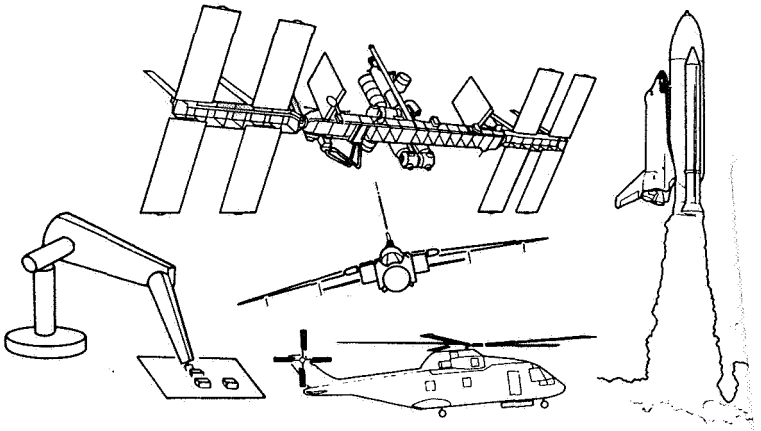
JPL Publication 93-02

Proceedings of the Fifth NASA/NSF/DOD Workshop on Aerospace Computational Control

M. Wette
G. K. Man
Editors



February 15, 1993

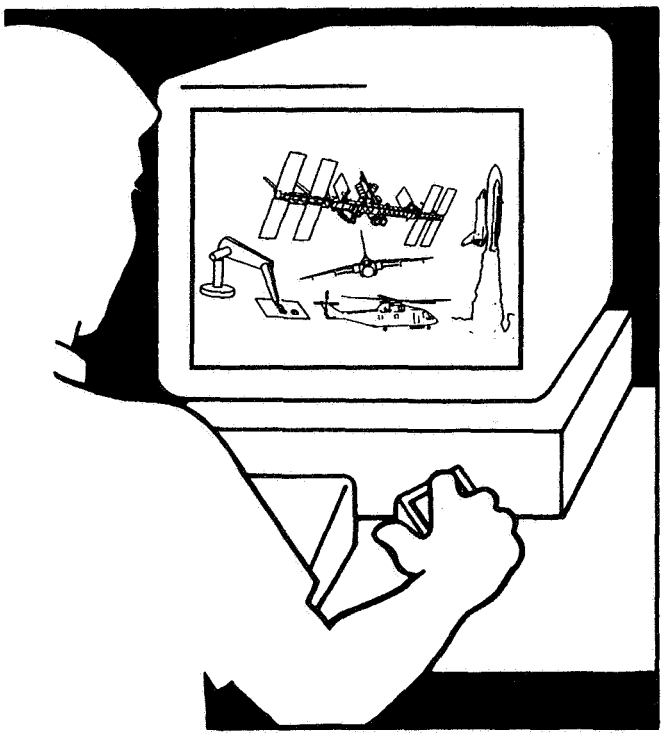
NASA
National Aeronautics and Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

JPL Publication 93-02

Proceedings of the Fifth NASA/NSF/DOD Workshop on Aerospace Computational Control

M. Wette
G. K. Man
Editors



February 15, 1993

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

Abstract

Introduction

The Fifth Annual Workshop on Aerospace Computational Control was one in a series of workshops sponsored by NASA, NSF and the DOD. The purpose of these workshops is to address computational issues in the analysis, design and testing of flexible multibody control systems for aerospace applications. The effort began with the 1987 Workshop on Multibody Simulation in Arcadia, California, and was followed by the 1988 Workshop on Computational Aspects in the Control of Flexible Systems in Williamsburg, Virginia. The 3rd Annual Conference on Aerospace Computational Control was held in Oxnard, California, and the 4th NASA Workshop on Computational Control of Flexible Aerospace Systems in Williamsburg, Virginia. The intention in holding these workshops is to bring together users, researchers and developers of computational tools in aerospace systems (spacecraft, space robotics, aerospace transportation vehicles, etc.) for the purpose of exchanging ideas on the state of the art in computational tools and techniques.

