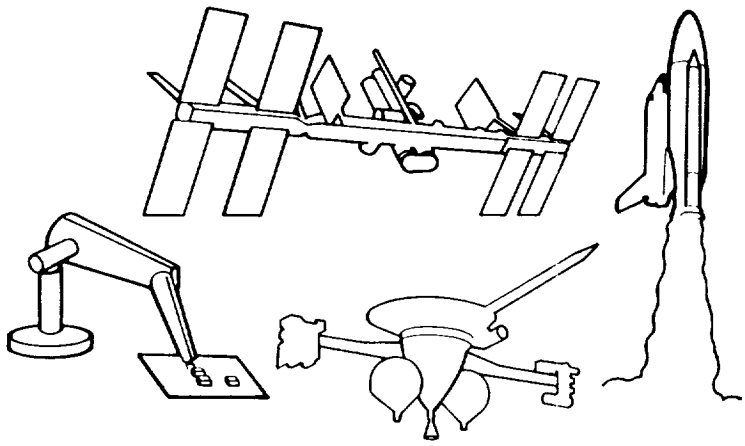
JPL Publication 89-45, Vol. 1

Proceedings of the 3rd Annual Conference on Aerospace Computational Control

Volume 1

D. E. Bernard
G. K. Man
Editors



December 15, 1989

National Aeronautics and
Space Administration

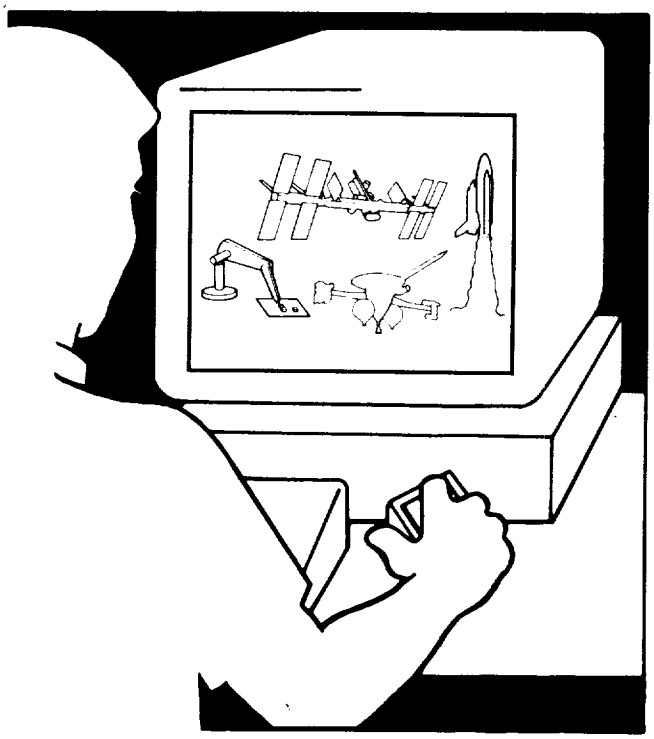
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

JPL Publication 89-45, Vol. 1

Proceedings of the 3rd Annual Conference on Aerospace Computational Control

Volume 1

D. E. Bernard
G. K. Man
Editors



December 15, 1989

NASA
National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

This publication was prepared by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

ABSTRACT

This is Volume I of two volumes of the Proceedings of the 3rd Annual Conference on Aerospace Computational Control.

The term Computational Control was coined this year to encompass that range of computer-based tools and capabilities needed by aerospace control systems engineers for design, analysis, and testing of current and future missions. This year's conference furthered the dialogue in this area begun at the 1987 Workshop on Multibody Simulation in Pasadena and continued at the 1988 Workshop on Computational Aspects in the Control of Flexible Systems in Williamsburg, Virginia.

A group of over 200 engineers and computer scientists representing government, industry, and universities convened at Oxnard for a three-day intensive conference on computational control. The conference consisted of thirteen sessions with a total of 107 technical papers in addition to opening and closing panel discussions.

Conference topics included definition of tool requirements, advanced multibody simulation formulations, articulated multibody component representation descriptions, model reduction, parallel computation, real time simulation, control design and analysis software, user interface issues, testing and verification, and applications to spacecraft, robotics, and aircraft.

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

FINAL REPORT

ON THE

3rd ANNUAL CONFERENCE ON AEROSPACE COMPUTATIONAL CONTROL

Douglas E. Bernard
Jet Propulsion Laboratory
California Institute of Technology

21 NOVEMBER 1989

ACKNOWLEDGMENT

The 3rd Annual Conference on Aerospace Computational Control was designed to advance the state of the art by bringing together users, researchers and software developers in the field of computational tools for control system design. Bringing such a conference together required the support and participation of a large number of individuals. Guy K. Man chaired the conference organizing committee which included Larry Taylor of NASA Langley, Terry Hinnerichs of the Air Force Weapons Laboratory, and Mike Jahanshahi and Ray Miller of the Jet Propulsion Laboratory.

Special thanks go to Mike Jahanshahi for organizing the vendor display and Ray Miller for administrative support of the planning and operation of the conference.

Finally, this conference could not have taken place without the interest and financial support of NASA, NSF, and DOD.

EXECUTIVE SUMMARY

The term Computational Control was coined this year to encompass that range of computer-based tools and capabilities needed by aerospace control systems engineers for design, analysis, and testing of current and future missions. This year's conference advanced the dialogue in this area begun at the 1987 Workshop on Multibody Simulation in Pasadena and continued at the 1988 Workshop on Computational Aspects in the Control of Flexible Systems in Williamsburg, Virginia.

A group of over 200 engineers and computer scientists representing government, industry, and universities convened at Oxnard for a three-day intensive conference on computational control. The conference consisted of thirteen sessions with a total of 107 technical papers. During the conference, eleven prominent computer hardware and control system design software companies displayed their products via hands-on demonstrations.

It was confirmed at the conference that the current control design and simulation tools are a limiting factor in today's control design and testing and are inadequate for future needs. The areas of concern are: A) Control design tools break down for high order systems. B) Spacecraft simulation tools are too slow to be used effectively for design and testing. C) An integrated computer-aided modeling-design-simulation environment is needed to improve productivity.

INTRODUCTION

The objective of the Computational Control Program is to spur the development of a new generation of articulated multibody spacecraft and robot modeling, control design, and simulation software tools. In order to assure a research approach that meets the needs of the community, it is necessary to bring the researcher, hardware and software developer, and user communities together on a periodic basis to allow exchange of ideas on requirements and developing capability. This conference provides a forum for that exchange. The term Computational Control was coined this year and encompasses the effort begun at the 1987 Workshop on Multibody Simulation in Pasadena and continued at the 1988 Workshop on Computational Aspects in the Control of Flexible Systems in Williamsburg, Virginia. These workshops are now considered the first and second annual conferences (workshops) on Computational Control, respectively.

The objectives of the 3rd annual Conference on Aerospace Computational Control were: 1) To provide NASA, NSF, and DOD a window into current computational control research and development in the areas of multibody dynamics formulation, concurrent processing, computational techniques, and control analysis and simulation software. 2) To allow researchers and tool developers to exchange ideas and experience on the correctness, efficiency, and usability of current computational tools. 3) To identify needs in important research areas such as computer-aided control design and multibody simulation tools which can be dealt with by government, industry, and university combined efforts. 4) To strengthen cooperation between industry and university researchers and aid in the transfer of technology from the research level to the users.

Each of these objectives was successfully met. Among the more than 200 conference participants in Oxnard were representatives from government, aerospace and computer hardware and software industries, and universities. The conference was led off by an opening session centered on a panel discussion of requirements for computational control tools and the state-of-the-art of such tools. The bulk of the conference consisted of thirteen sessions with a total of 107 technical papers. The sessions ranged from multibody dynamics formulation to parallel processing methods to control design and analysis tools to user environments. During the conference, eleven prominent computer hardware and control system design software companies displayed their products via hands-on demonstrations. The conference concluded with a lively panel discussion focused on future directions with active participation by users, researchers, hardware and software developers, and sponsors.

It was confirmed at the conference that the current control design and simulation tools are a limiting factor in today's control design and testing and are inadequate for future needs. The areas of concern are: A) Control design tools break down for high order systems. B) Spacecraft simulation tools are too slow to be used effectively for design and testing. C) An integrated computer-aided

modeling-design-simulation environment is needed to improve productivity.

This report is organized into three sections: the first describes the opening session with the panel discussion on requirements and state-of-the-art, the second section discusses highlights of the technical sessions, and the third section captures the closing session with its panel discussion on future directions.

OPENING SESSION

Sponsor Remarks

The conference was opened with remarks from each of the sponsors. NASA was represented by John DiBattista, the manager of the NASA Guidance, Navigation, and Control Program, who indicated that he planned to make the Computational Control area one of his highest priorities for funding. Elbert Marsh, NSF program director for Dynamics, Systems, and Control, stressed the interest of NSF in supporting basic and fundamental research in science and engineering. This includes computational control to the extent that it addresses generic issues within dynamic systems and control. Terry Hinnerichs, Chief of the Control and Development Branch at AFWL, described future DOD space missions which will need the sort of expanded capability at which computational control is aiming.

Introduction to Computational Control

The conference chair, Guy K. Man, gave an introduction to Computational Control. He explained that this conference is a sequel to the effort begun at the 1987 Workshop on Multibody Simulation and continued at the 1988 Workshop on Computational Aspects in the Control of Flexible Systems. He described the recommendations from the first workshop that led to a needs assessment effort and the formation of a multibody simulation technology verification committee. The needs assessment showed that in selected areas of multibody simulation and computer aided control design, orders of magnitude of improvement are necessary. The progress was interrupted at this point due to cutoff of SDIO funding. Out of the verification effort and the needs assessment findings, the case was made to NASA that a new program was needed. Computational Control is that new program.

The objective of Computational Control is to develop a new generation of articulated multibody spacecraft and robot modeling, control design, and simulation prototype software tools. The motivation behind this objective is that current tools are a limiting factor in today's control design and verification, and are inadequate for future needs.

Panel Discussion: Requirements and State-of-the-Art

The panel discussion was organized to give computational control tool researchers, developers,

and users a forum to discuss their perceptions of the state of the art and future requirements. The panel comprised: Panel Chair Larry Taylor of NASA Langley, John Sunkel of NASA Johnson Space Center, Glenn Macala of Jet Propulsion Laboratory, John Breakwell of Lockheed Missiles and Space, Terry Hinnerichs of the Air Force Weapons Laboratory, Guy K. Man of Jet Propulsion Laboratory, Achille Messac of C. S. Draper Laboratories, Jim Turner of Photon Research Associates, and Alan Laub of University of California Santa Barbara. Most panel members gave opening remarks of 5–15 minutes; Alan Laub gave a full-length paper as requested.

A brief sketch of the panelists' opening remarks follows. Larry Taylor introduced the panel noting that if we continue business as usual, the computational burden will be overwhelming.

John Sunkel described NASA Space Station requirements. He described the space station as a multibody system. The approach to flexible dynamics on Space Station is to separate the flexible modes—starting at 0.1 Hz—from the control Bandwidth of 0.01 Hz. A high fidelity flexible body simulator is needed for verification.

A test-bed environment is being developed for Space Station support. It will include orbital dynamics, aerodynamics, gravity gradient effects, sensor and actuator models, and flexible multibody dynamics. An early flexible body simulation dramatizes the need to improve the speed of current simulations: a 141 mode model required 7-8 hours to simulate 300 seconds.

Glenn Macala discussed the NASA effort to define computational control tool user requirements. User requirements are intended to illuminate the path towards efficient realization of computational control tools. The approach has been to first call out the basic functions of the controls engineer in the development of a spacecraft control system. Next, the functions of the tools needed to support the control engineer in his task were defined. In addition, NASA mission models were identified that would be good candidates to use the next generation of computational control tools. The final step is to compare the needed functions with the mission models to set performance requirements on the tool functions.

The report "Computational Control Program User Requirements," JPL Publication 89-40, is in preparation.

John Breakwell gave a view from industry. His perspective is that control engineers typically stick with what worked the last time. The change that he sees is that the availability of real-time control analysis tools is giving the analyst more ability to go into the laboratory and contribute to the test and verification activity. He feels that tools which reduce the gap between designers and implementers would be of great value.

Terry Hinnerichs discussed Air Force space requirements. Focusing on SDI, he described a wide variety of space systems concepts. One example of where multibody dynamics and

computational control fit in was the problem of a fine pointing telescope attached to a noisy laser device.

Guy K. Man outlined the results of the multibody simulation codes user survey. A total of 243 questionnaires were sent to 78 organizations. There were 40 responses, covering 27 organizations and 21 codes. Data collected included documentation, welcome features, additional features desired, ease of use, typical applications, run time acceptability, and over 20 additional issues. The details will be included in a paper in the conference proceedings.

Achille Messac discussed the Control Structure Interaction (CSI) problem–redefining it as one of Control Structure Integrated Design. His message was that the combination of the optimal structure and the optimal control does not necessarily yield the optimal system.

Jim Turner discussed current and possible future directions for multibody dynamics software. He listed three technical breakthroughs which are stimulating current research: Order N algorithms, symbolic manipulation, and parallel processing. He also identified areas with little current capability which remain areas for potential growth. These included fluid/mechanical coupling, large deflection flexible dynamics, and event driven activities such as robot/payload interaction.

Alan Laub gave a survey paper covering control algorithms and software. He started by agreeing that much of today's software either is not extensible or is inadequate to address future requirements. He advanced the folk theorem that methods that are good for hand calculation are typically poor for digital computation.

Accuracy is seen to be dependent on: the conditioning of the problem, the numerical stability of the algorithm, and the specifics of the hardware and software implementation. In the area of modeling, problems were identified for transformations between transfer function and state space realizations for large order systems. Since much problem structure is lost when second order systems are recast as first order systems, the value of developing control and synthesis methods for state space models in second order form was stressed.

Laub described the status of current control algorithms as follows: modest size ($\leq$200 states), dense matrices, no exploitable structure, serial computing. Suggestions for future developments are: use reliable components (for example LAPACK as a starting point,) create an extendible package rather than including everything anyone could want, and plan a team approach resulting in a coordinated, broad based, focused, long-term, carefully planned effort.

Open Discussion

Following the panel discussion, Larry Taylor opened the floor to open discussion.

During the open discussion, Ray Gluck raised the issue of funding for computational control research. John DiBattista responded that there is a definite opportunity for increased funding—but

only if the community can clearly characterize the field as one ripe for solutions to problems of importance to NASA.

John Hedgepeth addressed the issue of assuring that the results of computational control research get into real applications. Comparing computational control to CSI he suggested that concrete examples are needed to make clear where this art can help in the design of advanced performance spacecraft in ways that are really needed by users of the system.

Ed Haug stressed the importance of Differential Algebraic Equations (DAE) in the simulation of constrained systems. Alan Laub agreed, adding that control of DAEs is much harder than control of ODEs.

TECHNICAL SESSIONS

Due to the size of the conference compared to previous computational control workshops, it was necessary to break the technical sessions up into parallel sessions. The individual technical papers are described in the conference proceedings.

Any list of highlights would certainly omit important papers, but a few items will be mentioned to capture some of the flavor of the technical sessions. In the multibody formulation area, von Flotow addressed the issue of the need for non-linear strain-displacement relations in dynamic analysis–describing three options as consistent, inconsistent, and ruthless. Rodriguez outlined a powerful new dynamic analysis approach–"spatial operator algebra." In the parallel processing area, many authors discussed methods and advantages of fine and coarse grain parallelization. Slafer addressed the state-of-the-art in real-time hardware-in-the-loop simulation, and Howe put forward new numerical integration techniques which help increase simulation speed.

A number of papers discussed dynamic analysis applications ranging from the space station to manipulators to fighter aircraft. Others discussed emerging integrated capabilities: Russell discussed ISM at Boeing, Bauer and Downing discussed INCA and ASTEC at NASA Goddard, Kumar discussed TREETOPS work at Dynacs, and Canfield and Hummel discussed ASTROS and MEAD at Wright-Patterson.

ADDITIONAL CONFERENCE ACTIVITIES

A banquet was held the second day of the conference with Ed Haug of the University of Iowa as the featured speaker. Haug took a global perspective, looking at the competitive position of the United States in the world. He concluded that one place that we could leverage our capabilities was

in expanded use of simulation technology to improve the performance and reliability of our products.

Eleven software and hardware vendors displayed their products during the technical sessions of the conference. These included Applied Dynamics International, AT&T/Circuit Studios, BBN Laboratories, Boeing Computer Services, Cray Research Inc., Integrated Systems, Mitchell, Gauthier & Associates, SIKOG Inc (LEVCO), Silicon Graphics, Systolic Systems, and 3-D Visions.

CLOSING SESSION

The closing session was designed to bring everyone back together after two days of parallel sessions for two general interest papers followed by a panel discussion on future directions.

Full-Length Papers

The first speaker was Ed Haug of the University of Iowa, who described the concept of a Verification Library for multibody simulation software. The idea of a verification library is to build up a public domain library to store data for example problems. The intent is to facilitate the comparison of test results and the results from different multibody simulation codes. The library would contain model information, simulation results, test definition data, and processed test data.

Three flexible multibody codes are being compared at this time: DADS, DISCOS, and CONTOPS. Fundamental differences between these codes include: DADS and DISCOS use absolute coordinates where CONTOPS uses relative coordinates, DISCOS uses Euler angles to define joints while CONTOPS and DADS use unit vectors. In DISCOS and CONTOPS, the user must specify a kinematically acceptable initial condition, while DADS will solve an optimization problem to provide the initial condition based on estimates of the states. These differences make the job of designing a translator between input data sets for these codes difficult. The complete paper discussing the above is in the conference Proceedings

The next speaker was John Doyle of the California Institute of Technology. Doyle gave his view of the history of control theory and its future directions. His main message was that for the "post-modern" control era we should emphasize control as a strategy for dealing with uncertainty.

Panel Discussion

Guy K. Man moderated the panel discussion with the theme "future directions." The panel was composed of government sponsors, users, software/hardware developers, and academicians. John DiBattista of NASA Headquarters and Terry Hinnerichs of the Air Force Weapons Laboratory represented government sponsors. Government and industry users of Computational Control tools included: John Sunkel of NASA Johnson Space Center, Loren Slafer of Hughes Aircraft, and John Breakwell of Lockheed Missiles and Space. Hardware and software developers were represented by

Jim Taylor of General Electric, Doug Petesch of Cray Research, and Dan Rosenthal of Symbolic Dynamics. Academic participants included George Lee of Purdue University, Alan Laub of University of California Santa Barbara, and John Doyle of the California Institute of Technology.

Guy K. Man started off the discussion by relating firsthand experience with existing tools leading to the conclusion that they are too slow for use for spaceflight projects. He mentioned the mission models currently being considered as computational control reference problems. Other possibilities mentioned by panelists and attendees included Lunar/Mars initiative spacecraft, space transportation systems, solar-electric propulsion, solar sail, national aerospace plane, SDI directed energy spacecraft, and tether systems.

George Lee mentioned a related program in flight controls sponsored by the Wright Patterson Research Center. He had two key messages based on their experiences: use the best quality low level numerics package you can find, and when considering user friendliness, think of the expert user as well as the novice.

Alan Laub stressed that current state-of-the-art numerical analysis software such as EISPACK and LINPACK were only possible due to stable long-term funding, and that year-to-year funding tends to lead to the taking of shortcuts.

The issue of the interaction between the control engineer and the computer scientist in computational control was raised. Amir Fijany raised the issue of the importance for the controls and dynamics specialist to know the power of parallel processing. Similarly, Dan Rosenthal related from his experience the value of bringing in the talents of computer scientists at an early point. The flow of tool requirements from control engineers to computer scientists was discussed, and additional comments stressed the feedback from the computer scientists back to the control engineers.

John Doyle sees a set of core control routines—comparable to computing eigenvalues of a matrix—that are essential to being able to solve the types of problems that we talk about, and need a lot of work for actual implementation. He also finds that MATLAB-type packages, while good for linear algebra, have completely inadequate data types for control.

The issue of the lack of standardization in interfaces between packages was raised. This is an issue that many groups are concerned about. Jim Taylor mentioned that the International Federation of Automatic Control (IFAC) has a working group studying standardization of data structures. Terry Hinnerichs referred to a Russell's paper at this conference describing a AFWL sponsored effort tying together structural dynamics, controls, thermal and optics codes. This development is the Integrated Structural Modeling (ISM) program, which is to go into beta testing in 1990.

Doug Petesch suggested that users explore supercomputing capabilities before presuming that the solution required parallel processing. He also mentioned the advantage that comes from doing software development as well as production runs on the supercomputer. The value of a hypothetical teraflop machine was discussed. It was agreed that while valuable, it would not solve all the problems—such as numerical difficulties with higher order systems. In addition some thought that the same speed could be achieved at a much lower cost using parallel processing.

Dick Vandervoort of DYNAC referred back to John Doyle's comments about uncertainty, noting that multibody dynamics modeling gives highly detailed models—more detailed than the control designer needs or wants. In fact most of it is uncertainty, and what we need is a way to model that uncertainty in a way that can be used in the control design

Perhaps John Hedgepeth's comment best summed up the reason to move forward with Computational Control. Summarizing the comments of several speakers, he noted that we are proceeding ahead conservatively in control system design for planned missions, and that without improvements in technology—we are choosing to design only spacecraft that we already know how to design.

An Assessment of Multibody Simulation Tools for Articulated Spacecraft

Guy K. Man and Samuel W. Sirlin
The Jet Propulsion Laboratory
The California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA 91109

Introduction

A survey of multibody simulation codes was conducted in the spring of 1988, to obtain an assessment of the state of the art in multibody simulation codes from the users of the codes. This information will be used to evaluate the need to develop future codes. If this need is established, it will also be used to guide the development of future capabilities. A questionnaire, covering 30 issues, was developed for this purpose. Sixty questionnaires were sent out to 50 organizations. We received 40 replies from 30 organizations covering 21 simulation codes.

This survey covers the most often used articulated multibody simulation codes in the spacecraft and robotics community. There was no attempt to perform a complete survey of all available multibody codes in all disciplines. Furthermore, this is not an exhaustive evaluation of even robotics and spacecraft multibody simulation codes, as the survey was designed to capture feedback on issues most important to the users of simulation codes. We must keep in mind that the information received was limited and the technical background of the respondent varied greatly. Therefore, this paper only reports the most often cited observations from the questionnaire. In this survey, it was found that no one code had both many users (reports) and no limitations.

The paper is organized as follows. The next section is a report on multibody code applications. Following applications is a discussion of execution time, which is the most troublesome issue for flexible multibody codes. The representation of component flexible bodies, which affects both simulation setup time as well as execution time, is presented next. Following component data preparation, two sections address the accessibility or usability of a code, evaluated by considering its user interface design and examining the overall simulation integrated environment. A summary of user efforts at code verification is reported, before a tabular summary of the questionnaire responses. Finally, some conclusions are drawn.

Applications

It is not surprising that general purpose multibody simulation codes are used to address spacecraft and robotics with articulated elements, because of the complexity of the equations

of motion. Some codes are equipped to simulate ground vehicles or aircraft. The actual applications include control design and verification, deployment simulation, machine design, impact simulation, tether simulation, explosion simulation, and human in-the-loop real time simulators for the Space Shuttle and Space Station. About 70% of the users reported that multibody codes are used for control system design and verification. The remaining 30% apply multibody codes for system design and dynamics studies. It is also reported that multibody codes are used to check other codes. For example, a rigid body code might be used to check a more complex flexible multibody code.

System order exercised varies from a robot arm with less than 10 states to a Space Station model with 150 states. One user reported trying a model with over two hundred states with modes up to 1000 Hz using DISCOS. Flexible body systems usually are higher order systems than rigid body systems, and the simulation duration is usually shorter. Simulation duration does however vary according to the application.

Execution Time

Excessive execution time is one of the two most cited shortcomings of current multibody codes. This concern applies especially to flexible multibody simulations. Existing codes are so slow for most problems that it is impossible to use them during the design phase when quick turnaround is needed. Highly simplified flexible models or even rigid body models must suffice, even though the control system designer fully recognizes that the result may be inaccurate. For example, consider a 20 degree of freedom flexible system with flexible modes less than 100 Hz. The CPU time to real time ratio for this example is 200/1 on a VAX class computer. As the system order becomes higher, the computational load will become much more stressing because the computational load scales as N^3, for the current codes, where N is the number of degrees of freedom.

It is interesting to note from the survey that the simulation durations for flexible body problems are usually short (seconds). Very few users are using flexible codes for long runs. From the data, simulation duration is seen to be inversely proportional to the system order. This telltale observation indicates that flexible multibody simulation codes are not being used extensively, because long duration simulation costs are too high. One of the respondents summarizes this quite well - "Multibody run times are seldom acceptable. You either pay the price or get approximate answers."

Getting an approximate answer is the only way to reduce excessive execution time for given hardware and software. Approximation of flexible multibody systems is done most commonly by reducing the order of the individual flexible body dynamics, in fact often reducing the bodies down to rigid bodies. More than one user reported that fast rigid body simulations were used to check the more complex flexible body simulations. If appropriate, the more complex flexible body simulation is abandoned in favor of the faster rigid body code. The model reduction approach has not been included in this survey. But some observations of this method are discussed in the next section.

Component Model Representation

One of the major concerns in modeling a flexible multibody system is how to obtain data for the component elastic bodies. Among the flexible multibody codes surveyed, all except one code use the modal representation for flexible bodies. The primary benefit of using a modal representation is "simplicity," which can lead to reduced computation time. There are a variety of modes one can use to represent the individual bodies of a multibody system, such as Hurty modes (Craig Bampton modes) or cantilever modes. The modes of each body have to be supplied to the multibody code which will then synthesize the modes of the component bodies to arrive at a system model for time simulation.

The body of knowledge of choosing the best component mode set for system synthesis is called component mode synthesis. This includes both choosing the type of modes to use (possibly several) as well picking out which of the modes must be retained for simulation fidelity. The various component mode sets were developed for this purpose, however they were developed for systems with no articulating components, essentially static system geometry. Flexible multibody codes, on the other hand, were developed to model dynamical systems with varying geometry. The applicability of component mode synthesis results to multibody analysis is not well understood and care must be taken in each specific case to ensure that the results are correct. One question that needs to be asked often is whether or not a set of component body modes is complete over the range of interest of the articulation angles (or translations). After the system is synthesized in the multibody code, a check can be made against a higher fidelity linear model such as a NASTRAN system model only for a fixed geometry.

It was pointed out in the previous section that model reduction is one of the commonly used methods to obtain an approximate solution. This is a necessary step for obtaining a reasonable simulation computational time. A variety of methods exist to pick out the significant modes. These arose out of the the slightly different model reduction problems of control theory. What model reduction criterion should one use to obtain an adequate approximate multibody model? No definitive answer exists, but the answer certainly depends on how the simulation is to be used. For example, if the simulation is to be used for control system design, then controllability and observability criteria may play a part. It may also be useful to consider the control design problem simultaneously with the model reduction problem. In some multibody codes the level of modal approximation used (for example the DISCOS mass distribution options) ties directly into the flexibility data preparation process, compounding the data preparation problem.

Considering these issues, it is not surprising that the survey indicates flexible data requirements are obscure, challenging, poorly treated and inadequate. Considerable research remains to be done in component model representation.

User Interface

One of the most common complaints concerning current multibody simulation codes is the lack of interactive model setup capability. Codes that have a friendly input format win praise from the users. Interactive, menu driven type preprocessors with good error messages are

indicated as highly desirable. Incomplete documentation of the code was the second most cited complaint. Sophisticated users also desire well commented source code, especially if the user manual is inadequate. Online manuals and documentation were mentioned by several users as highly desirable. For flexible multibody codes, the availability of the theoretical background on which the code is based is very important. In addition to documentation, application examples are found to be an effective way to train users. It was also indicated that examples are needed for modal data preparation.

At the output interface, data retrieval flexibility and graphics seemed to be most important. Users were divided on whether they wanted built-in plotting capabilities or whether they simply wanted the ability to export data to their own favorite plotting programs. The survey did not indicate any major concerns in this area.

Integrated Environment

Multibody codes are sometimes used as stand alone tools but more recently complex flexible multibody codes are often used in conjunction with other software such as finite element, control analysis, or optics codes. In the latter environment, the multibody code becomes a fundamental building block of an integrated design and analysis environment. Some of the codes surveyed, such as TREETOPS, are actually a microcosm of such an environment. This is a natural development trend, because multibody systems are becoming more complex, forcing system engineering work to become multidisciplinary. The survey pointed out that no environment can satisfy everybody. Every organization has its own culture and emphasis. Everybody has some pet code. It is therefore perhaps wise to treat multibody codes as modular building blocks which can be easily integrated into a bigger environment. If this is the approach we adopt, then there is a need to define a standard interface for data passage.

There are a number of additional capabilities that users found helpful beyond the generation and integration of equations of motion. These are: a library of joints sensors and actuators, the ability to obtain constraint forces, transfer functions, Jacobians, and key state matrices for external control analysis, the ability to incorporate user defined subroutines, and of course a model reduction preprocessor.

Verification

A number of codes have been independently checked by users with other codes. Most users utilize simple test cases and the principles of mechanics to check simulation results. Of the test cases reported, none are designed to check flexible body effects. This is not surprising, because if component model data generation is not well understood, the verification of the model must also be nontrivial. More work should be done in this area so as to add confidence in the simulation results and the flexible multibody codes.

It is suggested that a set of standard test cases be collected, which can be used by the community to check both existing and new simulations codes. The set should include simple test cases with known simulation results as well as actual experiments with test data. This

collection process has been started by NASA and the University of Iowa.

Tabulated Data

The user questionnaire data has been summarized in the following tables. Table 1 gives the availability of the multibody code, as well as how many respondents used each code and how extensive the information provided by the users was. Almost all of the software is available, either commercially or in the public domain (COSMIC).

Table 2 gives various details of how users used the codes, and what code features they liked or felt needed improvement. Typical applications and whether or not the run times were acceptably fast are given, as well as what component data the code required and, in general, how easy the code was to use. User comments are included on the quality of the documentation as well as what features of the code they liked, and what features could be added to improve productivity. Following are some comments on the individual columns.

There was a wide range in documentation quality, from "clear and precise" to "inadequate overall," which in general indicates how much care the developer took with making the code easy to use.

There was very little overlap from code to code of features that the users appreciated. Some users liked the user interface of some codes, while one code was notable for animation, and another for being database driven. Two features that did show up more than twice were the ability to add user-defined subroutines, and a library of application modules.

There was also a wide variety of additional features that the users desired, again with surprisingly few common needs from code to code. Significant needs included better documentation (mentioned for four codes), and a better data interface capability (also mentioned for four different codes).

The acceptability of the simulation run times was much more uniform. The rigid body codes were all considered acceptably fast, while the flexible body codes were, in general, acceptable only for small problems.

There was considerable variety in the user comments on the type of data needed for the representation of flexible components, indicating the lack of maturity of this area.

It is important to keep in mind that some multibody codes have been around for a long time and so have an extensive user and applications base, while other codes are new and have only been used by relatively few people on a small number of examples.

Conclusions

Examining the reports of existing multibody code users, three facts become evident:

1. It is difficult to use flexible multibody codes in design and analysis due to the long execution times for realistic problems,

2. Representation of component flexible bodies is poorly understood, and

3. Not enough thought has been given to the user interface by the code developers.

At the Workshop on Multibody Simulation in 1988, JPL made a commitment to coordinate a multibody code verification, as part of an ongoing effort of the whole multibody simulation community. This survey report, as well as the verification library begun jointly with the University of Iowa, represent the first steps toward meeting that commitment.

Acknowledgement

The Research described in this paper was performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

TABLE 1 MULTIBODY SIMULATION CODES INCLUDED IN THE SURVEY

Code	Contact	# of Replys	Information Received	Availability	Host Computer
ADAMS	Rajiv Rampali MDI Inc. 3055 Plymouth Rd. Ann Arbor, MI 48105	3	extensive	yes, commercial	• Cray • Prime • NEC • Cyber • IBM • SGI • Alliant • HP • Convex • Sun • DEC • Apollo • Intergraph
ALLFLEX	Jimmy Ho Lockheed Missiles & Space Co. Space System Division 1111 Lockheed Way Sunnyvale, CA	1	moderate	yes	• CRAY • VAX
AUTOLEV	David Levinson 93-30/250 Lockheed Palo Alto Research Lab 3251 Hanover Street Palo Alto, CA 94304	1	moderate	yes, commercial	IBM PC/CLONE
CONTOPS	John Sharkey NASA MSFC Code ED 12 Marshall Space Flight Center Alabama 35812	4	extensive	yes, COSMIC	• VAX 11/750 • MICRO VAX • HP 9000
DADS	CADSI, Inc. P.O. Box 203 Oakdale, IA 52319	5	extensive	yes, commercial	• VAX • Multiflow • Alliant • Prime • Apollo • Silicon Graphics • Cray • IBM PC • DEC • Computervision • IBM • Cyber • Sun • HP

Code	Contact	# of Replys	Information Received	Availability	Host Computer
DAMS	A.A. Shabana P.O. Box 4348 Dept. of ME U of Illinois @ Chicago Chicago, IL 60680	1	limited	yes	IBM
DISCOS	COSMIC U. of Georgia Computer Service Annex 382 E. Broad Street Athens, GA 30602	7	extensive	yes, COSMIC	• IBM 3090, 770 • VAX 11/780, 750 • MICRO VAX
DYNOCOMBS	Ron Huston U of Cincinatti Dept. of ME & IE Cincinatti, OH 45221-0072	1	moderate	yes	• IBM 4043 • VAX 1170
DYNA/EF3	Jim Lawrence NASA JSC Mail Code EF3 Houston, TX 77058	1	moderate	yes	Gould Concept 32
DYNAMUS	Farid Amirouche Dept. of ME P.O. Box 4348 U of Illinois @ Chicago Chicago, IL 60680	1	extensive	yes	• IBM 370 • VAX 11/780

CODE	DOCUMENTATION	WELCOME FEATURES	ADDITIONAL FEATURES DESIRED	USER EXPERIENCE	TYPICAL APPLICATIONS	EASY TO USE	RUN TIME ACCEPTABILITY	FLEXIBLE COMPONENT DATA
Generic Manipulator Simulation	• Moderately documented • Source available and commented	• Open and closed chain dynamics and loads	• Better I/O • Flexbody dynamics	limited	• Manipulator	yes	yes	Does not model flexibility
LATDYN	• Manual available • Source available and commented	• Easy to use finite element code • Modeling of contrd forces • Macros	• Need restart • Better naming convention • Available for newer computers • Static Analysis • Interactive preprocessor	moderate	• RMS slew on spacestation • Impact, lockup	yes	Problem dependant	Easy, finite element approach
MIRRORS	Partial, subroutines are not completely documented	• Database driven	• More generic N body code	moderate	• Loads • Shuttle RMS	no	• Not acceptable for flexible problems	• Lumped mass • Craig Bampton
MULTFLEX	• User manual • Source not available, lightly documented	• Easy to change configuration • Symbolic pre-processor • Code generation capability	• Better documentation • Graphic preprocessor • Torsional flexibility • Transfer function	moderate	• Shuttle RMS • Spacecraft	• Easy setup	• Problem dependent • Acceptable on Cray	• Mode shape functions

Code	Contact	# of Replys	Information Received	Availability	Host Computer
MULTFLEX	Martin Tong The Aerospace Corp. MS (M4/972) P.O. Box 92957 Los Angeles, CA 90009-2957	4	moderate	proprietary	• CDC Cyber 175 • Cray X-MP
NBOD2	COSMIC University of Georgia Computer Services Annex 382 E. Broad Street Athens, GA 30602	1	limited	yes, COSMIC	VAX 11/785
SD/EXACT	Dan Rosenthal 11585 Silvergate Drive Dublin, CA 94568	6	extensive	yes, commercial	• VAX 11/785 • IBM 3090 • VAX Station II • Sun Workstation • HP-9000 • MICRO VAX
SPASIS	COSMIC University of Georgia Computer Service Annex 382 E. Broad Street Athens, GA 30602	1	moderate	yes, COSMIC	• VAX 8650 • Harris-800
Station Control Simulation	John W. Sunkel NASA JSC Mail Code EH2 Houston, TX 77058	1	limited	yes	CYBER 830
TREETOPS	John Sharkey NASA MSFC Code ED 12 Marshall Space Flight Center Alabama 35812	3	moderate	yes, COSMIC	VAX

TABLE 2 USER EXPERIENCE

CODE	DOCUMENTATION	WELCOME FEATURES	ADDITIONAL FEATURES DESIRED	USER EXPERIENCE	TYPICAL APPLICATIONS	EASY TO USE?	RUN TIME ACCEPTABILITY	FLEXIBLE COMPONENT DATA
ADAMS	• Well documented • User's & Application Manual • Theory Notes • Source not available	• 9 CAD/CAE Interface • Linearization, Modal Analysis • Large Joint Library • Wide Variety of Forces • Accurate Flexibility • Full Feature Restart • Interface to MatrixX • Postprocessor with Animation • User Subroutines	• Built in control elements • Theory manual	•Extensive (>100 Engineers trained at MDI) •Fully staffed hotline support	AUTOMOTIVE • Subsystem & system modeling • Suspension Design • Ride Analysis AEROSPACE • Robotics • Spacecraft Simulation • Mechanism analysis and design	• Yes • 9 commercial preprocessors available • 3 high-end commercial graphics animation post-processors	Problem Dependent	• ANSYS/NASTRAN reduced stiffness Matrix • Direct physical properties • Handles all geometric nonlinearities • Standard beam and field elements
ALLFLEX	• Documentation • Source not available	• Easy and fast to assemble simulation	• Menu driven input	extensive	Spacecraft	yes	Problem dependent	Format is well defined
AUTOLEV	• Documented • On-line help command • Source not available	• Very versatile • PC based • Explicit symbolic equations	None thus far	moderate	• Robotic device • Spacecraft	yes	yes	Handles systems of rigid bodies only
CONTOPS	• I/O well documented • Inner workings not well documented • Code comments sketchy	• Interactive preprocessor • Plotting nicely done	• More help in error tracing • Modal data prep examples • More application examples • Sensor model • More boundary conditions for flex data • Frequency response data • Expand output list	extensive	• Robots • RMS • Spacecraft • Robot experiment • Smart Structures	yes	Fair, not acceptable for flexible problems	• More flexibility in flex-body representation • Flexible augmentation is obscure
DADS	• Well documented • Source not available • User's manual • Examples manual	• CAD/CAE interfaces • Preprocessor • Control system modeling elements • Postprocessor • Extremely stable integration	• More sophisticated postprocessor • Interface to control analysis package • Stop/restart • Error messages • Control Analysis capability	• Extensive • Fully staffed hotline support	• Servo control of robot arms • Vehicle simulation • Mechanisms • Spacecraft	yes	Problem dependent	• Component modal synthesis • Interfaces to NASTRAN, ANSYS, ABAQUS • Handles all geometric nonlinearities
DAMS	Source is available but not well commented	User subroutine facility	not available	moderate	• Machine design • Robot • Impact	Mixed	Problem Dependent	Normal modes

CODE	DOCUMENTATION	WELCOME FEATURES	ADDITIONAL FEATURES DESIRED	USER EXPERIENCE	TYPICAL APPLICATIONS	EASY TO USE	RUN TIME ACCEPTABILITY	FLEXIBLE COMPONENT DATA
DISCOS	• Source code commented • Documentation inadequate overall	• Very general • User facility subroutine • Documentation of theory (but not complete) • Examples • Very few bugs now • Wide user community • Trustworthy	• Much better documentation • Much better user interface • Faster code • Better numerical stability • Automatic redimensioning	extensive	• Spacecraft • Rapid retargeting maneuvers • Deployment • Propellent slosh • Check other code	• No • Need thorough understanding of code	• Problem dependent • Too slow for high order systems	• Normal modes • Any mode shape
DYNOCOMBS	• Well documented • Source available	• Efficiency • Broad Applicability	• Animated output • Menu driven input	moderate	• Towing of submerged cables • Robotic • Human body modeling	yes	yes	Flexibility is modeled at joints
DYNA/EF3	• Mathematical model documentation • Software design documentation • Source available and commented	• Extensive vehicle library	• Expert systems shell • Better control of data time, transport delay	moderate	• Human-in-the-loop real time simulator for shuttle and space station • Shuttle RMS shuttle/station docking	fair	yes (real time)	Does not model flexibility
DYNAMUS	Yes, clear, precise	• Friendly input format • Automatic elimination of singularities due to constraints • Handles geometric nonlinearities	N/A	moderate	Constraint Tree, Flexible Spacecraft deployment, Manipulator	yes	good	Normal modes Component mode analysis
FB2	• Not well documented • Source is commented	• Fast • Lots of options	• Better I/O • Better documentation • Large rotation between bodies	limited	• Spacecraft • Robot	yes	yes	Craig-Bampton modes

CODE	DOCUMENTATION	WELCOME FEATURES	ADDITIONAL FEATURES DESIRED	USER EXPERIENCE	TYPICAL APPLICATIONS	EASY TO USE	RUN TIME ACCEPTABILITY	FLEXIBLE COMPONENT DATA
FLEXIM	Well documented	• Very few bugs	Not available	limited	• Spacecraft	Yes, but initial investment to learn the code is high	N/A	• Hurty modes • A special NASTRAN postprocessor exists
Generic Manipulator Simulation	• Moderately documented • Source available and commented	• Open and closed chain dynamics and loads	• Better I/O • Flexbody dynamics	limited	• Manipulator	yes	yes	Does not model flexibility
LATDYN	• Manual available • Source available and commented	• Easy to use finite element code • Modeling of control forces • Macros	• Need restart • Better naming convention • Available for newer computers • Static Analysis • Interactive preprocessor	moderate	• RMS slew on space station • Impact, lockup	yes	Problem dependent	Easy, finite element approach
MIRRORS	Partial, subroutines are not completely documented	• Database driven	• More generic N body code	moderate	• Loads • Shuttle RMS	no	• Not acceptable for flexible problems	• Lumped mass • Craig Bampton
MULTFLEX	• User manual • Source not available, lightly documented	• Easy to change configuration • Symbolic pre-processor • Code generation capability	• Better documentation • Graphic preprocessor • Torsional flexibility • Transfer function	moderate	• Shuttle RMS • Spacecraft	• Easy setup	• Problem dependent • Acceptable on Cray	• Mode shape functions

CODE	DOCUMENTATION	WELCOME FEATURES	ADDITIONAL FEATURES DESIRED	USER EXPERIENCE	TYPICAL APPLICATIONS	EASY TO USE	RUN TIME ACCEPTABILITY	FLEXIBLE COMPONENT DATA
NBOD2	• Fairly well documented • Source available and commented	• Preprocessor code is helpful	• Indexing scheme on free and constrained joints can be improved • Simple application examples desired	limited	• Attitude control of spacecraft	mixed	yes, but should be improved	Does not model flexibility
SD/EXACT	• Good manual • Source code not available	• Configuration parameters can be changed at run time • Fast	• More efficient memory use • Handle flexibility • Graphics • Variable degree of freedom	extensive	• Spacecraft • Robot • Robot experiment • Code checking	yes	yes	Does not model flexibility
SPASIS	• Well Documented • Source available and commented	• Good documentation • User friendly inputs • Batch processing utilities • Extensive earth orbital dynamics	• More efficient aero-dynamics and propellant dynamics options • Animation • Reboost ability	moderate	• Large space platforms • Small space platforms • Propellant dynamics • Docking	yes	yes	Does not model flexibility
Station Control Simulation	• Code documented • Source available and commented	• Modular design • Self contained	• Model reduction code • Faster code	limited	• Space station GN&C design • RCS reboost maneuver	yes	Excessive when flexible model used	Outputs 2 and 4 from NASTRAN
TREETOPS	• Reasonably documented • Source is available but not thoroughly commented	• Interactive preprocessor • Speedup options • Library of actuators and sensors	• Certain flex inputs not fully defined • Graphics preprocessor • Cumbersome output • Better interface with user subroutines	moderate	• Shuttle and payloads • Spacestation • Wing flap • Check other codes	usually	Usually, high for flexbody payloads	• Difficult, additional software needed • Modes are restrictive

SPATIAL OPERATOR ALGEBRA FRAMEWORK FOR MULTIBODY SYSTEM DYNAMICS

G. Rodriguez, A. Jain, and K. Kreutz
Jet Propulsion Laboratory/California Institute of Technology
4800 Oak Grove Drive, MS 198-219, Pasadena, CA 91109

Abstract: This paper describes the Spatial Operator Algebra framework for the dynamics of general multibody systems. The use of a spatial operator–based methodology permits the formulation of the dynamical equations of motion of multibody systems in a concise and systematic way. The dynamical equations of progressively more complex rigid multibody systems are developed in an evolutionary manner beginning with a serial chain system, followed by a tree topology system and finally, systems with arbitrary closed loops. Operator factorizations and identities are used to develop novel recursive algorithms for the forward dynamics of systems with closed loops. Extensions required to deal with flexible elements are also discussed.

1 Introduction

The field of multibody dynamics is currently being challenged in two major ways. The increase in the size and complexity of spacecraft systems requires the development of tools that not only help manage the complexity of such systems, but also facilitate the development of novel dynamics formulation techniques and solution algorithms. Areas such as robotics involve multibody systems consisting of multiple robot manipulators interacting with each other and with complex environments. These are multibody systems with not only constantly time–varying topological structure, but also ones in which the constituent bodies change with time. Coping with this aspect requires versatile and flexible dynamics simulation tools.

In this paper, the Spatial Operator Algebra Framework [1] is used to develop a systematic procedure for concisely formulating the equations of motion and derive spatially recursive forward dynamics algorithms for multibody systems. The equations of motion of progressively more complex rigid multibody systems such as serial chains, tree topology systems and finally closed chain systems are developed. Operator factorizations and identities are then used to obtain efficient spatially recursive algorithms for the forward dynamics of such systems. Extensions to handle flexible link elements are also discussed.

2 Equations of Motion

We begin by briefly describing the coordinate–free *spatial notation* used throughout this paper. Given the linear and angular velocities v and ω, the linear force F, and moment N at a point on a body, the *spatial velocity* V, *spatial acceleration* α and the *spatial force* f in $\mathcal{R}^6$ are defined as follows:

$$V \triangleq \begin{pmatrix} \omega \\ v \end{pmatrix}, \quad \alpha \triangleq \dot{V}, \quad f \triangleq \begin{pmatrix} N \\ F \end{pmatrix}$$

The *rigid body transformation operator* $\phi(.) \in \mathcal{R}^{6\times 6}$ is defined as:

$$\phi(l) \triangleq \begin{pmatrix} I & \tilde{l} \\ 0 & I \end{pmatrix}$$

where l is a vector joining two points, and $\tilde{l}$ is the cross–product matrix associated with l which acts on a vector to produce the cross–product of l with the vector. $\phi(l)$ and $\phi^*(l)$ transform spatial forces and spatial velocities respectively between two points on a rigid body seperated by the vector l. For a rigid body, its spatial inertias M_C and M_O at its center of mass C and at another point O respectively, are defined as

$$M(C) \triangleq \begin{pmatrix} \mathcal{J}(C) & 0 \\ 0 & mI \end{pmatrix}, \quad \text{and} \quad M(O) \triangleq \phi(p)M(C)\phi^*(p) = \begin{pmatrix} \mathcal{J}(O) & m\tilde{p} \\ -m\tilde{p} & mI \end{pmatrix}$$

where p is the vector from O to C, m is the mass of the body, and $\mathcal{J}(C)$ and $\mathcal{J}(O)$ are the inertia tensors for the body about C and O respectively. The reader is referred to [2] for additional discussion on the use of spatial notation.

2.1 Dynamics of a Serial Rigid Multibody System

Serial rigid multibody systems form basic subsystems from which the dynamics of more general rigid multibody systems can be generated. In this section we derive the equations of motion of a serial multibody system consisting of n rigid links connected together by multiple dof joints. The links are numbered 1 through n from tip to base. We use the terms *outboard* (*inboard*) link to refer to a link on the path towards the tip (base).

The set of configuration variables for the serial chain are the collection of the joint configuration parameters. It is assumed that the k^{th} joint possesses $r_p(k)$ positional dofs parameterized by the vector of configuration variables $\theta(k)$ (of dimension at least $r_p(k)$), and that its $r_v(k)$ motion dofs are parameterized by the $r_v(k)$ dimensional joint velocity vector $\beta(k)$. The kinematical equations which relate $\dot{\theta}(k)$ to $\beta(k)$ depend on the specific nature of the k^{th} joint. It is assumed for notational convenience that all the joint constraints are homogeneous (i.e., catastatic). $H(k)$ is defined such that $H^*(k)$ is the $6 \times r_v(k)$ joint map matrix for the k^{th} joint whose columns span the space of permissible relative spatial velocities $\Delta_V(k)$ across the joint. The complexity of the dynamics algorithms for the serial chain is determined by the number of overall motion dofs $\mathcal{N} \triangleq \sum_{k=1}^{n} r_v(k)$ for the chain. The state of the multibody system is defined by the collection of $[\theta(.), \beta(.)]$ for all the joints, and is assumed known.

Since each link is rigid, it suffices to develop the equations of motion at a single reference point on each link, which is taken to be the inboard joint location $\mathcal{O}_k$ for the k^{th} link. With $V(k)$ denoting the spatial velocity, $\alpha(k)$ the spatial acceleration, $f(k)$ the spatial force and $T(k)$ the joint force at $\mathcal{O}_k$ for the k^{th} link, the following Newton–Euler recursive equations describe the equations of motion for the serial rigid multibody chain:

$$\left\{ \begin{array}{rcl} & V(n+1) = 0, \quad \alpha(n+1) = 0 & \\ \textbf{for k} & = & \mathbf{n \cdots 1} \\ V(k) & = & \phi^*(k+1,k)V(k+1) + H^*(k)\beta(k) \\ \alpha(k) & = & \phi^*(k+1,k)\alpha(k+1) + H^*(k)\dot{\beta}(k) + a(k) \\ \textbf{end loop} & & \end{array} \right. \tag{2.1}$$

$$\left\{ \begin{array}{rcl} & f(0) = 0 & \\ \textbf{for k} & = & \mathbf{1 \cdots n} \\ f(k) & = & \phi(k+1,k)f(k-1) + M(k)\alpha(k) + b(k) \\ T(k) & = & H(k)f(k) \\ \textbf{end loop} & & \end{array} \right.$$

$a(k)$ and $b(k)$ are the velocity dependent Coriolis acceleration and gyroscopic forces repectively for the k^{th} link at $\mathcal{O}_k$. $\phi(k, k-1)$ denotes the transformation operator from $\mathcal{O}_{k-1}$ to $\mathcal{O}_k$. For additional details regarding the derivation of these equations of motion, see [1]. We have made the simplifying assumption

that the tip force $f(0)$ is zero. Attaching a full 6 motion dof joint between the physical base and the inertial frame allows us to easily deal with the mobile base situation. For the inverse dynamics problem, the joint accelerations $\dot{\beta}$ are known, and Eq. (2.1) represents an $O(\mathcal{N})$ computational process involving a base-to-tip recursion to compute the velocities and accelerations, followed by a tip-to-base recursion to compute the joint forces.

In order to express the equations of motion given by Eq. (2.1) in a more compact form, we define the *stacked notation*. In this notation, the $V(k)$'s, $\alpha(k)$'s etc are viewed as components of vectors V, α etc. Then Eq. (2.1) can be written in the following compact form:

$$\begin{aligned} V &= \mathcal{E}_\phi^* V + H^* \beta \\ \alpha &= \mathcal{E}_\phi^* \alpha + H^* \dot{\beta} + a \\ f &= \mathcal{E}_\phi f + M\alpha + b \end{aligned} \tag{2.2}$$

where,

$$\mathcal{E}_\phi \triangleq \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ \phi(2,1) & 0 & \dots & 0 & 0 \\ 0 & \phi(3,2) & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \phi(n,n-1) & 0 \end{pmatrix}, \quad M \triangleq \begin{pmatrix} M(1) & 0 & \dots & 0 \\ 0 & M(2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & M(n) \end{pmatrix},$$

$$H \triangleq \begin{pmatrix} H(1) & 0 & \dots & 0 \\ 0 & H(2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & H(n) \end{pmatrix} \tag{2.3}$$

However, since $\mathcal{E}_\phi$ is nilpotent ($\mathcal{E}_\phi^n = 0$),

$$\phi \triangleq (I - \mathcal{E}_\phi)^{-1} = I + \mathcal{E}_\phi + \mathcal{E}_\phi^2 + \cdots + \mathcal{E}_\phi^{n-1} = \begin{pmatrix} I & 0 & \dots & 0 \\ \phi(2,1) & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \phi(n,1) & \phi(n,2) & \dots & I \end{pmatrix} \tag{2.4}$$

where,

$$\phi(i,j) \triangleq \phi(i,i-1) \cdots \phi(j+1,j)$$

Thus Eq. (2.2) can be reexpressed in the form,

$$\begin{aligned} V &= \phi^* H^* \beta \\ \alpha &= \phi^* (H^* \dot{\beta} + a) \\ f &= \phi(M\alpha + b) = \phi M \phi^* H^* \dot{\beta} + \phi(M\phi^* a + b) \\ T &= Hf = H\phi M \phi^* H^* \dot{\beta} + H\phi(M\phi^* a + b) \\ &= \mathcal{M}\dot{\beta} + \mathcal{C}, \quad \text{where} \quad \mathcal{M} \triangleq H\phi M \phi^* H^*, \quad \text{and} \quad \mathcal{C} \triangleq H\phi(M\phi^* a + b) \end{aligned} \tag{2.5}$$

$\mathcal{M} \in \mathcal{R}^{\mathcal{N}\times\mathcal{N}}$ is the *mass matrix* for the serial chain and $\mathcal{C} \in \mathcal{R}^{\mathcal{N}}$ consists of the velocity dependent Coriolis, centrifugal and gyroscopic joint forces. In the terminology of Kane's method [3], β are the *generalized speeds* and the elements of $\phi^* H^*$ are the *partial (spatial) velocities*.

$\mathcal{E}_\phi$, ϕ, H, and M are the first of the *spatial operators* that will be encountered. Recursive dynamical algorithms can be derived naturally by exploiting the special *state transition* properties [1] of the elements of spatial operators such as $\mathcal{E}_\phi$, ϕ etc.. For instance, given a vector y, the evaluation of the matrix-vector product ϕy *does not* require an $O(n^2)$ matrix-vector product computation, and not even the explicit computation

of the elements of ϕ, but rather, it can be evaluated using an $O(n)$ recursive algorithm involving only the elements of $\mathcal{E}_\phi$ and y. This is precisely the correspondence between the concise operator based high–level description of the equations of motion in Eq. (2.5) and the recursive algorithmic description in Eq. (2.1).

Spatially recursive $O(\mathcal{N})$ forward dynamics algorithms for serial chains have been developed in [4] based on the recognition of the isomorphism between the structure of the dynamics equations and the equations encountered in Kalman Filtering theory. These insights have formed the basis for the development of the Spatial Operator Algebra Framework for multibody dynamics.

2.2 Tree Topology Systems

In this section, the dynamics of rigid multibody systems with tree topological structure are discussed. A tree topology system may be viewed as a set of component serial chains (referred to as *branches*) coupled together via joints at their *terminal* links. The total number of branches is denoted ℓ. The index for the branches thus ranges from $1 \cdots \ell$, and consistent with the link numbering scheme in the previous section, the inboard branches are assigned indices larger than those for the outboard ones. The *connectivity function* $\imath(k)$ is defined as the index of the *direct predecessor* branch, i.e., the inboard branch to which the k^{th} branch is connected. The j^{th} branch is simply denoted a *predecessor* branch for the k^{th} branch if it belongs on the unique path from the k^{th} branch to the base, i.e., if $\imath^p(k) = j$ for some integer $p > 0$. The joint coupling two branches is assigned to the outboard branch. Figure 1 illustrates the link/branch numbering convention for tree topology systems.

The notation for serial chains from Section 2.1 is carried over to describe the branches in the tree structure, and an additional subscript is used to identify the specific branch in the system. Thus n_j and $\mathcal{N}_j$ denote the number of links and the number of motion dofs respectively, while V_j, M_j, $\mathcal{E}_{\phi_j}$, ϕ_j etc. denote the appropriate spatial velocity etc. quantities for the j^{th} branch. A link/joint is identified by the index of the branch it is on, plus its location within the branch. For instance, $V(k_j)$ (or more accurately $V_j(k)$) denotes the spatial velocity of the the k^{th} link of the j^{th} branch at its inboard joint location $\mathcal{O}(k_j)$. The overall stacked spatial velocity, acceleration etc. vectors for the tree are now denoted V, α, f etc. with $V \triangleq [V_1^* \ \cdots \ V_\ell^*]^*$ etc.. The total number of links n, and the total number of motion dofs $\mathcal{N}$ for the system are given by

$$n \triangleq \sum_{j=1}^{\ell} n_j \quad \text{and} \quad \mathcal{N} \triangleq \sum_{j=1}^{\ell} \mathcal{N}_j \tag{2.6}$$

Note that when the j^{th} branch is the direct predecessor of the k^{th} branch, i.e., $j = \imath(k)$, the joint connecting them is the n_k^{th} joint on the k^{th} branch and is located on link 1_j on the j^{th} branch. The transformation operator from the n_k^{th} joint to the 1_j^{th} joint is denoted $\phi(1_j, n_k)$. The spatial operator $\mathcal{E}_\phi$ is now defined in terms of its block matrix elements below. For $j, k \in 1 \cdots \ell$,

$$\mathcal{E}_\phi(j,k) = \begin{cases} \mathcal{E}_{\phi_j} & \text{for} \quad j = k \\ \begin{pmatrix} 0 & \cdots & 0 & \phi(1_j, n_k) \\ 0 & \cdots & 0 & 0 \\ \vdots & \cdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \end{pmatrix} & \text{for} \quad j = \imath(k), \text{ i.e. if } j \text{ is the direct predecessor branch of } k \\ 0 & \text{for} \quad j \neq \imath(k), \text{ i.e. if } j \text{ is } \textit{not} \text{ the direct predecessor of } k \end{cases} \tag{2.7}$$

0 denotes a zero matrix of dimension appropriate for the context. As a consequence of the numbering scheme used here, for $j < k$, the j^{th} branch cannot be a predecessor to the k^{th} branch and thus the $(j,k)^{th}$ block

element, $\mathcal{E}_\phi(j,k) = 0$. Thus $\mathcal{E}_\phi$ is a strictly lower triangular matrix. The analogs of Eq. (2.2) are as follows:

$$\begin{aligned} V &= \mathcal{E}_\phi^* V + H^* \beta \\ \alpha &= \mathcal{E}_\phi^* \alpha + H^* \dot{\beta} + a \\ f &= \mathcal{E}_\phi f + M\alpha + b \end{aligned} \tag{2.8}$$

Once again (analogous to Eq. (2.4)), $\mathcal{E}_\phi$ is nilpotent ($\mathcal{E}_\phi^n = 0$), and so

$$\phi \triangleq (I - \mathcal{E}_\phi)^{-1} = I + \mathcal{E}_\phi + \mathcal{E}_\phi^2 + \cdots + \mathcal{E}_\phi^{n-1} \tag{2.9}$$

The block structure of ϕ is described below:

$$\phi(j,k) = \begin{cases} \phi_j & \text{for } j = k \\ \{\phi(m_j, l_k)\}_{m,l} & \text{if } \exists\, p > 0 : j = i^p(k), \text{ i.e., if } j \text{ is a predecessor branch of } k \\ 0 & \text{if } j \neq i^p(k)\ \forall\, p > 0, \text{ i.e., if } j \text{ is } \textit{not} \text{ a predecessor branch of } k \end{cases} \tag{2.10}$$

Here $\{\phi(m_j, l_k)\}_{m,l}$ denotes a block matrix whose $(m,l)^{th}$ entry is given by $\phi(m_j, l_k)$ with $m \in 1 \cdots n_j$ and $l \in 1 \cdots n_k$. $\phi(m_j, l_k)$ is the transformation operator from joint l_k (on the k^{th} branch) to joint m_j (on the j^{th} branch) and is a generalization of the transformation operator $\phi(i,j)$ in Eq. (2.4) for serial chains. It is formed by sequentially composing all the individual transformation operators that lie on the *unique* path joining the two joints. The numbering scheme used here ensures that ϕ will be a lower triangular matrix. The operator ϕ has state transition properties analogous to the ϕ for serial chains, and as a consequence, it can be used for high–level and concise description of the dynamics of tree topology systems (as in Eq. (2.11) below), but with the full understanding that from the computational perspective, these equations directly map into recursive implementation procedures. From Eq. (2.8) and Eq. (2.4) it follows that,

$$\begin{aligned} V &= \phi^* H^* \beta \\ \alpha &= \phi^*(H^* \dot{\beta} + a) \\ f &= \phi(M\alpha + b) = \phi M \phi^* H^* \dot{\beta} + \phi(M\phi^* a + b) \\ T &= Hf = H\phi M \phi^* H^* \dot{\beta} + H\phi(M\phi^* a + b) \\ &= \mathcal{M}\dot{\beta} + \mathcal{C}, \quad \text{where} \quad \mathcal{M} \triangleq H\phi M \phi^* H^*, \quad \text{and} \quad \mathcal{C} \triangleq H\phi(M\phi^* a + b) \end{aligned} \tag{2.11}$$

$\mathcal{M} \in \mathcal{R}^{\mathcal{N}\times\mathcal{N}}$ denotes the mass matrix for the tree system. $\hat{T} \triangleq T - \mathcal{C}$ can be easily computed from the knowledge of the system state, and so the equations of motion for the system can be rewritten in the form

$$\mathcal{M}\dot{\beta} = \hat{T} \tag{2.12}$$

The forward dynamics problem requires then the solution of the joint accelerations $\dot{\beta}$ for a given set of joint forces $\hat{T}$. The mass matrix for the system is typically not available and potentially needs to be computed to solve the forward dynamics problem. However, in Section 3, a recursive $O(\mathcal{N})$ forward dynamics algorithm for tree topology systems, which does not require the explicit computation of the mass matrix $\mathcal{M}$, is derived.

Before proceeding on to closed topology systems, we first derive the structure of the Jacobian operator. Given n_C points, denoted C_k's, on the tree (see Figure 1), the Jacobian operator $J \in \mathcal{R}^{6n_C \times \mathcal{N}}$ defines the mapping between β and $\tilde{V}$, i.e., $\tilde{V} = J\beta$, where $\tilde{V} \in \mathcal{R}^{6n_C}$ denotes the vector of spatial velocities at these points. If C_k is on link m_j, then the spatial velocity at C_k is given by

$$\tilde{V}(k) = \phi^*(\mathcal{O}(m_j), C_k) V(m_j)$$

with $\phi(\mathcal{O}(m_j), C_k)$ denoting the rigid body transformation operator from C_k to the point $\mathcal{O}(m_j)$. With the block elements of $B \in \mathcal{R}^{6n \times 6n_C}$ defined as

$$B(m_j, k) = \begin{cases} \phi(\mathcal{O}(m_j), C_k) & \text{if } C_k \in m_j^{th} \text{ link} \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } k = 1 \cdots n_C \tag{2.13}$$

it follows that

$$\tilde{V} = B^* V = B^* \phi^* H^* \beta, \quad \text{i.e.,} \quad J = B^* \phi^* H^* \tag{2.14}$$

This gives us an expression for the desired Jacobian operator which will be used below when dealing with loop closure constraints for closed topology systems.

2.3 Closed Topology Systems

This paper develops a systematic procedure for the formulation of the equations of motion and derivation of forward dynamics solution algorithms for general topology multibody systems with time-varying topologies as well as changing constituent bodies. Based on the specific application, such systems may be conceptually partitioned as follows:

(a) The *primary* system consisting of the least time-variant part, i.e., the multibody subsystem with fixed topology and constituent bodies.

(b) The *secondary* system consisting of the multibody subsystem which may change from time to time.

(c) The set of *closure constraints* and/or boundary conditions between/within the primary and secondary systems which change with changes in the system topology.

Note the the subsystems described above are in the order of increasing time-variation. As an example, let us examine the robotics scenario of multiple manipulators interacting with each other and the environment to perform complex tasks. In this context, the manipulators belong to the primary system since their innate structure varies very little with time. The task objects vary from task to task and form the secondary system. The constraints between these two subsystems change during the execution of a task, such as grasping, mating, tool operation etc., and belong to the last category.

This partitioning allows us to derive a very general and yet systematic procedure for the development of dynamics algorithms which are responsive and adaptable to time-varying systems. The procedure involves a sequence of decoupled steps for each of the primary and secondary system dynamics, and one step in which they come together when the constraint forces are computed. Being structurally time-invariant, it is possible to put in place optimized algorithms for the dynamics of the primary system. The time-variant secondary system is typically of small complexity and thus the use of standard, though suboptimal, algorithms does not substantively degrade performance.

This decomposition of the closed topology system is a departure from the more traditional approach (see [5, 6]) of forming a spanning tree for the full system and computing the constraint forces at the points of closure. In these latter approaches, even small changes in the original system typically lead to whole new spanning trees for the system. This disallows any algorithmic optimization, and the algorithms are also not very amenable to coping with time-varying systems.

The primary and secondary systems in most applications have tree topological structure. However in general there may be internal closed loops within either system. In any case, by cutting an appropriate number of joints, each subsystem may be regarded as a tree topology system with additional kinematical constraints at the internal loop closure points. The equations of motion for tree topology systems derived in Eq. (2.12) will be used to describe the dynamics of the tree components of both the primary and secondary systems, with the subscripts "P" and "S" differentiating the two subsystems. Thus the dynamics of the tree part of the two systems are described by

$$\hat{T}_P = H_P \phi_P M_P \phi_P^* H_P^* \dot{\beta}_P = \mathcal{M}_P \dot{\beta}_P, \quad \text{and} \quad \hat{T}_S = H_S \phi_S M_S \phi_S^* H_S^* \dot{\beta}_S = \mathcal{M}_S \dot{\beta}_S \tag{2.15}$$

$\mathcal{M}_P$ and $\mathcal{M}_S$ denote the mass matrices, β_P and β_S the motion dof parameter vectors, $\hat{T}_P$ and $\hat{T}_S$ the bias-free internal joint forces for the primary and secondary subsystems respectively.

Combining the internal loop points of closure with the points of closure coupling the two systems, we obtain the overall points of closure for each of the subsystems. Let $\tilde{V}_P$ and $\tilde{V}_S$ denote the spatial velocities at these overall points of closure for the two systems, and following the discussion leading to Eq. (2.14), let $J_P = B_P^* \phi_P^* H_P^*$ and $J_S = B_S^* \phi_S^* H_S^*$ denote the Jacobian operators for the two systems corresponding to these points. Thus $\tilde{V}_P = J_P \beta_P$ and $\tilde{V}_S = J_S \beta_S$. The kinematical constraints due to the existence of internal closed loops within the primary and secondary systems leads to constraint equations of the form:

$$Q_P \tilde{V}_P = \tilde{U}_P \quad \text{and} \quad Q_S \tilde{V}_S = \tilde{U}_S$$

The coupling together of the primary and secondary systems via joints leeds to constraint equations of the form:

$$\tilde{Q}_P \tilde{V}_P + \tilde{Q}_S \tilde{V}_S = \tilde{U}_C$$

Defining

$$A_P \triangleq \begin{pmatrix} \tilde{Q}_P \\ Q_P \\ 0 \end{pmatrix}, \quad A_S \triangleq \begin{pmatrix} \tilde{Q}_S \\ 0 \\ Q_S \end{pmatrix}, \quad \text{and} \quad A \triangleq [A_P \;\; A_S]$$

the closure constraints can be collectively expressed in the form:

$$A \begin{pmatrix} \tilde{V}_P \\ \tilde{V}_S \end{pmatrix} = [A_P \;\; A_S] \begin{pmatrix} J_P & 0 \\ 0 & J_S \end{pmatrix} \begin{pmatrix} \beta_P \\ \beta_S \end{pmatrix} = [A_P J_P \;\; A_S J_S] \begin{pmatrix} \beta_P \\ \beta_S \end{pmatrix} = \begin{pmatrix} \tilde{U}_C \\ \tilde{U}_P \\ \tilde{U}_S \end{pmatrix} \triangleq \tilde{U} \qquad \textbf{(2.16)}$$

It is assumed that $[A_P J_P \;\; A_S J_S]$ is of full row rank $\mathcal{N}_E$. The overall number of motion dofs of the closed chain system is given by $\mathcal{N}_C \triangleq \mathcal{N} + \mathcal{N}_S - \mathcal{N}_E$, and if necessary, Eq. (2.16) can be used to find the $\mathcal{N}_C$ dimensional minimal set of motion generalized coordinates for system. Based on the principle of virtual work, Eq. (2.16) implies that the closure constraint joint forces are of the form

$$\begin{pmatrix} J_P^* A_P^* \\ J_S^* A_S^* \end{pmatrix} \tilde{f}$$

for some $\tilde{f} \in \mathcal{R}^{\mathcal{N}_E}$. This leads to the following overall equations of motion:

$$\begin{pmatrix} \mathcal{M}_P & 0 & J_P^* A_P^* \\ 0 & \mathcal{M}_S & J_S^* A_S^* \\ A_P J_P & A_S J_S & 0 \end{pmatrix} \begin{pmatrix} \dot{\beta}_P \\ \dot{\beta}_S \\ \tilde{f} \end{pmatrix} = \begin{pmatrix} \hat{T}_P \\ \hat{T}_S \\ U \end{pmatrix}, \quad \text{where} \quad U \triangleq \dot{\tilde{U}} - [(\dot{A_P J_P}) \;\; (\dot{A_S J_S})] \begin{pmatrix} V_P \\ V_S \end{pmatrix}$$

$$\Rightarrow \quad \begin{pmatrix} \mathcal{M}_P & 0 & J_P^* A_P^* \\ 0 & \mathcal{M}_S & J_S^* A_S^* \\ 0 & 0 & -[A_P \Lambda_P A_P^* + A_S \Lambda_S A_S^*] \end{pmatrix} \begin{pmatrix} \dot{\beta}_P \\ \dot{\beta}_S \\ \tilde{f} \end{pmatrix} = \begin{pmatrix} \hat{T}_P \\ \hat{T}_S \\ U - [A_P J_P \mathcal{M}_P^{-1} \hat{T}_P + A_S J_S \mathcal{M}_S^{-1} \hat{T}_S] \end{pmatrix} \qquad \textbf{(2.17)}$$

where

$$\Lambda_P \triangleq J_P \mathcal{M}_P^{-1} J_P^*, \quad \text{and} \quad \Lambda_S \triangleq J_S \mathcal{M}_S^{-1} J_S^*$$

Note that Λ_P and Λ_S are the effective "admittances" of the primary and secondary systems reflected to the points of closure. We now describe some special cases of the above setup:

- The joint constraints coupling the primary and secondary systems are typically on the relative spatial velocity across the joints at the points of closure. When this is true for all joints, an appropriate reordering of the elements of $\tilde{V}$ will result in $\tilde{Q}_P = -\tilde{Q}_S$. Furthermore, if no relative motion is permitted across the joint, i.e., there is rigid rather than loose coupling, then in fact $\tilde{Q}_P = I$ and $\tilde{Q}_S = I$. When this is the case for only some of the joints, only the corresponding rows have these special features.
- If the secondary system has no internal actuators or source of generalized forces, then $T_S = 0$.
- If the secondary system is a free rigid body with no internal degrees of freedom, then the motion generalized coordinates vector β_S is of dimension 6 and consists of the 3 translational and 3 rotational dof parameters.

3 Forward Dynamics of Closed Chain Systems

In this section we discuss a recursive method for solving the forward dynamics of closed chain rigid multibody systems. This method does not require the explicit computation of the primary and secondary tree system mass matrices $\mathcal{M}_P$ or $\mathcal{M}_S$, but does require the computation of the constraint force parameters.

From the equations of motion of the closed chain system with dynamical closure constraints given by Eq. (2.17), the solution of the forward dynamics problem can be solved by the following sequence of steps:

(A)	Solve $\mathcal{M}_P\dot{\beta}^I = \hat{T}_P$ for $\dot{\beta}^I_P$	Solve $\mathcal{M}_S\dot{\beta}^I_S = \hat{T}_S$ for $\dot{\beta}^I_S$
(B)	Compute $\tilde{\alpha}^I_P = J_P\dot{\beta}^I_P$	Compute $\tilde{\alpha}^I_S = J_S\dot{\beta}^I_S$
(C)	Compute $\Lambda_P = J_P\mathcal{M}_P^{-1}J_P^*$	Compute $\Lambda_S = J_S\mathcal{M}_S^{-1}J_S^*$
(D)	Solve $[A_P\Lambda_PA_P^* + A_S\Lambda_SA_S^*]\tilde{f} = (A_P\tilde{\alpha}^I_P + A_S\tilde{\alpha}^I_S) - U$ for $\tilde{f}$	
(E)	Solve $\mathcal{M}_P\dot{\beta}^\delta_P = -J_P^*A_P^*\tilde{f}$ for $\dot{\beta}^\delta_P$	Solve $\mathcal{M}_S\dot{\beta}^\delta_S = -J_S^*A_S^*\tilde{f}$ for $\dot{\beta}^\delta_S$
(F)	$\dot{\beta}_P = \dot{\beta}^I_P + \dot{\beta}^\delta_P$	$\dot{\beta}_S = \dot{\beta}^I_S + \dot{\beta}^\delta_S$

As a result of the partitioning, a changes in the closure constraints only effect A and thus only STEP D, while changes in the secondary system effect only the steps in the right half column. Recursive algorithms for carrying out each of these steps are derived below. The proofs of the various lemmas are omitted due to space limitations. However they follow precisely along the lines of the proofs for serial chains discussed in [7]. The explicit use of the subscripts indentifying the primary/secondary system is dropped (except for STEP (D)) since the discussion is equally applicable to either subsystem.

STEP (A) Solve $\mathcal{M}\dot{\beta}^I = \hat{T}$. (Forward Dynamics of a Tree Topology System)

Note that Step (A) is equivalent to solving the forward dynamics of a tree topology system, and we develop an $O(N)$ recursive algorithm for this solution. This algorithm is based on a new factorization of the mass matrix $\mathcal{M}$ in terms of square factors. which may be contrasted with the earlier non–square factorization in Eq. (2.11). This square factorization is then used to obtain an explicit expression for $\mathcal{M}^{-1}$.

The *articulated body inertia matrix* P is defined as the solution to the following equation:

$$M = P - \mathcal{E}_\phi[P - PH^*(HPH^*)^{-1}HP]\mathcal{E}_\phi^* \tag{3.1}$$

P is *block diagonal* and the elements on the diagonal (denoted $P(k_j)$) can be obtained using a recursive algorithm described in Eq. (A.1) in Appendix A. Physically, $P(k_j)$ is the *articulated body inertia* as seen at the k_j^{th} joint, i.e., it is the effective inertia of all the links outboard from the k_j^{th} joint assuming that the joint forces at all the outboard joints are zero.

For the subsequent development, it is convenient to define

$$\begin{aligned} D &\triangleq HPH^*, \quad G \triangleq PH^*D^{-1}, \quad K \triangleq \mathcal{E}_\phi G \\ \tau &\triangleq GH, \quad \overline{\tau} \triangleq I - \tau, \quad \mathcal{E}_\psi \triangleq \mathcal{E}_\phi\overline{\tau} \end{aligned} \tag{3.2}$$

Note that D, G, τ and $\overline{\tau}$ are all block diagonal. The structure of $\mathcal{E}_\psi$ is identical to that of $\mathcal{E}_\phi$ with its elements being given by

$$\psi(k_j, k_j - 1) \triangleq \phi(k_j, k_j - 1)\overline{\tau}(k_j - 1)$$

$\mathcal{E}_\psi$ is also nilpotent ($\mathcal{E}_\psi^n = 0$), and analogous to ϕ, ψ is defined as

$$\psi \triangleq (I - \mathcal{E}_\psi)^{-1} = I + \mathcal{E}_\psi + \mathcal{E}_\psi^2 + \cdots + \mathcal{E}_\psi^{n-1} \tag{3.3}$$

The structure of ψ is very similar to that of ϕ and it also possesses the state transition properties which are used to develop recursive algorithms. ϕ may be viewed as the transformation operator for composite bodies (i.e., as if all the joints are locked), while ψ is the transformation operator for articulated bodies (i.e., as if all the joint forces were zero). The following lemma yields a square factorization of $\mathcal{M}$.

Lemma 1: The mass matrix $\mathcal{M}$ has the following factorization:

$$\mathcal{M} = [I + H\phi K]D[I + H\phi K]^*, \tag{3.4}$$ ∎

The following lemma gives the explicit form for the inverse of $[I + H\phi K]$.

Lemma 2:

$$[I + H\phi K]^{-1} = [I - H\psi K] \tag{3.5}$$ ∎

Combining Lemma 1 and Lemma 2 leads to the following form for the inverse of the mass matrix.

Lemma 3:

$$\mathcal{M}^{-1} = [I - H\psi K]^* D^{-1}[I - H\psi K] \tag{3.6}$$ ∎

Thus,

$$\dot{\beta}^J = \mathcal{M}^{-1}\hat{T} = [I - H\psi K]^* D^{-1}[I - H\psi K]\hat{T} \tag{3.7}$$

The $O(\mathcal{N})$ recursive computation of the expression on the right is given in Eq. (A.2) in Appendix A.

STEP (B) Compute $\tilde{\alpha}^J = J\dot{\beta}^J$

From Eq. (2.14), $\tilde{\alpha}^J = B^*\hat{\alpha}^J$, where

$$\hat{\alpha}^J \triangleq \phi^* H^* \dot{\beta}^J \tag{3.8}$$

However we have that,

Lemma 4:

$$(I - H\psi K)H\phi = H\psi \tag{3.9}$$ ∎

Thus using Eq. (3.6) and the above lemma in Eq. (3.8),

$$\hat{\alpha}^J = \phi^* H^*[I - H\psi K]^* D^{-1}[I - H\psi K]\hat{T} = \psi^* H^* D^{-1}[I - H\psi K]\hat{T}$$

Comparing this with Eq. (3.7) we see that $\hat{\alpha}^J$ can be evaluated as an intermediate quantity in the $O(\mathcal{N})$ recursive algorithm for computing $\dot{\beta}^J$ described in STEP (A).

STEP (C) Compute $\Lambda = J\mathcal{M}^{-1}J^*$

Using Eq. (2.14) and Eq. (3.6),

$$\begin{aligned} \Lambda &= \{[I - H\psi K]H\phi B\}^* D^{-1}\{[I - H\psi K]H\phi B\} \\ &= B^*\psi^* H^* D^{-1} H\psi B = B^*\Omega B, \quad \text{where} \quad \Omega \triangleq \psi^* H^* D^{-1} H\psi \end{aligned} \tag{3.10}$$

where Eq. (3.9) has been used to simplify the above expression. A recursive $O(\mathcal{N})$ procedure for the computation of Ω is given in Eq. (A.5) in Appendix A. Note that without the simplification resulting from the use of Eq. (3.9), the computation of Λ would be an $O(\mathcal{N}^3)$ process.

STEP (D) Solve $[A_P \Lambda A_P^* + A_S \Lambda_S A_S^*]\tilde{f} = (A_P \tilde{\alpha}_P^f + A_S \tilde{\alpha}_S^f) - U$ for $\tilde{f}$

Now,

$$\tilde{f} = [A_P \Lambda A_P^* + A_S \Lambda_S A_S^*]^{-1}[(A_P \tilde{\alpha}_P^f + A_S \tilde{\alpha}_S^f) - U] \tag{3.11}$$

In this form this step is of $O(\mathcal{N}_E^3)$ complexity. However, when $(A_P \Lambda_P A_P^*)$ is invertible, we can obtain an alternative expression for $\tilde{f}$ by reexpressing Eq. (2.17) as follows:

$$\begin{pmatrix} \mathcal{M}_P & 0 & J_P^* A_P^* \\ 0 & \mathcal{M}_S & J_S^* A_S^* \\ 0 & A_S J_S & -A_P \Lambda_P A_P^* \end{pmatrix} \begin{pmatrix} \dot{\beta} \\ \dot{\beta}_S \\ \tilde{f} \end{pmatrix} = \begin{pmatrix} \hat{T}_P \\ \hat{T}_S \\ U - A_P \tilde{\alpha}_P^f \end{pmatrix}$$

and consequently,

$$\begin{pmatrix} \mathcal{M}_P & 0 & J_P^* A_P^* \\ 0 & \mathcal{M}_S + J_S^* A_S^* (A_P \Lambda A_P^*)^{-1} A_S J_S & 0 \\ 0 & A_S J_S & -A_P \Lambda_P A_P^* \end{pmatrix} \begin{pmatrix} \dot{\beta} \\ \dot{\beta}_S \\ \tilde{f} \end{pmatrix} =$$

$$\begin{pmatrix} \hat{T}_P \\ \hat{T}_S + J_S^* A_S^* (A_P \Lambda_P A_P^*)^{-1}[U - A_P \tilde{\alpha}_P^f] \\ U - A_P \tilde{\alpha}_P^f \end{pmatrix}$$

From the above equation it follows that

$$\begin{aligned} \dot{\beta}_S &= [\mathcal{M}_S + J_S^* A_S^* (A_P \Lambda_P A_P^*)^{-1} A_S J_S]^{-1}[T_S - J_S^* A_S^* (A_P \Lambda A_P^*)^{-1}(A_P \tilde{\alpha}_P^f - U)] \\ \tilde{f} &= (A_P \Lambda A_P^*)^{-1}[(A_P \tilde{\alpha}_P^f + A_S J_S \dot{\beta}_S) - U] \\ &= (A_P \Lambda_P A_P^*)^{-1}[(A_P \tilde{\alpha}_P^f + A_S \tilde{\alpha}_S) - U], \quad \text{where} \quad \tilde{\alpha}_S \triangleq J_S \dot{\beta}_S \end{aligned}$$

Note the similarity between the forms of Eq. (3.11) and the above equation for $\tilde{f}$. The computational cost of the above operation is a combination of the cost of inverting $A_P \Lambda_P A_P^*$, and the $O(\mathcal{N}_S^3)$ step of solving a square linear system of equations of size $\mathcal{N}_S$. The cost of inverting $A_P \Lambda_P A_P^*$ depends on its structure: its sparsity reflects the degree of coupling between the closed loops in the system. The cost is typically much less than the worst case of $O(\mathcal{N}_E^3)$. In many application domains such as robotics, $A_P \Lambda_P A_P^*$ is in fact block diagonal and is thus invertible in $O(\mathcal{N}_E)$ steps [1]. In addition, for most applications $\mathcal{N}_S \ll \mathcal{N}_E$, and this new formulation can lead to considerable computational savings.

The inverse of $[A_P \Lambda_P A_P^* + A_S \Lambda_S A_S^*]$ will not exist if $[A_P J_P \quad A_S J_S]$ is not of full rank, i.e., the configuration is such that the number of motion dofs for the system have changed. It is therefore necessary to reformulate the constraint equation Eq. (2.16) so as to preserve the full rank property. Such changes of rank can occur at kinematically singular configurations.

STEP (E) Compute $\beta^\delta = -\mathcal{M}^{-1} J^* A^* \tilde{f}$

We have from Eq. (3.6) and Eq. (2.14) that

$$\beta^\delta = -[I - H\psi K]^* D^{-1}[I - H\psi K] H \phi B A^* \tilde{f}$$

Using Lemma 4 this simplifies to

$$\beta^\delta = -[I - H\psi K]^* D^{-1} H \psi B A^* \tilde{f} \tag{3.12}$$

The recursive $O(\mathcal{N})$ implementation of the above step is given in Eq. (A.6) in Appendix A.

The overall complexity of this spatially recursive forward dynamics algorithm ranges between $O(\mathcal{N}+\mathcal{N}_S)+O(\mathcal{N}_E^3)$ for the worst case and $O(\mathcal{N}+\mathcal{N}_S)+O(\mathcal{N}_E)+O(\mathcal{N}_S^3)$ in the best case.

By treating the primary and secondary system as one system, which amounts to defining the quantities $\psi \triangleq diag(\psi_P, \psi_S)$, $H \triangleq diag(H_P, H_S)$ etc., and using the above results, the overall closed topology forward dynamics algorithm can be restated in the following form:

$$\dot{\beta} = [I - H\psi K]^* D^{-\frac{1}{2}} \Big[I - b(A\Lambda A^*)^{-1} b^* \Big] D^{-\frac{1}{2}} [I - H\psi K]\hat{T}, \quad \text{where} \quad b \triangleq H\psi B A^* \tag{3.13}$$

Note that when there are no closed loops in the overall system, $A = 0$, and the middle term reduces to I, and we recover the form for for the forward dynamics of tree topology systems in Eq. (3.7).

4 Flexible Multibody Dynamics

In this section we briefly describe the extensions to handle the case of flexible links. We use the serial chain discussed in Section 2.1 as an illustrative example, but now assume that the links in the chain are flexible. It is assumed (without losing any generality) that finite element models are available for all the links, and in particular, the k^{th} link is characterized by: n_k node points with the location of of the j^{th} node denoted $\mathcal{Q}_k(j)$'s, the vector of displacement variables $u_k^f \in \mathcal{R}^{6n_k}$, a free–free mass–matrix $m_k \in \mathcal{R}^{6n_k \times 6n_k}$, a stiffness matrix $K_k \in \mathcal{R}^{6n_k \times 6n_k}$. The ordering of the nodes is such that $\mathcal{Q}_k(1)$ is on the same element as $\mathcal{O}_{k-1}$ and $\mathcal{Q}_k(n_k)$ is on the same element as $\mathcal{O}_k$. Treating the k^{th} link as being pinned at $\mathcal{O}_k$, this implies that $u_k^f(n_k) = 0$, and thus the true flexible dofs are given by the vector $u_k = h_k u_k^f$, where $h_k^T \equiv [I, 0]$. Note that $u_k \in \mathcal{R}^{6(n_k-1)}$ and $h_k \in \mathcal{R}^{6n_k \times 6(n_k-1)}$.

With $V(k)$ denoting the spatial velocity of the k^{th} link at $\mathcal{O}_k$,

$$\begin{aligned} V(k) &= \phi^*(k+1,k)V(k+1) + H^*(k)\beta(k) + \phi^*\Big(\mathcal{Q}_{k+1}(1), \mathcal{O}_k\Big) u_{k+1}(1) \\ &= \phi^*(k+1,k)V(k+1) + H^*(k)\beta(k) + C_{k+1}^* h_k^* u_{k+1} \qquad (4.1) \\ &\text{where} \quad C_{k+1}^* \triangleq [\phi^*\Big(\mathcal{Q}_{k+1}(1), \mathcal{O}_k\Big),\ 0,\ \cdots\ 0] \in \mathcal{R}^{6 \times 6n_k} \qquad (4.2) \end{aligned}$$

Thus,

$$V = \phi^*[H^*\beta + C^* h^* u]$$

with C defined as the block matrix with C_2 to C_n along its first block subdiagonal, and h is the block diagonal matrix with j^{th} block diagonal element being h_j. Let $V_k^f \in \mathcal{R}^{6n_k}$ denote the vector of spatial velocities on the k^{th} link at the n_k node points. Then

$$\begin{aligned} V_k^f &= B_k^* V(k) + h_k^* u_k, \quad \text{where} \quad B_k \triangleq [\phi(\mathcal{O}_k, \mathcal{Q}_k(1)), \cdots, \phi(\mathcal{O}_k, \mathcal{Q}_k(n_k))] \in \mathcal{R}^{6 \times 6n_k} \\ \Rightarrow \quad V^f &= B^* V + h^* u = B^*\phi^*[H^*\beta + C^* h^* u] + h^* u = B^*\phi^* H^*\beta + [I + B^*\phi^* C^*]h^* u \\ &= [I + B^*\phi^* C^*][h^* \ \ B^* H^*]X, \quad \text{where} \quad X \triangleq \begin{pmatrix} u \\ \beta \end{pmatrix} \qquad (4.3) \end{aligned}$$

B denotes the block diagonal matrix with the B_k's along its diagonal. We have used the facts that,

$$B_k C_k = \phi(k, k-1) \quad \Rightarrow \quad BC = \mathcal{E}_\phi \quad \Rightarrow \quad BC\phi = \phi - I \quad \Rightarrow \quad B[I + C\phi B] = \phi B$$

Note that X is the vector of motion dofs for the serial links and includes both the rigid and flexible dof parameters.

Using Eq. (4.3), the kinetic energy for the whole chain is given by

$$T = \frac{1}{2}(V^f)^* m V^f = \frac{1}{2} X^* \mathcal{M}^f X, \quad \text{where} \quad \mathcal{M}^f \triangleq \begin{pmatrix} HB \\ h \end{pmatrix} [I + C\phi B] m [I + C\phi B]^* \begin{pmatrix} HB \\ h \end{pmatrix}^*$$

$\mathcal{M}^I$ is the mass matrix for the flexible serial chain. Given this factored form for the mass matrix, similar techniques to those used for the rigid multibody case in the earlier sections result in alternate factorizations and inversion of the mass matrix, and recursive forward dynamics algorithms. The reader is referred to [8] for additional details. Just as for the rigid multibody case, the algorithms for flexible serial chains directly extend to the flexible general topology multibody systems.

5 Conclusions

This paper describes the Spatial Operator Algebra Framework for the dynamics of general multibody systems. Based on their rate of time-variation, the multibody system is partitioned into a primary subsystem, a secondary subsystem and the set of closure constraints. This allows the development of forward dynamics algorithms which are not only recursive and efficient, but also capable of easily coping with time varying multibody systems. The solution procedure consists of a sequence of steps on parallel paths involving the dynamics of the spanning trees for the primary and secondary systems. The two paths come together for one step in order to compute the constraint forces. Using the spatial algebra techniques to develop novel factorizations of the mass matrix and operator identities, efficient recursive algorithms for carrying out each of these steps is developed. The overall algorithm does not require the computation of the mass matrix, and its complexity is linear in the number of dofs for the tree systems. In addition, the impact on the complexity of the algorithm, of the degree of coupling among the closed loops in the system topology is made clear, and it is shown that the in the best circumstance, the algorithmic complexity is also linear in the number of closure constraint equations. During the development, an $O(\mathcal{N})$ forward dynamics algorithm for tree topology systems is also developed. For the sake of clarity, the focus of much of the paper was on multibody systems with rigid links. However the extensions necessary to deal with flexible elements are discussed.

6 Acknowledgement

The research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

References

[1] G. Rodriguez and K. Kreutz, "Recursive mass matrix factorization and inversion: an operator approach to open and closed-chain multibody dynamics," JPL Publication 88-11, Jet Propulsion Laboratory, Pasadena, CA, 1988.

[2] A. Jain, "Unified formulation of dynamics for serial rigid multibody systems," Eng. Memo. 347-89-264, (Internal Document), Jet Propulsion Laboratory, Pasadena, CA, 1989.

[3] T. Kane and D. Levinson, *Dynamics: Theory and Applications.* McGraw-Hill, 1985.

[4] G. Rodriguez, "Kalman filtering, smoothing and recursive robot arm forward and inverse dynamics," *IEEE J. Robotics Automat.*, vol. 3, Dec. 1987. (JPL Publication 86-48, 1986).

[5] H. Brandl, R. Johanni, and M. Otter, "An algorithm for the simulation of multibody systems with kinematic loops," in *World Congress on the Theory of Machines and Mechanisms (7th), Seville, Spain,* 1987.

[6] D. Bae and E. Haug, "A recursive formulation for constrained mechanical system dynamics: Part II. Closed loop systems," *Mech. Struct. & Mach.*, vol. 15, no. 4, pp. 481-506, 1987-88.

[7] G. Rodriguez, K. Kreutz, and A. Jain, "A spatial operator algebra for manipulator modeling and control," in *IEEE Conf. Rob. and Aut., Scottsdale, Az*, May 1989.

[8] G. Rodriguez, "Spatial operator approach to multibody manipulator inverse and forward dynamics," in *IEEE Conf. Rob. and Aut., Cincinnati, OH*, May 1990.

A Appendix

Based on the special structure of ϕ, ψ etc., it is possible to evaluate many of the dynamical expressions in a recursive manner and we describe some recursive algorithms in this appendix. First we define some notational shorthand to simplify the description of the algorithms that follow:

$$\begin{aligned} x(n_j+1) &\Longrightarrow x(1_{\imath(j)}) \\ y(n_j+1, n_j) &\Longrightarrow y(1_{\imath(j)}, n_j) \\ y(1_j, 0_j)x(0_j) &\Longrightarrow \sum_{m \in \imath^{-1}(j)} y(1_j, n_m)x(n_m) \\ y(1_j, 0_j)x(0_j)y^*(1_j, 0_j) &\Longrightarrow \sum_{m \in \imath^{-1}(j)} y(1_j, n_m)x(n_m)y^*(1_j, n_m) \end{aligned}$$

where $y(.,.)$ and $x(.)$ stand for some appropriate arrays. Thus wherever a term with indices as in the left column appears, its meaning is actually given by the corresponding term in the column on the right. !pr

- A recursive method for the computation of the block diagonal elements of P as defined by Eq. (3.1) and the entries of D, G, K, $\mathcal{E}_\psi$ and $\overline{\tau}$ defined in Eq. (3.2) are given by:

$$\left\{\begin{array}{l} \textbf{for j} = \mathbf{1 \cdots \ell} \\ \quad \left\{\begin{array}{rcl} & \text{If } \imath^{-1}(j) = \emptyset, \text{ then } P(0_j) = 0 & \\ \textbf{for k} = \mathbf{1_j \cdots n_j} & & \\ P(k) &=& \psi(k, k-1)P(k-1)\psi^*(k, k-1) + M(k) \\ D(k) &=& H(k)P(k)H^*(k) \\ G(k) &=& P(k)H^*(k)D^{-1}(k) \\ \overline{\tau}(k) &=& I - G(k)H(k) \\ \psi(k+1, k) &=& \phi(k+1, k)\overline{\tau}(k) \\ K(k+1, k) &=& \phi(k+1, k)G(k) \\ \textbf{end loop} & & \end{array}\right. \\ \textbf{end loop} \end{array}\right. \qquad \textbf{(A.1)}$$

- The recursive computation of $\dot{\beta}^J = [I - H\psi K]^* D^{-1}[I - H\psi K]\hat{T}$ in Eq. (3.7) in STEP (A) can be carried out via the $O(\mathcal{N})$ tree topology forward dynamics algorithm described below. It also results in the computation of $\hat{\alpha}^J = \psi^* D^{-1}[I - H\psi K]\hat{T}$ required in STEP (B) as an intermediate quantity.

$$\left\{\begin{array}{l} \textbf{for j} = \mathbf{1 \cdots \ell} \\ \quad \left\{\begin{array}{rcl} & \text{If } \imath^{-1}(j) = \emptyset, \text{ then } z(0_j) = 0, \hat{T}(0_j) = 0 & \\ \textbf{for k} = \mathbf{1_j \cdots n_j} & & \\ z(k) &=& \psi(k, k-1)z(k-1) + K(k, k-1)\hat{T}(k-1) \\ \epsilon(k) &=& T(k) - H(k)z(k) \\ \nu(k) &=& D^{-1}(k)\epsilon(k) \\ \textbf{end loop} & & \end{array}\right. \\ \textbf{end loop} \end{array}\right.$$

$$
\left\{
\begin{array}{l}
\hat{\alpha}^f(n_\ell + 1) = 0 \\
\textbf{for j} = \boldsymbol{\ell \cdots 1} \\
\quad \left\{
\begin{array}{l}
\textbf{for k} = \mathbf{n_j \cdots 1_j} \\
\quad \hat{\alpha}^f(k) = \psi^*(k+1,k)\hat{\alpha}^f(k+1) + H^*(k)\nu(k) \\
\quad \hat{\beta}^f(k) = \nu(k) - K^*(k+1)\hat{\alpha}^f(k+1) \\
\textbf{end loop}
\end{array}
\right. \\
\textbf{end loop}
\end{array}
\right. \tag{A.2}
$$

- STEP (C) requires the computation of $\Lambda = B^*\Omega B$. In order to obtain a $O(\mathcal{N})$ recursive scheme for the computation of Ω we first define the matrix Υ as the one satisfying the equation:

$$H^*D^{-1}H = \Upsilon - \mathcal{E}_\psi^*\Upsilon\mathcal{E}_\psi \tag{A.3}$$

Υ as defined above is a block diagonal matrix and its elements can be computed recursively. We now obtain the following decomposition of Ω.

Lemma 5:

$$\Omega = \Upsilon + \tilde{\psi}^*\Upsilon + \Upsilon\tilde{\psi} \tag{A.4}$$ ■

Noting that $\tilde{\psi}$ is strictly lower triangular, we can then recognize that Υ as nothing but the diagonal elements of Ω. We now present a recursive scheme to compute the block diagonal elements of Υ and of Ω.

$$
\left\{
\begin{array}{l}
\Upsilon(n_\ell + 1) = 0 \\
\textbf{for j} = \boldsymbol{\ell \cdots 1} \\
\quad \left\{
\begin{array}{l}
\textbf{for k} = \mathbf{n_j \cdots 1_j} \\
\quad \Upsilon(k) = \psi^*(k+1,k)\Upsilon(k+1)\psi(k+1,k) + H^*(k)D^{-1}(k)H(k) \\
\quad \left\{
\begin{array}{l}
\Omega(k,k) = \Upsilon(k) \\
\textbf{for m} = \mathbf{k-1 \cdots 1_j} \\
\quad \Omega(k,m) = \Omega^*(m,k) = \Upsilon(k,m+1)\psi(m+1,m) \\
\textbf{end loop}
\end{array}
\right. \\
\textbf{end loop}
\end{array}
\right. \\
\textbf{end loop}
\end{array}
\right.
$$

The above recursion yields the elements Ω_j on the block diagonal of Ω. Since Ω is symmetric, the off-diagonal elements satisfy $\Omega_{j,l} = \Omega_{l,j}^*$, and can be computed from the diagonal elements as follows. $\Omega_{l,j}$ for $l \in 1 \cdots (j-1)$ can be obtained via the following recursive scheme:

$$
\left\{
\begin{array}{l}
\textbf{if } \imath^p(\mathbf{l}) = \mathbf{j} \textbf{ for some } p > 0 \\
\quad \left\{
\begin{array}{l}
\textbf{for k} = \mathbf{n_j \cdots 1_j} \\
\quad \left\{
\begin{array}{l}
\textbf{for m} = \mathbf{n_l \cdots 1_l} \\
\quad \Omega(k,m) = \Omega^*(m,k) = \Omega(k,1_j)\psi(1_j,m) \\
\textbf{end loop}
\end{array}
\right. \\
\textbf{end loop}
\end{array}
\right. \\
\textbf{else} \\
\quad \Omega_{j,l} = \Omega_{l,j}^* \equiv 0 \\
\textbf{end if}
\end{array}
\right. \tag{A.5}
$$

- The $O(\mathcal{N})$ recursive implementation of $\beta^b = -[I - H\psi K]^* D^{-1} H\psi B A^* \tilde{f}$ in Eq. (3.12) in Step (e) is given below:

$$
\left\{
\begin{array}{l}
\qquad \text{Define } \hat{x} \triangleq -BA^*\tilde{f} \\
\textbf{for j} = 1 \cdots \ell \\
\quad \left\{
\begin{array}{l}
\qquad \text{If } i^{-1}(j) = \emptyset, \text{ then } z(0_j) = 0, \hat{x}(0_j) = 0 \\
\textbf{for k} = 1_j \cdots n_j \\
\quad z(k) = \psi(k, k-1)z(k-1) + K(k, k-1)\hat{x}(k-1) \\
\quad \epsilon(k) = -H(k)z(k) \\
\quad \nu(k) = D^{-1}(k)\epsilon(k) \\
\textbf{end loop}
\end{array}
\right. \\
\textbf{end loop}
\end{array}
\right.
$$

$$
\left\{
\begin{array}{l}
\qquad \alpha(n_\ell + 1) = 0 \\
\textbf{for j} = \ell \cdots 1 \\
\quad \left\{
\begin{array}{l}
\textbf{for k} = n_j \cdots 1_j \\
\quad \alpha(k) = \psi^*(k+1, k)\alpha(k+1) + H^*(k)\nu(k) \\
\quad \dot{\beta}^b(k) = \nu(k) - K^*(k+1)\alpha(k+1) \\
\textbf{end loop}
\end{array}
\right. \\
\textbf{end loop}
\end{array}
\right. \qquad \textbf{(A.6)}
$$

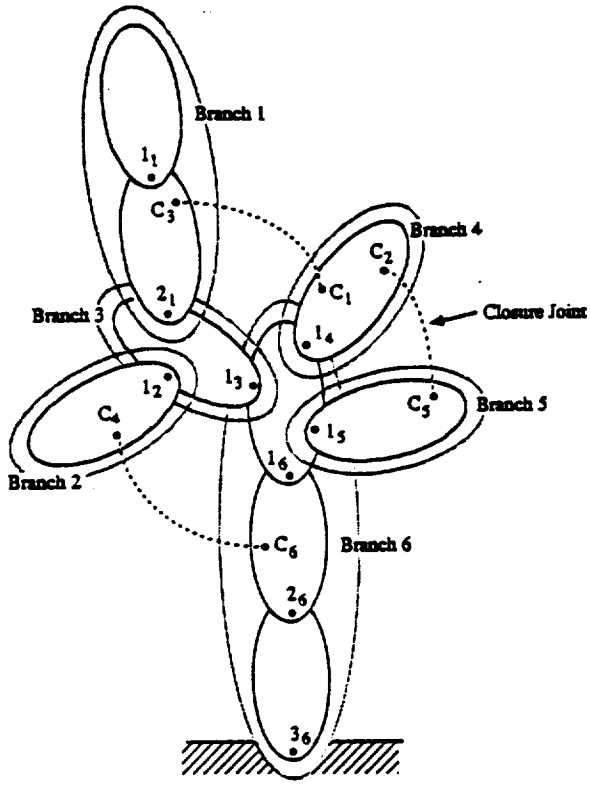


Figure 1: Illustration of link/branch numbering convention for multibody system

An Order (n) Algorithm for the Dynamics Simulation of Robotic Systems

H.M. Chun, J.D. Turner, and H.P. Frisch
NASA Goddard Space Flight Center

Abstract

The ability to simulate and analyze the dynamics of complicated multibody systems has been of great benefit to engineers for the past decade. Applications have included robotics, land vehicles, and spacecraft. However, many of the commercially available software have been computationally intensive and are costly and time-consuming for analyzing large systems. Fortunately, recent developments in Order (n) algorithms and parallel processing for multibody dynamics simulation have drastically reduced the computer time needed to simulate systems involving many bodies.

This paper presents the formulation of an Order (n) algorithm for DISCOS (Dynamics Interaction Simulation of Controls and Structures), which is an industry-standard software package for simulation and analysis of flexible multibody systems. For systems involving many bodies, the new Order (n) version of DISCOS is much faster than the current version. Results of the experimental validation of the dynamics software are also presented. The experiment is carried out on a seven-joint robot arm at NASA's Goddard Space Flight Center.

The algorithm used in the current version of DISCOS requires the inverse of a matrix whose dimension is equal to the number of constraints in the system. Generally, the number of constraints in a system is roughly proportional to the number of bodies in the system, and matrix inversion requires $O(p^3)$ operations, where p is the dimension of the matrix. The current version of DISCOS is therefore considered an Order (n^3) algorithm. In contrast, the Order (n) algorithm requires inversion of matrices which are small, and the number of matrices to be inverted increases only linearly with the number of bodies.

The newly-developed Order (n) DISCOS is currently capable of handling chain and tree topologies as well as multiple closed loops. Continuing development will extend the capability of the software to deal with typical robotics applications such as put-and-place, multi-arm hand-off and surface sliding.

Aspects of Efficient and Reliable Multibody System Simulation

R. Schwertassek and C. Führer
Institut für Dynamik der Flugsysteme
and
W. Rulka
MAN Technologie

Abstract

Multibody system equations can be generated in various forms. All of these may be interpreted as results of two basic approaches, the augmentation- and the elimination-method. The former method yields the descriptor form of the system motion, a set of differential-algebraic equations (DAE), and the latter the state space representation, a minimal set of ordinary differential equations (ODE). Both of these methods are surveyed. Particular emphasis is on the discussion of recursive computational schemes, generating the equations of motion with a number of operations, which is proportional to the number N of system bodies ($O(N)$-formulations).

For simulation purposes one would like to create that set of system equations, which can be generated most efficiently and for which the most efficient and reliable solution techniques are available. Numerical solution techniques for ODE have been studied in great detail and they are well-developed. By contrast, DAE have not been investigated for such a long time. In view of new developments in the latter field the generation of all the equations required for an efficient and reliable solution of DAE describing multibody system motion is discussed. These methods, i.e. an $O(N)$-formulation and new techniques for solving DAE, are implemented in the SIMPACK code. Its capabilities are illustrated by simulation of multibody robot models.

SYSTEMATIC GENERATION OF MULTIBODY EQUATIONS OF MOTION SUITABLE FOR RECURSIVE AND PARALLEL MANIPULATION

Parviz E. Nikravesh, Gwanghun Gim, Ara Arabyan, Udo Rein

Computer-Aided Engineering Laboratory
Aerospace and Mechanical Engineering Department
University of Arizona
Tucson, AZ 85721

ABSTRACT

This paper summarizes the formulation of a method known as the joint coordinate method for automatic generation of the equations of motion for multibody systems. For systems containing open or closed kinematic loops, the equations of motion can be reduced systematically to a minimum number of second order differential equations. The application of recursive and nonrecursive algorithms to this formulation, computational considerations and the feasibility of implementing this formulation on multiprocessor computers are discussed.

1. INTRODUCTION

In the past decade, the joint (or natural) coordinate method has been implemented in formulating the equations of motion. The methodology for open loop systems is well developed in a variety of forms [1-5]. For these systems, the method yields a minimal set of equations of motion since the joint coordinates are independent. The joint coordinates are no longer independent when closed kinematic loops exist in a system. For multibody systems containing simple closed loops, constraint equations between joint coordinates may be derived easily. However, for most spatial closed loops, the derivation of these constraints can be rather complicated. A simple method for automatic generation of the closed loop constraints, and a technique to generate a minimal set of differential equations of motion has been reported in reference [6].

This paper briefly describes the method of joint coordinates for the systematic generation of the equations of motion for open and closed loop systems. These equations are shown to be suitable for recursive and nonrecursive algorithms, serial or parallel processing, and symbolic manipulation. While the discussion principally focuses on multi-rigid-body systems, the assumption of rigidity may be relaxed by introducing the finite element technique in modeling the deformation of bodies. This formulation has been incorporated in a set of large-scale computer programs which have been used extensively to simulate the dynamic behavior of a variety of multibody systems.

2. EQUATIONS OF MOTION

The equations of motion for a multibody mechanical system can be expressed in different forms depending upon the choice of the coordinates and velocities that describe the configuration and motion of the system. In the following subsections, the equations of motion are first expressed in terms of absolute coordinates and velocities of the bodies in the system. Then these equations are reduced to a minimal set of equations for open-loop systems. Furthermore, the equations are generalized for systems containing closed kinematic loops.

2.1 Absolute Coordinate Formulation

In order to specify the position of a rigid body in a global non-moving xyz coordinate system, it is sufficient to specify the spatial location of the origin (center of mass) and the angular orientation of a body-fixed $\xi\eta\zeta$ coordinate system as shown in Fig. 1. For the ith body in a multibody system, vector $\mathbf{q}_i$ denotes a vector of coordinates which contains a vector of Cartesian translational coordinates $\mathbf{r}_i$ and a set of rotational coordinates. Matrix $\mathbf{A}_i$ represents the rotational transformation of the $\xi\eta\zeta_i$ axes relative to the xyz axes. A vector of velocities for body i is defined as $\mathbf{v}_i$, which contains a 3-vector of translational velocities $\dot{\mathbf{r}}_i$ and a 3-vector of angular velocities $\boldsymbol{\omega}_i$. The components of the angular velocity vector $\boldsymbol{\omega}_i$ are defined in the xyz coordinate system rather than the body-fixed coordinate system. A vector of accelerations for this body is denoted by $\dot{\mathbf{v}}_i$, which contains $\ddot{\mathbf{r}}_i$ and $\dot{\boldsymbol{\omega}}_i$. For a multibody system containing b bodies, the vectors of coordinates, velocities, and accelerations are $\mathbf{q}$, $\mathbf{v}$, and $\dot{\mathbf{v}}$ which contain the elements of $\mathbf{q}_i$, $\mathbf{v}_i$, and $\dot{\mathbf{v}}_i$, respectively, for i = 1, ..., b.

The kinematic joints between the bodies can be described by m-independent holonomic constraints as

$$\boldsymbol{\Phi}(\mathbf{q}) = \mathbf{0} \tag{1}$$

The first and second time derivatives of the constraints yield the kinematic velocity and acceleration equations

$$\dot{\boldsymbol{\Phi}} \equiv \mathbf{D}\mathbf{v} = \mathbf{0} \tag{2}$$

$$\ddot{\boldsymbol{\Phi}} \equiv \mathbf{D}\dot{\mathbf{v}} + \dot{\mathbf{D}}\mathbf{v} = \mathbf{0} \tag{3}$$

where D is the Jacobian matrix of the constraints. The equations of motion are written as [7]

$$\mathbf{M}\dot{\mathbf{v}} - \mathbf{D}^T\boldsymbol{\lambda} = \mathbf{g} \tag{4}$$

where $\mathbf{M}$ is the inertia matrix containing the mass and the inertia tensor of all bodies, $\boldsymbol{\lambda}$ is a vector of m Lagrange multipliers, and $\mathbf{g} = \mathbf{g}(\mathbf{q}, \mathbf{v})$ contains the gyroscopic terms and the forces and moments that act on the system. The inertia matrix is not a constant matrix since the rotational equations of motion are written in terms of the global components of the angular accelerations. The term $\mathbf{D}^T\boldsymbol{\lambda}$ in Eq. 4 represents the joint reaction forces and moments. Equations 1-4 represents a set of differential-algebraic equations of motion for a multibody system when absolute coordinates are used. These equations will have the same form whether the system is open- or closed-loop.

2.2 Joint Coordinates and Open Loop Systems

The constrained equations of motion expressed by Eqs. 1-4 can be converted to a smaller set of equtions in terms of a set of coordinates known as joint coordinates. For multibody systems with open kinematic loops, this conversion yields a set of differential equations equal to the number of degrees of freedom. We may consider one branch of an open-loop system as shown in Fig. 2. The bodies are numbered in any desired order. The relative configurations of two adjacent bodies are defined by one or more so-called joint (or natural) coordinates equal in number to the number of relative degrees of freedom between these bodies. For example, $\boldsymbol{\theta}_{ij}$ contains two angles if there is a universal joint between bodies i and j, or it contains only one translational variable if there is a prismatic joint between the two bodies. The vector of joint coordinates for the system is denoted by $\boldsymbol{\theta}$ containing all of the joint coordinates and the absolute coordinates of a base (reference) body if the base body is not the ground (floating base). Therefore, vector $\boldsymbol{\theta}$ has a dimension equal to the number of degrees of freedom of the system. The vector of joint velocities is defined as $\dot{\boldsymbol{\theta}}$, which is the time derivative of $\boldsymbol{\theta}$. It can be shown that there is a linear transformation between $\dot{\boldsymbol{\theta}}$ and $\mathbf{v}$ as [1-4]

$$\mathbf{v} = \mathbf{B}\dot{\boldsymbol{\theta}} \tag{5}$$

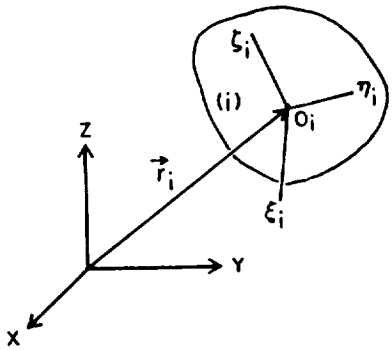


Figure 1 Body-fixed and global coordinate systems.

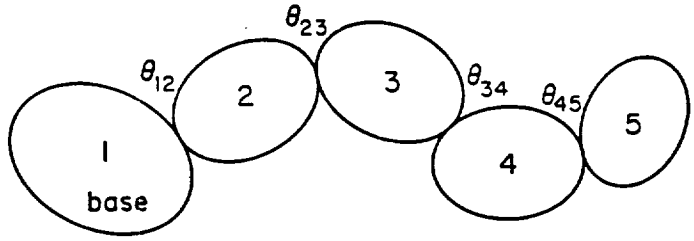


Figure 2 An open-loop system.

Matrix **B** is orthogonal to the Jacobian matrix **D**. This can be shown by substituting Eq. 5 in Eq. 2 to find $\mathbf{DB}\dot{\boldsymbol{\theta}} = \mathbf{0}$. Since $\dot{\boldsymbol{\theta}}$ is a vector of independent velocities, then

$$\mathbf{DB} = \mathbf{0} \tag{6}$$

The time derivative of Eq. 5 gives the transformation formula for the accelerations,

$$\dot{\mathbf{v}} = \mathbf{B}\ddot{\boldsymbol{\theta}} + \dot{\mathbf{B}}\dot{\boldsymbol{\theta}} \tag{7}$$

Substituting Eq. 7 in Eq. 4, premultiplying by $\mathbf{B}^T$, and using Eq. 6 yields

$$\bar{\mathbf{M}}\ddot{\boldsymbol{\theta}} = \mathbf{f} \tag{8}$$

where

$$\bar{\mathbf{M}} = \mathbf{B}^T\mathbf{M}\mathbf{B} \tag{9}$$

$$\mathbf{f} = \mathbf{B}^T(\mathbf{g} - \mathbf{M}\dot{\mathbf{B}}\dot{\boldsymbol{\theta}}) \tag{10}$$

Equation 8 represents the generalized equations of motion for an open-loop multibody system when the number of the selected coordinates is equal to the number of degrees of freedom.

2.3 **Joint Coordinates and Closed-Loop Systems** [6]

The equations of motion for open-loop systems, represented by Eq. 8, can be modified for systems containing closed kinematic loops. Assume that there is one or more closed kinematic loops in a multibody system, such as the closed-loop shown in Fig. 3(a). In order to derive the equations of motion for such a system, the closed-loop is cut at one of the kinematic joints in order to obtain an open-loop system as shown in Fig. 3(b). For this cut system, which will be called the reduced system, the joint coordinates are defined as for any open-loop system. It is clear that if we now close this system at the cut joint(s), the joint coordinates will no longer be independent, i.e., for each closed-loop there exist one or more algebraic constraints between the joint coordinates of that loop.

The constraint equations for the closed kinematic loops may be expressed implicitly as

$$\boldsymbol{\Psi}(\boldsymbol{\theta}) = \mathbf{0} \tag{11}$$

The time derivative of the constraints yields

$$\dot{\boldsymbol{\Psi}} \equiv \mathbf{C}\dot{\boldsymbol{\theta}} = \mathbf{0} \tag{12}$$

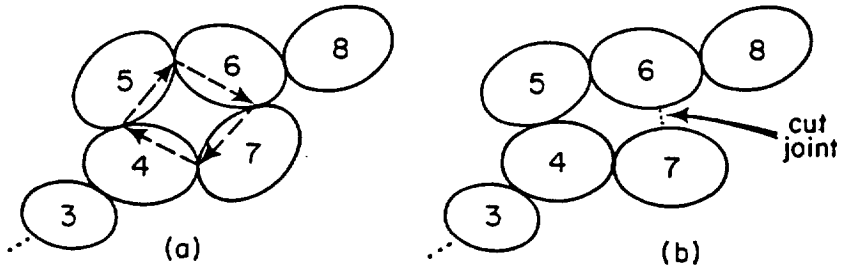


Figure 3 A system containing a closed-loop. (a) The closed-loop. (b) Its reduced open-loop representation.

where **C** is the Jacobian matrix of these constraints. Similarly the acceleration constraints for the closed-loops are written as

$$\ddot{\Psi} \equiv \mathbf{C}\ddot{\boldsymbol{\theta}} + \dot{\mathbf{C}}\dot{\boldsymbol{\theta}} = \mathbf{0} \tag{13}$$

Now the equations of motion of Eq. 8 are modified for closed loop systems as

$$\bar{\mathbf{M}}\ddot{\boldsymbol{\theta}} - \mathbf{C}^{\mathrm{T}}\boldsymbol{\nu} = \mathbf{f} \tag{14}$$

where $\boldsymbol{\nu}$ is a vector of Lagrange multipliers associated with the constraints of Eq. 11. Equations 11-14 represent the equations of motion for a multibody system when the number of selected joint coordinates is greater than the number of degrees of freedom of the system.

It is shown in [6] that the Jacobian matrix **C** in Eqs. 12-14 can be obtained systematically. If the Jacobian of the constraints for the cut-joint(s) is denoted by $\mathbf{D}^*$, then the product of this matrix and matrix **B** yields the **C** matrix as

$$\mathbf{D}^*\mathbf{B} \rightarrow \mathbf{C} \tag{15}$$

The product $\mathbf{D}^*\mathbf{B}$, in most cases, will have redundant rows. Matrix **C** is found after the redundant rows are eliminated. Since the elements of both matrices $\mathbf{D}^*$ and **B** are available explicitly in symbolic form, the elements of **C** can also be determined symbolically. Note that **C** is a small matrix in comparisson with $\mathbf{D}^*$ and **B**. Furthermore, since the elements of **C** can be determined symbolically, the term $\dot{\mathbf{C}}\dot{\boldsymbol{\theta}}$ in Eq. 13 can also be found symbolically.

2.4 Minimum Number of Equations of Motion for Closed-Loop Systems

For a multibody system containing closed kinematic loops, the Lagrange multipliers of Eq. 14 can be eliminated in order to obtain a minimal set of equations of motion in terms of a set of independent joint accelerations. For this purpose, a set of independent joint coordinates are selected as a subset of vector $\boldsymbol{\theta}$. Then a velocity transformation matrix **E** can be found such that [6]

$$\dot{\boldsymbol{\theta}} = \mathbf{E}\dot{\boldsymbol{\theta}}_{(i)} \tag{16}$$

where $\dot{\boldsymbol{\theta}}_{(i)}$ is the vector of independent joint velocity with a dimension equal to the number of degrees of freedom of the system. Note that the joint velocities outside the closed-loops and the independent joint velocities within the closed-loops are not affected by this conversion. The matrix **E** is orthogonal to the **C** matrix; i.e.,

$$\mathbf{C}\mathbf{E} = \mathbf{0} \tag{17}$$

The time derivative of Eq. 17 gives

$$\ddot{\theta} = E\ddot{\theta}_{(i)} + \dot{E}\dot{\theta}_{(i)} \tag{18}$$

Substituting of Eq. 18 in Eq. 14, premultiplying by E^T, and using Eq. 16 yields

$$\bar{M}'\ddot{\theta}_{(i)} = f' \tag{19}$$

where

$$\bar{M}' = E^T\bar{M}E \tag{20}$$

$$f' = E^T(f - \bar{M}\dot{E}\dot{\theta}_{(i)}) \tag{21}$$

Equation 19 represents the minimum number of equations of motion describing the dynamics of a multibody system containing closed kinematic loops.

Matrix E can be found in either explicit form or in numerical form for most closed kinematic loops. For this purpose, we can partition vector $\dot{\theta}$ into dependent and independent sets, $\dot{\theta}_{(d)}$ and $\dot{\theta}_{(i)}$, and correspondingly we can partition matrix C into two submatrices $C_{(d)}$ and $C_{(i)}$. Therefore, Eq. 16 becomes

$$C_{(d)}\dot{\theta}_{(d)} + C_{(i)}\dot{\theta}_{(i)} = 0$$

This equation can be written as,

$$\dot{\theta}_{(d)} = - C_{(d)}^{-1}C_{(i)}\dot{\theta}_{(i)}$$

or,

$$\dot{\theta} = \begin{bmatrix} I \\ -C_{(d)}^{-1}C_{(i)} \end{bmatrix} \dot{\theta}_{(i)} \tag{22}$$

where a proper selection of the independent joint velocities guarantees that $C_{(d)}$ is a nonsingular matrix. Comparing Eqs. 16 and 22 yields an expression for E as

$$E = \begin{bmatrix} I \\ -C_{(d)}^{-1}C_{(i)} \end{bmatrix} \tag{23}$$

Since the elements of matrix C are available explicitly, it may be possible to find the elements of E explicitly. This is due to the fact that the operation of Eq. 23 is performed separately on each closed-loop, and in addition, the C matrix for a closed-loop is relatively small in practical applications.

Equation 21 states that for evaluating the equations of motion, in addition to matrix E, matrix $\dot{E}$ is also needed. An apparent approach is to evaluate the time derivative of Eq. 23. However, since the product $\dot{E}\dot{\theta}_{(i)}$ is needed in Eq. 21, we can evaluate this product directly. For this purpose, Eq. 13 is written in a partitioned form as

$$C_{(d)}\ddot{\theta}_{(d)} + C_{(i)}\ddot{\theta}_{(i)} = -\dot{C}\dot{\theta}$$

This equation is then rearranged as

$$\ddot{\theta}_{(d)} = - C_{(d)}^{-1}C_{(i)}\ddot{\theta}_{(i)} - C_{(d)}^{-1}\gamma\dot{C}\dot{\theta}$$

or,

$$\ddot{\theta} = \begin{bmatrix} I \\ -C_{(d)}^{-1} C_{(i)} \end{bmatrix} \ddot{\theta}_{(i)} + \begin{bmatrix} 0 \\ -C_{(d)}^{-1} \hat{C} \dot{\theta} \end{bmatrix} \tag{24}$$

Comparison of Eqs. 18 and 24 yields

$$\dot{E}\dot{\theta}_{(i)} = \begin{bmatrix} 0 \\ -C_{(d)}^{-1} \hat{C} \dot{\theta} \end{bmatrix} \tag{25}$$

In the process of solving Eq. 19, the independent joint accelerations and velocities are integrated to obtain the independent joint velocities and coordinates. Then Eq. 16 (or Eq. 12) is used to find the dependent joint velocities. However, prior to that the dependent joint coordinates need to be computed. In order to find the dependent joint coordinates, the constraints of Eq. 11 must be solved for each closed-loop. These constraints are nonlinear in θ (or q) and they are not available explicitly. However, an iterative formula for finding the dependent joint coordinates can be derived. By applying the Newton-Raphson method to Eq. 11, the iterative formula is found as

$$C \, \Delta\theta = - \phi^* \tag{26}$$

where $\Delta\theta$ denotes the corrections in vector θ and ϕ^* denotes the violation in the constraints. Note that the violation in the constraints of Eq. 11 is the same as the violation in the constraints written in terms of the absolute coordinates of the bodies common to the cut joint(s). For the sake of simplicity, the iteration formula of Eq. 26 is shown in its abstract form. We assume it is understood that in this equation the known (independent) elements of θ and their corresponding columns of C have been manipulated properly. Furthermore, if there is more than one closed kinematic loop in the system, this iterative process can be applied separately to the constraint equations of each loop. The important point to draw from Eq. 26 is that explicit expressions for the joint coordinate constraints, represented by Eq. 11, are not needed.

3. COMPUTATIONAL CONSIDERATIONS

For most multibody systems, the inertia matrix **M** and the vector **g** containing the external forces/moments and the gyroscopic terms can be constructed systematically [7]. A systematic construction of the velocity transformation matrix **B**, for open or reduced open loop systems, has been shown in [4]. This matrix is constructed from the topology of the system by assembling smaller block matrices representing different type of kinematic joints. Since matrix **B** can be constructed in explicit form in terms of the absolute coordinates **q**, then matrix **B** can also be determined and expressed in explicit form as a function of **q** and **v**. For multibody systems containing closed kinematic loops, matrix C for Eqs. 11-14 and matrix E for Eq. 18 must be constructed. The process outlined in the preceding sections will be demonstrated by a simple example.

Example: Consider the multibody system shown in Fig. 4(a), containing four moving bodies. Body 1 is connected to the ground by a prismatic joint T_1, and there are four revolute joints, R_2 through R_5, with parallel axes connecting the bodies in a closed-loop. If the closed-loop is cut at R_5, the reduced system of Fig. 4(b) is obtained. For the reduced system, four joint coordinates, θ_1, θ_2, θ_3, and θ_4 are defined, where θ_1 represents relative translation between body 1 and the ground, and the other three joint coordinates represent relative rotations between the adjacent bodies. Four unit vectors, u_1 through u_4, are defined along the four joint axes. The vector of absolute velocities **v** is a 24-vector, where the vector of relative velocities $\dot{\theta}$ is a 4-vector. The velocity transformation matrix for this reduced system is:

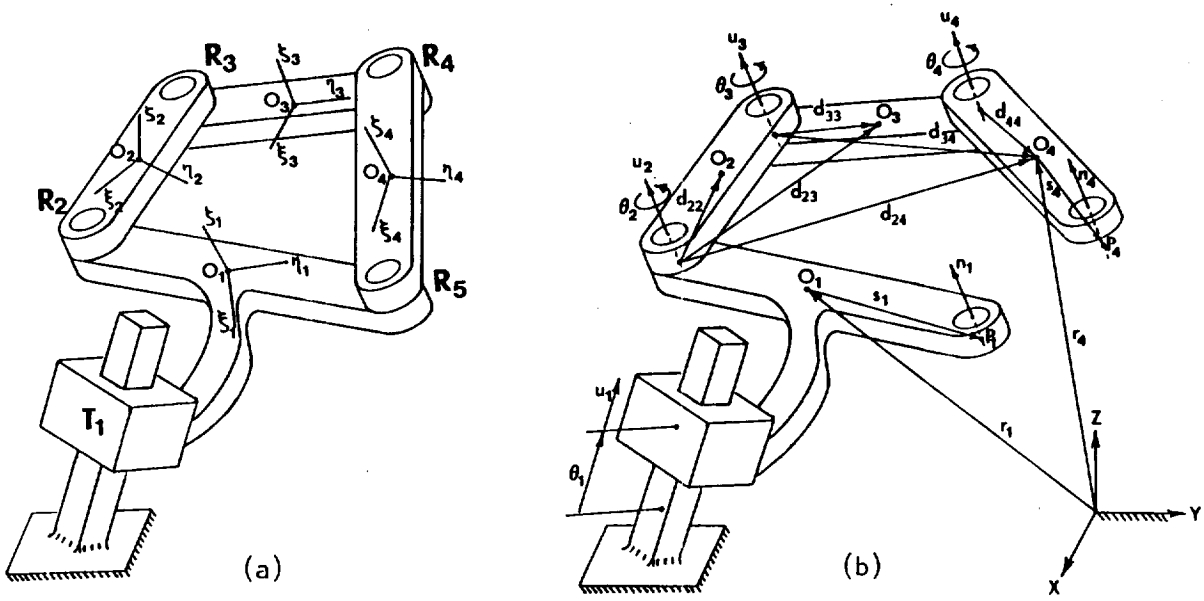


Figure 4 (a) A multibody system with four moving bodies containing a closed-loop.
(b) Its reduced open-loop representation.

$$\mathbf{B} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{u}_1 & -\tilde{\mathbf{d}}_{22}\mathbf{u}_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{u}_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{u}_1 & -\tilde{\mathbf{d}}_{23}\mathbf{u}_2 & -\tilde{\mathbf{d}}_{33}\mathbf{u}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{u}_2 & \mathbf{u}_3 & \mathbf{0} \\ \mathbf{u}_1 & -\tilde{\mathbf{d}}_{24}\mathbf{u}_2 & -\tilde{\mathbf{d}}_{34}\mathbf{u}_3 & -\tilde{\mathbf{d}}_{44}\mathbf{u}_4 \\ \mathbf{0} & \mathbf{u}_2 & \mathbf{u}_3 & \mathbf{u}_4 \end{bmatrix}$$

where $\mathbf{d}_{ij}$ vectors, for i, j = 2, 3, 4, are shown in Fig. 4(b). In this matrix, $\tilde{\mathbf{a}}$ represents a 3x3 skew-symmetric matrix made of the components of a 3-vector **a**, and $\tilde{\mathbf{a}}\mathbf{b}$ represents the cross product of two vectors **a** and **b**. The structure of **B** shows that the matrix is constructed from 6x1 block matrices $\begin{bmatrix} \mathbf{u}_1 \\ \mathbf{0} \end{bmatrix}$, representing a prismatic joint along the unit vector $\mathbf{u}_1$, and 6x1 block matrices $\begin{bmatrix} -\tilde{\mathbf{d}}_{ij}\mathbf{u}_i \\ \mathbf{u}_i \end{bmatrix}$, representing revolute joints along unit vectors $\mathbf{u}_i$, i = 2, 3, and 4 [4].

The constraint equations for the cut revolute joint R_5, i.e., Φ^*, between bodies 1 and 4 in terms of the absolute coordinates of these bodies can be expressed as [7]:

$$\Phi^3 \equiv \mathbf{r}_1 + \mathbf{s}_1 - \mathbf{r}_4 - \mathbf{s}_4 = 0$$

$$\Phi^2 \equiv \tilde{\mathbf{n}}_1\mathbf{n}_4 = 0$$

where Φ^3 represents three algebraic equations stating that two points P_1 and P_4 on the joint axis must coincide, and Φ^2 states that two unit vectors $\mathbf{n}_1$ and $\mathbf{n}_4$ along the axis of R_5 must remain parallel. It should be noted that the cross product of two vectors yields three abgebraic constraints, and only two out of three are independent. The time derivative of these constraints yields [7]:

$$\dot{\phi}^* \equiv \begin{cases} \dot{r}_1 + \tilde{\omega}_1 s_1 - \dot{r}_4 - \tilde{\omega}_4 s_4 = 0 \\ \tilde{n}_4 \tilde{n}_1 \omega_1 - \tilde{n}_1 \tilde{n}_4 \omega_4 = 0 \end{cases}$$

or,

$$\begin{bmatrix} I & -\tilde{s}_1 & 0 & 0 & 0 & 0 & -I & \tilde{s}_4 \\ 0 & \tilde{n}_4\tilde{n}_1 & 0 & 0 & 0 & 0 & 0 & -\tilde{n}_1\tilde{n}_4 \end{bmatrix} \begin{bmatrix} \dot{r}_1 \\ \omega_1 \\ \dot{r}_2 \\ \omega_2 \\ \dot{r}_3 \\ \omega_3 \\ \dot{r}_4 \\ \omega_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Therefore, the coefficients of the velocity components provide the Jacobian matrix for this cut revolute joint as:

$$D^* = \begin{bmatrix} I & -\tilde{s}_1 & 0 & 0 & 0 & 0 & -I & \tilde{s}_4 \\ 0 & \tilde{n}_4\tilde{n}_1 & 0 & 0 & 0 & 0 & 0 & -\tilde{n}_1\tilde{n}_4 \end{bmatrix}_{(5x24)}$$

Note that this is a 5x24 matrix. The product D^*B is found to be

$$D^*B = \begin{bmatrix} 0 & (\tilde{d}_{24}+\tilde{s}_4)u_2 & (\tilde{d}_{34}+\tilde{s}_4)u_3 & (\tilde{d}_{44}+\tilde{s}_4)u_4 \\ 0 & -\tilde{n}_1\tilde{n}_4 u_2 & -\tilde{n}_1\tilde{n}_4 u_3 & -\tilde{n}_1\tilde{n}_4 u_4 \end{bmatrix}_{(5x4)}$$

which is a 5x4 matrix since the columns of the matrix correspond to the four joint coordinates. Note that the first column of the matrix is all zeros, and this column corresponds to θ_1 which is not in the closed-loop. Based on the initial assumption, the four revolute joint axes are parallel, therefore, the cross product $\tilde{n}_4 u_2 = \tilde{n}_4 u_3 = \tilde{n}_4 u_4 = 0$. This means that the last two rows of the 5x4 matrix are zeros and they can be eliminated from the matrix. This leaves a 3x4 matrix as:

$$D^*B = [0 \quad (\tilde{d}_{24}+\tilde{s}_4)u_2 \quad (\tilde{d}_{34}+\tilde{s}_4)u_3 \quad (\tilde{d}_{44}+\tilde{s}_4)u_4]_{(3x4)}$$

If for a given configuration numerical values are substituted for the components of the vectors in this matrix, it will be found that the three rows are not independent -- one row is redundant and can be eliminated. For example, for a particular configuration, the numerical values of the elements of this matrix may be:

$$D^*B = \begin{bmatrix} 0 & 0 & -1.4142 & -1.4142 \\ 0 & 0 & -1.4142 & -1.4142 \\ 0 & -1.4142 & -1.4142 & 0 \end{bmatrix}_{(3x4)} \qquad (a)$$

This result should have been expected, since the closed-loop is a four-bar mechanism with one relative degree of freedom between its four bodies. Since there are three joint coordinates associated with this four-bar mechanism, there must be only two independent constraints between them. Therefore, matrix D^*B of Eq. (a) becomes

$$C = \begin{bmatrix} 0 & 0 & -1.4142 & -1.4142 \\ 0 & -1.4142 & -1.4142 & 0 \end{bmatrix}_{(2x4)} \qquad (b)$$

Since matrix C is available in explicit form, its time derivative and hence the product $\dot{C}\dot{\theta}$ can be found for Eq. 13.

After elimination of the redundant row matrix C can be written in abstract form as

$$C = \begin{bmatrix} 0 & c_1 & c_2 & c_3 \\ 0 & c_4 & c_5 & c_6 \end{bmatrix}$$

where c_1 , ..., c_6 are known explicitly from Eq. (a). If we choose θ_2 as the independent joint coordinate for the closed-loop, and noting that θ_1 is independent from the loop, we can have

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \end{bmatrix} = E \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

where

$$E = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & e_1 \\ 0 & e_2 \end{bmatrix} , \quad e_1 = \frac{c_3c_4 - c_1c_6}{c_2c_6 - c_3c_5} , \text{ and } \quad e_2 = \frac{c_1c_5 - c_2c_4}{c_2c_6 - c_3c_5}$$

If the numerical values of the elements of C from Eq. (b) are used, matrix E will be found to be

$$E = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & -1 \\ 0 & 1 \end{bmatrix}$$

4. RECURSIVE VERSUS NONRECURSIVE ALGORITHMS

The equations of motion for open or closed loop systems (Eq. 8 or Eqs. 11-14) can be generated and solved for the accelerations either recursively or nonrecursively. Since the inertia matrix $\bar{M}$ is symmetric, a standard nonrecursive technique such as L^TDL factorization can be employed to solve for the unknown accelerations. An alternative recursive approach for finding the unknown accelerations was first developed by Featherstone [8]. The idea is to remove one body from a multibody system and then to consider the remaining system as a new multibody system. Articulated body inertias are the properties which make the remaining system act as the original one. The articulated inertias are calculated by projecting the mass and inertia of the removed body onto the remaining system. Repeatedly removing one body from the multibody system leads to a system with only one body for which the accelerations can easily be calculated. The accelerations of the removed bodies will then be obtained by back substitution. Wehage interpreted this process mathematically by using a large number of equations of motion [9-11]. The unknowns of the equations are the absolute and joint accelerations, as well as joint reaction forces. The equations of motion are solved by applying matrix partitioning. This more theoretical approach allows for a very general formulation of the recursive

projection algorithm. Wehage shows that a recursive algorithm is equivalent to a "block LU factorization."

For an open loop system containing n joints, a nonrecursive matrix factorization algorithm requires a CPU time of approximately Order(n^2). For the same system, a recursive algorithm requires a CPU time of Order(n). However, the factor in front of O(n) or O(n^2) can make one algorithm more or less efficient than the other, depending upon n. Therefore, some examples are shown in this section to clarify this concept.

In the recursive algorithms, the velocity transformation matrix **B** can be represented as the product of two matrices [9],

$$\mathbf{B} = \mathbf{G}^{-1}\mathbf{H} \tag{27}$$

The matrices **G** and **H** are found from velocity transformation equations between consecutive bodies as

$$\mathbf{v}_j = \mathbf{G}_j\mathbf{v}_i + \mathbf{H}_j\dot{\boldsymbol{\theta}}_j \tag{28}$$

The equations of motion for an open loop system; i.e., Eq. 8, are then written as

$$(\mathbf{G}^{-1}\mathbf{H})^T\mathbf{M}\mathbf{G}^{-1}\mathbf{H}\ddot{\boldsymbol{\theta}} = (\mathbf{G}^{-1}\mathbf{H})^T(\mathbf{g} - \mathbf{M}\mathbf{G}^{-1}\boldsymbol{\gamma}) \tag{29}$$

where $\boldsymbol{\gamma}$ contains quadratic velocity terms which can be constructed from

$$\boldsymbol{\gamma}_j = \dot{\mathbf{G}}_j\mathbf{v}_i + \dot{\mathbf{H}}_j\dot{\boldsymbol{\theta}}_j \tag{30}$$

A detailed description of the recursive algorithm used in this study can be found in [12].

Two simple examples are considered here for comparison of the CPU times. Figure 5 shows a highly parallel and a highly serial system. In both systems, the number of joints n is increased, starting from one, between simulations. For the parallel system with only revolute joints, the nonrecursive method is more efficient than the recursive method regardless of number of joints, as shown in Fig. 6(a). It can be observed that both algorithms yield CPU times that increase almost linearly as n is increased. For the serial system, however, there is a breakpoint beyond which the recursive algorithm becomes more efficient than the nonrecursive algorithm. As shown in Fig. 6(b, c, d), the breakpoint for the number of joints n is six, nine, and five when the serial system contains only revolute joints, prismatic joints, or spherical joints respectively.

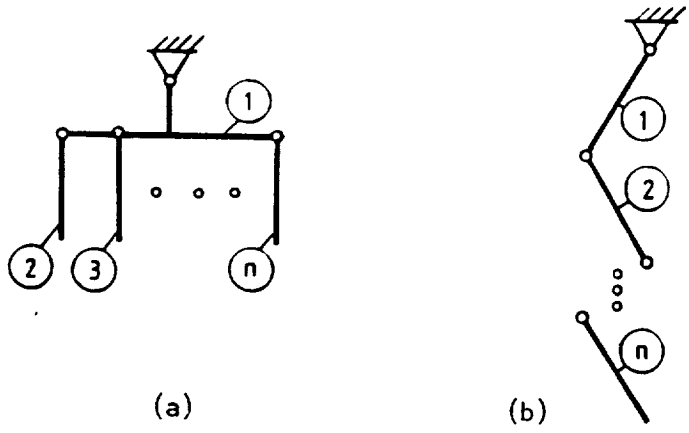


Figure 5 Two systems with (a) a highly parallel and (b) a highly serial structures.

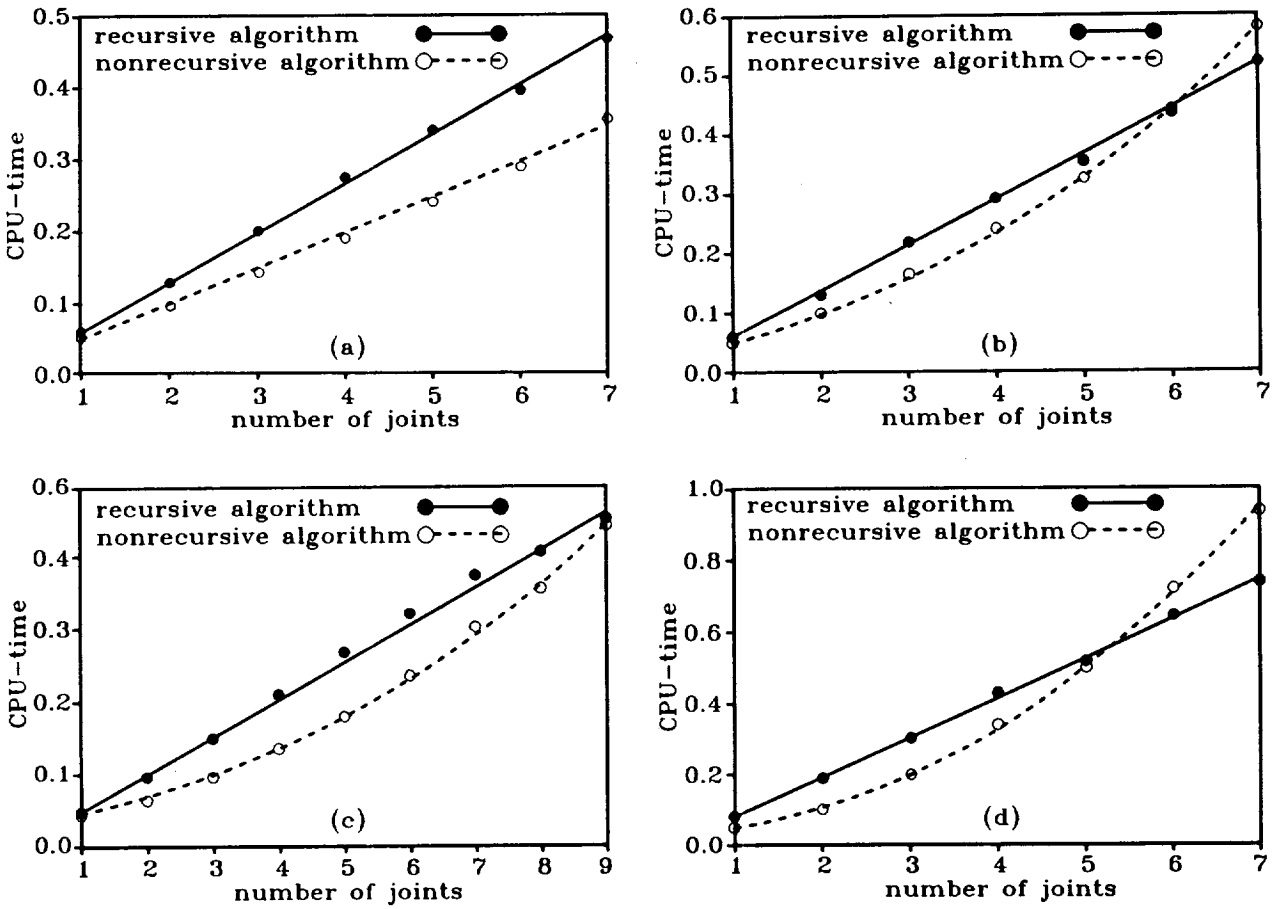


Figure 6 CPU time for one function evaluation versus number of joints for (a) a highly parallel system, and (b), (c), (d) highly serial systems containing only revolute, prismatic, and spherical joints respectively.

In reference [11], the recursive projection algorithm of open loop systems is modified for systems containing closed kinematic loops. This algorithm has been tested and the result is reported in [12]. It is shown that since a closed loop is cut at one of the joints to form a reduced open loop system, the breakpoint for the number of joints in a closed loop is approximately double of that of an open loop system. For example, if the closed loop contains only revolute joints, there should be approximately twelve or more bodies in the loop for the recursive algorithm to exhibit more efficiency than the nonrecursive method.

For mechanical systems with only rigid bodies, it is rather unlikely to have open or closed loops with enough number of bodies to make a recursive algorithm more efficient than a nonrecursive one. However, when one or more of the bodies in a system are considered as deformable, then the concept of recursive projection technique becomes highly attractive.

5. PARALLEL COMPUTATIONAL CONSIDERATIONS

When computation on a multiple-instruction multiple-data (MIMD) multiprocessing system is considered, obvious parallelisms arising from the topology (e.g. multiple branches) can be exploited. However, a true measure of the suitability of a computational scheme for parallel processing is the degree of intrinsic parallelism in the scheme for the worst case (i.e. single branch open-loop linkage).

The formulation described by Eq. 8 was applied to a 6 degree-of-freedom Stanford arm that consists of a base body plus 6 links all of which are connected to each other serially by revolute joints except link 3 which is connected to link 2 by a prismatic joint. The algorithm to compute $\ddot{\theta}$ at each time step was represented as a data-flow graph. It is known that the shortest possible time to traverse the graph from its beginning to its end is the length of the longest possible path in the graph, or the critical path.

The maximum speedup for any multiprocessing system is, therefore, the ratio of the serial computation time to the time corresponding to the critical path of the data-flow graph, since the latter is a property of the computational scheme alone. The length of the critical path was computed for a Stanford arm possessing 1 through 6 degrees of freedom. A plot of the maximum speedup with the degrees of freedom in the Stanford arm is shown in Fig. 7. The plot indicates the possibility of large speedups (11 to 76) if a suitable multiprocessing system and a proper scheduling algorithm are used. It is also observed that the length of the critical path of the data-flow graph resulting from the formulation increases linearly as the number of degrees of freedom increases in a serially connected multibody system. This suggests that the maximum speedup will approach a constant (ratio of the rate of increase of serial computation time to the rate of increase of critical path length) for a large number of degrees of freedom.

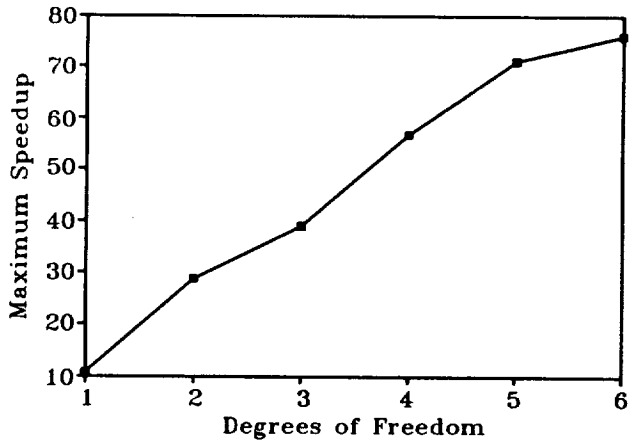


Figure 7 Maximum speedup versus number of degrees of freedom for the Stanford arm.

The velocity transformation of Eq. 5 offers other possibilities for parallel processing. It is evident from the approach described above that if the coordinate set, θ, used to describe the configuration of the system is such that the critical path of the data-flow graph does not increase or increases very slowly as new degrees of freedom are added to the system, then the maximum speedup will increase linearly or approach a very high limit at large numbers of degrees of freedom. Therefore, if matrix **B** can be chosen such that the resulting θ is this type of a coordinate set, then the formulation would be very suitable for parallel processing because of very large potential gains in speedup. Work is currently under way to determine such matrices **B** automatically on the basis of the topology imposed by each type of joint in a multibody system.

6. DEFORMABLE BODIES

In the dynamic analysis of multibody systems, the elastodynamic effects may play an important role on the behavior of the system. The most popular technique for describing the flexibility of the components of a system is the finite element method. In standard finite element formulation, the gross motion (large displacements, large deformations) is not taken into account. However, in order to analyze flexible multibody systems, such phenomena must be considered. Several researchers have suggested procedures that successfully introduce elastodynamic effects into multibody dynamics

equations [13-15]. The main problem with the inclusion of elastodynamic effects is that the flexible bodies may have a relatively complex geometry. This implies that a large number of nodes may be necessary and, therefore, a system of equations with a large number of degrees of freedom will result. In order to gain computational efficiency, a formulation based on the modal superposition method has been suggested to reduce the number of degrees of freedom of the model [14].

The method of joint coordinates for multi-rigidbody dynamics can be extended to a system of mixed rigid and flexible bodies. The equations of motion for the rigid bodies are written in terms of the absolute coordinates, similar to Eqs. 1-4, and the equations of motion for the flexible bodies are written in term of either nodal or modal coordinates. Additional kinematic constraints may be necessary to represent the connectivities between rigid-to-flexible and flexible-to-flexible bodies. Then, a velocity transformation process, similar to that of rigid bodies. (Eqs.5-10) can be applied to remove the algebraic constraints and their corresponding Lagrange multipliers. The resultant equations of motion for an open loop system can be converted to a minimal set of differential equations. In the case of closed loop systems, the equations of motion may or may not contain algebraic constraints and Lagrange multipliers.

For systems containing flexible bodies with linear elastic material properties, the equations of motion with modal coordinates can be used. However, in some applications, it may be necessary to consider the equations of motion in terms of the nodal coordinates in order to obtain more accurate results.

7. CONCLUSION

The equations of motion for multibody systems containing closed kinematic loops can be written either as a set of differential-algebraic equations (Eq. 11-14) or as a set of ordinary differential equations (Eq. 19). The elements of the constrained equations of motion given by Eqs. 11-14 can be constructed efficiently. Although all of the joint coordinates are not independent of each other, and hence the number of integration variables is not a minimum, the numerical integration of these equations can be performed efficiently. For example, a dynamic simulation of the system shown in Fig. 4, including several force elements, was performed using two different formulations -- the absolute coordinate formulation of Eqs. 1-4 and the joint coordinate formulation of Eqs. 11-14. The numerical integration of the equations of motion in both cases was carried out using a predictor-corrector Adams-Bashforth algorithm on a desktop workstation. The CPU time for simulating ten seconds of dynamic response was 352 seconds for Eqs. 1-4 and 75 seconds for Eqs. 11-14. The results obtained from these and other simulations have shown that the formulation of Eqs. 11-14 yields about five times or more efficiency over the formulation of Eqs. 1-4. The degree of efficiency depends on the number of bodies, number of joints, and the connectivity between the bodies.

Equation 19 provides the minimum number of equations of motion for a multibody system containing closed kinematic loops. The number of equations and the number of integration variables are smaller when compared to those of Eqs. 11-14. Therefore, it may appear that the numerical solution of Eq. 19 to be more efficient than that of Eqs. 11-14. However, a careful examination of the elements of Eq. 19 would reveal that the overhead associated with evaluating these elements may be more than the overhead associated with the additional number of equations and integration variables of Eqs. 11-14. Numerical simulations of several problems using the two methods have shown that the computation time associated with these two formulations are about the same.

Systematic generation of the elements of the equations of motion with the joint coordinates makes the formulation ideal for symbolic generation of these elements. Computer programs have been developed that symbolically generate the equations of motion for rigid body systems. The equations are generated in an optimized fashion to improve the computational efficiency of the dynamic simulation. The programs dealing with rigid and flexible bodies evaluate the equations of motion numerically. The equations of motion for deformable bodies can be considered either in terms of the modal or the nodal coordinates.

Another interesting feature of these equations is that the process of solving the equations of motion for unknown accelerations can be performed either recursively or nonrecursively. It has been shown that for highly serial systems with long chains, a recursive process may yield computational efficiency. Further adaptation of these equations to multiprocessor computers results in a highly efficient simulation package.

8. REFERENCES

1. Wittenburg, J., Dynamics of Systems of Rigid Bodies, B. G. Teubner, 1977.

2. Jerkovsky, W., "The Structure of Multibody Dynamics Equations," J. Guidance and Control, Vol. 1, No. 3, pp. 173-182, May-June 1978.

3. Roberson, R.E., and Schwertassek, R., Dynamics of Multibody Systems, Springer-Verlag, 1988.

4. Kim, S.S., and Vanderploeg, M.J., "A General and Efficient Method for Dynamic Analysis of Mechanical Systems Using Velocity Transformations," ASME J. Mech., Trans., and Auto. in Design, Vol. 108, No. 2, pp. 176-182, June 1986.

5. Ashrafiuon, H., Computer-Aided Optimal Design of Multibody Mechanical Systems Using Symbolic Computation, Ph.D. Dissertation, Mechanical Engineering, SUNY Buffalo, NY, 1988.

6. Nikravesh, P.E., and Gim, G., "Systematic Construction of The Equations of Motion for Multibody Systems Containing Closed Kinematic Loops," Proc., 15th ASME Design Automation Conference, Montreal, Canada, 1989.

7. Nikravesh, P.E., Computer-Aided Analysis of Mechanical Systems, Prentice Hall, 1988.

8. Featherstone, R., Robot Dynamics Algorithms, Kluwer Academic Publishers, 1987.

9. Wehage, R.A., "Application of Matrix Partitioning and Recursive Projection to O(n) Solution of Constrint Equations of Motion," Proc., 20th ASME Mechanisms Conference, Orlando, Florida, 1988.

10. Wehage, R.A., "Solution of Multibody Dynamics Using Natural Factors and Iterative Refinement -- Part I: Open Kinematic Loops," Proc., 15th ASME Design Automation Conference, Montreal, Canada, 1989.

11. Wehage, R.A., "Solution of Multibody Dynamics Using Natural Factors and Iterative Refinement -- Part II: Closed Kinematic Loops," Proc., 15th ASME Design Automation Conference, Montreal, Canada, 1989.

12. Rein, U., Implementation of a Recursive Projection Algorithm in a General Purpose Computer Program, Joint Diploma Thesis, Mechanical Engineering, Univ. of Arizona and Univ. of Stuttgart, 1989.

13. Song, J.O., and Haug, E.J., "Dynamic Analysis of Planar Flexible Mechanisms," Computer Methods in Applied Mechanics and Engineering, Vol. 24, pp. 359-381, 1980.

14. Shabana, A., and Wehage, R.A., "Variable Degree of Freedom Component Mode Analysis of Inertia Variant Flexible Mechanical Systems'" ASME J. of Mech., Trans., and Auto. in Design, Vol. 105, pp. 371-378, 1983.

15. Sunada, W., and Dubowsky, S., "The Application of Finite Element Method to the Dynamic Analysis of Flexible Spatial and Co-Planar Linkage Systems," ASME J. of Mech. Design, Vol. 103, pp. 643-651, 1981.

Recursive Linearization of Multibody Dynamics Equations of Motion

Tsung-Chieh Lin and K. Harold Yae

Center for Simulation and Design Optimization of Mechanical System
Department of Mechanical Engineering
The University of Iowa
Iowa City, Iowa 52242

Abstract

The equations of motion of a multibody system are nonlinear in nature, and thus pose a difficult problem in linear control design. One approach is to have a first-order approximation through the numerical perturbations at a given configuration, and to design a control law based on the linearized model. In this paper, a linearized model is generated analytically by following the footsteps of the recursive derivation of the equations of motion.

The equations of motion are first written in a Newton-Euler form, which is systematic and easy to construct; then, they are transformed into a relative coordinate representation, which is more efficient in computation. A new computational method for linearization is obtained by applying a series of first-order analytical approximations to the recursive kinematic relationships. The method has proved to be computationally more efficient because of its recursive nature. It has also turned out to be more accurate because of the fact that analytical perturbation circumvents numerical differentiation and other associated numerical operations that may accumulate computational error, thus requiring only analytical operations of matrices and vectors.

The power of the proposed linearization algorithm is demonstrated, in comparison to a numerical perturbation method, with a two-link manipulator and a seven degrees of freedom robotic manipulator. Its application to control design is also demonstrated.

1. Introduction

The behavior of a nonlinear dynamic system can be approximated by a linearized model in the neighborhood of a reference configuration. Intuitively, linear models of dynamic systems can be obtained by simply omitting nonlinear effects of the nonlinear dynamic systems, such as Coriolis forces, centrifugal forces, and the interaction forces between bodies. Such a model, however, cannot satisfy the needs of computer-aided design of control systems for multibody systems because the intuitive simplification is usually case-dependent. Therefore, a better linearized dynamic model based on a general purpose dynamics model is necessary. The approach that fits this requirement most is first-order approximations of the nonlinear dynamic models, which yield valid results in the neighborhood of the reference configuration for dynamic and control analysis. A straightforward approach to obtain first-order approximations of multibody systems is first to generate the analytical closed-form, nonlinear equations of motion of the systems, and then to generate the linearized equations of motion using first-order Taylor expansions. Unfortunately, these analytical equations of motion are generally not available because they are too complex to be generated.

Due to the difficulty of analytically generating the closed-form, nonlinear equations of motion of a multibody system, a numerical perturbation method is usually applied to obtain a linearized model at a certain configuration. For instance, in DISCOS [1] the numerical perturbation method is employed to generate a linearized model for stability analysis of a

multibody system at a selected configuration. Following a similar idea, Liang [2] implemented a numerical perturbation method with DADS [3] for multibody mechanical system control. Moreover, the numerical perturbation method has been widely implemented in dynamic and control analysis. For example, Sohoni and Whitesell [4] applied it in ADAMS, and Singh, Likins, and Vendervoortt applied [5] it to generate linear models of flexible body systems.

In the implementation of the numerical perturbation method, iterative computations are employed to ensure that the resulting linearized models are accurate. However, the iterative computation sometimes may not generate a satisfactory linearized model because of failure in convergence. Therefore, a trade-off between the accuracy and convergence must be made to generate a useful linearized model. An accurate linearized model is difficult to be generated with good computational efficiency when the numerical perturbation approaches are applied. In resolving this problem, the numerical perturbation method must be avoided during the linearization procedure.

On the other hand, symbolic programming languages can be used to devise efficient computational techniques to obtain the linearized manipulator models. Vukobratovic and Nenad [6] proposed the linearization technique that first generates the nonlinear dynamic models of the manipulator by means of symbolic programming languages, and then takes first-order approximation from the given nonlinear model. Following the same approach, Neuman and Murray [7] linearized symbolically the Lagrangian dynamic robot model about a nominal trajectory to generate the linearized and trajectory sensitivity models of a manipulator. Balafoutis, Misra, and Patel [8] further extended Neuman's approach to obtain more computational efficiency in generating linearized models by using the fact that the derivatives of trigonometric functions need not be computed explicitly, and that the partial derivatives of the homogeneous transformation matrices may be obtained merely by row and column manipulations. The same idea was applied to generate linearized models for flexible multibody systems by Jonker [9]. Thus, although this approach has the advantage of not using the numerical perturbation method, there is at the same time a disadvantage: it relies heavily on symbolic programming languages. Consequently, the approach is restricted to special case studies only until a general purpose symbolic manipulation package for the dynamic modelling becomes available.

In searching for a general purpose computer-aided dynamic analysis algorithm, Bae and Haug [10,11,12] developed a recursive formulation, which was later improved by Bae, Hwang and Haug [13,14]. In this approach, the equations of motion are first written in a Newton-Euler form, which is systematic and easy to construct. They are then transformed into a relative coordinate representation, which is efficient for computation. This approach is extended in this paper to efficiently generate a linearized model using the recursive computational structure and applying the analytical linear approximations of the recursive kinematic relationships, without applying numerical perturbations. The computational efficiency and opportunity for parallelism of the recursive algorithm would make it possible to linearize successively for adaptive dynamics control.

An analytical linearization algorithm is derived by using the recursive variational derivation, and by linearizing kinematic relationships analytically. In the recursive formulation, the equations of motion are obtained through a series of coordinate transformations. By analytically taking first-order approximations of kinematic relationships between Cartesian, state vector, and joint variables and then applying these linearized relationships in the recursive variational derivation, linearized equations of motion are generated in joint space. The proposed linearization algorithm is shown in Fig. 1 and is explained as follows:

(1) Variational equations of motion are obtained in Cartesian space and the generalized mass and force are approximated with first-order Taylor expansions.

(2) First-order approximate kinematic relationships are obtained between Cartesian variables and state vector variables [14] and are substituted into the variational

equations obtained in (1). Linearized variational equations in state vector space are generated.

(3) First-order Taylor expansions for the kinematical relationships between state vector variables and joint variables are obtained, and then substituted into the approximate variational equations obtained in (2).

(4) Linearized equations of motion are obtained from the approximate variational equations in joint space. At this stage, open-chain mechanisms are expressed in terms of independent coordinates and closed-chain mechanisms are expressed in a mixed differential algebraic equation (DAE) form.

(5) Linearized equations of motion expressed only in terms of independent coordinates are written in state space form for control applications.

The rest of the paper is organized as following. In Section 2, linearized kinematic relationships are expressed in terms of the generalized state vector, which is used to simplify expressions and to obtain compact equations. In Section 3, linearized relative kinematics relations are derived for two contiguous bodies. The linearized equations of motion are developed in Section 4. In Section 5, numerical examples of the recursive linearization method are given. In addition, control designs based on the linearized models and linear control theory are demonstrated. Finally, conclusions are presented in Section 6.

2. Generalized State Vector Notation

In this section, a first-order Taylor expansion is applied to approximate the relationship between Cartesian variables and generalized velocity state variables. The generalized velocity state vector, called the velocity state, is used to simplify expressions in later derivations. It is defined as [13]

$$\hat{Y}_P = \begin{bmatrix} \dot{r}_P + \tilde{r}_P \omega_P \\ \omega_P \end{bmatrix} \qquad (1)$$

where the subscript P represents the origin of a body-fixed frame, as shown in Fig. 2. The Cartesian velocity of point P can be written as

$$Y_P = \begin{bmatrix} \dot{r}_P \\ \omega_P \end{bmatrix} \qquad (2)$$

where $\dot{r}_P$ and ω_P are the translational velocity of point P and the angular velocity of a body-fixed frame at point P, respectively.

From the velocity expressions in Eqs. 1 and 2, the Cartesian velocity Y_P is expressed as

$$Y_P = \begin{bmatrix} I & -\tilde{r}_P \\ 0 & I \end{bmatrix} \hat{Y}_P$$

$$\equiv T_P \hat{Y}_P \qquad (3)$$

Replacing $\dot{r}_P$ by the virtual displacement δr_P and ω_P by the virtual rotation $\delta\pi_P$, yield the variation of the position state vector.

$$\delta Z_P \equiv T_P \, \delta\hat{Z}_P \qquad (4)$$

The Cartesian acceleration of the body-fixed frame shown in Fig. 2 is defined as the time derivative of Eq. 3,

$$\dot{Y}_P = T_P \, \dot{\hat{Y}}_P - V_P \qquad (5)$$

where $V_p \equiv \begin{bmatrix} \dot{\tilde{r}}_P \omega_P \\ 0 \end{bmatrix}$ is a velocity coupling vector.

Since the right sides of Eqs. 1 to 5 are explicitly expressed in terms of Cartesian variables, their first-order expansions can be obtained analytically with respect to perturbations in the Cartesian variables.

First, by expanding T_P at a reference configuration, Eq. 4 can be represented as

$$\delta Z_P = (T_P^o + d T_P^o + O(\Delta)^2) \delta \hat{Z}_P \qquad (6)$$

where d denotes a first-order perturbation with respect to Cartesian variables; the superscript o signifies that the quantity is evaluated at a reference configuration, i.e.,

$$d T_P^o = \frac{\partial T_P}{\partial Z_P} \Big|_{\text{evaluated at o}}^{\Delta}$$

and Δ denotes perturbed quantity, which is expressed in terms of the variables in the Cartesian space. The perturbation of matrix T_p can be obtained as

$$dT_P = \begin{bmatrix} 0 & -d\tilde{r}_P \\ 0 & 0 \end{bmatrix} \qquad (7)$$

where the partial derivative of the position vector of point P can be expressed as

$$dr_P = d\hat{r}_P - \tilde{r}_P \, d\pi_P \qquad (8)$$

Similarly, the acceleration relationship, Eq. 5, can be expanded as

$$\dot{Y}_P = [T_P^o + d T_P^o + O(\Delta)^2] \dot{\hat{Y}}_P - [V_P^o + d V_P^o + O(\Delta)^2] \qquad (9)$$

The derivatives of the position state variable and the Cartesian variable can be related from Eq. 6 as

$$dZ_P = T_P \, d\hat{Z}_P \qquad (10)$$

and the derivative of the Cartesian velocity vector can be written as

$$dY_P = d (T_P \hat{Y}_P)$$

$$= T_P \, d \hat{Y}_P + dT_P \hat{Y}_P \qquad (11)$$

where

$$dT_P \hat{Y}_P = \begin{bmatrix} \tilde{\omega}_P & -\tilde{\omega}_P \tilde{r}_P \\ 0 & 0 \end{bmatrix} d\hat{Z}_P \qquad (12)$$

where $d \hat{Y}_P$ and $d \hat{Z}_P$ are the perturbations of velocity and position state vectors. Based on the relationships in Eqs. 9, 10, and 11, the derivative of the variables in the Cartesian space can be expressed in terms of the derivatives of the state variables.

3. Relative Kinematics of Two Contiguous Bodies

In this section, a first-order Taylor approximation is derived to represent relative kinematic relationships between contiguous bodies that are constrained by a kinematic joint, as shown in Fig. 3. The relative kinematic relation between the velocities of the two contiguous bodies i and j is defined as [13]

$$\hat{Y}_j = \hat{Y}_i + B_{ij} \dot{q}_{ij} \qquad (13)$$

where

$$B_{ij} = \begin{bmatrix} \frac{\partial d_{ij}}{\partial q_{ij}} + (\tilde{r}_j + \tilde{s}_{ji}) H_{ij} \\ H_{ij} \end{bmatrix}$$

Relationships between virtual displacements and rotations of these bodies are obtained by replacing the velocity vectors in Eq. 13 with virtual translational and rotational vectors; i.e.,

$$\delta\hat{Z}_j = \delta\hat{Z}_i + B_{ij}\,\delta q_{ij} \qquad (14)$$

Taking the time derivative of Eq. 13 yields the acceleration relationship

$$\dot{\hat{Y}}_j = \dot{\hat{Y}}_i + B_{ij}\,\ddot{q}_{ij} + D_{ij} \qquad (15)$$

where

$$D_{ij} = \dot{B}_{ij}\,\dot{q}_{ij}$$

As shown in Eqs. 13, 14, and 15, the state variables of body j are expressed in terms of the state variable of body i and the joint variables used to define the relative motion between bodies i and j. The first-order Taylor expansions of Eqs. 14 and 15 with respect to the state variable of body i and the joint variables can be expressed as

$$\delta\hat{Z}_j = \delta\hat{Z}_i + [\,B^0_{ij} + dB^0_{ij} + O(\Delta)^2]\,\delta q_{ij} \qquad (16)$$

$$\dot{\hat{Y}}_j = \dot{\hat{Y}}_i + [\,B^0_{ij} + dB^0_{ij} + O(\Delta)^2]\,\ddot{q}_{ij+}\,[\,D^0_{ij} + dD^0_{ij} + O(\Delta)^2] \qquad (17)$$

where dB_{ij} and dD_{ij} are computed in terms of the state variables of body i and the joint variables.

Linearized equations of motion are generated based on the linearized joint kinematic relationships and linearized relationships between the state space variables of bodies i and j. From Eqs. 13 and 14, relations between the perturbations of state variables and relative coordinates are expressed as

$$d\hat{Z}_j = d\hat{Z}_i + B_{ij}\,dq_{ij} \qquad (18)$$

$$d\hat{Y}_j = d\hat{Y}_i + dB_{ij}\,\dot{q}_{ij} + B_{ij}\,d\dot{q}_{ij} \qquad (19)$$

$$d\dot{\hat{Y}}_j = d\dot{\hat{Y}}_i + dB_{ij}\,\ddot{q}_{ij} + B_{ij}\,d\ddot{q}_{ij} + dD_{ij} \qquad (20)$$

4. Linearized Equations of Motion of a Tree Structural Mechanism

The linearized equations of motion of a tree structure mechanism that contains n joints and n+1 bodies, as shown in Fig. 4, are presented in this section. By going through the procedure of variational derivation [13] and by replacing the nonlinear kinematic relationships with their first-order approximations, one can generate the linearized equations of motion for an open-chain system. Applying the linearized kinematic relationships to the recursive variational approach yields the linearized equations of motion that are written in terms of the joint variables.

4.1 Variational Equations in Cartesian Space

The variational form of the Newton-Euler equations of motion for an n-body system is written as [3]

$$0 = \sum_{i=0}^{n} \delta Z_i^T\,(M_i\dot{Y}_i - Q_i) \qquad (21)$$

where $\mathbf{M}_i$ is the mass matrix and $\mathbf{Q}_i$ is the generalized force. The variational equation must hold for all kinematically admissible variations $\delta\mathbf{Z}_i$, i = 1,...,n; i.e., the kinematic constraints on the system must be satisfied by $\delta\mathbf{Z}_i$. Then approximate variational equations can be generated from a set of approximations of the generalized mass and force for each body, which are expressed as

$$\mathbf{M}_i = \mathbf{M}_i^o + d\mathbf{M}_i^o + O(\Delta)^2 \qquad (22)$$

$$\mathbf{Q}_i = \mathbf{Q}_i^o + d\mathbf{Q}_i^o + O(\Delta)^2 \qquad (23)$$

where $d\mathbf{M}_i$ and $d\mathbf{Q}_i$ are expressed in terms of Cartesian variables.

4.2 Variational Equations in State Vector Space

Approximated variational equations in state vector space can be obtained by substituting Eqs. 6, 10, 22, and 26 into the equations of motion in Cartesian space and by replacing Cartesian variables with the state variables. The approximated variational equations can be written as

$$0 = \sum_{i=0}^{n} \delta\hat{\mathbf{Z}}_i^T \{[\mathbf{T}_i^{oT}\mathbf{M}_i^o\mathbf{T}_i^o\dot{\hat{\mathbf{Y}}}_i - \mathbf{T}_i^{oT}(\mathbf{M}_i^o\mathbf{V}_i^o + \mathbf{Q}_i^o)] + [(d\mathbf{T}_i^{oT}\mathbf{M}_i^o\mathbf{T}_i^o + \mathbf{T}_i^{oT}d\mathbf{M}_i^o\mathbf{T}_i^o +$$

$$\mathbf{T}_i^{oT}\mathbf{M}_i^o d\mathbf{T}_i^o\dot{\hat{\mathbf{Y}}}_i - (d\mathbf{T}_i^{oT}\mathbf{M}_i^o\mathbf{V}_i^o + \mathbf{T}_i^{oT}d\mathbf{M}_i^o\mathbf{V}_i^o + \mathbf{T}_i^{oT}\mathbf{M}_i^o\mathbf{V}_i^o + d\mathbf{T}_i^{oT}\mathbf{Q}_i^o + \mathbf{T}_i^{oT}d\mathbf{Q}_i^o)] + O(\Delta)^2\} \qquad (24)$$

where $\mathbf{V}_i$ is a velocity coupling term, which is defined in Eq. 5. In order to simplify Eq. 24, the notation of the generalized mass matrix $\hat{\mathbf{M}}_i$ and force vector $\hat{\mathbf{Q}}_i$ in the state vector space [13] will be used henceforth. The equations of motion are thus expressed as

$$0 = \sum_{i=0}^{n} \delta\hat{\mathbf{Z}}_i^T \{(\hat{\mathbf{M}}_i^o\dot{\hat{\mathbf{Y}}}_i - \hat{\mathbf{Q}}_i^o) + d\hat{\mathbf{M}}_i^o\dot{\hat{\mathbf{Y}}}_i - d\hat{\mathbf{Q}}_i^o + O(\Delta)^2\} \qquad (25)$$

where

$$d\hat{\mathbf{M}}_i = d\mathbf{T}_i^T\mathbf{M}_i\mathbf{T}_i + \mathbf{T}_i^T d\mathbf{M}_i\mathbf{T}_i + \mathbf{T}_i^T\mathbf{M}_i d\mathbf{T}_i$$

$$d\hat{\mathbf{Q}}_i = d\mathbf{T}_i^T\mathbf{M}_i\mathbf{V}_i + \mathbf{T}_i^T d\mathbf{M}_i\mathbf{V}_i + \mathbf{T}_i^T\mathbf{M}_i d\mathbf{V}_i + d\mathbf{T}_i^T\mathbf{Q}_i + \mathbf{T}_i^T d\mathbf{Q}_i$$

and $d\mathbf{M}_i$, $d\mathbf{T}_i$, $d\mathbf{V}_i$ and $d\mathbf{Q}_i$, which are expressed in terms of the perturbations of the Cartesian variables, can be rewritten in terms of the perturbations of the state variables by substituting the relationship between the Cartesian variables and the state variables into their expressions.

4.3. Linearized Equations of Motion in Joint Space

The approximated variational equations of motion in state vector space can be rewritten in terms of joint variables. As the results in Section 3 indicate, the variables in state vector space can be transformed into joint variables. The linearized equations in joint space can be obtained by applying the following procedures. Substituting the approximated kinematic relationship between bodies n and n-1 into the variational equations yields

$$0 = \sum_{i=0}^{n-1} \delta\hat{\mathbf{Z}}_i^T \{(\hat{\mathbf{M}}_i^o\dot{\hat{\mathbf{Y}}}_i - \hat{\mathbf{Q}}_i^o) + d\hat{\mathbf{M}}_i^o\dot{\hat{\mathbf{Y}}}_i - d\hat{\mathbf{Q}}_i^o + O(\Delta)^2\} +$$

$$\delta\hat{\mathbf{Z}}_{n-1}^T \{(\hat{\mathbf{M}}_n^o\dot{\hat{\mathbf{Y}}}_n - \hat{\mathbf{Q}}_n^o) + d\hat{\mathbf{M}}_n^o\dot{\hat{\mathbf{Y}}}_n - d\hat{\mathbf{Q}}_n^o + O(\Delta)^2\} + \qquad (26)$$

$$\delta q_n^T \{ (B_n^{T^o} + dB_n^{T^o} + O(\Delta)^2) [(\hat{M}_n^o \dot{\hat{Y}}_n - \hat{Q}_n^o) + d\hat{M}_n^o \dot{\hat{Y}}_n - d\hat{Q}_n^o + O(\Delta)^2]\}$$

where the perturbations are taken with respect to the state variables of the inboard bodies and the relative variables that are used to defined the relative motion between the bodies n and n-1. Moreover, the relative kinematic matrix $B_{n-1\ n}$ is denoted as B_n to simplify the expressions during the derivation.

Because joint n is not subject to any relative constraint between the connected bodies, the virtual displacement of the joint coordinate is arbitrary. Thus the coefficient of δq_n^T is equal to zero; i.e.,

$$0 = (B_n^{T^o} + dB_n^{T^o} + O(\Delta)^2) [(\hat{M}_n^o \dot{\hat{Y}}_n - \hat{Q}_n^o) + d\hat{M}_n^o \dot{\hat{Y}}_n - d\hat{Q}_n^o + O(\Delta)^2] \quad (27)$$

Substituting the first-order Taylor expansion of the acceleration vector into state vector space, and substituting the relationships between state and joint spaces into Eq. 27, gives the equation of motion corresponding to joint n.

$$0 = B_n^{T^o} (\hat{M}_n^o \dot{\hat{Y}}_n^o - \hat{Q}_n^o) + d (B_n^{T^o} \hat{M}_n^o \dot{\hat{Y}}_n^o) - d(B_n^{T^o} \hat{Q}_n^o) + O(\Delta)^2 \quad (28)$$

where

$$d (B_n^T \hat{M}_n \dot{\hat{Y}}_n) = dB_n^T \hat{M}_n \dot{\hat{Y}}_n + B_n^T d\hat{M}_n \dot{\hat{Y}}_n + B_n^T \hat{M}_n d\dot{\hat{Y}}_n$$

$$d(B_n^T \hat{Q}_n) = dB_n^T \hat{Q}_n + B_n^T d\hat{Q}_n$$

and $d\hat{M}_n$ and $d\hat{Q}_n$ can be written in terms of the state variables of body n-1 and the relative variables that are used to define the relative motion between bodies n and n-1. Moreover, the equation of motion corresponding to joint n at the reference configuration is

$$0 = B_n^{T^o} (\hat{M}_n^o \dot{\hat{Y}}_n^o - \hat{Q}_n^o) \quad (29)$$

Substituting the relationship in Eq. 29 into Eq. 28 and omitting higher order terms yields

$$0 = d (B_n^{T^o} \hat{M}_n^o \dot{\hat{Y}}_n^o) - d(B_n^{T^o} \hat{Q}_n^o) \quad (30)$$

where $\dot{\hat{Y}}_n$ can be expressed in terms of the derivative of a set of independent variables

x (= [$\dot{\hat{Y}}_0^T$ $\hat{Y}_0^T$ $\hat{Z}_0^T$ $\ddot{q}_1^T$ $\dot{q}_1^T$ q_1^T ... $\ddot{q}_n^T$ $\dot{q}_n^T$ q_n^T]^T) by substituting the relationships from Eqs. 18, 19, and 20 recursively for j from n to 1.

Following the same arguments used in obtaining Eq. 30, one can obtain the linearized equations of motion corresponding to joint i as

$$0 = B_i^{T^o} d(K_i \dot{\hat{Y}}_i + K_{i+1} B_{i+1} \ddot{q}_{i+1} + ... + K_n B_n \ddot{q}_n - L_i)^o +$$

$$dB_i^{T^o} (K_i \dot{\hat{Y}}_i + K_{i+1} B_{i+1} \ddot{q}_{i+1} + ... + K_n B_n \ddot{q}_n - L_i)^o$$

$$\text{for } i = 1, ..., n \quad (31)$$

where $K_i = \hat{M}_i + K_{i+1}$, $L_i = \hat{Q}_i + L_{i+1} - K_{i+1} D_{i+1}$, $K_n = \hat{M}_n$, and $L_n = \hat{Q}_n$.

By repeating the above procedure in backward path sequence to the base body, one can obtain the linearized equation of motion for the base body as

$$d(K_0 \dot{\hat{Y}}_0 + K_1 B_1 \ddot{q}_1 + ... + K_n B_n \ddot{q}_n - L_0)^o = 0 \quad (32)$$

For the open-chain mechanism, the linearized equations of motion at the reference configuration, which are represented in Eqs. 31 and 32, can be obtained recursively. During the derivation, every perturbed term in the linearized equations can be computed either from the perturbed variables in the Cartesian variables, which are computed analytically, or from the analytically linearized relationships among the Cartesian, state vector, and joint coordinate spaces.

5. Numerical Examples

In this section, the applications of the recursive linearization algorithm are illustrated by two examples: a two-link manipulator and a robot arm with seven degrees of freedom. The accuracy and computational efficiency of the proposed algorithm are demonstrated by comparing the models obtained from the recursive algorithm with those obtained from the analytical approach and from a numerical perturbation method. In the case of the two-link system, an exact linearization is accomplished by the use of the symbolic manipulator (MACSYMA) [15]. However, to generate the exact linearized model for a complicated system is very difficult, even with a symbolic manipulator. In the second example, a numerical perturbation is applied to the robot with seven degrees of freedom in order to generate a reference linearized model with which the results of the recursive linearization algorithm are compared.

5.1 A Two-Link Manipulator

In this subsection, a two-link manipulator, as shown in Fig. 7, is modeled and tested. Since all the joints are revolute, one independent coordinate is assigned to each joint. The manipulator can be modeled as a system of two differential equations. For this system, the linearization can also be carried out analytically by using the symbolic manipulator (MACSYMA). Therefore, it is possible to check the accuracy of the recursive linearization algorithm by comparing the linear models obtained from both approaches: recursive linearization and MACSYMA implementation.

The recursive linearization produced a linearized model at the specified configuration that was defined by setting θ_1, θ_2, $\dot{\theta}_1$, and $\dot{\theta}_2$ to zero. The linearized equation of motion is written as

$$\begin{bmatrix} d\dot{\theta}_2 \\ d\dot{\theta}_1 \\ d\ddot{\theta}_2 \\ d\ddot{\theta}_1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -5.191 & 3.4612 & 0 & 0 \\ 1.1537 & -4.0380 & 0 & 0 \end{bmatrix} \begin{bmatrix} d\theta_2 \\ d\theta_1 \\ d\dot{\theta}_2 \\ d\dot{\theta}_1 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -0.0824 & 0.0294 \\ 0.0294 & -0.0176 \end{bmatrix} \begin{bmatrix} dT_2 \\ dT_1 \end{bmatrix} \tag{33}$$

where T_1 and T_2 are actuating torques that are applied at the revolute joints. At the same specified configuration, the symbolic manipulation generated the exact linearized model, which is identical to the one obtained from the recursive linearization approach. A comparison of the numerical and analytical results shows that the proposed recursive linearization algorithm can generate a correct linearized model at a given configuration.

After the linearized model is obtained, a linear controller can be designed by applying the linear model to existing control design tools. A linear regulator is designed to control the motion of the manipulator by using the Pro-Matlab package. The pole placement algorithm [16] is used to compute the full state feedback gain matrix for the nonlinear dynamic model. The effectiveness of this regulator is tested by applying an initial deviation of the system and using the regulator to stabilize it. As expected, the linearized model can well represent the nonlinear model. Therefore, a small initial deviation is tested first. The results of 0.05 radian initial deviations are shown in Figs. 6 and 7. The nonlinear system can be stabilized by

the linear regulator. Similar results are presented in Figs. 8 and 9 for 1.0 radian initial deviations.

However, the pole placement algorithm for a multiple-input multiple-output system does not have a unique solution [17]. The feedback gain obtained from the Pro-Matlab package is an iterating solution, which is designed to find an insensitive set for the configuration change. However, this algorithm requires a lot of computation to generate an optimal gain matrix. Thus, this algorithm cannot be used for an on-line computation for the real time simulation. To fulfill the on-line computation requirement, a simple and stable pole placement algorithm is needed. A case particularly interesting is to determine the feedback controller [17] in such a way that the closed loop equation is decomposed into a set of n decoupled second-order differential equations.

$$0 = d\ddot{\theta}_i + 2\xi_i \omega_i d\dot{\theta}_i + \omega_i^2 d\theta_i ; \qquad i = 1, ..., n \tag{34}$$

where the damping factor ξ_i and the undamped frequency ω_i of each tracking error are specified by the designer. Defining the nxn constant diagonal matrices $\Lambda_1 = \text{diag}_i\{2\xi_i \omega_i\}$ and $\Lambda_2 = \text{diag}_i\{\omega_i^2\}$, one can obtain the desired decoupled closed loop equation from Eq. 34 as

$$\mathbf{0} = d\ddot{\theta} + \Lambda_1 d\dot{\theta} + \Lambda_2 d\theta \tag{35}$$

The closed loop equation of the linearized model with a proportional-derivative (PD) controller can be written in a second order differential equation form as

$$\mathbf{0} = \mathbf{M}\, d\ddot{\theta} + (\mathbf{P}_1 - \mathbf{K}_V) d\dot{\theta} + (\mathbf{P}_2 - \mathbf{K}_P)\, d\theta \tag{36}$$

where $\mathbf{K}_P$ is the position feedback gain matrix and $\mathbf{K}_V$ is the velocity feedback gain matrix. Equating coefficients in Eqs. 35 and 36 gives the desired closed-loop feedback gain matrices as

$$\mathbf{K}_V = \mathbf{M}\Lambda_1 - \mathbf{P}_1 \tag{37}$$

$$\mathbf{K}_P = \mathbf{M}\Lambda_2 - \mathbf{P}_2 \tag{38}$$

Consequently, a linear regulator is designed to control the dynamic system. As shown in Figs. 10, 11, 12, and 13, the linear regulator can stabilize the nonlinear dynamic model for both small and large initial deviation cases.

5.2 A Robot with Seven Degrees of Freedom

Figure 10 shows a robot arm that has seven degrees of freedom. The system consists of eight bodies, including the base body, which is designated as ground. The adjacent bodies are connected by revolute joints. Joints 1 to 7 are identified as Shoulder Roll, Shoulder Pitch, Elbow Roll, Elbow Pitch, Wrist Roll, Wrist Pitch, and Toolplate Roll.

Since adjacent bodies are connected by revolute joints, one generalized coordinate is assigned to each joint. The motion of this system can be described by seven generalized coordinates; the dynamic system is thus formulated as a system of seven differential equations. When a reference configuration is selected, a linearized model can be generated at this configuration using the proposed linearization algorithm.

The configuration that is shown in Fig. 10 is selected as a reference configuration: the angles of all the joints are zero ($q_1, q_2, ..., q_7 = 0$), and the velocities of all the joints are also zero ($\dot{q}_1, ..., \dot{q}_7 = 0$). At this reference configuration, the linearized model that is obtained from the recursive linearization algorithm is expressed as

$$d\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{M}^{-1}\mathbf{P}_2 & \mathbf{M}^{-1}\mathbf{P}_1 \end{bmatrix} d\mathbf{x} + \begin{bmatrix} \mathbf{0} \\ \mathbf{M}^{-1}\mathbf{P}_3 \end{bmatrix} d\mathbf{u} \tag{39}$$

where $\mathbf{x} = [q_1\ q_2\ q_3\ q_4\ q_5\ q_6\ q_7\ \dot{q}_1\ \dot{q}_2\ \dot{q}_3\ \dot{q}_4\ \dot{q}_5\ \dot{q}_6\ \dot{q}_7]^T$, $\mathbf{u}$ is the actuating torque vector, $\mathbf{M}$ is the generalized mass matrix that is expressed in terms of joint variables, and

$$M^{-1}P_1 = 0_{7x7}$$

$$M^{-1}P_2 = \begin{bmatrix} 15.737 & 0 & -1.607 & 0 & -.1451 & 0 & -.18e{-}8 \\ 0 & 15.228 & 0 & 2.8549 & 0 & -.3485 & 0 \\ .41697 & 0 & 9.2375 & 0 & -.4480 & 0 & -.13e{-}7 \\ 0 & -11.44 & 0 & -6.361 & 0 & .5637 & 0 \\ -29.11 & 0 & 3.711 & 0 & 21.533 & 0 & .93e{-}7 \\ 0 & -5.0345 & 0 & -13.19 & 0 & -15.76 & 0 \\ 14.914 & 0 & 5.9428 & 0 & -25.38 & 0 & -.58e{-}7 \end{bmatrix}$$

$$P_3 = I$$

In this case, to generate a closed-form analytical expression for the linearized model is too difficult for accuracy checking even if the symbolic manipulation is employed. Instead, two comparisons were derived to make certain that the linearized model in Eq. 39 accurately represents the nonlinear model. In the first comparison, both the nonlinear and linearized models were perturbed with the same amount, and then the resulted acceleration changes were examined. In the second comparison, two linearized models–one obtained from the recursive approach and the other obtained from the numerical perturbation method–are examined.

In the first comparison, perturbation of the generalized coordinate $\mathbf{x}$ by 10^{-6} incurs the relative error of the acceleration changes between the linearized and nonlinear model as

$$\frac{\| d\dot{x}^* - d\dot{x} \|_2}{\| d\dot{x}^* \|_2} = 2.739\ e{-}6 \qquad (40)$$

where $d\dot{x}$ is the acceleration change obtained from the linearized model and $d\dot{x}^*$ is obtained from the nonlinear model.

In the second comparison, a simple numerical perturbation without any convergence checking is implemented to generate a linearized model, which will serve as a reference in comparing the recursive linearization with the numerical perturbation. Comparing the linearized model obtained from the numerical method with those obtained from the recursive approach, one can observe that at the given configuration both approaches generate nearly identical linearized models, in which the relative difference is less than 10^{-6}.

However, the recursive algorithm proves to be more efficient than the numerical perturbation method. In comparison with the numerical perturbation method, the recursive linearization took half the cpu time to generate a linear model, even though the numerical perturbation method used here was a relatively simple one. If a convergence checking algorithm was employed for the numerical perturbation method, it would take even longer to generate a linear model.

The simple pole placement used in the previous example was used again to design a linear regulator. The desired closed-loop poles were selected to make the simulation results similar to the experimental results. After properly selecting the desired poles, we used this linear regulator to control the nonlinear dynamic model. The step response of Joint 4 is shown in Fig. 11. From this result, it is clear that a simple regulator based on a linearized model simulates the behavior of a complicate control system around a reference configuration.

6. Conclusion

In these examples, we have shown that the proposed linearization algorithm is both efficient and accurate in generating a linear model at given configurations. These linear

models are converted to the standard state space forms, which are convenient for linear control design. Moreover, the driving force input for a required motion around a given configuration can be predicted by using the linearized model. When a large gross motion is involved in a prescribed trajectory, more than one linearized model may be necessary for robust control. In such a case, the computation of linearization must be fast enough to update the linearized model before it fails to represent the system adequately. With the emerging parallel processing computers and computation algorithm, the use of successive linearization will be possible for on-line adaptive control.

References

1. Bodley, C., Devers, A., Park, A., and Frisch, H., A digital Computer Program for Dynamic Iteration and Simulation of Controls and Structures (DISCO), NASA Tech. Paper 1219, May 1978.
2. Liang, C.G., and Lance, G.M., Dynamic Analysis and Control Synthesis of Integrated Mechanical Systems, Ph.D. Thesis, University of Iowa, 1985.
3. Haug, E.J., Computer Aided Kinematics and Dynamics of Mechanical Systems, Volume I: Basic Method, Allyn & Bacon, Boston,1989.
4. Sohoni, V.N., and Whitesell, J., "Automatic Linearization of Constrained Dynamical Models," Journal of Mechanism, Transmission, and Automation in Design, Transactions of the ASME, vol. 108, Sept., 1986, p. 300-304.
5. Singh, R.P., Likins, P.W., and VenderVoortt, R.J., "Automated Dynamics and Control Analysis of Constrained Multibody System," Robotic & Manufacturing Automation, 1985, p. 109-113.
6. Vukobratovic, M. and Nenad, K., "Computer-oriented Method for Linearization of Dynamic Models of Active Spatial Mechanisms," Mechanism and Machine Theory, Vol. 17, No. 1, 1982, p.21-32.
7. Neuman, C., "Linearization and Sensitivity Functions of Dynamic Robot Models," IEEE Transactions on Systems, Man, and Cybernetics, vol. 14, no. 6, 1984, p.805-818.
8. Balafoutis, C.A., Misra, P., and Patel, R.V., "Recursive Evaluation of Linearized Dynamic Robot Models," IEEE J. of Robotics and Automation, Vol. RA-2, No. 3, 1986, p. 146-155.
9. Jonker, J.B., A Computer-Oriented Method for Linearization of The Dynamic Mechanism Equations, Lab. Report no. 822, Department of Mechanical Engineering, Delft University of Technology, The Netherlands, 1986.
10. Bad, D.S. and Haug, E.J., "A recursive Formulation for Constrained Mechanical Systems, Part I- Open Loop," Mechanics of Structures and Machines, Vol. 15, No. 4, 1987.
11. Bad, D.S. and Haug, E.J., "A recursive Formulation for Constrained Mechanical Systems, Part II- Closed Loop," Mechanics of Structures and Machines, Vol. 15, No. 4, 1987.
12. Bad, D.S., Haug, E.J., and Kuhl, J.G., "A recursive Formulation for Constrained Mechanical Systems, Part III- Parallel Processor Implementation," Mechanics of Structures and Machines, Vol. 16, No. 2, 1988.
13. Bae, D.S., Hwang, R.S., and Haug, E.J., "A Recursive Formulation for Real-Time Dynamic Simulation," Submitted to Journal of Mechanisms, Transmissions, and Automation in Design.
14. Hwang, R.S., Bae, D.S., Kuhl, J.G., and Haug, E.J., "Parallel Processing for Real-Time Dynamic Simulation," Submitted to Journal of Mechanisms, Transmissions, and Automation in Design.
15. VAX UNIX Macsyma Reference Manual, Version 11, Symbolics, Inc. 1985.
16. Kautsky, J., Nichols, N.K., and Van Dooren, P., "Robust Pole Assignment in Linear State Feedback," International Journal of Control, Vol. 41, No. 5, p. 1129-1155, 1985.

17. Brady, Hollerbach, Johnson, Lozano-P'eroz, and Mason, editor, Robot Motion: Planning and Control, The MIT Press, Cambridge, 1983.

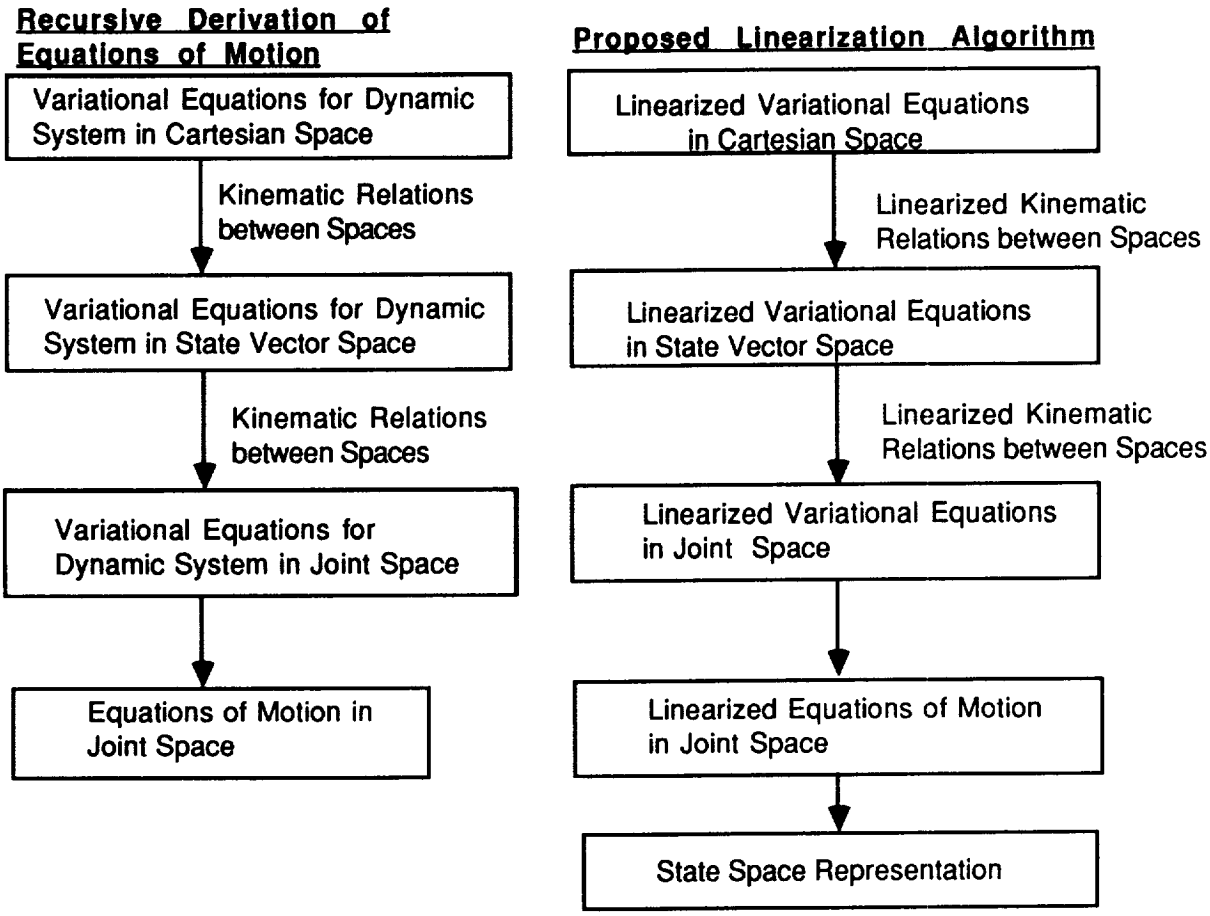


Figure 1. Recursive Derivation Flow for Dynamic Equations of Motion and Proposed Linearization Algorithm

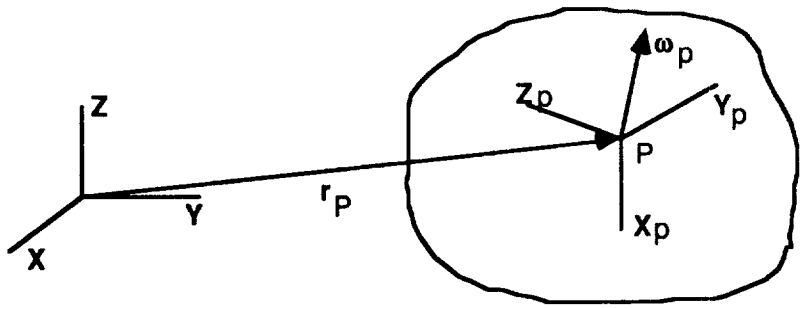


Figure 2. Body and Global Frames Representations

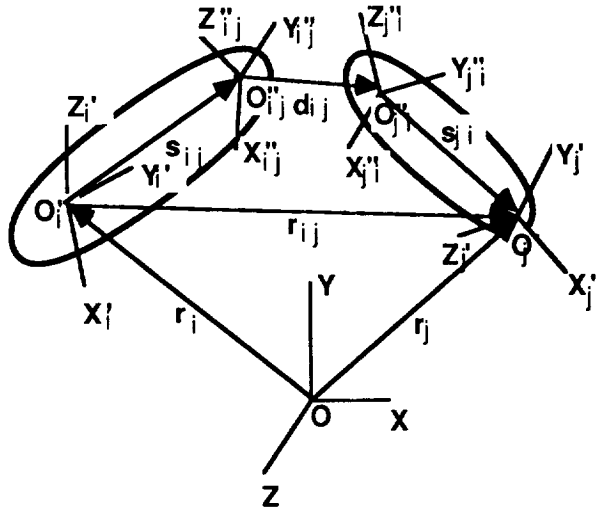


Figure 3. Pair of Contiguous Bodies

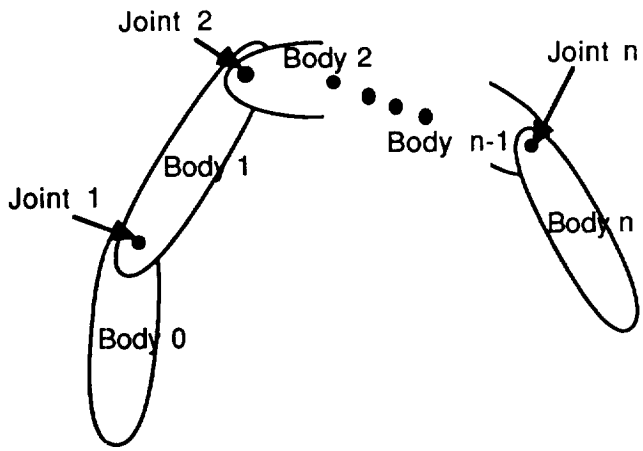


Figure 4. An n-Joint Open -Chain Mechanism

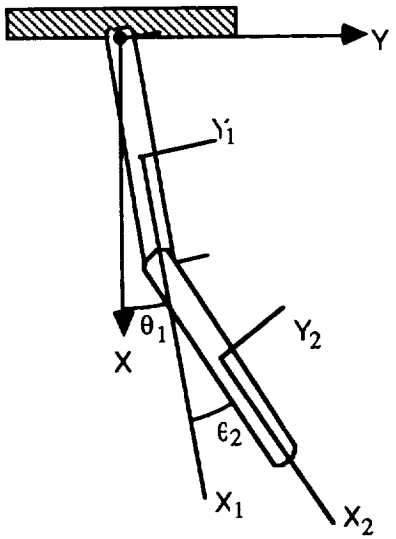


Figure 5. A Two Links Manipulator Arm

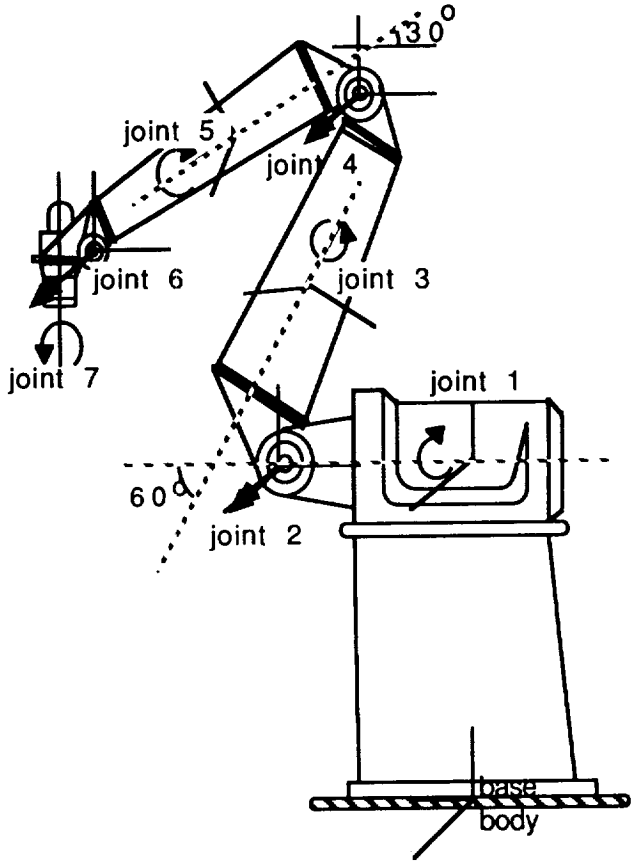


Figure 10. A Seven Degrees of Freedom Robot

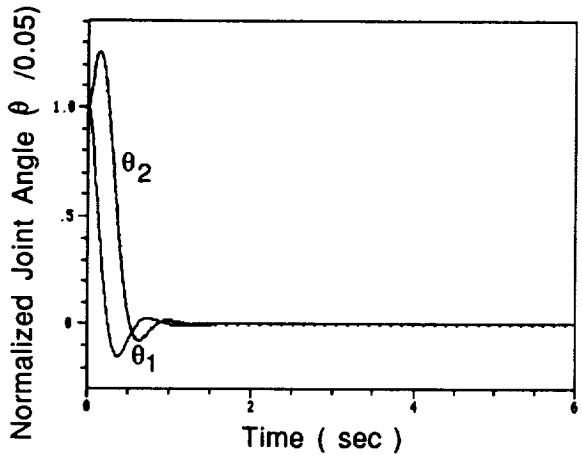


Figure 6. Response of small Initial Deviation

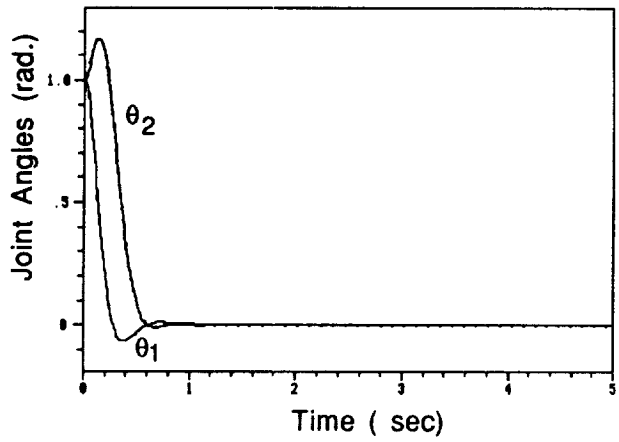


Figure 7. Response of large Initial Deviation

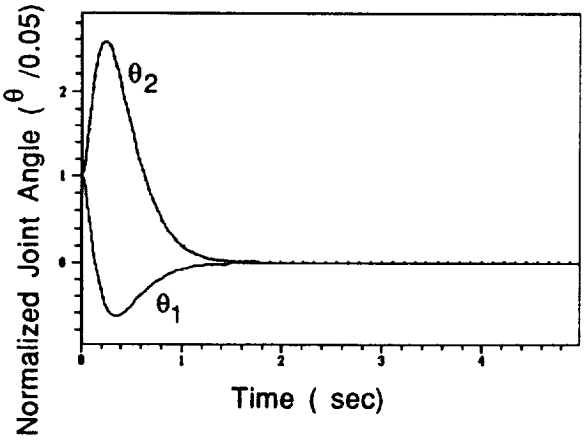


Figure 8. Response of small Initial Deviation

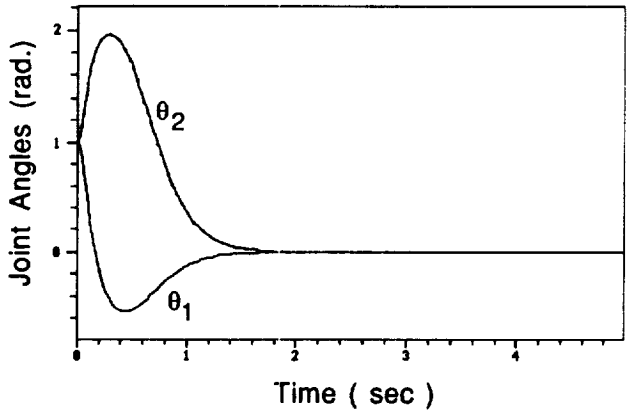


Figure 9. Response of large Initial Deviation

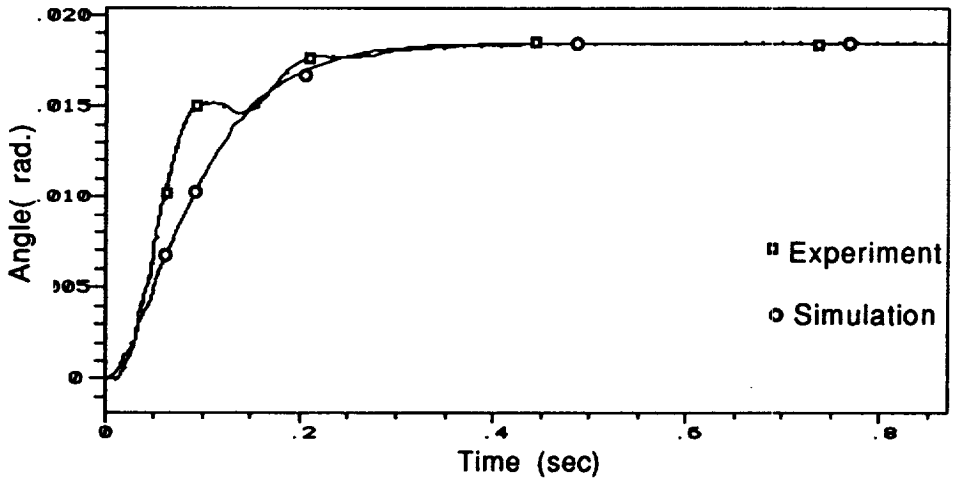


Figure 11. Step Response of Joint 4

An Innovations Approach to Decoupling of Multibody Dynamics and Control

G. Rodriguez
Jet Propulsion Laboratory
California Institute of Technology

Abstract

The paper solves the problem of hinged multibody dynamics using an extension of the innovations approach of linear filtering and prediction theory to the problem of mechanical system modeling and control. This approach has been used quite effectively to diagonalize the equations for filtering and prediction for linear state space systems. It has similar advantages in the study of dynamics and control of multibody systems. The innovations approach advanced here consists of expressing the equations of motion in terms of two closely related processes: (1) the innovations process e, a sequence of moments, obtained from the applied moments T by means of a spatially recursive Kalman filter that goes from the tip of the manipulator to its base; (2) a residual process, a sequence of velocities, obtained from the joint-angle velocities by means of an outward smoothing operations. The innovations e and the applied moments T are related by means of the relationships $e = (I - L)T$ and $T = (I + K)e$. The operation $(I - L)$ is a causal lower triangular matrix which is generated by a spatially recursive Kalman filter and the corresponding discrete-step Riccati equation. Hence, the innovations and the applied moments can be obtained from each other by means of a causal operation which is itself causally invertible. The residuals n and the joint-angle velocities $\dot{q}$ are related by $n = (I - K^*)\dot{q}$ and $\dot{q} = (I - L^*)n$ in which $(I - L^*)$ is also an anticausal, upper-triangular, matrix. Hence, the residuals and the joint-angle velocities are related by means of an anticausal operation which is itself anticausally invertible. The use of the residuals process is of interest because it diagonalizes the composite multibody system kinetic energy. In other words, the kinetic energy $J(\dot{q}, q)$ in the system can be written as $J = 1/2 n^T Dn$ in which D is a diagonal matrix. The Lagrangian equations of motion that result from this diagonal form for the kinetic energy are completely decoupled in the sense that the equation for the residual velocity at any given joint is independent from the similar equations at all of the remaining joints. The innovations process appears as a driving term in these equations. Use of the innovations, in place of the physically applied joint moments, decouples the equations even further. The equations of motion for joint k involves only the value of the innovations at the same joint. The final equations of motion are therefore diagonalized in the sense that the equation for any given joint is independent from the equations at the other

joints. The diagonal form of the equations of motion results in significant simplification of dynamic analysis, simulation, stability analysis, and control design. This simplicity is illustrated by arriving a very simple decoupled control algorithms for robotic manipulator control.

EFFICIENT DYNAMIC SIMULATION FOR MULTIPLE CHAIN ROBOTIC MECHANISMS

Kathryn W. Lilly and David E. Orin

Department of Electrical Engineering
The Ohio State University
Columbus, OH 43210

Abstract

An efficient $O(mN)$ algorithm for dynamic simulation of simple closed-chain robotic mechanisms will be presented in this paper, where m is the number of chains, and N is the number of degrees of freedom for each chain. It is based on computation of the operational space inertia matrix (6×6) for each chain as seen by the body, load, or object. Also, computation of the chain dynamics, when opened at one end, is required, and the most efficient algorithm is used for this purpose. Parallel implementation of the dynamics for each chain results in an $O(N) + O(\log_2 m+1)$ algorithm.

I. Introduction

Recently, there has been an increasing interest in robotic systems with multiple chains forming simple closed kinematic loops. Such systems of interest in space robotics applications include multilegged vehicles, multiple manipulators, and dexterous hands. Each is characterized by multiple chains of links (legs, arms, or fingers) in support of a body, load, or object. Real-time simulation of these systems is important for remote operation, but difficult to achieve at present. An even greater challenge to the computational engineer is that of *super-real-time simulation*, that is, planning seconds of motion in milliseconds. This has been shown to be of value in the control of a multilegged vehicle when predicting the action of the present control to ensure safety and stability along a planned trajectory.

The fundamental goal of this paper is the development of an efficient algorithm for the dynamic simulation of the time-varying topological systems discussed above. Previous researchers have presented algorithms for these and similar configurations based on equation augmentation [1], constraint propagation [2], and recursive computation [3,4], but these methods are often difficult to apply and/or computationally inefficient. The new simulation algorithm derived here makes use of efficient computations for the individual supporting chains to produce an efficient simulation method for the complete robot system. The dynamic properties of each chain are described in a simple, physically understandable manner, which facilitates the straightforward analysis of the combined dynamics of the entire mechanism.

Multiple chain robotic systems can take many forms, some of them quite complex. Simple closed-chain mechanisms are a subset of multiple chain systems with specific structural characteristics. The structure of a simple closed-chain mechanism is characterized by m actuated chains which support a single common reference member [1]. A supporting chain is identified as an independent functional unit in the closed chain system which has two ends, each terminated by a single link. Each chain may have an arbitrary number of links and degrees of freedom, and closed kinematic loops within a chain are permitted. The removal of the reference member breaks the closed loops formed by the multiple chains.

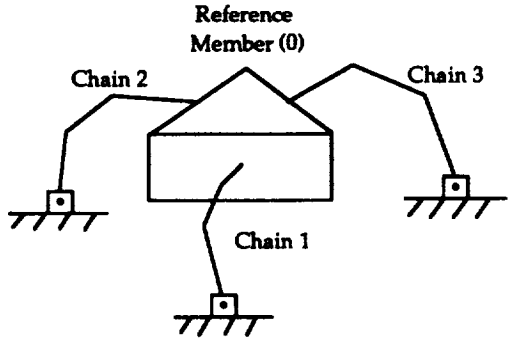


Figure 1: Example of a Type 0 Simple Closed-Chain Mechanism

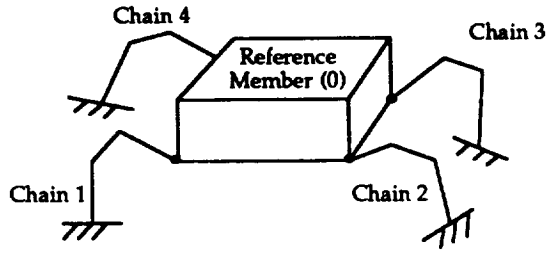


Figure 2: Example of a Type 1 Simple Closed-Chain Mechanism

There are two basic types of simple closed-chain mechanisms called Type 0 and Type 1, respectively [1]. These two types are defined based on the nature of the interactions which occur between the links of each chain and the reference member or support surface. Figure 1 illustrates a typical Type 0 mechanism which may be used to model multiple manipulators or dexterous hands. Note that the support surface, shown here as a fixed inertial frame for a multiple manipulator configuration, might also represent the moving "palm" of a dexterous hand. In either case, for a Type 0 mechanism, the base link of each chain is connected to the support surface by an actuated joint structure, while the last link interacts with the reference member through an unpowered contact. Figure 2 illustrates a Type 1 simple closed-chain mechanism which may be used to model multilegged vehicles. For a Type 1 mechanism, the last link of each chain interacts with the support surface through an unpowered contact, while the base link is connected to the reference member by an actuated joint structure. For both Type 0 and Type 1 mechanisms, the reference member (object, load, or body) is numbered 0, while the chains are numbered arbitrarily from 1 to m. Chain k ($k = 1, \ldots, m$) has N_k degrees of freedom, where N_k may be less than, equal to, or greater than 6.

In order to apply the same algorithm to both types in this work, the support surface will be considered to act as the "base" of each chain. We will refer to the terminal link which interacts with the support surface as link 1, and the terminal link which interacts with the reference member will be called the last link or end effector (link N). The far end of link N is the "tip" of the chain. The interactions and connections which occur between bodies or links in the system (including those at the support surface and at the reference member) will be described using the general joint model of [5,6]. This

includes both powered joint structures and unpowered contacts. The motion of the support surface is assumed to be known.

In this paper, an $O(mN)$ recursive algorithm for the dynamic simulation of simple closed-chain mechanisms is derived for m chains with N degrees of freedom each. The algorithm is based on the efficient computation of the (6×6) operational space inertia matrix [7] for each chain as seen by the body, load, or object. The operational space inertia matrix, $\mathbf{\Lambda}$, may be used to obtain the net effect of the chain dynamics at its tip. The computation of the chain dynamics when the chain is open at one end is also required, and the most efficient algorithm is used for this purpose. Given $O(N)$ algorithms for these two fundamental computations for each chain [4,5,8,9], an $O(mN)$ algorithm for the simulation of the entire multiple chain system is formulated.

In the next section, the notational and modelling conventions used in the formulation of the new algorithm are summarized. In the third section, the dynamic properties of the individual supporting chains and the common reference member are discussed, and the appropriate dynamic equations are developed. The operational space inertia matrix and the open-chain dynamics of each chain are of special significance in this discussion. In the fourth section, the $O(mN)$ dynamic simulation algorithm for simple closed-chain mechanisms is derived. The final algorithm is presented as a series of five steps, which are summarized in a convenient tabular form. The computational requirements of the new algorithm, including parallel implementation considerations, are presented in the fifth section. Finally, the results of this work are summarized and some overall conclusions are given in the final section.

II. Notation

Many of the notational conventions used in this paper are based on concepts introduced by Roberson and Schwertassek in [6] and used by Brandl, Johanni, and Otter in [5]. They are similar in many ways to those described by Featherstone in [9] for robot dynamics, although there are a few minor differences. As in each of these, spatial notation will be used to develop the dynamic equations for the chains and reference member. With spatial notation, velocity, acceleration, and force vectors are all 6×1 column vectors, where each incorporates the appropriate linear and angular components. In this paper, the spatial velocity of the reference member, $\mathbf{v}_0$, is written:

$$\mathbf{v}_0 = [\ (\omega_0)_x \quad (\omega_0)_y \quad (\omega_0)_z \quad (v_0)_x \quad (v_0)_y \quad (v_0)_z\ ]^T, \tag{1}$$

where $(\omega_0)_x$, $(\omega_0)_y$, and $(\omega_0)_z$ are the components of the angular velocity of the reference member about $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$, and $\hat{\mathbf{z}}$, respectively, usually resolved in the reference member frame (frame 0) or an inertial coordinate system. The three components, $(v_0)_x$, $(v_0)_y$, and $(v_0)_z$, represent the linear velocity of the coordinate origin of frame 0. Similarly, the spatial acceleration of the reference member is expressed as:

$$\mathbf{a}_0 = [\ (\alpha_0)_x \quad (\alpha_0)_y \quad (\alpha_0)_z \quad (a_0)_x \quad (a_0)_y \quad (a_0)_z\ ]^T, \tag{2}$$

where the individual components now correspond to resolved angular and linear acceleration vectors. Spatial force vectors have a corresponding structure:

$$\mathbf{f}_k = [\ (n_k)_x \quad (n_k)_y \quad (n_k)_z \quad (f_k)_x \quad (f_k)_y \quad (f_k)_z\ ]^T, \tag{3}$$

where, in this case, $\mathbf{f}_k$ will be used to represent the spatial force exerted on the reference member by chain k. The first three components, $(n_k)_x$, $(n_k)_y$, and $(n_k)_z$, represent the elements of a three-dimensional moment vector, while $(f_k)_x$, $(f_k)_y$, and $(f_k)_z$ are the elements of a three-dimensional force vector.

In general, the transformation of a spatial velocity or acceleration vector from one coordinate system to another one may be accomplished by the following spatial multiplication [9]:

$$^{j}\mathbf{p} = {}^{j}\mathbf{X}_{i}\,{}^{i}\mathbf{p}, \tag{4}$$

where ${}^{i}\mathbf{p}$ is the vector expressed with respect to the ith coordinate system, ${}^{j}\mathbf{p}$ is the same vector expressed with respect to the jth coordinate system, and ${}^{j}\mathbf{X}_{i}$ is the 6×6 spatial transformation matrix. This spatial transformation is defined as follows:

$${}^{j}\mathbf{X}_{i} = \begin{bmatrix} {}^{j}\mathbf{A}_{i} & \mathbf{0} \\ {}^{j}\mathbf{A}_{i}\,\tilde{\mathbf{b}}_{j}^{T} & {}^{j}\mathbf{A}_{i} \end{bmatrix}, \tag{5}$$

where ${}^{j}\mathbf{A}_{i}$ is the 3×3 rotation transformation between the two coordinate systems, and $\mathbf{b}_{j}$ is the 3×1 position vector from the origin of frame i to the origin of frame j, with components expressed in frame i. The 3×3 matrix, $\tilde{\mathbf{b}}_{j}$, is an anti-symmetric matrix defined by the rule:

$$\tilde{\mathbf{c}} = \begin{bmatrix} 0 & -c_3 & c_2 \\ c_3 & 0 & -c_1 \\ -c_2 & c_1 & 0 \end{bmatrix}. \tag{6}$$

In spatial notation, inertia matrices are also expressed as 6×6 matrices. An inertia matrix may be defined for each individual link of a chain, as well as the reference member, in its own corresponding coordinate system. For the reference member, this matrix, $\mathbf{I}_0$, is represented as follows:

$$\mathbf{I}_0 = \begin{bmatrix} \bar{\mathbf{I}}_0 & \tilde{\mathbf{h}}_0 \\ \tilde{\mathbf{h}}_0^{T} & \mathbf{M}_0 \end{bmatrix}, \tag{7}$$

where $\mathbf{M}_0$ is a 3×3 diagonal matrix of the mass of the reference member, and $\bar{\mathbf{I}}_0$ is the 3×3 moment of inertia tensor at the origin of coordinate frame 0. The matrix $\bar{\mathbf{I}}_0$ is symmetric and positive definite, but not necessarily diagonal. The 3×3 matrix, $\tilde{\mathbf{h}}_0$, is equal to $m_0\tilde{\mathbf{s}}_0$, where m_0 is the mass of the reference member, and $\mathbf{s}_0$ is the position vector of the center of gravity of the reference member from the coordinate origin of frame 0. Because $\bar{\mathbf{I}}_0$ and $\mathbf{s}_0$ are defined in coordinate system 0, the matrix $\mathbf{I}_0$ is constant.

To include general joints and contacts with multiple degrees of freedom in a multibody system, an extended model of the interconnections and interactions between individual bodies of that system is required. In this paper, the general joint model of Roberson and Schwertassek [6] is used for this purpose. This model is also used by Brandl, Johanni, and Otter in [5].

Briefly, each interconnection and/or interaction between two bodies in a simple closed-chain mechanism, hereafter referred to as a "general joint", is described in terms of two orthogonal vector spaces, ϕ and ϕ^c. The matrix ϕ is of dimension $6 \times n$, where n represents the number of degrees of freedom of the general joint, and it has full column rank. It represents the free modes of the joint, and its columns make up a basis for this free vector space. We will refer to ϕ as the *motion space* of the general joint. The matrix ϕ^c, which is $6 \times (6 - n)$ and also of full rank, represents the constrained modes of the general joint. It is orthogonal to ϕ, and may be called the *constraint space* of the joint. Both ϕ and ϕ^c are usually resolved in the joint frame, and thus, they are both constant.

III. Dynamic Properties of Individual Chains and Reference Member

Each chain in a simple closed-chain mechanism is governed by the dynamic equations of motion for a single chain. For chain k, $k = 1, \ldots, m$, these are:

$$\tau_k = \mathbf{H}_k \ddot{\mathbf{q}}_k + \mathbf{C}_k \dot{\mathbf{q}}_k + \mathbf{G}_k + \mathbf{J}_k^T \mathbf{f}_k, \tag{8}$$

where

$$\begin{aligned}
\tau_k &= N_k \times 1 \text{ applied general joint torque/force vector,} \\
\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_k &= N_k \times 1 \text{ general joint position, rate, and acceleration vectors,} \\
\mathbf{H}_k &= N_k \times N_k \text{ joint space inertia matrix,} \\
\mathbf{C}_k &= N_k \times N_k \text{ centripetal/Coriolis matrix,} \\
\mathbf{G}_k &= N_k \times 1 \text{ gravity vector,} \\
\mathbf{J}_k &= 6 \times N_k \text{ Jacobian matrix,}
\end{aligned}$$

and $\mathbf{f}_k$ is the (6×1) spatial force vector exerted by chain k on the reference member.

Note that $\mathbf{H}_k$, $\mathbf{C}_k$, $\mathbf{G}_k$, and $\mathbf{J}_k$ are functions only of the general joint position and rate vectors, $\mathbf{q}_k$ and $\dot{\mathbf{q}}_k$, respectively. Recall also that the "base" of each chain is the support surface, and the "tip" of each chain touches the reference member. The components of $\mathbf{q}_k$ and τ_k correspond to the general joints of each chain, starting with the joint between link 1 and the support surface and ending with the joint preceding link N. The basic unknowns in Eq. (8) are the general joint accelerations, $\ddot{\mathbf{q}}_k$, and the components of the force vector, $\mathbf{f}_k$, in the constrained directions of the general joint at the tip of chain k.

We may use the dynamic equations of motion to partition the joint acceleration and spatial tip acceleration vectors of each chain into the difference of two terms, one known and one unknown. For each chain, we may write [11]:

$$\ddot{\mathbf{q}}_k = (\ddot{\mathbf{q}}_k)_{open} - (\mathbf{H}_k^{-1}\mathbf{J}_k^T)\,\mathbf{f}_k, \tag{9}$$

$$= (\ddot{\mathbf{q}}_k)_{open} - \boldsymbol{\Omega}_k\,\mathbf{f}_k, \tag{10}$$

where $(\ddot{\mathbf{q}}_k)_{open}$ is the vector of joint accelerations for chain k in an open, unconstrained configuration ($\mathbf{f}_k = \mathbf{0}$), and $\boldsymbol{\Omega}_k$ is a function of the joint positions for chain k. Likewise, for $\ddot{\mathbf{x}}_k$, the tip acceleration for each chain:

$$\ddot{\mathbf{x}}_k = (\ddot{\mathbf{x}}_k)_{open} - (\mathbf{J}_k\,\mathbf{H}_k^{-1}\mathbf{J}_k^T)\,\mathbf{f}_k, \tag{11}$$

$$= (\ddot{\mathbf{x}}_k)_{open} - \boldsymbol{\Lambda}_k^{-1}\mathbf{f}_k, \tag{12}$$

where $(\ddot{\mathbf{x}}_k)_{open}$ is the spatial tip acceleration vector for chain k in an open, unconstrained configuration, and $\boldsymbol{\Lambda}_k^{-1}$ is the inverse operational space inertia matrix for chain k, defined at the tip of the chain [7,11].

The open-chain terms, $(\ddot{\mathbf{q}}_k)_{open}$ and $(\ddot{\mathbf{x}}_k)_{open}$, are completely defined for each chain given the present state joint positions and rates, $\mathbf{q}_k$ and $\dot{\mathbf{q}}_k$, the applied joint torques/forces in the free directions, τ_k, and the motion of the support surface. An appropriate open-chain Direct Dynamics algorithm may be used to calculate these terms. The $O(N)$ recursive algorithms of [5,10] are very efficient for this computation, and the linear order of computation is highly desirable. Because the joint positions are known, $\boldsymbol{\Omega}_k$ and $\boldsymbol{\Lambda}_k^{-1}$ are also defined. Efficient algorithms for $\boldsymbol{\Omega}_k$ and $\boldsymbol{\Lambda}_k^{-1}$ for a single chain are derived in [11], including an $O(N)$ recursive algorithm which is the most efficient for $N \geq 6$.

The dynamic behavior of the reference member may be described using a spatial force balance equation for that body. The sum of the spatial forces exerted by each chain on the reference member and any other external spatial forces (including gravity) are equal to the resultant force on the reference member. Using spatial notation, we may write the force balance equation as follows:

$$\mathbf{F}_0 = \sum_{k=1}^{m} {}^0\mathbf{f}_k + \mathbf{g}_0, \tag{13}$$

where

$\mathbf{F}_0$ = 6×1 resultant spatial force vector applied to the reference member,
${}^0\mathbf{f}_k$ = 6×1 spatial force vector applied by chain k to the reference member,

and

$\mathbf{g}_0$ = 6×1 external spatial force vector applied to the reference member (including gravity).

Each force term in Eq. (13) is defined with respect to the coordinate frame attached to the reference member (frame 0). Applying the basic Newton-Euler equations, we may also write the resultant vector, $\mathbf{F}_0$, as follows:

$$\mathbf{F}_0 = \mathbf{I}_0 \mathbf{a}_0 + \mathbf{v}_0 \times \mathbf{I}_0 \mathbf{v}_0, \tag{14}$$
$$= \mathbf{I}_0 \mathbf{a}_0 + \mathbf{b}_0, \tag{15}$$

where

$\mathbf{I}_0$ = 6×6 spatial inertia of the reference member,
$\mathbf{a}_0$ = 6×1 spatial acceleration of the reference member,
$\mathbf{v}_0$ = 6×1 spatial velocity of the reference member.

Both $\mathbf{v}_0$ and $\mathbf{a}_0$ refer to the motion of the coordinate origin of frame 0. The spatial inertia matrix, $\mathbf{I}_0$, is also defined at this point, and it is known and constant. Because $\mathbf{v}_0$ is given for the present state, the velocity-dependent term, $\mathbf{b}_0$, is known. If we combine Eqs. (13) and (15), we finally obtain the following dynamic equation for the reference member:

$$\sum_{k=1}^{m} {}^0\mathbf{f}_k + \mathbf{g}_0 = \mathbf{I}_0 \mathbf{a}_0 + \mathbf{b}_0. \tag{16}$$

In this equation, the basic unknowns are $\mathbf{a}_0$ and the components of ${}^0\mathbf{f}_k$ in the constrained directions of the general joint at the tip of chain k.

IV. Multiple Chain Algorithm

In developing an efficient algorithm for the dynamic simulation of simple closed-chain mechanisms, we are naturally led to consider the relationship between the physical structure of the robotic system and the computational structure of the desired algorithm. Intuitively, it seems apparent that the structural parallelism present in a simple closed-chain mechanism should lead to computational parallelism in the solution of the Direct Dynamics problem for that mechanism.

More specifically, in a simple closed-chain mechanism, the m actuated chains act on the reference member in parallel, and their motion is coupled with that of the reference member. If the reference member is removed, the chains may function independently. Computationally, the physical removal of the reference member corresponds to solving for the forces which are exerted on it by each chain. Once these forces are known, the system is equivalent to a group of independent chains with known tip forces. The joint accelerations may then be computed for each chain separately. Given enough processors (one per chain), the computations for each chain may be carried out in parallel.

We will illustrate the basic methodology of the new simulation algorithm by first examining a simple special case. Consider m manipulators rigidly grasping a common object. Each manipulator has six degrees of freedom, and no chain is in a singular position. For simplicity, we will express all of the relevant equations in absolute coordinates. Because each chain tip is rigidly attached to the reference member, we may write:

$$\ddot{\mathbf{x}}_k = \mathbf{a}_0 \tag{17}$$

for each chain k, $k = 1,\ldots,m$. Thus, the operational space dynamic equation for each chain, as given in Eq. (12), takes the form:

$$\mathbf{a}_0 = (\ddot{\mathbf{x}}_k)_{open} - \Lambda_k^{-1}\mathbf{f}_k. \tag{18}$$

Because no chain is in a singular position, and each chain has a full six degrees of freedom, Λ_k is defined [11]. We may, therefore, solve for the spatial tip force exerted by chain k on the reference member, $\mathbf{f}_k$, as follows:

$$\mathbf{f}_k = \Lambda_k \left[(\ddot{\mathbf{x}}_k)_{open} - \mathbf{a}_0\right]. \tag{19}$$

With this equation we have established an explicit relationship between the spatial tip force, $\mathbf{f}_k$, and the spatial acceleration of the reference member, $\mathbf{a}_0$. This expression may be used in the reference member dynamic equation, given in Eq. (16), to obtain:

$$\sum_{k=1}^{m} \Lambda_k \left[(\ddot{\mathbf{x}}_k)_{open} - \mathbf{a}_0\right] = \mathbf{I}_0\,\mathbf{a}_0 + \mathbf{b}_0 - \mathbf{g}_0. \tag{20}$$

The only unknown in Eq (20) is $\mathbf{a}_0$, the spatial acceleration of the reference member. Collecting terms, we may write:

$$\left[\mathbf{I}_0 + \sum_{k=1}^{m} \Lambda_k\right] \mathbf{a}_0 = \left[\sum_{k=1}^{m} \Lambda_k (\ddot{\mathbf{x}}_k)_{open} - \mathbf{b}_0 + \mathbf{g}_0\right]. \tag{21}$$

We may now solve for $\mathbf{a}_0$ from this linear system of algebraic equations using any linear system solver. Note that the characteristic matrix is just the sum of the operational space inertia matrices of the individual chains and reference member, and is only 6×6. With $\mathbf{a}_0$ known, we may also solve explicitly for the spatial tip force $\mathbf{f}_k$, $k = 1,\ldots,m$, using Eq. (19). Thus, the motion of the reference member and the spatial force exerted at the tip of each chain are completely defined. The simple closed-chain mechanism is effectively decoupled. Each manipulator may now be treated as an independent chain with a known spatial tip force. The joint accelerations for each chain may be computed separately using an appropriate Direct Dynamics algorithm and then integrated to obtain the next state.

The method outlined above is quite straightforward. Of course, the illustrated example represents a special case. We will now develop a similar approach for a general simple closed-chain mechanism. Consider a mechanism with m chains, each with an arbitrary number of degrees of freedom, N. The interaction between each chain tip and the reference member is arbitrary and will be modelled using the general joint model of [6]. To begin, we will derive an explicit relationship between the spatial acceleration of each chain tip and the spatial acceleration of the reference member. The spatial acceleration of the tip of chain k is denoted by $\ddot{\mathbf{x}}_k$. The relative spatial acceleration between the tip of chain k and the reference member, $\ddot{\mathbf{x}}_k^r$, resolved in the orthogonal vector spaces of the general joint between them, may be written:

$$\ddot{\mathbf{x}}_k^r = (\phi)_k\,\alpha_k + (\phi^c)_k\,\alpha_k^c, \tag{22}$$

where $(\phi)_k$ and $(\phi^c)_k$ are the motion space and constraint space of the general joint at the tip of chain k, respectively. The quantities α_k and α_k^c are the corresponding components of relative acceleration in the free and constrained directions. For each chain, $(\phi)_k$, $(\phi^c)_k$, and α_k^c are known, while α_k is unknown. The sum of $\ddot{\mathbf{x}}_k$ and $\ddot{\mathbf{x}}_k^r$ is just the spatial acceleration of the reference member on the far side of the general joint between it and chain k, $\mathbf{a}_0^k$. Thus, we may write:

$$\mathbf{a}_0^k = \ddot{\mathbf{x}}_k + \ddot{\mathbf{x}}_k^r, \tag{23}$$

$$= \ddot{\mathbf{x}}_k + (\phi)_k \alpha_k + (\phi^c)_k \alpha_k^c. \tag{24}$$

We may also express $\mathbf{a}_0^k$ in terms of the spatial acceleration of the reference member, $\mathbf{a}_0$, as follows:

$$\mathbf{a}_0^k = \mathbf{X}_0^k \mathbf{a}_0 + \zeta_0^k, \tag{25}$$

where $\mathbf{X}_0^k \equiv {}^N\mathbf{X}_0^k$ is the spatial transformation between coordinate frame 0 and the coordinate frame associated with the general joint at the tip of chain k. The quantity ζ_0^k is the 6×1 bias acceleration vector which is a function of the position and spatial velocity of the reference member. Because the present state of the entire system is given, both $\mathbf{X}_0^k$ and ζ_0^k are known.

Equating the two expressions above for $\mathbf{a}_0^k$, we obtain the following:

$$\ddot{\mathbf{x}}_k + (\phi)_k \alpha_k + (\phi^c)_k \alpha_k^c = \mathbf{X}_0^k \mathbf{a}_0 + \zeta_0^k. \tag{26}$$

This equation matches the spatial accelerations at the coupling point between chain k and the reference member, giving an explicit relationship between $\ddot{\mathbf{x}}_k$ and $\mathbf{a}_0$ when the coupling is arbitrary. The basic unknowns in Eq. (26) are $\ddot{\mathbf{x}}_k$, α_k, and $\mathbf{a}_0$. All other vectors and matrices may be computed rather simply from the initial information given for the simulation problem.

To decouple the chains and the reference member, we need an explicit mathematical relationship between the spatial force exerted by chain k on the reference member, $\mathbf{f}_k$, and the spatial acceleration of the reference member, $\mathbf{a}_0$. Equation (26) relates the spatial acceleration of the reference member and the spatial acceleration of the tip of chain k. We may eliminate α_k, the unknown components of the relative acceleration, by projecting Eq. (26) onto the constraint space of the corresponding general joint as follows:

$$(\phi^c)_k^T \left[\ddot{\mathbf{x}}_k + (\phi)_k \alpha_k + (\phi^c)_k \alpha_k^c\right] = (\phi^c)_k^T \left[\mathbf{X}_0^k \mathbf{a}_0 + \zeta_0^k\right]. \tag{27}$$

By definition [11]:

$$(\phi^c)_k^T (\phi)_k = \mathbf{0}, \tag{28}$$

and

$$(\phi^c)_k^T (\phi^c)_k = \mathbf{1}. \tag{29}$$

Thus, we may write:

$$(\phi^c)_k^T \ddot{\mathbf{x}}_k + \alpha_k^c = (\phi^c)_k^T \left[\mathbf{X}_0^k \mathbf{a}_0 + \zeta_0^k\right]. \tag{30}$$

Equation (12) defines $\ddot{\mathbf{x}}_k$ in terms of the desired force vector, $\mathbf{f}_k$. If we combine Eqs. (12) and (30), we obtain:

$$(\phi^c)_k^T \left[(\ddot{\mathbf{x}}_k)_{open} - \mathbf{\Lambda}_k^{-1} \mathbf{f}_k\right] = (\phi^c)_k^T \left[\mathbf{X}_0^k \mathbf{a}_0 + \zeta_0^k\right] - \alpha_k^c, \tag{31}$$

or

$$\left[(\phi^c)_k^T\,\Lambda_k^{-1}\right]\,\mathbf{f}_k = \left[\alpha_k^c - (\phi^c)_k^T\,\zeta_0^k + (\phi^c)_k^T\,(\ddot{\mathbf{x}}_k)_{open}\right] - \left[(\phi^c)_k^T\,\mathbf{X}_0^k\right]\,\mathbf{a}_0. \tag{32}$$

The first bracketed term on the right side of Eq. (32) is completely known. The only unknowns in this equation are the constraint components of the force vector, $\mathbf{f}_k$, and the spatial acceleration, $\mathbf{a}_0$. We may now pursue an explicit relationship between these two vectors.

Like the relative acceleration vector, $\mathbf{f}_k$ may also be resolved in the orthogonal vector spaces of the general joint at the tip of chain k as follows:

$$\mathbf{f}_k = (\phi)_k\,\mathbf{h}_k + (\phi^c)_k\,\mathbf{h}_k^c, \tag{33}$$

where $\mathbf{h}_k$ is the vector of known force components in the free directions, and $\mathbf{h}_k^c$ is the vector of unknown force components in the constrained directions. Combining Eqs. (32) and (33), we obtain:

$$\begin{aligned}(\phi^c)_k^T\,\Lambda_k^{-1}\left[(\phi)_k\,\mathbf{h}_k + (\phi^c)_k\,\mathbf{h}_k^c\right] &= \left[\alpha_k^c - (\phi^c)_k^T\,\zeta_0^k + (\phi^c)_k^T\,(\ddot{\mathbf{x}}_k)_{open}\right] \\ &\quad - \left[(\phi^c)_k^T\,\mathbf{X}_0^k\right]\,\mathbf{a}_0.\end{aligned} \tag{34}$$

If the spatial acceleration of the reference member is known, we may find an explicit solution for the unknown force components at the tip of chain k from the following set of linear algebraic equations:

$$\begin{aligned}\left[(\phi^c)_k^T\,\Lambda_k^{-1}(\phi^c)_k\right]\,\mathbf{h}_k^c &= \left\{\alpha_k^c - (\phi^c)_k^T\left[\zeta_0^k - (\ddot{\mathbf{x}}_k)_{open} + \Lambda_k^{-1}(\phi)_k\,\mathbf{h}_k\right]\right\} \\ &\quad - \left[(\phi^c)_k^T\,\mathbf{X}_0^k\right]\,\mathbf{a}_0, \qquad (35)\\ &= \mathbf{S}_k - \left[(\phi^c)_k^T\,\mathbf{X}_0^k\right]\,\mathbf{a}_0, \qquad (36)\end{aligned}$$

where $\mathbf{S}_k$ is known. Even when $\mathbf{a}_0$ is unknown, we may still find a solution for $\mathbf{h}_k^c$ in terms of the unknown $\mathbf{a}_0$. The solution will have the following form:

$$\begin{aligned}\mathbf{h}_k^c &= \mathbf{M}_k\left[\mathbf{S}_k - (\phi^c)_k^T\,\mathbf{X}_0^k\,\mathbf{a}_0\right], \qquad (37)\\ &= \mathbf{M}_k\,\mathbf{S}_k - \left[\mathbf{M}_k\,(\phi^c)_k^T\,\mathbf{X}_0^k\right]\,\mathbf{a}_0. \qquad (38)\end{aligned}$$

If $(n_c)_k$ is the number of degrees of constraint for the general joint at the tip of chain k, then $\mathbf{M}_k$ is the $(n_c)_k \times (n_c)_k$ transformation matrix which solves for $\mathbf{h}_k^c$. By carefully considering the rank of the coefficient matrix, this general solution can still be used for a chain in a singular position or a chain with less than six original degrees of freedom [11]. This solution procedure requires $O[(n_c)_k^3]$ scalar operations.

Note that if chain k is rigidly grasping the reference member, then the constraint space for this general joint, $(\phi^c)_k$, is the 6×6 identity matrix. In this case, also note that α_k^c and $\mathbf{h}_k$ are identically zero for each chain. If chain k has six degrees of freedom and is not in a singular position, then $\mathbf{M}_k$ will be exactly equal to Λ_k, the operational space inertia matrix for chain k, and the solution for $\mathbf{h}_k^c$ will be:

$$\mathbf{h}_k^c = \Lambda_k\left[\mathbf{S}_k - \mathbf{X}_0^k\,\mathbf{a}_0\right]. \tag{39}$$

This solution corresponds to the simple example discussed at the beginning of this section, but now expressed in local coordinates.

Given the general solution for $\mathbf{h}_k^c$ in Eq. (38), the force vector, $\mathbf{f}_k$, may now be written:

$$\begin{aligned}\mathbf{f}_k &= (\phi)_k\,\mathbf{h}_k + (\phi^c)_k\,\mathbf{h}_k^c, &(40)\\ &= [(\phi)_k\,\mathbf{h}_k + (\phi^c)_k\,\mathbf{M}_k\,\mathbf{S}_k] - \left[(\phi^c)_k\,\mathbf{M}_k\,(\phi^c)_k^T\,\mathbf{X}_0^k\right]\,\mathbf{a}_0, &(41)\\ &= \mathbf{P}_k - \mathbf{R}_k\,\mathbf{a}_0, &(42)\end{aligned}$$

where $\mathbf{P}_k$ and $\mathbf{R}_k$ are of dimension 6×1 and 6×6, respectively, and both may be computed from known quantities. We now have an explicit equation relating the force vector exerted by chain k and the spatial acceleration of the reference member. We may combine this information with the dynamic equation for the reference member to solve for $\mathbf{a}_0$ explicitly.

The dynamic equation for the reference member given in Eq. (16) may be rewritten as follows:

$$\sum_{k=1}^{m}(\mathbf{X}_0^k)^T\mathbf{f}_k + \mathbf{g}_0 = \mathbf{I}_0\,\mathbf{a}_0 + \mathbf{b}_0. \tag{43}$$

where $\mathbf{f}_k$ is the spatial force exerted by chain k on the reference member, expressed in the coordinate frame of the general joint at the chain tip. If the expression for $\mathbf{f}_k$ in Eq. (42) is used in Eq. (43), we obtain:

$$\sum_{k=1}^{m}(\mathbf{X}_0^k)^T(\mathbf{P}_k - \mathbf{R}_k\,\mathbf{a}_0) = \mathbf{I}_0\,\mathbf{a}_0 + \mathbf{b}_0 - \mathbf{g}_0. \tag{44}$$

Summing like terms, we may write:

$$\left[\mathbf{I}_0 + \sum_{k=1}^{m}(\mathbf{X}_0^k)^T\mathbf{R}_k\right]\,\mathbf{a}_0 = \left[\sum_{k=1}^{m}(\mathbf{X}_0^k)^T\mathbf{P}_k - \mathbf{b}_0 + \mathbf{g}_0\right], \tag{45}$$

or, expanding $\mathbf{R}_k$,

$$\left[\mathbf{I}_0 + \sum_{k=1}^{m}(\mathbf{X}_0^k)^T(\phi^c)_k\,\mathbf{M}_k\,(\phi^c)_k^T\,(\mathbf{X}_0^k)\right]\,\mathbf{a}_0 = \left[\sum_{k=1}^{m}(\mathbf{X}_0^k)^T\mathbf{P}_k - \mathbf{b}_0 + \mathbf{g}_0\right]. \tag{46}$$

In Eq. (46), the 6×1 spatial acceleration vector of the reference member, $\mathbf{a}_0$, is the only unknown. We may find a solution for $\mathbf{a}_0$ from the given set of linear algebraic equations using any efficient linear system solver. Because the required system solution always involves a 6×6 coefficient matrix, the computational cost of solving for $\mathbf{a}_0$ is constant.

The coefficient matrix of $\mathbf{a}_0$ in Eq. (46) represents the combined inertial properties of all the chains and the reference member. The inertial properties of each chain are first projected to the tip of that chain by computing the inverse operational space inertia matrix, Λ_k^{-1}. Along the free directions of the general joint which connects the chain tip and the reference member, the projected inertia of the chain is not felt by the reference member. Along the constrained directions of the joint, however, the corresponding components of Λ_k^{-1} are reflected across to the reference member. These components, spatially transformed to the coordinate origin of frame 0, are combined with the spatial inertia of the reference member, $\mathbf{I}_0$. This combination represents the effective operational space inertia of the simple closed-chain mechanism defined at the coordinate origin of frame 0. It is the effective inertia "felt" by the reference member in the present state. The bracketed term on the right side of Eq. (46) represents the spatial forces which act on the reference member at the given instant.

Once the spatial acceleration of the reference member is known, the spatial force vector applied by chain k to the reference member is defined by Eq. (42). That is, with $\mathbf{a}_0$ given, we may compute $\mathbf{f}_k$ as follows:

$$\mathbf{f}_k = \mathbf{P}_k - \mathbf{R}_k \mathbf{a}_0. \tag{47}$$

Recall that $\mathbf{f}_k$ is defined with respect to the coordinate frame of the general joint between chain k and the reference member. The explicit knowledge of $\mathbf{f}_k$ allows us to treat chain k as an independent chain with a known tip force. We may now solve for the general closed-chain joint accelerations for chain k using Eq. (10), repeated here for convenience:

$$\ddot{\mathbf{q}}_k = (\ddot{\mathbf{q}}_k)_{open} - \mathbf{\Omega}_k \mathbf{f}_k. \tag{48}$$

The application of Eq. (48) to every actuated chain in the simple closed-chain mechanism results in a complete solution to the Direct Dynamics problem for this robotic system. The next state positions and velocities may be computed by integrating the appropriate quantities for each chain and the reference member. As discussed in [11], small amounts of negative position and rate feedback may be employed to counteract the drift which is a result of the integration process, and which would violate the kinematic constraints.

The algorithm developed here for simple closed-chain mechanisms may be presented as a series of five steps. They are as follows:

1. The Open Chain Solution,
2. Calculation of the Spatial Acceleration of the Reference Member,
3. Calculation of the Spatial Chain Tip Forces,
4. Calculation of the Closed-Chain Joint Accelerations,
5. Integration for the Next State.

The fundamental computations required in each of these steps are summarized in Table 1. In Step 1, the Direct Dynamics problem is solved for each chain of the mechanism assuming that the reference member has been removed and each chain is in an open, unconstrained state. The general open-chain acceleration vectors, $(\ddot{\mathbf{q}}_k)_{open}$ and $(\ddot{\mathbf{x}}_k)_{open}$, are computed for each chain, along with the position-dependent matrices, $\mathbf{\Omega}_k$ and $\mathbf{\Lambda}_k^{-1}$. In Step 2, Eq. (45) is used to find an explicit solution for $\mathbf{a}_0$, the spatial acceleration of the reference member, via linear system solution. The quantities $(\mathbf{M}_k \mathbf{S}_k)$ and $[\mathbf{M}_k (\phi^c)_k^T \mathbf{X}_0^k]$, required for both $\mathbf{P}_k$ and $\mathbf{R}_k$, are computed in the determination of the explicit relationship between $\mathbf{h}_k^c$ and $\mathbf{a}_0$. This relationship is found by linear system solution using Eq. (36), with the solution taking the form of Eq. (38). In Step 3, this solution is used in Eq. (47) to solve for the spatial force vector exerted on the reference member by each chain. In Step 4, the general closed-chain joint accelerations are computed for each chain using Eq. (48), given the spatial tip force vector. In Step 5, the appropriate rates and accelerations are integrated to obtain the next state positions and rates for the system.

Note that the first step may be carried out for all chains in parallel, if enough processors are available (one per chain). Once the second step is complete and $\mathbf{a}_0$ is known, the third, fourth, and fifth steps may also be carried out for all chains simultaneously. Thus, taking advantage of the structural parallelism inherent in the simple closed-chain system has led to parallelism in the computational structure of the simulation algorithm.

V. Computational Requirements

We will now consider the computational requirements of the dynamic simulation algorithm for simple closed-chain mechanisms. First, the number of scalar operations required for each chain of the mechanism will be tabulated, followed by the number of operations required to compute the spatial

Table 1: Dynamic Simulation Algorithm for Simple Closed-Chain Mechanisms

Given: α_k^c, $\mathbf{h}_k$, $(\phi^c)_k$, $(\phi)_k$, and with
$\mathbf{X}_0^k$, $\mathbf{b}_0$, $\mathbf{g}_0$, ζ_0^k determined from the reference member state;

Step 1. Compute $(\ddot{\mathbf{q}}_k)_{open}$, $(\ddot{\mathbf{x}}_k)_{open}$, $\mathbf{\Omega}_k$, and $\mathbf{\Lambda}_k^{-1}$; $k = 1, \ldots, m$.

Step 2. Solve for $\mathbf{a}_0$:

$$\left[\mathbf{I}_0 + \sum_{k=1}^{m} (\mathbf{X}_0^k)^T \mathbf{R}_k\right] \mathbf{a}_0 = \left[\sum_{k=1}^{m} (\mathbf{X}_0^k)^T \mathbf{P}_k - \mathbf{b}_0 + \mathbf{g}_0\right],$$

with

$$\begin{aligned}
\mathbf{P}_k &= [(\phi)_k \mathbf{h}_k + (\phi^c)_k \mathbf{M}_k \mathbf{S}_k], \\
\mathbf{R}_k &= \left[(\phi^c)_k \mathbf{M}_k (\phi^c)_k^T \mathbf{X}_0^k\right], \\
\mathbf{S}_k &= \alpha_k^c - (\phi^c)_k^T \left[\zeta_0^k - (\ddot{\mathbf{x}}_k)_{open} + \mathbf{\Lambda}_k^{-1} (\phi)_k \mathbf{h}_k\right],
\end{aligned}$$

and where $(\mathbf{M}_k \mathbf{S}_k)$ and $\left[\mathbf{M}_k (\phi^c)_k^T \mathbf{X}_0^k\right]$ are determined by the solution of:

$$(\mathbf{M}_k^{-1})\, \mathbf{h}_k^c = \left[(\phi^c)_k^T \mathbf{\Lambda}_k^{-1} (\phi^c)_k\right] \mathbf{h}_k^c = \mathbf{S}_k - \left[(\phi^c)_k^T \mathbf{X}_0^k\right] \mathbf{a}_0.$$

Step 3. Solve for $\mathbf{f}_k$; $k = 1, \ldots, m$:

$$\mathbf{f}_k = \mathbf{P}_k - \mathbf{R}_k \mathbf{a}_0.$$

Step 4. Solve for $\ddot{\mathbf{q}}_k$; $k = 1, \ldots, m$:

$$\ddot{\mathbf{q}}_k = (\ddot{\mathbf{q}}_k)_{open} - \mathbf{\Omega}_k \mathbf{f}_k.$$

Step 5. Integrate to obtain the next state positions and rates for the system.

Table 2: Computations Per Chain in the Simple Closed-Chain Dynamic Simulation Algorithm

Calculation	#Mult.	#Add.	#Mult. ($N = 6$, $n_c = 3$)	#Add. ($N = 6$, $n_c = 3$)
$\ddot{\mathbf{q}}_{open}, \ddot{\mathbf{x}}_{open}$	$250N - 182$	$220N - 167$	1318	1153
$\mathbf{\Omega}, \mathbf{A}^{-1}$	$400N - 621$	$320N - 528$	1779	1392
$\mathbf{P}, \mathbf{R}$	$\frac{1}{6}n_c^3 + 6\frac{1}{2}n_c^2 + 5\frac{1}{3}n_c + 26$	$\frac{1}{6}n_c^3 + 6n_c^2 + \frac{5}{6}n_c + 10$	105	71
$\mathbf{X}^T\mathbf{P}, \mathbf{X}^T\mathbf{R}$	$36n_c + 20$	$36n_c - 24$	128	84
$\mathbf{f}$	36	36	36	36
$\ddot{\mathbf{q}}$	$6N$	$6N$	36	36
Total:	$656N - 767$ $+(\frac{1}{6}n_c^3 + 6\frac{1}{2}n_c^2 + 41\frac{1}{3}n_c + 46)$	$546N - 659$ $+(\frac{1}{6}n_c^3 + 6n_c^2 + 36\frac{5}{6}n_c - 14)$	3402	2772

acceleration of the reference member. The computational complexity of the complete algorithm will then be discussed, and the parallel implementation of this algorithm will be considered.

Table 2 lists the number of scalar operations (multiplications, additions) required in the simulation algorithm for each chain of a simple closed-chain mechanism. The operations are tabulated for the case of an N degree-of-freedom serial-link chain with simple revolute and/or prismatic joints only. The $O(N)$ Direct Dynamics algorithm of [5] is used to compute the open-chain terms, $\ddot{\mathbf{q}}_{open}$ and $\ddot{\mathbf{x}}_{open}$. The $O(N)$ Force Propagation Method of [11] is used to compute $\mathbf{\Omega}$ and $\mathbf{\Lambda}^{-1}$. All $\ddot{\mathbf{q}}_{open}$, $\ddot{\mathbf{x}}_{open}$, $\mathbf{\Omega}$, and $\mathbf{\Lambda}^{-1}$ are computed in Step 1 of the simulation algorithm.

In Step 2 of the simulation algorithm, the spatial acceleration of the reference member is calculated using Eq. (45). For this task, $\mathbf{X}^T\mathbf{P}$ and $\mathbf{X}^T\mathbf{R}$ must be computed for each chain. The number of operations required to compute $\mathbf{P}$, $\mathbf{R}$, $\mathbf{X}^T\mathbf{P}$, and $\mathbf{X}^T\mathbf{R}$ are also listed in Table 2. In this case, the number of operations is a function of the number of degrees of constraint at the general joint between the chain tip and the reference member (n_c). This number can never be greater than six. The computational complexity of these calculations is $O(n_c^3)$ due to the linear system solution required in the computation of both $\mathbf{P}$ and $\mathbf{R}$ (see Table 1).

The spatial force vector, $\mathbf{f}$, exerted by each chain on the reference member, and the closed-chain joint accelerations for the chain, $\ddot{\mathbf{q}}$, are calculated in Steps 3 and 4 of the simulation algorithm, respectively. The appropriate equations are given in Table 1. The operations required to calculate these vectors complete the table. The operations required for the special case of $N = 6$ and $n_c = 3$ are given in the last two columns of Table 2. This value of n_c could correspond to a hard point contact between a manipulator tip and surface of a load object when the tip is not slipping.

Given the computations required for each individual chain, the number of scalar operations needed to compute the spatial acceleration of the reference member, $\mathbf{a}_0$, is given in Table 3. Equation (45) is used to obtain the solution, which requires $O(m)$ spatial additions and a single 6×6 symmetric linear system solution. Thus, the number of operations required for $\mathbf{a}_0$ is a function only of m, the number of chains in the simple closed-chain mechanism. The example of three chains ($m = 3$) is given in the last two columns of this table.

To determine the total number of scalar operations required to simulate the entire simple closed-chain mechanism, the number of operations required for a single chain is simply multiplied by m,

Table 3: Computations for the Spatial Acceleration of the Reference Member

Calculation	#Mult.	#Add.	#Mult. ($m = 3$)	#Add. ($m = 3$)
$\mathbf{a}_0$	86	$27m + 71$	86	152

the number of chains, and added to the computations required for $\mathbf{a}_0$ for the same number of chains. Thus, the computational complexity of the complete simulation algorithm is $O(mN)$ for a given value of $n_c \leq 6$.

The total computational complexity discussed in the previous section only considered the execution of the simulation algorithm on a single processor. In order to speed up the simulation, parallel processing may be investigated. If a single processor is used for the entire system of m chains, the computational complexity of the simulation algorithm is $O(mN)$ for a given $n_c \leq 6$. Given $\mathbf{a}_0$, all computations for each chain may be carried out independently. Thus, if m processors are available, the computational tasks associated with each chain may be performed in parallel, and the computational complexity of the operations required for the m chains may be reduced to $O(N)$. Of course, the computations required to compute $\mathbf{a}_0$ must also be considered. These operations may also be implemented in parallel on the m available processors. Equation (45) requires $O(m)$ spatial additions to compute $\mathbf{a}_0$. On $(m + 1)/2$ parallel processors, this task may be carried out in $O(\log_2 m+1)$ operations by using the recursive doubling approach [12]. Thus, on m parallel processors, the computational complexity of the entire dynamic simulation algorithm may be reduced to $O(N) + O(\log_2 m+1)$.

VI. Summary and Conclusions

In this paper, a general and efficient dynamic simulation algorithm for simple closed-chain mechanisms was derived. The algorithm is applicable to both Type 0 and Type 1 mechanisms. Both types of mechanisms are modelled in a convenient and general manner through the use of the general joint concept. The operational space inertia matrix of each chain is used to project the dynamic properties of the chain to its tip where it is coupled to the reference member. By combining the operational space inertia of each chain with the model of the general joint at each chain tip, a solution may be found for the spatial acceleration of the reference member and the spatial force vector exerted on it by each chain. Once the force vectors are completely defined, the system is effectively decoupled, and the joint accelerations for each chain may be computed separately.

The computational complexity of the new simulation algorithm is $O(mN)$ when implemented on a single processor. The linear dependence on N is a significant improvement over previous simulation algorithms such as that presented in [1]. The computational complexity of the new algorithm may be further reduced to $O(N) + O(\log_2 m+1)$ if it is implemented on m processors in parallel.

VII. Acknowledgments

Support for this research was provided by a Presidential Fellowship from The Ohio State University and by the National Science Foundation under Computational Engineering Grant No. EET-8718434.

References

[1] S. Y. Oh and D. E. Orin, "Dynamic Computer Simulation of Multiple Closed-Chain Robotic Mechanisms," in *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, pp. 15–20, San Francisco, CA, April 1986.

[2] R. H. Lathrop, "Constrained (Closed-Loop) Robot Simulation by Local Constraint Propagation," in *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, pp. 689–694, San Francisco, CA, April 1986.

[3] H. Brandl, R. Johanni, and M. Otter, "An Algorithm for the Simulation of Multibody Systems with Kinematic Loops," in *Proceedings of the IFToMM Seventh World Congress on the Theory of Machines and Mechanisms*, Sevilla, Spain, September 1987.

[4] G. Rodriguez and K. Kreutz, "Recursive Mass Matrix Factorization and Inversion: An Operator Approach to Open- and Closed-Chain Multibody Dynamics," Technical Report 88-11, Jet Propulsion Laboratory, Pasadena, CA, March 1988.

[5] H. Brandl, R. Johanni, and M. Otter, "A Very Efficient Algorithm for the Simulation of Robots and Similar Multibody Systems Without Inversion of the Mass Matrix," in *Proceedings of IFAC/IFIP/IMACS International Symposium on the Theory of Robots*, Vienna, Austria, December 1986.

[6] R. E. Roberson and R. F. Schwertassek, *Introduction to the Dynamics of Multibody Systems.* Springer-Verlag, Berlin, 1987.

[7] O. Khatib, "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation," *IEEE Journal of Robotics and Automation*, vol. RA-3, no. 1, pp. 43–53, 1987.

[8] K. W. Lilly and D. E. Orin, "*O(N)* Recursive Algorithm for the Operational Space Inertia Matrix of a Robot Manipulator," submitted to *11th IFAC World Congress*, August 1990.

[9] R. Featherstone, *Robot Dynamics Algorithms.* Boston: Kluwer Academic Publishers, 1987.

[10] R. Featherstone, "The Calculation of Robot Dynamics Using Articulated-Body Inertias," *The International Journal of Robotics Research*, vol. 2, no. 1, pp. 13–30, Spring 1983.

[11] K. W. Lilly, "Efficient Dynamic Simulation of Multiple Chain Robotic Mechanisms," Ph.D. Thesis, The Ohio State University, 1989.

[12] M. Amin-Javaheri and D. E. Orin, "Parallel Algorithms for Computation of the Manipulator Inertia Matrix," in *Proceedings of NASA Conference on Space Telerobotics*, Pasadena, CA, Jan./Feb. 1989.

A Nearly-Linear Computational-Cost Scheme for the Forward Dynamics of an N-Body Pendulum

Jack C. K. Chou
Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas, P. O. Box 830688, MP 32
Richardson, Texas 75083-0688, USA
Tel: (214)690-2132

Abstract

The dynamic equations of motion of an n-body pendulum with spherical joints are derived to be a mixed system of differential and algebraic equations (DAE's). The DAE's are kept in implicit form to save arithmetic and preserve the sparsity of the system and are solved by the robust implicit integration method. At each solution point, the predicted solution is corrected to its exact solution within given tolerance using Newton's iterative method. For each iteration, a linear system of the form $J\Delta\mathbf{X} = \mathbf{E}$ has to be solved. The computational cost for solving this linear system directly by LU factorization is $O(n^3)$, and it can be reduced significantly by exploring the structure of J. This paper shows that by recognizing the recursive patterns and exploiting the sparsity of the system the multiplicative and additive computational costs for solving $J\Delta\mathbf{X} = \mathbf{E}$ are $O(n)$ and $O(n^2)$, respectively. The formulation and solution method for an n-body pendulum is presented. The computational cost is shown to be nearly linearly proportional to the number of bodies.

1 INTRODUCTION

The general modeling and formulation of an open-chain multi-body system with spherical joints was presented by Chou, Singhal, and Kesavan in 1986 [1]. In this paper, we are interested in a single open kinematic chain without branching which is a special configuration of a general open-chain system. Much attention was paid to this open-chain system by researchers, such as Armstrong [2], because the system is simple and its configuration is similar to robot arms. In Armstrong's work, he presented an $O(n)$ algorithm for the computation of robot forward dynamics. However, in the conclusion of his paper he stated that his program worked only for the joints with three degrees of freedom (i.e., spherical joints). He also claimed that his algorithm can be enhanced to include prismatic and revolute joints. In the author's opinion, once we have joints which possess less than three rotational degrees of freedom connecting two bodies, the bodies are rotationally dependent. In this case, the equations of motion can not be decoupled easily, and we may not be able to find an $O(n)$ algorithm for the complete calculation of robot forward dynamics unless we use very simple and primitive integration methods or other techniques such as parallel computing. In other words, if we use an unreliable integration routine we may get invalid solutions.

A series of n rigid bodies joined sequentially by spherical joints form an n-body pendulum. The bodies are allowed to rotate freely in space, but the motion of translation between adjacent bodies is constrained by the joint. This sequence of bodies can swing in any direction with one end fixed at the ceiling through a spherical joint. Here, we derive the equations of motion of this pendulum in the form of a mixed set of differential and algebraic equations (DAE's) and solve the equations using the robust implicit integration method developed by Petzold [3,4]. The DAE's are kept in implicit form to save arithmetic and preserve the sparsity of the system. At each solution point, the predicted solution is corrected to its exact solution within the given tolerance using Newton's iterative method. For each iteration, a linear system of the form $J\Delta\mathbf{X} = \mathbf{E}$ has to be solved. The computational cost for solving this linear system directly by LU factorization is $O(n^3)$. In order to reduce computations, we explore the structure of J and take advantage of the system sparsity. This paper shows that by recognizing the recursive patterns and exploiting the sparsity of the system the multiplicative and additive computational costs for solving $J\Delta\mathbf{X} = \mathbf{E}$ are $O(n)$ and $O(n^2)$, respectively.

2 MATHEMATICAL MODEL

An n-body pendulum was modeled with graphs by Chou and was presented in [5]. Based on this graph-theoretic model, the mathematical model was derived as a set of DAE's and was written symbolically as

$$\boxed{\mathbf{E}(\mathbf{X}, \dot{\mathbf{X}}, t) = \mathbf{0}} \tag{1}$$

where the vector of unknown variables is

$$\mathbf{X} = \begin{bmatrix} \mathbf{R}_b^T & \mathbf{V}_b^T & \mathbf{P}_b^T & \mathbf{S}_b^T \end{bmatrix}^T \tag{2}$$

The vectors $\mathbf{R}_b$ and $\mathbf{V}_b$ are the collection of displacements and velocities of all the bodies, and the vectors $\mathbf{P}_b$ and $\mathbf{S}_b$ are the orientation parameters (we use Euler parameters here) and their derivatives. The superscripts "u" and "v" indicate the dependent set and the independent set, respectively. The equations include the following six sets of equations:

$$\mathbf{E} \equiv \begin{bmatrix} {}^r\mathbf{F}(\dot{\mathbf{V}}_b, \mathbf{P}_b, \mathbf{S}_b, \dot{\mathbf{S}}_b, t) \\ \hline {}^t\mathbf{G}(\mathbf{R}_b, \mathbf{P}_b, t) \\ {}^t\dot{\mathbf{G}}(\dot{\mathbf{V}}_b, \mathbf{P}_b, \mathbf{S}_b, t) \\ \hline {}^r\mathbf{I}^1(\mathbf{P}_b, t) \\ {}^r\mathbf{I}^2(\mathbf{P}_b, \mathbf{S}_b, t) \\ {}^r\mathbf{I}^3(\dot{\mathbf{P}}_b^v, \mathbf{S}_b^v, t) \end{bmatrix} = \mathbf{0} \tag{3}$$

and they are a total of $14n$ scalar equations in $14n$ unknown variables.

3 SOLVING THE FULL SYSTEM

At each solution point, we can solve the system equations $\mathbf{E}$ directly, using the implicit integration method [3,4] where the predicted solution is corrected to its exact solution within a given tolerance using Newton's iterative method. For each iteration, a liner system

$$\boxed{J\,\Delta\mathbf{X} = \mathbf{E}}$$

has to be solved by LU factorization. The matrix J is the system Jacobian matrix which is specified by the formula given by Petzold [4]. The vectors $\Delta\mathbf{X}$ and $\mathbf{E}$ are the vectors of corrections and residuals, respectively. Letting the Jacobian of an implicit function $\mathbf{F}$ with respect to the variable $\mathbf{V}$ be $\boldsymbol{F}_V$, we can write the Jacobian matrix J and the vectors $\Delta\mathbf{X}$ and $\mathbf{E}$ of the full system symbolically as follows:

$$\left[\begin{array}{ccccc|c} {}^t\boldsymbol{G}_R & 0 & 0 & 0 & {}^t\boldsymbol{G}_{P^u} & {}^t\boldsymbol{G}_{P^v} \\ 0 & {}^t\dot{\boldsymbol{G}}_V & {}^t\dot{\boldsymbol{G}}_{S^u} & {}^t\dot{\boldsymbol{G}}_{S^v} & {}^t\dot{\boldsymbol{G}}_{P^u} & {}^t\dot{\boldsymbol{G}}_{P^v} \\ 0 & 0 & {}^r\boldsymbol{I}^2_{S^u} & 0 & 0 & {}^r\boldsymbol{I}^2_{P^v} \\ 0 & 0 & 0 & {}^r\boldsymbol{I}^3_{S^v} & 0 & {}^r\boldsymbol{I}^3_{P^v} \\ 0 & 0 & 0 & 0 & {}^r\boldsymbol{I}^1_{P^u} & {}^r\boldsymbol{I}^1_{P^v} \\ \hline 0 & {}^r\boldsymbol{F}_V & {}^r\boldsymbol{F}_{S^u} & {}^r\boldsymbol{F}_{S^v} & {}^r\boldsymbol{F}_{P^u} & {}^r\boldsymbol{F}_{P^v} \end{array}\right] \left[\begin{array}{c} \Delta\mathbf{R}_b \\ \Delta\mathbf{V}_b \\ \Delta\mathbf{S}_b^u \\ \Delta\mathbf{S}_b^v \\ \Delta\mathbf{P}_b^u \\ \hline \Delta\mathbf{P}_b^v \end{array}\right] = \left[\begin{array}{c} {}^t\mathbf{G} \\ {}^t\dot{\mathbf{G}} \\ {}^r\mathbf{I}^2 \\ {}^r\mathbf{I}^3 \\ {}^r\mathbf{I}^1 \\ \hline {}^r\mathbf{F} \end{array}\right] \tag{4}$$

Solving the full system means that the system is solved by implicit integration in which the full linear system, $J\Delta\mathbf{X} = \mathbf{E}$, is solved by LU factorization without reducing its size. However, some structural information in J and $\mathbf{E}$ can be utilized to reduce computations when they are evaluated.

3.1 Recursiveness and Special Structures

Some equations in $\mathbf{E}$ possess recursive properties which can be further utilized to reduce computational cost. This recursive nature also causes the sub-matrices in J to have special structures which can be exploited to minimize the computational cost.

3.1.1 Translational Kinematic Constraints ${}^t\mathbf{G}$

In [5], the translational kinematic equations were derived as

$${}^t\mathbf{G}_i \equiv \mathbf{R}_{b_i} - \mathbf{R}_{o_{p_1}} - \boldsymbol{A}_i\mathbf{r}_i + \sum_{k=1}^{i} \boldsymbol{A}_k\mathbf{a}_k = \mathbf{0} \tag{5}$$

Let

$$\begin{cases} \boldsymbol{\lambda}_i &= \sum_{k=1}^{i} \boldsymbol{A}_k \mathbf{a}_k \\ \mathbf{x}_i &= \boldsymbol{A}_i \mathbf{a}_i \\ \boldsymbol{\lambda}_0 &= \mathbf{0} \end{cases}$$

then, (5) becomes ${}^t\mathbf{G}_i \equiv \mathbf{R}_{b_i} - \mathbf{R}_{sp_1} - \boldsymbol{A}_i \mathbf{r}_i + \boldsymbol{\lambda}_i = \mathbf{0}$, or recursively

$$ {}^t\mathbf{G}_i \equiv \mathbf{R}_{b_i} - \mathbf{R}_{sp_1} - \boldsymbol{A}_i \mathbf{r}_i + (\boldsymbol{\lambda}_{i-1} + \mathbf{x}_i) = \mathbf{0} \tag{6}$$

When we evaluate the residual vector ${}^t\mathbf{G}$, we only have to compute $\mathbf{R}_{b_i}$, $\mathbf{R}_{sp_1}$, $\boldsymbol{A}_i \mathbf{r}_i$, and $\mathbf{x}_i$ for $i = 1, ..., n$ once. Instead of computing $\boldsymbol{\lambda}_i$ for each equation, we only compute $\mathbf{x}_i$ and add the previous $\boldsymbol{\lambda}_{i-1}$; that is, $\boldsymbol{\lambda}_i = \boldsymbol{\lambda}_{i-1} + \mathbf{x}_i$.

The Jacobian entries corresponding to the equation ${}^t\mathbf{G}$ are ${}^t\boldsymbol{G}_R$ and ${}^t\boldsymbol{G}_P$. The Jacobian of ${}^t\mathbf{G}$ corresponding to $\mathbf{R}_b$ can be shown to be a unit matrix easily. Here we show that ${}^t\boldsymbol{G}_P$ has a special structure. It is written as

$${}^t\boldsymbol{G}_P \equiv \frac{\partial\, {}^t\mathbf{G}}{\partial\, \mathbf{P}_b} = \begin{bmatrix} \frac{\partial\, {}^t\mathbf{G}_1}{\partial\, \mathbf{p}_{b_1}} & \frac{\partial\, {}^t\mathbf{G}_1}{\partial\, \mathbf{p}_{b_2}} & \cdots & \frac{\partial\, {}^t\mathbf{G}_1}{\partial\, \mathbf{p}_{b_n}} \\ \frac{\partial\, {}^t\mathbf{G}_2}{\partial\, \mathbf{p}_{b_1}} & \frac{\partial\, {}^t\mathbf{G}_2}{\partial\, \mathbf{p}_{b_2}} & \cdots & \frac{\partial\, {}^t\mathbf{G}_2}{\partial\, \mathbf{p}_{b_n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\, {}^t\mathbf{G}_n}{\partial\, \mathbf{p}_{b_1}} & \frac{\partial\, {}^t\mathbf{G}_n}{\partial\, \mathbf{p}_{b_2}} & \cdots & \frac{\partial\, {}^t\mathbf{G}_n}{\partial\, \mathbf{p}_{b_n}} \end{bmatrix} \tag{7}$$

Since ${}^t\mathbf{G}_i$ is a function of $\mathbf{p}_{b_1}$, $\mathbf{p}_{b_2}$, ..., and $\mathbf{p}_{b_i}$, for each row i ($i = 1, ..., n$) we have

$$\frac{\partial\, {}^t\mathbf{G}_i}{\partial\, \mathbf{p}_{b_k}} = \mathbf{0};\ k = i+1, ..., n \tag{8}$$

The matrix in (7) becomes

$${}^t\boldsymbol{G}_P = \begin{bmatrix} \frac{\partial\, {}^t\mathbf{G}_1}{\partial\, \mathbf{p}_{b_1}} & 0 & 0 & \cdots & 0 \\ \frac{\partial\, {}^t\mathbf{G}_2}{\partial\, \mathbf{p}_{b_1}} & \frac{\partial\, {}^t\mathbf{G}_2}{\partial\, \mathbf{p}_{b_2}} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\, {}^t\mathbf{G}_n}{\partial\, \mathbf{p}_{b_1}} & \frac{\partial\, {}^t\mathbf{G}_n}{\partial\, \mathbf{p}_{b_2}} & \frac{\partial\, {}^t\mathbf{G}_n}{\partial\, \mathbf{p}_{b_3}} & \cdots & \frac{\partial\, {}^t\mathbf{G}_n}{\partial\, \mathbf{p}_{b_n}} \end{bmatrix} \tag{9}$$

and it is a blocked lower-triangular matrix. Each blocked entry is a 3×4 matrix. The entries in each column i ($i = 1, ..., n$) in (9) are derived, in Appendix D.1 in [5], as

$$\begin{cases} \frac{\partial\, {}^t\mathbf{G}_i}{\partial\, \mathbf{p}_{b_i}} &= 2\boldsymbol{E}_i\, \bar{\mathbf{a}}_i - 2\boldsymbol{E}_i\, \bar{\mathbf{r}}_i \\ \frac{\partial\, {}^t\mathbf{G}_k}{\partial\, \mathbf{p}_{b_i}} &= 2\boldsymbol{E}_i\, \bar{\mathbf{a}}_i;\ k = i+1, \ldots, n \end{cases} \tag{10}$$

Hence, (9) becomes

$${}^t\boldsymbol{G}_P = \begin{bmatrix} 2\boldsymbol{E}_1(\bar{\mathbf{a}}_1 - \bar{\mathbf{r}}_1) & 0 & 0 & \cdots & 0 \\ 2\boldsymbol{E}_1\, \bar{\mathbf{a}}_1 & 2\boldsymbol{E}_2(\bar{\mathbf{a}}_2 - \bar{\mathbf{r}}_2) & 0 & \cdots & 0 \\ 2\boldsymbol{E}_1\, \bar{\mathbf{a}}_1 & 2\boldsymbol{E}_2\, \bar{\mathbf{a}}_2 & 2\boldsymbol{E}_3(\bar{\mathbf{a}}_3 - \bar{\mathbf{r}}_3) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 2\boldsymbol{E}_1\, \bar{\mathbf{a}}_1 & 2\boldsymbol{E}_2\, \bar{\mathbf{a}}_2 & 2\boldsymbol{E}_3\, \bar{\mathbf{a}}_3 & \cdots & 2\boldsymbol{E}_n(\bar{\mathbf{a}}_n - \bar{\mathbf{r}}_n) \end{bmatrix} \tag{11}$$

To compute ${}^t\boldsymbol{G}_P$, we only have to evaluate $2\boldsymbol{E}_i\, \bar{\mathbf{a}}_i$, $2\boldsymbol{E}_i\, \bar{\mathbf{r}}_i$, and $2\boldsymbol{E}_i(\bar{\mathbf{a}}_i - \bar{\mathbf{r}}_i)$ once for $i = 1, \ldots, n$.

The matrix ${}^t\boldsymbol{G}_P$ can be further partitioned into ${}^t\boldsymbol{G}_{P^*}$ and ${}^t\boldsymbol{G}_{P^*}$ as in (4). This involves permuting selected columns; however, the special structure is retained for these matrices after they are partitioned.

3.1.2 Translational Velocity Kinematic Constraints ${}^t\dot{\mathbf{G}}$

Translational velocity constraints are written as

$$ {}^t\dot{\mathbf{G}}_i \equiv \mathbf{V}_{b_i} - \dot{\boldsymbol{A}}_i \mathbf{r}_i + \sum_{k=1}^{n} \dot{\boldsymbol{A}}_k \mathbf{a}_k = \mathbf{0} \tag{12} $$

Let

$$ \begin{cases} \dot{\boldsymbol{\lambda}}_i = \sum_{k=1}^{i} \dot{\boldsymbol{A}}_k \mathbf{a}_k \\ \dot{\mathbf{x}}_i = \dot{\boldsymbol{A}}_i \mathbf{a}_i \\ \dot{\boldsymbol{\lambda}}_0 = \mathbf{0} \end{cases} $$

then, (12) becomes ${}^t\dot{\mathbf{G}}_i \equiv \mathbf{V}_{b_i} - \dot{\boldsymbol{A}}_i \mathbf{r}_i + \dot{\boldsymbol{\lambda}}_i = \mathbf{0}$, or recursively

$$ {}^t\dot{\mathbf{G}}_i \equiv \mathbf{V}_{b_i} - \dot{\boldsymbol{A}}_i \mathbf{r}_i + (\dot{\boldsymbol{\lambda}}_{i-1} + \dot{\mathbf{x}}_i) = \mathbf{0} \tag{13} $$

Similar to (6), we only have to compute $\mathbf{V}_{b_i}$, $\dot{\boldsymbol{A}}_i \mathbf{r}_i$, and $\dot{\mathbf{x}}_i$ once. Instead of computing $\dot{\boldsymbol{\lambda}}_i$, we compute $\dot{\boldsymbol{\lambda}}_{i-1} + \dot{\mathbf{x}}_i$ recursively to save computations.

The Jacobian matrices correspond to ${}^t\dot{\mathbf{G}}$ are ${}^t\dot{\boldsymbol{G}}_V$, ${}^t\dot{\boldsymbol{G}}_S$, and ${}^t\dot{\boldsymbol{G}}_P$ in which ${}^t\dot{\boldsymbol{G}}_V$ can be shown to be a unit matrix easily, and ${}^t\dot{\boldsymbol{G}}_S$ and ${}^t\dot{\boldsymbol{G}}_P$ will be shown to possess special structures. The Jacobian matrix of ${}^t\dot{\mathbf{G}}$ with respect to $\mathbf{S}_b$ is defined as

$$ {}^t\dot{\boldsymbol{G}}_S \equiv \frac{\partial\, {}^t\dot{\mathbf{G}}}{\partial\, \mathbf{S}_b} = \begin{bmatrix} \frac{\partial\, {}^t\dot{\mathbf{G}}_1}{\partial\, \mathbf{s}_{b_1}} & \frac{\partial\, {}^t\dot{\mathbf{G}}_1}{\partial\, \mathbf{s}_{b_2}} & \cdots & \frac{\partial\, {}^t\dot{\mathbf{G}}_1}{\partial\, \mathbf{s}_{b_n}} \\ \frac{\partial\, {}^t\dot{\mathbf{G}}_2}{\partial\, \mathbf{s}_{b_1}} & \frac{\partial\, {}^t\dot{\mathbf{G}}_2}{\partial\, \mathbf{s}_{b_2}} & \cdots & \frac{\partial\, {}^t\dot{\mathbf{G}}_2}{\partial\, \mathbf{s}_{b_n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\, {}^t\dot{\mathbf{G}}_n}{\partial\, \mathbf{s}_{b_1}} & \frac{\partial\, {}^t\dot{\mathbf{G}}_n}{\partial\, \mathbf{s}_{b_2}} & \cdots & \frac{\partial\, {}^t\dot{\mathbf{G}}_n}{\partial\, \mathbf{s}_{b_n}} \end{bmatrix} \tag{14} $$

Since ${}^t\dot{\mathbf{G}}_i$ is a function of $\mathbf{s}_{b_1}$, $\mathbf{s}_{b_2}$, ..., and $\mathbf{s}_{b_i}$, for each row i $(i = 1, ..., n)$ we have

$$ \frac{\partial\, {}^t\dot{\mathbf{G}}_i}{\partial\, \mathbf{s}_{b_k}} = 0;\ k = i+1, ..., n \tag{15} $$

Hence, (14) becomes

$$ {}^t\dot{\boldsymbol{G}}_S = \begin{bmatrix} \frac{\partial\, {}^t\dot{\mathbf{G}}_1}{\partial\, \mathbf{s}_{b_1}} & 0 & 0 & \cdots & 0 \\ \frac{\partial\, {}^t\dot{\mathbf{G}}_2}{\partial\, \mathbf{s}_{b_1}} & \frac{\partial\, {}^t\dot{\mathbf{G}}_2}{\partial\, \mathbf{s}_{b_2}} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\, {}^t\dot{\mathbf{G}}_n}{\partial\, \mathbf{s}_{b_1}} & \frac{\partial\, {}^t\dot{\mathbf{G}}_n}{\partial\, \mathbf{s}_{b_2}} & \frac{\partial\, {}^t\dot{\mathbf{G}}_n}{\partial\, \mathbf{s}_{b_3}} & \cdots & \frac{\partial\, {}^t\dot{\mathbf{G}}_n}{\partial\, \mathbf{s}_{b_n}} \end{bmatrix} \tag{16} $$

The entries in each column i $(i = 1, ..., n)$ in (16) are derived, in Appendix D.1 in [5], as

$$ \begin{cases} \frac{\partial\, {}^t\dot{\mathbf{G}}_i}{\partial\, \mathbf{s}_{b_i}} = 2\boldsymbol{E}_i\, \bar{\mathbf{a}}_i - 2\boldsymbol{E}_i\, \bar{\mathbf{r}}_i \\ \frac{\partial\, {}^t\dot{\mathbf{G}}_k}{\partial\, \mathbf{s}_{b_i}} = 2\boldsymbol{E}_i\, \bar{\mathbf{a}}_i;\ k = i+1, \ldots, n \end{cases} \tag{17} $$

Comparing (17) with (10), we find that

$$ {}^t\dot{\boldsymbol{G}}_S \equiv \frac{\partial\, {}^t\dot{\mathbf{G}}}{\partial\, \mathbf{S}_b} = \frac{\partial\, {}^t\mathbf{G}}{\partial\, \mathbf{P}_b} \equiv {}^t\boldsymbol{G}_P \tag{18} $$

Computing ${}^t\dot{\boldsymbol{G}}_S$ is not required. Once we compute ${}^t\boldsymbol{G}_P$, we get ${}^t\dot{\boldsymbol{G}}_S$.

The Jacobian, ${}^t\dot{G}_P$, is different. Since ${}^t\dot{G}_i$ is a function of $\mathbf{p}_{b_1}$, $\mathbf{p}_{b_2}$, ..., and $\mathbf{p}_{b_i}$, ${}^t\dot{G}_P$ is a blocked lower-triangular matrix like ${}^t\dot{G}_S$ and is written as

$$ {}^t\dot{G}_P \equiv \frac{\partial\, {}^t\dot{\mathbf{G}}}{\partial\, \mathbf{P}_b} = \begin{bmatrix} \frac{\partial\, {}^t\dot{\mathbf{G}}_1}{\partial\, \mathbf{p}_{b_1}} & 0 & 0 & \cdots & 0 \\ \frac{\partial\, {}^t\dot{\mathbf{G}}_2}{\partial\, \mathbf{p}_{b_1}} & \frac{\partial\, {}^t\dot{\mathbf{G}}_2}{\partial\, \mathbf{p}_{b_2}} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\, {}^t\dot{\mathbf{G}}_n}{\partial\, \mathbf{p}_{b_1}} & \frac{\partial\, {}^t\dot{\mathbf{G}}_n}{\partial\, \mathbf{p}_{b_2}} & \frac{\partial\, {}^t\dot{\mathbf{G}}_n}{\partial\, \mathbf{p}_{b_3}} & \cdots & \frac{\partial\, {}^t\dot{\mathbf{G}}_n}{\partial\, \mathbf{p}_{b_n}} \end{bmatrix} \tag{19} $$

The entries in each column i ($i = 1, ..., n$) in (19) are derived, in Appendix D.1 in [5], as

$$ \begin{cases} \frac{\partial\, {}^t\dot{\mathbf{G}}_i}{\partial\, \mathbf{p}_{b_i}} = 2\dot{\boldsymbol{E}}_i\, \bar{\mathbf{a}}_i - 2\dot{\boldsymbol{E}}_i\, \bar{\mathbf{r}}_i \\ \frac{\partial\, {}^t\dot{\mathbf{G}}_k}{\partial\, \mathbf{p}_{b_i}} = 2\dot{\boldsymbol{E}}_i\, \bar{\mathbf{a}}_i;\ k = i+1, \ldots, n \end{cases} \tag{20} $$

Hence, (19) becomes

$$ {}^t\dot{G}_P = \begin{bmatrix} 2\dot{\boldsymbol{E}}_1(\bar{\mathbf{a}}_1 - \bar{\mathbf{r}}_1) & 0 & 0 & \cdots & 0 \\ 2\dot{\boldsymbol{E}}_1\, \bar{\mathbf{a}}_1 & 2\dot{\boldsymbol{E}}_2(\bar{\mathbf{a}}_2 - \bar{\mathbf{r}}_2) & 0 & \cdots & 0 \\ 2\dot{\boldsymbol{E}}_1\, \bar{\mathbf{a}}_1 & 2\dot{\boldsymbol{E}}_2\, \bar{\mathbf{a}}_2 & 2\dot{\boldsymbol{E}}_3(\bar{\mathbf{a}}_3 - \bar{\mathbf{r}}_3) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 2\dot{\boldsymbol{E}}_1\, \bar{\mathbf{a}}_1 & 2\dot{\boldsymbol{E}}_2\, \bar{\mathbf{a}}_2 & 2\dot{\boldsymbol{E}}_3\, \bar{\mathbf{a}}_3 & \cdots & 2\dot{\boldsymbol{E}}_n(\bar{\mathbf{a}}_n - \bar{\mathbf{r}}_n) \end{bmatrix} \tag{21} $$

To compute ${}^t\dot{G}_P$, we only have to evaluate $2\dot{\boldsymbol{E}}_i\, \bar{\mathbf{a}}_i$, $2\dot{\boldsymbol{E}}_i\, \bar{\mathbf{r}}_i$, and $2\dot{\boldsymbol{E}}_i(\bar{\mathbf{a}}_i - \bar{\mathbf{r}}_i)$ once for $i = 1, \ldots, n$.

The matrices ${}^t\dot{G}_S$ and ${}^t\dot{G}_P$ can be further partitioned into ${}^t\dot{G}_{S^u}$, ${}^t\dot{G}_{S^r}$, ${}^t\dot{G}_{P^u}$, and ${}^t\dot{G}_{P^r}$ as in (4). This involves permuting selected columns. However, after permuting columns the special structures are retained for these partitioned matrices.

3.1.3 Torque-Balance Equations ${}^r\mathbf{F}$

The torque-balance equations for an n-body pendulum were derived as

$$ {}^r\mathbf{F}_i \equiv \mathbf{T}^i_{b_i} - \tilde{\mathbf{r}}_i \boldsymbol{A}_i^T M_i(\dot{\mathbf{V}}_{b_i} - \mathbf{g}) + \tilde{\mathbf{a}}_i \boldsymbol{A}_i^T \sum_{k=i}^{n} M_k(\dot{\mathbf{V}}_{b_k} - \mathbf{g}) = \mathbf{0} \tag{22} $$

Let

$$ \begin{cases} \boldsymbol{\tau}_i = \sum_{k=i}^{n} M_k(\dot{\mathbf{V}}_{b_k} - \mathbf{g}) \\ \mathbf{z}_i = M_i(\dot{\mathbf{V}}_{b_i} - \mathbf{g}) \\ \boldsymbol{\tau}_{n+1} = \mathbf{0} \end{cases} $$

then, we can write (22) as ${}^r\mathbf{F}_i \equiv \mathbf{T}^i_{b_i} - \tilde{\mathbf{r}}_i \boldsymbol{A}_i^T \mathbf{z}_i + \tilde{\mathbf{a}}_i \boldsymbol{A}_i^T \boldsymbol{\tau}_i = \mathbf{0}$, or recursively

$$ {}^r\mathbf{F}_i \equiv \mathbf{T}^i_{b_i} - \tilde{\mathbf{r}}_i \boldsymbol{A}_i^T \mathbf{z}_i + \tilde{\mathbf{a}}_i \boldsymbol{A}_i^T (\mathbf{z}_i + \boldsymbol{\tau}_{i+1}) = \mathbf{0} \tag{23} $$

When we compute the residual of ${}^r\mathbf{F}$, we only have to evaluate $\mathbf{T}^i_{b_i}$, $\mathbf{z}_i$, $\tilde{\mathbf{r}}_i \boldsymbol{A}_i^T \mathbf{z}_i$, and $\tilde{\mathbf{a}}_i \boldsymbol{A}_i^T \boldsymbol{\tau}_i$ once for $i = 1, ..., n$. Instead of computing $\boldsymbol{\tau}_i$ for each equation, we compute $\mathbf{z}_i$ and accumulate it recursively to save computations. The recursive term, $\mathbf{z}_i + \boldsymbol{\tau}_{i+1}$, runs in backward sequence; that is, $i = n, ..., 1$.

The Jacobian of ${}^r\mathbf{F}$ has three parts: ${}^r\boldsymbol{F}_V$, ${}^r\boldsymbol{F}_S$, and ${}^r\boldsymbol{F}_P$. They also possess special structures. First, the Jacobian of ${}^r\mathbf{F}$ with respect to $\mathbf{V}_b$ is written as

$$ {}^r\boldsymbol{F}_V \equiv \frac{\partial\, {}^r\mathbf{F}}{\partial\, \mathbf{V}_b} + \sigma \frac{\partial\, {}^r\mathbf{F}}{\partial\, \dot{\mathbf{V}}_b} = \sigma \frac{\partial\, {}^r\mathbf{F}}{\partial\, \dot{\mathbf{V}}_b} = \sigma \begin{bmatrix} \frac{\partial\, {}^r\mathbf{F}_1}{\partial\, \dot{\mathbf{V}}_{b_1}} & \frac{\partial\, {}^r\mathbf{F}_1}{\partial\, \dot{\mathbf{V}}_{b_2}} & \cdots & \frac{\partial\, {}^r\mathbf{F}_1}{\partial\, \dot{\mathbf{V}}_{b_n}} \\ \frac{\partial\, {}^r\mathbf{F}_2}{\partial\, \dot{\mathbf{V}}_{b_1}} & \frac{\partial\, {}^r\mathbf{F}_2}{\partial\, \dot{\mathbf{V}}_{b_2}} & \cdots & \frac{\partial\, {}^r\mathbf{F}_2}{\partial\, \dot{\mathbf{V}}_{b_n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\, {}^r\mathbf{F}_n}{\partial\, \dot{\mathbf{V}}_{b_1}} & \frac{\partial\, {}^r\mathbf{F}_n}{\partial\, \dot{\mathbf{V}}_{b_2}} & \cdots & \frac{\partial\, {}^r\mathbf{F}_n}{\partial\, \dot{\mathbf{V}}_{b_n}} \end{bmatrix} \tag{24} $$

Since ${}^r\mathbf{F}_i$ is a function of $\dot{\mathbf{V}}_{b_i}$, $\dot{\mathbf{V}}_{b_{i+1}}$, ..., and $\dot{\mathbf{V}}_{b_n}$, for each column i $(i = 1, ..., n)$ we have

$$\sigma \frac{\partial {}^r\mathbf{F}_k}{\partial \dot{\mathbf{V}}_{b_i}} = 0; \; k = i+1, ..., n \tag{25}$$

The matrix (24) becomes a blocked upper-triangular matrix and is written as

$${}^r\boldsymbol{F}_V = \sigma \begin{bmatrix} \frac{\partial {}^r\mathbf{F}_1}{\partial \dot{\mathbf{V}}_{b_1}} & \frac{\partial {}^r\mathbf{F}_1}{\partial \dot{\mathbf{V}}_{b_2}} & \cdots & \frac{\partial {}^r\mathbf{F}_1}{\partial \dot{\mathbf{V}}_{b_n}} \\ 0 & \frac{\partial {}^r\mathbf{F}_2}{\partial \dot{\mathbf{V}}_{b_2}} & \cdots & \frac{\partial {}^r\mathbf{F}_2}{\partial \dot{\mathbf{V}}_{b_n}} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial {}^r\mathbf{F}_n}{\partial \dot{\mathbf{V}}_{b_n}} \end{bmatrix} \tag{26}$$

The entries of each row i $(i = 1, ..., n)$ in ${}^r\boldsymbol{F}_V$ are derived as

$$\begin{cases} \sigma \frac{\partial {}^r\mathbf{F}_i}{\partial \dot{\mathbf{V}}_{b_i}} = -\sigma M_i \tilde{\mathbf{r}}_i A_i^T + \sigma M_i \tilde{\mathbf{a}}_i A_i^T \\ \sigma \frac{\partial {}^r\mathbf{F}_i}{\partial \dot{\mathbf{V}}_{b_k}} = \sigma M_k \tilde{\mathbf{a}}_i A_i^T; \; k = i+1, \; ..., \; n \end{cases} \tag{27}$$

Hence, (26) becomes

$${}^r\boldsymbol{F}_V = \sigma \begin{bmatrix} M_1(\tilde{\mathbf{a}}_1 - \tilde{\mathbf{r}}_1)A_1^T & M_2\tilde{\mathbf{a}}_1 A_1^T & M_3\tilde{\mathbf{a}}_1 A_1^T & \cdots & M_n\tilde{\mathbf{a}}_1 A_1^T \\ 0 & M_2(\tilde{\mathbf{a}}_2 - \tilde{\mathbf{r}}_2)A_2^T & M_3\tilde{\mathbf{a}}_2 A_2^T & \cdots & M_n\tilde{\mathbf{a}}_1 A_1^T \\ 0 & 0 & M_3(\tilde{\mathbf{a}}_3 - \tilde{\mathbf{r}}_3)A_3^T & \cdots & M_n\tilde{\mathbf{a}}_3 A_3^T \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & M_n(\tilde{\mathbf{a}}_n - \tilde{\mathbf{r}}_n)A_n^T \end{bmatrix} \tag{28}$$

To compute ${}^r\boldsymbol{F}_V$, we only have to evaluate $\sigma M_i(\tilde{\mathbf{a}}_i - \tilde{\mathbf{r}}_i)A_i^T$ and $\sigma M_k \tilde{\mathbf{a}}_i A_i^T$ once for $i = 1, ..., n$.

The Jacobian of ${}^r\mathbf{F}$ with respect to $\mathbf{P}_b$ is similar to (24). However, since ${}^r\mathbf{F}_i$ is a function of $\mathbf{p}_{b_i}$, the off-diagonal entries become zero:

$$\frac{\partial {}^r\mathbf{F}_k}{\partial \mathbf{p}_{b_i}} = \frac{\partial {}^r\mathbf{F}_i}{\partial \mathbf{p}_{b_k}} = 0; \; k = i+1, ..., n \tag{29}$$

for $i = 1, ..., n$. Hence, ${}^r\boldsymbol{F}_P$ becomes a diagonally blocked matrix and is written as

$${}^r\boldsymbol{F}_P \equiv \frac{\partial {}^r\mathbf{F}}{\partial \mathbf{P}_b} = \begin{bmatrix} \frac{\partial {}^r\mathbf{F}_1}{\partial \mathbf{p}_{b_1}} & 0 & \cdots & 0 \\ 0 & \frac{\partial {}^r\mathbf{F}_2}{\partial \mathbf{p}_{b_2}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial {}^r\mathbf{F}_n}{\partial \mathbf{p}_{b_n}} \end{bmatrix} \tag{30}$$

where

$$\begin{aligned} \frac{\partial {}^r\mathbf{F}_i}{\partial \mathbf{p}_{b_i}} &= \frac{\partial \mathbf{T}_{b_i}^i}{\partial \mathbf{p}_{b_i}} - \tilde{\mathbf{r}}_i \frac{\partial (A_i^T \mathbf{z}_i)}{\partial \mathbf{p}_{b_i}} + \tilde{\mathbf{a}}_i \frac{\partial (A_i^T \boldsymbol{\tau}_i)}{\partial \mathbf{p}_{b_i}} \\ &= \frac{\partial \mathbf{T}_{b_i}^i}{\partial \mathbf{p}_{b_i}} - 2\tilde{\mathbf{r}}_i G_i \overset{+}{\mathbf{z}}_i + 2\tilde{\mathbf{a}}_i G_i \overset{+}{\boldsymbol{\tau}}_i \\ &= \left\{ 2I_i \tilde{G}_i + 4G_i \dot{G}_i^T I_i \dot{G}_i - 4(\mathbf{s}_{b_i}^T \mathbf{p}_{b_i}) I_i \dot{G}_i - 2\tilde{\boldsymbol{\psi}}_i \dot{G}_i \right\} \\ &\quad - 2\tilde{\mathbf{r}}_i G_i \overset{+}{\mathbf{z}}_i + 2\tilde{\mathbf{a}}_i G_i \overset{+}{\boldsymbol{\tau}}_i \end{aligned} \tag{31}$$

where $\boldsymbol{\psi}_i \equiv I_i \boldsymbol{\omega}_i = 2I_i G_i \dot{\mathbf{p}}_{b_i}$ and $\tilde{\boldsymbol{\psi}}_i \equiv \boldsymbol{\psi} \times$ are defined in Appendix D.2 in [5].

${}^r\boldsymbol{F}_S$ has a structure identical to that in ${}^r\boldsymbol{F}_P$. The off-diagonal entries are zero:

$$\frac{\partial {}^r\mathbf{F}_k}{\partial \mathbf{s}_{b_i}} + \sigma \frac{\partial {}^r\mathbf{F}_k}{\partial \dot{\mathbf{s}}_{b_i}} = \frac{\partial {}^r\mathbf{F}_i}{\partial \mathbf{s}_{b_k}} + \sigma \frac{\partial {}^r\mathbf{F}_i}{\partial \dot{\mathbf{s}}_{b_k}} = \mathbf{0};\ k = i+1, ..., n \tag{32}$$

for $i = 1, ..., n$. The Jacobian ${}^r\boldsymbol{F}_S$ is diagonally blocked and is written as

$${}^r\boldsymbol{F}_S \equiv \frac{\partial {}^r\mathbf{F}}{\partial \mathbf{S}_b} + \sigma \frac{\partial {}^r\mathbf{F}}{\partial \dot{\mathbf{S}}_b} = \begin{bmatrix} \frac{\partial {}^r\mathbf{F}_1}{\partial \mathbf{s}_{b_1}} + \sigma \frac{\partial {}^r\mathbf{F}_1}{\partial \dot{\mathbf{s}}_{b_1}} & 0 & \cdots & 0 \\ 0 & \frac{\partial {}^r\mathbf{F}_2}{\partial \mathbf{s}_{b_2}} + \sigma \frac{\partial {}^r\mathbf{F}_2}{\partial \dot{\mathbf{s}}_{b_2}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial {}^r\mathbf{F}_n}{\partial \mathbf{s}_{b_n}} + \sigma \frac{\partial {}^r\mathbf{F}_n}{\partial \dot{\mathbf{s}}_{b_n}} \end{bmatrix} \tag{33}$$

where

$$\begin{aligned} \frac{\partial {}^r\mathbf{F}_i}{\partial \mathbf{s}_{b_i}} + \sigma \frac{\partial {}^r\mathbf{F}_i}{\partial \dot{\mathbf{s}}_{b_i}} &= \frac{\partial {}^r\mathbf{T}_{b_i}}{\partial \mathbf{s}_{b_i}} + \sigma \frac{\partial {}^r\mathbf{T}_{b_i}}{\partial \dot{\mathbf{s}}_{b_i}} \\ &= \left\{ 4\dot{G}_i G_i^T I_i G_i - 4(\mathbf{s}_{b_i}^T \mathbf{p}_{b_i}) I_i G_i + 2\tilde{\boldsymbol{\psi}}_i G_i \right\} - \sigma(2 I_i G_i) \end{aligned} \tag{34}$$

The derivation of (31) and (34) is given in Appendix D.2 in [5].

3.1.4 Equations Concerning Euler Parameters: ${}^r\mathbf{I}^1$, ${}^r\mathbf{I}^2$, and ${}^r\mathbf{I}^3$

When we compute the residuals of ${}^r\mathbf{I}^1$, ${}^r\mathbf{I}^2$, and ${}^r\mathbf{I}^3$, there is no applicable recursive structure that can be utilized. However, their Jacobians do have some special structures because these equations are derived for each body. Matrices with diagonally blocked structures are to be expected.

First, the normality constraint for each set of Euler parameters is written as

$${}^r I_i^1 \equiv \mathbf{p}_{b_i}^T \mathbf{p}_{b_i} - 1 = 0 \tag{35}$$

Since ${}^r I_i^1$ is a function of $\mathbf{p}_{b_i}$ only, we have

$$\frac{\partial {}^r I_i^1}{\partial \mathbf{p}_{b_k}} = \frac{\partial {}^r I_k^1}{\partial \mathbf{p}_{b_i}} = \mathbf{0};\ k = i+1, ..., n \tag{36}$$

From (36), the Jacobian matrix of ${}^r\mathbf{I}^1$ with respect to $\mathbf{p}_{b_1}$, $\mathbf{p}_{b_2}$, ..., and $\mathbf{p}_{b_n}$ must be a diagonally blocked matrix. The same argument can be applied to ${}^r\mathbf{I}^2$ and ${}^r\mathbf{I}^3$ such that their Jacobians with respect to $\mathbf{P}_b$ and $\mathbf{S}_b$ are diagonally blocked.

Since the blocks concerning these three sets of equations are locally processed, as shown in [5], the Jacobian matrices ${}^r I_{p_v}^2$, ${}^r I_{p_v}^3$, and ${}^r I_{p_v}^1$ possess a similar structure which is illustrated as follows:

$$\begin{bmatrix} {}^r I_{p_v}^2 \\ \hline {}^r I_{p_v}^3 \\ \hline {}^r I_{p_v}^1 \end{bmatrix} = \left[\begin{array}{cccc} \epsilon_1 & 0 & \cdots & 0 \\ 0 & \epsilon_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \epsilon_n \\ \hline \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_n \\ \hline \nu_1 & 0 & \cdots & 0 \\ 0 & \nu_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \nu_n \end{array}\right] \tag{37}$$

where

$$\begin{cases} \epsilon_i = [d_i\ e_i\ f_i] \\ \sigma_i = \begin{bmatrix} -\sigma & 0 & 0 \\ 0 & -\sigma & 0 \\ 0 & 0 & -\sigma \end{bmatrix} \\ \nu_i = [a_i\ b_i\ c_i] \end{cases}$$

are defined in Appendix B.1 in [5].

4 SOLVING THE REDUCED SYSTEM

Solving the reduced system means to solve the linear system using the technique of dimension reduction. Partitioning the linear system, $J\Delta\mathbf{X} = \mathbf{E}$, according to the partition in (4) gives

$$\begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{X}_1 \\ \Delta\mathbf{X}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{E}_1 \\ \mathbf{E}_2 \end{bmatrix}$$

Since J_{11} is a unit upper-triangular matrix and is non-singular, we can obtain

$$\begin{bmatrix} J_{11} & J_{12} \\ 0 & J_R \end{bmatrix} \begin{bmatrix} \Delta\mathbf{X}_1 \\ \Delta\mathbf{X}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{E}_1 \\ \mathbf{E}_R \end{bmatrix}$$

For the independent corrections $\Delta\mathbf{X}_2$, we solve

$$\boxed{J_R\Delta\mathbf{X}_2 = \mathbf{E}_R} \tag{38}$$

by LU factorization where

$$\begin{cases} J_R = J_{22} - J_{21}J_{11}^{-1}J_{12} \\ \mathbf{E}_R = \mathbf{E}_2 - (J_{21}J_{11}^{-1})\mathbf{E}_1 \end{cases} \tag{39}$$

The dependent corrections $\Delta\mathbf{X}_1$ are obtained by backward substitutions:

$$\Delta\mathbf{X}_1 = J_{11}^{-1}(\mathbf{E}_1 - J_{12}\Delta\mathbf{X}_2) \tag{40}$$

4.1 Solving for Independent Corrections

The independent corrections $\Delta\mathbf{X}_2$ are obtained by solving the linear system $J_R\Delta\mathbf{X}_2 = \mathbf{E}_R$. The evaluation of J_R and $\mathbf{E}_R$ is accomplished by developing a set of formulae such that the sparsity of the linear system is completely utilized. From (4), we derive the following formulae for J_R and $\mathbf{E}_R$:

$$\begin{aligned} J_R = {}^{r}F_{P^{\nu}} &- ({}^{r}F_V)({}^{t}\dot{G}_{P^{\nu}}) - ({}^{r}\Psi_4)({}^{r}I^2_{P^{\nu}}) \\ &- ({}^{r}\Psi_5)({}^{r}I^3_{P^{\nu}}) - ({}^{r}\Psi_6)({}^{r}I^1_{P^{\nu}}) \end{aligned} \tag{41}$$

and

$$\mathbf{E}_R = {}^{r}\mathbf{F} - ({}^{r}F_V)\,{}^{t}\dot{\mathbf{G}} - ({}^{r}\Psi_4)\,{}^{r}\mathbf{I}^2 - ({}^{r}\Psi_5)\,{}^{r}\mathbf{I}^3 - ({}^{r}\Psi_6)\,{}^{r}\mathbf{I}^1 \tag{42}$$

where

$$\begin{cases} {}^{r}\Psi_4 = {}^{r}F_{S^{\nu}} - ({}^{r}F_V)({}^{t}\dot{G}_{S^{\nu}}) \\ {}^{r}\Psi_5 = {}^{r}F_{S^{\nu}} - ({}^{r}F_V)({}^{t}\dot{G}_{S^{\nu}}) \\ {}^{r}\Psi_6 = {}^{r}F_{P^{\nu}} - ({}^{r}F_V)({}^{t}\dot{G}_{P^{\nu}}) \end{cases} \tag{43}$$

When we compute J_R and $\mathbf{E}_R$ using the above formulae, several matrix multiplications will be performed. The special structures discussed in the previous section can be exploited to reduce computation cost.

4.2 Solving for Dependent Corrections

Once we obtain the independent corrections, $\Delta\mathbf{X}_2$, the dependent corrections, $\Delta\mathbf{X}_1$, can be computed by $J_{11}^{-1}(\mathbf{E}_1 - J_{12}\Delta\mathbf{X}_2)$. However, using this formula directly is not practical since J_{11} has a very simple structure with many zeros. In order to utilize the sparsity completely, a set of formulae are derived symbolically for the dependent corrections. They are as follows:

$$\Delta\mathbf{P}^{u}_{b} = {}^{r}\mathbf{I}^1 - ({}^{r}I^1_{P^{\nu}})\Delta\mathbf{P}^{\nu}_{b} \tag{44}$$

$$\Delta\mathbf{S}^{\nu}_{b} = {}^{r}\mathbf{I}^3 - ({}^{r}I^3_{P^{\nu}})\Delta\mathbf{P}^{\nu}_{b} \tag{45}$$

$$\Delta\mathbf{S}^{u}_{b} = {}^{r}\mathbf{I}^2 - ({}^{r}I^2_{P^{\nu}})\Delta\mathbf{P}^{\nu}_{b} \tag{46}$$

$$\begin{aligned} \Delta\mathbf{V}_b = {}^{t}\dot{\mathbf{G}} &- ({}^{t}\dot{G}_{S^{u}})\Delta\mathbf{S}^{u}_{b} - ({}^{t}\dot{G}_{S^{\nu}})\Delta\mathbf{S}^{\nu}_{b} \\ &- ({}^{t}\dot{G}_{P^{u}})\Delta\mathbf{P}^{u}_{b} - ({}^{t}\dot{G}_{P^{\nu}})\Delta\mathbf{P}^{\nu}_{b} \end{aligned} \tag{47}$$

$$\Delta\mathbf{R}_b = {}^{t}\mathbf{G} - ({}^{t}G_{P^{u}})\Delta\mathbf{P}^{u}_{b} - ({}^{t}G_{P^{\nu}})\Delta\mathbf{P}^{\nu}_{b} \tag{48}$$

Again, when we compute the dependent corrections using the above formulae, the special structures discussed in the previous section can be exploited.

5 LINEAR COMPUTATIONAL-COST SCHEME

Rewriting the full system (4) by

- combining $\mathbf{S}_b^u$ and $\mathbf{S}_b^v$ to form $\mathbf{S}$ in the original order
- combining $\mathbf{P}_b^u$ and $\mathbf{P}_b^v$ to form $\mathbf{P}$ in the original order
- combining ${}^r\mathbf{I}^1$ and ${}^r\mathbf{I}^3$ to form $\mathbf{I}$
- combining $-2\sigma\,{}^r\mathbf{I}^2$ (scaled by -2σ) and ${}^r\mathbf{F}$ to form $\mathbf{F}$
- re-arranging the columns and rows
- dropping the subscripts and superscripts for clarity and simplicity

gives

$$\begin{bmatrix} G_R & G_P & 0 & 0 \\ 0 & I_P & I_S & 0 \\ 0 & F_P & F_S & F_V \\ 0 & \dot{G}_P & \dot{G}_S & \dot{G}_V \end{bmatrix} \begin{bmatrix} \Delta\mathbf{R} \\ \Delta\mathbf{P} \\ \Delta\mathbf{S} \\ \Delta\mathbf{V} \end{bmatrix} = \begin{bmatrix} \mathbf{G} \\ \mathbf{I} \\ \mathbf{F} \\ \dot{\mathbf{G}} \end{bmatrix} \tag{49}$$

or

$$\begin{bmatrix} U & G_P & 0 & 0 \\ 0 & U & I_S & 0 \\ 0 & F_P & F_S & F_V \\ 0 & \dot{G}_P & \dot{G}_S & U \end{bmatrix} \begin{bmatrix} \Delta\mathbf{R} \\ \Delta\mathbf{P} \\ \Delta\mathbf{S} \\ \Delta\mathbf{V} \end{bmatrix} = \begin{bmatrix} \mathbf{G} \\ \mathbf{I} \\ \mathbf{F} \\ \dot{\mathbf{G}} \end{bmatrix} \tag{50}$$

where G_R and $\dot{G}_V$ were shown to be unit matrices. In order to make I_P a unit matrix, Gaussian elimination has to be applied to I_P, I_S, and $\mathbf{I}$ locally. An example for one body is given in Appendix B.2 in [5]. We intend to solve this linear system with a computational cost which is linearly proportional to the number of bodies in the system.

5.1 Basic Theory

Applying the technique of dimension reduction to (50) gives

$$\begin{bmatrix} U & G_P & 0 & 0 \\ 0 & U & I_S & 0 \\ 0 & 0 & J_F & 0 \\ 0 & 0 & 0 & J_R \end{bmatrix} \begin{bmatrix} \Delta\mathbf{R} \\ \Delta\mathbf{P} \\ \Delta\mathbf{S} \\ \Delta\mathbf{V} \end{bmatrix} = \begin{bmatrix} \mathbf{G} \\ \mathbf{I} \\ \mathbf{E}_F \\ \mathbf{E}_R \end{bmatrix} \tag{51}$$

where

$$\begin{cases} J_R = U - (\dot{G}_S - \dot{G}_P I_S)\, J_F^{-1} F_V \\ \mathbf{E}_R = \dot{\mathbf{G}} - (\dot{G}_P)\,\mathbf{I} - (\dot{G}_S - \dot{G}_P I_S)\, J_F^{-1}(\mathbf{F} - F_P\,\mathbf{I}) \end{cases} \tag{52}$$

and

$$\begin{cases} J_F = F_S - F_P I_S \\ \mathbf{E}_F = \mathbf{F} - (F_P)\,\mathbf{I} - (F_V)\,\Delta\mathbf{V} \end{cases} \tag{53}$$

Instead of solving (50) directly by LU factorisation, we solve

$$\boxed{J_R \Delta\mathbf{V} = \mathbf{E}_R} \tag{54}$$

to obtain $\Delta\mathbf{V}$; then we solve

$$\boxed{J_F \Delta\mathbf{S} = \mathbf{E}_F} \tag{55}$$

to obtain $\Delta\mathbf{S}$. The rest of corrections can be computed by

$$\begin{cases} \Delta\mathbf{P} = \mathbf{I} - (I_S)\,\Delta\mathbf{S} \\ \Delta\mathbf{R} = \mathbf{G} - (G_P)\,\Delta\mathbf{P} \end{cases} \tag{56}$$

5.2 Invertibility

In [6], we have shown that J_R is non-singular if J is non-singular. From (52), we have to find J_F in order to evaluate J_R. We would like to show that J_F is invertible when $\sigma \rightarrow \infty$.

The matrix J_F is diagonally blocked because it is computed by $F_S - F_P I_S$ where F_S, F_P, and I_S are all diagonally blocked matrices. Each block at the diagonal position in J_F is computed by its corresponding block in F_S, F_P, and I_S. Hence, we write

$$J_{F_i} = F_{S_i} - F_{P_i} I_{S_i} \tag{57}$$

for each diagonal block. From (B.12) in Appendix B given in [5], we find that I_{S_i} is a matrix scaled by $\frac{1}{\sigma}$. Hence, we may write

$$F_{P_i} I_{S_i} = \frac{1}{\sigma} \Upsilon_i$$

Also, from (34) we find that F_{S_i} may be written as

$$F_{S_i} = -\sigma(2 I_i G_i) + \Gamma_i$$

Hence, we can write

$$J_{F_i} = -\sigma(2 I_i G_i) + \Gamma_i - \frac{1}{\sigma} \Upsilon_i \tag{58}$$

If $\sigma \rightarrow \infty$, we have

$$J_{F_i} \approx -\sigma(2 I_i G_i) \tag{59}$$

However, we have added one more equation into the **F** block as mentioned previously. This increases the number of equations in each block in **F** from three to four. The equation is ${}^r\mathbf{I}_i^2$ scaled by -2σ; that is,

$$-2\sigma\, {}^r\mathbf{I}_i^2 \equiv -2\sigma\, \mathbf{p}_i^T \mathbf{s}_i = 0 \tag{60}$$

The Jacobian corresponding to **s** is

$$-2\sigma\, \mathbf{p}_i^T \tag{61}$$

Combining it with (59) gives

$$J_{F_i} \approx -\sigma(2 I_i \overset{+}{\mathbf{P}}_i^T) \tag{62}$$

Adding equation (60) into block **F** is equivalent to using Euler's equations in quaternion space [7]. We have proved that the coefficient matrix with respect to $\dot{\mathbf{s}}$ in Euler's equations for inertial torque in 4-space is a non-singular matrix [7]. This matrix is exactly the same as $2I_i(\overset{+}{\mathbf{P}}_i)^T$ in (62). Therefore, J_{F_i} as well as J_F are invertible if $\sigma \rightarrow \infty$.

5.3 Summary of Computational Cost

The detailed discussion of the linear-cost scheme was presented in [5]. In terms of multiplicative($\times$) and additive ($+$) operations, the computational cost required for solving the reduced linear system for the linear-cost scheme is summarized also in detail in [5] as Tables 2 to 6. Adding up the totals in the tables gives the grand total of operations required for the linear-cost scheme:

$$\begin{cases} \times: & 657n - 324 \\ +: & 15n^2 + 574n - 321 \end{cases} \tag{63}$$

This cost is not for obtaining one solution point; it is the cost of solving the reduced linear system for one Newton's iteration. The computational cost for reaching a solution point depends on output step size and the number of iterations.

6 IMPLEMENTATION AND RESULTS

A program, called LINPEN (LINear-cost scheme for simulating an n-body PENdulum), was coded to simulate a pendulum. The program is equipped with modules for reporting various physical quantities such as displacement, velocity, and acceleration and was run on a VAX 750 computer. The motion of a user-specified pendulum can be displayed on a GRINNELL display terminal.

Each body is drawn as a standardized 3-D polygon and is projected in perspective. on the terminal screen according to its simulated position and orientation. An example of simulating a 3-body pendulum for ten continuous positions is illustrated in Figure 1. The figure looks exactly the same as displayed on the display terminal.

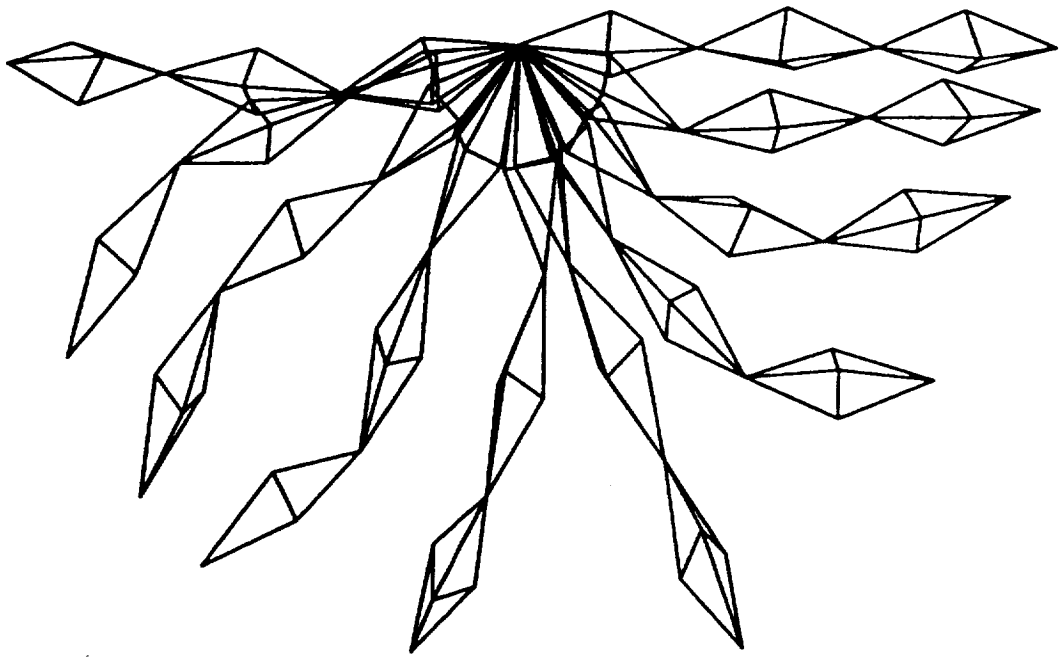


Figure 1: Simulation of a 3-body pendulum.

6.1 Comparative Study in Computational Cost

For each solution point, a linear system, $J\,\Delta\mathbf{X} = \mathbf{E}$, may have to be solved by LU factorization several times. There are four schemes available to accomplish this:

- solving the full linear system by LU factorization where J is approximated by numerical difference
- solving the full linear system by LU factorization where J is evaluated by the exact Jacobian matrix of $\mathbf{E}$
- solving the reduced linear system by LU factorization where J is evaluated by the exact Jacobian and is further reduced to its optimal size using the technique of dimension reduction
- solving the reduced linear system by LU factorization where J is evaluated by the exact Jacobian and is further reduced to its optimal size using the linear-cost scheme

For each iteration, the analytical count of operations required to solve the full system, reduced system, and the linear-cost system with exact Jacobian is tabulated in Table 1. Although the additive operations for the linear-cost system is $O(n^2)$, we may consider the cost linear because the amount of time required to perform a multiplication or division on a computer is about the same and is considerably greater than that required to perform an addition or subtraction.

The simulation of an n-body pendulum was run for the four systems. For each system, the program was run ten times from one body to ten bodies. One hundred solution points were generated, for each run, with an output time-step of 0.01 seconds and a tolerance of 10^{-7}. Total CPU time for solving four systems is calculated in terms of CPU time per residual-call or CPU time per Jacobian-call. Total CPU time per residual-call versus the number of bodies is plotted in Figure 2.

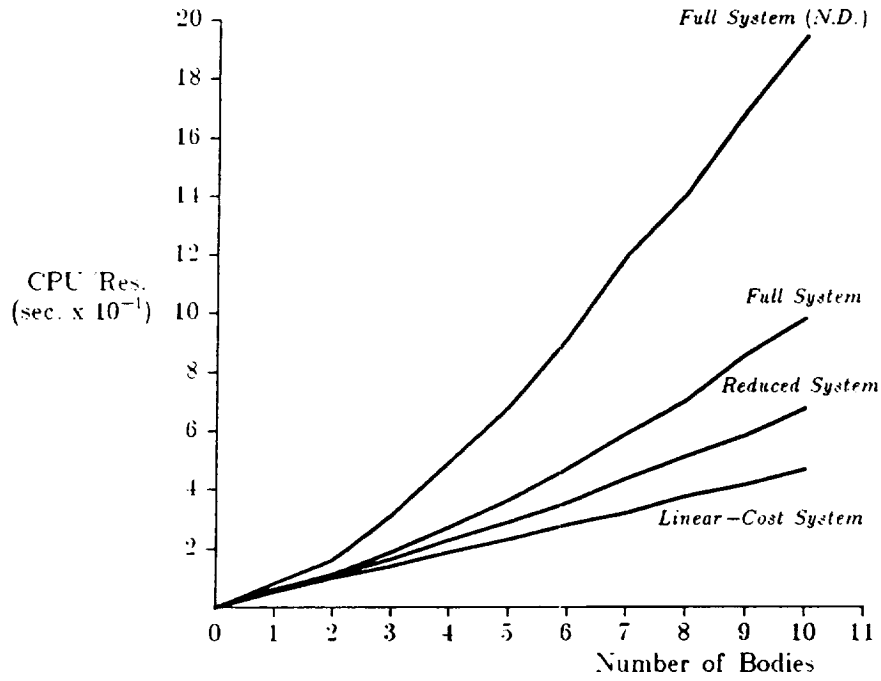


Figure 2: Computational cost in terms of CPU time for four systems.

An N-Body Pendulum Grand Total of Operations for Three Systems with Exact Jacobian Matrix		
System	×	+
Full System	$915n^3 + 196n^2 - 5n$	$915n^3 + 196n^2 - 19n$
Reduced System	$33n^3 + 129n^2 + 50n$	$25n^3 + 51n^2 + 25n$
Linear-Cost System	$657n - 324$	$15n^2 + 574n - 321$

Table 1: Analytical count of operations required for solving three systems with analytical Jacobian matrices.

7 REMARKS

The mathematical model of an n-body pendulum with spherical joints and its solution method have been presented. The mathematical model derived here is a mixed system of differential and algebraic equations (DAE's) in implicit form. A model of state-space equations can be derived if we do further complex substitutions. However, in the form of DAE's the equations of motion provide more sparsity, and this sparsity can be exploited when numerical solution methods are applied. The modeling and formulation of an n-body pendulum is the first study, for its similarity to a robotic manipulator in structure and for its simplicity in equations.

We also present four solution methods for solving the equations of motion of this n-body pendulum. The first method solves the linear system, $\boldsymbol{J}\,\Delta\mathbf{X} = \mathbf{E}$, directly by LU factorization where $\boldsymbol{J}$ is approximated by numerical difference. The second method solves the linear system directly by LU factorization, but $\boldsymbol{J}$ is evaluated by its analytical Jacobian matrix with some structural consideration.

The third method employs the technique of dimension reduction to transform the full system to its reduced system, $\boldsymbol{J}_R\,\Delta\mathbf{V} = \mathbf{E}_R$, such that LU factorization is performed on the reduced system only. This technique utilizes the sparsity of the system completely and saves a considerable number of computations. The four methods also employ the technique of dimension reduction to reduce the size of the system. However, they further exploit the structure of the system such that the reduced Jacobian generated possesses a very special structure which can be utilized to solve the system in a nearly-linear computational cost.

The comparative study in computational cost for these four systems in the paper gives strong evidence of the success of the theory developed.

References

[1] J. C. K. Chou, K. Singhal, and H. K. Kesavan. Multi-body systems with open chains: Graph-theoretic model. *Mechanism and Machine Theory*, 21(3):273–284, 1986.

[2] W. W. Armstrong. Recursive solution to the equations of motion of an n-link manipulator. *Proceeding of 5th World Congress on Theory of Machines and Mechanisms*, 2:1343–1346, July 1979.

[3] L. R. Petzold. Differential/algebraic equations are not ODE's. *SIAM J. Sci. Stat. Comput.*, 3(3):367–384, September 1982.

[4] L. R. Petzold. A description of DASSL: A differential/algebraic system solver. *Scientific Computing*, pages 65–68, North-Holland, 1983.

[5] Jack C. K. Chou. *Modeling, Formulation, and Solution Scheme for an N-Body Pendulum.* Technical Report No. TR-89008, Program in Engineering Science, Erik Jonsson School of Engineering and Computer Science, University of Texas at Dallas, Richardson, Texas, 1989.

[6] Jack C. K. Chou. *Computer-Aided Design Methods for Three-Dimensional Constrained Mechanical Systems.* Ph.D. Dissertation, Department of Systems Design Engineering, University of Waterloo, Waterloo, Ontario, Canada, 1988.

[7] Jack C. K. Chou. *Quaternions, Finite Rotation, and Dynamics.* Technical Report No. TR-89007, Program in Engineering Science, Erik Jonsson School of Engineering and Computer Science, University of Texas at Dallas, Richardson, Texas, 1989.

Sliding Control of Pointing and Tracking with Operator Spline Estimation*

Thomas A. W. Dwyer III† Fakhreddine Karray and Jinho Kim

3rd Annual Conference on Aerospace Computation Control

Abstract

It is shown in this paper how a variable structure control technique could be implemented to achieve precise pointing and good tracking of a deformable structure subject to fast slewing maneuvers. The correction torque that has to be applied to the structure is based on estimates of upper bounds on the model errors. For a rapid rotation of the deformable structure, the clastic response can be modeled by oscillators driven by angular acceleration, and where stiffness and damping coefficients are also angular velocity and acceleration dependent. By transforming this "slew-driven" elastic dynamics into bilinear form (by regarding the vector made up of the angular velocity, squared angular velocity and angular acceleration components, which appear in the coefficients as the input to the deformation dynamics), an operator spline can be constructed, that gives a low order estimate of the induced disturbance. Moreover, a "worst case" error bound between the estimated deformation and the unknown exact deformation is also generated, which can be used where required in the sliding control correction.

---

*Supported in part by SDIO/IST and managed by AFSOR under contract F49620-87-C-01013, and by NASA grant NAG-1-613

†Professor of Aeronautical and Astronautical Engineering, University of Illinois, Urbana, Il 61801

Graduate Research Assistants, Department of Aeronautical and Astronautical Engineering, University of Illinois, Urbana, Il 61801

A Survey on the Structured Singular Value

Andy Packard
Dept of Electrical and Computer Engineering
UC Santa Barbara
and
Michael Fan
Systems Research Center
University of Maryland

Abstract

The structured singular value, μ, is an important linear algebra tool to study a class of matrix perturbation problems. It is useful for analyzing the robustness of stability and performance of uncertain, (nominally) linear systems. Computation of (M) is difficult, and usually, upper and lower bounds are all that can be reliably computed. Upper bounds give conservative estimates of the sizes of allowable perturbations. The maximum singular value of a matrix M is an upper bound for (M). As an upper bound, it can be improved by finding a transformations to the data (ie. M) which do not change the structured singular value, but do reduce the maximum singular value. Typically, upper bound algorithms involve searches over sets of transformations to yield the tightest bound. Lower bound algorithms produce small, destabilizing perturbations. In general, the algorithms are intelligent searches for minimum-norm solutions to multivariable polynomial equations, and are based on various optimality conditions that hold at the global (and, unfortunately, some local) minima. This paper reviews the current methods to compute both of these types of bounds, covering theoretical justification and extensive numerical experience with the various algorithms.

Algorithms for Computing the Multivariable Stability Margin

Jonathan A. Tekawy[*], *Michael G. Safonov*[†] *and Richard Y. Chiang*[‡]

Abstract

Stability margin for multiloop flight control systems has become a critical issue, especially in highly maneuverable aircraft designs where there are inherent strong cross-couplings between the various feedback control loops. To cope with this issue,we have developed computer algorithms based on nondifferentiable optimization theory. These algorithms have been developed for computing the Multivariable Stability Margin (MSM). The MSM of a dynamical system is the "size" of the smallest structured perturbation in component dynamics that will destabilize the sytem. These algorithms have been coded and appear to be reliable. As illustrated by examples, they provide the basis for evaluating the robustness and performance of flight control systems.

1 Introduction

Accurate knowledge of the dynamical model associated with the design of modern flight control system is becoming more difficult to obtain. This is especially true for the design of the next generation fighters where many of the performance specifications go beyond the capability of the aircraft currently in service. Robust control analysis methods have received considerable attention in recent years as a possible solution to the problem of controlling systems for which the given model contains significant uncertainty [Saf 1] [Doyle2]. The central feature of these methods is their effectiveness in handling an unknown-but-bounded class of plants, instead of the nominal plant only.

[*]Flight Control Research, Aircraft Div., Northrop Corporation, Hawthorne, CA 90250
[†]E. E. - Systems Dept., University of Southern California, Los Angeles, CA 90089-0781
[‡]Flight Control Research, Aircraft Div., Northrop Corporation, Hawthorne, CA 90250

The uncertainty in the nominal plant arises from several different sources: For gain-scheduled aerospace vehicle control systems, typical uncertainties in the plant at each design point consist mainly of modeling errors due to uncertain aerodynamic coefficients, linearization, model reduction, neglected dynamics, time-delays, etc. Aerodynamic coefficients developed from wind tunnel testing or computational fluid dynamics usually are different from those obtained from actual flight data. Linearization will also affect the nominal plant behavior. Nonlinear effects such as actuator saturation and rate limits are neglected altogether when a model is linearized. Parameter drift will also affect the nominal plant. There may be dynamical modes which are intentionally or unknowingly neglected. In addition phase loss which results from time delay also leads to an uncertainty bound.

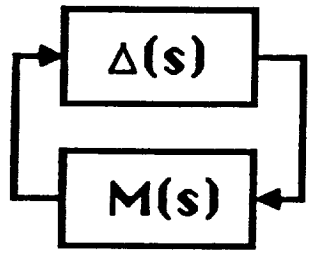
The uncertainty may be loosely classified as falling into two categories, structured and unstructured. Structured uncertainty arises from specific component or parameter variations. Two examples of structured uncertainty are variations in weight and drifting aerodynamic parameters. Unstructured uncertainty is any other sort of uncertainty which can be regarded as a frequency-dependent norm bounded perturbation matrix. High frequency modeling errors are one type of unstructured uncertainty. The linearization of the nonlinear equations of motion contribute to both classes of uncertainty. Actuator rate and position limits have a distinctive signature which can easily be isolated, so that they fall into the class of structured uncertainties. On the other hand, the effects of nonlinear kinematic terms can only be bounded, therefore necessitating an unstructured uncertainty representation.

2 Robustness Measure

Safonov [Saf 2], who built upon different, but related, conic-sector nonlinear stability theory work of Zames [Zames], reinterpreted the conic-sector stability concepts in order to deal with uncertainty and robustness issues. Safonov, Doyle and Fan, to name a few, have contributed to the continuing development in this area [Saf 3] [Saf 4] [Doyle2] [Fan].

Basically the robustness measure is done by lumping uncertain deviations from a nominal system $M(s)$ into an uncertain matrix $\Delta(s)$ resulting in an uncertain feedback system with loop transfer function $\Delta(s)M(s)$ as shown in figure 1.

Then, the Multivariable Stability Margin (MSM), "K_m", is defined as the smallest stable, norm-bounded perturbation $\Delta(s)$ that can destabilize the system. While K_m is in general difficult to compute, a reasonably tight lower bound $\underline{K}_m$ theoretically can be computed using diagonally scaled singular values [Saf 3] [Doyle2] [Fan]. The plot of $\underline{K}_m(w)$ vs. frequency identifies tolerable levels of parameter uncertainty as a function of frequency.



$$\Delta = diag[\Delta_1, \Delta_2, ..., \Delta_i, .., \Delta_n]$$

Figure 1: Robustness analysis model.

For unstructured uncertainty, the maximum singular value has been shown to be useful in bounding the multivariable stability margin. However, the bound can be very conservative in the case of structured uncertainty. The singular value analysis will attempt to find the worst direction of the uncertainty that in reality impossible to exist. To deal with the case of diagonally structured Δ, Safonov [Saf 4] introduced the two-sided structured Multivariable Stability Margin (MSM), denoted K_m, and Doyle [Doyle2] introduced the term Structured Singular Value (SSV), denoted μ, to describe the reciprocal, $\mu(M(s)) = 1/K_m(M(s))$. Diagonal perturbations are quite general and flexible if one considers parametric uncertainties (e.g. aerodynamic coefficients). Traditionally one defines K_m and μ for "two-sided" magnitude-bounded uncertainties which may be either positive or negative; but in cases where the sign of the uncertain Δ_i is known a priori one may modify the definitions of K_m and μ accordingly.

When the uncertainties are known to cover both positive and negative perturbations, the SSV of Doyle and MSM of Safonov provide a "tight" (to within 15%) condition for robust stability. This condition is measured by representing directly the individual sources of uncertainties in the form of block diagonal perturbations.

Definition: Given transfer functions $G(s), a(s)$ and $b(s)$, we write

$$G(s) \in sector[a, b]$$

if

$$|G(jw) - C(jw)| \leq |r(jw)| \qquad \forall w$$

where

$$C(s) = (a(s) + b(s))/2$$

and

$$r(s) = (a(s) - b(s))/2$$

Assuming $M(s)$ and $\Delta(s)$ to be stable then the one-sided MSM, "K_{m_1}" and, the two-sided MSM, "K_{m_2}" are defined by the following (see figure 2 and 3):

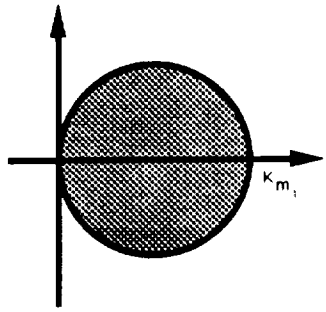


Figure 2: One-sided K_m

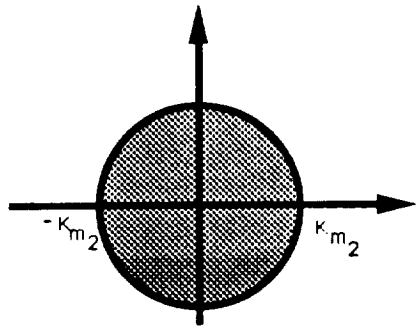


Figure 3: Two-sided K_m

- One-sided K_m :

  The system is stable for all Δ with

$$\Delta_i \in sector[0, K_{m_1}] \quad \forall i = 1, \cdots, n. \tag{1}$$

- Two-sided K_m :

  The system is stable for all Δ with

$$\Delta_i \in sector[-K_{m_2}, K_{m_2}] \quad \forall i = 1, \cdots, n. \tag{2}$$

For any diagonal matrix D, a practical upper bound on $\mu = 1/K_{m_2}$ is $\sigma_{max}(DMD^{-1})$. Further, it is known that for 3 or fewer Δ_i 's that the minimum over D of this upper bound is actually equal to μ [Doyle2]. Safonov and Doyle proved that the minimization problem of $\sigma^2_{max}(DMD^{-1})$ is convex in $D' = log(D)$, so that every local minimum is a global minimum. Furthermore, computational experience has shown that minimum of $\sigma_{max}(DMD^{-1})$ over D is within 15% of μ. So, we choose to work with this upper bound, $\underline{K}^{-1}_{m_2}$, to calculate the reciprocal of two-sided MSM. Practical upper bound for the reciprocal of one-sided MSM can be easily derived by using conic

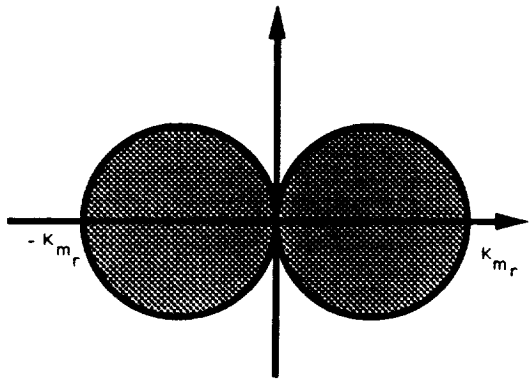


Figure 4: $\underline{K}_{m_r}$

sector property of Zames [Zames]; viz. $K_{m_1}^{-1}$ is bounded above by

$$\underline{K}_{m_1}^{-1} = max[\frac{1}{2} min_D \lambda_{max}(DMD^{-1} + (DMD^{-1})^*), 0] \tag{3}$$

By modifying the former equation to include the permutation matrix ϕ_i, a less conservative bound for the two-sided real MSM, K_{m_r}, is given by

$$\underline{K}_{m_r}^{-1} = max[\frac{1}{2} max_{\phi_i} min_D \lambda_{max}(DM\phi_i D^{-1} + (DM\phi_i D^{-1})^*), 0] \tag{4}$$

Here

$$\phi_i \in \Phi, \quad i = 1, \cdots, 2^n$$

and Φ is the set of all permutation $n \times$ n diagonal matrices.

$$\Phi = diag[(\pm 1), ..., (\pm 1)]$$

The bound (4) is similar to the one proposed by Jones [Jones]:

$$\frac{1}{2} min_D max_{\phi_i} \lambda_{max}(DM\phi_i D^{-1} + (DM\phi_i D^{-1})^*) \tag{5}$$

Although, equation (5) will lead to more conservative bounds. Geometrically equation (4) is shown in figure 4.

It should be mentioned at this point that several software packages are available to compute the two-sided MSM ,however , they are not accessible to the authors to be evaluated.

3 $\underline{K}_m$ Computation

The computation of σ_{max} is straight forward using available numerical software (Linpack). For both one-sided, real and two-sided cases, we have to solve an optimization

problem. However for all of these problems, the analytical gradient is available, so accurate solution can be obtained. However, when several eigenvalues or singular values coalesce (i.e., have multiplicity greater than one) the function is nondifferentiable ("creases" produce direction-dependent derivatives), so that a more complex algorithm of computing a descent direction is required. Before actually solving either case, one can approximately prescale the system matrix M by substituting for M the matrix DMD^{-1} where D minimizes the Frobenius norm of DMD^{-1} [Osborne].

The monotonic transformation of $D \rightarrow D'$, with $D = Exp(D')$, transforms the problem into a well behaved convex optimization [Saf 5]. The initial guess for D was taken to be equal to the identity. This initial guess was used only for the first frequency value in the given range. The solution obtained for a particular frequency point was then used as an initial guess for the next value. Suppose that the largest eigenvalue is simple, then a descent direction is calculated directly using the Davidon-Fletcher-Powell technique. In the case that the largest eigenvalue has multiplicity greater than 1 and the function is not continuously differentiable, a generalized gradient is used to determine a descent direction. Once this is done, the minimal point can be found in the specified descent direction by using a well known "binary search" algorithm of Bolzano. These steps are repeated until the global minimum is located (i.e. gradient is zero). Convexity of $\sigma^2_{max}(e^{D'}Me^{-D'})$ and $\lambda_{max}\frac{1}{2}(e^{D'}Me^{-D'} + (e^{D'}Me^{-D'})^*)$ ensures that this procedure is convergent to the global minimum.These steps can be summarized as follows:

1. Initialize $M_1 = M$; $D'_1 = 0$; $k = 1$.
2. Scale $M_{k+1} = e^{D'_k}M_k e^{-D'_k}$; set $D'_{k+1} = 0$
3. Find the search direction
   - Davidon Fletcher Powell (DFP) deflected gradient.
   - DFP generalized gradient for multiplicity ≥ 2
4. Unidirectional search.
   - Method of Bolzano (Fig. 5).
5. $D'_{k+1} \leftarrow D'_{k+1}$+ stepsize * search direction.
6. $k = k + 1$; go to step 2.

Step 5. of the algorithm involves varying the diagonal scaling matrix D' along a line by adjusting the scalar parameter stepsize. The size of $e^{D'_{k+1}}$ could approach the value of ∞. To prevent this, as the stepsize grows, M_{k+1}, is repeatedly updated to $M_{k+1} \leftarrow e^{D'_{k+1}}Me^{-D'_{k+1}}$ and stepsize and D'_{k+1} are reset to zero. This is done as often as needed to prevent numerical overflow when $e^{D'_{k+1}}$ is evaluated.

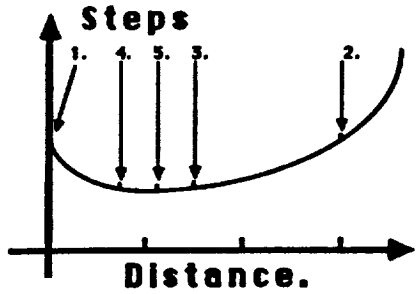


Figure 5: Method of Bolzano.

3.1 Generalized Gradient.

This discussion is not at all self contained and only key results will be stated. An excellent reference for this algorithm is [Polak]. In the case where the greatest eigenvalue/singular value has multiplicity greater than one, the function ceases to be differentiable. In this case the gradient is not defined and more complicated "generalized gradient" methods must be used to compute the descent direction. The generalized gradient at a nondifferentiable point is defined as the nearest point to the origin in the convex-hull of the set of directional derivatives at neighboring points; thus the computation of the generalized gradient at any point is itself a convex nonlinear programming problem. We employ an algorithm similar to that of [Doyle2, Polak1] to compute the generalized gradients of $\sigma^2_{max}(e^{D'}Me^{-D'})$ and $\lambda_{max}\frac{1}{2}(e^{D'}Me^{-D'} + (e^{D'}Me^{-D'})^*)$. Geometrically the algorithm is shown in figure 6 and summarized as follows:

- Generalized Gradient is defined as:

$$\nabla_{gen} \triangleq N_r(Co\{\nabla(x)|\ \ \|x\| = 1\})$$

  where

  Co(.) - the convex hull of the set (.).

  Nr(.) - the nearest point to the origin of the set (.).

  $\{\nabla(x)|\ \ \|x\| = 1\}$ - the set of directional derivatives.

- Iterative algorithm for computing ∇_{gen}:

  1. Initialize $k = 1$.
  2. Guess x_k and set $y_k = \nabla(x_k)$.

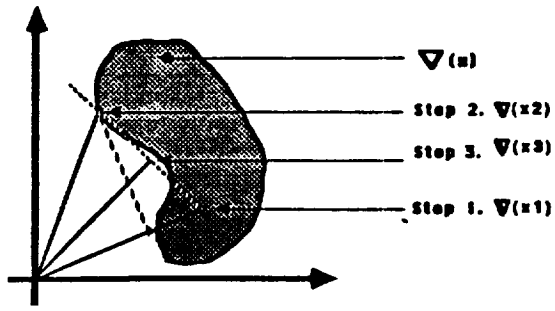


Figure 6: Generalized gradient.

3. Find x_{k+1} by minimizing $(y_k^* \nabla(x_{k+1}))$ subject to $\|x_{k+1}\| = 1$.
4. Find $y_{k+1} = \mathrm{Nr}(\ \mathrm{Co}(y_k, \nabla(x_{k+1}))$.
5. Increment $k \leftarrow k + 1$, go to step 3.

3.2 Davidon-Fletcher-Powell Scaling.

The unmodified generalized gradient determines a steepest descent direction. The steepest descent direction is simply minus the generalized gradient. Steepest descent usually works quite well during early stages of the optimization process but if the Hessian (second derivative) matrix has a large condition number, the method usually behaves poorly, and small zig-zagging steps, called "stitching", take place (see fig. 7). Stitching problems also occur when the multiplicity of σ_{max} or λ_{max} is 3 or more. Therefore we use the Davidon-Fletcher-Powell (DFP) method to modify the generalized gradient in order to handle this phenomenon. This technique uses the previously calculated generalized gradient to estimate the Hessian and effectively rescale the function to make its Hessian better conditioned. This quadratic fit method requires fewer gradient evaluations and tends to converge faster. It should also be noted that the likelihood of stitching-induced premature termination of the algorithm (as can occur in the unscaled steepest descent technique) can be greatly reduced with the DFP scaling.

4 Lateral Directional Flight Control Example

4.1 Example 1

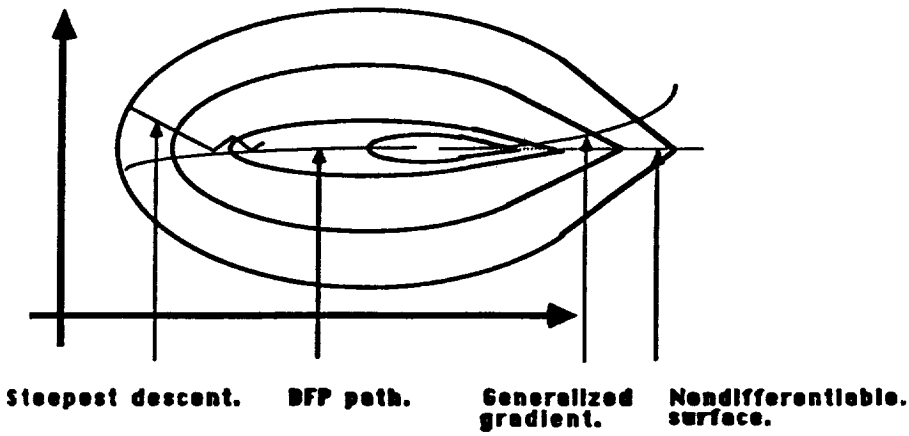


Figure 7: Minimization paths.

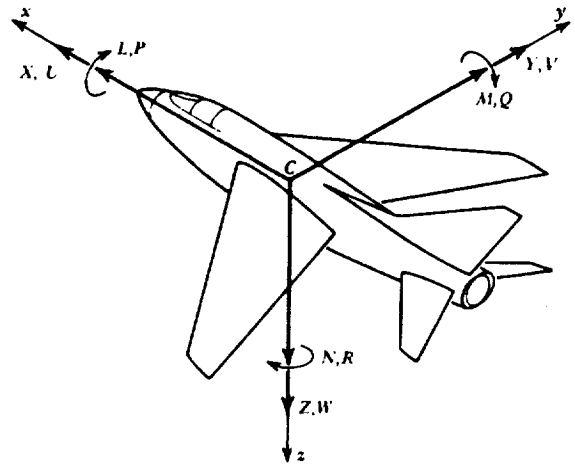


Figure 8: Axis systems and sign convention

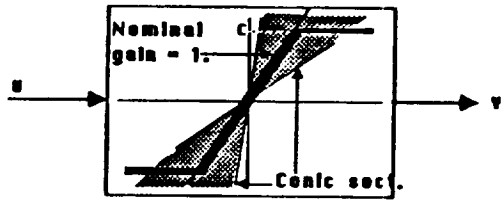


Figure 9: Two-sided actuator uncertainty model.

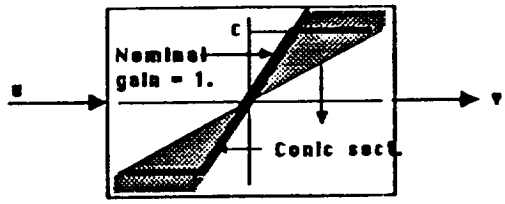


Figure 10: One-sided actuator uncertainty model.

Nonlinear elements of actuators can be treated as linear conic-sector elements with structured uncertainty. This uncertainty can be modeled as one-sided or two-sided uncertain gains within the actuator model. This can be shown through an example. Consider a saturation curve below. If the input size is always less than C, then the saturation element is equivalent to a gain element with a magnitude of one, however, if u exceeds C, one may model the saturation element as a two-sided uncertain gain Δ in parallel with a nominal gain of one as shown in figure 9. A better approach is to model the saturation element by a gain with a nominal value of one and a one-sided negative uncertainty as shown in figure 10. Clearly, the one-sided model will produce less conservative margins than the two-sided model.

A design example is presented below in which MSM algorithm is asked to check the robustness of a typical lateral/directional flight control systems with respect to the actuator uncertainty (e.g. position saturation) and the reduction in the effectiveness of all control surfaces. The state-space matrices are given in figure 11. The controller uses roll rate, P, yaw rate, R, and the lateral acceleration, N_y, for feedback (see figure 12). By putting "extender wires" on the uncertainty blocks Δ_i and pulling them out into a separate "block", one can check the system robustness.

The plot of $\underline{K}_{m_1}$, $\underline{K}_{m_2}$ and σ_{max} are in figure 13. Note that the σ_{max} and $\underline{K}_{m_2}$ which have the minimum values of .015 and .42 respectively are equal or less than $\underline{K}_{m_1}$ for all frequency, and therefore shown to be more conservative than one-sided

A	B	=	-3.4d-1	-4.8d-1	-2.8d+2	3.2d+1	0.0d+0	-9.2d-4	-6.8d+0
			5.2d-3	-3.0d+0	4.6d-1	-8.3d-6	0.0d+0	1.2d+1	-7.2d-2
			1.5d-2	-1.4d-1	-1.9d+0	-7.5d-7	0.0d+0	3.8d-1	3.2d+0
			0.0d+0	1.0d+0	0.0d+0	0.0d+0	0.0d+0	0.0d+0	0.0d+0
			0.0d+0	0.0d+0	1.0d+0	0.0d+0	0.0d+0	0.0d+0	0.0d+0
C	D		0.0d+0	5.7d+1	0.0d+0	0.0d+0	0.0d+0	0.0d+0	0.0d+0
			0.0d+0	0.0d+0	5.7d+1	0.0d+0	0.0d+0	0.0d+0	0.0d+0
			-6.3d-3	-3.6d-2	7.5d-1	5.1d-8	0.0d+0	6.2d-2	1.6d-1

Figure 11: State-space matrices.

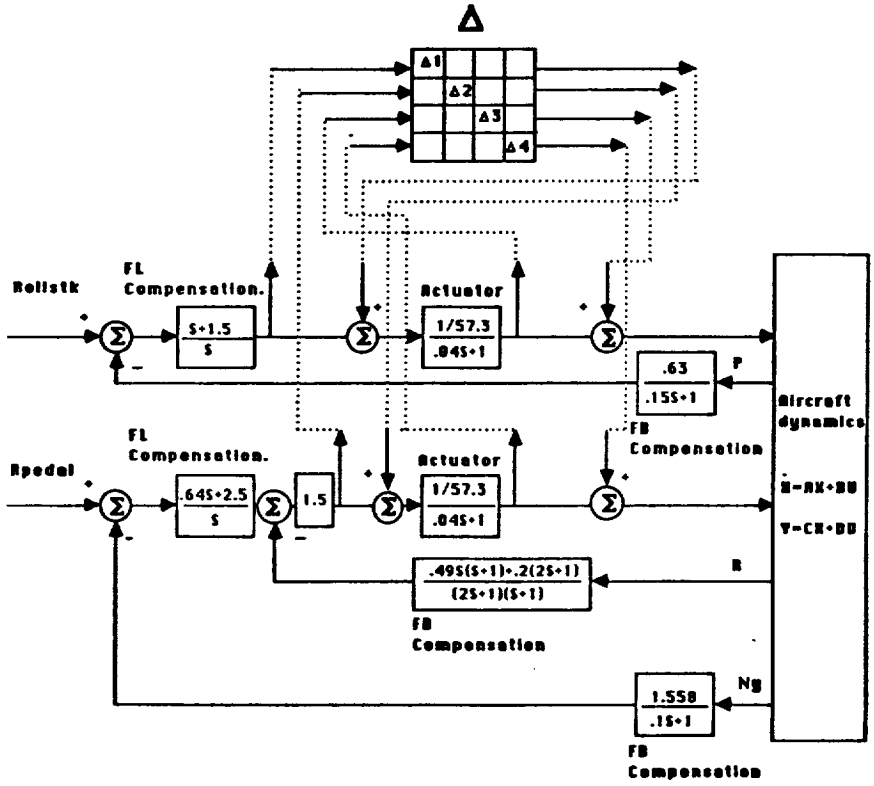


Figure 12: Lateral directional flight control with uncertainties at the controller output and plant input.

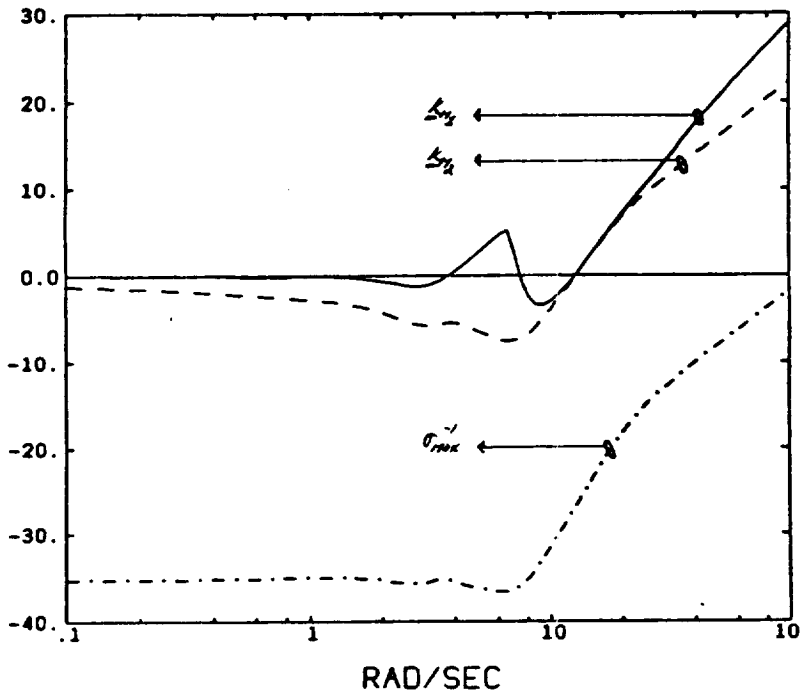


Figure 13: MSM at the plant input and controller output.

structured stability margin $\underline{K}_{m_1}$. This results from the one-sided structured multiplicative uncertainty that is not accounted for in the computation (i.e. nonlinear elements of actuators and gain reduction tolerance at the controller outputs). To properly account for the sign of the uncertainty and its structural information, the one-sided MSM was computed and it is shown to have a better robustness measure. $\underline{K}_{m_1}$ has the minimum value of 0.677, indicating that the system can simultaneously tolerate at least a 67.7 percent reduction in the effectiveness of all control surfaces and the actuator inputs up to at least three times the saturation value without instability.

4.2 Example 2

The MSM's minimal value $\underline{K}_{m_{peak}}$ also can be used to quantify a control systems tolerance of simultaneous gain and phase variations at all the plant inputs and outputs. This is done so the system has good stability robustness with respect to the uncertainties at the two actuator commands and the three sensor outputs (see figure 14). These uncertainties come from various sources. Model accuracy deteriorates at higher frequencies due to unmodeled aeroservoelastic effect. Several potential error sources exist within the assumed perfect sensors. Model reduction of the actuators can also be considered as one of the effects at the plant inputs.

Shown in figure 15 is the Bode plot of the MSM vs. frequency. The minimal value, denoted $\underline{K}_{m_{2_{peak}}} = .4196$, gives an indication of the minimal size of structured perturbations required to destabilize the system or equivalently diagonal perturbation

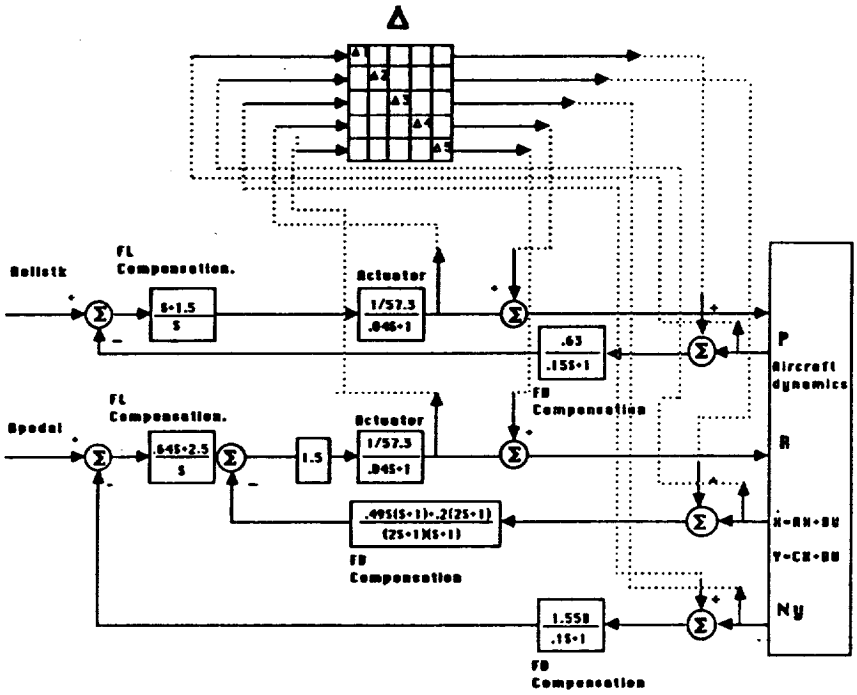


Figure 14: Lateral directional flight control with uncertainties at the plant input and output.

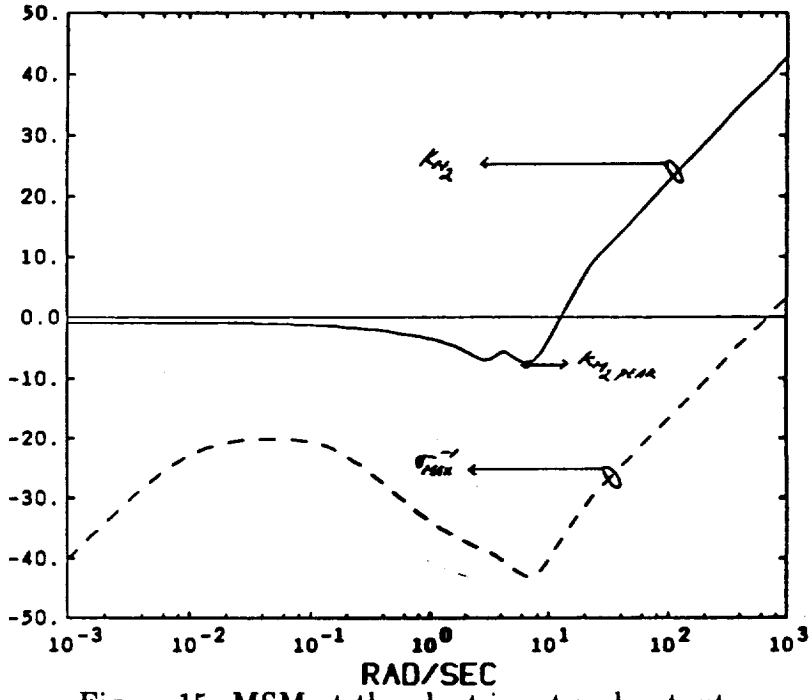


Figure 15: MSM at the plant input and output.

as large as 41.96 percent can be tolerated at any frequency; at higher frequencies, perturbation magnitude as large as $w/10$ can be tolerated.

5 Conclusion

Computer algorithms for determining the multivariable stability margin "K_m" have been developed. The algorithms provide a reliable tool for evaluating the robustness of control systems with significant gain and/or parameter uncertainties. The computation for the one-sided and two-sided structured stability margin were done using nondifferentiable optimization theory. Robustness analysis was performed on a typical lateral directional flight control system problem with large uncertainty.

References

[Doyle2] Doyle, J. C. (1982a). Analysis of feedback systems with structured uncertainties. *IEE Proc.*, **129**,part D, 242-250.

[Fan] Fan, M. K. and Tits, A. L. (1986). Characterization and Efficient Computation of the Structured Singular Value. *IEEE Trans. on Automatic Control*, **AC-31**, 734-743.

[Jones] Jones, R. D. (1987). Structured Singular Value Analysis for Real Parameter Variations. *1987 AIAA Conference in Guidance and Control.*

[Osborne] Osborne, E. E. (1960). On pre-conditioning of matrices. *JACM*, **vol. 7**, 338-345.

[Polak] Polak, E. and Mayne, D. Q. (1981). On the solution of singular value inequalities over a continuum of frequencies. *IEEE Trans. on Automatic Control*, **AC-26**, 690-694.

[Saf 1] Safonov, M. G., A. J. Laub and G. Hartmann (1981). Feedback properties of multivariable systems: The role and use of the return difference matrix. *IEEE Trans. on Automatic Control*, **AC-26**, 47-65.

[Saf 2] Safonov, M. G. (1980). *Stability and Robustness of Multivariable Feedback Systems.* MIT Press, Cambridge, MA.

[Saf 3] Safonov, M. G. (1982). Stability margins of diagonally perturbed multivariable feedback systems. *IEE Proc.*, **129**,part D, 251-256.