The Fifth Workshop, held in Santa Barbara, California, provided the attendees a window into current research, development and applications in computational issues related to the development of flexible multibody systems. The objectives of the workshop were to provide

- NASA, NSF and DOD a window into current research
- tool users and tool developers an opportunity to exchange ideas and experience on the correctness, efficiency, and usability of current computational tools
- researchers and users an opportunity to see what areas are ripe for application and what areas require more effort
- academia and industry an opportunity to strengthen cooperation in order to aid the transfer of new technologies to applications

Attendees presented their current work and discussed their views on what directions should be taken. The focus of the workshop was in the following areas:

Modeling: component model representation, multi-flexible body modeling, robotics modeling, model reduction, model verification, modal synthesis, damping, algorithms & software

Flexible Multibody Simulation: recursive and non-recursive algorithms, real-time simulation, symbolic algorithms, software

CAE Issues: application-specific tools, tool integration, database systems, user interfaces, visualization, algorithms & software

Control System Design: analysis and design tools and techniques, large-scale problems, numerical algorithms, numerical analysis, software

Testing and Verification: techniques, software systems, hardware systems, ground test systems, flight experiments

Applications: real-time hardware-in-the-loop test implementations, graphics and animation, experience with design tools, testbeds

The Computational Control Program

The Fifth Annual Workshop on Aerospace Computational Control was sponsored in part by NASA's Computational Control Program, which is currently implemented at five NASA centers (the Goddard Space Flight Center, the Jet Propulsion Laboratory, the Johnson Space Center, the Marshall Space Flight Center, and the Langley Research Center). The program was started in 1987 after observers found that the limitations of current tools could possibly jeopardize future NASA flight projects. The objectives of this program are to develop a new generation of algorithms and prototype software tools for the design and testing of spacecraft and robots. The emphasis of this program is to provide technology which will provide the capability to handle the more demanding requirements of future missions while enhancing productivity and reducing risk. Within this program several existing codes (e.g., TREETOPS, DISCOS) have been validated and enhanced and several new tools have been developed (e.g., DARTS, Caesy, COMPARE, Space Station Workstation). These tools are finding acceptance in wide-ranging projects both inside and outside of NASA.

Discussion

The Workshop was concluded with a lively panel discussion. The purpose of this panel discussion was to allow participants to access the current state-of-the-art and new directions for future work.

It was pointed out that much progress has been made in the modeling area. Researchers are now aware that many valid techniques exist to derive the equations of motion. Work on the Component Model Representation technique is continuing; the technique becomes very important in situations when spacecraft is being designed, built and verified in terms of subsystems.

There was quite a bit of discussion on the need for new emphasis in modeling. First, it was pointed out that damping should be considered more in design. Hence, the modeling of damping should be stressed in the design process as well as in the design tools. Second, more attention should be paid to the relationship between model fidelity, validation and simulation. Higher fidelity models may be more prone to modeling errors due to the complexity involved in the modeling process. Also, higher fidelity models, with numerous high-frequency modes, may cause problems with algorithms for integrating the systems of differential equations. In effect, one should only model what is necessary – for example, what lies within the sensor bandwidth. In general, the modeling process should be more "in tune" with the end application whether it be simulation (where numerical "stiffness" is a concern), control system design (where knowledge of the model uncertainties is important) or whatever. Finally, a comment was made that with parallel computing, the computational bottleneck is no longer a problem and hence model reduction should no longer be a concern. There was some disagreement with this argument.

There has been remarkable progress in real-time simulation. The emphasis on derivation of equations has given way to the age of parallel processing. Many of the papers presented showed that parallel processing is being used quite successfully in practice to provide multibody simulation in real time. It was mentioned that the development of algorithms should be integrated with the development of the algorithms for numerical integration. In essence, an algorithm must be found which matches the problem and the available computing resources. Also, with the advances made in real-time simulation, the need for better algorithms must still be established.

There has been some progress in computer-aided engineering. However, the remark was made that the processes of setting up, debugging and interpreting results from an analysis are still significant bottlenecks in the overall design and analysis process. Also, there exists a need for tools to transfer data between programs. Many users write custom programs for doing these chores.

Work should be done to develop highly functional and reliable tools for this task: this is a solvable problem.

In the control-system design area, it was mentioned that access to model uncertainty is an important factor in design. Another remark was made that the community should work on the establishment and development of benchmark problems for flexible multibody control problems. This would give students and engineers access to "real world" problems and would provide test cases for commercial tools.