[Saf 4] Safonov, M. G. and M. Athans (1981). A multiloop generalization of the circle criterion for stability margin analysis. *IEEE Trans. on Automatic Control*, **AC-26**, 415-422.

[Saf 5] Safonov, M. G. and J. C. Doyle (1983). Optimal scaling for multivariable stability margin singular value computation. *Proc. MECO/EES'83 Symposium*, Athens, Greece.

[Zames] Zames, G. (1966). On the input-output stability of time-varying nonlinear feedback systems - Parts I and II. *IEEE Trans. on Automatic Control*, **AC-11**, 228-238 and 465-476, 1966.

Robustness Analysis for Real Parametric Uncertainty*

Athanasios Sideris
Dept. of Electrical Engineering
California Institute of Technology

Abstract

This paper has a twofold purpose. First, to review some key results in the literature in the area of robustness analysis for linear feedback systems with structured model uncertainty, and secondly to present some new results.

Model uncertainty is described as a combination of real uncertain parameters and norm bounded unmodeled dynamics. We will mainly focus on the case of parametric uncertainty. An elementary and unified derivation of the celebrated theorem of Kharitonov and the Edge Theorem will be presented. Next, an algorithmic approach for robustness analysis in the cases of multilinear and polynomic parametric uncertainty (i.e. the closed loop characteristic polynomial depends multilinearly and polynomially respectively on the parameters) is given. The latter cases are most important from practical considerations.

Some novel modifications in this algorithm which result in a procedure of polynomial time behavior in the number of uncertain parameters will be outlined. Finally, we show how the more general problem of robustness analysis for combined parametric and dynamic (i.e. unmodeled dynamics) uncertainty can be reduced to the case of polynomic parametric uncertainty, and thus be solved by means of our algorithm.

*To be presented at the 3rd Annual Conference on Aerospace Computational Control, Oxnard CA, August 1989.

Computational Issues in the Analysis of Adaptive Control Systems

Robert L. Kosut
Integrated Systems Inc.

Abstract

Adaptive systems under slow parameter adaptation can be analyzed by the method of averaging. This provides a means to assess stability (and instability) properties of most adaptive systems, either continuous-time or (more importantly for practice) discrete-time, as well as providing an estimate of the region of attraction. Although the method of averaging is conceptually straightforward, even simple examples are well beyond hand calculations. Specific software tools are proposed which can provide the basis for user-friendly environment to perform the necessary computations involved in the averaging analysis.

Experimental Experience with Flexible Structures

Gary J. Balas
California Institute of Technology

Abstract

This paper will focus on a flexible structure experiment developed at the California Institute of Technology. The main thrust of the experiment is to address the identification and robust control issues associated with large space structures by capturing their characteristics in the laboratory. The design, modeling, identification and control objectives will be discussed within this paper. Also, the subject of uncertainty in structural plant models and the frequency shaping of performance objectives will be expounded upon. Theoretical and experimental results of control laws designed using the identified model and uncertainty descriptions will be presented.

A DISTURBANCE BASED CONTROL/STRUCTURE DESIGN ALGORITHM

Mark D. McLaren [1]
Gary L. Slater [2]

Department of Aerospace Engineering and Engineering Mechanics
ML # 70, University of Cincinnati, Cincinnati, OH 45221

1 Introduction

In the past, the structure and its control system have been designed independently. Structural design and optimization, and control system design and optimization, have each been areas of separate research, each progressing vigorously along its own path. However, spurred on by recent proposals of new, large, highly constrained space structures, the question has arisen as to whether an integrated structural/control design procedure might not be more appropriate. The first papers actively investigating the question of simultaneous structure and control design began appearing in the literature around 1983 [1,2,3]. Since then, there has been a growing interest in this subject from other authors, although the field itself is still in relative infancy. Using a conventional design approach for a controlled structure, one would first optimize the structure alone, then design a control system for this baseline structure. This process may then be iterated until both the structure and control system meet necessary constraints and objectives.

Some authors ([4,5,6,7], for example) take a "classical" approach to the simultaneous structure/control optimization by attempting to simultaneously minimize the *weighted* sum of the total mass and a quadratic form, subject to all of the structural and control constraints. In this paper, the optimization will be based on the dynamic response of a structure to an external unknown stochastic disturbance environment [8]. Such a "response to excitation approach" is common to both the structural and control design phases, and hence represents a more natural control/structure optimization strategy than relying on artificial and vague control penalties. The design objective is to find the structure and controller of minimum mass such that all the prescribed constraints are satisfied.

Two alternative solution algorithms will be presented which have been applied to this problem. Each algorithm handles the optimization strategy and the imposition of the nonlinear constraints in a different manner. Two controller methodologies, and their effect on the solution algorithm, will be considered. These are full state feedback and direct output feedback, although the problem formulation is not restricted solely to these forms of controller. In fact, although full state feedback is a popular choice among researchers in this field (for resons that will become apparent), its practical application is severely limited. The controller/structure interaction is inserted by the imposition of appropriate closed-loop constraints, such as closed-loop output response and control effort constraints. Numerical results will be obtained for a representative flexible structure model to illustrate the effectiveness of the solution algorithms.

2 General Problem Formulation

The integrated control/structure design optimization problem can be stated as follows: find the vector of structural and controller parameters that minimizes the mass of the structure subject to a set of prescribed stochastic disturbances, with limitations on the available control energy and on a set of allowable output responses. This can be written in the form of a nonlinear mathematical programming problem as

[1] Graduate Student
[2] Professor

Minimize, with respect to $\boldsymbol{p}$, the weight $J(\boldsymbol{p})$, subject to

$$\boldsymbol{g}(\boldsymbol{p}) \leq 0$$

$$\dot{\boldsymbol{x}} = F\boldsymbol{x} + G\boldsymbol{u} + G_w\boldsymbol{w}$$

$$g_{cc_i} = \frac{\mathrm{E}[\boldsymbol{u}_i^T R_i \boldsymbol{u}_i]}{\beta_i^2} - 1 \leq 0 \qquad \text{for } i = 1, \ldots, n_\beta \tag{1}$$

$$g_{oc_i} = \frac{\mathrm{E}[\boldsymbol{y}_{d_i}^T W_i \boldsymbol{y}_{d_i}]}{\alpha_i^2} - 1 \leq 0 \qquad \text{for } i = 1, \ldots, n_\alpha$$

$$\boldsymbol{y}_{d_i} = H_{d_i}\boldsymbol{x} \qquad \text{for } i = 1, \ldots, n_\alpha$$

$$\boldsymbol{p}_\ell \leq \boldsymbol{p} \leq \boldsymbol{p}_u$$

where

$\boldsymbol{p}$	is an N-vector of design variables,
$\boldsymbol{g}$	is an m-vector of structural constraints,
$\boldsymbol{x}$	is an n-vector of state variables,
$\boldsymbol{u}$	is an n_u-vector of control forces,
$\boldsymbol{w}$	is an n_w-vector of stochastic disturbances,
F	is the $(n \times n)$ matrix containing the system dynamics,
G	is the $(n \times n_u)$ matrix containing information on the locations and orientations of the actuators,
G_w	is the $(n \times n_w)$ matrix containing information on the points of application and orientation of the disturbances,
g_{cc_i}	is the i^{th} control effort constraint cost function,
$\boldsymbol{u}_i$	is the n_{u_i}-order partition of $\boldsymbol{u}$ representing the control forces involved in g_{cc_i},
β_i	is the maximum allowable value of the i^{th} expected control effort function $\mathrm{E}[\boldsymbol{u}_i^T R_i \boldsymbol{u}_i]$,
R_i	is an $(n_{u_i} \times n_{u_i})$ control force weighting matrix,
g_{oc_i}	is the i^{th} output response constraint cost function,
α_i	is the maximum allowable value of the i^{th} expected output response function $\mathrm{E}[\boldsymbol{y}_{d_i}^T W_i \boldsymbol{y}_{d_i}]$,
W_i	is an $(n_{d_i} \times n_{d_i})$ output response weighting matrix,
$\boldsymbol{y}_{d_i}$	is the i^{th} design output n_{d_i}-vector,
H_{d_i}	is an $(n_{d_i} \times n)$ matrix giving the relationship between the state variables and $\boldsymbol{y}_{d_i}$,
$\boldsymbol{p}_\ell$	is an N-vector of minimum design variable values, and
$\boldsymbol{p}_u$	is an N-vector of maximum design variable values.

The side constraints are the strict bounds $\boldsymbol{p}_\ell$ and $\boldsymbol{p}_u$ on the design variables, and are vector inequalities that are imposed element by element. These design variable bounds are not included explicitly as constraints in the problem formulation. Note that the structural weight and the structural constraints $\boldsymbol{g}$ will in general not be functions of the controller design variables unless the controller mass is included in the design. Note also that g_{cc_i} is a weighted mean square control effort, and g_{oc_i} is a weighted mean square output response. Multiple output response constraints are allowed, although only one of these will in general be active at the optimum design. However, all of the control effort constraints will generally be active at the optimum design.

In this work $\boldsymbol{w}$ is a zero mean Gaussian white noise disturbance with covariance X_w. The structure will respond to this disturbance with some transient behaviour, in addition to a steady-state response. It seems

reasonable to optimize the structure for the steady-state response to the disturbance rather than the transient response because the transient behaviour will normally be of secondary importance to the response objectives (such as long term pointing accuracy). Also, for steady state optimization, the differential equation constraint (state equation) can be replaced with a steady state covariance equation, so that the control effort and output response constraints may be recast in terms of this covariance. Therefore the two-point boundary value problem is eliminated and the numerical solution of the problem is significantly simplified.

2.1 Full State Feedback Control

The simplest form of feedback control is to feedback the entire state vector, with $\boldsymbol{u} = -K\boldsymbol{x}$, where K is the $(n_u \times n)$ state feedback gain matrix. The controller design variables for this case will be the $n_u n$ elements of K. Substituting this control into the state equation, and assuming that the disturbance $\boldsymbol{w}$ is zero mean Gaussian white noise, the state covariance matrix X for this case can be found from the Lyapunov equation

$$F_{cl}X + XF_{cl}^T + G_w X_w G_w^T = 0 \tag{2}$$

where $F_{cl} = (F - GK)$ is the *stable* closed-loop dynamical matrix for the full state feedback case, $X = \mathrm{E}[\boldsymbol{x}\boldsymbol{x}^T]$ is the $(n \times n)$ symmetric state covariance matrix, and $X_w = \mathrm{E}[\boldsymbol{w}\boldsymbol{w}^T]$ is the $(n_w \times n_w)$ symmetric covariance matrix for the stochastic disturbances.

Expressions for the controller constraints in terms of this covariance matrix can then be obtained as

$$g_{cc_i} = \frac{\mathrm{tr}[K_i^T R_i K_i X]}{\beta_i^2} - 1 \qquad \text{for } i = 1, \ldots, n_\beta \tag{3}$$

$$g_{oc_i} = \frac{\mathrm{tr}[H_{d_i}^T W_i H_{d_i} X]}{\alpha_i^2} - 1 \qquad \text{for } i = 1, \ldots, n_\alpha \tag{4}$$

where K_i is the $(n_{u_i} \times n)$ partition of K corresponding to $\boldsymbol{u}_i$. It is assumed that the $\boldsymbol{u}_i$ are independent, and that $\boldsymbol{u}$ and K are ordered as

$$\boldsymbol{u}^T = \begin{bmatrix} \boldsymbol{u}_1^T & \boldsymbol{u}_2^T & \cdots & \boldsymbol{u}_{n_\beta}^T \end{bmatrix}, \qquad K^T = \begin{bmatrix} K_1^T & K_2^T & \cdots & K_{n_\beta}^T \end{bmatrix} \tag{5}$$

Note that $\sum_{i=1}^{n_\beta} n_{u_i} = n_u$, and that the columns of G can be interchanged to force condition (5) to be satisfied. Using full state feedback, the first-order necessary (Kuhn-Tucker) conditions for optimality can be analytically solved to give [8]

$$K = R^{-1}G^T\Lambda_x, \qquad \text{where} \qquad F^T\Lambda_x + \Lambda_x F - \Lambda_x G R^{-1} G^T \Lambda_x + W = 0 \tag{6}$$

and where R and W are respectively the $(n_u \times n_u)$ and $(n \times n)$ matrices defined as

$$R = \mathrm{diag}\left\{ \left(\frac{\lambda_{u_i}}{\beta_i^2} \right) R_i \right\}, \qquad W = \sum_{i=1}^{n_\alpha} \left(\frac{\lambda_{y_i}}{\alpha_i^2} \right) H_{d_i}^T W_i H_{d_i}. \tag{7}$$

The variables λ_{u_i} and λ_{y_i} come from the Kuhn-Tucker conditions, and are the Lagrange multipliers associated with the i^{th} control effort and output response constraints respectively.

Equations (6) define the solution to the optimal control problem

$$\min \; J_c = \int_0^\infty \left[\boldsymbol{x}^T W \boldsymbol{x} + \boldsymbol{u}^T R \boldsymbol{u} \right] \mathrm{d}t \tag{8}$$

where K is the optimal steady-state gain matrix, and Λ_x is the steady-state solution to the associated Riccati equation. Although this LQR property only holds true at the optimum point, it is computationally convenient to assume that at every point in the design cycle, the control design variables will be found as the solution to the optimal control problem (8). Therefore, the numerical optimization problem can be reduced to optimization over just the structural design variables, along with an optimal control problem solution which will be a function of the Lagrange multiplier vectors $\boldsymbol{\lambda}_u$ and $\boldsymbol{\lambda}_y$. The immediate benefit of this is a reduced dimensionality nonlinear programming problem. In addition, since the regulator solutions always give a stable closed-loop

system, no explicit check must be performed on the system stability during the solution procedure. The LQR assumption in this proble formulation is similar to the approach taken in [9,10], and others, where R and W are fixed, and not chosen to satisfy the constraints.

2.2 Direct Output Feedback Control

For most real systems, the state vector will be very large, and the use of full state feedback would result in a controller of unacceptably high dimension, assuming additionally that the entire state is available. However, usually only a small subset of the system states will be available to the designer, in the form of the output measurement vector. These can include actual system states along with linear combinations of the system states, in the form $\boldsymbol{y} = H\boldsymbol{x}$, where $\boldsymbol{y}$ is the n_y-vector of outputs and H is the $(n_y \times n)$ output matrix giving the relationship between the outputs and the system states. If these output states are to be used in the feedback loop, the resulting control is termed direct output feedback, with the control forces defined to be $\boldsymbol{u} = -K\boldsymbol{y}$, where K is the $(n_u \times n_y)$ output feedback gain matrix. In this case, the controller design variables will be the $n_u n_y$ elements of K.

Substituting this control into the state equation, the state covariance matrix X for this case can be found as the solution to the same Lyapunov equation (2), except that now the closed-loop dynamics are given by $F_{cl} = (F - GKH)$. Note that since K does not satisfy any special conditions (such as the LQR conditions), the closed-loop system F_{cl} is not guaranteed to be stable for any K. If F_{cl} is unstable at any stage in the solution procedure, the covariance matrix X cannot be found from equation (2). Therefore, for this case, hard constraints on the closed-loop system eigenvalues must be imposed at every step in the design proceedure. "Hard" in this sense means that special precautions must be taken in the solution procedure such that these constraints can never be violated.

Expressions for the controller constraints for direct output feedback in terms of the covariance matrix can then be found to be

$$g_{cc_i} = \frac{\mathrm{tr}[H^T K_i^T R_i K_i H X]}{\beta_i^2} - 1 \qquad \text{for } i = 1, \ldots, n_\beta \tag{9}$$

$$g_{oc_i} = \frac{\mathrm{tr}[H_{d_i}^T W_i H_{d_i} X]}{\alpha_i^2} - 1 \qquad \text{for } i = 1, \ldots, n_\alpha \tag{10}$$

3 Solution Algorithms

In the general problem formulation presented in the previous section, the constraint functions are generally highly nonlinear implicit functions of the design variables. Solution of this problem could be attempted by the direct application of nonlinear programming techniques; that is, using the exact functional expressions for the constraints. However, this approach quickly becomes computationally very expensive as the dimensionality increases since the full objective and constraint functions must be evaluated at every step, and their respective gradients at most, if not all, steps thoughout the design procedure. Such evaluations tend to be computationally very expensive.

Approximation techniques, where the implicit nonlinear problem is replaced by a sequence of explicit approximate (although not necessarily linear) problems, have been shown to yield efficient and powerful algorithms for structural design optimization (see, for example, [11,12]). In this paper, two solution techniques based on approximation techniques will be tested on the integrated control/structure design optimization problem. The methods will be compared with respect to the ease of use, generality of application, and numerical robustness to changes in move-limits and other solution parameters.

3.1 Sequential Nonlinear Approximations

In this method, the fully constrained nonlinear optimization problem is solved by the iterative construction and numerical solution of a sequence of explicit approximate problems. The approximate problems are first-order Taylor's series expansions (with respect to either the inverse design variables ([13], for example), or with respect to hybrid design variables [14]) of the objective and constraint functions. Depending on the intermediate

variables chosen, the approximate functions may still be nonlinear functions of the design variables. Therefore, the numerical solution is accomplished using a mathematical programming code, specifically the modified method of feasible directions as implemented in ADS [15].

The solution process begins with some initial structure, which is analyzed using the finite element technique. At this point, the gradients of the active constraint set are evaluated, and the approximate problem is formed, with respect to the current design. Expressions for the gradients of all constraints considered can be evaluated analytically. The approximate problem is solved with ADS using an active constraint set strategy to reduce the dimensionality of the approximate problem by deleting the inactive constraints. Move-limits on the design variables are imposed during the solution to ensure that the design remains within the region for which the approximation functions are of acceptable quality. The choice of move-limits and how they change can have a significant effect on convergence, and will often be determined from numerical experience with the particular problem at hand.

After the solution of the approximate problem, the structure and its control system are deemed optimal if a convergence test on either the absolute or relative objective function change over a specified number of successive global iterations is satisfied. Otherwise, the objective and active constraint gradients are evaluated for the new design, a new approximate problem formed, and the process above is repeated in an iterative manner. The solution procedure ends when the design variables converge, or when the number of iterations exceeds some preset maximum.

Scaling the structure and controller to the closest constraint surface may be possible in some cases, because of the special assumed form of the controller. Scaling to structural constraints has been performed in other work (see [16]) and will not be covered here. If full state feedback is used, it is possible and practical to scale the structure to the closest control effort constraint and closest output response constraint simultaneously. The variables with which the structure is scaled are the structural design variables (elemental areas or thicknesses), and the Lagrange multipliers associated with the two controller constraints λ_u and λ_y (where for clarity, and without loss of generality, the subscripts on the λ's that refer to the particular control effort or output response constraints under consideration have been dropped).

Note that changing the values of λ_u and λ_y cannot independently change the values of $u_{ms} = \mathrm{tr}(K^T R K X)$ and $y_{ms} = \mathrm{tr}(H_d^T W H_d X)$, because in the LQR problem, only the ratio of λ_u to λ_y is important. One can choose the ratio (λ_u/λ_y) to satisfy one of the control constraints — say u_{ms}. Then y_{ms} will not in general be satisfied. Suppose y_{ms} is too large (i.e. $y_{ms} > \alpha^2$) at the particular point where u_{ms} is satisfied. Then the *only* way one can satisfy the y_{ms} constraint is to increase the sizes of at least some of the structural members. This seems reasonable because if the control constraints could be satisfied by simply choosing appropriate controller parameters, then there would be no interaction between structural optimization and controller optimization. Intuitively, it can be seen that this is not the case. Note that each member of the structure will be scaled by the same amount to fulfill our goals. Obviously, this method is not absolutely mandated, and some other approach could be used where the design variables are not scaled equally. However, this would then be *resizing* rather than *scaling*, a process normally left to the nonlinear programming algorithm.

The final scaling aim is to set $u_{ms} = \beta^2$ and $y_{ms} = \alpha^2$. To perform the scaling, it is assumed that, at iteration i, the values $(u_{ms})_i$ and $(y_{ms})_i$ will change, as a result of changes to $(\lambda_u)_i$ and the $(p_j)_i$, according to the equations

$$\frac{(u_{ms})_{i+1}}{(u_{ms})_i} = \Delta_{i+1}^{a_i}\,\delta_{i+1}^{b_i}, \qquad \frac{(y_{ms})_{i+1}}{(y_{ms})_i} = \Delta_{i+1}^{c_i}\,\delta_{i+1}^{d_i} \tag{11}$$

where a_i, b_i, c_i and d_i are constants, and where

$$\Delta_{i+1} = \frac{(\lambda_u)_{i+1}}{(\lambda_u)_i}, \qquad \delta_{i+1} = \frac{(p_j)_{i+1}}{(p_j)_i} \tag{12}$$

If initial (educated) guesses for these constants can be made, they can be updated in an adaptive manner during the scaling proceedure.

Move-limits are imposed on the design variables during each approximate problem solution. This is done in an attempt to restrain the design variables to a region in which the explicit function approximations remain reasonably accurate. However, deciding how to impose these move-limits is a non-trivial task. The local curvature of the design space (i.e. how nonlinear are the actual constraint surfaces in the region about the expansion point of the approximations) will determine the move-limits, with more strict move-limits applied in regions

of high curvature, and less strict move-limits imposed in regions of low curvature. Since second-derivative information is required to estimate curvatures, and since such evaluations are very expensive computationally, imposing move-limits is usually reduced to an art based on past experience. Quasi-Newton methods obtain the second derivatives using only first derivative information, however, these methods typically take N iterations to fill the Hessian, and can be very costly if N is large.

For the purpose of this work, a move-limits factor γ is imposed in an exponential form. If the current design variable and approximation expansion vector is $\boldsymbol{p}$, then the upper and lower bounds on the design variables for the current approximate problem are defined as

$$\boldsymbol{p}_u = \gamma \boldsymbol{p}, \qquad \boldsymbol{p}_\ell = \frac{1}{\gamma}\boldsymbol{p} \tag{13}$$

where $\gamma \geq 1$. The limits specified in equation (13) must be imposed element by element. Note that since the design variables in this example will be structural design variables only, they are restricted to be positive. Obviously, equation (13) must be modified if the design variables can be negative. The exponential form of the move-limit factor is defined by the particular choice of γ_{min} and γ_{max} (typically 1.2 and 1000 respectively in this work).

3.2 Continuation and Sequential Linear Programming

The complex nature of the constraint functions in the nonlinear optimization problem, especially the controller constraints, leads to various convergence problems in the context of a classical gradient based nonlinear programming code such as ADS. As the problem dimensionality increases, convergence will usualy become increasingly difficult to accomplish, as step sizes reduce to satisfy the local linearity assumptions inherent in gradient based solution techniques. Another method for the solution of mathematical programming problems that has recently become popular is the use of continuation methods to impose nonlinear constraints coupled with sequential linear programming (SLP) [17,18]. The continuation procedure is a conceptually simple method of applying restrictive constraints gradually from less restrictive ones, which replaces the most demanding constraint functions of the form $g(\boldsymbol{p}) \leq 0$, by a set of neighbouring constraint functions $\mathcal{G}_i$, defined by

$$\mathcal{G}_i(\boldsymbol{p}_i, \gamma_i) = g(\boldsymbol{p}_i) - (1-\gamma_i) g(\boldsymbol{p}_0) \leq 0 \qquad \text{for } i = 0, \ldots, M \tag{14}$$

where $\boldsymbol{p}_0$ is the arbitrarily chosen initial design point, and γ_i is a continuation parameter satisfying

$$0 = \gamma_0 \leq \gamma_1 \leq \cdots \leq \gamma_M = 1 \tag{15}$$

Note that for $\gamma_0 = 0$, when $\boldsymbol{p} = \boldsymbol{p}_0$, the new constraint function $\mathcal{G}_0$ is identically satisfied. If convergence is acheived for $\gamma = 1$, then the original constraints will be recovered in M steps. The step size $\Delta\gamma = \gamma_i - \gamma_{i-1}$ (and hence M), can be chosen small enough so that assumptions on local linearity can be almost arbitrarily satisfied.

Linear Programming (LP) methods are a powerful approach to handling a large number of locally linear constraints, and due to the wide availability of very efficient LP codes, are an attractive alternative to nonlinear programming methods. The neighbouring problems generated by the continuation procedure can be written in the form

$$\text{Minimize } J(\boldsymbol{p}_i), \text{ subject to } \mathcal{G}_i(\boldsymbol{p}_i, \gamma_i) \leq 0 \tag{16}$$

To transform these equations into a linear programming problem, the equations are linearized about the current point $\boldsymbol{p}_i$, and move-limits on the maximum parameter changes allowable locally are imposed. Expanding the objective and constraint equations in (16) to first order in a Taylor's series expansion about $\boldsymbol{p}_i$ gives the locally linearized problems in linear programming form as

$$\text{Minimize, with respect to } \Delta \boldsymbol{p}_i,\ \Delta J_i = \left[\frac{\partial J}{\partial \boldsymbol{p}}\right]_{\boldsymbol{p}_i} \Delta \boldsymbol{p}_i, \text{ subject to}$$

$$\left[\frac{\partial \mathcal{G}_i(\boldsymbol{p}_i, \gamma_i)}{\partial \boldsymbol{p}}\right]_{\boldsymbol{p}_i} \Delta \boldsymbol{p}_i + \boldsymbol{g}(\boldsymbol{p}_i) - (1 - \gamma_i)\boldsymbol{g}(\boldsymbol{p}_0) \leq 0 \tag{17}$$

$$-\boldsymbol{\epsilon} \leq \Delta \boldsymbol{p}_i \leq \boldsymbol{\epsilon}$$

for $i = 0, \ldots, (M-1)$. All elements of $\boldsymbol{\epsilon}$ are assumed positive, and the vector inequality is imposed element by element.

The algorithm begins with an initial structure, which is analyzed using the finite element technique. The initial problem ($\gamma_0 = 0$) is solved, and the continuation parameter γ is incremented from $\gamma_0 = 0$ by $\Delta\gamma$ to γ_1. Note that if initially $K = 0$, so that the control effort allowed for the initial local problem is zero, this initial problem becomes a pure structural optimization subject to the dynamic output repsonse constraints. The increment $\Delta\gamma$ is set by an a priori choice of M, the number of continuation steps, although $\Delta\gamma$ need not be constant throughout the solution procedure. Successful implementation of the continuation method has been reported when $\Delta\gamma$ was chosen as initially quite small and increased to a larger value during the solution [18]. The choice of $\Delta\gamma$ is closely coupled with the choice of the nominal design variable move limit vector $\boldsymbol{\epsilon}_0$. There is in fact a tradeoff between the satisfaction of the local linearity assumption through $\boldsymbol{\epsilon}$, and the ability to converge to the neighbouring problem through $\Delta\gamma$. Usually, for each particular problem, some numerical trial and error will be required to find those values of $\boldsymbol{\epsilon}_0$ and $\Delta\gamma$ that yield an efficient solution technique.

The gradients of the objective and constraint functions are calculated at the current design, and then the associated linear programming problem (17) is solved by a linear programming code. In this work, the linear programming routine E04MBF from NAGLIB (National Algorithms Group LIBrary) was used, although other routines inserted at this point should provide the same solution. Since the local linearity assumption will never be exactly satisfied, the actual constraint values at the new point, specified by the solution to the linear programming problem, will be different than that predicted. Therefore, the constraints $\mathcal{G}_i$ may not be satisfied following the linear programming solution step.

Since a converged subproblem solution is required before increasing the continuation parameter, this local problem is iterated locally until convergence is obtained. At each local iteration, new gradients are calculated, and the move limits on the design variables are reduced so that $\boldsymbol{\epsilon} = c\boldsymbol{\epsilon}_0$, where a value of $c = 0.75$ was used in this work. If convergence to the local problem does not occur within 15 iterations (where move limits are about 1% of their nominal values), the move limits are increased to their nominal values $\boldsymbol{\epsilon}_0$, and the local iterations are repeated. Numerical experience with this algorithm has shown that this procedure is flexible enough numerically so that converged subproblem solutions can be obtained in a reasonable number of local iterations, as long as the neighbouring problems are "close enough". Practically, this means that either the constraint values of the initial system should be "close" to their final desired values, or that M should be large. Once the local problem has been solved, the continuation parameter γ is incremented, and the new local problem solved as before. At the M^{th} continuation step, $\gamma = 1$ and the original problem is recovered, so that the solution to the M^{th} local problem is the solution to the original problem.

If closed-loop stability constraints are violated at any stage in the solution procedure, these must be imposed immediately. To achieve this, it is possible to employ a method that never requires the calculation of the closed-loop eigenvalue derivatives, saving considerable computational expense. Of course, if in addition to overall stability there are constraints on closed-loop damping ratios or bandwidth, then the evaluation of the closed-loop eigenvalue derivative may be necessary at some point. The method used in this paper is to simply bisect $\Delta \boldsymbol{p}_i$ and perform another analysis until a stable system configuration is obtained.

4 Gradient Analysis

For the numerical optimization procedure to be practical, especially as the dimensionality increases to realistic structures, it is essential that it be possible to evaluate the first-order sensitivities of the complex constraint functions in an efficient manner. The objective function (the weight) is a linear function of the finite element

thicknesses and/or cross sectional areas (for truss type finite elements), so that its gradient is easy to calculate at any point in the design space.

4.1 Gradients for the case of Full State Feedback Control

The gradients of the controller constraints with respect to a structural design variable p_j are given by

$$\frac{\partial g_{cc_i}}{\partial p_j} = \frac{1}{\beta_i^2}\mathrm{tr}\left[\left(\frac{\partial K_i^T}{\partial p_j}R_iK_i + K_i^TR_i\frac{\partial K_i}{\partial p_j}\right)X + \mathcal{P}_i\mathcal{H}_j\right] \tag{18}$$

$$\frac{\partial g_{oc_i}}{\partial p_j} = \frac{1}{\alpha_i^2}\mathrm{tr}\left[\left(\frac{\partial H_{d_i}^T}{\partial p_j}W_iH_{d_i} + H_{d_i}^TW_i\frac{\partial H_{d_i}}{\partial p_j}\right)X + \mathcal{Q}_i\mathcal{H}_j\right] \tag{19}$$

where $\mathcal{P}_i$, $\mathcal{Q}_i$, and $\mathcal{H}_j$ are evaluated using the following set of equations:

$$F_{cl}^T\mathcal{P}_i + \mathcal{P}_iF_{cl} + K_i^TR_iK_i = 0 \tag{20}$$

$$F_{cl}^T\mathcal{Q}_i + \mathcal{Q}_iF_{cl} + H_{d_i}^TW_iH_{d_i} = 0 \tag{21}$$

$$\mathcal{H}_j = \left[\frac{\partial F_{cl}}{\partial p_j}X + X\frac{\partial F_{cl}^T}{\partial p_j} + \frac{\partial G_w}{\partial p_j}X_wG_w^T + G_wX_w\frac{\partial G_w^T}{\partial p_j}\right] \tag{22}$$

$$\frac{\partial F_{cl}}{\partial p_j} = \left(\frac{\partial F}{\partial p_j} - \frac{\partial G}{\partial p_j}K - G\frac{\partial K}{\partial p_j}\right) \tag{23}$$

$$\frac{\partial K}{\partial p_j} = R^{-1}\left[\frac{\partial G^T}{\partial p_j}\Lambda_x + G^T\frac{\partial \Lambda_x}{\partial p_j}\right] \tag{24}$$

$$F_{cl}^T\frac{\partial \Lambda_x}{\partial p_j} + \frac{\partial \Lambda_x}{\partial p_j}F_{cl} = -\left[\left(\frac{\partial F}{\partial p_j} - \frac{\partial G}{\partial p_j}K\right)^T\Lambda_x + \Lambda_x\left(\frac{\partial F}{\partial p_j} - \frac{\partial G}{\partial p_j}K\right) + \frac{\partial W}{\partial p_j}\right] \tag{25}$$

$$\frac{\partial W}{\partial p_j} = \sum_{i=1}^{n_\alpha}\left(\frac{\lambda_{y_i}}{\alpha_i^2}\right)\left(\frac{\partial H_{d_i}^T}{\partial p_j}W_iH_{d_i} + H_{d_i}^TW_i\frac{\partial H_{d_i}}{\partial p_j}\right) \tag{26}$$

Note that gradients with respect to controller design variables need not be evaluated since these design variables were effectively removed from the optimization problem by the LQR constraint.

4.2 Gradients for the case of Direct Output Feedback Control

The gradients of the controller constraints with respect to a structural design variable p_j are given by

$$\frac{\partial g_{cc_i}}{\partial p_j} = \frac{1}{\beta_i^2}\mathrm{tr}\left[\left(\frac{\partial H^T}{\partial p_j}K_i^TR_iK_iH + H^TK_i^TR_iK_i\frac{\partial H}{\partial p_j}\right)X + \mathcal{Y}_i\mathcal{N}_j\right] \tag{27}$$

$$\frac{\partial g_{oc_i}}{\partial p_j} = \frac{1}{\alpha_i^2}\mathrm{tr}\left[\left(\frac{\partial H_{d_i}^T}{\partial p_j}WH_{d_i} + H_{d_i}^TW_i\frac{\partial H_{d_i}}{\partial p_j}\right)X + \mathcal{Z}_i\mathcal{N}_j\right] \tag{28}$$

where $\mathcal{Y}_i$, $\mathcal{Z}_i$, and $\mathcal{N}_j$ are evaluated using the following set of equations:

$$F_{cl}^T\mathcal{Y}_i + \mathcal{Y}_iF_{cl} + H^TK_i^TR_iK_iH = 0 \tag{29}$$

$$F_{cl}^T\mathcal{Z}_i + \mathcal{Z}_iF_{cl} + H_{d_i}^TW_iH_{d_i} = 0 \tag{30}$$

$$\mathcal{N}_j = \left[\frac{\partial F_{cl}}{\partial p_j}X + X\frac{\partial F_{cl}}{\partial p_j}^T + \frac{\partial G_w}{\partial p_j}X_wG_w^T + G_wX_w\frac{\partial G_w^T}{\partial p_j}\right] \tag{31}$$

$$\frac{\partial F_{cl}}{\partial p_j} = \left(\frac{\partial F}{\partial p_j} - \frac{\partial G}{\partial p_j}KH - GK\frac{\partial H}{\partial p_j}\right) \tag{32}$$

The gradients of the controller constraints with respect to the elements of the gain matrix K can be written in matrix expression form as

$$\frac{\partial g_{cc_i}}{\partial K} = \frac{2}{\beta_i^2}\left[\{\mathcal{R}_i K H - G^T y_i\} X H^T\right] \tag{33}$$

$$\frac{\partial g_{oc_i}}{\partial K} = -\frac{2}{\alpha_i^2}\left[G^T Z_i X H^T\right] \tag{34}$$

where, for scalar s and matrix A with elements a_{ij}, $\left[\frac{\partial s}{\partial A}\right]_{ij} = \frac{\partial s}{\partial a_{ij}}$, and where

$$\mathcal{R}_i = \text{diag}\ \{0 \cdots 0\ \ R_i\ \ 0 \cdots 0\ \}_{(n_u \times n_u)} \tag{35}$$

Note that the R_i in equation (35) is of order $(n_{u_i} \times n_{u_i})$, and is in the i^{th} diagonal block of $\mathcal{R}_i$.

5 Example: The DRAPER I Teterahedral Truss Structure

The DRAPER I structure [19] is a tetrahedral truss attached to the ground by three right-angled bipods, as shown in Figure 1. Although attached to the ground, this model will act as a typical flexible structure pointing subsystem (e.g. antenna, radar, optical) attached to a rigid core. Any motion would then be with respect to this rigid core, and transmit forces to it. Consequently, this model has no rigid body degrees of freedom. The finite element model has 12 truss elements, since the joints are pinned and transmit no moments. There are four nodes that are free to move in all directions, so the model contains 12 degrees of freedom. The structural design variables are the cross-sectional areas of each of the 12 truss elements. Since there are 12 degrees of freedom in the model for this structure, the state-space model will be 24^{th} order. There will be 6 inputs corresponding to the 6 legs of the structure, and a varying number of outputs, depending on the problem at hand.

For the purposes of this work, material parameters of $\rho = 0.1$ lb/in and E = Young's Modulus = 20 kpsi were used. The dimensional values E and ρ were chosen to give initial numerical values of structural frequencies for the dimensional model roughly comparable to those of the non-dimensional model. The model contains no nonstructural mass. Elements 7 through 12, the three right-angled bipods, take on the duties of force actuators (and possibly colocated velocity and/or displacement sensors). Only one output response constraint is defined ($n_\alpha = 1$), with the design output vector y_d representing the line-of-sight error of the top vertex [(x, y) displacements of vertex 1]. The disturbances, labelled w_1 and w_2 in Figure 2, are assumed to be independent, zero mean, Gaussian disturbances with intensity 1.0.

The damping added to the state space system will depend on the state space realization used. For cases where a realization based on physical variables is used, the damping matrix C is formed to be $C = 0.1M + 0.001K_s$, For cases where a realization based on modal variables was used, the damping ratio of each mode was specified to be 0.1% of the modal frequencies during the formation of the state matrices. The weighting matrices R and W are set to the identity matrices, so that equal weighting is given to all components of u and y_d. The minimum cross-sectional areas for all elements was specified as 0.1 in^2. For this problem, no static structural constraints were specified in this model of the DRAPER I structure, the intent being to investigate the effect of the closed-loop controller constraints on the structural design optimization.

5.1 Full State Feedback Control

In this section, the sequential nonlinear approximations solution algorithm, with the addition of the scaling procedure outlined in Section 3.1, is applied to the full state feedback control of the DRAPER I structure. The effect of the scaling procedure used here can be determined by applying the sequential approximations algorithm in the form of a direct output feedback problem with $H = I$ with no scaling assumed. The continuation solution algorithm is also best handled in the form of a direct output feedback problem since no special scaling is assumed. Both solution algorithms applied to the case of full state feedback are discussed in Section 5.2, where direct output feedback control is considered. For brevity, only limited results for both

controller methodologies are presented in this paper, but a full discussion of this example can be found in reference [20].

Runs were made optimizing the DRAPER I structure using an inverse design variable approximation for all constraint functions. The initial structure was defined with all structural design variables set at 10 in^2, and with the Lagrange multipliers λ_u and λ_y set at 1.0. This set of initial conditions will be termed the symmetric set of initial conditions, for they specify a structure with a number of vibrational modes of the same frequency (repeated eigenvalues). A range of allowable expected output response (α^2) of 1×10^{-5} in^2 to 1×10^{-4} in^2 in steps of 1×10^{-4} in^2, and allowable expected control effort (β^2) of 50 lb^2 to 80 lb^2 in steps of 10 lb^2 were used.

Table 1 summarizes the resulting minimum weight in pounds found for the case where a state-space realization based on the modal displacements and velocities was used. Intuitively, two trends would be expected in the data displayed in Table 1. The optimum weight should decrease as the allowable control effort β^2 is increased at constant allowable output response α^2 (left to right across the table), and the optimum weight should decrease as the allowable output response α^2 is increased at constant allowable control effort β^2 (down the table). With reference to Table 1, we can see that this trend is observed in a macroscopic sense only, there being several examples where this trend is not observed. For example, considering the first column of Table 1, which corresponds to $\beta^2 = 50$ lb^2 for varying α^2, we see only two exceptions to the expected trends, these being at α^2 values of 6×10^{-5} and 9×10^{-5}. Similar results are observed in all other columns and rows of Table 1. Results obtained using a state-space realization based on the physical displacements and velocities of nodal points are more consistent than when using the modal variables, although still not totally uniform. It might be pointed out that the results when using a physical realization were consistently easier to obtain, there being no need to alter the nominal value of γ_{min} to obtain convergence, and the number of global iterations required for convergence being consistenty lower.

Some understanding of these contradictory results can be found by considering Table 2, which gives the optimal element areas found for $\beta^2 = 50$ and for the varying α^2 corresponding to the first column of Table 1. Also given in this table is the number of global iterations required for convergence, the final values of the Lagrange multipliers (which then defines the LQR controller), and the initial value of the structural design variables (all the same for the symmetric set of initial conditions) at which the initial scaled system satisfies the constraints. Immediately apparent from Table 2 is a number of seemingly separate regions of the design space into which this structure has converged. For example, the final designs for $\alpha^2 = 5 \times 10^{-5}$ and $\alpha^2 = 7 \times 10^{-5}$ seem to be similar in relative structure. Here, "similar" refers to the relative sizing of the structural members, in that design variables that are "larger" in one design are "larger" in the other. Both these designs are however distinctly different from those for $\alpha^2 = 1 \times 10^{-5}$ and $\alpha^2 = 3 \times 10^{-5}$, which themselves are similar. The conclusion seems to be that we are converging into different regions of the design space with our solution algorithm, and that there are numerous local minima. Several columns of Table 2 seem to define their own region of the design space, being dissimilar to any other column. In other words, our design space seems to have multidimensional corrugations leading to multiple local minima. The solutions will lie somewhere on the intersection hyperplane between the surface of constant allowable output response and the surface of constant allowable control effort.

This corrugated nature of the design space can be illustrated by considering the solutions obtained, for the same constraint case, when starting from different initial conditions. For the case of $\beta^2 = 75$ and $\alpha^2 = 1 \times 10^{-5}$, Table 3 summarizes the results of runs made when modal state space realizations were used, and when only the initial conditions are varied. The different initial conditions are defined by setting all structural elements equal except the first (element 1), to which is added a percentage of the size of other elements. Even with this limited variation in the initial conditions, there are seemingly many distinct regions in the design space into which the system may converge. A picture of the constraint surfaces as a one-dimensional slice of the multidimensional space will emerge if these optimal structures are varied into each other in a linear fashion, and the constraint values are calculated between each case. That is, the structural design variables and Lagrange multipliers are changed linearly from the optimal values in one case to those in another case. Then the constraint surfaces obtained would be those seen when travelling in a straight line between each successive point.

The results of such an analysis are shown in Figures 3 for the cases corresponding to those given in Table 3. As expected, the weight varies linearly between the cases, but it is the constraint curves that are much more revealing. For example, considering Figure 3, one can see that between case 1 and case 2, there is a "ridge" of output response larger than the maximum allowable value. Similarly, the control effort first decreases, then also increases to a ridge of high value. This corresponds to a hump in the constraint surfaces between the two

points in the design space. Assuming that we would see such behaviour when moving in every direction away from case 1 and case 2, rather than just in a direction between the two as shown in Figure 3, then the design points corresponding to these cases would represent local minima. In this situation, the design can become "trapped" in such a locally convex region, causing the solution algorithm to converge to different points.

With reference to the same Figure 3, one can see that both the output response and control efforts are virtually constant between cases 2 and 3, while the weight increases slightly from 2053.0 lb to 2090.6 lb. This indicates that case 2 and case 3 actually represent the same optimal solution, with the difference being accounted for in the variance allowed by the convergence criteria used. The direction in the design space represented by the movement from case 2 to case 3 would lie in the intersection hyperplane of the surfaces of constant control effort and output response constraints, and would be at a shallow angle to the linear surface of constant weight. Figure 3 graphically illustrates a design space that is a very complicated function of the design variables, in which multiple local minima abound.

There are some other tests that can be made on the hypothesis that the design is becoming trapped in local minima. If the design is actually trapped in a local minimum, the solution should stay in the vicinity of that minimum if the problem is changed only slightly. That is, if a converged solution is used as the initial conditions for an optimization run where the constraint objectives are changed by a "small" amount, then the new problem should converge to a point that is "close to" the initial point. Table 4 represents such a situation. Here, the solution was first obtained for the case where $\beta^2 = 50$ and $\alpha^2 = 1 \times 10^{-5}$, and where a modal state space realization and inverse design variable approximations were used. This converged solution was then used as the initial conditions for the cases $\beta^2 = 50$ and $\alpha^2 = 2 \times 10^{-5}$, and $\beta^2 = 60$ and $\alpha^2 = 1 \times 10^{-5}$. Moving down each column, and across the top row, of Table 4, the converged solution from the previous case was used as the initial condition for the new problem. As can be seen from Table 4, the two expected trends in the data, as mentioned previously, are now observed without exception. The optimal solutions for the first column of Table 4, corresponding to the cases where $\beta^2 = 50$, are given in Table 5 The solutions now appear to be in the same local region of the design space, as evidenced by the relative sizing of the optimal structures. For example, note that in all converged designs, structural elements 9, 10, and 12 are at their lower gage limit of 0.1 in^2, and that the first structural element is the largest by far. The optimal solutions for the $\beta^2 = 60$ cases from Table 4 also appear to be in this same region of the design space. These results test the hypothesis that designs are converging to local minima, and indicate that the local optima are real and not simply figments of a numerical imagination.

5.2 Direct Output Feedback Control

Recall that when using direct output feedback, no simplifying assumptions can be made regarding the controller design variables (elements of K), such as the LQR assumption used in the case of full state feedback. Therefore, no scaling of the structure and controller to the closest constraint surface is performed. If a full state feedback case is to be solved in the form of direct output feedback with $H = I$, the number of design variables will increase significantly over the number of design variables created when other types of controllers are considered. This is because the number of states in the plant model will generally be large for anything but trivial systems, hence K will have many elements, all of which will be treated explicitly as design variables. However, such a situation is considered here to aid a comparison between the two solution algorithms, and the results obtained in the previous section. For the continuation algorithm, convergence was obtained after the specified number of global iterations, set by the specification of $\Delta\gamma$. Also note that, since only one each of the output response and control effort constraints are specified at this stage ($n_\alpha = 1$, $n_\beta = 1$), these can both be set as equality constraints without loss of generality since they will both be active at an optimum.

Table 6 gives the optimal weights found using the sequential approximations solution algorithm in the case when a physical state space realization and inverse design variable approximations are used, for $\beta^2 = 50$ and $\beta^2 = 60$, and for varying α^2. Compared to the similar case when scaling was performed, the optimal weights found here are larger in every case. Additionally, the solution times were significantly larger because of the number of iterations required for convergence. Note that some values in Table 6 are for situations where an average steady state value was obtained, but where the design was jumping around too much over each global iteration for convergence to occur. This was so even though the move-limits for every case in Table 6 were set at a relatively small $\pm 2.5\%$. A smaller move-limit would aid convergence, but slow it considerably. Also, smaller move-limits may cause premature convergence if the design is at a point where the constraint surfaces

and the surface of constant weight are nearly parallel.

Table 7 gives the optimal weights found using the continuation algorithm for the same cases listed in Table 6. All cases listed were obtained with the continuation parameter $\Delta\gamma = 0.01$ until $\gamma = 0.5$, when $\Delta\gamma$ became 0.02, so that for every case convergence was achieved in 75 global iterations. Also listed in Table 7 are the move-limits on the maximum parameter changes allowable locally (ϵ in Section 3.2). These are set to give a tradeoff between the satisfaction of the local linearity assumption and convergence to the local neigbouring problem, and must be found by numerical experimentation. Note that in both situations represented by Tables 6 and 7, minimum move-limits on the elements of K are set so that these elements can change sign if desired.

The solutions given in Tables 6 and 7 were obtained when the initial structure was defined with all truss elements of equal cross-sectional area. The particular values for this area are given in Tables 6 and 7 for each case, and were chosen so that the initial output response was "close to" its desired final value. With all structural elements at 120 in^2, 90.0 in^2, and 60.0 in^2, the initial output responses were 2.067×10^{-5}, 3.674×10^{-5}, and 8.266×10^{-5} respectively. The initial control effort was zero of course, since all elements of K were set initially to zero.

Comparing the optimal weights from Tables 6 and 7, it can be seen that those obtained using the continuation solution algorithm are significantly lower than those obtained using the sequential approximations solution algorithm in every case. The optimal weights are also much more consistent, in terms of the expected trends as α^2 increases, when using the continuation method. Convergence was obtained for every case listed in Table 7, whereas for three of the cases listed in Table 6, no convergence was obtained within 300 iterations. The reason for the convergence failure can be illustrated by considering the convergence histories given in Figures 4 and 5, for typical cases from Tables 6 and 7 respectively. The inset in Figure 4 shows some of the histories toward the end of the solution in more detail. These can be seen to be very rough, as compared to the histories in Figure 5 which are smooth everywhere. These parameter oscillations for the sequential approximations solution technique are indicative of a move-limit set too high, so that the local linearity assumption is violated. However, since the convergence is very flat toward the end of the solutions, a smaller move-limit is likely to cause premature convergence, or to significantly slow down the convergence for very little additional objective reduction.

The optimal weights found using the continuation method listed in Table 7 compare very favourably with those found for the same cases ($\beta^2 = 50$ and varying α^2) when the full state feedback LQR assumption was used to simplify solution. In almost every case, the optimal weight is approximately equal to or lower than those found earlier. However, the sequential approximations solution algorithm results are much worse, being significantly larger than the optimal weights found earlier in every case. The sequential approximations solution algorithm performs so poorly because it tends to become trapped in a local minimum close to the specified initial conditions.

Consider now cases where the full state is not available for feedback. In the DRAPER I structure, the six right-angled bipods are usually assumed to take on the duties of force actuators and colocated rate sensors, so that $H = G^T$ (output state of dimension six). Tables 8 and 9 list the optimal weights found for this reduced-order output vector for the same cases used in Table 6, when the sequential approximations and continuation solution algorithms respectively are used. Note that the optimal weights consistently obey the trends that are expected as the allowable output response and controller effort are altered. Even so, the designs can converge into completely different regions of the design space for any particular case. This is illustrated by Table 10, which gives the optimal structural design variables for the cases represented by the second column of Table 9. Note that every solution does not define a unique local minima. Many of the solutions seem to lie in the same local minima region, for example the solution for the case $\alpha^2 = 1{,}7$, and 9×10^{-5}. Once again, the design space is found to have many local minima.

The two solution algorithms here seem to predict quite similar optimal weights for the cases considered, although in general those found using the continuation method are slightly better. However, convergence for the results using the sequential approximations solution algorithm were more difficult to obtain than those for the continuation method. Altering the nominal move limits (set at $\pm 2.5\%$), and perhaps reducing it toward the latter stages of each solution would aid convergence, but at increased effort on the part of the user. The continuation results are easier to obtain since they require less individualized attention.

Comparing Tables 8 and 9 to Table 7, it can be seen that the optimal weights found when $H = G^T$ are much larger than those found when full state feedback was used. The reason for this is the particular placement of the disturbance forces relative to the design output states (which determine the output response). For the

case considered here, the DRAPER I structure is disturbed at node 1, the displacement of which is to be kept below the design objective value α^2. With full state feedback, the displacement states of node 1 are available for feedback, whereas if $H = G^T$, these states are not available, and the effect of the displacement of node 1 is available only indirectly through its effect on the velocities along the six bipods that make up the sensors. The importance of these states can be seen by examining the optimal gain matrices for the full state feedback case. The largest gains associated with displacements and velocities in these matrices appear in the columns corresponding to node 1 degrees of freedom, indicating that the states associated with node 1 are very important. When they are not available, the controller does not have as much information about the state of node 1, the node it is trying to control, as it does in the case of full state feedback, and will increase the structural stiffness (and hence mass) to compensate.

To illustrate further, consider output feedback with the displacement and velocity states of node 1 added to the output vector used previously. The optimal weights for the same $\beta^2 = 50$ and $\beta^2 = 60$ cases used in Table 9 with this new output vector (of dimension 12 now) are given in Tables 11 and 12, when the sequential approximations and continuation solution algorithms are used respectively. There are now larger differences between the weights found using the two solution algorithms than in the case when $H = G^T$ only, with the continuation method giving the best results (with one exception). The weights obtained using the continuation algorithm are now very close to those obtained when using full state feedback (in Table 7), although still a little larger in every case. This is because even with this larger output vector, the displacement states for nodes two through four are still not used in the controller.

All the results presented so far were generated with the external stochastic disturbance intensities set at one ($X_w = I$). If these intensities are varied as $X_w = x_w I$, the effect of varying x_w on the optimal weight is shown in Figure 6. These results were generated using the continuation solution algorithm, for the case when $\alpha^2 = 1 \times 10^{-5}$ and $\beta^2 = 50$, and when all structural design variables were initially set at 120 in^2. As can be seen, the relationship between the disturbance intensity and the optimal structural weight appears basically linear in the range shown. However, a linear fit of the data does not produce an optimum weight of zero for a zero intensity disturbance, as would be expected. Therefore, the relationship cannot be exactly linear. The results also indicate that the disturbance level chosen for the previous results ($x_w = 1$) was significant for the range of output response and control effort constraint objectives used.

As long as the neighbouring problems in the continuation algorithm were close enough, which was satisfied by starting from an initial point where the constraint values were "close to" their final desired locations, no difficulties were experienced in obtaining convergence. The solution times for the continuation algorithm were 2–3 times longer than for solutions by the sequential approximations algorithm, since on average approximately 8–12 local iterations were required for convergence to the neighbouring problem for each global iteration. The performance of the sequential approximations solution algorithm decreases as the dimensionality increases, as evidenced by the sequence of cases where $H = G^T$ (48 design variables), $H = G^T$ plus node 1 displacement and velocity states (84 design variables), and $H = I$ (156 design variables). However, the continuation method seemed much less sensitive to the problem dimension. The continuation solution method was found to be generally superior, for this problem at least, to the sequential approximations solution method, with respect to the confidence in obtaining a "good" converged solution. The disparity in solution times was acceptable because of the ease with which solutions were obtained using the continuation method, and because the solutions found seemed to be generally much better than those found using the sequential approximations algorithm.

6 Conclusions

In this work, the integrated control/structure design optimization problem has been investigated from a response to disturbances point of view. Both full state and output feedback controllers were employed in the control strategy, and two solution methods were compared. It was found that for this problem, the continuation method coupled with a sequential linear programming approach performed better than the more traditional type of nonlinear approximations approach, in the sense that it was more robust to changes in the arbitrary parameters set by the user, obtained better results, and the results were easier to obtain. The design space was found to exhibit multiple local minima in which the solution could become trapped, although the continuation solution method seemed to handle the corrugated design space better than the other method. In future work, more diverse controller types must be considered, along with structures consisting of more complicated finite

elements than simple truss members. Additionally, more realistic problems of higher dimension must be solved, to demonstrate the practicality of this design procedure.

Acknowledgments

This work was partially funded by the Air Force Office of Sponsored Research at the Flight Dynamics Laboratory, Wright Aeronautical Laboratiories, and also by a grant from CRAY Research Inc. Computer facilities on a CRAY-XMP were provided by the Ohio Supercomputer Center in Columbus, Ohio.

References

[1] Hanks, B.R. and Skelton, R.E., "Designing Structures for Reduced Response by Modern Control Theory", Proceedings of the 24th AIAA Structures, Structural Dynamics and Materials Conference, Lake Tahoe, Nevada, May 2–4, 1983.

[2] Komkov, V., "Simultaneous Control and Optimization for Elastic Systems", Proceedings of the Workshop on Applications of Distributed System Theory to the Control of Large Space Structures, JPL Publication 83-46, ed. by G. Rodriguez, July 1983.

[3] Hale, A.L., Lisowski, R.J. and Dahl, W.E., "Optimizing Both the Structure and the Control of Maneuvering Flexible Structures", Proceedings of the AAS/AIAA Astrodynamics Conference, Lake Placid, New York, August 22–24, 1983.

[4] Salama, M., Hamidi, M. and Demsetz, L., "Optimization of Controlled Structures", Proceedings of the JPL Workshop on Identification and Control of Flexible Space Structures, San Diego, California, June 4–6, 1984.

[5] Messac, A., Turner, J. and Soosaar, K., "An Integrated Control and Minimum Mass Structural Optimization Algorithm for Large Space Structures", Proceedings of the JPL Workshop on Identification and Control of Flexible Space Structures, San Diego, California, June 4–6, 1984.

[6] Miller, D.F. and Shim, J., "Gradient Based Combined Structural and Control Optimization", *J. Guidance and Control*, Vol. 10, No. 3, May–June 1987, pp 291–298.

[7] Onoda, Junjiro and Haftka, Raphael T., "Simultaneous Structure/ Control Optimization of Large Flexible Spacecraft", AIAA paper 87-0823.

[8] Slater, G.L., "A Disturbance Model for the Optimization of Control/Structure Interactions for Flexible Dynamic Systems", AIAA Guidance, Navigation and Control Conference, Minneapolis MN, August 15–17, 1988, pp 57–63; AIAA Paper 88-4058-CP.

[9] Khot, N.S., Eastep, F.E. and Venkayya, V.B., "Simultaneous Optimal Structural/Control Modifications to Enhance the Vibration Control of a Large Flexible Structure", Proceedings of the AIAA Guidance, Navigation and Control Conference, Snowmass, CO, Aug. 19–21, 1985, pp 459–466.

[10] Khot, N.S., "Minimum Weight and Optimal Control Design of Space Structures", NATO Advanced Study Institute, Computer Aided Optimal Design: Structural and Mechanical Systems, Troia, Portugal, June 29–July 11, 1986.

[11] Schmit, L.A., and Farshi, B., "Some Approximation Concepts for Efficient Structural Synthesis",*AIAA Journal*, Vol. 12, No. 5, 1974, pp 692–699.

[12] Grandhi, R.V. and Venkayya, V.B., "Structural Optimization with Frequency Constraints", *AIAA Journal*, Vol 26, No. 7, July 1988, pp 858–866.

[13] Slater, G.L. and Kandadai, R.D., "Gain Optimization with Non-Linear Controls", *Optimal Control Applications and Methods*, Vol. 5, pp 207–219, 1984.

[14] Starnes, J.H. Jr. and Haftka, R.T., "Preliminary Design of Composite Wings for Buckling, Strength and Displacement Constraints", *Journal of Aircraft*, Vol. 16, No. 8, August 1979, pp 564–570.

[15] Kirk, D.E., *Optimal Control Theory, An Introduction*, Prentice-Hall, Englewood Cliffs, New Jersey, 1970.

[16] Canfield, R.A., Grandhi, R.V. and Venkayya, V.B., "Comparison of Optimization Algorithms for Large Structures", Report AFWAL-TM-86-204-FIBR, Wright-Patterson Air Force Base, May 1986.

[17] Lim, K.B. and Junkins, J.L., "Robustness Optimization of Structural and Controller Parameters", *J. Guidance, Control and Dynamics*, Vol. 12, No. 1, Jan.–Feb. 1989, pp 89–96.

[18] Horta, L.G., Juang, J-N. and Junkins, J.L., "A Sequential Linear Optimization Approach for Controller Design", *J. Guidance, Control and Dynamics*, Vol. 9, No. 6, Nov.–Dec. 1986, pp 699–703.

[19] Strunce, R., Lin, J., Hegg, D and Henderson, T., "Actively Controlled Structures Theory", Final Report, Vol 2 of 3, R-1338, Charles Stark Draper Laboratory, Cambridge, Mass. Dec. 1979.

[20] McLaren, M.D., "Controller Methodologies for the Integrated Control/Structure Design Optimization of Large Flexible Structures", Ph.D. Dissertation, University of Cincinnati, December 1989.

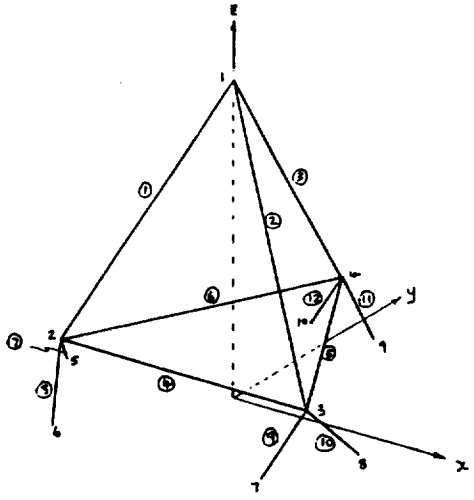


Figure 1: The DRAPER I Structure

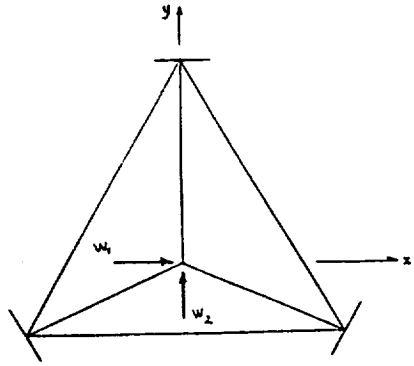


Figure 2: The disturbance model

α^2 ($\times 10^{-5}$)	β^2			
	50	60	70	80
1	2847.1	2991.4	2107.4	2924.9
2	2077.3	2030.8	1914.4	1289.7
3	1878.0	1658.9	1472.9	1216.8
4	1548.3	1409.7	1653.2	1021.5
5	1418.9	1319.2	1459.5	937.4
6	1460.6	1209.5	964.9	1009.8
7	1209.0	1076.0	928.7	1052.7
8	1066.7	1214.1	987.9	790.2
9	1126.3	940.5	854.0	720.1
10	971.9	1035.1	766.4	970.6

Table 1: Optimal weight using a modal state-space realization, the symmetric set of initial conditions, and inverse design variable approximations.

	α^2 ($\times 10^{-5}$)				
	1	2	3	4	5
Scaled DV	90.437	63.947	52.212	45.213	40.441
Final DV					
1	125.893	73.091	70.557	49.717	43.868
2	23.336	53.545	36.664	21.137	23.151
3	45.632	15.268	14.833	35.217	21.510
4	6.389	10.704	5.599	6.318	4.957
5	10.973	5.375	12.207	4.213	9.637
6	9.800	6.209	6.953	5.365	4.914
7	17.627	8.610	10.775	5.638	9.310
8	17.719	0.124	8.802	0.150	9.901
9	0.100	13.250	0.100	0.100	0.100
10	0.100	9.332	0.711	0.313	8.307
11	18.206	0.100	13.746	9.473	8.352
12	0.100	0.100	0.100	9.266	0.100
λ_u	1.6710	1.4574	1.2620	1.4851	0.8632
λ_y	1.0	1.0	1.0	1.0	1.0
iter. for convergence	44	19	15	32	32

	α^2 ($\times 10^{-5}$)				
	6	7	8	9	10
Scaled DV	36.918	34.180	31.973	29.979	28.594
Final DV					
1	25.936	32.194	9.723	16.757	20.331
2	30.650	21.255	27.425	32.495	38.508
3	28.247	20.736	28.618	16.565	11.434
4	11.168	5.141	4.083	8.798	2.041
5	10.922	7.713	6.737	4.348	2.369
6	6.795	4.260	3.815	6.692	2.934
7	0.101	8.299	0.100	0.100	0.152
8	2.501	9.363	0.100	9.628	0.885
9	0.245	0.100	6.482	1.332	3.242
10	9.107	8.183	8.323	0.100	5.016
11	7.744	7.379	8.218	7.737	2.532
12	8.583	0.100	6.782	10.099	0.100
λ_u	0.7522	0.8544	0.7937	0.9117	1.7622
λ_y	1.0	1.0	1.0	1.0	1.0
iter. for convergence	21	18	22	22	49

Table 2: Optimal design variables for $\beta^2 = 50$

	Case 1	Case 2	Case 3	Case 4	Case 5
Final Wt.	2451.3	2053.0	2090.6	1915.6	2067.0
Final DV					
1	73.849	41.076	55.595	38.306	60.319
2	70.699	49.211	55.391	68.734	48.760
3	13.262	74.820	58.403	44.107	50.104
4	16.100	2.146	1.6065	1.227	2.841
5	7.730	1.843	1.4806	1.612	2.169
6	8.068	1.819	1.5676	3.161	2.834
7	3.864	0.100	0.100	0.100	0.170
8	20.163	0.100	0.100	0.451	0.100
9	21.568	0.100	0.100	7.939	8.984
10	5.694	0.100	0.100	0.100	0.100
11	0.100	0.100	0.100	0.100	0.100
12	0.100	0.100	0.100	0.100	9.014
λ_u	0.9765	3.1055	3.3391	2.6755	2.5236

For all $j \neq 1$, the initial conditions are:
Case 1: $p_1 = p_j$
Case 2: $p_1 = p_j + 3\%$
Case 3: $p_1 = p_j + 6\%$
Case 4: $p_1 = p_j + 10\%$
Case 5: $p_1 = p_j + 50\%$

For all cases, $(\lambda_u)_0 = 1.0$, $(\lambda_y)_0 = 1.0$

Table 3: Optimal values when $\beta^2 = 75$ and $\alpha^2 = 1 \times 10^{-5}$ for differing initial conditions, when using a modal state space model

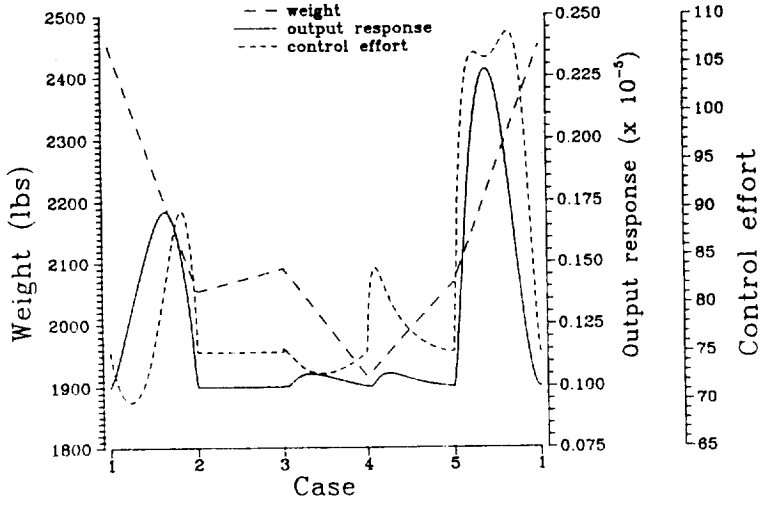


Figure 3: Constraint surfaces between cases listed in Table 3

α^2 ($\times 10^{-5}$)	β^2			
	50	60	70	80
1	2847.1	2466.0	2166.6	1969.2
2	2045.9	1782.1	1579.7	1443.2
3	1685.6	1478.8	1327.5	1197.2
4	1470.9	1298.0	1162.0	1052.0
5	1325.2	1174.2	1049.3	952.1
6	1217.0	1083.2	965.9	880.0
7	1134.1	1012.4	901.4	823.8
8	1065.7	955.3	849.4	776.5
9	1014.5	907.9	806.4	735.3
10	961.3	861.0	770.0	700.2

Table 1: Optimal weight using a modal state-space realization and inverse design variable approximations, where the initial condition for each case is the converged solution from the previous case.

α^2 ($\times 10^{-5}$)	initial dv's (structural)	optimal weight		iterations	
		$\beta^2 = 50$	$\beta^2 = 60$	$\beta^2 = 50$	$\beta^2 = 60$
1	120.0	3847.9	3548.4	220	300*
2	90.0	2724.3	2490.7	202	300*
3	90.0	2191.7	2083.1	231	300*
4	90.0	1950.8	1771.3	188	300*
5	90.0	1680.5	1702.5	300*	181
6	90.0	1529.7	1443.9	300*	300*
7	60.0	1443.0	1395.0	207	181
8	60.0	1325.6	1247.4	300*	300*
9	60.0	1688.2	1591.6	119	123
10	60.0	1604.4	1487.0	126	127

* indicates no convergence in specified number of global iterations

Table 6: Optimal weight using sequential appproximations solution algorithm without scaling for full state feedback, a physical state space realization, and inverse design variable approximations.

α^2 ($\times 10^{-5}$)	initial dv's (structural)	optimal weight	
		$\beta^2 = 50$	$\beta^2 = 60$
1	120.0	3255.6	3013.7
2	90.0	2291.0	2116.4
3	90.0	1864.9	1737.2
4	90.0	1617.7	1630.3
5	90.0	1450.0	1447.6
6	90.0	1321.6	1221.0
7	60.0	1226.8	1135.0
8	60.0	1143.9	1064.0
9	60.0	1085.4	1002.9
10	60.0	1024.3	961.8

Table 7: Optimal weight using continuation solution algorithm without scaling for full state feedback, and a physical state-space realization.

α^2 ($\times 10^{-5}$)	initial dv's (structural)	β^2	
		50	60
1	120.0	4764.9	4502.3
2	90.0	3553.6	3170.5
3	90.0	2744.5	2586.7
4	90.0	2511.8	2388.8
5	90.0	2090.7	2004.6
6	90.0	1954.7	1830.1
7	60.0	1797.0	1698.4
8	60.0	1683.6	1585.0
9	60.0	1584.3	1496.5
10	60.0	1503.4	1420.1

Table 9: Optimal weight using continuation solution algorithm for direct output feedback with $H = G^T$, and a physical state space realiza-

	α^2 ($\times 10^{-5}$)				
	1	2	3	4	5
Final DV					
1	125.893	82.406	66.777	57.864	50.877
2	23.336	19.075	16.010	14.170	13.257
3	45.632	36.194	29.987	26.286	23.752
4	6.389	6.372	5.386	4.736	4.105
5	10.973	8.135	6.487	5.952	5.429
6	9.800	8.558	7.433	6.631	6.031
7	17.627	12.148	12.085	9.342	11.208
8	17.719	10.689	8.446	6.718	5.422
9	0.100	0.100	0.100	0.100	0.100
10	0.100	0.100	0.100	0.100	0.100
11	18.206	11.326	8.820	8.172	7.764
12	0.100	0.100	0.100	0.100	0.100
λ_u	1.6710	1.6281	1.6293	1.6325	1.6273
λ_y	1.0	1.0	1.0	1.0	1.0
iter. for convergence	44	16	2	2	2
	α^2 ($\times 10^{-5}$)				
	6	7	8	9	10
Final DV					
1	46.745	42.850	40.259	37.112	33.742
2	12.302	11.800	11.108	11.403	11.179
3	21.926	20.458	19.239	18.610	17.761
4	3.996	3.621	3.446	4.045	3.745
5	5.144	4.708	4.451	3.751	4.167
6	5.556	5.241	4.926	4.681	4.634
7	7.501	9.330	8.344	7.053	6.621
8	5.286	4.390	4.167	5.319	5.070
9	0.100	0.100	0.100	0.100	0.100
10	0.100	0.100	0.100	0.100	0.100
11	7.237	6.603	6.216	4.678	5.357
12	0.100	0.100	0.100	0.100	0.100
λ_u	1.6243	1.6163	1.6167	1.5670	1.5540
λ_y	1.0	1.0	1.0	1.0	1.0
iter. for convergence	2	2	2	4	3

Table 5: Optimal design variables for $\beta^2 = 50$ cases given in Table 4

α^2 ($\times 10^{-5}$)	initial dv's (structural)	optimal weight		iterations	
		$\beta^2 = 50$	$\beta^2 = 60$	$\beta^2 = 50$	$\beta^2 = 60$
1	120.0	4830.5	4616.8	400*	400*
2	90.0	3626.9	3262.8	223	400*
3	90.0	2903.5	2668.4	400*	400*
4	90.0	2393.5	2324.4	400*	400*
5	90.0	2152.3	2054.5	290	400*
6	90.0	2067.1	1893.6	312	265
7	60.0	1811.6	1815.2	336	254
8	60.0	1704.1	1639.7	400*	400*
9	60.0	—	—	—	—
10	60.0	1629.4	1487.0	425	500*

* indicates no convergence in specified number of global iterations

Table 8: Optimal weight using sequential appproximations solution algorithm for direct output feedback with $H = G^T$, and a physical state-space realization.

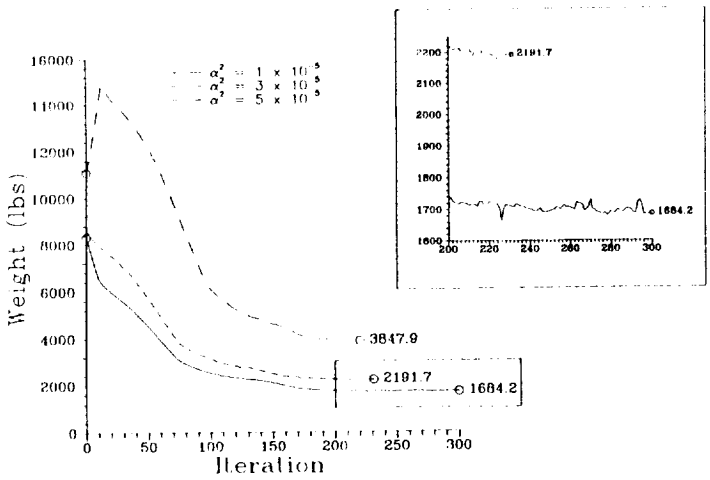


Figure 4: Convergence histories for particular cases from Table 6

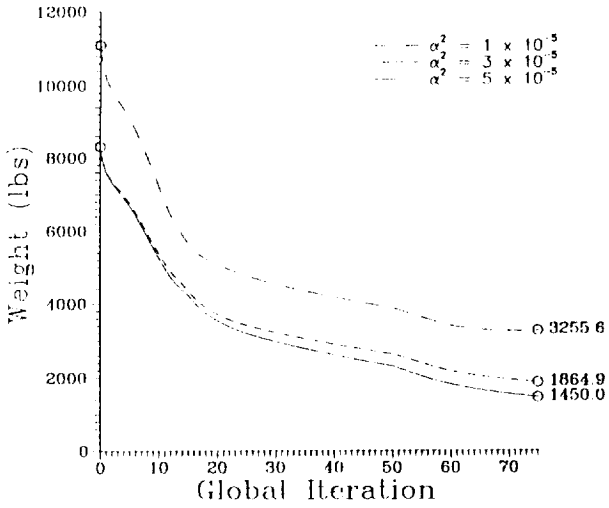


Figure 5: Convergence histories for particular cases from Table 7

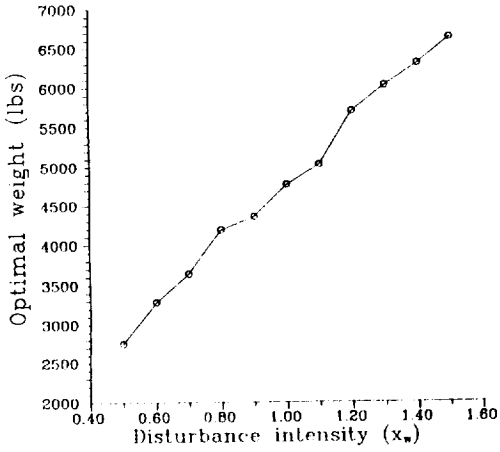


Figure 6: Optimal weight for varying external disturbance intensity $\Lambda_w = x_w I$ with $\alpha^2 = 1 \times 10^{-5}$ and $\beta^2 = 50$

	α^2 ($\times 10^{-5}$)				
	1	2	3	4	5
Initial DV	120.000	90.000	90.000	90.000	90.000
Final DV					
1	92.172	55.944	41.010	44.107	36.632
2	72.274	55.967	52.340	40.128	32.776
3	71.334	62.487	41.938	38.878	31.848
4	41.867	27.830	24.475	24.192	17.445
5	47.282	33.755	24.327	19.986	18.412
6	41.794	33.553	26.973	23.608	20.363
7	41.341	2.365	5.462	17.750	8.363
8	41.314	20.562	1.604	17.277	1.317
9	2.795	20.742	23.982	1.761	10.880
10	9.518	2.424	24.159	12.659	17.091
11	9.541	24.020	1.570	14.006	20.089
12	2.795	23.903	5.612	1.667	1.458
	α^2 ($\times 10^{-5}$)				
	6	7	8	9	10
Initial DV	90.000	60.000	60.000	60.000	60.000
Final DV					
1	28.739	34.922	25.632	30.032	23.272
2	39.315	26.729	32.700	23.857	28.426
3	28.871	26.916	24.976	24.192	22.333
4	17.154	16.238	14.880	14.139	13.401
5	17.249	17.279	15.037	15.367	13.472
6	19.143	16.109	16.386	14.174	14.662
7	4.076	15.734	3.447	14.073	3.130
8	1.143	15.739	0.996	13.904	0.850
9	16.803	1.039	14.683	0.896	13.294
10	16.869	3.667	14.621	3.215	13.149
11	1.200	3.627	1.013	3.291	0.849
12	3.803	1.038	3.387	0.899	3.069

Table 10: Optimal design variables for $\beta^2 = 50$ cases from Table 9

α^2	initial dv's	optimal weight		iterations	
($\times 10^{-5}$)	(structural)	$\beta^2 = 50$	$\beta^2 = 60$	$\beta^2 = 50$	$\beta^2 = 60$
1	120.0	3889.6	3630.7	221	247
2	90.0	2620.3	2434.5	400*	400*
3	90.0	2161.2	1975.4	400*	400*
4	90.0	1917.1	1732.4	277	400*
5	90.0	1747.7	1544.5	208	400*
6	90.0	1603.4	1541.1	199	189
7	60.0	1680.4	1365.0	160	281
8	60.0	1383.7	1234.0	187	400*
9	60.0	1252.3	1156.9	400*	400*
10	60.0	1206.3	1112.2	311	400*

* indicates no convergence in specified number of global iterations

Table 11: Optimal weight using sequential appproximations solution algorithm with a physical state-space realization, for direct output feedback with $H = G^T$, plus node 1 displacements and velocities

α^2	initial dv's	β^2	
($\times 10^{-5}$)	(structural)	50	60
1	120.0	3516.5	3232.6
2	90.0	2402.4	2213.0
3	90.0	1935.5	1790.2
4	90.0	1671.5	1543.3
5	90.0	1492.9	1379.2
6	90.0	1364.6	1261.6
7	60.0	1459.5	1166.3
8	60.0	1184.0	1093.4
9	60.0	1116.8	1186.1
10	60.0	1063.4	983.3

Table 12: Optimal weight using continuation solution algorithm with a physical state-space realization, for direct output feedback with $H = G^T$ plus node 1 displacements and velocities

Using Deflation in the Pole Assignment Problem with Output Feedback

George Miminis

Department of Computer Science, Memorial University of Newfoundland

St. John's, Newfoundland, Canada A1C 5S7

Abstract

A direct algorithm is suggested for the computation of a linear output feedback for a multi input, multi output system such that the resultant closed-loop matrix has eigenvalues that include a specified set of eigenvalues. The algorithm uses deflation based on unitary similarity transformations. Thus we hope the algorithm is numerically stable, however, this has not been proven as yet.

1 Introduction

Deflation is a technique that has been efficiently used in the solution of the standard eigenvalue problem of a matrix A as well as other eigenvalue related problems. According to this technique once an eigenpair (λ_1, x_1) of A is computed, we continue the process with a matrix that possesses only the remaining eigenvalues of A, and possibly a zero eigenvalue in the place of λ_1. In this way we are left to solve a smaller problem. Deflation can be accomplished by a variety of algorithms. Some of them are, Hoteling's deflation, Wielandt's deflation, deflation based on similarity transformations, deflation by restriction, etc. An excellent review of the first three methods can be found in [7, pp. 584-600], whereas deflation by restriction can be found in [5, p. 84]. Lately, deflation has been used in the solution of eigenassignment problems. For example, Wielandt's deflation is used in [6] to solve the partial eigenvalue allocation problem with state feedback for continuous time systems. In this paper we will be concerned with deflation based on unitary similarity transformations, and how it can be used in the pole assignment problem with output feedback, or as it is often known, the eigenvalue allocation problem with output feedback (MEVAO).

In section 2 we define the MEVAO. In section 3 we define transmission zeros and discuss how they affect the MEVAO. In section 4 we give an algorithm for the solution of the MEVAO. Finally in section 5 give some numerical examples to demonstrate the performance of our algorithm.

Lower case Greek letters represent scalars, upper case Roman represent matrices, while lower case Roman represent column vectors and indices. Superscripts T, H indicate transpose and conjugate transpose respectively. The notation k = i(r)j means that k takes all the values starting from i until j with step r. e_i represents the ith column of the identity matrix I, and $\lambda(\cdot)$ the set of eigenvalues of a matrix, counting multiplicities. $\mathbf{R}(\cdot)$, $\mathbf{N}(\cdot)$ will represent the space spanned by the columns of a matrix, and the null space of a matrix respectively. $\boldsymbol{R}, \mathbf{C}$ will represent the set of real and complex numbers respectively.

2 The Problem

The MEVAO problem that we will attack in this paper may be defined as follows:

We are given a controllable and observable system

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{1}$$

$$y(t) = Cx(t) \tag{2}$$

where $A \in \boldsymbol{R}^{n\times n}$, $B \in \boldsymbol{R}^{n\times m}$, $C \in \boldsymbol{R}^{p\times n}$, rank (B)=m, rank(C)=p, $x(t) \in \boldsymbol{R}^n$ is the state of the system at time t and u(t) the input at time t. We are also given a self conjugate set of eigenvalues λ_i, $i = 1(1)r$ with r=min{n,m+p-1}, we need to compute $K \in \boldsymbol{R}^{m\times p}$ such that $\lambda(A - BKC) \supseteq \{\lambda_i \mid i = 1(1)r\}$. As a result, a linear output feedback $u(t) = -Ky(t)$ may be computed that will make (1) become

$$\dot{x}(t) = (A - BKC)x(t).$$

The resulting system will have desirable properties for the contror engineer.

3 Transmission Zeros

In the remaining of the paper unless otherwise stated, we will assume without loss of generality that $m \leq p$.

Transmission zeros (*trzs*) play an important role in the MEVAO, thus it is vital that they are well understood. Our experience with the literature on *trzs* has been rather disappointing. In our search for a definition we have found several; some of them even contracting one another. Thus we

decided to resort to the physical motivation of *trzs*, and from that to derive a sensible definition. Some observations that may give some physical motivation to the concept of a transmission zero (of a controllable-observable system in our case) were found in [2], p.41. From this we concluded that, physically, a transmission zero (*trz*) of a system is a specific frequency of the system for which there exists a nonzero input that will yield a zero output. This basically is definition 1 below. In the following we will present three definitions of *trzs*, and we will prove their equivalence. The reason we present these three defintions is because the first is directly derived from the physical motivation of *trzs*, the second is very useful in matrix computations, and the third is frequently encountered in the literature.

Definition 1: A number $\zeta \in \mathbf{C}$ is a *trz* of (1),(2) if and only if there exists nonzero input u that can yield a zero output $y = G(\zeta)u = 0$, where

$$G(s) = C(sI - A)^{-1}B \quad , with \quad s \in \mathbf{C}$$

is the transfer function of (1),(2). □

The number of *trzs* of (1),(2) can be at most $n - max\{m, p\}$ (see for example [2], p.65). The set of all *trzs* of (1),(2) will be denoted by Z_{tr}. Often it is helpful to use the following definition of a *trz*.

Definition 2: $\zeta \in \mathbf{C}$ is a *trz* of (1),(2) if and only if

$$\exists u \neq 0 \ : \begin{pmatrix} A - \zeta I & B \\ C & 0 \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix} = 0 \quad with \quad x = (\zeta I - A)^{-1}Bu \ .$$

□

Frequently we encounter in the literature the following definition of *trzs*.

Definition 3: Let $L^{-1}(\zeta)G(\zeta)R^{-1}(\zeta) = \begin{pmatrix} \frac{\epsilon_1(\zeta)}{\psi_1(\zeta)} & & & \\ & \ddots & & \\ & & \frac{\epsilon_{m-1}(\zeta)}{\psi_{m-1}(\zeta)} & \\ & & & \frac{\epsilon_m(\zeta)}{\psi_m(\zeta)} \\ & & 0 & \end{pmatrix}$ be the Smith-McMillan (S-M) form of $G(\zeta)$, where $L(\zeta)$, $R(\zeta)$ are polynomial matrices and we have assumed that G(s) has normal rank m. Then ζ is a *trz* of (1) if and only if there exists $i = 1, \cdots, m$ such that $\epsilon_1(\zeta) = 0$. □

In the S-M form of $G(\zeta)$ the following properties hold

$$\epsilon_1(\zeta) \mid \epsilon_2(\zeta) \mid \cdots \mid \epsilon_m(\zeta) \quad \textit{and} \quad \psi_m(\zeta) \mid \psi_{m-1}(\zeta) \mid \cdots \mid \psi_1(\zeta) \,.$$

Because of the above properties we see that there cannot be $trz\ \zeta$ that will also be a root of $\psi_m(\zeta)$. Since, if it were then $\epsilon_m(\zeta) = 0$ and ζ would have been cancelled in $\epsilon_m(\zeta) \mid \psi_m(\zeta)$. Hence ζ would not be a trz, which is a contradiction. The roots of $\psi_i(\zeta)$, $i = 1(1)m$ are the poles of (1),(2), counting multiplicities. In our attempt to prove the equivalence of the above definitions we will be using the expression $G(\zeta)u$ with $\zeta \in Z_{tr}$. One then may wonder about the existance of $G(\zeta)u$ when ζ is also a pole of (1),(2). The following lemma will help us clarify this point.

Lemma 1: Let $\zeta \in \mathbf{C}$ be a trz as well as a pole of (1),(2), then there exists $u \neq 0$ such that $\lim_{s\to\zeta} G(s)u$ exists.

Proof: Let ζ be a pole as well as a trz of (1), then from the S-M form of $G(\zeta)$ we have that

$$\exists (i,j) \quad with \quad i < j \quad : \quad \psi_i(\zeta) = \epsilon_j(\zeta) = 0 \,.$$

Let us now for illustration assume $p = m = 3$ and $\psi_1(\zeta) = \epsilon_2(\zeta) = 0$. Then by taking $\tilde{u}^T = (0, \eta, \theta) \neq 0$ we have the following from the S-M form of G(s).

$$\begin{aligned} \lim_{s\to\zeta} L^{-1}(s)G(s)R^{-1}(s)\tilde{u} &= \begin{pmatrix} \lim_{s\to\zeta} \frac{\epsilon_1(s)}{\psi_1(s)} & & \\ & \lim_{s\to\zeta} \frac{\epsilon_2(s)}{\psi_2(s)} & \\ & & \lim_{s\to\zeta} \frac{\epsilon_3(s)}{\psi_3(s)} \end{pmatrix} \begin{pmatrix} 0 \\ \eta \\ \theta \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ 0 \\ \theta \lim_{s\to\zeta} \frac{\epsilon_3(s)}{\psi_3(s)} \end{pmatrix} \end{aligned}$$

which is defined since $\psi_3(\zeta) \neq 0$. Since $L(s)$, $R(s)$ are nonsingular (they have determinants independent of s, see for example [7], p.21), $\lim_{s\to\zeta} L^{-1}(s)$, $\lim_{s\to\zeta} R^{-1}(s)$ are defined.

Hence if we take $u = \lim_{s\to\zeta} R^{-1}(s)\tilde{u}$ then $\lim_{s\to\zeta} G(s)u$ is defined for every $\zeta \in Z_{tr}$ that is also a pole of (1),(2). □

In the following when we write $G(\zeta)u$ with $\zeta \in Z_{tr}$ also being a pole of (1), we will mean $\lim_{s\to\zeta} G(s)u$. Similarly when we write $x = (\zeta I - A)^{-1}Bu$ we will mean $x = \lim_{s\to\zeta}(sI - A)^{-1}Bu$, when $\zeta \in Z_{tr}$ is also a pole of (1),(2).

Theorem 1: Definitions 1,2 and 3 are equivalent.

Proof: To prove that definition 1 is equivalent to definition 2 we proceed as follows:

$$\begin{aligned}
\zeta \in Z_{tr} &\overset{\text{def}_1}{\iff} \{\exists\, u \neq 0 \;:\; C(\zeta I - A)^{-1}Bu = 0\} \\
&\iff \{\exists\, u \neq 0 \;:\; Cx = 0 \quad with \quad x = (\zeta I - A)^{-1}Bu\} \\
&\iff \left\{\exists\, u \neq 0 \;:\; \begin{pmatrix} A - \zeta I & B \\ C & 0 \end{pmatrix}\begin{pmatrix} x \\ u \end{pmatrix} = 0 \quad with \quad x = (\zeta I - A)^{-1}Bu\right\} \\
&\overset{\text{def}_3}{\iff} \zeta \in Z_{tr}\ .
\end{aligned}$$

To prove the equivalence of definitions 1 and 3 we proceed as follows:

$$\begin{aligned}
\zeta \in Z_{tr} &\overset{\text{def}_1}{\iff} \{\exists\, u \neq 0 \;:\; G(\zeta)u = 0\} \\
&\iff \left\{\exists\, u \neq 0 \;:\; L(\zeta)\underbrace{\begin{pmatrix} \operatorname{diag}_{i=1(1)m} \frac{\epsilon_i(\zeta)}{\psi_i(\zeta)} \\ 0 \end{pmatrix}}_{M(\zeta)} R(\zeta)u = 0\right\} \qquad (3)
\end{aligned}$$

Since $L(s)$, $R(s)$ are nonsingular for $\forall s \in \mathbf{C}$, $M(\zeta)$ must not have full column rank in (2). Hence

$$\begin{aligned}
(3) &\iff \{\exists\, i \in \{1,2,\cdots,m\} \;:\; \epsilon_i(\zeta) = 0\} \\
&\overset{\text{def}_3}{\iff} \zeta \in Z_{tr}\ .
\end{aligned}$$

□

In the last theorem we silently assumed that the normal rank q of $G(s)$ is m, or equivalently the normal rank $n+q$ of $P(s) = \begin{pmatrix} A - sI & B \\ C & 0 \end{pmatrix}$ is $n+m$. However, theorem 1 holds even for the degenerate case where $q < min\{m,p\}$ $(= m$ according to our assumption). To show this we proceed as follows:

First we prove $Z_{tr} = \mathbf{C}$ using definition 1. Since $q < m$ the following is true

$$\{\forall \zeta \in \mathbf{C} \quad \exists u \neq 0 \quad : \quad G(\zeta)u = 0\} \overset{\text{def}_1}{\Longleftrightarrow} Z_{tr} = \mathbf{C}\,. \tag{4}$$

To prove the equivalence of definitions 1 and 2 we see

$$Z_{tr} = \mathbf{C} \overset{(4)}{\Longleftrightarrow} \left\{ \forall \zeta \in \mathbf{C} \quad \exists u \neq 0 \quad : \quad \begin{pmatrix} A - \zeta I & B \\ C & 0 \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix} = 0 \quad with \quad x = (\zeta I - A)^{-1} B u \right\}$$

To prove the equivalence of definitions 1 and 3 we see

$$\begin{aligned} Z_{tr} = \mathbf{C} \quad &\overset{(4)}{\Longleftrightarrow} \quad \left\{ \forall \zeta \in \mathbf{C} \quad \exists u \neq 0 \quad : \quad L(\zeta) \begin{pmatrix} \underset{i=1(1)m}{\text{diag}} \frac{\epsilon_i(\zeta)}{\psi_i(\zeta)} \\ 0 \end{pmatrix} R(\zeta) u = 0 \right\} \\ &\Longleftrightarrow \quad \{\forall \zeta \in \mathbf{C} \quad \exists i \in \{1, \cdots, m\} \; : \; \epsilon_i(\zeta) = 0\}. \end{aligned}$$

Hence another definition of *trzs* that is often found in literature, and according to which $\zeta \in Z_{tr}$ if and only if $G(\zeta)$ and $P(\zeta)$ go below their normal rank, does not agree in the degenerate case with definitions 1, 2 and 3. This is so because not $\forall \zeta \in \mathbf{C}$ will make $G(\zeta)$ and $P(\zeta)$ go below their normal rank, hence $Z_{tz} \neq \mathbf{C}$, according to this definition. In what follows we will assume that *trzs* are defined by definitions 1, 2 or 3.

Definition 4: A number $\zeta \in \mathbf{C}$ is a strong transmission zero (*strz*) of (1),(2) if and only if $G(\zeta) = 0$, or equivalently $\epsilon_1(\zeta) = 0$ (in the $S - M$ form of $G(\zeta)$), or equivalently

$$\left\{ \forall u \neq 0\,, \quad \begin{pmatrix} A - \zeta I & B \\ C & 0 \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix} = 0 \;, \quad with \quad x = (\zeta I - A)^{-1} B u \right\}.$$

□

Theorem 2: $\zeta \in \mathbf{C}$ cannot be assigned by output feedback if and only if ζ is a strong transmission zero.

Proof: Let $G(\zeta) \neq 0$, then if $\zeta \in \lambda(A)$ we take $K = 0$ and ζ is placed. If $\zeta \notin \lambda(A)$ then there exists $u \neq 0$ such that

$$y = G(\zeta)u \neq 0 \quad \Rightarrow \left\{ \begin{array}{l} x = (\zeta I - A)^{-1} B u \\ y = Cx \neq 0 \end{array} \right\}.$$

If we take $K = -\frac{uy^T}{y^Ty}$ then we see that

$$\begin{aligned}[\zeta I - (A - BKC)]\, x &= (\zeta I - A)x + BKCx \\ &= Bu - B\frac{uy^T}{y^Ty}\, y \;\; = 0\end{aligned}$$

Thus the eigenpair (ζ, x) has been placed, which proves the necessity of the propositionn given by the theorem. To prove its sufficiency assume $G(\zeta) = 0$, but nevertheless ζ has been placed. Then there exists $x \neq 0$ such that for some K

$$\begin{aligned}(A - \zeta I)x - BKCx = 0 \;\; &\Rightarrow \;\; x - (A - \zeta I)^{-1}BKCx = 0 \\ &\Rightarrow \;\; Cx - C(A - \zeta I)^{-1}BKCx = 0 \\ &\Rightarrow \;\; Cx + G(\zeta)KCx = 0 \\ &\Rightarrow \;\; Cx = 0\ .\end{aligned}$$

Hence from $(A - \zeta I)x - BKCx = 0 \;\Rightarrow\; (A - \zeta I)x = 0 \;\Rightarrow\; \zeta \in \lambda(A)$. This is a contradiction, since if $\zeta \in \lambda(A)$ the $G(\zeta)$ is not defined, however it is assumed that $G(\zeta) = 0$. □

4 The Algorithm

We give an algorithm based on deflation by unitary similarity transformations.

Step 1: Compute a feasible eigenvector x_1 of $A_1 - B_1K_1C_1 \;\equiv\; A - BKC$ corresponding to, say λ_1 with $\| x_1 \|_2 = 1$. We will show in the sequel how we may compute such an x_1.

Step 2: Allocate the eigenpair (λ_1, x_1) to $A_1 - B_1K_1C_1$ as follows: Let V_1 be a unitary transformation such that $V_1^HC_1x_1 = \delta_1 e_1$ and $B_1 = (U_0, U_1)\begin{pmatrix} R_1 \\ 0 \end{pmatrix}$ a QR decomposition of B_1, then

$$\begin{aligned}(A_1 - B_1K_1C_1)x_1 = \lambda_1 x_1 \;\; &\Longleftrightarrow \;\; A_1x_1 - B_1(K_1V_1)(V_1^HC_1x_1) \;=\; \lambda_1 x_1 \\ &\Longleftrightarrow \;\; B_1\tilde{K}_1\delta_1 e_1 \;=\; (A_1 - \lambda_1 I)x_1\ , \quad with \quad \tilde{K}_1 = K_1V_1 \\ &\Longleftrightarrow \;\; B_1k_1\delta_1 = (A_1 - \lambda_1 I)x_1\ , \quad with \quad k_1 = \tilde{k}_1 e_1 \\ &\Longleftrightarrow \;\; \begin{pmatrix} R_1 \\ 0 \end{pmatrix} k_1\delta_1 \;=\; \begin{pmatrix} U_0^H(A_1 - \lambda_1 I)x_1 \\ U_1^H(A_1 - \lambda_1 I)x_1 \end{pmatrix} \qquad (5)\end{aligned}$$

From (5) we see that if x_1 was computed in step 1 so that $x_1 \in \mathrm{N}[U_1^T(A_1 - \lambda_1 I)]$, and k_1 is computed by solving

$$R_1 k_1 = \delta_1^{-1} U_0^H (A_1 - \lambda_1 I) x_1 \tag{6}$$

then (5) is satisfied and the eigenpair (λ_1, x_1) is allocated. An eigenvector x_1 that satisfies $x_1 \in \mathrm{N}[U_1^T(A_1 - \lambda_1 I)]$ will be called feasible.

Step 3: Compute unitary $Q_1 = (x_1, \tilde{Q}_1)$. Hence from $Q_1 e_1 = x_1 \Rightarrow Q_1^H x_1 = e_1$ we see that Q_1 can be a Householder transformation or a sequence of rotations.

Step 4: Perform the unitary similarity transformation

$$\begin{aligned} Q_1^H(A_1 - B_1 K_1 C_1)Q_1 &= \begin{pmatrix} x_1^H \\ \tilde{Q}_1^H \end{pmatrix} [A_1 - B_1 \tilde{K}_1 (V_1^H C_1)](x_1, \tilde{Q}_1) \\ &= \begin{pmatrix} x_1^H \\ \tilde{Q}_1^H \end{pmatrix} [(A_1 x_1 - B_1 k_1 \delta_1)\,,\, (A_1 \tilde{Q}_1 - B_1 \tilde{K}_1 (V_1^H C_1)\tilde{Q}_1)] \end{aligned} \tag{7}$$

From step 3, k_1 has been computed so that $A_1 x_1 - B_1 k_1 \delta_1 = \lambda_1 x_1$. If we now set $\tilde{K}_1 = (k_1, K_2)$ and

$$\begin{aligned} V_1^H C_1 \tilde{Q}_1 &= \begin{pmatrix} c_1^H \\ C_2 \end{pmatrix} \quad \textit{then} \\ (7) &= \begin{pmatrix} x_1^H \\ \tilde{Q}_1^H \end{pmatrix} [\lambda_1 x_1 \;,\; A_1 \tilde{Q}_1 - B_1(k_1 c_1^H + K_2 C_2)] \\ &= \left(\begin{array}{c|c} \lambda_1 & x_1^H [A_1 \tilde{Q}_1 - B_1(k_1 c_1^H + K_2 C_2)] \\ \hline & \tilde{Q}_1^H A_1 \tilde{Q}_1 - \tilde{Q}_1^H B_1 k_1 c_1^H - \tilde{Q}_1^H B_1 K_2 C_2 \end{array}\right) \\ &= \left(\begin{array}{c|c} \lambda_1 & * \\ \hline & A_2 - B_2 K_2 C_2 \end{array}\right) \end{aligned}$$

where $A_2 = \tilde{Q}_1^H A_1 \tilde{Q}_1 - \tilde{Q}_1^H B_1 k_1 c_1^H$, $B_2 = \tilde{Q}_1^H B_2$.

Step 5: Continue the allocations with $A_2 - B_2 K_2 C_2$. □

There are two points that need to be clarified for the allocation of λ_1 in the above algorithm. One is, how to identify whether λ_1 is a *strz* or not, and the other is the computation of x_1.

Lemma 2: If $\{\lambda_1$ is a *strz* of (1) $\} \Rightarrow \delta_1 = 0$.

Proof: Let λ_1 be a *strz* of (1) then

$$\left\{\forall u_1 \neq 0 \ \exists x_1 \ : \ \begin{pmatrix} A_1 - \lambda_1 I & B_1 \\ C_1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ u_1 \end{pmatrix} = 0 \right\} \Rightarrow \ C_1 x_1 = 0 \ \Rightarrow \ \delta_1 = 0 \, .$$

We may note however that the above lemma on its own is not adequate to identify *strz*. In the sequel we will compute our feasible eigenvector x_1 in such a way that if $\delta_1 = 0$ then λ_1 will be a *strz*. As we will show, computing x_1 in this way will also improve the numerical stability of our algorithm. To accomplish the above we proceed as follows:

We know that

$$| \, \delta_1 \, | = \| \, C_1 x_1 \, \|_2 \Rightarrow \ \{\delta_1 = 0 \iff x_1 \in \mathbf{N}(C_1)\} \ .$$

Thus if we compute x_1 so that

$$x_1 \in \mathbf{N}[U_1^H (A_1 - \lambda_1 I)] - \mathbf{N}(C_1)$$

when

$$\mathbf{N}[U_1^H (A_1 - \lambda_1 I)] - \mathbf{N}(C_1) \neq \{\}$$

then we may safely conclude that $\delta_1 = 0 \ \Rightarrow \ \{\lambda_1$ is a *strz* $\}$. If the dimension of

$$\mathbf{N}[U_1^H (A_1 - \lambda_1 I)] - \mathbf{N}(C_1)$$

is greater than one, we have some freedom in choosing our x_1. To see how we may exploit this freedom, we observe from (6) that if $| \, \delta_1 \, |$ is unreasonably small then we may unnecessarily lose accuracy when we solve (6). To avoid this, let

$$C_1^T = (H_0, H_1) \begin{pmatrix} Z_1 \\ 0 \end{pmatrix}$$

be a QR decomposition of C_1^T, then $\mathbf{R}(H_0) = \mathbf{R}(C_1^T)$ and $\mathbf{R}(H_1) = \mathbf{N}(C_1)$. Assuming now $\| \, x_1 \, \|_2 = 1$ we get

$$| \, \delta_1 \, | = \| \, C_1 x_1 \, \|_2 = \| \, Z_1^T H_0^T x_1 \, \|_2 \leq \| \, Z_1 \, \|_2 \| \, H_0^T x_1 \, \|_2 \, .$$

But $\| \, H_0^T x_1 \, \|_2 = \sigma_{max}(H_0^T x_1) = cos\theta$, $\sigma_{max}(\cdot)$ is the maximum singular value of a matrix, and θ is the smallest angle between $\mathbf{R}(C_1^T)$ and $\mathbf{R}(x_1)$ (see[1]). Hence from

$$| \delta_1 | \leq \| z \|_2 \cos\theta$$

in order to make $| \delta_1 |$ as large as possible we need to make θ as small as possible. Thus we will choose $x_1 \in \mathbf{N}[U_1^H(A_1 - \lambda_1 I)]$ that forms an angle θ with $\mathbf{R}(H_0)$ as close to zero as possible. The following theorem taken from [1,p.582] and modified to meet our requirements, will give us a way to compute θ as it was required above.

Theorem 3: Let H_0 and W_1 form unitary bases of two subspaces. Let also

$$H_0^H W_1 = Y \Sigma X^H$$

be the singular value decomposition of $H_0^H W_1$, then the smallest angle between $\mathbf{R}(H_0$ and $\mathbf{R}(W_1)$ takes place between vectors $H_0 Y e_1$ and $W_1 X e_1$.

Proof: See [1]

Theorem 3 suggests the following algorithm for the computation of x_1:

Compute the QR decomposition of C_1^T, B_1, and $[U_1^H(A_1 - \lambda_1 I)]^H = (W_0, W_1) \begin{pmatrix} T \\ 0 \end{pmatrix}$.

Compute the singular value decomposition of $V_0^H W_1 = Y \Sigma X^H$.

Take $x_1 = W_1 X e_1$.

Finally to eliminate any doubt that $\delta_1 = 0$ when λ_1 is a *strz* we prove the following lemma.

Lemma 3: $\mathbf{N}[U_1^H(A_1 - \lambda_1 I)] \subseteq \mathbf{N}(C_1) \iff \{\lambda_1$ is a *strz* of (1),(2)$\}$

Proof: First we prove

$$\mathbf{N}[U_1^H(A_1 - \lambda_1 I)] = \{(\lambda_1 I - A_1)^{-1} B_1 u_1 \mid u_1 \in \mathbb{R}^m\} . \tag{8}$$

To do so, we see that, given $u_1 \neq 0$ and computing x_1 to satisfy

$$(A_1 - \lambda_1 I)x_1 = -B_1 u_1 \iff \begin{pmatrix} U_0^H \\ U_1^H \end{pmatrix} (A_1 - \lambda_1 I)x_1 = \begin{pmatrix} -R_1 \\ 0 \end{pmatrix} u_1$$
$$\Rightarrow \quad x_1 \in \mathbf{N}[U_1^H(A_1 - \lambda_1 I)].$$

Thus

$$\{\forall u_1 \in \mathbb{R}^m , (\lambda_1 I - A_1)^{-1} B_1 u_1 \in \mathbf{N}[U_1^H(A_1 - \lambda_1 I)]\} \Longrightarrow$$

$$\{(\lambda_1 I - A_1)^{-1} B_1 u_1 \mid u_1 \in \mathbb{R}^m\} \subset \mathbf{N}[U_1^H (A_1 - \lambda_1 I)] \,. \tag{9}$$

Let $x_1 \,:\, U_1^H (A_1 - \lambda_1 I) x_1 = 0$, then, since we can always compute

$$u \in \mathbb{R}^m \,:\, R_1 u_1 = U_0^H (A_1 - \lambda_1 I) x_1 \,,$$

we have $\begin{pmatrix} U_0^H \\ U_1^H \end{pmatrix} (A_1 - \lambda_1 I) x_1 = \begin{pmatrix} -R_1 \\ 0 \end{pmatrix} u_1 \;\Rightarrow\; x_1 = (\lambda_1 I - A_1)^{-1} B_1 u_1 \,,$

from which we get

$$\mathbf{N}[U_1^H (A_1 - \lambda_1 I)] \subset \{(\lambda_1 I - A_1)^{-1} B_1 u_1 \mid u_1 \in \mathbb{R}^m\} \,. \tag{10}$$

From (9), (10) we may derive (8).

Suppose now that

$$\begin{aligned} \{\lambda_1 \textit{ is a strz of } (1),(2)\} &\iff \{\forall u_1 \in \mathbb{R}^m \;\; C_1 (\lambda_1 I - A_1)^{-1} B_1 u_1 = 0\} \\ &\iff C_1 \mathbf{N}[U_1^H (A_1 - \lambda_1 I)] = 0 \\ &\iff \mathbf{N}[U_1^H (A_1 - \lambda_1 I)] \subset \mathbf{N}(C_1) \,. \end{aligned}$$

□

The algorithm that has been described so far in this section, can be used to allocate only one eigenvalue. We will show however that we may use it to allocate $min\{n, m+p-1\}$ eigenvalues. To do so we need to observe that only the first column of the current K_i is needed for the allocation of one eigenvalue. We also need to consider the fact that $\lambda(A_i - B_i K_i C_i) = \lambda(A_i^T - C_i^T K_i^T B_i^T)$. Given these two points then we may use the following algorithm, which for illustration we describe for $m = 2$ and $p = 3$, thus

$$K_1 = \begin{pmatrix} \times & \times & \times \\ \times & \times & \times \end{pmatrix} \,.$$

We allocate eigenvalues until the number of rows of the current K_i become greater than the number of columns. In our example this happens after the allocation of two eigenvalues, hence

$$K_3 = \begin{pmatrix} \times \\ \times \end{pmatrix} \,.$$

At this point we continue working with the transposed system $A_3^T - C_3^T K_3^T B_3^T$. Since K_3^T has two columns we are able to allocate two more eigenvalues instead of just one that we would have allocated if we had continued with K_3. In the general case we keep transposing until we run out of eigenvalues or columns. Thus the total number of eigenvalues that we manage to allocate using this algorithm is

$$m + p - 1 = 4 \ .$$

Note that by following this algorithm we also satisfy the condition $m \leq p$ at the beginning of each allocation. This condition has been assumed throughout this paper. Note that m, p are associated with the columns of the matrix on the left of K_i or K_i^T and the rows of the matrix on the right of K_i or K_i^T respectively, rather than with B_i and C_i.

5 Numerical Examples

In this section we give two numerical examples to demonstrate the performance of our algorithm. The computation was performed on double precision (56-bit mantissa) using PC-MATLAB on a Toshiba T5100 which uses an 80387 coprocessor equipped with the IEEE floating point standard of arithmetic. In the computation below we show, up to 12 decimal digits of accuracy.

Example 1:

$$A = \begin{pmatrix} 0.581314086914 & 0.504058837890 & 0.559921264648 & 0.741333007812 & 0.303421020507 \\ 0.166717529296 & 0.157394409179 & 0.665222167968 & 0.933609008789 & 0.144439697265 \\ 0.353500366210 & 0.441650390625 & 0.373367309570 & 0.771942138671 & 0.078048706054 \\ 0.836242675781 & 0.200820922851 & 0.072296142578 & 0.598373413085 & 0.638671875000 \\ 0.244094848632 & 0.936126708984 & 0.607955932617 & 0.154357910156 & 0.154678344726 \end{pmatrix}$$

$$B^T = \begin{pmatrix} 0.549163818359 & 0.843856811523 & 0.178649902343 & 0.409255981445 & 0.595489501953 \\ 0.942672729492 & 0.008666992187 & 0.239028930664 & 0.142791748046 & 0.671371459960 \\ 0.613098144531 & 0.065872192382 & 0.300445556640 & 0.576828002929 & 0.726318359375 \end{pmatrix}$$

$$C = \begin{pmatrix} 0.561660766601 & 0.332702636718 & 0.355514526367 & 0.279266357421 & 0.531448364257 \\ 0.427825927734 & 0.648269653320 & 0.666625976562 & 0.572494506835 & 0.972076416015 \\ 0.455917358398 & 0.134307861328 & 0.497573852539 & 0.212463378906 & 0.653732299804 \end{pmatrix}$$

The eigenvalues to be allocated are

{0.704589843750, 0.421768188476, 0.572113037109, 0.396102905273, 0.127380371093}.

The K computed by our algorithm was found to be

$$K = \begin{pmatrix} -9.415631257941 & -1.520241736469 & 11.520696978251 \\ -36.155174309311 & -4.515055559417 & 41.184483490595 \\ 33.590525809043 & 4.071903303950 & -36.923138190434 \end{pmatrix}$$

The eigenvalues of $A - BKC$ were computed and they were $\lambda(A - BKC) =$ {0.704589843749, 0.421768188479, 0.572113037109, 0.396102905270, 0.127380371093}.

Example 2: In the above example we had $n = m + p - 1$, thus all n eigenvalues were allocated. However, if $m + p - 1 < n$ then $n - (m + p - 1)$ eigenvalues of $A - BKC$ will take values that our algorithm has absolutely no control over. The following example demonstrates this point.

$$A = \begin{pmatrix} 0.356336616284 & -0.356626507847 & -1.198870998736 \\ -0.210549376245 & 2.165165719183 & -0.882324378697 \\ -0.355939324751 & -0.506195941988 & 0.721098719251 \end{pmatrix}$$

$$B = \begin{pmatrix} 1.606955436561 & -0.407338058168 \\ 0.062823512419 & -0.595038063525 \\ -1.611627967397 & 0.616657720777 \end{pmatrix}$$

$$C = \begin{pmatrix} -1.182820557312 & 0.343687857622 & -0.357421120340 \end{pmatrix}$$

The eigenvalues to be allocated are {-0.406648486670, -0.366406484747, 0.853280016421}.

The K computed by our algorithm was found to be

$$K = \begin{pmatrix} -2.053475126112 \\ -6.466096811958 \end{pmatrix}$$

The eigenvalues of $A - BKC$ were computed and they were $\lambda(A - BKC)$ = {-0.406648486670, -0.366406484747, 1.707616832510}.

Our algorithm allocates one eigevalue at a time, thus complex eigenvalues need complex arithmetic. As a result, K may be complex. Investigation is under way to derive an algorithm that will allocate two eigenvalues at a time in a double step. In this way we will allocate a complex conjugate pair of eigenvalues in one double step using only real arithmetic. Hence K will be real.

6 Conclusion

We presented an algorithm for the pole assignment problem for multi-input, multi-output systems using output feedback. The algorithm uses deflation based on unitary similarity transformations and it allocates $min\{n, m + p - 1\}$ eigenvalues. The same kind of deflation has been used in [3] to solve the corresponding pole assignment problem using state feedback. Since the algorithm in [3] has been proven to be numerically stable we hope the algorithm in this paper has the same property too. However, this needs to be proven by doing a rounding error analysis of the algorithm, and we plan to do this in the near future.

References

[1] Bjorck, A. & Golub, G.H., "Numerical Methods for Computing Angles Between Linear Subspaces", Mathematics of Computation, v.27, No.123, pp. 579-594, 1973.

[2] Macfarlane, A.G.J. & Karcanias, N., "Poles and Zeros of Linear Multivariable Systems: A Survey of the Algebraic, Geometric and Complex-Variable Theory", Int. J. Control, v.24, No.1, pp. 33-74, 1976.

[3] Miminis, G.S. & Paige, C.C., "A Direct Algorithm for Pole Assignment of Time Invariant multi-input Linear Systems using State Feedback", Automatica, v.24, No.3, pp. 343-356, 1988.

[4] Miminis, G.S., "The use of Deflation in Eigenassignment Problems", To appear in the Proceedings of the MTNS89 conference, Amsterdam, June 1989.

[5] Parlett, B., "The Symmetric Eigenvalue Problem", Prentice-Hall, 1980.

[6] Saad, Y., "Projection and Deflation Methods for Partial Pole Assignment in Liner State Feedback", IEEE Trans. Autom. Control, v.33, No.3, pp. 290-297, 1988.

[7] Wilkinson, J., "The Algebraic Eigenvalue Problem", Oxford, 1965.

N90-23005

COMBINED CONTROL-STRUCTURE OPTIMIZATION

M. Salama, M. Milman, R. Bruno, R. Scheid
Jet Propulsion Laboratory
California Institute of Technology

S. Gibson
University of California, Los Angeles

1. INTRODUCTION

The strong interaction between structural dynamics and active control is a well-recognized challenge in the analysis of controlled flexible structures. But it is only recently that the same interaction has been exploited in the design process. The traditional design approach in which the control design comes very late in the development — typically after the structure has been designed and built — is no longer viable. Although this approach has produced satisfactory results for the attitude control of relatively rigid space structures, it will not be suitable for the ambitious space missions that require precise controlled pointing of telescopes, interferometers and the vibration suppression for science instruments mounted on large flexible structures. In such systems, designing the structure and designing its control become entwined. This dictates early consideration of the control design — well before any detailed structural design is finalized. And just as the structure is optimally designed to meet such performance metrics as minimum mass or response to external disturbances, it should be optimally designed to meet its ultimate control performance as well.

A natural way to introduce the control element into the overall design process is through an optimization procedure that combines the structural and control design criteria into a single problem formulation. A number of authors [1-6] have taken this perspective. In terms of the types of design parameters and constraints considered, Ref. (2) is probably the most extensive in that the design variables include structure parameters, actuator locations and feedback matrix. Static output feedback is used, and the performance objectives include total mass and robustness measures. Constraints are imposed on the eigenvalue placements, performance bounds, and structural constraints. Since not all of the constraints are commensurate, they are relaxed using a homotopy approach. Just as with Ref. (6), the approach taken in the present paper is not to produce the "best" optimized point design, but to produce a family of Pareto optimal designs representing options that assist in early trade studies. The philosophy is that these are candidate designs to be passed on for further consideration, and their function is more to guide the system design rather than to represent the ultimate design.

An optimization approach that is consistent with this philosophy is to utilize multiple cost objectives that include an LQG cost criterion in conjunction with structural cost(s), and possibly other control related costs.

After introducing the combined objective formulation in the setting of vector optimization, we derive the necessary conditions for Pareto optimality. No behavioral constraints are explicitly imposed in the problem formulation and a homotopy approach is used to generate a family of optimum designs. The intent of this paper is to explore this design approach and to provide an exposition of the computational aspects of the problem using numerical examples.

2. COMBINED OPTIMIZATION

The combined optimization approach can be best appreciated when contrasted with the traditional sequential optimization. In the sequential optimization, the structure is first optimized by selecting the design variable, α (e.g. member sizes) which minimize a structural criterion $J_s(\alpha)$ - often taken as the mass of the structure subject to some behavioral constraints $h(\alpha)\geq 0$ on deformations, stresses, open-loop frequencies, etc.

$$\min_{\alpha} J_s(\alpha) \; ; \qquad h(\alpha)\geq 0, \; \alpha \epsilon D \tag{2.1}$$

where D is the physical domain for α. Second, having completely specified the optimal structural design α^*, the control optimization is carried out with α^* fixed. For example, LQG or H_∞ optimal control designs pose the problem:

$$\min_{C} J_c(\alpha^*, C) \tag{2.2}$$

where J_c represents either of the control criterion, and C is allowed to vary over the class of stabilizing compensators for the plant.

By contrast, in the combined optimization formulation, the goal is to first merge the criteria of interest (here J_s and J_c) into one using non-negative multipliers β, and δ, then optimize the combined criterion over the original design variable space α, C:

$$\min_{\alpha, C} [\beta J_s(\alpha) + \delta J_c(\alpha, C] \tag{2.3}$$

The following expression compares the results of the two optimization procedures outlined above.

$$\min_{\alpha, C} [\beta J_s(\alpha) + \delta J_c(\alpha, C)] \leq [\min_{\alpha} \beta J_s(\alpha) + \min_{C} \delta J_c(\alpha^*, C)] \tag{2.4}$$

The right-hand side of (2.4) corresponds to performing the sequential optimization by solving (2.1) for α^*, followed by solving (2.2) for C*. Note that the optimal solution of the right-hand side is independent of β and δ- but not so for the combined optimization embodied by the left-hand side. In terms of the total cost, expression (2.4) states the fact that the combined optimization is never inferior to the sequential optimization. In the vector optimization setting, the relative weighting of β and δ serves as a parameter that allows the generation of an entire family of Pareto optima.

In the present paper, only two objective criteria are dealt with. But it is not difficult to generalize the approach to incorporate other criteria such

as minimum open-loop frequency and certain controller robustness measures. In general these criteria are noncommensurate, and there is no unique solution that minimizes all criteria $J_1,\ldots,J_N$ simultaneously. Thus, one must look for Pareto optimal solutions as outlined below.

First one assembles the criteria $J_i:D\to R$, $i=1,\ldots,N$ into a single criterion $J:D\to R^N$, $J(\alpha)=(J_1(\alpha),\ldots,J_N(\alpha))^T$. Then the cone $C_o = \{x\epsilon R^N: x_i\geq 0,\ i=1,\ldots,N\}$ is defined to induce a partial ordering $\leq$ on R^N by $x\leq y$ if $y-x\ \epsilon C_o$. Now let $\alpha\epsilon D$. A design vector $\alpha^*\epsilon D$ is said to be (strongly) Pareto optimal if $J(\alpha)\leq J(\alpha^*)$ implies $J(\alpha)=J(\alpha^*)$. A necessary condition for Pareto optimality is contained in the following theorem due to Lin [7].

Theorem 2.1: If α^* is Pareto optimal for the combined criterion J, and D is an open set, then there exists a nonzero vector $Z\epsilon C_o$ such that $Z^T J_*(\alpha^*)=0$. Here J_* denotes the differential of J.

For the two-term optimization problem in (2.3), we find that the Pareto optimal solution to $J = (J_s,\ J_c)^T$ can be generated by solving for the necessary conditions for extremizing the following convex combination J_λ:

$$J_\lambda = (1-\lambda)\ J_s(\alpha) + \lambda J_c(\alpha,C);\quad \lambda\epsilon[0,1] \tag{2.5}$$

where λ replaces β, δ in (2.3). The form of Eq. (2.5) suggests a homotopy (or continuation) approach for generating a family of Pareto optima as λ is propagated from 0 to 1.

3. NECESSARY CONDITIONS

We begin with the n_s degree-of-freedom dynamical system

$$M(\alpha)\ddot{r} + D(\alpha)\dot{r} + K(\alpha)r = G_1u + G_2v \tag{3.1}$$

where M, D, and K are the $n_s x n_s$ mass, damping and stiffness, G_1 is the constant $n_s x n_u$ control influence matrix, and G_2 is the constant $n_s x n_d$ disturbance matrix. The vectors r, u, and v are respectively, physical degrees-of-freedom, control forces and disturbances. Let $x = (r,\dot{r})^T$. Then the first order state equation is

$$\dot{x} = A(\alpha)x + B_1(\alpha)u + B_2(\alpha)\ v + v' \tag{3.2}$$

where

$$A = \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{bmatrix},\quad B_1 = \begin{bmatrix} 0 \\ M^{-1}G_1 \end{bmatrix},\quad B_2 = \begin{bmatrix} 0 \\ M^{-1}G_2 \end{bmatrix} \tag{3.2a}$$

and an additional disturbance v' independent of α has been introduced for greater flexibility of the formulation. We assume that (3.2) has measured output variables y and controlled output variables z:

$$\begin{aligned} y &= C_1x + w, \\ z &= C_2x \end{aligned} \tag{3.3}$$

and that v, v', and w are uncorrelated white noise processes with intensities Q_v, V, and Q_w, respectively.

In the remainder of this paper, the total mass of the structure is assumed to represent the structural criterion J_s. Thus, for a structure consisting of n_a one-dimensional finite elements, each having a cross-sectional area α_i, length ℓ_i and density ρ:

$$J_s = \sum_i^{n_a} \rho \ell_i \alpha_i \tag{3.4}$$

For the control criterion J_c associated with (3.2) and (3.3), we assume the LQG index

$$J_c = \lim_{t\to\infty} E\{z^T(t)\tilde{D}z(t) + x^T D_1 x + u^T(t)Ru(t)\} \tag{3.5}$$

where E is the expectation, $\tilde{D}$ and D_1 are non-negative definite weighting matrices, and R is positive definite. Although D_1 is assumed independent of α, D could possibly depend on α. The latter case would arise, for instance, if the first term in (3.5) were to represent the total energy in the system with $z = (r,\dot{r})^T$ and $D = \mathrm{diag}(K,M)$. Under standard assumptions of stabilizability and detectability, the optimal compensator C* for (3.5) is implemented by [8]:

$$\begin{aligned} u_o &= -B_1^T P x_o \\ \dot{x}_o &= (A-K_f C)x_o + K_f y + B_1 u_o, \end{aligned} \tag{3.6}$$

and the optimal cost J_c associated with this compensator is

$$J_c(C) = \mathrm{tr}\{P(B_2 Q_v B_2^T + V) + P_f P B_1 R^{-1} B_1^T P\}. \tag{3.7}$$

where P and P_f are the unique positive definite solutions to the algebraic Riccati equations

$$\begin{aligned} A^T P + PA + D_o - PB_1 R^{-1} B_1^T P &= 0, \\ AP_f + P_f A^T + B_2 Q_v B_2^T + V - P_f C_1^T Q_w^{-1} C_1 P_f &= 0 \end{aligned} \tag{3.8}$$

and $D_o = C_2^T \tilde{D} C_2 + D_1$, and $K_f = P_f C_1^T Q_w^{-1}$

With the above representation for J_c, we seek the optimality conditions for

$$\min_\alpha J_\lambda(\alpha) = (1-\lambda)J_s(\alpha) + \langle \nu, \alpha \rangle + \lambda \mathrm{tr}\{P(B_2 Q_v B_2^T + V) + P_f P B_1 R^{-1} B_1^T P\} \tag{3.9}$$

subject to the constraints in (3.8). In (3.9) an n_a-dimensional vector $\nu \epsilon R^{n_a}$ has been introduced to "regularize" the problem and to serve the purpose of initializing the homotopy path.

Proposition 3.1 Let Σ_+ denote the set of n_x xn_x positive definite matrices. The optimality conditions for α^* to be a local extremum of (3.9) subject to (3.8) are the zeros of the function H: RxR^{na} x Σ_+ x Σ_+ → R^{na} x Σ_+ x Σ_+ defined by $H = [H_1, H_2, H_3, H_4, H_5]$, where

$$H_1(\lambda,\alpha,Z_1,Z_2,P,P_f) =$$

$$(1-\lambda)\frac{\partial J_s}{\partial \alpha_i} + \nu_i + \lambda tr\{P \frac{\partial (B_2 Q_v B_2^T)}{\partial \alpha_i} + PP_fP \frac{\partial (B_1 R^{-1} B_1^T)}{\partial \alpha_i} + Z_1[\frac{\partial A^T}{\partial \alpha_i}P + P\frac{\partial A}{\partial \alpha_i} + \frac{\partial D_o}{\partial \alpha_i}$$

$$-P \frac{\partial (B_1 R^{-1} B_1^T)}{\partial \alpha_i} P] + Z_2[\frac{\partial A}{\partial \alpha_i} P_f + P_f \frac{\partial A^T}{\partial \alpha_i} + \frac{\partial (B_2 Q_v B_2^T)}{\partial \alpha_i} - P_f \frac{\partial (C_1^T Q_w^{-1} C_1)}{\partial \alpha_i} P_f]\},$$

$$i=1,\ldots,n_a \qquad (3.10a)$$

$$H_2(\lambda, \alpha, Z_1, Z_2, P, P_f) =$$

$$(A-B_1R^{-1}B_1^TP)Z_1+Z_1(A^T-PB_1R^{-1}B_1^T)+B_2Q_vB_2^T+P_fPB_1R^{-1}B_1^T+B_1R^{-1}B_1^TPP_f \qquad (3.10b)$$

$$H_3(\lambda, \alpha, Z_1, Z_2, P, P_f) =$$

$$(A^T-C_1^TQ_w^{-1}C_1P_f)Z_2+Z_2(A-P_fC_1^TQ_w^{-1}C_1)+PB_1R^{-1}B_1^TP \qquad (3.10c)$$

$$H_4(\lambda, \alpha, Z_1, Z_2, P, P_f) = A^TP+PA+D_o-PB_1R^{-1}B_1^TP, \qquad (3.10d)$$

$$H_5(\lambda, \alpha, Z_1, Z_2, P, P_f) = AP_f+P_fA^T+B_2Q_vB_2^T+V-P_fC_1^TQ_w^{-1}C_1P_f \qquad (3.10e)$$

The proof of proposition (3.1) is omitted here to conserve space, but is given in detail in Ref. [9]. Thus, for the LQG formulation, the necessary conditions involve solving two algebraic Riccati equations (3.10d, e), two Lyapunov equations (3.10b, c), and a gradient equation (3.10a) for α_i, i=1,...,n_a. In

the case of the LQR formulation, it is easily verified that the optimization statement expressed by (3.9) reduces to

$$\min_{\alpha} J_\lambda(\alpha)=(1-\lambda)J_s(\alpha)+\langle\nu,\alpha\rangle+\lambda tr\{P(B_2Q_vB_2^T+V)\} \tag{3.11}$$

and that the optimality conditions reduce to finding the zeros of the simpler function $H=[H_1, H_2, H_3]$ involving one Riccati and one Lyapunov equation (instead of two as in LQG), in addition to the gradient equations:

$$H_1(\lambda,\alpha,Z,P) = (1-\lambda)\frac{\partial J_s}{\partial\alpha_i} + \nu_i + \lambda tr\{P\frac{\partial(B_2Q_vB_2^T)}{\partial\alpha_i}$$

$$+Z[2P\frac{\partial A}{\partial\alpha_i}+\frac{\partial D_o}{\partial\alpha_i} - P\frac{\partial(B_1R^{-1}B_1^T)}{\partial\alpha_i}P]\}; \quad i=1,\ldots,n_a \tag{3.12a}$$

$$H_2(\lambda,\alpha,Z,P) = A_cZ + ZA_c^T + B_2Q_vB_2^T + V, \tag{3.12b}$$

$$H_3(\lambda,\alpha,Z,P) = A^TP + PA + D_o - PB_1R^{-1}B_1^TP, \tag{3.12c}$$

with $A_c = A - B_1R^{-1}B_1^TP$.

4. HOMOTOPY STRATEGY

For all $\lambda\epsilon[0,1]$, our goal is to minimize (3.9) in the case of the LQG formulation or (3.11) in the case of the LQR formulation by finding the design variables α that satisfy the corresponding optimality conditions (3.10) or (3.12). The basic strategy is: given the solution at a value λ_o, smoothly propagate it to a new solution at $\lambda_o+\Delta\lambda$ via some local mechanism such as Newton method or iterative optimization. This strategy has been analyzed in detail in Ref. (9). In the following, only a summary of the results is given without proofs, assuming the LQR formulation.

Let x denote a generic point $(\alpha, Z, P)\epsilon R^{n_a}x\sum_+ x\sum_+$ so that $H(\lambda, x)$ stands for $H(\lambda, \alpha, Z, P)$. In determining the zeros of H, the following proposition asserts that in a small neighborhood about the optimal at $\lambda=0$, there is a smooth path parameterized by λ consisting of the global optimal solution.

Proposition 4.1: Suppose that $\min J_s(\alpha)$ has a unique global solution α^* satisfying the second order sufficiency condition on the positivity of the

Hessian $[J_s(\alpha^*)]_{,\alpha_i \alpha_j} > 0$. Further, assume that J_s is coercive so that $|J_s(\alpha)| \to \infty$ as $|\alpha| \to \infty$. Then there exists $\epsilon > 0$ such that (3.11) has a unique global solution for $\lambda < \epsilon$.

The next proposition provides a sufficient condition for the path to remain locally optimal.

Proposition 4.2: Let $\phi(\lambda) = (\lambda, x(\lambda))$ denote a smooth path in $[0,r) \times R^{na} \times \sum_+ \times \sum_+$ with $H(\phi(\lambda)) = 0$ and $H_{,x}$ invertible for $\lambda \in [0,r)$. Such an r is guaranteed by Proposition 4.1. Then $\alpha(\lambda)$ is a local minimum for J_λ for each $\lambda \in [0,r)$.

The purpose of the following lemma is to demonstrate that the zero set of H is "generically" well-behaved.

Lemma 4.3: Suppose that $H(0,x)=0$ has a unique solution. Then for almost every choice of $(\nu, V, D_1) \in R^{na} \times \sum_+ \times \sum_+$, the solution path emanating from $(0, x^*)$ is diffeomorphic to the real **R** and every other component of $H^{-1}(o)$ is diffeomorphic to either a circle or **R**.

Thus, following the path defined by one of the zero curves of H, not just the one emanating from the optimal at $\lambda=0$, will not lead to a pathological behavior such as bifurcations or curves with infinite length in bounded sets. Another fundamental and generally difficult question that arises when employing homotopy methods is whether or not the path remains bounded. The following result provides a partial answer to this question.

Proposition 4.4: Suppose that J_s, B_i, D_o, and A are all polynomials in $\alpha_1, \ldots \alpha_{na}$, and assume coercivity of $J_s(\alpha)/|\alpha|$. If $H(0,x)=0$ has a unique solution, then for any $\epsilon > 0$ and for almost every triple $(\nu, V, D_1) \in R^{na} \times \sum_+ \times \sum_+$, the component of $H^{-1}(0)$ containing $(0,x^*)$ is a bounded set in $[0,1-\epsilon] \times R^{na} \times \sum_+ \times \sum_+$.

5. NUMERICAL RESULTS

The numerical experiments described in this section demonstrate the results of the foregoing theory. Two prototype examples are used; both employing the LQR formulation. Implementation of the homotopy strategy of the previous section is achieved by iterative optimization, wherein the solution path for minimizing (3.11) in terms of the homotopy parameter λ starts at $\lambda=0$ with $\alpha_1, \ldots, \alpha_{na}$ initialized to a predetermined sufficiently small allowable size α_o. At this point in the solution space, J_λ is fully weighted toward minimizing J_s only. Minimization of $J_{\lambda o}$ thus yields α_o^* for which $H(\lambda_o, x_o^*)=0$. For the next iteration and for every succeeding one, λ is incremented and the weighting is shifted gradually toward J_c. For a typical iteration j, the following steps are performed:

(i) $\lambda_j \leftarrow \lambda_{j-1} + \Delta\lambda_{j-1}$

(ii) Initialize the minimization of $J_{\lambda j}$ by using $\alpha_j \leftarrow \alpha^*_{j-1}$. This will result in α_j^* for which conditions (3.12) hold.

In performing the minimization in (ii) above, we employed the Automated Design Synthesis (ADS) system of general purpose subroutines [10]. ADS provides a wide selection of options at three levels: strategy, optimization, and line search. Available strategies include sequential linear and quadratic programming, and sequential unconstrained minimization coupled with various penalty methods. At the optimization level, one can choose between Fletcher-Reeves algorithm and the variable metric method of Broydon-Fletcher-Goldfarb-Shanno (BFGS) for unconstrained minimization, or Zoutendijk's method of feasible directions and modifications thereof for constrained minimization. For one-dimensional line search, the options include a combination of polynomial interpolation/extrapolation, solution bounding, and the method of Golden Section. Not all combination of options are compatible at the three levels, and the program parameters must be adjusted to suit the problem at hand. For this purpose, an analytical function was contrived which possessed such features as: easy to compute closed form solution, multiple minima, and insensitivity of the functions gradient near the minima to design parameters. Several compatible options were tried and the program parameter values (e.g., move limits and convergence criteria on the absolute and relative changes in objective function between two successive iterations) were adjusted until the closed form and numerical solution agreed within as few iterations as possible. As a result of these numerical experiments, the popular BFGS variable metric method for unconstrained minimization emerged as the one of choice for use in the combined control-structure optimization examples that follow. During the one dimensional line search, the minimum is located by first computing the bounds, then using polynomial interpolation.

Example 1: The cantilever beam of Fig. 1 resembling a flexible appendage of a large structure is modelled by three finite elements with two degrees-of-freedom (dof) at each node. The structural design variables are the x-sectional areas α_1, α_2, α_3 of the elements. The disturbance υ represents a transverse sound pressure modelled by uncorrelated unit impulses at t=0 concentrated at the three nodal transverse dof. The control force u is applied at the free end along the transverse dof direction. With the J_s given by (3.4), we seek the minimum of (3.11) for $\lambda\epsilon[0,1]$, subject to conditions (3.12). Here, the weighting matrices were taken as $D=10^2$ x diag(K,M), and $R=10^{-4}$, and the initial α_i used were $\alpha_i=\alpha_o=1\text{x}10^{-7}$, i=1,...,3.

Table 1 lists the family of Pareto optimal designs α_i^* that minimize J_λ, $\lambda\epsilon[0,.99]$ along with the corresponding values for J_c, J_s and J_λ. The variations of the Pareto optimal $J_c^*(\lambda)$, $J_s^*(\lambda)$ and $J_\lambda^*(\lambda)$ are shown in Fig. 2. A number of observations can be made from Table 1 and Fig. 2:

(i) The noncommensurate nature of the two costs J_c and J_s is apparent as the weight is shifted between them: while J_c is a strictly decreasing function of λ, J_s is a strictly increasing function of λ. This is consistent since a stiffer structure requires less control energy.

(ii) Except near $\lambda=0$, the optimal structural shapes that minimize J_λ for the disturbance, choice of D and R, and control forces described above are essentially $\alpha_1=\alpha_2$ near the fixed end, and a much smaller α_3 near the free end. This is a physically plausible optimum shape that minimizes the mixture of high strain energy density near the fixed end and high kinetic energy density near the free end. Other choices of disturbance, control location and D, R are expected to alter the optimal beam shape.

(iii) Although the design at λ=0 is guaranteed to be globally optimal (Proposition 4.1), it is possible that designs generated as λ is continued may be only locally optimal. To assess this possibility, the optimal designs listed for λ=.200, λ=.700 and λ=.900 were re-examined separately. For each case, the minimization was restarted with randomly selected initial α_i values. In most cases, the minimization converged to the same or to a higher minimum than obtained in Table. 1.

Table 1. Pareto Optimal Designs of Example 1

λ	Optimal Design x10^{-3} α_1	α_2	α_3	J_c	J_s	J_λ
0.0	.00010	.00010	.00010	350000.	.007	.007
.000001	.00035	.00038	.00032	96000.	.026	.122
.00001	.00116	.00114	.00077	30800.	.076	.384
.0001	.00570	.00625	.00287	5870.	.369	.955
.005	.0585	.0708	.0187	495.	3.68	6.14
.010	.0801	.0802	.0212	391.	4.52	8.89
.020	.113	.111	.0300	265.	6.33	11.52
.040	.159	.155	.0438	177.	8.90	15.64
.100	.237	.249	.0761	102.	14.00	22.77
.200	.367	.349	.112	61.8	20.62	28.86
.300	.425	.457	.154	46.1	25.80	31.92
.400	.563	.532	.189	35.50	31.97	32.99
.500	.622	.650	.231	27.93	37.43	32.68
.600	.795	.751	.279	21.33	45.45	30.97
.700	.886	.934	.346	16.86	53.92	27.98
.800	1.20	1.14	.44	11.78	69.39	23.30
.900	1.68	1.59	.62	7.30	96.98	16.27
.940	1.93	2.07	.82	5.40	119.9	12.27
.980	3.37	3.22	1.37	2.58	198.2	6.49
.990	3.95	4.40	1.81	1.81	253.6	4.33

Example 2: The beam in this example (See Fig. 3) simulates a flexible appendage (length = 45 m) attached to a rigid hub (inertia = 50 kg.m^2) to which a control torque is applied to counteract the transverse unit impulse at the free end. The beam is modelled by three finite elements of constant width = .001 m, but whose nodal depths $d_1,\ldots,d_4$ represent design variables having a lower bound = .001 m. Here again, J_s represents the total mass of the flexible beam (excluding the hub). For the control objective J_c, the response energy is weighed by D_1 so as to minimize the transverse free end displacement, and R is taken = 1.x10^{-4}.

Table 2 and Fig. 4 represent analogous results to those presented for Example 1 in Table 1 and Fig. 2. In addition to observation (i) of the previous example - which holds here as well - the following remarks can be made with reference to the results of this example:

(i) For small values of λ (e.g. λ=0.1), where the total mass J_s dominates the minimization of J_λ, the optimal shapes tend to have a small slope from $d_1 \rightarrow d_2 \rightarrow d_3$, followed by a sharper slope from $d_3 \rightarrow d_4$. As λ increases, minimizing J_s becomes less important than minimizing J_c (tip displacement response energy plus control energy). As a result, the beam becomes gradually stiffer, and the monotonic slope from $d_1 \rightarrow d_2 \rightarrow d_3 \rightarrow d_4$ associated with small λ values gradually disappears at $\lambda \simeq .45$, then gives way to a pronounced inflection of slope for $d_3 \rightarrow d_4$ for $\lambda > .45$. This results in a larger allocation of mass at the tip. This type of shape is physically consistent with the requirements of the two parts of the control objective J_c: a stiffer structure near the hub that is reduced toward the tip (free end) makes best distribution of mass, while minimizing the tip displacement response. On the other hand, a large mass at the tip (where the disturbance exists) makes the disturbance less effective - thus requiring less control effort.

(ii) To confirm the above interpretation, the case of λ=0.7 in Table 2 (i.e. $R=10^{-4}$) was re-examined for smaller and larger values of R; $R=10^{-6}$ and $R=10^{-2}$, respectively. As Fig. 5 shows, smaller values of R give more weighting to the tip displacement response energy part of J_c, thus giving rise to the optimum shape being a stiffer structure near the hub which is reduced toward the tip. Conversely, larger values of R (e.g. $R=10^{-2}$) give more relative weighting to the control energy part of J_c, which is best minimized by the presence of the heavier tip mass. It is interesting to note the similar effect of varying R and varying λ on the optimal shapes.

Table 2. Pareto Optimal Designs for Example 2.

λ	Optimum Design d_1	d2	d_3	d_4	J_c	J_s	J_λ
.000	.001	.001	.001	.001	$1.6\text{x}10^{+10}$	.075	.075
.0001	.02404	.02363	.02291	.01366	3355.26	1.628	1.963
.001	.03552	.03485	.03359	.02065	482.6	2.404	2.884
.010	.05223	.05134	.04940	.03043	70.78	3.54	4.21
.100	.07699	.07559	.07157	.05303	10.03	5.28	5.757
.200	.08759	.08563	.08031	.06659	5.33	6.05	5.906
.300	.09550	.09267	.08657	.07847	3.54	6.62	5.703
.400	.10258	.09867	.09187	.09085	2.56	7.15	5.31
.500	.10944	.10389	.09670	.10470	1.93	7.66	4.79
.600	.11715	.10948	.10135	.12176	1.46	8.22	4.17
.700	.12657	.11576	.10681	.14516	1.08	8.92	3.43
.800	.13944	.12322	.11304	.18055	.77	9.87	2.59
.900	.16369	.13641	.12430	.24923	.47	11.63	1.59
.920	.17299	.14104	.12893	.27374	.41	12.28	1.355
.940	.18542	.14815	.13565	.30544	.34	13.18	1.107
.960	.20578	.16117	.14818	.35131	.26	14.64	0.83
.980	.25035	.19468	.18421	.42401	.16	17.83	0.52
.990	.29401	.25488	.24878	.48165	.09	22.20	0.32

(iii) Of general interest to problems in combined optimization - at least numerically - is the question as to the degree of "roughness" of the hyper-surface $J_\lambda(\alpha)$. A partial answer to this question is provided in Fig. 6 after introducing idealizations that reduce the number of variables from four $(d_1,\ldots,d_4)$ to only two (d_3, d_4), so that a three dimensional plot could be generated. Figure 6 shows such a surface in the neighborhood of the optimum for the case λ=0.7 in Table 2. This is achieved by fixing d_1=.13, allowing d_3 and d_4 to assume various values larger and smaller than those in Table 2 for λ=0.7, and letting $d_2=\frac{1}{2}(d_1+d_3)$. Assuming that the idealizations above (which led to reducing the dimensionality of the J_λ surface) did not alter the basic topology of J_λ surface, it appears from Fig. 6 that J_λ is a smooth function of the design variables - at least in the neighborhood of the minimum. Furthermore, with these idealizations it appears that J_λ is relatively flat near the minimum along the d_4 axis, and that the optimum shape is some linear combination of the four basic shapes depicted at the corners.

6. CONCLUSIONS

An approach for combined control-structure optimization keyed to enhancing early design trade-offs has been outlined and illustrated by numerical examples. The approach employs a homotopic strategy and appears to be effective for generating families of designs that can be used in these early trade studies.

Analytical results were obtained for classes of structure/control objectives with LQG and LQR costs. For these, we have demonstrated that global optima can be computed for small values of the homotopy parameter. Conditions for local optima along the homotopy path were also given. Details of two numerical examples employing the LQR control cost were given showing variations of the optimal design variables along the homotopy path. The results of the second example suggest that introducing a second homotopy parameter relating the two parts of the control index in the LQG/LQR formulation might serve to enlarge the family of Pareto optima, but its effect on modifying the optimal structural shapes may be analogous to the original parameter λ.

ACKNOWLEDGEMENT

The research described in this paper was performed as part of the Control Structure Interaction (CSI) Program at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

REFERENCES

1. D. S. Bodder and L. Junkins, "Eigenvalue Optimization Algorithms for Structural/Control Design Iterations," Amer. Contr. Conf., San Diego, CA, June, 1984.

2. K. Lim and J. Junkins, "Robust Optimization of Structural and Controller Parameters," Journal of Guidance, Vol. 12, January, 1989, pp. 89-96.

3. N. S. Khot, F. E. Eastep, and V. B. Venkaya, "Simultaneous Optimal Structural/Control Modifications to Enhance the Vibration Control of a Large Flexible Structure," Proc. of the AIAA Guid., Nav. and Contr. Conf., 1985, pp. 459-466.

4. S. K. Morrison, Y. Ye, C. F. Gregory, Jr., R. Kosut, and M. E. Regelbrugge, "Integrated Structural/Controller Optimization for Large Space Structures," AIAA Guid. and Contr. Conf., Minneapolis, MN, 1988.

5. M. Salama, J. Garba, and F. Udwadia, "Simultaneous Optimization of Controlled Structures," Journal of Computational Mechanics, Vol. 3, 1988, pp. 275-282.

6. M. Milman, R. Scheid, M. Salama, and R. Bruno, "Methods for Combined Control-Structure Optimization," Proceedings of the VPI Symposium on the Dynamics and Control of Large Structures, Blacksburg, VA, May, 1989.

7. J. G. Lin, "Maximal Vectors and Multi-Objective Optimization," Journal of Opt. Theory and Appl., Vol. 18, January, 1976, pp. 41-65.

8. H. Kwakernaak and R. Sivan, "Linear Optimal Control System," Wiley Inter-Science, NY, 1972.

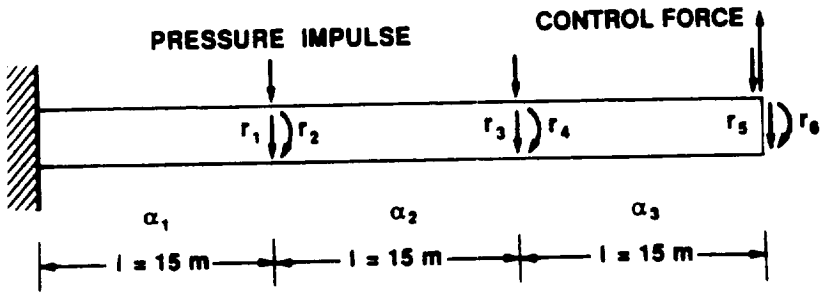
9. M. Milman, M. Salama, R. Scheid, R. Bruno, and S. Gibson, "Integrated Control-Structure Design: A Multiobjective Approach," JPL Report D-6767, (internal report), October, 1989.

10. G. Vanderplaats, "ADS - A Fortran Program for Automated Design Synthesis," NASA CR-172460, October, 1984.

C13

Figure 1

EXAMPLE 1.
CANTILEVER BEAM PROBLEM



STRUCTURAL MODEL: MASS DENSITY = 1660 Kg/m^3, MODULUS = 9.56 x 10^{10} N/m^2
MODAL DAMPING = 0.5%

CONTROL: DISTURBANCE = TRANSVERSE PRESSURE IMPULSE CONCENTRATED AT THE NODES
RESPONSE ENERGY WEIGHTED BY $\tilde{D}$ = Diag (K,M) x 10^2,
CONTROL ENERGY WEIGHTED BY R = 1 x 10^{-4}

DESIGN VARIABLES: $\alpha_1, \alpha_2, \alpha_3 \geq 1 \times 10^{-7}$

Figure 2

CANTILEVER BEAM OPTIMIZATION

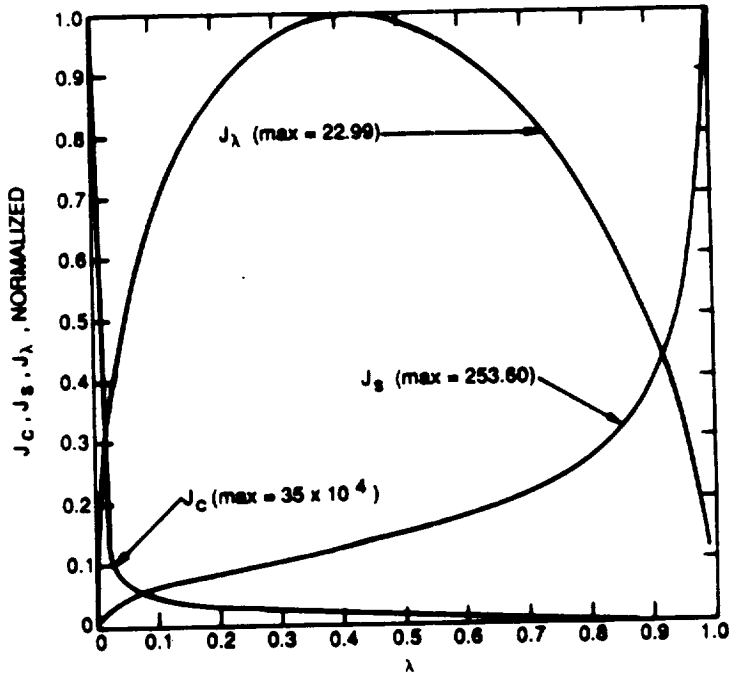
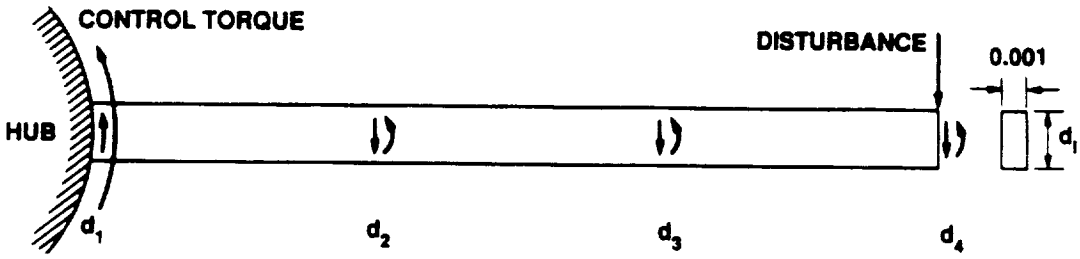


Figure 3

EXAMPLE 2.
HUB-BEAM PROBLEM



STRUCTURAL MODEL: MASS DENSITY = 1660 Kg/m^3, MODULUS = 9.56 x 10^{10} N/m^2
MODAL DAMPING = 0.5%,

CONTROL: DISTURBANCE = UNIT IMPULSE
RESPONSE ENERGY WEIGHTED TO MINIMIZE END DISPLACEMENT, R = 1 x 10^{-4}

DESIGN VARIABLES: $d_1, \cdots, d_4 \geq 0.001$

Figure 4

HUB-BEAM OPTIMIZATION

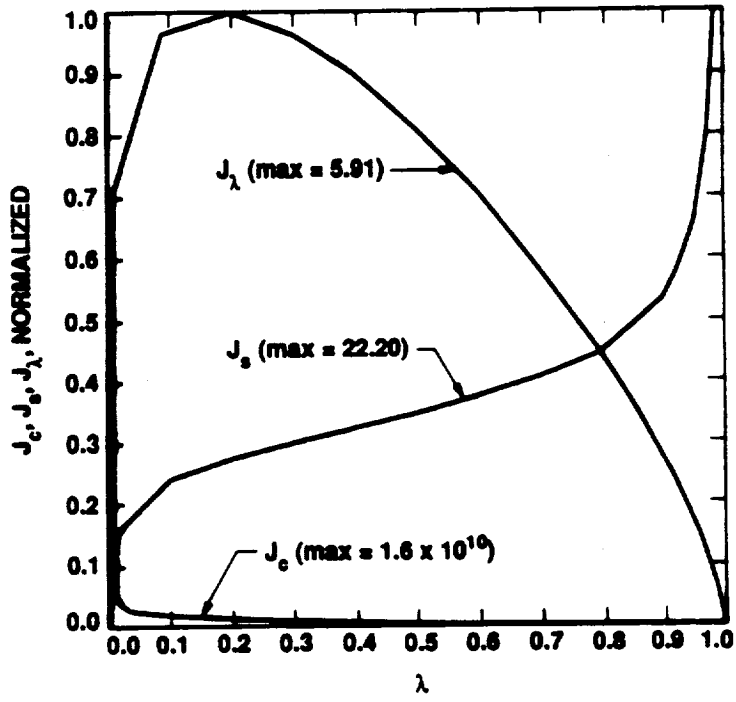


Figure 5

OPTIMUM SHAPES

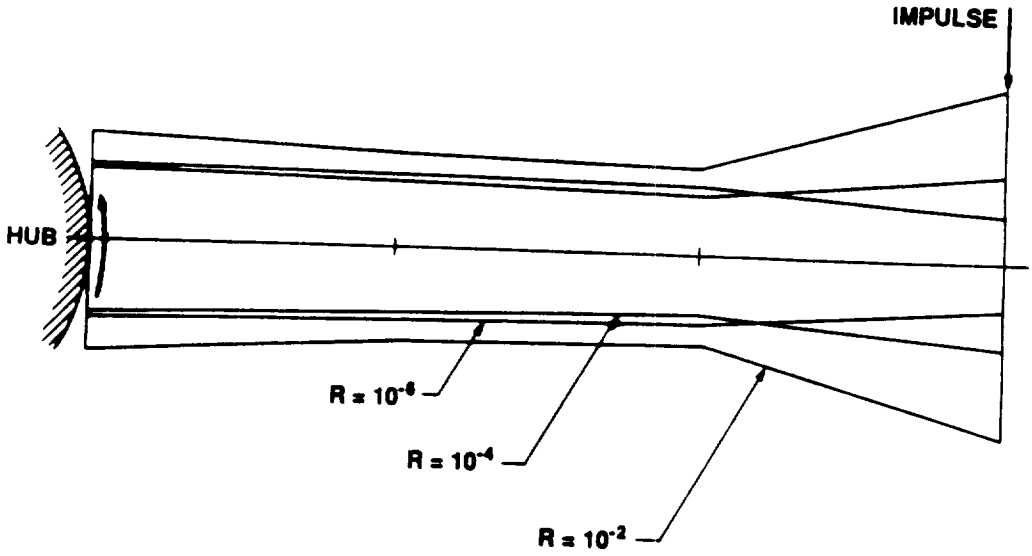
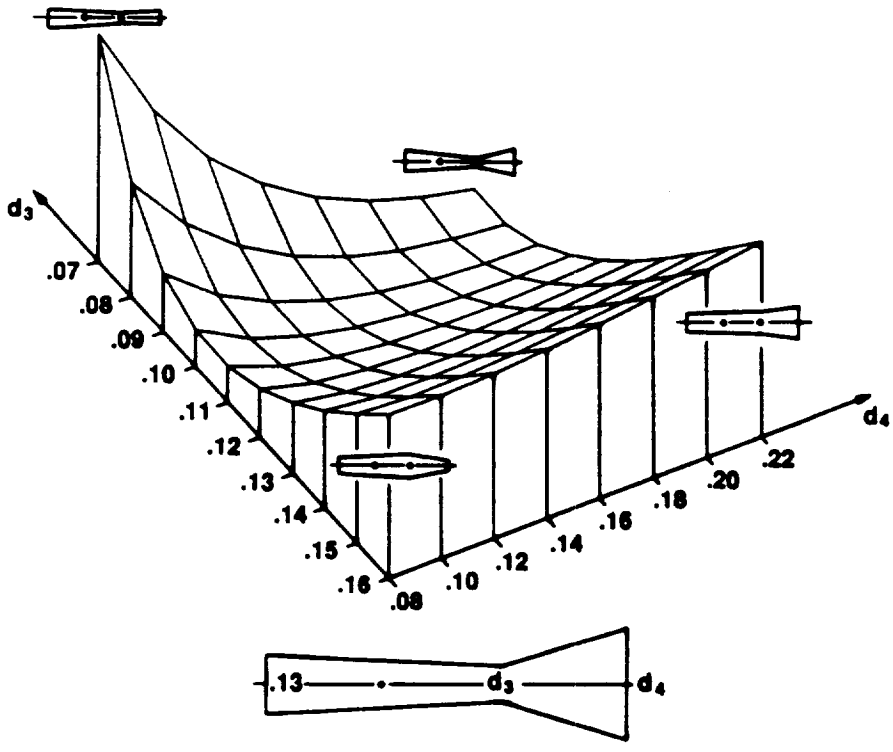


Figure 6

$J_\lambda(d_3,d_4)$ SURFACE NEAR THE MINIMUM, $\lambda = 0.7$



PARALLEL PROCESSING OF REAL-TIME DYNAMIC SYSTEMS SIMULATION ON OSCAR (Optimally SCheduled Advanced multiprocessoR)

Hironori Kasahara*, Hiroki Honda and Seinosuke Narita
Dept. of Electrical Engineering, Waseda University
3-4-1 Ohkubo Shinjuku-ku, Tokyo, 169, Japan

Abstract

This paper presents parallel processing of real-time dynamic systems simulation on a multiprocessor system named OSCAR. In the simulation of dynamic systems, generally, the same calculation are repeated every time step. However, we cannot apply the Do-all or the Do-across techniques for parallel processing of the simulation since there exist data dependencies from the end of an iteration to the beginning of the next iteration and furthermore data-input and data-output are required every sampling time period. Therefore, parallelism inside the calculation required for a single time step, or a large basic block which consists of arithmetic assignment statements, must be used. In the proposed method, near fine grain tasks, each of which consists of one or more floating point operations, are generated to extract the parallelism from the calculation and assigned to processors by using optimal static scheduling at compile time in order to reduce large run time overhead caused by the use of near fine grain tasks. The practicality of the scheme is demonstrated on OSCAR (Optimally SCheduled Advanced multiprocessoR) which has been developed to extract advantageous features of static scheduling algorithms to the maximum extent.

I. INTRODUCTION

High speed dynamic systems simulation, or solution of ordinary differential equations, has been required to simulate dynamic behaviors of various systems such as airplanes, missiles, nuclear reactors and robots, in real-time. So far, the dynamic systems simulation has generally been performed on traditional analog or hybrid computers or on general-purpose digital computers by using a simulation language like CSMP (Continuous Systems Modeling Program). However, these approaches have several problems, for example, operational accuracy and realization of non-linear functions for the analog computers and high processing cost and real-time input-output for the general purpose main-frame computers.

In an attempt to resolve these problems, the use of parallel processing techniques[13-14] has attracted much attention. In fact, various parallel processing schemes, especially parallel processing using

---

* Prof. Kasahara is also a Visiting Research Scholar of Center for Supercomputing R & D, Univ. of Illinois at Urbana-Champaign, 305 Talbot Laboratory, 104 South Wright Street, Urbana, IL 61801.

multiprocessor systems[14-15], have been so far proposed[1-4]. The differences among the schemes lie in the choice of task granularity and task scheduling. For example, Korn [1] and Koyama [4] employed a large task size (coarse task-grain) approach where the computation for the numerical integration of each equation in a set of first-order simultaneous differential equations was selected as a task. The generated tasks are assigned properly by the user to a relatively small number of processors. The functional distribution approach by Gilbert, et al[2], dealt with each fundamental operation (four fundamental arithmetic operations, integration and so on) as a task to be assigned to a dedicated hardware operational unit. Yoshikawa, et al[3], also adopted an approach where each fundamental arithmetic operation was assigned to one processor. The common problem left unsolved to these approaches was poor parallel processing efficiency stemming from the lack of efficient methods which allocate the generated tasks onto an arbitrary number of parallel processors in an optimal manner. This paper proposes a parallel processing scheme for the solution of the above-mentioned problem by using static minimum execution time multiprocessor scheduling algorithms[5][10] already developed by the authors for optimum task allocation. The proposed parallelizing compilation scheme consists of the following processes: task generation, optimal task scheduling, and generation of machine codes to be executed on respective processor element.

The effectiveness and practicality of the proposed scheme are demonstrated on OSCAR's processor cluster with sixteen 32-bit RISC-like processor elements which has been designed to extract advantageous features of static scheduling at compile time to the maximum extent.

II. A PARALLEL PROCESSING SCHEME USING STATIC SCHEDULING

Generally, dynamics of most continuous-time systems can be modeled by the following explicit first-order simultaneous ordinary differential equations:

$$dx_i/dt=f_i(t,x_1,x_2,\ldots,x_m) \quad (i=1,2,\ldots,m)$$

Therefore, the dynamics systems simulation can be regarded as the solution of the ordinary differential equations. Hence, this paper handles parallel solution of the equations using various numerical integration formulae such as Euler, Trapezoidal, 3rd- and 4th-order Adams Bashforth, 4th-order Runge Kutta and 4th-order Adams Moulton (predictor-corrector method) listed in Table 1. In applying these integration formulae, the computation required for each integration step consists of arithmetic assignment statements to evaluate the derivative of each equation and to perform numerical integration. Between consecutive iterations, there exist data dependencies[16-17] from the end of an iteration to the beginning of the next iteration. Furthermore, for real-time simulation, data input and data output are required every iteration or few iterations, namely every sampling period. Therefore, we cannot apply Do-all and Do-across techniques to parallel processing of the dynamic systems simulation which are popular parallel processing schemes for a Do loop on a multiprocessor system[18][19].

Taking into consideration these facts, in order to realize efficient parallel processing of the simulation, we must parallel process a block of arithmetic assignment statements, or a basic block, in each iteration.

However, the parallel processing of the basic block on a multiprocessor system has been thought to be very difficult since data transfer overhead and synchronization overhead are relatively large. The proposed scheme allows us to minimize these overheads and to realize efficient processing by generating optimized machine codes based on the static schedule at compilation time.

A. Task Generation

As mentioned before, in the dynamic systems simulation, we must process each iteration in parallel though we can sometimes unroll a few iterations if data input and output should be made every few iterations. In order to process the iteration in parallel, first of all, we must generate tasks with suitable granularity, which are basic units assigned to processors. As for the task granularity, several levels may be perceived: equation level, operation element level, and intermediate level. In the case of equation level granularity, the computation related to each subscript i for each numerical integration formula listed in TABLE 1 (the computation of a derivative and that of numerical integration corresponding to each formula for each variable X_i) is considered to be a task. When operation element level granularity is adopted, the computation for each derivative or for each numerical integration is subdivided into finer fundamental operation elements such as the four arithmetic operations and trigonometric functions, each of which is taken as a task and allocated to the processors (fine granularity). In the intermediate task granularity, several floating point operations are combined to form a task. For instance, when Van der Pol's equations

$$dx_1/dt = x_2$$
$$dx_2/dt = \varepsilon x_2 - x_1^2 * \varepsilon x_2 - x_1$$

is decomposed into fairly small intermediate-level tasks, three multiplication tasks, two subtraction tasks, and two integration tasks (including several floating point operations) are generated. Fig.1 depicts the block diagram representation of the seven tasks, with data dependencies explicitly shown.

There exists no general rule for determining the best task granularity applicable to all kinds of dynamic systems. When parallel processing is performed on a multiprocessor system with little data transfer and synchronization overheads among processor elements, the operation element level granularity is known to be most advantageous to achieve minimum processing time because parallelism can be exploited to the maximum extent. For a large-scale problem (the order of simultaneous equations is very high) or a multiprocessor system with poor data transfer capabilities, however, the operation element level granularity does not always give the best performance. In other words, much attention must be paid to such factors as processor speed, interprocessor data transfer speed, size and parallelism inherent to the problem in hand, and complexities of scheduling mechanisms (both software and hardware) [7]. Namely, we must choose the best granularity for each problem and each multiprocessor system. For this reason, the proposed parallel processing scheme provides with two methods for the input of simulation source programs. The first method employs a simplified simulation language shown in Fig.2, which allows direct input of mathematical equations. The user can specify arbitrary task granularity from the operation element level to

the equation level. The second method facilitates the input of block diagram representations such as those employed for analog computer. As shown in Fig.1, each operational element of analog computer (adder, integrator, etc.) can be taken as a task, to realize near fine granularity. Medium granularity can also be dealt with by combining automatically several tasks with near fine granularity. (This process is referred to as task fusion). In what follows, emphasis will be placed on the case of near fine granularity, namely the finest granularity that can be treated by use of the proposed scheme on a multiprocessor system named OSCAR mentioned later.

Next, the proposed parallel processing scheme analyzes precedence relations caused by data dependencies among the generated tasks and represents the task precedence relations by a task graph like Fig. 3 which is a directed acyclic graph (DAG). The precedence constraints represent the restrictions existing among tasks regarding the execution order of tasks. The existence of task i precedent to task j means that the execution of task j cannot be initiated before the completion of task i. The precedence relation can be examined by the data flow analysis among tasks. When the data flow analysis is made, the output variable of each integration task is treated as an initial value. Each node in the task graph stands for a task and an arc between a pair of nodes for the precedence constraint. Nodes 0 and 8 are not actual nodes but dummy nodes introduced for the sake of convenience. They represent the entry node and the exit node, respectively. The figure beside each node represents the estimated processing time of the corresponding task. Since the actual processing time does not usually take on a fixed value but varies with the data to be processed, the average value or the worst-case value is employed as the input[7], which is used in the scheduling algorithms to be described in the subsequent section. When the average value is used for each task, the resultant schedule gives the minimum value of the average processing time of the task set. Similarly, when the worst-case value is used, the worst-case processing time is minimized. However, OSCAR, which is a target machine in this paper, can execute all instructions including a few floating point operations in one clock by employing RISC like processor. Therefore, we don't have the above mentioned problem on OSCAR, a compiler can estimate accurate processing time of each task.

Once a task graph is generated, the minimum possible processing time achieved by parallel processing of the tasks can be estimated as the critical path length t_{cr} of the task graph. In Fig.3, the critical path is shown by double-line segments.

An unique task graph can also be generated by following simple procedures in the case of the block diagram input mode. The task graph shown in Fig.3 represents the computation in one integration step when the tasks are generated in the size of near fine granularity and the numerical integration method employed is Euler, Trapezoidal or 3rd- or 4th-order Adams Bashforth. The integration task involves computation specific to each numerical integration method. When the 4th-order Runge-Kutta method is employed, k_1 through k_4 need to be evaluated, and the computation described by this task graph is repeated four times or the expanded task graph involving the computation repeated four times is processed for each integration step. In the former case, the content of each integration task to be processed differs with the iteration count in order to evaluate k_1 through k_4 and their weighted average. Similarly, when a predictor-corrector method such as the 4th-order Adams-Moulton is used, the task graph is computed twice or the expanded task graph to

represent the unrolled computation is processed for each integration step. In the former case, the computation corresponding to the predictor of the integration task is performed first, followed by the computation for the corrector.

As mentioned earlier, the task graph shown in Fig.3 represents the case where the tasks are generated in the size of near fine granularity. When coarse granularity at the equation level is employed for task generation, the portion surrounded by the dashed lines becomes a task. Also, in the case of fine granularity, the portion of each integration task is replaced by a subgraph generated by subdividing it into the operation element level.

It should be mentioned here that parallel processing scheme proposed in this paper is so designed that the tasks generated in either fine or near fine granularity level can be fused automatically without sacrificing much parallelism. As a simple example, when there exist a pair of successor task (son node) with only in-edge and the predecessor task (father node) with only one out-edge, the two tasks are fused into a single task. Even such an easy task fusion technique allows the optimization of resister utilization and avoids unnecessary data transfer for more efficient parallel processing.

B. Scheduling Algorithms

In order to process the set of tasks on a multiprocessor system efficiently, the assignment of tasks onto the parallel processors and the execution order among the tasks assigned to the same processor must be determined optimally. The problem which determines the optimal assignment and execution order can be treated as the traditional multiprocessor scheduling problem of which the objective function is the minimization of the parallel processing time or schedule length [5][8]. To state formally, the scheduling problem is to determine such a nonpreemptive schedule that the execution time or the scheduling length be minimum, given a set of n computational tasks T=(T1, ... ,Tn), precedence constraints among the tasks and n processors with the same processing capability. This problem, however, has been known as a "strong" NP-hard problem [9]. In other words, unless P=NP, it is impossible to construct not only a pseudo-polynomial time optimization algorithm but also a fully polynomial time approximation scheme. With this fact in mind, the authors have successfully constructed a heuristic algorithm named CP/MISF and an efficient practical algorithm called DF/IHS [5]. The former algorithm can provide very precise approximate solutions quite rapidly because of its very low time complexity. The latter algorithm can obtain optimal solutions or approximate solutions with guaranteed accuracies from optimal solutions by combining CP/MISF and depth-first search. In what follows, the two algorithms are explained very briefly. For further details, the reader is referred to the literature [5].

1) CP/MISF(Critical Path/Most Immediate Successors First) Method
This method essentially is a kind of list scheduling algorithms.

step.1 Determine the level l_i for each task. The l_i is the longest path from N_i to the exit node.

step.2 Construct the priority list in the descending order of l_i and the number of immediately successive tasks.

step.3 Execute list scheduling [8] on the basis of the priority list.

Since the list scheduling may be regarded as a method to construct the

schedule for the case where a set of tasks are processed in parallel in the data-driven manner considering the priority assigned to each task, it can be easily extended to dynamic scheduling at run time.

Furthermore, the list scheduling can be also modified to eliminate unnecessary data transfer among processors. In the modified algorithm CP/DT/MISF method[10], when the tasks with the same priority are allocated to a processor, a task is allocated to the processor which needs the minimum data transfer to execute the task. This simple modification significantly decreases the data transfer overhead for the multiprocessor system with poor data transfer performance.

Its average performance was evaluated for a total of over nine thousand test cases by comparing the CP/MISF solutions with the lower bound function[11]. Optimal solutions were obtained for 67 percent of the cases tested. Approximate solutions with errors of less than 5 percent were obtained for 87 percent of the cases and those with errors of 10 percent for 98.5 percent of the cases. The worst-case performance of CP/MISF, i.e., the error of the worst-case solution t obtained by CP/MISF from the true optimal solution t_{opt} is given by

$$(t-t_{opt})/t_{opt} \leq 1/m \quad [5].$$

In addition, the time complexity of CP/MISF is $O(n^2+mn)$. For problems with about one thousand tasks, it only takes a few ten seconds on a HITAC M280H system. In summary, CP/MISF is suitable for the solution of very large problems with hundreds or even thousands of tasks.

2) DF/IHS (Depth First/Implicit Heuristic Search) Method
DF/IHS is an optimization/approximation algorithm to determine schedules (solutions) which are always more precise than those by CP/MISF. The method combines CP/MISF and depth-first search in a special manner and reduces markedly space complexity (memory requirements) and average computation (search) time. It is so practical and powerful that optimal schedules for most large-scale problems involving a few hundred tasks for a total of some ten parallel processors can be determined in several seconds to one hundred seconds on an M280H. Optimal solutions could be obtained for 75% of the test problems where the upper limit of search time was set to 180 seconds [5]. The effectiveness of DF/IHS may be recognized by considering the fact that use of dynamic programming could provide optimal solutions for small problems with less than 40 tasks even for two parallel processors. In the case of parallel processing on a limited number of processors, it is known that there exist such task graphs that the minimum processing time cannot be attained by data driven execution or the list scheduling [12]. For these task graphs, use of DF/IHS can determine the optimal schedule that gives rise to the minimum processing time by forcing some processors to be idle for a certain time period. This fact implies the possibility of more efficient parallel processing than data flow machines. In summary DF/IHS is very useful when CP/MISF fails to obtain an accurate solution for problems with several hundred tasks.

C. Machine Code Generation

For the efficient execution on an actual multiprocessor system, the optimal machine codes tailored to the given system must be generated by using the scheduled results. The scheduled results give us the information about tasks to be executed on each processor element, the execution order

of tasks on the same processor element, the rough estimates of waiting time of the tasks which wait for the data from other tasks assigned to other processors, the tasks to be synchronized and so on. Therefore, we can generate the machine codes for each processor by putting together the codes for the tasks assigned to the processor and attaching the codes for synchronization and data transfer among processors. The "version number" method is used for the synchronization among tasks. The version number corresponds to the number of times of iterations or integration steps. Each "writer" task updates the version number on the common memory to the number of current integration step for itself after it finishes writing the shared data. And each "reader" task checks the version number if the number is the same as the number of current integration step to the reader task. All processor elements (PE's) have the same version numbers during one integration step and update or increase the number at the end of the integration step. Updating the version number on each PE by respective PE's allows us to eliminate the need to update the version number (or to reset a flag used in test & set or semaphore) attached to each shared data on a common memory when the next integration step is started. Therefore, the version number method can minimize the frequency of access to the common memory for task synchronization in this application.

We can also optimize the codes to minimize various processing overheads by making full use of all information which is obtained as the result of static scheduling. For example, the information about task assignment and execution order allows the optimized use of the registers of the processor when the tasks allocated to the same processor exchange data. The optimal use of registers reduces the processing time markedly. The knowledge about the estimated waiting time helps prevent the degradation of data transfer performance caused by frequent bus access to check the existence of the required data (data level synchronization) by the waiting task. In other words, if it is estimated that the task must wait the data for a long time, the frequency to check a flag on a common memory is reduced. In addition, we can minimize the synchronization overhead by carefully taking into consideration the information about the tasks to be synchronized, the task assignment and the execution order. For example, let tasks A, B and C be allocated to processor 1 and tasks D and E to processors 2 and 3 respectively as shown in Fig.4 and data among the tasks be transferred via a common memory. Then task B does not need to check the flag which shows the completion of task A because both tasks are allocated to the same processor. Task E has no need to check the flag which indicates the completion of task D because the termination of task D has already been confirmed by task C or B.

In the parallel processing scheme, the transfer of output data of integration tasks is not represented on a task graph since data flow analysis is performed on the assumption that output data of the integration tasks has been given as initial values. In actual processing, however, those data must be transferred to several tasks allocated on other PE's between the end of an integration step and the beginning of the next integration step since, during one integration step, all the tasks except the integration tasks use the output data of the integration tasks generated in the previous integration step. The data transfer at one time causes bus congestion. In order to prevent the bus congestion, two copies of machine codes for each PE which are assigned different data storages are generated and executed alternatively for every integration step. Generating the two copies of codes allows each integration task in a copy of codes to write or transfer its output data, as soon as it completes

execution, onto a data storage assigned for the next integration step or another copy of codes. In other words, it allows distributed bus access and also to eliminate data synchronization to check the completion of the integration tasks because the output data of the integration tasks has already been transferred before the end of each integration step.

The optimal machine codes for each PE generated in the way mentioned above are loaded to the local instruction memory of each processor element and executed asynchronously. The four steps of the proposed parallel processing scheme described in this section can be performed automatically by a special purpose compiler.

III. PERFORMANCE EVALUATION ON OSCAR

This section discusses the performance evaluation of the proposed parallel processing scheme on a prototype multiprocessor supercomputing system named OSCAR being developed by the authors.

In the following, as an example of parallel processing of the practical dynamic systems simulation for evaluating the performance of the proposed scheme on OSCAR, dynamics simulation of a hot strip mill control in a steel making plant is treated. The simulation program can be represented by a block diagram shown in Fig. 5. In this example, near fine task granularity has been chosen in which each integration task consists of several floating point operations and the other tasks consist of only one floating point operation. By the task generation method using near fine granularity, fifty-one tasks involving nine integration tasks were generated. Fig. 6 is a task graph generated from Fig.5 automatically by a special purpose compiler.

OSCAR is a hierarchical multiprocessor system which has a plurality of processor clusters as shown in Fig.7. Its goal is to realize, by the combined use of static scheduling and dynamic scheduling, efficient parallel processing of Fortran programs and a variety of applications including those which have so far been difficult to process efficiently because of a lot of scalar assignments involved.

One processor cluster(PC) hardware has already been completed. On the PC, various parallel processing application will be implemented. The PC involves sixteen processor elements, three common memories, a local control processor and three shared buses. Each PE consists of a 32-bit custom-made RISC-like processor with 64 general purpose registers which executes all instructions including a few floating point operations in one clock (clock:200ns), a 256-KW local data memory, a 2-KW two-port memory to communicate with other PE's, two banks of 128-KW instruction memory and a DMA controller. The DMA controller realizes high-speed transfer of a block of data to the common memories and the two-port memories of other PE's and dynamic loading of a set of instruction codes from the common memories to one of the instruction memory banks during execution. The reduced instruction set and the one-clock-execution of the all instructions make the estimation of task processing time for the scheduling easy and accurate. For interprocessor communication, three types of data transfer modes are provided such as broadcast mode, direct data transfer mode to the two-port memory of another PE or indirect data transfer mode via a common memory. Each mode can be used for both single word data transfer and block data transfer. Each common memory accepts simultaneous accesses from three buses. The data transfer speed of the three buses totals to 60MByte/s.

When we operate one PC of OSCAR, a Unix-based workstation is used as the host computer which generates machine codes for each PE by using static schedule providing the minimal processing time and downloads the codes to each PE. In the generated codes, bus access timing by PE's, data transfer modes and use of 64 registers employed to exchange data among tasks assigned on the same PE are optimized. In addition, redundant task synchronization is also eliminated as mentioned before. An timing chart representing execution of the machine codes is shown Fig.8. This chart can be regarded as a precise simulated result of actual parallel processing on OSCAR. In the figure, for PE3, characters such as "LD30","25R","wait"," PE5" and so on are written. These characters mean to load input data for task 30 from a local data memory to registers, execute task 30 and keep its result in registers, and wait for a while to directly transfer the output data of task 30 to PE5. At that time, PE3 waits for bus access since PE1 is accessing bus for data broadcasting. In OSCAR, another PE cannot access the busses while a PE is broadcasting data. Furthermore "U26 51"," PE2","WAIT","FC44" and "38R" represent to execute task 51 by using output data of task 26 on registers, transfer its output data to PE2, wait for output data of task 44 from PE5, check a flag showing completion of data transfer from task 44, execute task 38 and keep its output data on a register.

Fig.9 shows the measured parallel processing time on OSCAR (solid lines) and simulated parallel processing time (dotted lines and chained lines) of 51 tasks in Fig.6. In this example, 4th-order Adams-Bashforth method was used. The measured processing time on OSCAR of the near fine granularity tasks was reduced from 108.7 us for one PE to 37.2 us (1/2.92) for seven PE's. Next, the task fusion technique which generates a coarser granularity task by combining several tasks in order to reduce data transfer overhead with the minimum loss of parallelism is evaluated. As a simple example, those tasks surrounded by dotted lines in Fig. 6 can be fused and twenty-two medium granularity tasks are generated automatically. Processing time of the medium granularity tasks (after task fusion) decreases from 105.8 us for one PE to 36.8 us (1/3.01) for seven PE's. From the results, it has been confirmed that the determination of the most suitable task granularity is very important and that the automatic task fusion is useful.

The two dotted lines show the simulated processing time. It is clear from the figure that there exists little difference between the measured processing time and the simulated processing time or an execution image of machine codes generated by using static scheduling. In the light of this fact, we can conclude that the generation of the precisely optimized machine code using static scheduling is very useful for OSCAR.

The processing time shown above, however, represents the degraded performance of OSCAR since OSCAR is still in a stage of operation testing. Though OSCAR can normally transfer two words data in 5 clocks, the processing time were measured in a degraded operating condition where two words data transfer takes 9 clocks. Therefore data transfer overhead will be reduced by half in the normal operating condition. The chained lines in Fig.9 show the precisely simulated processing times in the normal condition for the near fine granularity before task fusion and the medium granularity after task fusion. The processing time after task fusion decreases from 104.8 us for one PE to 28.8 us for seven PE's (1/3.64). From the experiment mentioned above, it has been confirmed that OSCAR's architecture, especially one clock execution of all instructions and three types of data transfer modes, allows us to efficiently parallel process

the dynamic systems simulation by extracting the advantageous features of static scheduling to the maximum extent.

V. CONCLUSIONS

In this paper, the authors have proposed a parallel processing scheme of the dynamic systems simulation using static optimal multiprocessor scheduling algorithms and shown that the scheme allows us to realize efficient parallel processing on OSCAR which has been designed to extract the advantageous features of static scheduling to the maximum extent. More precisely speaking, the special purpose compiler for OSCAR using the proposed scheme can generate suitable granularity tasks, the minimal execution time schedule and optimized machine codes for each processor in which data transfer and synchronization overheads are minimized and the registers on each processor are used optimally.

Furthermore, it has been confirmed that the architectural support in OSCAR for a parallelizing compiler using static scheduling is very useful. The authors are planning to develop a practical dynamic systems simulator using OSCAR which can simulate dynamics of flying objects like airplanes and missiles, nuclear reactors, robot systems and various industrial plants.

REFERENCES

[1] G. A. Korn, "Back to Parallel Computation:Proposal for a Completely New On-line Simulation System Using Standard Minicomputers for Low-cast Multiprocessing," Simulation , Vol.19, pp. 37-44 (Aug.1972).

[2] E. O. Gilbert, and R. M. Howe, "Design Consideration in a Multiprocessor Computer for Continuous System Simulation," Proc. National Computer Conf., pp. 385-393, AFIP Press, Reston (1978).

[3] R. Yoshikawa, T. Kimura, Y. Nara and H. Aiso, "A Multi-microprocessor Approach to a High-speed and Low-cast Continuous-system Simulation," Proc. National Computer Conf., pp. 931-936, AFIP Press, Reston (1977).

[4] S. Koyama, K. Makino, N. Miki, Y. Iino and Y. Iseki, "On the Parallel Processor Array of Hokkaido University High-speed System Simulator "Hoss"," Proc. 8th IFAC World Cong., pp. 1715-1720, Pergamon Press, Oxford (1981).

[5] H. Kasahara and S. Narita, "Practical Multiprocessor Scheduling Algorithms for Efficient Parallel Processing," IEEE Trans. Comput., Vol.c-33, pp. 1023-1029 (Nov.1984).

[6] H. Kasahara and S. Narita, "Parallel Processing of Robot-arm Control Computation on a Multimicroprocessor System," IEEE J. of Robotics and Automation, Vol.RA-1, pp. 104-113 (June 1985).

[7] H. Kasahara and S. Narita, "An Approach to Supercomputing Using Multiprocessor Scheduling Algorithms," Proc. IEEE First International Conf. on Supercomputing Systems, pp. 139-148 (Dec.1985).

[8] E. G. Coffman, "Computer and Job-shop Scheduling Theory," Wiley, New York (1976).

[9] M. R. Garey and D. S. Jonson, "Computers and Intractability : A Guide to the Theory of NP-Completeness," Freeman, San Francisco (1979).

[10] H. Kasahara and S. Narita, "Load Distribution among Real-time Control Computers Connected via Communication Media," Proc. IFAC 9th

Triennial World Congress, pp. 2695-2700 (1984).
[11] E. B. Fernardez and B. Bussel, "Bound on the number of processors and time for multiprocessor optimal schedules," IEEE Trans. comput., Vol.c-22, pp. 745-751, (Aug. 1973).
[12] C. V. Ramamoothy, K. M. Chandy and M. J. Gonzalez Jr., "Optimal scheduling strategies in a multiprocessor system," IEEE Trans. comput., Vol.c-21, pp. 137-146, (Feb.1972).
[13] R.W. Hockney and C.R. Josshope,"Parallel Computers 2: Architecture, Programming and Algorithms," Adam Hilger, 1988.
[14] K.Hwang and F.A.Briggs, "Computer Architecture and Parallel Processing," McGRAW-HILL, 1984.
[15] D.D.Gajski and Peir,"Essential Issues in Multiprocessor Systems," IEEE Computers,Vol.C-18,No.6,pp.9-27,Jun.1985.
[16] U.Banerjee, Dependence Analysis for Supercomputing, Kluwer Academic Publisher,1988
[17] D.A.Padua,D.J.Kuck and D.H.Lawrie,"Highspeed Multiprocessors and Compilation Techniques,"IEEE Trans. Comput. Vol.29,No.9,Sep.,1980.
[18] D.J.Padua, and M.J.Wolfe,"Advanced Compiler Optimizations for Supercomputers," C.ACM, Vol.29,No.12,pp.1184-1201,Dec.1986.
[19] C. Polychronopoulos,"Parallel Programming and Compilers," Kluwer Academic Publishers, 1988.

TABLE I. NUMERICAL INTEGRATION METHODS

Trapezoidal	$X_{i,n+1}=X_{i,n}+h\ (3\dot{X}_{i,n}-\dot{X}_{i,n-1})\ /2$ Where $X_{i,n}=f_i\ (t_n,\ X_{1,n},\ \cdots,\ X_{s,n})$
4th_Order Runge Kutta	$X_{i,n+1}=X_{i,n}+\ (k_{1,i}+2k_{2,i}+2k_{3,i}+k_{4,i})\ /6$ $k_{1,i}=h\ f_i\ (t,\ X_{1,n},\ X_{2,n},\ \cdots,\ X_{s,n})$ $k_{2,i}=h\ f_i\ (t+h/2,\ X_{1,n}+k_{1,1}/2,\ X_{2,n}+k_{1,2}/2,\ \cdots,\ X_{s,n}+k_{1,s}/2)$ $k_{3,i}=h\ f_i\ (t+h/2,\ X_{1,n}+k_{2,1}/2,\ X_{2,n}+k_{2,2}/2,\ \cdots,\ X_{s,n}+k_{2,s}/2)$ $k_{4,i}=h\ f_i\ (t,\ X_{1,n}+k_{3,1},\ X_{2,n}+k_{3,2},\ \cdots,\ X_{s,n}+k_{3,s}/2)$
4th_Order Adams Moulton	$X^p_{i,n+1}=X^c_{i,n}+h\ (55\dot{X}^c_{i,n}-59\dot{X}^c_{i,n-1}+37\dot{X}^c_{i,n-2}-9\dot{X}^c_{i,n-3})\ /24$ $X^c_{i,n+1}=X^c_{i,n}+h\ (9\dot{X}^p_{i,n+1}+19\dot{X}^c_{i,n}-5\dot{X}^c_{i,n-1}+\dot{X}^c_{i,n-2})\ /24$

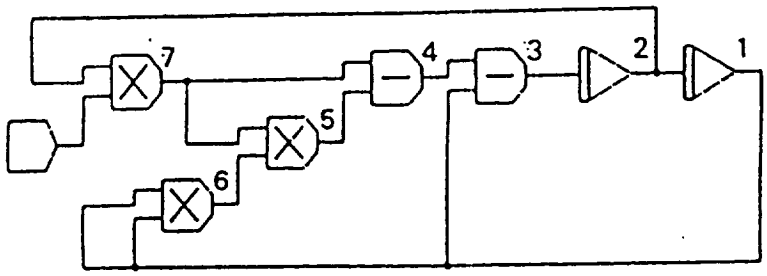


Fig.1 Block diagram for Van der Pol eq..

```
begin
   a=integral(b,0.01);   (1)
   b=integral(c,0.01);   (2)
   c=d-a;                (3)
   d=g-e;                (4)
   e=f*g;                (5)
   f=a*a;                (6)
   g=b*1                 (7)
end.
```

Fig.2 Assignment statements for Van der Pol eq..

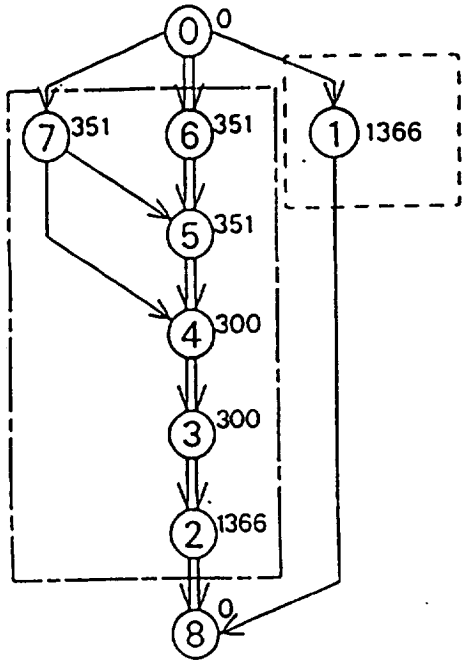


Fig.3 Task graph for Van der Pol eq.

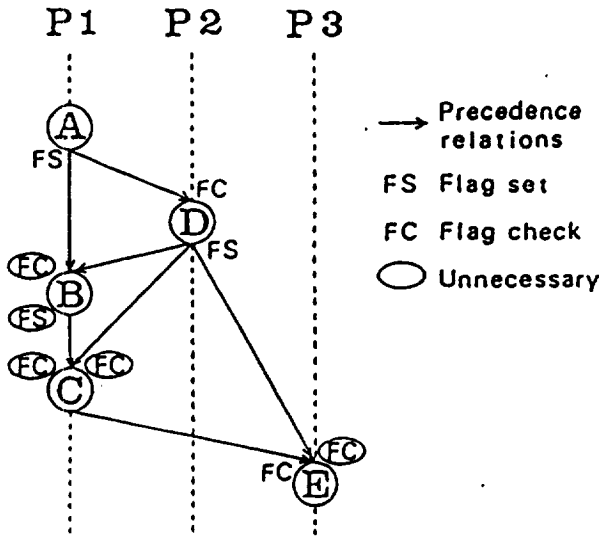


Fig.4 Minimization of synchronization overhead.

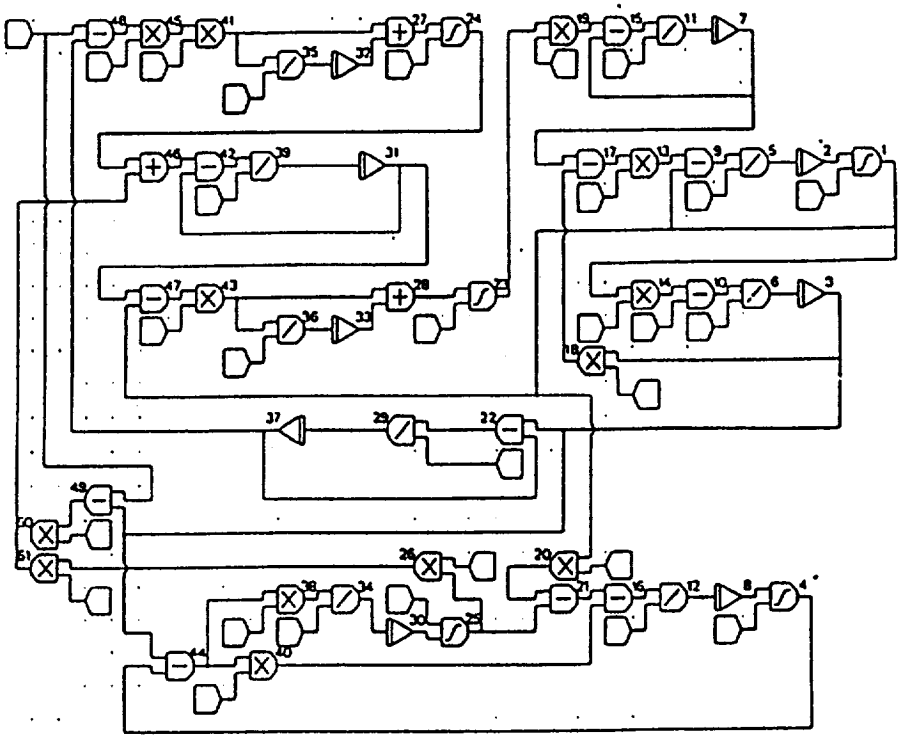


Fig.5 An example of block diagram.

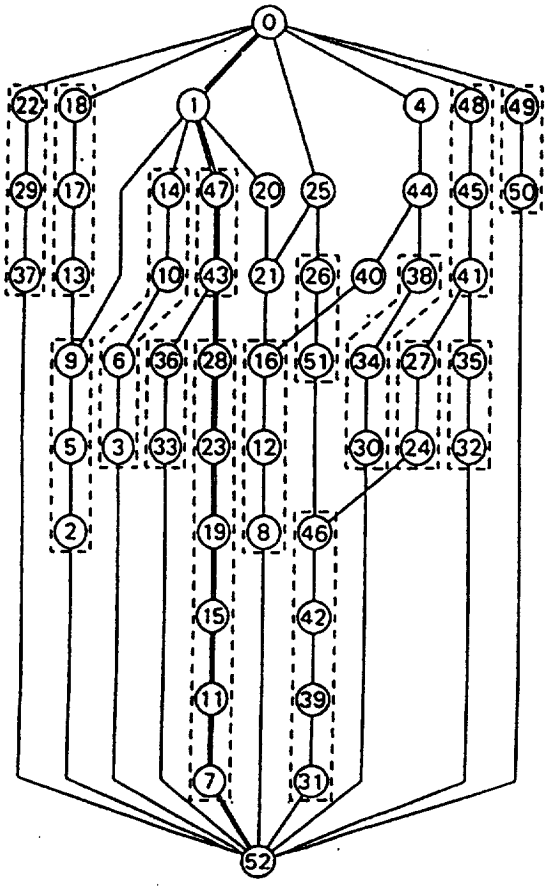


Fig.6 Task graph for Fig.5.

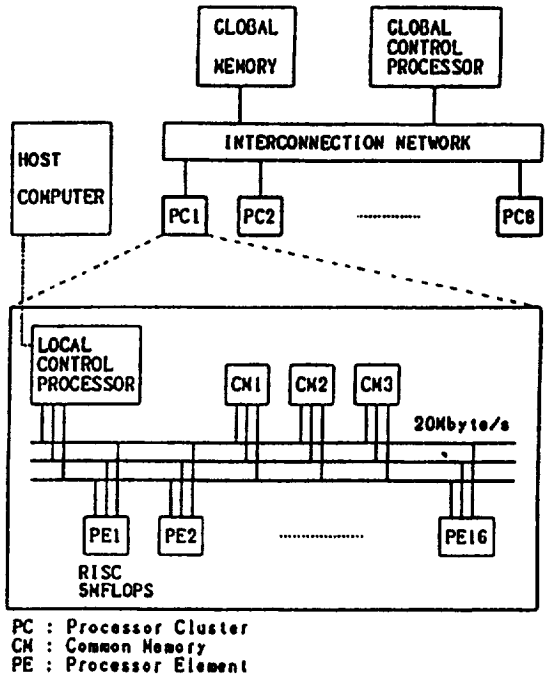


Fig.7 OSCAR (Optimally SCheduled Advanced multiprocessoR)

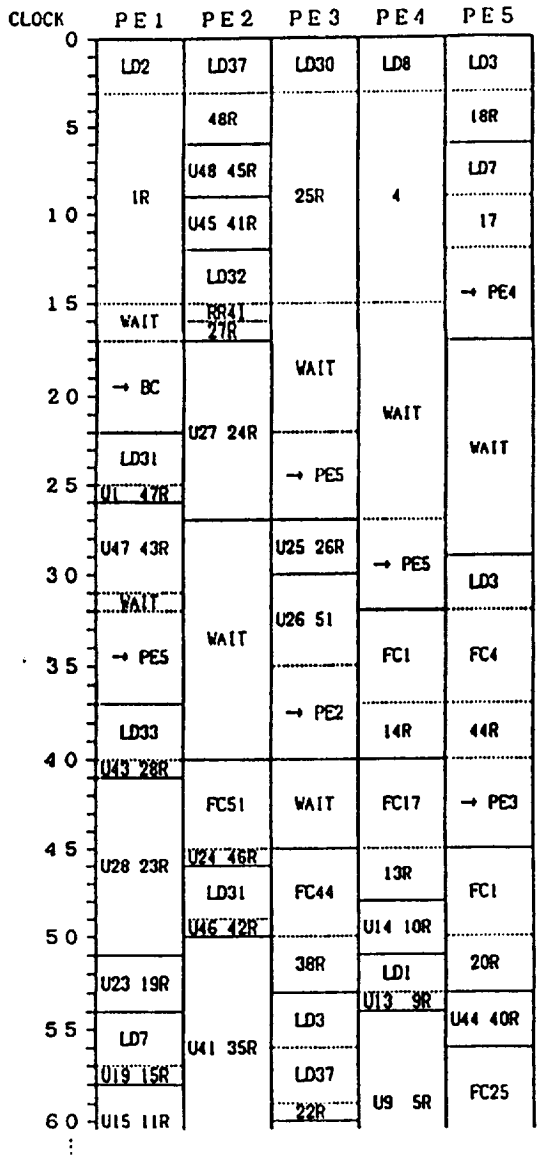


Fig.8 Execution image of machine codes on OSCAR.

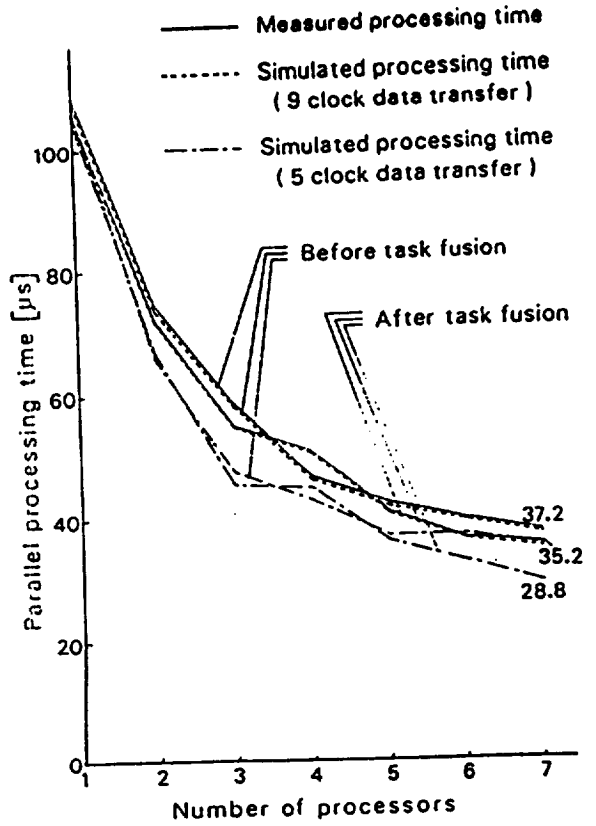


Fig.9 Parallel processing time measured on OSCAR and simulated parallel processing time.

PARALLEL AND VECTOR COMPUTATION FOR STOCHASTIC OPTIMAL CONTROL APPLICATIONS

F. B. HANSON
Department of Mathematics, Statistics, and Computer Science
University of Illinois at Chicago
P. O. Box 4348; M/C 249
Chicago, IL 60680
3rd CACC

Abstract. A general method for parallel and vector numerical solutions of stochastic dynamic programming problems is described for optimal control of general nonlinear, continuous time, multibody dynamical systems, perturbed by Poisson as well as Gaussian random white noise. Possible applications include lumped flight dynamics models for uncertain environments, such as large scale and background random atmospheric fluctuations. The numerical formulation is highly suitable for a vector multiprocessor or vectorizing supercomputer, and results exhibit high processor efficiency and numerical stability. Advanced computing techniques, data structures, and hardware help alleviate Bellman's *curse of dimensionality* in dynamic programming computations.

1. Introduction. The primary motivation for this research is to provide a provide a general computational treatment of stochastic optimal control applications in continuous time. In addition, fast and efficient methods are being developed by the optimization of stochastic dynamic programming algorithms for larger multibody problems. The optimization will help alleviate *Bellman's curse of dimensionality*, in that the computational problem greatly increases as the dimension of the state space increases. Optimization consists of parallelization and vectorization techniques to enhance performance on advanced computers, such as parallel processors and vectorizing supercomputers. General Markov random noise in continuous time consists of two kinds, Gaussian and Poisson. Gaussian white noise, being continuous but nonsmooth, is used to model background random fluctuations, such as turbulence and external field variations. Poisson white noise (its frequency spectrum is also flat like Gaussian noise), being discontinuous, is useful for modeling large random fluctuations, such as shocks, collisions, unexpected external events and large environmental changes. Our general feedback control approach combines the treatment of both linear and nonlinear (i.e., singular and nonsingular) control through the use of small and non-small quadratic costs. The methods also handle deterministic and stochastic control in the same code, making it convenient for checking the effects of stochasticity on the application. Some actual applications are models of resources in an uncertain environment [15], [11], [8]. Some potential applications are flight dynamics under random wind conditions [2] and other resource models [12].

The Markov, multibody dynamical system is illustrated in Figure 1 and is governed by the stochastic differential equation (SDE):

$$d\mathbf{y}(s) \;=\; \mathbf{F}(\mathbf{y},s,\mathbf{u})ds \;+\; G(\mathbf{y},s)d\mathbf{W}(s) \;+\; H(\mathbf{y},s)d\mathbf{P}(s)\,, \tag{1.1}$$

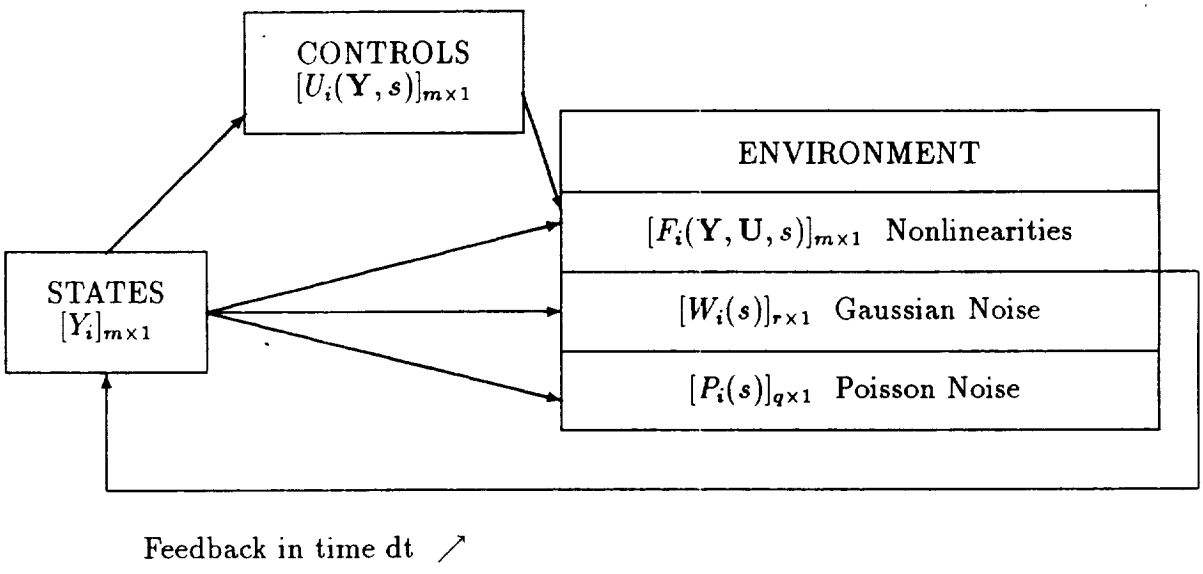


Figure 1: The multibody dynamical system.

$$\mathbf{y}(t) = x \; ; \; 0 < t < s < t_f \; ; \; \mathbf{y}(s) \in \mathcal{D}_y \; ; \; \mathbf{u} \in \mathcal{D}_u \; ,$$

where $\mathbf{y}(s)$ is the $m \times 1$ multibody state vector at time s starting at time t, $\mathbf{u} = \mathbf{u}(\mathbf{y}, s)$ is the $n \times 1$ feedback control vector, $\mathbf{W}$ is the r-dimensional normalized Gaussian white noise vector, $\mathbf{P}$ is the independent q-dimensional Poisson white noise vector with jump rate vector $[\lambda_i]_{q\times 1}$, $\mathbf{F}$ is the $m \times 1$ deterministic nonlinearity vector, G is an $m \times r$ diffusion coefficient array, and H is an $m \times q$ Poisson amplitude coefficient array.

The control criterion is the optimal expected cost performance,

$$V^*(\mathbf{x}, t) = \min_{\mathbf{u}} \left[MEAN_{\mathbf{P},\mathbf{W}} \left[V[\mathbf{y}, s, \mathbf{u}, \mathbf{P}, \mathbf{W}] \mid \mathbf{y}(t) = \mathbf{x} \right] \right] , \tag{1.2}$$

over some specified optimal control set $\mathcal{D}_u$, where the total cost is

$$V[\mathbf{y}, t, \mathbf{u}, \mathbf{P}, \mathbf{W}] = \int_t^{t_f} ds \; C(\mathbf{y}(s), s, \mathbf{u}(\mathbf{y}(s), s)) \; , \tag{1.3}$$

on the time horizon $(t, \; t_f)$. In (1.3), the instantaneous cost function $C = C(\mathbf{x}, t, \mathbf{u})$ is assumed to be a quadratic function of the control,

$$C(\mathbf{x}, t, \mathbf{u}) = C_0(\mathbf{x}, t) + \mathbf{C}_1^T(\mathbf{x}, t)\mathbf{u} + \tfrac{1}{2}\mathbf{u}^T C_2(\mathbf{x}, t)\mathbf{u} \; . \tag{1.4}$$

The unit cost of the control increases with $\mathbf{u}$ when C_2 is positive definite. For example, the cost criterion could be minimal fuel consumption, minimum distance to target or minimum time to target. No final salvage value is assumed at final time, so V is zero at $t = t_f$.

In addition, the deterministic, nonlinear dynamics in (1.1) are assumed to be linear in the controls,

$$\mathbf{F}(\mathbf{x}, t, \mathbf{u}) = \mathbf{F}_0(\mathbf{x}, t) + F_1(\mathbf{x}, t)\mathbf{u} \; , \tag{1.5}$$

but nonlinear in the multibody state variable $\mathbf{x}$.

For numerical purposes, it is more convenient to convert equations (1.1)-(1.2) to an effectively deterministic partial differential equation using Bellman's of optimality as illustrated in the optimization step from optimal control vector $\mathbf{U}^*$ to optimal expected costs V^* in Fig. 2. The Bellman functional PDE of stochastic dynamic programming,

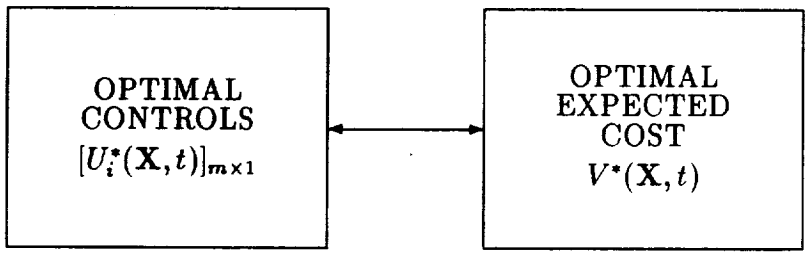


Figure 2: The optimization step from controls to costs.

$$
\begin{aligned}
0 \;=\; & V_t^* \;+\; L[V^*] \;\equiv\; V_t^* \;+\; \mathbf{F}_0^T \nabla V^* \;+\; \tfrac{1}{2} G G^T(\mathbf{x},t) : \nabla\nabla^T V^* \\
& +\; \sum_{l=1}^{q} \lambda_l \cdot [\; V^*(\mathbf{x} + \mathbf{H}_l(\mathbf{x},t),t) - V^*(\mathbf{x},t) \;] \qquad (1.6) \\
& +\; C_0 \;+\; (\tfrac{1}{2}\mathbf{U}^* - \mathbf{U}_R)^T C_2 \mathbf{U}^* \;,
\end{aligned}
$$

follows from the generalized *Itô* chain rule for Markov SDEs as in [7] and [15], where $\mathbf{U}^*$ is the optimal feedback control computed by constraining the unconstrained or regular control,

$$\mathbf{U}_R(\mathbf{x},t) \;=\; -C_2^{-1}(\mathbf{C}_1 \;+\; F_1{}^T \nabla V^*) \;, \qquad (1.7)$$

to the control set $\mathcal{D}_u$. In general, the Bellman equation (1.6) is nonlinear with discontinuous coefficients due to the last term, $(\frac{1}{2}\mathbf{U}^* - \mathbf{U}_R)^T C_2 \mathbf{U}^*$, in (1.6) and due to the compact relationship between the constrained, optimal control and the unconstrained, regular control,

$$U_i^*(\mathbf{x},t) = \min[U_{\max,i}, \min[U_{\min,i}, U_{R,i}(\mathbf{x},t)]], \qquad (1.8)$$

for $i = 1$ to n controls, where $\mathbf{U}_{\min}$ is the minimum control constraint vector and $\mathbf{U}_{\max}$ is the maximum. As the constraints are attained, the optimal control $\mathbf{U}^*$, changes from the regular control, $\mathbf{U}_R$, to the single bang control values, $\mathbf{U}_{min}$ or $\mathbf{U}_{max}$, which in general are functions of state and time. In (1.6), the symbol (:) denotes the scalar matrix product $A : B = \sum_{i=1}^{m} \sum_{j=1}^{m} A_{ij} B_{ij}$, assuming B is symmetric. It is important to note that the principal equation, the Bellman equation (1.6), is an exact equation for the optimal expected value V_* and does not involve any sampling approximations such as the use of random number generators in simulations.

Since there is no final salvage value and since (1.6) is a backward equation (unlike the usual diffusion equation, which is a forward equation), the final condition is that $V^*(x,t_f) = 0$ using (1.2) and (1.3). On the other hand, boundary conditions for the PDE of stochastic dynamic programming (1.6) are not as simple or as straightforward to state. This is because the boundary conditions vary significantly with the form the deterministic linearity function $\mathbf{F}$, the Gaussian noise W, and the Poisson noise P. Thus for treatment of general boundary conditions, it is most practical to directly integrate (1.6) for the special values of x, or to use the objective functional directly as defined in (1.2) and (1.3). The problem with boundary conditions is also present in stochastic application in continuous time, even when there is no control variable or optimization in the problem.

As the number of multibody state variables, m, increases, the spatial dimension rises, and computational difficulties are present that can compare to those of three-dimensional fluid dynamics computations. This is the famous Bellman's *curse of dimensionality* [3]. Thus there is a great need to make use of advanced-architecture computers, to use parallelization as well as vectorization. The Panel on Future Directions in Control Theory [6] stresses the importance of making gains in such areas as nonlinear control, stochastic control, optimal feedback control and computational methods for control. This paper is a preliminary report on our efforts to treat all of the above mentioned areas combined from the computational point of view.

2. Numerical Methods. The integration of the Bellman equation (1.6) is backward in time, because V^* is specified finally at the final time $t = t_f$, rather than at the initial time. A summary of the discretization in state and backward time is given below:

$$\begin{aligned}
\mathbf{x} \;&\longrightarrow\; \mathbf{X_j} = [X_{ij_i}]_{m\times 1} = [X_{i1} + (j_i - 1)\cdot DX_i]_{m\times 1}\,, \\
\mathbf{j} \;&=\; [j_i]_{m\times 1}\,, \quad \text{where } j_i = 1 \text{ to } M_i\,, \quad \text{for } i = 1 \text{ to } m\,; \\
t \;&\longrightarrow\; T_k = t_f - (k-1)\cdot DT\,, \quad \text{for } k = 1 \text{ to } K\,; \\
V^*(\mathbf{X_j}, T_k) \;&\longrightarrow\; V_{\mathbf{j},k}\,; \quad L[V^*](\mathbf{X_j}, T_{k+\frac{1}{2}}) \longrightarrow L_{\mathbf{j},k+\frac{1}{2}}\,;
\end{aligned} \tag{2.1}$$

where DX_i is the mesh size for state i and DT is the step size in backward time.

The numerical algorithm is a modification of the predictor corrector, Crank Nicolson methods for nonlinear parabolic PDEs in [5]. Modifications are made for the switch term and delay term calculations. Derivatives are approximated with an accuracy that is second order in the local truncation error, at all interior and boundary points. The Poisson induced functional or delay term, $V^*(\mathbf{x} + \mathbf{H}_l, t)$, changes the local attribute of the usual PDE to a global attribute, such that the value at a node $[\mathbf{X} + \mathbf{H}_l]_\mathbf{j}$ will in general not be a node. Linear interpolation, with special handing of point near the boundaries, maintains the numerical integrity compatible with the numerical accuracy of the derivative approximations. Even though the Bellman equation (1.6) is a single PDE, the process of solving it not only produces the optimal expected value V^*, but also the optimal expected control law U^*. This is because the PDE is a functional PDE, in which the computation of the regular control is fed back into the optimal value and the optimal value feeds back into regular control through its gradient. The nonstandard part of the algorithm is to decompose this tightly coupled analytical feedback so that both the value and the control can be calculated by successive

iterations, such that each successive approximation of one improves the next approximation of the other. While our procedure may look superficially like a standard application of finite differences, it is not due to the nonstandard features mentioned above. For these reasons, we are not aware of any other successful stochastic dynamic programming code that treats anywhere near the generality of applications that we treat. Variations of this algorithm have been successfully utilized in [15] and [8].

Prior to calculating the values, $V_{\mathbf{j},k+1}$, at the new $(k+1)^{st}$ time step for $k = 1$ to $K-1$, the old values, $V_{\mathbf{j},k}$ and $V_{\mathbf{j},k-1}$, are assumed to be known, with $V_{\mathbf{j}0} \equiv V_{\mathbf{j}1}$. The algorithm begins with an *extrapolator (x) start*:

$$V^{(\mathbf{x})}_{\mathbf{j},k+\frac{1}{2}} = \tfrac{1}{2}(3 \cdot V_{\mathbf{j},k} - V_{\mathbf{j},k-1}) , \tag{2.2}$$

which are then used to compute updated values of the gradient of V^*, the second order derivatives, Poisson functional terms (V^* at $(\mathbf{x}+\mathbf{H})$), regular controls $\mathbf{U}_R$, optimal controls $\mathbf{U}^*$, and finally the new value of the Bellman equation spatial functional $L_{\mathbf{j},k+0.5}$. The extrapolation step greatly speeds up the convergence of the corrector step, except at the initial step. These evaluations are used in the *extrapolated predictor (xp) step*:

$$V^{(\mathbf{xp})}_{\mathbf{j},k+1} = V_{\mathbf{j},k} + DT \cdot \tfrac{1}{2} L^{(\mathbf{x})}_{\mathbf{j},k+\frac{1}{2}} . \tag{2.3}$$

which are then used in the *predictor evaluation (xpe) step*:

$$V^{(\mathbf{xpe})}_{\mathbf{j},k+\frac{1}{2}} = \tfrac{1}{2}(V^{(\mathbf{xp})}_{\mathbf{j},k+1} + V_{\mathbf{j},k}) , \tag{2.4}$$

an approximation which preserves numerical accuracy and which is used to evaluate all terms comprising $L_{\mathbf{j},k+0.5}$. The evaluated predictions are used in the *corrector (xpec) step*:

$$V^{(\mathbf{xpec},\gamma+1)}_{\mathbf{j},k+1} = V_{\mathbf{j},k} + DT \cdot L^{(\mathbf{xpe},\gamma)}_{\mathbf{j},k+\frac{1}{2}} \tag{2.5}$$

for $\gamma = 0$ to γ_{max} while stopping criterion unmet, with *corrector evaluation (xpece) step*:

$$V^{(\mathbf{xpece},\gamma+1)}_{\mathbf{j},k+\frac{1}{2}} = \tfrac{1}{2}(V^{(\mathbf{xpec},\gamma+1)}_{\mathbf{j},k+1} + V_{\mathbf{j},k}) . \tag{2.6}$$

The predicted value is taken as the zeroth correction. The stopping criterion for the corrections is a heuristically motivated comparison to a predictor corrector convergence criterion for a linearized, constant coefficient PDE [13]. The stopping criterion is computed with a robust mesh selection method, so that only a few corrections are necessary. The selection of the mesh ratio, the ratio of the time step DT to the norm of the space or state step $D\mathbf{X}$, guarantees that the corrections will converge whether the Bellman equation (1.6) is parabolic-like (with Gaussian noise) or hyperbolic-like (without Gaussian), according to whether or not an explicit second derivative is in the equation.

Parallelization and vectorization of the algorithm was done by what might be called the "Machine Computational Model Method," i.e., tuning the code to optimizable constructs

that are automatically recognized by the compiler, with the Alliant FX/8 vector multiprocessor [1] in mind. All inner double loops were reordered to fit the Alliant *concurrent outer - vector inner* (*COVI*) model. All non-short single loops were made *vector-concurrent* Short loops became *scalar-concurrent* only. Multiple nested loops were reordered with the two largest loops innermost. A total of 37 out of 39 loops was optimized. Detailed results for a two-state and two-control model with Poisson noise are reported in [9]. Very similar techniques work for the vectorizing Cray supercomputers, except that only inner loops are vectorized. Vectorizing and parallelizing techniques are very similar, because vectorization is really a primitive kind of parallelization and because both are inhibited by many of the same types of data dependencies.

The relative performance of column oriented versus row oriented code is discussed in [10]. Dongarra, Gustavson, and Karp [4] have demonstrated that loop reordering gives vector or supervector performance for standard linear algebra loops on a Cray 1 type column oriented FORTRAN environment with vector registers. However, for the stochastic dynamic programming application, the dominant loops are non-standard linear algebra loops, so that the preference for column oriented loops is not a rule, as demonstrated on the Alliant vector multiprocessor [10].

Current efforts are concentrated on implementing the code on the Cray X-MP/48 and Cray 2 for more general multi-state and multi-control applications. In order to implement the code for arbitrary state space dimension, a more flexible data structure is needed for the problem arrays, $\mathbf{F}$, G and H, as well as for the solution arrays, V along with its derivatives and the control $\mathbf{U}$. In the straight-forward, original data structure, an array like the nonlinearity vector requires one index, $js(is)$, to denote a numerical node for each state variable is:

$$\mathsf{F(is, js(1), js(2), \cdots, js(m))} \tag{2.7}$$

for each state equation, $is = 1$ to m. It is assumed that there are a common number $M = M_1 = \cdots = M_m$ of nodes per state, so that $js(is) = 1$ to M for $is = 1$ to m states. As a consequence, the typically dominant loops containing the nonlinearity function $\mathbf{F}$, the solution gradient DV or similarly sized array are nested to a depth of at least $m + 1$. A typical loop has the form

```
      do 1 i = 1, m
         do 1 j1 = 1, M
                 :
            do 1 jm = 1, M
                    :
1                F(i,j1,j2,···,jm) = ·····
```

This state size dependent loop nest depth level of $m + 1$ inhibits the development of general multibody algorithms, especially when the state size m is incremented and the number of loops in each nest have to be changed. Also, vectorization is inhibited for compilers that vectorize only the most inner loop. Parallel and vector optimization is important, due to the size of the work load, which is $\mathcal{O}(m \cdot M^m)$, for the dominant loop illustrated above. As the

number of states grows the computational load will grow like some multiple of

$$m \cdot M^m = m \cdot e^{m \ln(M)},$$

i.e., the load grows exponentially in the number of states m. This exponential growth is merely a quantitative expression of *Bellman's curse of dimensionality.*

One way around this inhibiting structure (2.7) is to use a vector data structure:

$$\mathsf{FV(is,jv)} \tag{2.8}$$

for the nonlinearity vector as an example, such that all the numerical nodes are collected into a single vector indexed by the global state index jv, where $jv = 1$ to M^m over all state nodes. Assuming that the number of nodes per state are fixed at M, then for a fixed set of state node indices $\{js(1), js(2), \cdots, js(m)\}$, the global state vector index is computed from the direct mapping formula

$$jv = \sum_{i=1}^{m} (js(i) - 1) \cdot M^{i-1} + 1, \tag{2.9}$$

in the case of fixed state mesh size, $M_i = M$ for all states i.

Both the direct mapping from the original data structure to the vector data structure and the inverse mapping are needed to compute the amplitude functions, $\mathbf{F}$, G and H, as well as the derivatives of V^*, because these quantities depend on the original formulation. The pseudo-inverse of the vector index in (2.9) can be shown to permit the recovery of the individual state indices by way of integer arithmetic:

$$js(is; jv) = 1 + [jv - 1 - \sum_{i=is+1}^{m} (js(i; jv) - 1) \cdot N^{i-1}]/N^{is-1}, \tag{2.10}$$

recursively, for $is = m$ to 1, by back substitution, with $\sum_{i=m+1}^{m} a_i \equiv 0$. The vector data structure of (2.8) to (2.10) results in major do loop nests of the order of 1 to 2, rather than order of $m + 1$. A typical vector data structure loop has the form

```
      do  2  i = 1, m     ! parallel loop.
          do  2  jv = 1, M**m     ! vector loop.
                    :
2                 FV(i,jv) = ......
```

resulting in a reduction of the loop nest depth from $m + 1$ to 2, independent of the number of states m. Preliminary implementation of the vector data structure is available on the Alliant multiprocessor and on the Cray X-MP/48.

One major disadvantage of the vector data structure given in (2.10) is that the largest degree of parallelism available to a parallel processor or multiprocessor in the most outer or state number loop is m, the number of states. This task load can be better scheduled on parallel processors by block decomposition or strip mining of the vector data structure loop in the index iv, so that the single inner loop is split into two evenly balanced loops (cf., Polychronopoulos [14]). Thus, dividing the vector data structure into blocks can enhance

parallelism. Let MBLK be the number of state nodes in each block and then the total number of blocks will be

$$\text{NBLK} = M^m/\text{MBLK},$$

assumed to be an integer for simplicity. Consequently, the blocked version of the typically dominant loop will have the form

```
      do 3 i = 1, m
         do 3 jblk = 1, MBLK    ! parallel loop.
            jv1 = 1 + MBLK*(jblk - 1)
            jv2 = MBLK*jblk
            do 3 jv = jv1, jv2    ! vector loop.
                  ⋮
3                FV(i,jv) = ·····
```

This form should result in better parallel optimization when there are more than m available parallel processors.

The advantages of the algorithm is that it 1) permits the treatment of general continuous time Markov noise or deterministic problems without noise in the same code, 2) maintains feedback control, 3) permits the cheap control limit to linear singular control to be found from the same quadratic cost code, 4) stable mesh selection can be used to control the number of corrector steps, and 5) produces very vectorizable and parallelizable code whose performance is described in the next section.

3. Results and Discussion. The stochastic dynamic programming code arose from renewable resource modeling problems of Hanson and co-worker Ryan, with various one-state, one-control models treated in [15] and [11]. Two-state, two-control models were treated by Hanson [8]. In the two-state model [8], the two controls represent removals from the system by respective commercial and recreational users of the system. Poisson noise is used to represent natural catastrophic events. Applications to aerospace problems only entails modification of the dynamical system and performance criteria input by appropriate aerospace input functions and parameters.

The dynamic programming code has been optimized for parallelization and vectorization [9] using Hanson's two-state model [8] as a test example, and the Alliant FX/8 vector multiprocessor as the advanced hardware. The Alliant FX/8 at the Advanced Computing Research Facility (ACRF) at Argonne National Laboratory was used for benchmarking the code. This Alliant FX/8 has eight vector computing elements (*CEs*). Each of the *CEs* has eight vector registers whose length is 32 eight-byte elements, and the *CEs* are connected to a 128 KB cache. Some automatic parallelization and vectorization is performed, but significant increases are still attainable by the removal of optimization hindering data dependencies. Benchmark performance was measured for many mesh sizes and on all processor configurations. Almost all loops were of the highly optimized parallel and vector type for the Alliant. Over 65% efficiency was achieved over a wide range of tests [9]. The temporal mesh was chosen to be about four times more refined than the spatial mesh, $K = 4 \cdot (M-1) + 1$, for a fixed number of spatial nodes M and for constant numerical stability conditions. In

addition, vector stride effects (resonance effects related to multiples of the vector register length of 32 on the FX/8) were found with non-standard performance in both column and row referencing environments [10].

The present results have been obtained for a three-state, three-control modification of Hanson's two-state resource model [8] and by implementing the vector data structure mentioned above. The present application contains a new interacting state with competition. The present code is in a form where it is much more convenient to change the application, the advanced computer intrinsics, and the number of states.

Table 1 compares the performance of the code on the ACRF Alliant FX/8 vector multiprocessor at Argonne National Laboratory, the NCSA Cray X-MP/48 vector supercomputer at Urbana, and the University of Illinois at Chicago IBM3081K as a scalar uniprocessor reference. The Cray X-MP/48 is a four processor pipelined vector multiprocessor, but the use of the X-MP is much more costly to use in parallel than the Alliant and so only single processor results are reported here for the X-MP. The Cray executing on one vector processor outperforms the Alliant using either one vector processor or the full eight vector processors, due to the more powerful pipelined processing unit on the Cray. The advantages of block decomposition with MBLK = 32 for eight Alliant processors are illustrated in the table, where the eight processor time has been reduced from about 52 to 33 seconds when $M = 16$, while the one processor time has increased dramatically for the block method. The IBM3081K scalar uniprocessor is much slower when $M = 8$ unblocked spatial nodes than any of the Alliant or Cray values at $M = 8$. However, as the spatial mesh size is refined to $M = 16$ spatial points, with a corresponding increase in work load, the IBM3081K performs between the one and eight processor Alliant, but still significantly below the CRAY performance.

Table 1: Comparative Performance of IBM 3081K, Alliant and Cray, for three state model.

Nodes		Method	IBM 3081K	Alliant FX/8		Cray X-MP
state	time		vs fortran, opt(3)	fortran -O		cft77
M	K		$p = 1$	$p = 1$	$p = 8$	$p = 1$
8	29	unblocked	38.513	8.653	2.980	0.144
16	61	unblocked	85.377	147.391	51.619	2.058
8	29	blocked	—	13.693	1.998	—
16	61	blocked	—	223.426	32.729	—

The performance of the stochastic programming code under parallel and vector operation is investigated in more detail on the ACRF Alliant FX/8, which has better parallel capability than the Cray X-MP/48. The Cray X-MP/48 is also a vector multiprocessor, but the multiprocessing features are not as easily accessed as on the Alliant, where parallelization is more transparent. In Figure 3, the blocked and unblocked code is compared on the Alliant FX/8 with time $T(p)$ plotted against the number of processors p. The unblocked code runs faster as the number of processors increases from one, but then ceases to run any

faster beyond $p = 3$ processors due to the fact that the maximum parallelism available is the three iterations in the three-state outer loop. The blocked code, using a block size of MBLK = 32 (the vector register length on the Alliant) runs faster the more processors used out of the eight vector processors. However, the unblocked code is faster for $p < 5$, but slower for $p > 5$. The trade-off point between the blocked and unblocked code is $p = 5$, with the block overhead slowing down the code for $p < 5$, but the benefit of parallelism is found for $p > 5$.

Figure 4 shows the speedup, $S(p) = T(1)/T(p)$, versus the number of processors p. The unblocked code clearly exhibits a speedup plateau for $p \geq 3$ and the unblocked code exhibits nearly ideal speedup, $S(p) \simeq p$ for all p. However, this figure illustrates the danger of comparing speedups, because the unblocked case is better for $p < 5$ as demonstrated in Figure 3. In Figure 5, the efficiency, $E(p) = S(p)/p$ or speedup per processor, versus the number of processors p is shown. Again, the blocked efficiency is much higher than the unblocked efficiency, independent of the actual performance.

4. CONCLUSIONS. Stochastic dynamic programming algorithm can be optimized to permit numerical solution of larger state space problems using vector multiprocessors. In order to handle a large number of state variables, a large number of parallel processors would be desirable, but Bellman's *curse of dimensionality* appears to very much weakened. Parallelization, vectorization, and general supercomputing are important in the solution of the larger problems. Robust mesh selection techniques are necessary to achieve stable algorithms. These techniques are generally applicable to other vector and parallel computers. The general code is valid for general Markov noise in continuous time, feedback control, nonlinear dynamics, nonlinear control and the cheap control limit.

Future directions include applications to aerospace problems, improved development of general code for an arbitrary number of state variables, enhanced code portability, extensions to Kalman filtering for imperfect observations, and optimization for other advanced architectures.

Acknowledgements. Work was supported, in part, by several Faculty Research Participantships, a Faculty Research Leave at Argonne National Laboratory Advanced Computing Research Facility, and by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U. S. Department of Energy, Under Contracts W-31-109-38 and DE-AC05-84-R21400; the National Science Foundation Grant DMS-8806099; the National Center for Supercomputing Applications in Urbana. The author wishes to acknowledge the work of C.-W. Leung for the advanced analysis of the vector data structure and for investigations of applications of Cray multitasking to the problem, and to S.-L. Chung for continued development of optimizations for the algorithm.

REFERENCES

[1] Alliant, **FX/FORTRAN Programmer's Handbook**, Alliant Computer Systems Corporation, Acton, Mass., 1985.

[2] M. Athans, D. Castanon, K. P. Dunn, C. S. Greene, W. H. Lee, N. R. Sandell, Jr.,

and A. S. Willsky, *The stochastic control of the F-8C aircraft using a multiple model adaptive control (MMAC) method - Part I: Equilibrium flight*, **IEEE Trans. Autom. Control**, vol. AC-22, pp. 768-780, 1977.

[3] R. E. Bellman, **Adaptive Control Processes: A Guided Tour**. Princeton: Princeton University Press, 1961.

[4] J. J. Dongarra, F. G. Gustavson, and A. Karp, *Implementation of linear algebra algorithms of dense matrices on a vector pipeline machine*, **SIAM Rev.**, vol. 26, pp. 91-112, 1984.

[5] J. Douglas, Jr., and T. DuPont, *Galerkin methods for parabolic equations*, **SIAM J. Num. Anal.**, vol. 7, pp. 575-626, 1970.

[6] **Future Directions in Control Theory: A Mathematical Perspective**, W. H. Fleming, Chairman. Philadelphia: Society for Industrial and Applied Mathematics, 1988.

[7] I. I. Gihman and A. V. Skorohod, **Controlled Stochastic Processes**. New York: Springer-Verlag, 1979.

[8] F. B. Hanson, *Bioeconomic model of the Lake Michigan alewife fishery*, **Can. J. Fish. Aquat. Sci.**, vol. 44, Suppl. II, pp. 298-305, 1987.

[9] F. B. Hanson, *Computational dynamic programming for stochastic optimal control on a vector multiprocessor*, **Argonne National Laboratory, Mathematics and Computer Science Division Technical Memorandum ANL/MCS-TM-113**, June 1988, 26 pages.

[10] F. B. Hanson, *Parallel computation for stochastic dynamic programming: Row versus column code orientation*, in **Proceedings 1988 Conference on Parallel Processing, Vol. III Algorithms and Applications**, D. H. Bailey, Editor. University Park: Pennsylvania State University Press, 1988, pp. 117-119.

[11] F. Hanson and D. Ryan, *Optimal harvesting with density dependent random effects*, **Natural Resource Modeling**, vol. 2, No. 3, pp. 439-455, 1988.

[12] D. Ludwig, *Optimal harvesting of a randomly fluctuating resource I: Application of perturbation methods*, **SIAM J. Appl. Math.**, vol. 37, pp. 166-184, 1979.

[13] K. Naimipour and F. B. Hanson, *Convergence of a numerical method for the Bellman equation of stochastic optimal control with quadratic costs*, In Preparation, 1989.

[14] C. D. Polychronopoulos, *Parallel Programming and Compilers*. Boston: Kluwer Academic Publishers, 1988, pp. 26-27.

[15] D. Ryan and F. B. Hanson, *Optimal harvesting of a logistic population in an environment with stochastic jumps*, **J. Math. Biol.**, vol. 24, pp. 259-277, 1986.

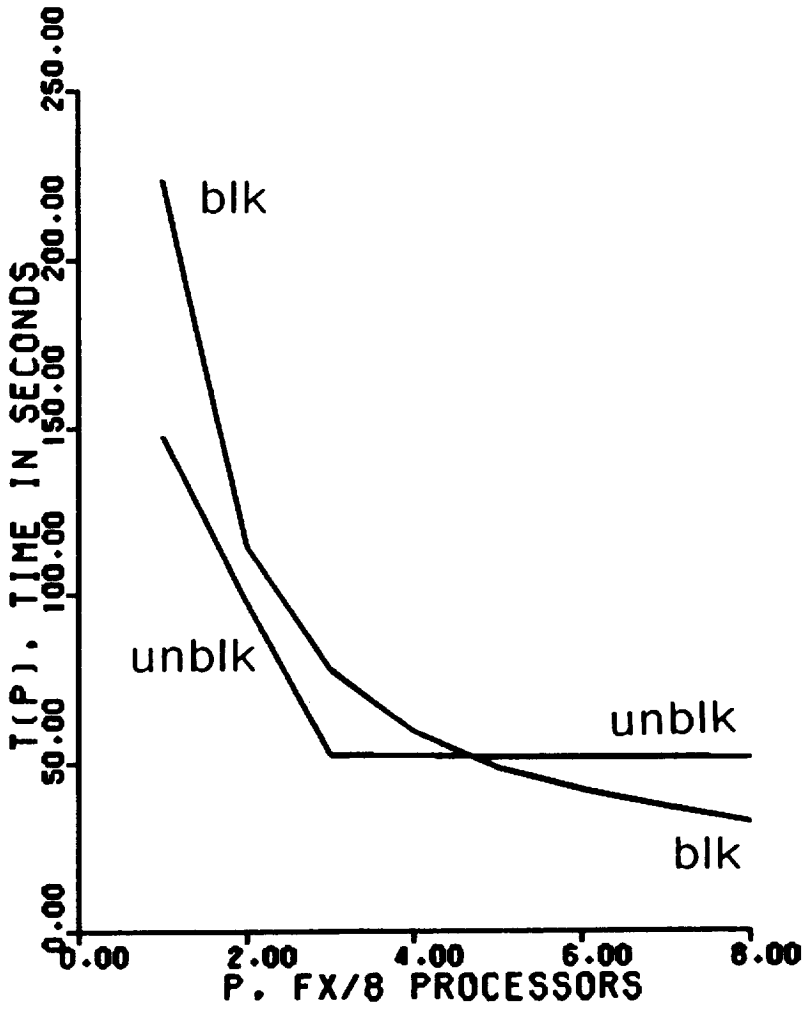


Figure 3: Comparison of blocked (blk) and unblocked (unblk) versions of the code. Time $T(p)$ is in seconds and p is the number of processors. Results are for $m = 3$ states, $M = 16$ spatial nodes and $K = 61$ temporal nodes.

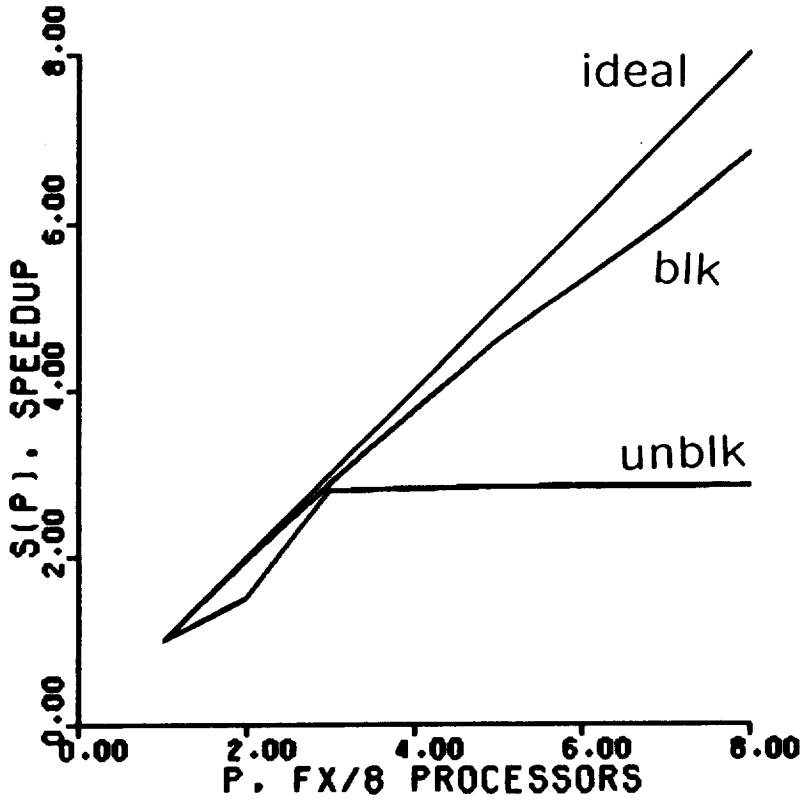


Figure 4: Speedups for blocked (blk) and unblocked (unblk) versions of the code. Speedup is denoted by $S(p) = T(1)/T(p)$ and p is the number of processors. The notation (ideal) denotes the ideal case, $S(p) = p$. Results are for $m = 3$ states, $M = 16$ spatial nodes and $K = 61$ temporal nodes.

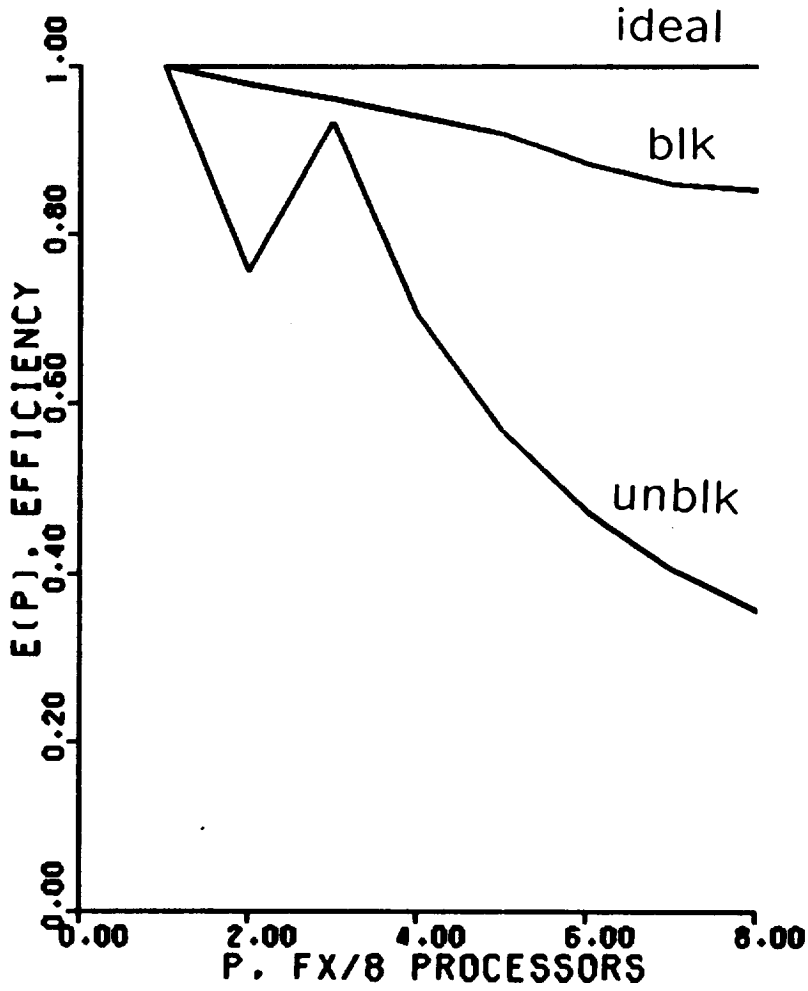


Figure 5: Efficiency for blocked (blk) and unblocked (unblk) versions of the code. Efficiency is denoted by $E(p) = S(p)/p$ and p is the number of processors. The notation (ideal) denotes the ideal case, $E(p) = 1$. Results are for $m = 3$ states, $M = 16$ spatial nodes and $K = 61$ temporal nodes.

A Robot Arm Simulation with a Shared Memory Multiprocessor Machine

Sung-Soo Kim and Li-Ping Chuang
The Center for Simulation and Design Optimization
The University of Iowa

Abstract

A parallel processing scheme for a single chain robot arm is presented for high speed computation on a shared memory multiprocessor. A recursive formulation that is derived from a virtual work form of the d'Alembert equations of motion is utilized for robot arm dynamics. A joint drive system that consists of a motor rotor and gears is included in the arm dynamics model, in order to take into account gyroscopic effects due to the spinning of the rotor. The fine grain parallelism of mechanical and control subsystem models is exploited, based on independent computation associated with bodies, joint drive systems, and controllers. Efficiency and effectiveness of the parallel scheme are demonstrated through simulations of a telerobotic manipulator arm. Two different mechanical subsystem models, i.e., with and without gyroscopic effects, are compared, to show the trade-off between efficiency and accuracy.

A Unifying Framework for Rigid Multibody Dynamics and Serial and Parallel Computational Issues

Amir Fijany and Abhinandan Jain
Jet Propulsion Laboratory/California Institute of Technology

Abstract

In this paper we present a unifying framework for various formulations of the dynamics of open-chain rigid multibody systems and assess their suitability for serial and parallel processing. The framework is based on the derivation of intrinsic, i.e., coordinate-free, equations of the algorithms which provides a suitable abstraction and permits a distinction to be made between the computational redundancy in the intrinsic and extrinsic equations. A set of spatial notation is used which allows the derivation of the various algorithms in a common setting and thus clarifies the relationships among them. The three classes of algorithms viz., $O(n)$, $O(n^2)$ and $O(n^3)$ or the solution of the dynamics problem are investigated. We begin with the derivation of $O(n^3)$ algorithms based on the explicit computation of the mass matrix and it provides insight into the underlying basis of the $O(n)$ algorithms. From a computational perspective, the optimal choice of a coordinate frame for the projection of the intrinsic equations is discussed and the serial computational complexity of the different algorithms is evaluated. The three classes of algorithms are also analyzed for suitability for parallel processing. It is shown that the problem belongs to the class of NC and the time and processor bounds are of $O(\log_2^2(n))$ and $O(n^4)$, respectively. However, the algorithm that achieves the above bounds is not stable. We show that the fastest stable parallel algorithm achieves a computational complexity of $O(n)$ with $O(n^4)$, respectively. However, the algorithm that achieves the above bounds is not stable. We show that the fastest stable parallel algorithm achieves a computational complexity of $O(n)$ with $O(n^2)$ processors, and results from the parallelization of the $O(n^3)$ serial algorithm.

Parallel Algorithms and Architecture for Computation of Manipulator Forward Dynamics

Amir Fijany and **Antal K. Bejczy**

Jet Propulsion Laboratory, California Institute of Technology

Abstract- In this paper parallel computation of manipulator forward dynamics is investigated. Considering three classes of algorithms for the solution of the problem, that is, the $O(n)$, the $O(n^2)$, and the $O(n^3)$ algorithms, parallelism in the problem is analyzed. It is shown that the problem belongs to the class of *NC* and that the time and processors bounds are of $O(\log_2^2 n)$ and $O(n^4)$, respectively. However, the fastest stable parallel algorithms achieve the computation time of $O(n)$ and can be derived by parallelization of the $O(n^3)$ serial algorithms. Parallel computation of the $O(n^3)$ algorithms requires the development of parallel algorithms for a set of fundamentally different problems, that is, the Newton-Euler formulation, the computation of the inertia matrix, decomposition of the symmetric, positive definite matrix, and the solution of triangular systems. Parallel algorithms for this set of problems are developed which can be efficiently implemented on a unique architecture, a triangular array of $n(n+1)/2$ processors with a simple nearest-neighbor interconnection. This architecture is particularly suitable for VLSI and WSI implementations. The developed parallel algorithm, compared to the best serial $O(n)$ algorithm, achieves an asymptotic speedup of more than two orders-of-magnitude in the computation the forward dynamics.

I. INTRODUCTION

The manipulator forward dynamics problem concerns the determination of the motion of the mechanical system resulting from the application of a set of joint forces/torques which is essential for dynamic simulation. The motivation for devising fast algorithms for forward dynamics computation stems from applications which require extensive off-line simulation as well as applications which require real-time dynamic simulation capability. In particular, for many anticipated space teleoperation applications, a faster-than-real-time simulation capability will be essential. In fact, in the presence of unavoidable delay in information transfer, such a capability would allow a human operator to preview a number of scenarios before run-time [1].

The forward dynamics problem can be stated as follows: Given the vectors of actual joint positions (Q) and velocities ($\dot{Q}$), the external force (f_E) and moment (n_E) exerted on the End-Effector (EE), and the vector of applied joint

forces/torques (τ), find the vector of joint accelerations ($\ddot{Q}$). Integrating the vector of joint accelerations leads to the new values for Q and $\dot{Q}$, and the process is then repeated for the next τ. The first step in computing the forward dynamics is to derive a linear relation (for the given manipulator configuration described by the vector of joint positions) between the vector of joint accelerations and the vector of applied inertial forces/torques.

Given the dynamic equations of motion as

$$A(Q)\ddot{Q} + C(Q,\dot{Q}) + G(Q) + J^t(Q)F_E = \tau \qquad (1)$$

and defining the bias vector b as

$$b = C(Q,\dot{Q}) + G(Q) + J^t(Q)F_E \qquad (2)$$

the linear relation is derived:

$$A(Q)\ddot{Q} = \tau - b = \Gamma \qquad (3)$$

where Q, $\dot{Q}$, and $\ddot{Q}$ are nx1 vectors and F_E, a 6x1 vector, is a combined representation of f_E and n_E. A(Q) is an nxn symmetric, positive definite, inertia matrix, and J is the 6xn Jacobian matrix (t denotes matrix transpose). The bias vector b represents the contribution due to coriolis and centrifugal terms $C(Q,\dot{Q})$, gravitional terms G(Q), and the external force and moment. Hence, in Eq. (3), Γ is the nx1 vector of applied inertia forces/torques. The bias vector b can be obtained by solving the inverse dynamics problem, using the Newton-Euler (N-E) formulation [2], while setting the vector of joint accelerations to zero. The computation of the vectors b and Γ represent the common first step in any algorithm for solving the forward dynamics problem.

The proposed algorithms for the forward dynamics problem differ in their approaches to solving Eq. (3), which directly affects their asymptotic computation complexity. These algorithms can be classified as:

1) The $O(n)$ algorithms [3]-[6] which, by taking a more explicit advantage of the structure of problem, e.g., by using the Articulated-Body Inertia [3]-[4] and recursive factorization and inversion of the inertia matrix [5]-[6], solve Eq. (3) in $O(n)$ steps without explicit computation and inversion of the inertia matrix.

2) The $O(n^2)$ conjugate gradient algorithms [7, 10] which iteratively solve Eq. (3) without explicit computation and inversion of the inertia matrix. The conjugate gradient algorithm is guaranteed to converge to the solution in at most n iterations which, given the $O(n)$ computational complexity of each iteration, leads to an overall $O(n^2)$ computational complexity.

3) The $O(n^3)$ algorithms [7] which solve Eq. (3) by explicit computation and inversion of the inertia matrix, leading to an $O(n^3)$ computational complexity.

However, any analysis of the relative efficiency of these algorithms should be based on the realistic size of the problem, i.e., the number of Degree-Of-Freedom (DOF), rather than the asymptotic complexity. In fact, the comparative study in [3]-[4] shows that the $O(n^3)$ Composite Rigid-Body algorithm is the most efficient for n less than 12. It should be pointed out

that efficiency of the $O(n^3)$ and $O(n^2)$ algorithms has been recently improved [9]-[10]. However, despite these improvements, even the fastest $O(n^3)$ algorithm is far from providing the efficiency required for real-time or faster-than-real-time simulation. This observation clearly suggests that the exploitation of a high degree of parallelism in the computation is the key factor in achieving the required efficiency.

The analysis of the efficiency of the different algorithms for parallel computation is more complex than that for serial computation. In the next section, the three classes of algorithms are analyzed based on their efficiency for parallel computation and it is shown that the $O(n^3)$ algorithms are also the most efficient for parallel computation. However, parallelization of the $O(n^3)$ algorithms represents a challenging problem since it requires the development of parallel algorithms for computation of a set of fundamentally different problems, i.e., the N-E formulation, the inertia matrix, the factorization of the inertia matrix, and the solution of triangular systems.

Lee and Chang [15] were first to investigate the computation of the forward dynamics by parallelization of the $O(n^3)$ algorithms. Considering an SIMD architecture with n processors interconnected through a generalized-cube network, a modified version of their $O(\log_2 n)$ algorithm in [16] and an $O(n)$ parallel version of the Composite Rigid-Body algorithm were developed for computation of the N-E formulation and the inertia matrix. A parallel $O(n^2)$ Cholesky algorithm and the $O(n)$ Column-Sweep algorithms were also proposed for the factorization of the inertia matrix and the solution of the resulting triangular systems, leading to an $O(n^2)$ complexity of the overall computation. However, the main drawbacks of the proposed algorithms reside in the complexity of the required interconnection network and the $O(n^2)$ communication complexity which mainly results from the excessive data alignment needed for different algorithms.

In this paper, we present a set of efficient parallel algorithms for the computation of the forward dynamics, using the $O(n^3)$ algorithms, which can be implemented on a two-dimensional array of $n(n+1)/2$ processors with a nearest neighbor interconnection. The overall of communication complexity, even with such simple interconnection structure, is limited to $O(n)$ and no additional data alignment between the computation of the different algorithms is required, which further reduces the overhead in the parallel computation. A new algorithm for computation of the inertia matrix is developed which, though not efficient for serial processing, achieves the best performance for parallel computing in terms of both computation and communication complexity while demanding simple architectural features for its implementation. The parallel algorithm for computing the inertia matrix achieves the time lower bound of $O(\log_2 n)+O(1)$ on the processor array. Synchronous data-flow parallel algorithms are also developed for factorization of the inertia matrix and the solution of the resulting triangular systems on the processor array.

This paper is organized as follows. In Section II, parallelism and time and processors bounds in the computation of the forward dynamics are investigated. In Section III, parallel algorithm for computation of the inertia matrix is developed. In Section IV, parallel computation of the bias vector and the linear system solution are briefly discussed. Finally, some concluding remarks are made in Section V.

II. PARALLELISM IN FORWARD DYNAMICS COMPUTATION

A. Time and Processor Bounds in the Computation

The analysis of time and processors bounds in parallel computation of a given problem is of fundamental theoretical importance. It can determine the inherent parallelism in the problem and the bound on the number of processors required for exploiting maximum parallelism and achieving the time lower bound in the computation. However, besides the theoretical importance, it can also provide, as is the case for forward dynamics problems, useful insights into devising more practical and efficient parallel algorithms (in the sense of both computation time and number of processors) for the problem.

Let P denote the class of problems that can be solved sequentially in a time bounded by a polynomial of the input size, n. Also, let NC (for "Nick's Class" [18]) stand for the class of problems that can be solved in parallel in a time of $O(\log_2^k n)$, for some constant k, with a number of processors bounded by a polynomial of n. One open question regarding the complexity of parallel algorithms is whether $P = NC$, which is thought to be very unlikely [19]. It is clear that $NC \subseteq P$. For k = 1, the time of $O(\log_2 n)+O(1)$ represents the natural time lower bound in the computation. However, most of the kinematic and dynamic problems in robotics belong to the class of NC [8]. Furthermore, it is possible to devise parallel algorithms which achieve the time lower bound of $O(\log_2 n)+O(1)$ in solving these problems [8,14,16,17]. In the following, we study the time and processors bounds in the computation of the forward dynamics by different algorithms.

Using the N-E formulation, the bias vector can be computed in a time of $O(\log_2 n)+O(1)$ with $O(n)$ processors [15]-[16]. This implies that the time and processors bounds in the forward dynamics computation are determined by those in the solution of Eq. (3). Note that, with $O(n)$ processors, the integration of the computed joint accelerations can be performed in a time of $O(1)$.

The solution of Eq. (3) by the $O(n)$ algorithms results in a set of first-order nonlinear recurrences which can be represented (at an abstarct level) as

$$X_i = C_i + \phi_2(X_{i+1})/\phi_1(X_{i+1}) = C_i + \phi(X_{i+1}) \qquad (4)$$

where C_i is constant, ϕ_1 and ϕ_2 are polynomials of first and second degree, and deg ϕ = max (deg ϕ_1, deg ϕ_2) = 2. It is well-known that, regardless of the number of processors, the computation of nonlinear recurrences of the form of Eq. (4) and with deg ϕ>1 can be speeded up only by a constant factor [20]-[21]. This is due to the fact that the data dependency in nonlinear

recurrences and, particularly, those containing division, is stronger than in linear recurrences [22]. Hence, the parallelism in the $O(n)$ algorithms is bounded, that is, their parallelization leads to the $O(n)$ algorithms which are faster than the serial algorithm only by a constant factor. Note that a rather simple model was used for presentation of the nonlinear recurrences of the $O(n)$ algorithms while they are far more complex than those usually studied in literature, e.g., in [21]-[22] (see [8] for a more detailed discussion).

For the conjugate gradient algorithms in [7], [10], the computation of each iteration, as is shown in [15], can be done in a time of $O(\log_2 n)$ with n processors, leading to the $O(n\log_2 n)$ parallel algorithms. This implies that the parallelism in conjugate gradient algorithm is unbounded. Asymptotically, however, the parallel conjugate gradient algorithms are slower than the best serial algorithms, the $O(n)$ algorithms, for the solution of the problem.

The inertia matrix can be computed in $O(\log n)+O(1)$ steps with $O(n^2)$ processors [8], [11], [13]. The implication of this result is that it further reduces the analysis of the time and processors bounds in the forward dynamics problem to that in a more generic problem, the linear system solution. Csanky has shown that the linear system can be solved in $O(\log_2^2 n)$ steps with $O(n^4)$ processors [23] . This implies that the forward dynamics problem belongs to the class of *NC*. Note that, using Cramer's rule, the linear system solution can be computed in $O(\log n)$ steps with $O(n!)$ processors [20]. But such a result has neither theoretical nor practical importance.

However, Csanky's algorithm is unpractical since, besides using too many processors, it is numerically unstable [25]. The best stable algorithms for linear system solution achieve a time of $O(n)$ with $O(n^2)$ processors [24]-[25]. Hence, parallelization of the $O(n^3)$ algorithms results in the stable $O(n)$ parallel algorithms with $O(n^2)$ processors, which indicates an unbounded parallelism.

The above analysis shows that the forward dynamics problem belongs to the class of *NC* and that the best known upper bounds on the time and processors are $O(\log_2^2 n)$ and $O(n^4)$, respectively. Practically, however, the fastest (and stable) parallel algorithm for its computation is of $O(n)$. With respect to these results the main question is, given the fact that both the serial $O(n)$ and $O(n^3)$ algorithms result in the $O(n)$ parallel algorithms, which one is more efficient for parallelization?

Let $\alpha_1 n+\beta_1$ denote the polynomial complexity of the serial $O(n)$ algorithms. There is a limited parallelism in both coarse grain and fine grain (in matrix-vector operation) forms in these algorithms [8]. Exploitation of this parallelism leads to the parallel algorithms with polynomial complexity as $\alpha_2 n+\beta_2$ where, due to the limited parallelism, α_1 is reduced to α_2 only by a small factor. Furthermore, exploitation of both coarse and fine grain parallelism requires additional architectural complexity. For the $O(n^3)$

algorithms, the polynomial complexity of the resulting parallel $O(n)$ algorithm is of the form $\alpha_3 n+\gamma_3\lceil\log_2 n\rceil+\beta_3$ where α_3 is smaller than α_1 by more than two orders-of-magnitude. As a result, while the algorithm is asymptotically faster than the serial $O(n)$ algorithms and their parallel versions by a high constant factor, it is also more efficient for small n. The price to be paid for this efficiency, of course, is an architecture with $O(n^2)$ processors. However, the efficiency of the parallel algorithm and the suitability of the architecture for VLSI and WSI implementation strongly support the choice of $O(n^3)$ algorithms for parallel computing.

III. PARALLEL COMPUTATION OF INERTIA MATRIX

A. Basic Algorithms for computation of inertia matrix

From Eq. (3) the elements of the inertia matrix can be computed as

$$a_{ij} = a_{ji} = \Gamma_j \tag{5}$$

for the condition given by

$$\ddot{Q}_i = 1 \text{ and } \ddot{Q}_{k\neq i} = 0 \qquad \text{For } k = 1, 2, \ldots, n \tag{6}$$

Two physical interpretations can be thought for the above condition, with each interpretation leading to a distinct class of algorithms as

1) The first i-1 links do not have any motion, that is, they are static, and the accelerations and the forces/torques of the last n-i+1 links result from the unit acceleration of link i. This interpretation leads to the first class of algorithms, designated as the class of Newton-Euler Based (NEB) algorithms, in which the diagonal and lower off-diagonal elements of the inertia matrix are computed. In [7] an algorithm of this class is presented, designated as the Original NEB (ONEB) algorithm, which computes the inertia matrix by successive applications of the N-E formulation.

2) The last n-i+1 links can be considered as a single composite rigid system, since they do not have any relative motion, which is accelerating in space, leading to the exertion of forces and moments on the first i-1 static links. This interpretation leads to the second class of algorithms, designated as the class of Composite-Rigid Body (CRB) algorithms, in which the diagonal and upper off-diagonal elements of the inertia matrix are computed. In [7] an algorithm of this class, designated as the Original CRB (OCRB) algorithm, is presented in which the center of mass and the first and the second moment of mass with respect to the center of mass of a set of composite systems are computed.

The comparative study in [7] shows that the OCRB algorithm achieves a significantly greater efficiency over the ONEB algorithm. In [8]-[9], we have developed an algorithm, designated as the Variant of CRB (VCRB) algorithm, which avoids the redundancies in the OCRB algorithm and represents the most efficient algorithm (to date) for computing the inertia matrix. Note that, however, due to the symmetry of the problem, both interpretations and hence both classes of algorithms should lead to the same results and computational

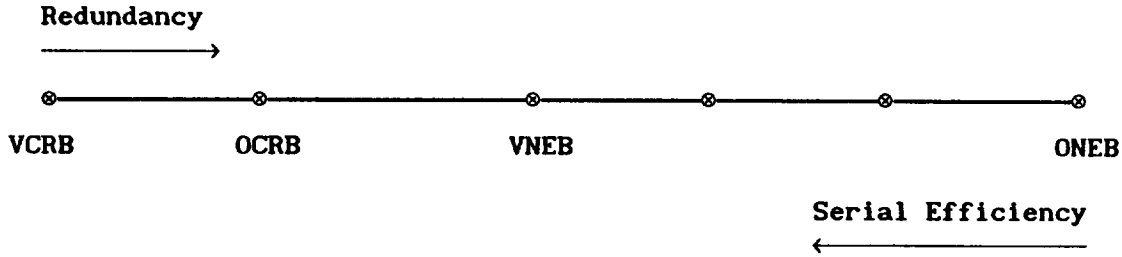


Fig. 1. Comparison of different algorithms for computation of inertia matrix

efficiency. In [8]-[9], we have shown that, by introducing or reducing the redundancy in the computation, the algorithms of the two classes can be transformed to one another and, particularly, to the most efficient one, the VCRB algorithm. Figure 1 shows the relative serial efficiency of and redundancy in different algorithms.

Although the results presented in Fig. 1 answer the question of the serial efficiency of different algorithms, it does not indicate which algorithm provides the most suitable features for parallelization. For serial processing, removing any redundancy increases the computational efficiency. For parallel processing, however, depending on its impact on the data dependency in the computation, this may increase or decrease the efficiency. The fact that arbitrary algorithms can be developed by introducing or removing different types of redundancy in the computation represents an additional degree-of-freedom that can be exploited to derive an algorithm which, though perhaps not efficient for serial computing, is the most suitable for parallelization. In [8], [12], we have shown that the NEB algorithms are more suitable for parallel computing than the CRB algorithms. In fact, they not only achieve a better computational complexity (in the parallel sense) but also require a less complex communication and synchronization mechanism. This better efficiency for parallelization mainly results from the fact that the evaluation of the columns of the inertia matrix by the NEB algorithms is order independent and hence can be done in parallel.

B. A Variant of Newton-Euler Based (VNEB) Algorithm

Four different types of redundancy can be recognized in the ONEB algorithm, which can be eliminated, respectively, by [8], [9], [13]:

1) Choosing a more suitable coordinate frame for projection of the equations.
2) Optimizing the N-E formulation for the condition given by Eq. (6).
3) Using a more efficient variant of the optimized N-E formulation.
4) Introducing a two-dimensional recursion in the computation.

Note that the first redundancy resides in the *extrinsic* equations and results from the choice of coordinate frame for projection of the *intrinsic* equations while the second, the third, and the fourth redundancies reside in the intrinsic equations and are inherent in the formulation. As stated before, by removing all redundancies in the intrinsic equations, the ONEB algorithm can be transformed to the VCRB algorithm. However, removing any type of redundancy in the NEB algorithms, as far as the order independence property

is preserved, will also increase the efficiency of their parallel versions. In this regard, only the removal of the fourth redundancy, which leads to the introduction of a two-dimensional recursion in the computation, results in the loss of the order independence property of the algorithm. In the following, a Variant of the NEB algorithm, designated as the VNEB algorithm, is presented which is developed by removing the first three redundancies.

The derivation of the Variant of N-E Based (VNEB) algorithm is fully discussed in [8], [9], [12], [13]. Here, for the sake of completeness, a brief description of the algorithm is given. The algorithm is presented by the intrinsic equations. In this paper, according to the Gibbs notation, vectors are underlined once and tensors (tensors of order 2) twice. Also, in order to simplify and, particularly, unify the derivation of the serial and parallel algorithms, a set of notations, given in Table I and Fig. 3, are used. The VNEB algorithm is then written as

For $i = 1, 2, \ldots, n$

For $j = i, i+1, \ldots, n$

$$\dot{\underline{\omega}}(j,i) = \underline{Z}(i) \tag{7}$$

$$\dot{\underline{V}}(j,i) = \dot{\underline{V}}(j-1,i) + \dot{\underline{\omega}}(j,i)\times\underline{P}(j,j-1) = \dot{\underline{\omega}}(j,i)\times\underline{P}(j,i) \tag{8}$$

$$\underline{F}(j+1,j,i) = M(j)\dot{\underline{V}}(j,i) + \dot{\underline{\omega}}(j,i)\times\underline{H}(j) \tag{9}$$

$$\underline{N}(j+1,j,i) = \underline{K}(j)\dot{\underline{\omega}}(j,i) \tag{10}$$

For $j = n, n-1, \ldots, i$

$$\underline{F}(n+1,j,i) = \underline{F}(j+1,j,i) + \underline{F}(n+1,j+1,i) \tag{11}$$

$$\underline{N}(n+1,j,i) = \underline{N}(j+1,j,i) + \underline{N}(n+1,j+1,i) + \underline{P}(j+1,j)\times\underline{F}(n+1,j+1,i) \tag{12}$$

with $\underline{F}(n+1,n+1,i) = \underline{N}(n+1,n+1,i) = 0$

$$a_{ji} = \underline{Z}(j).\underline{N}(n+1,j+1,i) \tag{13}$$

C. Parallel Algorithm for Computation of Inertia Matrix

The serial computational complexity of evaluating the inertia matrix is of $O(n^2)$. No serial algorithm can achieve a better asymptotic complexity since, given n inputs (joint positions), the evaluation of the $O(n^2)$ outputs, the elements of the inertia matrix, requires $O(n^2)$ distinct steps in the computation. Based on the VCRB algorithm, we have already shown that the inertia matrix can be computed in $O(\log_2 n)+O(1)$ steps with $O(n^2)$ processors [8], [11]. It is interesting to note that not only the same bounds on time and processors can be much more easily derived by parallelization of the VNEB algorithm but also the resulting parallel algorithm, compared to the parallel VCRB algorithm, reduces the coefficients on the polynomial complexity.

For the implementation of the parallel algorithm achieving the time lower bound, we consider a two-dimensional array of $n(n+1)/2$ processor-memory modules represented as PR_{ij}, for $i = 1, 2, \ldots, n$ and $j = i, i+1, \ldots, n$

(Fig. 3 shows the array for n = 6). For the parallel algorithm, the equations are projected onto the EE coordinate frame, coordinate frame n+1. An n DOF all revolute joints manipulator is considered for which the joint variables are the joint angles, that is, $Q_j = \theta_j$. It is assumed that the joint variables θ_j (or $S\theta_j$ and $C\theta_j$) and constant parameters $S\alpha_j$, $C\alpha_j$, ${}^{j+1}P(j+1,j)$, ${}^{j+1}H(j)$, ${}^{j+1}K(j)$, and $M(j)$ reside in the memory of all processors of Row j. The analysis of the mapping the algorithm onto the architecture of Fig. 3 is fully presented in [12]. Here, due to the lack of space, we only give the a brief description of the algorithm in terms of its computational steps and cost.

For the parallel algorithm, the ith column of the inertia matrix is computed by the processors of the ith column of the processor array. The fact that $\dot{\omega}(j,i) = Z(i)$ for $j = i, i+1, \ldots, n$, implies that global communication of $Z(i)$ among the processors of the ith column is required. This requirement can be avoided by introducing two recurrences as

$$\dot{\omega}(j,i) = \dot{\omega}(j-1,i) = Z(i) \tag{14}$$

$$P(j,i) = P(j-1,i) + P(j,j-1) \tag{15}$$

Equation (14) does not need any computation while, for the parallel algorithm, the computation of Eq. (15) is required. By computing Eqs. (14)-(15) as a set of coupled recurrences, the terms $\omega(j-1,i)$ can be considered as the data associated with Eq. (15). Using such a scheme increases the communication complexity of parallel evaluation of Eq. (15) but will result in the global distribution of $Z(i)$ among the processors of the ith column. The computation of the parallel algorithm is then performed as follows.

Step 1:
1) Parallel compute $R(j+1,j)$ by all processors of the jth row.

For $j = 1, 2, \ldots, n$

  For $i = 1, 2, \ldots, j$

    PR_{ji} : $R(j+1,j)$ (16)

2) Parallel compute $R(n+1,j)$ by processors of the ith column.

For $i = 1, 2, \ldots, n$
  For $j = i, i+1, \ldots, n$
    For $\eta = 1$ step 1 until $\lceil \log_2(n+1-j) \rceil$, Do (17)

$$R(j+2^{\eta},j) = R(n+1,j) \qquad j+2^{\eta} > j+2^{\eta-1} \geq n+1$$

$$R(j+2^{\eta},j) = R(n+1,j) = R(n+1,j+2^{\eta-1})R(j+2^{\eta-1},j) \qquad j+2^{\eta} \geq n+1 > j+2^{\eta-1}$$

$$R(j+2^{\eta},j) = R(j+2^{\eta},j+2^{\eta-1})R(j+2^{\eta-1},j) \qquad n+1 > j+2^{\eta} > j+2^{\eta-1}$$

    End_Do

3) Rotate $R(n+1,j)$ by processors of Row j to the processors of Row j-1.
For $j = 1, 2, \ldots, n$
  For $i = 1, 2, \ldots, j$
    PR_{ji}: $R(n+1,j+1)$ (18)

with $R(n+1,n+1) = U$ (Unit Matrix)

Note that, as the result of the above data transfer, both the terms $R(n+1,j)$ and $R(n+1,j+1)$ reside in the memory of all the processors of the jth row.

4) Parallel compute ${}^{n+1}Z(j)$, ${}^{n+1}P(j+1,j)$, ${}^{n+1}H(j)$, and ${}^{n+1}K(j)$ by all the processors of the jth row.

For $j = 1, 2, \ldots, n$

For $i = 1, 2, \ldots, j$

a) PR_{ji} : ${}^{n+1}Z(j) = R(n+1,j)\,{}^{j}Z(j)$ (19)

with ${}^{j}Z(j) = [0\ 0\ 1]^t$

b) PR_{ji}: ${}^{n+1}P(j+1,j) = R(n+1,j+1)\,{}^{j+1}P(j+1,j)$ (20)

with ${}^{j+1}P(j+1,j) = [a_i\ \ d_i S\alpha_i\ \ d_i C\alpha_i]^t$

c) PR_{ji}: ${}^{n+1}H(j) = R(n+1,j+1)\,{}^{j+1}H(j)$ (21)

d) PR_{ji}: ${}^{n+1}K(j) = R(n+1,j+1)\,{}^{j+1}K(j)R(j+1,n+1)$ (22)

Note that for the processors of the nth row Eqs. (21)-(22) do not need any computation since the terms ${}^{n+1}H(n)$ and ${}^{n+1}K(n)$ are given constant parameters. As the result of the computation of Step 1, all the vectors and the tensors are projected onto the coordinate frame n+1. In the following, the absence of superscripts denotes that the computations are performed in this frame.

Step 2:

1) Parallel compute $P(j,i)$ and $\dot{\omega}(j,i)$ by processors of the ith column.

For $i = 1, 2, \ldots, n$

For $j = i, i+1, \ldots, n$

For $\eta = 1$ step 1 until $\lceil \log_2(n+1-j) \rceil$, Do

$$\dot{\omega}(j+2^{\eta},i) = \dot{\omega}(j-2^{\eta},i) = Z(i) \tag{23}$$

$$P(j+2^{\eta},j) = P(i,j) \qquad j+2^{\eta} > j+2^{\eta-1} \geq i$$

$$P(j+2^{\eta},j) = P(i,j) = P(i,j+2^{\eta-1}) + P(j+2^{\eta-1},j) \qquad j+2^{\eta} \geq i > j+2^{\eta-1} \tag{24}$$

$$P(j+2^{\eta},j) = P(j+2^{\eta},j+2^{\eta-1}) + P(j+2^{\eta-1},j) \qquad i > j+2^{\eta} > j+2^{\eta-1}$$

End_Do

2) Parallel compute $\dot{V}(j,i)$, $F(j+1,j,i)$, $N(j+1,j,i)$ by processors of the ith column.

For $i = 1, 2, \ldots, n$

For $j = i, i+1, \ldots n$

PR_{ji}: $\dot{V}(j,i) = \dot{\omega}(j,i) \times P(j,i)$ (25)

$$PR_{ji}:\ F(j+1,j,i) = \dot{\omega}(j,i)xH(j) + M(j)\dot{V}(j,i) \tag{26}$$

$$PR_{ji}:\ N(j+1,j,i) = K(j)\dot{\omega}(j,i) + H(j)x\dot{V}(j,i) \tag{27}$$

Step 3:

1) Parallel compute F(n+1, j, i) and N(n+1, j, i) by processors of the ith column.

For i = 1, 2, ..., n

For j = i, i+1, ..., n

For η = 1 step 1 until $\lceil \log_2(n+1-j) \rceil$, Do (28)

$F(j+2^{\eta},j,i) = F(n+1,j,i)$ $\quad j+2^{\eta} > j+2^{\eta-1} \geq n+1$

$F(j+2^{\eta},j,i) = F(n+1,j,i) = F(j+2^{\eta},j+2^{\eta-1},i)+F(j+2^{\eta-1},j,i)$ $\quad j+2^{\eta} > n+1 > j+2^{\eta-1}$

$F(j+2^{\eta},j,i) = F(j+2^{\eta},j+2^{\eta-1},i)+F(j+2^{\eta-1},j,i)$ $\quad n+1 > j+2^{\eta} > j+2^{\eta-1}$

End_Do

For η = 1 step 1 until $\lceil \log_2(n+1-j) \rceil$, Do (29)

$N(j+2^{\eta},j,i) = N(n+1,j,i)$ $\quad j+2^{\eta} > j+2^{\eta-1} \geq n+1$

$N(j+2^{\eta},j,i) = N(n+1,j,i) = N(n+1,j+2^{\eta-1},i)+N(j+2^{\eta-1},j,i)+$

$P(j+2^{\eta-1},j)xF(n+1,j+2^{\eta-1},i)$ $\quad j+2^{\eta} \geq n+1 > j+2^{\eta-1}$

$N(j+2^{\eta},j,i) = N(j+2^{\eta},j+2^{\eta-1},i)+N(j+2^{\eta-1},j,i)+$

$P(j+2^{\eta-1},j)xF(j+2^{\eta},j+2^{\eta-1},i)$ $\quad n+1 > j+2^{\eta} > j+2^{\eta-1}$

End_Do

2) Parallel compute a_{ji} by PR_{ji}.

For i = 1, 2, ..., n

For j = i, i+1, ...n

$$PR_{ji}:\ a_{ji} = Z(j).N(n+1,j,i) \tag{30}$$

As stated before, the time lower bound in, as well as the computational cost of, parallel evaluation of the inertia matrix, using the VNEB algorithm, is determined by that of the parallel evaluation of the first column of the inertia matrix. In other words, the computational cost of the n recurrences in Eqs. (17), (23), (24), (28), and (29) is determined by that of the largest ones, i.e., for i = 1, which are of size n. Furthermore, the computational cost of all the $O(n^2)$ terms in Eqs. (16), (19)-(22), (25)-(27), and (30) is determined by the cost of one term since for each column n terms are computed in parallel and the computation for n columns, as will be discussed later, is overlapped. Let *m* and *a* denote the cost of multiplication and addition, respectively. The computational cost of the parallel algorithm is then evaluated as follows:

Step 1: The cost of Eq. (16) is $4m$; The cost of Eq. (17) is $(27m+18a)\lceil\log_2 n\rceil$; Eq. (18) represents a simple data rotation; Eq. (19) does not need any computation; The cost of Eqs. (20), (21), and (22) is $(9m+6a)$, $(9m+6a)$, and $(54m+36a)$, respectively. The cost of this step is $(27m+18a)\lceil\log_2 n\rceil+(76m+48a)$.

Step 2: Eq. (23) does not need any computation; The cost of Eq. (24) is $(3a)\lceil\log_2 n\rceil$; The cost of Eqs. (25), (26), and (26) is $(6m+3a)$, $(9m+6a)$, and $(15m+12a)$, respectively. The cost of this step is $(3a)\lceil\log_2 n\rceil+(30m+21a)$.

Step 3: The cost of Eqs. (28) and (29) is $(3a)\lceil\log_2 n\rceil$ and $(6m+9a)\lceil\log_2 n\rceil$, respectively; The cost of Eq. (30) is $(3m+2a)$. The cost of this step is $(6m+12a)\lceil\log_2 n\rceil+(3m+2a)$.

Adding the cost of Steps 1-3, the computation cost of the algorithm is obtained as $(33m+33a)\lceil\log_2 n\rceil+(109m+71a)$. As stated before, mapping the developed parallel algorithm onto the processor array is fully presented in [12] where it is shown that, even using a simple nearest neighbor interconnection structure, a communication complexity of $O(n)$ can be achieved. Also, the mechanisms for global and local synchronization for the processor array are presented. Figure 4 shows the resulting distribution of the elements of the inertia matrix among the processors.

IV. PARALLEL COMPUTATION OF N-E FORMULATION AND SOLUTION OF LINEAR SYSTEM

As stated before, the bias vector can be computed by evaluating the N-E formulation while setting the vector of joint accelerations to zero. We use the parallel algorithm presented in [15] for computing the bias vector. This computation is performed by the processors of the first column with the equations being projected onto the frame n+1. Therefore, the results of the computation of the first step except Eqs. (21)-(22) can be used while, similar to the terms ${}^{n+1}K(j)$ in Eq. (22), the terms ${}^{n+1}J(j)$, for $j = 1, 2, \ldots, n$, are also needed to be computed by the processors of the first column. The cost of evaluation of the vector Γ is then obtained as $15a\lceil\log_2 n\rceil+(141m+101a)$. With the nearest neighbor interconnection among the processors of the first column a communication complexity of $O(n)$ is achieved. In order to achieve the proper sequencing of the computation of the inertia matrix, the bias vector, and the linear system solution, a data-driven mechanism can be employed. That is, the processors of all columns, except the processors of the first column, by completion of the computation of their corresponding column of the inertia matrix enter the wait state while the processors of the first column start the computation of the bias vector and the vector Γ. The activity of the processors of column 2 through column n in linear system solution is triggered by receiving the corresponding data and by the completion of the computation of the vector Γ.

Figure 4 shows the organization of the data resulting from the computation of the inertia matrix and the vector Γ. Given this data organization, we have developed synchronous data-flow parallel algorithms for the full solution of Eq. (3), that is, for decomposition of the inertia matrix, using the square-root-free variant of the Cholesky factorization, and the solution of the

resulting lower and upper triangular linear systems, which are presented in detail in [12]. The computation cost of the solution of Eq. (3) is obtained as $(n-1)(3m+2a)+(1m+1a)$, where the cost of division and multiplication is taken to be the same. A communication complexity of $O(n)$ is also achieved.

Adding the cost of evaluation of the inertia matrix, the bias vector and the vector Γ, and the linear system solution, the computational cost of the forward dynamics problem is obtained as $(3m+2a)n+(33m+48a)\lceil \log_2 n \rceil+(238m+171a)$. The computational cost of the best serial $O(n)$ algorithm, the Articulated-Body algorithm [3]-[4], is evaluated as $(380m+302a)n-(198m+173a)$ [15]. If the time of multiplication and addition is taken to be the same, then, for n = 6, the developed parallel algorithms achieve a speedup of 6 compared to the best serial $O(n)$ algorithm. Note, however, that as n increases, e.g., for redundant manipulators, the speedup also significantly increases. To see this, let us write the computational cost of the serial $O(n)$ algorithm and the parallel algorithms as $682n+O(1)$ and $5n+O(\log_2 n)+O(1)$, with the time of multiplication and addition taken to be the same. It can be seen that, asymptotically, the parallel algorithms achieve a speedup of more than two orders-of-magnitude. For small n, the computational cost of the parallel algorithms is dominated by the $\lceil \log_2 n \rceil$-dependent and constant terms on the polynomial complexity. Hence, for small n, the computational complexity of the developed parallel algorithms can be practically considered as $O(\log_2 n)+O(1)$.

V. CONCLUSION

We developed a set of parallel algorithms for computing the forward dynamics problem. These algorithms exploit a high degree of parallelism in the problem and achieve a significant speedup in the computation. Furthermore, they can be efficiently implemented on a two-dimensional array of processors with a nearest neighbor interconnection. This architecture is particularly suitable for practical implementation using VLSI and WSI technologies. Due to their simple architectural requirements, these algorithms, with some modifications, can also be efficiently implemented on rather more general-purpose architectures, e.g., a two-dimensional array of Transputers.

A key factor in our approach to the parallel computation of the forward dynamics is the minimization of the resulting overhead. The overall communication complexity is of $O(n)$. The overhead is further minimized since there is no need for any data alignment between the computation of different algorithms and the intermediate data resulting from the different algorithms are generated and consumed within the array. Also, the final result of the computation, that is, the vector of joint accelerations, is computed by the processors of the first column. Therefore, they can be output using the same channels for inputting the data to the array (Fig. 5). This is particularly critical for VLSI and WSI implementation since, by using only n bidirectional Input/Output channels, the number of required pins is kept small.

ACKNOWLEDGEMENT
The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration (NASA).

References

[1] M.H. Milman and G. Rodriguez,"Cooperative Dual Arm Manipulator Issues and Task Approach," *Jet Propulsion Lab., Eng. Memorandum 347*, Nov. 1987.

[2] J.Y.S. Luh, M.W. Walker, and R.P. Paul,"On-line Computation Scheme for Mechanical Manipulator," *Trans. ASME J. Dyn. Syst., Meas., and Control*, Vol. 102, pp. 69-76, June 1980.

[3] R. Featherstone,"The Calculation of Robot Dynamics Using Articulated-Body Inertia," *Int. J. Robotics Research*, Vol.2(2), 1983.

[4] R. Featherstone, *Robot Dynamics Algorithms*. Ph.D. Dissertation, Univ. of Edinburgh, 1984.

[5] G. Rodriguez,"Kalman Filtering, Smooting and Recursive Robot Arm Forward and Inverse Dynamics," *IEEE J. Robotics and Automation*, Vol. RA-5, Dec. 1987. Also in Jet Propulsion Lab. Publication 86-48, Dec. 1986.

[6] G. Rodriguez and K. Kreutz,"Recursive Mass Matrix Factorization and Inversion: An Operator Approach to Open- and Closed-Chain Multibody Dynamics," *Jet Propulsion Lab. Publication* 88-11, May 1988.

[7] M.W. Walker and D.E. Orin,"Efficient Dynamic Computer Simulation of Robotics Mechanisms," *Trans. ASME J. Dyn. Syst., Meas., and Contr.*, vol. 104, pp. 205-211, 1982.

[8] A. Fijany, *Parallel Algorithms and Architectures in Robotics*. Ph.D. Dissertation, Univ. of Paris XI (Orsay), Sept. 1988.

[9] A. Fijany and A.K. Bejczy,"An Efficient Algorithm for Computation of the Manipulator Inertia Matrix," To appear in J. of Robotic Systems, Feb. 1990.

[10] A. Fijany and R.E. Scheid,"Efficient Conjugate Gradient Algorithms for Computation of the Manipulator Forward Dynamics," *Proc. of NASA Conf. on Space Telerobotics*, Pasadena, CA, Jan. 1989.

[11] A. Fijany and A.K. Bejczy,"A Class of Parallel Algorithms for Computation of the Manipulator Inertia Matrix," *IEEE Trans. Robotics and Automation*, Vol. RA-5, No. 5, pp. 600-615, Oct. 1989.

[12] A. Fijany and A.K. Bejczy," Parallel Algorithms and Architecture for Computation of the Manipulator Forward Dynamics," Submitted to *IEEE Trans. Robotics and Automation*.

[13] A. Fijany,"A New Class of Parallel and Pipeline Algorithms for Computation of the Manipulator Inertia Matrix," In preparation.

[14] A. Fijany and J.G. Pontnau,"Parallel Computation of the Jacobian for Robot Manipulators," Proc. IASTED Int. Conf. on Robotics & Automation, Santa Barbara, May 1987.

[15] C.S.G. Lee and P.R. Chang,"Efficient Parallel Algorithms for Robot Forward Dynamics Computation," *IEEE Trans. Syst. Man Cybern.*, Vol. SMC-18, no. 2, pp. 238-251, March/April 1988.

[16] C.S.G. Lee and P.R. Chang,"Efficient Parallel Algorithm for Robot Inverse Dynamics Computations," *IEEE Trans. Syst. Man Cybern.*, Vol. SMC-16(4), pp. 532-542, July/Aug. 1986.

[17] L.H. Lathrop,"Parallelism in Manipulator Dynamics," *Int. J. Robotics Res.*, Vol. 4(2), Summer 1985.

[18] S.A. Cook,"An Overview of Computational Complexity," *Com. ACM*, Vol. 26, June 1983.

[19] J.S. Vitter and R.A. Simons,"New Classes of Parallel Complexity: A Study of Unification and other Complete Problems for *P*," IEEE Trans. Computer, Vol. C-35(5), May 1986.

[20] H.T. Kung,"New Algorithms and Lower Bounds for the Parallel Evaluation of

Certain Rational Expressions and Recurrences," *J. of ACM*, Vol. 23, No. 2, pp. 252-261, April 1976.
[21] J. Miklosko and V.E. Kotov (Eds.), *Algorithms, Software and Hardware of Parallel Computers*. New York: Springer-Verlag, 1984.
[22] L. Hyafil and H.T. Kung,"The Complexity of Parallel Evaluation of Linear Recurrences," *J. ACM*, Vol. 24, No. 3, pp. 513-521, July 1977.
[23] L. Csanky,"Fast Parallel Matrix Inversion Algorithms," *SIAM J. of Computing*, Vol. 5, No. 4, pp. 618-623, Dec. 1976.
[24] A.H. Sameh and D.J. Kuck,"On Stable Parallel Linear System Solvers," *J. of ACM*, Vol. 25, No. 1, pp. 81-91, Jan. 1978.
[25] A. Bojanczyk, R.P. Brent, and H.T. Kung,"Numerically Stable Solution of Dense Systems of Linear Equations Using Mesh-Connected Processors," *SIAM J. of Stat. Comput.*, Vol. 5, No. 1, March 1984.

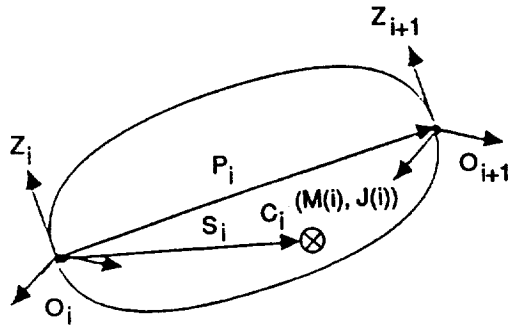


Fig. 2. Link, Frames, and Kinematic and Dynamic Parameters.

a_i, d_i, & α_i	Length, Distance, and Twist Angle of link i, respectively.
Q_i, $\dot{Q}_i$, & $\ddot{Q}_i$	Position, Velocity, and Acceleration of joint i, respectively.
M(i)	Mass of link i.
J(i)	Second moment of mass of link i about its center of mass (C_i).
H(i)	First moment of mass of link i about point O_i.
K(i)	Second moment of mass of link i about point O_i.
Z(i)	Axis of joint i
P(i,j)	Position vector from point O_j to point O_i.
R(i,j)	A 3x3 matrix representing the orientation of coordinate frame j with respect to coordinate frame i.
$\dot{\omega}$(i,j)	Angular acceleration of link i resulting from the unit acceleration of joint j.
$\dot{V}$(i,j)	Linear acceleration of link i (point O_i) resulting from the unit acceleration of joint j.
F(k+1,i,j)	Force exerted on point O_i due to the acceleration of links i through k, i.e., the links contained between points O_i and O_{k+1}, resulting from the unit acceleration of joint j.
N(k+1,i,j)	Moment exerted on point O_i due to the acceleration of link i through k, resulting from the unit acceleration of joint j.

Table I. Notation used in derivation of serial and parallel algorithms

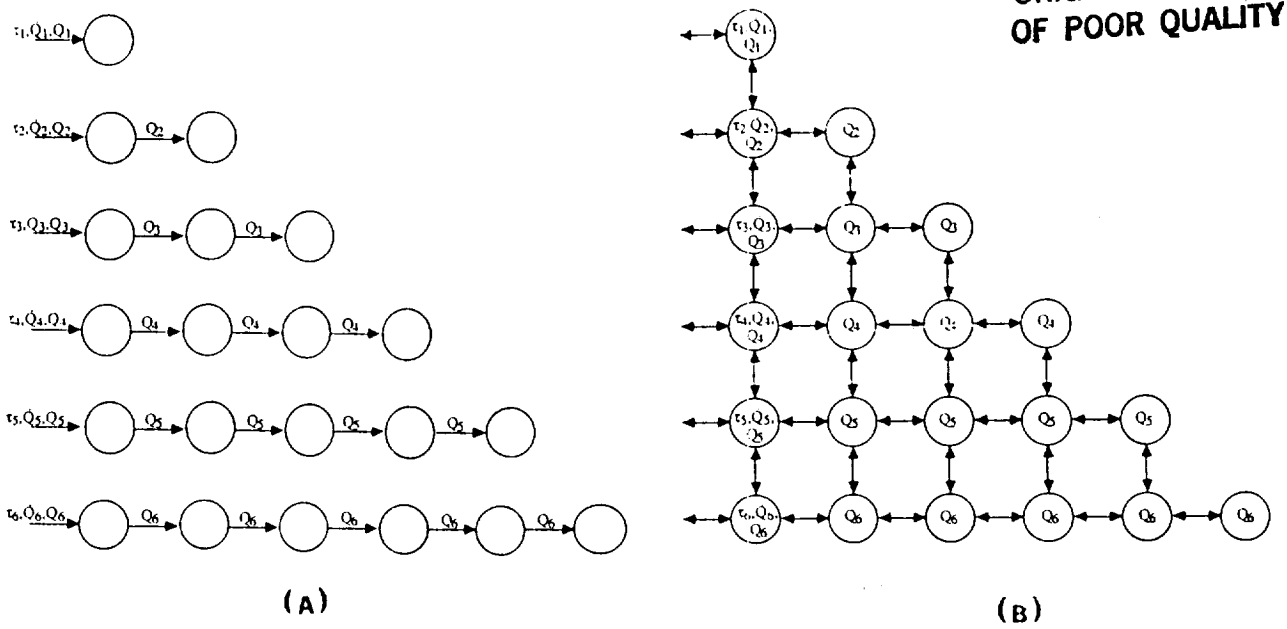


FIGURE 3. A TWO-DIMENSIONAL PROCESSOR ARRAY (A) DATA INPUT TO PROCESSOR ARRAY, (B) DISTRIBUTION OF INPUT DATA AMONG PROCESSORS.

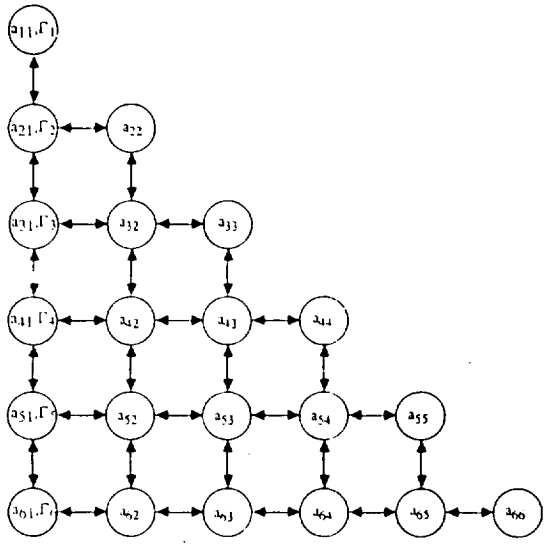


FIGURE 4. ORGANIZATION OF DATA RESULTING FROM COMPUTATION OF THE INERTIA MATRIX AND THE BIAS VECTOR.

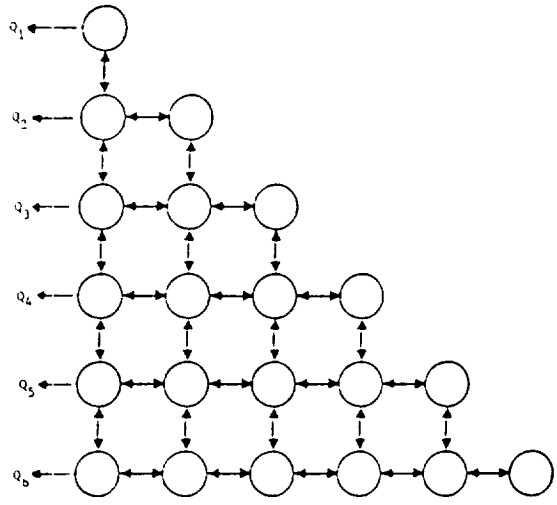


FIGURE 5. DATA OUTPUT FROM PROCESSOR ARRAY.

COMPUTATIONAL DYNAMICS FOR ROBOTICS SYSTEMS USING A NON-STRICT COMPUTATIONAL APPROACH

David E. Orin and Ho-Cheung Wong
Dept. of Electrical Engineering
The Ohio State University
Columbus, OH 43210

P. Sadayappan
Dept. of Computer & Information Science
The Ohio State University
Columbus, OH 43210

Abstract

This paper proposes a Non-Strict computational approach for real-time robotics control computations. In contrast to the traditional approach to scheduling such computations, based strictly on task dependence relations, the proposed approach relaxes precedence constraints and scheduling is guided instead by the relative sensitivity of the outputs with respect to the various paths in the task graph. An example of the computation of the Inverse Dynamics of a simple inverted pendulum is used to demonstrate the reduction in effective computational latency through use of the Non-Strict approach. A speedup of 5 has been obtained when the processes of the task graph are scheduled to reduce the latency along the crucial path of the computation. While error is introduced by the relaxation of precedence constraints, the Non-Strict approach has a smaller error than the conventional Strict approach for a wide range of input conditions.

I. Introduction

Complex robot dynamics computations have typically been represented using directed task precedence graphs, in order to facilitate the exploitation of parallelism in their execution [1,2,3]. The nodes of such a task graph represent computational modules, with the directed edges imposing a strict partial order on their execution sequence. While such a computational approach is faithfully accurate to the underlying physical model of the robot system if executed instantaneously, in practice the computational latency can be significant even with state-of-the-art computers. The use of pipelining also does not reduce this latency, even though it increases computational throughput.

The Non-Strict computational approach is motivated by a need to reduce the "effective latency" from input to output for complex robotics computations. The terms Strict and Non-Strict are derived from the manner in which precedence is treated in scheduling the tasks of a task graph such as the one shown in Fig. 1. The approach proposed here has the

same motivation as the "Imprecise Results" approach recently proposed in [4], but uses a different model. The key concept behind our Non-Strict approach is that of relaxing the strict precedence constraints imposed by a conventional task-graph model, to facilitate trading off accuracy for timeliness in real-time computation over sampled, continuously varying inputs. Rather than sample all inputs simultaneously and then schedule tasks in strict adherence to the precedence constraints dictated by the edges of the task graph, tasks are scheduled so as to minimize delay between input and output along paths in the task graph that most strongly affect the output, perhaps using some previously computed values along less crucial paths.

Consider the task graph shown in Fig. 1. The circles represent computational tasks to be performed and the arcs represent data flow and thus precedence. According to the conventional Strict model of computation, the input, $\ddot{q}$, should first be sampled. Then it should proceed through the sequence of 1-2-3-4-5, completing the computation by furnishing the output, τ. The latency between input and output is just the total computation time; it represents a lag in the real-time control implementation and tends to destabilize the system.

If the output of process 5 is more strongly affected by the output of process 4 and further, if process 4 is strongly affected by the output of process 1 and only weakly dependent on process 3, then a better schedule, which does not strictly adhere to the natural precedence relationships, may be developed. In particular, it may be best to sample the input, compute process 1, resample the input, compute process 4 using the last computed value of process 3, and compute process 5 to generate the output. After process 5 is complete, then processes 2 and 3 may be scheduled which gives the following ordering on the computations 1-4-5-2-3 which does not follow strict precedence. However, if the 1-4-5 path from input to output is the most crucial to the computation, then the effective latency has been reduced. This is especially true if the computation times of processes 2 and 3 are long as compared to the others.

From the above example, several important characteristics of the Non-Strict computational approach may be noted:

1. It is appropriate to the case of real-time control in which the same computation is to be repeated over successive time samples of the inputs.

2. The main objective is to relax the precedence so as to reduce the effective latency from input to output. Rather than sampling the inputs once for a computation so that the results are strictly correct, albeit delayed, before each process which requires an input, a fresh sample of the input is taken so that the effects of delay are minimized. Also, computation generally proceeds along the crucial paths using the freshest (but past) values generated by the other less crucial processes. While this is often not according to precedence, again it can have the effect of reducing the latency.

3. While the Non-Strict approach does introduce error into the computation by violating task-graph precedence constraints, and can therefore only be an approximation of the

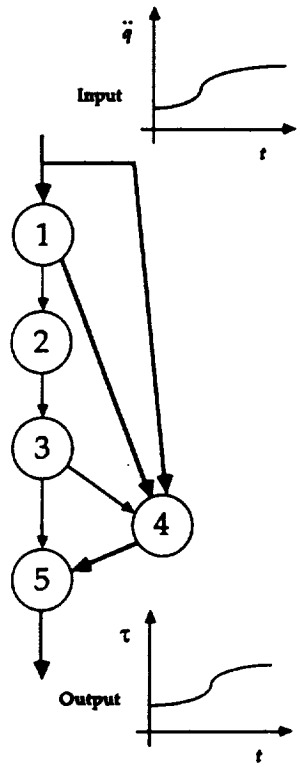


Figure 1: Example Task Graph to Illustrate Non-Strict Computation.

ideal output, the maximum instantaneous error and average error are often much less when compared to that resulting from the use of a Strict approach (which produces an output with the same shape as the ideal, but delayed in time).

4. By relaxing precedence constraints, potential parallelism may be increased. That is, the task precedence relationships constrain the flow of computation to certain sequences/orderings. With these relaxed, a greater amount of parallel scheduling of processes may proceed.

In order to further develop the basic concepts, an example of the dynamics of a simple inverted pendulum will be given. A description of the system, including its dynamic equation of motion and task graph for its evaluation, will be given in the next section. Following that, results using the Non-Strict computational approach will be given and will be compared with the Strict approach. Finally, the paper ends with a summary of the results, and conclusions on which to base further investigations.

II. Simple Inverted Pendulum Example

The computation of the dynamics of the simple inverted pendulum system shown in Fig. 2 is to be considered. The system consists of a single link which is connected through a one-degree-of-freedom revolute joint to a fixed base. Gravity acts in the vertical direction, and an actuator is mounted at the joint to power the pendulum movement. While the model is quite simple, it has been used in the past to study the dynamics of biped walking in the single support phase of locomotion [5].

The basic computational task is to solve the Inverse Dynamics problem [6] in real time. The dynamic equation of motion for the pendulum may be written as follows:

$$\tau = I\ddot{q} + B\dot{q} - mgl \sin(q) \tag{1}$$

where

τ = Actuator torque,
$q, \dot{q}, \ddot{q}$ = Joint position (as referenced to the vertical), velocity, and acceleration
I = Moment of inertia of the link about the joint axis,
B = Joint actuator damping coefficient,
m = Mass of the link, and
l = Position of the center of gravity of the link from the joint axis.

Note that values for the system parameters are also given in Fig. 2.

For Inverse Dynamics, the joint position, velocity, and acceleration are given and the joint torque is to be computed. Inverse Dynamics is used in advanced control schemes for robotic systems to determine the torque required for a desired motion trajectory and must be computed at real-time rates of up to a few hundred hertz [7].

A task graph for the Inverse Dynamics computation is shown in Fig. 3. There are 6 processes with the top number in the circle giving the process number. The operation performed is given in the middle section of the circle while the computation time is given in the bottom part and is normalized to units of a basic computation time, Δ. Note that it is assumed that adds and multiplies take a single unit of time and that the sine computation takes ten times as long. For the purposes of this example, it is assumed that the input position, q, is a simple sinusoid:

$$q(t) = \sin(\omega t) . \tag{2}$$

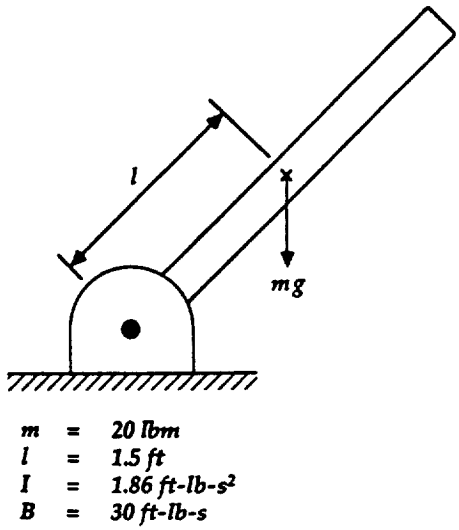


Figure 2: Simple Inverted Pendulum System.

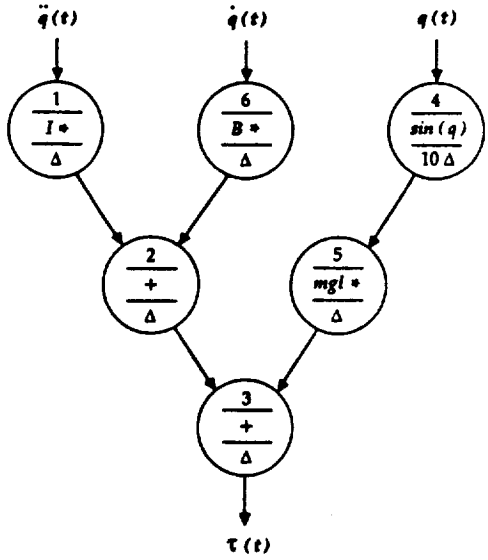


Figure 3: Task Graph for Inverse Dynamics Computation.

where ω is the frequency.

III. Computational Results

In this section, using the Inverse Dynamics of an inverted pendulum as an example, a number of results will be given which illustrate the Non-Strict computational approach and compare it with the Strict approach. The output of the computation under the Strict and Non-Strict approach are contrasted for a simple sinusoidal input to the system. If the computational delays in evaluating the component operations in the task graph were zero, then both the Strict and Non-Strict approaches would yield identical results. With the Strict approach, the generated output is identical in shape to the ideal output, but shifted in time by the sum of the computational delays of the tasks in the task graph. With the Non-Strict approach, the output (in general) approximates the shape of the ideal output but is not identical in form. However, the overall error (including the effects of time delay) can often be significantly less than the error with the Strict computation.

In this section, the two approaches are compared using two measures – 1) *mean square error*, and 2) *effective latency*. The *mean square error* over a period is defined as:

$$mean\ square\ error\ =\ \frac{1}{T}\int_0^T [\tau_i(t) - \tau(t)]^2\ dt \tag{3}$$

where T is the period of the input and $\tau_i(t)$ is the ideal output. The *effective latency* Δt_{eff} is defined in the following way:

$$\Delta t_{eff}\ =\ \Delta t \quad :\quad Min\ \frac{1}{T}\int_0^T [\tau_i(t - \Delta t) - \tau(t)]^2\ dt. \tag{4}$$

Thus a best fit is attempted between a delayed version of the ideal output and the Non-Strict output, and the shift in the delayed version of the ideal output is interpreted as an effective latency of the computation. As long as the mean square error between the output and the shifted ideal output is sufficiently small, the error in output may be related to that arising from a time delay of the ideal output. Thus, the outputs of the Strict and Non-Strict approaches may be compared in terms of effective computational latencies, where the scheme with the lower effective latency is preferable.

In the following, speedup through reduction in the effective latency of the computation will be shown. The relationship of the error in both the Strict and Non-Strict cases will be given as a function of the input frequency. The objective is to determine a range of frequencies for which the Non-Strict approach will give speedup while maintaining reasonable limits on the error. Finally, the relationship between the effective latency and the input frequency will be investigated and will provide motivation for adaptive scheduling strategies.

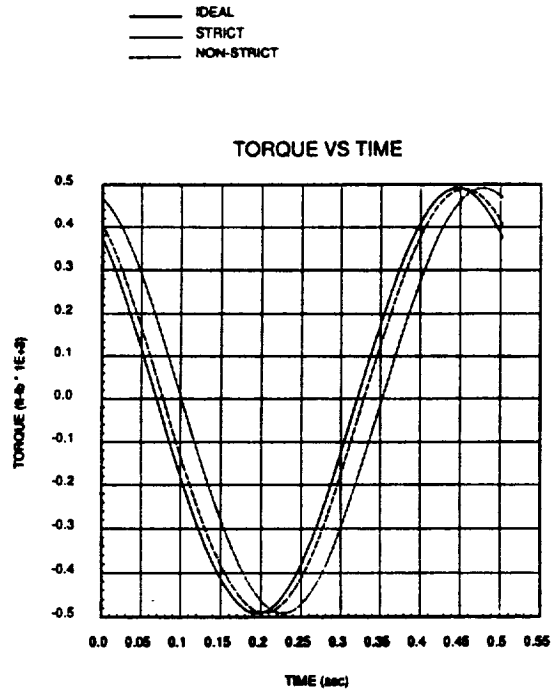


Figure 4: Plot of the Torque for the Ideal, Strict, and Non-Strict Cases Over a Period ($\omega = 12.5\ rad/s$ and $\Delta = \pi/1500\ s$).

A. Reduction in the Effective Latency

Fig. 4 compares the outputs of Strict and Non-Strict evaluations of the task graph of Fig. 3 against the ideal output obtained with zero computational delay. Results for $\omega = 12.5\ rad/s$ and $\Delta = \pi/1500\ s$ are given. If the tasks are scheduled in the order 1-2-3-4-5-6 instead of an order dictated by strict adherence to task precedence constraints, then speedup should result by a reduction of the effective latency of the computation. In particular, at high frequencies when the inertial term dominates the torque computation, it is anticipated that the computational delay may be as little as 3Δ since this is the most crucial path in that case. Of course, the schedule is not according to strict precedence and some amount of error will be introduced.

Note that the Strict curve is an exact form of the ideal but delayed in time by 15Δ. While the Non-Strict curve is not an exact form of the ideal, it gives the least amount of error for almost the entire period. For the present results, little difference can be seen between the shape of the ideal and Non-Strict curves. In fact, the Non-Strict curve does not quite reach the desired peak and the slopes are slightly different along the trajectory. However, the effective latency has been reduced considerably to as little as 20-30% of the Strict case.

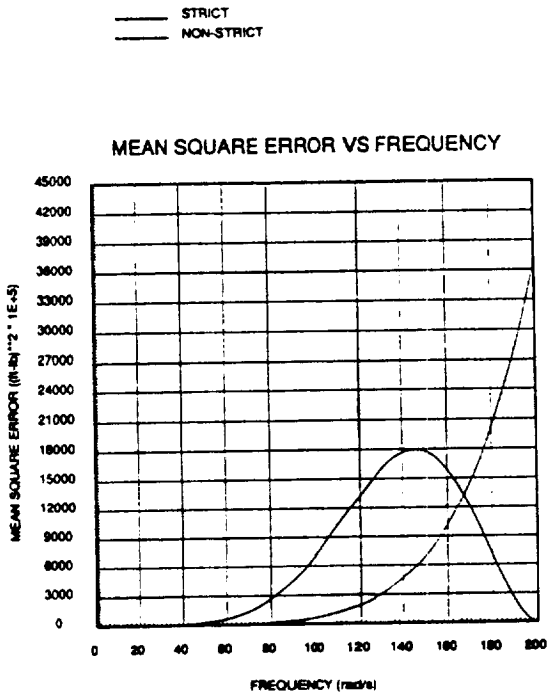


Figure 5: Mean Square Error for the Strict and Non-Strict Cases as a Function of the Frequency ($\Delta = \pi/1500\ s$).

B. Error Vs. Frequency

If the frequency is increased such that the period is exactly 15Δ, then the Strict approach will give perfect results. The question becomes: is there a range of frequencies over which the Non-Strict approach will give the best results (least error)? Toward this end, the error of the Strict and Non-Strict approaches with respect to the ideal have been evaluated as a function of the frequency. Typical results are given in Fig. 5.

The results indicate that the Non-Strict approach has a smaller error up to a frequency of approximately $\omega = 168\ rad/s$ when $\Delta = \pi/1500$. As might be expected, this crossover frequency varies with the value of Δ chosen. As the value of Δ increases, the crossover value of frequency decreases. In fact, as long as the value for Δ is less than a certain fraction of the period, then the Non-Strict approach will be better. For the example system, this ratio has been determined as:

$$\frac{\Delta}{T} \leq 0.056. \tag{5}$$

For this example, if the Strict computation delay (15Δ) is less than approximately 80% of

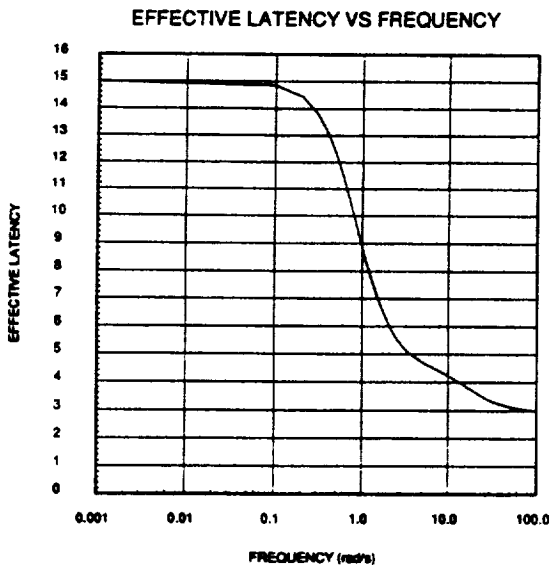


Figure 6: Effective Latency as a Function of Frequency for the Schedule (1-2-3-4-5-6).

the period, then the Non-Strict approach will give better results.

C. Effective Latency Vs. Frequency

The schedule assumed for the previous results tends to work well for high input frequencies, since in this region the crucial path is 1-2-3 and is scheduled first. Then for very high frequencies, it is anticipated that the overall effective latency will be approximately 3Δ which is the computational delay along this particular path. At low frequencies, the gravitational term will dominate the inertial term. With the same schedule, then, it is anticipated that the effective latency will be longer. In fact, the delay from the input q to the output τ is seen to be 15Δ. Therefore, the effective latency of the computation should be in the range of $3 - 15\Delta$ and will vary with the frequency.

Results have been obtained for the effective latency as a function of the frequency and are shown in Fig. 6. As expected, the effective delay is 15Δ at low frequencies and decreases to 3Δ at high frequencies. The crossover takes place in the region between $\omega = 0.1$ to $\omega = 10.0$ during which the significance of the inertial, damping, and gravitational terms changes.

If operation of the inverted pendulum is at lower frequencies, then an alternate schedule may be more appropriate. In particular, the schedule 4-5-3-1-6-2 should have an effective latency of 12Δ at low frequencies – better than the previous schedule. Results for this case are given in Fig. 7. The results are as expected. Note that the delay at high frequencies is 15Δ because of the long delay from sampling $\ddot{q}$ to the output. Also note that the damping

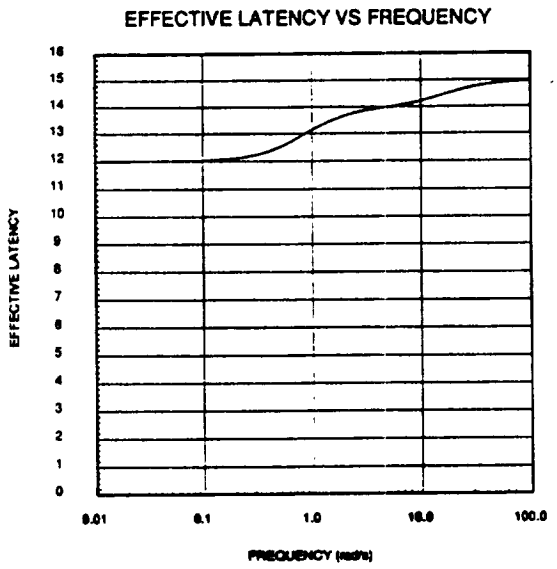


Figure 7: Effective Latency as a Function of Frequency for the Schedule (4-5-3-1-6-2).

term with $\dot{q}$ appears to dominate at approximate $\omega = 4$ so that the effective delay is 14Δ.

In fact, in general, the schedule should vary depending upon the rate of change of the inputs, and their effect on the output. For low frequencies, the path 4-5-3 should be favored while for high frequencies, the path 1-2-3 should be favored. This gives rise to a need for an adaptive scheduling scheme and will be considered in future investigations.

IV. Summary and Conclusions

This paper has proposed a Non-Strict computational approach attractive for real-time applications such as robotics control. In this approach, the precedence set in a task graph model of the computation is relaxed so as to reduce the effective latency from input to output. This is achieved by appropriately ordering the execution of the processes so that the most crucial path from input to output is given priority. While some amount of error is introduced since the precedence relationships are not strictly adhered to, there are many cases in which the overall error is less than that introduced by the sheer delay of the Strict case.

The Non-Strict approach has been applied to compute Inverse Dynamics [6] for a simple inverted pendulum. At high frequencies of motion, the effective latency was reduced from 15Δ to 3Δ, giving a speedup of 5. For a simple sinusoidal input on the joint angle, the error for the Non-Strict case was better than the Strict case as long as the Strict computational delay was less than approximately 80% of the period. This is the case in practical

applications. The effective latency for the Non-Strict case was shown to be a function of the frequency of the sinusoidal input. This results from a change in the crucial path through the task graph as the frequency varies. Two different schedules showed the range of possibilities for reducing the effective latency and motivated the need for an adaptive scheduling scheme.

There are a number of avenues for further work which are presently being explored. First of all, speedup has been obtained here for the case of a serial machine. The use of a Non-Strict computational approach can clearly be expected to have an impact on exploitation of parallelism. The strict precedence constraints imposed by a conventional task-graph model tend to limit parallelism. The relaxation of precedence constraints with the Non-Strict approach should thus be conducive to the better exploitation of parallelism. However, in the parallel context, the determination of an optimal scheduling order becomes more difficult, with the two inter-related criteria of 1) task ordering for minimization of "effective latency" through use of a measure of "task crucialness", and 2) minimization of computational latency through maximization of processor utilization.

A second issue is that of adaptive scheduling. Based on the rate of change of a process output and its significance to the computation, a constantly adapting schedule should give better results than one which is fixed. Also, processes need not be scheduled at the same rate and such multi-rate schemes may be especially important in effectively utilizing the resources of a parallel processor system. Further work in scheduling strategies is needed.

Another area in which the work may be extended involves the use of prediction to reduce the effective latency and computational error. In particular, if prediction is made on the inputs before use by a process, then it may be possible to reduce the effective process latency and therefore the overall latency. Investigations into appropriate schemes for using the prediction are needed.

Finally, the applicability of the Non-Strict computational approach to a wider class of real-time control problems should be investigated. The preliminary results presented in this paper show much promise for speedup and should be applied to other difficult real-time computational problems.

The Non-Strict computational approach seems promising as a first step towards a framework for specifying real-time robot dynamics computations in a machine-independent manner. It is current practice to choose the model/algorithm with a view to what is implementable to provide real-time response when executed on a given target machine. The flexible and adaptive scheduling of tasks inherent with the Non-Strict approach suggests that a single algorithmic representation can be adequate for slow and fast processors alike, for single as well as multi-processors. This seems quite feasible especially if computationally inexpensive predictors/extrapolators are used as alternatives in conjunction with more accurate but computationally demanding task blocks. A powerful target machine might schedule the computationally demanding blocks every time needed and thus obtain great accuracy, whereas a weaker target machine might schedule those computationally intensive blocks relatively infrequently and use the predictors in between, thereby trading off accuracy

for timeliness.

Work is currently in progress in evaluating the Non-Strict approach with more complex robot dynamics computations, and in investigating the use of predictors and multi-rate task scheduling strategies. A testbed is also under development on a BBN Butterfly multiprocessor to investigate Non-Strict scheduling in a parallel setting. It is hoped that with further work the Non-Strict approach can be established as an effective approach for complex real-time domains such as robotics control.

Acknowledgements

This work was supported by the National Science Foundation under Computational Engineering Grant No. EET-8718434.

References

[1] J. Barhen, "Robot Inverse Dynamics on a Concurrent Computation Ensemble," in *Proc. of 1985 ASME International Conference on Computers in Engineering*, pp. 415–429, Boston, MA, August 1985.

[2] J. Y. S. Luh and C. S. Lin, "Scheduling of Parallel Computation for a Computer-Controlled Mechanical Manipulator," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-12, no. 2, pp. 214–234, March 1982.

[3] D. E. Orin, K. W. Olson, and H. H. Chao, "Systolic Architectures for Computation of the Jacobian for Robot Manipulators," in *Computer Architectures for Robotics and Automation*, J. H. Graham, Ed., pp. 39–67, New York: Gordon and Breach Science Publishers, 1987.

[4] K. J. Lin, S. Natarajan, and J. W. S. Liu, "Imprecise Results: Utilizing Partial Computations in Real-Time Systems," in *Proc. of 8th IEEE Real-Time Systems Symposium*, pp. 210–217, December 1987.

[5] H. Hemami, I. C. Wall, F. O. Black, and G. L. Golliday, "Single Inverted Pendulum Biped Experiments," *Journal of Interdis. Model. & Simulation*, vol. 2, no. 3, pp. 211–227, 1985.

[6] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul, "On-line Computational Scheme for Mechanical Manipulator," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 102, pp. 69–76, June 1980.

[7] C. H. An, C. G. Atkeson, and J. M. Hollerbach, *Model-Based Control of a Robot Manipulator*. Cambridge, MA: The MIT Press, 1988.

"Computational Chaos" in Massively Parallel Neural Networks

Jacob Barhen and Sandeep Gulati
Jet Propulsion Laboratory/California Institute of Technology

Abstract

A fundamental issue which directly impacts the scalability of current theoretical neural network models to massively parallel embodiments, in both software as well as hardware, is the inherent and unavoidable concurrent asynchronicity of emerging fine-grained computational ensembles and the possible emergence of chaotic manifestations. Previous analyses attributed dynamical instability to the topology of the interconnection matrix, to parasitic components or to propagation delays. However, we have observed the existence of "emergent" computational chaos in a "concurrently asynchronous" framework, independent of the network topology. In this paper we present a methodology enabling the effective asynchronous operation of large-scale neural networks. Necessary and sufficient conditions guaranteeing concurrent asynchronous convergence are established in terms of contracting operators. Lyapunov exponents are computed formally to characterize the underlying nonlinear dynamics. Simulation results are presented to illustrate network convergence to the correct results, even in the presence of "large" delays.

Multi-Flexible-Body Dynamics Capturing Motion-Induced Stiffness

Arun K. Banerjee*, Mark E. Lemak**, and John M. Dickens***
Lockheed Missiles & Space Co.

Abstract

This paper presents a multi-flexible-body dynamics formulation incorporating a recently developed theory for capturing motion induced stiffness for an arbitrary structure undergoing large rotation and translation accompanied by small vibrations. In essence, the method consists of correcting prematurely linearized dynamical equations for an arbitrary flexible body with generalized active forces due to geometric stiffness corresponding to a system of twelve inertia forces and nine inertia couples distributed over the body. Equations of motion are derived by means of Kane's method. A useful feature of the formulation is its treatment of prescribed motions and interaction forces. Results of simulations of motions of three flexible spacecraft, involving stiffening during spinup motion, dynamic buckling, and a repositioning maneuver, demonstrate the validity and generality of the theory.

---

* Senior Staff Engineer
** Senior Research Engineer
*** Staff Engineer

NONLINEAR STRAIN-DISPLACEMENT RELATIONS AND FLEXIBLE MULTIBODY DYNAMICS

Carlos E. Padilla
Andreas H. von Flotow
Massachusetts Institute of Technology
Cambridge, Massachusetts

ABSTRACT

This paper considers dynamics of chains of flexible bodies undergoing large rigid body motions, but small elastic deflections. The role of nonlinear strain-displacement relations in the development of the motion equations correct to first order in elastic deflections is investigated. The general form of these equations linearized only in the small elastic deflections is presented, and the relative significance of various nonlinear terms is studied both analytically and through the use of numerical simulations. Numerical simulations are performed for a two link chain constrained to move in the plane, subject to hinge torques. Each link is modeled as a thin beam. Slew maneuver simulation results are compared for models with and without properly modeled kinematics of deformation. The goal of this case study is to quantify the importance of the terms in the equations of motion which arise from the inclusion of nonlinear strain-displacement relations. It is concluded that unless the consistently linearized equations in elastic deflections and speeds are available and necessary, the inconsistently (prematurely) linearized equations should be replaced in all cases by "ruthlessly" linearized equations: equations in which all nonlinear terms involving the elastic deflections and speeds are ignored.

1 INTRODUCTION

In recent years a fundamental limitation of the finite element formulation of flexible deformations in flexible multibody simulation programs such as TREETOPS[1], DISCOS[2], etc. has been pointed out [3,4,5]. This limitation could be characterized as a premature linearization of velocity expressions that is implicit in a linear finite element or modal formulation of the motion equations for flexible bodies [4]. Kane et al. [6] demonstrated this flaw of such an approach numerically by simulating a simple system consisting of a flexible beam attached to a rigid base spinning in the plane. This simulation yielded the surprising and intuitively wrong result of the beam diverging during a spin up maneuver [3]. Probably because of this simple example the "prematurely linearized" equations of motion are said to lack the "spin-stiffening effect." Further study of this simple system showed that this limitation of the traditional approach does not significantly affect simulation results for some maneuvers typical of space dynamical systems, such as repositioning slewing maneuvers and station keeping [4].

This paper presents a general discussion of the equations of motion of a flexible multibody system undergoing motion with large rigid body rates (not just angular velocities) and rigid body configuration changes, but with infinitesimal elastic deformations. The generic equations of motion for such systems are presented; linearized only in elastic deformation amplitudes. The implicit assumption here is that this is a useful set of motion equations: nonlinear in rigid body rates and coordinates but linear in flexible coordinates and rates. Some of the terms in these motion equations cannot be obtained with linear kinematics of elastic deformation (i.e., the traditional linear finite element or modal formulation). This paper illuminates the form of these practically unobtainable (in the general case) terms, evaluates their relative importance and examines the possible consistent simplifications of the motion equations.

The practical impact of these simplifications is investigated with the use of two case studies. The consistent equations of motion, derived with nonlinear strain-displacement relations, are explicitly given for two systems: 1) a single Bernoulli-Euler beam cantilevered to a rigid base undergoing large rigid body, but small flexible motions in the plane; and, 2) a two-link, revolute, planar, flexible manipulator consisting of three rigid bodies (shoulder, elbow and end effector) connected by two Bernoulli-Euler beams. These two examples will serve to emphasize the general discussion and to quantify the magnitudes of the neglected and retained terms in the equations of motion. This will be done both analytically and through comparison of simulation results.

Before proceeding to a more general discussion of the "correct" linearized equations of motion, a brief explanation of how these equations are obtained through proper linearization, and of the role of nonlinear strain-displacement relations is in order.

2 NONLINEAR STRAIN-DISPLACEMENT AND PROPER LINEARIZATION

The problem that concerns us is that of obtaining the correctly linearized equations of motion for an important class of systems which exhibit large rigid body motions but small elastic deflections. By far the most common practice to date is to handle the flexibility through discretization of the desired continuous system. This is achieved by representing the solution as a finite series of time-dependent generalized elastic coordinates multiplied by space-dependent functions, as in an assumed modes approach, or in a finite element formulation [7]. Whichever formulation one uses, in light of the class of systems under study, the next step is to assume that these elastic coordinates, together with the generalized elastic speeds, are infinitesimally small. In other words, we assume these coordinates and speeds to be small enough so that only terms linear in them are kept in the equations of motion, as terms of second order or higher are negligible.

Now that our goal is clearly stated, it should be an easy matter to obtain the linearized equations of motion as long as we consistently drop all terms nonlinear in the elastic coordinates and the corresponding generalized elastic speeds. Of course, the word "consistently" is the catch. When do we linearize? That is to say: Does it matter at what step in our derivation of the equations of motion we start to linearize? To answer this question we have to consider the process by which we derive these equations.

Let us consider two of the more widely known methods to derive motion equations for complex systems: Lagrange's equations of motion [8] and Kane's dynamical equations [9]. Lagrange's equations for a holonomic system with n generalized coordinates q_k:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_k}\right) - \frac{\partial L}{\partial q_k} = Q_k, \qquad (k = 1, \ldots, n)$$
$$L = T - V \tag{1}$$

where the Lagrangian L is a function of the system kinetic and potential energies (T and V respectively). Q_k are n generalized non-potential forces. The important thing to note is that using this method to derive motion equations requires differentiating both the potential and kinetic energies of the system with respect to the generalized coordinates and speeds. If these q_i and u_i ($=dq_i/dt$) were to represent our generalized elastic coordinates, we see that the above differentiations imply that some terms linear in q_i and u_i in the energy expressions become terms of zeroth order in q_i and u_i. More importantly, we see that terms of second order in the generalized coordinates and speeds in the energy expressions become terms of first order in the resulting equations of motion. Clearly then in order to obtain equations of motion correct to first order in q_i and u_i we need to have energy expressions for our system that are correct to second order in these same elastic generalized coordinates and speeds. More specifically the requirement demands in general that the expressions for displacements and velocities used in determining potential and kinetic energies be correct to second order in the elastic coordinates and speeds. Only by doing this can we ensure consistent linearization.

Kane's dynamical equations for a holonomic system of n particles with n generalized speeds u_i:

$$
\begin{aligned}
& F_r + F_r^* = 0 \qquad\qquad (r = 1, \ldots, n) \\
& F_r^* = \sum_{i=1}^{n} \mathbf{v}_r^{P_i} \bullet \mathbf{R}_i^* \quad , \quad \mathbf{R}_i^* = - m_i \vec{a}_i \\
& \mathbf{v}^{P_i} = \sum_{i=1}^{n} \mathbf{v}_r^{P_i} u_i + \mathbf{v}_t \quad , \quad u_i = \dot{q}_i
\end{aligned}
\tag{2}
$$

where F_r is the generalized active force, F_r^* is the generalized inertia force, $\mathbf{R}_i^*$ is the inertia force for particle P_i in an inertial reference frame, and $\mathbf{v}^{P_i}$ is the velocity of this particle in the same frame. $\mathbf{v}_r^{P_i}$ is the r-th partial velocity of particle P_i in the inertial frame. Using a similar argument as above, it is easy to see that since the partial velocity has to be correct to first order in the generalized coordinates and speeds, and again this term is obtained through differentiation of the velocity with respect to the generalized speeds, the velocity has to be correct to second order in q_i and u_i until we form the partial velocities. This is necessary if we want our equations of motion to be consistently linear in the generalized coordinates and speeds.

3 FORM OF THE EQUATIONS OF MOTION FOR A CHAIN OF ELASTIC BODIES

The equations of motion of an open chain of elastic bodies can be expressed quite generally as [10]:

$$
\begin{bmatrix} M_{RR}(x,q) & M_{RE}(x,q) \\ M_{ER}(x,q) & M_{EE}(x,q) \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} T_{C,R} \\ T_{C,E} \end{bmatrix} + \begin{bmatrix} T_{ext,R} \\ T_{ext,E} \end{bmatrix}
+ \begin{bmatrix} F_R(x, q, \dot{x}, u) \\ F_E(x, q, \dot{x}, u) \end{bmatrix}, \qquad u = \dot{q}
\tag{1}
$$

where x is a vector of rigid body generalized coordinates; q is a vector of the elastic generalized coordinates; M_{RR}, M_{RE}, M_{ER}, M_{EE} form the configuration-dependent mass matrix; T_C is a vector of control forces (as in joint-torque actuators in a manipulator); T_{ext} is a vector of other generalized external forces; K_{EE} is a constant stiffness matrix (see equation (2) below) and F is a vector of nonlinear inertial (coriolis and centripetal) forces.

We are often interested in the important class of systems for which the elastic deformations remain small so we can ignore terms of second order in q and u. Strictly speaking this requires that $\|q\|$ and $\|u\|$ be infinitesimally small. However we know that if, for example, our flexible body is modelled as a beam, it is sufficient that the elastic deformations do not exceed one tenth of the length of the beam in order to use linear Bernoulli-Euler beam theory. At any rate, given this assumption of small elastic deflections, we could expand the previous equation in order to show more explicitly the form of the nonlinear terms:

$$
\begin{bmatrix} M_{RR}(x,q) & M_{RE}(x,q) \\ M_{ER}(x,q) & M_{EE}(x,q) \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} T_{C,R} \\ T_{C,E} \end{bmatrix} + \begin{bmatrix} T_{ext,R} \\ T_{ext,E} \end{bmatrix}
$$

$$
- \begin{bmatrix} 0 & 0 \\ 0 & K_{EE} \end{bmatrix} \begin{bmatrix} x \\ q \end{bmatrix} + \sum_{i=1}^{n} f_{1ii}(x) \dot{x}_i^2 + \frac{1}{2} \sum_{i=1}^{n} \sum_{j \neq i}^{n} f_{1ij}(x) \dot{x}_i \dot{x}_j
$$

$$+\frac{1}{2}\sum_{i=1}^{n} f_{2ii}(x)q\dot{x}_i^2+\frac{1}{2}\sum_{i=1}^{n}\sum_{j\neq i}^{n} f_{2ij}(x)q\dot{x}_i\dot{x}_j$$

$$+\sum_{i=1}^{n} f_{3i}(x)\dot{q}\dot{x}_i-\begin{bmatrix} M^*_{RR}(x,q) \\ M^*_{ER}(x,q)\end{bmatrix}\ddot{x},$$

$$\left(M^*_{RR}(x,q)\right)_{ij}=m^T_{1ij}(x)q,\quad \left(M^*_{ER}(x,q)\right)_{ij}=m^T_{2ij}(x)q \tag{2}$$

where n above is the number of rigid body coordinates; f_{1ij} is a column matrix, but f_{2ij} and f_{3i} are n+m by m matrices, where m is the number of elastic coordinates.

In light of the discussion in the previous section, we could further write:

$$\begin{aligned} f_{2ij}(x) &= f^1_{2ij}(x)+f^2_{2ij}(x) \\ m_{1ij}(x) &= m^1_{1ij}(x)+m^2_{1ij}(x) \\ m_{2ij}(x) &= m^1_{2ij}(x)+m^2_{2ij}(x) \end{aligned} \tag{3}$$

where the superscript 2 terms can only be obtained through the use of displacement and velocity expressions, in the development of the equations of motion, that are accurate to second order in the elastic generalized coordinates and speeds (q and u). This requires the use of nonlinear strain-displacement relations [11,12], nonlinear kinematic constraints [6,4], or the use of a nonlinear "geometric stiffening" term appended to the incorrectly linearized equations of motion [5,13].

The complexity of equations (2) and the difficulty involved in obtaining the nonlinear terms have prompted attempts at simplification. It is common [10] for example to assume small velocities and drop all terms nonlinear in rates. This results in rate-linear motion equations that greatly simplify the dynamicist's task. It has been pointed out [14], however, that in the case of n-link rigid manipulators in any configuration, the velocity and acceleration terms of the dynamic equations have the same relative significance at any speed of movement. The fact that the omission of these terms does not significantly affect simulation results is attributed to the fact that gravity and joint friction usually overpower inertial terms. These results have not been extended to chains of flexible bodies. One might argue that in some limit (i.e., vanishingly small q) the equations of motion of the flexible multibody system should reduce to those of the rigid multibody system. Then it seems that a good case could be made for the inclusion of at least nonlinear terms in the rigid body rates in our rate-linear equations (i.e., $f_{1ij}(x)$), particularly considering the fact that future, fast, space manipulators with low joint frictions are part of the class of systems under consideration.

Faced with this, we can proceed in two ways with respect to the equations of motion: we can be consistent, or we can be selective. To be consistent requires keeping all terms of order $\|q\|$ and $\|u\|$ in equation (2), i.e., no simplification. We also encounter the problem that the superscript 2 terms in equations (3) are not readily available in the general case since they depend on nonlinear elastic theory or nonlinear kinematics of deformation for their derivation. We could just make do with the superscript 1 terms in equations (3) (standard approach) but this would not be consistent nor justifiable since there is no *a priori* reason to guarantee $\|f_{2ij}{}^1\| >> \|f_{2ij}{}^2\|$, for example.

We are forced then to be selective, at least in the general case. Now we have to rely mostly on experience and simulation to determine which terms are important and which negligible under given conditions. Using the simple example of a beam radially cantilevered to a spinning hub, an empirical speed limit has been proposed beyond which the standard linear finite element or modal formulations of the model give erroneous results [5]. This limit is specified as follows: the magnitude of the spinning rate of the system has to be one order of magnitude less than the fundamental bending frequency of the beam. We use this simulation result to claim the following:

unless the equations of motion exact to first order in elastic generalized coordinates and speeds are available, we are limited to rigid body angular rates less than an order of magnitude lower than the lowest fundamental bending frequency of our system. In view of this, we might be able to model our system accurately enough by just keeping the rate-linear equations together with terms f_{1ij}. In other words, we might as well drop all nonlinear terms involving elastic coordinates and speeds. We further claim that speed or acceleration limits also exist in translational rates or accelerations, arising from those mass matrix terms that depend on q and cannot be obtained through the standard approaches. In the next section we investigate these claims analytically and through simulation.

In what follows we shall refer to three types of models for a flexible multibody system under study. The "consistent" model will be that which retains all terms to first order in q and u, that is, the consistently linearized model. The "inconsistent" model shall be that obtained through linear kinematics of deformation, that is, one whose equations omit the superscript 2 terms mentioned above. Finally, a "ruthlessly linearized," or simply "ruthless" model, shall be one in which the nonlinear terms which include elastic coordinates and speeds are ignored, including those terms in the mass matrix which depend on elastic coordinates. In other words, in our "ruthless" model equations we ignore terms f_{2ij} and f_{3i}, and we assume the mass matrix depends on rigid body configuration only.

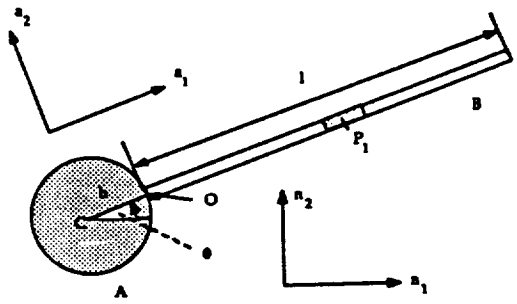


Fig. 1: Single Beam Attached to a Moving Base

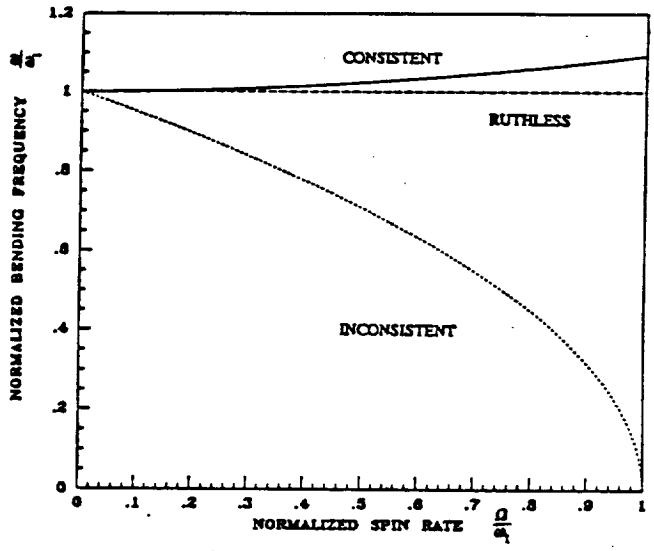


Fig. 2: Fundamental Bending Frequency of Spinning Beam vs Spin Rate

4 TWO CASE STUDIES

4.1 One flexible body example

Consider a simple, slender, uniform beam cantilevered to a rigid body free to move in the plane (see Fig. 1). The frame N, defined by the unit vectors $\mathbf{n}_1$, $\mathbf{n}_2$, $\mathbf{n}_3$, is inertial, and we introduce the rotating frame A defined by the unit vectors $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$, attached to body A and whose $\mathbf{a}_1$ axis lies along the undeformed neutral axis of the beam B initially. The consistently linearized equations of motion for this system have been derived using Kane's dynamical equations together with nonlinear strain-displacement relations. Shear and rotary inertia effects have been ignored (i.e., slender beam assumption). The equations of motion, exact to first-order in generalized elastic coordinates and speeds are [15]:

$$
\begin{bmatrix}
m_A + m_B & 0 & -\sum_{i=1}^{n} E_i q_i & \cdots & 0 & \cdots \\
0 & m_A + m_B & bm_B + e & \cdots & E_i & \cdots \\
-\sum_{i=1}^{n} E_i q_i & bm_B + e & b^2 m_B + 2eb + I_B + {}^{'}I_A & \cdots & bE_i + F_i & \cdots \\
\vdots & \vdots & \vdots & \ddots & \vdots & \iddots \\
-\sum_{i=1}^{n} \mu_{ij} q_i & E_j & bE_j + F_j & \cdots & G_{ij} & \cdots \\
\vdots & \vdots & \vdots & \iddots & \vdots & \ddots
\end{bmatrix}
\begin{bmatrix} \dot{v}_1 \\ \dot{v}_2 \\ \dot{v}_3 \\ \vdots \\ \ddot{q}_i \\ \vdots \end{bmatrix} =
$$

$$
\begin{bmatrix}
(m_A + m_B) v_2 v_3 + v_3^2 (bm_B + e) + 2v_3 \sum_{i=1}^{n} E_i \dot{q}_i \\
-(m_A + m_B) v_3 v_1 + v_3^2 \sum_{i=1}^{n} E_i q_i \\
-v_2 v_3 \sum_{i=1}^{n} E_i q_i - v_3 v_1 (bm_B + e) \\
\vdots \\
- v_3 v_1 E_j + v_3^2 \sum_{i=1}^{n} G_{ij} q_i - v_3^2 \sum_{i=1}^{n} (b\mu_{ij} + \eta_{ij}) q_i - v_2 v_3 \sum_{i=1}^{n} \mu_{ij} q_i \\
\vdots
\end{bmatrix} +
$$

$$
\begin{bmatrix}
\cdots & 0 & \cdots \\
\cdots & 0 & \cdots \\
\cdots & 0 & \cdots \\
\ddots & \vdots & \iddots \\
\cdots & -H_{ij} & \cdots \\
\iddots & \vdots & \ddots
\end{bmatrix}
\begin{bmatrix} \vdots \\ q_i \\ \vdots \end{bmatrix} \tag{1}
$$

where following Kane et al. [16] we have defined:

$$
m_B \equiv \int_0^L \rho dx, \quad e \equiv \int_0^L x \rho dx, \quad I_B \equiv \int_0^L x^2 \rho dx
$$

$$
H_{ij} \equiv \int_0^L EI \phi_{2i}''(x) \phi_{2j}''(x) dx, \quad E_i \equiv \int_0^L \phi_{2i}(x) \rho dx
$$

$$
F_i \equiv \int_0^L x \phi_{2i}(x) \rho dx, \quad G_{ij} \equiv \int_0^L \phi_{2i}(x) \phi_{2j}(x) \rho dx
$$

$$
(i, j = 1, \ldots, n)
$$

and we have further defined:

$$\mu_{ij} \equiv \int_0^L \rho \int_0^x \phi'_{2i}(\sigma)\phi'_{2j}(\sigma)d\sigma dx,$$

$$\eta_{ij} \equiv \int_0^L x\rho \int_0^x \phi'_{2i}(\sigma)\phi'_{2j}(\sigma)d\sigma dx$$

In deriving the above equations, we have assumed no external forces act on the beam for simplicity. m_A is the mass of the rigid body A; v_1 and v_2 are translational speeds of body A in the directions of $\mathbf{a}_1$ and $\mathbf{a}_2$ respectively; v_3 is the rotational speed of body A; q_i are the n generalized elastic coordinates, where we have discretized the transverse elastic deformation of the beam using assumed modes.

Specializing equation (1) to the prescribed motion of uniform rotation of the base,

$$v_1 = v_2 = 0, \quad v_3 = \Omega = \text{constant}$$

we get:

$$\sum_{i=1}^{n} G_{ij}\ddot{q}_i + \sum_{i=1}^{n}\left[H_{ij} + \Omega^2(b\mu_{ij} + \eta_{ij} - G_{ij})\right]q_i = 0$$
$$(j = 1, \ldots, n) \tag{2}$$

Note that the terms μ_{ij} and η_{ij} cannot be obtained using linear kinematics of deformation but are obtained through the use of nonlinear strain-displacement relations. The term

$$\Omega^2(b\mu_{ij} + \eta_{ij})$$

is known as the geometric stiffness matrix for this specialized rigid body motion (rotation). It is easy to observe from just this analytical study that in the absence of these terms our equations lose stiffness with increasing angular rates Ω, since:

$$(K)_{ij} = H_{ij} - \Omega^2 G_{ij} = \int_0^L EI\phi''_{2i}(x)\phi''_{2j}(x)dx$$
$$- \Omega^2 \int_0^L \rho\phi_{2i}(x)\phi_{2j}(x)dx \tag{3}$$

We note that for a variety of mode shapes [17]:

$$\int_0^L \phi_{2i}(x)\phi_{2j}(x)dx = \int_0^L \phi''_{2i}(x)\phi''_{2j}(x)\left(\frac{L}{\lambda_i}\right)^2\left(\frac{L}{\lambda_j}\right)^2 dx$$
$$= \begin{cases} L, & i = j \\ 0, & i \neq j \end{cases}$$
$$\lambda_i^2 = \omega_i L^2\sqrt{\frac{\rho}{EI}} \tag{4}$$

so

$$(K)_{ij} = \begin{cases} \omega_i^2\rho L - \Omega^2\rho L, & i = j \\ 0, & i \neq j \end{cases}$$

and it is clear that as long as $\Omega << \omega_i$ the incorrect de-stiffening effect will not be apparent. This clearly evidences the angular speed limit mentioned at the end of the previous section. As shall be seen, this rotational rigid body rate limit is the most restrictive on the validity of other than the correctly linearized equations for the maneuvers considered. Figure 2 shows the first bending eigenfrequency as a function of Ω, as predicted by the three modelling approaches, each using the same assumed modes.

If we now specialize equation (1) to the prescribed motion of constant translational acceleration:

$$du_1/dt = g = \text{constant}, \quad v_2 = v_3 = 0$$

we obtain:

$$\sum_{i=1}^{n} G_{ij}\ddot{q}_i + \sum_{i=1}^{n} \left(H_{ij} - g\mu_{ij}\right) q_i = 0 \qquad (j = 1, \ldots, n) \tag{5}$$

μ_{ij} is in this case the operative geometric stiffness matrix. Now we can see that for g large enough, we again obtain de-stiffening. Another way of looking at it is to realize that for g large enough the stiffness matrix becomes non-positive definite which implies that the beam buckles due to its own weight. The predicted buckling is correct, and would have been lost with the inconsistently or the ruthlessly linearized approaches. This suggests that a translational rate or acceleration limit also exists beyond which our model is again grossly incorrect if we do not use equations exact to first order. Inspection of equation (1) suggests that another rate limit exists, this one on the product of rigid body rotational and translational rates, $v_2 v_3$. Banerjee's recent work [13] suggests that as many as 12 such independent limits on rigid body motion exist and must be considered for general three-dimensional motion. It is doubtful that for typical stop-to stop slew and repositioning maneuvers these limits become operative, since the magnitude of the applied forces is limited by the requirement that the flexible body not deform excessively. This anticipation motivates the case study of the next section.

4.2 Two-link flexible arm example

Consider a two-link flexible, revolute arm composed of three rigid bodies connected by two slender uniform beams (see Fig. 3). The equations of motion for this system were derived using Kane's dynamical equations together with nonlinear strain-displacement relations. The equations are thus exact to first order in the beams' elastic generalized coordinates and speeds. We ignore independent axial extensions of the beams (i.e., the axial strain at the neutral axis is assumed zero for each beam). The elbow joint is actually modelled as two bodies: one attached to link 1 in a cantilevered way and the other, free to rotate with respect to the first, with link 2 attached to it in a cantilevered way also. Both elbow bodies are rigid and share the hinge point but their mass centers are allowed to be offset from the hinge point. As in the previous examples, the continuous transverse displacements of the beams are discretized using an assumed modes approach. One percent modal damping is added to the model to represent structural damping. Actuation is assumed in the form of shoulder and elbow torques. The equations of motion for this system, which are quite extensive, are available in [12].

An examination of the equations of motion for this system makes it clear that the complexity of the mass matrix and nonlinear inertial terms could be diminished immensely by dropping most terms involving the generalized elastic coordinates and speeds (i.e., the "ruthless" case). It would be advantageous to know, then, if these terms have a significant effect on the dynamics of a chain of elastic bodies if we are limited to the low rotational speeds and translational accelerations encountered during slew maneuvers with joint torques limited by the requirement that flexible deflections remain small. In order to investigate this, we propose to examine the three models mentioned at the end of section three, to note, consistent, inconsistent, and ruthless, as applied to the two link arm. The

complexity of the motion equations precludes an analytical study akin to the one presented in the previous section. We will therefore rely entirely on numerical simulation and compare the performance of the three "models" of the arm when it is subjected to a smooth slew maneuver.

5 NUMERICAL SIMULATION

The motion equations for the two-link, flexible, planar manipulator, consistently linearized in small elastic deflections and speeds, were programmed in FORTRAN and implemented in a VAXstation 2000. The code allows for a maximum of four cantilevered modes per beam. The mass matrix is inverted using LU decomposition and a Runge-Kutta fourth-order scheme with adaptive time step is utilized for the time integration [18]. Energy and angular momentum checks are built into the simulation. Table 1 shows the physical properties assumed for the arm (see also Fig. 3). These numbers were chosen to mimic an actual experimental testbed built at Martin Marietta by Dr. Eric Schmitz [19].

Physical Properties of Planar Manipulator with Two Flexible Links			
Mass of shoulder body (kg)	20.0	Length of link 1 (m)	0.9144
Mass density of link 1 (kg/m)	1.33937	Length of link 2 (m)	0.9144
Mass of elbow body (kg)	14.0		
Mass density of link 2 (kg/m)	0.669685	Other lengths (see Fig. 3):	
Mass of tip body (kg)	2.0	b_1 (m)	0.0762
		b_{21} (m)	0.0762
Moments of Inertia (about axis perpendicular to plane):		b_{22} (m)	0.0127
Shoulder body (kgm^2)	0.01	b_t (m)	0.0508
Elbow body (kgm^2)	0.03		
Tip body (kgm^2)	0.01		

Table 1: Physical Properties of the Two-link Manipulator

All the trajectories presented below were run in open loop after the torques had been computed from the inverse dynamics problem (given the desired angular trajectories) assuming rigid links for the manipulator. For all simulation runs, only two assumed modes per link were used, since this gave adequate results and was computationally much cheaper than running the full four modes per link. With two modes per link, the first two system vibration frequencies were obtained to within three percent of the value obtained using four modes per link.

In the following, reference is made to time-scaled trajectories. Time-scaling of nominal trajectories [20] is achieved by replacing time as the independent variable by the new variable

$$r = \alpha t, \qquad \alpha > 0$$

where α is a constant. When α is greater than one, the trajectory is sped up, while if α is less than one it is slowed down. From this it is apparent that rates scale like α, while accelerations, and thus torques, scale like the square of α. These relations are used in the following sections to select a scaling factor that yields a desired maximum value of angular rate, or a given maximum value of torque to obtain desired maximum link tip deflections.

5.1 A Smooth Slew Maneuver

In section 4.1 it was predicted that the more severe limit on the validity of other than consistently linearized equations would be the limit on rigid body angular rates. In the case of chains of flexible bodies, as exemplified by the two-link manipulator in this simulation, the nonlinearity arising from dependence on configuration makes it difficult to select a "characteristic" angular rate in

the general case. This suggests a case by case approach. Several different slews were compared in reference [12]. We report here only one, perhaps the most interesting.

The smooth trajectory is obtained by assuming a form of the joint angular time histories of the equivalent rigid manipulator that is quintic in time. This allows the specification of angle, angular rate and angular acceleration at the initial and final times of the trajectories [21]. This trajectory is a deployment maneuver. Both links undergo significant rotational motion, and the outboard link translates. Fig. 4 shows the computed torques for the nominal trajectory.

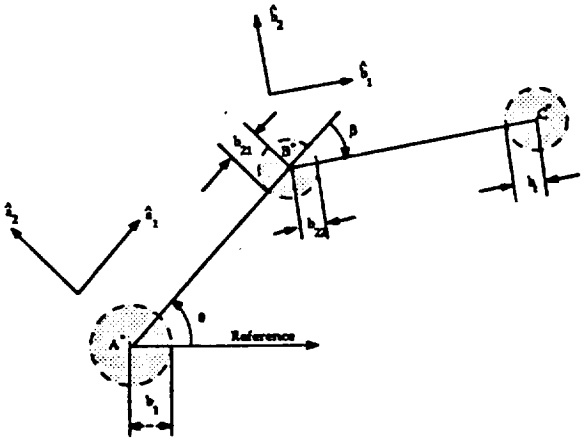


Fig. 3: Schematic of two-link Planar Manipulator

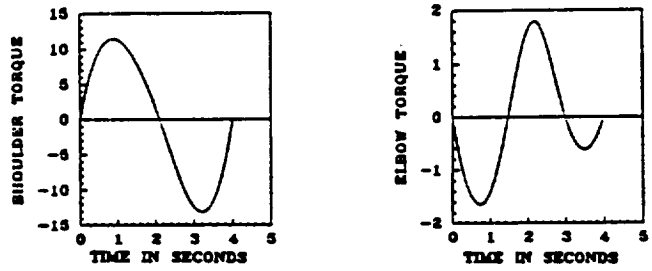


Fig. 4: Computed Torques for Second Smooth Trajectory (Nm)

Figure 5 shows the nominal trajectory. In this figure, plots for both the ruthless and the consistent models have been overlaid. The inconsistent model fails for this case. The failure is a numerical divergence during time integration. This is perhaps related to the fact that the angular rates for the nominal maneuver are as large as thirty percent of the "fundamental vibration frequency." The system frequencies for the manipulator with locked joints are seen to fluctuate from 3 rad/sec to 4 rad/sec for corresponding elbow relative angles of zero to 135 degrees [12]. For this reason, in the case at hand reference is made to "one" fundamental frequency, and it is assumed that it lies in the range specified above and is about 3.5 rad/sec. For the two models shown, the agreement is again excellent for the shoulder angle and tip deflections, with the elbow angular position being off by only a maximum of ten percent relative error.

Convergence of all three models is achieved if the nominal maneuver is slowed down (α=0.1685). Figure 6 shows that for this case, all three models yield identical results. This confirms the predictions in section 4.1, and further suggests that within the limit of validity of the inconsistent model, the ruthless is as good as the inconsistent, and actually better since it is much easier to obtain. Note that the angular rates are well within ten percent of the fundamental.

The last case considered consists of the nominal trajectory scaled upwards in time (α=1.2). As in the previous section, it was desired to reach the "hard" simulation limit of link tip deflections of about ten percent of the link lengths. As Figure 7 shows, only the ruthless model did not fail under the given speed-up of the trajectory, even though tip deflections should only be about four percent of link lengths. This divergence is traceable to a near singularity of the mass matrix, related to the fact that the links are modeled with distributed mass, while they are in fact nearly "massless springs." The manipulator mass distribution is dominated by the elbow and the tip mass. Reference [12] reports that the ratio of the largest to smallest mass matrix singular values is on the order of 10^6, and that this range is configuration dependent. It is not known why this numerical ill-conditioning of the

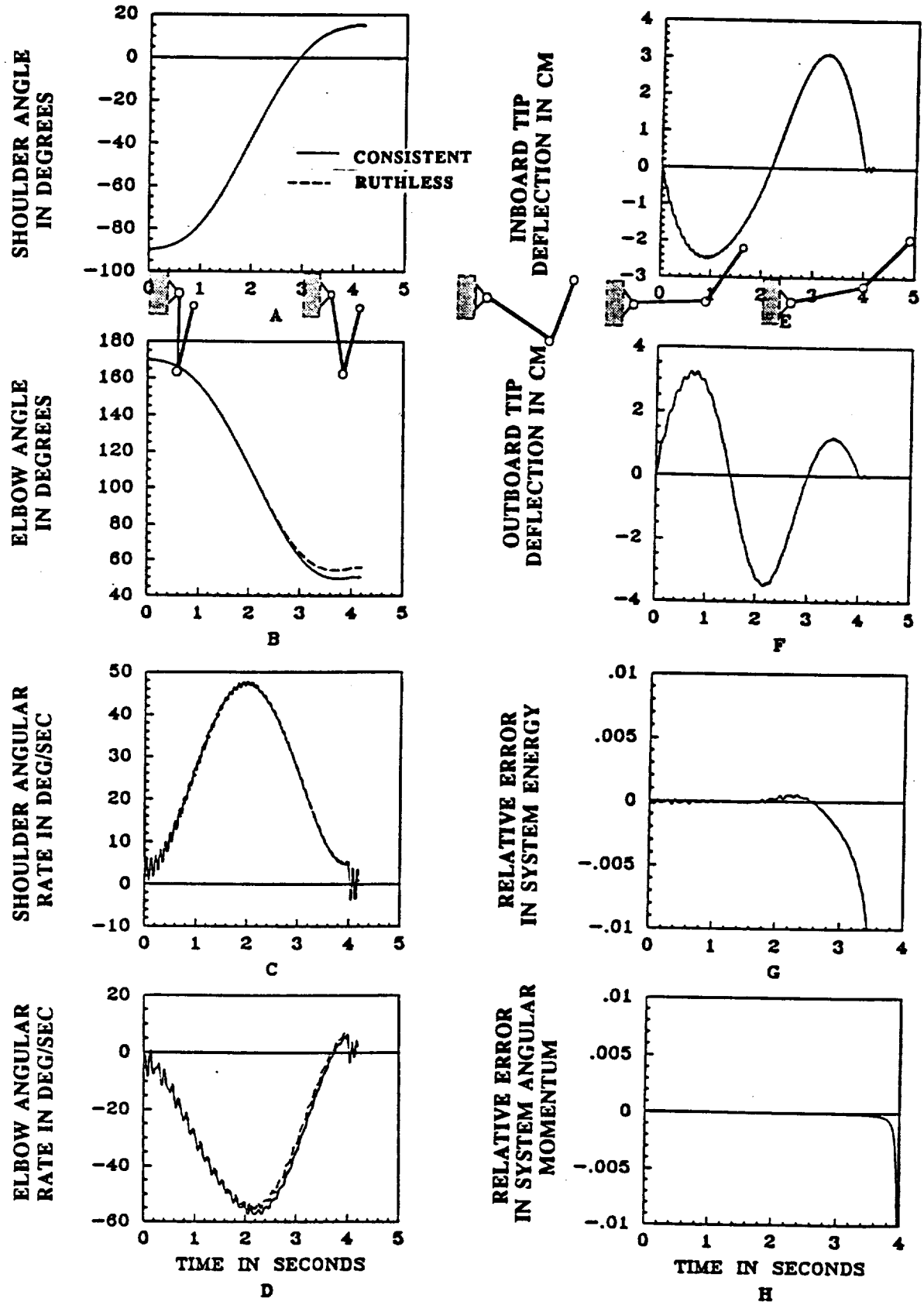


Fig. 5: Smooth Slew Maneuver with α=1.0

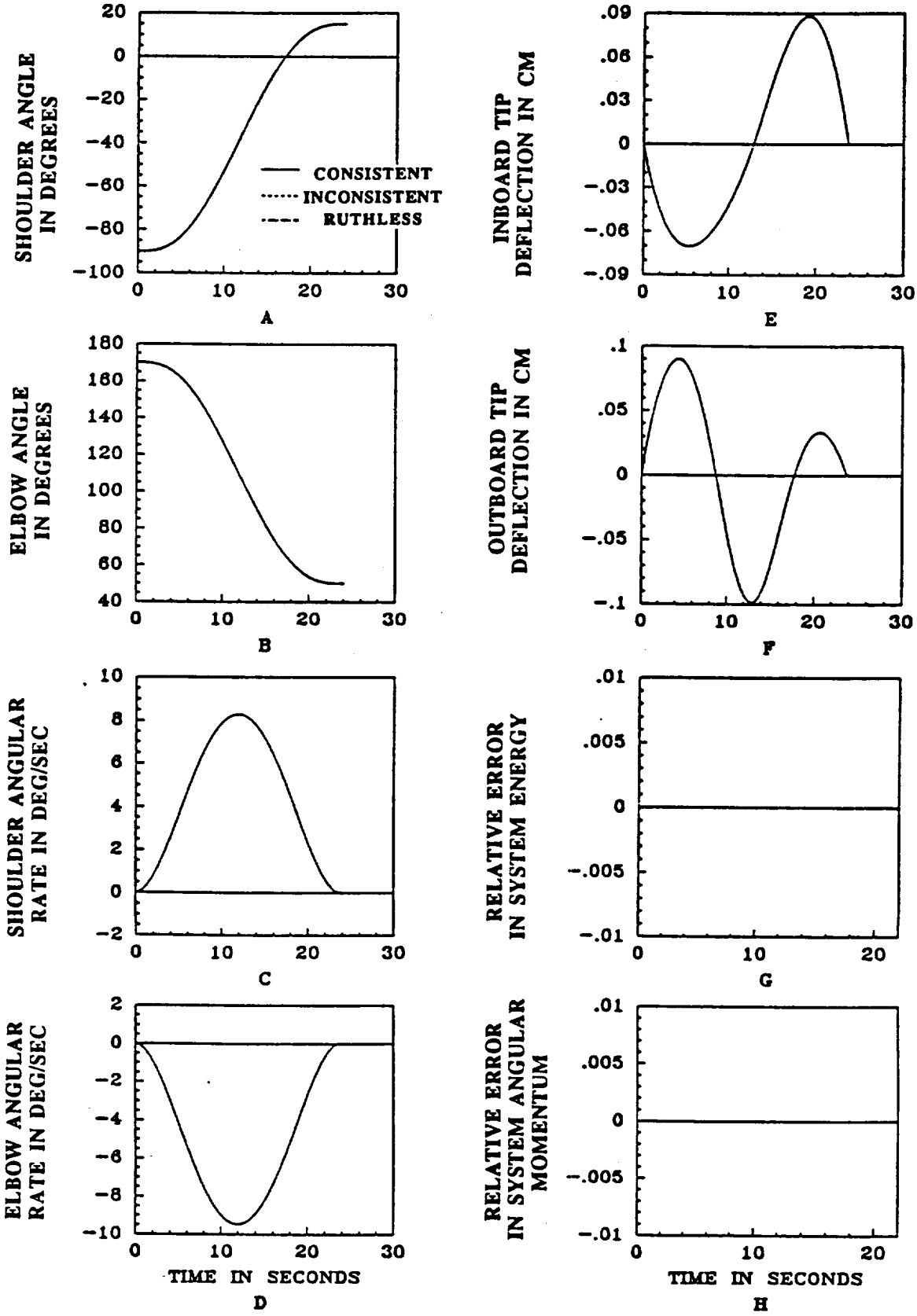


Fig. 6: Smooth Slew Maneuver with α=0.1685

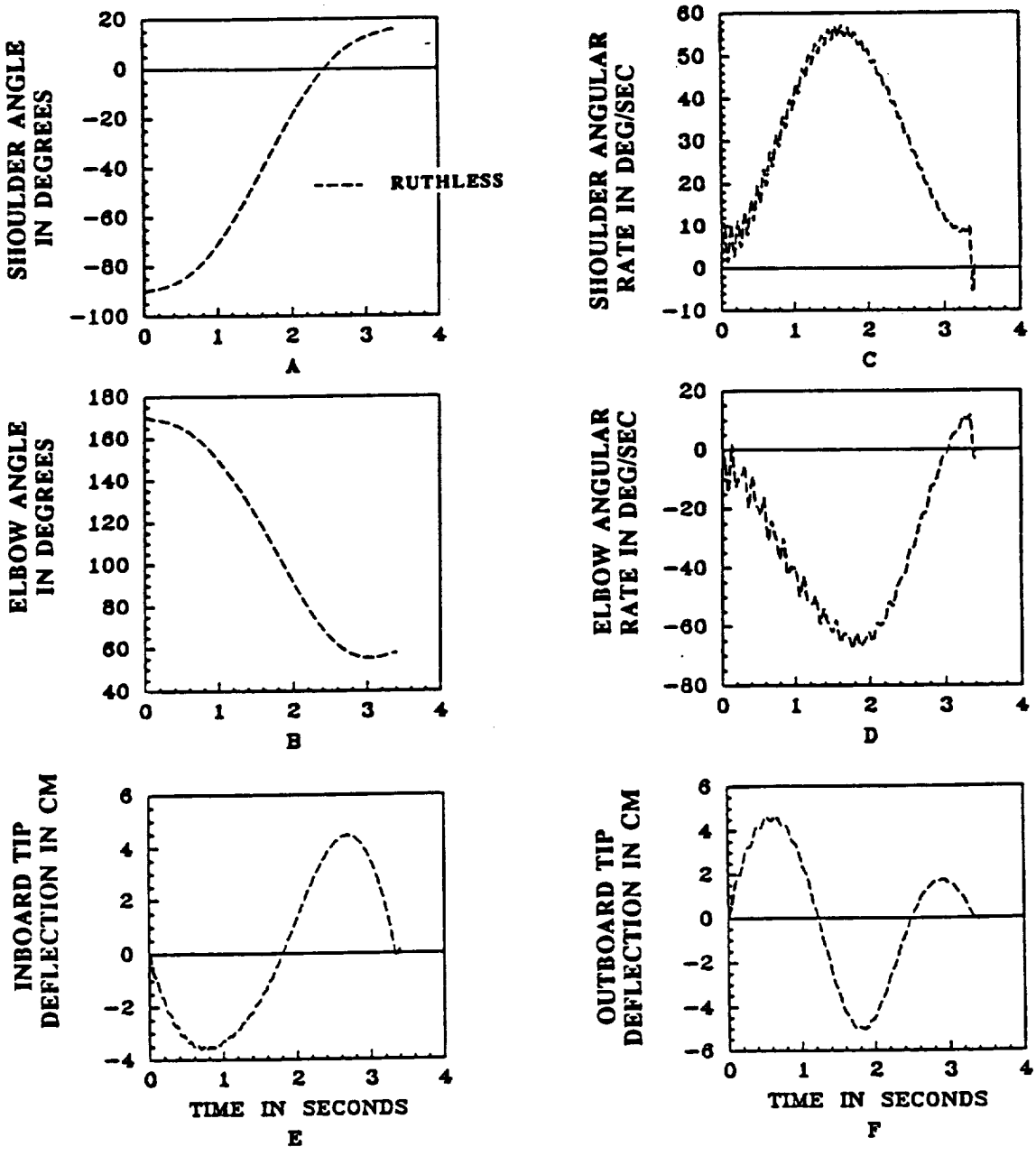


Fig. 7: Smooth Slew Maneuver with α=1.2

mass matrix appears to be exacerbated by including dependence upon elastic deformation, as in the consistent model.

In summary, the above results again show a strong correlation between the limit of validity of the inconsistent model and the maximum values of angular rates. In this case, however, no "characteristic" rigid body rate is apparent. The limit at which the inconsistent model fails seems to be even before any of the two angular rates (shoulder or elbow) reach ten percent of the fundamental vibration frequency. A strong point can still be made, nevertheless, in that the ruthless model is as good as the inconsistent whenever the inconsistent is valid, and more conservative since the ruthless model does not fail. Even at high angular rates the ruthless model yields results that are quantitatively very close to the consistent model results.

Finally, it is worth pointing out the excellent agreement in the tip deflections for both links, in all cases. This is probably due to the fact that the equations of motion (see [12]) are elastically decoupled, and, while inertially coupled, the elastic degrees of freedom mass matrix (M_{EE} of section 3) does not depend on elastic nonlinear terms due to linearization. Also, the agreement in shoulder angle and angular rates is remarkable. This indicates that, depending on what state variable is of interest in a given trajectory, the ruthless model will be as good as the more cumbersome consistent model. In all cases, the ruthless is more conservative and better conditioned than the inconsistent model.