It became clear from the discussion that the process of verification is not always a well-defined process. There is more work needed to aid in providing verification techniques for dynamic models, reduced-order models, model code, etc. Another interesting point was brought up concerning model validation. It is a fact that many models being used for dynamical systems (e.g., liquid fuel dynamics) can't be verified on the ground. A suggestion was made that future space missions should be designed to provide flight data for model identification and verification. Although validation of real-time simulation code against an analytical model is more tractable and more often used, it does not provide the same confidence that would be provided by validation against flight data.

Later discussions with attendees brought about the idea that with emphasis now on smaller, cheaper spacecraft, the challenge is to provide a design and test environment which supports rapid turnaround. This will involve tools for integrated prototype designs, allowing an engineer to analyze critical design trade-offs early in the design process.

Current Challenges

The talks and discussions at this workshop provided good inputs on requirements for continuing and future research and development. The current challenges for work in computational control include

Speed: The faster we can get simulations to run, the better off we are. Parallel and vector processing have paid off well, and work in this area should be continued.

Damping: We need to start incorporating damping into the modeling process. This implies that the associated modeling and design tools must be extended to handle damping.

Benchmark Problems: Benchmark problems (with associated data) should be developed and/or collected to provide problems to students as well as tool developers.

Computer-Aided Engineering: The "bridging tools" to provide the capability to process and interpret data efficiently are still lacking. We need some good tools here.

Concurrent Engineering: The process of designing, building and verifying systems needs to be speeded up. This is a new focus area that should receive new effort.

Much progress has been made over the past few years, but many challenges still remain. We hope the enthusiasm with which the community has approached these problems continues.

CONTENTS

FLEXIBLE MULTIBODY SIMULATION

TESTING, VERIFICATION , AND APPLICATIONS I

CONTROL SYSTEM DESIGN

TESTING, VERIFICATION, AND APPLICATIONS II

Geometric Stiffening in Multibody Dynamics Formulations

Inna Sharf*
Department of Mechanical Engineering
University of Victoria
Victoria, British Columbia, Canada
V8W 3P6

Abstract

In this paper we discuss the issue of geometric stiffening as it arises in the context of multibody dynamics. This topic has been treated in a number of previous publications in this journal and appears to be a debated subject. The controversy revolves primarily around the "correct" methodology for incorporating the stiffening effect into dynamics formulations. The main goal of this work is to present the different approaches that have been developed for this problem through an in-depth review of several publications dealing with this subject. This is done with the goal of contributing to a precise understanding of the existing methodologies for modelling the stiffening effects in multibody systems. Thus, in presenting the material we attempt to illuminate the key characteristics of the various methods as well as as show how they relate to each other. In addition, we offer a number of novel insights and clarifying interpretations of these schemes. The paper is completed with a general classification and comparison of the different approaches.

1 Introduction

The issue of geometric stiffening, also referred to as dynamic stiffening, centrifugal stiffening and foreshortening has been a topic of many recent publications dealing with the dynamics of flexible bodies for applications to multibody systems. Kane *et al.*[1] first observed that the majority of existing multibody dynamics formulations and accordingly the dynamics simulation packages do not incorporate the geometric stiffening effect. They have attributed this flaw to the "convenional approach" for describing the deformation of elastic bodies, which yields a set of dynamics equations that inherently lack the geometric stiffening terms. Kane *et al.* proposed an alternative approach, correcting this flaw, and applied it to develop a set of equations for the deformation of a beam attached to a moving base.

Eke and Laskin[2] took up the issue raised in Ref. 1 and investigated regimes of validity of existing formulations with the simulation package DISCOS on a spin-up beam example. They qualified the error in conventional approach as a "premature linearization" of the displacement field. This was later supported by Padilla and von Flotow[3] and Banerjee and Dickens[4].

Shortly after Kane *et al.*'s publication, two commentaries appeared on the material presented in Ref. 1. In particular, London[5] pointed out that geometric stiffening has been

*Assistant Professor

previously considered by many researchers in a number of applications. Furthermore, he observed that several approaches have been employed to include this effect in the dynamics equations and compiled a table characterizing the various methods. In a technical note,[6] Hanagud and Sarkar state that, contrary to the claim made in Ref. 1, the conventional method for modelling the kinematics of elastic deformation *can* be used.

The existing controversy over the nature of geometric stiffening, the debate on " the correct" approach to model it and the seeming incongruity of the methods used to include the effect in the motion equations—all of these have motivated us to review several of the works on this subject. In doing so, we have attempted to understand precisely how geometric stiffening is incorporated into the dynamics equations in different approaches, what assumptions and approximations are made in the derivation, what motivated these and whether they are justified. This paper contains the main results of our review.

Our starting point will be the landmark paper by Kane *et al.*[1] and the subsequent commentaries.[5,6] Following that, we give a thorough treatment of the works by Likins *et al.*,[7] Vigneron[8] and Kaza and Kvaternik[9] and a summary of the relevant material from the publications by Lips and Modi[10] and Hughes and Fung.[11] Section 4 contains the main results from Laskin *et al.*,[12] Meirovitch[13,14] and Banerjee *et al.*.[4,15] In reviewing the works of these researchers, we do not simply repeat their derivations, nor do we include the dynamics equations developed in these publications. Instead, we concentrate on the fundamental assumptions made in formulating the basic elements necessary for deriving these equations, where the "formulation" ends when the development becomes a purely mechanical process. For instance, in the cases where dynamics equations are derived *via* Hamilton's principle, we limit ourselves to stating kinetic and potential energy functions, and do not go through the procedure of applying the variational principle. This allows us to compare the various approaches based on the fundamental physical assumptions.

In addition to presenting the key features of different procedures, making comparisons and establishing relationships between them, we also provide clarifications and give some new insights. We conclude the paper with a discussion in which we disclose some of the existing misconceptions, classify the approaches and comment on their suitability for multibody dynamics simulation.

2 Kane *et al.* and Commentaries

2.1 Main Results of Kane, Ryan and Banerjee

In Ref. 1, Kane, Ryan and Banerjee develop the dynamics equations of a general flexible beam built into a rigid base. The base body can undergo arbitrary, but prescribed translational and rotational motion. The generality of the beam refers to the fact that its geometric and material properties are not assumed to be constant, but can vary along the length of the beam. In addition, Kane *et al.* do not make the common assumption that the elastic and centroidal axes coincide. As a result, their motion equations contain terms dependent on the components of the eccentricity vector, e_2 and e_3.

The formulation of equations in Ref. 1 differs from many existing procedures in several respects. First, it incorporates the effect of the transverse displacement on the axial displacement in the kinematic description of the deformation. This is achieved *indirectly* by expressing the distance along the deformed elastic axis as a nonlinear function of the transverse displacements

with:

$$x + s(x,t) = \int_0^{x+u_1} \left[1 + \left(\frac{\partial u_2}{\partial \sigma}\right)^2 + \left(\frac{\partial u_3}{\partial \sigma}\right)^2\right]^{1/2} d\sigma \tag{1}$$

The above equation is the same as Eq. (19) of Ref. 1, where the variable s denotes "the stretch in the beam along the elastic axis."

The second major difference relates to the choice of elastic deformations that are employed to describe kinematics of the deformed beam. The standard procedure is to use three orthogonal elastic displacements u_1, u_2 and u_3 to represent the displacement field in a deformable body. Kane *et al.* employ the *stretch* s with the transverse translations u_2 and u_3 as a set of generalized coordinates. Thus, when discretizing the continuous displacement field, they discretize the stretch variable instead of the axial displacement u_1. This is expressed by Eqs. (25) and (26) of Ref. 1, which we rewrite here for convenience as:

$$\begin{aligned} s(x,t) &= \sum_{j=1}^{\nu} \phi_{1j}(x)\, q_j(t) \\ u_i(x,t) &= \sum_{j=1}^{\nu} \phi_{ij}(x)\, q_j(t), \qquad i = 2,3 \end{aligned} \tag{2}$$

Accordingly, the dynamics equations based on the above premise reprsent a model for the time-evolution of $\{s, u_2, u_3\}$, or rather, the corresponding discrete elastic coordinates. The "conventional approach" involves discretizing the orthogonal set of elastic displacements $\{u_1, u_2, u_3\}$ with:

$$u_i = \sum_{j=1}^{\nu} \phi_{ij}\, q_j, \qquad i = 1,2,3 \tag{3}$$

or in matrix form:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \mathbf{u} = \mathbf{\Phi q} \tag{4}$$

Kane *et al.* argue that in the standard procedure (3), the three elastic deformations cannot account for the fact that every transverse displacement gives rise to an axial displacement, because the form (3) inherently precludes such an interdependence.