5.2 Other Trajectories

The above qualitative results for the smooth slew maneuver have also been confirmed for two other maneuvers: a smooth maneuver in which the elbow motion is kept to a minimum and the two-link manipulator is slewed in extended configuration like a "beam"; and a time-optimal, bang-bang control slew maneuver where joint torques were assumed limited. Details of these results, together with an analysis of numerical considerations, can be found in [12].

6 SUMMARY AND CONCLUSIONS

Having looked into the general form of the linearized dynamics equations for chains of flexible bodies undergoing large rigid body motions, but small elastic deflections, we concluded that some terms cannot be obtained through the use of linear strain-displacement relations. These terms were seen to be critical in the simple rotating beam example as they provide the geometric stiffness terms necessary to obtain physical results. The absence of these terms in inconsistently linearized equations limits their validity to relatively gentle rigid body motions. The fact that these terms are unobtainable for the general case of an arbitrary flexible body led us to consider possible simplifications of the general motion equations, consistent with such restrictions on their applicability.

The two alternative models studied, the ruthlessly linearized model and the inconsistent model, are subject to several limits in applicability. While the consistent model requires we keep elastic coordinates and speeds small, the two alternative models will only be accurate if we further maintain low rigid body angular rates. There also exists some translational acceleration or speed limit that needs to be considered, although for the cases studied this limit was of no consequence. Within the domain of validity of both simplified models, it appears the ruthless model yields results as accurate as the correct consistently linearized model. In addition, preliminary results promise that the ruthless model will result in large reductions in computational time in the simulation of large flexible multibody systems. This coupled to the simplification of the dynamicist's task inherent in the adoption of ruthlessly linearized models makes this option an attractive alternative.

From the above it is clear that the inconsistent model should never be used. Further, the more cumbersome and hard to obtain consistent model should only be used when necessary (i.e., when the domain of validity of the simplified models is exceeded). Finally, it is our strong belief that the ruthless model deserves widespread use.

Simulation misbehavior at certain trajectories for relatively high rigid body angular rates was tracked down to numerical ill-conditioning of the configuration dependent mass matrix. This problem was attributed to modelling error inherent in choosing cantilever (clamped-free) modes to

model the flexible deflections of the manipulator links. In ref. [12] it is suggested that this results in effectively modelling one (or more) massless degrees of freedom. Thus it became apparent that physical modelling of bodies, and adequate selection of assumed modes and numerical procedures, can be as important as sensible simplifications of the motion equations.

ACKNOWLEDGEMENT

This research work was supported in part by a National Science Foundation Graduate Fellowship.

REFERENCES

[1] User's Manual for TREETOPS, "A Control System Simulation for Structures with a Tree Topology," Singh, Likins, et al., NASA contract no. NAS8-34588, Rev. B, November 1984.

[2] Bodley, C. S., et al., "A Digital Computer Program for the Dynamic Interaction Simulation of Controls and Structure (DISCOS)," Vols. 1 & 2, NASA Technical Paper 1219, May 1978.

[3] Ryan, R. R., "Flexible Multibody Dynamics: Problems and Solutions," JPL D-5190, Vol. I, Proceedings of the Workshop on Multibody Simulation, April 15, 1988, pp. 103-190.

[4] Ryan, R. R., "Flexibility Modeling Methods in Multibody Dynamics," Paper no. AAS 87-431, AAS/AIAA Astrodynamics Specialist Conference, Kalispell, Montana, August 10-13, 1987.

[5] Spanos, J., Laskin, R. A., "Geometric Nonlinear Effects in Simple Rotating Systems," JPL D-5190, Vol. I, Proceedings of the Workshop on Multibody Simulation, April 15, 1988, pp. 191-218.

[6] Kane, T. R., Ryan, R. R., Banerjee, A. K., "Dynamics of a Cantilever Beam Attached to a Moving Base," *Journal of Guidance, Control, and Dynamics*, Vol. 10, No. 2, March-April 1987, pp. 139-151.

[7] Meirovitch, L., *Elements of Vibration Analysis*, 2nd ed., McGraw-Hill Book Company, 1986.

[8] Pars, L. A., *A Treatise on Analytical Dynamics*, John Wiley & Sons, Inc., New York, N.Y., 1965.

[9] Kane, T. R., Levinson, D. A., *Dynamics: Theory and Applications*, McGraw -Hill Book Company, 1985.

[10] Hughes, P. C., "Multibody Dynamics for Space Station Manipulators: Dynamics of a Chain of Elastic Bodies," DYNACON Report SS-2, DYNACON Enterprises Ltd., Feb. 1985.

[11] Laskin, R. A., Likins, P. W., Longman, R. W., "Dynamical Equations of a Free-Free Beam Subject to Large Overall Motions," AAS/AIAA Astrodynamics Conference, Lake Tahoe, NV, August 3-5, 1981.

[12] Padilla, C.E., "Nonlinear Strain-Displacement Relations in the Dynamics of a Two-link Flexible Manipulator," M.S. Thesis, Dept. of Aeronautics and Astronautics, Massachusetts Institute of Technology, May 1989.

[13] Banerjee, A.K., Dickens, J.M., "Dynamics of an Arbitrary Flexible Body Undergoing Large Rotation and Translation with Small Vibration," AIAA Structures, Structural Dynamics and Materials Conference, Mobile, AL, April 3-5, 1989.

[14] Brady, M., Hollerbach, J. M., et al., *Robot Motion: Planning and Control*, MIT Press, Cambridge, MA, 1982, pp 55-60.

[15] von Flotow, A. H., Lecture Notes, MIT Aeronautics and Astronautics Dept., April 1988, pp. 850-858.

[16] Kane, T. R., Likins, P. W., Levinson, D. A., *Spacecraft Dynamics*, McGraw-Hill Book Company, 1983, pp. 318-326.

[17] Blevins, R.D., *Formulas for Natural Frequency and Mode Shape*, Robert E. Krieger Publishing Company, Malabar, Florida, 1979.

[18] Press, W.H., Flanner, B.P., Teukolsky, S.A., Vetterling, W.T., *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, Cambridge, London, 1986, pp. 550-559.

[19] Schmitz, E., "Dynamics and Control of a Planar Manipulator with Elastic Links," AAS 87-047, pp. 445-456.
[20] Hollerbach, J.M., "Dynamic Scaling of Manipulator Trajectories," Journal of Dynamic Systems, Measurement, and Control, Vol. 106, March 1984, pp. 102-106.
[21] Craig, J.J., *Introduction to Robotics Mechanics and Control*, Addison-Wesley Publishing Company, 1986, pp.191-219.

Nonlinear Finite Element Formulation for the Large Displacement Analysis in Multibody System Dynamics

J. Rismantab-Sany
B. Chang
A. A. Shabana
Department of Mechanical Engineering
University of Illinois at Chicago

Abstract

A total Lagrangian finite element formulation for the deformable bodies in multibody mechanical systems that undergo finite relative rotations is developed. The deformable bodies are discretized using finite element methods. The shape functions that are used to describe the displacement field are required to include the rigid body modes that describe only large translational displacements. This does not impose any limitations on the technique because most commonly used shape functions satisfy this requirement. The configuration of an element is defined using four sets of coordinate systems: Body, Element, Intermediate element, Global. The body coordinate system serves as a unique standard for the assembly of the elements forming the deformable body. The element coordinate system is rigidly attached to the element and therefore it translates and rotates with the element. The intermediate element coordinate system, whose axes are initially parallel to the element axes, has an origin which is rigidly attached to the origin of the body coordinate system and is used to conveniently describe the configuration of the element in undeformed state with respect to the body coordinate system. A mixed sets of Cartesian translational and rotational coordinates are used in order to define the location and orientation of the deformable body coordinate system with respect to global inertial frame of reference. The nonlinear dynamic equations of motion developed, for deformable multibody systems that undergo large relative displacements, are expressed in terms of a unique set of time-used. These invariants can be generated for each finite element prior to dynamic analysis. The invariants of the deformable body can be obtained by assembling the invariants of the elements using a standard finite element processor. The nonlinear formulation presented in this paper has been implemented in the general purpose computer program DAMS (Dynamic Analysis of Multibody Systems) that automatically constructs and numerically solves the nonlinear equations of motion of multibody systems consisting of interconnected rigid and deformable bodies. The linearization used in other finite element methods (such as the updated Lagrangian formulation) for describing the large displacements of deformable bodies is also discussed.

Optimum Control Forces for Multibody Systems with Intermittent Motion

S. K. Ider* and F. M. L. Amirouche†
Department of Mechanical Engineering
University of Illinois at Chicago

Abstract

It is common practice in the analysis of constrained multibody systems to apply the constraints as a set of separate algebraic equations and embed them into the governing equations for the specified time of the simulation. However more realistic systems are those where different constraints could be applied at different times, and hence the multibody system must be able to accommodate for the sudden changes. If the multibody system doesn't develop the appropriate initial conditions to satisfy the constraints whenever they are applied, the system performance will then be hampered and it will fail to accomplish its tasks.

If a multibody system is subjected to constraint equations that are released and applied at different times during the motion they characterize the so-called intermittent constraints. This paper objective is to address the continuity of motion when a dynamical system is suddenly subjected to constraint conditions. Motion discontinuity due to the initial constraint violation is avoided by prior control forces that adjust the motion and yield velocity and acceleration consistent at the point of application of the constraint. The optimum control forces are determined for a specified control interval. The method proposed provides an optimum adjustment of the system's motion and assures that the stresses developed at the system components are kept within acceptable limits. The procedures developed will be illustrated making use of inequality constraints applied to obstacle avoidance problems in robotics.

*PhD, Member ASME
†Assistant Professor, Member ASME, AIAA

Development of Efficient Computer Program for Dynamic Simulation of Telerobotic Manipulation

J. Chen & Y. J. Ou

Department of Mechanical Engineering
University of Maryland
College Park, MD 20742

Abstract

Research in robot control has generated interests in computationally efficient forms of dynamic equations for multi-body systems. For a simply connected open-loop linkage, dynamic equations arranged in recursive form has been found to be particularly efficient. A general computer program capable of simulating open-loop manipulator with arbitrary number of links has been developed based on an efficient recursive form of Kane's dynamic equations. Also included in the program is some of the important dynamics of the joint drive system, i.e., the rotational effect of the motor rotors. Further efficiency is achieved by the use of symbolic manipulation program to generate the Fortran simulation program tailored for a specific manipulator based on the parameter values given. This paper describes the formulations and the validation of the program, and it also shows some results.

Introduction

In the development of a robotic manipulator, simulation program can be an important design tool. It can be used to support detailed mechanical design by revealing the constraint forces and torques at different locations during certain maneuvers. It can also be applied to test different control laws without concerns of damaging the actual manipulators. If real time simulation can be developed, training of telerobot operators and testing of actual control hardware and software can become possible.

The success of a simulation in providing the useful and accurate information depends on the model fidelity, the formulation of equations of motion and the numerical solution of the equations. There is no such thing as "the" simulation of a dynamical system because the fidelity of the model determines what the results are like. There is always room for higher fidelity and so there is no end to it. But quite often, a modest increase of model fidelity is accompanied by a significant increase in equation complexity and numerical difficulty, and thus computation time. To achieve reasonable efficiency in the computation, one has to investigate the merits of different solution algorithms, different dynamical formulations and different levels of model fidelity. Additionally, one has to validate that the program is correctly representing the model.

Many researchers have worked on efficient formulations of dynamic equations for robot manipulators[2-9]. Most of them model robot as consisting of rigid bodies connected together with revolute or translational joints. Details of the joint drive systems have been mostly ignored. It is shown in [7] that joint drive systems have potentially significant effects on robot dynamics and hence should be included in the model. Also shown in [7] is a procedure to obtain the dynamical equations of a robot with a speed-reduction drive system from the equations of a direct drive robot. This procedure will be followed to develop a more comprehensive robot simulation program.

It has been known [2,5,6] that the important aspect of efficient formulations is the recursive development of kinematic and dynamic quantities to reduce the number of transformations among vector bases. It is also known that recursive Lagrange's formulation is still less efficient than the recursive Newton-Euler's formulations. However, Newton-Euler's formulation will not be advantageous if more complicated model of the system is analyzed. Since the program under development is anticipated to be expanded for more comprehensive modeling of manipulator systems, Kane's method is chosen because of its systematic features. An efficient formulation has been developed by applying recursive schemes in Kane's equations for a general manipulator system. The forward and backward recursions are established based on the bounds on the summation signs in the equations.

If properly developed, it is expected that a customized simulation program for a particular manipulator should be more efficient than a general purpose simulation program. For the development of simulation program, there is always a trade-off between generality and efficiency. But through the application of symbolic manipulation to eliminate unnecessary computations that occur for a particular model, it is possible to improve simultaneously the generality and the efficiency of a simulation program. Symbolic manipulation language MACSYMA has been used to develop a program called MSP (Manipulator Simulation Program) for manipulators that are made up of a single chain of any number of rigid bodies connected by revolute joints. Gear reduction effects of some simple joint drive systems are also efficiently incorporated in the program following the procedure in [7].

Independent formulation and programming of the system kinetic energy and the system angular momentum about a base-fixed point on the 1st joint axis are developed for validation purposes. Test cases which involve conservation of these quantities have been selected to validate the simulation programs. The objective of this paper is to present the formulation involved in the development of this program. Computation efficiency and significance of gear reduction effect are also to be discussed.

Mathematic Model

An open chain manipulator with N degrees of freedom as shown in Fig. 1 is analyzed for the development of MSP. Each link is driven with a motor and a gear reduction mechanism, an example of which is shown in Fig. 2. The base is considered fixed in the earth E (assumed to be an inertial reference frame). Couples are generated at motors through electromagnetic interactions, and gear reductions amplify the resulted moments on the links about the joints. It is assumed that the motor rotor and its rigidly attached part is the only massive element in a joint drive system that will contribute to the modifications of the equations of motion from that of a multibody direct drive system.

The links are labeled consecutively B_1 to B_N starting from the link connected to the base. The base is referred to as link B_0. The ideal revolute joints between links are numbered such that joint i connects link B_i to link B_{i-1}. An orthogonal unit vector basis $\underline{x}_i$, $\underline{y}_i$ and $\underline{z}_i$ fixed in B_i is defined in such a way that the unit vector $\underline{z}_i$ (i = 1,..., N) is directed along the axis of joint i. A particular configuration called the null configuration of a manipulator is one in which relative joint angles between links are all equal to zeros. The joint angles q_i (i = 1,..., N) are positive when right-handed rotation from the null configuration about $\underline{z}_i$ occurs. In this paper, the motor driving link B_i is assumed to be mounted on link B_{i-1} and unit vector $\underline{e}_i$ is defined to be parallel to the rotation axis of the motor rotor.

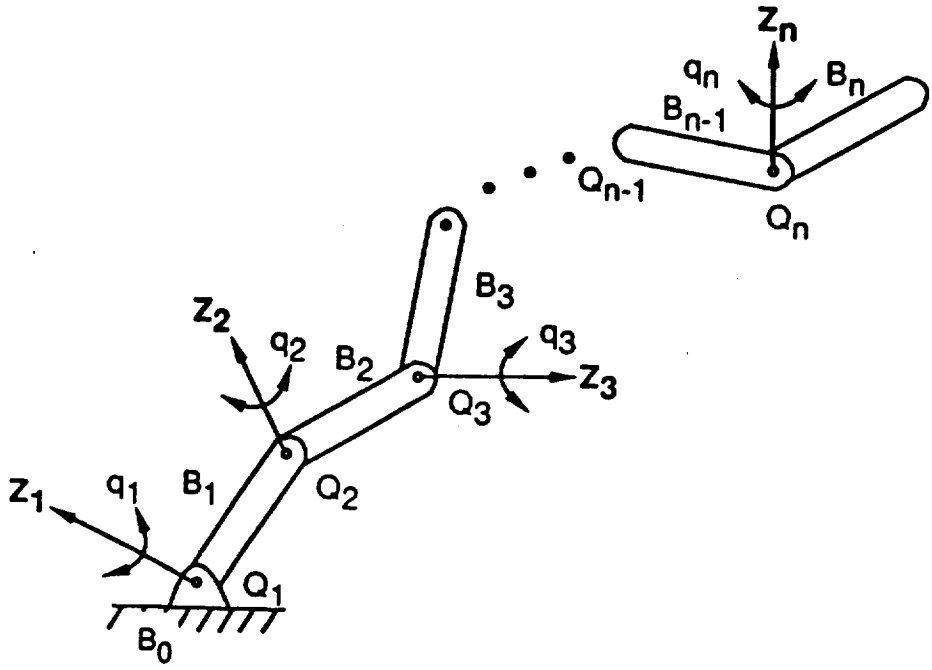


Fig. 1. Multi-Link System

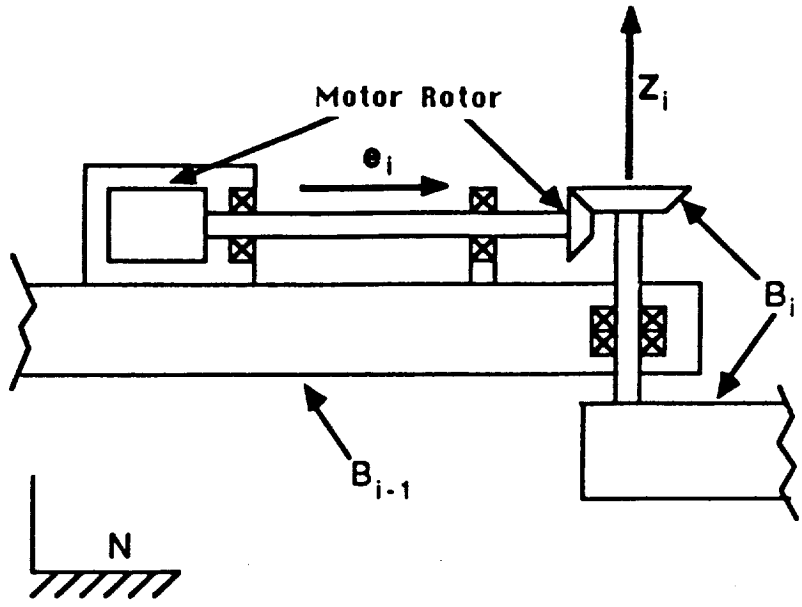


Fig. 2. Motor Rotor and Gear Reduction Mechanism

Formulation of Dynamical Equation

The following presentation of the formulations will be in terms of vectors and dyadics which are quantities independent of unit vector bases and can be represented by column and square matrices, respectively, when expressed in a particular basis.

Kane's dynamical equations[1] are

$$F_r + F_r^* = 0 \qquad (r = 1,\ldots,N) \quad (1)$$

where F_r and F_r^* are the generalized active and inertia forces associated with the r-th generalized speed, respectively. Since the system is holonomic, the number of generalized speeds are equal to the number of degrees of freedom. Here the generalized speeds are chosen to be simply the derivatives of the generalized coordinates. Assuming that the only contributing active forces are the motor torques $T_r\underline{e}_r$, the gravitational forces and the external load on the last link, represented by a force $\underline{F}^e$ acting through the mass center and a torque $\underline{T}^e$, one can write

$$F_r = \mu_r T_r + \sum_{i=1}^{N} M_i(\underline{V}_{i,r} \cdot \underline{G}) + \underline{V}_{N,r} \cdot \underline{F}^e + \underline{\omega}_{N,r} \cdot \underline{T}^e \qquad (r = 1,\ldots,N) \quad (2)$$

where μ_r is the gear ratio for the r-th joint, M_i is the mass of i-th link, $\underline{G}$ is the gravitational acceleration vector, and $\underline{V}_{i,r}$ and $\underline{\omega}_{i,r}$ are the r-th partial velocity of B_i^*, mass center of B_i, and the r-th partial angular velocity of B_i, in E, respectively. The generalized inertia forces are due to the inertia force and torque associated with each link, and they are

$$F_r^* = -\sum_{i=1}^{N} M_i(\underline{V}_{i,r} \cdot \underline{a}_i) - \sum_{i=1}^{N} \underline{\omega}_{i,r} \cdot (\hat{\underline{I}}_i \cdot \underline{\alpha}_i + \underline{\omega}_i \times \hat{\underline{I}}_i \cdot \underline{\omega}_i) \qquad (r = 1,\ldots,N) \quad (3)$$

where $\underline{a}_i$ and $\underline{\alpha}_i$ are the acceleration of B_i^* and the angular acceleration of B_i in E, respectively, and $\hat{\underline{I}}_i$ is the central inertia dyadic of B_i. Therefore, the dynamic equations become

$$\sum_{i=1}^{N} M_i(\underline{V}_{i,r} \cdot \underline{a}_i) + \sum_{i=1}^{N} \underline{\omega}_{i,r} \cdot (\hat{\underline{I}}_i \cdot \underline{\alpha}_i + \underline{\omega}_i \times \hat{\underline{I}}_i \cdot \underline{\omega}_i)$$

$$= \mu_r T_r + \sum_{i=1}^{N} M_i(\underline{G} \cdot \underline{V}_{i,r}) + \underline{V}_{N,r} \cdot \underline{F}^e + \underline{\omega}_{N,r} \cdot \underline{T}^e \qquad (r = 1,\ldots,N) \quad (4)$$

From Fig. 1, we can obtain the following kinematic equations.

$$\underline{\omega}_i = \sum_{j=1}^{i} \dot{q}_j \underline{z}_j \qquad (i = 1,\ldots,N) \quad (5)$$

$$\underline{V}_i = \sum_{j=1}^{i} \dot{q}_j(\underline{z}_j \times \underline{r}_{ji}^c) \qquad (i = 1,\ldots,N) \quad (6)$$

$$\underline{\omega}_{i,r} = \underline{z}_r \xi_r^i \qquad (i = 1,\ldots,N) \quad (7)$$

$$\underline{V}_{i,r} = (\underline{z}_r \times \underline{r}_{ri}^c) \xi_r^i \qquad (i = 1,\ldots,N) \quad (8)$$

where $\underline{r}_{ji}^c$ is the position vector from Q_j to B_i^* and

$$\xi_r^i = \begin{cases} 1 & \text{if } i \geq r \\ 0 & \text{if } r > i \end{cases} \qquad (i,r = 1,\ldots,N) \quad (9)$$

The angular and linear accelerations can be derived as

$$\underline{\alpha}_i = \sum_{j=1}^{i} \ddot{q}_j \underline{z}_j + \underline{\alpha}_i' \qquad (i = 1,\ldots,N) \quad (10)$$

$$\underline{a}_i = \sum_{j=1}^{i} \ddot{q}_j (\underline{z}_j \times \underline{r}_{ji}^c) + \underline{a}_i' \qquad (i = 1,\ldots,N) \quad (11)$$

where

$$\underline{\alpha}'_i = \sum_{j=1}^{i} \dot{q}_j(\underline{\omega}_j \times \underline{z}_j) \tag{12}$$

$$\underline{a}'_i = \sum_{j=1}^{i} \dot{q}_j \frac{{}^{E}d}{dt}(\underline{z}_j \times \underline{r}^c_{ji})$$

$$= \underline{\alpha}'_i \times \underline{r}^c_{ji} + \underline{\omega}_i \times(\underline{\omega}_i \times \underline{r}^c_{ji}) + \sum_{j=1}^{i-1} \dot{q}_j \underline{z}_j \times(\frac{{}^{E}d}{dt} \underline{r}^L_{ji}) \tag{13}$$

and $\underline{r}^L_{ji}$ is the position vector from Q_j to Q_i. A left superscript on a time differentiation symbol represents the reference frame in which the differentiation is to be performed[1]. With equations (5-13) substituted, the equations of motion can be rewritten as

$$\sum_{i=1}^{N} A'_{ri} \ddot{q}_i = \mu_r T_r - \hat{T}_r \qquad (r = 1,\ldots,N) \tag{14}$$

where

$$A'_{ri} = \underline{z}_r \cdot \underline{f}_{ri} \tag{15}$$

$$\underline{f}_{ri} = \sum_{j=i}^{N}[\hat{\underline{I}}_j \cdot \underline{z}_i + M_j \underline{r}^c_{rj} \times (\underline{z}_i \times \underline{r}^c_{ij})] \qquad (i \geq r) \tag{16}$$

$$\hat{T}_r = \underline{z}_r \cdot \hat{\underline{T}}_r \tag{17}$$

$$\hat{\underline{T}}_r = \{\sum_{i=r}^{N}[M_i(\underline{r}^c_{ri} \times \hat{\underline{a}}_i) + \underline{T}'_i]\} - \underline{T}^e - (\underline{r}^c_{rN} \times \underline{F}^e) \tag{18}$$

$$\hat{\underline{a}}_i = \underline{a}'_i - \underline{G} \tag{19}$$

$$\underline{T}'_i = \hat{\underline{I}}_i \cdot \underline{\alpha}'_i + \underline{\omega}_i \times \hat{\underline{I}}_i \cdot \underline{\omega}_i \tag{20}$$

Because of symmetry, only upper triangular terms of matrix $[A'_{ri}]$ need to be evaluated. The following formulations are used to evaluate the vector quantity $\underline{f}_{ri}$.

$$\underline{f}_{ii} = \underline{I}_i \cdot \underline{z}_i \qquad (i = 1,\ldots,N) \tag{21}$$

$$\underline{I}_i = \sum_{j=i}^{N}\{\hat{\underline{I}}_j + M_j[(\underline{r}^c_{ij})^2 \underline{U} - \underline{r}^c_{ij}\,\underline{r}^c_{ij}]\}$$

$$= \underline{I}_{i+1} + \underline{K}_i + 2(\underline{r}^L_{i(i+1)} \cdot \underline{r}^*_{i+1})\underline{U} - \underline{r}^L_{i(i+1)}\underline{r}^*_{i+1} - \underline{r}^*_{i+1}\underline{r}^L_{i(i+1)}$$

$$(i = N-1,\ldots,1) \tag{22}$$

$$\underline{K}_i = \hat{\underline{I}}_i + M_i[(\underline{r}^c_{ii})^2 \underline{U} - \underline{r}^c_{ii}\,\underline{r}^c_{ii}] + (\sum_{j=i+1}^{N} M_j)[(\underline{r}^L_{i(i+1)})^2 \underline{U} - \underline{r}^L_{i(i+1)}\underline{r}^L_{i(i+1)}]$$

$$(i = 1,\ldots,N) \tag{23}$$

$$\underline{r}^*_i = \sum_{j=i}^{N} M_j \underline{r}^c_{ij}$$

$$= \underline{r}^*_{i+1} + M_i \underline{r}^c_{ii} + (\sum_{j=i+1}^{N} M_j) \underline{r}^L_{i(i+1)} \qquad (i = N-1,\ldots,2) \tag{24}$$

The dyadic quantity $\underline{K}_i$ is a constant in B_i, i.e, if $\underline{K}_i$ is expressed in terms of $\underline{x}_i$, $\underline{y}_i$, $\underline{z}_i$ basis, the coefficients are constants. Equations (22) and (24) are backward recursive formulas that can be evaluated by establishing the following:

$$\underline{I}_N = \hat{\underline{I}}_N + M_N[(\underline{r}^c_{NN})^2 \underline{U} - \underline{r}^c_{NN}\,\underline{r}^c_{NN}]$$

$$= \underline{I}^{B_N/Q_N} \tag{25}$$

$$\underline{r}^*_N = M_N \underline{r}^c_{NN} \tag{26}$$

where $\underline{I}^{B_N/Q_N}$ is the inertia dyadic of B_N relative to Q_N. For the off-diagonal terms of $[A_{ri}]$, a backward recursive formula involving r, i.e.,

$$\underline{f}_{ri} = \underline{f}_{(r+1)i} + \underline{r}^{L}_{r(r+1)} \times (\underline{z}_i \times \underline{r}^{*}_i) \qquad \begin{pmatrix} i = N, \ldots, 2 \\ r = i-1, \ldots, 1 \end{pmatrix} \qquad (27)$$

can be used with $\underline{f}_{ii}$ being the starting vector that should have been evaluated from equation (21).

For $\hat{\underline{T}}_r$, the following forward recursive formulations can be used to evaluate the necessary quantities.

$$\underline{\omega}_i = \underline{\omega}_{i-1} + \dot{q}_i \underline{z}_i \qquad (i = 1, \ldots, N) \qquad (28)$$

$$\underline{\alpha}'_i = \underline{\alpha}'_{i-1} + \dot{q}_i (\underline{\omega}_i \times \underline{z}_i) \qquad (i = 1, \ldots, N) \qquad (29)$$

$$\underline{u}_i = \hat{\underline{a}}_i - \underline{\underline{A}}_i \cdot \underline{r}^{c}_{ii} = \underline{u}_{i-1} + \underline{\underline{A}}_{i-1} \cdot \underline{r}^{L}_{(i-1)i} \qquad (i = 1, \ldots, N) \qquad (30)$$

where

$$\underline{\underline{A}}_i = \underline{\alpha}'_i \times \underline{\underline{U}} + (\underline{\omega}_i \times \underline{\underline{U}}) \cdot (\underline{\omega}_i \times \underline{\underline{U}}) \qquad (i = 1, \ldots, N) \qquad (31)$$

The starting values for equations (28-30) are

$$\underline{\omega}_0 = 0 \qquad (32)$$

$$\underline{\alpha}'_0 = 0 \qquad (33)$$

$$\underline{u}_0 = -\underline{G} \qquad (34)$$

The introduction of dyadic quantity $\underline{\underline{A}}_i$ is to reduce the overall computation by reusing it in two equations in the remaining formulations. The dyadic obtained by cross multiplying a vector with a unit dyadic can be represented by a skewsymmetric square matrix when it is expressed in a particular unit vector basis. This skew symmetric square matrix is commonly encountered when a cross multiplication of column matrices is replaced with a matrix multiplication.

The following backward recursive formulation can be used to evaluate the kinetic quantities $\hat{\underline{T}}_i$:

$$\underline{p}_i = \underline{p}_{i+1} + \underline{\underline{A}}_i \cdot [(\sum_{j=i+1}^{N} M_j)\, \underline{r}^{L}_{i(i+1)} + M_i\, \underline{r}^{c}_{ii}] \qquad (i = N-1, \ldots, 1) \qquad (36)$$

$$\hat{\underline{T}}_i = \hat{\underline{T}}_{i+1} + \underline{r}^{L}_{i(i+1)} \times \underline{p}_{i+1} + \underline{L}_i + [\, M_i\, \underline{r}^{c}_{ii} + (\sum_{j=i+1}^{N} M_j)\, \underline{r}^{L}_{i(i+1)}] \times \underline{u}_i \qquad (i = N-1, \ldots, 1) \qquad (37)$$

where

$$\underline{L}_i = \{\, \underline{\underline{A}}_i \cdot [\underline{\underline{K}}_i - \frac{1}{2} (\underline{\underline{K}}_i : \underline{\underline{U}})\, \underline{\underline{U}}]\}_v \qquad (i = 1, \ldots, N) \qquad (38)$$

and a subscript v next to a dyadic in equation (38) denotes the vector of the dyadic, which is formed by summing the cross products of the prefactors and the postfactors of all the dyads in the dyadic. The reason for using the expression in equation (38) is to reduce computation counts. In fact, $\underline{L}_i$ can be expressed in a form identical to equation (20). Conversely, equation (20) can be replaced by

$$\underline{T}'_i = \{\, \underline{\underline{A}}_i \cdot [\hat{\underline{\underline{I}}}_i - \frac{1}{2} (\hat{\underline{\underline{I}}}_i : \underline{\underline{U}})\, \underline{\underline{U}}]\}_v \qquad (i = 1, \ldots, N) \qquad (39)$$

where the expression in brackets [ ] is the dyadic whose representation in a particular unit vector basis is the inertia matrix with half its trace subtracted from each diagonal element. The dyadic operations used above follow the convention introduced by Gibbs[10] in late eighteen hundreds.

The starting values for the recursive equations, equations (36) and (37), are

$$\underline{p}_N = M_N\, \underline{\underline{A}}_N \cdot \underline{r}^{c}_{NN} \qquad (40)$$

$$\hat{\underline{T}}_N = M_N\, \underline{r}^{c}_{NN} \times (\underline{u}_N - \underline{F}^{e}/M_N) + \underline{L}_N - \underline{T}^{e} \qquad (41)$$

Equations (14) represent the equations of motion of a direct drive open-chain system. The modifications to equations (14) for an open-chain system with motors and gear reduction mechanisms shown in Fig. 2 are based on the difference between generalized inertia forces contributing to these two systems. They are[7]

$$(F_r^*)_s = (F_r^*)_{s'} + G_r^* \qquad (r = 1,\ldots,N) \qquad (42)$$

where subscript s represents a manipulator system S which has a motor and gear reduction mechanism in each link similar to that shown in Fig. 2, while subscript s' represents the manipulator system S' which has the same mass and inertia distribution as system S, but the motor rotors and gear reduction mechanisms are considered to be fixed in the links on which they are mounted. Here and throughout this paper, the motor driving i-th joint is assumed to be mounted on link B_{i-1} (i=1,...,N). System S', therefore, represents a direct-drive system. The difference terms between these two systems are

$$G_r^* = \begin{cases} 0 & (r > i) \\ -\mu_r J_r(\mu_r \ddot{q}_r + \underline{\alpha}_{r-1}\cdot \underline{e}_r) & (r = i) \\ \sum_{i=r+1}^{N} -\mu_i J_i[(\underline{z}_r \cdot \underline{e}_i)\ddot{q}_i - \dot{q}_i(\underline{\omega}_{i-1}\times \underline{z}_r)\cdot \underline{e}_i] & (r < i) \end{cases} \qquad (43)$$

With the additional terms added, the dynamic equations for S become

$$\sum_{i=1}^{N} A_{ri}\,\ddot{q}_i = \mu_r T_r - \hat{T}_r + G_r \qquad (r = 1,\ldots,N) \qquad (44)$$

where

$$A_{ri} = A'_{ri} + \begin{cases} \mu_i J_i(\underline{z}_r\cdot\underline{e}_i) & (\text{if } r \neq i) \\ \mu_i^2 J_i & (\text{if } r = i) \end{cases} \qquad (45)$$

$$G_r = -\mu_r J_r\,\underline{\alpha}'_{r-1}\cdot\underline{e}_r + \sum_{i=r+1}^{N} \mu_i J_i \dot{q}_i(\underline{\omega}_{i-1}\times \underline{z}_r)\cdot\underline{e}_i \qquad (46)$$

It can be noticed that matrix $[A_{ri}]$ is still symmetric. Hence, only those difference terms in the upper triangle of matrix $[A_{ri}]$ need to be evaluated.

Symbolic Manipulation

Direct numerical approach in evaluating the above equations can be inefficient if there are terms involving multiplication with 0 or 1, or addition with 0. The use of multi-dimensional arrays in a general purpose program further reduces the computational efficiency. These are some of the reasons why a general simulation program cannot achieve the highest possible efficiency. Symbolic language such as MACSYMA can be applied to eliminate these inefficiencies. Theoretically, one can use MACSYMA to derive equations explicitly in terms of all joint angles and their derivatives and then to reorganize the equations for efficient computation. But for a manipulator with high number of degrees of freedom, this requires enormous memory space and CPU time, and cannot be optimally simplified because the simplification is limited by the capability of MACSYMA. It has been found that the recursive formulation as presented above is particularly advantageous because computation is already optimized. Only simple symbolic operations need to be applied to generate the recursive equations of motion in FORTRAN coding, and hence the computer time required for this process is not excessively long.

Program Validation

Complete validation of a simulation program is next to impossible. But without being subject to some forms of validation, a program cannot be trusted. When a program is applied on a reasonably complicated system, one cannot rely on simple statements like "the results make sense" as validation.

For a manipulator system with 3 or more links, intuition as to its motion when subject to certain inputs does not work well at all. A more systematic approach need to be adopted. This is a very important subject for the dynamic simulation of robots, but it does not seem to attract much attention. The approach taken by the authors was to select some conditions under which the system will have some scalar quantities, such as energy or measure numbers of an angular momentum vector, that are conserved throughout the motion. Since there exists a slight possibility that chance may cause errors not to be detected, an independently developed program is used to evaluate the validation quantities. Formulation of the system kinetic energy, potential energy and the system angular momentum about a base-fixed point on the first joint axis for validation purposes will be described next followed by the description of test cases chosen.

The kinetic energy formulation for a direct-drive manipulator S' is

$$K' = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} \dot{q}_i \dot{q}_j \tag{47}$$

For a manipulator with gear reduction in its drive system, slight modification is required. Consider two manipulator system S and S' as described before. If they have the same motion, then the kinetic energies K and K' of systems S and S', respectively, are related by[7]

$$K = K' + \sum_{i=1}^{N} (\frac{1}{2} \mu_i^2 J_i \dot{q}_i^2 + \mu_i J_i \dot{q}_i \, \underline{\omega}_{i-1} \cdot \underline{e}_i) \tag{48}$$

The potential energy V of the system are due to gravity only, and it is

$$V = -\sum_{i=1}^{N} M_i \, \underline{G} \cdot \underline{r}_{1i}^{c} \tag{49}$$

There is no difference between the potential energy expressions for S and S' because they have the same mass distribution. The angular momentum vector $\underline{H}'$ of a direct-drive manipulator S' about Q_1 is

$$\underline{H}' = \sum_{j=1}^{N} \sum_{B_j} \underline{r}_1^P \times (m_P \, {}^E\underline{V}^P) \tag{50}$$

where P is a generic particle in B_j, $\sum_{B_j}$ represent the summing over all the particles in B_j, $\underline{r}_1^P$ is the position vector from Q_1 to P, and m_P and ${}^E\underline{V}^P$ are, respectively, the mass and the velocity in E of P. Only the measure number of $\underline{H}'$ in $\underline{z}_1$ direction is used in the validation process. Comparing the kinetic energy formulation with that of $\underline{H}'$, one can obtain

$$\underline{H}' \cdot \underline{z}_1 = \sum_{i=1}^{N} A_{1i} \dot{q}_i \tag{51}$$

where A_{1i} can be found in Eq. (15) with r = 1. The angular momenta $\underline{H}$ and $\underline{H}'$ about Q_1 of systems S and S', respectively, are related by

$$\underline{H} = \underline{H}' + \sum_{i=1}^{N} \mu_i J_i \dot{q}_i \underline{e}_i \tag{52}$$

Hence, equations (48-52) provide the necessary formulas for the evaluation of the conservation quantities.

The validation cases used are

I. Conservation of total energy, K+V :

$T_r = 0 \quad (r = 1, \ldots, N)$

II. Conservation of $\underline{H} \cdot \underline{z}_1$:

a. $g = 0$, $T_1 = 0$, $\mu_1 = 1$, $\underline{e}_1 = \underline{z}_1$

b. $g \neq 0$, $T_1 = 0$, $\mu_1 = 1$, $\underline{e}_1 = \underline{z}_1 = \pm \underline{G}/g$

where g is the gravitational constant.

Under conditions described above, many simulation runs of manipulators have been performed to validate the program. The results of these runs show that numerical variations of the conservation quantities are of the order of magnitude that is appropriately correlated to the absolute integration error tolerances. Here integration subroutines using Adams-Bashforth-Moulton predictor-corrector scheme and using Runge-Kutta-Verner method have been separately applied for numerical integration.

Discussions

For a general manipulator, the total numbers of operations to obtain A_{ri} and $\mu_r T_r - \hat{T}_r$ $(r,i = 1,\ldots,N)$ in Eq. (14) for a direct drive manipulator are $11N(N-1)+150(N-1)-15$ multiplications and $7N(N-1)+119(N-1)-14$ additions. Table 1 lists the numbers of operations required for each of the equations in the evaluation of matrix $[A_{ri}]$ and $\hat{T}_r$. The counting of operations follows that presented in [6]. Notice that unit vector basis transformation has to be performed in each recursion step. Here external force and torque applied on the last link are not included. Therefore, for a general 6 link manipulator, 1075 multiplications and 791 additions are required. This is much lower than the 1541 multiplications and 1196 additions needed in Method 3 of [2]. For the six dof PUMA 600 presented in [9], Our MSP program generates FORTRAN code that requires 351 multiplications and 281 additions to perform the computation that takes 392 multiplications and 294 additions in [8].

Table 1. Number of Operations

equations	multiplications	additions
(22)	57(N-1)-33	36(N-1)-25
(24)	8(N-1)	7(N-1)-3
(27)	11N(N-1)-8(N-1)	7N(N-1)-4(N-1)
(28)	8(N-1)	5(N-1)
(29)	10(N-1)	6(N-1)
(30)	17(N-1)	13(N-1)
(31)	6(N-1)	9(N-1)+1
(36)	17(N-1)	13(N-1)
(37)	20(N-1)	19(N-1)
(38)	15(N-1)+3	15(N-1)+1
(40)	9	6
(41)	6	5
Total	11N(N-1)+150(N-1)-15	7N(N-1)+119(N-1)-14

Adding $\ddot{q}_i \underline{z}_i$ to the right hand side of equation (29), equations (28-41) together with equation (17) become inverse dynamic formulation for a direct drive robot. This inverse dynamic evaluation is similar to algorithm 3 in [6]. By applying MACSYMA to the formula, some unnecessary computations can be removed. For instance, if N is 6, the number of computation for the manipulator with twist angles equal to 0° or 90° is 340 multiplications and 290 additions compared to 388 multiplications and 370 additions in [6]. For the simpler manipulator with $\underline{r}^c_{ii}$ and $\underline{r}^l_{i(i+1)}$ having only one nonzero element, the numbers of computation are 245 multiplications and 204 additions compared to 277 multiplications and 255 additions in [6]. The reductions are due to some additional multiplications and additions with zero quantities that are counted in [6] because the authors of [6] did not actually expand the equations for the counting.

In order to give additional indication of efficiency, the 7 link Robot Research Corporation [11] manipulator shown in Fig. 3 with parameters listed

in the Appendix is considered. The numbers of operations for the system in the form of Eq. (14) are 651 multiplications and 505 additions with effects of motor rotors included. The numbers for direct-drive system are 548 multiplications and 439 additions. Therefore, adding the effects of motor rotors requires 103 multiplications and 66 additions, which is about 17% of the total computation needed for the direct-drive system.

In the interest of demonstrating the effect of motor rotors, a constant motor torque (T_1= 0.625 N-m) is applied on the first joint of the 7 link manipulator shown in Fig. 3. Two sets of equations are solved for comparison. Set 1 is equation (14), which represents the direct drive system S' and set 2 is equation (44), which is the complete equations of system S. The results from set 1 are shown in Figs. 4 and 5 while those from set 2 are in Figs. 6 and 7. The differences of the results are so substantial that it is clear that set 1 is not representative of the actual system.

Among all the additional terms due to motor rotor, the terms $\mu_i^2 J_i \ddot{q}_i$ (i=1, . . . , N) are most significant due to the large values of μ_i. Another set of equations, set 3, established by adding only $\mu_i^2 J_i$ to diagonal elements A_{ii} of $[A'_{ri}]$ matrix in equations (14), is also solved for comparison. A sinusoidal motor torque (T_1= 3.125 cos 0.8πt kg-m) is applied on the first joint of the manipulator. Some of the results from set 2 are shown in Fig. 8 while the differences of the results between set 2 and set 3 are shown in Fig. 9. It is clear that the differences are relatively small in the duration of 5 seconds.

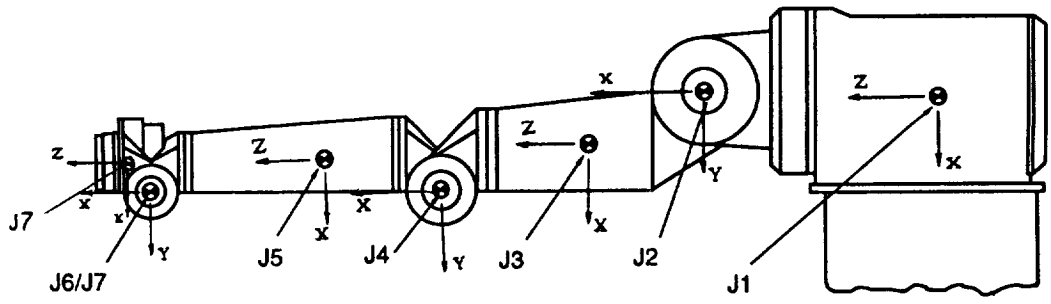


Fig. 3. A 7 Link Manipulator

Conclusions

The initial development of an efficient simulation program for telerobotic manipulators is described. An efficient recursive formulation of Kane's dynamic equations for a class of manipulators which are made up of links connected with revolute joints and driven by motors with reduction mechanisms is presented. The recursions are established according to the summation bounds in the equations. Comparison of operation counts with other published formalisms shows advantages of the present approach. Furthermore, effects of rotor inertia and speed reduction are included in the formulation to yield a more faithful model of the actual system. Symbolic manipulation is also applied to generate customized simulation program for additional improvement of the computational efficiency. Aside from the discussions on efficiency, steps taken to validate the simulation program are presented. Finally, simulation results show that effects of rotors in drive system with high speed reduction cannot be ignored in the simulation of a manipulator.

Acknowledgement

The funding support from NASA Goddard Space Flight Center (Grant NAG5-1019) for this work is greatly appreciated. The authors also like to thank Mark Routson for his help in developing the program for validation and Win-bin Shieh for testing the simulation program MSP.

Reference

[1] Kane, T. R. and D. A. Levinson , Dynamics: Theory and applications, McGraw-Hill Book Company, 1985.
[2] Walker, M. W. and D. E. Orin, "Efficient Dynamic Computer Simulation of Robotic Mechanisms", Journal of Dynamic Systems, Measurement, and Control, vol. 104, Sep. 1982.
[3] Rosenthal, D. E. and M. A. Sherman, "Symbolic Multibody Equations via Kane's Method(SD/EXACT)" AAS/AIAA Astrodynamics Specialist Conference, New York, Aug. 22-25, 1983
[4] Kane, T. and D. Levinson, "The Use of Kane's Dynamical Equations in Robotics", the Int. J. of Robotics Research Vol. 2, No. 3, 1983.
[5] Hollerbach, J. M., "A recursive Largrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity", IEEE, Vol. SMC-10, No. 11, Nov. 1980.
[6] Balafoutis, C. A., R. V. Patel and P. Misra, "Efficient Modeling and Computation of Manipulator Dynamics Using Orthogonal Cartesian Tensors", IEEE J. of Robotics and Automation, Vol. 4, NO. 6. Dec. 1988.
[7] Chen, J., "The Effects of Gear Reduction on Robot Dynamics". Submitted to ASME Journal of Dynamic System, Measurement and Control.
[8] Neuman C. P. and J. J. Murray, "Customized Computational Robot Dynamics", Journal of Robotic Systems, Vol. 4, No. 4, 1987, pp. 503-526.
[9] Paul, R. P., M. Rong and H. Zhang, "The Dynamics of the PUMA Manipulator", in Proceedings of 1983 American Control Conference, San Francisco, California.
[10]Gibbs, J. W. and E. B. Wilson, Vector Analysis, Yale University Press, New Haven, 1931 (seventh printing).
[11]Karlen, J. P., J. M. Thompson and J. D. Farrell, "Design and Control of Modular, Kinematically-Redundant Manipulators", Second AIAA/NASA/USAF Symposium on Automation, Robotics and Advanced Computing for the National Space Program, March 9-11, 1987.

Fig. 4. Solution of Equations Set 1
(joints 1–3)

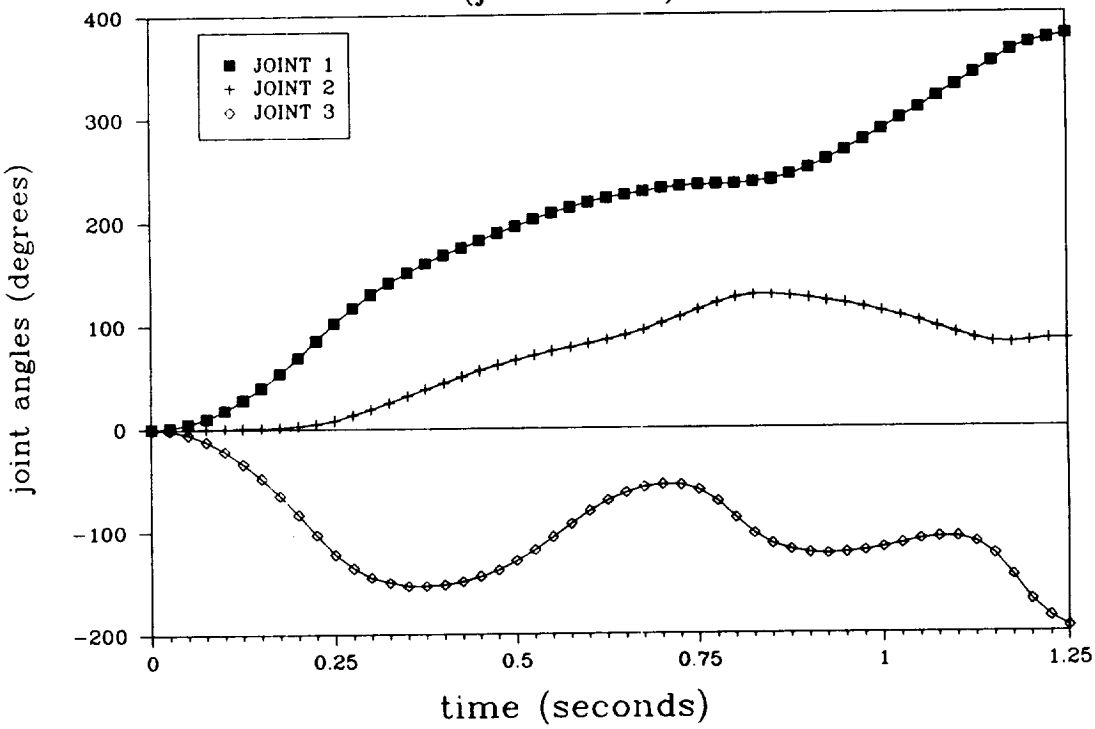


Fig. 5. Solution of Equations Set 1
(joints 4–7)

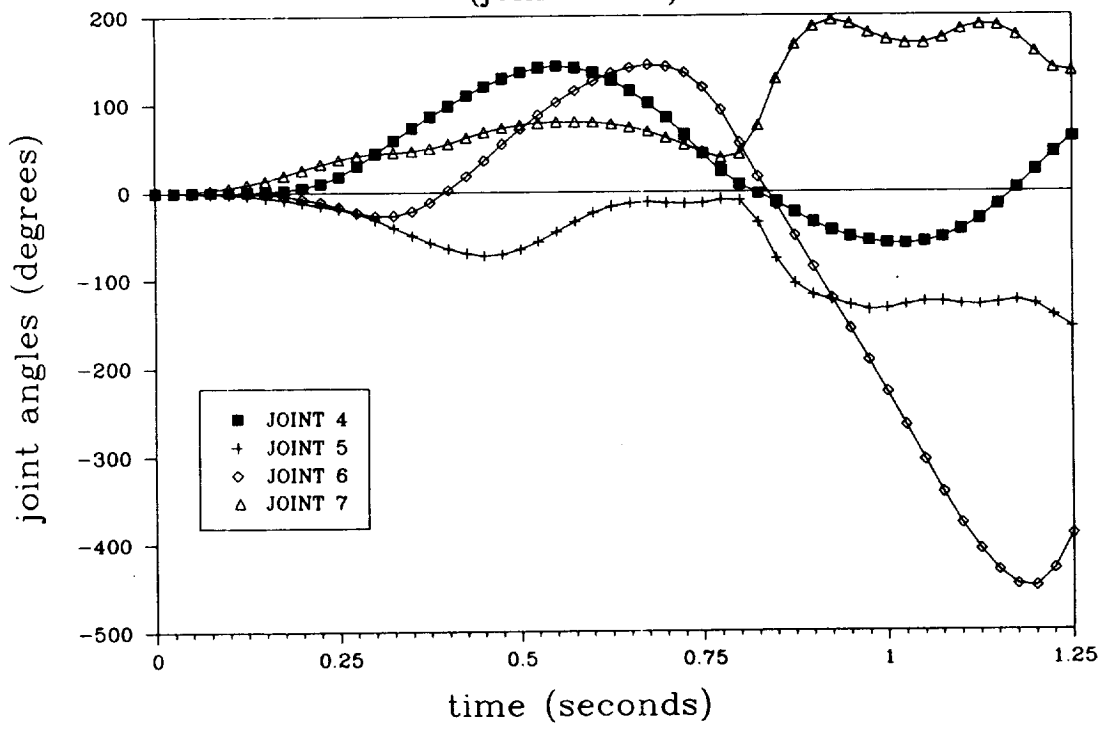


Fig. 6. Solution of Equations Set 2
(joints 1–3)

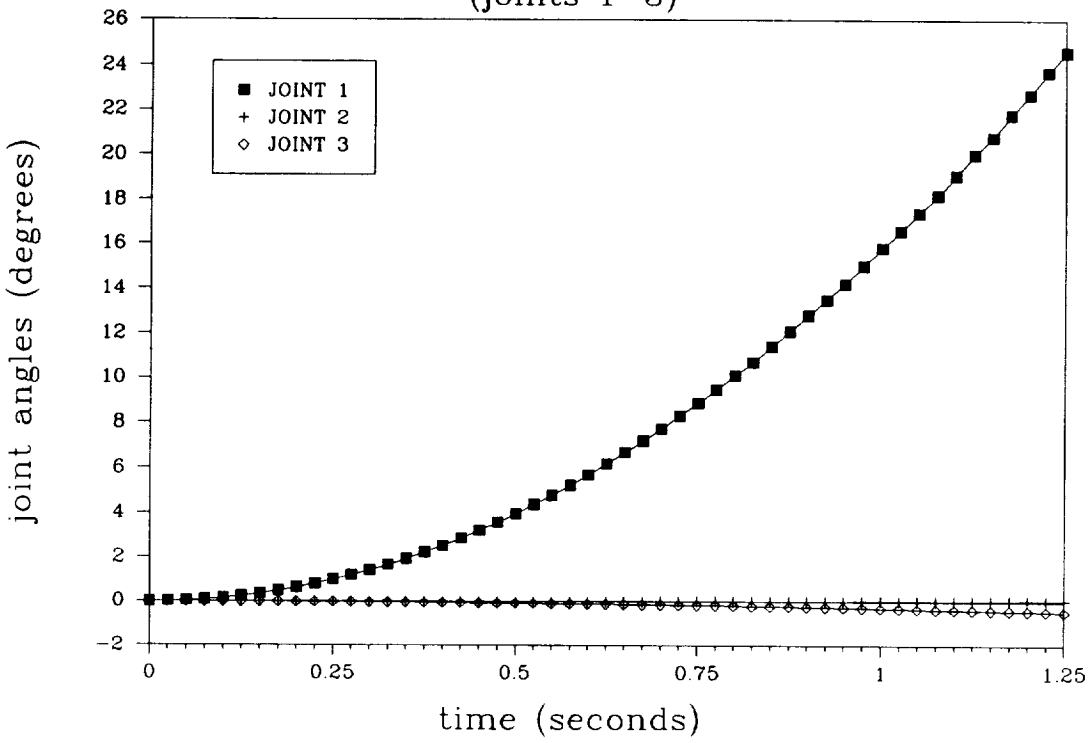


Fig. 7. Solution of Equations Set 2
(joints 4–7)

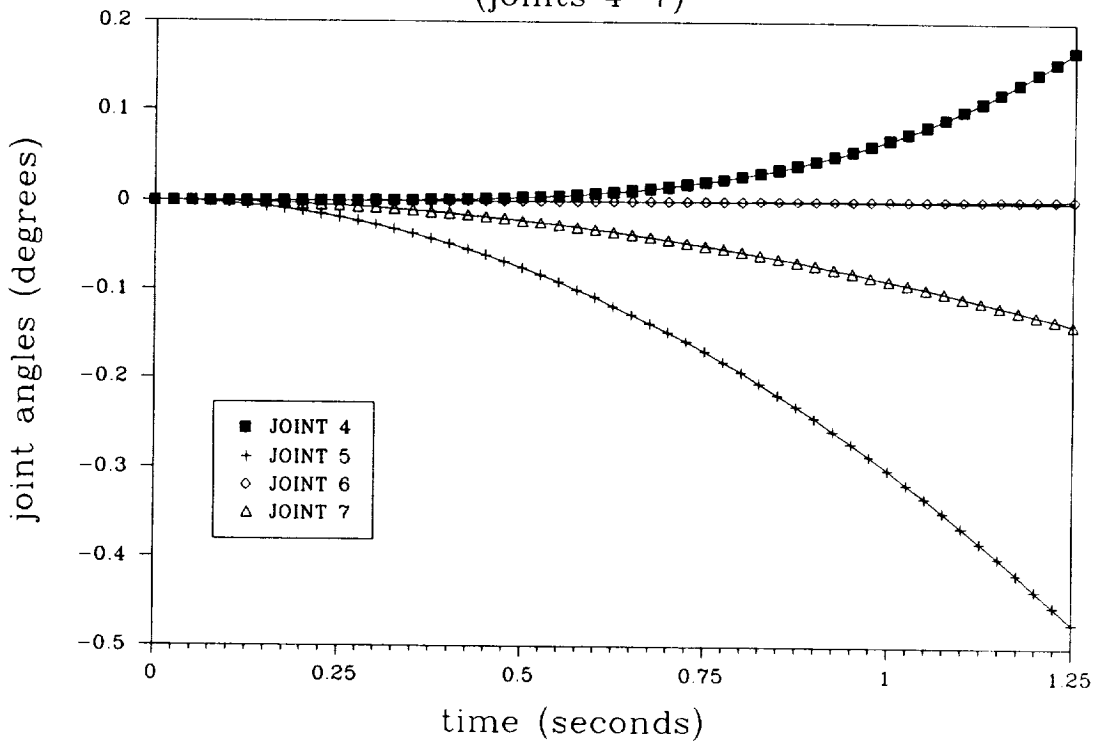


Fig. 8. Solution of Equations Set 2
(joints 1,5,6,7)

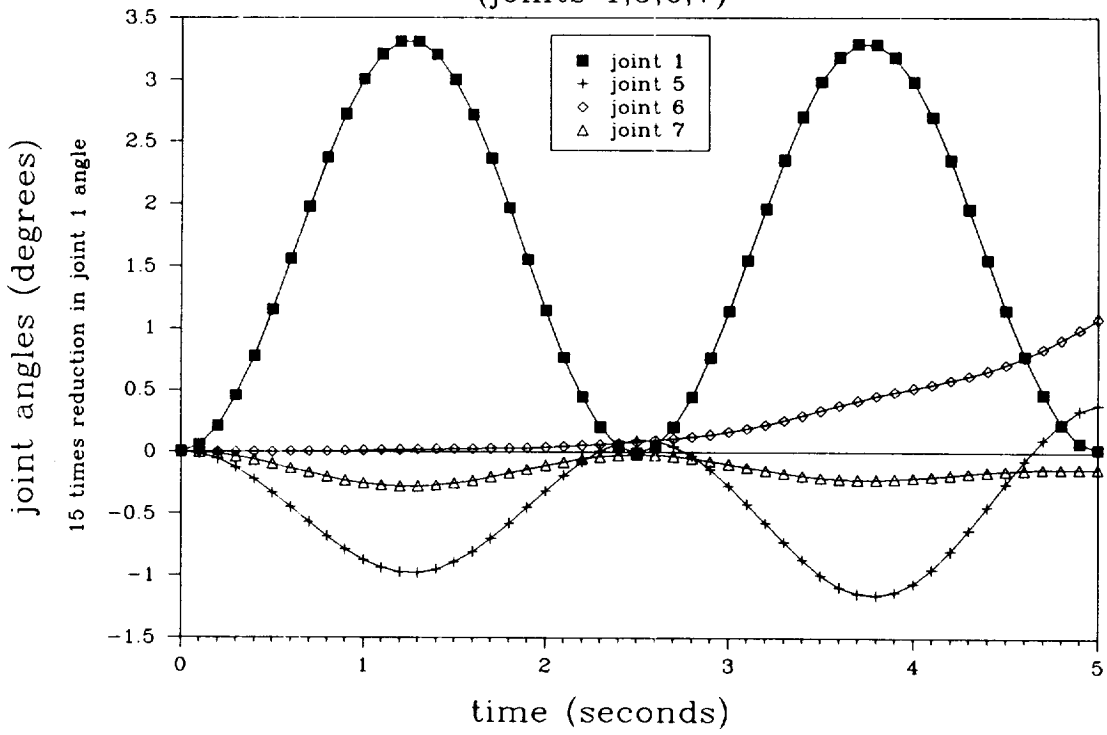
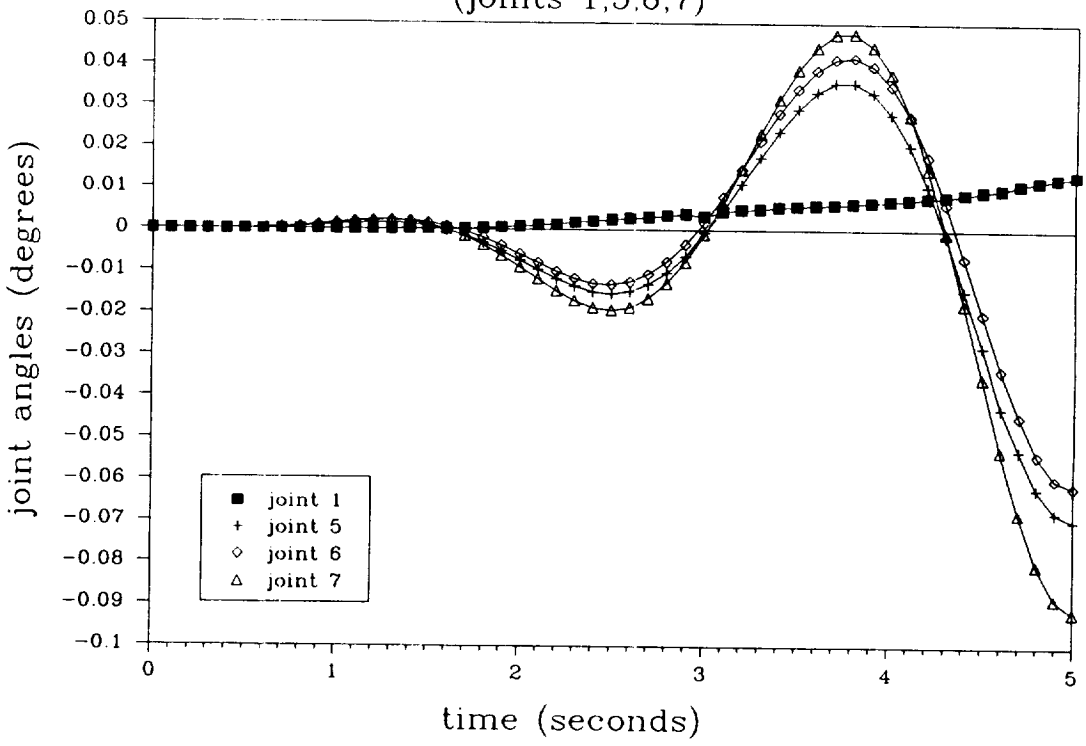


Fig. 9. Difference of Solutions of Set 2 & 3
(joints 1,5,6,7)



Appendix

The parameters of a 7 link Robot Research Corporation manipulator used in the simulation from which results are presented in this paper are in the following.

Mass (including motor and speed reduction mechanism): (kg)

link	1	2	3	4	5	6	7
mass	104.2	55.4	28.8	20.4	11.1	4.7	4.7

Moment of inertia (including motor and speed reduction mechanism): ($kg\text{-}m^2$)

	I_{xx}	I_{yy}	I_{zz}
link 1	2.05	2.05	0.7
link 2	0.4	0.4	0.3
link 3	0.6	0.6	0.2
link 4	0.2	0.2	0.2
link 5	0.1	0.1	0.04
link 6	0.03	0.03	0.03
link 7	0.03	0.03	0.03

Speed reduction ratio, rotor inertia (including attachment) and rotor axis (in link fixed unit vector basis):

	speed ratio	rotor inertia ($kg\text{-}m^2$)	rotor axis
base	160	0,003	[0, 0, 1]
link 1	160	0.003	[1, 0, 1]
link 2	200	0.0003	[0, 1, 0]
link 3	200	0.0003	[1, 0, 0]
link 4	200	0.0002	[0, 1, 0]
link 5	200	0.0002	[1, 0, 0]
link 6	160	0.0002	[0, 1, 0]

Joint to mass center position vector (m) r^c_{ii}, i=1,..., 7 :

	X	Y	Z
link 1	0	0	0
link 2	0	0	0
link 3	0	0	0
link 4	0	0	0
link 5	0	0	0
link 6	0	0	0
link 7	0	0	0

Joint to joint position vector (m) $r^L_{i(i+1)}$, i=1,..., 6 :

	X	Y	Z
link 1	0.0	0.0	0.5
link 2	0.3	0.1	0.0
link 3	0.1	0.0	0.3
link 4	0.3	-0.1	0.0
link 5	0.08	0.0	0.4
link 6	0.19	-0.08	0.0

The Coupling Effects of Kinematics and Flexibility on the Lagrangian Dynamic Formulation of Open Chain Deformable Links

Koorosh Changizi
Department of Mechanical Engineering
University of Illinois at Chicago

Abstract

In this paper a nonlinear Lagrangian formulation for the spatial kinematic and dynamic analysis of open chain deformable links consisting of cylindrical joints that connects pair of flexible links is developed. The special cases of revolute or prismatic joint can also be obtained form the kinematic equations. The kinematic equations are described using 4x4 matrix method. The configuration of each deformable link in the open loop kinematic chain is identified using a coupled set of relative joint variables, costant geometric parameters, and elastic coordinates. The elastic coordinates define the link deformation with respect to a selected joint coordinate system that is consistent with the kinematic constraints on the boundary of the deformable link. These coordinates can be introduced using approximation techniques such as Rayleigh-Ritz method, finite element technique or any other desired approach. The large relative motion between neighboring links are defined by a set of joint coordinates which describes the large relative translational and rotational motion between two neighboring joint coordinate systems. The origin of these coordinate systems are rigidly attached to the neighboring links at the joint definition points along the axis of motion. The geometry of the deformable links are included in the formulation by two costant parameters which accounts for the length and twist of the deformable link in tis undeformed state. The kinematic equations that define the global position and velocity of an arbitrary point on a deformable link is developed in terms of the relative joint variable, constant geometric parameters, and elastic coordinates of deformable links. These kinematic equations are then used to develop the energy expression of the deformable link. The nonlinear terms that represent the dynamic coupling between the large relative motion and the mall elastic deformations is identified and presented in terms of a set of time-invariant quantities that depend on the assumed displacement field and provide a systematic approach to study the spatial dynamics of open loop kinematic chains. The system differential equations are then developed and expressed in terms of these set of invariant quantities using Lagrange's equation of motion.

Explicit Modeling of Composite Plates and Beams in the Dynamics of Multibody Systems

F. M. L. Amirouche* , S. K. Ider† , and M. Moumene†
Department of Mechanical Engineering
University of Illinois at Chicago

Abstract

The state of the art dynamic response analysis of flexible multibody systems is currently restricted to elastic bodies with homogeneous materials. The requirements for high speed operation has made it necessary to use lightweight multi layered composite bodies in robotic systems and space structure applications. Dynamic modeling and analysis of such systems are particularly important since the effects of body flexibility to the performance are likely to be more pronounced.

In this paper first the eight-noded isoperimetric quadrilateral element with independent rotational and displacement degrees of freedom is extended to laminated composite elements. The element includes an arbitrary number of bonded layers, each of which may have a different thickness. The transverse shear deformation which is a predominant factor in the analysis of laminated composite structures is taken into account in developing the stiffness and mass matrices. The corresponding 3-D mode shapes are then incorporated to the multibody system dynamical equations. Floating body reference frames allow the selection of different boundary conditions, and the dynamical equations contain all the nonlinear interactions between the rigid and elastic motion. Example simulations are presented to illustrate the methods proposed.

*Assistant Professor, Member ASME, AIAA
†PhD, Member ASME

EXPERIMENTAL VERIFICATION OF DYNAMIC SIMULATION

K. Harold Yae, Howyoung Hwang, and Su-Tai Chern
Center for Simulation and Design Optimization
Department of Mechanical Engineering
The University of Iowa
Iowa City, Iowa 52242
(319) 335-5683

ABSTRACT

The dynamics model here is a backhoe, which is a four degree of freedom manipulator from the dynamics standpoint. Two types of experiment are chosen that can also be simulated by a multibody dynamics simulation program. In the experiment, recorded were the configuration and force histories; that is, velocity and position, and force output and differential pressure change from the hydraulic cylinder, in the time domain.

When the experimental force history is used as driving force in the simulation model, the forward dynamics simulation produces a corresponding configuration history. Then, the experimental configuration history is used in the inverse dynamics analysis to generate a corresponding force history. Therefore, two sets of configuration and force histories--one set from experiment, and the other from the simulation that is driven forward and backward with the experimental data--are compared in the time domain. More comparisons are made in regard to the effects of initial conditions, friction and viscous damping.

INTRODUCTION

With recent developments in dynamic simulation software, there have been steady improvements in analysis and design of multibody mechanical systems. The performance of a software package has been frequently compared with that of another, but rarely with experimental data. In this research, dynamic simulation is compared with experimental data in time domain.

Through the dynamic simulation, the rigid-body (or flexible-body) equations of motion generate the positions, velocities and accelerations of the components of a given system, and the reaction forces at the system's joints. The equations of motion are usually idealized by not including Coulomb friction and viscous damping, and by simplifying actuating force elements. These idealizations manifest the limitations in the mathematical modelling of a dynamic system. There are also limitations in experimentation. The experiments provide the factual data of the actual system. But such data are not completely reliable because of errors in measurements and subsequent data analysis and interpretation.

The system chosen for the research is J.I. Case 580K backhoe. From a dynamics' standpoint, it is a manipulator of four degrees of freedom (dof) with an operator in the loop. Three dofs are controlled by hand levers for digging, scooping up, and dumping operations, and one dof by a pair of foot pedals for left and right swing motion. These four dofs are individually controlled by hydraulic cylinders that comprise a complicated circuit.

APPROACHES

The approach taken here was to divide the simulation task into multibody dynamics and control element modeling, each of which was separately validated and then later

combined together. In this paper, only the validation of the multibody dynamics is presented. The multibody dynamics includes the modeling of each component and its joints, with the assumption that applied forces or torques are supplied by the control elements. The validation of will thus enable unbiased evaluation of the multibody dynamic simulation, without being influenced by the modeling technique of the controller, i.e., in this case, the hydraulic cylinders and circuitry.

The verification effort started with defining a set of static and dynamic quantities that were both measurable in the experiment and obtainable from the simulation, and that were capable of describing the system status at any specified time. Such quantities were identified as positions and velocities of the system components, and forces acting on the system's joints. In the simulation, the post-processing analysis recovered these quantities easily. In the experiment, however, each quantity demands its own transducer with signal conditioning and data analysis. As a result, experiments were carefully orchestrated with the available equipments, so that the mathematical model could simulate the same operation as in the experiment.

Although there is no established method of validating dynamic simulation in the time domain, the strategy adopted here makes use of the forward and backward (inverse) dynamic analyses, with experimentally known time histories of position and joint forces. The position history that had been measured in the experiment was input into inverse dynamic analysis, which generates a force history that would have driven the simulation model along the input position history. Under the ideal condition such that the dynamic simulation describes the exactly same behavior of the actual system, the two force histories — one from the experiment, the other from the inverse dynamic analysis — should be the same. But, in reality, there inevitably exists a discrepancy between these two. This discrepancy is viewed as a measure of the validation. Similarly, the force history that had been obtained in the experiment was fed into forward dynamic analysis, which generates position history that would have been exactly the same as measured under the ideal condition. Again this position history was compared with the experimental position history.

EXPERIMENTS

Since the boom carries most of the load, its static and dynamic stress analysis is the major concern in design and analysis. Once the dynamic model is validated, it should generate reliable joint reaction forces for dynamics and stress analysis. The experimental effort was thus concentrated on the boom and its hydraulic cylinder. The transducers were attached to boom are a load cell that measures the boom cylinder force output, a differential pressure transducer between the supply and drain sides of the boom cylinder, and a position/velocity transducer for the boom cylinder piston movement. The experiments were conducted by actuating the boom cylinder with various fixed configurations of the dipper and bucket assembly. Among those various configurations, two of them were selected for experimentation and simulation. First, the bucket and the dipper were tucked in under the boom, as shown in Fig. 1. Second, the bucket and the dipper were stretched out, as shown in Fig.3 .

Experimental data were digitized, inspected, and recorded in the IBM PC/AT at the experimental site. Later in the lab, the PC was connected to the local network to unload the data to an Apollo workstation. The data were then retrieved, filtered, interpreted, and supplied for comparisons of experimental with theoretical results in static stress analysis and dynamic behavior, and verification of hydraulic actuator models.

Two types of experimental data are used in the dynamic simulations: force and relative displacement histories of the boom cylinder for the forward and backward simulation. Both quantities were measured while the backhoe was being operated through a predefined trajectory. The relative displacement was measured by a position transducer

with one end attached to the piston and the other end to the cylinder housing. The cylinder force was measured by a load cell placed at the piston end of the cylinder.

Experiment I

In the first experiment, the backhoe is in folded-up configuration; that is, the dipper and bucket cylinders are fully extended, so the dipper and bucket are tucked in under the boom. The angle between the boom and the dipper is about 42 degrees. The only degree of freedom allowed is the rotation around the revolute joint between the boom and the swing tower. At the beginning of an experiment, the boom was in upright position, making an angle of 4 degrees with the vertical (Fig. 1). The boom was slowly lowered from the upright position until it reached 38 degrees of boom angle, then stood still a few seconds, and was brought back up to the original position. The duration of this operation was about 30 seconds.

A typical relative cylinder displacement history measured by the position transducer is shown in Fig. 2. Since the boom cylinder extends while the boom drops downward, the rising trend of the displacement history should be interpreted as the downward motion of the boom.

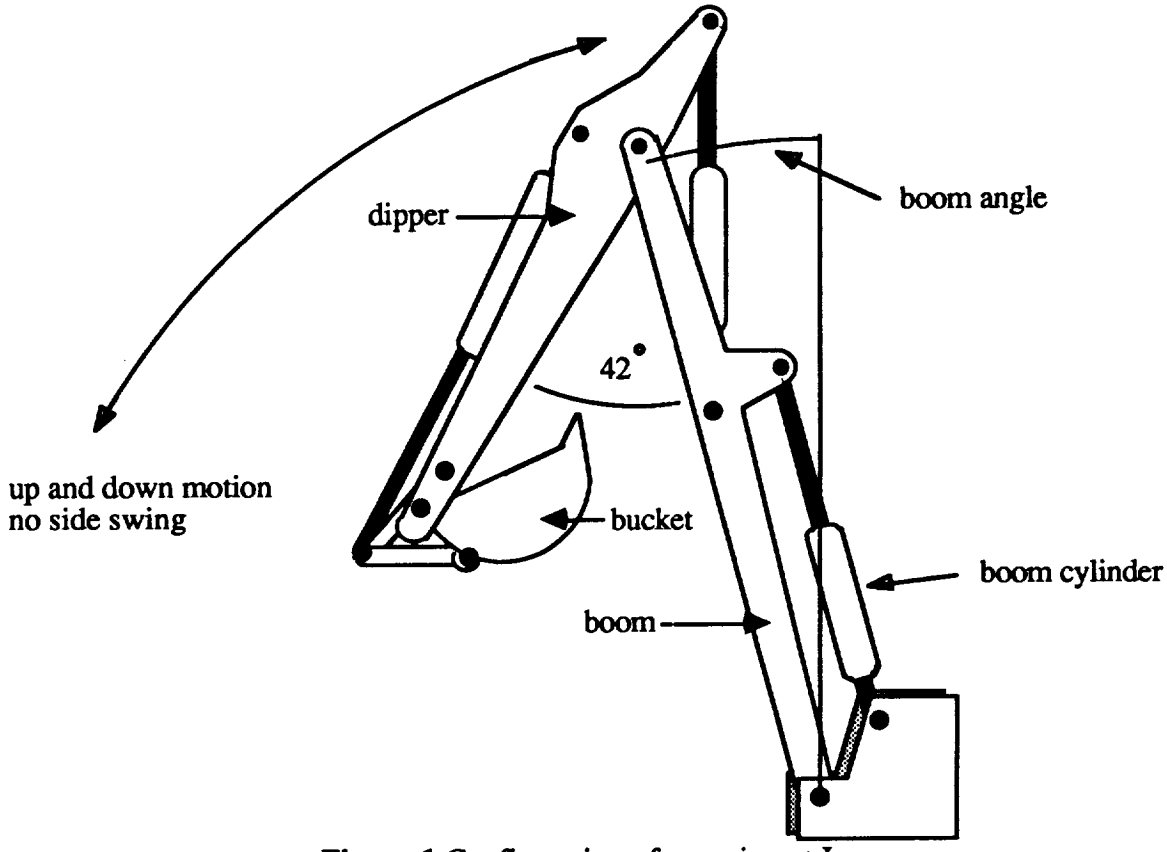
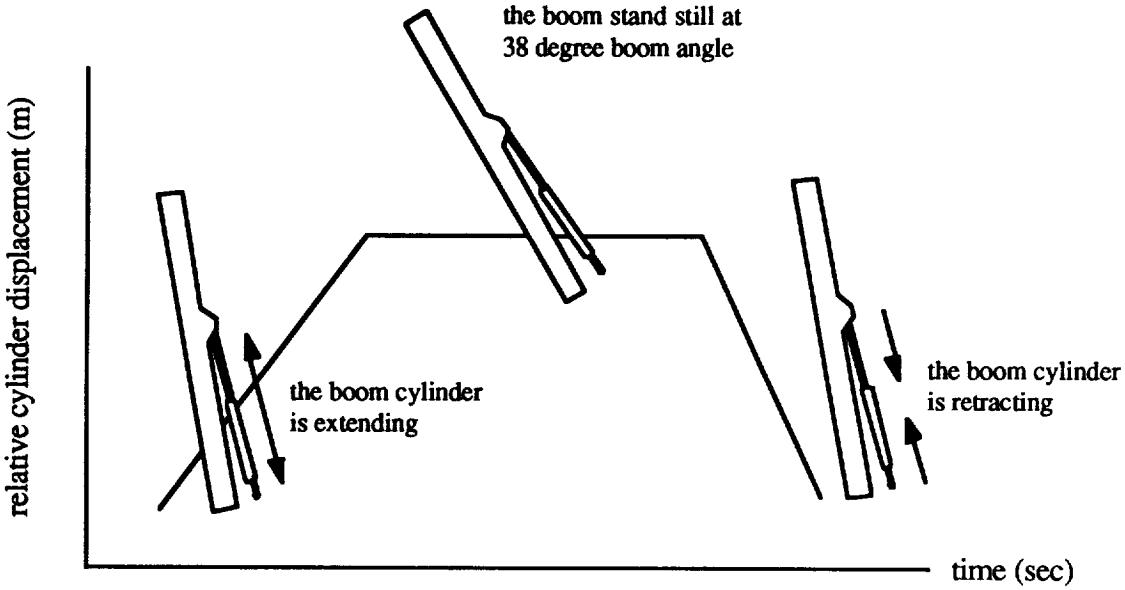


Figure 1 Configuration of experiment I



The motion of the backhoe can be divided into three stages

Figure 2 Position history of experiment I

Experiment II

In the second experiment, the dipper and bucket cylinders were fully retracted so that the backhoe stretched out to its longest reach (Fig. 3). The bucket initially rested on the ground. The boom slowly lifted the bucket up until the bucket reached about 2 m above the ground. Then it brought the bucket back to its original position. This time the whole operation took about 8 seconds. Figure 4 shows a typical relative cylinder displacement history measured in experiment II.

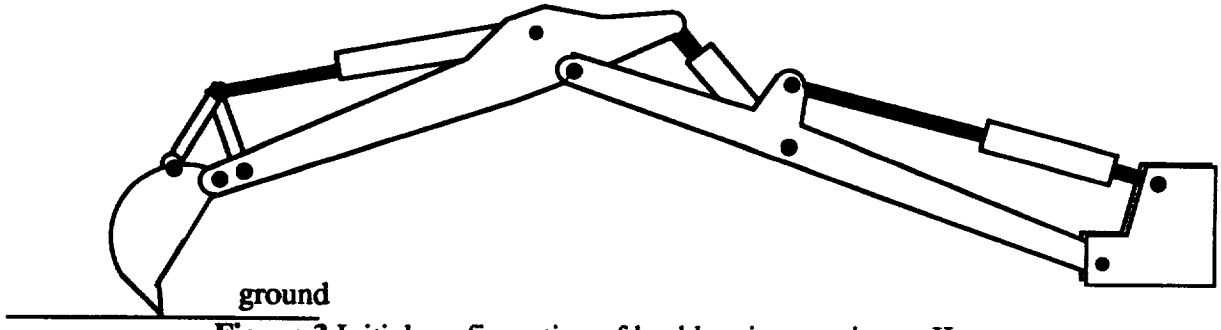


Figure 3 Initial configuration of backhoe in experiment II

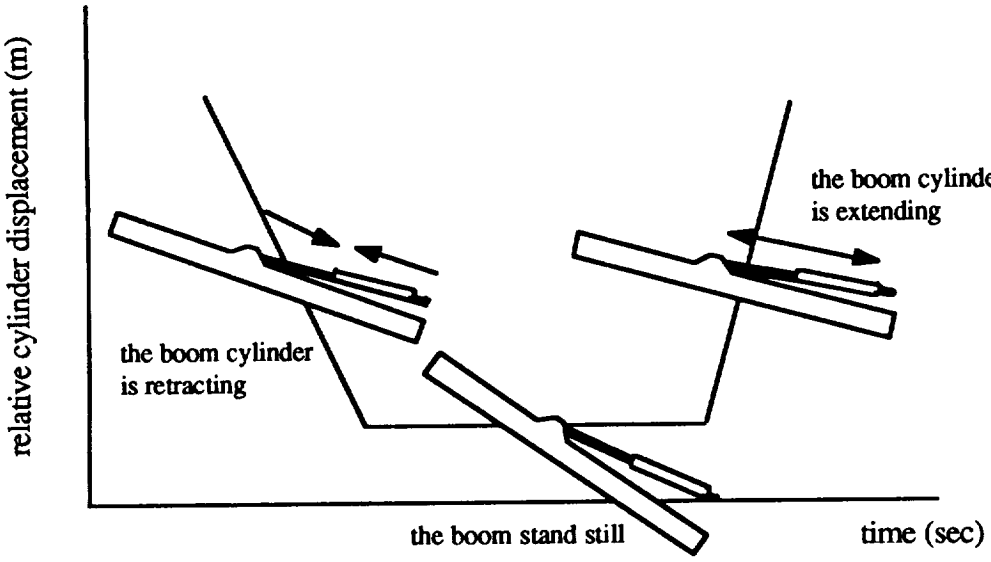


Figure 4 Position history of experiment II

DYNAMIC SIMULATION

Modelling

The dynamic modelling of the backhoe started with a relatively simple model including only the major components, and then added more components such as pins until every single component was accounted for. Table 1 lists the major components and the types of joint used in the model (also see Fig. 6). One of the most significant changes made in model refinements is the addition of the weights of pins and hydraulic fluid. These masses have been regarded insignificant until we found that the simulation model was lacking in the total inertia.

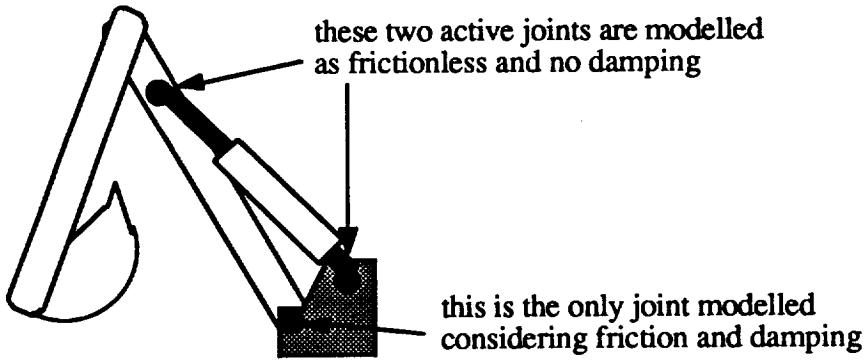


Figure 5 Active joints of backhoe

Among the joints, three of them are active during the experimentation, the locations of these active joints are shown in Fig. 5. There evidently exist viscous and Coulomb friction damping forces at these joints. Since the revolute joint between the tower and the boom is the biggest joint among them, the complexity of the analysis is reduced by modelling that joint as the only joint with viscous and friction damping.

Table 1 Bodies and Joint Types

Body name	Joint Type	Body 1	Body 2
Boom	revolute	Tower	Boom
Dipper	revolute	Boom	Dipper
Bucket	revolute	Dipper	Bucket
Coupler I of bucket	revolute	Coupler I	Coupler II
Coupler II of bucket	cylindrical	Bucket	Coupler II
Boom cylinder	cylindrical	Boom cylinder	Tower
Boom piston	cylindrical	Dipper cylinder	boom
Dipper cylinder	cylindrical	Bucket cylinder	Dipper
Dipper piston	spherical	Dipper	Coupler I
Bucket cylinder	spherical	Boom Piston	Boom
Bucket piston	spherical	Dipper Piston	Dipper
	spherical	Bucket piston	Coupler I

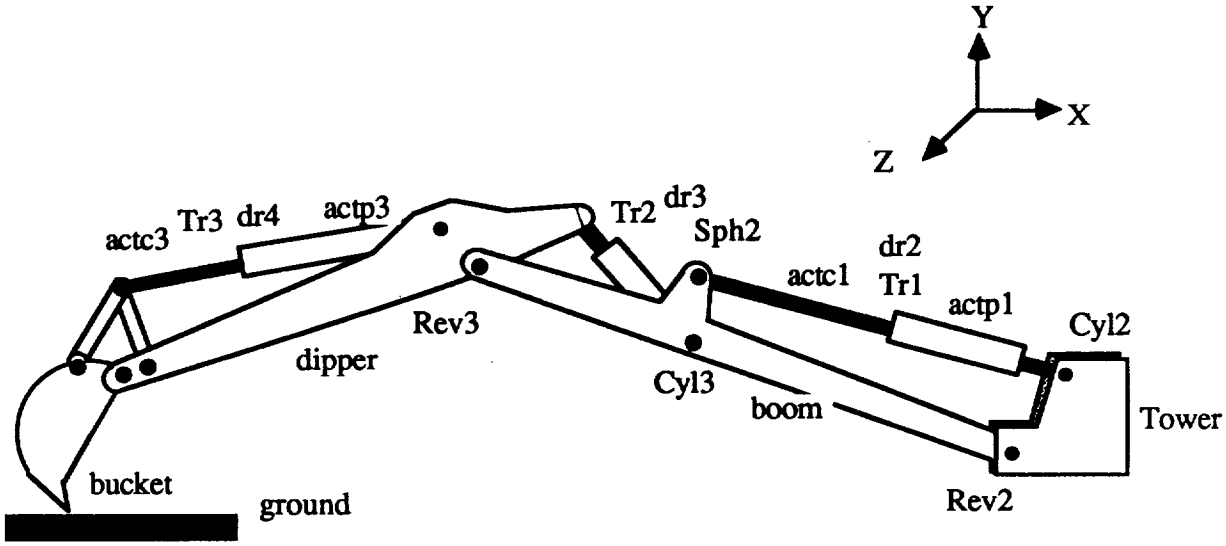


Figure 6 Joint Definition

Initial configurations

In the time-domain validation, the initial conditions of the simulation must first and foremost equal those of the experiment. This requires static equilibrium analysis in the simulation and accurate measurements of position and reaction forces in the experiment. In fact, some initial configurations had to be excluded on the grounds that they were statically indeterminate.

At the beginning of experiment I when the system is in static equilibrium, the force measured at that moment was used to calculate the exact initial position of the backhoe in the experiment. The position measurements were also available from the experiment, but were much less accurate than the force measurement, because a little error in position measurement resulted in a huge error in the corresponding equilibrium force at the initial configuration, which was almost vertical.

In experiment II, a different approach was taken to determine the initial configuration. Since the initial position of the tip of bucket was precisely known, the initial configuration was determined by using this fact. The mass center of the swing tower is defined as the reference coordinate center of a simulation, so the vertical distance from

the reference center to the ground has to be measured. This distance was measured to be about 0.8m. The initial configuration of experiment II is thus obtained based this information and the kinematic relations between bodies of the backhoe model.

SIMULATION AND COMPARISON

In comparison with experiment I, several viscous damping ratios have been tested in the dynamic simulations. Figure 7 shows force comparison in which viscous damping does not play a significant role. Indeed, it was a slow operation, so the viscous damping force was expected to be small. However, the effect of viscous damping is pronouncedly exhibited in the displacement comparison. In Fig. 8, the viscous damping coefficients of 10 and 15 (kN/m/sec) make the simulation close to the experimental data.

In Fig. 9, the simulation with the viscous damping coefficient of 15 (kN/m/sec) continues to move upward (actual motion downward) even when the actual system stopped and stood still, thus exposing the absence of Coulomb friction in the simulation model. The existence of Coulomb friction is also observed in Fig. 7. The force from the simulation is not reduced by the amount of Coulomb friction force, whereas the applied force has already reflected loss from Coulomb friction. Therefore, the simulation force would be equal to the sum of the Coulomb friction and applied forces if the simulation exactly matched with the experiment. In the first half where the friction force is in the same direction with the applied force, the simulation force appears above the actual applied force. In the second half where the friction force is in the opposite direction to the applied force, the simulation force appears below the actual applied force.

In experiment II, the long stretch of the backhoe in combination with a faster maneuver induced vibrations that are visible in Fig. 10. But the simulation shows no vibration but follows the general trend, because the system is modelled with rigid body dynamics. Figure 11 shows a good agreement between the simulation and experiment in position history.

DISCUSSION

When the experimental position history was input to inverse dynamic analysis, it was differentiated twice to obtain velocity and acceleration. Along with this digitized position history, however, noises and discontinuities were also differentiated twice, thereby creating quite a few "jerks", which in turn made the simulation force fluctuate spuriously. To correct this problem, three smooth curves of first and third order polynomials were pieced together to approximate the experimental position history. At the two junction points, spurious peaks are still observed in Fig. 7 and 10.

Experimental estimation of viscous and Coulomb friction damping should accompany the analytical effort in which several dynamic simulations were performed with different damping coefficients. These damping forces were not so significant in Fig. 7. But their effect on the displacement is quite noticeable as shown in Fig. 8.

The validation in the time domain requires that the initial condition of the simulation should equal that of the experiment. This requirement is most of times very difficult to satisfy, because it involves static equilibrium analysis in the simulation and accurate measurements of position and reaction forces in the experiment.

FIG. 7 EXPERIMENT I: FORCE COMPARISON

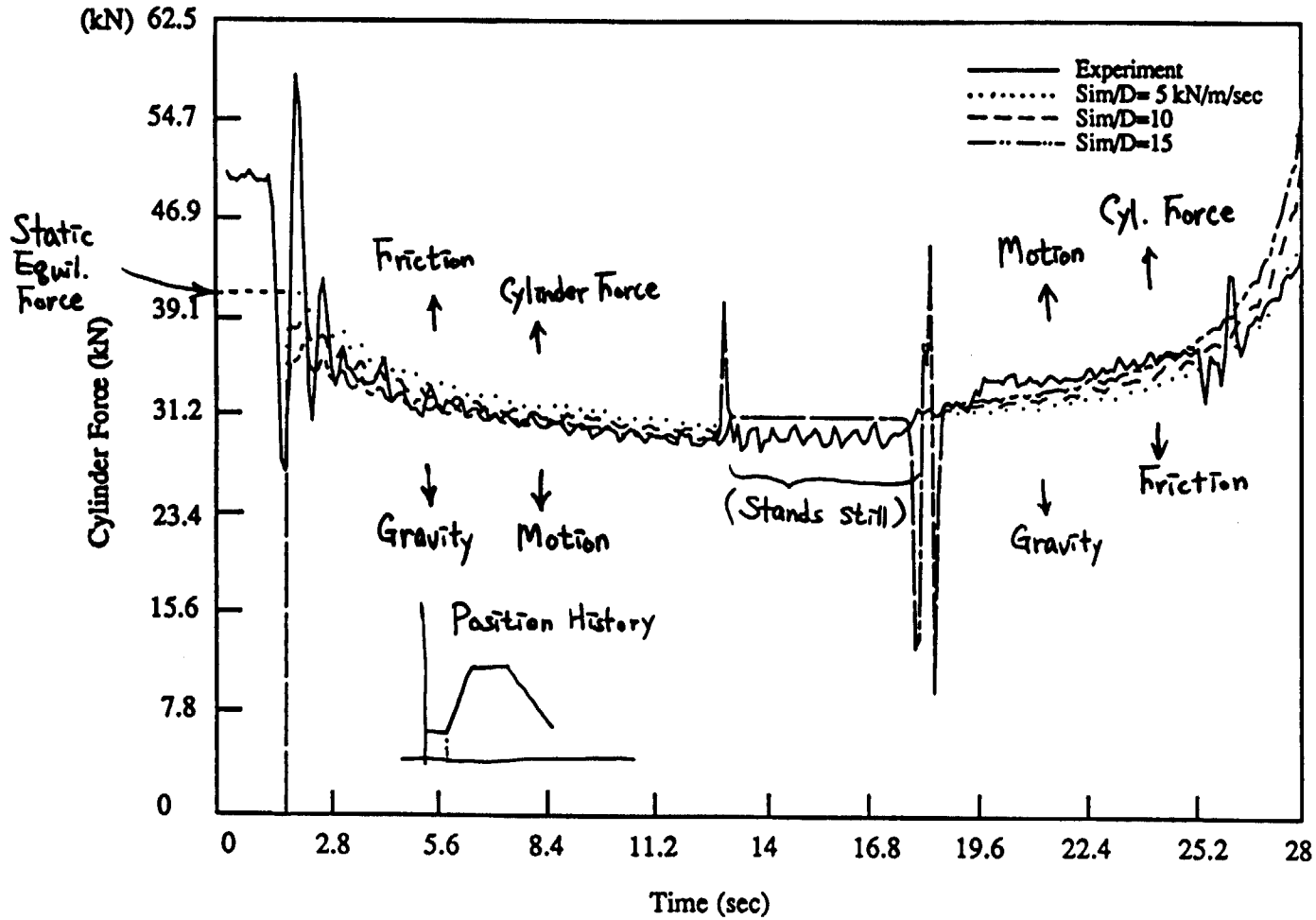


FIG. 8 EXPERIMENT I: POSITION COMPARISON

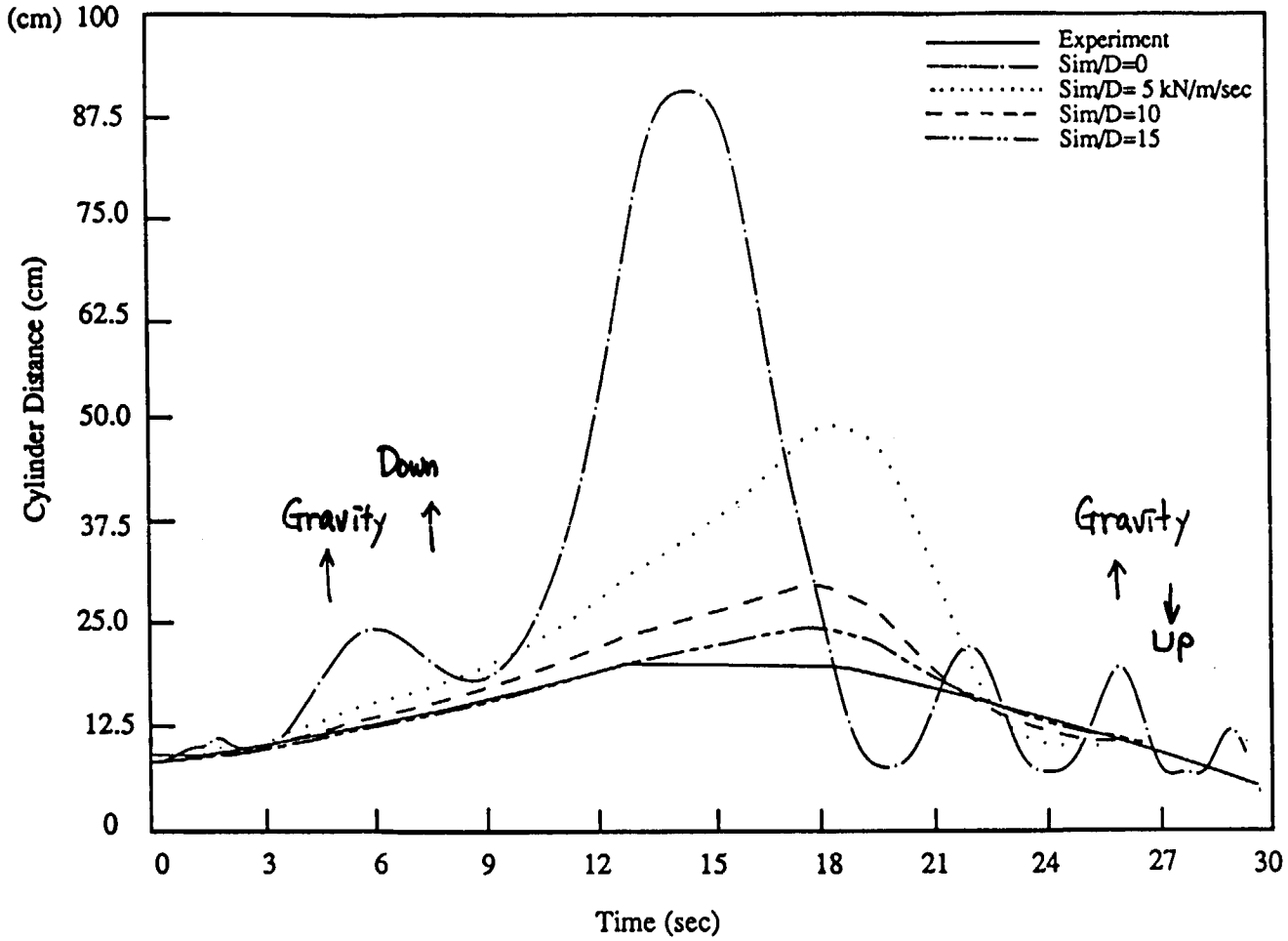


FIG. 9 EXPERIMENT I: POSITION COMPARISON

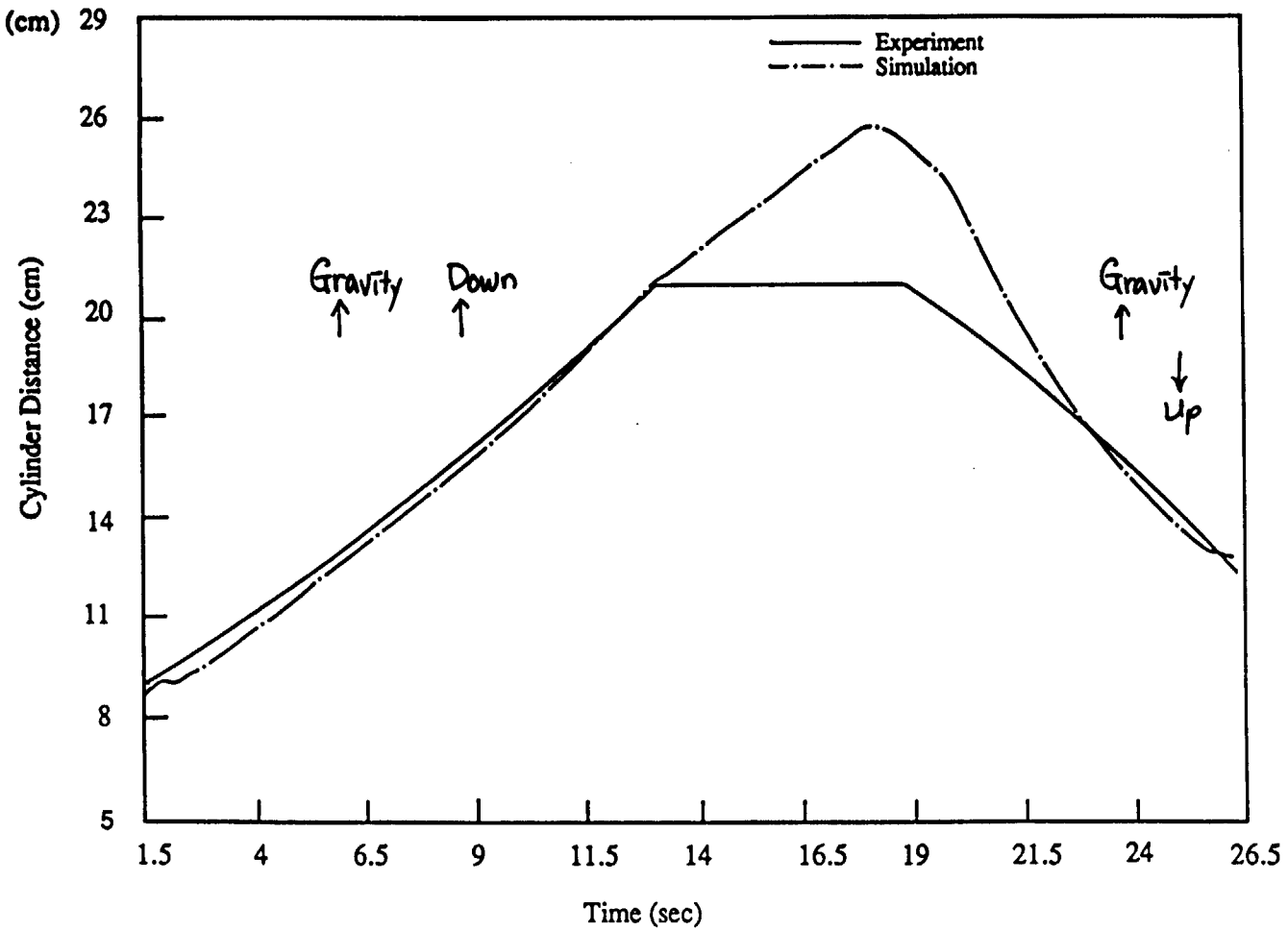


FIG. 10 EXPERIMENT II: FORCE COMPARISON

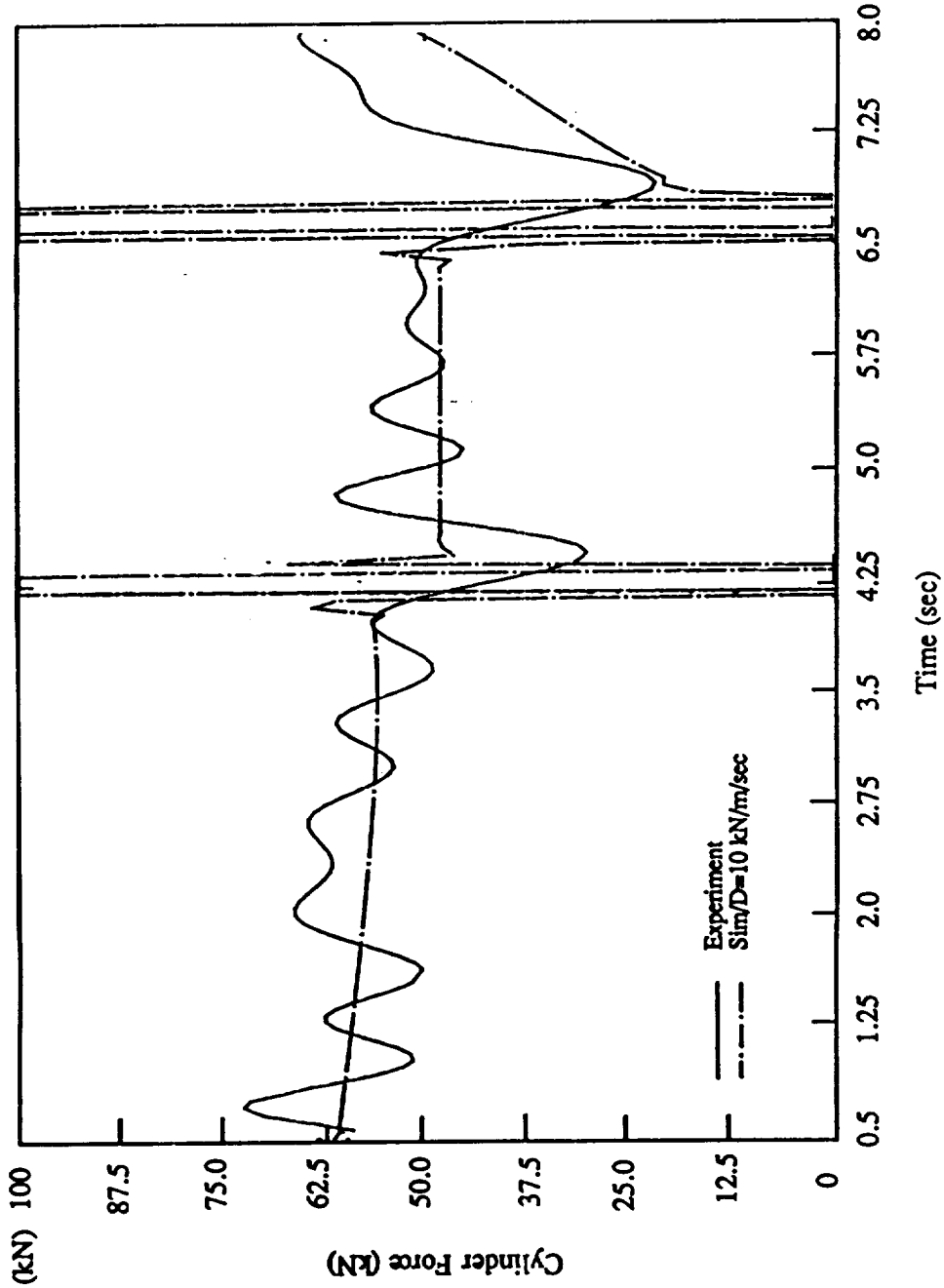
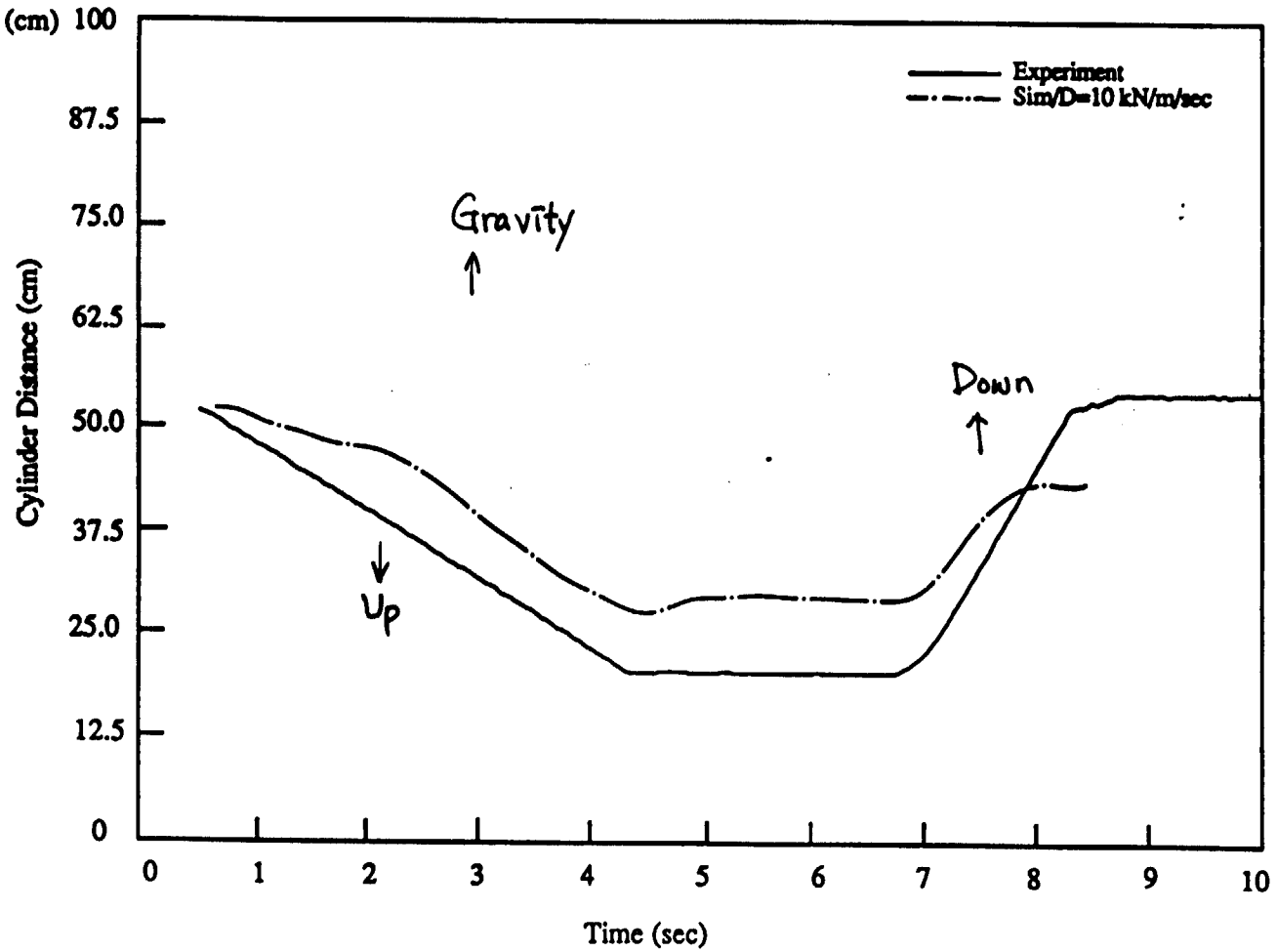


FIG. 11 **EXPERIMENT II: POSITION COMPARISON**



Frequency Response Modeling and Control of Flexible Structures: Computational Methods[1]

William H. Bennett

TECHNO–SCIENCES, INC.
7833 Walker Dr. – Suite 620
Greenbelt, MD 20770

ABSTRACT

The dynamics of vibrations in flexible structures can be conveniently modeled in terms of frequency response models. For structural control such models capture the *distributed parameter* dynamics of the elastic structural response as an irrational transfer function. For most flexible structures arising in aerospace applications the irrational transfer functions which arise are of a special class of pseudo-meromorphic functions which have only a finite number of right half plane poles. In this paper, we demonstrate computational algorithms for design of multiloop control laws for such models based on optimal Wiener-Hopf control of the frequency responses. The algorithms employ a sampled-data representation of irrational transfer functions which is novel and particularly attractive for numerical computation. One key algorithm for the solution of the optimal control problem is the spectral factorization of an irrational transfer function. We highlight the basis for the spectral factorization algorithm together with associated computational issues arising in optimal regulator design. We also highlight options for implementation of wide band vibration control for flexible structures based on the sampled-data frequency response models. A simple flexible structure control example is considered to demonstrate the combined frequency response modeling and control algorithms.

1 Introduction

Frequency response methods offer several advantages for modeling the dynamics of small amplitude vibrations in flexible structures. Such models capture the *distributed parameter* dynamics of the elastic structural response as an irrational transfer function from localized actuation to localized deformation measurements. Interest in frequency response models can arise from a desire to predict modal frequencies with increased accuracy over that obtainable from finite element methods. The frequency domain approach is well suited to optimal control law synthesis with specific requirements for precision vibration supression and isolation. Most computational methods for optimal control synthesis available to design engineers focus on the manipulation of state space models. For flexible structure control, state space models are problematic since the question of model order required must be resolved as part of the optimal control computation. This paper reports progress in the development and testing of computational methods for design of precision control systems for mechanical structures with

[1]Work supported by SDIO and Air Force Wright Research and Development Center under contract F33615-88-C-3215.

$\mathbf{C}_+$	open right half complex s-plane, $\Re e\ s > 0$
$\mathbf{H}_\infty$	Hardy space of complex functions, analytic and essentially bounded in $\mathbf{C}_+$
$\mathbf{H}_2$	Hardy space of complex functions, $f(s)$, analytic in $\mathbf{C}_+$ and such that; $\frac{1}{2\pi j}[\int_{-j\infty}^{j\infty} \|f(s)\|^2\ ds]^{1/2} < \infty$ for $s \in \mathbf{C}_+$.
$\mathbf{RH}_\infty$	rational functions in $\mathbf{H}_\infty$

Table 1.1: Notation

elastic effects based on direct frequency response models. The frequency response models can arise from finite element analysis, transfer function methods, wave propagation models, and/or empirical measurements. Moreover, the computational approach offers a framework for integration of frequency response data from various modeling approachs which offer varying precision in different frequency bands. The current paper extends the efforts reported in [1].

We will use the following notation and conventions in this paper. The transpose of a column vector will be denoted as x^T, $Tr\ X$ is the trace of the square matrix X, and $j = \sqrt{-1}$. A Laplace (resp. z) transform will normally be indicated by dependent variable; $x(s)$ (resp. $x(z)$), however, we often drop the explicit dependence where the meaning is clear from the context. The notation $u_*(s) = u^T(-s)$ will be frequently used. $E\{x(t)\}$ indicates the expectation of the random process $x(t)$. In this work all random processes are assumed wide sense stationary and ergodic so that expectation can be replaced with ensemble average where convenient. The notation contained in Table 1.1 specifies the classes of transfer function models considered at various points. A rational function has a (partial fraction) expansion $A(s) = \{A(s)\}_+ + \{A(s)\}_- + \{A(s)\}_\infty$ where $\{.\}_+$ (resp. $\{.\}_-$) is analytic in $\Re e s > 0$–the causal part ($\Re e s < 0$–the anti-causal part) and $\{.\}_\infty$ is the part associated with poles at infinity. Thus the operation $\{A(s)\}_+$ is *causal projection* of the frequency response model.

In section 2 we provide an overview of frequency domain models and modern Wiener-Hopf design of multiloop control systems. We motivate the role of frequency response modeling and optimal control and identify critical computational steps required for the method. Section 3 discusses a new approach to the required computations for Wiener-Hopf control which extend the algebraic constructions for rational transfer functions to certain irrational cases. We highlight the role of coprime factorization in design of distributed systems. Section 4 considers a simple, but nontrivial distributed parameter system design. Finally, in section 5 we discuss new options for real time control implementation suggested by the computational approach of section 3.

2 Optimal Control of Frequency Response Models: Wiener-Hopf Design

Frequency domain models have been used to articulate the full range of opportunities for feedback compensation for internal model stabilization. Algebraic constructions based on Laplace transform models of linear, time-invariant system dynamics have been used to describe alternatives for standard control computations and realizations for stabilizing controllers [2,

3]. This together with Wiener-Hopf optimization provides a general approach for resolving tradeoffs in regulator design where natural, frequency domain specifications for model-based, control performance are available. We remark that restricting attention to rational transfer function models in computational approachs to flexible structure control has been primarily motivated by convenience [1]. Specific results which extend the constructions of coprime factorization and internal stabilization to a certain class of irrational transfer functions have been obtained [4]. In the present effort we restrict attention to transfer functions in $\mathbf{H}_\infty$and meromorphic with the exception of a small number of right half plane poles.

Optimal regulator design via Wiener-Hopf methods. Techniques for the solution of $\mathbf{H}_2$ optimization problems in multiloop feedback systems have received considerable attention in the control theory literature for a number of years. A comprehensive approach to Wiener-Hopf design using transfer function models is given by Youla et al [5].

A general framework for resolution of tradeoffs in multiloop control design was recently outlined by Park and Bongiorno [6]. In general, control design involves the resolution of choices in the use of dynamic (feedback) compensation with respect to a nominal dynamic model of the system response to an n-vector control, u and an m-vector of exogenous system disturbances, e, as seen by p available sensors, y and ℓ (possibly nonmeasurable) regulated variables. A frequency domain model for the control design problem (shown in Figure 2.1) can be expressed using Laplace transforms as,

$$\begin{aligned} \begin{pmatrix} y(s) \\ z(s) \end{pmatrix} &= G(s) \begin{pmatrix} u(s) \\ e(s) \end{pmatrix} \\ &= \begin{bmatrix} -G_{yu} & G_{ye} \\ G_{zu} & G_{ze} \end{bmatrix}(s) \begin{pmatrix} u(s) \\ e(s) \end{pmatrix}. \end{aligned} \tag{2.1}$$

The control architecture is assumed to involve feedback,

$$u(s) = C(s)y(s). \tag{2.2}$$

Then the closed loop compensation will alter the response of the system to disturbances as seen in terms of the *regulated variables* as[2],

$$\begin{aligned} z &= [G_{zu}CS, I] \begin{bmatrix} G_{ye} \\ G_{ze} \end{bmatrix} e \\ &= G_{zu}CSG_{ye}e + G_{ze}e \end{aligned} \tag{2.3}$$

where the system closed loop *sensitivity operator* is $S = [I + G_{yu}C]^{-1}$.

A major consideration in the classical methods of frequency domain design is closed loop stability. In such methods stability considerations must be continually evaluated (using root locus or Nyquist plots) as performance tradeoffs are evaluated. For single loop designs of relatively low order systems, classical frequency domain methods focus attention on the tradeoff between stability margins and performance. The modern approach is to use optimization to

[2]Suppressing dependence on the Laplace variable s.

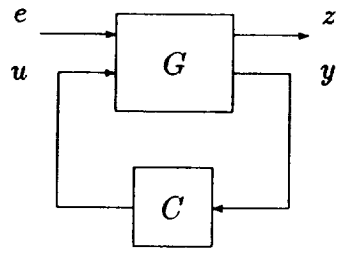


Figure 2.1: General Multiloop Control System Design Problem

resolve complicated engineering tradeoffs in multiloop design—subject to the constraint of system internal stability.

The well known Youla parametrization of all stabilizing feedback controllers C which stabilize a given plant model was orginally derived for rational matrix transfer functions which have *coprime factorizations* over the ring of polynomials in the Laplace s-variable [5]. It is now understood that the construction goes through without modification for the ring of stable rational functions, $\mathbf{RH}_\infty$which includes all rational transfer functions analytic in the closed right half plane including the point at infinity [7]. Thus under the assumption that the plant transfer function has right and left factorizations,

$$G_{yu} = ND^{-1} = D_\ell^{-1}N_\ell, \tag{2.4}$$

coprime over $\mathbf{RH}_\infty$, then there exist $X, Y, X_\ell, Y_\ell \in \mathbf{RH}_\infty$ such that

$$D_\ell X_\ell + N_\ell Y_\ell = I, \qquad XD + YN = I, \tag{2.5}$$

and each controller, C, which obtains $R = CS$ with $R \in \mathbf{RH}_\infty$ can be *parametrized* by the factorization formulae,

$$C = (X - KN_\ell)^{-1}(Y + KD_\ell) \tag{2.6}$$

$$= (Y_\ell + KD)(X_\ell - KN)^{-1} \tag{2.7}$$

for some $K \in \mathbf{RH}_\infty$. The importance of this construction is that optimization procedures can be applied directly to the choice of $K \in \mathbf{RH}_\infty$ without concern for closed loop stability. This fact was first exploited by Youla et al[5] in the description of Wiener-Hopf optimal control for frequency response models. More recently, the parametrization has been utilized by Desoer and his students [8].

Our concern here is with $\mathbf{H}_2$ optimization for a certain class of irrational transfer functions which are "pseudo-meromorphic" in the sense stable coprime factorizations exist; i.e., the constructions in (2.4)-(2.7) obtain closed loop stability with $N, D, N_\ell, D_\ell, X, Y, X_\ell, Y_\ell, K \in \mathbf{H}_\infty$[1, 4]. In the research reported herein we avoid algebraic constructions related to stable, coprime factorization of irrational transfer functions as considered by Desoer [7] and instead, focus on the development of numerical algorthims for approximating the frequency response

of the required objects. Such transfer functions can be adequately approximated over finite frequency ranges by (rather high order) rational transfer functions. Models of this type arise in the study of vibrations of multibody systems with flexible interactions [9] and wave propagation in flexible mechanical structures [10, 11].

PSD modeling of control processes for performance specification. A wide class of standard control design problems including symultaneous requirements for tracking, disturbance rejection and accomodation, etc. can be represented in the form of the general linear, time-invariant regulator problem of Fig. 2.1 and (2.1)-(2.2) where the objective is to choose a controller which stabilizes the closed loop system and minimizes a performance criterion in the form,

$$J = \frac{1}{2\pi j}\int_{-j\infty}^{j\infty} Tr[Q(s)P_z(s)]\,ds, \tag{2.8}$$

where P_z is an effective Power Spectral Density (PSD) of the regulated variables, z.

The representation of regulation performance in terms of PSD is quite practical for a variety of design problems arising in multiloop systems and provides a frequency dependent specification of control performance consistent with design requirements for vibration rejection. One can extend the significance of PSD modeling to include a wide range of practical design considerations. The regulation PSD, P_z, can be related to modeling assumptions on the exogenous inputs in terms of the closed loop transfer functions;

$$P_z = [G_{uz}R,\ I]\begin{pmatrix} G_{ye} \\ G_{ze} \end{pmatrix} P_e\,(G_{ye*},\ G_{ze*})\,. \tag{2.9}$$

PSD models of exogenous inputs may include deterministic transient effects together with steady state stochastic PSD, Φ_e; viz.,

$$P_e(s) = \lambda_1 E\{e(s)e_*(s)\} + \lambda_2\Phi_{ee}(s), \tag{2.10}$$

Formulation of the performance objective, J, may include real, positive, λ_i, $i = 1,2$ which permit scaling relative importance of steady state and transient considerations to the composite performance and $Q(s)$ is included to permit frequency weighting. PSD modeling has recently received increased emphasis in the study of vibration control in acoustic regimes [11].

Park and Bongiorno [6] also highlight the use of PSD models for minimizing closed loop system sensitivity to model uncertainty. Let the system model uncertainty be given as a frequency dependent, additive perturbation, $G := G+\Delta$, which can be expressed in partitioned form as,

$$\Delta = \begin{bmatrix} -\Delta_{yu} & \Delta_{ye} \\ \Delta_{zu} & \Delta_{ze} \end{bmatrix}.$$

Following [6] an effective model uncertainty PSD, $P_\Delta = E\{\Delta\Delta_*\}$, can be reflected to the system regulated outputs, and via superposition, a composite performance objective of the form,

$$J = \frac{1}{2\pi j}\int_{-j\infty}^{j\infty} Tr\left\{Q[G_{uz}R,\ I]\begin{bmatrix} \Phi_{yy} & \Phi_{yz} \\ \Phi_{zy} & \Phi_{zz} \end{bmatrix}\right\} ds$$

is obtained for optimal regulator design, where the system model uncertainty can be obtained commensurate with the performance specifications as an effective disturbance PSD,

$$\begin{bmatrix} \Phi_{yy} & \Phi_{yz} \\ \Phi_{zy} & \Phi_{zz} \end{bmatrix} = \begin{pmatrix} G_{ye} \\ G_{ze} \end{pmatrix} P_e \left(G_{ye*}, G_{ze*} \right) + \mu P_\Delta. \tag{2.11}$$

Here the partitioned terms can be expressed in terms of a priori modeling assumptions; [6]

$$\Phi_{yy} = G_{ye} P_e G_{ye*} + \mu (E\{\Delta_{yu}(\Delta_{yu})_*\} E\{\Delta_{ye}(\Delta_{ye})_*\}), \tag{2.12}$$

$$\Phi_{yz} = G_{ye} P_e G_{ze*} + \mu (E\{\Delta_{yu}(\Delta_{zu})_*\} E\{\Delta_{ye}(\Delta_{ze})_*\}) = \Phi_{zy*}, \tag{2.13}$$

$$\Phi_{zz} = G_{ze} P_e G_{ze*} + \mu (E\{\Delta_{zu}(\Delta_{zu})_*\} E\{\Delta_{ze}(\Delta_{ze})_*\}). \tag{2.14}$$

It is by now widely recognized that frequency domain response considerations are extremely important for robust control design and that performance objectives formulated in the frequency domain are important tools for resolving design tradeoffs of relevance to practical design problems. However, the common wisdom is that state space modeling offers the most reliable numerical framework for the computational problems which arise in optimal regulator design. The Wiener-Hopf approach identifies the solution for the optimal controller in an explicit form which highlights the role of the algebraic constructions generic to stabilization and the quantitative computations required for identifying an optimal controller. Thus given the system architecture (2.1)-(2.2), appropriately chosen stable coprime factors for the plant (2.4), a nominal stabilizing controller given in terms of its coprime factors as solutions of the Diophantine relations (2.5), and performance PSD's (2.12)-(2.14), then an optimal closed loop system response is obtained (assuming a solution exists) by the formula,

$$R = D\Lambda^{-1} \left(\{\Lambda D^{-1} Y \Omega\}_- - \{\Lambda_*^{-1} D_* G_{zu*} Q \Phi_{zy} D_{\ell*} \Omega_*^{-1}\}_+ \right) \Omega^{-1} D_\ell. \tag{2.15}$$

The explicit form given here depends on operations of causal projection and the solution of two causal, spectral factorizations;

$$D_* G_{zu*} Q G_{zu} D = \Lambda_* \Lambda \tag{2.16}$$

$$D_\ell \Phi_{yy} D_{\ell*} = \Omega \Omega_* \tag{2.17}$$

with $\Lambda, \Lambda^{-1}, \Omega, \Omega^{-1} \in \mathbf{H}_\infty$. The required controller can then be obtained in the explicit form, $C = (I - RG_{yu})^{-1} R$.

The computational steps required to identify candidate optimal control solutions for the regulator problem include: 1) stable coprime factorization (as in (2.4), 2) identify candidate solution to Diophantine relations (2.5), 3) causal spectral factorization, and 4) causal projection. We contend that such computations can be effectively supported (in finite precision arithmetic) by obtaining state space realizations [3] *only for relatively low order, rational transfer functions.* In the sequel, we specifically avoid such an approach since we are ultimately concerned with the approximate solution of large (or even infinite) dimensional models.

3 Frequency Response Computations for Optimal Control.

Our approach to optimal control computation is motivated by distributed parameter models which arise in flexible structure control. The approach we have in mind is based on sampling

and interpolation of the frequency response models for the system. The choice of sampling and the resulting high order, rational approximations are obtained in the context of the optimal control problem as summarized above.

A computational approach to spectral factorization. Recall that a transfer function $H(s) \in \mathbf{H}_2 \cap \mathbf{H}_\infty$ has a unique *spectral factorization* $H(s) = F(s)F_*(s)$ with $F \in \mathbf{H}_\infty$ if:

1. $\overline{H(s)} = H(\bar{s})$; i.e., $H(s)$ is the transform of a real-valued function $h(t)$.
2. $H(s) = H_*(s)$; i.e., $H(s)$ is "para-hermittian".
3. $H(s)$ is of normal rank; i.e., full rank almost everywhere in $\mathbf{C}$.
4. $H(i\omega)$ is positive, semi-definite and bounded for $\omega \in \mathbf{R}$.

To see that causal projection is a closely related problem consider the following. If $H(s)$ is scalar, then with $\Phi(s) = \ln H(s)$ we obtain

$$\Phi(s) = \{\Phi(s)\}_+ + \{\Phi(s)\}_-, \tag{3.1}$$

$$= \ln F(s) + \ln F_*(s), \tag{3.2}$$

so that the causal, spectral factorization is related to causal projection via the logarithmic transformation; $F(s) = \exp\{\ln H(s)\}_+$.

Our goal is to obtain numerically stable approximations to these related problems for transfer functions in $\mathbf{H}_\infty$. For application to precision control of flexible structures we require wide band frequency domain models so that even rational approximations will be of relatively high order. An approach to model order reduction which has recently received attention in the literature is based on Fourier series approximation of irrational frequency responses [12] in $\mathbf{H}_\infty$. Our approach to computations for such models is also based on sampling and interpolation of the spectrum, but is motivated by computational requirements for Wiener-Hopf optimization. From the above discussion of causal projection we motivate a class of algorithms of interest from basic properties of the Hilbert transforms applied to the frequency response $\Phi(j\omega)$. Recall that the Hilbert transform of a time signal $f(t)$ is defined as a convolution; $\check{f}(t) = \int_{-\infty}^{\infty} \frac{f(\tau)}{\pi(t-\tau)}\, d\tau$ and it's Fourier transform has the property,

$$\check{f}(\omega) = \begin{cases} -jf(\omega), & \omega > 0 \\ jf(\omega), & \omega > 0 \end{cases}.$$

The inverse Fourier transform of $\check{\Phi}$ is $-j\mathrm{sgn}(t)\phi(t)$ where $\phi(t)$ is the inverse Fourier transform of $\Phi(\omega)$. A consequence is that the casual projection can be obtained as

$$\{\Phi(\omega)\}_+ = \frac{1}{2}[\Phi(\omega) + j\check{\Phi}(\omega)].$$

In previous studies we reported computational algorithms for causal projection and scalar spectral factorization by numerical evaluation of the Hilbert transform integral. Computational cost was high due to the fact that the Hilbert transform integral is convergent only in

the Cauchy principal value sense [13]. An alternate method for causal projection and spectral factorization was considered in [14] based on sampling and interpolation of the system frequency response. The algorithm developed in [14] employs results of Stenger [15] on numerical solution of Wiener-Hopf integrals by sampling and interpolation. Details of the algorithm used for the current studies and computer implementation are given in [14].

In the multiloop, optimal regulator design problem we require the solution of two matrix spectral factorization problems (analogous to the solution of control and filtering Riccati equations for time domain models). The computational approach exploited in the current study is based on a Newton-Raphson iteration for the matrix causal spectral factor;

$$F_{n+1}(i\omega) := \left\{[F_n^*(i\omega)]^{-1} H(i\omega)[F_n(i\omega)]^{-1}\right\}_+ F_n(i\omega). \tag{3.3}$$

The recursion (3.3) can be replaced with a numerically well conditioned problem by iteration on the inverse spectral factor;

$$[F_{n+1}]^{-1} := [F_n]^{-1}\left(I + \left\{[F_n^*]^{-1} H\,[F_n]^{-1} - I\right\}_+\right)^{-1}. \tag{3.4}$$

By initializing with F_0 (an $m \times m$ diagonal matrix) with diagonal elements equal to the spectral factors of the diagonal elements of H the second term of (3.4) remains a perturbation of the identity (since $[F_n^*]^{-1}H[F_n]^{-1} - I \to 0$) which regularizes the computations. The algorithm used in this work is based on that reported in [14] and is a modified form of the method reported in [16].

Computation of stable coprime factorizations for flexible structure models. Simple models of structural components with elastic effects typically lead to transfer functions in $\mathbf{H}_\infty$once realistic damping models are included. Linear vibration models of more complex structures arising in aerospace applications usually will have transfer function models with only a finite number of poles in the closed right half plane. Restricting attention to such transfer functions we indicate a simple procedure for coprime factorization over $\mathbf{H}_\infty$.

Let $\mathcal{S} \subseteq \mathbf{H}_\infty$ be a set of transfer functions analytic in a half plane including $\mathbf{C}_+$. Under the above assumption any such transfer function $P(s)$ can be expressed in the form,

$$P(s) = P_{\mathcal{S}}(s) + P_{\mathcal{S}'}(s) \tag{3.5}$$

where $P_{\mathcal{S}} \in \mathcal{S} \subseteq \mathbf{H}_\infty$ and $P_{\mathcal{S}'}$ is rational and analytic in the complement of $\mathcal{S}$ with (a finite number of) poles outside $\mathcal{S}$. A stable coprime factorization can be readily obtained for the (typically low order) transfer function as, $P_{\mathcal{S}'} = \tilde{N}_r \tilde{D}_r^{-1}$, by well known state space constructions [3]. Then P has stable coprime factorization,

$$P = N_r D_r^{-1} = [\tilde{N}_r - P_{\mathcal{S}}\tilde{D}_r]\tilde{D}_r^{-1}, \tag{3.6}$$

where N_r, D_r are $\mathcal{S}$-stable. The separation of terms in (3.5) is readily carried out given $P(s)$ by computing the residues of the finite number of unstable poles contributing to $P_{\mathcal{S}'}$.

4 Control Computations for an Elastic Structure

To illustrate the computational approach for a simple elastic structure we consider the simply supported Euler beam with torque control at one end. The beam lateral deformation is given by $y(t,z)$ with $0 \leq z \leq L$ and has dynamics described by the dimensionless PDE;

$$\frac{\partial y^2}{\partial t^2} - 2\zeta \frac{\partial^3 y}{\partial t \partial z^2} + \frac{\partial^4 y}{\partial z^4} = 0. \tag{4.1}$$

with boundary conditions at $z = 0$,

$$y(t,0) = 0, \qquad \left.\frac{\partial^2 y}{\partial z^2}\right|_{z=0} = 0,$$

and at $z = L$,

$$y(t,L) = 0, \qquad \left.\frac{\partial^2 y}{\partial z^2}\right|_{z=L} = \tau,$$

with the control moment, τ, applied at the right hand end of the beam. The transfer function $(\tau \rightarrow y)$ for beam control is

$$G_{yu}(s,z) = L^2 \frac{\sin\lambda_1 \sinh\lambda_2 \frac{z}{L} - \sin\lambda_1 \frac{z}{L} \sinh\lambda_2}{(\lambda_1^2 + \lambda_2^2)\sin(\lambda_1)\sinh(\lambda_2)}, \tag{4.2}$$

where $\lambda_1^2 = (-\zeta + i\sqrt{1-\zeta^2})sL^2$, $\lambda_2^2 = (\zeta + i\sqrt{1-\zeta^2})sL^2$, L is the beam length, ζ is the damping factor, and z is the observation point to be regulated on the beam. The transfer function is meromorphic, and $G_{yu}(s,z) \in \mathbf{H}_\infty$ for any $0 \leq z \leq L$.

The regulator problem considered arises from a requirement for asymptotic rejection of constant load disturbances at a point $\frac{z}{L} = 0.7$. For the current numerical studies we take $L = 10.$, and the effective damping ratio, $\zeta = 0.01$. Stable coprime factorization is trivial and we take $N_r = N_\ell = G_{yu}$, $D_r = D_\ell = 1$. Exogenous inputs here include the output load disturbance d and measurement noise model n and are described by their effective PSD models representing constant (step) load disturbance and narrowband sensor noise as shown in Figure 4.2. The frequency response of G_{yu} is shown in Figure 4.1 with 1024 uniform frequency samples over a bandwidth of $0 < \omega < 100$. Clearly, the frequency response is irrational and no obvious rational approximation is evident.

The optimal control design is regulation of the beam deflection at $z/L = .7$ and the performance objective is given as,

$$J = \frac{1}{2\pi j} \int_{-j\infty}^{j\infty} Tr\{\Phi_y + \mu\Phi_u\}\, ds$$

where the tracking cost is modeled by PSD, Φ_y, and the control saturation PSD is $\Phi_u = E\{uu_*\}$. Then given a constraint on the control power the scalar $\mu > 0$ plays the role of a Lagrange multiplier for the optimal design. In this case the required spectral factors;

$$(G_{yu*}G_{yu} + \mu) = \Lambda_*\Lambda \tag{4.3}$$

$$(G_{yd}\Phi_d P_{yd*} + \Phi_n) = \Omega\Omega_* \tag{4.4}$$

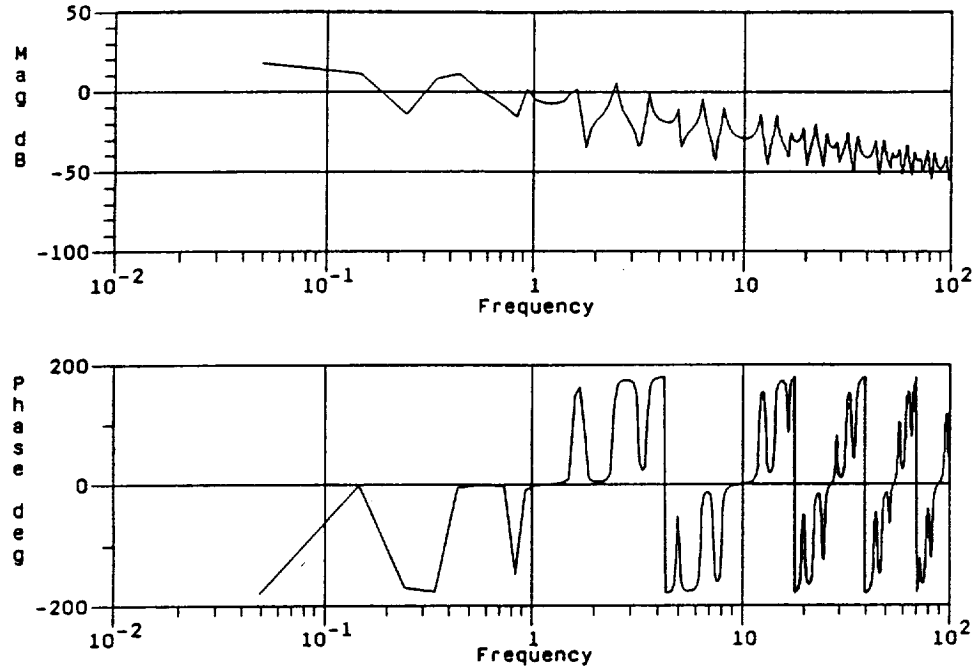


Figure 4.1: Frequency response for pinned-pinned beam control.

were obtained for $\mu = .1$ by the frequency sampled procedure and are displayed in Figure 4.3. We remark that the computations of the indicated spectral factors effectively replace the computational step of solving a pair Riccati matrix equation for the control (resp. filter) problem typically encountered in state-space methods for control design. For distributed parameter systems, solution of the Riccati equation (a PDE) requires discretization which is accomplished using the current algorithms by sampling and interpolation of the frequency response. Thus numerical precision is concentrated over frequency bands significant for the given control problem and with sampling under direct control of the designer. The solution obtained is effectively a high order rational approximation of the optimal solution with frequency response interpolation points chosen by the design engineer. The optimal controller frequency response thus obtained is shown for $\mu = .1$ in Figure 4.4.

5 Frequency Sampling Filters for Real Time Control Implementation

The frequency response computations for Wiener-Hopf control outlined and illustrated in the previous sections identify various frequency sampled approximations to the ideal, possible irrational frequency response for the desired optimal controller. Bandwidth and sampling can be chosen by the design engineer to represent specific concerns based on models and/or control performance. The frequency sampled computations obtain a *specification* for the frequency response of the ideal (optimal) controller via its sampled representation. The design engineer now has several options for implementing the controller depending on available hardware. In contrast to the state space approach for finite dimensional systems, several new realization opportunities are suggested by the frequency sampling approach.

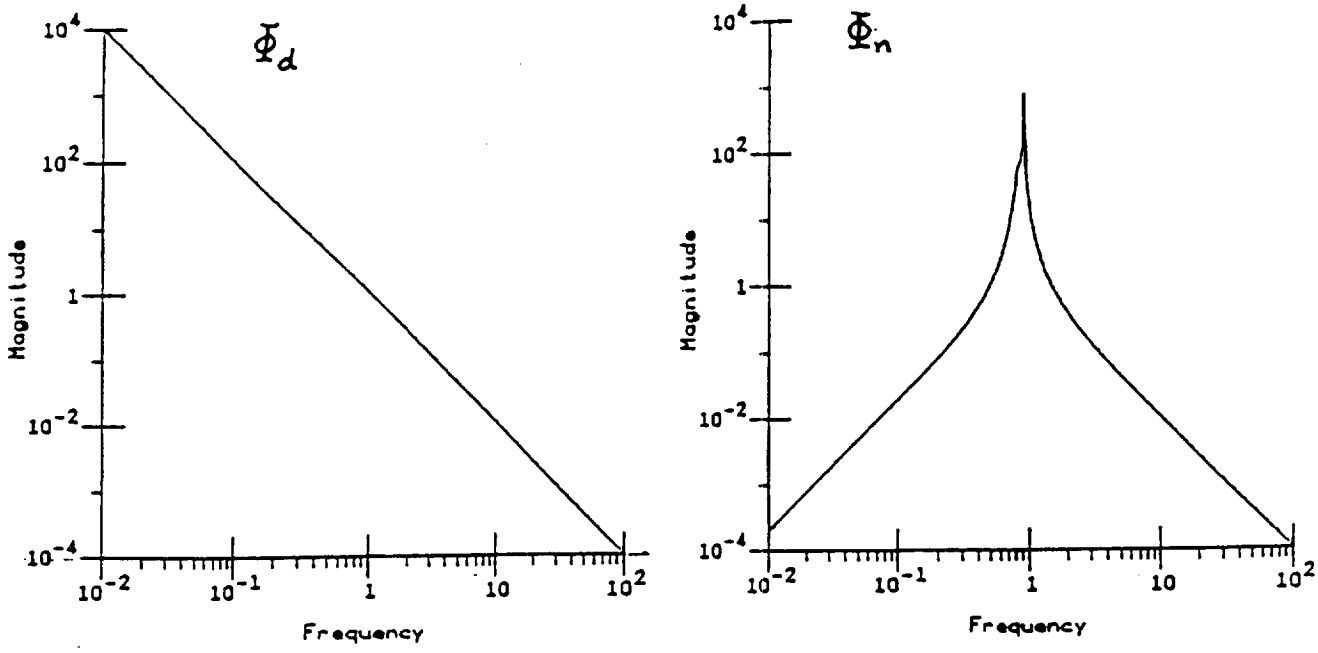


Figure 4.2: PSD for disturbance and sensor noise inputs for beam control.

A principal concern in implementation of high precision control laws for flexible structure control is the order of the realization required for the online controller. The controller order is usually taken to mean the dimension of the state variable realization of the transfer function $C(s)$ which will be implemented for realtime control. Implementation using analog components of high order models is limited by circuit complexity, reliability and cost. As a result considerable effort has been expended in methods for model order reduction. One approach to controller realization which follows from the frequency sampled computations of the previous section is to compute reduced order, continuous time, state space realizations for the controller by techniques such as in [12].

Digital computer implementations are primarily limited by computational speed and algorithm complexity effecting the ultimate obtainable sampling rate and considerations for reduced order realization of the controller may be required. However, the emergence of specialized computer hardware implemented in VLSI single chip circuits for digital signal processing opens new opportunities for realization of realtime control for flexible structures. We prefer to consider realization options for the optimal controller in discrete time for implemenation on a digital computer. Realization of the controller specified by its frequency samples can be obtained using a FIR digital filter implementation.

Given the specified frequency samples obtained for the optimal controller,

$$C_k = C(j\omega_k),$$

at frequencies, $\omega_k = k\omega_{BW}/N$, where ω_{BW} is bandwidth and N the number of uniformly spaced frequency samples we describe the digital filter realization using z-transforms. With discrete time sampling rate $\omega_s > 2\omega_{BW}$ the frequency samples correspond to interpolation points in the z-plane given by[3], $z_k = e^{jk2\pi/N_1}$, for $k = 0, \ldots, N-1$. The z transform which

[3]Bandwidth and sampling requirements would typically require padding the sequence of frequency samples of length N with $N_1 - N$ zero values to avoid aliasing.

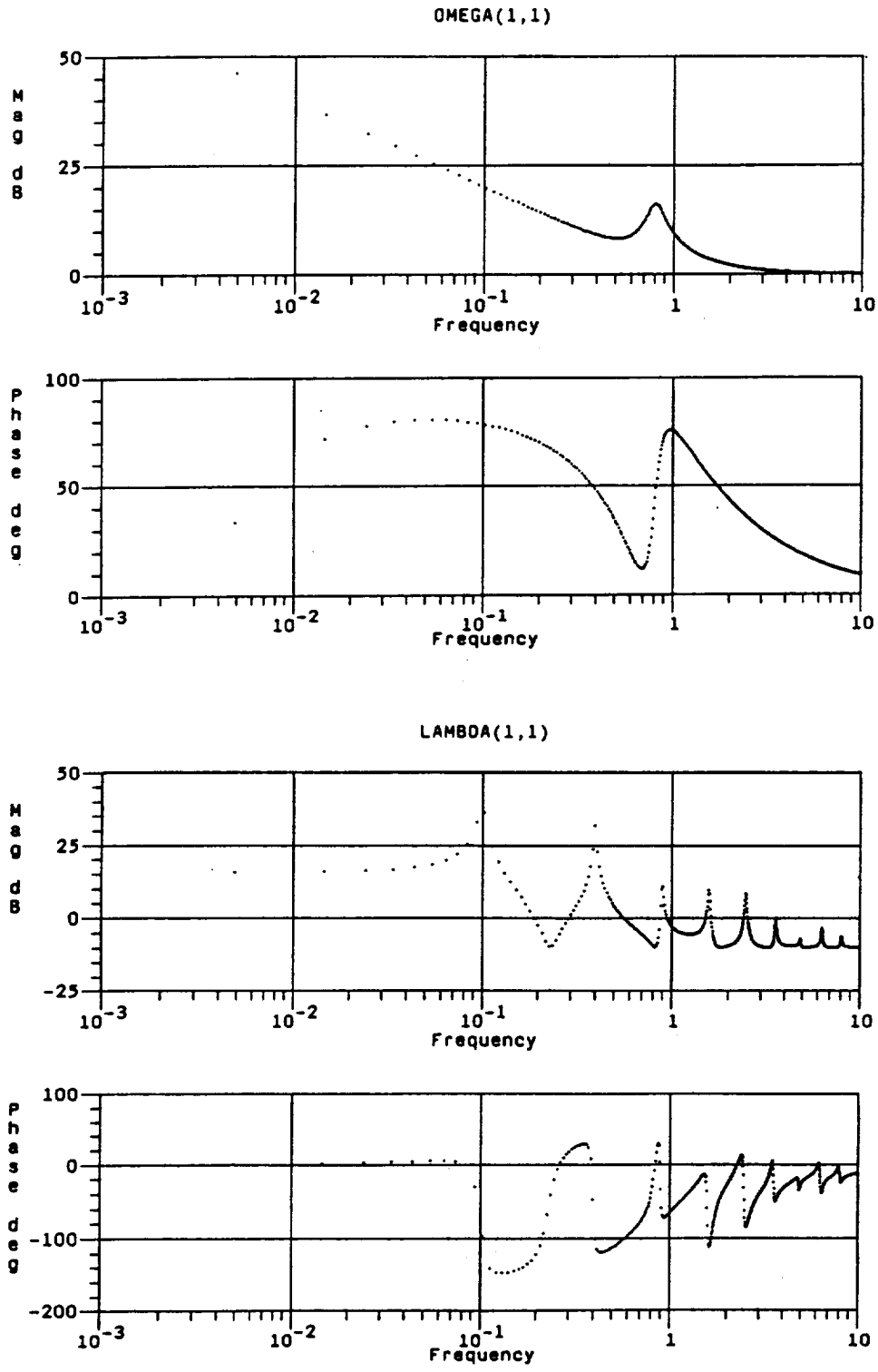


Figure 4.3: Spectral Factors for pinned-pinned beam control.

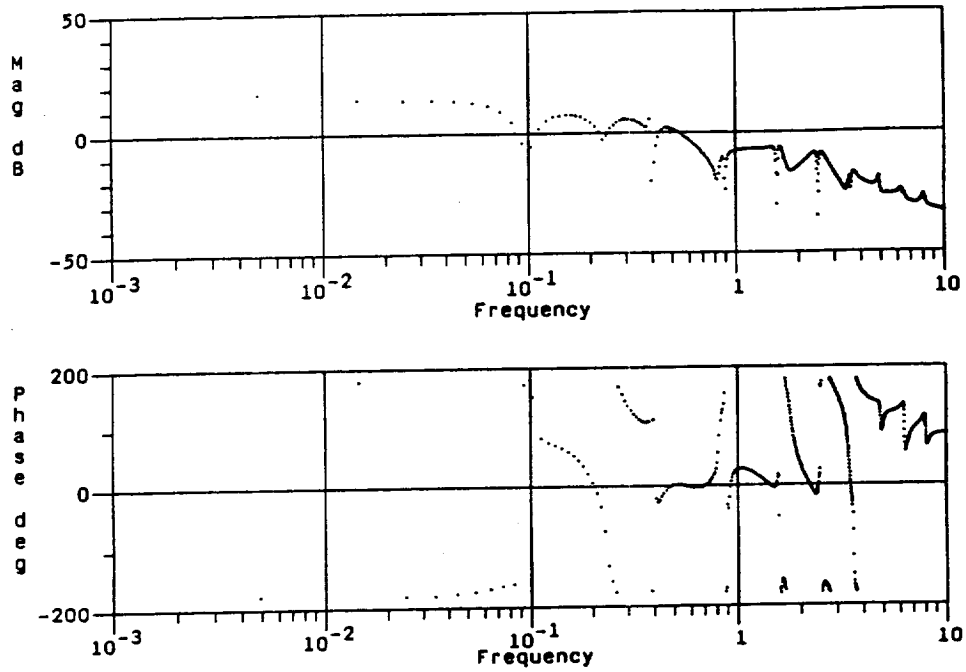


Figure 4.4: Optimal Controller Frequency Response for $\mu = .1$.

realizes the frequency sampling filter is

$$C(z) = \sum_{k=0}^{N-1} C_k F_k(z)$$

where the interpolating functions are,

$$F_k(z) = \frac{1 - z^{-N}}{N(1 - e^{jk2\pi/N} z^{-1})},$$

with $k = 0, \ldots, N-1$. A standard computation shows that the frequency sampling filter has transfer function

$$C(z) = \frac{1 - z^{-N}}{N} \sum_{k=0}^{N-1} \frac{C_k}{1 - e^{jk2\pi/N} z^{-1}} = \sum_{\ell=0}^{N-1} c_\ell z^{-\ell} \tag{5.1}$$

where the coefficients,

$$c_\ell = \frac{1}{N} \sum_{k=0}^{N-1} C_k e^{j(2\pi/N)\ell k} \tag{5.2}$$

for $\ell = 0, \ldots, N-1$, are the *Inverse Discrete Fourier Transform (IDFT)* of the sequence $C_0, \ldots, C_{N-1}$. The final form in (5.1) shows that the realization is a FIR realization. Such realizations are nonrecursive and are efficiently implemented using high speed, single chip DSP processors which utilize highly pipelined architectures to achieve high throughput.

6 Conclusions and Directions

Wiener-Hopf optimization of frequency domain models has been shown to offer significant advantages for computation of precision controllers for irrational transfer functions arising in control of flexible structures. Computational algorithms for causal spectral factorization

and causal projection can be implemented based on frequency sampled representation of the model response. Such models can be obtained from transfer function models or from frequency response measurements of the controlled structure. Computations based on frequency response sampling have been demonstrated for irrational transfer function models arising in the control of flexible structures.

Requirements for precision control will involve frequency response models which are characterized by a large number of flexible modes within the control bandwidth. However, for control of relatively large, flexible space structures control bandwidth and resulting sampling requirements for discrete time control implementations are well within the state-of-the-art for high speed digital computers. Frequency sampling filters based on nonrecursive implementations can be efficiently implemented in modern DSP single chip processors for realtime control of such systems.

Application of FIR realizations for realtime, closed loop control have not received much consideration in the literature primarily due to increased phase lag by comparison with a recursive realization. However, the rapidly developing technology for realtime DSP using special purpose architectures offers throughput capabilities which may reduce the achievable computational delay to within acceptable limits for certain applications. In such cases, high order realizations may be feasible using nonrecursive implemenations which cannot be realized by recursive methods.

References

[1] W. H. Bennett, "Computation and implementation of precision control for flexible structures," in *Proc. Com Con 88*, Oct. 1988.

[2] C. Desoer, R.-W. Liu, J. Murray, and R. Saeks, "Feedback systems design: The fractional representation approach to analysis and synthesis," *IEEE Trans. on Auto. Cntrl.*, vol. AC-25, no. 3, pp. 399–412, 1980.

[3] M. Vidyasagar, *Control System Synthesis: A Factorization Approach.* MIT Press, 1985.

[4] J. S. Baras, "Complex variable methods in the control of distributed systems," 1985. Presented at NSF, SIAM, AFOSR Workshop on Control of Distributed Systems.

[5] D. Youla, H. Jabr, and J. J.J. Bongiorno, "Modern Wiener-Hopf design of optimal controllers–parts I and II," *IEEE Trans. on Auto. Cntrl.*, vol. AC-30, no. 7, pp. 3–13 and 319–338, 1976.

[6] K. Park and J. J.J. Bongiorno, "A general theory for the Wiener-Hopf design of multivariable control systems," *IEEE Trans. on Auto. Cntrl.*, vol. AC-34, no. 6, pp. 619–626, 1989.

[7] C. Desoer, R.-W. Liu, J. Murray, and R. Saeks, "Feedback system design: The fractional representation approach," *IEEE Trans. on Auto. Control*, vol. AC–25, pp. 399–412, 1980.

[8] C. Desoer and C. L. Gustafson, "Algebraic theory of linear multivariable feedback systems," *IEEE Trans. Auto. Cntrl.*, vol. AC–29, no. 10, pp. 909–917, 1984.

[9] W. Bennett and H. Kwatny, "Continuum modeling of flexible structures with application to vibration control," *AIAA J.*, September 1989. to appear.

[10] A. von Flotow, "Traveling wave control for large spacecraft structures," *AIAA J. Guidance*, vol. 9, no. 4, pp. 462–468, 1986.

[11] D. Miller, A. von Flotow, and S. R. Hall, "Active modification of wave reflection and transmission in flexible structures," in *Proc. 1987 Amer. Cntrl. Conf.*, pp. 13128–1324, 1987.

[12] G. Gu and P. Khargonekar, "Approximation of infinite-dimensional systems," *IEEE Tran. on Auto. Cntrl.*, vol. AC-34, no. 6, pp. 610–618, 1989.

[13] W. Bennett and N. Barkakati, "FlexCAD: Prototype software for modeling and control of flexible structures," in *Proc. IEEE CACSD Symp.*, Sept. 1986.

[14] W. Bennett and I. Yan, "A computer algorithm for causal spectral factorization," in *Proc. 1988 NAECON*, May 1988.

[15] F. Stenger, "The approximate solution of Wiener-Hopf integral equations," *J. Math. Analysis and Applic.*, vol. 37, pp. 687–724, 1979.

[16] J. Davis and R. Dickinson, "Spectral factorization by optimal gain iteration," *J. Appl. Math.*, vol. 43, pp. 389–301, 1983.

Efficient Computer Algebra Algorithms for Polynomial Matrices in Control Design

J. S. Baras
D. C. MacEnany
R. Munach

Abstract

The theory of polynomial matrices plays a key role in the design and analysis of multi-input multi-output control and communications systems using frequency domain methods. Examples include coprime factorizations of transfer functions, canonical realizations from matrix fraction descriptions, and the transfer function design of feedback compensators. Typically, such problems abstract in a natural way to the need to solve systems of Diophantine equations (the so-called generalized Bezout equations) or systems of linear equations over polynomials. These and other problems involving polynomial matrices can in turn be reduced to polynomial matrix triangularization procedures, a result which is not surprising given the importance of matrix triangularization techniques in numerical linear algebra. There, we deal with matrices with entries from a field and Gaussian elimination plays a fundamental role in understanding the triangularization process. In the case of polynomial matrices we are dealing with matrices with entries from a ring for which Gaussian elimination is not defined and triangularization is accomplished by what is quite properly called Euclidean elimination.

Unfortunately, the nmerical stability and sensitivity issues which accompany floating point approaches to Euclidean elimination are not very well understood at present. In this paper we present new algorithms which circumvent entirely such numerical issues through the use of exact, symbolic methods in computer algebra. The use of such error-free algorithms guarantees that the results are accurate to within the precision of the model data—the best that can be hoped. Care must be taken in the design of such algorithms due to the phenomenon of *intermediate expression swell*, the price paid for

Integrated Control-System Design via Generalized LQG (GLQG) Theory

Dr. Dennis S. Bernstein
Dr. David C. Hyland
Dr. Stephen Richter
Harris Corporation

Prof. Wassim M. Haddad
Department of Mechanical and Aerospace Engineering
Florida Institute of Technology

Abstract

Thirty years of control systems research has produced an enormous body of theoretical results in feedback synthesis. Yet such results see relatively little practical application, and there remains an unsettling gap between classical single-loop techniques (Nyquist, Bode, root locus, pole placement) and modern multivariable approaches (LQG and H∞ theory). Large scale, complex systems, such as high performance aircraft and flexible space structures, now demand efficient, reliable design of multivariable feedback controllers which optimally tradeoff performance against modeling accuracy, bandwidth, sensor noise, actuator power, and control law complexity. This presentation will describe a methodology which encompasses numerous practical design constraints within a single unified formulation. The approach, which is based upon coupled systems of modified Riccati and Lyapunov equations, encompasses time-domain linear-quadratic-Gaussian theory and frequency-domain H theory, as well as classical objectives such as gain and phase margin via the Nyquist circle criterion. In addition, this approach encompasses the optimal projection approach to reduced-order controller design. The current status of the overall theory will be reviewed including both continuous-time and discrete-time (sampled-data) formulations. The presentation will focus on the

Modern CACSD using the Robust-Control Toolbox

Richard Y. Chiang and Michael G. Safonov
Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, CA. 90089-0781

Abstract

The *Robust-Control Toolbox* [1] is a collection of 40 "M-files" which extend the capability of PC/PRO-MATLAB to do modern multivariable robust control system design. Included are robust analysis tools like singular values and structured singular values, robust synthesis tools like continuous/discrete H^2/H^∞ synthesis and LQG Loop Transfer Recovery methods and a variety of robust model reduction tools such as Hankel approximation, balanced truncation and balanced stochastic truncation, etc.

In this paper, we will describe the capabilities of our toolbox and illustrate them with examples to show how easily they can be used in practice. Examples include structured singular value analysis, H^∞ loop-shaping and large space structure model reduction.

1 Introduction

The fundamental issue in robust control theory – *to find a stabilizing controller that achieves feedback performance despite the plant uncertainty*, is still the same issue addressed by the classical 1930's feedback theory of Black, Bode and Nyquist (ref. Fig. 1.1). Modern robust control theory has resolved many of the issues concerning the "gap" between the theory and practice that had grown to troublesome proportions in the 1970's. One key to bridging the "gap" has been the singular value Bode plat. Recent progress in Structured Singular Value (SSV), H^∞ optimal control theory and the model reduction techniques utilizing singular values have made the modern robust control theory *highly practical.*

The inavailability of quality software implementing the techniques of robust control theory has, until very recently, significantly limited the access of both researchers and engineering practitioners to these techniques.

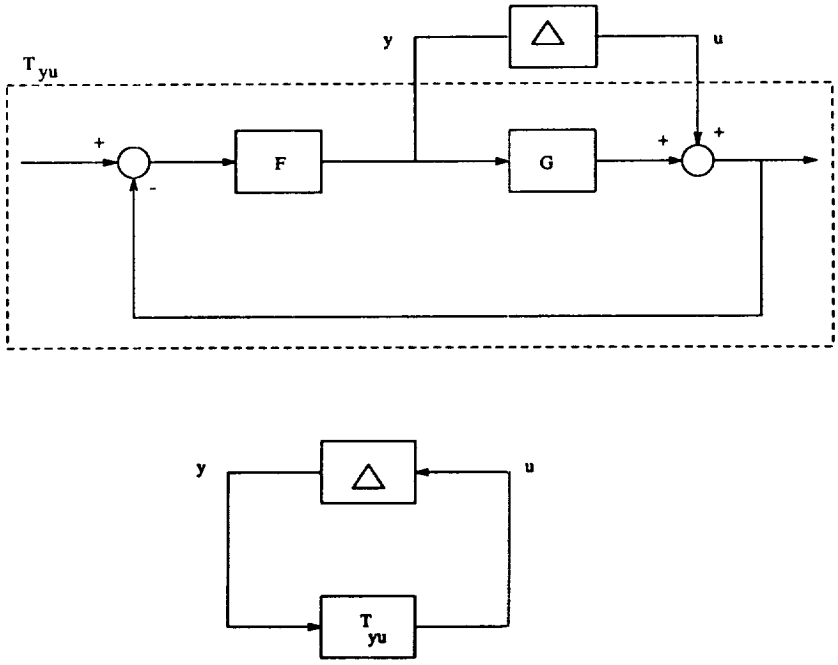


Fig. 1.1 Robust Control Problem.

Our *Robust-Control Toolbox* implements the most up-to-date robust control theory like Perron SSV, optimal descriptor 2-Riccati H^∞ formulae, LQG/LTR and singular value based model reduction techniques, ..., etc. The toolbox consists of a library of 40 functions which extend the capabilities of $PC/PRO - MATLAB^{TM}$ and the $PC/PRO - MATLAB$ *Control Toolbox*. The toolbox itself represents four man years research work done at USC.

These 40-functions can be catalogued into 3 major areas:

- **Robust Analysis**
  - Singular Values
  - Characteristic Gain Loci
  - Structured Singular Values
- **Robust Synthesis**
  - LQG/LTR, Frequency-Weighted LQG
  - H^2, H^∞
- **Robust Model Reduction**
  - Optimal Descriptor Hankel (with Additive Error Bound)
  - Schur Balanced Truncation (with Additive Error Bound)

– Schur Balanced Stochastic Truncation (with Multiplicative Error Bound)

In this paper, we will highlight the most important functions in the toolbox, demonstrate how easily they can be used and show what kind of results can be achieved with practical problems. As for the details of the modern robust control theory, they can be found in [7], [2] and the references therein.

2 Robust Analysis

The objective of robust analysis is to find a proper measure of the multivariable stability margin (MSM) against uncertainty. Uncertainty may take many forms, but among the most significant ones are noise/disturbance signals, transfer function modeling errors and unmodeled nonlinear dynamics, etc. Uncertainty in any form is no doubt the major issue in most control system designs.

Several tools to measure MSM are available: [1], [8]

- Singular values (Safonov, 1977; Doyle, 1978)
- Perron eigenvalues (Safonov, 1982)
- Diagonal scaling via nonlinear programming (Doyle, 1982; Tekawy et al., 1989)

Let's define the MSM first:

Definition 1 *Multivariable Stability Margin (MSM)*K_m, $\mu(\cdot)^{-1}$

$$K_m(T_{yu}) \triangleq \mu(T_{yu})^{-1} = \inf_{\Delta}\{\overline{\sigma}(\Delta)|det(I - T_{yu}\Delta) = 0\}.$$

In other words, it's the smallest $\overline{\sigma}(\Delta)$ that can make the determinant $(I - T_{yu}\Delta)$ singular (or the closed-loop system unstable). See Fig. 1.1.

A theorem summarizes the whole MSM idea:

Theorem 1 *The system is stable for all stable* Δ_i *with* $\|\Delta_i\|_\infty < 1$, *if the MSM* $K_m(T_{yu})> 1$.

Unfortunately, exact computation of K_m (or μ^{-1}) would require solution of a non-convex optimization problem and is therefore impractical. Fortunately, computable upper bounds on K_m are available, viz.,

$$K_m^{-1}(T_{yu}) = \mu(T_{yu}) \leq \inf_{D\in\mathcal{D}} \|DT_{yu}D^{-1}\|_\infty \leq \inf_{D\in\mathcal{D}} \|Dabs(T_{yu})D^{-1}\|_\infty \ll \|T_{yu}\|_\infty$$

where $\mathcal{D} := \{diag(d_1 I, \ldots, d_n I)|d_i > 0\}$.

Then using these upper bounds (some may be more conservative than others), one can assure that the system is stable against the norm-bounded uncertainty $\|\Delta\|_\infty \leq K_m$.

A comparison of the available upper bounds reveals that some are much easier to compute than others. See table 2.1.

Table 2.1

Method	Property	Computation	Reference
Optimal Diagonal Scaling	$n = 3$, exact K_m $n > 3$, $\exists$ 15 % gap	demanding	Doyle, 1982 Tekawy et al., 1989
Perron Eigenvector Diagonal Scaling	very close to optimal diagonal scaling	easy	Safonov, 1982
Singular Value	can be very conservative	easy	Safonov, 1977 Doyle, 1978

Let's see the following example.

Example: Given a system G(s) with multiplicative uncertianty at its input. Find the MSM.

$$G(s) = \begin{bmatrix} \frac{4}{s+4} & 0 \\ \frac{4s}{s+4} + \frac{8s}{s+8} & \frac{8}{s+8} \end{bmatrix}$$

Theorem 1 implies $\|\Delta\|_\infty \leq (\|G(I+G)^{-1}\|_\infty)^{-1}$.

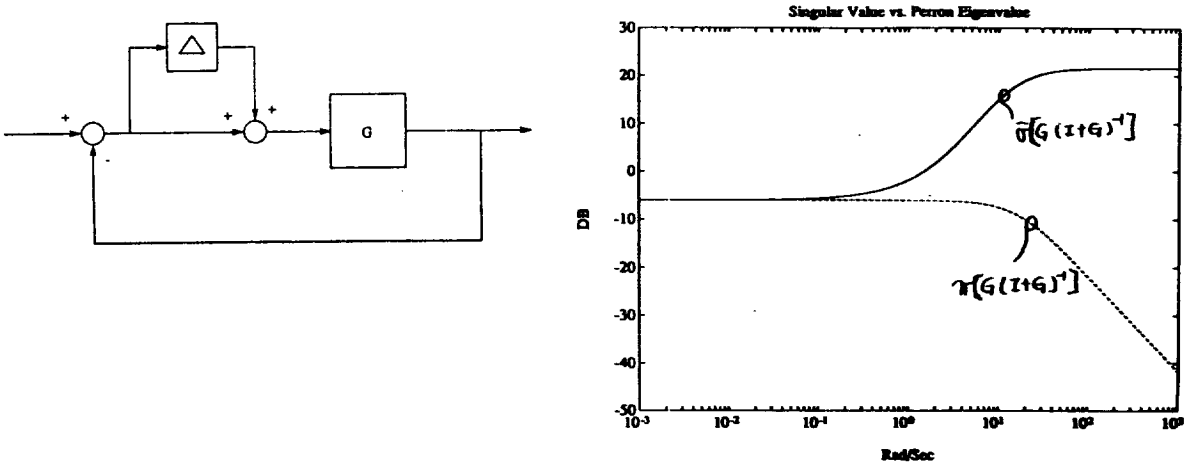


Fig. 2.2 Singular Value vs. K_m (upper bound).

This example reveals that the singular value upper bound is too conservative in "robust analysis". Whereas, Perron SSV is much simpler to compute than diagonally scaled nonlinear programming μ.

3 Robust Synthesis

Classical control system designers often do "loop-shaping" to meet design specifications. So do modern robust control system designers. "Loop-shaping" for mul-

tivariable systems is done via the singular-value Bode plot. However, to shape the *loop transfer function* $L(s)$ is nothing but to shape the *sensitivity function* $S(s) = (I + L(s))^{-1}$ and the *complementary sensitivity function* $T(s) = L(s)(I + L(s))^{-1}$. See Fig. 3.1

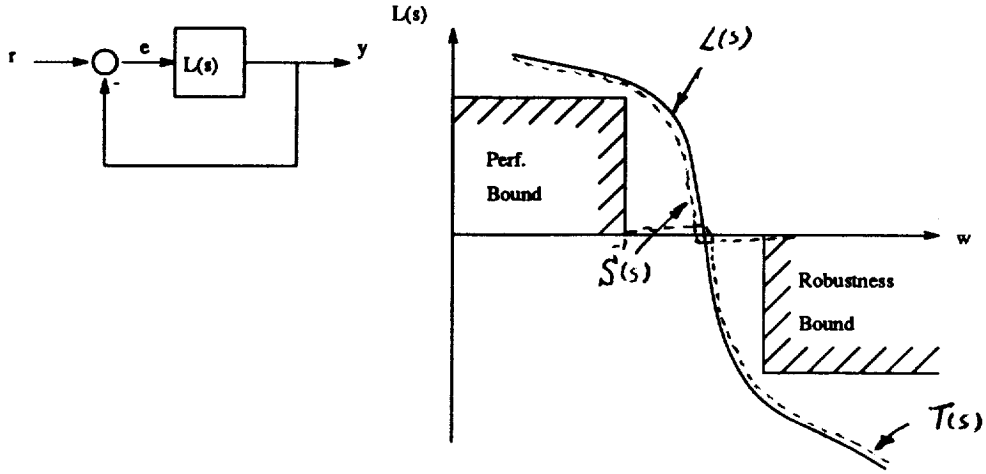


Fig. 3.1 SISO & MIMO Loop-Shaping.

There are several loop-shaping methods available in the *Robust-Control Toolbox* (see Table 3.1), but $\mathbf{H}^\infty$ is one of our favorites.

Table 3.1

Methods	Advantages	Disadvantages
LQR (lqr.m)	• guaranteed stability margin • pure gain controller	o need full-state feedback o need accurate model o possibly many iterations
LQG (lqg.m)	• use available noise data	o no stability margin guaranteed o need accurate model o possibly many iterations
LQG/LTR (ltru.m,ltry.m)	• guaranteed stability margin • systematic design procedure	o high gain controller o possibly many iterations o design focus on one point
$\mathbf{H}^2$ (h2lqg.m)	• address stability and sensitivity • almost exact loop shaping • closed-loop always stable	o possibly many iterations
$\mathbf{H}^\infty$ (hinf.m)	• address stability and sensitivity • exact loop shaping • direct one-step procedure	

Example: Classical loop shaping vs. $\mathbf{H}^\infty$ for 2nd order low-damped system.

Given a plant $G(s)$ which is 2nd order with damping 0.05 at 20 rad/sec, find a controller to meet frequency response Bode plot (see Fig. 3.2)

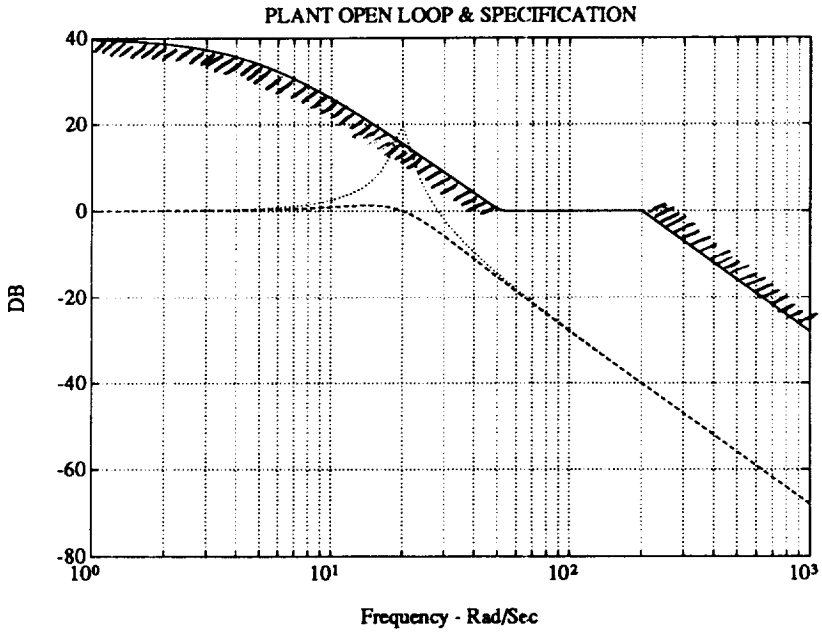


Fig. 3.2 2nd order plant open loop ($\zeta : 0.05, 0.5$) and the $L(s)$ spec.

A classical design might be decomposed into the following: (see Fig. 3.3)

Step 1: Rate feedback to improve damping.

Step 2: Design high frequency (phase margin, BW, roll-off..).

Step 3: Design low frequency (DC gain, disturbance rejection..).

The classical result is shown in Fig. 3.4. Now, let's see how $\mathbf{H}^\infty$ approaches the problem.

$\mathbf{H}^\infty$ Problem Formulation

We are solving the so-called $\mathbf{H}^\infty$ Small-Gain Problem ([3]) using the numerically robust "optimal" descriptor 2-Riccati formulae of Safonov, Limebeer and Chiang [4] [5].

$\mathbf{H}^\infty$ Small-Gain Problem:

Given a plant $P(s)$ (ref. Fig. 3.5), find a stabilizing controller $F(s)$ such that the closed-loop transfer function $T_{y_1 u_1}$ is internally stable and its infinity-norm is less than or equal to one.

But what makes $\mathbf{H}^\infty$ work is its unique and remarkable "all-pass" property:

At $\mathbf{H}^\infty$ optimal, the frequency response of $T_{y_1 u_1}$ is all-pass and equal to one (i.e., $\|T_{y_1 u_1}\| \cong 1$)!

This means that designers can achieve EXACT frequency domain loop-shaping via suitable weighting strategies. For example, one may augment the plant with frequency

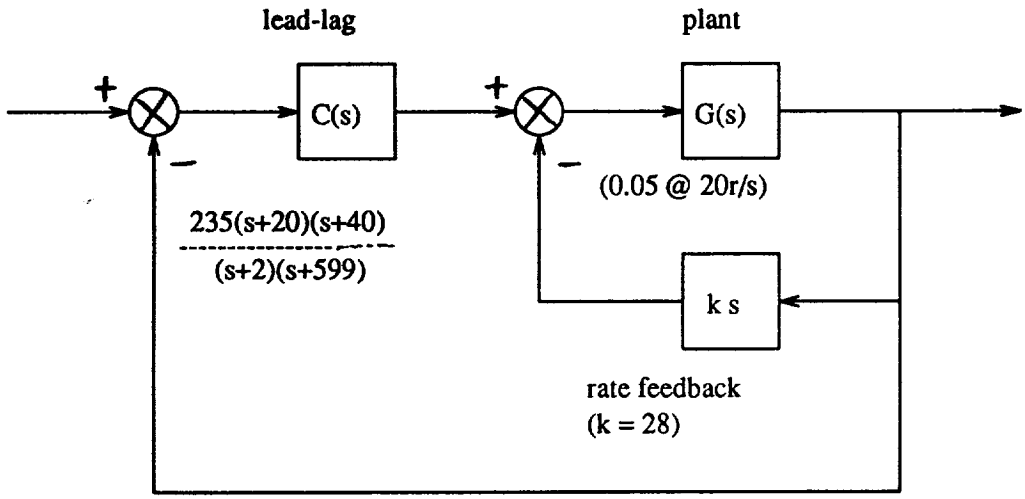


Fig. 3.3 Classical loop-shaping block diagram.

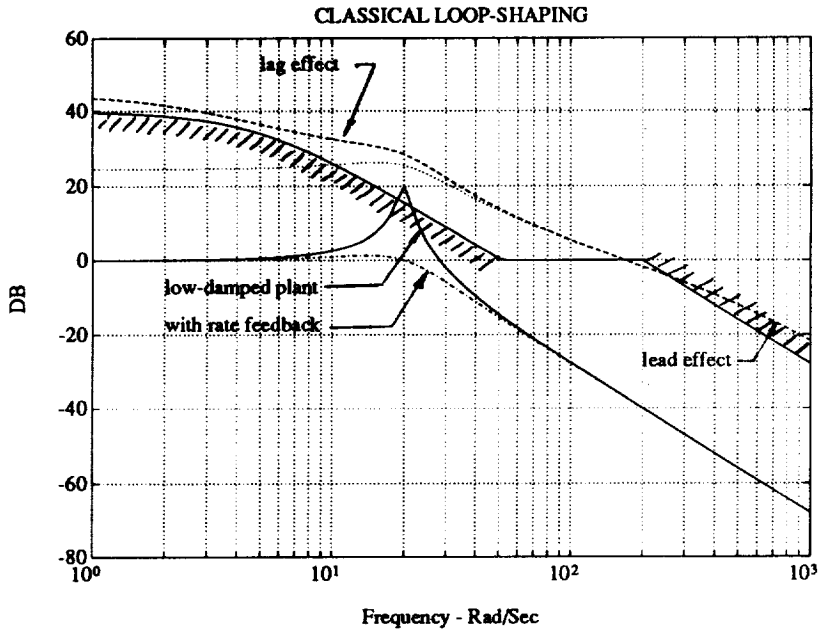


Fig. 3.4 Classical loop-shaping.

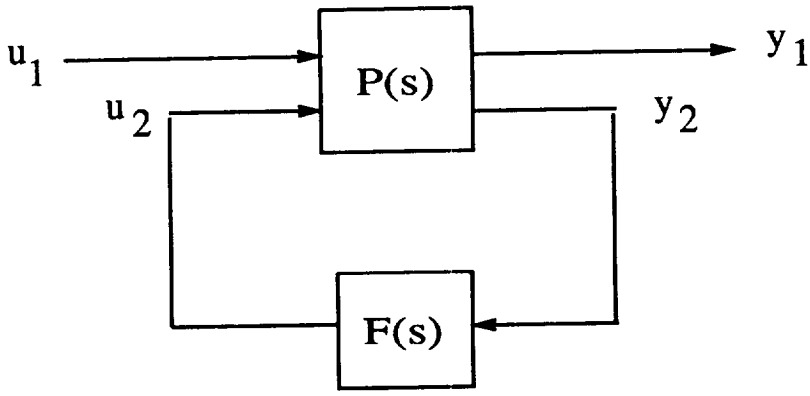


Fig. 3.5 H^{∞} Small-Gain Problem.

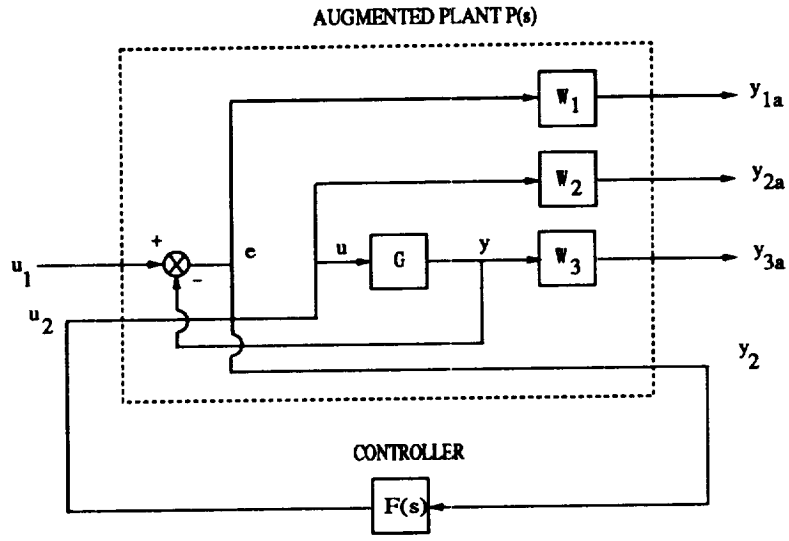


Fig. 3.6 Weighting strategy block diagram.

dependent weights W_1, W_2 and W_3 as shown in Fig. 3.6. Then, the *Robust-Control Toolbox* functions **augtf.m** and **augss.m** will perform the augmentation and create a state-space for function **hinf.m** to find an H^∞ controller. Of course, these frequency weighting functions have to be chosen so that a stabilizing solution satisfying the H^∞ norm constraint exists.

In a typical application, either $W_2(s)$ or $W_3(s)$ would be absent, leading to weighted H^∞ costs of the forms

$$\min_{F(s)} \left\| \begin{bmatrix} W_1 S \\ W_3 T \end{bmatrix} \right\|_\infty \leq 1 \qquad or \qquad \min_{F(s)} \left\| \begin{bmatrix} W_1 S \\ W_2 F S \end{bmatrix} \right\|_\infty \leq 1.$$

In our example, the frequency domain spec. can be split into W_1 and W_3:

$$W_1^{-1} = \rho \frac{(0.2s+1)^2}{100(0.005s+1)^2}; \qquad W_3^{-1} = \frac{40000}{s^2}$$

as shown in Fig. 3.7.