The general methodology employed in Ref. 1 to derive an explicit (literal) set of motion equations for the elastic coordinates is that presented in Kane and Levinson.[16] The procedure requires one to construct the *generalized inertia* and *generalized active* forces. The former are developed in Ref. 1 according to the algorithm outlined by Kane and Levinson. The generalized *active* forces, which for the particular system considered result from internal forces, are derived from the strain energy function. The expression for this function U is given in terms of the components of the force and torque vectors which act on a cross-section of the beam (see Eq. (51) in Ref. 1). Thus, it takes the form of a sum of six integrals, three for each of the force and torque, where the integrand of each integral term is a *quadratic* function of the appropriate load. In order to determine the generalized internal force by using Eq. (50) of Ref. 1, which in fact is a statement of Castigliano's theorem, one needs to formulate the strain energy as a function of the generalized coordinates. To this end, Kane *et al.* express each of the six loads as a *linear* function of elastic deformations or their spatial derivatives.

For reasons that will become apparent in §2.4, we draw attention to one particular term in the strain energy function. This term represents the contribution of the axial load and will be denoted here by U_P. It corresponds to the first term in Eq. (51) of Ref. 1, which we rewrite, omitting the subscripts as:

$$U_P = \int_0^L \frac{P^2}{2EA}\, dx \tag{5}$$

In linear analysis, one approximates the axial load as a linear function of the axial displacement gradient with $P = EA\frac{\partial u_1}{\partial x}$. The expression for P given in Ref. 1 by the left Eq. (58) takes the same form, but with the axial translation u_1 replaced with the stretch variable s. With that, the axial load contribution to the strain energy function becomes:

$$U_P = \frac{1}{2}\int_0^L EA\left(\frac{\partial s}{\partial x}\right)^2 dx \tag{6}$$

The above are what we view as the key features of the dynamics formulation for a flexible beam attached to a moving base that has been put forward by Kane, Ryan and Banerjee.[1] In the following two subsections, we summarize the main points of a technical comment and an engineering note, both of which are related to Ref. 1. These appeared in two issues of the 1988 volume of the Journal of Guidance and Control, shortly after the publication of Kane *et al.* Section 2 is concluded with a discussion of the two commentaries.

2.2 London's Comments

In summarizing London's comments, we have grouped them into two categories. The first one includes comments which deal with the "qualitative" aspects of Kane, Ryan and Banerjee's work, such as literature review. In the second category, we include comments related to the quantitave or technical aspects of the analytical development presented in Ref. 1.

Category I. London observes that Kane *et al.* "create the impression that a new theory has been discovered," refering to the theory to model foreshortening. London bases this statement on the fact that Kane *et al.* do not give any references as to the origin of their results for the nonlinear description of the kinematics of the deformed beam (primarily, Eq. (19) in Ref. 1).

Regarding the literature review on the subject of modelling flexible beams attached to a moving base, London's criticisms are twofold. First, he suggests that the references in Kane *et al.*'s manuscript are incomplete. To be specific, they neglect to mention the work by Lips and Modi[10] on the dynamics of beams undergoing a three-axis spin, as well as the work by Hughes and Fung,[11] which treats the problem of stability of spinning satellites with flexible appendages. Second, London implies that many of the references that are included have not been given a proper and/or appropriate credit. In this regard, he particularly notes the work of Kaza and Kvaternik[9] which is classified by Kane *et al.* into a group of papers "in connection with aircraft dynamics" addressing "questions concerning tapered, twisted and rotating beams." Similar treatment is given to the works of Likins *et al.*[7] and Vigneron,[8] which are grouped under those in the field of "spacecraft dynamics" with a "particular interest" in "the effect of vehicle elasticity on attitude motions."

Category II. With regards to the technical merits of Kane *et al.*'s formulation, London draws attention to the following four points:

1. London questions why the authors of Ref. 1 "choose to represent the three elastic degrees of freedom by five variables $\{u_1, u_2, u_3, s, \zeta\}$."
2. It is pointed out that the use of $x + u_1$ as an upper limit of the integral in Eq. (1) (Eq. (19) in Ref. 1) creates problems in evaluating the modal integrals β_{ij} and γ_{ij} (Eq. (41) in Ref. 1).
3. London questions why the foreshortening is not considered during evaluation of the strain energy.
4. It is stated that the final equations derived by Kane "are linear in terms of vibration coordinates (but still retain higher-order spin effects) as is already the case with most other works."

2.3 Hanagud and Sarkar's Note

Contrary to what is claimed in Ref. 1, Hanagud and Sarkar believe that the axial and transverse motions can be treated independently with the standard discretization procedure and the stiffening effect can be accounted for if "the nonlinear effects are properly included in the formulation."[6]

Hanagud and Sarkar present a formulation where, as in the conventional approach, they discretize the axial displacement u_1, not the stretch variable s. Hanagud and Sarkar derive the differential equations for the corresponding discrete elastic coordinates by using the same general methodology as employed in Ref. 1, which as they point out is sometimes referred to as Kane's method. Similar to the development of Kane *et al.*, they also determine the generalized active forces which are the elastic (internal) forces, from the strain energy function. However, Hanagud and Sarkar formulate the strain energy as a *quartic* function of the spatial derivatives of u_1, u_2 and u_3. This is accomplished by employing the *nonlinear* strain-displacement relations, through which the aforementioned nonlinear effects are introduced into the formulation. Lastly, Hanagud and Sarkar do not linearize the final equations of motion, but retain terms of second and third order.

An important contribution of Hanagud and Sarkar's work is an observation that the expression for the stretch variable presented by Kane *et al.* (Eq. (19) in Ref. 1) is inconsistent with the rest of their development. The inconsistency results from the fact that this relation for s is applicable if one expresses the transverse displacements u_2 and u_3 as a function of the *deformed* coordinate X which corresponds to the axial projection of a generic point in the deformed configuration of the beam. (We have chosen to follow the traditional notation employed in the theory of elasticity, where one distinguishes the deformed and undeformed coordinates by different-case letters.) Since in their formulation, Kane *et al.* express the translations u_2 and u_3 in terms of the undeformed coordinate x (see Eq. (2)), the consistent expression for the stretch variable is:

$$x + s(x,t) = \int_0^x \left[\left(1 + \left. \frac{\partial u_1(x,t)}{\partial x} \right|_{x=\sigma} \right)^2 + \left(\left. \frac{\partial u_2(x,t)}{\partial x} \right|_{x=\sigma} \right)^2 + \left(\left. \frac{\partial u_3(x,t)}{\partial x} \right|_{x=\sigma} \right)^2 \right]^{1/2} d\sigma \quad (7)$$

The above equation, although looks different, is equivalent to Eq. (2) in Ref. 6. We also observe that it embodies the nonlinear formulation of the strain. In (7), we have used the notation $\left. \frac{\partial(\cdot)}{\partial x} \right|_{x=\sigma}$ to emphasize that elastic translations must be expressed as a function of the undeformed coordinate x, while σ in this case is a dummy integration variable. We also point

out a change in the upper limit of the integral from $x + u_1$ in Eq. (1) to x in Eq. (7) which, of course, is a consequence of the transformation from deformed to undeformed axial coordinate.

2.4 Discussion of the Commentaries

In the following, we offer our opinions on the various points made in the commentaries. We hope that these will serve to elucidate the more subtle features of the formulation proposed by Kane *et al.* A couple of the issues raised in the commentaries will be further addressed in the final section of the paper.

We agree with London in saying that when one reads the manuscript,[1] one gets an impression that its authors propose a new theory to model the deformation of a beam. Fundamentally, the theory is *not* new and a number of formulations which incorporate the foreshortening effect and the resultant stiffening of the beam during its rotation, have been previously published. (The material presented in §3 and §4 will support this fact.) Indeed, Kane *et al.* refer to some of these works, but only in a superficial and in some cases misleading manner. This notwithstanding, we feel that the formulation developed in Ref. 1 does have a couple of novel features. These are: (i) generality of the system modelled and (ii) use of the stretch variable as a generalized coordinate in deriving the dynamics equations. Although we do not share Kane's conviction that the dynamics equations *must be* formulated in terms of the stretch variable in order to predict stiffening of the beam, this particular feature of their procedure provides, at the least, an interesting alternative to the conventional approach.

Continuing with London's comments in the second category, we offer the following observations:

1. In our interpretation of the formulation,[1] the stretch s is introduced to replace the axial displacement u_1. This is made abundantly clear in section IV of Ref. 1 where s takes place of what usually appears as u_1. The variable ζ is employed as a short-hand for the combination $x + u_1$. In this light, we do not agree with London's statement that Kane *et al.* propose to use five variables to represent a three-degree-of-freedom displacement field.