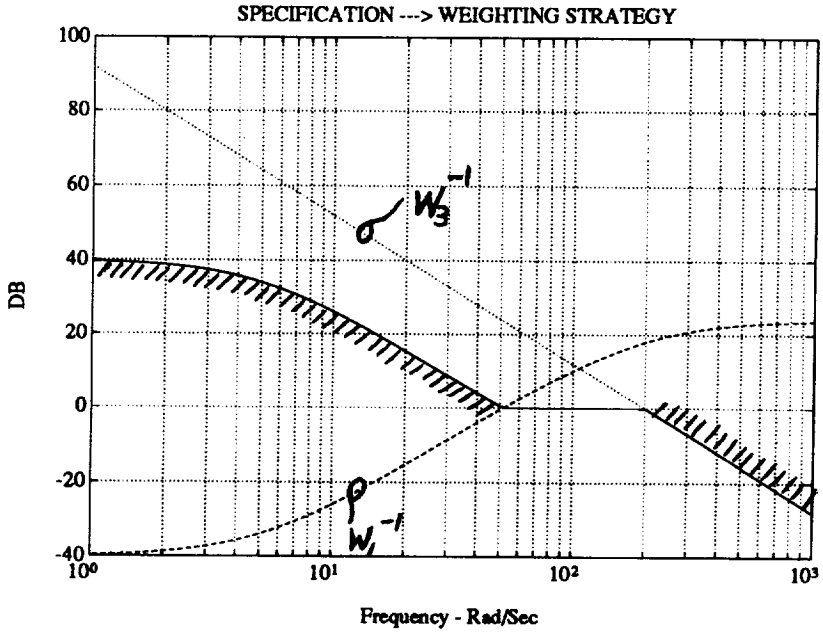


Fig. 3.7 Weighting strategy for 2nd order problem spec.

The results are shown in Fig. 3.8 for different ρ's. Clearly, in the limit (as ρ goes to 3.16) the cost function becomes "all-pass". The parameter ρ of W_1 is the only parameter on which we iterate for design; the *Robust-Control Toolbox* script-file **hinfgama.m** automates this iteration.

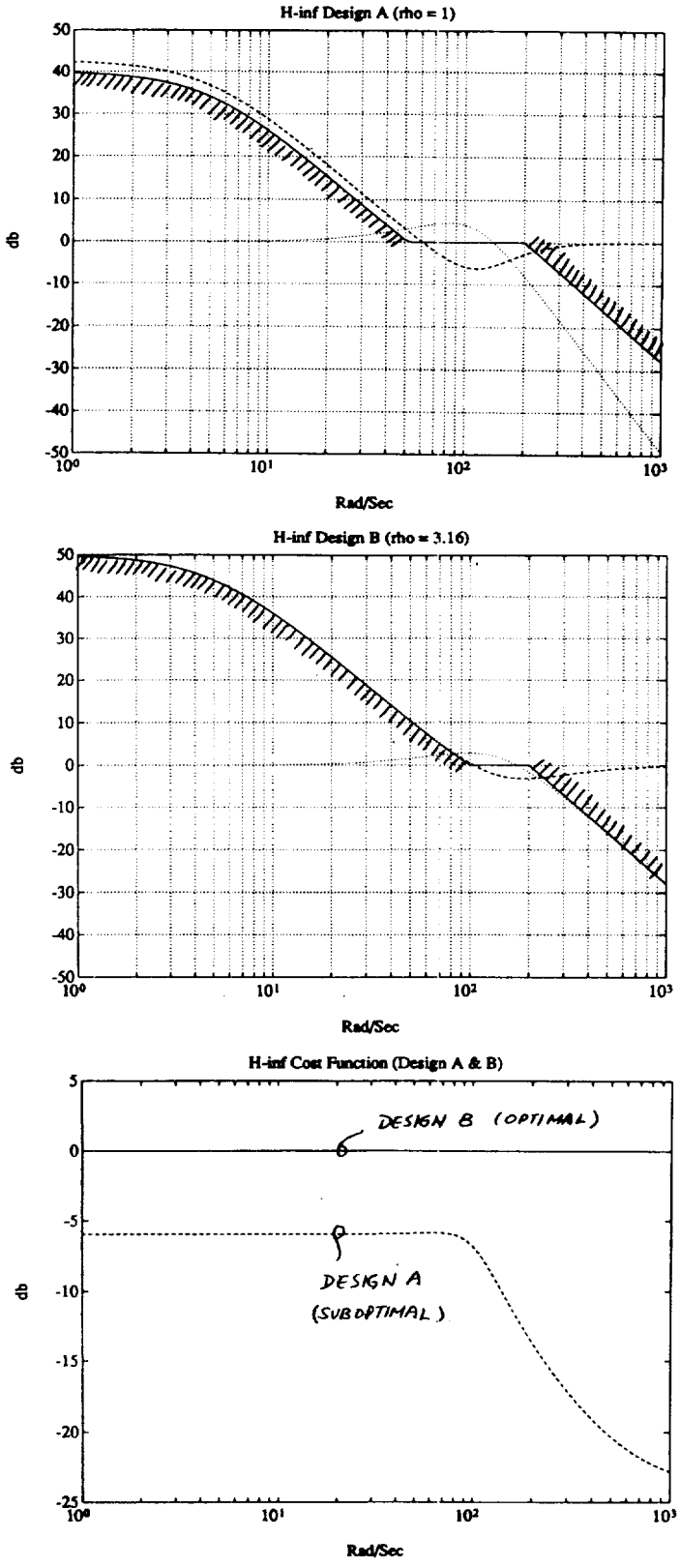


Fig. 3.8 H^{∞} results for 2nd order system

3.1 H^∞ Software Execution and Sample Run

To do an H^∞ control design with the *Robust-Control Toolbox* is relatively simple. Table 3.2.1 shows the complete user inputs for our sample problem.

Table 3.2.1

```
>> nug = [0 0 400]; dng = [1 2 400];
>> [ag,bg,cg,dg] = tf2ss(nug,dng);
>> sysg = [ag bg;cg dg]; xg = 2;
>> w1 = [2.5e-5 1.e-2 1;
     0.01*[4.e-2 4.e-1 1]];
>> w2 = [ ];
>> w3 = [1 0 0;0 0 40000];
>> [A,B1,B2,C1,C2,D11,D12,D21,D22] = augtf(sysg,xg,w1,w2,w3);
>> hinf
```

Table 3.2.2 shows the output which appears on the screen for a successful run of hinf.m.

Table 3.2.2

```
              << H-inf Optimal Control Synthesis >>

          Computing the 4-block H-inf optimal controller
        using the S-L-C loop-shifting/descriptor formulae

- - - Solving for the H-inf controller F(s) using U(s) = 0 (default)  - - -

   Solving riccati equations and performing H-infinity
   existence tests:
      1.  Is D11 small enough?                          OK
      2.  Solving state-feedback (P) riccati ...
           a.  No Hamiltonian jw-axis roots?            OK
           b.  A-B2*F stable (P >= 0)?                  OK
      3.  Solving output-injection (S) riccati ...
           a.  No Hamiltonian jw-axis roots?            OK
           b.  A-G*C2 stable (S >= 0)?                  OK
      4.  max eig(P*S) < 1 ?                            OK
  ------------------------------------------------------
     all tests passed -- computing H-inf controller ...
                       DONE!!!
  ------------------------------------------------------
```

4 Robust Model Reduction via Basis-Free Techniques

In the design of controllers for complicated systems, model reduction arises in several places: 1). Plant model reduction, 2). Controller model reduction, 3). Simulation of large size problem.

However, naive implementations of model reduction methods such as Rosenbrock's stair-case algorithm, Moore's balanced truncation, optimal Hankel approximation and balanced stochastic truncation, etc., can fail on even relatively simple problems due to numerical instability.

The *Robust-Control Toolbox* implements the "basis-free" version of the latter three of the model reduction techniques, which are not only numerically robust but also tend to achieve the ultimate result for practical problems. In particular, they all possess the following special features:

1. They bypass the ill-conditioned balancing transformation, so that they can easily deal with the "non-minimal" systems.

2. They employ Schur decomposition to robustly compute the orthogonal bases for eigenspaces required in intermediate steps.

These methods all enjoy attractive $\mathbf{L}^\infty$ error bounds – either an *additive error bound* or a *multiplicative error bound*.

- Additive Methods:
  - Optimal descriptor Hankel MDA (**ohklmr.m**).
  - Schur balanced truncation (**balmr.m, schmr.m**).
- Multiplicative Method:
  - Schur balanced stochastic truncation (**reschmr.m**).

4.1 Robust Model Reduction Theorems

For robust control system design, it is desirable that the reduced order model satisfies the conditions of one of the following two theorems (ref. [1] [2]). Otherwise, the controller design based on the "blind"reduced order plant model can be unstable !

Theorem 2 Additive Robustness Theorem: *(see Fig. 4.1)*
If $\bar{\sigma}(\tilde{\Delta}_A) \leq \underline{\sigma}(\tilde{G})$ *for* $\omega \leq \omega_r$ *(with* $\tilde{\Delta}_A$ *and* $\tilde{G}$ *open loop stable), then the closed-loop system will be stable provided that the control bandwidth is less than* ω_r*, where* $\omega_r := max\{\omega \mid \underline{\sigma}(\tilde{G}(j\omega)) \geq \bar{\sigma}(\tilde{\Delta}_A(j\omega))\}$.

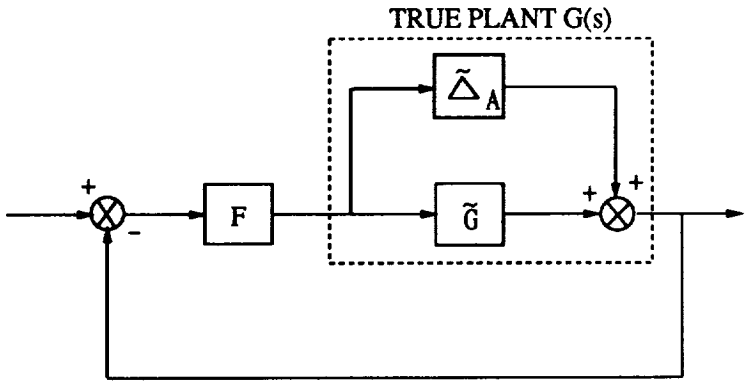


Fig. 4.1 Additive modeling error.

Theorem 3 Multiplicative Robustness Theorem: *(see Fig. 4.2)*
If $\bar{\sigma}(\tilde{\Delta}_M) \leq 1$ *for* $\omega \leq \omega_r$ *(with* $\tilde{\Delta}_M$ *and* $\tilde{G}$ *open loop stable), then the closed-loop system will be stable provided that the control bandwidth is less than* ω_r*, where* $\omega_r := max\{\omega \mid \bar{\sigma}(\tilde{\Delta}_M(j\omega)) \leq 1\}$.

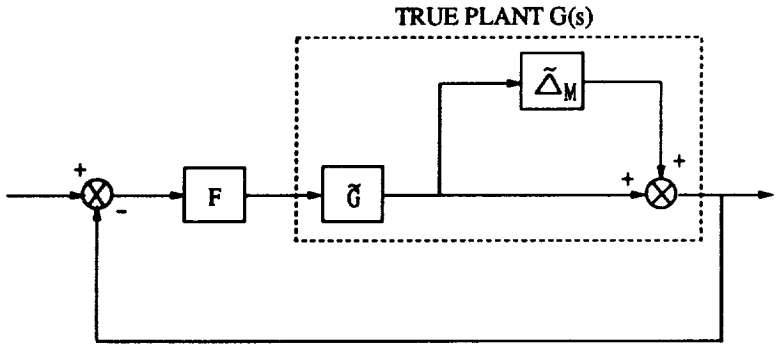


Fig. 4.2 Multiplicative modeling error.

4.2 Examples of Model Reduction

Example 1: Find a 3-state reduced order model for the transfer function

$$G(s) = \frac{0.05(s^7 + 801s^6 + 1024s^5 + 599s^4 + 451s^3 + 119s^2 + 49s + 5.55)}{s^7 + 12.6s^6 + 53.48s^5 + 90.94s^4 + 71.83s^3 + 27.22s^2 + 4.75s + 0.3}$$

with Hankel singular values of the phase matrix

σ_1	σ_2	σ_3	σ_4	σ_5	σ_6	σ_7
0.9959	0.9972	0.9734	0.7166	0.5635	0.0021	0

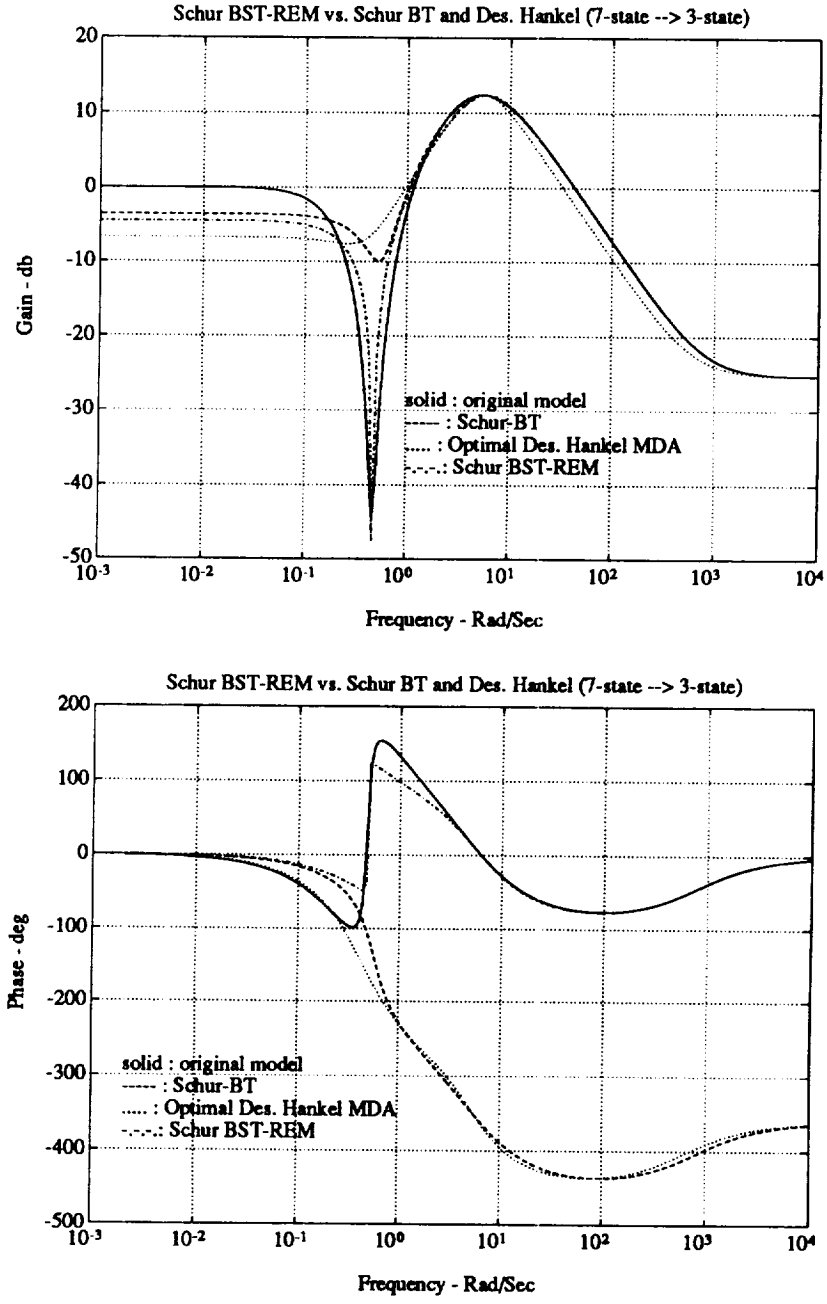


Fig. 4.3 Schur BST-REM vs. Schur BT and Descriptor Hankel.

The results produced by the *Robust-Control Toolbox* functions – ohklmr.m, schmr.m, reschmr.m, are shown in Fig. 4.3. Clearly, the Schur BST-REM method that keeps the reduced model staying inside a prescribed relative error bound produces the ultimate result in model reduction. Note that $\sigma_7 = 0$ indicates only the "basis-free" methods can handel the problem without numerical difficulty.

Example 2: Model reduction for a large space structure [6] (see Fig. 4.4).

Our design reequirement is to find a controller to track LOS loops in 300 Hz BW and to reduce plant disturbance response by a factor of 100.

The Hankel singular values after the inner loops are closed indicate that the system is non-minimal. Therfore, only the "basis-free" methods such as – Schur BT (schmr.m) or Schur BST-REM (reschmr.m) can be used. Again, only the Schur BST-REM can match the original model up to a "robust frequency" which is high enough so that the required BW of 300 Hz can be satisfied (see Fig. 4.5 & Fig. 4.6).

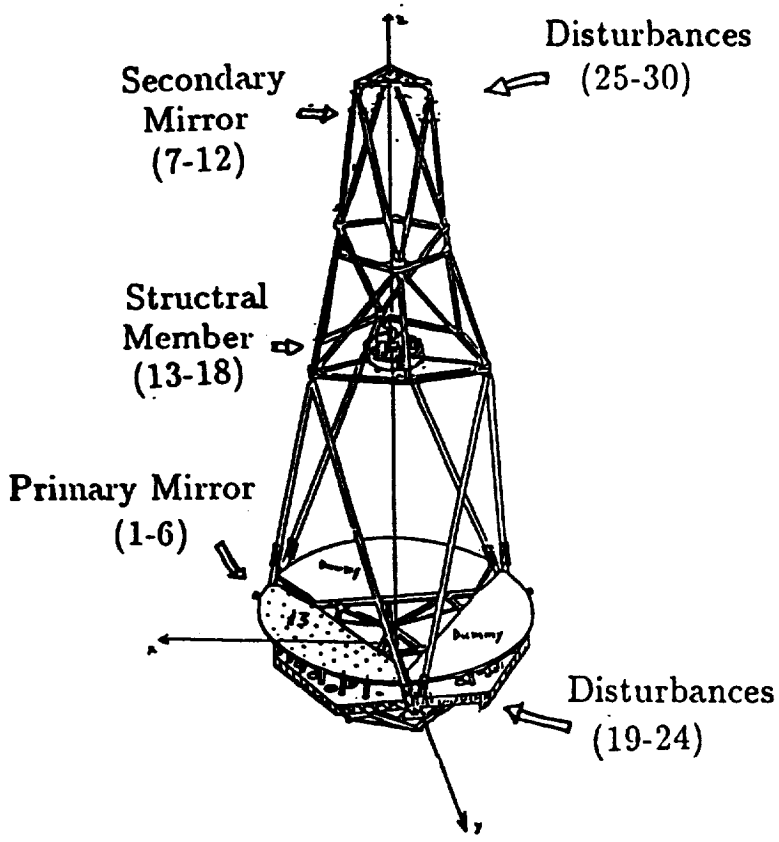


Fig. 4.4 Large space structure.

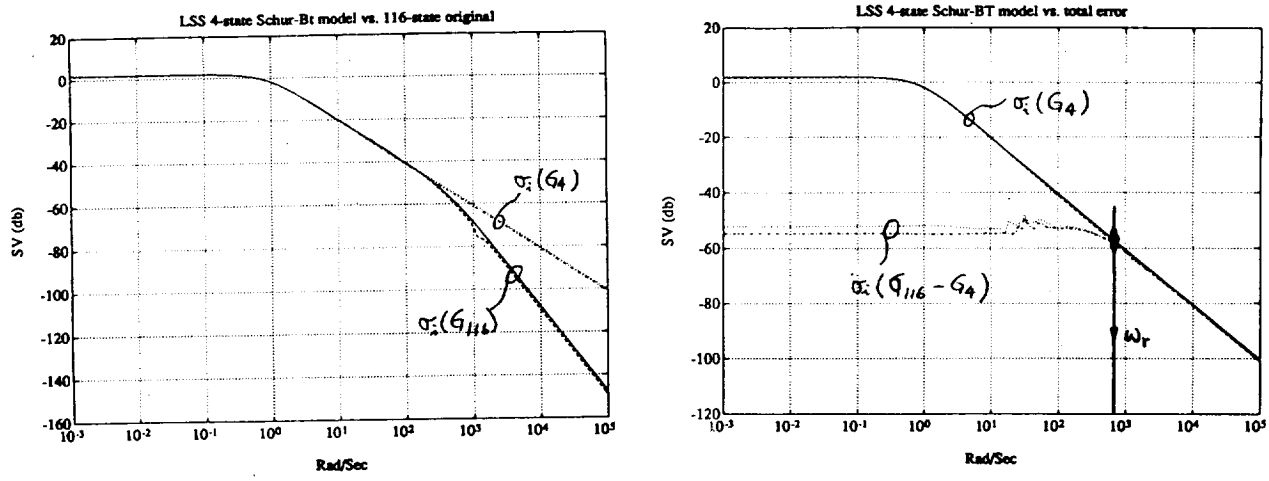


Fig. 4.5 Model reduction using Schur Balanced Truncation.

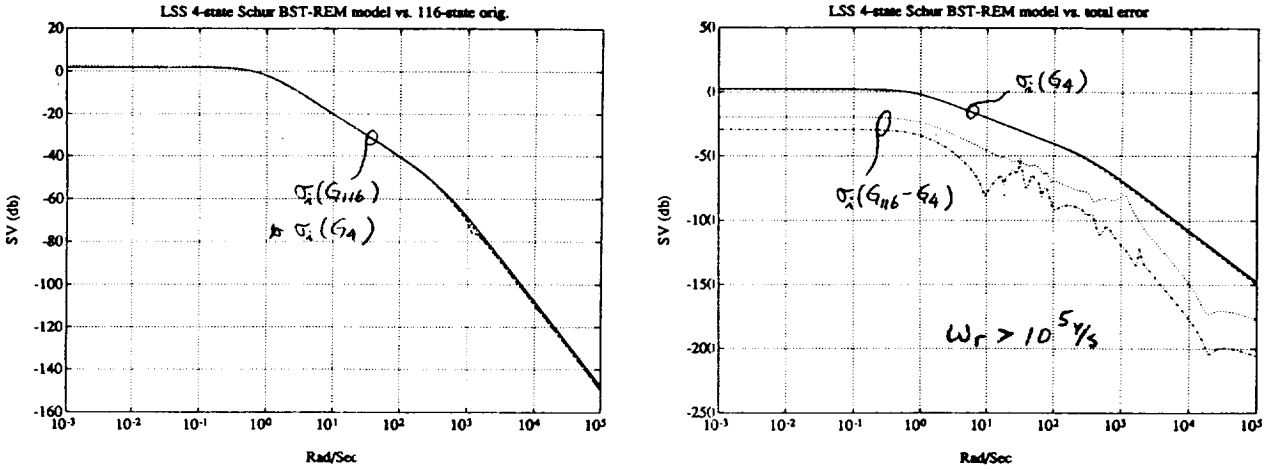


Fig. 4.6 Model reduction using Schur Balanced Stochastic Truncation.

5 Importance to the Control Community

If the ultimate goal of research is to apply the theory to reality, then the contribution of *Robust-Control Toolbox* is clear:
It provides a vital bridge between modern robust control theory and real control applications.

The following diagram (Fig. 5.1) shows how different groups of people with different backgrounds can utilize the *Robust-Control Toolbox* to achieve their personal goals. For example classical control designers can use the toolbox to understand the theory or to apply on a design. Non-robust control researchers can study the code provided by the toolbox together with the papers referenced therein to become familiar with the robust control theory. Students can use the toolbox either for robust control research or for realistic design studies. It seems that the *Robust-Control Toolbox* can serve people from a variety of backgrounds.

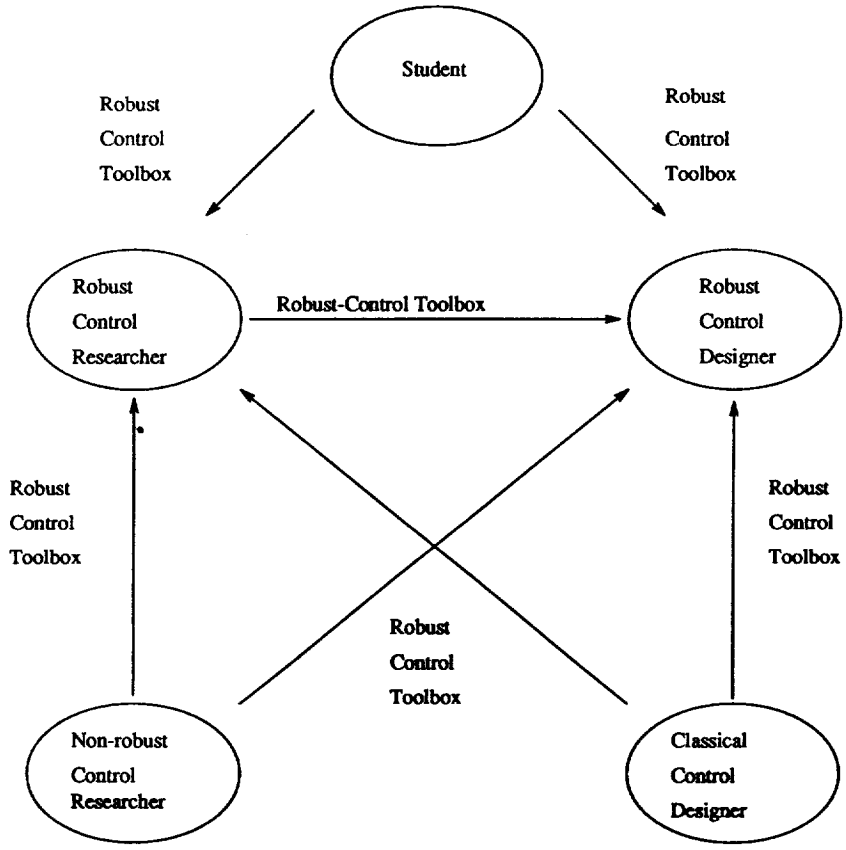


Fig. 5.1 The importance of the *Robust-Control Toolbox*.

6 Summary

The *Robust-Control Toolbox*

- Provides a bridge between modern robust control theory and the real-world applications.
- Has the most up-to-date Robust-Control theories and algorithms.
- Is in *readable* M-files, so it's educational.
- Contains the tools one needs to do robust control system **design, analysis** and **model reduction.**
- Is direct, powerful and easy-to-use.

References

[1] R. Y. Chiang and M. G. Safonov, *Robust-Control Toolbox.* So. Natick, MA: MathWorks, 1988.

[2] R. Y. Chiang, *Modern Robust Control Theory.* Ph.D. dissertation, USC, 1988.

[3] M. G. Safonov and R. Y. Chiang, "CACSD using the State-Space L^∞ Theory - A Design Example", *IEEE Trans. on Automat. Contr*, vol. AC-33, no. 5, pp. 477-479, 1988.

[4] M. G. Safonov and D. J. N. Limebeer, "Simplifying the H^∞ Theory via Loop-Shifting", *Proc. IEEE Conf. on Decision and Control*, Austin, TX, Dec. 7-9, 1988.

[5] M. G. Safonov, D. J. N. Limebeer and R. Y. Chiang, "Simplifying the H^∞ Theory via Loop-Shifting, Matrix Pencil and Descriptor Concepts", to appear *Int. J. Control* 1989.

[6] M. G. Safonov, R. Y. Chiang and H. Flashner, "H^∞ Control Synthesis for a Large Space Structure," *Proc. of American Contr. Conf.*, Atlanta, GA. June 15-17, 1988. Submitted to *J. of Guidance and Control*, June, 1989.

[7] M. G. Safonov, *Robustness and Stability Aspects of Stochastic Multivariable Feedback System Design*, Ph.D. dissertation, MIT, 1977. Also, M. G. Safonov, *Stability and Robustness of Multivariable Feedback Systems.* Cambridge, MA: MIT Press, 1980.

[8] J. Tekawy, M. G. Safonov and R. Y. Chiang, "Computer Algorithms for Multivariable Stability Margin", *Proc. of 3rd Annual Conference on Aerospace Computational Control*, Oxnard, CA, Aug. 28-30, 1989.

H^2- and H^∞-Design Tools for Linear Time-Invariant Systems

Uy-Loi Ly
Department of Aeronautics and Astronautics FS-10
University of Washington
Seattle, Washington 98195

Abstract

Recent advances in optimal control have brought design techniques based on optimization of H^2 and H^∞ norm criteria, closer to be attractive alternatives to single-loop design methods for linear time-invariant systems. Significant steps forward in this technology are the deeper understanding of performance and robustness issues of these design procedures and means to perform design trade-offs. However acceptance of the technology has been hindered by the lack of convenient design tools to exercise these powerful multivariable techniques, while still allowing single-loop design formulation. Presented in this paper is a unique computer tool for designing arbitrary low-order linear time-invariant controllers that encompasses both performance and robustness issues via the familiar H^2 and H^∞ norm optimization. Application to disturbance rejection design for a commercial transport is demonstrated.

I. Introduction

Past three decades have laid a foundation on the theory of optimal control. Issues have been actively pursued in algorithms for numerical solution of optimum designs, feedback properties of optimal linear feedback (and feedforward) controllers and associated theoretical results of existence and uniqueness. Filtering of these wealth of technology down to current practitioners have been agonizingly slow. Demonstration and acceptance of these design techniques in typical flight systems such as SAS (stability augmentation systems), manual controls and autopilot designs, are almost non-existent. Hindrances in this effort are related to concerns raised in the following areas: design simplicity, ease-of-modification during flight-test and incorporation of designers' intuition and experiences in these "optimum" systems. Presented in this paper is the development of a design tool that covers much of the advances in multivariable controls and its potential application to flight controls.

II. Background and Motivation

Historically multivariable controls have been extensively developed based on optimal control of linear time-invariant systems. Class of design problems addressed in the past are optimal linear regulator using full-state feedback or estimate-state output feedback. Research efforts to extend the usefulness of multivariable control designs within the reach of experienced control designers are concentrated in the following areas:

- Measures of design robustness in the presence of modeling uncertainties[†1-21],

---

† Numbers indicate references.

- Synthesis methods to achieve trade-offs in performance and robustness[7,22-35],
- Model and controller order reduction[36-39],
- Direct synthesis of reduced-order controllers of given structure to achieve trade-off in closed-loop performance and robustness, and at the same time facilitate design integration over different operating conditions[23].

A well-known property of feedback concept is its ability to regulate in the presence of plant uncertainties. Measures of robustness are traditionally based on loop stability margins[40] (i.e. phase and gain margins of individual control loops while maintaining other loops closed at nominal gains). These single-loop robustness tests provide useful design criteria for the evaluation of current flight control systems. Recent development in robustness analysis techniques allow designers to examine design sensitivity in the multiloop and multivariable settings. To detect conditions for design sensitivity, one makes use of the singular values of loop return-difference matrix at appropriate plant input/output locations. In addition μ-measure is defined to characterize design robustness to uncertainties that are expressed in terms of plant parameter variations or those that have a predetermined structure. The latter robustness measure, μ-measure, is difficult to evaluate exactly; but it provides the most accurate description of design robustness in the presence of structured uncertainties. Current research direction is to devise numerical schemes that approximate closely[12] (i.e. providing "least" conservative upper and lower bounds) or, exactly calculate the μ-function for some specific types of structured uncertainties[13-21]. Synthesis methods to improve design robustness based on a general μ-measure are not available.

Guaranteed robustness of (-6dB,∞) in gain margins and (-60°,60°) in phase margins from optimal linear regulator have motivated researchers in developing design procedures to retain or recover these robustness properties for state-estimate feedback controllers[7] (i.e. as in optimal linear quadratic gaussian (LQG) designs). Fundamental understanding in the robustness recovery process resulted in design methods classified under the category of LQG/LTR where, for example, loop properties of a full-state feedback regulator are asymptotically recovered with appropriate selection of process noise models[41], or in the framework of "asymptotic" pole assignment[42]. One possible drawback of these procedures is the tendency of having unnecessarily high gains in the estimator design during the loop transfer recovery.

Controllers obtained from a LQG/LTR design procedure are usually of high order (i.e. order of the controlled plant model augmented with models for actuation, sensor and design-shaping filter dynamics). Implementation of these controllers in digital flight processors may be feasible with anticipated advances in computing technology; but will undoubtedly be challenging from the point of view of design traceability, reliability and maintainability. However it is often possible to reduce the controller order using standard model reduction techniques such as modal residualization[43], balanced truncation[44] and optimal Hankel norm approximation[37]. Careful considerations must be made especially in the reduction of controllers so that closed-loop stability, performance and robustness are preserved. Frequency-weighted reduction schemes have been developed to address these issues with some success[39].

A remaining problem is the final integration of "suboptimal" reduced-order controllers to operate over a wide range of flight conditions. This is usually achieved in an ad-hoc manner (e.g. curve-fitting gain parameters optimized at individual flight conditions as a function of some physical quantities such as calibrated airspeed, aircraft weight, aircraft center of gravity and dynamic pressure).

With some of the above outstanding issues unresolved, it is evident that wide acceptance of these design techniques has been difficult. Additional reason behind this difficulty in technology transfer is

the lack of intuitiveness in the design approach to handle low-order controllers of given (i.e. predetermined) structures (e.g. washout filter in the yaw damper design of a lateral control system for turn-coordination, simple a-priori gain scheduling according to physical quantities such as aircraft weights, dynamic pressure and calibrated airspeed, synthesis of dedicated structural filters for control of lightly passively damped elastic modes, etc...).

Research in direct synthesis of reduced-order controllers for multivariable controls are being actively pursued and have resulted in some promising design algorithms both in the continuous-time[23,32] and discrete-time[45,46] domains. Although theoretical development of the design techniques have made significant strive, insights in applying them to the synthesis of practical flight control systems are yet to be established. To facilitate the evaluation and technology transfer of multivariable control design concepts, there is a need for an efficient and versatile design tool that is able to resolve the above issues related to design implementation, performance, robustness and integration (i.e. gain schedule) over a wide range of operating conditions.

III. Design Tool Development

The objective of the design tool is to provide a unified framework for applying recent advances in robust multivariable controls to flight systems. To achieve this goal, development of efficient and versatile computational algorithms is needed. Scope of the design concept and procedure will hopefully enable and motivate experienced designers to appreciate the importance and value of multivariable controls. Steps taken to accomplish the stated objectives are as follows:

- Formulation of control design problems and solution algorithms for the synthesis of robust low-order controllers,
- Implementation of design algorithms in useful CAD tools for ease in obtaining design solutions to a wide class of linear feedback/feedforward controllers,
- Ability to formulate other design specifications using linear and nonlinear equality and inequality constraints.

Control Design Problem: Multivariable controls have primarily focused on applying optimization to the design of control systems. Extensive work conducted to-date are on control of linear time-invariant systems. The problem is the synthesis of linear controllers that meet specific closed-loop performance and robustness over a range of linearized plant conditions. Surprisingly this problem is identical to the one that experienced designers have to confront in their daily design work where traditional single-input single-output (SISO) methods prevail. Inadequacies of these SISO design techniques are well-known: neglect of cross-coupling effects, difficulty to satisfy multiple design requirements, highly dependent on the designers' experience, trial-and-error. On the other hand, advantages behind SISO design procedures are: the simplicity of its final controller structure, the ease-of-incorporating designers' experience into the design process and design flexibility for post-flight test modification. A useful design tool would combine these existing SISO design features into multivariable control synthesis.

Design Procedure Based on H^2 and H^∞-Optimization : Design methods for multivariable controls can be categorized into two general classes depending on whether the control-laws are synthesized based on minimizing a performance measure while satisfying other design constraints, or just simply meeting design constraints (e.g. eigenstructure assignment).

In the category of performance-oriented methods, control algorithms are generally developed from optimization of some weighted norms of the plant outputs and control inputs in a closed-loop

system subject to deterministic or stochastic disturbances. Two commonly used measures are H^2 and H^∞-norms with interpretation in both frequency and time-domains. Until recently, and mostly for mathematical convenience, majority of feedback and feedforward control-law synthesis are based on H^2-norm. Associated design schemes are classified under the following methods: linear quadratic regulator (LQR), linear quadratic gaussian (LQG) design and, linear quadratic gaussian design with loop transfer recovery (LQG/LTR).

Over the past decade, practitioners of these techniques have gathered valuable experiences in applying these techniques to flight controls. Iterative procedures have been developed to achieve trade-offs between performance and control bandwidths[54,55]. However implementation of these designs remains an area of concern and need further research development. Often these designs with full-state feedback structure are implemented using a state estimator or observer. Procedures to obtain design robustness in state-estimate feedback are done through the mechanism of Riccati equation[41] or eigenvalue placement[42] starting from sufficient conditions for loop transfer recovery.

These procedures offer valuable insights into the design feasibility based on requirements of closed-loop stability, performance and robustness. Unfortunately, difficulty in extending these results to encompass traditional design philosophy (e.g. output feedback, low-order controller with structure intuitive to designers, gain scheduling,...) remains. Attempts to fit these multiloop high-order controllers into low-order and conceptually simple designs using, for example, controller order reduction are not trivial and have been partially successful[39,43]. This remains to be an area of continued research interests.

A completely different, direct and practical approach to multivariable controls would be via the route of parameter optimization. The control design procedure described in this paper is one of such methods. It is based on the optimization of an objective function using any pre-defined controller structure and subject to additional linear and nonlinear design constraints[56]. The formulation allows direct intervention of control designers through the set-up of the design objective function, the controller structure and constraints on closed-loop stability, performance and robustness to plant uncertainties. This design concept was originally developed in reference 23 for linear time-invariant systems using objective function based on H^2-norm. The design algorithm was efficiently implemented into the computer-aided-design package SANDY. Evaluation of the objective function and its gradients with respect to the controller parameters are performed analytically for a diagonalizable closed-loop system. Subsequent improvement have been made in the area of numerical optimization (e.g. found in the constrained optimization code NPSOL[56]), and in the development of typical constraints encountered in flight control systems such as closed-loop damping, covariance bounds on output and control variables. Encouraging results have been obtained in a variety of control design applications[22,26-28,31,48].

Reference 23 has also demonstrated the early application of such a technique in simple design situations. Later applications have been in the design of missile autopilot[27], design of a reliable stability and command augmentation system for a commercial transport[22,26], design of an improved lateral ride quality control system[48,49]. Usefulness of such a design tool has been further investigated in the control of a remotely-piloted vehicle[50,51] and nonrigid manipulators[52,53]. Reference 53 actually applied and verified in experiments results achieved using the design algorithm[23] to the synthesis of robust compensators for flexible structures with uncertain parameters.

This research has led to the development of a unified multivariable control design concept that addresses virtually all flight control design problems such as stability augmentation systems, gust load alleviation, manual and automatic control modes. Typical flight control systems can be formulated exactly in the same situation as designers would when conducting designs using single-

loop frequency-domain techniques. However, in the design solution, multivariable control methods based on H^2 and H^∞-optimization will be used instead to derive the appropriate design gains. Section V illustrates briefly the design philosophy in the synthesis of a longitudinal control system of a commercial transport.

With this unique design concept developed for solution of optimal H^2-norm type of problems, the work is later extended to address control issues related to H^∞-norm[47]. The overall scope is to provide a unified design algorithm for low-order controller synthesis that utilizes criteria based on both H^2 and H^∞-norms[57,58]. To achieve this objective an efficient numerical algorithm has been developed to solve the following optimal control problems:

(a) Mixed H^2 and H^∞-Design Objective: Synthesis of feedback/feedforward controllers of fixed (i.e. arbitrary) order and structure is based on the minimization of the objective function J given by

$$J = \sum_{i=1}^{N_p} \left\{ W_\infty^i \left\| \begin{matrix} Q^{i\,1/2} H_{zw}^i(j\omega) \\ R^{i\,1/2} H_{uw}^i(j\omega) \end{matrix} \right\|_\infty^2 + W_2^i \left\| \begin{matrix} Q^{i\,1/2} H_{zw}^i(j\omega) \\ R^{i\,1/2} H_{uw}^i(j\omega) \end{matrix} \right\|_2^2 \right\} \tag{1.a}$$

or

$$J = \lim_{t_f \to \infty} \sum_{i=1}^{N_p} \left\{ W_\infty^i \sup_{w^i(t)} \frac{\int_0^{t_f} [z^T(t)Q^i z(t) + u^T(t)R^i u(t)]dt}{\int_0^{t_f} w^{i\,T}(t) w^i(t)]dt} + W_2^i \, E[z^T(t_f)Q^i z(t_f) + u^T(t_f)R^i u(t_f)] \right\} \tag{1.b}$$

Note that $\mathbf{H}_{zw}(s)$ and $\mathbf{H}_{uw}(s)$ are the transfer matrices between the disturbance inputs $\mathbf{w}(s)$ and the performance outputs $\mathbf{z}(s)$ and controls $\mathbf{u}(s)$ of the closed-loop system respectively.

This formulation covers design criteria that are expressible either in terms of H^2-norm or H^∞-norm, or a combination of the two. Another feature of this set-up is its ability to address design robustness to plant uncertainties (e.g. structured and unstructured uncertainties at both the plant inputs and plant outputs, plant parameter variations) through the use of aggregated closed-loop responses over a set of plant conditions, signified by the summation index i ($i=1,N_p$) and N_p is the total number of design conditions. An objective function that spans multiple plant conditions further provides a means to establish gain scheduling across the entire design envelope.

Reference 23 demonstrated the usefulness of this design formulation in controlling an 8^{th}-order flexible mechanical system under a non-collocated sensor/control configuration. A second-order controller has been designed that is robust to large variation in moment of inertia of one of the disks in a four-disk system[23,53]. The resulting robust controller turns out to be non-minimum phase. This result agrees with the SISO control synthesis procedure[59] for active vibration control.

The design algorithms for evaluating both H^2 and H^∞-norms use an equivalent time-domain characterization. The equivalence is established using the familiar Parseval theorem[60].

(b) H^2 Design Objective with H^∞-Bound Constraints: Alternatively one can define the control problem being the minimization of an objective function J based on H^2-norm,

$$J = \sum_{i=1}^{N_p} W_2^i \left\| \begin{matrix} Q^{i\,1/2} H_{zw}^i(j\omega) \\ R^{i\,1/2} H_{uw}^i(j\omega) \end{matrix} \right\|_2^2 \tag{2}$$

subject to additional constraints

$$\left\| \begin{matrix} Q^{i\,1/2} H^{i}_{zw}(j\omega) \\ R^{i\,1/2} H^{i}_{uw}(j\omega) \end{matrix} \right\|_{\infty} \leq \gamma_i \tag{3}$$

for some positive scalar γ_i (i=1,N_p). Recent work in H^∞-optimization[33] follow similar past development in optimal control for H^2-norm problems to a single plant model. Algorithms have been developed for state-feedback and output-feedback controllers (of the same order as the plant model) that satisfy given H^∞-bounds (e.g. equation (3)). These methods can be applied iteratively to yield solution of an H^∞-optimal control problem.

The resulting LQG-like controllers that are solutions to the H^∞-optimal control problems suffer the same drawbacks associated with traditional LQG controllers. Moreover, solutions of low-order controllers (i.e. strictly less than the order of the plant model) for H^∞-optimization are still not available. Our method provides a convenient framework for H^∞-optimization using the early design concept developed in reference 23 for low-order controllers. The outcome is a unified design procedure that allows control practitioners to examine requirements based on current findings in H^∞-bounds or other related measures (e.g. μ-measure, worst-case perturbations in parametric uncertainties) for performance and robustness.

Finite-Time Quadratic Performance Index : One unique feature of the design algorithms for H^2 and H^∞-norm calculation is the usage of a finite-time quadratic performance index. The objective function $J(t_f)$ (with a finite terminal time t_f) for both L^2 and L^∞ norms is given by the following equations,

- For H^2-norm :
  - Random Initial Conditions:

$$J(t_f) = \sum_{i=1}^{N_p} w_{p_i} \int_0^{t_f} e^{2\alpha t}\, E[z^T(t)Q^i z(t)+u^T(t)R^i u(t)]dt \leq J(t_f \rightarrow \infty) \tag{4.a}$$

  - Random Forcing Inputs:

$$J(t_f) = \sum_{i=1}^{N_p} w_{p_i}\, E_\alpha[z^T(t_f)Q^i z(t_f)+u^T(t_f)R^i u(t_f)] \leq J(t_f \rightarrow \infty) \tag{4.b}$$

or

$$J(t_f) = \frac{1}{t_f} \sum_{i=1}^{N_p} w_{p_i} \int_0^{t_f} e^{2\alpha t}\, E[z^T(t)Q^i z(t)+u^T(t)R^i u(t)]dt \leq J(t_f \rightarrow \infty) \tag{4.c}$$

- For H^∞-norm:

$$J(t_f) = \sum_{i=1}^{N_p} w_{p_i} \frac{\int_0^{t_f} [z^T(t)Q^i z(t)+u^T(t)R^i u(t)]dt}{\int_0^{t_f} w^{i\,T}(t) w^i(t)dt} \tag{5}$$

with $w^i(t)=w_o^i \exp(j\omega_o^i t)$ where w_o^i and ω_o^i are respectively the direction vector and frequency of the "worst-case" inputs w(t) in the H^∞-norm evaluation at the i^{th} plant condition.

The formulation based on a finite-time horizon provides not only appropriate lower bounds to these exact norms, but also an indication on internal stability for disturbable and detectable systems. It is well-known that, if treated entirely in frequency domain, synthesis procedure that minimizes either the L^2 or L^∞-norms of plant outputs would not necessarily guarantee closed-loop stability. With the proposed formulation, if an optimum solution exists from the minimization of $J(t_f)$ for sufficiently large t_f, then closed-loop stability will be achieved for a detectable, disturbable and stabilizable system.

Another design concern in multivariable controls is the effect of input directionality[61] upon the closed-loop performance. Sensitivity of closed-loop responses to command inputs in the presence of plant uncertainties is often neglected or not explicitly defined in design techniques such as LQ, LQG, LQG/LTR. The design procedure described herein is based on parameter optimization of design objective that incorporates directly responses to specific commands. In this manner, effects of input directionality are obviously captured in the design objective through appropriately chosen input directions and with the usage of multiple plant conditions. It is envisioned that the ill-conditioned problem[61] of multivariable controls would no longer be a design issue.

IV. Design Tool Implementation (SANDY)

Figure 1 shows a schematics of the CAD design tool SANDY. The design tool is innovative and will serve a useful medium for the introduction of multivariable control concepts to a vast number of traditional control designers.

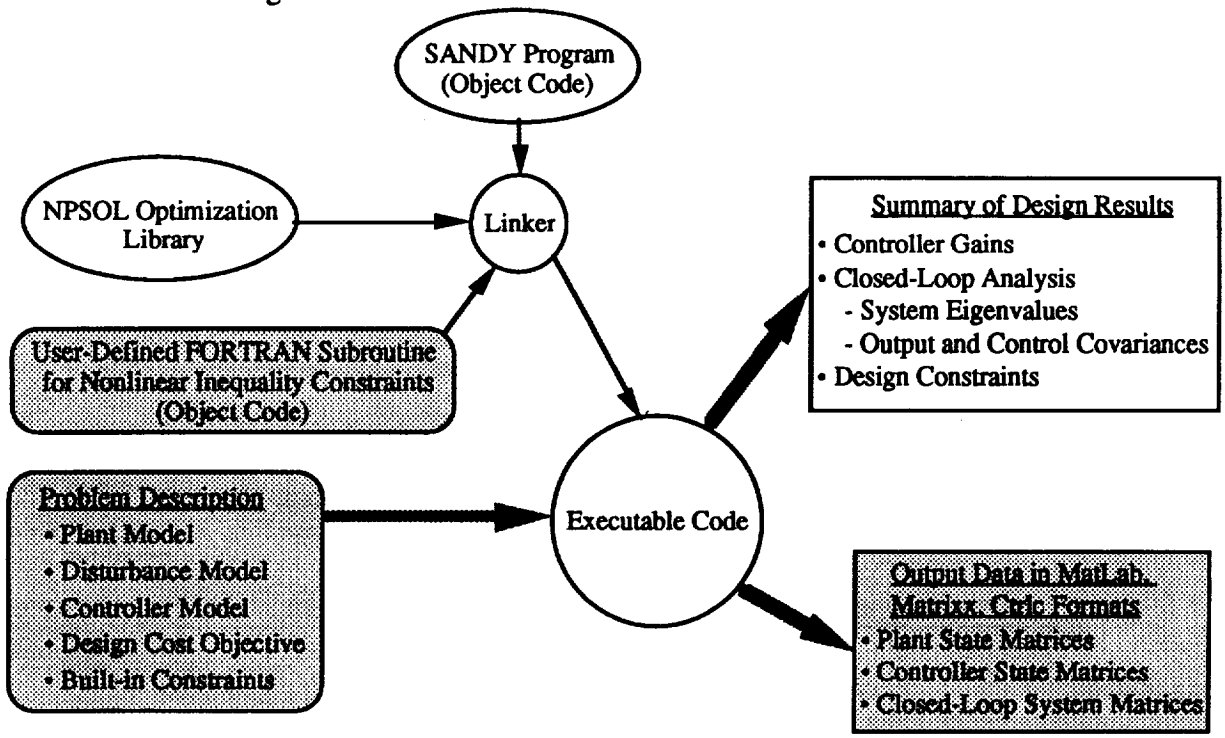


Figure 1 Schematics of the Design Tool Implementation SANDY

A procedure is set up to link the design code SANDY with the optimization library NPSOL and any user-defined Fortran subroutines defining nonlinear inequality constraints for the control design problem. This capability provides great flexibility to incorporate any additional design specifications to the problem without affecting the core program. An executable code is then generated to run the

design optimization. This version of the executable code can be run repetitively without the need to relink when the designers switch between design conditions, alter the design parameters while keeping the same design constraints as defined in the user-defined Fortran subroutines.

Characteristics of the disturbance model, weighting parameters in the design objective, selection of the controller design parameters and options in built-in design constraints are entered in one single data file. State matrices for the plant models and the controller model are defined in separate data files. Printout of the design results will be provided at the end of the program execution. Future development will include the generation of design data files for the plant models, optimized controller model and the respective closed-loop systems in compatible formats for the analysis packages such as Matlab, Matrixx and Ctrlc.

V. Design Example

Preliminary development of the above design algorithm for mixed H^2 and H^∞-optimization has been applied to the disturbance rejection problem for a B767 aircraft (Figure 2). The design objective is to synthesize a low-order feedback controller that minimizes the aircraft normal acceleration $n_{zcg}(t)$ responses to vertical gust turbulence w(t). The performance relates to both peak responses (i.e. worst-case) and mean-square responses to Dryden spectra.

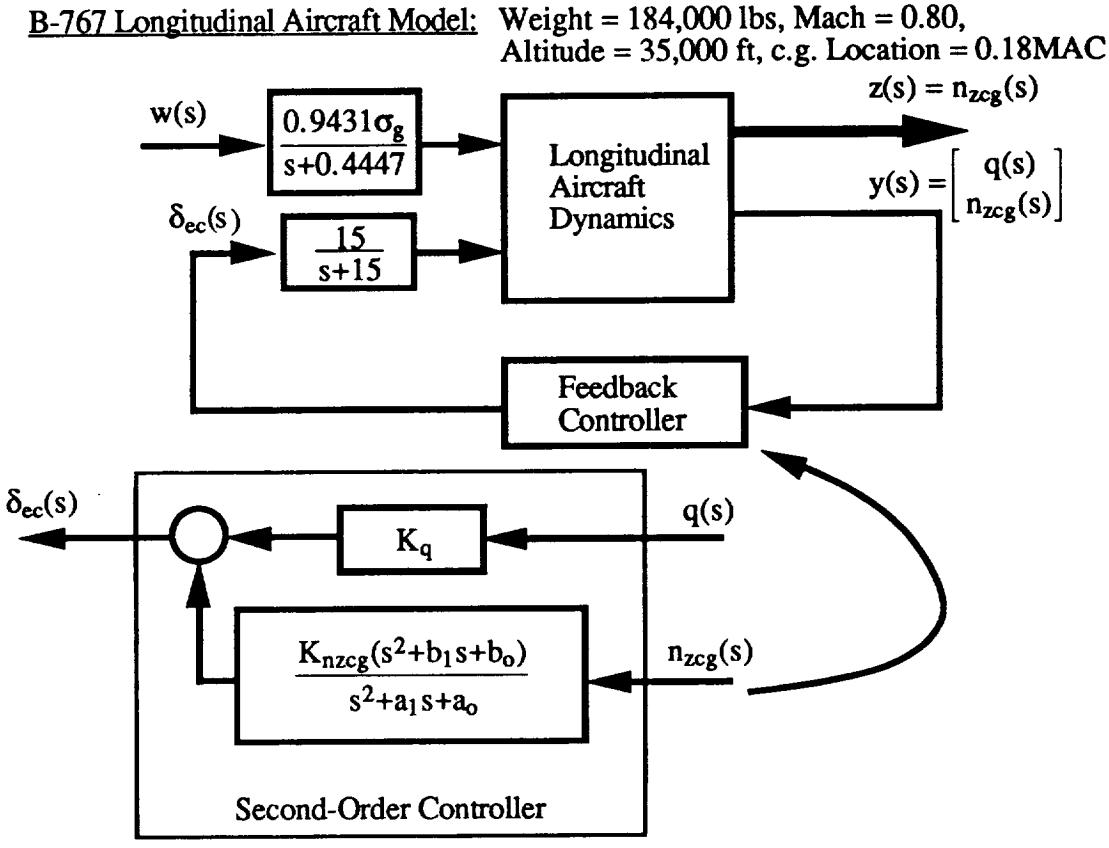


Figure 2 Disturbance Rejection Design for a B-767 Aircraft
Using Mixed H^2 and H^∞-Optimization

Mixed H^2 and H^∞- performance objective is given by

$$J = \lim_{t_f \to \infty} \{W_\infty \sup_{w(t)} \frac{\int_0^{t_f} [10n_{zcg}^2(t)+\delta_{ec}^2(t)]dt}{\int_0^{t_f} w^2(t)dt} + W_2 E[10n_{zcg}^2(t_f)+\delta_{ec}^2(t_f)]\} \quad (6)$$

where $\delta_{ec}(t)$ is the elevator control. State matrices of the design model are given in the appendix.

The controller is set up to have output feedback (Figure 2) on pitch rate q(t) and a second-order lead-lag feedback compensation on the normal acceleration $n_{zcg}(t)$. Figure 3 summarizes three controller designs illustrating trade-off achieved in mixed H^2 and H^∞-norm optimization:

(a) H^2-norm optimization: With W_2=1.0 and W_∞=0.0, this design simply solves the minimization of the mean square responses to Dryden turbulence using the controller structure shown in Figure 2.

(b) Mixed H^2 and H^∞-norm optimization: With W_2=1.0 and W_∞=1.0, this design is a predominantly H^∞-norm problem yielding results similar to those achieved with algorithms described in Reference 33. In this design the mean-square responses of $n_{zcg}(t)$ is roughly 16 percent higher than the H^2-norm optimized design (Case a) while the H^∞-norm is reduced by 20 percent. To recover the performance of H^2-optimized design (Case a), we simply increase the penalty W_2 on the H^2-norm performance.The following improvement is achieved with small degradation in H^∞-norm as seen in the next design case.

(c) Improved mixed H^2 and H^∞-norm optimization: With W_2=10.0 and W_∞=1.0, this design provides proper balance between H^2 and H^∞-norm performance. The resulting H^2-norm is almost the same as that of the H^2-optimized design (~ 1.3 percent higher) and the H^∞-norm is about 2.4 percent higher than that achieved in case (b).

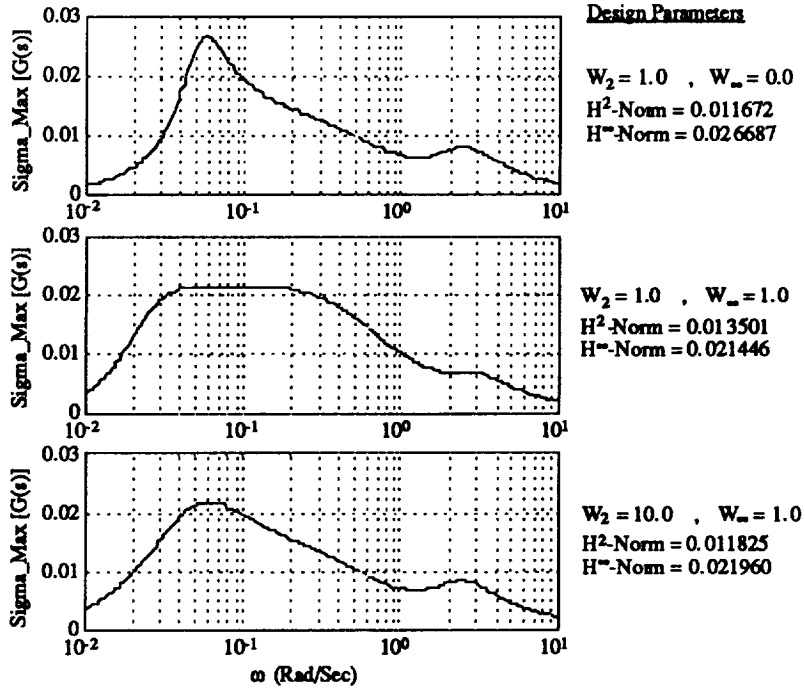


Figure 3 Comparison Between Mixed H^2 and H^∞-Norm Optimized Designs

Results of the H^∞-norm optimization are similar to those achieved by state-feedback or full-order output feedback designs using methods described in Reference 33 as indicated in Figure 4. Advantage of the current design approach is its simplicity and low order structure.

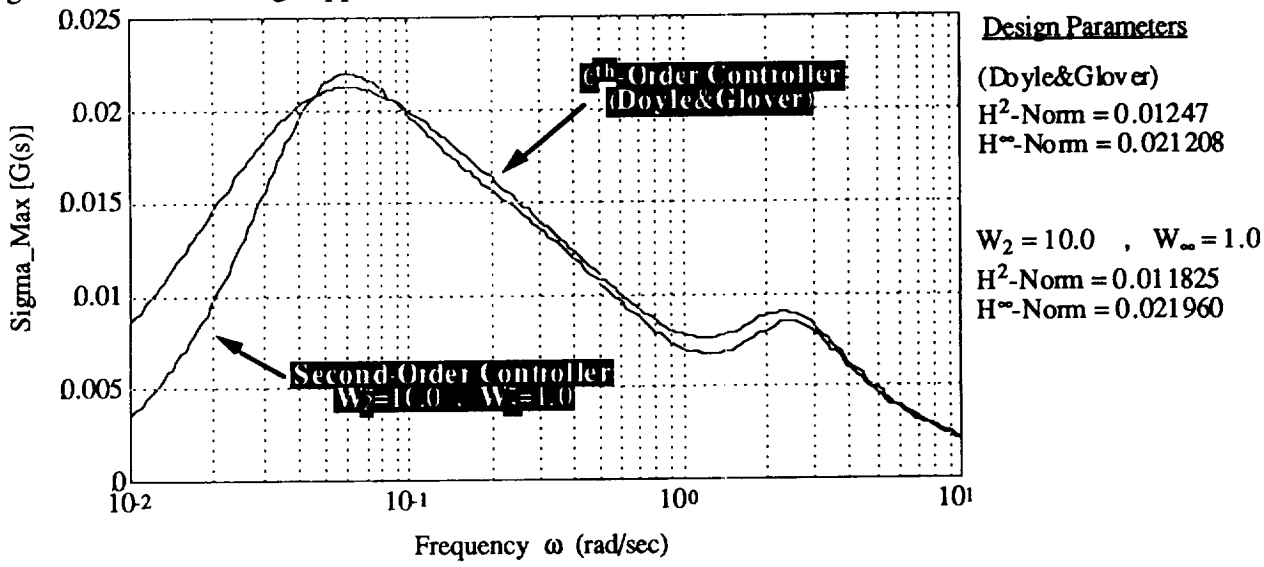


Figure 4 Comparison with Existing H^∞-Optimization Method

VI. Future Directions

Recent work have provided several useful mathematical measures for robustness characterization of multivariable feedback control design, numerical algorithms for their "exact" calculation and their usage in robust control-law synthesis. Basically there are two kinds of robustness measures depending on whether they are defined based on frequency-domain (i.e. Nyquist stability) or time-domain (i.e. Lyapunov stability) criteria.

Methods in frequency domain have made significant strive since the early work[12] initiated by Doyle. Analysis techniques to determine the μ-measure for structured uncertainty are still emerging and are most likely computationally intensive[13-18,20-21]. Complexity of these algorithms is partly the result of the wide variety of possibilities in the modeling of the uncertainty block Δ (Figure 5).

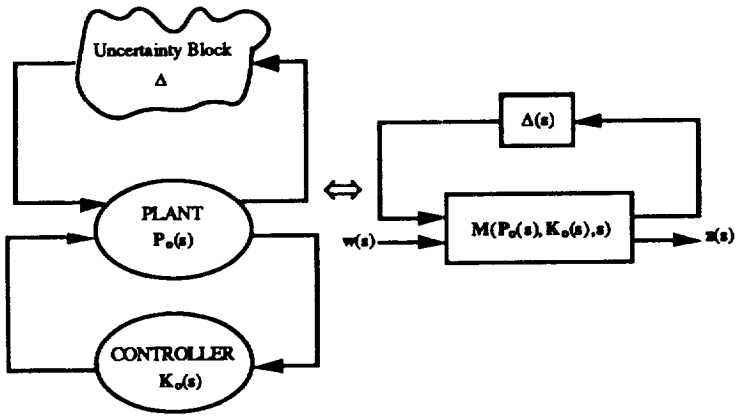


Figure 5 Robust Control Synthesis Based on Worst-Case Uncertainties of $\Delta(s)$

Generally, the more structure (i.e. information) one assigns to the uncertainty block Δ, the more difficult it is to determine the necessary and sufficient bounds on μ-measure[33].

One approach in robust control-law synthesis is to make use of recent methods for "exact" calculation of μ or related measures to define the worst-case uncertainty model, say Δ^*, and incorporate this condition into a closed-loop model $[I-\Delta^*M(s,K_o(s))]$ for re-optimization of the controller $K_o(s)$ embedded in the transfer matrix M(s).

For robustness measures based on time-domain approach[11], bounds on plant uncertainties ($\Delta A,\Delta B,\Delta C,\Delta D$) in the state matrices (A,B,C,D) are determined (Figure 6) providing sufficient conditions for closed-loop stability. These analysis procedures can be elaborated to obtain specific worst-case plant conditions. As before once established, these conditions can be implemented into the design procedure SANDY where one of the plant conditions represents the worst-case plant model from which appropriate design constraints for robustness can be defined.

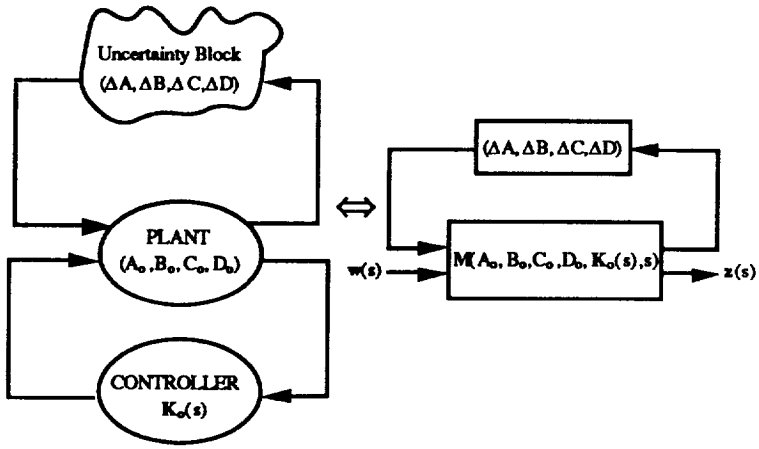


Figure 6 Robust Control Synthesis Based on Worst-Case Plant Perturbation ($\Delta A^*,\Delta B^*,\Delta C^*,\Delta D^*$)

Future work will also examine theoretical development of these robust design algorithms and their numerical implementation. These methods will be applied to the synthesis of flight control problems (e.g. SAS, autopilots, ride quality control, modal suppression, etc...) that include robustness issues such as multiloop stability margins, plant parameter variation and unmodelled high-frequency dynamics. Specifically we address the set-up of objective function and relevant design constraints (e.g. closed-loop damping, handling qualities in terms of short-period frequency, overshoots, command and control bandwidths, stability margins, limited control activities, etc...) for the following flight control problems,

(a) Stability Augmentation Systems:
- Pitch augmentation system
- Yaw damper design
- Disturbance rejection: ride quality, load factor reduction
- Structural mode stabilization: control of lightly damped structural modes

(b) Command Augmentation Systems:
- Integral Controls
- Autopilots: airspeed, altitude, flight path control
- Manual control with handling qualities
- Target tracking

As one might envision, the control synthesis depends strongly on the design objectives (e.g. inclusion of integral control, washout filters for decoupling in steady-state control, anti-aliasing filters, time delay, etc...) regardless of the methods used in the determination of feedback and feedforward control gains. One of our research goals is to identify components in the synthesis

model essential to the design problems stated in (a) and (b) based on common knowledge of the design requirements in each particular situation.

References

[1] Doyle,J.C.,"Robustness of Multiloop Linear Feedback Systems," Proceedings of the 17th-IEEE Conference on Decision and Control, January 1979, San Diego, CA, pp.12-18.

[2] Lehtomaki,N.A.,Sandell,N.R.,Athans,M.,"Robustness Results in Linear Quadratic Gaussian Based Multivariable Control Designs," IEEE Transactions on Automatic Controls, Vol.AC-26, No.1, February 1981,pp.75-92.

[3] Mukhopadhyay,V.,Newsome,J.R.,"A Multiloop System Stability Margin Study Using Matrix Singular Values," Journal of Guidance Control and Dynamics, Vol.7, September-October 1984, pp.582-587.

[4] Mukhopadhyay,V.,Newsome,J.R.,"Application of Matrix Singular Value Properties for Evaluating Gain and Phase Margins of Multiloop Systems," AIAA Guidance and Control Conference, August 9-11 1982, San Diego, CA, pp.420-428.

[5] Ly, U., "Robustness Analysis of a Multiloop Flight Control System," AIAA Guidance and Control Conference, Gatlinburg, Tennessee, August 1983, pp. 155-165.

[6] Barrett,M.F.,"Conservatism with Robustness Tests for Linear Feedback Control Systems," 19th-IEEE Control and Decision Conference, December 1980, pp.885-890.

[7] Ridgely,D.B.,Banda,S.S.,"Introduction to Robust Multivariable Control," AFWAL-TR-85-3102.

[8] Yeh,H.,Banda,S.S.,Ridgely,D.B.,"Stability Robustness Measures Utilizing Structural Information," International Journal of Control, Vol.41, No.2, February 1985, pp.365-387.

[9] Yeh,H.,Banda,S.S.,Ridgely,D.B.,"Nonconservative Evaluation of Uniform Stability Margins of Multivariable Feedback Systems," Journal of Guidance, Control and Dynamics, Vol.8, No.2, March-April 1985, pp.165-172.

[10] Yeh,H.,Banda,S.S.,Ridgely,D.B.,"Regions of Stability for Gain or Phase Variations in Multivariable Systems," Proceedings of the 23rd Conference on Decision and Control, Las Vegas, Nevada, December 1984, pp.1409-1422.

[11] Yedavalli,R.K.,Banda,S.S.,Ridgely,D.B.,"Time-Domain Stability Robustness Measures for Linear Regulators," Journal of Guidance, Control and Dynamics, Vol.8, No.4, July-August 1985, pp.520-525.

[12] Doyle,J.C.,"Analysis of Feedback Systems with Structured Uncertainty," IEE Proceedings, Part D, No.6, November 1982.

[13] Sideris,A., De Gaston, R.R.E.,"Multivariable Stability Margin Calculation with Uncertain Correlated Parameters," Proceedings of the 25th Conference on Decision and Control, Athens, Greece, December 1986.

[14] Safonov,M.G.,"Stability Margins of Diagonally Perturbed Multivariable Feedback Systems," IEE Proceedings, Vol.129, Pt. D, No.6, November 1982, pp.251-256.

[15] Sideris,A., Sanchez Pena,R.S.,"A General Program to Compute the Multivariable Stability Margin for Systems with Parametric Uncertainty," Proceedings of the American Control Conference, June 1988.

[16] De Gaston,R.R.E., Safonov,M.G.,"Exact Calculation of the Multiloop Stability Margin," IEEE Transactions on Automatic control, Vol.33, No.2, February 1988.

[17] Packard,A., Fan,M.K.H., Doyle,J.C.,"A Power Method for the Structured Singular Value," Proceedings of the 27th Conference on Decision and Control, Austin, Texas, December 1988.

[18] Fan,M.K.H., Tits,A.L.,"Characterization and Efficient Computation of the Structured Singular Value," IEEE Trans. Auto. Control, Vol.AC-31, No.8, pp.734-743, August 1986.
[19] Freudenberg,J.S., Looze,D.P., Cruz,J.B.,"Robustness Analysis Using Singular Value Sensitivities," Int. J. Control, Vol.35, No.1, pp.95-116, 1982.
[20] Helton,W.,"A Numerical Method for Computing the Structured Singular Value," Systems and Control Letters, pp.21-26, 1988.
[21] Dailey,R.L., Gangsaas,D.,"Worst-Case Analysis of Flight Control Systems Using Structural Singular Values," AIAA/AHS/ASEE Aircraft Design, Systems and Operations Meeting, Seattle, WA. July 31-August 2,1989.
[22] Gangsaas,D., Bruce,K.R., Blight,J.D. and Ly,U., "Application of Modern Synthesis to Aircraft Control: Three Case Studies," IEEE Trans. on Automatic Control, Vol. AC-31, No.11, November 1986.
[23] Ly,U.,"A Design Algorithm for Robust Low-Order Controller," Ph.D. Dissertation, Department of Aeronautics and Astronautics, Stanford University, November 1982.
[24] Ly,U. and Cannon,R.H., "A Direct Method for Designing Robust Optimal Control Systems," AIAA Guidance and Control Conference, Palo Alto, California, August 1978, pp. 440-448.
[25] Vinkler,A.P., Wood,L.J., Ly,U. and Cannon,R.H., "Minimum Expected Cost Control of Linear Systems with Uncertain Parameters - Applications to Remotely Piloted Vehicle Flight Control Systems," AIAA Guidance and Control Conference, Boulder, Colorado, August 6-8, 1979.
[26] Ly,U., "Optimal Low-Order Flight Critical Pitch Augmentation Control Law for a Transport Airplane," AIAA Guidance and Control Conference, Seattle, Washington, August 1984, pp. 743-757.
[27] Jones,R.D., Bossi,J. and Ly,U., "Multivariable Regulator Design for Robustness and Performance: A Realistic Example," American Control Conference, Seattle, Washington, June 1986, pp. 285-288.
[28] Ly,U., Bryson,A.E. and Cannon,R.H., "Design of Low-Order Compensators Using Parameter Optimization," Automatica, Vol.21, No.3, pp.315-318, 1985.
[29] Francis,B.A.,"A Course in H^∞ Control Theory," Volume 88 of Lecture Notes in Control and Information Sciences, Springer-Verlag, 1987.
[30] Halyo,N.,"A Combined Stochastic Feedforward and Feedback Control Design Methodology with Application to Autoland Design," NASA Contractor Report 4078, July 1987.
[31] Ly,U. and Ho,J.K.,"Fault Tolerant Control Laws," NASA CR-NAS1-17635 Task No. 10, 1986.
[32] Bernstein,D., Haddad,W.,"LQG Control with an H^∞-Performance Bound: A Riccati Equation Approach," Proceedings of the American Control Conference, Atlanta, Georgia, June 1988, pp. 796-802.
[33] Doyle,J.C., Glover,K.,"State-Space Formulae for All Stabilizing Controllers that Satisfy an H^∞-Norm Bound and Relations to Risk Sensitivity," Systems and Control Letters 11, 1988, pp. 167-172, North Holland.
[34] Doyle,J.C., Lenz,K., Packard,A.,"Design Examples Using μ-Synthesis Space Shuttle Lateral Axis FCS During Reentry," Proceedings of the 25th Conference on Decision and Control, Athens, Greece, December 1986.
[35] Valavani,L., Voulgaris,P.,"High Performance H^2 and H^∞ Designs for the Supermaneuverable F18/HARV Fighter Aircraft," submitted for publication.

[36] Moore, J.B., Telford, A. and Ly, U., "Controller Reduction Methods Maintaining Performance and Robustness", to be presented at the 27th CDC Conference, December 1988.
[37] K. Glover,"All Optimal Hankel-Norm Approximation of Linear Multivariable Systems and their L^∞-Error Bounds," Internat. J. Control, 1984, Vol.39, No.6, pp.1115-1193.
[38] Liu, Y., Anderson, B.D.O. and Ly, U., "Coprime Factorization Controller Reduction with Bezout Identity Induced Frequency-Weighting," submitted for publication in Automatica, 1988.
[39] Ly,U.,"Model Reduction Techniques and Their Applications to Controller Reduction," Boeing Document D180-30802-1, March 1988.
[40] Military Specifications,"Flying Qualities of Piloted Airplanes," MIL-F-8785C, November 1980.
[41] Doyle,J.C.,Stein,G.,"Robustness with Observers," presented at the 1978 Conference on Decision and Control.
[42] Saberi,A.,Sannuti,P.,"Observer Design for Loop Transfer Recovery and for Uncertain Dynamical Systems," to appear in IEEE Transactions on Automatic Control.
[43] Gangsaas,D.,Ly,U.,Norman,D.C.,"Practical Gust Load Alleviation and Flutter Suppression Control-Laws Based on a LQG Methodology," AIAA 19th Aerospace Sciences Meeting, January 12-15 1981, St. Louis, Missouri.
[44] Pernebo,L.,Silverman,L.M.,"Model Reduction via Balanced State Space Representations," IEEE Trans. Automat. Control, Vol.27, No.2, pp.382-387, April 1982.
[45] M.C. Berg, "The Design of Multirate Digital Control Systems," Ph.D. Thesis, Stanford University, Department of Aeronautics and Astronautics, Stanford, California, NSG-4002, March 1986.
[46] M.C. Berg and G. Yang, "A New Algorithm for Multirate Digital Control Law Synthesis", Proc. of the 27th Conference on Decision and Control, Austin, Texas, December 1988.
[47] Ly,U.,"A Design Algorithm for H^∞-Optimization Based on a Quadratic Cost Function," in preparation.
[48] Ho,J.K.,Cooper,S.R.,Tran,C.B.,Chakravarty,A.,"On the Design of Robust Compensators for Airplane Modal Control," 1987 American Control Conference, Minneapolis, Minnesota, June 1987.
[49] Chakravarty,A.,"In-Flight Evaluation of a Modal Suppression Yaw Damper," Proceedings of the AIAA Guidance, Navigation and Control Conference, August 17-19, 1987, Monterey, California, pp.163-169.
[50] Gardner,B.E.,"Feedforward/Feedback Control Logic for Robust Target-Tracking," Ph.D Thesis, Dept. of Aeronautics and Astronautics, Stanford University, December 1984.
[51] Gardner,B.E.,"Time-Invariant Controllers for Target Tracking," J. Guidance and Control, pp.330-337, Vol.10, No.4, July-August 1987.
[52] Hollars,M.G.,Cannon,R.H.Jr.,"Experiments on the End-Point Control of a Two-Link Robot with Elastic Drives," presented at the AIAA Guidance,Navigation and Control Conference, August 18-20 1986.
[53] Rosenthal, D.E.,"Experiments in Control of Flexible Structures with Uncertain Parameters," SUDAAR 542, Stanford University, Stanford, March 1984.
[54] Blight,J.D.,Gangsaas,D.,Richardson,T.M.,"Control-Law Synthesis for an Airplane with Relaxed Static Stability," J. Guidance, Control, and Dynamics, Vol.9, No.5, September-October 1985, pp.546-554.

[55] Thompson,C.M.,Coleman,E.E.,Blight,J.D.,"Integral LQG Controller Design for a Fighter Aircraft," Proceedings of the AIAA Guidance, Navigation and Control Conference, August 17-19, 1987, Monterey, California, pp.866-895.

[56] Gill,P.E., Murray,W., Saunders,M.A. and Wright,M.H.,"User's Guide for NPSOL (Version 4.0): A Fortran Package for Nonlinear Programming," Technical Report SOL 86-2, January 1986.

[57] Robel,G.,"On Computing the Infinity Norm," to appear in IEEE Transaction on Automatic Controls.

[58] Boyd,S., Balakrishnan,V. and Kabamba,P.,"On Computing the H^{∞} Norm of a Transfer Matrix," 1988 American Control Conference, Atlanta, Georgia; also to appear in Mathematics of Controls, Signals, and Systems, 1988.

[59] Wie,B.,Byun,K.,"New Generalized Structural Filtering Concept for Active Vibration Control Synthesis," J. Guidance, Control, and Dynamics, Vol.12, No.2, March-April 1989, pp.147-154.

[60] Kreider,D.L.,Kuller,R.G.,Ostberg,D.R.,Perkins,F.W.,"An Introduction to Linear Analysis," Addison Wesley Publishing Company, Inc., 1966.

[61] Skogestad,S.,Morari,M.,Doyle,J.C.,"Robust Control of Ill-Conditioned Plants: High-Purity Distillation," IEEE Transactions on Automatic Controls, Vol.33, No.12, December 1988.

Appendix

The following are state matrices of the synthesis model with states $\{u,\alpha,q,\theta,x_w,\delta_e\}$, inputs $\{\delta_{ec},w\}$ and outputs $\{q,n_{zcg}\}$,

```
a =-1.6750e-02   1.1214e-01   2.8000e-04  -5.6083e-01   1.6750e-02  -2.4320e-02
   -1.6400e-02  -7.7705e-01   9.9453e-01   1.4700e-03   1.6400e-02  -6.3390e-02
   -4.1670e-02  -3.6595e+00  -9.5443e-01            0   4.1670e-02  -3.6942e+00
             0            0   1.0000e+00            0            0            0
             0            0            0            0  -4.4470e-01            0
             0            0            0            0            0  -1.5000e+01
b =          0            0
             0            0
             0            0
             0            0
             0   9.4310e-01
    1.5000e+01            0
c =          0            0   1.0000e+00            0            0            0
    6.9400e-03   3.2795e-01   2.3100e-03            0  -6.9400e-03   2.6790e-02
d =    0     0
       0     0
```

An Algorithm for the Solution of Dynamic Linear Programs

Mark L. Psiaki†
Cornell University, Ithaca, New York

Abstract

The algorithm's objective is to efficiently solve Dynamic Linear Programs (DLP) by taking advantage of their special staircase structure. This algorithm constitutes a stepping stone to an improved algorithm for solving Dynamic Quadratic Programs, which, in turn, would make the nonlinear programming method of Successive Quadratic Programs more practical for solving trajectory optimization problems. The ultimate goal is to bring trajectory optimization solution speeds into the realm of real-time control.

The algorithm exploits the staircase nature of the large constraint matrix of the equality-constrained DLPs encountered when solving inequality-constrained DLPs by an active set approach. A numerically-stable, staircase QL factorization of the staircase constraint matrix is carried out starting from its last rows and columns. The resulting recursion is like the time-varying Riccati equation from multi-stage LQR theory. The resulting factorization increases the efficiency of all of the typical LP solution operations over that of a dense matrix LP code. At the same time numerical stability is ensured. The algorithm also takes advantage of dynamic programming ideas about the cost-to-go by relaxing active pseudo constraints in a backwards sweeping process. This further decreases the cost per update of the LP rank-1 updating procedure, although it may result in more changes of the active set than if pseudo constraints were relaxed in a non-stagewise fashion. The usual stability of "closed-loop" Linear/Quadratic optimally-controlled systems, if it carries over to strictly linear cost functions, implies that the savings due to reduced factor update effort may outweigh the cost of an increased number of updates.

An aerospace example is presented in which a ground-to-ground rocket's distance is maximized. This example demonstrates the applicability of this class of algorithms to aerospace guidance. It also sheds light on the efficacy of the proposed pseudo constraint relaxation scheme.

Introduction

The objective of the present work is to develop and test a special-purpose algorithm for the solution of Dynamic Linear Programs (DLP). The general form of a DLP is as follows:

find: $$\mathbf{x} = \begin{bmatrix} \mathbf{x}_0^T & \mathbf{x}_1^T & \cdots & \mathbf{x}_N^T \end{bmatrix}^T \tag{1a}$$

to minimize: $$J = \begin{bmatrix} \mathbf{c}_0^T & \mathbf{c}_1^T & \cdots & \mathbf{c}_N^T \end{bmatrix} \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} \tag{1b}$$

subject to: $$\begin{bmatrix} \mathbf{A}_{00} & \mathbf{A}_{01} & & \\ & \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{0} \\ & & \ddots & \\ \mathbf{0} & & & \mathbf{A}_{NN} \end{bmatrix} \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} - \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_N \end{bmatrix} \begin{Bmatrix} = \\ \leq \end{Bmatrix} \mathbf{0} \tag{1c}$$

where the $\mathbf{x}_i$ vectors constitute the decision vector time history, the $\mathbf{c}_i$ vectors are linear cost coefficients, and the $\mathbf{A}_{ii}$ and $\mathbf{A}_{ii+1}$ matrix blocks and the $\mathbf{b}_i$ vectors define the linear problem constraints. The bracketed equality and inequality

---

† Assistant Professor, Mechanical and Aerospace Engineering.

signs in eq. 1c indicate that both forms of constraints may be present; some rows may be equalities while others are inequalities. The problem in eq. 1a-1c is also known as a staircase LP.

A reason for interest in this problem from an aerospace controls point of view comes from its relationship to the following multi-stage trajectory optimization problem:

find: $$\mathbf{x} = \left[\mathbf{u}_0^T \; \mathbf{x}_1^T \; \mathbf{u}_1^T \; \mathbf{x}_2^T \; \ldots \; \mathbf{u}_{N-1}^T \; \mathbf{x}_N^T \right]^T \tag{2a}$$

to minimize: $$J = \sum_{k=0}^{N-1} L(\mathbf{x}_k, \mathbf{u}_k, k) \; + \phi(\mathbf{x}_N) \tag{2b}$$

subject to: $$\mathbf{x}_0 \text{ given} \tag{2c}$$

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, k) \qquad \text{for } k = 0 \ldots N-1 \tag{2d}$$

$$\mathbf{a}(\mathbf{x}_k, \mathbf{u}_k, k) \left\{ \begin{matrix} = \\ \leq \end{matrix} \right\} \mathbf{0} \qquad \text{for } k = 0 \ldots N-1 \tag{2e}$$

$$\mathbf{a}(\mathbf{x}_N, N) \left\{ \begin{matrix} = \\ \leq \end{matrix} \right\} \mathbf{0} \tag{2f}$$

where the $\mathbf{u}_i$ and $\mathbf{x}_{i+1}$ vectors constitute the control and state vector time histories, the L() and $\phi()$ functions define the nonlinear stagewise and terminal costs, eq. 2c defines the initial conditions, eq. 2d is the discrete-time dynamics difference equation, and constraints such as 2e and 2f may be present to restrict the states, or the control inputs, or both.

The current paper is part of a research program that seeks a fast and reliable way to solve the problem in eq. 2a-2f. The ultimate goal is to do real-time aerospace guidance by repeatedly solving this problem. The program is taking a two-pronged approach, algorithm improvement and parallelization of computations. This paper relates to the first prong, algorithm improvement. Fletcher's L_1 penalty function, trust region adaptation of the method of successive quadratic programs (SQP) [1] is one algorithm for solving such a nonlinear program (NP). This algorithm has fast local convergence properties, it ensures global convergence (to a local minimum), and it is good at handling inequality constraints. The application of this algorithm to the nonlinear trajectory optimization problem results in Quadratic Programming (QP) sub-problems with a special structure, the dynamic programming structure. Efficient solution of the NP requires efficient solution of the dynamic QP (DQP). Special-purpose algorithms for efficient solution of a DLP can be similar to special-purpose algorithms for efficient solution of a DQP. Thus, the present paper, in concentrating on DLP, constitutes a sort of warm-up exercise for later development of a DQP algorithm.

In addition to providing a warm-up, the present work provides a point of comparison with other research efforts in the field. Little or no work has been done on special-purpose DQP algorithms [2,3], but much attention and effort has been devoted to special-purpose DLP algorithms (e.g., Refs. 4-7). Fourer provides a useful overview of different avenues of approach that have been tried [7]. He groups algorithms for problem 1a-1c into three categories: Compact Basis, Nested Decomposition, and Transformation. All get the correct answer, but none have proved particularly successful in that none consistently out-perform the general sparse simplex method with regard to computation time.

The present algorithm is in the compact basis category; it works with a staircase factorization of the active constraints. The factorization used, a staircase QL factorization, is consistent with the plan for subsequent upgrading to handle the quadratic cost case. It has numerical stability, and there is no trade-off between numerical stability and factor compactness; general sparse matrix LP codes must deal with such trade-offs. The focus of the entire project is on aerospace guidance problems, hence the submatrices of the problem, the $\mathbf{A}_{ii}$ and $\mathbf{A}_{ii+1}$ blocks, are relatively dense. Therefore, there is hope that the current algorithm will out-perform general sparse matrix LP algorithms on these problems (e.g., algorithms such as MINOS [8]).

The body of this paper concentrates on explanation of the algorithm, with an example and conclusions at the end. Before describing the algorithm, problem 1a-1c is related to a general LP on the one hand and to a control-type LP on the other hand. The algorithm description begins with a review of the application of the L_1 penalty function method to a general LP. The staircase QL factorization then gets presented along with methods for multiplier

solution, decision vector solution, and rank-1 update. The algorithm explanation concludes with the presentation of a specialized order for problem solution that could further reduce the computational burden. The example at the end of the paper demonstrates the algorithm's applicability to aerospace trajectory optimization and examines whether the special solution ordering yields increased computational efficiency.

Equivalent Problem Forms

The problem in eq. 1a-1c is a special case of the following general LP form:

$$\text{find:} \quad \boldsymbol{\mathcal{x}} \tag{3a}$$

$$\text{to minimize:} \quad J = c^T \boldsymbol{\mathcal{x}} \tag{3b}$$

$$\text{subject to:} \quad A \boldsymbol{\mathcal{x}} - b \left\{ \begin{matrix} = \\ \leq \end{matrix} \right\} 0 \tag{3c}$$

where $\boldsymbol{\mathcal{x}}$ is defined in eq. 1a and **A**, **b**, and **c** are defined as:

$$A = \begin{bmatrix} A_{00} & A_{01} & & \\ & A_{11} & A_{12} & 0 \\ & & \ddots & \\ 0 & & & A_{NN} \end{bmatrix}, \quad b = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_N \end{bmatrix}, \quad c = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_N \end{bmatrix} \tag{4}$$

Form 3a-3c is fully general. It is the LP form used in developing general active constraint algorithms [1].

A more specialized DLP problem statement clarifies the relationship of these problems to controls:

$$\text{find:} \quad u_k \text{ for } k = 0 \ldots N\text{-}1 \text{ and } x_k \text{ for } k = 1 \ldots N \tag{5a}$$

$$\text{to minimize:} \quad J = \sum_{k=0}^{N-1} \left\{ \begin{bmatrix} c_{x_k}^T & c_{u_k}^T \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} \right\} + c_{x_N}^T x_N \tag{5b}$$

$$\text{subject to:} \quad x_0 \text{ given} \tag{5c}$$

$$x_{k+1} = F_k x_k + G_k u_k + h_k \qquad \text{for } k = 0 \ldots N\text{-}1 \tag{5d}$$

$$A_{x_k} x_k + A_{u_k} u_k - b_{xu_k} \left\{ \begin{matrix} = \\ \leq \end{matrix} \right\} 0 \qquad \text{for } k = 0 \ldots N\text{-}1 \tag{5e}$$

$$A_{x_N} x_N - b_{xu_N} \left\{ \begin{matrix} = \\ \leq \end{matrix} \right\} 0 \tag{5f}$$

where there is a direct correspondence between eq. 2a-2f and eq. 5a-5f, all functions having only linear and constant terms in the latter problem. The following definitions put problem 5a-5f in the format of problem 1a-1c:

$$\boldsymbol{\mathcal{x}}_0 = u_0, \quad A_{00} = \begin{bmatrix} A_{u_0} \\ G_k \end{bmatrix}, \quad b_0 = \begin{bmatrix} b_{xu_0} \\ -h_0 \end{bmatrix} - \begin{bmatrix} A_{x_0} \\ F_0 \end{bmatrix} x_0, \quad \text{and } c_0 = c_{u_0} \tag{6a}$$

$$\boldsymbol{\mathcal{x}}_k = \begin{bmatrix} x_k \\ u_k \end{bmatrix}, \quad A_{kk} = \begin{bmatrix} A_{x_k} & A_{u_k} \\ F_k & G_k \end{bmatrix}, \quad b_k = \begin{bmatrix} b_{xu_k} \\ -h_k \end{bmatrix}, \quad \text{and } c_k = \begin{bmatrix} c_{x_k} \\ c_{u_k} \end{bmatrix} \qquad \text{for } k = 1 \ldots N\text{-}1 \tag{6b}$$

$$A_{kk+1} = \begin{bmatrix} 0 & 0 \\ -I & 0 \end{bmatrix} \qquad \text{for } k = 0 \ldots N\text{-}1 \tag{6c}$$

$$\boldsymbol{\mathcal{x}}_N = x_N, \quad A_{NN} = A_{x_N}, \quad b_N = b_{x_N}, \quad \text{and } c_N = c_{x_N} \tag{6d}$$

Thus, the problem in eq. 1a-1c is related to controls.

Algorithm

The L_1 Exact Penalty Function and an Active Set LP Method†

The L_1 exact penalty function method enforces equality and inequality constraints as in 3c by adding a penalty cost to the problem cost that is a weighted L_1 norm of the constraint violations. The penalty function reformulation of problem 3a-3c becomes

$$\text{find:} \qquad \boldsymbol{x} \tag{7a}$$

$$\text{to minimize:} \qquad J = \mathbf{c}^T \boldsymbol{x} + \mu_{max}\{\|\mathbf{A}_e \boldsymbol{x} - \mathbf{b}_e\|_1 + \|[\mathbf{A}_{in} \boldsymbol{x} - \mathbf{b}_{in}]^+\|_1\} \tag{7b}$$

where μ_{max} is a large positive constant, where constraints $[\mathbf{A},\mathbf{b}]$ in eq. 3c have been split into the equality constraints, $[\mathbf{A}_e,\mathbf{b}_e]$, and the inequality constraints, $[\mathbf{A}_{in},\mathbf{b}_{in}]$, and where $[a]^+ = \max(a,0)$. This technique is actually very similar to the adjoining of constraints in Lagrange and Kuhn-Tucker formulations. The only difference is that the constraint multipliers are effectively limited in magnitude by μ_{max}. If μ_{max} is greater than the magnitude of the largest multiplier in the solution of problem 3a-3c, then problem 7a-7b has the same solution. The advantage of this technique is that it solves the problem in a single phase, optimizing while achieving feasibility. It is a variant of the LP technique known as the big M method. The primary reason for using it in the current paper is to make the algorithm compatible with the proposed method for eventually solving the NP in eq. 2a-2f.

Active constraints refer to those constraints that are satisfied as strict equalities. The active set method for solving problem 7a,7b consists of the following steps.

Main Algorithm:

1. Guess solution, $\boldsymbol{x}$, and split $[\mathbf{A},\mathbf{b}]$ into active and inactive rows:

$$\begin{bmatrix} \mathbf{A}_{act},\mathbf{b}_{act} \\ \mathbf{A}_{inact},\mathbf{b}_{inact} \end{bmatrix} = \mathbf{P}\,[\mathbf{A},\mathbf{b}] \tag{8}$$

   where $\mathbf{P}$ is a permutation matrix, and where the active constraints at the guessed solution must yield an $\mathbf{A}_{act}$ that is square and nonsingular.

2. Compute $\mathbf{A}_{act}^{-1}$

3. Compute $\mu_{act} = -(\mathbf{A}_{act}^{-1})^T (\mathbf{c} + \mathbf{A}^T_{inact}\mu_{inact})$ where the elements of μ_{inact} are either $-\mu_{max}$, 0, or $+\mu_{max}$ depending on whether the corresponding inactive constraint is an equality or an inequality and depending on whether it is positive or negative at the guessed solution, $\boldsymbol{x}$.

4. Find the element, i, of μ_{act} that is furthest out of the allowable range for the penalty function, $(\mu_{act})_i \in [-\mu_{max}, +\mu_{max}]$ for equality constraints, $(\mu_{act})_i \in [0, +\mu_{max}]$ for inequality constraints. Stop if no elements are out of range; the current $\boldsymbol{x}$ is optimum.

5. Compute the search direction, $\delta\boldsymbol{x} = \mathbf{A}_{act}^{-1}\mathbf{e}_i$, where $\mathbf{e}_i$ is the unit vector with all zeros except for a 1 in row i.

6. Compute the new guessed solution, $\boldsymbol{x} = \boldsymbol{x} + \alpha\,\delta\boldsymbol{x}$, where α has the same sign as $(\mu_{act})_i$ and where its magnitude is chosen to be the smallest value that makes one of the inactive constraints active, min $|\alpha|$ such that $[\mathbf{A}_{inact}(\boldsymbol{x} + \alpha\,\delta\boldsymbol{x}) - \mathbf{b}_{inact}]_j = 0$, for some row j.

---

† The discussion of this section deals with the problem form in eq. 3a-3c. The discussion is based on algorithms presented in Ref. 1.

7. Interchange rows i and j between $[\mathbf{A}_{act}, \mathbf{b}_{act}]$ and $[\mathbf{A}_{inact}, \mathbf{b}_{inact}]$, update $\mathbf{A}_{act}^{-1}$, and go to step 3.

The equation in step 3 results from differentiation with respect to $\mathbf{x}$ of the augmented cost in eq. 7b. Despite the nondifferentiability of eq. 7b for some values of $\mathbf{x}$, differentiation can be done if the active constraints are treated as adjoined equality constraints with unknown multipliers μ_{act} rather than as L_1 penalty terms. Steps 4 and 5 determine a descent direction. The step length is determined in step 6. The step length rule ensures that the entire step is in a descent direction of the piecewise-linear augmented cost. Because of the step length rule, the new active set changes by only one row. Step 7 takes advantage of this fact in the recomputation of $\mathbf{A}_{act}$'s inverse.

The algorithm used in this paper starts out by assuming that a set of pseudo constraints are active. This yields the identity matrix for the initial $\mathbf{A}_{act}$, and step 2 is trivial. The allowable range for the pseudo constraint multipliers is different than for the actual problem constraints, $\mu_{pseudo} \in [0,0]$. They get dropped from the active set in the course of the algorithm unless the problem solution is not unique.

For each constraint addition/deletion the algorithm cycles through steps 3-7. Most of the computational load per cycle is caused by manipulations with the inverse of the $\mathbf{A}_{act}$ matrix, multiplication by it in steps 3 and 5 and rank-1 updating of it in step 7. The main idea of this paper -- indeed the main idea of all compact basis schemes -- is to compactly represent factors of $\mathbf{A}_{act}$ that are suitable for carrying out multiplications by $\mathbf{A}_{act}$'s inverse and that are easy to update when $\mathbf{A}_{act}$ undergoes a rank-1 row change.

Staircase QL Factorization for Staircase LP

$\mathbf{A}_{act}$ inherits a staircase structure from A as in eq. 4. The compact QL factorization used here performs a stage-wise backwards sweep to factor the nonzero blocks of $\mathbf{A}_{act}$. The following recursion yields matrices that constitute a staircase QL factorization of $\mathbf{A}_{act}$:

$$\mathbf{D}_{NN} = \mathbf{A}_{NN_{act}} \tag{9a}$$

$$\mathbf{Q}_{kk}\begin{bmatrix} \mathbf{A}_{k\text{-}1k\text{-}1_{act}} & \mathbf{A}_{k\text{-}1k_{act}} \\ \mathbf{0} & \mathbf{D}_{kk} \end{bmatrix} = \begin{bmatrix} \mathbf{D}_{k\text{-}1k\text{-}1} & \mathbf{0} \\ \mathbf{D}_{kk\text{-}1} & \mathbf{L}_{kk} \end{bmatrix} \quad \text{for } k = N, \ldots, 1 \tag{9b}$$

$$\mathbf{Q}_{00}\mathbf{D}_{00} = \mathbf{L}_{00} \tag{9c}$$

where the $\mathbf{A}_{k\text{-}1k\text{-}1_{act}}$ and $\mathbf{A}_{k\text{-}1k_{act}}$ matrices are the nonzero blocks of the staircase $\mathbf{A}_{act}$ matrix, and where the $\mathbf{Q}_{kk}$ are orthonormal and the $\mathbf{L}_{kk}$ are lower triangular. The QL factorization is stored in the matrices $\mathbf{Q}_{kk}$, $\mathbf{L}_{kk}$, and $\mathbf{D}_{kk}$ for k = N,...,0 and the matrices $\mathbf{D}_{kk\text{-}1}$ for k = N,...,1. The $\mathbf{D}_{ij}$ matrices have no special properties. Equations 9b and 9c are only implicit relations for these factors, but the factors can be explicitly evaluated via Householder transformations. At stage k, the factorization begins with the data $\mathbf{A}_{k\text{-}1k\text{-}1_{act}}$, $\mathbf{A}_{k\text{-}1k_{act}}$, and $\mathbf{D}_{kk}$, and it computes $\mathbf{Q}_{kk}$, $\mathbf{L}_{kk}$, $\mathbf{D}_{k\text{-}1k\text{-}1}$, and $\mathbf{D}_{kk\text{-}1}$. The result $\mathbf{D}_{k\text{-}1k\text{-}1}$ then completes the necessary data for stage k-1.

Numerical stability of the factorization is ensured by the use of orthogonal transformations only. The computational complexity of the factorization algorithm is linear in the number of stages and cubic in the dimension(s) of the blocks, which is as efficient as can be expected if the blocks are dense. The factor storage is linear in the number of stages and quadratic in the dimension(s) of the blocks, which again is the best that can be achieved with dense blocks.

Equations 9a-9b are a DLP equivalent to the matrix Riccati equation of time-varying, multi-stage Linear Quadratic Regulator theory. The lower blocks of the right hand side of eq. 9b act as a closed-loop dynamic difference equation as will be shown in the next section. The upper block on the right hand side, $[\mathbf{D}_{k\text{-}1k\text{-}1}, \mathbf{0}]$, propagates active constraint effects backwards in time; it summarizes the constraints that $\mathbf{x}_{k\text{-}1}$ must satisfy in order to make possible the satisfaction of all constraints from stage k-1 onwards. Note that the number of rows in $[\mathbf{D}_{k\text{-}1k\text{-}1}, \mathbf{0}]$ does not necessarily equal the number of rows in $[\mathbf{A}_{k\text{-}1k\text{-}1_{act}}, \mathbf{A}_{k\text{-}1k_{act}}]$.

Staircase QL Solution for the Multiplier and Decision Vector Time Histories

The basic operations involved to do steps 3 and 5 of the LP algorithm presented above involve solution of a linear system by orthogonal transformation and forward or backward substitution. This is similar to the forward and backward substitutions of the simplex method and its variants. The transformations and substitutions are done in stage-wise recursions using the stagewise factors. Performing the following backward recursion then forward recursion yields the active constraint multipliers.

Backward recursion for intermediate multipliers λ_N through λ_0:

$$L_{NN}^T \lambda_N = - c_N - A_{NN_{inact}}^T \mu_{N_{inact}} - A_{N\text{-}1N_{inact}}^T \mu_{N\text{-}1_{inact}} \tag{10a}$$

$$L_{kk}^T \lambda_k = - D_{k+1k}^T \lambda_{k+1} - c_k - A_{kk_{inact}}^T \mu_{k_{inact}} - A_{k\text{-}1k_{inact}}^T \mu_{k\text{-}1_{inact}} \qquad \text{for } k = N\text{-}1, \ldots, 1 \tag{10b}$$

$$L_{00}^T \lambda_0 = - D_{10}^T \lambda_1 - c_0 - A_{00_{inact}}^T \mu_{0_{inact}} \tag{10c}$$

Forward recursion for intermediate multipliers β_0 through β_N and for active constraint multipliers $\mu_{0_{act}}$ through $\mu_{N_{act}}$:

$$\beta_0 = Q_{00}^T \lambda_0 \tag{11a}$$

$$\begin{bmatrix} \mu_{k_{act}} \\ \beta_{k+1} \end{bmatrix} = Q_{k+1k+1}^T \begin{bmatrix} \beta_k \\ \lambda_{k+1} \end{bmatrix} \qquad \text{for } k = 0 \ldots N\text{-}1 \tag{11b}$$

$$\mu_{N_{act}} = \beta_N \tag{11c}$$

Step 5 of the LP algorithm is accomplished by solving the system of equations $A_{act}\delta x = e_i$. To illustrate how the staircase QL factorization does this, the following equations present its use in solving the alternate system $A_{act}x = b_{act}$. Again, a backward recursion followed by a forward recursion yields the solution.

Backward recursion for intermediate nonhomogeneous constraint terms d_N through d_0 and g_N through g_0:

$$d_N = b_{N_{act}} \tag{12a}$$

$$\begin{bmatrix} d_{k-1} \\ g_k \end{bmatrix} = Q_{kk} \begin{bmatrix} b_{k-1_{act}} \\ d_k \end{bmatrix} \qquad \text{for } k = N, \ldots, 1 \tag{12b}$$

$$g_0 = Q_{00}\, d_0 \tag{12c}$$

Forward recursion for the decision vectors x_0 through x_N:

$$L_0 x_0 = g_0 \tag{13a}$$

$$L_{kk} x_k = - D_{kk-1} x_{k-1} + g_k \qquad \text{for } k = 1 \ldots N \tag{13b}$$

As stated earlier, eq. 13b is like the closed-loop dynamic difference equation of multi-stage LQR theory. The matrix L_{kk} is lower triangular and allows for easy solution for x_k in terms of x_{k-1} and g_k.

Rank-1 Update of Staircase QL Factorization

This section explains how to efficiently update the staircase QL factorization after a single constraint addition/deletion. This procedure must be carried out every time step 7 of the main algorithm is encountered. One could recompute the entire factorization, but the practicality of all LP codes hinges on their ability to update the factors for much less work than would be required to recompute them from scratch.

The general add/drop updating scheme for the staircase QL factorization must update the results of eq. 9a-9b when an arbitrary row j at stage kadd gets added to the active constraint set and another arbitrary row i at stage kdrop gets deleted from the active constraint set. Thus, $[A_{kk_{act}}, A_{kk+1_{act}}]|_{k=kadd}$ gets a new row and $[A_{kk_{act}}, A_{kk+1_{act}}]|_{k=kdrop}$ loses a row. The stages kadd and kdrop can have any relationship to each other, and the update algorithm must be able to handle all possible cases. Three different cases can occur, kadd > kdrop, kadd = kdrop, and kadd < kdrop.

Efficient rank-1 update can be accomplished by a series of stage-wise rank-1 updates linked together in an appropriate manner. Three different stagewise rank-1 updating algorithms are needed to do this. The first algorithm updates the factors computed in eq. 9b when a new row has been added to the bracketed expression on the left-hand side of that equation. The second algorithm updates these same factors in the case of a row deletion from the bracketed expression on the left-hand side. This second algorithm also modifies $Q_{k\text{-}1k\text{-}1}$ by a single Householder transformation. The third stagewise algorithm updates these same factors when D_{kk} has undergone an arbitrary rank-1 change. Recall that the bracketed matrix on the left-hand side of eq. 9b represents the input data for a given stage and the Q_{kk} matrix together with the bracketed expression on the right-hand side represents the result of the stagewise factorization. The following discussion explains each of these three algorithms and the way in which they work together to accomplish the multi-stage rank-1 update.

First, consider what happens to the stage k factorization when a new row gets added to either $[A_{k\text{-}1k\text{-}1_{act}}, A_{k\text{-}1k_{act}}]$ or D_{kk}. The algorithm begins by adding a row and a column to Q_{kk} with all 0s except for a 1 at the intersection of the new row and the new column. Thus, Q_{kk} remains orthonormal. Suppose the new constraint row is $[p^T_{k1}, p^T_{k2}]$, then the new row and column of Q_{kk} are added so that eq. 9b temporarily becomes:

$$\begin{bmatrix} Q_{kk_{11}} & 0 & Q_{kk_{12}} \\ 0^T & 1 & 0^T \\ Q_{kk_{21}} & 0 & Q_{kk_{22}} \end{bmatrix} \begin{bmatrix} A_{k\text{-}1k\text{-}1_{act}} & A_{k\text{-}1k_{act}} \\ p^T_{k1} & p^T_{k2} \\ 0 & D_{kk} \end{bmatrix} = \begin{bmatrix} D_{k\text{-}1k\text{-}1} & 0 \\ p^T_{k1} & p^T_{k2} \\ D_{kk\text{-}1} & L_{kk} \end{bmatrix} \tag{14}$$

where the $Q_{kk_{ij}}$ matrix blocks are just the blocks of the original Q_{kk} matrix. Suppose n_k is the dimension of the x_k decision vector. Then it is also the dimension of the square lower-triangular matrix L_{kk}. A series of n_k Givens rotations can be performed to zero out p^T_{k2} while preserving the lower-triangular structure of L_{kk}. The first Givens rotation uses the last row of L_{kk} as the pivot row and zeros out the last element of p^T_{k2}, and successive rotations use successively higher rows of L_{kk} as the pivot and zero out successive elements of p^T_{k2} going from right to left. Suppose these rotations are G_1 to G_{n_k}. Then the new stage k factorization becomes:

$$Q_{kk_{new}} = G_{n_k} \bullet \ldots \bullet G_1 \bullet \begin{bmatrix} Q_{kk_{11}} & 0 & Q_{kk_{12}} \\ 0^T & 1 & 0^T \\ Q_{kk_{21}} & 0 & Q_{kk_{22}} \end{bmatrix} \tag{15a}$$

$$\begin{bmatrix} D_{k\text{-}1k\text{-}1} & 0 \\ d^T_{k1} & 0^T \\ D_{kk\text{-}1_{new}} & L_{kk_{new}} \end{bmatrix} = G_{n_k} \bullet \ldots \bullet G_1 \bullet \begin{bmatrix} D_{k\text{-}1k\text{-}1} & 0 \\ p^T_{k1} & p^T_{k2} \\ D_{kk\text{-}1} & L_{kk} \end{bmatrix} \tag{15b}$$

$$D_{k\text{-}1k\text{-}1_{new}} = \begin{bmatrix} D_{k\text{-}1k\text{-}1} \\ d^T_{k1} \end{bmatrix} \tag{15c}$$

where the last equation has been included to emphasize the fact that the new $D_{k\text{-}1k\text{-}1}$ differs from the old $D_{k\text{-}1k\text{-}1}$ by only a single new row. This fact sets the stage for the use of this same algorithm at stage k-1. The new Q_{kk} is orthonormal because the augmented matrix is orthonormal and because all Givens rotations are orthonormal. Thus, the new factors have all of the required properties for use in the LP algorithm described above.

Next, consider what happens to the stage k factorization when a row gets deleted from either $[\mathbf{A}_{k\text{-}1k\text{-}1_{act}}, \mathbf{A}_{k\text{-}1k_{act}}]$ or $\mathbf{D}_{kk}$. The following development is based on ideas for QP from Ref. 9. Write $\mathbf{Q}_{kk}$ in the form

$$\mathbf{Q}_{kk} = \begin{bmatrix} \mathbf{Q}_{kk11} & \mathbf{q}_{kk12} & \mathbf{Q}_{kk13} \\ \mathbf{q}^T_{kk21} & \mathbf{q}_{kk22} & \mathbf{q}^T_{kk23} \\ \mathbf{Q}_{kk31} & \mathbf{q}_{kk32} & \mathbf{Q}_{kk33} \end{bmatrix} \tag{16}$$

where the middle column conforms in matrix multiplication with the constraint row that is getting deleted -- rows of $\mathbf{D}_{kk}$ can be referred to as constraints; they are propagated active constraints. The bottom blocks, $[\mathbf{Q}_{kk31}, \mathbf{q}_{kk32}, \mathbf{Q}_{kk33}]$, have n_k rows, the same as in the bottom blocks on the right hand side of eq. 9b, $[\mathbf{D}_{kk\text{-}1}, \mathbf{L}_{kk}]$.

The stagewise deletion algorithm starts with a Householder transformation in which the $\mathbf{q}_{kk22}$ row in the above representation is used as the pivot row to zero out $\mathbf{q}_{kk12}$ in the first rows. Next, a series of n_k Givens rotations is used to zero out successive elements of $\mathbf{q}_{kk32}$ starting with the topmost element and working downwards. Again, the $\mathbf{q}_{kk22}$ row in the above representation is used as the pivot. If the Householder transformation is $\mathbf{H}$ and the Givens rotations are $\mathbf{G}_1$ to $\mathbf{G}_{n_k}$, then the following changes to the stage k factorization result:

$$\begin{bmatrix} \mathbf{Q}_{kk11_{new}} & \mathbf{0} & \mathbf{Q}_{kk12_{new}} \\ \mathbf{0}^T & 1 & \mathbf{0}^T \\ \mathbf{Q}_{kk21_{new}} & \mathbf{0} & \mathbf{Q}_{kk22_{new}} \end{bmatrix} = \mathbf{G}_{n_k} \bullet \ldots \bullet \mathbf{G}_1 \bullet \mathbf{H} \bullet \begin{bmatrix} \mathbf{Q}_{kk11} & \mathbf{q}_{kk12} & \mathbf{Q}_{kk13} \\ \mathbf{q}^T_{kk21} & \mathbf{q}_{kk22} & \mathbf{q}^T_{kk23} \\ \mathbf{Q}_{kk31} & \mathbf{q}_{kk32} & \mathbf{Q}_{kk33} \end{bmatrix} \tag{17a}$$

$$\mathbf{Q}_{kk_{new}} = \begin{bmatrix} \mathbf{Q}_{kk11_{new}} & \mathbf{Q}_{kk12_{new}} \\ \mathbf{Q}_{kk21_{new}} & \mathbf{Q}_{kk22_{new}} \end{bmatrix} \tag{17b}$$

$$\begin{bmatrix} \mathbf{D}_{k\text{-}1k\text{-}1_{new}} & \mathbf{0} \\ \mathbf{p}^T_{k1} & \mathbf{p}^T_{k2} \\ \mathbf{D}_{kk\text{-}1_{new}} & \mathbf{L}_{kk_{new}} \end{bmatrix} = \mathbf{G}_{n_k} \bullet \ldots \bullet \mathbf{G}_1 \bullet \mathbf{H} \bullet \begin{bmatrix} \mathbf{D}_{k\text{-}1k\text{-}1} & \mathbf{0} \\ \mathbf{D}_{kk\text{-}1} & \mathbf{L}_{kk} \end{bmatrix} \tag{17c}$$

where $[\mathbf{p}^T_{k1}, \mathbf{p}^T_{k2}]$ corresponds to the constraint that is getting dropped. Orthonormality of the original $\mathbf{Q}_{kk}$ matrix ensures the form of the result on the left-hand side of eq. 17a. Note that the matrix $\mathbf{D}_{k\text{-}1k\text{-}1_{new}}$ is a function only of $\mathbf{D}_{k\text{-}1k\text{-}1}$ and $\mathbf{H}$; the Givens rotations do not affect it. Therefore, another Householder transformation, $\mathbf{H}''$, can be constructed based on the same Householder vector. It yields:

$$\begin{bmatrix} \mathbf{D}_{k\text{-}1k\text{-}1_{new}} \\ \mathbf{d}^T_{k\text{-}1_{drop}} \end{bmatrix} = \mathbf{H}'' \, \mathbf{D}_{k\text{-}1k\text{-}1} \tag{18}$$

where $\mathbf{d}_{k\text{-}1_{drop}}$ is not necessarily equal to $\mathbf{p}_{k1}$. This sets the stage for propagation of the constraint deletion process backwards to stage k-1. If $\mathbf{Q}_{k\text{-}1k\text{-}1}$ gets transformed according to

$$\mathbf{Q}_{k\text{-}1k\text{-}1_{interim}} = \mathbf{Q}_{k\text{-}1k\text{-}1} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}'' \end{bmatrix} \tag{19}$$

then $\mathbf{Q}_{k\text{-}1k\text{-}1_{interim}}$ is still orthonormal because $\mathbf{H}''$ is orthonormal, and because $\mathbf{H}''$ is equal to its transpose, constraint $[\mathbf{0}^T, \mathbf{d}^T_{k\text{-}1_{drop}}]$ is the constraint that must get dropped at stage k-1. The foregoing algorithm can accomplish the deletion at this next preceding stage.

The algorithm that performs the multi-stage rank-1 update of the staircase QL factorization starts with the highest stage at which either a constraint addition or deletion occurs. It uses whichever of the two foregoing stagewise updating algorithms is appropriate to propagate the addition or deletion backwards. It continues until it reaches a stage at which both an addition and a deletion must take place. One, but not both, of the changes at this stage may be the result of a backwards propagation. At this stage of the concurrent add/drop, the multi-stage algorithm first does a single-stage constraint addition followed by a single-stage constraint deletion with no change of stage in between.

If the index of this stage is k, then $\mathbf{D}_{k\text{-}1k\text{-}1}$ will differ from its pre-update value by a rank-1 change at most. This can be shown by recognizing that the result on the left-hand side of eq. 15c becomes the input data on the right-hand side of eq. 18 when an add followed by a drop both occur at the same stage:

$$\begin{bmatrix} \mathbf{D}_{k\text{-}1k\text{-}1_{new}} \\ \mathbf{d}^T_{k\text{-}1_{drop}} \end{bmatrix} = \mathbf{H}'' \begin{bmatrix} \mathbf{D}_{k\text{-}1k\text{-}1} \\ \mathbf{d}^T_{k1} \end{bmatrix} \tag{20}$$

$\mathbf{H}''$ is a Householder transformation; it differs from the identity matrix by a rank-1 matrix, hence the conclusion about the change in $\mathbf{D}_{k\text{-}1k\text{-}1}$. Define this rank-1 change in terms of the vectors $\mathbf{r}_{k\text{-}1}$ and $\mathbf{s}_{k\text{-}1}$:

$$\mathbf{D}_{k\text{-}1k\text{-}1_{new}} = \mathbf{D}_{k\text{-}1k\text{-}1} + \mathbf{r}_{k\text{-}1}\mathbf{s}^T_{k\text{-}1} \tag{21}$$

If either $\mathbf{r}_{k\text{-}1}$ or $\mathbf{s}_{k\text{-}1}$ is the $\mathbf{0}$ vector, then the multi-stage rank-1 update is complete. If not, then another stagewise updating algorithm is needed.

The final stagewise updating algorithm must update the stagewise factors for an arbitrary rank-1 change in the data $\mathbf{D}_{kk}$. It is allowed to produce at most a rank-1 change in $\mathbf{D}_{k\text{-}1k\text{-}1}$. This restriction on its effect on $\mathbf{D}_{k\text{-}1k\text{-}1}$ makes it self recursive for all subsequent stagewise factorizations in the backwards chain. It can be used for updating the factorizations of all stages that precede the concurrent constraint addition/deletion stage. It can be used recursively until no more updating is needed.

One might suppose that the necessary algorithm has already been developed in a work such as Ref. 10. That paper is a good reference for rank-1 modifications, and it defines the general methodology used in the algorithm below, but the relevant algorithm from [10] would result in a rank-2 change to $\mathbf{D}_{k\text{-}1k\text{-}1}$. This would destroy the stagewise recursive applicability of the algorithm, hence the modified algorithm presented below.

Suppose there has been a rank-1 modification to $\mathbf{D}_{kk}$ as in eq. 21 (except at stage k instead of stage k-1). Then, eq. 9b gets modified:

$$\mathbf{Q}_{kk}\begin{bmatrix} \mathbf{A}_{k\text{-}1k\text{-}1_{act}} & \mathbf{A}_{k\text{-}1k_{act}} \\ \mathbf{0} & \left[\mathbf{D}_{kk}+\mathbf{r}_k\mathbf{s}^T_k\right] \end{bmatrix} = \begin{bmatrix} \mathbf{D}_{k\text{-}1k\text{-}1} & \mathbf{0} \\ \mathbf{D}_{kk\text{-}1} & \mathbf{L}_{kk} \end{bmatrix} + \begin{bmatrix} \mathbf{v}_{k\text{-}1} \\ \mathbf{w}_k \end{bmatrix}\begin{bmatrix} \mathbf{0}^T & \mathbf{s}^T_k \end{bmatrix} \tag{22}$$

where

$$\begin{bmatrix} \mathbf{v}_{k\text{-}1} \\ \mathbf{w}_k \end{bmatrix} = \mathbf{Q}_{kk}\begin{bmatrix} \mathbf{0} \\ \mathbf{r}_k \end{bmatrix} \tag{23}$$

and where $\mathbf{v}_{k\text{-}1}$ and $\mathbf{D}_{k\text{-}1k\text{-}1}$ have the same number of rows, $n_{dk\text{-}1}$. The algorithm starts by reducing the v-w vector to a vector with zeros in all of its entries except the last two. This is done by first applying a Householder transformation, $\mathbf{H}_1$, to the first $n_{dk\text{-}1}+1$ rows to zero out the first $n_{dk\text{-}1}$ rows. Then a series of Givens rotations, $\mathbf{G}_1$ to $\mathbf{G}_{nk\text{-}1}$, is applied to successive pairs of rows of the resulting vector to zero out successive elements until only the last two elements are left nonzero. These same transformations are applied to all terms on both sides of eq. 22, and the two terms on the right hand side of the equation are added together with the following (partial) result:

$$G_{n_{k-1}} \cdot \ldots \cdot G_1 \cdot H_1 \cdot \left\{ \begin{bmatrix} 0 \\ L_{kk} \end{bmatrix} + \begin{bmatrix} v_{k-1} \\ w_k \end{bmatrix} s^T{}_k \right\} = \begin{bmatrix} \alpha v_{k-1} & 0 & 0 & \ldots & 0 \\ * & * & & & \\ * & * & * & & 0 \\ * & * & * & * & \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix} \tag{24}$$

where α is a scalar and where asterisks (*) indicate nonzero scalar elements of the matrix. The top row of vector entries in the right hand matrix corresponds to the upper right-hand **0** block in the bracketed expression on the right side of eq. 9b. The bottom rows constitute a matrix with nonzero elements on the first diagonal above the main, on the main diagonal, and below the main diagonal. All elements on diagonals that are 2 or more above the main diagonal are zero. The nonzero entry in the first column of the top block, αv_{k-1}, results from application of the $\mathbf{H}_1$ Householder transformation to matrix $[\mathbf{0}^T, L^T{}_{kk}]^T$.

The remaining transformations are applied in order to restore lower triangularity to the matrix on the right hand side of eq. 24. First, a series of n_k-1 Givens rotations, $\mathbf{G}_{n_k}$ to $\mathbf{G}_{2n_k-2}$, is applied to successive pairs of rows of the matrix starting from the last two rows and working up to the first two rows in the lower block. Each rotation zeros out one of the above-diagonal elements. At the end of this operation the matrix has the form

$$\begin{bmatrix} \alpha v_{k-1} & 0 & 0 & \ldots & 0 \\ * & & & & \\ * & * & & & 0 \\ * & * & * & & \\ \vdots & \vdots & \vdots & \ddots & \\ * & * & * & * & \\ * & * & * & * & * \end{bmatrix} \tag{25}$$

so that the lower block is lower triangular. The final part of the algorithm is to apply a last Householder transformation, $\mathbf{H}_2$, to zero out the first column of the top block. These operations result in the following factor updates:

$$Q_{kk_{new}} = \mathbf{H}_2 \cdot \mathbf{G}_{2n_k-2} \cdot \ldots \cdot \mathbf{G}_1 \cdot \mathbf{H}_1 \cdot Q_{kk} \tag{26a}$$

$$\begin{bmatrix} D_{k\text{-}1k\text{-}1_{new}} & 0 \\ D_{kk\text{-}1_{new}} & L_{kk_{new}} \end{bmatrix} = \mathbf{H}_2 \cdot \mathbf{G}_{2n_k-2} \cdot \ldots \cdot \mathbf{G}_1 \cdot \mathbf{H}_1 \cdot \left\{ \begin{bmatrix} D_{k\text{-}1k\text{-}1} & 0 \\ D_{kk\text{-}1} & L_{kk} \end{bmatrix} + \begin{bmatrix} v_{k-1} \\ w_k \end{bmatrix} \begin{bmatrix} 0^T & s^T{}_k \end{bmatrix} \right\} \tag{26b}$$

In both of the Householder transformations, the first $n_{d_{k-1}}$ elements of the transformation vector are parallel to v_{k-1}, and none of the Givens rotations affect the first $n_{d_{k-1}}$ rows of the bracketed expression on the right of eq. 26b. Therefore, $D_{k\text{-}1k\text{-}1_{new}}$ differs from $D_{k\text{-}1k\text{-}1}$ only by a rank-1 matrix:

$$D_{k\text{-}1k\text{-}1_{new}} = D_{k\text{-}1k\text{-}1} + v_{k-1} y^T{}_{k-1} \tag{27}$$

where the vector y_{k-1} can be determined from the algorithm presented above. Thus, the algorithm updates stage k according to the rank-1 change in the stage's input data, and it produces a similar rank-1 change in the input data for stage k-1. The multi-stage rank-1 updating algorithm propagates these rank-1 changes backwards until at some stage one or both of the vectors in the rank-1 change are zero. This occurs at least by the time stage k = 0 is reached because $D_{-1\text{-}1}$ has zero dimension.

Numerical stability of the factorization update is ensured by the use of orthogonal transformations only. The computational complexity of the multi-stage update algorithm is linear in the number of stages affected and quadratic in the dimension(s) of the blocks, which is as efficient as can be expected if the blocks are dense. If the number of stages affected by a particular row interchange of active and inactive constraints can kept small, then the cost of the update will be small. This fact provides the motivation for the solution scheme presented in a later section.

Possible Improvements to Banded Staircase QL Factorization

Several issues come to mind in considering the forgoing use of a QL factorization for an LP basis factorization. They all revolve around a single question: is the entire factorization needed to implement the LP algorithm? For instance, a general LP method has been developed that uses LQ factorization but does not store Q [11]. Not storing the Q factors would yield a great savings in memory and computation time if it carried over to the present multi-stage algorithm. This presents no difficulty to the procedures for solving for the multiplier and decision vectors, steps 3 and 5 of the main LP algorithm. The problem with not storing Q occurs in the factor update, step 7. There is no apparent way to do the single-stage constraint deletion or the single-stage rank-1 modification without storing at least some of the $\mathbf{Q_{kk}}$ matrix. Reference 9 has some ideas in its section on quadratic programming that could be used to climinate storage of the lower part of $\mathbf{Q_{kk}}$. Alternatively, storage of $\mathbf{D_{kk}}$ and $\mathbf{D_{kk\text{-}1}}$ could be eliminated. Savings in computation time and memory would be about the same for either scheme, about 30% savings. These issues may be explored in a later work.

Backwards-Sweeping Pseudo Constraint Relaxation and an Alternate Method of Selecting the Active Constraint to Drop

In theory, all dynamic programming problems can be solved by first computing the cost-to-go at each stage, then solving a single stage optimization at each stage. Part of the cost for each of these single stage problems is the cost-to-go that results from the stage's decisions. For DLPs and for their associated L_1 penalty function problems, the cost-to-go at a given stage is a piecewise-linear convex function of the decisions at that stage. This convexity property gives rise to a hope that DLPs may have a property like the stability property of their quadratic-cost counterparts, multi-stage LQR problems. In the DLP context, this property might mean that a small change in the decisions at a given stage would give rise to even smaller changes in the state at subsequent stages. This might translate into a grouping of constraint additions and deletions at stages nearly following the stage at which the decision variations are taking place.

If this property exists, it can be exploited without the necessity of computing the entire cost-to-go function. If all of the active constraint multipliers for constraints following a given stage are within their allowable range, then the guessed solution is an optimal trajectory for all stages following that stage. Also, the local linear piece of the cost-to-go function is known. Suppose the given stage can be optimized without causing any of the multipliers at subsequent stages to exceed their L_1 penalty function bounds. Suppose also that all of the original pseudo constraints are active for the preceding stages. Then, the rank-1 updates that would have to be done during the optimization of that stage might involve changes to very few stages. The assumption about the the pseudo constraints ensures that the updates will not affect any of the stages preceding stage k-1 if stage k is being optimized. The possibility of stability implies that max(kadd,kdrop) might, in most cases, not be much larger than k.

A change is needed to the main LP procedure presented above. It allows the multipliers at stages subsequent to the stage being optimized to vary outside of their L_1 penalty function bounds. The modification needs to be in the selection of the active constraint that gets dropped on each cycle. In the main algorithm, the dropped constraint is the same as the non-optimal constraint that gets relaxed in steps 4-6. This could cause an active constraint multiplier that was within its bounds to go out of its bounds. If the multiplier corresponded to a constraint at a subsequent stage, then the optimality of the subsequent stages would break down.

This situation can be avoided by performing a search in the active constraint multiplier space for the active constraint to be dropped. This search is the dual of that carried out in steps 5 and 6 of the main algorithm, and the search direction is defined by relaxing the L_1 penalty function constraint on the multiplier associated with inactive constraint j, the inactive constraint that is becoming active. The size of the step in multiplier space is chosen to be the smallest that brings one of the active constraint multipliers to a bound which the multiplier would violate if the step size were larger; the new active constraint must be included in this test. The active constraint whose multiplier

bound limits this step size is the active constraint that gets dropped. It is not necessarily the constraint whose relaxation was dictated in step 3 of the main algorithm.

With this scheme in place, only pseudo constraints will have multipliers that are out of bounds in the step 3 optimality test. The number of non-optimal active constraints will never increase. In turn, each pseudo constraint will eventually be the constraint that gets chosen for dropping, though this may happen while another pseudo constraint is being relaxed.

A special order has been chosen for relaxing pseudo constraints to take advantage of the possibility of savings from "stability". The modified main algorithm starts by testing and relaxing only stage-N pseudo constraints in steps 3-6. This continues until all of the stage-N pseudo constraints have dropped from the active list or have zero multipliers. Then the algorithm switches to exclusive consideration of the stage-(N-1) pseudo constraints in steps 3-6. It performs add/drop cycles until all of these pseudo constraints get dropped or have zero multipliers. It continues this stagewise pseudo constraint relaxation scheme in a backwards sweep all the way to stage 0. The guessed solution is optimal after the last stage-0 pseudo constraint has been dropped or has had its multiplier go to zero. The trajectory from stage k to stage N is an optimal trajectory once all of the stage-k pseudo constraints have been dropped or have had their multipliers go to zero (although it probably will not be the final optimal trajectory associated with the solution to the overall problem).

Comparison of Algorithm Complexity with Matrix Riccati Equation

Table 1 compares the present algorithm's computational complexity with that of related algorithms for a typical aerospace controls problem. The time-varying multi-stage Matrix Riccati equation actually does not compute an A_{act} because it solves a different optimization problem. It has been included because control engineers are more familiar with it. The three QL factorization entries assume that the factors are built up from initial pseudo constraints via 900 rank-1 updates. In the last two entries, assumptions are made about the average number of stages affected per rank-1 update. The table clearly indicates that the staircase QL factorization makes a tremendous improvement in comparison to the dense factorization; the improvement will not be nearly so great in comparison to a general sparse matrix code. Also, large improvements are expected from the special ordering of the pseudo constraint relaxation. Note that all of the algorithms are far more costly than the implementation of a time-varying LQR solution. Inequality constraints are difficult to handle.

Table 1.

A Comparison of Effort for Factorization of A_{act} for a Typical Aerospace Control Example
(100 stages, 6 state vector elements, 3 control vector elements)

Solution Method	Effort (No. of Mult., Div., & Sqrt.)
Matrix Riccati Equation	112,000
Dense Matrix QL, Not storing Q	2,920,000,000
Staircase QL, Arbitrary order of pseudo constraint relaxation	90,700,000
Staircase QL, Special order of pseudo constraint relaxation	4,400,000

Aerospace Example

A simple aerospace control problem has been solved with the algorithm in order to demonstrate the usefulness of this class of algorithms on aerospace problems and in order to study the algorithm's behavior. The problem is one of fixed-time maximization of the distance travelled by a thrust- and impulse-limited ground-to-ground rocket. The continuous-time problem is:

find: $\mathbf{u}(t)$ for $0 \le t \le t_f = 12$ sec (28a)

to minimize: $J = -[1\ 0\ 0\ 0]\mathbf{x}(t_f)$ (28b)

subject to: $\mathbf{x}(0) = \mathbf{0}$ (28c)

$$\frac{dx}{dt} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 & 0 \\ 32.2 & 0 \\ 0 & 0 \\ 0 & 32.2 \end{bmatrix} u + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -32.2 \end{bmatrix} \tag{28d}$$

$$\|u(t)\|_2 \leq 5 \text{ g} \tag{28e}$$

$$\int_0^t \|u(\tau)\|_2 d\tau \leq 10 \text{ g-sec} \tag{28f}$$

$$-[0\ 0\ 1\ 0]x(t) \leq 0 \text{ ft.} \tag{28h}$$

which is a point-mass model of motion in the vertical plane. The acceleration (thrust) limit is 5 gs and the impulse limit is 10 g-sec. The first two state vector elements are horizontal position and velocity; the last two elements are vertical position and velocity. Only thrust and a uniform gravity field act on the rocket. The first control vector element is horizontal acceleration; the second element is vertical acceleration. Constraint 28h keeps the rocket above the ground.

In order solve this problem with this paper's algorithm, the control time history has been approximated by a 24-stage zero-order hold. Additionally, the norms in constraints 28e and 28f have been approximated by functions with octagonally-shaped contours. The 2-norm's contours are spherical; so, this approximation introduces some modelling error. Fixing the end time seems unnatural, but it is necessary in order to be able to model the problem as an LP. An NP model is needed to handle the free-end-time case.

The LP code solved this problem in 55 min. on an IBM PC-AT with an 80287 coprocessor. It started from a first guess that violated inequality constraint 28h at every stage and that foolishly tried to maintain a constant thrust for the entire trajectory. Figures 1-3 compare the multi-stage LP solution with the exact continuous-time solution. In Fig. 1, the LP solution does better than the exact solution because of mis-modeling; it takes advantage of some extra thrust available at some points of the octagon norm. The thrust magnitude and angle time histories, Fig. 2 & 3, are both close to the exact solution, and the discrepancies are due to the same modeling error.

Figure 4 gives a 2-dimensional histogram of the constraint addition/deletion frequency. The left-hand horizontal axis indicates the stage at which the pseudo constraints are being relaxed in the special backwards-chaining process. The right-hand horizontal axis indicates the stage at which constraint additions and deletions are occurring during that relaxation process. The vertical axis gives the frequency of additions/deletions at the given right-hand-axis stage during pseudo-constraint relaxation at the given left-hand-axis stage. The extreme left-hand side of the figure shows no constraint addition or deletions -- none can occur at any stage before stage k-1 when the pseudo constraints at stage k are the ones being relaxed. The peaks on and near the center diagonal of the graph lend support to the conjecture that most of the constraint additions/deletions will happen at stages near the pseudo-constraint-relaxation stage. Note, however, that a moderate amount of constraint addition/deletion activity occurred near the terminal stage throughout the optimization. Nevertheless, the average factor update was relatively cheap. Altogether, about 800 rank-1 updates occur during the optimization. The total number of decision vectors in the time history is 312 -- 9 extra states are needed to model the impulse constraint in eq. 28f.

Conclusions

An algorithm has been presented for solving Dynamic Linear Programs. It takes advantage of the staircase structure of the active constraint matrix by factorizing it into staircase QL factors. These are derived in a stagewise fashion and play a role similar to that played by the time-varying matrix Riccati equation in multi-stage LQR theory. All of the usual linear programming functions have been implemented with the staircase QL factorization: decision vector solution, multiplier solution, and rank-1 updating. Each function has a computational complexity of $O(n^2N)$ or less, where n is a block dimension and N is the number of stages. This is the best that can be expected for dense blocks. Numerical stability is assured via the exclusive use of QL factors and is independent of pivoting strategies.

The algorithm is a modified active set implementation of the big M method with pseudo-constraint initialization. The modification restricts the set of non-optimal constraints that can be relaxed at one time to a single stage. This restriction gets iterated through all the stages in a backwards chain. Also, the modification chooses the

C 15

constraint that gets dropped in a way that assures optimality of the final portion of the solution time history. The modified strategy's goal is to reduce the average complexity of the rank-1 updates.

A 24-stage example problem has been solved. The algorithm solves the 312-dimensional problem in about 800 add/drop cycles, requiring 55 min. on an IBM PC-AT. The average update complexity is significantly reduced by the modified active set strategy.

Acknowledgement

This research was supported in part by the National Aeronautics and Space Administration under Grant No. NAG-1-1009.

References

1. Fletcher, R., **Practical Methods of Optimization**, 2nd Edition, J. Wiley & Sons, (New York 1987).
2. Dantzig, G.B., Personal Communication, Aug. 1989.
3. Fourer, R., Personal Communication, Aug. 1989.
4. Dantzig. G.B., "Programming of Interdependent Activities II: Mathematical Model," **Econometrica**, Vol. 17, 1949, pp. 200-211.
5. Dantzig, G.B., and Wolfe, P., "Decomposition Principle for Linear Programs," **Operations Research**, Vol. 8, Jan.-Feb. 1960, pp. 101-111.
6. Propoi, A., and Krivonozhko, V., "The Simplex Method for Dynamic Linear Programs," Proceedings of the IIASA Workshop on Large-Scale Linear Programming, June 2-6, 1980, (Laxenburg, Austria, 1981), pp. 299-363.
7. Fourer, R., "Solving Staircase Linear Programs by the Simplex Method," Proceedings of the IIASA Workshop on Large-Scale Linear Programming, June 2-6, 1980, (Laxenburg, Austria, 1981), pp. 179-259.
8. Murtaugh, B.A., and Saunders, M.A., "MINOS 5.0 Users Guide," Report SOL 83-20, Department of Operations Research, Stanford U., (Stanford, California, 1983).
9. Coleman, T.F., **Large Sparse Numerical Optimization**, Springer-Verlag, (New York, 1984)
10. Gill, P.E., Golub, G.H., Murray, W., and Saunders, M.A., "Methods for Modifying Matrix Factorizations," **Mathematics of Computation**, Vol. 28, April 1974, pp. 505-535.
11. Saunders, M., "Large-Scale Linear Programming Using the Cholesky Factorization," Ph.D. Dissertation, Stanford University, (Stanford, California, 1972).

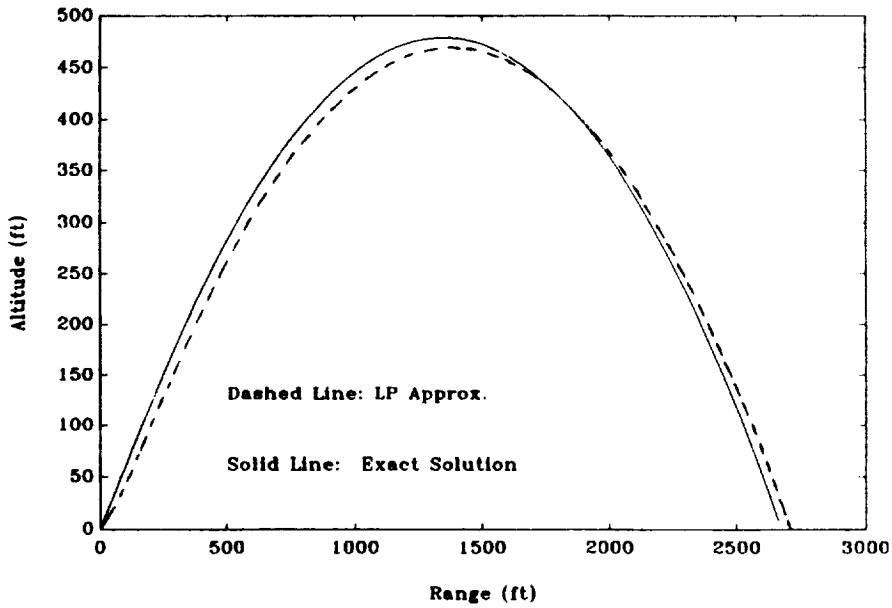


Fig. 1 Altitude vs. Range Trajectory for Ground-to-Ground Missile Range Maximization

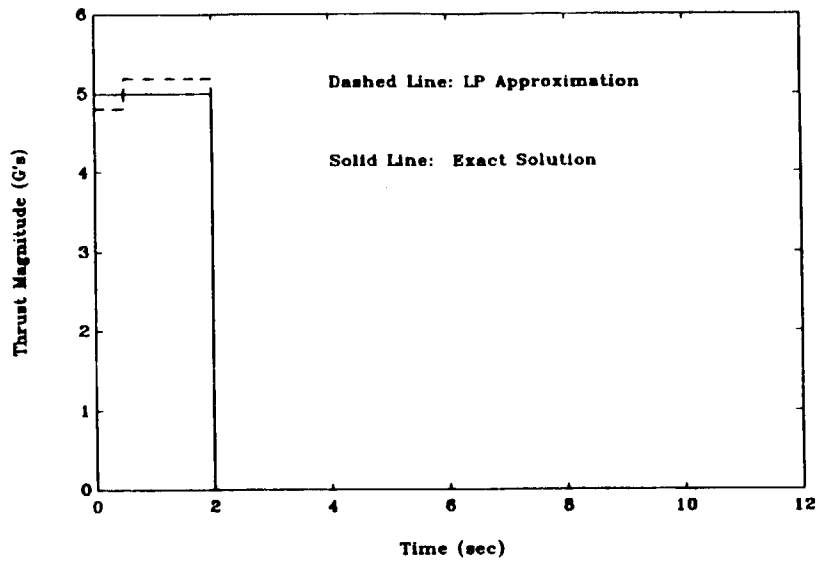


Fig. 2 Thrust (Acceleration) Magnitude Time History for Ground-to-Ground Missile Range Maximization

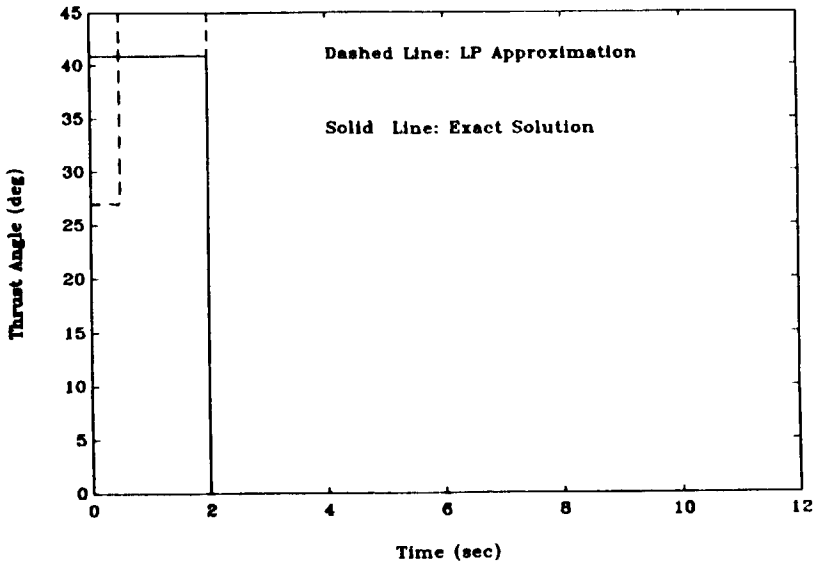


Fig. 3 Thrust Angle Time History for Ground-to-Ground Missile Range Maximization

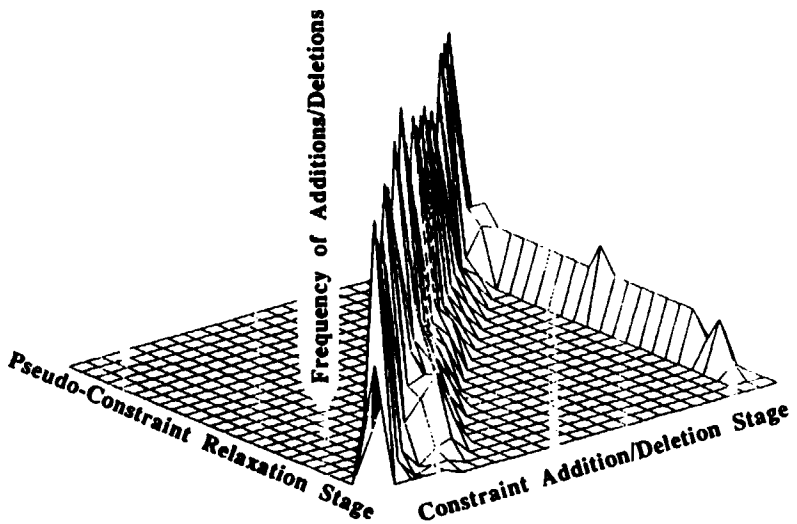


Fig. 4 Two-Dimensional Histogram of Frequency of Constraint Additions and Deletions at a Given Stage for a Given Pseudo-Constraint Relaxation Stage

A Finite Element Based Method for Solution of Optimal Control Problems

Robert R. Bless [1], Dewey H. Hodges [2], and Anthony J. Calise [3]

School of Aerospace Engineering
Georgia Institute of Technology, Atlanta, GA 30332

Abstract

A temporal finite element based on a mixed form of the Hamiltonian weak principle is presented for optimal control problems. The mixed form of this principle contains both states and costates as primary variables that are expanded in terms of elemental values and simple shape functions. Unlike other variational approaches to optimal control problems, however, time derivatives of the states and costates do not appear in the governing variational equation. Instead, the only quantities whose time derivatives appear therein are virtual states and virtual costates. Also noteworthy among characteristics of the finite element formulation is the fact that in the algebraic equations which contain costates, they appear linearly. Thus, the remaining equations can be solved iteratively without initial guesses for the costates; this reduces the size of the problem by about a factor of two. Numerical results are presented herein for an elementary trajectory optimization problem which show very good agreement with the exact solution along with excellent computational efficiency and self-starting capability. The goal of this work is to evaluate the feasibility of this approach for real-time guidance applications. To this end, a simplified two-stage, four-state model for an advanced launch vehicle application is presented which is suitable for finite element solution.

Introduction

Future space transportation and deployment needs are critically dependent on the development of reliable and economical launch vehicles that will provide flexible, routine access to orbit. A particular requirement now receiving attention is that for an advanced technology heavy-lift vehicle. Future space transportation systems will need to place large payloads – 100,000 to 150,000 pounds – into low Earth orbit at an order of magnitude lower cost per pound. Such systems will also require on-board algorithms that maximize system performance as measured by autonomy, mission flexibility, in-flight adaptability, reliability, accuracy and payload capability. They must be computationally efficient, robust, self-starting, and capable of functioning independently of ground control. Also, the algorithms must be designed with the anticipation that the launch vehicle will undergo evolutionary growth [1].

One approach to optimal guidance consists of repeatedly solving a two-point boundary-value problem that results from the traditional necessary conditions for optimality in an optimal control problem formulation. The vehicle state at discrete instants of time along a trajectory can be viewed as a new starting condition, and the remainder of the trajectory is reoptimized for that condition. The open loop optimal control is applied for a short interval of time, and feedback is introduced by reoptimization at the next time instant. This process presupposes that the two-point boundary-value problem can be reliably solved in a time interval that is small compared to the control update interval. Knowledge of the previous solution helps in providing a good starting point for the optimization process; however, no method has been demonstrated that can operate reliably in a real time environment.

This paper examines a finite element approach to addressing this problem. Hamilton's principle has traditionally been used in *analytical* mechanics as a method of obtaining the governing equations of motion for

[1] Graduate Research Assistant. Student Member, AIAA.
[2] Professor. Associate Fellow, AIAA.
[3] Professor. Member, AIAA.

continuous systems. In the 1970's a form of Hamilton's principle called Hamilton's law of varying action was first used by Bailey (see, for example, [2]) to obtain direct solutions to dynamics problems in the time domain, thus introducing Hamilton's principle into *computational* mechanics. During the last decade, it was shown by Borri *et al.* [3] and by Peters and Izadpanah [4] that these direct methods, when expressed in a weak form, could be competitive with numerical solution of the corresponding ordinary differential equations. Later work by Borri *et al.* [5] has shown that a mixed form of the weak principle has further computational advantages, namely that shape functions can be chosen from a far less restrictive class of functions.

In [6], Hodges and Bless have shown that optimal control problems can be solved in a virtually identical way to that of the mixed form of Hamilton's weak principle. Hence, the method as used in [6] and the present paper has been called the weak Hamiltonian method for optimal control problems. Finite element methods have some advantages over other solution procedures; one advantage is that finite element methods provide the possibility for development of algorithms which converge reliably. The present method, at least for the problems investigated to date, is essentially self-starting. This meets a key operational requirement for on-board algorithms. However, application of the finite element method to optimal control problems is rather new. For example, Patten [7] used a Ritz-Galerkin technique with Lagragean interpolation polynomials. One advantage of the present formulation is the allowance for a simpler choice of shape functions. The computational savings which may stem from this are now under investigation.

In this paper, a weak form governing optimal control problems is derived, and a finite element procedure is outlined for the solution of such problems. Numerical results for the solution of a simple trajectory optimization problem are presented and compared with the exact solution to demonstrate the accuracy and efficiency of the weak Hamiltonian finite element formulation. In anticipation of applying the present method to optimal guidance of a rocket booster, a simplified two-stage model suitable for this problem is presented. In [6] a one-stage model was analyzed and finite element results were compared to a numerical solution obtained using a multiple shooting method [8]. Of particular interest are the self-starting operation and the performance in terms of execution time and accuracy versus the number of elements used to represent the time span of the trajectory.

Weak Principle for Optimal Control

A definite analogy exists between the mixed formulation of Hamilton's weak principle in dynamics and the first variation of the performance index in optimal control theory. Specifically, there is an analogy between the generalized coordinates and generalized momenta in dynamics and the states and co-states in optimal control theory. Only a brief development of the weak Hamiltonian method for optimal control problems is presented herein. More details on the development and the analogy with dynamics problems may be found in [6].

General Development

We start with a performance index taken from Eq. (2.8.4) of Bryson and Ho [9]. Its first variation will be taken in a standard manner, except that states, costates, and controls will have arbitrary variations. Rather than setting its first variation equal to zero, however, it will be set equal to an expression which contains the terms that are necessary to transform all boundary conditions to the natural or "weak" type. The final weak form is then obtained by integration of this equation by parts in such a way that no derivatives of states or costates appear.

It should be noted that the fundamental relationships are not being changed. To make certain of this, we will ensure that the resulting formulation produces the Euler-Lagrange equations and boundary conditions which have already been established in optimal control theory (see, for example, [9], Eqs. 2.8.15 – 2.8.21).

In order to clearly understand what is meant by a "weak" formulation and the derivation of the weak formulation that is to follow, we first study a more simple problem. Let us start with a functional of the form

$$J = \int_A^B F(y, y', x)dx \tag{1}$$

where A and B are fixed numbers. The necessary conditions for an extremal are defined by

$$\delta J = \int_A^B \left(\frac{\partial F}{\partial y} \delta y + \frac{\partial F}{\partial y'} \delta y' \right) dx = 0 \tag{2}$$

Introducing $\partial F / \partial y' = f$ for notational convenience and integrating by parts we obtain

$$\int_A^B \left(\frac{\partial F}{\partial y} - f' \right) \delta y \, dx + f \delta y \Big|_A^B = 0 \tag{3}$$

The integrand in the above equation is the familiar Euler-Lagrange equation. The trailing term leads to the boundary conditions. If y is specified at $x = A$ or $x = B$, then $\delta y = 0$ at $x = A$ or $x = B$ respectively. This is referred to as a *strong* boundary condition. If y is not specified at one of the endpoints, then $f = 0$ at that endpoint. This is referred to as a natural or *weak* boundary condition. The key points to remember are that the trailing term itself is zero at each endpoint and that specifying y requires that $\delta y = 0$ at a point.

In our weak formulation, we want all the boundary conditions to be of the weak type. Thus, even if $y(A) = 0$ then $\delta y(A) \neq 0$. To allow for this mathematically, we introduce a new variable $\hat{f}$ which represents the discrete value of f at an endpoint. The variation of J is now set equal to $\hat{f} \delta y |_A^B$ yielding

$$\delta J = \int_A^B \left(\frac{\partial F}{\partial y} \delta y + f \delta y' \right) dx = \hat{f} \delta y \Big|_A^B \tag{4}$$

This is referred to as a weak form. If we integrate by parts, we obtain

$$\int_A^B \left(\frac{\partial F}{\partial y} - f' \right) \delta y \, dx = (\hat{f} - f) \, \delta y \Big|_A^B \tag{5}$$

Note that the Euler-Lagrange equation is the same as before and that the two boundary conditions are that $f = \hat{f}$ at the two endpoints; thus, the trailing terms are still constrained to be zero, but in a weak sense and δy need not ever vanish.

The advantages of the weak formulation are not apparent from the above discussion. However, when we apply this type of formulation to problems in dynamics (see, for example [6]), or optimal control theory [6], and use finite elements, then we have a powerful problem-solving tool.

Consider a system defined by a set of n states x and a set of m controls u. Furthermore, let the system be governed by a set of state equations of the form $\dot{x} = f(x, u, t)$. We may denote elements of the performance index, J, with an integrand $L(x, u, t)$ and a discrete function of the final states and time $\phi[x(t_f), t_f]$. In addition, any terminal constraints placed on the states may be placed in the set of q functions $\psi[x(t_f), t_f]$ and adjoined to the performance index by a set of q discrete Lagrange multipliers ν. Finally, we will adjoin the state equations to the performance index with a set of Lagrange multiplier functions $\lambda(t)$ which are referred to as costates. This yields a performance index of the form

$$J = \int_{t_0}^{t_f} (\lambda^T \dot{x} - L - \lambda^T f) \, dt - \phi \Big|_{t_f} - \nu^T \psi \Big|_{t_f} \tag{6}$$

Taking the first variation of J and setting it equal to an expression chosen so that all boundary conditions are of the weak type, one obtains

$$
\begin{aligned}
\delta J = & \int_{t_0}^{t_f} \left[\delta\lambda^T(\dot{x}-f) + \delta\dot{x}^T\lambda - \delta L - \delta f^T\lambda\right] dt \\
& + \delta t_f \left(\lambda^T\dot{x} - L - \lambda^T f - \frac{\partial\phi}{\partial t} - \nu^T\frac{\partial\psi}{\partial t}\right)\Big|_{t_f} \\
& - \delta x_f^T\hat{\lambda}_f - \delta\nu^T\psi\big|_{t_f} \\
& = \delta t_f\left(\lambda^T\dot{x}\right)\big|_{t_f} - \delta x_0^T\hat{\lambda}_0 - \delta\lambda^T(\hat{x}-x)\Big|_{t_0}^{t_f}
\end{aligned}
\tag{7}
$$

where

$$
\hat{\lambda}_f \equiv \left[\left(\frac{\partial\phi}{\partial x}\right)^T + \left(\frac{\partial\psi}{\partial x}\right)^T\nu\right]\Bigg|_{t_f} \tag{8}
$$

The right hand side of Eq. (7) contains terms necessary to form all of the proper boundary conditions as natural ones. The quantities $\hat{x}$ and $\hat{\lambda}$ are discrete values of the states and co-states at the initial (with subscript 0) and final times (with subscript f). Depending on the problem, these values will either be specified or left as unknowns.

From Eq. (7), we can directly write down a weak formulation. Before this is done, however, let us examine this expression to ensure that it produces the correct Euler-Lagrange equations and boundary conditions. Integrating the $\delta\dot{x}^T$ term in Eq. (7) by parts and expanding the variation of L, one obtains

$$
\begin{aligned}
& \int_{t_0}^{t_f}\Bigg\{\delta\lambda^T(\dot{x}-f) - \delta u^T\left[\left(\frac{\partial L}{\partial u}\right)^T + \left(\frac{\partial f}{\partial u}\right)^T\lambda\right] \\
& \quad - \delta x^T\left[\left(\frac{\partial L}{\partial x}\right)^T + \left(\frac{\partial f}{\partial x}\right)^T\lambda + \dot{\lambda}\right]\Bigg\}dt \\
& \quad - \delta\nu^T\psi\Big|_{t_f} - \delta t_f\left(L + \lambda^T f + \frac{\partial\phi}{\partial t} + \nu^T\frac{\partial\psi}{\partial t}\right)\Big|_{t_f} \\
& \quad + \delta x_f^T\left(\lambda_f - \hat{\lambda}_f\right) - \delta x_0^T\left(\lambda_0 - \hat{\lambda}_0\right) \\
& \quad + \delta\lambda_f^T(\hat{x}_f - x_f) - \delta\lambda_0^T(\hat{x}_0 - x_0) = 0
\end{aligned}
\tag{9}
$$

where x_0, λ_0, x_f, and λ_f represent the values of those functions at the initial and final times, respectively. The coefficients of $\delta\lambda^T$, δx^T, and δu^T in the integrand are the three correct Euler-Lagrange equations, Eqs. 2.8.15 - 2.8.17 from [9]. There are also six trailing terms in Eq. (9) from which the boundary conditions can be determined. The equations corresponding to the first four and the sixth of these terms correspond to the correct boundary conditions in [9]. Namely, the requirement for the coefficient of $\delta\nu^T$ to vanish yields Eq. (2.8.21). The requirement for the coefficient of δt_f to vanish is equivalent to Eq. (2.8.20). The requirement for the coefficient of δx_f^T to vanish shows that the final value of λ equals $\hat{\lambda}_f$ as given in Eq.(8), which corresponds to Eq. (2.8.19). If $\hat{\lambda}_0$ is chosen as zero, the requirement for the coefficient of δx_0^T to vanish requires the initial value of λ to equal zero; on the other hand, the requirement for the coefficient of $\delta\lambda_0^T$ to vanish requires the initial value of x to equal $\hat{x}_0$, in accordance with Eq. (2.8.18). Finally, the requirement for the coefficient of $\delta\lambda_f^T$ to vanish demands that the final value of x equal the discrete value $\hat{x}_f$; this has no counterpart in [9] since the elements of $\hat{x}_f$ are usually unknown.

Having satisfied our requirement that none of the fundamental equations are altered, we may now derive our weak formulation from Eq. (7). In order to allow for the simplest possible shape functions when we introduce a finite element discretization, we do not want time derivatives of x and λ to appear in the weak formulation. Therefore, we integrate the $\dot{x}$ term by parts in Eq. (7) yielding

$$
\begin{aligned}
&\int_{t_0}^{t_f} \left\{ \delta \dot{x}^T \lambda - \delta x^T \left[\left(\frac{\partial L}{\partial x} \right)^T + \left(\frac{\partial f}{\partial x} \right)^T \lambda \right] \right. \\
&\left. - \delta \dot{\lambda}^T x - \delta \lambda^T f - \delta u^T \left[\left(\frac{\partial L}{\partial u} \right)^T + \left(\frac{\partial f}{\partial u} \right)^T \lambda \right] \right\} dt \\
&- \delta t_f \left(L + \lambda^T f + \frac{\partial \phi}{\partial t} + \nu^T \frac{\partial \psi}{\partial t} \right) \Big|_{t_f} - \delta \nu^T \psi |_{t_f} \\
&- \delta x_f^T \hat{\lambda}_f + \delta x_0^T \hat{\lambda}_0 + \delta \lambda_f^T \hat{x}_f - \delta \lambda_0^T \hat{x}_0 = 0
\end{aligned}
\tag{10}
$$

This is the governing equation for the weak Hamiltonian method for optimal control problems. It will serve as the basis for the finite element discretization described below. It should be noted that normally one will encounter various types of inequality constraints in problems that deal with optimal control. Inequality constraints will be the subject of future research.

Finite Element Solution

Note in Eq. (10) that time derivatives of δx and $\delta \lambda$ are present. However, no time derivatives of x, λ, u or δu exist. Therefore, it is possible to implement linear shape functions for δx and $\delta \lambda$ within elements and constant shape functions for x, λ, u, and δu within elements.

For simplicity, let us break up the time interval into N segments of equal length $\Delta t = \frac{t_f - t_0}{N}$. Let the values of time be given by t_i for $i = 1, 2, \ldots, N+1$ at the points where the time interval is broken, the so-called nodes. Here $t_0 = t_1$ and $t_f = t_{N+1}$. Then, introduce a nondimensional elemental time τ such that

$$
\tau = \frac{t - t_i}{t_{i+1} - t_i} = \frac{t - t_i}{\Delta t} \qquad (0 \leq \tau \leq 1) \tag{11}
$$

Now, in accordance with the above guidelines, and letting $i = 1, 2, \ldots, N$, we can choose simple linear shape functions

$$
\begin{aligned}
\delta x &= \delta x_{(i)}(1 - \tau) + \delta x_{(i+1)} \tau \\
\delta \lambda &= \delta \lambda_{(i)}(1 - \tau) + \delta \lambda_{(i+1)} \tau
\end{aligned}
\tag{12}
$$

where the arbitrary, discrete virtual states and virtual costates are defined at every node point. (The nodal indices are enclosed in parentheses to avoid confusion with the state column matrix index.) For the states and costates, we can choose even simpler shape functions

$$
x = \begin{cases} \hat{x}_{(i)} & \text{if } \tau = 0; \\ \bar{x}_{(i)} & \text{if } 0 < \tau < 1; \\ \hat{x}_{(i+1)} & \text{if } \tau = 1 \end{cases}
\tag{13}
$$

and

$$
\lambda = \begin{cases} \hat{\lambda}_{(i)} & \text{if } \tau = 0; \\ \bar{\lambda}_{(i)} & \text{if } 0 < \tau < 1; \\ \hat{\lambda}_{(i+1)} & \text{if } \tau = 1 \end{cases}
\tag{14}
$$

where in both cases, $i = 1, 2, \ldots, N$. For u and δu, since their time derivatives do not appear in the formulation, we let

$$u = \bar{u}_{(i)}$$
$$\delta u = \delta \bar{u}_{(i)} \tag{15}$$

where $i = 1, 2, \ldots, N$.

Substitution of Eqs. (12) - (15) into Eq. (10) along with Eq. (11) and

$$t = t_i + \tau \Delta t$$
$$dt = \Delta t d\tau \tag{16}$$

yields a set of nonlinear algebraic equations which can be assembled in accordance with standard finite element practice. For the sake of brevity, these rather lengthy equations for the general case are not written out explicitly here. They are, however, written out in [6] wherein it is noted that there are $2n(N+1)$ equations for the states and costates, mN equations for the controls, q equations for the end-point constraints, and 1 equation for the unknown time equation. In the assembly process, $\hat{x}_{(i)}$ and $\hat{\lambda}_{(i)}$ drop out of the equations for $i = 2, \ldots, N$ (*i.e.*, for all but the ends of the time interval $t_0 = t_1$ and $t_f = t_{N+1}$). Thus, the total number of unknowns is $2n(N+2)$ for the states and costates, mN for the controls, q for the ν's, and 1 for the unknown time t_{N+1}. Thus, there are $2n$ more unknowns than there are equations, which allows for one to choose, say, $\hat{x}_0$ in accordance with physical constraints and $\hat{\lambda}_f$ in accordance with Eq. (8) and solve for the rest. The resulting equations may be explicitly perturbed to obtain the Jacobian and solved iteratively by a Newton-Raphson method or by any method that is suitable for nonlinear algebraic equations with very sparse Jacobians.

It is also pointed out in [6] that $n(N+1)$ of these algebraic equations contain the $n(N+1)$ unknown costates and that these equations are linear in the costates. Thus, the costates can be solved for symbolically in terms of the other unknowns, and the remaining equations can be solved *circumventing the need for initial estimates of the costates.* This decreases the size of the problem by approximately a factor of two.

Example: A Free-Final-Time Problem

In this section a relatively simple optimal control problem is solved by means of the Hamiltonian weak formulation, and the results are compared with the exact solution. Of particular interest is the computational effort for varying numbers of elements, the ability of the method to converge for various values of the system parameters without needing new initial estimates of the unknowns, and, most important, the accuracy of the method versus the number of elements.

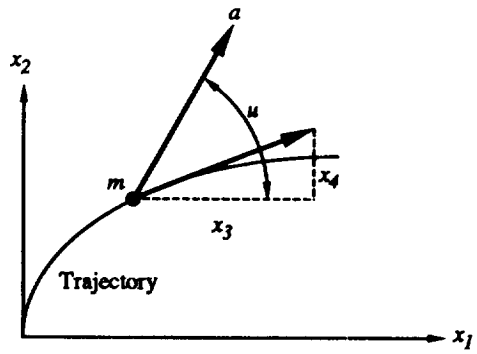


Fig. 1: Nomenclature for planar motion with thrust acceleration = a

Problem Statement

The problem is taken from [9], article 2.7, problem 9. A particle of mass m is acted upon by a thrust force of magnitude ma. Assuming planar motion and making use of an inertial coordinate system x_1 and x_2 as shown in Fig. 1, we may write the dynamical equations as

$$\dot{x} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} x + \begin{Bmatrix} 0 \\ 0 \\ a\cos u \\ a\sin u \end{Bmatrix} \tag{17}$$

where x_1 is the horizontal component of position, x_2 is the vertical component of position, x_3 is the horizontal component of velocity, and x_4 is the vertical component of velocity. The control is the thrust angle u, and $x(0) = \hat{x}_0 = \lfloor 0 \quad 0 \quad 0 \quad 0 \rfloor^T$. We want to obtain given values h of the final vertical component of position and U of the horizontal component of velocity. The final value of the vertical component of the velocity, must vanish, but we do not care what the final value of the horizontal component of position is. For the optimal control problem, the initial time is taken to be zero, and the final time T is to be minimized so that $\phi = 0$ and $L = 1$ which yields $J = T$. The elements of the end-point constraint function ψ must vanish where

$$\psi = \begin{Bmatrix} x_2 - h \\ x_3 - U \\ x_4 \end{Bmatrix} \tag{18}$$

In accordance with Eq. (8)

$$\hat{\lambda}_f = \begin{Bmatrix} 0 \\ \nu_1 \\ \nu_2 \\ \nu_3 \end{Bmatrix} \tag{19}$$

Substitution of these equations into the weak form, Eq. (10), yields a system of algebraic equations whose size depends on N.

Results and Discussion

These equations are solved by a Newton-Raphson algorithm with trivial initial guesses (that are never changed regardless of input parameters) for $N = 2$. These results are then used to obtain the initial guesses for arbitrary N by simple interpolation. In all results obtained to date for this problem, no additional steps are necessary to obtain results as accurate as desired.

Representative numerical results for x_1/h versus x_2/h are presented in Fig. 2 for a case with $\frac{ah}{U^2} = 0.75$. Also, the control angle u versus dimensionless time t/T is presented in Fig. 3. Based on other results, not shown due to space limitations, accuracy for the costates is comparable to that for the states and for u. It can easily be seen that $N = 8$ gives acceptable results for all variables.

It should be noted that the computer time on a Cyber 990 is only about 2 seconds for $N = 2$, $N = 4$, and $N = 8$ and 3 seconds for $N = 16$. Thus, it is relatively insensitive to N.

Overall, the method provides very accurate results for this problem with only a few elements and for minimal computational effort. Furthermore, in results that are not presented herein, the Hamiltonian is seen to converge nicely to zero all along the trajectory as the number of elements increases.

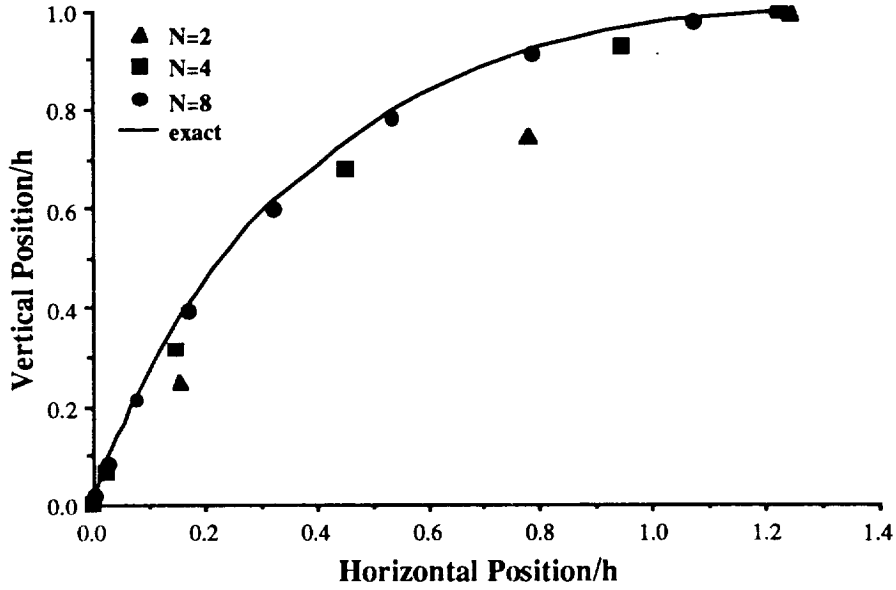


Fig. 2: Dimensionless vertical position x_2/h versus horizontal position x_1/h

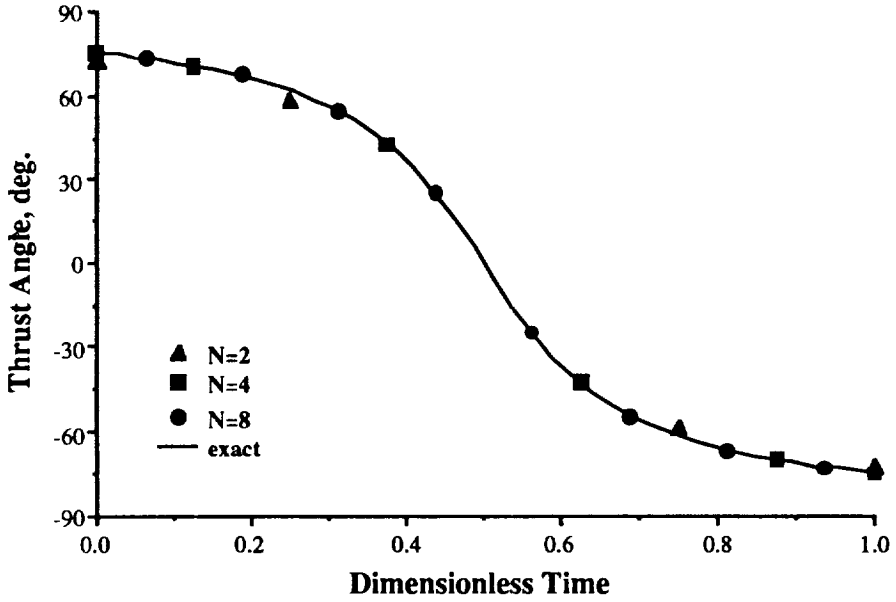


Fig. 3: Control angle u versus t/T

An Advanced Launch System Model

In this section, a model is presented which is suitable for evaluating the potential usefulness of the weak Hamiltonian finite element approach in real time guidance of an advanced launch system. A two-stage vehicle is considered that is simplified by not allowing for any inequality constraints.

We confine our attention to vertical plane dynamics of a vehicle flying over a spherical, non-rotating earth as depicted in Fig. 4. This results in the following state model for the states m (mass), h (height), E (energy per unit mass), and γ (flight-path angle):

$$\dot{m} = -\frac{T}{9.81 I_{sp}}; \qquad \dot{h} = V \sin\gamma; \qquad \dot{E} = \left(\frac{T-D}{m}\right) V$$
$$\dot{\gamma} = \left(\frac{T + qSC_{L\alpha}}{mV}\right)\alpha + \left(\frac{V}{r} - \frac{\mu}{r^2 V}\right)\cos\gamma \tag{20}$$

where T is the thrust, D is the drag, and V is the velocity. Here α, the angle of attack, has been adopted as a control variable.

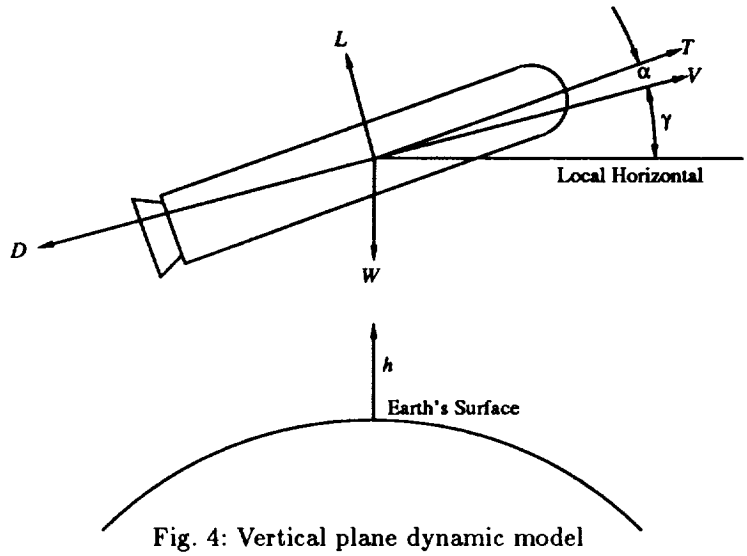


Fig. 4: Vertical plane dynamic model

The aerodynamic, propulsion, and atmospheric models are given by the following equations:

$$\begin{aligned}
T &= T_{vac} - A_e p \\
p &= p_0(1 - 0.00002255h)^{5.256} \quad \text{for } h \le 11000\text{m} \\
p &= p_{11} \exp\left(-\frac{h - 11000}{6350}\right) \quad \text{for } h \ge 11000\text{m} \\
r &= R_e + h; \quad V = \sqrt{2\left(E + \frac{\mu}{r}\right)}; \quad q = \frac{\rho V^2}{2} \\
\rho &= \rho_0 \exp\left(-\frac{h}{6700}\right); \quad M = \frac{V}{a} \\
D &= qS\left[C_{D0}(M) + \alpha^2 C_{N\alpha}(M)\right] \\
C_{L\alpha}(M) &= C_{N\alpha}(M) - C_{D0}(M) \\
a &= a_0\sqrt{1 - 0.00002255h} \quad \text{for } h \le 11000\text{m} \\
a &= 295.03\text{ms}^{-2} \quad \text{for } h > 11000\text{m}
\end{aligned} \tag{21}$$

The vehicle parameters chosen for this model are based on a Saturn IB launch vehicle SA-217 [10] and are

$$\begin{aligned}
I_{sp_1} &= 263.4\text{s}; & I_{sp_2} &= 430.4\text{s} \\
T_{vac_1} &= 8155800\text{N}; & T_{vac_2} &= 1186200\text{N} \\
A_{e_1} &= 8.47\text{m}^2; & A_{e_2} &= 5.29\text{m}^2; \quad S = 33.468\text{m}^2
\end{aligned} \tag{22}$$

where subscripts "1" and "2" refer to the first and second stages respectively. The aerodynamic coefficient data C_{D0} and $C_{N\alpha}$ are presented as functions of the Mach number M in Tables 1 and 2. The physical constants

used in the above model are the earth's gravitational constant $\mu = 3.9906 \times 10^{14}\text{m}^3\text{s}^{-2}$, the earth's mean radius $R_e = 6.378 \times 10^6\text{m}$, the sea-level atmospheric pressure $p_0 = 101320\text{Nm}^{-2}$, the atmospheric pressure at 11km $p_{11} = 22637\text{Nm}^{-2}$, the sea-level density of air $\rho_0 = 1.225\text{kg m}^{-3}$, and the sea-level speed of sound in air $a_0 = 340.3\text{ms}^{-1}$.

M	C_{D0}
0.20	1.00
0.75	0.45
0.98*	0.80
1.00	0.80
1.02*	0.80
3.50	0.20
6.00	0.02

Table 1: Aerodynamic coefficient C_{D0} versus Mach number
(* denotes a common end point of two quadratic polynomial curves)

M	$C_{N\alpha}$
0.00	6.20
0.50	6.35
0.98*	7.70
1.00	7.70
1.02*	7.70
2.50	5.20
4.40*	4.70
5.00	5.50
6.00	6.00

Table 2: Aerodynamic coefficient $C_{N\alpha}$ versus Mach number
(* denotes a common end point of two quadratic polynomial curves)

The performance index is

$$J = \phi|_{t_f} = m|_{t_f} \tag{23}$$

and the final time t_f is open. The initial conditions specified are $m(0) = 5.2 \times 10^5\text{kg}$, $h(0) = 1800\text{m}$, $E(0) = -6.25 \times 10^7\text{m}^2\text{s}^{-2}$, and $\gamma(0) = 75°$. The final energy is specified as $E(t_f) = -1.25 \times 10^7\text{m}^2\text{s}^{-2}$. The burnout mass of the first stage is 192000 kg and the drop-mass of the booster is 51000 kg.

For this two-stage model, we must modify our formulation somewhat. We must accomodate for the unknown staging time t_s, the constraint on the mass at t_s (as opposed to a constraint on the states at the final time), the jump in the mass at t_s, the jump in the mass costate at t_s, the condition for a continuous Hamiltonian at t_s (continuous since ϕ and ψ are not explicit functions of time), and finally the change in state equations at t_s (due to the change in thrust). We further point out that the control u is discontinuous at the staging time. However, since only discrete mid-point values of u are solved for, the jumps are allowed to occur automatically at the nodes.

The new performance index (with $L = 0$) is

$$J = \int_0^{t_s} [\lambda^T(\dot{x} - f_1)\,dt + \int_{t_s}^{t_f} [\lambda^T(\dot{x} - f_2)\,dt - \nu_1\psi_1\Big|_{t_s} - \phi\Big|_{t_f} - \nu_2\psi_2\Big|_{t_f} \tag{24}$$

where f_1 and f_2 represent the state equations before and after t_s respectively, $\psi_1 = \hat{m}(t_s^-) - 192000$ and $\psi_2 = \hat{E}(t_f) + 12500000$.

The weak formulation is derived exactly as before and the same shape functions can be employed. This time, however, we will discretize the time from 0 to t_s with N_1 elements and the time from t_s to t_f with N_2 elements. The algebraic equations shown in [6] for the weak formulation are readily modified to account for the various jumps and state equation discontinuities. The resulting equations are

$$
\begin{aligned}
&\sum_{i=1}^{N_1}\Bigg\{\delta x_i^T\left[-\bar{\lambda}_i - \frac{\Delta t_1}{2}\left(\frac{\partial \bar{f}_1}{\partial \bar{x}}\right)_i^T \bar{\lambda}_i\right] + \delta\lambda_i^T\left[\bar{x}_i - \frac{\Delta t_1}{2}\left(\bar{f}_1\right)_i\right] \\
&+\delta x_{i+1}^T\left[\bar{\lambda}_i - \frac{\Delta t_1}{2}\left(\frac{\partial \bar{f}_1}{\partial \bar{x}}\right)_i^T \bar{\lambda}_i\right] - \delta\lambda_{i+1}^T\left[\bar{x}_i + \frac{\Delta t_1}{2}\left(\bar{f}_1\right)_i\right] \\
&-\delta\bar{u}_i^T\left[\Delta t_1\left(\frac{\partial \bar{f}_1}{\partial \bar{u}}\right)_i^T \bar{\lambda}_i\right]\Bigg\} - \delta t_s\left[\hat{H}(t_s^+) - \hat{H}(t_s^-)\right] - \delta\nu_1\left(\hat{m}_{N_1+1}^- - 192000\right) \\
&+\delta x_1^T\hat{\lambda}_0 - \delta\lambda_1^T\hat{x}_0 - \delta x_{N_1+1}^T\left(\hat{\lambda}_{N_1+1}^- - \hat{\lambda}_{N_1+1}^+\right) + \delta\lambda_{N_1+1}^T\left(\hat{x}_{N_1+1}^- - \hat{x}_{N_1+1}^+\right) \\
&+\sum_{i=N_1+1}^{N_1+N_2}\Bigg\{\delta x_i^T\left[-\bar{\lambda}_i - \frac{\Delta t_2}{2}\left(\frac{\partial \bar{f}_2}{\partial \bar{x}}\right)_i^T \bar{\lambda}_i\right] + \delta\lambda_i^T\left[\bar{x}_i - \frac{\Delta t_2}{2}\left(\bar{f}_2\right)_i\right] \\
&+\delta x_{i+1}^T\left[\bar{\lambda}_i - \frac{\Delta t_2}{2}\left(\frac{\partial \bar{f}_2}{\partial \bar{x}}\right)_i^T \bar{\lambda}_i\right] - \delta\lambda_{i+1}^T\left[\bar{x}_i + \frac{\Delta t_2}{2}\left(\bar{f}_2\right)_i\right] \\
&-\delta\bar{u}_i^T\left[\Delta t_2\left(\frac{\partial \bar{f}_2}{\partial \bar{u}}\right)_i^T \bar{\lambda}_i\right]\Bigg\} - \delta t_f\left[\hat{H}(t_f)\right] - \delta\nu_2\left(\hat{E}_f + 12500000\right) \\
&+\delta\lambda_{N_1+N_2+1}^T(\hat{x}_f) - \delta x_{N_1+N_2+1}^T(\hat{\lambda}_f) = 0
\end{aligned}
\tag{25}
$$

From Eqs. (8) and (23) it is seen that $\hat{\lambda}_f$ is given by $\lfloor 1 \quad 0 \quad \nu_2 \quad 0 \rfloor^T$. Also, we note that the only jumps are in the mass state and the mass costate and these jumps are

$$
\begin{aligned}
\hat{m}(t_s^-) - \hat{m}(t_s^+) &= 51000 \\
\hat{\lambda}_m(t_s^-) - \hat{\lambda}_m(t_s^+) &= \nu_1
\end{aligned}
\tag{26}
$$

The finite element equations are solved using the method of Levenberg-Marquardt as coded in the IMSL subroutine ZXSSQ [11]. Running a case for a few elements generates a good approximation for larger numbers of elements. Initial guesses do not need to be very accurate, but the method is not nearly as computationally efficient as a Newton-Raphson procedure where sparsity in the Jacobian could be exploited.

In Figs. 5 and 6 numerical results for the ALS model are given for 4 and 8 elements in each time interval. (The number of elements in each interval is completely arbitrary.) In Fig. 5 the altitude profile is shown and the control history is shown in Fig. 6. From past experience with the one-stage model, we believe the $N_1 = 8$ and $N_2 = 8$ to be a converged result. Furthermore, we see that even the $N_1 = 4$ and $N_2 = 4$ result gives a reasonable approximation to the solution. It should be noted that these results are not realistic because of the absence of state constraints, and because we have large angles of attack (more than 30° at some points) even though we assumed small angles in the state equations. However, they do suffice to illustrate the power of the method.

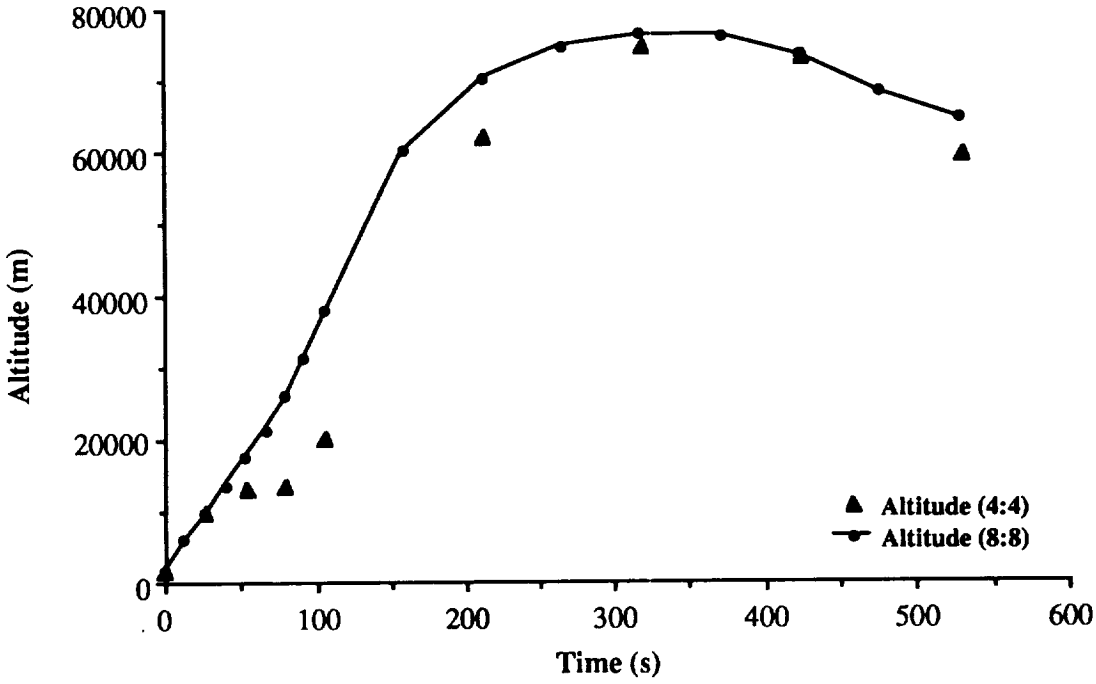


Fig. 5: Altitude profile versus time

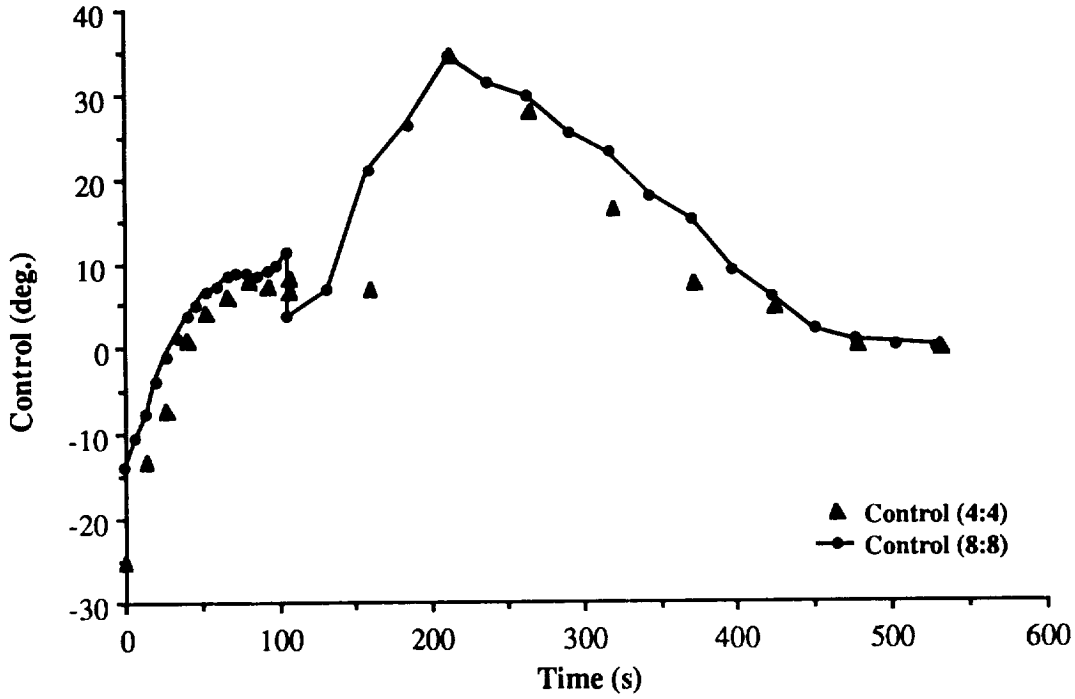


Fig. 6: Angle of attack profile versus time

As an indication of the accuracy of the method in a global sense, the Hamiltonian was observed to converge to zero (the exact answer) all along the trajectory. The finite element results are converging to the exact solution as N increases.

Conclusion

In this paper we present a weak Hamiltonian formulation for optimal control problems. Results are presented based on the weak Hamiltonian finite element formulation for a simple optimal control problem which show the method is reliable, efficient, and accurate. For this and other problems it is easily programmed with a self-starting algorithm.

To address the future needs of real-time guidance for future space transportation systems, we present a four-state model for the dynamics of mass, altitude, energy, and flight-path angle. The angle of attack is the control. The results show the power and efficacy of the present approach. It has several advantages for applications in real-time guidance. It is not only reliable and efficient but has excellent self-starting capability. Furthermore, initial guesses of the costates are not needed, and the method exhibits a graceful degradation in performance with reduction in number of elements.

For future research we intend to concentrate on improving computational efficiency by exploitation of the relatively significant level of sparsity in the Jacobian. We also will incorporate inequality constraints, and begin to address the avoidance of atmospheric anomalies.

Acknowledgement

This work was supported by NASA Grant NAG-1-939 of which Dr. D. D. Moerder is the technical monitor.

References

1. Hardtla, J. W., Piehler, M. J., and Bradt, J. E., "Guidance Requirements for Future Launch Vehicles," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Paper No. 87-2462, August, 1987.
2. Bailey, C. D., "Application of Hamilton's Law of Varying Action," *AIAA Journal*, Vol. 13, 1975, pp. 433 - 451.
3. Borri, M., *et al.*, "Dynamic Response of Mechanical Systems by a Weak Hamiltonian Formulation," *Computers and Structures*, Vol. 20, No. 1 - 3, 1985, pp. 495 - 508.
4. Peters, David A., and Izadpanah, Amir, "*hp*-Version Finite Elements for the Space-Time Domain," *Computational Mechanics*, Vol. 3, pp. 73 - 88, 1988.
5. Borri, M., *et al.*, "Primal and Mixed Forms of Hamilton's Principle for Constrained and Flexible Dynamical Systems: Numerical Studies," ARO/AFOSR Workshop on Nonlinear Dynamics, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, June 1 - 3, 1988.
6. Hodges, Dewey H., and Bless, Robert R., "A Weak Hamiltonian Finite Element Method for Optimal Control Problems," *Journal of Guidance, Control, and Dynamics*, submitted for publication, 1989.
7. Patten, William Neff, "Near Optimal Feedback Control for Nonlinear Aerodynamic Systems with an Application to the High-Angle-of-Attack Wing Rock Problem," AIAA Paper 88-4052-CP, 1988.
8. Bulirsch, R., "The Multiple Shooting Method for Numerical Solution of Nonlinear Boundary Value Problems and Optimal Control Problems (in German)," Carl-Cranz-Gesellschaft, Tech. Rpt., Heidelberg, 1971.
9. Bryson, Arthur E. Jr., and Ho, Yu-Chi, *Applied Optimal Control*, Blaisdell Publishing Company, Waltham, Massachusetts, 1969, Chapter 2.
10. Anon., "Saturn IB SA-217 Reference Launch Vehicle," NASA TM X-53686, 1968.
11. Anon., The International Mathematical and Statistical Library, IMSL Inc., Houston, Texas, Ch. Z, 1984.

A Methodology For Formulating A Minimal Uncertainty Model For Robust Control System Design and Analysis

Christine M. Belcastro
MS 489
NASA Langley Research Center
Hampton, VA. 23665

B.-C. Chang
ME&M Dept.
Drexel University
Philadelphia, PA. 19104

Robert Fischl
ECE Dept.
Drexel University
Philadelphia, PA. 19104

Abstract

In the design and analysis of robust control systems for uncertain plants, the technique of formulating what is termed an "M-Δ model" has become widely accepted and applied in the robust control literature. The "M" represents the transfer function matrix M(s) of the nominal system, and "Δ" represents an uncertainty matrix acting on M(s). The uncertainty can arise from various sources, such as structured uncertainty from parameter variations or multiple unstructured uncertainties from unmodeled dynamics and other neglected phenomena. In general, Δ is a block diagonal matrix, and for real parameter variations the diagonal elements are real. As stated in the literature, this structure can always be formed for any linear interconnection of inputs, outputs, transfer functions, parameter variations, and perturbations. However, very little of the literature addresses methods for obtaining this structure, and none of this literature (to the authors' knowledge) addresses a general methodology for obtaining a minimal M-Δ model for a wide class of uncertainty. Since having a Δ matrix of minimum order would improve the efficiency of structured singular value (or multivariable stability margin) computations, a method of obtaining a minimal M-Δ model would be useful. This paper presents a generalized method of obtaining a minimal M-Δ structure for systems with real parameter variations.

1. Introduction

Robust control theory for both analysis and design has been the subject of a vast amount of research in this decade [1-35]. In particular, robust stability and performance have been emphasized in much of this work, as, for example, in the development of H^∞ control theory [10-15, 19-23]. Moreover, the development of robust control system design and analysis techniques for unstructured [1-9, 13, 19, 21] as well as structured [16-35] plant uncertainty continues to be the subject of much research - particularly the latter. Unstructured plant uncertainty arises from unmodeled dynamics and other neglected phenomena, and is complex in form. This uncertainty is called "unstructured" because it is represented as a norm-bounded perturbation with no particular assumed structure. Plant uncertainty is called "structured" when there is real parameter uncertainty in the plant model, or when there is unstructured complex uncertainty occurring in the system at multiple points simultaneously. Plant parameter uncertainty can arise from modeling errors (which usually result from assumptions and simplifications made during the modeling process and/or from the unavailability of dynamic data on which the model is based), or from parameter variations that occur during system operation.

Robust control design and analysis methods for systems with unstructured uncertainty is accomplished via singular value techniques [1-9, 21]. For systems with structured plant uncertainty, however, the structured singular value (SSV) [16-27] or multivariable stability margin (MSM) [28-33] must be used. In order to compute the SSV or MSM, the system is usually represented in terms of an M-Δ model. The "M" represents the transfer function matrix M(s) of the nominal system, and "Δ" represents an uncertainty matrix acting on M(s). In general, Δ is a block diagonal matrix, and for real parameter uncertainites the diagonal elements are real. As indicated in the literature [17,18,20,28], this structure can always be formed for any linear interconnection of inputs, outputs, transfer functions, parameter variations, and perturbations. However, very little of the literature discusses methods for obtaining an M-Δ model. For unstructured uncertainites, this model is very easy to obtain. However, for real parameter variations, forming an M-Δ model can be very difficult. In [29], De Gaston and Safonov present an M-Δ model for a third-order transfer function with uncertainty in the location of its two real poles and in its gain factor. Although the given M-Δ model is easily obtained for this simple example, other examples do not yield such a

straight-forward result. A general state model of M(s) for additive real perturbations in the system A matrix (where A is assumed to be closed-loop) is discussed in [34]. Unfortunately, this model is not general enough for many examples, since system uncertainty is restricted to the A matrix and the uncertainty class is restricted to be linear. Morton and McAfoos [26] present a general method for obtaining an M-Δ model for linear (affine) real perturbations in the system matrices (A,B,C,D) of the open-loop plant state model. In this model, an interconnection matrix P(s) is constucted first for separating the uncertainties from the nominal plant, and then M(s) is formed by closing the feedback loop. The M-Δ model thus formed can be used in performing robustness analysis of a previously determined control system. If the feedback loop is not closed, μ-synthesis techniques [19-21] can be applied to the M-Δ model for robust control system design. Morton's result essentially reduces to that of [34] when the perturbations occur only in the A matrix (and the A matrix of [34] is assumed to be open-loop). An algorithm for easily computing M(s) based on Morton's result is presented in [35]. Although this method of constructing an M-Δ model is general for linear uncertainties, many realistic problems require a more general class of uncertainties. Furthermore, no consideration is given to obtaining a minimal M-Δ model, where "minimal" refers to the dimension of the Δ (or M) matrix. Since the M-Δ model is a nonunique representation, it would be prudent to obtain one of minimal dimension so that the complexity of the SSV or MSM computations during robust control system design or analysis could be minimized. However, none of the literature (to the authors' knowledge) addresses the issue of minimality.

This paper presents a methodology for constructing a minimal M-Δ model for systems with real parametric multilinear uncertainties, where the term "multilinear" is defined as follows:

Definition: A function is multilinear if the functional form is linear (affine) when any variable is allowed to vary while the others remain fixed. For example, $f(a,b,c) = a + ab + bc + abc$ is a multilinear function.

Thus, the allowance of multilinear functions of the uncertain parameters provides a means of handling cross-terms in the transfer function coefficients. A procedure is proposed for obtaining this model in state-space form for uncertain single-input single-output (SISO) systems, given the system transfer function in terms of the uncertain parameters. An extension of this result to multiple-input multiple-output (MIMO) systems will be given in a subsequent publication. In this development, M(s) will represent the nominal open-loop plant, so that the resulting M-Δ model may be used for robust control system analysis or design. The state-space form used in modeling M(s) is an extension of Morton's result for real parametric linear (affine) uncertainties [26]. The paper is divided into the following sections. A formal statement of the problem to be solved in this paper is presented in Section 2, followed by a discussion of minimality considerations in Section 3. The approach is presented in Section 4, a proposed solution to the problem is presented in Section 5, and the proposed procedure for finding a minimal M-Δ model is summarized in Section 6. Several examples demonstraing the proposed solution are given next in Section 7, followed by some concluding remarks in Section 8.

2. Problem Statement

Given the transfer function of an uncertain system, G(s,δ), in either factored or unfactored form, as a function of the uncertain real parameters, δ, **find** a minimal M-Δ model of the form depicted below in Figure 1:

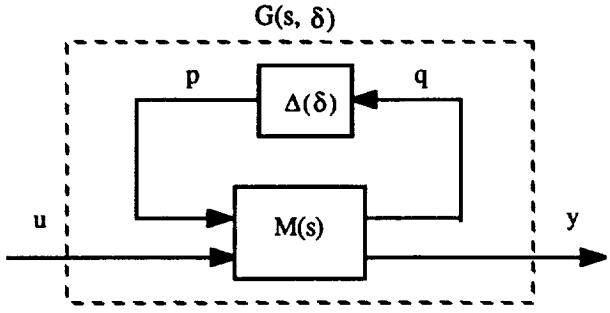


Figure 1. Block diagram of the General M - Δ Model

such that:

1. The diagonal uncertainty matrix, Δ, is of minimal dimension.
2. The model of the nominal plant, M(s), is in state-space form.

The model must handle multilinear uncertainty functions in any or all of the transfer function coefficients. In order to construct a minimal M-Δ model, the dimension of the Δ matrix must be minimized. Hence, factors which have been found to affect the dimension of the M-Δ model will be discussed next, followed by the approach used in forming a solution to this problem.

3. Minimality Considerations

In constructing an M-Δ model of an uncertain system, the Δ matrix can become unnecessarily large due to repeated uncertain parameters on its main diagonal. It is therefore of interest to examine the factors which can cause this repetition, so that the number of repeated uncertain parameters can be minimized. A factor which can be shown to cause unnecessary repetition in the Δ matrix is the particular realization used in representing the system. Examples can be constructed which demonstrate this effect, and it appears that a cascade realization (and, in particular, cascaded uncertain real poles and zeros) is a desirable form for obtaining a minimal M-Δ model. Thus, a general cascade-form realization will be part of the approach taken in constructing a minimal M-Δ model. A problem arises, however, in that some transfer functions have a form which precludes cascading uncertain real poles or zeros, such as:

$$G(s,\delta) = \frac{b_1 s^2 + b_2 s + b_3}{(s+\theta_1)(s+\theta_2)} \quad , \quad G(s,\delta) = \frac{(s+\theta_1)(s+\theta_2)}{s^2 + a_1 s + a_2}$$

where θ_1 and θ_2 are assumed here to be uncertain (and hence a function of δ). Cascading the poles and zeros for either case would result in improper transfer function blocks to be realized. For these cases, it is unavoidable for the minimal Δ matrix to have repeated uncertain parameters on the main diagonal. However, for each inseparable pole or zero pair it is only necessary to repeat one uncertain parameter. This issue will be addressed in the proposed solution, and a minimal M-Δ model for the first transfer function above will be given as an example.

Another factor which affects the dimension of the M-Δ model is the form of the coefficients in the system transfer function. If any of the coefficients is a nonlinear function of the uncertain parameters instead of a multilinear function (e.g., there are squared uncertain terms in any of the coefficients), then extra dependent uncertain parameters must be defined in order to represent these terms in a multilinear form. For example, δ_1^2 would be represented as $\delta_1\delta_2$ where $\delta_2 = \delta_1$, and both δ_1 and δ_2 would appear in the Δ matrix. Thus, for this case, it is again necessary that the minimal Δ matrix contain repeated uncertain parameters on its main diagonal. An example illustrating this situation will be presented later.

These issues are addressed in the proposed solution for constructing a minimal M-Δ model. The approach taken in forming this solution is described in the next section.

4. Approach

Based on the problem definition and the minimality considerations outlined above, several issues will be addressed in forming a solution to the problem of constructing a minimal M-Δ model given the transfer function of an uncertain system. First, a general cascade-form realization will be found which can be used to obtain a minimal M-Δ model. Second, the minimal Δ matrix will be determined for any uncertain system such that extra dependent parameters are assigned to account for inseparable pairs of uncertain real poles or zeros as well as non-multilinear (e.g., squared) terms. Third, a method of obtaining a state-space realization of M(s) for any uncertain system will be found. Therefore, the proposed approach for constructing a minimal M-Δ model is given as follows:

1. Obtain a cascade-form realization of the system so that the state-space uncertain model can be written as:

$$\begin{aligned} \dot{x} &= A\,x + B\,u \\ y &= C\,x + D\,u \end{aligned} \qquad (1)$$

$$\text{where: } A = A_o + [\Delta A],\ B = B_o + [\Delta B],\ C = C_o + [\Delta C],\ D = D_o + [\Delta D] \qquad (2)$$

The terms with the "o" subscript (A_o, B_o, C_o, D_o) represent the nominal matrix components, and the "Δ" terms $(\Delta A, \Delta B, \Delta C, \Delta D)$ represent the uncertain matrix components. To eliminate confusion of the Δ notation, the diagonal uncertainty matrix, Δ, of the M-Δ model will be represented as [Δ], and the ΔA, ΔB, ΔC, and ΔD matrices will be represented as [ΔA], [ΔB], [ΔC], and [ΔD] wherever clarification is required.

2. Obtain a minimal M-Δ model as described in the problem definition and pictured in Figure 1, where:

a. The minimal uncertainty matrix, Δ, is defined as:

$$\Delta = \text{diag}[\delta_1, \delta_2, \delta_3, \ldots, \delta_m] = \text{diag}[\delta_I, \delta_D] = \text{diag}[\delta] \tag{3}$$

where: $\Delta \in R^{m \times m}$, $\delta_I \in R^{m_I}$, $\delta_D \in R^{m_D}$, $\delta \in R^m$

and: m = the minimal number of uncertain parameters
m_I = the number of independent parameters given in $G(s,\delta)$
m_D = the minimal number of dependent (or repeated) parameters

Also: $$p = [\Delta]\, q \tag{4}$$

where: p = the uncertain parameters input to M(s), $p \in R^{m_p}$
q = the uncertain variables output from M(s), $q \in R^{m_q}$

Since an M-Δ model is minimal if the dimension of the Δ matrix, m, is minimal, where m depends on m_I and m_D, with m_I being given and fixed, a formal definition of a minimal M-Δ model can be stated as follows:

Definition: An M-Δ model is minimal if m_D - i.e., the number of dependent (or repeated) parameters in the Δ matrix - is minimal (or zero, if possible).

b. The state-space model of the nominal plant, M(s), is an extension of Morton's result [26] and has the following form:

$$\begin{aligned} \dot{x} &= A_o x + \begin{bmatrix} B_{xp} & B_o \end{bmatrix} \begin{bmatrix} p \\ u \end{bmatrix} \\ \begin{bmatrix} q \\ y \end{bmatrix} &= \begin{bmatrix} C_{qx} \\ C_o \end{bmatrix} x + \begin{bmatrix} D_{qp} & D_{qu} \\ D_{yp} & D_o \end{bmatrix} \begin{bmatrix} p \\ u \end{bmatrix} \end{aligned} \tag{5}$$

where B_{xp}, C_{qx}, D_{qp}, D_{qu}, and D_{yp} are constant matrices. Thus, M(s) can also be written in the equivalent shorthand notation defined as follows:

$$M(s) = \begin{bmatrix} M_{11}(s) & M_{12}(s) \\ M_{21}(s) & M_{22}(s) \end{bmatrix} := \left[\begin{array}{c|cc} A_o & B_{xp} & B_o \\ \hline C_{qx} & D_{qp} & D_{qu} \\ C_o & D_{yp} & D_o \end{array}\right] \tag{6}$$

where:
$$\begin{aligned} M_{11}(s) &= q(s)/p(s) = C_{qx}(sI - A_o)^{-1} B_{xp} + D_{qp} \\ M_{12}(s) &= y(s)/p(s) = C_{qx}(sI - A_o)^{-1} B_o + D_{qu} \\ M_{21}(s) &= q(s)/u(s) = C_o(sI - A_o)^{-1} B_{xp} + D_{yp} \\ M_{22}(s) &= y(s)/u(s) = C_o(sI - A_o)^{-1} B_o + D_o \end{aligned} \tag{7}$$

It should be noted that in [26] the D_{qp} matrix was required to be zero. In this paper, however, D_{qp} is allowed to be nonzero in order to model the multilinear (cross-product) uncertain terms.

The results for constructing a minimal M-Δ model via this approach are presented in the next section.

5. Proposed Solution

The proposed solution will be presented in two parts: the results for obtaining a cascade-form realization of the uncertain system will be summarized first, followed by the results for obtaining the state-space realization of a minimal M-Δ model.

5.1 Cascade-Form Realization

Given the transfer function of an uncertain system in terms of its uncertain parameters, $G(s,\delta)$, it is desired to realize the system in a cascade form of first- and second-order subsystems. Thus, if the transfer function is given in unfactored form, the numerator and denominator polynomials must be factored into first- and second-order terms. The given transfer function will then be represented as follows:

$$G(s,\delta) = K_y(\delta)\, G_C(s,\delta)\, G_R(s,\delta)\, K_u(\delta) \tag{8}$$

where K_u and K_y represent input and output gain terms, respectively, and G_R and G_C represent the real (first-order) and complex (second-order) transfer function components, respectively. Then :

$$G_R(s,\delta) = G_{R_k}(s,\delta)\, G_{R_{k-1}}(s,\delta) \ldots G_{R_2}(s,\delta)\, G_{R_1}(s,\delta) \tag{9}$$

$$G_C(s,\delta) = G_{C_l}(s,\delta)\, G_{C_{l-1}}(s,\delta) \ldots G_{C_2}(s,\delta)\, G_{C_1}(s,\delta) \tag{10}$$

$$G_{R_i}(s,\delta) = \frac{\beta_{2i-1}\, s + \beta_{2i}}{s + \alpha_i} \tag{11}$$

$$G_{C_i}(s,\delta) = \frac{b_{3i-2}\, s^2 + b_{3i-1}\, s + b_{3i}}{s^2 + a_{2i-1}\, s + a_{2i}} \tag{12}$$

and:

k = number of real (first-order) blocks

l = number of complex (second-order) blocks.

Any or all of these transfer function coefficients may be uncertain. The uncertainty may arise from either the coefficient itself being uncertain, or from the coefficient being a function of one or more uncertain variables. Therefore, for either case, any of the coefficients may be a function of δ. Furthermore, the uncertain variables may have either an additive or multiplicative form:

$$\varepsilon = \varepsilon_0 + \delta_\varepsilon \quad , \quad \varepsilon = \varepsilon_0 (1 + \delta_\varepsilon) \tag{13}$$

The following cascade-form state-space realization of this system is proposed:

$$G(s) = \left[\begin{array}{cc|c} A_R & 0 & B_R K_u \\ B_C C_R & A_C & E_C D_R K_u \\ \hline K_y D_C C_R & K_y C_C & K_y D_C D_R K_u \end{array}\right] \tag{14}$$

where:

$$A_R = \begin{bmatrix} A_{R_1} & 0 & \cdots & 0 & 0 \\ B_{R_2}C_{R_1} & A_{R_2} & \cdots & 0 & 0 \\ B_{R_3}D_{R_2}C_{R_1} & B_{R_3}C_{R_2} & \cdots & 0 & 0 \\ B_{R_4}D_{R_3}D_{R_2}C_{R_1} & B_{R_4}D_{R_3}C_{R_2} & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ B_{R_{k-1}}D_{R_{k-2}}\cdots D_{R_2}C_{R_1} & B_{R_{k-1}}D_{R_{k-2}}\cdots D_{R_3}C_{R_2} & \cdots & A_{R_{k-1}} & 0 \\ E_{R_k}D_{R_{k-1}}\cdots D_{R_2}C_{R_1} & B_{R_k}D_{R_{k-1}}\cdots D_{R_3}C_{R_2} & \cdots & B_{R_k}C_{R_{k-1}} & A_{R_k} \end{bmatrix} \quad (15)$$

$$B_R = \begin{bmatrix} B_{R_1} \\ B_{R_2}D_{R_1} \\ B_{R_3}D_{R_2}D_{R_1} \\ B_{R_4}D_{R_3}D_{R_2}D_{R_1} \\ \vdots \\ B_{R_{k-1}}D_{R_{k-2}}\cdots D_{R_2}D_{R_1} \\ B_{R_k}D_{R_{k-1}}\cdots D_{R_2}D_{R_1} \end{bmatrix} \quad (16)$$

$$C_R = \begin{bmatrix} D_{R_k}D_{R_{k-1}}\cdots D_{R_2}C_{R_1} & D_{R_k}D_{R_{k-1}}\cdots D_{R_3}C_{R_2} & \cdots & D_{R_k}C_{R_{k-1}}C_{R_k} \end{bmatrix} \quad (17)$$

$$D_R = \begin{bmatrix} D_{R_k}D_{R_{k-1}}\cdots D_{R_2}D_{R_1} \end{bmatrix} \quad (18)$$

The A_C, B_C, C_C, and D_C matrices have the exact same form as (15) - (18), except that the subscripts "R" and "k" are replaced by "C" and "l", respectively. The submatrices are defined as follows:

$$A_{R_i} = -\alpha_i \qquad B_{R_i} = 1$$
$$C_{R_i} = \beta_{2i} - \alpha_i \beta_{2i-1} \qquad D_{R_i} = \beta_{2i-1} \quad (19)$$

$$A_{C_i} = \begin{bmatrix} 0 & 1 \\ -a_{2i} & -a_{2i-1} \end{bmatrix} \qquad B_{C_i} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$
$$C_{C_i} = \begin{bmatrix} (b_{3i} - a_{2i}\, b_{3i-2}) & (b_{3i-1} - a_{2i-1}\, b_{3i-2}) \end{bmatrix} \qquad D_{C_i} = b_{3i-2} \quad (20)$$

The realizations $\{ A_{R_i}, B_{R_i}, C_{R_i}, D_{R_i} \}$ and $\{ A_{C_i}, B_{C_i}, C_{C_i}, D_{C_i} \}$ represent the i^{th} real (first-order) and complex (second-order) systems $G_{R_i}(s,\delta)$ and $G_{C_i}(s,\delta)$, respectively. Thus, for the real subsystems, i = 1, 2, ... , k, and for the complex subsystems, i = 1, 2, ... , l.

The resulting cascade-form realization of the uncertain system is therefore given from (14) as:

$$A = \begin{bmatrix} A_R & 0 \\ B_C C_R & A_C \end{bmatrix} \qquad B = \begin{bmatrix} B_R K_u \\ B_C D_R K_u \end{bmatrix}$$
$$C = \begin{bmatrix} K_y D_C C_R & K_y C_C \end{bmatrix} \qquad D = K_y D_C D_R K_u \tag{21}$$

The above model is a general cascade-form realization for any uncertain open-loop SISO transfer function. The model does not, however, handle nonmonic denominator polynomials with uncertain leading coefficients. This would result in fractional (i.e., rational) matrix elements in the realization with uncertain parameters in the denominator of these elements. For real uncertain poles or zeros, two factors determine whether the real (first-order) or complex (second-order) block form should be used. The first is the nature of the uncertainty associated with these terms, and the second is the form of the transfer function. If the real pole or zero locations are the uncertain parameters and the transfer function form allows these poles or zeros to be separated out, then the real block form should be used. If the transfer function form does not allow this separation, then the complex block form must be used. Furthermore, if there is a pair of uncertain poles or zeros that cannot be cascaded, then the resulting minimum Δ matrix will have a repeating parameter on the main diagonal for each inseparable pole or zero pair. Alternatively, if the coefficients of the second-order polynomial associated with the real poles are the uncertain parameters, then the complex block form should be used. These cases will be illustrated in the **Examples** section of this paper. The formulation of the minimal M-Δ model will be presented next.

5.2 Minimal M-Δ Model

In formulating the minimal M-Δ model, the minimal Δ matrix must be determined first, followed by the state-space realization of M(s). Thus, the results for formulating this model will be presented in this order.

5.2.1 Minimal Δ Matrix

The minimal Δ matrix is defined as in (3) with:

$$m = m_I + m_D \tag{22}$$

where m_I is the number of independent uncertain parameters, and m_D is the number of dependent uncertain parameters that must be added. The uncertain independent parameters are those defined in $G(s,\delta)$. However, as discussed previously, the dependent uncertain parameters are those independent parameters that must be repeated due to non-multilinear terms in the transfer function coefficients and/or pairs of uncertain real poles or zeros that cannot be cascaded. Thus, for Δ to be minimal, m_D (or δ_D) should be minimized. It can be shown that if the system transfer function is formed from a given minimal M-Δ model of an uncertain system, the coefficients of the numerator and denominator polynomials will be multilinear functions of the uncertain parameters. Unfortunately, the converse is not necessarily true in general because of the dependence of the M-Δ model on the realization used for the plant. If the general cascade-form realization posed in this paper is used, however, the multilinear form of the transfer function coefficients can be used to establish that $m = m_I$ (i.e., $m_D = 0$), unless there are real uncertain pairs of poles or zeros that cannot be cascaded. Furthermore, it can be shown that if the coefficients of all the factors of the numerator and denominator polynomials are multilinear functions, then the coefficients of the expanded polynomials will also be multilinear. However, if there are non-multilinear uncertain terms in the transfer function, then dependent parameters must be defined (and added to Δ) to represent the non-multilinear term in a multilinear form. Moreover, if the non-multilinear term is of the form δ^n, then n-1 dependent parameters must be defined. If there are pairs of real uncertain poles or zeros that cannot be cascaded, then one additional dependent parameter must be added for each pair, and the dependent parameter can be either of the uncertain real parameters in the pair. Therefore, the number m, as determined by these rules, is the minimal dimension of the Δ matrix for the uncertainty class considered in this paper. Once this minimal dimension is determined, the Δ matrix can be defined as a diagonal matrix, as in (3), with the specified uncertain parameters on the main diagonal. Examples which illustrate these cases will be presented later in Section 6.

5.2.2 State-Space Realization of M(s)

Once the cascade-form realization has been determined, the system can be modeled as in (1) and (2), where $[\Delta A]$, $[\Delta B]$, $[\Delta C]$, and $[\Delta D]$ are known functions of the uncertain parameters. Since any non-multilinear terms have

been redefined in a multilinear form when the minimal Δ matrix is determined, these matrices are multilinear functions of the parameters. In order to obtain a state-space model for M(s) as defined in (5), expressions for these uncertainty matrices must be determined in terms of the matrices B_{xp}, C_{qx}, D_{qp}, D_{qu}, and D_{yp} from the model. Using (4) and (5), these expressions are determined as follows:

$$\begin{aligned}
[\Delta A] &= B_{xp}[\Delta](I - D_{qp}[\Delta])^{-1} C_{qx} = B_{xp}(I - [\Delta] D_{qp})^{-1} [\Delta] C_{qx} \\
[\Delta B] &= B_{xp}[\Delta](I - D_{qp}[\Delta])^{-1} D_{qu} = B_{xp}(I - [\Delta] D_{qp})^{-1} [\Delta] D_{qu} \\
[\Delta C] &= D_{yp}[\Delta](I - D_{qp}[\Delta])^{-1} C_{qx} = D_{yp}(I - [\Delta] D_{qp})^{-1} [\Delta] C_{qx} \\
[\Delta D] &= D_{yp}[\Delta](I - D_{qp}[\Delta])^{-1} D_{qu} = D_{yp}(I - [\Delta] D_{qp})^{-1} [\Delta] D_{qu}
\end{aligned} \tag{23}$$

The inverse term makes computation of D_{qp} very difficult. Furthermore, the matrix inversion can cause [ΔA], [ΔB], [ΔC], and [ΔD] to have fractional (i.e., rational) elements with uncertain parameters in the denominator, which is not allowed in the uncertainty class being considered. Thus, it is desirable to represent this term in expanded form as follows:

$$(I - [\Delta] D_{qp})^{-1} = I + [\Delta] D_{qp} + ([\Delta] D_{qp})^2 + ([\Delta] D_{qp})^3 + \ldots \tag{24}$$

where the latter form in (23) has been assumed. Then the above equations can be rewritten as:

$$\begin{aligned}
[\Delta A] &= B_{xp}[\Delta] C_{qx} + B_{xp}\{[\Delta] D_{qp} + ([\Delta] D_{qp})^2 + ([\Delta] D_{qp})^3 + \ldots\}[\Delta] C_{qx} \\
[\Delta B] &= B_{xp}[\Delta] D_{qu} + B_{xp}\{[\Delta] D_{qp} + ([\Delta] D_{qp})^2 + ([\Delta] D_{qp})^3 + \ldots\}[\Delta] D_{qu} \\
[\Delta C] &= D_{yp}[\Delta] C_{qx} + D_{yp}\{[\Delta] D_{qp} + ([\Delta] D_{qp})^2 + ([\Delta] D_{qp})^3 + \ldots\}[\Delta] C_{qx} \\
[\Delta D] &= D_{yp}[\Delta] D_{qu} + D_{yp}\{[\Delta] D_{qp} + ([\Delta] D_{qp})^2 + ([\Delta] D_{qp})^3 + \ldots\}[\Delta] D_{qu}
\end{aligned} \tag{25}$$

The second group of terms add in the cross-terms of the multilinear uncertainty functions. Each term in the series adds a higher-order cross-product term. Since [ΔA], [ΔB], [ΔC], and [ΔD] are multilinear functions with a finite number of terms, the D_{qp} matrix can be defined to have a special nilpotent structure such that:

$$D_{qp}^{r+1} = 0 \tag{26}$$

and:

$$(I - [\Delta] D_{qp})^{-1} = I + [\Delta] D_{qp} + ([\Delta] D_{qp})^2 + \ldots ([\Delta] D_{qp})^r \tag{27}$$

where r is the order of the highest cross-term occuring in [ΔA], [ΔB], [ΔC], and [ΔD], i.e.:

$$r = \max(O_A, O_B, O_C, O_D) \tag{28}$$

where O_A, O_B, O_C, and O_D represent the order of the highest-order cross-product term in [ΔA], [ΔB], [ΔC], and [ΔD], respectively. Cross-product term order is defined as:

$$\text{order}(\delta_1 \delta_2 \delta_3 \ldots \delta_i) = i - 1 \tag{29}$$

where $i = 1, 2, \ldots, m$. Thus, the maximum value of r is $r_{max} = m-1$, where m is the dimension of the Δ matrix.

The required structure for D_{qp} to satisfy (26) and (27) is given as follows:

$$\begin{aligned}
&1.) \quad d_{ii} = 0; \qquad\qquad i = 1, 2, \ldots, m \\
&2.) \quad \text{If } d_{ij} \neq 0, \text{ then for } i = 1, 2, \ldots, m \text{ and } j = 1, 2, \ldots, m: \\
&\qquad a.) \quad d_{ji} = 0; \\
&\qquad b.) \quad d_{i\oplus 1, j\oplus 1} = 0 \text{ or } d_{i\oplus 2, j\oplus 2} = 0 \text{ or } \ldots \text{ or } d_{i\oplus(m-1), j\oplus(m-1)} = 0
\end{aligned} \tag{30}$$

where the symbol "$\oplus$" represents "modulo m" addition. The desired equations can therefore be written as:

$$
\begin{aligned}
[\Delta A] &= B_{xp}\,[\Delta]\,C_{qx} + B_{xp}\,\{\,[\Delta]\,D_{qp} + ([\Delta]\,D_{qp})^2 + \ldots\ ([\Delta]\,D_{qp})^r\,\}\,[\Delta]\,C_{qx} \\
[\Delta B] &= B_{xp}\,[\Delta]\,D_{qu} + B_{xp}\,\{\,[\Delta]\,D_{qp} + ([\Delta]\,D_{qp})^2 + \ldots\ ([\Delta]\,D_{qp})^r\,\}\,[\Delta]\,D_{qu} \\
[\Delta C] &= D_{yp}\,[\Delta]\,C_{qx} + D_{yp}\,\{\,[\Delta]\,D_{qp} + ([\Delta]\,D_{qp})^2 + \ldots\ ([\Delta]\,D_{qp})^r\,\}\,[\Delta]\,C_{qx} \\
[\Delta D] &= D_{yp}\,[\Delta]\,D_{qu} + D_{yp}\,\{\,[\Delta]\,D_{qp} + ([\Delta]\,D_{qp})^2 + \ldots\ ([\Delta]\,D_{qp})^r\,\}\,[\Delta]\,D_{qu}
\end{aligned}
\tag{31}
$$

where "r" is defined in (28). Since the [ΔA], [ΔB], [ΔC], and [ΔD] matrices are known for the given system, the equations in (31) are used to determine B_{xp}, C_{qx}, D_{qu}, D_{yp}, and D_{qp}. Once these matrices are obtained, the state-space model of M(s) is found. Hence, a minimal M-Δ model has been formed.

A procedure which summarizes the necessary steps in obtaining a minimal M-Δ model using these results is presented next.

6. Summary of Procedure

The following is a summary of the procedure implied by the above proposed approach for forming a minimal M-Δ model of a given uncertain system:

i.) Obtain the system transfer function in factored form. The coefficients of each factor should be a multilinear function of the uncertain parameters. If necessary, define new dependent parameters to represent any non-multilinear terms in a multilinear form.

ii.) Define the number of parameters in the Δ matrix, m, using (22). In so doing, determine if any new parameters are required to model inseparable uncertain real pole or zero pairs. If there are inseparable real pairs, either uncertain parameter in the pair may be repeated.

iii.) Define the minimal Δ matrix as in (3), using the independent parameters defined in the given transfer function as well as those defined in steps i.) and ii.) above.

iv.) Obtain a cascade-form realization for the system as a function of the uncertain parameters.

v.) Express the system matrices as in (2).

vi.) Determine the maximum order of cross-product terms, r, in [ΔA], [ΔB], [ΔC], and [ΔD] as defined by (28) and (29). Then [ΔA], [ΔB], [ΔC], and [ΔD] have the form represented in (31), where D_{qp} has the special (nilpotent) structure summarized by (30).

vii.) Express [ΔA], [ΔB], [ΔC], and [ΔD] as:

$$
\begin{aligned}
[\Delta A] &= [\Delta A_0] + [\Delta A_1] + [\Delta A_2] + \ldots + [\Delta A_r] \\
[\Delta B] &= [\Delta B_0] + [\Delta B_1] + [\Delta B_2] + \ldots + [\Delta B_r] \\
[\Delta C] &= [\Delta C_0] + [\Delta C_1] + [\Delta C_2] + \ldots + [\Delta C_r] \\
[\Delta D] &= [\Delta D_0] + [\Delta D_1] + [\Delta D_2] + \ldots + [\Delta D_r]
\end{aligned}
\tag{32}
$$

where the subscript i represents the cross-terms of i^{th} order in each uncertainty matrix.

viii.) The B_{xp}, C_{qx}, D_{yp}, and D_{qu} matrices are found using the expansion described in [26] for the uncertainty matrices having zero-order cross-product terms; i.e. define:

$$
M = \begin{bmatrix} \Delta A_o & \Delta B_o \\ \Delta C_o & \Delta D_o \end{bmatrix} = M_1\,\delta_1 + M_2\,\delta_2 + \cdots + M_m\,\delta_m
\tag{33}
$$

where the M_i matrices are appropriately partitioned. For the case of repeated parameters due to inseparable real poles or zeros, the M_i matrix associated with the repeated parameter must be nonzero. These matrices can be decomposed into the product of appropriately partitioned column and row matrices as follows:

$$M_i = \begin{bmatrix} M_{B_i} \\ M_{D_{1i}} \end{bmatrix} \begin{bmatrix} M_{C_i} & M_{D_{2i}} \end{bmatrix} \tag{34}$$

where M_{B_i} forms the i^{th} column of B_{xp}, $M_{D_{1i}}$ forms the i^{th} column of D_{yp}, M_{C_i} forms the i^{th} column of C_{qx}, and $M_{D_{2i}}$ forms the i^{th} column of D_{qu}. Thus:

$$\begin{aligned} B_{xp} &= \begin{bmatrix} M_{B_1} & M_{B_2} & \cdots & M_{B_m} \end{bmatrix} \\ D_{yp} &= \begin{bmatrix} M_{D_{11}} & M_{D_{12}} & \cdots & M_{D_{1m}} \end{bmatrix} \\ C_{qx}^T &= \begin{bmatrix} M_{C_1}^T & M_{C_2}^T & \cdots & M_{C_m} \end{bmatrix}^T \\ D_{qu}^T &= \begin{bmatrix} M_{D_{21}}^T & M_{D_{22}}^T & \cdots & M_{D_{2m}} \end{bmatrix}^T \end{aligned} \tag{35}$$

ix.) Use the higher-order cross-terms of [ΔA], [ΔB], [ΔC], and [ΔD], as in (32), to determine the elements of the D_{qp} matrix. Begin with the first-order terms and specify as many elements as possible. Continue with the second-order terms, and proceed until all elements of D_{qp} are specified. Check D_{qp} to ensure that the required special structure of (30) and, hence, (26) is satisfied.

x.) Form the minimal M-Δ model as given in (3), (5) and (6), and as pictured in Figure 1.

It should be noted that the matrices M_{B_i}, M_{C_i}, $M_{D_{1i}}$, and $M_{D_{2i}}$, obtained in decomposing the M_i matrices in (34), are not necessarily unique. A method of formalizing this decomposition for computer implementation will not be addressed in this paper. However, an algorithm is presented in [35] which accomplishes this decomposition for a more restrictive uncertainty class. Some examples will be given next to illustrate these results.

7. Examples

The following examples illustrate the proposed procedure presented above. Due to space limitations, details of each step will not be included. However, the results that are presented for each example should be fairly easily obtained.

Example 7.1

Consider the following uncertain system:

$$G(s,\delta) = \frac{(\beta_1 s + \beta_2)(\beta_3 s + \beta_4)(b_1 s^2 + b_2 s + b_3)}{(s + \alpha_1)(s + \alpha_2)(s^2 + a_1 s + a_2)}$$

where:
$$\alpha_1 = \alpha_{1_0} + \delta_{\alpha_1}\ ,\ \alpha_2 = \alpha_{2_0} + \delta_{\alpha_2}\ ,\ a_1 = a_{1_0} + \delta_{a_1}\ ,\ a_2 = a_{2_0} + \delta_{a_2}$$
$$\beta_1 = \beta_{1_0} + \delta_{\beta_1}\ ,\ \beta_2 = \beta_{2_0} + \delta_{\beta_2}\ ,\ \beta_3 = \beta_{3_0} + \delta_{\beta_3}\ ,\ \beta_4 = \beta_{4_0} + \delta_{\beta_4}$$
$$b_1 = b_{1_0} + \delta_{b_1}\ ,\ b_2 = b_{2_0} + \delta_{b_2}\ ,\ b_3 = b_{3_0} + \delta_{b_3}$$

The cascade-form realization of this example is found in a straight-forward manner to be:

$$A = \begin{bmatrix} -\alpha_1 & 0 & 0 & 0 \\ \beta_2 - \alpha_1\beta_1 & -\alpha_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \beta_3(\beta_2 - \alpha_1\beta_1) & \beta_4 - \alpha_2\beta_3 & -a_2 & -a_1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 \\ \beta_1 \\ 0 \\ \beta_1\beta_3 \end{bmatrix}$$

$$C = [\, b_1\beta_3(\beta_2-\alpha_1\beta_1) \quad b_1(\beta_4-\alpha_2\beta_3) \quad (b_3-a_2b_1) \quad (b_2-a_1b_1)\,] \qquad D = [\, \beta_1\beta_3b_1\,]$$

This system has eleven independent uncertain parameters. There are no non-multilinear terms in the transfer function coefficients, and there are no inseparable uncertain real pole or zero pairs. Thus, m = 11 and the minimal Δ matrix is defined as:

$$\Delta = \text{diag}\,[\,\delta_{\alpha_1}, \delta_{\alpha_2}, \delta_{a_1}, \delta_{a_2}, \delta_{\beta_1}, \delta_{\beta_2}, \delta_{\beta_3}, \delta_{\beta_4}, \delta_{b_1}, \delta_{b_2}, \delta_{b_3}\,]$$

For this example, uncertainty arises in the A, B, C, and D matrices. The ΔA, ΔB, ΔC, and ΔD matrices are fairly complicated for this example, and the order of the highest cross-product term is three, i.e. r = 3. Following the procedure outlined in Section 6, the results are determined to be:

$$B_{xp} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \beta_{1o} & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \beta_{1o}\beta_{3o} & \beta_{3o} & 1 & 1 & \beta_{3o} & \beta_{3o} & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$D_{yp} = [\, b_{1o}\beta_{1o}\beta_{3o} \;\; b_{1o}\beta_{3o} \;\; b_{1o} \;\; b_{1o} \;\; b_{1o}\beta_{3o} \;\; b_{1o}\beta_{3o} \;\; b_{1o} \;\; b_{1o} \;\; 1 \;\; 1 \;\; 1\,]$$

$$C_{qx}{}^T = \begin{bmatrix} -1 & 0 & 0 & 0 & -\alpha_{1o} & 1 & (\beta_{2o}-\alpha_{1o}\beta_{1o}) & 0 & \beta_{3o}(\beta_{2o}-\alpha_{1o}\beta_{1o}) & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & -\alpha_{2o} & 1 & (\beta_{4o}-\alpha_{2o}\beta_{3o}) & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -a_{2o} & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -a_{1o} & 1 & 0 \end{bmatrix}$$

$$D_{qu}{}^T = [\, 0\;0\;0\;0\;1\;0\;\beta_{1o}\;0\;\beta_{1o}\beta_{3o}\;0\;0\,]$$

$$D_{qp} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \beta_{1o} & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \beta_{1o}\beta_{3o} & \beta_{3o} & 1 & 1 & \beta_{3o} & \beta_{3o} & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Equations (5) and (6) can now be used to obtain the state-space model of M(s).

Example 7.2

This example illustrates the case of an inseparable uncertain real pole pair. Consider the following system:

$$G(s,\delta) = \frac{b_1 s^2 + b_2 s + b_3}{(s+\theta_1)(s+\theta_2)}$$

where: $b_1 = b_{1_o} + \delta_{b_1}$, $b_2 = b_{2_o} + \delta_{b_2}$, $b_3 = b_{3_o} + \delta_{b_3}$

$\theta_1 = \theta_{1_o} + \delta_{\theta_1}$, $\theta_2 = \theta_{2_o} + \delta_{\theta_2}$

Since the numerator is second order with uncertain coefficients, the uncertain real poles in the denominator cannot be separated into the real cascade form. The denominator must therefore be expanded, and the complex (second-order) block used in the realization, which is given as follows:

$$A = \begin{bmatrix} 0 & 1 \\ -\theta_1\theta_2 & -(\theta_1+\theta_2) \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$C = \begin{bmatrix} (b_3 - \theta_1\theta_2 b_1) & (b_2 - (\theta_1+\theta_2)b_1) \end{bmatrix}, \qquad D = [b_1]$$

Since there is one inseparable uncertain pair of poles, either δ_{θ_1} or δ_{θ_2} must be repeated in the Δ matrix. (It can be shown that if this is not done, D_{qp} will not have the required structure and hence the higher-order cross-product terms will not be modeled correctly.) Since there are no non-multilinear terms in any of the transfer function coefficients, m = 6 (i.e., five given independent uncertain parameters plus one dependent parameter for the inseparable pole pair). The resulting Δ matrix can therefore be defined as follows:

$$\Delta = \text{diag}\,[\,\delta_{\theta_1}, \delta_{\theta_1}, \delta_{\theta_2}, \delta_{b_1}, \delta_{b_2}, \delta_{b_3}\,]$$

where δ_{θ_1} was arbitrarily chosen to be repeated. Using the proposed procedure, the following results are obtained:

$$B_{xp} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 0 & 0 & 0 \end{bmatrix}, \quad D_{yp} = [-b_{1o} \;\; -b_{1o} \;\; -b_{1o} \;\; -1 \;\; 1 \;\; 1]$$

$$C_{qx} = \begin{bmatrix} \theta_{2o} & 0 \\ 0 & 1 \\ \theta_{1o} & 1 \\ \theta_{1o}\theta_{2o} & (\theta_{1o}+\theta_{2o}) \\ 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad D_{qu} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \\ 0 \\ 0 \end{bmatrix}$$

$$D_{qp} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1/\theta_{2o} & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Example 7.3

This example illustrates the case of non-multilinear terms in the transfer function. Consider the second-order uncertain system:

$$G(s,\delta) = \frac{1}{s^2 + 2\sigma s + \omega^2} \quad , \qquad \text{where:} \qquad \sigma = \sigma_o + \delta_\sigma \quad , \quad \omega = \omega_o + \delta_\omega$$

This is a second-order system with uncertain complex poles. The uncertainty appears in the real and imaginary components of the complex poles. The constant coefficient, ω^2, is not a multilinear function of the uncertain parameters. Substituting for σ and ω in the above transfer function yields the following equation:

$$s^2y = u - 2\sigma_o sy - \omega_o^2 y - 2\delta_\sigma sy - (2\omega_o\delta_\omega + \delta_\omega^2)y$$

In forming an M-Δ model, the problematic term is δ_ω^2 because it is not multilinear. In order to represent this equation in multilinear form, the following dependent variable is defined:

$$\delta_3 = \delta_\omega$$

so that:

$$s^2y = u - 2\sigma_o sy - \omega_o^2 y - 2\delta_\sigma sy - (2\omega_o\delta_\omega + \delta_\omega\delta_3)y$$

Then the following equations can be defined:

$$q_1 = -2\,sy \qquad p_1 = \delta_\sigma q_1$$
$$q_2 = -y \qquad p_2 = \delta_\omega q_2$$
$$q_3 = -2\,\omega_o y + p_2 \qquad p_3 = \delta_3 q_3$$

Thus:

$$s^2y = u - 2\sigma_o sy - \omega_o^2 y + p_1 + p_3$$

The realization of M(s) for the resulting M-Δ model can be depicted as follows:

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_o^2 & -2\sigma_o \end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0 & 0 & 0\\ 1 & 0 & 1\end{bmatrix}\begin{bmatrix}p_1\\ p_2\\ p_3\end{bmatrix} + \begin{bmatrix}0\\ 1\end{bmatrix} u$$

$$\begin{bmatrix}q_1\\ q_2\\ q_3\end{bmatrix} = \begin{bmatrix} 0 & -2 \\ -1 & 0 \\ -2\omega_o & 0 \end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}\begin{bmatrix}p_1\\ p_2\\ p_3\end{bmatrix}$$

$$y = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}$$

The Δ matrix is given by $\Delta = \text{diag}\,[\,\delta_\sigma, \delta_\omega, \delta_3\,]$, where $\delta_3 = \delta_\omega$.

These examples illustrate the proposed procedure for forming a minimal M-Δ model of an uncertain system. Although all the steps involved in obtaining these results have not been included, the stated results should provide a guide in performing the steps of the proposed procedure. It should be noted that, for ease of hand computation, the examples included only the simplistic (and less realistic) case in which the coefficients themselves are the uncertain parameters. However, it is emphasized that the proposed procedure does handle the more realistic case in which the uncertain transfer function coefficients are multilinear functions of the uncertain parameters.

8. Concluding Remarks

This paper has presented a proposed procedure for forming a minimal M-Δ model of an uncertain system given its transfer function in terms of the uncertain parameters. The uncertainty class considered in this paper allows the transfer function coefficients to be multilinear functions of the uncertain parameters, and the uncertainties may arise in the A, B, C, and D matrices of the system model. The proposed procedure involves realizing the system in a cascade form, determining the minimal Δ matrix of uncertain parameters, and obtaining a state-space model for the nominal system, M(s). Three examples were given to illustrate the proposed procedure. The first example had eleven independent uncertain parameters, which arose in the A, B, C, and D matrices of the system realization. The second example had uncertain parameters arising in the A, C, and D matrices only. This example illustrated the formulation of a minimal M-Δ model for a system with inseparable real uncertain poles, and involved repeating an uncertain parameter in the Δ matrix. The last example had uncertainty in the A matrix only, and illustrated a method for handling non-multilinear terms.

Further work on the proposed procedure will be to include systems having a nonmonic characteristic polynomial with an uncertain leading coefficient, as well as systems having an inner feedback loop which may or may not have uncertainties. The latter case may require a modification in the formulation of the cascade realization. Although the procedure presented in this paper is for SISO systems, an extension to MIMO systems will be forthcoming, and should primarily involve modifying the cascade-form realization. Other areas of future work include development of a simple means of verifying the minimality of a given M-Δ model, and development of a method of reducing a nonminimal M-Δ model to a minimal form.

REFERENCE LIST

[1.] J. C. Doyle and G. Stein, "Multivariable Feedback Design: Concepts for a Classical/Modern Synthesis", IEEE Trans. on Aut. Contr., Vol. AC-26, pp.4-16, Feb. 1981.

[2.] M. J. Chen and C. A. Desoer, "Necessary and Sufficient Condition for Robust Stability of Linear Distributed Feedback Systems", Int. J. Control,Vol. 35, No. 2, pp. 255-267, 1982.

[3.] N. A. Lehtomaki, N. R. Sandell, and M. Athans, "Robustness Results in Linear-Quadratic-Gaussian Based Multivariable Control Design", IEEE Trans. on Aut. Contr., Vol. AC-26, pp. 75-92, Feb. 1981.

[4.] N. A. Lehtomaki, D. A. Castanon, B.C. Levy, G. Stein, N. R. Sandell Jr., and M. Athans, "Robustness and Modeling Error Characterization", IEEE Trans. on Aut. Contr., Vol. AC-29, No. 3, March 1984.

[5.] I. Postelthwaite, J. M. Edmunds, and A. G. J. MacFarlane, "Principal Gains and Principal Phases In The Analysis of Linear Multivariable Feedback Systems", IEEE Trans. on Aut. Contr., Vol. AC-26, pp. 32-46, 1981.

[6.] M. G. Safonov, A. J. Laub, and G. L. Hartmann, "Feedback Properties of Multivariable Systems: The Role and Use of the Return Difference Matrix", IEEE Trans. on Aut. Contr., Vol. AC-26, pp. 47-65, 1981.

[7.] J. B. Cruz, J. S. Freudenberg, and D. P. Looze, "A Relationship Between Sensitivity and Stability of Multivariable Feedback Systems", IEEE Trans. on Aut. Contr., Vol. AC-26, pp. 66-74, 1981.

[8.] J. C. Doyle, "Robustness of Multiloop Linear Feedback Systems", 17th IEEE Conf. on Decision and Control, San Diego, USA, Jan. 1979.

[9.] J. S. Freudenberg, D. P. Looze, and J. B. Cruz, "Robustness Analysis Using Singular Value Sensitivities", Int. J. Control, Vol. 35, pp. 95-116, 1982.

[10.] B. A. Francis, J. W. Helton, and G. Zames, "H^∞- Optimal Feedback Controllers for Linear Multivariable Systems", IEEE Trans. on Aut. Contr., Vol. AC-29, pp. 888-900, Oct. 1984.

[11.] B.-C. Chang and J. B. Pearson, "Optimal Disturbance Reduction in Linear Multivariable Systems", IEEE Trans. on Aut. Contr., Vol. AC-29, pp. 880-887, Oct. 1984.

[12.] B.-C. Chang, "A Stable State-Space Realization in the Formulation of H^∞ Norm Computation", IEEE Trans. Aut. Contr., Vol. AC-32, pp. 811-815, Sept. 1987.

[13.] B.-C. Chang, S. S. Banda, and T. E. McQuade, "Fast Iterative Computation of Optimal Two-Block H^∞-Norm", IEEE Trans. on Aut. Contr., Vol. AC-34, pp. 738-743, July 1989.

[14.] B.-C. Chang, "Size Reduction in Four-Block H^∞ Formulation", Proceedings of 26th IEEE Conf. on Decision and Control, Dec. 1987.

[15.] J. C. Doyle, K. Glover, P. Khargonekar, and B. A. Francis, "State-Space Solutions to Standard H_2 and H_∞ Control Problems", 1988 American Control Conference, Atlanta, GA, June 1988.

[16.] M. G. Safonov and J. Doyle, "Minimizing Conservativeness of Robustness Singular Values", in S. G. Tzefestas (ed.), Multivariable Control, D. Reidel Publishing Company, 1984.

[17.] J. Doyle, "Analysis of Feedback Systems with Structured Uncertainties", IEE Proc. Part D, Vol. 129, No. 6, pp. 242-250, 1982.
[18.] J. C. Doyle, J. E. Wall, and G. Stein, "Performance and Robustness Analysis with Structured Uncertainty", Proceedings of the 21st IEEE Conference on Decision and Control, 1982.
[19.] J. Doyle, "Synthesis of Robust Controllers and Filters", Proc. of 22nd IEEE Conf. on Decision and Control, Dec. 1983.
[20.] J. Doyle, "Structured Uncertainty in Control System Design", Proc. of the 24th IEEE Conf. on Decision and Control, Dec. 1985.
[21.] J. Doyle, Lecture Notes, ONR/Honeywell Workshop on Advances in Multivariable Control, Minneapolis, MN, Oct. 1984.
[22.] M. G. Safonov, "Optimal H^{∞} Synthesis of Robust Controllers For Systems with Structured Uncertainty", Proc. of 25th IEEE Conf. on Decision and Control, pp. 1822-1825, Dec. 1986.
[23.] J. C. Doyle, "A Review of μ For Case Studies in Robust Control", IFAC 10th Triennial World Congress, Munich, FRG, 1987.
[24.] M. K. H. Fan and A. L. Tits, "Characterization and Efficient Computation of the Structured Singular Value", IEEE Trans. Aut. Contr., Vol. AC-31, pp. 734-743, Aug. 1986.
[25.] M. K. H. Fan and A. L. Tits, "m-Form Numerical Range and the Computation of the Structured Singular Value", IEEE Trans. Aut. Contr., Vol. AC-33, pp. 284-289, 1988.
[26.] B. G. Morton and R. M. McAfoos, "A Mu-Test for Real Parameter Variations", Proc. of the1985 American Control Conference.
[27.] R. D. Jones, "Structured Singular Value Analysis for Real Parameter Variations", 1987 AIAA Conference on Guidance and Control.
[28.] M. G. Safonov, "Stability Margins of Diagonally Perturbed Multivariable Feedback Systems", Proc. of the 20th IEEE Conf. on Decision and Control, Dec. 1981.
[29.] R. E. De Gaston and M. G. Safonov, "Exact Calculation of the Multiloop Stability Margin", IEEE Trans. on Aut. Contr., Vol. AC-33, pp. 156-171, 1988.
[30.] R. S. S. Pena and A. Sideris, "A General Program to Compute the Multivariable Stability Margin for Systems with Parametric Uncertainty", Proc. of the 1988 American Control Conference.
[31.] A. Sideris and R. S. S. Pena, "Fast Computation of the Multivariable Stability Margin for Real Interrelated Uncertain Parameters", Proc. of the 1988 American Control Conference.
[32.] B.-C. Chang, O. Ekdal, H. H. Yeh, and S. S. Banda, "Computation of the Real Structured Singular Value via Polytopic Polynomials", Proc. of the 1989 AIAA Guidance, Navigation and Control Conference.
[33.] H. Bouguerra, B.-C. Chang, H. H. Yeh, and S. Banda, "A Fast Algorithm for Checking the Stability of Convex Combinations of Stable Polynomials", Technical Report, Dept. of Mechanical Engineering, Drexel University, Jan. 1989.
[34.] G. A. Hewer, C. Kenney, and R. Klabunde, "A State Space Model of Structured Singular Values", Proc. of the 27th IEEE Conf. on Decision and Control, Dec. 1988.
[35.] M. K. Manning and S. S. Banda, "Algorithm to Obtain M-Δ Form for Robust Control", Proc. of IEEE Conf. on Systems Engineering, Aug. 24-26, 1989, Dayton, Ohio.

Space Station Dynamics, Attitude Control and Momentum Management

John W. Sunkel
NASA-Johnson Space Center
Ramen P. Singh and Ravi Vengopal
Dynacs Engineering Co.

Abstract

The Space Station Attitude Control System software test-bed provides a rigorous environment for the design, development and functional verification of GN & C algorithms and software. All Space Station systems and sub-systems that are controlled or monitored by the GN & C software are simulated. The simulation presents a major computational challenge, starting from the simulation of full nonlinear flexible body dynamics including the orbital environment and Mobile Servicing System (MSS) operations, to task scheduling, sensor dynamics and inter-module communication. In addition, the complex tasks of providing flight algorithm sequencing and control and input command validation needs to be addressed.

This paper describes the approach taken for the simulation of the vehicle dynamics and environmental models using a computationally efficient algorithm. The simulation includes capabilities for docking/berthing dynamics, prescribed motion dynamics associated with the Mobile Remote Manipulator System (MRMS) and microgravity disturbances. The vehicle dynamics module interfaces with the test-bed through the centroal Communicator facility which is in turn driven by the Station Control Simulator (SCS) Executive. The Communicator addresses issues such as the interface between the discrete flight software and the continuous vehicle dynamics, and multi-programming aspects such as the complex flow of control in real-time programs. Combined with the flight software and redundancy management modules, the facility provides a flexible, user-oriented simulation platform.

Approximate Minimum-Time Trajectories for 2-link Flexible Manipulators

G.R. Eisler, D.J. Segalman, R.D. Robinett†
Sandia National Laboratories, Albuquerque, New Mexico

Abstract

Powell's nonlinear programming code, VF02AD, has been used to generate approximate minimum-time tip trajectories for 2-link semi-rigid and flexible manipulator movements in the horizontal plane. The manipulator is modeled with an efficient finite-element scheme for an n-link, m-joint system with horizontal-plane bending only. Constraints on the trajectory include boundary conditions on position and energy for a rest-to-rest maneuver, straight-line tracking between boundary positions, and motor torque limits. Trajectory comparisons utilize a change in the link stiffness, EI, to transition from the semi-rigid to flexible case. Results show the level of compliance necessary to excite significant modal behavior. Quiescence of the final configuration is examined with the finite-element model.

Introduction

Trajectory planning is essential in budgeting a manipulator's actuator efforts to maximize productivity. For repetitive tasks, the minimum-time maneuver goes hand-in-hand with this goal. A variety of approaches have been advanced for rigid manipulator control, taking advantage of the fact that all or some of the controls take the form of switching functions between actuator bounds. Bobrow [1] used an intuitive approach to generate optimal switching controls, as well as proving the boundedness of the controls. Weinreb and Meier [2][3] used calculus of variations approaches to incorporate control bounds in the problem formulation. In a second study, Bobrow [4] used numerical optimization to generate spline fits to the switching controls.

Switching functions do not lend themselves to maintaining tip accuracy for non-rigid structures. One would hope that the applied controls do take advantage of the bounds to maximize performance, but a clear analytical directive for this does not exist at the present.

In filling this void, parameter optimization techniques can provide approximate optimal performance solutions for systems driven by complex, highly nonlinear dynamic models with arbitrary equality or inequality constraints. Of these solution techniques, Powell's Recursive Quadratic Programming algorithm [5], embodied in the code VF02AD, has proven to be a robust tool for a variety of aerospace applications [6][7][8], and will be used in this study. The primary drawback to this or other numerical optimization methods is the dependancy on accurate gradient approximations of the performance index and constraints with respect to the parameters.

The ensuing discussion initially describes the structural dynamics model of the manipulator, followed by the optimal control problem and parameterization of the controls, and ends with the results of a computational experiment.

The manipulator structure modeled in this study, and presently under fabrication, is a 2-link cantilever arrangement constrained to slew in the horizontal plane. Tall, thin links are used to

†Members of the Technical Staff, Engineering Analysis Department

minimize vertical plane droop. The hub or joint-1 actuator slews both links, an interlink motor, and tip payload. The interlink or joint-2 actuator located at the end of link-1 slews the second link and the tip payload. The joint-1/joint-2 actuator torque ratio is about 4/1. The complete manipulator is about 0.5 meters (m) tall and 1.2m long.

The Structural Model

There is a long literature discussing the difficulties of simulating the vibrations of rotating structures[9] [10] [11]. The problem seems to arise from kinematics that are of second order importance in nonrotating problems, but become of first order importance in the presence of rotational accelerations. Additonally, there are constraints inherent to the flexible link problem which must be satisfied: motions occur entirely in a horizontal plane; one end of the chain of links is attached to a stationary hub; and each flexible link is inextensible. These considerations motivate the development of a mathematical model that faithfully carries the full kinematics of the problem. It is also necessary to devise such a model in a form that will lend itself to real-time calculations.

The need to meet these apparently conflicting demands motivated the development of a model specialized to flexible, multilink structures. That apparently successful strategy is outlined below. The full kinematics are retained by expressing the configuration as functions of convected coordinates. This is a traditional approach in nonlinear elasticitity [12]. Further, the kinematic variables are selected so that all geometric constraints (fixed hub, planar motion, and non-extension) are automatically satisfied.

Since motions are assumed to occur entirely in a plane, it is also assumed that the elastic lines of the links as well as the mass centers of the cross sections all lie in the same plane. Each cross section is identified by its arc-length distance from the hub, so that the orientation of the center of the cross section s at time t is

$$\vec{\beta}(s,t) = cos(\theta(s,t))\vec{i} + sin(\theta(s,t))\vec{j}$$

The location of the center of cross section s at time t is obtained by integration of the above unit tangent vector:

$$\vec{x}(s,t) = \int_0^s \vec{\beta}(s',t)ds'$$

Similarly, the velocity at the cross section s at time t is obtained by integration of the time derivative of $\vec{\beta}(s,t)$:

$$\dot{\vec{x}}(s,t) = \int_0^s \dot{\theta}(s',t)\vec{\gamma}(s',t)ds'$$

where

$$\vec{\gamma}(s,t) = -sin(\theta(s,t))\vec{i} + cos(\theta(s,t))\vec{j}$$

The above description of configuration - entirely in terms of $\theta(s,t)$ - causes all of the geometrical constraints to be satisfied automatically. Additionally, the above description expresses the configuration in terms of one unknown field (θ), instead of the more conventional two or three fields (x, y, & θ).

The governing equations of the dynamics are derived using those kinematics and a frame-invariant variational method - Hamilton's principle. A finite element discretization is used to cast the resulting integro-differential equations for $\theta(s,t)$ and its first and second derivatives into a system of fully-coupled, nonlinear algebraic equations. Particularly important for the application at hand is the observation that since all spatial integrals are with respect to the convected coordinate, s, those integrals are configuration-independent and need be done only once. The nonlinearities

remain, and a new nonlinear system must be solved at each time step, but the time consuming quadrature process can be done in advance of the dynamics simulation. These features are illuminated below through derivation.

Hamilton's principle is that

$$\delta \int_{t_1}^{t_2} \left[KE(t') - SE(t') + WE(t') \right] dt' = 0 \tag{1}$$

where KE is kinetic energy, SE is strain energy, and WE is external work. Quantities associated with the end times, t_1 and t_2 have to do with initial and final conditions and the conservation of momenta, but the governing equations necessary to model motion of the flexible structure are obtained by consideration of the integrand alone:

$$\delta KE(t) - \delta SE(t) + \delta WE(t) = 0 \tag{2}$$

for all $t_1 \leq t \leq t_2$.

The kinetic energy is that of the flexible links plus that of all concentrated masses and concentrated moments of inertia:

$$KE(t) = \frac{1}{2} \int_0^L \rho(s)\, \dot{\vec{x}}(s,t) \cdot \dot{\vec{x}}(s,t) ds + \frac{1}{2} \sum_{k=1}^{masses} M_k\, \dot{\vec{x}}(s_k,t) \cdot \dot{\vec{x}}(s_k,t) + \frac{1}{2} \sum_{l=1}^{inertias} I_l\, \dot{\theta}(s_l,t) \cdot \dot{\theta}(s_l,t)$$

The strain energy is that of the flexible links:

$$SE(t) = \frac{1}{2} \int_0^L \kappa(s) \frac{\partial \vec{\beta}(s,t)}{\partial s} \cdot \frac{\partial \vec{\beta}(s,t)}{\partial s} ds$$

Discretization of the above energy terms is obtained by discretizing the tangent vector $\vec{\beta}$ as:

$$\vec{\beta}(s,t) = \sum_{n=1}^{nodes} \vec{\beta}_n(t) p_n(s)$$

where the shape functions, p_n, have support over intervals that are small relative to the anticipated radii of curvature. The above condition on the support of the basis functions is necessary to assure compliance with the condition of nonextension. The resulting energies are:

$$KE(t) = \frac{1}{2} \sum_{m=1}^{nodes} \sum_{n=1}^{nodes} \dot{\theta}_m(t) \dot{\theta}_n(t) \vec{\gamma}_m(t) \cdot \vec{\gamma}_n(t) M_{m,n} \tag{3}$$

and

$$SE(t) = \frac{1}{2} \sum_{m=1}^{nodes} \sum_{n=1}^{nodes} \vec{\beta}_m(t) \cdot \vec{\beta}_n(t)\, K_{m,n} \tag{4}$$

where

$$M_{m,n} = \int_0^L \rho(s)\, q_m(s)\, q_n(s) ds + \sum_{k=1}^{masses} M_k\, q_m(s_k)\, q_n(s_k) + \sum_{l=1}^{inertias} I_l\, \delta_K(l,m)\, \delta_K(l,n) \ ,$$

$$q_m(s) = \int_0^s p_m(\hat{s}) d\hat{s} \ , \qquad K_{m,n} = \int_0^L \kappa(s)\, p'_m(s)\, p'_n(s) ds \ ,$$

δ_K is the Kronecker delta function, $\hat{s}$ is a dummy variable, and $p'_{m,n}$ are spatial derivatives.

After appropriate integration by parts, the integrand of equation 2 becomes:

$$\sum_{n=1}^{nodes} \delta\theta_m(-\vec{\gamma}_m(t)\cdot\ddot{\vec{\beta}}_n(t)M_{m,n} - \vec{\gamma}_m(t)\cdot\vec{\beta}_n(t)K_{m,n} + \vec{\gamma}_m(t)\cdot\vec{\tau}_n(t)\delta_K(m,n)) = 0 \qquad (5)$$

for all nodes m. In the above equation, τ_n is the external torque applied at node n. After $\ddot{\vec{\beta}}_n(t)$ is expanded:

$$\ddot{\vec{\beta}}_n(t) = \ddot{\theta}_n(t)\vec{\gamma}_n(t) + (\dot{\theta}_n(t))^2\vec{\beta}_n(t)$$

and Equation 5 is invoked for all $\delta\theta_m$, a complete set of *nodes* second order equations in the *nodes* unknowns, $\ddot{\theta}_n$, results as follows:

$$\sum_{n=1}^{nodes} \vec{\gamma}_m(t)\cdot\vec{\gamma}_n(t)\ddot{\theta}_n(t)M_{m,n} - \vec{\gamma}_m(t)\cdot\vec{\beta}_n(t)(\dot{\theta}_n(t))^2M_{m,n} + \vec{\gamma}_m(t)\cdot\vec{\beta}_n(t)K_{m,n} = \vec{\gamma}_m(t)\cdot\vec{\tau}_m(t) \quad (6)$$

The above problem formulation, involving only one unknown field, automatically satisfying all constraints, and requiring only one evaluation of element mass and stiffness matrices, lends itself to rapid numerical calculation. A computer code to generate and solve the above described system of equations for each time step has been written, tested and used by VF02AD. Both the derivation and code mentioned above are described more fully in Ref. [13].

One particularly interesting calculation, discussed in the literature as being difficult to solve, is that of a beam accelerated around a hub to an angular frequency above the first bending frequency of the beam. This is a particularly stringent test of flexible-dynamics codes, testing numerical robustness, as well as the correctness of the physics. Figure 1 shows the motion of the tip of such a flexible beam relative to the tip of a rigid beam rotating at the hub velocity. Initially, the flexible beam lags its rigid counterpart; it then overtakes and oscillates about the rigid beam. These results are in near exact agreement with the results presented in [9], obtained from a much more complex model.

Optimal Trajectory Shaping

The principle goal in this study is to combine the physics of the structure with optimization techniques to generate actuator torque histories for accomplishing a useful task with minimal degradation in performance. A secondary objective is to shortcut the work of an eventual feedback controller, which will be needed to compensate for modeling errors.

Looking towards maximizing productivity in some repetitive task, a minimum-time tip trajectory was chosen for investigation. Constraints on such a trajectory include: completing a rest-to-rest maneuver, tracking a specified path $(x(t), y(t))_{tip}$, slewing between specified endpoints $[(x(t_o), y(t_o)),\ (x(t_f), y(t_f))]_{tip}$, and not exceeding actuator torque limits $\tau_{1,2_{max}}$.

The configuration initially starts at rest. Driving a flexible structure to rest at the final time, t_f, necessitates end constraints on both kinetic and potential or strain energies $(KE(t_f), SE(t_f))$. The chosen path is a straight line and actuator torques limits are constants. The problem can be

restated as

minimize: $J = t_f$

subject to: - finite element model
- input actuator torques, $\tau_{1,2}(t)$
- known initial conditions

constrained by:

$$\begin{bmatrix} \\ C_j(t_f) \\ \\ \\ \\ \\ \end{bmatrix}_{j=1,7} = \begin{bmatrix} x_{tip}(t_f) - x_{specified}(t_f) \\ y_{tip}(t_f) - y_{specified}(t_f) \\ \int_0^{t_f} [y_{tip}(x_{tip}(t)) - y_{line}(x_{tip}(t))]\,dt \\ KE(t_f) \\ SE(t_f) \\ \int_0^{t_f} (|\tau_1(t)| - |\tau_{1_{max}}|)dt \\ \int_0^{t_f} (|\tau_2(t)| - |\tau_{2_{max}}|)dt \end{bmatrix} \begin{matrix} = \\ = \\ = \\ = \\ = \\ \leq \\ \leq \end{matrix} \begin{bmatrix} \\ 0 \\ \\ \\ \\ \\ \end{bmatrix}$$

Note that the equality tracking constraint, C_3, and inequality torque constraints, C_6, C_7, are formulated as integrals. In addition, equality constraints on energy are ***point*** constraints. Both of these items will have profound effects on the example trajectories to be generated.

Parameterization of the Controls

To approximate optimum system performance from the aforementioned structural model, a suitable parameterization of the controls, $\tau_{1,2}(t)$, is necessary. The choice of "parameterizable " torque functions is essentially limitless. For this study, the simplest case of using tabular values of torque, τ_i, as parameters at equal-spaced fixed times, t_i, for both joints was chosen, or

$$\tau_1(t_i), \tau_2(t_i), \quad i = 1, n \qquad 0 \leq t_i \leq t_f,$$

which results in $2n$ control parameters.

However, since the final time is changing due to minimization, the loss of control history definition would result if the times at which the control parameters are defined remain fixed in an absolute sense. To correct this, $\tau_{1,2}$ were specified at equally-spaced, nondimensional ***node*** points, $\zeta_i = t_i/t_f$, where

$$\tau_1(\zeta_i), \tau_2(\zeta_i), \quad i = 1, n \qquad 0 \leq \zeta_i \leq 1,$$

This allows the torque histories to "stretch" naturally over the trajectory length. Using this modification, it is necessary to add t_f as a parameter also, resulting in $2n + 1$ control parameters to be found.

Numerical derivatives of the performance index, t_f, and the constraints, $C_j(t_f)$, provided to VF02AD are central finite-difference approximations. To generate derivatives with respect to a torque parameter, $\tau_{1,2_i} = \tau_{1,2}(t_i)$, the parameter is perturbed equidistant about its nominal value, and complete trajectories (or integrations of the structural equations) are computed to the current nominal t_f to produce perturbed $C_j(t_f)$ values. Since derivatives are computed over the current ***fixed*** t_f, the derivatives, $\partial t_f/\partial \tau_{1,2_i} = 0$, and only the derivatives, $\partial C_j(t_f)/\partial \tau_{1,2_i} \neq 0$. Naturally both $t_f, C_j(t_f)$ gradients with respect to t_f, evaluated over the current nominal torque histories, are nonzero, (where $\partial t_f/\partial t_f = 1.$).

Results

The following finite-element structural model was used for the manipulator to produce the

sample trajectories.

ITEM	COMPOSITION	LENGTH (m)	MASS (kg)	STIFFNESS, EI (newton-m^2)
joint-1 bracket	1 element	.0635	.545	10^5
link-1	3 elements	.5040	.640	10^2,10^3
1st joint-2 bracket + joint-2	1 element + 1 point mass	.1070	5.415	10^5
2nd joint-2 bracket	1 element	.1040	.830	10^5
link 2	3 elements	.4890	.313	10^2,10^3
Totals:	9 elements	1.2675	7.743	

The two values of stiffness, EI, for links 1,2 represent the trajectory comparison for this study. The brackets, modeled with a stiffness of 10^5, are considered rigid. Point moments of inertia were used to define mass distribution for the brackets. No payload was used in this comparison. The joints were assumed to have no compliance or damping.

The two trajectories, computed on a CRAY-XMP, were integrated for 100 time steps, where $\Delta t = .01 t_f$. Trajectory evaluations for gradient computations executed in 0.75 secs. The torque histories for each joint were composed of 21 tabular values, where $\Delta\zeta = .05$. Torque bounds were chosen as $\pm 16, \pm 4$ newton-m (n-m) for joints 1 and 2. The path to be tracked for this study was the line connecting (x, y) pairs, (0.0,1.13) and (1.13,0.0). A composite of the slew motion for the "flexible" case ($EI_{links} = 100$ newtons-m^2) is given in Fig.2. The parameterized torque histories that created this slew represent 100 iterations of VF02AD after initialization with the parameter solution values from the "semi-rigid" case ($EI_{links} = 1000$ n-m^2). The tip path traced is reasonably straight, but does contain some small ripples - a result of the integral statement of the tracking constraint, $C_3(t_f)$.

Figure 3 graphically depicts the difference between the semi-rigid and flexible links. Shown is the angular velocity of the finite-element node adjacent to joint 1. The frequency of vibration for $EI_{links} = 100$ is about 19 hz. From examination of Fast Fourier transforms (FFTs) of the finite-element output of the system disturbed about the initial, midtime, and final positions, this appears to be one of the lower modes. Note the low angular velocity of the semi-rigid system after $t/t_f = 0.9$, implying that a significant amount of time is being expended in order to bring the system to "rest". This phenomena is definitely at odds with purely rigid system behavior. The $KE(t_f) = 0$ constraint imposes a zero final angular velocity in both cases. Also, note that t_f for the flexible case is slightly greater than the semi-rigid one. This demonstrates the approximate nature of the solutions, insofar as the semi-rigid solution not converging as well.

Fig.4 shows the τ_1 profiles. These still retain some of the boundedness qualities of purely rigid configurations. However, they begin and end near zero instead of the bounds (± 16 n-m), and the transition between bounds is comparitively gradual. The oscillatory behavior in the $EI_{links} = 100$ torque is probably counteracting the excitement of the lowest structural modes, which would have the greatest impact on the position constraints. Note the abruptness of the controls near the end in an attempt to quiet the structure. The τ_2 torques in Fig.5 show minimal activity for most of the trajectory, except close to the end in order to accomplish the rest state. Note that this closing maneuver starts sooner with the more flexible link structure.

The straight-line tracking error in millimeters (mm) is shown in Fig.6. Both torque histories appear to limit the error to ± 5 mm except near the end of the $EI_{links} = 1000$ trajectory, where the error momentarily "escapes" before returning to zero to satisfy the end position constraints, $(C_1(t_f), C_2(t_f))$. One drawback to the integral formulation is that it can relax tracking performance in isolated parts of the trajectory, yet yield a reasonably low residual (≈ 0) for $C_3(t_f)$. It may be

necessary to add interior point constraints to significantly decrease this error. A less complex alternative is to reduce the EI_{links} more gradually between solutions, in the fashion of a homotopy scheme, until the desired value is reached.

The kinetic energy of the flexible structure at peak rates ($t/t_f \approx 0.45$) is, not surprisingly, higher than the semi-rigid one as shown in Fig.7. It is interesting that KE appears to be devoid of oscillatory behavior in both cases. Note again, the rest phase of the trajectories above $t/t_f = 0.9$. Strain energy is shown in Fig.8. This again displays the contrast first seen in Fig.3. The semi-rigid structure produces relatively little strain, while the flexible-link configuration again contains the 19 hz mode with a sizable increase in energy magnitude. Note the peaks in both cases, mirroring the sharp τ_1 changes in Fig.4. Also, note the enforcement of the $SE(t_f) = 0$ point constraint at the end. The SE "floor" in both cases corresponds to the KE peaks in Fig.7.

The finite-element model was also able to examine the quiescence of the final state. At t_f, joint node accelerations can be zeroed and joint node positions fixed at their nominal final values. The corresponding torques applied to maintain these values can then be computed. Fig.9 shows the residual τ_1 being applied after the approximate optimal $t_f = 1.505$ secs for $EI_{links} = 100$ has been reached. The oscillations in this residual appear to contain frequencies in the vicinity of 2.5 and 35 hz. The lower appears to correspond to a fixed-free mode of the first link with the remainder of the system mass concentrated at the end. This mode was not seen during the optimized part of this trajectory. The second frequency may be a higher harmonic of the mode seen earlier, or possibly a numerical artifact. The small magnitude of this residual coupled with the spacing of the modes may be ideal targets for attack by a linear feedback control scheme. As an open-loop alternative, augmenting the tabular torque controls with frequency-based, parameterized functionals may be sufficient to suppress these oscillations.

Conclusions

A robust, parameter optimization tool has been able to generate actuator torque histories for approximate, minimum-time slewing maneuvers containing a variety of continuous and point constraints for a 2-link flexible manipulator. The parameters, or actuator torques, for each link were tabular values at fixed node points during the maneuver. Perturbations were made to each parameter to approximate final time and constraint gradients. The efficient formulation of the finite-element model made the numerical optimization procedure a realistic endeavor.

The accuracy of the straight-line tip tracking was good. Additional interior constraints and/or a more gradual change in the stiffness should yield further improvements. For the trajectory used in this study, joint-1 applied most of the input, which included cancelling lower-mode vibrations for the structural as a whole. Energy constraints were effective in bringing the structure to rest at t_f. It was also demonstrated that final energy constraints do not preclude vibrations during the slew. The intended production use of the manipulator will dictate whether this is a hindrance or not. Finally, the secondary goal of providing enough vibration control to maximize the success of a linear feedback controller in treating residual oscillations appears feasible.

References

[1] J.E. Bobrow, S. Dubowksy and J.S. Gibson, "Time-Optimal Control of Robotic Manipulators along Specified Paths", *Int. J. Robotics Res.*, Vol 4.,no.3, Fall 1985.

[2] Weinreb, A., Bryson, A.E., "Optimal Control of Systems with Hard Control Bounds", *IEEE Transactions on Automatic Control*, Vol.AC-30, No.11, Nov. 1985, pp.1135-1138.

[3] E.B. Meier and A.E. Bryson, "An efficient algorithm for time-optimal control of a two-link manipulator,", in AIAA conference on Guidance, Navigation, and Control, Monterey, CA, Aug 1987, pp 204-212.

[4] Bobrow, J.E., "Optimal Robot Path Planning Using the Minimum-Time Criterion", *IEEE J. Robotics and Automation*, Vol.4, No.4, August 1988.

[5] Powell, M.J.D., "A Fast Algorithm for Nonlinearly Constrained Optimization Calculations", *Proceedings of the Biennnial Conference on Numerical Analysis*, Springer-Verlag, Berlin, 1978, pp.144-157.

[6] Eisler, G.R., Hull, D.G., "Maximum Terminal Velocity Turns at Nearly Constant Altitude", *Journal of Guidance, Control, and Dynamics*, Vol.11, No.2, March-April 1988, pp.131-136.

[7] Outka, D.E., "Parameter Optimization Capability in the Trajectory Code PMAST", SAND86-2917, Sandia National Laboratories, Albuquerque, 1987.

[8] Robinett, R.D., "A Unified Approach to Vehicle Design, Control, and Flight Path Optimization", The Center for Strategic Technology, Texas A&M University, SS87-1, 1987.

[9] J. C. Simo and L. Vu-Quoc 'The role of non-linear theories in transient dynamic analysis of flexible structures', *Journal of Sound and Vibration, Vol. 119*, 1987, pp 487-508.

[10] T. R. Kane, R. R. Ryan, and A. K. Banerjee, 'Dynamics of a cantilever beam attached to a moving base', *Journal of Guidance, Control and Dynamics, Vol. 10*, 1987, pp 139-151.

[11] S. Hanagud and S. Sarkar, 'Problem of the dynamics of a cantilever beam attached to a moving base', *Journal of Guidance, Control and Dynamics*, Vol. 12, 1989, pp 438-441.

[12] A. E. Green and W. Zerna, *Theoretical Elasticity*, Clarendon Press, Oxford, 1954.

[13] D. J. Segalman, 'A Mathematical Formulation for the Rapid Simulation of a Flexible Multilink Manipulator' SAND89-2308, Sandia National Laboratories, Albuquerque, New Mexico.

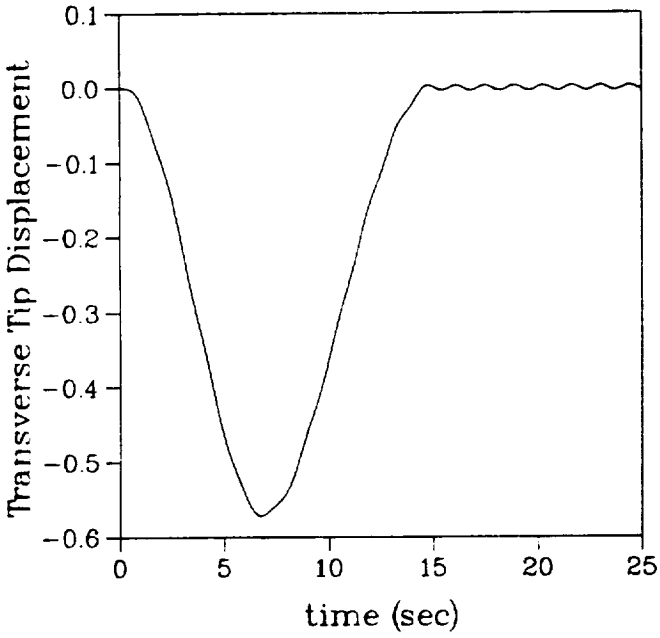


Figure 1: Single beam relative tip motion

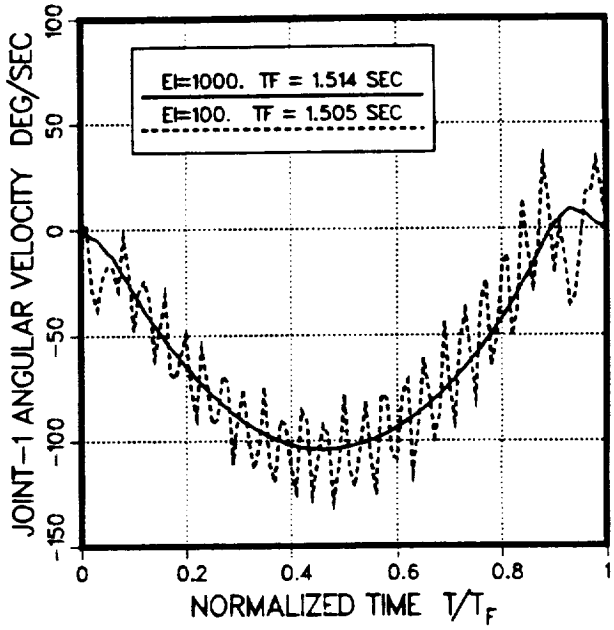


Figure 3: Joint 1 node angular velocity

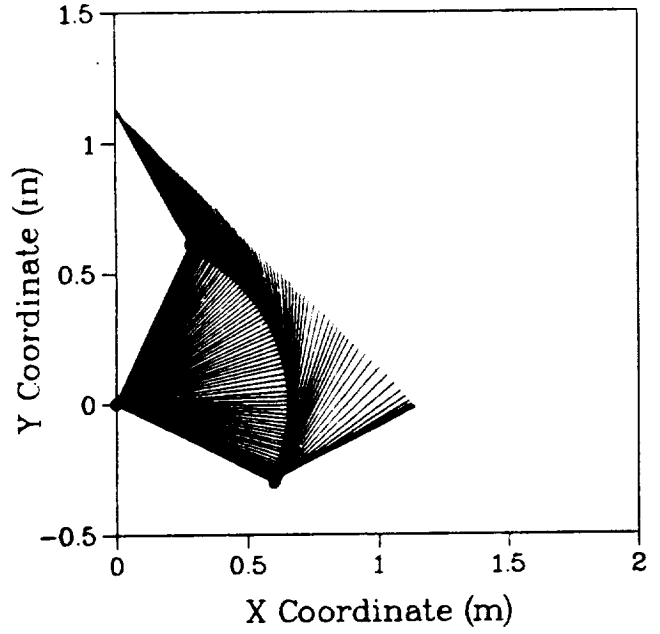


Figure 2: Composite motion for $EI_{link} = 100$

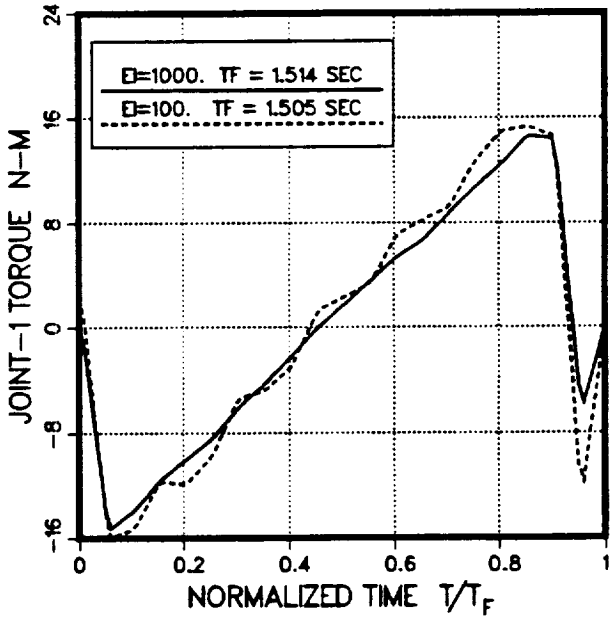


Figure 4: τ_1 joint torque histories

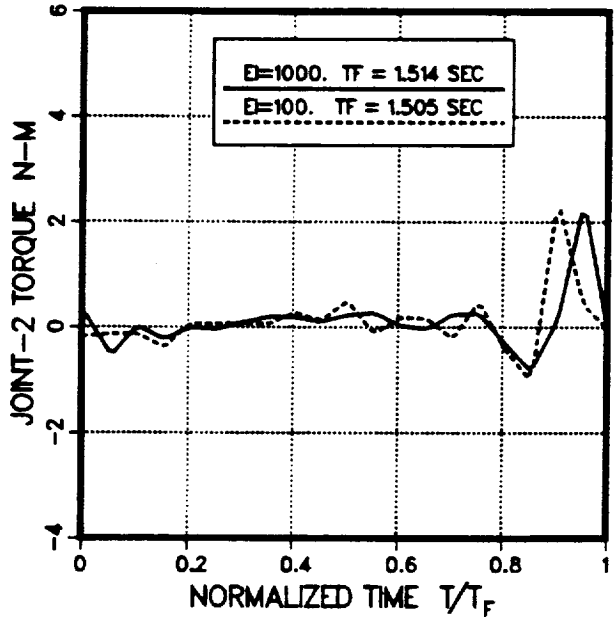


Figure 5: τ_2 joint torque histories

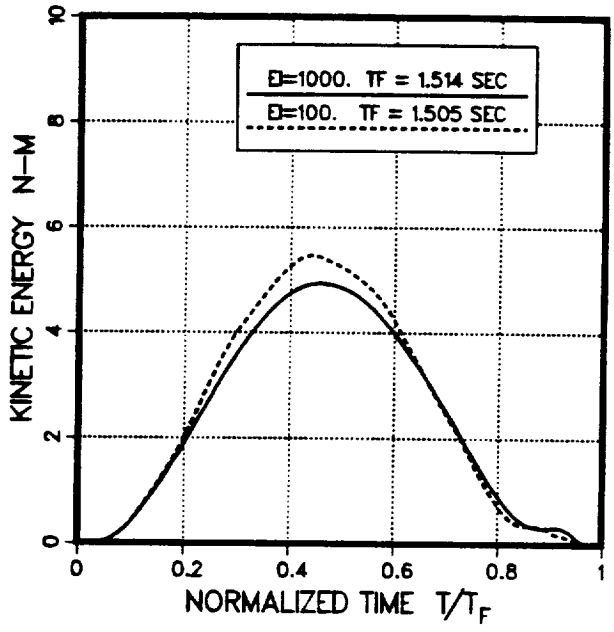


Figure 7: Kinetic energy histories

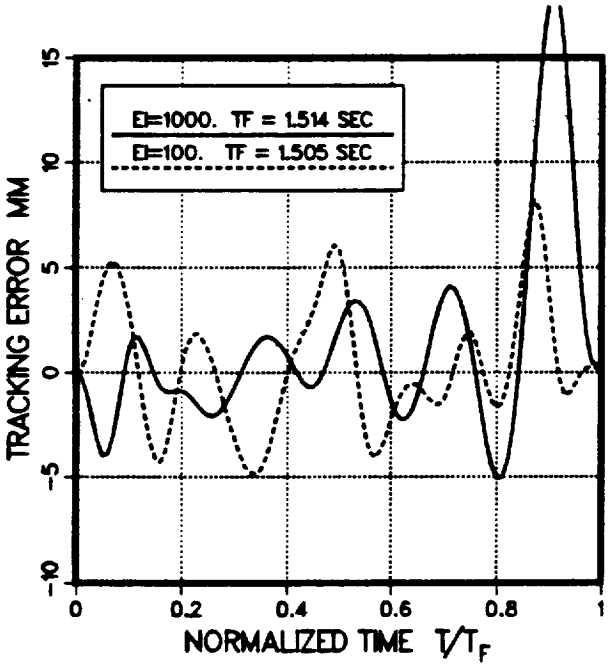


Figure 6: Straight-line tracking error

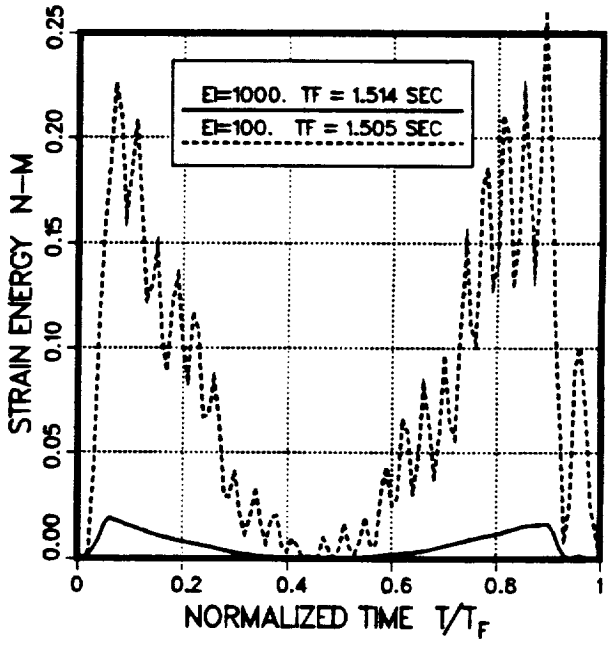


Figure 8: Strain energy histories

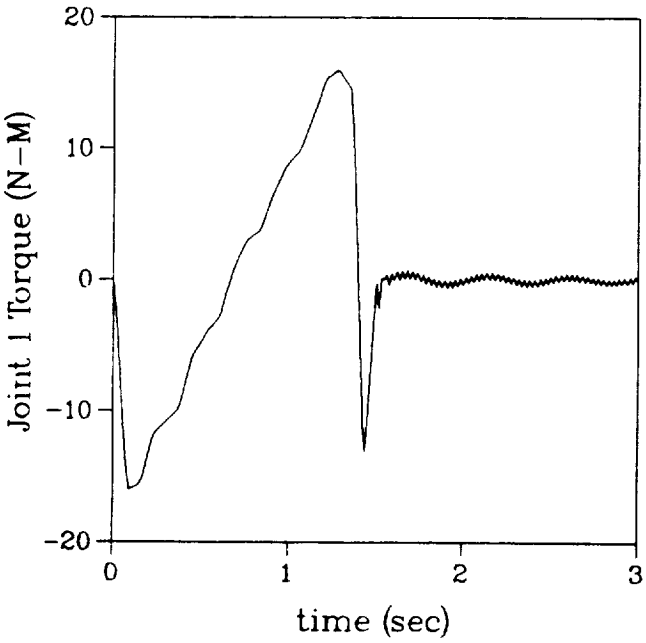


Figure 9: Residual τ_1 for $EI_{links} = 100$

MODELING OF CONTROL FORCES FOR KINEMATICAL CONSTRAINTS IN THE DYNAMICS OF MULTIBODY SYSTEMS-A NEW APPROACH

SITKI KEMAL IDER
Assistant Professor
Department of Mechanical Engineering
Middle East Technical University, Ankara, Turkey

ABSTRACT

Conventionally kinematical constraints in multibody systems are treated similar to geometrical constraints and are modeled by constraint reaction forces which are perpendicular to constraint surfaces. However, in reality, one may want to achieve the desired kinematical conditions by control forces having different directions in relation to the constraint surfaces. In this paper the conventional equations of motion for multibody systems subject to kinematical constraints are generalized by introducing general direction control forces. Conditions for the selections of the control force directions are also discussed. A redundant robotic system subject to prescribed end-effector motion is analyzed to illustrate the methods proposed.

1. INTRODUCTION

In many applications of multibody systems certain points are desired to follow prescribed paths, such as the end-effector in a robotic system. Such kinematical conditions are treated constraint equations to determine the system motion and the control forces.

In this paper those constraints which arise from geometrical restrictions such as closed loops and physical guides are termed geometrical constraints. On the other hand, kinematical constraints are defined as those conditions which represent desired motions or desired paths of certain points or bodies.

In the conventional methods of analysis, regardless of the fundamental dynamic equations (Newton-Euler, Lagrange, Kane, etc.) used, the constraints are modeled by constraint reaction forces which are perpendicular to the constraint surfaces. (See Arnold [1], Hemami and Weimer [2], Kamman and Huston [3], Wehage and Haug [4], Nikravesh [5], Kim and Vanderploeg [13], Amirouche and Jia [6].)

However kinematical constraints do not have to be satisfied by constraint reaction forces, and usually have to be realized by control forces applied by the actuators in the system. Hence the conventional

solution procedure imposes an arbitrary restriction to the directions of the control forces. Depending on the places of the actuators in the system one may want to achieve the desired kinematical conditions by control forces having different directions in relation to the constraint surfaces.

In this paper the conventional equations of motion are generalized by inroducing general direction control forces for kinematical constraints, that replaces the constraint force representation. And the dynamic equations for multibody systems subject to geometrical and kinematical constraints are developed. By the proposed method of solution the control forces and the system motion are solved simultaneously.

This paper is divided into five sections. After the introduction, the second section outlines the conventional equations of motion for constrained multibody systems. In the third section the general direction control forces for kinematical constraints are introduced and the conditions for the control force directions are discussed. In section four simulations of a redundant manipulator by the conventional and the proposed methods are presented.Conclusions form the last section.

2. CONVENTIONAL EQUATIONS OF MOTION

Consider a mechanical system where $q_1,\ldots,q_n$ are a set of generalized coordinates chosen for convenience to specify the configuration of the system. Let the system be subject to c constraints. Kane's equations for an arbitrary system of particles and rigid bodies can be expressed as (Kamman and Huston [3], Baumgarte [7]),

$$F^* + F + F^c = 0 \tag{1}$$

where

$$F_p^c = \lambda_i \frac{\partial f_i}{\partial y_p} \qquad i=1,\ldots,c \ , \quad p=1,\ldots,n \tag{2}$$

F^*, F and F^c are the vectors of generalized inertia, external and constraint forces respectively. In equation (2) $f_i=0$, $i=1,\ldots,c$ are the constraint equations in the acceleration level, λ_i are undetermined multipliers, and $y_1,\ldots,y_n$ are the generalized speeds of the system chosen for convenience as independent linear combinations of $\dot{q}_p$. The tranformation between $\dot{q}_p$ and y_p, e.g. Euler angle derivatives and relative angular velocity components can be expressed as

$$\dot{q}_h = T_{hp}\, y_p \qquad h,p=1,\ldots,n \tag{3}$$

where T_{hp} are functions of q_p (Kane and Levinson [8]).

The generalized inertia forces can be expressed in the following form (Huston and Passarello [9]),

$$F^* = M\,\dot{y} + Q \tag{4}$$

where M is the generalized mass matrix of the unconstrained system being functions of q_p, and Q contains the quadratic velocity terms.

The holonomic and nonholonomic constraint equations can be expressed in the acceleration level as below

$$B_{ip}\,\dot{y}_p = h_i \qquad i=1,\ldots,c \tag{5}$$

In eq. (5) B is cxn constraint matrix, and B_{ip} and h_i are in general functions of q_p and y_p.

Note that for holonomic constraints $\Phi_i(q_p,t)=0$,

$$B_{ip} = \frac{\partial \Phi_i}{\partial q_h}\,T_{hp}$$

and for velocity level nonholonomic constraints $\Psi_i(q_p,y_p,t)=0$,

$$B_{ip} = \frac{\partial \Psi_i}{\partial y_p}$$

Then, using eq. (2), the generalized constraint forces are

$$F^c = B^T \lambda \tag{6}$$

The undetermined multipliers λ_i represent the restraining constraint forces and moments generated by the constraints at the points of application (Ider and Amirouche [11]).

Equations (1) and (5) represent the governing dynamical equations. Combining these and making use of equations (4) and (6), we have

$$\begin{bmatrix} M & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \dot{y} \\ \lambda \end{bmatrix} = \begin{bmatrix} F-Q \\ h \end{bmatrix} \tag{7}$$

The accelerations obtained from eq. (7) are then used for numerical integration for the time history of y_p and, through the use of eq. (3), the generalized coordinates q_p.

Lagrange multipliers could be eliminated to reduce the equations for computational efficiency. To this end let C represent a nx(n-c) matrix which is orthogonal complement to B, obtained either by Singular value decomposition (Singh and Likins [12]), Zero eigenvalue method (Kamman and Huston [3]), Q-R decomposition (Kim and Vanderploeg [13], Amirouche and Jia [6]), or row equivalence transformation (Ider and Amirouche [10]). Premultiplying eq. (1) by C^T yields

$$C^T(F^*-F) = 0 \tag{8}$$

Combining equations (8) and (5) and utilizing eq. (4), we obtain the reduced equations

$$\begin{bmatrix} C^T M \\ \hline B \end{bmatrix} \dot{y} = \begin{bmatrix} C^T(F-Q) \\ \hline h \end{bmatrix} \tag{9}$$

Equations (9) and (3) form a set of 2n first order ordinary differential equations that can be numerically integrated to obtain the time history of y_p and q_p.

When relative joint coordinates are selected as the generalized coordinates and the corresponding partial velocity vectors are developed using recursive multibody kinematics (Huston and Passarello [9], Ider and Amirouche [10]), constraint equations for joint connections are automatically eliminated. Hence, in this paper an open tree-like system represents an unconstrained system where n is the total number ofthe free joint degrees of freedom.

3. CONTROL FORCES FOR KINEMATICAL CONSTRAINTS

Now consider that a tree-like multobody system is subject to geometrical and kinematical constraints. Kinematical constraints represent desired motions or desired paths of certain points or bodies. They are the conditions that have to be realized by the actuators in the system. The desired motions could be specified at position, velocity or acceleration levels and could be holonomic or nonholonomic.

Whether one uses Newton-Euler, Lagrange or Kane's equations or other variations of these, in the conventional approach the constraints in the system are modeled by constraint reaction forces which are perpendicular to the constraint surfaces. In the case of geometrical constraints perpendicular reaction forces at the application points are generated, hence the above approach is necessary. On the other hand kinematical constraints could be achieved by a number of alternative control forces whose directions in the generalized space can be be selected by physical considerations. Therefore the conventional equations of motion should be generalized by considering general direction control forces for kinematical constraints.

Consider c constraint equations (5), and let c_1 of the constraints in the system be geometrical and the remaining c_2 ($c_2=c-c_1$) be kinematical. The constraint matrix B and the vector of constraint force magnitudes λ can be partitioned such that

$$B = [B^{G^T} \quad B^{K^T}]^T \tag{10}$$

and

$$\lambda = [\lambda^{G^T} \quad \lambda^{K^T}]^T \tag{11}$$

where B^G is a $c_1$xn matrix, B^K is a c_2 matrix, λ^G is a c_1 dimensional vector and λ^K is a c_2 dimensional vector.

Addition of control forces to the equations of motion yields

$$M\dot{y} + Q + B^{G^T}\lambda^G + B^{K^T}\lambda^K + A^T\mu = F \tag{12}$$

where A is a $c_2$xn control force matrix where each row represents the direction of the control force for each kinematical constraint in the generalized space, and μ is c_2 dimensional vector of control force magnitudes. Now assume that the control force directions and magnitudes are selected such that the restraining constraint forces λ^G become zero. This leads to

$$M\dot{y} + Q + B^{G^T}\lambda^G + A^T\mu = F \tag{13}$$

Eq. (13) can be written in the following form

$$M\dot{y} + Q + Z^T\nu = F \tag{14}$$

where Z is the augmented matrix of constraint and control force directions

$$Z = [B^{G^T} \quad A^T]^T \tag{15}$$

and is the vector of constraint and control force magnitudes,

$$\nu = [\lambda^{G^T} \quad \mu^T]^T \tag{16}$$

Once the control force directions A are selected by physical considerations eq. (14) needs to be solved together with eq. (5) to determine the control force magnitudes simultaneously with the generalized accelerations. Hence the augmented equations of motion are

$$\begin{bmatrix} M & Z^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \dot{y} \\ \nu \end{bmatrix} = \begin{bmatrix} F-Q \\ h \end{bmatrix} \tag{17}$$

Alternatively the equations could be reduced in a manner similar to Section 2. To this end, let $\bar{C}$ be a nx(n-c) matrix orthogonal complement to Z. Premultiplying eq. (14) by $\bar{C}^T$ and augmenting with eq. (5) leads to

$$\begin{bmatrix} \bar{C}^T M \\ \hline B \end{bmatrix} \dot{y} = \begin{bmatrix} \bar{C}^T(F-Q) \\ \hline h \end{bmatrix} \tag{18}$$

In the case when the reduced equations are used ν can be obtained from eq. (14) utilizing the computed accelerations, as

$$\nu = (Z\,Z^T)^{-1}\,Z\,(F-My-Q) \tag{19}$$

Note that A should be selected such that rank Z=c, because otherwise there will be less than c_2 control forces to control c_2 kinematical conditions.

The augmented equations have a solution if and only if the augmented mass matrix is nonsingular, in which case the prescribed conditions are achieved with the corresponding control forces. On the other hand if it is not physically possible to realize the kinematical constraints with the selected control force directions, this reveals itself as a singular (or near singular) augmented mass matrix. Therefore the condition for the existence of solution could be expressed as follows: Directions A should be chosen such that a linear combination of the rows of $\bar{C}^T M$ should not be a linear combination of the rows of B. In other words the vector space spanned by the rows of $\bar{C}^T M$ and the vector space spanned by the rows of B should be nonintersecting (except the zero vector). Since the dimension of the vector space spanned by the rows of B is c, and that of $\bar{C}^T M$ is n-c, the possibilities for $\bar{C}^T M$ are infinitely many (provided that n-c>0). Hence one can construct various vector spaces for $\bar{C}^T M$ by different selections of the control force directions A. $C^T M$ that correspond to directions B is only one of them.

For redundant systems, i.e. when c<n, it has been observed that one usually has several physically meaningful control force directions to satisfy the given kinematical conditions.

In the special case when A is selected such that its rows are linear combinations of the rows of B^K, then since $\bar{C}$ is the same as C in the conventional model, y becomes the same as the conventional case and $Z^T \nu$ becomes equal to $B^T \lambda$. However ν_i may be different than λ_i depending on A.

Similarly for nonredundant systems, i.e. n=c, B is nxn, and rows of Z are necessarily linear combinations of the rows of B. In this case $\bar{C}$ is null matrix and the above procedure reduces to the conventional method where $Z^T \nu$ is equal to $B^T \lambda$.

It should be noted that $\bar{C}^T M$ and B may form nonintersecting vector spaces even if C^T and B do not. Hence realization of the prescribed motions is possible even in the extreme case when a control force direction is tangent to the corresponding constraint surface. This is due to the inertia coupling between the generalized coordinates.

4. SIMULATIONS OF A REDUNDANT ROBOTIC SYSTEM

In the three link manipulator shown in Figure 1, the configuration of the system can be described by three generalized coordinates θ_1, θ_2, θ_3. The generalized speeds y_p are defined as

$$y_1 = \dot{\theta}_1 \ , \ y_2 = \dot{\theta}_2 \ , \ y_3 = \dot{\theta}_3$$

The data used are $L_1=L_2=L_3=1.0m$, $m_1=30kg$, $m_2=m_3=18kg$, $I_1=10$ $kg.m^2$, $I_2=I_3=8.64$ $kg.m^2$.

The end-effector (point A) is desired to move in the horizontal direction with a constant velocity v^A. Hence the constraint equations in the system are

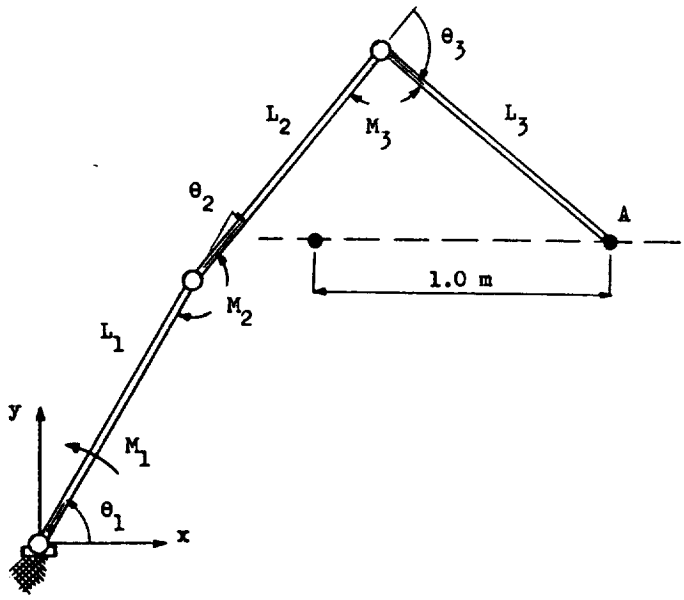


Figure 1. Three link manipulator

$$L_1c_1 + L_2c_{12} + L_3c_{13} = v^At + 1.9088$$
$$L_1s_1 + L_2s_{12} + L_3s_{13} = 0.9893 \qquad (20)$$

where $c_1=\cos\theta_1$, $c_{12}=\cos(\theta_1+\theta_2)$, $c_{13}=\cos(\theta_1+\theta_2+\theta_3)$, and similarly $s_1=\sin\theta_1$, $s_{12}=\sin(\theta_1+\theta_2)$, $s_{13}=\sin(\theta_1+\theta_2+\theta_3)$. At the acceleration level the constraint equations are given by eq. (5) where B and h are

$$B = \begin{bmatrix} L_1s_1+L_2s_{12}+L_3s_{13} & L_2s_{12}+L_3s_{13} & L_3s_{13} \\ L_1c_1+L_2c_{12}+L_3c_{13} & L_2c_{12}+L_3c_{13} & L_3c_{13} \end{bmatrix} \qquad (21)$$

and

$$h = \begin{bmatrix} -L_1s_1y_1^2-L_2s_{12}(y_1+y_2)^2-L_3s_{13}(y_1+y_2+y_3)^2 \\ -L_1c_1y_1^2-L_2c_{12}(y_1+y_2)^2-L_3c_{13}(y_1+y_2+y_3)^2 \end{bmatrix} \qquad (22)$$

Initially the system is at the configuration $\theta_1=60^o$, $\theta_2=-10^o$, $\theta_3=-90^o$. The initial generalized speeds are $y_1=0.386$ rad/sec, $y_2=0$, $y_3=-0.9618$ rad/sec, which correspond to $v^A=-1$ m/sec.

First the system is simulated using the conventional method. The generalized constraint forces can be expressed from equations (6) and (21) as,

$$\begin{bmatrix} F_1^c \\ F_2^c \\ F_3^c \end{bmatrix} = \begin{bmatrix} \lambda_1 (L_1 s_1 + L_2 s_{12} + L_3 s_{13}) + \lambda_2 (L_1 c_1 + L_2 c_{12} + L_3 c_{13}) \\ \lambda_1 (L_2 s_{12} + L_3 s_{13}) + \lambda_2 (L_2 c_{12} + L_3 c_{13}) \\ \lambda_1 L_3 s_{13} + \lambda_2 L_3 c_{13} \end{bmatrix} \tag{23}$$

In particular we wish to determine joint moments denoted as M_i, $i=1,2,3$ that would achieve the desired kinematical conditions. F_i^c, $i=1,2,3$ represent the required joint moments. It is seen from eq. (23) that all three joint moments are nonzero, i.e. motors are required at all three joints.

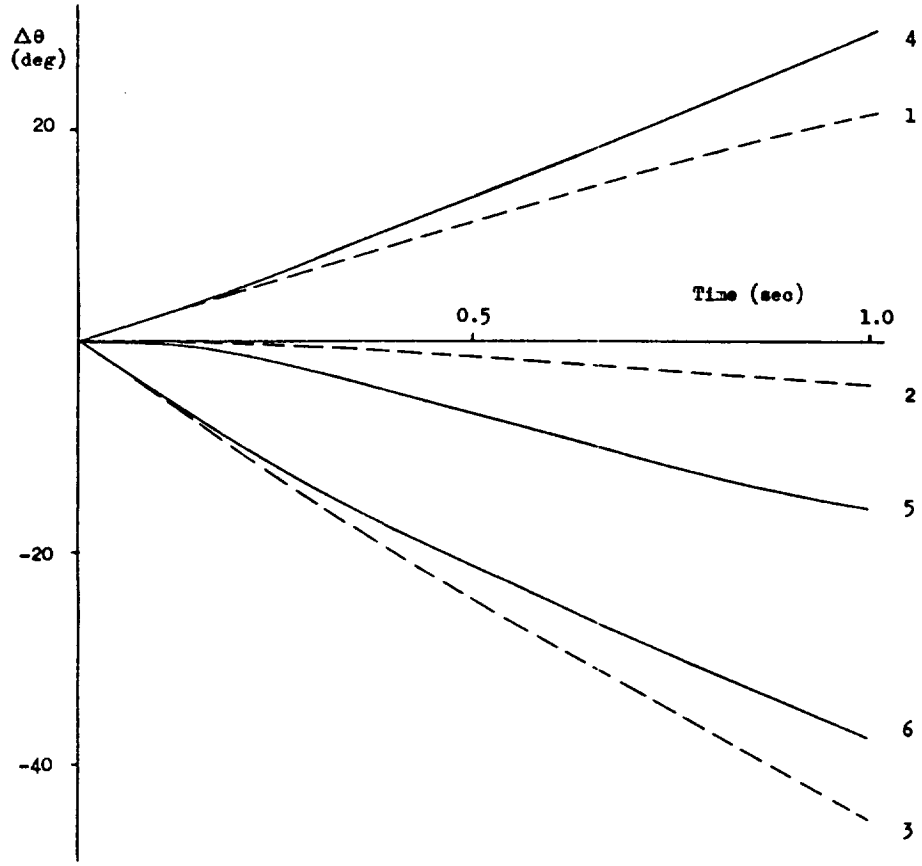


Figure 2. Displacements.
Conventional method : 1. $\Delta\theta_1$, 2. $\Delta\theta_2$, 3. $\Delta\theta_3$
Control force method: 4. $\Delta\theta_1$, 5. $\Delta\theta_2$, 6. $\Delta\theta_3$

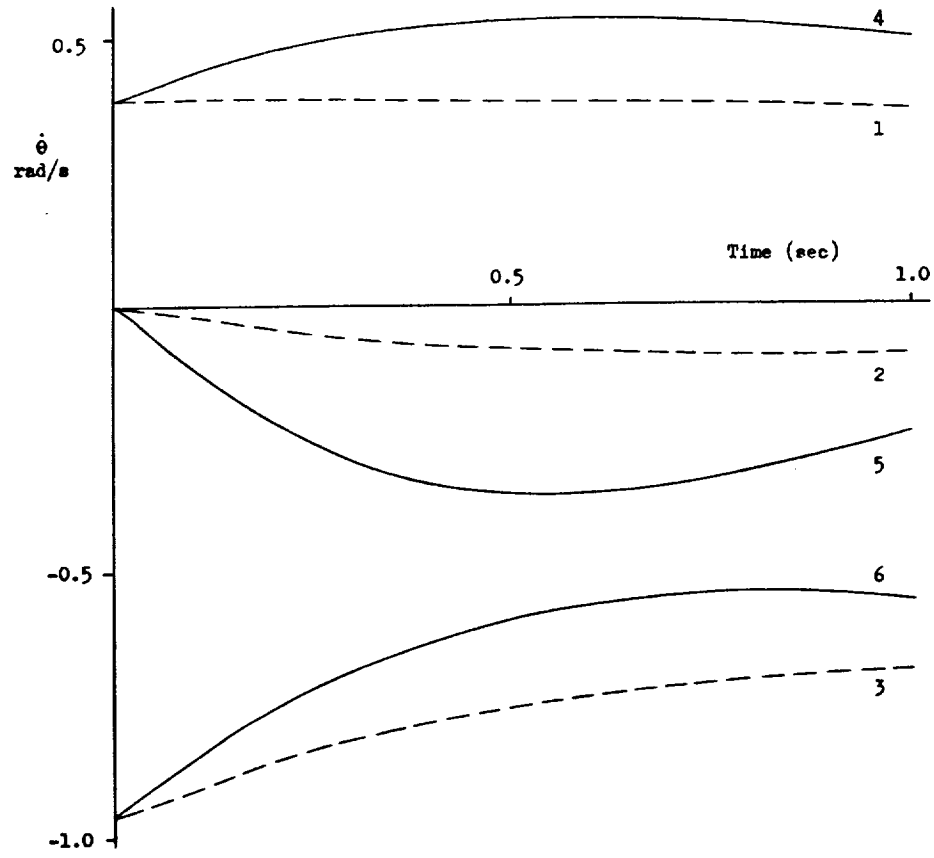


Figure 3. Velocities.
Conventional method : 1. $\dot{\theta}_1$, 2. $\dot{\theta}_2$, 3. $\dot{\theta}_3$
Control force method: 4. $\dot{\theta}_1$, 5. $\dot{\theta}_2$, 6. $\dot{\theta}_3$

The simulation is performed for 1 sec., until the end effector moves 1m in -x direction. $\Delta\theta_i$ and $\dot{\theta}_i$, i=1,2,3 are plotted in Figures 2 and 3 respectively. The joint moments M_i required to obtain the desired motion of the end effector as obtained by the conventional method are shown in Figure 4.

Second the system is resimulated by the proposed control force approach. As an illustration the control force directions are selected such that no moments are needed at the lower joint of link 1. The corresponding control force directions are

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{24}$$

Note that since there are no geometrical constraints in this system, the

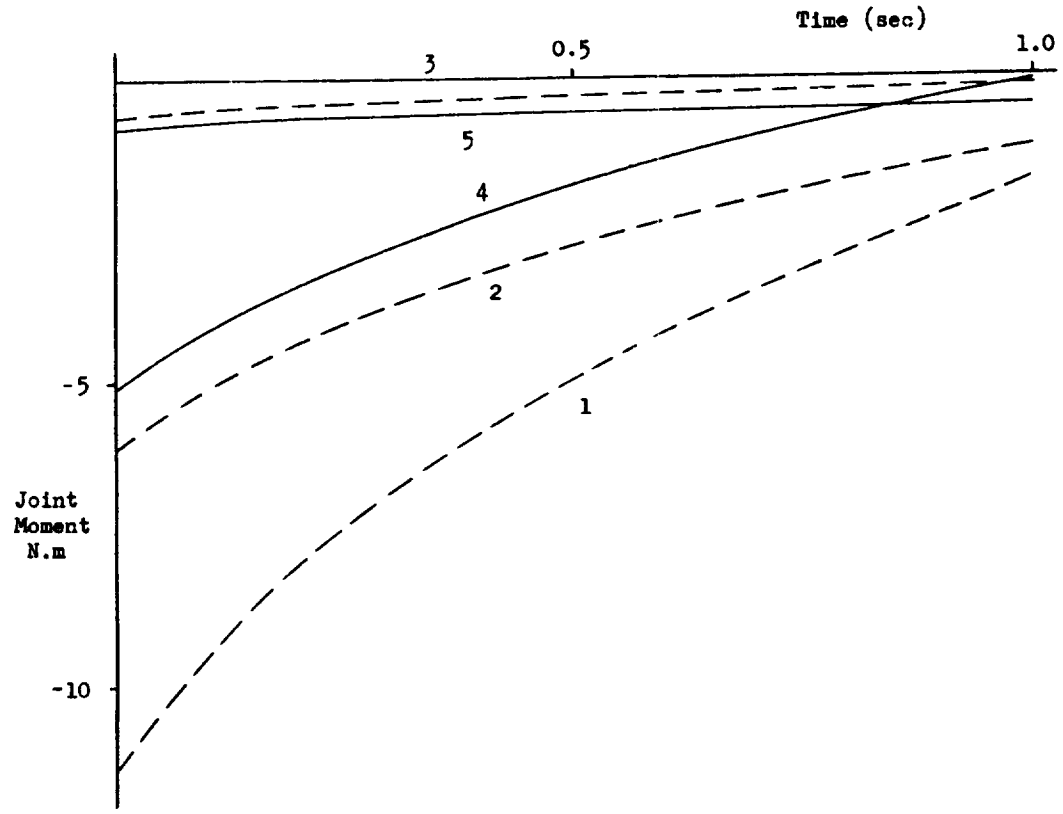


Figure 4. Joint moments.
Conventional method : 1. M_1, 2. M_2, 3. M_3
Control force method: 4. M_2, 5. M_3

matrix Z is identical to A, and the vector ν is identical to μ. The control forces $Z^T\nu$ are

$$Z^T\nu = \nu_1\begin{bmatrix}0\\1\\0\end{bmatrix} + \nu_2\begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}0\\\nu_1\\\nu_2\end{bmatrix} \tag{25}$$

Hence, in this case, the required joint moments for the prescribed end effector motion are $M_1=0$, $M_2=\nu_1$, $M_3=\nu_2$.

The augmented mass matrix was observed to be full rank as expected. θ_i and $\dot{\theta}_i$, i=1,2,3 for a simulation of 1 sec. are plotted in Figures 2 and 3. The joint moments are plotted in Figure 4. Note that in this case a motor is not needed at the lower joint of link 1.

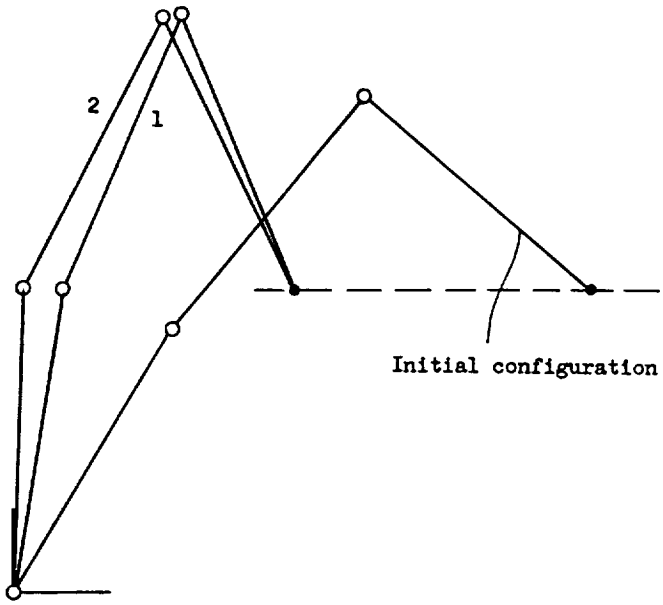


Figure 5. Final configurations: 1. Conventional method; 2. Control force method.

The system's motion differ in the above two approaches although in both cases the end effector performs the same motion. The configurations at t=1 sec. corresponding to the conventional model and the control force model are shown in Figure 5.

5. CONCLUSIONS

This paper presented a general procedure for the dynamic modeling of multibody systems subject to kinematical constraints. General direction control forces have been introduced that replace the conventional constraint reaction forces, hence increasing the ways of realization of the prescribed motions. It is shown that the possible conrol force directions are more than one, and the criteria for the existence of solution have been presented.

The method proposed in this paper involves selecting the control force directions in the generalized space by physical considerations, and then solving their magnitudes simultaneously with the corresponding system motion. As a result one can design alternative control forces that can be applied by the actuators in the system. The method is expected to be especially useful to control the extra degrees of freedom in systems that have joint flexibility or joint clearance.

REFERENCES

1. Arnold, V. I., 1978, Mathematical Methods in Classical Mechanics, Springer-Verlag, New York.

2. Hemami, H., Weimer, F. C., 1981, "Modelling of Nonholonomic Dynamic Systems with Applications", ASME Journal of Applied Mechanics, Vol. 48, No.1, pp. 177-182.

3. Kamman, J. W., Huston, R. L., 1984, "Dynamics of Constrained Multibody Systems", ASME Journal of Applied Mechanics, Vol. 51, No. 4, pp. 899-903

4. Wehage, R. A., Haug, E. J., 1982, "Generalized Coordinate Pertitioning for Coordinate Reduction in Analysis of Constrained Dynamic Systems", ASME Journal of Mechanical Design, Vol. 104, pp. 247-255.

5. Nikravesh, P. E., 1984, "Some Methods for Dynamics of Constrained Mechanical Systems: A Survey", NATO ASI Series, Vol. F9, Springer-Verlag, Berlin, pp. 351-368.

6. Amirouche, F. M. L., Jia, T., 1987, "Automatic Elimination of the Undetermined Multipliers in Kane's Equations using a Pseudo-Uptriangular Decomposition Method", Computers and Structures, Vol.27, p. 203.

7. Baumgarte, J., 1972, "Stabilization of Constraints and Integrals of Motion in Dynamical Systems", Computational Methods in Applied Mechanical Engineering, Vol. 1, pp. 1-16.

8. Kane, T. R., Levinson, D. A., 1985, Dynamics: Theory and Applications, McGraw-Hill, New York.

9. Huston, R. L., Passarello, C. E., 1980, "Multibody Structural Dynamics Including Translation between the Bodies", Computers and Structures, Vol. 12, pp. 713-720.

10. Ider, S. K., Amirouche, F. M. L., 1988, "Coordinate Reduction in the Dynamics of Constrained Multibody Systems", ASME Journal of Applied Mechanics, Vol. 55, No. 2, pp. 899-905.

11. Ider, S. K., Amirouche, F. M. L., 1989, "Numerical Stability of the Constraints Near Singular Positions in the Dynamics of Multibody Systems", Computers and Structures, Vol. 32, No.5.

12. Singh, R. P., Likins, P. W., 1985, "Singular Value Decomposition for Constrained Dynamical Systems", ASME Journal of Applied Mechanics, Vol. 52, pp. 943-948.

13. Kim, S. S., Vanderploeg, M. J., 1986, "QR Decomposition for State Space Representation of Constrained Mechanical Dynamic Systems", ASME J.of Mech., Trans. and Auto. in Design", Vol.108, pp.183-188.

APPLICATION OF NUMERICAL OPTIMIZATION TECHNIQUES TO CONTROL SYSTEM DESIGN FOR NONLINEAR DYNAMIC MODELS OF AIRCRAFT

C. Edward Lan and Fuying Ge
Department of Aerospace Engineering
The University of Kansas, Lawrence, Kansas 66045

ABSTRACT

Control system design for general nonlinear flight dynamic models is considered through numerical simulation. The design is accomplished through a numerical optimizer coupled with analysis of flight dynamic equations. In the analysis, the general flight dynamic equations are numerically integrated and dynamic characteristics are then identified from the dynamic response. The design variables are determined iteratively by the optimizer to optimize a prescribed objective function which is related to desired dynamic characteristics. Generality of the method allows nonlinear effects of aerodynamics and dynamic coupling to be considered in the design process. To demonstrate the method, nonlinear simulation models for an F-5A and an F-16 configurations are used to design dampers to satisfy specifications on flying qualities and control systems to prevent departure. The results indicate that the present method is simple in formulation and effective in satisfying the design objectives.

INTRODUCTION

At high angles of attack, the aerodynamic forces and moments are, in general, time-dependent and nonlinear functions of motion variables. Therefore, the traditional control system design method based on a linearized dynamic system are not appropriate. In addition, the aerodynamic, kinematic, and inertial coupling phenomena are important to the high angle-of-attack flight dynamics of modern aircraft. As a result, a number of high angle-of-attack control concepts have emerged (refs. 1-4). Therefore, a suitable control system design method must be capable of incorporating these coupling phenomena with considerations of time-dependent, nonlinear aerodynamic forces and moments. A control system designed without considering these coupling phenomena often has a detrimental effect on the departure/spin resistance (ref. 5). Another feature of high-alpha control system is the simultaneous utilization of several control surfaces or devices. Therefore, a design method capable of handling multiple input and output is necessary. A current approach to solving this problem is by extensive piloted simulation (ref. 5).

Methods in optimal control theory represent possible approaches to solving these problems under consideration. These methods are derived through calculus of variation. However, computational methods in existence require linearization of dynamic equations and aerodynamics about trimmed conditions (ref. 6). Another alternative is to apply numerical optimization techniques without linearization as they are frequently used in structural and aerodynamic designs of large systems. A similar approach has also been used in other control applications in ref. 7.

In the present method, a numerical optimization technique based on conjugate gradients and feasible directions (ref. 8) is coupled with an analysis method which is to obtain numerical solutions of the nonlinear six degree-of-freedom dynamic equations. This analysis method is to provide information needed in the design process, such as damping ratios, frequencies, motion variables involved in dynamic instabilities, etc. Since the analysis method can deal with nonlinearities in the dynamics and the aerodynamics and with any general constraints on the control system configuration, the control system designed with a numerical optimization technique can be very realistic and effective.

NUMERICAL APPROACHES

Typically, a control system may be designed to enhance flying qualities, to prevent flight departure, and to have an effective maneuver control system. To demonstrate the present method, only the first two objectives will be considered. That is, one is to design dampers at a moderate angle of attack to satisfy specifications on flying qualities and the other to design a control system to prevent flight departure at high angles of attack in a maneuver. Numerical formulations to solve these problems are described in the following.

Design to Satisfy Flying Qualities Specifications

The general system of equations can be written as

$$m(\dot{u} - vr + wq) = mg_x + F_{A_x} + F_{T_x} \tag{1a}$$

$$m(\dot{v} + ur - wp) = mg_y + F_{A_y} + F_{T_y} \tag{1b}$$

$$m(\dot{w} - uq + vp) = mg_z + F_{A_z} + F_{T_z} \tag{1c}$$

$$I_{xx}\dot{p} - I_{xz}\dot{r} - I_{xz}pq + (I_{zz} - I_{yy})rq = L_A + L_T \tag{1d}$$

$$I_{yy}\dot{q} + (I_{xx} - I_{zz})pr + I_{xz}(p^2 - r^2) = M_A + M_T \tag{1e}$$

$$I_{zz}\dot{r} - I_{xz}\dot{p} + (I_{yy} - I_{xx})pq + I_{xz}qr = N_A + N_T \tag{1f}$$

$$\dot{\phi} = p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \tag{1g}$$

$$\dot{\theta} = q\cos\phi - r\sin\phi \tag{1h}$$

$$\dot{\psi} = (q\sin\phi + r\cos\phi)\sec\theta \tag{1i}$$

$$\alpha = \tan^{-1}(w/u) \tag{1j}$$

$$\beta = \sin^{-1}(v/\sqrt{u^2 + v^2 + w^2}) \tag{1k}$$

where (u, v, w) are the three linear velocity components of the aircraft; (p, q, r) are the angular velocity components; and (ϕ, θ, ψ) are the Euler angles in bank, pitch, and yaw, respectively. The subscripts (x, y, z) appearing on the right-hand side of Eqs. (1a) - (1c) denote components in the corresponding coordinate directions; and (A, T) denote aerodynamic and thrust components, respectively. g is the gravitational acceleration, and F's are the external forces, while (L, M, N) are the moments about the (x-, y-, z-) axes. In addition, m is the mass and I_{xx}, I_{xz}, etc., are the moments of inertia. The aerodynamic forces and moments (F_A, L_A, M_A, N_A), including the control effects, are represented in dimensionless coefficients in a tabulated form as functions of motion variables in this study. The motion variables are (u, v, w, p, q, r).

This system of equations is numerically integrated from an initial state (usually a trimmed level flight condition) after disturbances (such as impulsive control-surface deflections) are imposed to generate time-history data of motion variables.

For demonstrative purposes, it is assumed that dampers to provide flight characteristics satisfying flying-qualities specifications are to be determined. This problem has been solved in the past by conventional methods, such as the root-locus method, by using linearized equations of motion. It is considered here mainly to show the generality of the present method even without linearizing the equations of motion. In the present method, the necessary design information includes damping ratios, natural frequencies, and time constants of the vehicle motions. These characteristics are identified from calculated time-history results of motion variables after multiple-axis disturbances are imposed. The numerical method used for parameter identification is the method of differential corrections described in the following.

A general discretized system output in the time domain is assumed to be of the form:

$$f(t_k) = \sum_{i=1}^{n} e^{-\zeta_i \omega_{n_i} t_k} (A_i \cos\omega_i t_k + B_i \sin\omega_i t_k) + \sum_{j=1}^{m} C_j e^{-\sigma_j t_k} + Dt_k + E \qquad (2)$$

where $t_k = \Delta t(k - 1)$, $k = 1, 2, \ldots, K$, $\omega_i = \omega_{n_i}\sqrt{1 - \zeta_i^2}$ is the damped frequency of the i^{th} mode. The objective is to use Eq. (2) to fit the dynamic response data [$Q_k = x(t_k)$] through the method of least squares to determine the damping ratios (ζ_i) natural frequencies (ω_{n_i}) and time constants ($1/\sigma_j$), $i = 1, \ldots, n$; $j = 1, \ldots, m$. These parameters appear nonlinearly in Eq. (2). Other unknowns, A_i, B_i, C_j, D, and E, are linear parameters in Eq. (2). Because of nonlinearity, finding a solution of the resulting nonlinear algebraic equations from the least-square formulation is difficult. The best approach, as it has been determined in the present investigation, is the method of differential corrections. In other words, the unknown parameters are expressed as

$$\zeta_i = \zeta_{i_k} + \Delta\zeta_i$$

$$\omega_{n_i} = \omega_{n_{i_k}} + \Delta\omega_{n_i}, \text{ etc., } i = 1, \ldots, n, \; j = 1, \ldots, m$$

where ζ_{i_k}, $\omega_{n_{i_k}}$, ... are the initial approximations of the i^{th} or j^{th} unknowns. Using Taylor series expansions, it is obtained that

$$Q_k + \varepsilon_k = f_o(t_k) + \sum_{i=1}^{n} \left(\Delta\zeta_i \frac{\partial f}{\partial \zeta_{i_k}} + \Delta\omega_{n_i} \frac{\partial f}{\partial \omega_{n_{i_k}}} + \Delta A_i \frac{\partial f}{\partial A_{i_k}} + \Delta B_i \frac{\partial f}{\partial B_{i_k}}\right)$$

$$+ \sum_{j=1}^{m} \left(\Delta C_j \frac{\partial f}{\partial C_{j_k}} + \Delta\sigma_j \frac{\partial f}{\partial \sigma_{j_k}}\right) + \Delta D \frac{\partial f}{\partial D_k} + \Delta E \frac{\partial f}{\partial E_k} + \ldots \tag{3}$$

where ε_k is the residual. The least-square method is then applied to Eq. (3) in such a way that

$$G = \sum_{k=1}^{K} \varepsilon_k^2 = \text{minimum}$$

The differential corrections ($\Delta\zeta_i$, etc.) which minimize the G-function are determined by setting the first derivatives, $\partial G/\partial(\Delta\zeta_i)$, etc, to zero. Once the differential corrections are determined, they are added to the initial estimates of the unknowns and the process is repeated to determine a new set of differential corrections until G is a minimum or until there is no significant change in the unknowns. Typically, convergence is assumed if $G < 10^{-7}$.

After the necessary design information is obtained from the analysis part of the algorithm, the optimizer is called to perform the design process.

The damper design problem here may be formulated as follows: find the pitch rate feedback gain K_q, the roll rate feedback gain K_p, the yaw rate feedback gain K_r, the lateral acceleration feedback gain K_{ay}, and the aileron-to-rudder interconnect gain K_{ARI}, such that the following objective function is minimized:

$$OBJ = \frac{-B1}{\varepsilon + E1 \times |\zeta_{sp_1} - \zeta_{sp}|} + \frac{-B2}{\varepsilon + E2 \times |\zeta_{p_1} - \zeta_p|}$$

$$+ \frac{-B3}{\varepsilon + E3 \times |\zeta_{D_1} - \zeta_D|} + \frac{-B4}{\varepsilon + E4 \times |\omega_{n_{D_1}} - \omega_{n_D}|}$$

$$+ \frac{-B5}{\varepsilon + E5 \times |T_{r_1} - T_r|} + \frac{-B6}{\varepsilon + E6 \times |T_{s_1} - T_s|} \tag{4}$$

where ζ_{sp_1}, ζ_{p_1}, ζ_{D_1}, $\omega_{n_{D_1}}$, T_{r_1}, and T_{s_1} are specified values to satisfy MIL-F-8785B. ζ_{sp}, ζ_p, ζ_D, ω_{n_D}, T_t, and T_s are values obtained in the analysis part. In Eq. (4), Bi and Ei are some weighting factors. ε in the denominator of the objective function is a small number used to prevent the objective function from being infinite and is set to 10^{-14} in the present algorithm. The optimization problem is subject to constraints on magnitudes of damping ratios, frequencies, time constants, overshoot, etc. In the optimization process, the design variables are varied systematically by the optimizer to obtain numerically the gradients of the objective function and constraints. These gradients are then used through the methods of conjugate gradients and feasible directions to determine the appropriate design variables to minimize the objective function. The process continues until the objective function does not change and the constraints are all satisfied.

Design to Prevent Flight Departure

Again, Eqs. (1) are numerically integrated. During time integration, a certain maneuver flight is imposed to induce departure of the airplane. One example of the maneuver flight is to pull up the airplane (i.e., to increase the angle of attack) and then induce a high roll rate afterwards. The present algorithm is constructed on the assumption that a departure condition is identifiable from the magnitude of the state vector, or motion variables. Since the latter are directly obtained from time integration of Eqs. (1), no further data manipulation is needed to calculate the necessary design information.

The design objective is achieved by first assuming a control system structure. Then the design problem may be formulated for the demonstration cases to be presented as follows.

Determine the aileron-rudder interconnect gain (K_{ARI}), the side acceleration feedback gain (K_{ay}), and the yaw damper gain (K_r), etc., to

minimize the following objective function:

$$OBJ = -C_1 p_{max} - C_2 \alpha_{trim} - \frac{C_3}{|\alpha_{max}| + \varepsilon} - \frac{C_4}{|\psi_{max}| + \varepsilon} - \frac{C_5}{|\beta_{max}| + \varepsilon} - \frac{C_6}{|r_{max}| + \varepsilon} - \frac{C_7}{|\theta| + \varepsilon} - \frac{C_8}{|\phi_{trim}| + \varepsilon} \tag{5}$$

subject to various constraints depending on applications. Note that Eq. (5) indicates that p (the roll rate) is to be maximized and α_{max} in the transient motion, ψ_{max} (yaw angle), β_{max} (sideslip) and r_{max} (yaw rate) are to be minimized. α_{trim} is calculated as the average angle of attack over the whole time period and may be used to define the limiting angle of attack to be discussed later for the F-16. Specific applications are discussed in the next section.

Two fighter configurations will be used to demonstrate the algorithm, one being an F-5A and the other an F-16. A pitch damper design for the F-5A will be considered first. Control to prevent flight departure will be discussed next. For illustrative purposes, all system gains in the following consideration are assumed constant.

A Pitch Damper Design for an F-5A

The algorithm has been tested and found to work well at different flight conditions to design dampers under multiple input conditions. At low angles of attack, calculated results are found to be consistent with existing systems. To demonstrate this computational tool, consider designing a pitch damper at a Mach number of 0.3 and at an altitude of 10,000 ft. The corresponding α_{trim} is determined to be 11.7 deg. Assume that a damping ratio of 0.65 (ζ_{sp_1}) is required in the longitudinal dynamic response of the short-period mode. The optimization problem may then be formulated as follows:

Determine the pitch damper gain constant (K_q) to
minimize the difference in the actual (ζ_{sp}) and desired (ζ_{sp_1}) damping ratios; and
subject to the constraints that

$$0 < \zeta_{sp} < 1$$

$$0 < \omega_{n_{sp}} < 10 \text{ rad/sec.}$$

Limitations on the control system are that
the pitch rate feedback be limited to 4 deg/sec,
the elevator deflection limits are +5.5 deg to -17 deg., and
the elevator actuator rate limits are -26 to +26 deg/sec.

Fig. 1 shows that the pitch damper gain constant to satisfy this design problem is 4.36. The existing system with $K_q = 0.2$ is not adequate to provide a damping ratio of 0.65. Note that during the design process, motions along all axes have been imposed to provide any possible effect of inertial coupling.

Control System to Prevent Yaw Divergence of an F-5A

The second example is to design a control system to prevent yaw divergence of an F-5A during roll maneuver at high angles of attack. The aircraft is placed in a departure condition by a maximum constant elevator deflection to increase the angle of attack, followed by a constant roll control deflection of 2 deg. The optimal control problem is formulated as follows.

Determine the aileron-rudder interconnect gain (K_{ARI}), the side acceleration feedback gain (K_{ay}), and the yaw damper gain (K_r), to

maximize the roll rate, and
minimize the sideslip angle (β), the yaw rate (r), and the change in heading angle (ψ).

In other words, in Eq. (5) the terms associated with α_{trim} and $|\alpha_{max}|$ are not used. Some results for the time variation of motion variables are shown in Fig. 2. It is seen that the present method is effective in satisfying the design objective by reducing both the change in yaw and bank angles.

Control System to Prevent Pitch Departure of an F-16

The aerodynamic data are obtained from ref. 9. Since the F-16 is unstable in pitch, design of a pitch control system is of major concern. The system includes an angle-of-attack and normal-acceleration feedback control. The airplane is first pulled up by applying the full stabilator deflection command (-25°). The objective is to minimize Eq. (5) with

$$C_1 = 0.01,\ C_2 = 0.03,\ C_3 = 1,\ C_4 = 12,\ C_5 = 0.01$$

$$C_6 = 0.008,\ C_7 = 1,\ C_8 = 1$$

These weighting factors are chosen so that various terms in Eq. (5) have the same order of magnitude. The design variables are the various gain constants. Note that the α-feedback system is defined such that

$$\delta_e \text{ due to } \alpha\text{-feedback} = K_\alpha \alpha - K_c \tag{6}$$

Two flight conditions are examined, one without imposing a roll maneuver after pull-up and the other with a roll maneuver. Results for the first case are presented in Fig. 3. It is seen that if there is no angle-of-attack limiting system ($K_\alpha = 0$, $K_c = 0$), the airplane will trim at an angle of attack equal to about 66 deg, which is the deep-stall condition. On the other hand, the limiting system would limit the trim angle of attack to about 25 deg.

For the second case, a roll control of -10 deg is applied between t = 22 and 34 sec. Note that roll should induce pitch-up due to inertial coupling. The results shown in Fig. 4 indicate that no departure has occurred and α_{trim} is determined to be 24.7 deg. By changing the initial time at which the roll control is applied, α_{trim} can still be determined to be about 25 deg. Therefore, it may be concluded that maximizing α_{trim} is to define approximately the limiting angle of attack.

CONCLUSIONS

Application of numerical optimization techniques to control system design was demonstrated for F-5A and F-16 configurations at high angles of attack. The methodology accounted for nonlinearities in aerodynamics and dynamics. Specific examples were presented to design control systems to satisfy flying qualities requirements and to prevent flight departure. The results indicated that the present method was effective in satisfying design objectives.

REFERENCES

1. Gilbert, W. P.; Nguyen, L. T.; and Van Gunst, R. W. "Simulator Study of Applications of Automatic Departure-and-Spin Prevention Concepts to Variable-Sweep Fighter Airplane." NASA TM X-2928, 1973.
2. Nguyen, L. T.; Gilbert, W. P.; and Van Gunst, R. W. "Simulator Study of the Departure Resistance of Lightweight Fighter Airplane with Twin Vertical Tails." NASA TM X-3510, 1977.
3. Gilbert, W. P.; Nguyen, L. T.; and Van Gunst, R. W. "Simulator Study of the Effectiveness of an Automatic Control System Designed to Improve the High-Angle-of-Attack Characteristics of a Fighter Airplane." NASA TND-8176, 1976.
4. Gilbert, W. P.; and Libbey, C. E. "Investigation of an Automatic Spin Prevention System for Fighter Airplanes." NASA TND-6670, 1972.
5. Nguyen, L. T.; Gilbert, W. P.; and Ogburn, M. E. "Control System Techniques for Improved Departure/Spin Resistance for Fighter Aircraft." NASA TP-1689, 1989.
6. Linse, D. J.; and Downing, D. R. "The Design and Analysis of a High Angle of Attack Flight Control System." KU-FRL-776-1, The University of Kansas Center for Research, Inc., July 1987.
7. Fan, M. K. H.; Wang, L. S.; Koninckx, J.; and Tits, A. L. "CONSOLE: A CAD Tandem for Optimization-Based Design Interacting with User-Supplied Simulators." Workshop on Computational Aspects in the Control of Flexible Systems, NASA TM-101578, 1988, pp. 89-108.
8. Vanderplaats, G. N. "COPES/ADS--A FORTORN Control Program for Engineering Synthesis Using the ADS Optimization Program." Engineering Design Optimization, Inc., June 1985.
9. Nguyen, L. T., et al. "Simulator Study of Stall/Post-Stall Characteristics of a Fighter Airplane with Relaxed Longitudinal Static Stability." NASA TP-1538, December 1979.

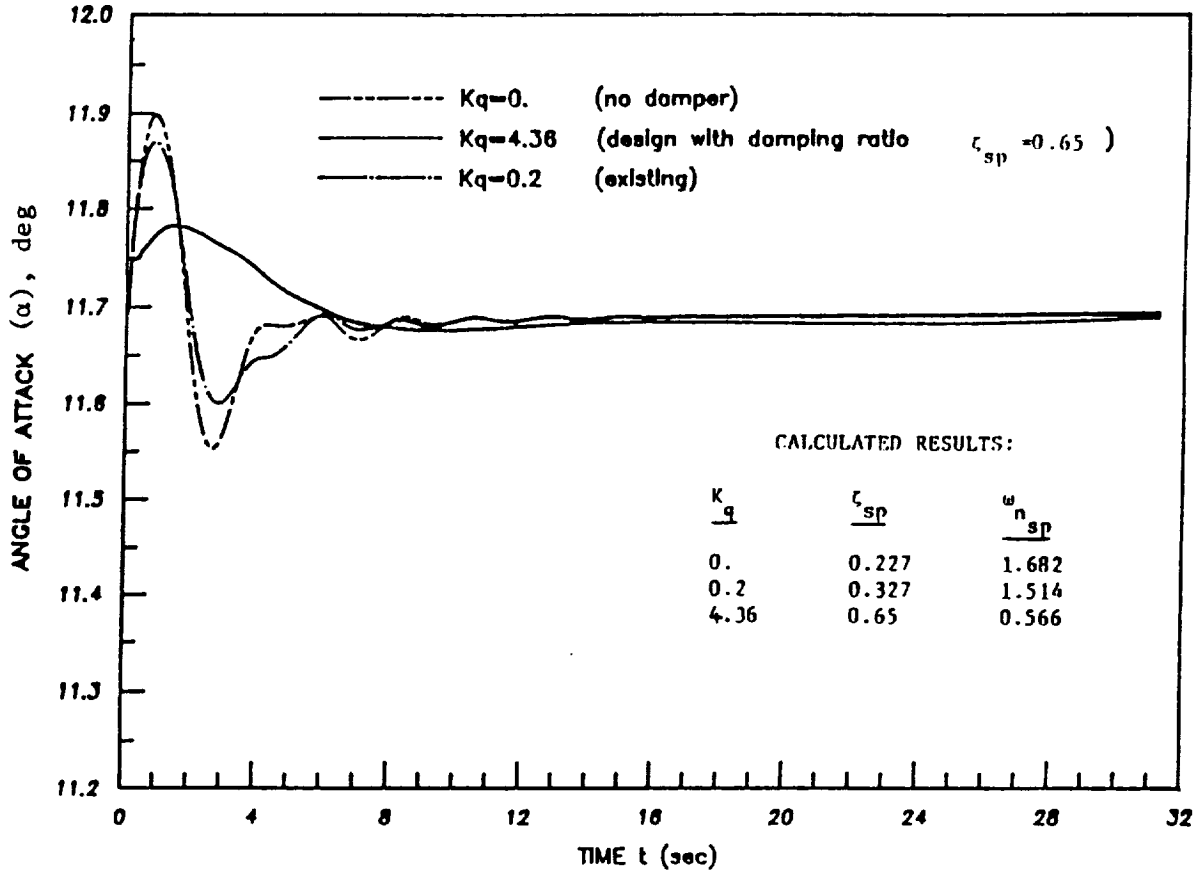
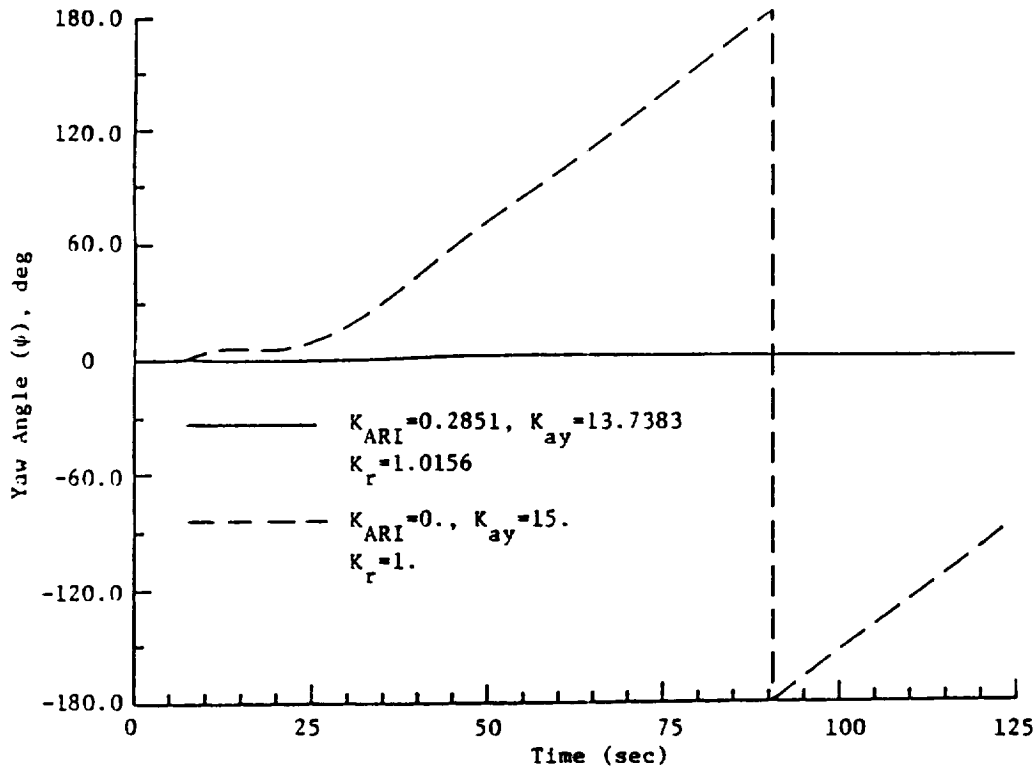


Figure 1 Time History of the Angle of Attack Responding to Impulsive Elevator, Aileron and Rudder Deflections for an F-5A with Pitch Damper (M=0.3, h=10,000 ft.).



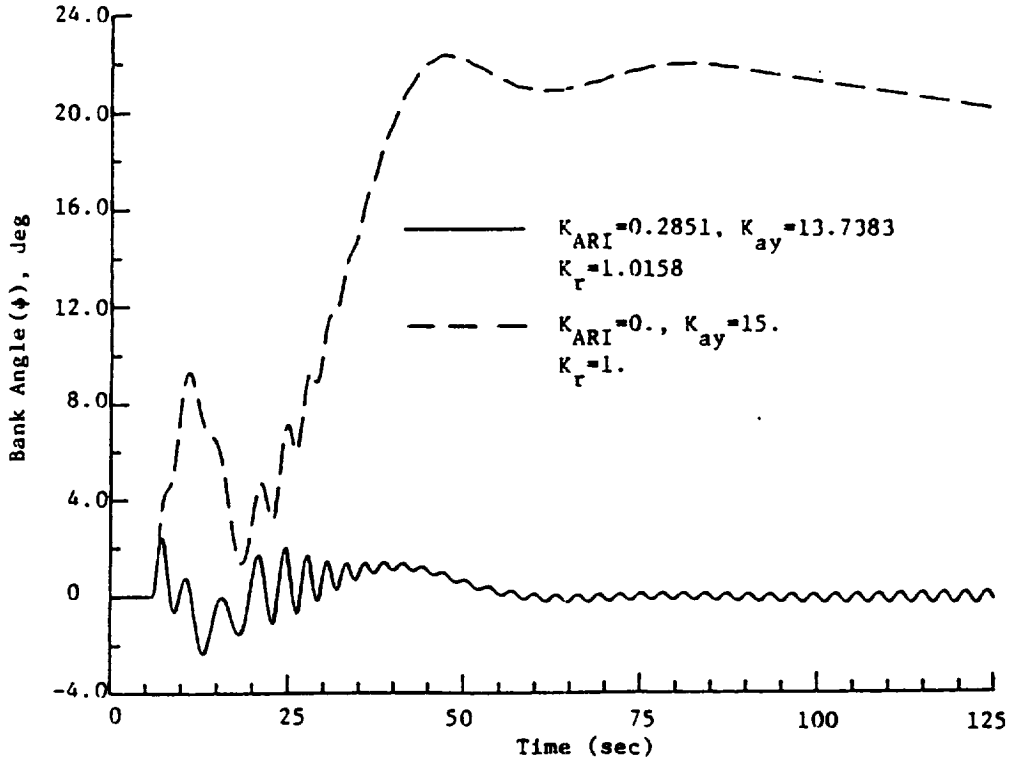


Figure 2 Effect of Control System on Yaw Divergence for an F-5A Configuration at M = 0.5 and h = 10,000 ft. in a Pull-up and Roll Maneuver

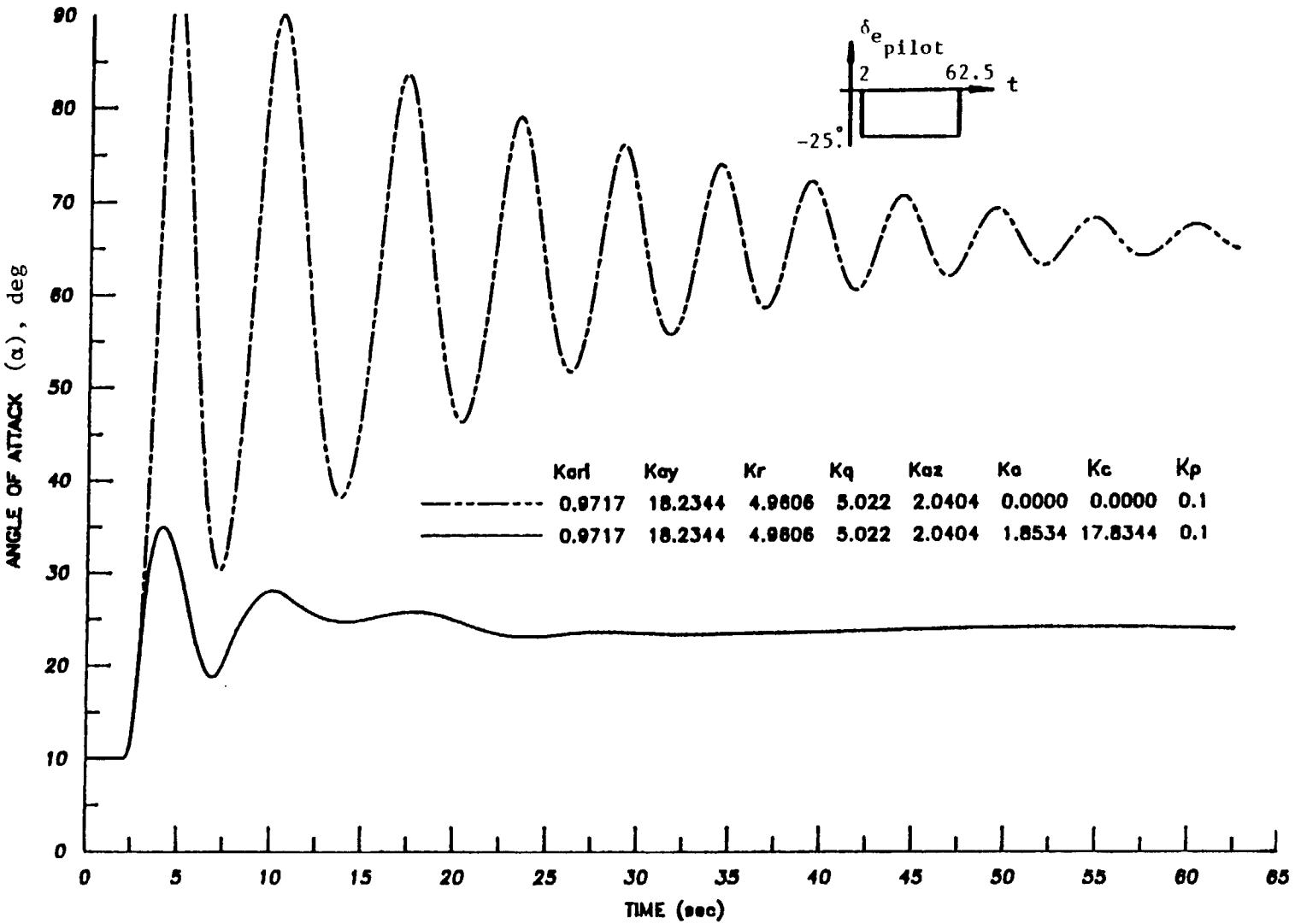


Figure 3 Control System Design to Prevent Pitch Departure of an F-16 Configuration in a Pull-up Maneuver without Roll at M = 0.5 and h = 30,000 ft.

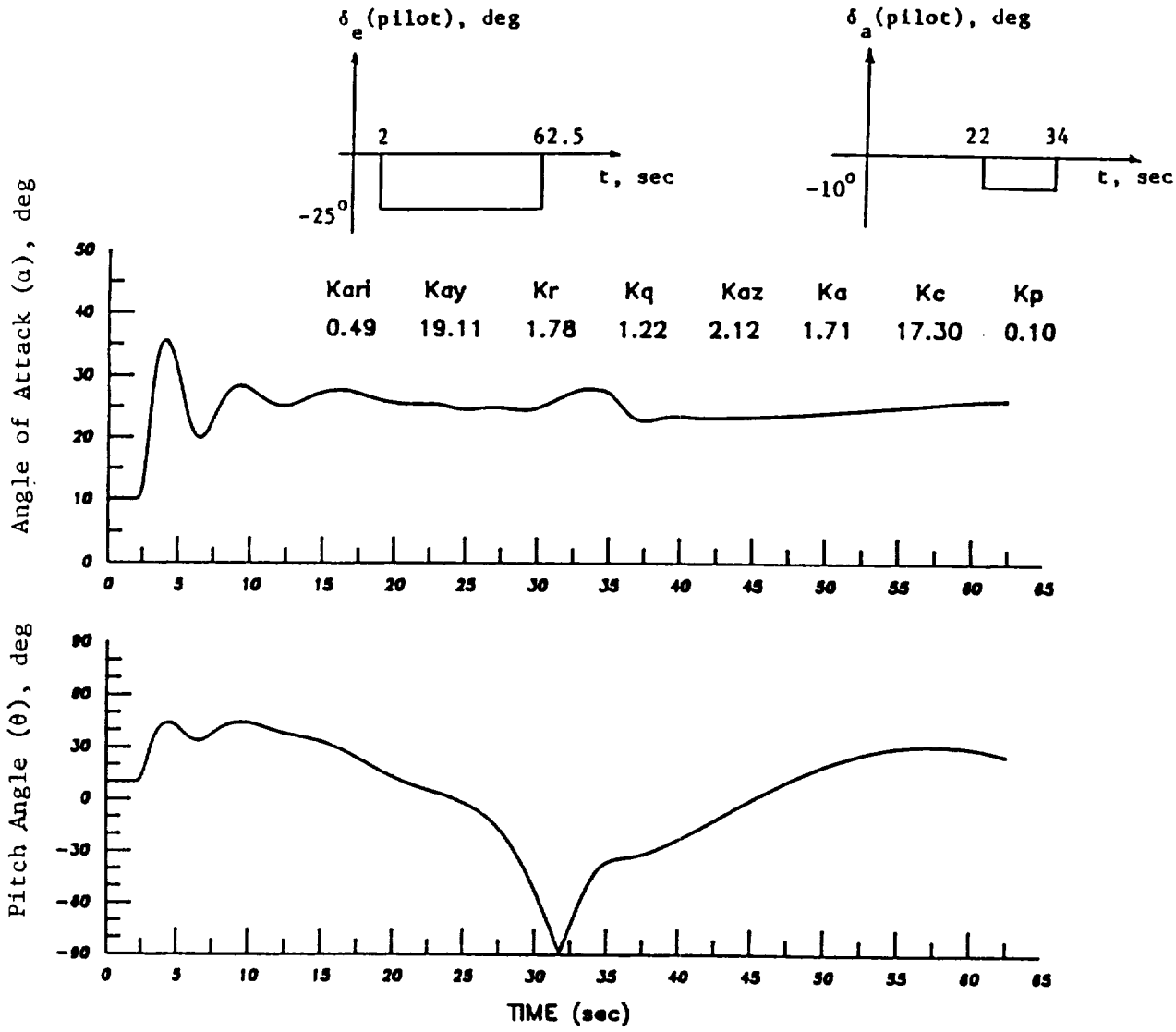


Figure 4 Control System Design to Prevent Pitch Departure of an F-16 Configuration in a Pull-up and Roll Maneuver at M = 0.5 and h=30,000 ft. Thrust = constant

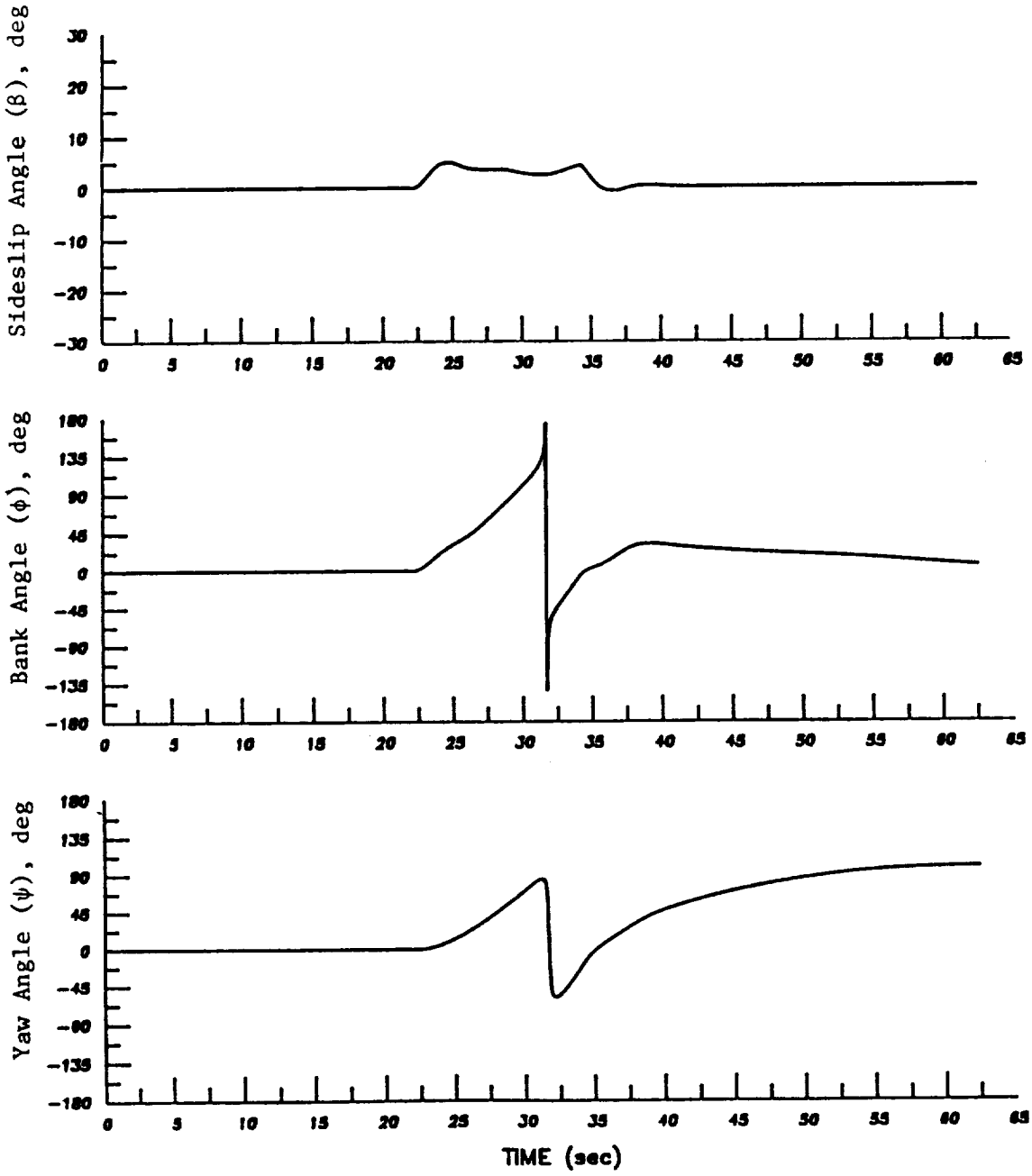


Figure 4 Concluded.

An Inverse Kinematics Algorithm For A Highly Redundant Variable-Geometry-Truss Manipulator

Frank Naccarato
Peter Hughes

University of Toronto Institute for Aerospace Studies
4925 Dufferin St., Downview, Ontario, Canada M3H 5T6

Abstract

A new class of robotic arm consists of a periodic sequence of truss substructures, each of which has several variable-length members. Such *variable-geometry-truss manipulators* (VGTMs) are inherently highly redundant and promise a significant increase in dexterity over conventional anthropomorphic manipulators. This dexterity may be exploited for both obstacle avoidance and controlled deployment in complex workspaces. The inverse kinematics problem for such unorthodox manipulators, however, becomes complex because of the large number of degrees of freedom, and conventional solutions to the inverse kinematics problem become inefficient because of the high degree of redundancy. This paper presents a solution to this problem based on a spline-like reference curve for the manipulator's shape. Such an approach has a number of advantages: (a) direct, intuitive manipulation of shape; (b) reduced calculation time; and (c) direct control over the effective degree of redundancy of the manipulator. Furthermore, although the algorithm has been developed primarily for variable-geometry-truss manipulators, it is general enough for application to a number of manipulator designs.

1 Introduction

A new class of robotic arm consists of a periodic sequence of truss substructures containing variable-length members. *Variable-geometry-truss manipulators* (VGTMs) have emerged from various designs for deployable/controllable trusses for space-based applications [10,12,16]. An example of a VGTM is illustrated in Figure 1. The manipulator is a statically determinate truss comprised of linear members hinged together at joints. Some members are actuated and can change their length; control of these lengths determines the manipulator's shape as well as the position and orientation of the end effector.

Truss manipulators promise a number of advantages over conventional anthropomorphic robot arms and space cranes, not the least of which is a significant increase in dexterity. This increased dexterity, however, has a price: a complex and unorthodox kinematic description. This paper will outline a kinematic methodology for VGTMs and set up a solution to the general

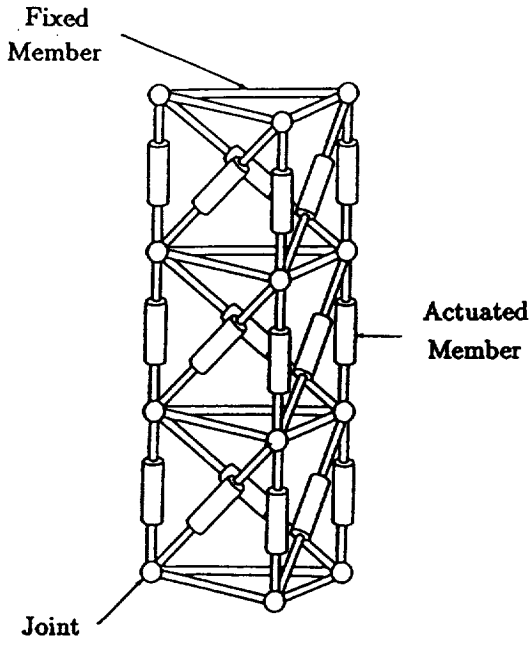


Figure 1: Example of a VGTM

inverse kinematics problem. Furthermore, a new inverse kinematics algorithm — based on spline-like reference shape curves — will be presented and compared with the more conventional solution.

2 Background to the Inverse Kinematics Problem

The purpose of any robotic arm is to manipulate an end effector along a desired trajectory; this motion depends on the combined action of the manipulator's actuators. The task of determining the correct actuator motion for a given end effector trajectory is the *inverse kinematics problem.*

2.1 Direct Kinematics

The end effector is defined by a set of n_e coordinates, $\mathbf{x}_e(t)$. This vector may contain up to six elements: three position and three orientation coordinates. The manipulator's *configuration* is defined by the vector $\mathbf{q}(t)$ containing n_q elements representing the manipulator's internal degrees of freedom. In general, we may state the *direct kinematics* relationship in the form

$$\mathbf{x}_e = \mathbf{f}(\mathbf{q}) \tag{1}$$

$$n_e \leq n_q \tag{2}$$

The inequality represents the possibility of *redundancy.*

2.2 Differential Kinematics

In general, the direct kinematics relationship (1) cannot be inverted directly to give the inverse kinematics solution. Instead, we resort to *differential kinematics* [1,15], whereby we arrive at a

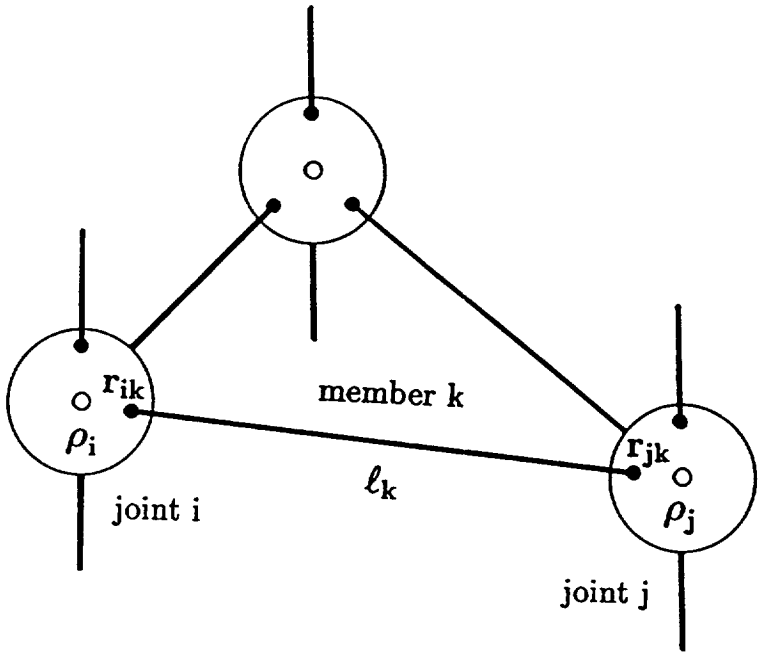


Figure 2: Geometric Representation of a VGTM

rate-linear system by expanding the time derivative of $\mathbf{x}_e(t)$:

$$\dot{\mathbf{x}}_e(t) = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}(t) \tag{3}$$

$$\mathbf{J} = \frac{\partial \mathbf{f}}{\partial \mathbf{q}^T} \tag{4}$$

The *Jacobian matrix* $\mathbf{J}$ has dimensions $n_e \times n_q$.

2.3 Inverse Kinematics Solution with Redundancy

For a redundant manipulator, the Jacobian is not square and a solution to (3) is not straightforward. A common formulation of a solution takes the form [6,7]

$$\dot{\mathbf{q}} = \mathbf{J}^+\dot{\mathbf{x}}_e + (1 - \mathbf{J}^+\mathbf{J})\dot{\mathbf{q}}_s \tag{5}$$

$$\mathbf{J}\mathbf{J}^+ = 1 \tag{6}$$

The *Moore-Penrose pseudoinverse* [13], $\mathbf{J}^+$, represents a least-squares solution to (3); alternative solutions have used weighted pseudoinverses [5]. The vector $\dot{\mathbf{q}}_s$ is an arbitrary configuration velocity that may be used to minimize a cost function, avoid obstacles or fulfil some other objective [3,8,11]. The operator $(1 - \mathbf{J}^+\mathbf{J})$ projects this velocity onto the null space of $\mathbf{J}$ so that the desired motion of the end effector is not affected.

3 Kinematic Description of VGTMs

Generalized techniques exist to develop kinematic descriptions for conventional anthropomorphic manipulators [2]. These techniques, however, are not well suited to truss manipulators. In this section, a general kinematic methodology for VGTMs will be developed.

3.1 VGTM Geometry

Figure 2 shows a geometric representation of a truss manipulator. The linear members of the truss (actuated and nonactuated) are represented by straight lines having lengths ℓ_k. Embedded in the joint mechanisms are the member *endpoints*, represented by the position vectors $\mathbf{r}_{ik}$. Associated with each joint — or truss node — is a *node coordinate vector*, $\boldsymbol{\rho}_i$; each node vector represents a fixed point within the joint mechanism that defines the joint's *position* in space. Since the truss structures are assumed to be statically determinate, we may calculate the *orientation* of each joint given the set of node vectors $[\boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \cdots, \boldsymbol{\rho}_N]$, where N is the number of joints in the truss.

3.2 Configuration Variables

For conventional robotic arms, the configuration variables are identical to the actuator variables, i.e., revolute joint angles and prismatic joint lengths. For VGTMs, however, the actuator variables — the nonfixed member lengths — are unsuitable when writing the direct kinematics; an intermediate set of variables must be used. The most appropriate choice for configuration variables are the *node coordinate vectors* $\boldsymbol{\rho}_i$, $(i = 1, 2, \ldots, N)$. These variables have the following properties that make them suitable:

1. Relationships may be found between the node vectors and useful external parameters. For instance, we may be interested in the position and orientation of a particular triangular face of the truss (where an instrument platform may be attached); these coordinates may be found if the node vectors describing the vertices of the triangle are known.

2. Since the truss is statically determinate, we may calculate the endpoint vectors $\mathbf{r}_{ik}$ given $\boldsymbol{\rho}_i$, $(i = 1, 2, \cdots, N)$; once the endpoints are known, we may determine the length of each actuator. Hence, the actuator coordinates for the truss may be arrived at through this set of configuration variables.

3.3 Kinematic Relationships

There are three kinematic relationships by which we may determine the node coordinate vectors:

Explicit External Constraints. This kinematic relationship involves node vectors that are known, explicit functions of time; these functions are defined independently of the truss configuration and hence are 'external.' If there are N_c such constrained nodes, we may group them into the vector $\boldsymbol{\rho}_c$ such that

$$\boldsymbol{\rho}_c = \text{col}[\boldsymbol{\rho}_i(t)] \tag{7}$$

$$i = 1, 2, \ldots, N_c \tag{8}$$

Typically, these form the base nodes of a VGTM. For completeness, we also group the unconstrained node vectors into $\boldsymbol{\rho}$:

$$\boldsymbol{\rho} = \text{col}[\boldsymbol{\rho}_i] \tag{9}$$

$$i = N_c + 1,\ N_c + 2, \ldots, N \tag{10}$$

Implicit External Constraints. This type of constraint describes the relationship of truss nodes to a set of external coordinates — such as the end effector coordinates $\mathbf{x}_e$ — and takes the form

$$\mathbf{f}(\rho, \rho_c) = \mathbf{x}_e(t) \tag{11}$$

where $\mathbf{f}(\rho, \rho_c)$ is a set of functions in the node coordinate vectors.

Internal Constraints. The n_l nonactuated members of a truss manipulator serve as hard constraints to the motion of the nodes. These internal constraints take the implicit form

$$(\mathbf{r}_{ik} - \mathbf{r}_{jk})^T(\mathbf{r}_{ik} - \mathbf{r}_{jk}) = \ell_k^2 \tag{12}$$

$$\mathbf{r}_{ik} = \mathbf{r}_{ik}(\rho, \rho_c) \tag{13}$$

$$k = 1, 2, \ldots, n_l \tag{14}$$

$$i, j = 1, 2, \ldots, N \tag{15}$$

The length of fixed member k is expressed in terms of its *endpoint vectors*, $\mathbf{r}_{ik}$. If the truss nodes are perfect pin joints, the endpoint vectors become identical to the node vectors.

3.4 Inverse Kinematics

A unique solution for ρ may be impossible to determine because of redundancy. To resolve this redundancy, we first find rate-linear expressions based on the constraint equations. From the implicit external constraints (11), we form

$$\mathbf{J}_u(\rho, \rho_c)\dot{\rho} + \mathbf{J}_c(\rho, \rho_c)\dot{\rho}_c = \dot{\mathbf{x}}_e \tag{16}$$

where

$$\mathbf{J}_u = \frac{\partial \mathbf{f}}{\partial \rho^T} \tag{17}$$

$$\mathbf{J}_c = \frac{\partial \mathbf{f}}{\partial \rho_c^T} \tag{18}$$

From the internal constraints we get a set of n_l equations, $(k = 1, 2, \ldots, n_l)$, that take the form

$$(\mathbf{r}_{ik} - \mathbf{r}_{jk})^T[(\mathbf{J}_{ik,u} - \mathbf{J}_{jk,u})\dot{\rho} + (\mathbf{J}_{ik,c} - \mathbf{J}_{jk,c})\dot{\rho}_c] = 0 \tag{19}$$

$$\mathbf{J}_{ik,u} = \frac{\partial \mathbf{r}_{ik}}{\partial \rho^T} \tag{20}$$

$$\mathbf{J}_{ik,c} = \frac{\partial \mathbf{r}_{ik}}{\partial \rho_c^T} \tag{21}$$

(N.B. If the nodes are ideal pin joints, the nonzero portions of these Jacobians reduce to identity matrices, and we are left with a much simpler set of expressions [18]). We may group these n_l equations into a system:

$$\mathbf{J}_g(\rho, \rho_c)\dot{\rho} = \dot{\mathbf{x}}_g(\rho, \rho_c) \tag{22}$$

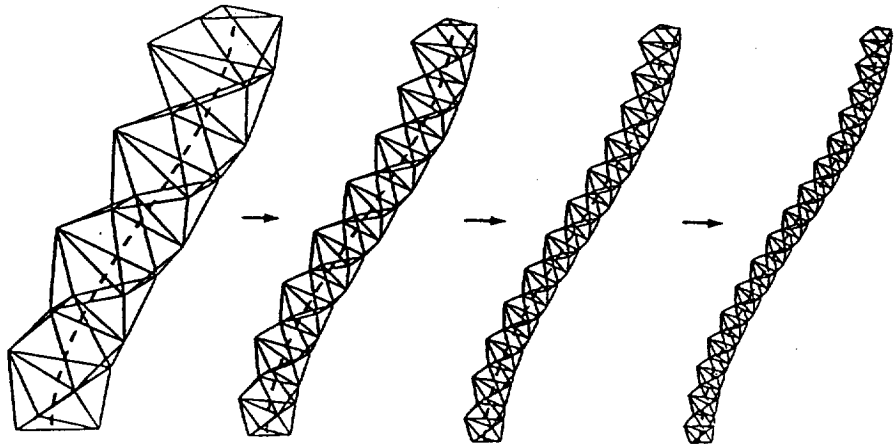


Figure 3: VGTMs Approach Continuum

where

$$\dot{\mathbf{x}}_g(\boldsymbol{\rho}, \boldsymbol{\rho}_c) = \text{col}[-(\mathbf{r}_{ik} - \mathbf{r}_{jk})^T(\mathbf{J}_{ik,c} - \mathbf{J}_{jk,c})\dot{\boldsymbol{\rho}}_c]_k \quad (23)$$

$$\mathbf{J}_g(\boldsymbol{\rho}, \boldsymbol{\rho}_c) = \text{col}[(\mathbf{r}_{ik} - \mathbf{r}_{jk})^T(\mathbf{J}_{ik,u} - \mathbf{J}_{jk,u})]_k \quad (24)$$

Combining (16) and (22), and defining

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\mathbf{x}}_e - \mathbf{J}_c\dot{\boldsymbol{\rho}}_c \\ \dot{\mathbf{x}}_g \end{bmatrix} \quad (25)$$

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_u \\ \mathbf{J}_g \end{bmatrix} \quad (26)$$

we arrive at the following system:

$$\mathbf{J}(\boldsymbol{\rho}, \boldsymbol{\rho}_c)\dot{\boldsymbol{\rho}} = \dot{\boldsymbol{x}} \quad (27)$$

The inverse kinematic solution of (27) takes the conventional form

$$\dot{\boldsymbol{\rho}} = \mathbf{J}^+\dot{\boldsymbol{x}} + (1 - \mathbf{J}^+\mathbf{J})\dot{\boldsymbol{\rho}}_s \quad (28)$$

4 Inverse Kinematics Using Reference Shape Curves

In the previous section, a solution to the inverse kinematics problem for VGTMs was found using the conventional methods outlined in §2. An alternative solution will now be presented, and a comparison will be made to the previous solution to illustrate some advantages associated with the new technique.

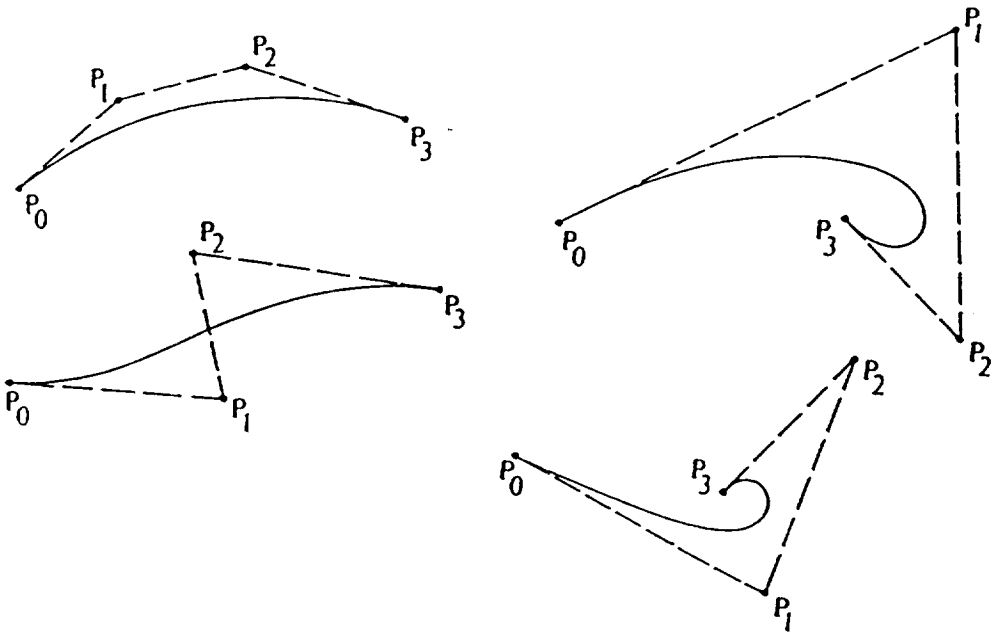


Figure 4: Space curves (Bezier)

4.1 Basic Concept

A distinctive feature of the VGTM design is its ability to assume curvilinear shapes [9]. Furthermore, as illustrated in figure 3, the shape of a truss manipulator may be made to conform *exactly* to an arbitrary space curve in the limit as the number of bays gets large. It is this serpentine property of VGTMs that has inspired an inverse kinematics technique based on the modelling of the manipulator's shape using *reference shape curves.*

The proposed solution algorithm is as follows:

1. When solving the inverse kinematics problem, *replace* the manipulator by a continuous space curve that satisfies the same external constraints.

2. Force the manipulator to *track* this new desired shape.

The reference shape curve may be thought of as the next dimensional extension of the end effector coordinates, $\mathbf{x}_e$: while the latter is defined at a point, the reference curve is a continuous coordinate *distribution* over a line. The second-order case will be a reference shape *surface* that may be used to model plate-like variable geometry trusses.

The primary reasons for pursuing such a technique are

1. A simplified, analytic expression for the manipulator's shape will help in modelling interactions with the robot's environment (e.g., collision detection, obstacle avoidance, controlled deployment).

2. Less complex kinematic relations, resulting from the simplified shape model, will improve the computational efficiency of the inverse kinematics solution.

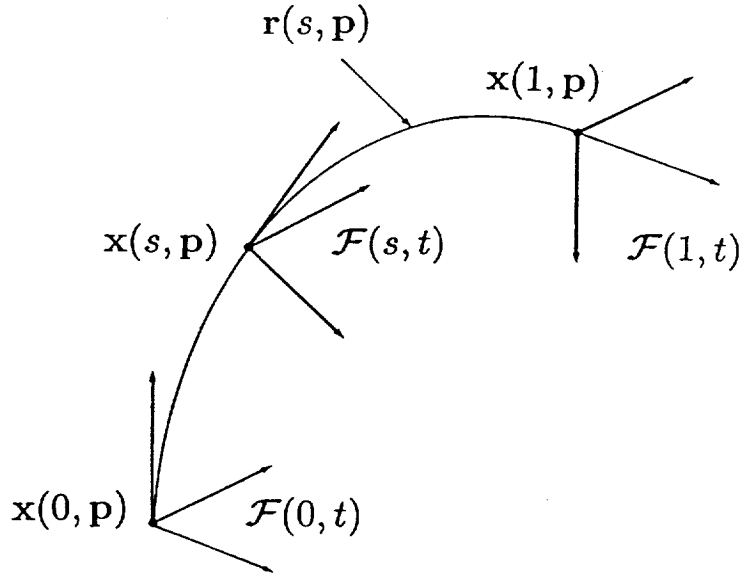


Figure 5: Coordinate Distribution Associated with Space Curve

4.2 Type of Curve

A certain class of space curve, found in computer graphics applications [17], has a number of properties that are desirable for this application. These spline-like curves — parameterized by s $(0 \leq s \leq 1)$ — have the form

$$\mathbf{r}(s,t) = \sum_{i=0}^{N} w_i(s)\mathbf{p}_i(t) \tag{29}$$

$$1 = \sum_{i=0}^{N} w_i(s) \tag{30}$$

$$\tag{31}$$

Example of such curves are *Bezier curves, B-splines and Beta-splines.* Because the *weighting functions* $w_i(s)$ are a partition of unity, a point on the curve $\mathbf{r}$ is a weighted average of the *control vertices* $\mathbf{p}_i$. Joining the control verticies sequentially forms an open *control polygon* that approximates the actual shape of the curve; closeness of fit between the control polygon and the curve depends on the form of the weighting functions. Figure 4 illustrates this relationship for a Bezier curve.

4.3 Coordinate Distribution

Associated with the curve is a distribution of position and orientation coordinates, $\mathbf{x}(s,\mathbf{p})$:

$$\mathbf{x}(s,\mathbf{p}) = \begin{bmatrix} \mathbf{r}(s,\mathbf{p}) \\ \theta(s,\mathbf{p}) \end{bmatrix} \tag{32}$$

$$\mathbf{p} = \mathrm{col}\,[\mathbf{p}_i] \tag{33}$$

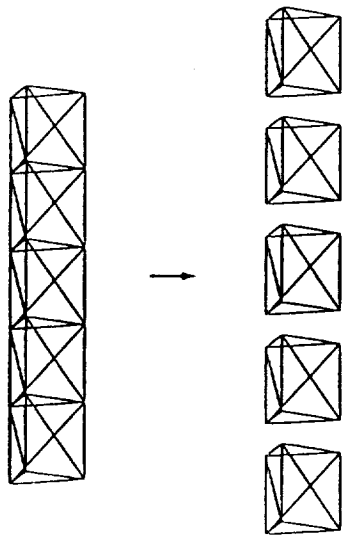


Figure 6: Partitioning of VGTM

Figure 5 shows that $\mathbf{x}(s,\mathbf{p})$ describes a frame of reference $\mathcal{F}(s,t)$ that has two properties:

1. Its origin intersects the curve at $\mathbf{r}(s,\mathbf{p})$.
2. One of its axes is tangent to the curve at the origin.

Since the tangent is a function of $\mathbf{p}$, two of the three attitude coordinates are explicit functions of the control vertices. The third attitude coordinate — representing a torsion about the tangent — may be specified independently of the control vertices, and hence depends on the parameter s alone.

4.4 Kinematics of a Curve

The relationship between the control verticies and the overall curve shape allows direct, intuitive manipulation of the curve. Hence, the control vertices $\mathbf{p}$ will be defined as the configuraton variables for the curve. The direct kinematics relation follows from the fact that the coordinate distribution, $\mathbf{x}$, is an extension of the original end effector coordinates, i.e.,

$$\mathbf{x}_e(t) = \mathbf{x}(1,t) \tag{34}$$

The rate-linear kinematic expression for any point on the curve is

$$\mathbf{J}_s(s,\mathbf{p})\,\dot{\mathbf{p}} = \dot{\mathbf{x}}(s,t) \tag{35}$$

where $\mathbf{J}_s$ — the *curve Jacobian* — is defined as

$$\mathbf{J}_s(s,\mathbf{p}) = \begin{bmatrix} \frac{\partial \mathbf{r}(s,\mathbf{p})}{\partial \mathbf{p}^T} \\ \frac{\partial \boldsymbol{\theta}(s,\mathbf{p})}{\partial \mathbf{p}^T} \end{bmatrix} \tag{36}$$

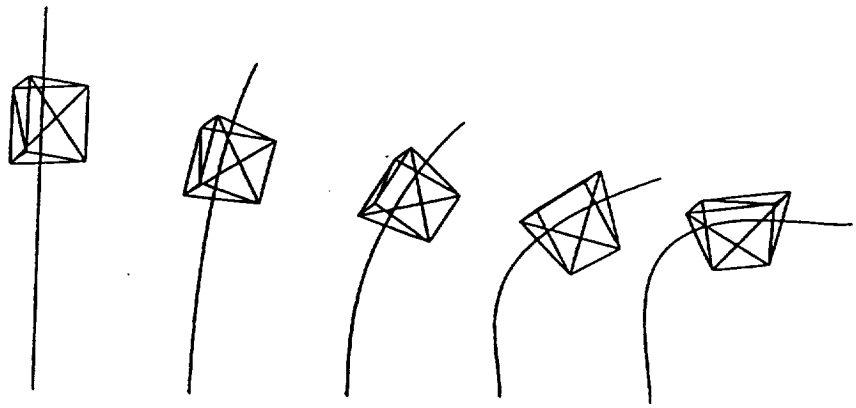


Figure 7: Tracking the Curve

To impose the implicit external constraint expressed in (34), we set

$$\mathbf{J}_s(1,\mathbf{p})\,\dot{\mathbf{p}} = \dot{\mathbf{x}}_e(t) \tag{37}$$

Following (5), the solution for the curve kinematics becomes

$$\dot{\mathbf{p}} = \mathbf{J}_s^{+}\dot{\mathbf{x}}_e + (1 - \mathbf{J}_s^{+}\mathbf{J}_s)\dot{\mathbf{p}}_s \tag{38}$$

As was the case for the general manipulator in §2 and for a VGTM in §3, the vector $\dot{\mathbf{p}}_s$ may be formulated to achieve a variety of objectives including obstacle avoidance.

4.5 Tracking the Reference Curve

The second step in the algorithm is to force the manipulator to track the reference curve. To do this, the manipulator is *partitioned* into smaller VGTMs (see figure 6); each segment may be described kinematically in the same manner as the whole VGTM. The implicit external constraints $\mathbf{x}_i(t)$ corresponding to each segment are then read off the reference curve at discrete points — $[s_0, s_1, \ldots, s_N]$ — called *curve nodes*, i.e.

$$\mathbf{x}_i(t) = \mathbf{x}(s_i, \mathbf{p}) \tag{39}$$

The VGTM segments may be solved in one of two ways:

1. *Recursively*, whereby the implicit constraints for the kth bay are given by $\mathbf{x}(s_k, \mathbf{p})$, and the explicit constraints are provided by the $(k-1)$st bay, or,
2. *In parallel*, whereby each bay is solve independantly using only the implicit information provided by the curve.

Figure 7 shows a manipulator segment tracking a curve that is executing a maneuver.

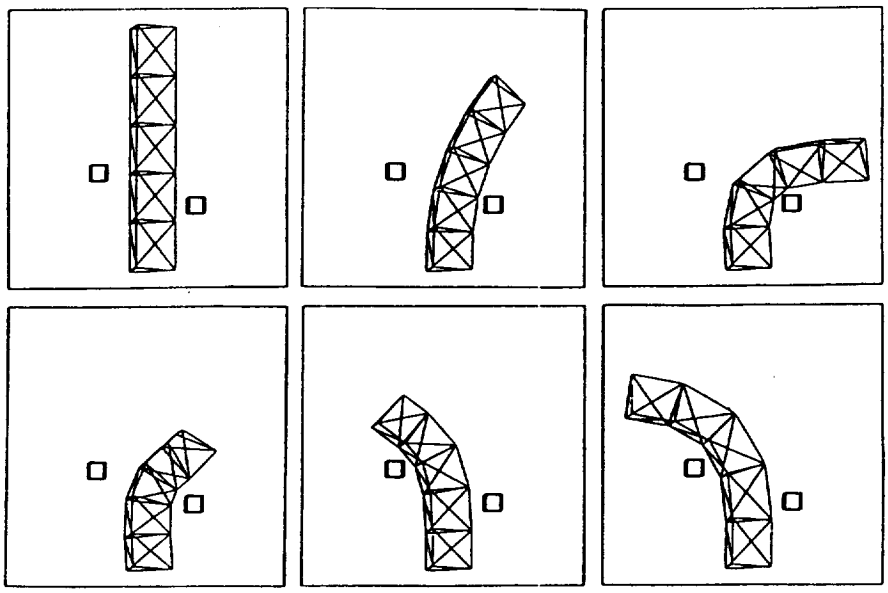


Figure 8: Example Maneuver

4.6 Example Maneuver

Figure 8 shows sample frames from a 50-step maneuver involving a five-bay VGTM. The arm is directed to avoid two cube-shaped obstacles. The inverse kinematics has been solved using the reference curve technique; clearly, the technique works adequately in that the desired end effector trajectory is tracked, and a safe clearance is maintained between the robot arm and the obstacles.

4.7 Advantages to the Reference Curve Technique

Some of the advantages of the reference curve technique are summarized below:

Improved Computational Efficiency To evaluate any improvement in computation time, a test trajectory was designed and *both* techniques — conventional and reference shape curve (recursive) — were used to solve the inverse kinematics for VGTMs of different lengths. The maneuver was partitioned into 50 time-steps, and the runs were carried out on an APOLLO^{TM} DN 4000 workstation. Figure 9 shows a comparison of the run-times for both methods; clearly, there is a marked improvement in computational efficiency when using the reference shape curve technique. This improvement may be attributed to the recursive nature of the tracking problem, as discussed in §4.5.

Parallel Structure of Problem As stated earlier, the tracking problem may be solved *in parallel* as well. This natural parallel structure makes the reference curve technique conducive to a *multiprocessing* environment [14]; in such an architecture, dedicated processors may be used to solve the inverse kinematics of the VGTM segments concurrently, thereby providing a quantum improvement in efficiency.

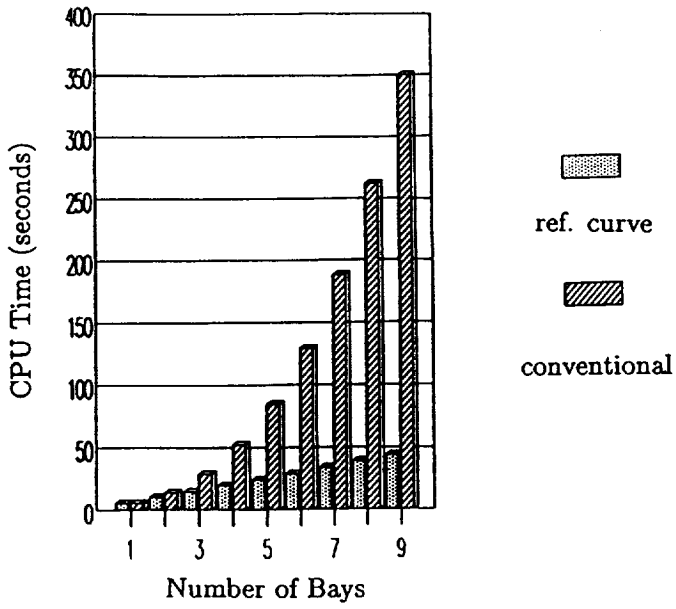


Figure 9: Run-time Comparison

Analytical Description of Manipulator Shape An analytical expression for the manipulator's shape is helpful when describing robot/environment interactions. An important part of many obstacle avoidance routines involves finding the closest point to the obstacle on the manipulator [4,5,11]. Using the shape curve equation and quickly-converging iterative solvers, we may readily solve for these critical points. An analytical description may also be incorporated into a mathematical model of the manipulator's workspace to predict collisions.

Variable Degree of Redundancy Since the global inverse kinematics problem has been shifted from the manipulator to the reference curve, we may specify the degree of redundancy *a priori* by choosing the number of control vertices used to describe the space curve. For instance, if it is known that the VGTM is to operate in a very simple, obstacle-free workspace, then only a minimum number of degrees of freedom — enough to satisfy the external constraints — need be included; conversely, a complex workspace will demand the use of more control vertices. There is a limit, of course: it will be impossible for the manipulator to track a shape curve that has more degrees of freedom than itself. Yet, this flexiblity in choosing the degree of redundancy may be used to reduce computational overhead.

4.8 Generality

While developed for application to truss manipulators, the reference curve method is general enough to find use as a redundancy resolution technique with more conventional robotic arms. The first step in the algorithm remains the same since the kinematics of the reference curve is independant of the manipulator. The second step — the tracking problem — may be applied to any robotic arm that can be easily segmented into smaller manipulators.

5 Conclusions

A generalized kinematic description has been presented for variable-geometry-truss manipulators. From this description, a solution to the inverse kinematics problem was found by conventional means. An alternative technique — based on reference shape curves — was developed and applied to VGTMs in the hopes of improving the efficiency of the straightforward approach, as well to inject some geometric insight in the description of the manipulator's shape. The new technique succeeded in solving the inverse kinematics problem with a significantly decreased computation time. Further advantages were noted:

- Technique is attractive to obstacle avoidance and collision detection applications;
- Problem structure is applicable to parallel processing;
- Variable degree of redundancy
- Generality

The preliminary success of this new method is encouraging for the development of real-time-control robotic facilities based on truss manipulators.

6 Acknowledgements

The authors would like to acknowledge research support from The Natural Sciences and Engineering Research Council of Canada, and would like to thank the following for their help: V. Pugliese, members of the Space Dynamics and Control Group, and Dynacon Enterprises Ltd.

7 References

1. CRAIG, J. *Introduction to Robotics, Mechanics and Control*, Addison-Wesley, 1986.
2. GOLDENBERG, A. ET AL. "A Complete Generalized Solution to the Inverse Kinematics of Robots" *IEEE Journal of Robotics and Automation*, Vol. Ra, No.1, March 1985.
3. HOLLERBACH, J.M. & SUH, K.C. "Redundancy Resolution of Manipulators Through Torque Optimization," *IEEE Journal of Robotics and Automation*, Vol. RA-3, No. 4, August 1987.
4. KHATIB, O. "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *International Journal of Robotics Research*, Vol. 5, No. 1, Spring 1986.
5. KIRCANSKI, M. & VUKOBRATOVIC, M. "Contribution to Control of Redundant Robotic Manipulators in an Environment with Obstacles," *Intl. Journal of Robotics Research*, Vol.5, No. 4, Winter 1986.
6. KLEIN, C.A. & HUANG, C.-H. "Review of Pseudoinverse Control for Use With Kinematically Redundant Manipulators," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-13, No. 3, March/April 1983.
7. LIEGEOIS, A. "Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanics," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-7, No. 12, Dec. 1977.

8. MACIEJEWSKI, A.A. & KLEIN, C.A. "Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments," *International Journal of Robotics Research*, Vol. 4, No. 3, Fall 1985.

9. MIURA, K. & FURUYA, H. "Variable Geometry Truss and Its Application to Deployable Truss and Space Crane Arms," *35th Congress of the International Astronautical Federation*, Lausanne, Switzerland, Oct. 7-13, 1984 [IAF-84-394].

10. MIURA, K. & FURUYA, H. "An Adaptive Structure Concept for Future Space Applications," *36th Congress of the International Astronautical Federation*, Stockholm, Sweden, Oct. 7-12, 1985 [IAF-85-211].

11. NAKAMURA, Y. & HANAFUSA, H. "Optimal Redundancy Control of Robot Manipulators," *International Journal of Robotics Research*, Vol. 6, No. 1, Spring 1987.

12. NAYFEH, A.H. "Kinematics of Foldable Discrete Space Cranes," *NASA CR176360*, 1985.

13. NOBLE, B. & DANIEL, J.W. *Applied Linear Algebra*, 2nd Ed., Prentice-Hall, New Jersey 1977.

14. OSTERHAUG, A. *Guide to Parallel Programming*, Sequent Computer Systems, Oregon, 1987.

15. PAUL, R. *Robot Manipulators*, MIT Press, Cambridge, 1981.

16. RHODES, M.D. & MIKULAS, M.M. "Deployable Controllable Geometry Truss Beam," *NASA TM-86366*, 1985.

17. ROGERS, D.F. & ADAMS, J.A. *Mathematical Elements for Computer Graphics*, McGraw-Hill, New York, 1976.

18. SINCARSIN, W.G. & HUGHES, P.C. *Trussarm: Candidate Geometries*, Dynacon Enterprises Ltd., Toronto, 1987.

DETERMINATION OF JOINT DRIVES FOR STABLE END-EFFECTOR MOTION IN FLEXIBLE ROBOTIC SYSTEMS

SITKI KEMAL IDER
Assistant Professor
Department of Mechanical Engineering
Middle East Technical University, Ankara, Turkey

ABSTRACT

The prescribed tasks in high speed robotic systems are severely deteriorated because of their manipulator dynamic deflections. On the other hand conventional dynamic modeling techniques fail to reveal appopriate control forces in flexible systems. In this paper the conventional dynamic equations of motion for systems subject to kinematical constraints are modified by a new concept of control force representation. The directions of the control forces are selected such that they correspond to the joint degrees of freedom. Then the joint control forces and torques that yield unperturbed prescribed motions are solved simultaneously with the system motion. A flexible manipulator is presented to illustrate the methods proposed.

1. INTRODUCTION

The operation of high speed robots is severely limited by their manipulator dynamic deflection. The vibrations deteriorate the accuracies of the prescribed tasks assigned to certain points and significantly reduce the robot arm production rate. Hence determination of the appropriate control forces and torques at the joints that yield stable prescribed motions is an important control problem.

In this paper geometrical constraints represent geometrical restrictions such as closed loops and physical guides. On the other hand kinematical constraints represent prescribed desired paths or prescribed motions of certain points or bodies. Such prescribed motions are to be realized by control forces applied by the actuators in the system which are usually placed at the joints.

In the conventional approach, constraints in the system are modeled by constraint reaction forces whose directions are perpendicular to the constraint surfaces. (See Yoo and Haug [1], Shabana [2], Kamman and Huston [3], Hemami and Weimer [4], Nikravesh [5].) Using conventional methods, when the prescribed motions are treated as constraint equations and embedded into the governing equations of motion, the corresponding generalized constraint reaction forces can be determined. However these forces cannot be utilized as physically possible control forces due to the

presence of components that correspond to the elastic coordinates. For this reason previous solution procedures involved feedback and adaptive control algorithms which in turn increase the complexity of the problem considerably.

In this paper the kinematical constraints are modeled by general direction control forces. The modified equations of motion for flexible multibody and robotic systems subject to geometrical and kinematical constraints are developed. The directions of the control forces are selected such that they correspond to the joint degrees of freedom. By this way joint control forces and torques that achieve the unperturbed prescribed motions are solved simultaneously with the corresponding system motion. The modeling of flexible multibody systems in joint space based on finite element method and component mode synthesis, as developed in references Ider [6], Ider and Amirouche [7] is also outlined. In the equations of motion all nonlinear interactions between the rigid body and elastic coordinates are automatically incorporated.

This paper is divided into seven sections. The first section provided an introduction. In the second section kinematics and constraint equations in flexible multibody systems are outlined. The conventional equations of motion for constrained systems are presented in the third section. In section four the problems with the conventional approach are discussed. In the fifth section the modified equations of motion with general direction control forces are developed. Sixth section presents the simulations of a flexible manipulator by the proposed method. Conclusions form the last section.

2. FLEXIBLE MULTIBODY KINEMATICS AND CONSTRAINT EQUATIONS

In a multibody system each joint connection can be described by a total of six degrees of freedom. The constrained joint coordinates are eliminated in the analysis, hence all possible joint types are allowed. The system may contain closed loops and any selected points may have prescribed motions. First the recursive dynamical equations are developed for a tree configuration which is obtained by cutting the closed loops open (using any arbitrary joint in the loop). Closed loops and prescribed motions are then imposed as a set of constraint equations.

In Figure 1, a typical deformable body B_k and its lower connecting body B_j are shown. The joint between B_k and B_j is the lower joint of B_k and is defined by points Q_k and Q_k^* and axis frames n^k and n^{k*} fixed at these points. The elastic deformations are modeled by finite element method with respect to a body reference axis frame denoted by N^k. N^k, in general, is not fixed to a point on the body. It follows the rigid body motion of B_k in a manner consistent with the specified boundary conditions [6,7].

The position of the system can be described by the relative joint coordinates of each body and the modal coordinates of the flexible bodies. Translation of n^k with respect to n^{k*} is denoted by vector z^k. For the relative rigid body rotation degrees of freedom, successive Euler angles in transforming n^k to n^{k*} can be used. The modal coordinates η_j^k, $j=1,..,m^k$

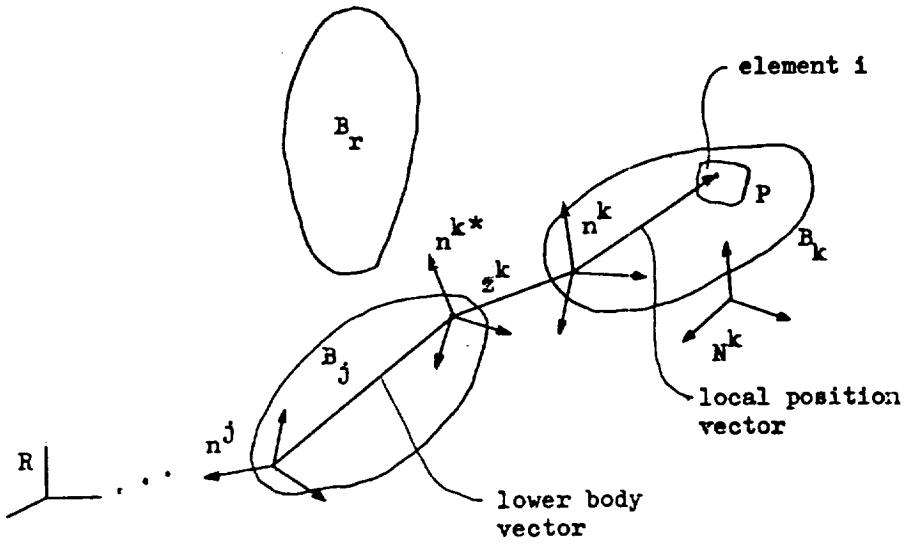


Figure 1. A multibody system

of each body B_k represent the normal modes of deformation obtained by component mode synthesis and m^k is the number of the modes considered.

Let vector $\hat{w}^k$ represent the angular velocity of n^k with respect to n^{k*}. Then the generalized speeds of the system can be conveniently selected as the relative angular velocity components, the relative translational velocity components and the modal coordinate derivatives. The vector of the system generalized speeds y can be defined as

$$y = [\hat{w}^T, \dot{z}^T, \dot{\eta}^T]^T \tag{1}$$

where

$$\hat{w} = [\hat{w}_1^1, \hat{w}_2^1, \hat{w}_3^1, \ldots, \hat{w}_1^N, \hat{w}_2^N, \hat{w}_3^N]^T \tag{2}$$

$$\dot{z} = [\dot{z}_1^1, \dot{z}_2^1, \dot{z}_3^1, \ldots, \dot{z}_1^N, \dot{z}_2^N, \dot{z}_3^N]^T \tag{3}$$

and

$$\dot{\eta} = [\dot{\eta}_1^1, \ldots, \dot{\eta}_{m^1}^1, \ldots, \dot{\eta}_1^N, \ldots, \dot{\eta}_{m^N}^N]^T \tag{4}$$

For the dynamical equations we need the velocity, in fixed frame R, of an arbitrary point P in finite element i of body B_k. To this end, the angular velocity of N^k in R, w^k, is obtained by summing successive relative angular velocites and can be compactly expressed as

$$w^k = \nu^k \hat{w} + \mu^k \dot{\eta} \tag{5}$$

where ν^k and μ^k are the partial angular velocity matrices composed of the coeffficients of the generalized speeds $\hat{w}$ and $\dot{\eta}$, respectively.

It can be shown [6,7] that the velocity of point P in R could be written as

$$v^{ki} = a^{ki} \hat{w} + b^k \dot{z} + c^{ki} \dot{\eta} \tag{6}$$

where a^{ki}, b^k and c^{ki} are the partial velocity arrays associated with $\hat{w}$, $\dot{z}$ and $\dot{\eta}$, and are functions of displacements only.

Depending on each joint type, the constrained joint coordinates are then eliminated in the generalized speed vectors $\hat{w}$ and $\dot{z}$ (equations (2), (3)). When the corresponding columns of the partial velocity matrices are eliminated, ν^k and a^{ki} are $3xn_1$ matrices, and b^k is a $3xn_2$ matrix, where n_1 is the total number of the free joint rotation degrees of freedom and n_2 is the total number of the free joint translation degrees of freedom. The remaining arrays μ^k and c^{ki} are $3xm$ ($m=m^1+...+m^N$).

Let the tree structure have n degrees of freedom ($n=n_1 + n_2 + m$), and let the total number of closed loop and prescribed motion type of constraints be c. Then the system's degrees of freedom reduce to n-c.

The constraint equations can be generated using the partial velocity matrices. For example, if a point say A in B_r has a prescribed motion, and the prescribed velocity vector of that point is given by g(t) with respect to R, then denoting the local undeformed vector from O_r to A by s^r, the resulting three constraint equations are

$$a^{ri} \hat{w} + b^r \dot{z} + c^{ri} \dot{\eta} = g \tag{7}$$

where a^{ri} and c^{ri} correspond to s^r.

Similarly if the refence axis frame of B_r has a prescribed angular velocity h(t), we have

$$\nu^r \hat{w} + \mu^r \dot{\eta} = h \tag{8}$$

For a closed loop type of constraint, let B_r and B_s connect with each other to form a closed loop in 3-D. Differentiation of the position vector equation expressing loop closure leads to the three velocity level constraint equations,

$$(a^{ri}-a^{si}) \hat{w} + (b^r-b^s) \dot{z} + (c^{ri}-c^{si}) \dot{\eta} = 0 \tag{9}$$

The holonomic and nonholonomic constraint equations can be compactly written as

$$B\,y = g \tag{10}$$

where B is a cxn constraint matrix and g contains prescribed velocities.

3. CONSTRAINT REACTION FORCES AND EQUATIONS OF MOTION

Kane's equations for the constrained system can be written as

$$F + F^* + S + F^c = 0 \tag{11}$$

where F, F^*, S and F^c are respectively the vectors of generalized external, inertia, stiffness and constraint forces.

The generalized inertia forces F^* can be written in the following form,

$$F^* = M\,\dot{y} + Q \tag{12}$$

where individual submatrices of M and Q can be expressed in terms of the partial velocity matrices [7] as,

$$M = \sum_k \sum_i \int_{V_{ki}} \begin{bmatrix} a^{ki^T} a^{ki} & & \text{sym.} \\ a^{ki^T} b^k & b^{k^T} b^k & \\ a^{ki^T} c^{ki} & b^{k^T} c^{ki} & c^{ki^T} c^{ki} \end{bmatrix} \rho\, dV \tag{13}$$

and

$$Q = \sum_k \sum_i \int_{V_{ki}} \begin{bmatrix} a^{ki^T} (a^{ki}\,\hat{w} + b^k\,\dot{z} + c^{ki}\,\dot{\eta}) \\ a^{ki^T} (a^{ki}\,\hat{w} + b^k\,\dot{z} + c^{ki}\,\dot{\eta}) \\ a^{ki^T} (a^{ki}\,\hat{w} + b^k\,\dot{z} + c^{ki}\,\dot{\eta}) \end{bmatrix} \rho\, dV \tag{14}$$

The stiffness vector S is obtained from the structural and geometrical stiffness matrices of each body expressed in modal coordinates.

The generalized constraint forces F^c can be expressed as

$$F^c = B^T \lambda \tag{15}$$

where λ is the vector of undetermined multipliers. Since the rows of B are the partial velocity vectors, λ_i, i=1,...,c represent the constraint reaction forces generated at the application of the constraints. A row of

B can also be viewed as the direction of that constraint in the generalized space.

Substitution of equations (12) and (15) into eq. (11) leads to

$$M\,y + S + Q + B^T \lambda = F \tag{16}$$

Our purpose is to find the accelerations for numerical integration. To this end the constraint equations in the acceleration level are

$$B\,y = \dot{g} - \dot{B}\,y \tag{17}$$

Equations (16) and (17) constitute n+c equations from which the accelerations and the undetermined multipliers can be obtained.

The multipliers could be eliminated for computational efficiency. To this end, let C denote a nx(n-c) matrix which is orthogonal complement to B [8]. Premultiplying eq. (16) by C^T, and combining the resulting equation with eq. (17), we obtain the augmented equations for the constrained system as below.

$$\begin{bmatrix} C^T M \\ \hline B \end{bmatrix} \dot{y} = \begin{bmatrix} C^T(F-S-Q) \\ \hline \dot{g}-\dot{B}y \end{bmatrix} \tag{18}$$

4. PROBLEMS WITH THE CONVENTIONAL APPROACH

In the conventional approach the constraints in the system are modeled by constraint reaction forces which are perpendicular to constraint surfaces. They represent the reactions of the environment. However kinematical constraints represent desired motions and are meant to be realized by internal control forces. Kinematical constraints are particularly important in robotic systems where certain points are assigned specific tasks that should be realized by joint actuators.

Let c_1 of the constraints in the system be geometric and the remaining c_2 (c_2=c-c_1) be kinematical. The matrix of the constraint force directions B and the vector of constraint force magnitudes λ can be partitioned such that

$$B = [B^{G^T} \quad B^{K^T}]^T \tag{19}$$

and

$$\lambda = [\lambda^{G^T} \quad \lambda^{K^T}] \tag{20}$$

where the dimensions of B^G, B^K, λ^G and λ^K are $c_1$xn, $c_2$xn, c_1 and c_2 respectively.

Then eq. (16) can be written in the following form

$$M\dot{y} + Q + S + F^G + F^K = F \tag{21}$$

where the generalized constraint forces F^G and F^K corresponding respectively to geometrical and kinematical constraints are

$$F^G = B^{G^T} \lambda^G \tag{22}$$

and

$$F^K = B^{K^T} \lambda^K \tag{23}$$

The constrained system as given by eq. (18) could be simulated to determine the generalized constraint forces. In view of eq. (21), then control forces numerically equal to F^K would yield the same motion of the system ensuring the realization of the desired motions.

However, in flexible systems F^K contains components that correspond to the elastic coordinates in addition to the components that correspond to the joint coordinates. While the latter can be applied by the joint actuators as control forces, the former cannot be produced by a physical means as control forces. That F^K has components in the direction of the elastic coordinates is apparent from eqs. (7) and (8) where the coefficients of the generalized speeds form the vectors of B^K in eq. (23). Hence, with the conventional approach it is not possible to design a set of control forces that can achieve unperturbed prescribed motions in flexible robotic and multibody systems.

5. CONTROL FORCES FOR KINEMATICAL CONSTRAINTS AND MODIFIED EQUATION OF MOTION

Since kinematical constraints are to be realized by control forces in the system, general direction control forces are introduced to the equations of motion, so that

$$M\dot{y} + Q + S + B^G \lambda^{G^T} + B^{K^T} \lambda^K + A^T \mu = F \tag{24}$$

where A is a $c_2 x n$ matrix of control force directions and μ is a c_2 dimensional vector of control force magnitudes. Let the control force directions are selected such that the constraint reaction forces corresponding to the kinematical constraints become zero. Then eq. (24) can be written as follows, as shown in accompanying paper (Ider [9]).

$$M\dot{y} + Q + S + Z^T \nu = F \tag{25}$$

where

$$Z^T = [B^{G^T} \quad A^T] \tag{26}$$

and

$$\nu^T = [\lambda^{G^T} \quad \mu^T] \tag{27}$$

The directions A need to be selected from physical considerations and then equations (25) and (17) can be solved together to compute the control force magnitudes and the corresponding generalized accelerations. A should be chosen such that rank of Z is c, so that c_2 kinematical conditions could be controlled.

Let $\bar{C}$ be a cx(n-c) matrix orthogonal complement to Z. Premultiplying eq. (25) by $\bar{C}^T$ and augmenting with eq. (17), we obtain reduced equations

$$\begin{bmatrix} \bar{C}^T M \\ \hline B \end{bmatrix} \dot{y} = \begin{bmatrix} \bar{C}^T(F-S-Q) \\ \hline \dot{g}-\dot{B}y \end{bmatrix} \tag{28}$$

The selected control force directions can realize the prescribed motions if and only if the augmented mass matrix in eq. (28) is non singular. Hence singularity of the augmented mass matrix represents a condition to test the solution. In other words, the directions A should be such that the vector space spanned by the rows of B and the vector space spanned by the rows of $\bar{C}^T$ are nonintersecting [9].

It has been observed that for flexible robotic systems if the control forces are selected in the directions along the corresponding joint degrees of freedom the augmented mass matrix becomes full rank and hence it is possible to realize the kinematical constraints by actuators at the joints. This will be illustrated by the simulations of a flexible manipulator in the next section.

6. SIMULATIONS OF A FLEXIBLE MANIPULATOR

In the planar manipulator shown in Figure 2, link 2 is a flexible link, while link 1 is treated rigid. The data used for link 1 are L_1=1m, m_1=30kg and I_1=10 kg.m^2. Link 2 is modeled by beam elements with deformation displacement and rotation nodal coordinates [10], and L_2=2.7m, a=1.8m, m_2=15kg, E=68.95x10^9 N/m^2 and area A=0.0005 m^2. The longitudinal deformation is neglected due to the axial stiffness and the transverse deflection is described by the first two modes since higher modes were observed to be negligible. Therefore the generalized coordinates of the system are θ_1, θ_2, and modal coordinates η_1 and η_2. The generalized speed vector

$$y = [\dot{\theta}_1, \quad \dot{\theta}_2, \quad \dot{\eta}_1, \quad \dot{\eta}_2]^T.$$

Initially the system is at rest, and θ_1=80° and θ_2=-160°. Point A on link 2 is required to deploy from the given position 1.5m horizontally. The prescribed motion of point A is given as

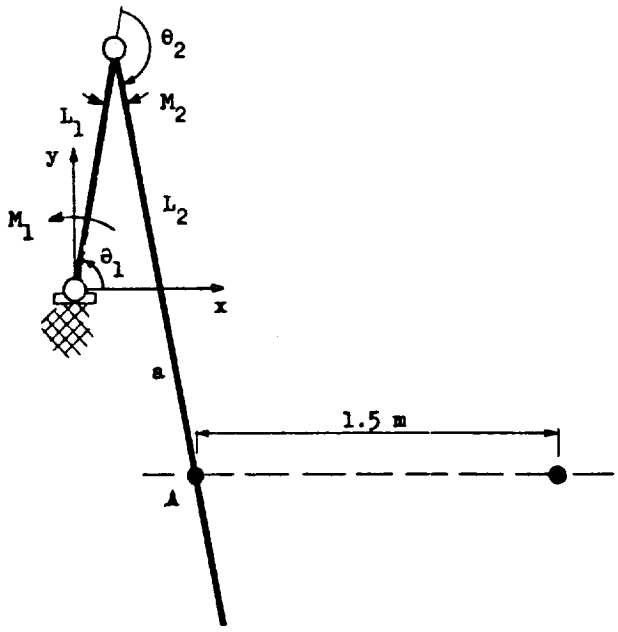


Figure 2. Flexible manipulator

$$x_A = 1.5\,\frac{1}{T}\left(t - \frac{T}{2\pi}\sin\frac{2\pi t}{T}\right) + 0.4862$$
$$y_A = -0.7878 \tag{29}$$

The constraint equations in the system can be expressed as

$$L_1c_1 + L_2c_{12} - s_{12}(\phi_1\eta_1 + \phi_2\eta_2) = x_A$$
$$L_1s_1 + L_2s_{12} + c_{12}(\phi_1\eta_1 + \phi_2\eta_2) = y_A \tag{30}$$

where ϕ_1 and ϕ_2 are the values that correspond to the location of point A in the first and second eigenvectors respectively. $c_1=\cos\theta_1$, $c_{12}=\cos(\theta_1+\theta_2)$, $s_1=\sin\theta_1$ and $s_{12}=\sin(\theta_1+\theta_2)$.
At the acceleration level the constraints are given by eq. (17) where B and g are

$$B = \begin{bmatrix} L_1s_1+L_2s_{12}+c_{12}(\phi_1\eta_1+\phi_2\eta_2) & L_2s_{12}+c_{12}(\phi_1\eta_1+\phi_2\eta_2) & \phi_1s_{12} & \phi_2s_{12} \\ L_1c_1+L_2c_{12}-s_{12}(\phi_1\eta_1+\phi_2\eta_2) & L_2c_{12}-s_{12}(\phi_1\eta_1+\phi_2\eta_2) & \phi_1c_{12} & \phi_2c_{12} \end{bmatrix}$$

and

$$g = [\dot{x}_A\ ,\ 0]^T. \tag{31}$$

First the system is simulated using the conventional method. Notice that both constraints in the system are kinematical. Our objective is to determine joint moments M_1 and M_2 that would produce unperturbed desired motion of point A. The generalized constraint forces in eq. (15) are

$$\begin{bmatrix} F_1^c \\ F_2^c \\ F_3^c \\ F_4^c \end{bmatrix} = \begin{bmatrix} \lambda_1\{L_1 s_1 + L_2 s_{12} + c_{12}(\Phi_1\eta_1 + \Phi_2\eta_2)]\} + \lambda_2\{L_1 c_1 + L_2 c_{12} - s_{12}(\Phi_1\eta_1 + \Phi_2\eta_2)\} \\ \lambda_1\{L_2 s_{12} + c_{12}(\Phi_1\eta_1 + \Phi_2\eta_2)\} + \lambda_2\{L_2 c_{12} - s_{12}(\Phi_1\eta_1 + \Phi_2\eta_2)\} \\ \lambda_1 \Phi_1 s_{12} + \lambda_2 \Phi_1 c_{12} \\ \lambda_1 \Phi_2 s_{12} + \lambda_2 \Phi_2 c_{12} \end{bmatrix} \quad (32)$$

The system is simulated for the deployment motion period T=1sec. The generalized constraint forces obtained are plotted in Figure 3. Notice that if one considers F_1^c and F_2^c as joint control moments, F_3^c and F_4^c will be left unaccounted. They cannot be converted to any set of physically applicable control forces or moments, and a simulation only with F_1 and F_2 as control moments M_1 and M_2 produces perturbations for point A.

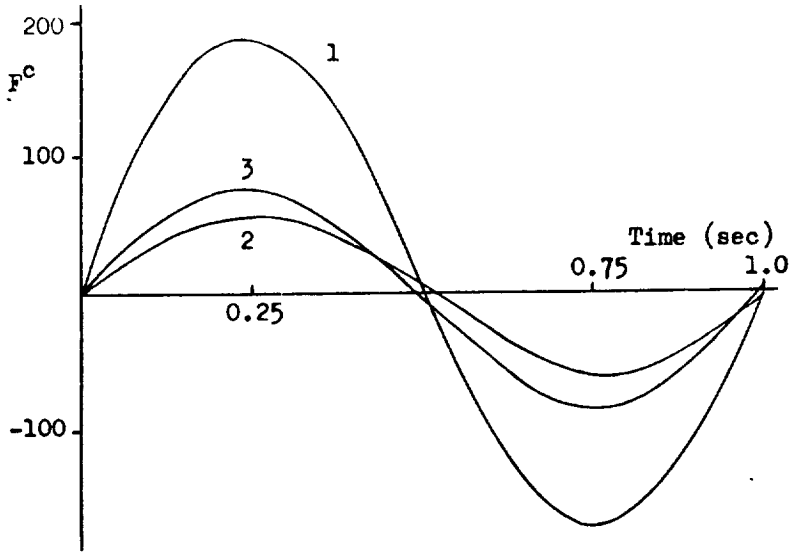


Figure 3. Generalized constraint forces using conventional method: 1. F_1^c , 2. F_2^c , 3. F_3^c , F_4^c

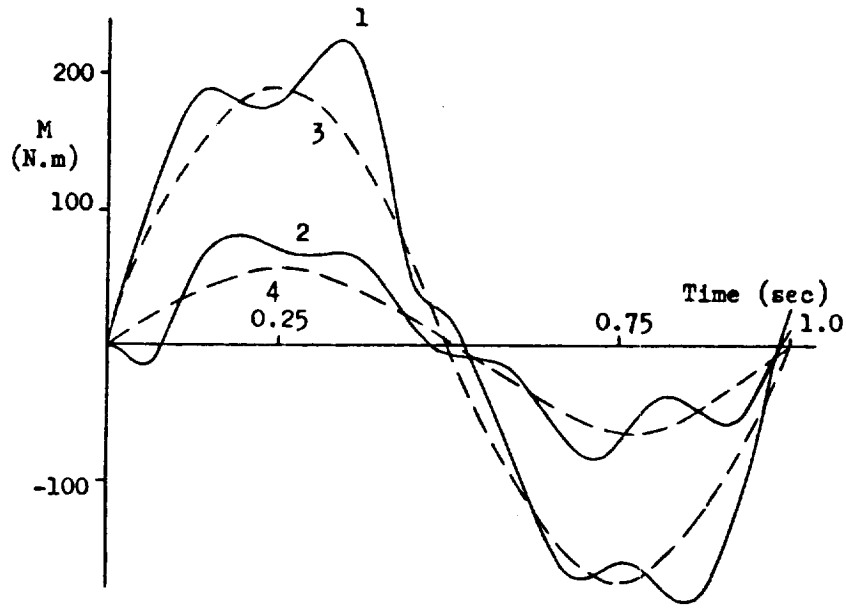


Figure 4. Joint moments for unperturbed motion of A.
Flexible system: 1. M_1 , 2. M_2
Rigid system: 3. M_1 , 4. M_2

The system is then resimulated by the control force approach presented in this paper. The control force directions are selected such that they correspond to the joint coordinates θ_1 and θ_2, i.e.

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{33}$$

The control forces $Z^T v$ become

$$Z^T v = v_1 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + v_2 \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ 0 \\ 0 \end{bmatrix} \tag{34}$$

This means that the required joint moments for unperturbed motion of point A are $M_1 = v_1$ and $M_2 = v_2$.

With the above control force directions, the augmented mass matrix was observed to be full rank, i.e. singularity (or near singularity) did not occur, conforming with physical expectations. The joint control moments M_1 and M_2 that produce unperturbed motion of point A are plotted in Figure 4.

For comparison, the system is resimulated with both bodies considered rigid, and the joint moments corresponding to the rigid system are also shown in Figure 4. The difference in the control moments for the flexible system accounts for the effects of the elastic deformations.

7. CONCLUSIONS

This paper presented a general procedure to determine the joint control forces and torques in flexible robotic systems, that realize prescribed motions in an unperturbed manner. The method is based on a new approach for modeling kinematical constraints by general direction control forces. The control forces have been selected along the directions of the joint degrees of freedom in the generalized space, and the control force magnitudes are solved simultaneously with the corresponding system motion.

It has been shown that with the conventional approach of perpendicular constraint forces a solution to the problem cannot be obtained.

In the analysis the flexible bodies have been modeled by finite element method and all interactions between the rigid and elastic motion have been included. By the procedures presented in this paper the body flexibilities can be controlled by applying forces and torques at the joints.

REFERENCES

1. Yoo, W. S., Haug, E. J., 1986, "Dynamics of Articulated Structures. Part I. Theory", Journal of Structural Mechanics, Vol. 14, pp. 105-126.

2. Shabana, A. A., 1985, "Automated Analysis of Constrained Systems of Rigid and Flexible Bodies", ASME Journal of Vibrations, Acoustics, Stress and Reliability in Design, Vol. 107, pp. 431-439.

3. Kamman, J. W., Huston, R. L., 1984, "Dynamics of Constrained Multibody Systems", ASME Journal of Applied Mechanics, Vol. 51, No. 4, pp. 899-903.

4. Hemami, H., Weimer, F. C., 1981, "Modeling of Nonholonomic Dynamic Systems with Applications", ASME Journal of Applied Mechanics, Vol. 48, No. 1, pp. 177-182.

5. Nikravesh, P. E., 1984, "Some Methods for Dynamics of Constrained Mechanical Systems: A Survey", NATO ASI Series, Vol. F9, Springer-Verlag, Berlin, pp. 351-368.

6. Ider, S. K., 1988, "Stability and Dynamics of Constrained Flexible Multibody Systems", PhD Thesis, University of Illinois at Chicago.

7. Ider, S. K., Amirouche, F. M. L., 1989, "Nonlinear Modeling of Flexible Multibody Systems Dynamics Subjected to Variable Constraints" ASME Journal of Applied Mechanics, Vol. 56, No. 2, pp. 444-450.

8. Ider, S. K., Amirouche, F. M. L., 1988, "Coordinate Reduction in the Dynamics of Constrained Multibody Systems", ASME Journal of Applied Mechanics, Vol. 55, No. 4, pp. 899-905.

9. Ider, S. K., 1989, "Modeling of Control Forces for Kinematical Constraints in the Dynamics of Multibody Systems-A New Approach", 3rd Annual Conf. on Aerospace Computational Control, Oxnard, CA, Sept.

10. Przemieniecki, J. S., 1968, Theory of Matrix Structural Analysis, McGraw-Hill, New York.

11. Naganathan, G., Soni, A. H., 1986, "Nonlinear Modeling of Kinematic and Flexibility Effects in Manipulator Design", ASME Paper, 86-DET-88.

12. Kane, T. R., Ryan, R. R., Banerjee, A. K., 1987, "Dynamics of a Cantilever Beam Attached to a Moving Base", Journal of Guidance, Control and Dynamics, Vol. 10, No.2, pp. 139-151.

CONTROL OF A FLEXIBLE PLANAR TRUSS USING PROOF MASS ACTUATORS

Constantinos Minas*
Ephrahim Garcia
Daniel J. Inman
Department of Mechanical and Aerospace Engineering
State University of New York at Buffalo
Buffalo, N.Y. 14260

Abstract

A flexible structure was modelled and actively controlled by using a single space realizable linear proof mass actuator. The NASA/UVA/UB actuator was attached to a flexible planar truss structure at an "optimal" location and it was considered as both passive and active device. The placement of the actuator was specified by examining the eigenvalues of the modified model that included the actuator dynamics, and the frequency response functions of the modified system. The electronic stiffness of the actuator was specified, such that the proof mass actuator system was tuned to the fourth structural mode of the truss by using traditional vibration absorber design. The active control law was limited to velocity feedback by integrating of the signals of two accelerometers attached to the structure. The two lower modes of the closed-loop structure were placed further in the LHS of the complex plane. The theoretically predicted passive and active control law was experimentally verified.

1. Introduction

Large continuous structures, like space structures tend to have tight restrictions on the actual response of the structure. A passive or active control design is often necessary for the structure to satisfy the desired response restrictions. The success of the passive and active control design is based on the accuracy of the model that describes the dynamic characteristics of the structure. Flexible distributed parameter systems can be successfully modelled by finite element analysis [1]. This category of structures is lightly damped and tends to have most of its mass concentrated at the joints [2]. Their natural frequencies are low and appear in closely spaced groups. The finite element model of the structure that consists of a mass and a stiffness matrix, can be reduced by traditional model reduction techniques by eliminating the insignificant displacements at the nodal points [3]. The dissipation energy of the system can be modelled by constructing a system damping matrix, by assuming a normal mode system [4], and by using the damping ratios obtained experimentally from modal parameter estimation methods [5, 6, 7]. In the case where the discrepancy between the analytical model and the experimentally obtained modal model is significant, the reduced order analytical damped model can be further modified [8], such that it is in agreement with the experimental natural frequencies, damping ratios and mode shapes [8,9,10,11,12,13]. It is important to realize that the design of the "optimal" control is based on the modified reduced order model, but it is actually applied to the real structure. Therefore, the model improvement mentioned above, becomes very important and its accuracy is vital in the success of the design of the control law.

The structure used here, is a planar truss constructed with space realizable links and joints in the configuration presented in fig.1. The truss is lightly damped and has the behavior of a large

---

*Currently with G.E. Corporate Research and Development Center, Advanced Projects Laboratory, Schenectady, N.Y. 12301.

space structure, with most of its mass concentrated at the joints [2]. It possesses low resonant frequencies that appear in closely spaced groups and has both translational and rotational modes of vibration.

The structure is passively and actively controlled by a single actuator. The actuator used in this experiment is the NASA/UVA/UB proof mass actuator system. The actuator dynamics are taken into consideration and a global model is constructed which includes both the structure and the actuator dynamics [14,15]. The location of the actuator is specified [16,17] by examining the eigenvalues of the uncontrolled global model and the frequency response functions of the global system. The actuator is considered as both a passive and an active device with two design variables, its electronic stiffness and the generated force. The electronic stiffness is specified such that the actuator proof-mass-electronic-spring system is tuned to one of the structural modes of the truss by using traditional vibration absorber design [18,19,20]. The generated force of the actuator is specified by using output feedback techniques. Here, the active control law was limited to velocity feedback by integrating the signals of two accelerometers attached to the structure. The objective is to move the two lower modes of the closed-loop structure further in the LHS of the complex plane and at the same time maintain stability of the closed-loop system [21,22]. The theoretically predicted passive and active control law are experimentally implemented and the results are evaluated.

2. Modeling

2.1 Construction of the Finite Element Model

The finite element model of the structure was constructed by using the commercially available MSC/PAL package for dynamic modeling. The structure weighed 7.335 Kg and was constructed with links and joints, mainly made of aluminum alloy. The density of the material was measured experimentally by using standard techniques. The Young's modulus of aluminum alloy was used, since the links and joints are mainly constructed with this material. The nodal points of the finite element model coincide with the location of the joints of the structure. Every nodal point was allowed to have three degrees of freedom, that is translation in the z-axis and rotations about the x and y-axis resulting in a 48-degree-of-freedom model (see Fig.1). The boundary conditions were assumed to be clamped for nodes 15 and 16 and free for the rest of the nodes, since the structure was supported as illustrated in fig.1. After the boundary conditions were applied the final model was a 42-degree-of-freedom model.

2.2 Mass Distribution

The mass distribution of a non-uniform structure is a problem, that should by no means be ignored. Here, two approaches were used. The first approach was to calculate an equivalent internal diameter of the hollow links, such that the links had the measured mass. The links were treated as uniform hollow tubes constructed with aluminum alloy with an equivalent length of 0.5m. The joints were modelled as a concentrated mass at the particular location and are treated as rigid. The natural frequencies of this model were calculated and are presented in table 1. The results were considered unsatisfactory and one of the links was disassembled for more insight to the mass distribution of the link. In the second approach, the real internal diameter of the links was used and the excessive mass was distributed to the nodes accordingly. The resulting natural frequencies of the model are compared to the experimental results in table 1. The finite element model was constructed using a finer grid which include more nodal points, specifically an additional nodal point at the mid-point of each link. The resulting model after the boundary conditions were applied was a 126-degree-of-freedom model.

It can be concluded that the 45-node(126-dof) model is not significantly better than the 16-node(42-dof) model in predicting the first fourteen natural frequencies. Therefore, it was found unnecessary to use the 45-node(126-dof) model in the determination of the control design of the structure, since the 16-node(42-dof) model was as accurate.

Table 1 : Comparison of the theoretical and experimental natural frequencies of the structure.

	FEM				TEST I (rot accel)
	Uniform mass distribution	Corrected mass distribution			
	42dof	42dof	126dof	14dof	SDOF analysis
Frequency in Hz					
1	1.38	1.045	1.048	1.039	1.07
2	4.56	3.467	3.468	3.469	3.54
3	10.88	8.050	8.050	8.051	7.94
4	26.98	19.894	19.894	19.902	-
5	29.68	21.746	21.748	21.750	-
6	30.94	22.077	22.074	22.087	22.54
7	42.63	30.468	30.472	30.477	32.61
8	53.79	39.268	39.252	39.326	40.35
9	68.46	48.524	48.521	48.552	-
10	72.61	51.746	51.704	51.842	52.51
11	82.93	58.645	58.629	58.718	61.41
12	101.93	71.169	71.116	71.275	65.62
13	102.88	72.090	72.039	72.285	78.24
14	116.52	80.741	80.610	80.920	91.74
15	236.64	219.856	183.903	-	187.13

2.3 Model Reduction

Most of the control algorithms are designed for first order systems. Transforming the 16-node(42-dof) model in the state space results in a 84-dof state space matrix. This matrix is quite large, and it was found that it is difficult to manipulate in vibration prediction, and control algorithms. Therefore, it was necessary to reduce the order of the model before performing control analysis and designing a control law. From the configuration of the model the rotational degrees of freedom can be considered as less significant than the translational ones, and can be eliminated from the model by using the Guyan reduction method [3]. The resulting reduced order model is a 14-dof model. Eigenvalue analysis of this model showed that this model maintained the first fourteen natural frequencies of the larger model quite accurately. The damping ratios determined from the modal test were used in the construction of the system's damping matrix, by assuming that the system exhibited normal mode behavior. The damping matrix is calculated by the following equation:

$$D = MU_F \mathrm{diag}(2\zeta_i\omega_i)U_F^{-1} \quad (1)$$

where U_F is the eigenvector matrix of $M^{-1}K$, and ζ_i are the experimentally obtained damping ratios. The final reduced order model is described by the following equation:

$$M\ddot{q}(t) + D\dot{q}(t) + Kq(t) = 0 \quad (2)$$

This equation describes only the dynamic characteristics of the structure. The actuator dynamics were considered important and they were included in the dynamic model.

2.4 Actuator Dynamics

The actuator that was used in this experiment was the NASA/UVA/UB proof mass actuator, presented in fig.2. The actuator system is comprised of a movable proof mass (m_{prf} = 0.225Kg), a fixed coil that applies an electromagnetic force on the proof mass, an analog interface board, a power amplifier and a linear variable differential transformer (LVDT) sensor. The LVDT transducer is an electromechanical transducer that measures the relative position of the proof mass with respect to the actuator housing. The actuator can be modelled as single degree of freedom mass-spring system, with a variable electronic stiffness and the ability to apply a force on the

structure at the attachment point. An equal and opposite force is applied on the proof mass of the actuator. The actuator is space-realizable in the sense that it does not have to be attached to the ground. The equations of motion are written by taking into account the actuator dynamics[15]. Let's assume that the actuator is attached to the structure at the ith nodal point. The global system that includes both the structure and the actuator dynamics, is of higher order, equal to the order of the original system plus the order of the actuator dynamics, and it is described by:

$$\begin{bmatrix} M_1 & 0 \\ 0 & m_{prf} \end{bmatrix}\begin{bmatrix} \ddot{q} \\ \ddot{q}_{prf} \end{bmatrix} + \begin{bmatrix} D_1 & & 0 \\ & -c_{act} & \\ 0 & 0 & c_{act} \end{bmatrix}\begin{bmatrix} \dot{q} \\ \dot{q}_{prf} \end{bmatrix} + \begin{bmatrix} K_1 & & 0 \\ & -k_{act} & \\ 0 & 0 & k_{act} \end{bmatrix}\begin{bmatrix} q \\ q_{prf} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ -1 \end{bmatrix} f_g \quad (3a)$$

where q_{prf} is the displacement of the proof mass (m_{prf}), the scalars k_{act} and c_{act} are the stiffness and damping of the electronic spring of the actuator, m_{par} is the parasitic mass of the actuator, f_g is force generated by the actuator, and the matrices M_1 ,D_1 and K_1 are the following matrices:

$$M_1 = M + m_{par}\text{diag}[0,...,0,1,0,...,0] \quad (3b)$$

$$K_1 = K + k_{act}\text{diag}[0,...,0,1,0,...,0] \quad (3c)$$

$$D_1 = D + c_{act}\text{diag}[0,...,0,1,0,...,0] \quad (3d)$$

This is referred to as the open-loop system and the mass, damping and stiffness matrices are denoted by subscript ($_{OL}$) for convenience. Note that the non-zero elements correspond to the ith row or/and column of the particular matrix or vector of the previous set of equations. The force f_g is the actuator-generated force applied on the structure. The electronic stiffness of the actuator can be selected in a variety of ways for various design approaches.

3. Passive Control Design

3.1 Structural Modification Design

The parasitic mass of the actuator housing has the same effect as adding a dead parasitic mass at the point of attachment. Increasing the mass of the structure is a structural modification, with the direct effect of reducing the lower natural frequencies of the system. The natural frequencies of the new model with the dead mass were examined both theoretically and experimentally, and the results are tabulated in table 2. The experimental results are presented in the form of point and transfer inertance (transfer function) plots. The transfer function of nodes 1 and 8, of both the original structure and the modified structure are presented in fig.3 and fig.4 respectively. The effect of attaching the PMA (inactive) was also examined. This configuration is equivalent of having a dead mass equal to the parasitic mass of the actuator housing plus the proof mass. However, when the actuator's electronic stiffness is activated, the proof mass becomes an additional degree of freedom, and it is not part of the parasitic mass any longer.

The results indicate that the modified structure has lower natural frequencies than the original structure. This is true for the first five structural modes as indicated in the table above. The experimental frequency response plots show that the level of the vibration response was reduced considerably, especially in the lower frequency region.

If the design methodology was limited to structural modification, it will be considered necessary to examine the effect of adding the dead mass at different nodal points. The results are presented in table 3. The design criterion that was used to place the actuator was to reduce the overall vibration level at node 1, because a sensitive device will be attached at that point. The actuator cannot be placed at node 1 because there is no room. Note that different design criterion results in different locations of the actuator. Placing the actuator at node 10 doesn't reduce the vibration at node 1 at all. Nodes 2, 3, and 4 have the same effect in reducing the vibration level of node 1. But the first structural mode is shifted at 0.92 Hz. This was considered undesirable because it is hard to control the low frequencies by active control. Placing the actuator at nodes 6,7 and 8 has the same effect in reducing the vibration level of node 1 and the first structural mode is not shifted considerably. Therefore, any of nodes 6,7, and 8 can be used as an "optimal" location of the actuator. The results that follow are for placing the actuator at node 8.

Table 2 : Comparison of the theoretical and experimental natural frequencies of the structure with and without the parasitic mass.

	FEM		TEST I		
	w/o	w	w/o	w dead mass	w PMA inactive
	Frequency in Hz				
1	1.04	0.97	1.07	1.01	1.02
2	3.47	2.94	3.54	3.09	2.96
3	8.05	8.00	7.94	7.69	7.88
4	19.90	16.42	-	17.01	16.03
5	21.75	21.44	-		22.39
6	22.09	22.06	22.54	22.02	23.50
7	30.48	28.53	32.61	30.08	29.50
8	39.33	39.12	40.35	39.78	39.33
9	48.55	46.40	-	-	-
10	51.84	51.45	52.51	49.31	50.68
11	58.72	58.52	61.41	54.57	57.36
12	71.27	70.71	65.62	65.02	66.29
13	72.28	72.28	78.24	77.73	78.41
14	80.92	80.74	91.74	84.8	-

Table 3 : Comparison of the theoretical natural frequencies of the structure with the parasitic mass at various nodal points.

	FEM								
	w/o	8	2	3	4	5	6	7	10
	Frequency in Hz								
1	1.04	0.97	0.93	0.93	0.92	0.98	0.98	0.98	1.01
2	3.47	2.94	3.39	3.40	2.94	2.96	3.41	3.40	3.42
3	8.05	8.00	7.71	7.66	7.65	7.95	7.93	7.95	7.28
4	19.90	16.42	18.25	18.41	17.47	15.54	19.84	19.88	19.52
5	21.75	21.44	21.74	21.45	20.17	21.75	20.52	20.24	20.52
6	22.09	22.06	21.98	22.07	21.77	21.96	21.94	21.75	22.00
7	30.48	28.53	30.09	30.02	29.60	27.79	30.07	30.43	29.83
8	39.33	39.12	39.17	38.15	37.87	36.87	39.30	38.92	37.06
9	48.55	46.40	45.12	46.65	48.35	48.35	43.27	45.40	43.03
10	51.84	51.45	51.67	49.02	49.76	50.89	49.56	51.40	51.83
11	58.72	58.52	54.15	57.71	58.47	58.54	58.60	56.68	56.07
12	71.27	70.71	68.85	68.34	70.27	70.62	67.91	68.31	68.53
13	72.28	72.28	71.87	72.26	72.26	72.09	71.67	72.23	71.34
14	80.92	80.74	80.44	80.13	80.69	80.67	79.27	79.27	77.81

3.2 Vibration absorber design

There are several criteria for tuning the absorber to a MDOF structure. The simplest criterion is to tune the natural frequency of the absorber to exactly one of the natural frequencies of the structure[18], that is:

$$\omega_a = \omega_i \tag{4a}$$

The design of the damped absorber results in an optimal tuned frequency given by[18]:

$$\omega_a = \frac{\omega_i}{1+\mu_i} \tag{4b}$$

where μ_i is the ratio of the mass of the absorber (here, the proof mass) over the mass of the SDOF structure (here, the modal mass at mode ω). The ratio μ_i or the modal mass can be calculated in a trial and error procedure. The difficulty of applying the second method is the fact that it is difficult to determine the optimal value for μ for the higher modes [22].

An optimal tuning criterion for MDOF systems was presented in reference [19]. The absorber frequency (ω_a) and damping coefficient (c_a) are given by:

$$\omega_a^2 = \omega_i^2 \frac{1+\mu_t}{(1+\mu_t+\mu_a)^2} \tag{5a}$$

$$c_a^2 = m_a^2 \omega_i^2 \mu_a \frac{1+\mu_t}{(1+\mu_t+\mu_a)^3} \tag{5b}$$

where,

$$\mu_t = m_t \phi_{i_j}^2 \text{ and } \mu_a = m_a \phi_{i_j}^2 \tag{5c}$$

The scalars m_t and m_a are the parasitic mass and the mass of the absorber, respectively, and the scalar ϕ_{i_j} is the jth entry of the associated eigenvector of the ith mode, where j is the degree of freedom corresponding to the location of the absorber. Note that the eigenvectors derived form the finite element model, are normalized with respect to the mass matrix.

3.2.2 Experimental implementation of the passive control design

The stiffness of the PMA can be electronically varied, such that the actuator system can be tuned to different frequencies. The PMA was attached to ground, and the LVDT signal was examined for random signal input that generates an electromagnetic force on the proof mass. The LVDT signal gives the relative position of the proof mass with respect to the housing of the actuator. As it can be clearly seen in the experimental bode plot in fig.5, the PMA system is well modelled by a SDOF system, with a natural frequency depending on the gain that determines the electronic stiffness. The stiffness is a function of the external gain (α), and other electromagnetic constants of the coil and the amplifier (included in the factor K) . The natural frequency of the system is given by:

$$\omega_a = 1/2\pi \sqrt{\alpha K/m_{prf}} \tag{6}$$

The damping in the actuator was identified as Coulomb damping due to the friction in the bearings. An equivalent viscous coefficient was calculated from the frequency response functions of the LVDT signal at particular tuning frequencies. It was found that the lower the tuning frequency becomes, the higher the equivalent damping becomes. This is actually due to the fact that at low frequencies the proof mass of the actuator cannot overcome the friction. As a consequence, the natural frequency of the SDOF model of the actuator dynamics cannot go lower than a certain frequency, since the stiffness is electronically determined and it depends on the relative motion of the proof mass with respect to the housing of the actuator. It was found that the actuator system behaves like an overdamped system when tuned to frequencies below 8 Hz. Therefore, it was practically impossible to tune the actuator to frequencies lower than 8 Hz. Note that, this range includes the three lower natural frequencies of the modified structure. Therefore, the PMA is tuned to the fourth mode, by using the criteria described above. The results from only the second criterion are presented here in the top part of fig.6, due to the fact that the plots from the simple criterion (equation 4a) and the optimal tuning criterion (equation 5) were very similar. It can be clearly seen that the vibration response is clearly reduced.

4. Active Control design

The active control law is implemented, by using one actuator and two sensors. The force generator signal of the actuator was then given by:

$$f_g = FCy(t) \quad (7)$$

where F the feedback gain matrix and C the output matrix. The sensors were placed at node 1 and node 4 as indicated in fig.1. Node 1 was chosen because this is the possible point of attachment of a sensitive device, where the vibration level is required to be reduced. Node 4 was chosen, because it moves in the opposite direction of node 1, when the structure is excited at one of its rotational modes. Here, accelerometers were used and their signals were integrated once by an analog computer, to give the corresponding velocity signals. The output position matrix was therefore zero, and the velocity output matrix was of the form:

$$C_1 = \begin{bmatrix} 1 & 0_{1\times 14} \\ 0_{1\times 3} \; 1 & 0_{1\times 11} \end{bmatrix} \quad (8)$$

The gain matrix is therefore given by:

$$F = [g_1 : g_2] \quad (9)$$

where g_1 and g_2 are the two gains to be determined. Substituting into the previous equation results in:

$$f_g = F\begin{bmatrix} 1 & 0_{1\times 14} \\ 0_{1\times 3} \; 1 & 0_{1\times 11} \end{bmatrix}\dot{q}(t) \quad (10)$$

The closed-loop system written in physical coordinate system, is given by the following equation:

$$M_{OL}\ddot{q}(t) + D_{OL}\dot{q}(t) + K_{OL}q(t) = B_{OL}FC_1\dot{q}(t) \quad (11)$$

The objective here is to calculate the gain matrix F such that the system has poles at the desired locations. The right hand side of the previous equation is expanded as:

$$B_{OL}FC_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ -1 \end{bmatrix}[g_1 : g_2]\begin{bmatrix} 1 & 0_{1\times 14} \\ 0_{1\times 3} \; 1 & 0_{1\times 11} \end{bmatrix} = \begin{bmatrix} & 0_{7\times 15} & \\ g_1\; 0\; 0 & & g_2 \; 0_{1\times 11} \\ & 0_{6\times 15} & \\ -g_1\; 0\; 0 & & -g_2 \; 0_{1\times 11} \end{bmatrix} \quad (12)$$

Note that this is a square sparse asymmetric matrix with only four non-zero elements. This results in a closed-loop system damping matrix of the form:

$$D_{CL} = \begin{bmatrix} D_1 & & 0 \\ & & -c_{act} \\ & & 0 \\ 0 & -c_{act} \; 0 & c_{act} \end{bmatrix} + \begin{bmatrix} & 0_{7\times 15} & \\ g_1\; 0\; 0 & & g_2 \; 0_{1\times 11} \\ & 0_{6\times 15} & \\ -g_1\; 0\; 0 & & -g_2 \; 0_{1\times 11} \end{bmatrix} \quad (13)$$

where c_{act}, corresponds to the equivalent viscous damping coefficient of the actuator system.

The objective here, was to decrease the amplitude of the vibration response at the low modes that have high participation factors. Note that, direct pole placement design could not be applied since with one actuator and two sensors, only one closed-loop pole can be placed. The gains were determined in an ad hoc design, from an algorithm that covered a broad region of values, with the main objective to move the lower two poles further in the LHS complex plane. The results are presented in table 6. It can be clearly seen that the closed-loop system is stable when the two gains g_1 and g_2, are in the region -10 to 10 and 0 to 15 respectively. A finer grid that covered the part of the stable region, where the damping of the first two modes was increased (g_1 from 0 to 10 and g_2 from 10 to 20) was also examined [22].

It was discovered that the "optimal " gain of F= [5 : 15] increases the damping on modes 1, 2, 4, 5, 6 and decreases the damping at mode 3. Note that, further increase of the gains towards the "optimal" direction, resulted in an unstable closed-loop system. The experimentally obtained transfer functions of nodes 1 and 8, are presented in fig.6, and they are compared directly with the open-loop system, tuned to the fourth structural mode. The results show clearly, a decrease in the response at modes 1 and 2. The decrease of the vibration response is not very large as desired, because of the following reasons:

(i) By using only one actuator and two sensors, we can only affect 4 elements of the 15x15 closed-loop damping matrix.

(ii) Further increase in the gains towards the "optimal " direction drives the third mode unstable.

(iii) We are trying to control a flexible structure with many significant modes that cannot be ignored.
(iv) We are only using velocity feedback

It was also illustrated experimentally that by increasing the gains at higher values drove the proof mass system unstable.

Table 6 : Determination of the feedback gain matrix

g_2 \ g_1	-20	-15	-10	-5	0	5	10	15	20	25	30
-20	U	U	U	U	U	U	U	U	U	U	U
-15	U	U	U	U	U	U	U	U	U	U	U
-10	U	U	U	U	U	U	U	U	U	U	U
-5	U	U	U	U	U	U	U	U	U	U	U
0	U	U	U	S	S	U	U	U	U	U	U
5	U	U	S	S	S	S	U	U	U	U	U
10	U	U	U	S	S	S	S	U	U	U	U
15	U	U	U	U	S	S	U	U	U	U	U
20	U	U	U	U	U	U	U	U	U	U	U

U = unstable, S = stable.

5. Closing Remarks

An experimental flexible planar truss structure was modelled and successfully controlled in a passive and active way by using a space realizable linear proof mass actuator system. The PMA was attached to the truss at a desired location, and tuned as traditional vibration absorber to one of the structural modes of the truss by using several criteria. The actuator dynamics were successfully modelled and taken into consideration in the design of the passive and active control law. The active control design was adopted in the form of output velocity feedback by integrating the signals of two accelerometers, attached to the structure. The limitations of this method were indicated and difficulties of applying output feedback on large flexible structures with several significant modes are identified and pointed out.

6. Acknowledgements

This work was supported in part by AFOSR grants no F49620-88-00018 and F49620-86-6-0111. Much of the equipment used was funded through instrumatation grants numbers AFOSR-85-0119 and AFOSR-88-450-0390.

7. References

(1) Shames, I.H. and Dym, C.L., 1985, Energy and Finite Element Methods in Structural Mechanics, Hemisphere Publishing Corp., Chapter 16, pp.643-657.
(2) Balas, M. J., 1982, "Trends in Large Space Structure Control Theory: Fondest hopes, Wildest Dreams," IEEE Transactions on Automatic Control, Vol. AC-27, No. 3, June, pp. 522-535.
(3) Guyan, R.J., 1965, "Reduction of Stiffness and Mass Matrices," AIAA Journal, Vol. 14, pp.1627-1628.
(4) Caughey, T.K. and O'Kelly, M.E., 1965, "Classical Normal Modes in Damped Linear Dynamic Systems," ASME Journal of Applied Mechanics, Vo. 32, pp.583-588.
(5) Ewins, D.J., 1986, Modal Testing: Theory and Practice, Research Studies Press Ltd., England.
(6) Ibrahim, S.R. and Mikulcik, E.C., 1976, "The Experimental Determination of Vibration Parameters from Time Responses," Shock and Vibration Bulletin, No. 46, pt.5, pp.187-196.
(7) Juang, J.N. and Pappa, R.S., 1985, "An Eigensystem Realization Algorithm (ERA) for Modal Parameter Identification and Model Reduction," AIAA Journal of Guidance, Control and Dynamics, vol.8, 5, Sept-Oct, pp.620-627.

(8) Kammer, D.C., 1987, "An Optimum Approximation for Residual Stiffness in Linear System Identification," Proceedings of the 28th SDM Conference, Monterey, California, pp.277-287.
(9) Berman, A., 1984, "System Identification of Structural Dynamic Models-Theoretical and Practical Bounds," Proceedings of the 25th SDM Conference, Palm Springs, California, May, pp.123-129.
(10) Berman, A. and Nagy, E.Y., 1983, "Improvement of a Large Analytical Model Using Test Data," AIAA Journal, Vol.21, Aug., pp.1168-1173.
(11) Minas, C. and Inman, D.J., 1988, "Correcting Finite Element Models with Measured Modal Results using Eigenstructure Assignment Methods," Proceedings of the 6th International Modal Analysis Conference, Orlando, Florida, February, pp.583-587.
(12) Minas, C. and Inman, D.J., 1989, "Matching Finite Element Models to Modal Data," ASME Journal of Vibration, Acoustics, Stress, and Reliability in Design, (accepted for publication, paper No. 88-487).
(13) Minas, C. and Inman, D.J., 1989, "Model Improvement by using Pole Placement Methods," Proceedings of the 12th Biennial ASME Conference in Vibration and Noise, Sept., Montreal, Canada.
(14) Zimmerman, D. C., Horner, G. C., and Inman, D. J., 1988, "Microprocessor Controlled Force Actuator," AIAA Journal of Guidance, Control, and Dynamics, Vol. 11, No. 3, May-June, pp.230-236.
(15) Harokopos, E. G. and Mayne, R. W., 1986, "Motor Characteristics in the Control of a Compliant Load," AIAA Journal of Guidance, Control, and Dynamics, Vol. 9, No. 1, Jan.-Feb., pp. 113-118.
(16) Chiang,H.D., Thorp, J.S., Wang, J.C., and Lu, J., 1989, "Optimal Controller Placements in Large Scale Linear Systems," Proceedings of the American Control Conference, pp.1615-1620.
(17) Min, I.J., Chang, and Soong,T.T., 1980, "Optimal Controller Placement in Modal Control of Complex Systems," Journal of Mathematical Analysis and Applications, Vol.75, pp.340-358.
(18) Den Hartog, 1956, Mechanical Vibrations, 4th edition, McGraw-Hill, New York.
(19) Juang, J.N., 1984, Optimal Design of a Passive Vibration Absorber for a Truss Beam," AIAA Journal of Guidance, Control and Dynamics, vol.7, 6, Nov.-Dec., pp.733-739.
(20) Miller, D.W. and Crawley, E.F., 1988, "Theoretical and Experimental Investigation of Space-Realizable Inertial Actuation for Passive and Active Structural Control," AIAA Journal of Guidance, Control and Dynamics, vol.11, 5, Sept-Oct, pp.449-458.
(21) Andry, A.N., Shapiro, E.Y., and Chung, J.C., 1983, "Eigenstructure Assignment for Linear Systems, "IEEE Transactions on Aerospace and Electronic Systems, " Vol. AES-19, no.5, September, pp.711-729.
(22) Minas, C., 1989, "Modeling and Active Control of Large Flexible Structures," Ph.d. dissertation, State University of New York at Buffalo .

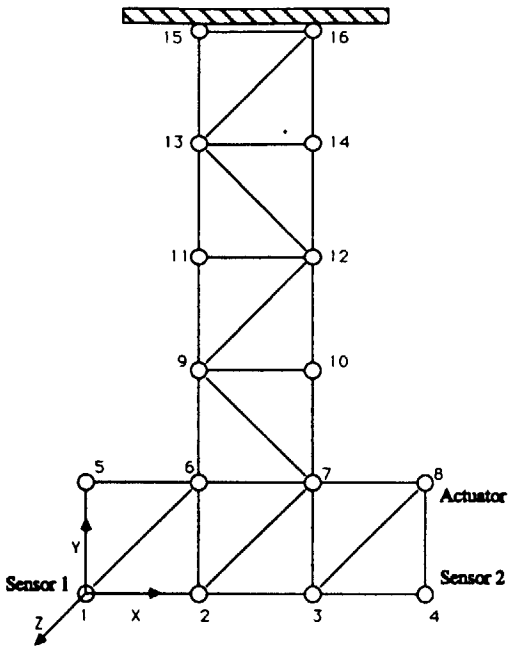


Fig.1 : Structure configuration with locations of sensors and actuator

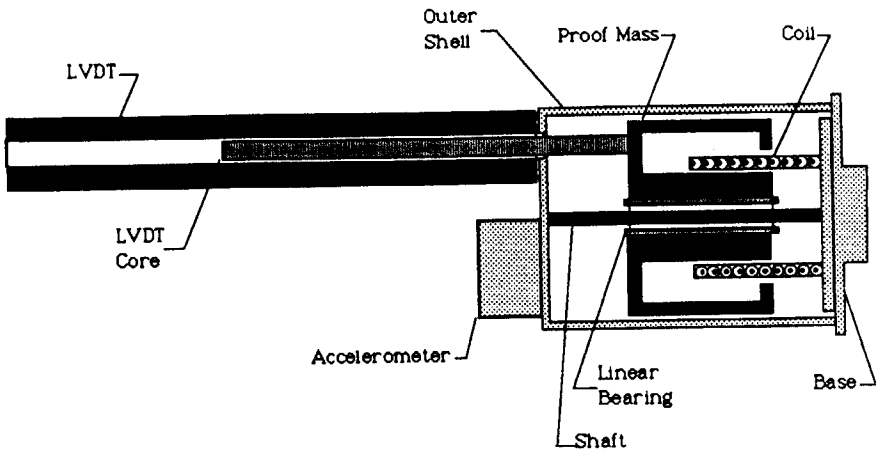


Fig 2 : Configuration of the proof mass actuator system

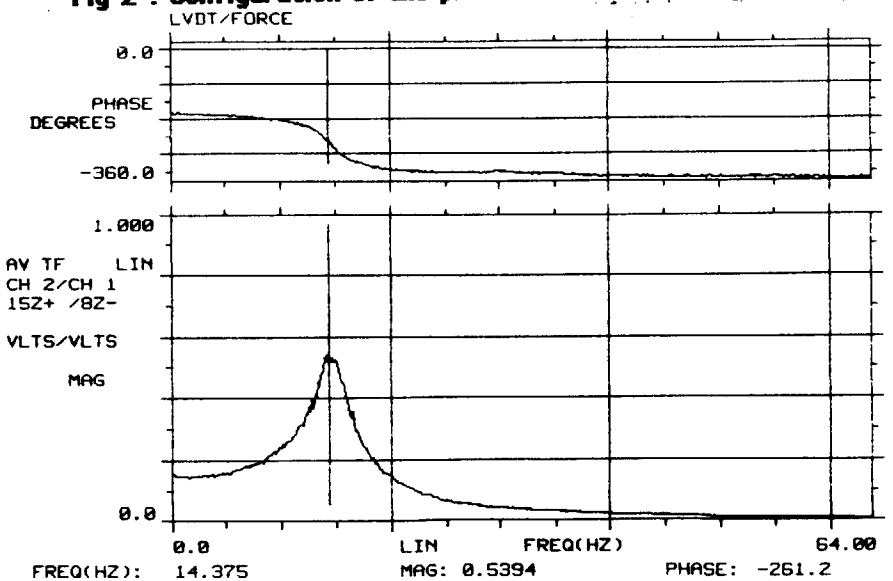


Fig.5 : Frequency Response function of the LVDT signal of the Proof-mass-actuator tuned to 14.4 Hz

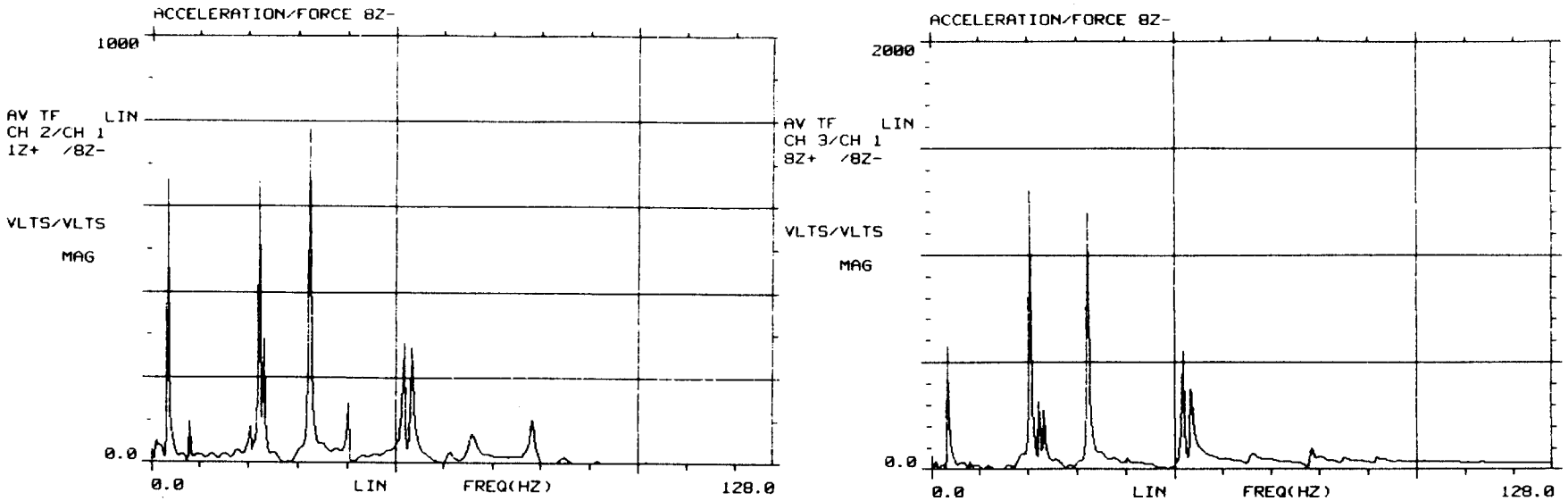


Fig.3 : Transfer function of the uncontrolled structure

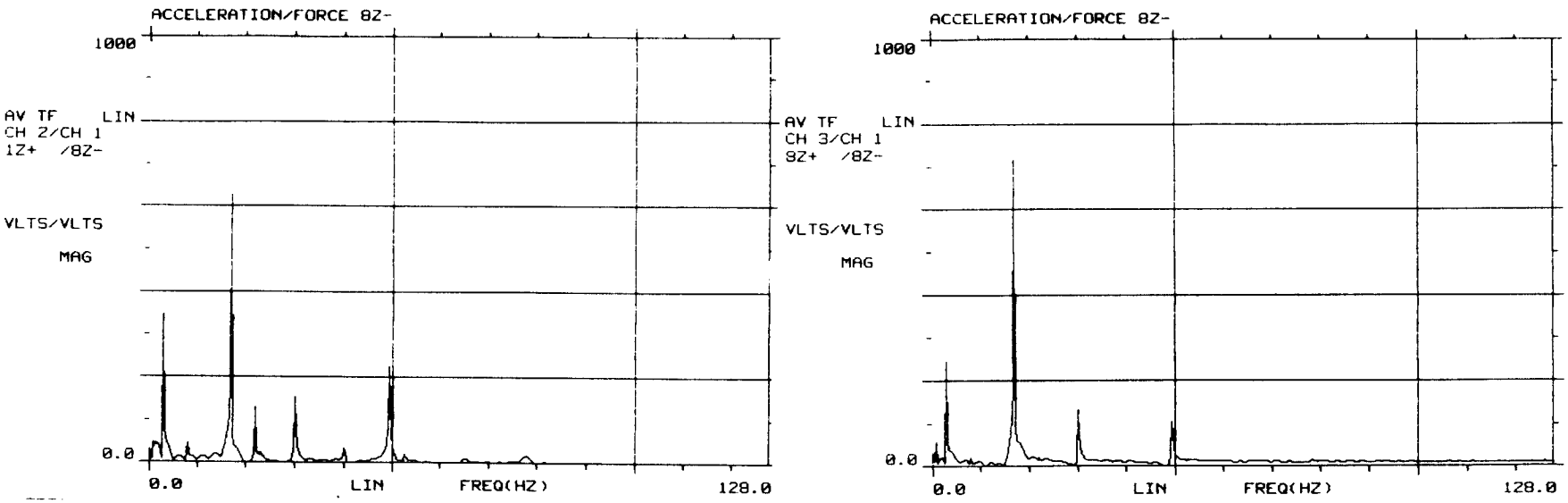


Fig.4 : Transfer function of the uncontrolled structure with parasitic mass

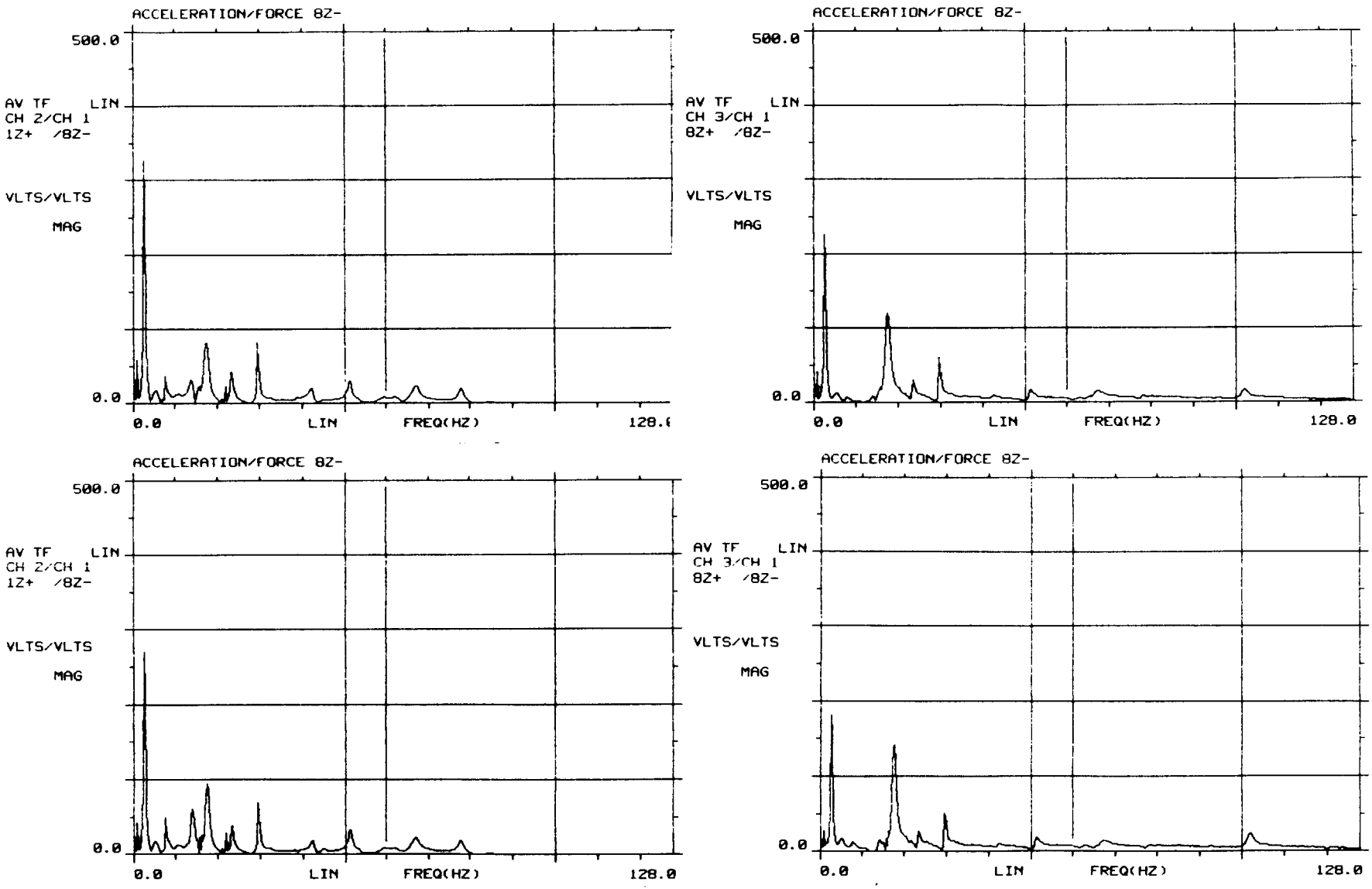


Fig.6 : Response of the passively controlled structure (top) and the actively controlled structure (bottom)

Simulation Studies Using Multibody Dynamics Code DART

James E. Keat
Photon Research Associates, Inc.

Abstract

DART is a multibody dynamics code developed by Photon Research Associates for the Air Force Astronautics Laboratory (AFAL). The code is intended primarily to simulate the dynamics of large space structures, particularly during the deployment phase of their missions. DART integrates nonlinear equations of motion numerically. The number of bodies in the system being simulated is arbitrary. The bodies' interconnection joints can have an arbitrary number of degrees of freedom between 0 and 6. Motions across the joints can be large. Provision for simulating on-board control systems is provided. Conservation of energy and momentum, when applicable, are used to evaluate DART's performance.

After a brief description of DART, the paper describes studies made to test the program prior to its delivery to AFAL. Three studies are described. The first is a large angle reorientating of a flexible spacecraft consisting of a rigid central hub and four flexible booms. Reorientation was accomplished by a single-cycle sine wave shape torque input. In the second study, an appendage, mounted on a spacecraft, was slewed through a large angle. Four closed-loop control systems provided control of this appendage and of the spacecraft's attitude. The third study simulated the deployment of the rim of a bicycle wheel configuration large space structure. This system contained 18 bodies. An interesting and unexpected feature of the dynamics was a pulsing phenomena experienced by the stays whose playout was used to control the deployment.

The paper concludes with a short description of the current status of DART.

On Trajectory Generation for Flexible Space Crane: Inverse Dynamics Analysis by LATDYN

G-S. Chen
Jet Propulsion Laboratory/California Institute of Technology
J. M. Housner
NASA Langley Research Center
S-C. Wu and C-W. Chang
The COMTEK Company

Abstract

For future in-space construction facility, one or more space cranes capable of manipulating and positioning large and massive spacecraft components will be needed. Because the space systems being constructed are relatively large and massive, the space cranes must have a reach on the order of 100-meter and be made of truss-type construction for structural efficiency. In order to optimize space crane's performance, an operational strategy consisting of gross-motion and fine-motion phases was proposed. Under this strategy, a space crane is commanded into position in a relatively fast pre-planned trajectory with relaxed requirements, and then "rigidized" by bracing against either the workpiece or an auxiliary support structure. After bracing, the subsequent fine motion will not involve the major crane bodies, and the precision movements between the workpieces can be performed without the adverse flexible crane body effect.

Inverse dynamics has been extensively studied as a basis for trajectory generation and control of robot manipulators. This paper will focus on trajectory generation in the gross-motion phase of space crane operation. Inverse dynamics of the flexible crane body is much more complex and intricate as compared with a rigid robot link. To model and solve the space crane's inverse dynamics problem, LATDYN program which employs a three-dimensional finite element formulation for the multibody truss-type structures will be used. The formulation is oriented toward a joint dominated structure which is suitable for the proposed space crane concept. To track a planned

trajectory, procedures will be developed to obtain the actuation profile and dynamics envelope which are pertinent to the design and performance requirements of the space crane concept.

MINIMUM ATTAINABLE RMS ATTITUDE ERROR USING CO-LOCATED RATE SENSORS

A. V. Balakrishnan†

Abstract

In this paper we announce a closed form analytical expression for the minimum attainable attitude error (as well as the error rate) in a flexible beam by feedback control using co-located rate sensors. For simplicity, we consider a beam clamped at one end with an offset mass (antenna) at the other end where the controls and sensors are located. Both control moment generators and force actuators are provided. The results apply to any beam-like lattice-type truss, and provide the kind of performance criteria needed under CSI — Controls-Structures-Integrated optimization.

---

† Research Supported in part under NAS1-18585 Task Assignment 49.

1. Introduction

One of the challenges in the Design Challenge For Flexible Flight Structure Control System Design formulated in the inaugural paper on SCOLE [1] was to hold the antenna pointing error within ±0.02 degrees after slewing by appropriate feedback control. In this paper we derive a closed form expression for the minimal achievable mean square pointing error using co-located rate sensors. A slightly simplified form of the SCOLE article (which eliminates rigid-body modes) is used: a cantilevered beam with an offset mass where the controls — both c.m.g.'s and force actuators — and the rate sensors are located. Our results are in terms of continuum model parameters — the uniform Bernoulli version is used. The beam dynamics are given in Section 2. The main results are in Section 3. We note that a technique for deriving equivalent Bernoulli beam parameters for various types of trusses is described by Noor and Anderson in [4]. Recently Noor and Russell [5] presented equivalent anisotropic Timoshenko beam models for beam-like lattice trusses with an arbitrary degree of modal coupling, which appear to yield excellent agreement with modal frequencies derived from finite element models. Our theory is able to handle these Timoshenko models, and moreover we can also use it for rigid-body modes, although they are not included here. Thus our results can be used for any beam-like lattice truss structure.

2. The Model

We consider a uniform Bernoulli beam clamped at one end with an offset mass (antenna) at the other end which also houses the sensors and actuators. See Figure 1. We allow for both force actuators and moment actuators. The sensors are rate gyros. Because of the clamping at one end, no rigid-body modes are involved and hence no attitude sensors are needed.

We allow bending in two mutually perpendicular planes containing the beam axis, as well as torsion in the plane perpendicular to the beam axis, all uncoupled. The continuum model (uniform Beroulli beam) dynamics can then be described by the following partial differential equations (similar to those in [2, 3]). Let the beam extend along the z-axis, $0 < s < L$, and let $u_\phi(s, t)$, $u_\theta(s, t)$, denote the bending displacements and $u_\psi(s, t)$ the torsion angle about the beam axis. Let in the usual notation (cf. [1]), EI_ϕ, EI_θ denote the flexural stiffness and GI_ψ the torsional rigidity. Let ρ denote the mass per unit area and A the cross-sectional area. Then we have:

$$\rho A \frac{\partial^2 u_\phi(s, t)}{\partial t^2} + EI_\phi \frac{\partial^4 u_\phi(s, t)}{\partial s^4} = 0, \qquad 0 < s < L;\ 0 < t$$

$$\rho A \frac{\partial^2 u_\theta(s, t)}{\partial t^2} + EI_\theta \frac{\partial^4 u_\theta(s, t)}{\partial s^4} = 0, \qquad 0 < s < L;\ 0 < t$$

$$\rho I_\psi \frac{\partial^2 u_\psi(s, t)}{\partial t^2} - GI_\psi u_\psi''(s, t) = 0, \qquad 0 < s < L;\ 0 < t$$

with the clamped boundary conditions at $s = 0$:

$$u_\phi(0, t) = u_\theta(0, t) = u_\psi(0, t) = 0$$

$$u_\phi'(0, t) = u_\theta'(0, t) = 0.$$

The antenna center of gravity is located at

$$(r_x, r_y, L).$$

The distance from the beam tip to antenna center of gravity is denoted by

$$|r| = \sqrt{r_x^2 + r_y^2}.$$

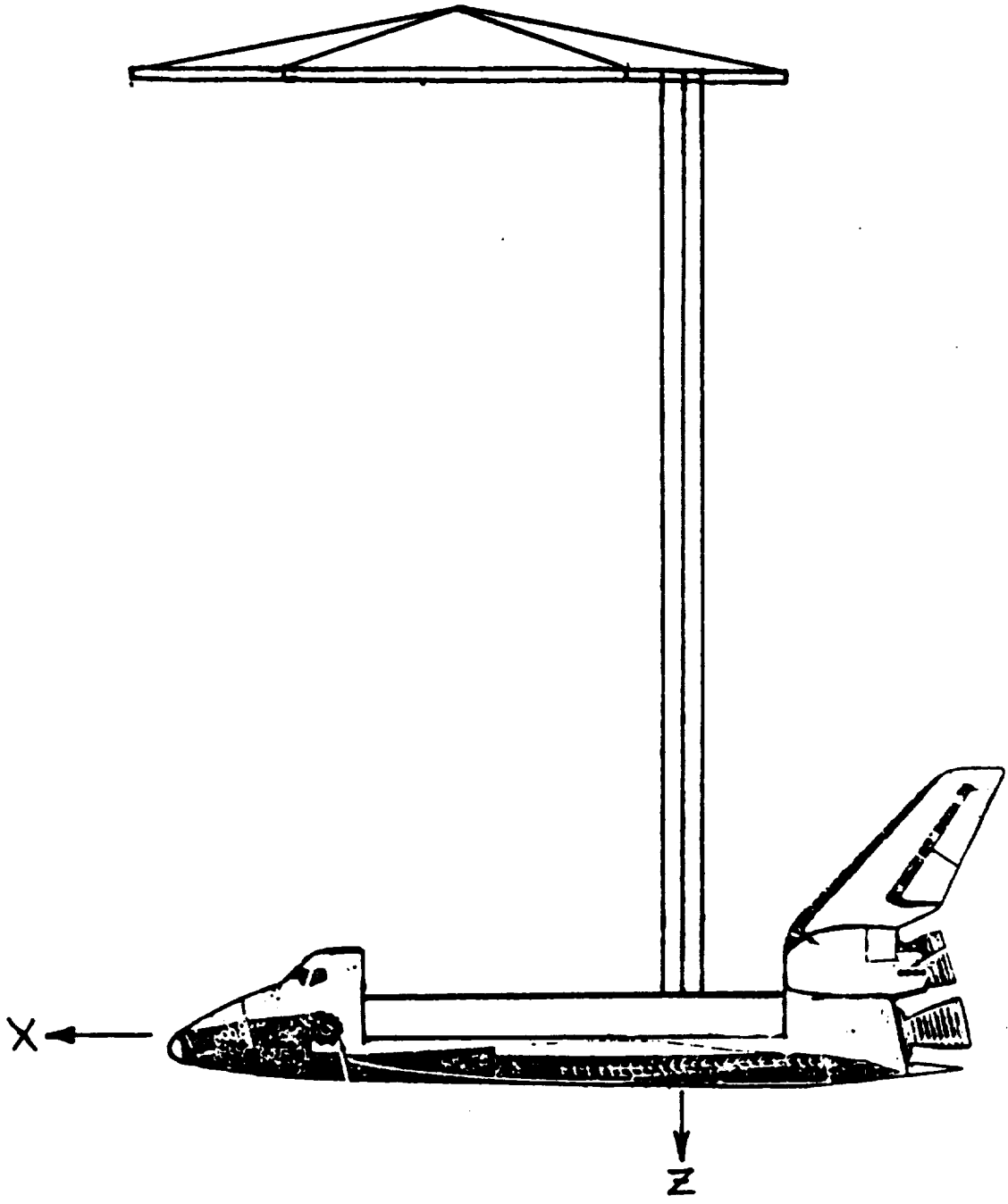


Figure 1: Shuttle/Antenna Configuration

The force balance equation at $s = L$ yields

$$m \begin{vmatrix} 1 & 0 & r_x \\ 0 & 1 & r_y \end{vmatrix} \begin{vmatrix} \ddot{u}_\phi(L,t) \\ \ddot{u}_\theta(L,t) \\ \ddot{u}_\psi(L,t) \end{vmatrix} + \begin{vmatrix} f_1(t) \\ f_2(t) \end{vmatrix} = \begin{vmatrix} EI_\phi u_\phi'''(L,t) \\ EI_\theta u_\theta'''(L,t) \end{vmatrix}$$

where m is the antenna mass and $f_1(\cdot)$, $f_2(\cdot)$ are the applied control forces. The torque balance equations yield

$$0 = \begin{vmatrix} EI_\phi u_\phi''(L,t) \\ EI_\theta u_\theta''(L,t) \\ GI_\psi u_\psi'(L,t) \end{vmatrix} + \hat{I}_a \dot{\omega} + M(t) + r \otimes \begin{vmatrix} f_1(t) \\ f_2(t) \\ 0 \end{vmatrix}$$

$$+ r \otimes \begin{vmatrix} \ddot{u}_\phi(L,t) + r_x \ddot{u}_\psi(L,t) \\ \ddot{u}_\theta(L,t) + r_y \ddot{u}_\psi(L,t) \end{vmatrix}$$

where the superdots indicate time derivatives and the primes the derivatives with respect to the spatial variable s; $\otimes$ denotes the vector cross-product and ω the angular rate vector

$$\omega = \begin{vmatrix} \dot{u}_\phi'(L,t) \\ \dot{u}_\theta'(L,t) \\ \dot{u}_\psi(L,t) \end{vmatrix},$$

$\hat{I}_a$ denotes the moment of inertia of the antenna about the beam tip ($s = L$) and finally, $M(t)$ denotes the applied control moment.

It is convenient to denote by $b(t)$ the boundary vector:

$$b(t) = \begin{vmatrix} u_\phi(L,t) \\ u_\theta(L,t) \\ u_\phi'(L,t) \\ u_\theta'(L,t) \\ u_\psi(L,t) \end{vmatrix}.$$

The boundary rate vector would thus be $\dot{b}(t)$. Hence our sensor model is:

$$v(t) = \dot{b}(t) + N_0(t)$$

where we assume that $N_0(t)$ is white Gaussian noise with spectral density matrix $d_0 I$, where I is the identity (5×5) matrix. Similarly we assume that the control actuators are also

characterized by additive white Gaussian noise. Denoting the applied control vector by $u(t)$:

$$u(t) = \begin{vmatrix} u_1(L,t) \\ u_2(L,t) \\ u_3(L,t) \\ u_4(L,t) \\ u_5(L,t) \end{vmatrix} .$$

we have

$$\begin{vmatrix} f_1(t) \\ f_2(t) \\ M(t) \end{vmatrix} = u(t) + N_s(t)$$

where $N_s(t)$ is white Gaussian with spectral density d_s. We shall also use M_b to denote the actuator mass/inertia matrix

$$M_b = \begin{vmatrix} m & 0 & 0 & 0 & mr_x \\ 0 & m & 0 & 0 & mr_y \\ 0 & 0 & & & \\ 0 & 0 & & \hat{I}_a & \\ mr_x & mr_y & & & \end{vmatrix}$$

where

$$\hat{I}_a = I_a + \begin{vmatrix} r_y^2 & -r_x r_y & 0 \\ -r_x r_y & r_x^2 & 0 \\ 0 & 0 & r_x^2 + r_y^2 \end{vmatrix}$$

where I_a is the antenna moment of inertia about its center of gravity. For any control input $u(\cdot)$ (which must perforce be a "feedback" control, based on the sensor data v(·)) the mean square pointing error is then expressed by:

$$\lim_{T\to\infty} \frac{1}{T} \left\{ \int_0^T u_\phi(L,t)^2 \, dt + \int_0^T u_\theta(L,t)^2 \, dt + |r|^2 \int_0^T u_\psi(L,t)^2 \, dt \right\}$$

and the mean square pointing *rate* is given by

$$\lim_{T\to\infty} \frac{1}{T}\left\{ \int_0^T \dot{u}_\phi(L, t)^2\, dt + \int_0^T \dot{u}_\theta(L, t)^2\, dt + |r|^2 \int_0^T \dot{u}_\psi(L, t)^2\, dt \right\}.$$

From the results in [6] it follows that the minimal attainable mean square pointing error is given by

$$aM_b^{-1}a^*$$

where

$$a = \text{row vector } (1, 1, 0, 0, |r|)$$

$$a^* = \text{transpose of } a$$

3. Main Results

We need some notation first. The mean square attitude response, whatever the feedback control used is defined by

$$\lim_{T\to\infty} \frac{1}{T}\left\{ \int_0^T u_\phi(t, \ell)^2\, dt + \int_0^T u_\theta(t, \ell)^2\, dt + |r|^2 \int_0^T u_\psi(t, \ell)^2\, dt \right\}. \tag{3.1}$$

This is recognized as the mean square displacement of the center of gravity of the antenna which is then also proportional to the mean square "pointing" error — see [1] for the relationships.

Next let u denote any (vector) of control inputs — a constant "step" input:

$$u = \begin{vmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{vmatrix}. \tag{3.1}$$

Solve the equations

$$\left.\begin{aligned} EI_\phi u_\phi''''(s) &= 0 \\ EI_\theta u_\theta''''(s) &= 0 \\ GI_\psi u_\psi''(s) &= 0 \end{aligned}\right\} \qquad 0 < s < L, \tag{3.2}$$

subject to the end conditions

$$\left.\begin{aligned} EI_\phi u_\phi'''(L) &= u_1 \\ EI_\theta u_\theta'''(L) &= u_2 \\ EI_\phi u_\phi''(L) + u_3 &= 0 \\ EI_\theta u_\theta''(L) + u_4 &= 0 \\ GI_\psi u_\psi'(L) + u_5 &= 0 \end{aligned}\right\} . \tag{3.3}$$

Note that the solution can be recognized as the steady-state response of the system to the step-input u, assuming that there is some damping. We only need to calculate the response to three specialized inputs:

Calculate the response to $u_\phi(L)$ to the special case, Case 1, where:

$$u_1 = 1$$

$$u_i = 0 , \qquad 2 \le i \le 5 .$$

Calculate the response $u_\theta(L)$ to the special case, Case 2, where:

$$u_1 = 0$$

$$u_2 = 1$$

$$u_3 = u_4 = u_5 = 0 .$$

Calculate the response $u_\psi(L)$ to the special case, Case 3, where

$$u_1 = u_2 = u_3 = u_4 = 0$$

$$u_6 = 1 .$$

Then the minimal achievable mean-square response whatever the choice of the feedback and whatever the mean-square control effort, is given by

$$\sqrt{d_s d_o}\ (u_\phi(L)^2 + u_\theta(L)^2 + r^2 u_\psi(L)^2) . \tag{3.4}$$

This is our main result. Unfortunately the derivation is beyond the scope of this report and

will be published elsewhere. To proceed further with (3.4) we calculate the solution of (3.2), (3.3) explicitly. Thus for any u, the solution is of the form

$$u_\phi(s) = a_3 s^3 + a_2 s^2, \qquad 0 < s < L$$

$$u_\theta(s) = b_3 s^3 + b_2 s^2, \qquad 0 < s < L$$

$$u_\psi(s) = c_1 s, \qquad 0 < s < L$$

where

$$a_3 = -\frac{u_1}{6EI_\phi}$$

$$b_3 = -\frac{u_2}{6EI_\theta}$$

$$a_2 = \frac{1}{2}\left[\frac{u_3}{EI_\phi} + \frac{u_1 \ell}{EI_\phi}\right]$$

$$b_2 = \frac{1}{2}\left[\frac{u_4}{EI_\theta} + \frac{u_1 L}{EI_\theta}\right]$$

$$c_1 = \frac{u_5}{GI_\psi}.$$

Thus for Case 1 we have

$$u_\phi(L)^2 = \frac{L^3}{3EI_\phi}$$

and for Case 2 we have

$$u_\theta(L)^2 = \frac{L^3}{3EI_\theta}$$

and for Case 3:

$$u_\psi(L)^2 = \frac{L}{GI_\psi}.$$

Hence the mean-square attitude error

$$= \sqrt{d_s d_o}\left(\frac{L^3}{3EI_\phi} + \frac{L^3}{3EI_\theta} + \frac{|r|^2 L}{GI_\psi}\right). \tag{3.5}$$

Note the appearance of the noise parameters in (3.5) in product form.

The technique for calculating the minimal mean square atttiude error in more complex models than that illustrated is the same: calculate the mean square step response (assuming

some damping) to unit step inputs.

In conclusion we suggest this result (3.5) can be the basis for combined structures-controls optimization — CSI, since the required structural parameters can be calculated for a lattice truss from the material gage and physical dimensions as in [4, 5]. We omit the details of these calculations.

References

1. L. W. Taylor and A. V. Balakrishnan. "A Mathematical Problem and a Spacecraft Control Laboratory Experiment (SCOLE): NASA/IEEE Design Challenge." *Proceedings of the NASA SCOLE Workshop, Langley Research Center, December 1984.*

2. A. V. Balakrishnan. "A Mathematical Formulation of the SCOLE Control Problem, Part I." NASA CR 172581. May 1985.

3. A. V. Balakrishnan. "Control of Flexible Flight Structures." In: *Analyse mathematique et applications*. Paris: Gauthier-Villars, 1988.

4. A. K. Noor and C. M. Anderson. "Analysis of Beam-like Trusses." *Computer Methods in Applied Mechanics and Engineering*, Vol. 20 (1979).

5. A. K. Noor and W. C. Russell. "Anisotropic Continuum Models for Beam-like Lattice Structures." *Computer Methods in Applied Mechanics and Engineering*, Vol. 57 (1986).

6. A. V. Balakrishnan. "A Mathematical Formulation of the SCOLE Control Problem, Part II: Optimal Compensator Design." NASA CR 181720. December 1988.

Characterization of Robotics Parallel Algorithms and Mapping onto a Reconfigurable SIMD Machine

C. S. G. Lee and C. T. Lin

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

ABSTRACT

The kinematics, dynamics, Jacobian, and their corresponding inverse computations are six essential problems in the control of robot manipulators. Efficient parallel algorithms for these computations are discussed and analyzed. Their characteristics are identified and a scheme on the mapping of these algorithms to a reconfigurable parallel architecture is presented. Based on the characteristics including type of parallelism, degree of parallelism, uniformity of the operations, fundamental operations, data dependencies, and communication requirement, it is shown that most of the algorithms for robotic computations possess highly regular properties and some common structures, especially the linear recursive structure. Moreover, they are well-suited to be implemented on a single-instruction-stream multiple-data-stream (SIMD) computer with reconfigurable interconnection network. The model of a reconfigurable dual network SIMD machine with internal direct feedback is introduced. A systematic procedure to map these computations to the proposed machine is presented. A new scheduling problem for SIMD machines is investigated and a heuristic algorithm, called neighborhood scheduling, that reorders the processing sequence of subtasks to reduce the communication time is described. Mapping results of a benchmark algorithm are illustrated and discussed.

1. Introduction

Robot manipulators are highly nonlinear systems and their dynamic performance is directly dependent on the efficiency of the kinematic and dynamic models, the control schemes/algorithms, and the computer architecture for computing the control schemes. In general, robot manipulators are usually servoed in the joint-variable space while the objects to be manipulated are usually expressed in the world (or Cartesian) coordinate system. In order to control the position and orientation of the manipulator end-effector, the robot controller is required to compute, at a sufficient rate, such tasks as coordinate transformation between the joint-variable space and the Cartesian space, generalized forces/torques to drive the joint motors, the manipulator inertia matrix for model-based control schemes, and the Jacobian matrix which relates the joint velocity in the joint-variable space to the Cartesian space. These are the basic robotic computations for the control of robot manipulators. They are equivalent to the computations of kinematics, dynamics, Jacobian, and their corresponding inverses. These six basic robotics computations are required at various stages of robot arm control and computer simulation of robot motion, and reveal a basic characteristic and common problem in robot manipulator control — *intensive computations with a high level of data dependency*. They have become major computational bottlenecks in the control of robot manipulators. Despite their impressive speed, conventional general-purpose uniprocessor computers cannot efficiently handle the kinematics and dynamics computations at the required computation rate because their architectures limit them to a mostly serial approach to computation. Furthermore, less efficient, serial computational algorithms must be used to compute these robotics computations on a uniprocessor computer. Consequently, the quest for real-time robot arm control and motion simulation rests on the study and development of parallel algorithms of lower computational complexity with faster computational structures. The ultimate goal is to achieve an *order-of-magnitude* and/or an *order-of-complexity* improvement in computational efficiency in these robotics computations by taking advantage of parallelism, pipelining, and architectures.

A common feature of today's research on robotic computational problems is that a specific problem, mostly the inverse dynamics or the inverse kinematics, is studied at a time, and usually an algorithmically-specialized architecture or processor is developed for that particular algorithm. Obviously, this specialized architecture can make the most use of the parallel properties of the algorithm. However, most advanced robot control schemes always require to solve a combination of some or all of the six basic robotic computations. One solution for this problem is to wire these specialized architectures or processors together. This method is inflexible because the combination of these components is dedicated to a particular control scheme and cannot be used efficiently for another scheme. Another solution is to connect the architectures or processors to a bus as peripherals of a general-purpose computer. This is more flexible, but the bus becomes a bottleneck and time is wasted in data movements between different computational processes. Another possible solution is focussed on partitioning the original algorithm/task into a set of subtasks with precedence relationship and then developing efficient scheduling algorithms to map these

subtasks onto a general-purpose multiprocessor system. This solution is much more flexible because most computational algorithms can be represented by directed task graphs. However, this approach may result in ignoring some inherent parallelism in robotics algorithms.

In this paper, we shall address these robotic computational problems, and major effort is focussed on finding a scheme which provides the flexibility needed to solve robotic computational problems on the same architecture while maintaining high efficiency by taking into account the inherent parallelism of robotics algorithms. To exploit the inherent parallelism of these robotics algorithms, our approach is first to characterize the set of parallel robotic algorithms based on the six specified characteristics and features, including type of parallelism, degree of parallelism, uniformity of the operations, fundamental operations, data dependency, and communication requirement. Our analysis shows that machines operating in the single-instruction-stream multiple-data-stream (SIMD) mode are the most efficient and suitable for our robotic algorithms. By fully considering the common characteristics and inherent parallelism of the robotics algorithms, a prototype of a medium-grained, reconfigurable, dual-network, SIMD machine with internal direct feedback has been designed for the computation of these kinematic and dynamic computational tasks. A systematic mapping procedure has been developed for scheduling these robotic computational tasks onto the proposed SIMD machine. This procedure builds a task table which contains the subtask assignment from the original parallel algorithm. Then a simplified task table and an input table are produced through the notation simplification. These two tables are then used as inputs to the neighborhood scheduling algorithm which reorders the processing sequence of the subtasks into a rescheduled task table to reduce the communication time. Finally, the subtasks in this rescheduled task table are mapped onto the proposed SIMD machine and a control table which describes the control sequence in the machine is produced. A benchmark algorithm which contains the characteristics of the six basic robotic computations has been implemented on the proposed SIMD machine, and the mapping results are included for discussion.

2. Characteristics of Parallel Algorithms

A key factor to the design of a parallel architecture for a group of algorithms is the understanding of their architectural requirements, and this requires us to identify the characteristics of these algorithms. This identification is usually helpful because the algorithms from a given application area such as robotics often possess an identifiable structure. In order to examine the characteristics of the six basic robotics parallel algorithms, a set of features which have the greatest effects on the execution of parallel algorithms is defined for robotics application [1].

- **Type of parallelism.** Two levels of parallelism can be identified.
  (a) *Job-level parallelism.* The original algorithm is reformulated to a parallel processable form. In this level, the variables carrying the same kind of information but with different indices (e.g., for different links or joints of a manipulator) are processed parallelly. Due to the nature of the robot's serial link structure, variables representing the same physical meaning are defined for each link such as joint velocities, joint accelerations, and joint torques. Usually, the same class of variables are produced through an identical computational procedure but with different set of data. This property is called uniformity of operations as defined below. So the job-level parallelism will often be amenable to the SIMD implementation and usually the required number of processors depends on the number of degrees of freedom of the manipulator (i.e., one processor for each joint).
  (b) *Task-level parallelism.* The original algorithm is decomposed into multiple subtasks. While the computation within a subtask is serial, the number of subtasks that can be processed concurrently is maximized by using some scheduling techniques. Obviously, this implies multiple-instruction-stream multiple-data-stream (MIMD) operations. Furthermore, for this level of parallelism, a subtask usually performs the same computation for different set of data, and hence the operation can be pipelined. An advantage of this task-level parallelism is that the required number of processors is independent of the number of degrees of freedom of the manipulator.
- **Degree of parallelism (Granularity).** Three levels of granularity are distinguished. In the *large grain granularity*, the parallelism is performed at the algorithmic level. That is, only the parallelism between different segments or subtasks is considered. For the *medium grain granularity*, the concurrency is considered at the operation level and the parallelism is performed based on some basic mathematical operations such as vector cross product and matrix-vector multiplication. If we consider the implementation of parallelism within the basic arithmetic operations, then the *fine grain granularity* is achieved. Different degrees of parallelism often imply different synchronization requirements. The finer the granularity is, the more frequent synchronization is required.
- **Uniformity of operations.** A robotics algorithm is said to possess uniformity of operations if the required computations for some set of variables, especially the joint variables, are uniform. An algorithm with operation uniformity can be implemented on an SIMD machine with higher efficiency.

- **Fundamental operations.** Algorithms in an application area usually perform similar mathematical operations. The identification of basic operations performed in the algorithm will dictate the processor capabilities needed.
- **Data dependency.** Three kinds of data dependency are classified for robotics algorithms: *local neighborhood dependency, special type dependency,* and *global dependency.* The local dependency means that the required operands in an operation come from its neighborhood; for example, from the results of last operation or using the same operands of last operation. The special type dependency is defined for some special equation or problem. There are some special types of data dependency that are peculiar and inherent to the robotics algorithms. Among them, the homogeneous (or hetero-homogeneous) linear recursive type of dependency which describes the data dependency in a homogeneous (or hetero-homogeneous) linear recursive equation appears most frequently. This linear recurrence structure plays a major role in the robotics algorithms because the variables of a joint are usually related to the corresponding variables of its adjacent joint due to the robot's serial link structure. Other special types of data dependency are defined for some well-known problems; for example, system of linear equations and Column-Sweeping algorithm for a triangular linear system. The global dependency means that the results of some operations may be required by other operations or equations that may appear in other places of the algorithm. Since few algorithms possess absolutely one kind of data dependency, we can just identify whether an algorithm is local data dependency oriented or not. The data dependency in an algorithm usually dictates memory organization, data allocation, and communication requirements.
- **Communication requirement.** The communication requirement decides the required interconnection type between processor and processor or between processor and memory. Three types of interconnection are considered: *one-to-one* connection, *permutation* and *broadcast* connections. Of course, the exact required interconnection type for each computation in an algorithm depends on many factors such as task assignment of each processor, data allocation in the memories, and data dependency of each computation. Hence, the exact required interconnection type can only be decided at the time of the algorithm-architecture mapping process. In examining the features of robotics parallel algorithms, only rough connection requirements can be observed.

3. Characterization of Basic Robotics Parallel Algorithms

Based on the above set of features, each of the six basic robotics algorithms have been carefully examined and analyzed to find the common features and characteristics among them [2]. Only the final results are presented here, which are useful for better understanding of the robotics computations and for designing a suitable parallel architecture for their computations.

Inverse Dynamics Problem. Among various methods for computing the inverse dynamics problem, the one based on the Newton-Euler (NE) equations of motion is the most efficient [3]. Since this method has been shown to possess the time lower bound of $O(n)$ running on uniprocessor computers, where n is the number of degrees-of-freedom of the manipulator, further substantial improvements in computational efficiency appear unlikely. Nevertheless, some improvements could be achieved by taking advantage of particular computation structures [4], customized algorithms/architectures for specific manipulators [5], parallel computations [6,7], and scheduling algorithms for multiprocessor systems [8-11].

Forward Dynamics Problem. Among various methods for solving the forward dynamics problem [12-14], the composite rigid-body method [12], based on the computation of the NE equations of motion, is widely used to develop efficient parallel algorithms [14-16]. The composite rigid-body method is suitable for parallel processing because efficient parallel algorithms for the inverse dynamics computation have been well developed and can be used to speed up the computation time.

Forward Kinematics Problem. Using the Denavit-Hartenberg matrix representation for establishing the link coordinate frames [17,18], the solution to the forward kinematics problem is the successive multiplication of the 4×4 homogeneous link transformation matrices for an n-link manipulator

$$\mathbf{T} = \mathbf{A}_0^1 \mathbf{A}_1^2 \mathbf{A}_2^3 \cdots \mathbf{A}_{i-1}^i \cdots \mathbf{A}_{n-1}^n \tag{1}$$

where $\mathbf{A}_{i-1}^i$ is the D-H link transformation matrix which relates the ith coordinate frame to the $(i-1)$th coordinate frame [17,18]. The above successive matrix multiplication equation can be reformulated in a homogeneous linear recursive form

$$\mathbf{T}_0^1 = \mathbf{A}_0^1 \quad \text{and} \quad \mathbf{T}_0^i = \mathbf{T}_0^{i-1} \mathbf{A}_{i-1}^i \quad \text{for } i = 2, \cdots, n, \tag{2}$$

from which the configuration of all the coordinate frames can be obtained at the time lower bound [7,19,20].

Forward Jacobian Problem. Existing methods in computing the Jacobian are mostly confined to uniprocessor computers. In particular, Orin/Schrader [21], and Yeung/Lee [22] exploited the linear recurrence characteristics of the Jacobian equations. These methods differed from each other only by a different selection of the reference

coordinate frame for computation. The reference coordinate frame is selected such that all the vectors and matrices and the Jacobian computed are referred to that reference coordinate system. They all have the computational order of $O(n)$ for an n-jointed manipulator.

Inverse Jacobian Problem. The inverse Jacobian algorithms for a general manipulator can be divided into two categories. One is to calculate the inverse or the generalized inverse Jacobian explicitly [23]. The other is to consider the inverse Jacobian problem as a system of linear equations and solve the joint rate from the Cartesian velocity implicitly [24]. For practical purposes, the latter approach is easier to be parallelized due to the use of some standard techniques to solve a system of linear equations such as the Gaussian elimination method.

Inverse Kinematics Problem. In general, the inverse kinematic position solution can be obtained by various techniques [18], among which the inverse transform [25] and the iterative method [26] are widely discussed. The inverse transform technique yields a set of explicit, non-iterative joint angle equations which involve multiplications, additions, square root, and transcendental function operations. The iterative methods can obtain robot independent joint solution, but they usually have some disadvantages: more computations than the closed-form solution, variable computation time and, more important, convergence problem, especially in the singular and degenerate cases. We shall examine the characteristics of the inverse transform technique and the iterative methods.

The equations for closed-form solution appear highly non-uniform [27]. To achieve higher parallelism for the inverse kinematics problem, the iterative method provides a better approach, since nearly every presented iterative method contains the computations of forward kinematics, forward Jacobian, and inverse Jacobian [26], which have been shown to be highly parallelized.

If we consider these six basic robotics computations as a set of tasks that we need to compute for the control of robot manipulators, then we need to find their common features and characteristics so that a parallel architecture can be designed to efficiently compute these tasks. The characteristics of the six basic robotics algorithms are tabulated in Table 1 and it shows that these algorithms do possess some important common features and characteristics. This is especially true for the inverse dynamics, the forward dynamics, the forward kinematics, and the forward Jacobian computations for the following three reasons. First, they are all suitable to be parallelized at the job-level and the parallelization can be performed at the large, medium, and fine grain granularities simultaneously, although different granularities are emphasized in each individual algorithm. Second, their operations are all uniform for the variables corresponding to each joint, and the most important fundamental operation is the matrix-vector operation. Finally, the strongest common feature is that they are all in homogeneous linear recursive form, for which the recursive doubling technique can be applied to achieve the time lower bound of $O(\lceil \log_2 n \rceil)$. The communication requirement indicates that one-to-one and some regular or irregular permutation capabilities are required for these four computational problems and the broadcast capability is necessary for the forward dynamics and the forward Jacobian algorithms. This indicates that some efficient, versatile network is required in the parallel architecture for their computations.

The inverse Jacobian and the inverse kinematics computations may seem less common to the above four algorithms. However, if less efficient methods to solve these two problems are chosen individually, then these two algorithms may possess some common features to the other four algorithms, and a common parallel architecture can be designed to match all these common characteristics for their computations. From previous discussions, we found that either the direct method or the iterative method for the inverse Jacobian is a proper candidate for parallel processing, while the direct method is more efficient with somewhat complex data dependencies. For the inverse kinematics problem, only the iterative method possesses regular properties similar to the other four computations.

With all the characteristics listed in Table 1, we shall next examine how to reformulate and parallelize these robotics algorithms from their original serial algorithms by complying to their common features [2]. The parallelization process is performed at the job level; that is, we try to express the original algorithms as a sequence of serial steps (jobs). Each individual step is accomplished through the cooperation of all the processors and for each step, the operations of each processor are almost identical by using one of their common features: the uniformity of operations. Hence, each step can be considered to be a single instruction in a serial program. Two different steps (or jobs) are identified after the parallelization process: *single* steps and *macro* steps. The notion of "single instruction" and "subroutine" of a serial program can be used to distinguish between these two steps. A single step corresponds to a single instruction in a serial program, while a macro step corresponds to a subroutine in a serial program. The macro steps require more complex parallel computations for all the processors, for example, the homogeneous linear recursive equation, the hetero-homogeneous linear recursive equation, and the system of linear equations are all macro steps. These macro steps are identified by their completeness and repeatity. The completeness means that the step can be treated as an individual problem. The technique to process these macro steps parallelly needs special consideration and the algorithm to solve these steps is so well-structured that finer decomposition is not helpful or even impossible, for example, the parallel recursive doubling technique for solving the homogeneous linear recursive equation, or the parallel Cholesky factorization technique for solving the system linear

equations with a symmetric-positive-definite square matrix. The repeatity means that the problem which can be solved in the step is so important and common that it appears repetitively at many other places; for example, many equations of robotics algorithms are in homogeneous linear recursive form, then the procedure for parallelly solving this problem can be applied to all these places. The method to parallelize each of these macro steps is designed separately.

Instead of computing all the six basic robotics algorithms, we synthesize a benchmark algorithm (see Table 2) which represents the general structure of the basic robotics parallel algorithms. This benchmark algorithm consists of six serial steps, and each step needs the cooperation of n processors. This benchmark algorithm will be used to demonstrate the whole process of mapping the "serial type" parallel algorithms onto a proposed parallel architecture in the following sections.

4. Design of Algorithmically-Specialized Parallel Architecture

In this section, an appropriate parallel architecture with the attributes that best match the common features of the six basic robotics parallel algorithms is designed. The important parallel architecture attributes include the type of machine (e.g., SIMD or MIMD mode), number of processors, synchronization requirement, processor capabilities, memory organization, and network requirement. Each of these attributes is affected by one or more features of the six basic robotics algorithms discussed in section 3. Detailed consideration for the design of this machine can be found in [2]. With all these requirements and attributes, the appropriate parallel architecture is a *reconfigurable, dual-network, SIMD* (DN-SIMD) machine for the computation of robotic algorithms.

The structure of the proposed DN-SIMD machine, as shown in Fig. 1, consists of multiple processing elements, two reconfigurable interconnection networks (RIN1 and RIN2), a set of global data registers (GDRs), three data buffers including register output buffer, PE output buffer and input data buffer (IDB), and a set of multiplexers. All of these are coordinated by a central control unit (CU) which is not shown in Fig. 1. The functions of each element are briefly described here.

1. *Processing Element* (PE). There are n identical PEs. Each PE is essentially an arithmetic logic unit (ALU) with attached working registers (see Fig. 1). All the ALUs perform the same programmable function synchronously in a lock-step fashion under the command of the CU. Some of the PEs can be masked (disabled) for some computation period, while other unmasked or enabled PEs perform computations. Each PE has two input working registers (IWRs) which are used to store two operands for each computation, and one output working register (OWR) which is used to store the current result of each computation. The operands in the IWRs are kept there until they are replaced. Thus, they can be used repetitively if one or two operands are common for a series of continuous computations. An inner loop connection within a PE is designed, which connects the OWR to one of the two IWRs. This provides an immediate inner-PE forwarding path such that the current result can be used as an operand for the next computation immediately.
2. *Global data registers* (GDRs). There are n groups of data registers which correspond to the n global memory modules. In each computation period, the registers with the same relative position in each group can be accessed under the control of the CU. The result of each computation from each PE will be stored in the GDRs only when either the result is the final output or the result will be used in later computations but not the immediate following one, which can make use of the internal forwarding path for data exchange among PEs or inner loop within PEs.
3. *Reconfigurable interconnection networks.* There are two sets of identical interconnection networks: RIN1 and RIN2. They are assumed to have full connectivity including one-to-one, permutation, and broadcast capabilities (e.g., the crossbar network). The RIN1 connects the GDRs to the PEs. This provides the paths for sending required operands to the appropriate PEs. The RIN2 makes the connection from the outputs of PEs to the inputs of PEs; this provides the direct paths for internal forwarding data exchange among PEs. It should be noted that, if necessary, the output of PE i can be stored into its corresponding memory module i. This is not affected by the RIN2.
4. *Data buffers.* There are three sets of data buffers. The register output buffer allows the "current computation" and the "RIN1 reconfiguration and operand fetch for the next computation" be processed at the same time. The PE output buffer allows the "current computation" and the "RIN2 reconfiguration and output data storing" be processed simultaneously. The input data buffer (IDB) is the buffer for operands directly from external input data.
5. *Multiplexer.* The n multiplexers in advance of the n PEs are used to select proper operands to enter PEs from three possible sources: GDR, IDB, and IFD exchange. They are also under the control of the control unit.

With the functions of these elements described above, the basic mathematical operations performed by each PE of the DN-SIMD machine involve at most two operands,

$$T = A \circ B \tag{3}$$

where A and B are two arbitrary operands and they can be scalar, vector or matrix, and "$\circ$" indicates the operation performed by the PE. When either A or B is null, the computation only involves one operand such as the transpose of a matrix. The operands A or B may come from five different sources. They are GDRs through RIN1, IFD exchange through RIN2, IDB, IWR within PE, and OWR within PE through inner loop connection. The result T may be sent to two possible destinations: GDRs directly, or PEs through RIN2 via IFD exchange. The possible input operands and output result transfer path diagrams are illustrated in Figs. 2(a) and 2(b) respectively. In Fig. 2(a), we demonstrate all the possible source combinations except the case that the operands come from the IWR within the PE. We assume that the time to transfer one operand from the GDR or the IDB to a PE (i.e., operand fetching) is the same as the time to transfer the output result from a PE to the GDR (i.e., result storing) and equals to the computation time of one basic PE operation. This time interval is called a *cycle*. Since operands fetching, computation, and result storing can be performed simultaneously due to the data buffers designed in this system, a three-stage pipelined operation can be performed on our DN-SIMD machine. Since a computation usually needs two operands A and B, and if A and B come from different sources, then they can be transferred to a PE simultaneously in one period. In this case, the three-stage pipelined operation proceeds normally. However, if A and B come from the same source (e.g., GDR or IDB), then it will take 2 cycles to transfer them. This situation is called the *double transmission required* (DTR) computation. In this case, a delay period must be added to the pipeline operation to synchronize the operation. This DTR computation obviously will slow down the system speed. Hence, we need to minimize the number of DTR computations in a computational task.

5. Mapping of Parallel Robotic Algorithms onto the Dual-Network SIMD Machine

Since our DN-SIMD machine was designed to best match the common characteristics of the six basic robotics parallel algorithms, the scheduling of their computations in our system is more straightforward with less difficulties as compared with other general mapping problems. Based on this characteristics matching, a systematic and efficient mapping procedure is developed to map the parallel robotic algorithms onto the proposed medium-grained DN-SIMD machine.

The proposed mapping procedure consists of three stages [2]. In the first stage, each of the single steps of these parallel robotic algorithms is further decomposed to a set of "subtasks" and each subtask possesses the basic mathematical form of consisting at most two operands. On the other hand, each of the macro steps in these algorithms is viewed as a subtask and is not decomposed at this stage. The first stage results in a series of parallel subtasks. In the second stage, these subtasks are reordered to reduce the number of DTR operations through a *neighborhood scheduling* algorithm. The reordered subtasks will be mapped onto the DN-SIMD machine directly in the third stage. In the final stage, the actual implementation of the macro steps in the parallel algorithms on the DN-SIMD machine is performed. Using the benchmark algorithm in Table 2 as an example, the details of these three stages of our mapping procedure are discussed in the following subsections.

5.1. Subtask Assignment

Since the proposed DN-SIMD is a medium-grained machine and is synchronized at each basic mathematical operation, each parallel algorithm must be decomposed into a series of subtasks. Each subtask is either in the basic mathematical form which involves at most two operands or in a well-defined macro step. Although this functional decomposition can be easily performed on the single steps, it is not the case for the macro steps, in which the data dependencies are so complex that the decomposition based on basic computational unit is not obviously feasible. So the macro step will be viewed as a single subtask in this stage. Consider the decomposition of the following equation

$$K = L \times (C + E) + G \times C \,. \tag{4}$$

Here we use three temporary variables, T_1, T_2, and T_3 to rewrite Eq. (4) into four simple equations in the basic mathematical form:

$$T_1 = C + E \;,\; T_2 = L \times T_1 \;,\; T_3 = G \times C \;, \text{ and } K = T_2 + T_3 \tag{5}$$

This same technique is applied to our decomposition process for single steps. For clarity, the benchmark algorithm is used as an example to demonstrate the technique. The decomposition result and the original algorithm are shown in Table 2. Here, two sets of variables are introduced: T_i's represent the immediate results (temporary variables) or the final outputs. If T_i is a macro subtask, then it is specially denoted as $\overline{T}_i$. I_i's represent the external input variables; that is, the variables that do not come from the outputs of other computations.

To ease the subtask scheduling in the second stage, notation simplification is performed on the above task table to produce a *simplified task table* as shown in Table 3. In this table, two arrays are defined: TB[i] contains the

identification of subtasks T_i's and OP[i] represents the corresponding operation for subtask TB[i]. Each element of OP[i] is either a macro subtask or in the form of $A \circ B$, where A and B may be T_i ($\overline{T_i}$) or I_i. Moreover, the superscript on A or B indicates the difference between the index i of the result, $T_{j_3}[i]$, and the index k or l of its operand $T_{j_1}[k]$ or $T_{j_2}[l]$, where $T_{j_3}[i] = T_{j_1}[k] \circ T_{j_2}[l]$. For example, the subtask $T_1[i] = T_2[i+2] \circ T_3[i-1]$ is denoted as $T_1 = T_2^2 \circ T_3^{-1}$. If their indices are equal, that is, $i = k$ or $i = l$, then the superscript is omitted. For example, the subtask $T_4[i] = T_5[i] \circ T_6[i]$ is denoted simply as $T_4 = T_5 \circ T_6$. The simplified task table is the final result of this stage and will be used as the input for the next stage.

5.2. Subtask Scheduling

To schedule the subtasks for computation, we first observe all the possible operand sources and their combinations for each computation. The operand may be one of the four possible types denoted as S_I, S_{OI}, S_T, and S_{OT} which correspond to four kinds of different sources. S_I denotes the operand from the IDB and it needs one period of transmission time. S_{OI} denotes the operand which is fetched by the previous computation (subtask) from the IDB and is still in the IWR within the PE, so no transmission is required for this operand. S_T denotes the operand from the GDR and this operand requires one cycle of transmission time via the network RIN1. S_{OT} denotes the operand from other sources including the following three possibilities: (i) The operand which is fetched by the previous computation from the GDR and is still in the IWR within the PE, so no transmission time is required; (ii) Current computation result through the inner loop; (iii) Current computation result through the internal forwarding path with data exchange provided by the network RIN2. The transmission time for the last two cases is ignored when compared to the system cycle time. Using these notations, all the possible combinations of operand sources including the situation of only one operand are listed below:

(S_I', S_I'')	(S_I, S_T)	(S_{OT}, S_T)	(S_{OI}, S_{OI})	(S_I)
(S_I, S_{OI})	$S_{OI}, S_{OT})$	(S_T', S_T'')	(S_{OT}, S_{OT})	(S_{OI})
(S_I, S_{OT})	(S_{OI}, S_T)	(S_I, S_I)	(S_T, S_T)	(S_T) , (S_{OT})

where the prime superscripts are used to distinguish different operands from the same kind of source. Among these situations, the combinations (S_I', S_I'') and (S_T', S_T'') are DTR operations and require two cycles to transmit two operands through the same transmission path. It is possible to eliminate DTR operations, if we reorder the processing sequence without violating the constraint of precedence relation. That is, in these two situations, one operand S_T' (or S_T'') can become the type S_{OT}, or S_I'(or S_I'') can become the type S_{OI}. Then, the DTR operation phenomena can be avoided and the unnecessary transmission can also be avoided for the efficient use of the same data repetitively and instantly.

A neighborhood scheduling algorithm for scheduling and reordering the execution of these subtasks to minimize the total number of DTR operations has been developed and is considered here.

Definition 1. For two subtasks in the kth and lth rows of the simplified task table, $TB[k]$ and $TB[l]$, assume $OP[k] = A \circ B$ and $OP[l] = C \circ D$, where A, B, C, and D are operands, each with one of these possible types: $\{I_j, T_j, T_j^i\}$. Then the subtask $TB[k]$ is called a *neighborhood* of $TB[l]$ if all the following conditions are satisfied:

(i) $k < l$,

(ii) $C = TB[k]$ or $C = TB^i[k]$ or $C = A$ or $C = B$ or
$D = TB[k]$ or $D = TB^i[k]$ or $D = A$ or $D = B$.

From the above definition, we know that if subtask $TB[l]$ has a previous subtask $TB[k]$ as its neighborhood ($k < l$) and moreover, if these two subtasks are next to each other; i.e., $l = k + 1$, then at least one operand of subtask $TB[l]$ comes directly from the result or operand of subtask $TB[k]$ without accessing the GDR or the IDB. This obviously will save the communication time to access global memories, and the subtask $TB[l]$ will never be a DTR subtask, thus minimizing the number of DTR subtasks.

Definition 2. A subtask in the kth row of the simplified task table $TB[k]$ is called a *double transmission required* (DTR) subtask if the following two conditions are satisfied:

(i) Its operand is one of these types:
$OP[k] = TB[m] \circ TB[n]$ for some m, $n < k$ and $m \neq n$.
$OP[k] = TB^i[m] \circ TB[n]$ for some m, $n < k$ and $m \neq n$.
$OP[k] = TB[m] \circ TB^j[n]$ for some m, $n < k$ and $m \neq n$.
$OP[k] = TB^i[m] \circ TB^j[n]$ for some m, $n < k$ and $m \neq n$.
$OP[k] = I[m] \circ I[n]$ for $m \neq n$.

(ii) $k = 1$ or $TB[k-1]$ is not a neighborhood of $TB[k]$ for $k > 1$.

Notice that for $OP[k] = TB[m] \circ I[n]$ and $OP[k] = TB^i[m] \circ I[n]$, the subtask $OP[k]$ is not a DTR subtask because its two operands can be transmitted simultaneously through two different set of connection lines. Moreover, a subtask involves only one operand is obviously a non-DTR subtask. For example, in the simplified task table of the benchmark algorithm, subtasks T_7, T_9, T_{12}, T_{13}, T_{15}, and T_{17} are all DTR subtasks as indicated in Table 3.

From the above definition, whether a subtask is a DTR subtask depends on its "position" in the simplified task table. A DTR subtask can become a non-DTR subtask if it is moved to the place exactly behind its neighborhood. Since it is possible that the movement of a DTR subtask may introduce another new DTR subtask, this reordering process is desirable only when it complies with the precedence constraint of the original algorithm and the number of DTR subtasks in the reordered task table is less than that in the original table. This forms the *scheduling problem*; that is, to reorder the processing sequence of subtasks to reduce the number of DTR subtasks as far as possible without violating the precedence constraint of the original algorithm. This reordering process can be performed by the following efficient neighborhood scheduling algorithm.

Algorithm N-Scheduling (*Neighborhood Scheduling Algorithm*).
Input: Simplified Task Table with n rows (i.e., n subtasks).
Output: Reordered Task Table.

N1. [Main Loop] Check each subtask to see if it is a DTR subtask. If yes, try to change its position.
For $k = 1$ **step** 1 **until** n **do**

N2. [Check DTR]
Check if $TB[k]$ is a DTR subtask according to definition 2? If not, go to step N4.

N3. [Main Body] Try to change the position of a DTR subtask to make it into a non-DTR subtask.
If $OP[k] = (TB[m]$ or $TB^a[m]) \circ (TB[n]$ or $TB^b[n])$,

then let $i \leftarrow max(m, n)$;
else let $i \leftarrow 1$; {* $OP[k] = I[m] \circ I[n]$ *}
End {If}
While $i < k-1$ **do**

If $TB[i]$ is a neighborhood of $TB[k]$, **then**

If {$TB[i+1]$ is a DTR subtask} or {the insertion of $TB[k]$ between $TB[i]$ and $TB[i+1]$ will not make $TB[i+1]$ a DTR subtask},

then insert $TB[k]$ behind $TB[i]$ to make $TB[k]$ the new $(i+1)$th subtask;
go to step N4
End {If}
End {If}
Let $i \leftarrow i+1$;
End {While}

N4. Continue {main loop}
End {For}

END. {N-Scheduling}

As an example, the N-Scheduling algorithm is applied to the benchmark algorithm. The input is the simplified task table in Table 3, which has a total of 18 subtasks and six of them are DTR subtasks. After applying the N-Scheduling algorithm to this simplified task table, the reordered task table is produced as shown in Table 3, in which all the DTR subtasks in the simplified task table have been removed.

5.3. Mapping Procedure

The reordered task table produced by the N-Scheduling algorithm can be mapped onto the proposed DN-SIMD machine in a rather straightforward way because these subtasks are all single-step, simple subtasks. If the subtasks are macro steps, then their mapping requires further consideration. Our mapping procedure at this final stage consists of two phases. In the first phase, the subtasks including single steps and macro steps which are viewed as single steps temporarily are mapped onto the DN-SIMD machine in a row directly. The actual mapping of the macro steps is considered in the second phase. The output of the mapping procedure is a control table as shown in Table 4. This table consists of ten columns and indicates the exact movement of the central control unit. The first column represents the identification of subtasks appearing in processing order. It also represents the result of the corresponding subtask. The second column indicates the first operand; it may be T_j (T_j^i, $\bar{T}_j$, $\vec{T}_j$) or I_j for some i. The third column indicates the source of the first operand, and there are five possibilities: the GDR, the IDB, the IFD, the IWR and the OWR within the PE. The fourth column describes which network is used (RIN1 or RIN2) and the required connection type on it to transmit the first operand if necessary. Columns 5 to 7 contain the

same information as the previous three columns, but for the second operand if it exists. Column 8 indicates the operation performed in this subtask. Column 9 indicates the destination of the result; it may be the GDR, the IFD, or both. If the IFD is needed, the connection type of network RIN2 is specified. Column 10 contains some comment on this subtask. For a macro subtask, these columns possess somewhat different meanings. Columns 2-7 indicate the corresponding information for the initial conditions of the macro subtask (similar to the parameters for a subroutine in a serial program). Columns 9-10 indicate the corresponding information for the final result of the macro subtask (similar to the return values of a subroutine in a serial program).

At the end of phase 1 of the mapping procedure, the control table of the benchmark algorithm is obtained as shown in Table 4. Since there are three macro subtasks in the control table, further mapping must be performed in phase 2. Among these macro subtasks, $\bar{T}_1$ and $\bar{T}_6$ are the HLR equations, and $\bar{T}_{11}$ is the HHLR equation. The mapping of HLR equations are demonstrated next.

The first-order homogeneous linear recurrence equation is defined as: Given $x(0) = a(0) =$ null, and $a(i)$, $1 \le i \le n$, find all the $x(i)$ for $1 \le i \le n$ from the following recursive equation

$$x(i) = x(i-1) \circ a(i) . \tag{6}$$

An efficient technique called the recursive doubling technique has been found to solve this recursive equation efficiently on an SIMD machine [7,19]. Using this technique, the parallel algorithm to solve Eq. (6) and the mapping diagram of this algorithm onto the proposed DN-SIMD machine are shown in Fig. 3. This diagram possesses the same information as a control table including the sources of operands, destination of result, network used and required connection types for each iteration. It takes an order of $O(\lceil \log_2(n+1) \rceil)$ iterations to produce the final results. Also notice that, in Fig. 3, we assume that the initial conditions $a(i)$'s come from the IDB. In fact, they may also come from the GDR depending on whether $a(i)$'s are external input variables or not. In that case, its mapping diagram is exactly the same except that the $a(i)$'s are from the GDR through the network RIN1 at the beginning. Similarly, the final results $x(i)$'s can be stored in the GDR or directly fedback to PEs depending on the necessity of the next subtask. Using the similar techniques, the mapping of HHLR equations can also be performed [2].

6. Conclusions

To design a global architecture for a set of parallel robotics algorithms, the characteristics of these algorithms are identified according to six fundamental features: degree of parallelism, uniformity of operations, fundamental operations, data dependency, and communication requirements. Considering the characteristics matching between the common features of the robotics algorithms and the architecture features. a medium-grained, DN-SIMD machine is designed. It consists of two sets of reconfigurable interconnection networks. One provides the communication between the PEs and the GDRs. The other provides the internal direct feedback paths among PEs to avoid unnecessary data storing and routing time. This machine performs three-stage pipelined operations and is synchronized at each basic mathematical calculation.

With the parallel robotics algorithms and the proposed DN-SIMD parallel machine, a systematic mapping procedure to schedule the subtasks of the parallel algorithms onto the parallel architecture is developed. This mapping procedure consists of three stages. At the first stage, mathematical decomposition is performed on the parallel algorithms to achieve a series of subtasks and each subtask is either in the basic mathematical form which involves at most two operands, or a well-structured macro subtask such as the linear recurrence equations. At the second stage, to shorten the communication time, the processing sequence of subtasks is reordered to minimize the total number of DTR subtasks using the Neighborhood Scheduling algorithm. At the final stage, the reordered subtasks are mapped onto the DN-SIMD machine. In this process, the single-step subtasks can be mapped directly, while the macro-step subtasks need further design and special technique such as the recursive doubling technique for solving the linear recurrence equations. A benchmark algorithm was used throughout as an example to illustrate the mapping procedure.

7. References

[1] L. H. Jamieson, "Characterizing Parallel Algorithms," in *The Characteristics of Parallel Algorithms*, L. H. Jamieson et al. (Eds.), The MIT Press, 1987.

[2] C. T. Lin, "Parallel Algorithms and Reconfigurable Architecture for Robotics Computations," MSEE Thesis, School of Electrical Engineering, Purdue University, West Lafayette, IN, August 1989.

[3] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul, "On-line Computational Scheme for Mechanical Manipulator," *Trans. ASME J. Dynam. Syst., Meas. Contr.*, Vol. 102, pp. 69-76, June 1980.

[4] C. S. G. Lee, T. N. Mudge, and J. L. Turney, "Hierarchical Control Structure Using Special Purpose Processor for the Control of Robot Arm," *Proc. 1982 Conf. Patt. Recog. and Image Processing*, Las Vegas, Nevada, pp.

634-640, June 14-17, 1982.

[5] R. Nigam, C. S. G. Lee, "A Multiprocessor-Based Controller for the Control of Mechanical Manipulators," *IEEE J. of Robotics and Automation*, Vol. RA-1, No. 4, pp. 173-182, Dec. 1985.

[6] L. Lathrop, "Parallelism in Manipulator Dynamics," *Int'l J. of Robotics Res.*, Vol. 4, No. 2, pp. 80-102, Summer 1985.

[7] C. S. G. Lee and P. R. Chang, "Efficient Parallel Algorithm for Robot Inverse Dynamics Computation," *IEEE Trans. on Syst. Man. Cybern.*, Vol. SMC-16, No. 4, pp. 532-542, July/Aug. 1986.

[8] J. Y. S. Luh and C. S. Lin, "Scheduling of Parallel Computation for a Computer-controlled Mechanical Manipulator," *IEEE Trans. on Syst. Man. Cybern.*, Vol. SMC-12, No. 2, pp. 214-234, March 1982.

[9] H. Kasahara, and S. Narita, "Parallel Processing of Robot-arm Control Computation on a Multimicroprocessor System," *IEEE J. of Robotics and Automation*, Vol. RA-1, No. 2, pp. 104-113, June 1985.

[10] C. L. Chen, C. S. G. Lee, and E. S. H. Hou, "Efficient Scheduling Algorithms for Robot Inverse Dynamics Computation on a Multiprocessor System," *IEEE Trans. on Syst. Man. Cybern.*, Vol. SMC-18, No. 5, pp. 729-743, September/October 1988.

[11] C. S. G. Lee and C. L. Chen, "Efficient Mapping Algorithms for Scheduling Robot Inverse Dynamics Computation on a Multiprocessor System," to appear in *IEEE Trans. on Syst. Man. Cybern.*

[12] M. W. Walker and D. E. Orin, "Efficient Dynamic Computer Simulation of Robot Mechanisms," *Trans. ASME J. Dynam. Syst. Meas. and Contr.*, Vol. 104, pp. 205-211, Sept. 1982.

[13] R. Featherstone, "The Calculation of Robot Dynamics Using Articulated-body Inertia," *Int. J. Robotics Res.*, Vol. 2, No. 1, pp. 13-30, Spring 1983.

[14] C. S. G. Lee and P. R. Chang, "Efficient Parallel Algorithms for Robot Forward Dynamics Computation," *IEEE Trans. on Syst. Man. Cybern.*, Vol. SMC-18, No. 2, pp. 238-251, Mar./Apr. 1988.

[15] M. Amin-Javaheri and D. E. Orin, "A Systolic Architecture for Computation of the Manipulator Inertia Matrix," *Proc. of 1987 IEEE Int'l Conf. on Robotics and Automation*, Raleigh, North Carolina, pp. 647-653, March 30-April 3, 1987.

[16] A. Fijany and A. K. Bejczy, "An Efficient Method for Computing the Manipulator Inertia Matrix," *2nd Int. Symp. on Robotics and Manufacturing Research*, Albuqurque, Nov., 1988.

[17] J. Denavit and R. B. Hartenberg, "A Kinematic Notation for Lower-pair Mechanisms based on Matrices,"

[18] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*, New York: McGraw-Hill, 1987.

[19] P. M. Kogge and H. S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," *IEEE Trans. on Computer*, Vol. c-22, pp. 789-793, Aug. 1973.

[20] A. Fijany and J. G. Pontnau, "Parallel Computation of the Jacobian for Robot Manipulators," *Proc. IASTED*, Santa Barbara, May 1987.

[21] D. E. Orin and W. W. Schrader, "Efficient Computation of the Jacobian for Robot Manipulators," *the Int. J. of Robotics Research*, Vol. 3, No. 4, pp. 66-75, Winter 1984.

[22] T. B. Yeung and C. S. G. Lee, "Efficient Parallel Algorithms and VLSI Architectures for Manipulator Jacobian Computation," *IEEE Trans. on Syst. Man. Cybern.*, Vol. SMC-19, No. 5, September/October 1989.

[23] P. R. Chang and C. S. G. Lee, "Residue Arithmetic VLSI Array Architecture for Manipulator Pseudo-Inverse Jacobian Computation," *IEEE Trans. on Robotics and Automation*, Vol. RA-5, No. 5, October 1989.

[24] R. Featherstone, "Position and Velocity Transformations Between Robot End-effector Coordinates and Joint Angles," *the Int. J. of Robotics Research*, Vol. 2, No. 2, Summer 1983, pp. 35-45.

[25] R. P. Paul, B. E. Shimano, and G. Mayer, "Kinematic Control Equations for Simple Manipulators," *IEEE Trans. on Syst. Man. Cybern.*, Vol. SMC-11, pp. 494-455, 1981.

[26] Y. T. Tsai and D. E. Orin, "A Strictly Convergent Real-time Solution for Inverse Kinematics of Robot Manipulators," *J. of Robotic Systems*, Vol. 4, No. 4, pp. 477-501, 1987.

[27] C. S. G. Lee and P. R. Chang, "A Maximum Pipelined CORDIC Architecture for Robot Inverse Kinematic Position Computation," *IEEE J. Robotics and Automation*, Vol. RA-3, No. 5, pp. 445-458, Oct. 1987.

Table 1. Characteristics of Basic Robotics Algorithms.

Algorithms	CHARACTERISTICS					
	Type of Parallelism	Degree of Parallelism	Uniformity of Operations	Fundamental Operations	Data Dependency	Communication Requirement
Inverse Dynamics	Job level	Large grain	Yes	Matrix-Vector	HLR	(Regular) Permutation
Forward Dynamics	Job level	Large grain	Yes	Scalar ops. Reciprocal Matrix-Vector	HLR,HHLR SHLR, PNE System of Linear Eqs.	one-to-one Permutation Broadcast
Forward Kinematics	Job level	Medium or Fine grain	Yes	Matrix Mult. Trigonometric	HLR	(Regular) Permutation
Forward Jacobian	Job level	Medium or Fine grain	Yes	Matrix-Vector	HLR (Forward & Backward)	(Irregular) Permutation Broadcast
Inverse Jacobian (Direct)	Job level	Medium or Fine grain	Yes	Scalar ops. Reciprocal Vector ops.	Global	Permutation Broadcast
Inverse Jacobian (iterative)	Job level	Medium or Fine grain	Yes	Scalar ops. Reciprocal Vector ops.	Local	Permutation Broadcast
Inverse Kinematics (Direct)	Task level	Fine grain	No	Scalar ops. Reciprocal Square root Trigonometric	Global	one-to-one Broadcast
Inverse Kinematics (iterative)	Job level	Medium or Fine grain	Yes	Scalar ops. Reciprocal Matrix-Vector Trigonometric	Local	one-to-one Permutation Broadcast

Table 2. Robotics Benchmark Algorithm and Subtask Assignment.

G_i	Equations	T_i	Subtasks
G_1	$A[i]=A[i-1]\times B[i]$, $2\le i\le n$, $A[1]=B[1]$	$\bar{T}_1$	$\bar{T}_1[i]=\bar{T}_1[i-1]\times I_1[i]$, $2\le i\le n$, $\bar{T}_1[1]=I_1[1]$
G_2	$C[i]=A[i]\cdot D[i]$	T_2	$T_2[i]=\bar{T}_1[i]\cdot I_2[i]$
G_3	$E[i]=A[i]\cdot F[i]$	T_3	$T_3[i]=\bar{T}_1[i]\cdot I_3[i]$
G_4	$G[i]=G[i-1]+(E[i]H[i])J[i]$, $2\le i\le n$	T_4	$T_4[i]=T_3[i]I_4[i]$
	$G[1]=(E[1]H[1])J[1]$	T_5	$T_5[i]=T_4[i]I_5[i]$
		$\bar{T}_6$	$\bar{T}_6[i]=\bar{T}_6[i-1]+T_5[i]$, $2\le i\le n$, $\bar{T}_6[1]=T_5[1]$
G_5	$K[i]=L[i](C[i]+E[i])+G[i]+C[i]$	T_7	$T_7[i]=T_2[i]+T_3[i]$
	$M[i]=N[i]M[i-1]+K[i]$, $2\le i\le n$	T_8	$T_8[i]=T_7[i]I_6[i]$
	$M[1]=K[1]$	T_9	$T_9[i]=\bar{T}_6[i]+T_2[i]$
		T_{10}	$T_{10}[i]=T_8[i]+T_9[i]$
		$\bar{T}_{11}$	$\bar{T}_{11}[i]=I_7[i]\bar{T}_{11}[i-1]+T_{10}[i]$, $1\le i<n$, $\bar{T}_{11}[n]=T_{10}[n]$
G_6	$O[i]=(E[i-1]+M[i])+(G[i]+K[i+2])$	T_{12}	$T_{12}[i]=\bar{T}_6[i]+T_{10}[i+2]$
	$P[i]=A[i]\cdot G[i]$	T_{13}	$T_{13}[i]=T_3[i-1]+\bar{T}_{11}[i]$
	$Q[i]=M[i]\cdot C[i+1]+O[i+1]\cdot P[i]$	T_{14}	$T_{14}[i]=T_{12}[i]+T_{13}[i]$
		T_{15}	$T_{15}[i]=\bar{T}_1[i]\cdot\bar{T}_6[i]$
		T_{16}	$T_{16}[i]=T_{14}[i+1]\cdot T_{15}[i]$
		T_{17}	$T_{17}[i]=T_2[i+1]\cdot\bar{T}_{11}[i]$
		T_{18}	$T_{18}[i]=T_{16}[i]+T_{17}[i]$

where in this table, except separate indication, i is from 1 to n.
$B[i]$, $D[i]$, $F[i]$, $H[i]$, $J[i]$, $L[i]$, $N[i]$; $i=1,\ldots,n$; are assumed to be input variables.
$B[i]$ is a 3×3 matrix. $D[i]$, $F[i]$ are 3×1 vectors.
$H[i]$, $J[i]$, $L[i]$, $N[i]$ are all scalars.

Table 3. Simplified and Reordered Task Table of Benchmark Algorithm.

	SIMPLIFIED TASK TABLE			REORDERED TASK TABLE		
ROW	TB[ROW]	OP[ROW]	DTR	TB[ROW]	OP[ROW]	DTR
1	$\bar{T}_1$	$\bar{T}_1^{-1} \times I_1$		$\bar{T}_1$	$\bar{T}_1^{-1} \times I_1$	
2	T_2	$\bar{T}_1 \cdot I_2$		T_2	$\bar{T}_1 \cdot I_2$	
3	T_3	$\bar{T}_1 \cdot I_3$		T_3	$\bar{T}_1 \cdot I_3$	
4	T_4	$T_3 I_4$		T_7	T_2+T_3	
5	T_5	$T_4 I_5$		T_4	$T_3 I_4$	
6	$\bar{T}_6$	$\bar{T}_6^{-1}+T_5$		T_5	$T_4 I_5$	
7	T_7	T_2+T_3	×	$\bar{T}_6$	$\bar{T}_6^{-1}+T_5$	
8	T_8	$T_7 I_6$		T_{15}	$\bar{T}_1 \cdot \bar{T}_6$	
9	T_9	$\bar{T}_6+T_2$	×	T_9	$\bar{T}_6+T_2$	
10	T_{10}	T_8+T_9		T_8	$T_7 I_6$	
11	$\bar{T}_{11}$	$I_7 \bar{T}_{11}^{-1}+T_{10}$		T_{10}	T_8+T_9	
12	T_{12}	$\bar{T}_6+T_{10}^{+2}$	×	T_{12}	$\bar{T}_6+T_{10}^{+2}$	
13	T_{13}	$T_3^{-1}+\bar{T}_{11}$	×	$\bar{T}_{11}$	$I_7 \bar{T}_{11}^{-1}+T_{10}$	
14	T_{14}	$T_{12}+T_{13}$		T_{17}	$T_2^{+1} \cdot \bar{T}_{11}$	
15	T_{15}	$\bar{T}_1 \cdot \bar{T}_6$	×	T_{13}	$T_3^{-1}+\bar{T}_{11}$	
16	T_{16}	$T_{14}^{+1} \cdot T_{15}$		T_{14}	$T_{12}+T_{13}$	
17	T_{17}	$T_2^{+1} \cdot \bar{T}_{11}$	×	T_{16}	$T_{14}^{+1} \cdot T_{15}$	
18	T_{18}	$T_{16}+T_{17}$		T_{18}	$T_{16}+T_{17}$	

where T_1 and T_6 are HLR equations and T_{11} is an HHLR equation.

Table 4. Control Table for Benchmark Algorithm.

1	2	3	4	5	6	7	8	9			10
T_i	Operand 1	Source 1	Network Type	Operand 2	Source 2	Network Type	Operation	Output Destination			Comment
								GDR	IFD	RIN2	
$\bar{T}_1$	$\bar{T}_1^{-1}$	IFD	*	I_1	IDB	-	MM	×		-	* HLR Eqn.
T_2	$\bar{T}_1$	OWR	-	I_2	IDB	-	MV	×		-	
T_3	$\bar{T}_1$	IWR	-	I_3	IDB	-	MV	×		-	
T_7	T_3	OWR	-	T_2	GDR	RIN1-1	VA	×		-	
T_4	T_3	IWR	-	I_4	IDB	-	SV			-	
T_5	T_4	OWR	-	I_5	IDB	-	SV			-	
$\bar{T}_6$	$\bar{T}_6^{-1}$	IFD	*	T_5	OWR	-	VA	×		-	* HLR Eqn.
T_{15}	$\bar{T}_6$	OWR	-	$\bar{T}_1$	GDR	RIN1-1	MV	×		-	
T_9	$\bar{T}_6$	IWR	-	T_2	GDR	RIN1-1	VA	×		-	
T_8	T_7	GDR	RIN1-1	I_6	IDB	-	SV			-	
T_{10}	T_8	OWR	-	T_9	GDR	RIN1-1	VA	×	×	2	
T_{12}	T_{10}^2	IFD	RIN2-2	$\bar{T}_6$	GDR	RIN1-1	VA	×		-	
$\bar{T}_{11}$	I_7	IDB	-	*	*	*	*			-	* HHLR Eqn.
T_{17}	$\bar{T}_{11}$	OWR	-	T_2^{+1}	GDR	RIN1-3	VI	×		-	
T_{13}	$\bar{T}_{11}$	IWR	-	T_3^{-1}	GDR	RIN1-4	VA			-	
T_{14}	T_{13}	OWR	-	T_{12}	GDR	RIN1-1	VA		×	3	
T_{16}	T_{14}^{+1}	IFD	RIN2-3	T_{15}	GDR	RIN1-1	VI			-	
T_{18}	T_{16}	OWR	-	T_{17}	GDR	RIN1-1	SA	×		-	Result

Connection type 1: straight connection;
Connection type 2: uniform module shift ($d = 2$);
Connection type 3: uniform module shift ($d = 1$)
Connection type 4: uniform module shift ($d = -1$)

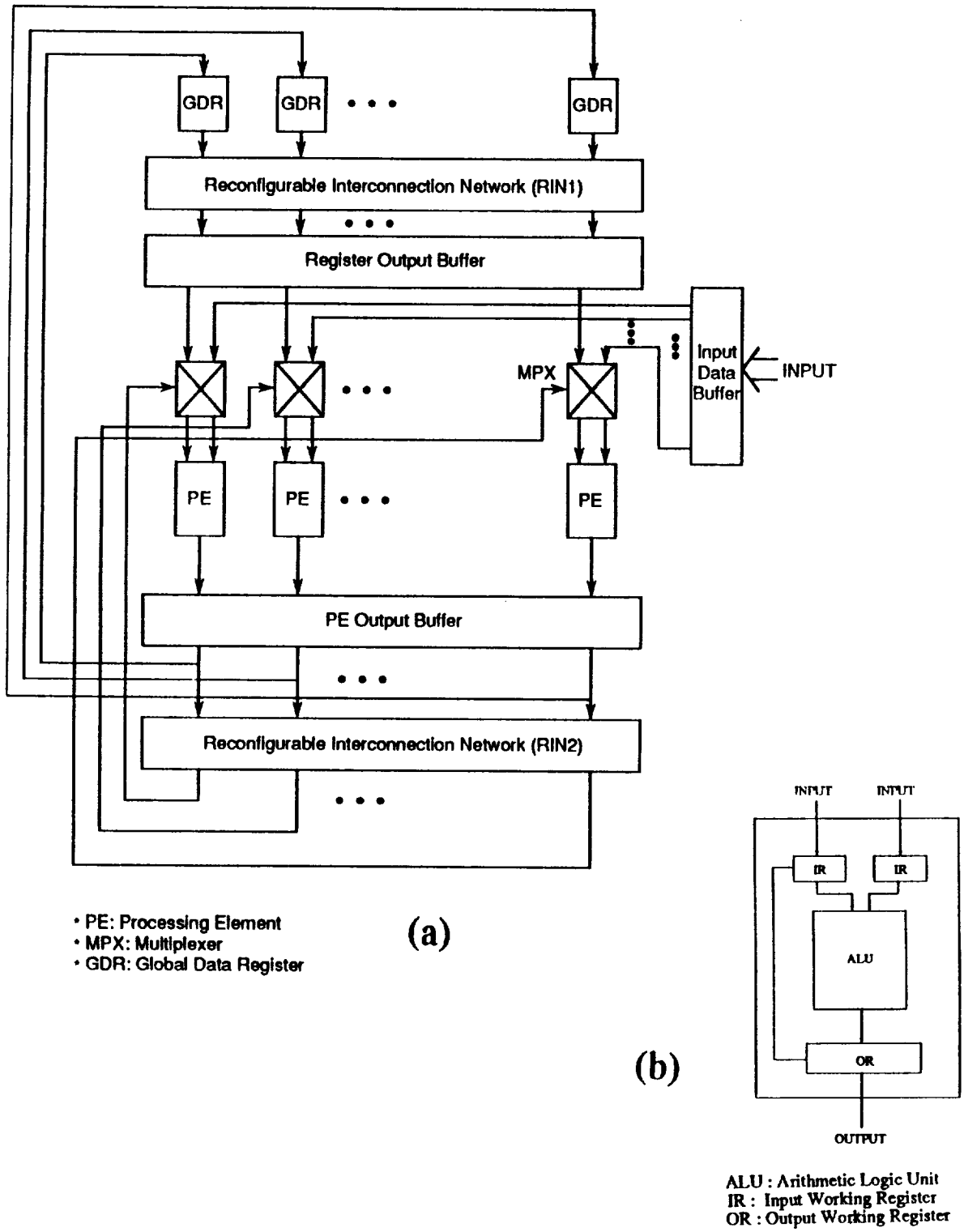
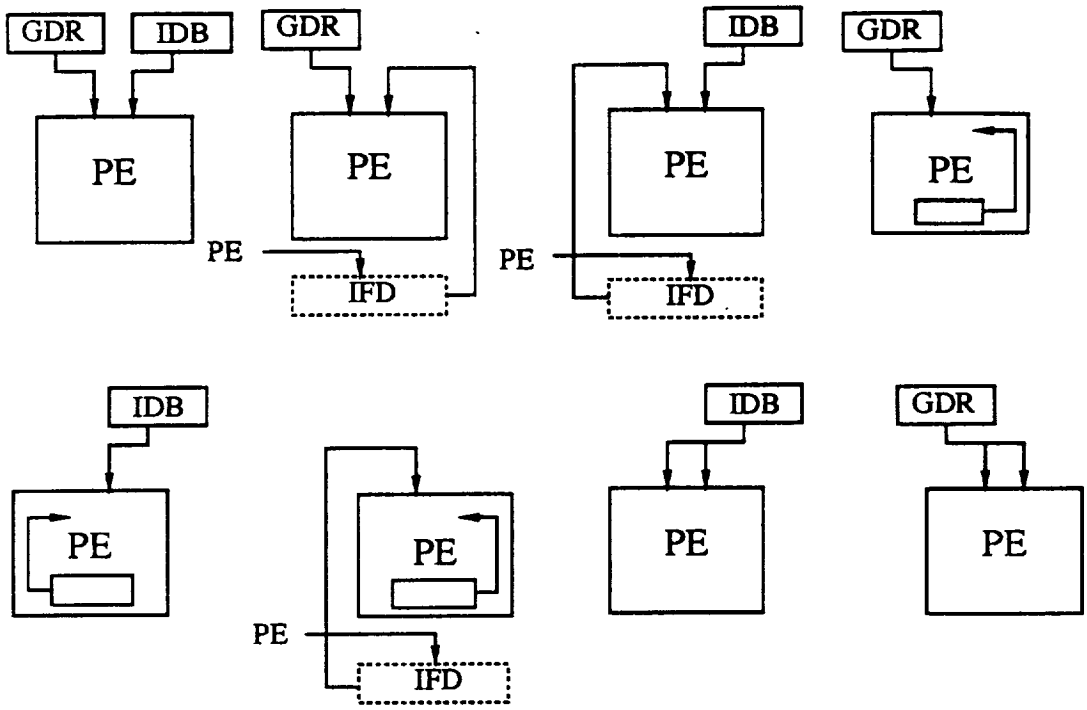
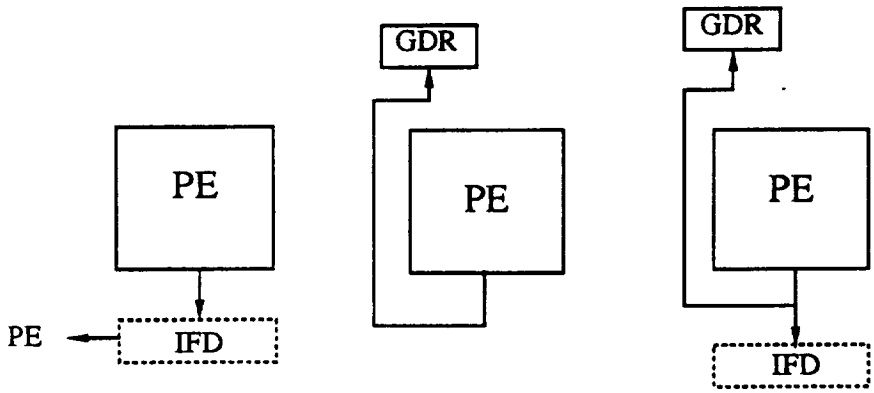


Figure 1. (a) Structure of Dual Network SIMD Machine.
(b) The Structure of Processing Element.



(a) Input Operands Flow Diagrams



(b) Output Result Flow Diagrams

Figure 2. Data Path Flow of Processing Element.

Algorithm FOHRA (*First-Order Homogeneous Recurrence Algorithm*).

F1. [Initialization] Given the terms a_i, $0 \le i \le n$, let $X^{(k)}(i)$ be the ith sequence at the kth splitting and $s = \lceil \log_2(n+1) \rceil$. Set the sequence at the initial step, $X^{(0)}(i) \leftarrow a_i$, $0 \le i \le n$.

F2. [Compute x_i parallelly]
for $k \leftarrow 1$ **to** s, **do**

$$X^{(k)}(i) = \begin{cases} X^{(k-1)}(i-2^{k-1}) * X^{(k-1)}(i) & , \text{ if } 2^{k-1} \le i \le n \\ X^{(k-1)}(i) & , \text{ if } 0 \le i < 2^{k-1} \end{cases}$$

end {for}

Set $x_i \leftarrow X^{(s)}(i)$, $1 \le i \le n$.

END FOHRA.

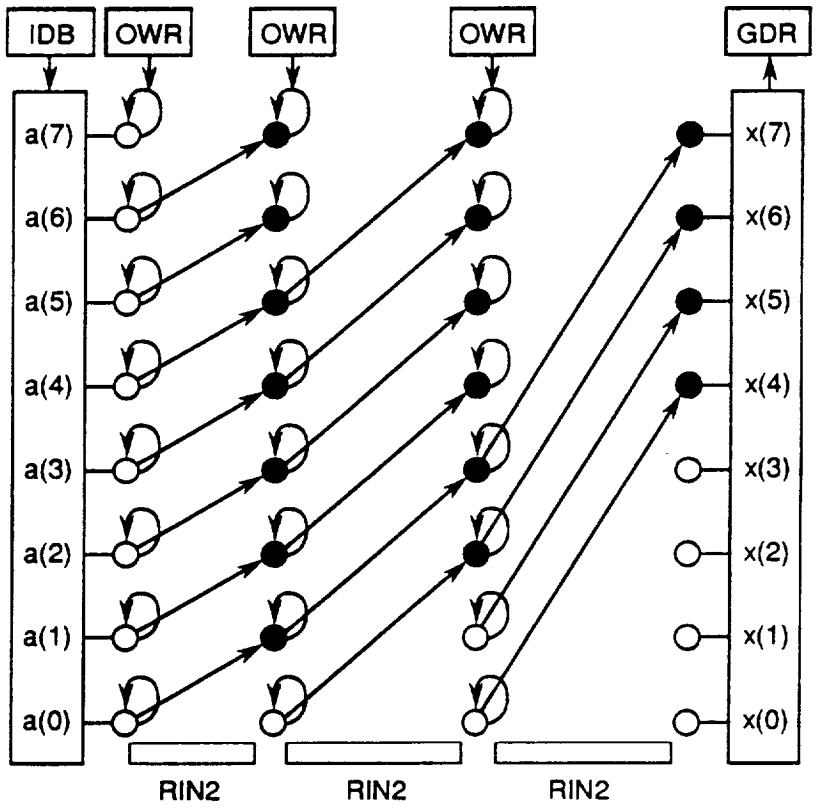


Figure 3. Mapping Diagram of First-Order Homogeneous Linear Recurrence Equations on DN-SIMD Machine.