2. According to the development presented in Ref. 1, the modal integrals β_{ij}, γ_{ij} are by definition time-dependent through $u_1 = u_1(x,t)$ in their upper limit of integration. This implies that they must be evaluated at each time step in the numerical integration of the motion equations. (Although that would certainly add to the computational cost of the simulation, it should not pose a problem otherwise.) The situation changes, however, if one corrects formulation in Ref. 1 in accordance with observations made by Hanagud and Sarkar. That can be achieved by replacing the inconsistent expression for the stretch, given by Eq. (1), with a consistent form of Eq. (7). With this correction, the aforementioned modal integrals become *independent* of time and, therefore, can be evaluated prior to the numerical integration of the motion equations.

3. This particular comment by London has motivated us to consider closely the expression for the strain energy function employed in Ref. 1. In the process, we have singled out the "axial" strain energy, previously denoted by U_P, as the only contribution which may possibly comprise the foreshortening effect. The following brief development shows that indeed it does.

   The function U_P, as can be seen from (6), is quadratic in the spatial derivative of the stretch, $\frac{\partial s}{\partial x}$. Thus, in order to illuminate the nature of this strain energy, we need to obtain

an explicit expression for the stretch gradient. As noted previously, Eq. (7) provides the proper form for s that should be employed in Ref. 1. Abbreviating the notation, we rewrite (7) for s with:

$$s(x,t) = \int_0^x \left[\left(1 + \frac{\partial u_1}{\partial \sigma}\right)^2 + \left(\frac{\partial u_2}{\partial \sigma}\right)^2 + \left(\frac{\partial u_3}{\partial \sigma}\right)^2 \right]^{1/2} d\sigma - x \tag{8}$$

Differentiating the above with respect to x we get:

$$\frac{\partial s}{\partial x} = \left[\left(1 + \frac{\partial u_1}{\partial x}\right)^2 + \left(\frac{\partial u_2}{\partial x}\right)^2 + \left(\frac{\partial u_3}{\partial x}\right)^2 \right]^{1/2} - 1 \tag{9}$$

and upon expansion of the first term, the required gradient takes the form:

$$\frac{\partial s}{\partial x} = \left[1 + 2\frac{\partial u_1}{\partial x} + \left(\frac{\partial u_1}{\partial x}\right)^2 + \left(\frac{\partial u_2}{\partial x}\right)^2 + \left(\frac{\partial u_3}{\partial x}\right)^2 \right]^{1/2} - 1 \tag{10}$$

Let us now introduce the axial strain $\varepsilon_{0,xx}$, where the 0 subscript signifies that it refers to the elastic axis. (Note that u_1, u_2 and u_3 are defined in Ref. 1 as translations of points along the elastic axis only.) The strain $\varepsilon_{0,xx}$ can be expressed in terms of the elastic displacements with a well-established strain-displacement relation. It has the following *exact* and *nonlinear* form:

$$\varepsilon_{0,xx} = \frac{\partial u_1}{\partial x} + \tfrac{1}{2}\left[\left(\frac{\partial u_1}{\partial x}\right)^2 + \left(\frac{\partial u_2}{\partial x}\right)^2 + \left(\frac{\partial u_3}{\partial x}\right)^2 \right] \tag{11}$$

With the above, the stretch gradient of Eq. (10) can be succinctly written as:

$$\frac{\partial s}{\partial x} = [1 + 2\varepsilon_{0,xx}]^{1/2} - 1 \tag{12}$$

Before we continue, it is worthwhile to point out that in all formulations dealing with the subject of geometric stiffening in the context of multibody dynamics, it is always assumed, although not always stated, that the strains are small, and specifically, $\varepsilon_{0,xx} << 1$. Therefore, we can make use of the Binomial Theorem to simplify Eq. (12). Retaining the first two terms in the binomial expansion we get:

$$\frac{\partial s}{\partial x} \approx [1 + \varepsilon_{0,xx}] - 1 = \varepsilon_{0,xx} \tag{13}$$

Finally, substituting for $\frac{\partial s}{\partial x}$ from (13), the axial contribution to the strain energy function employed by Kane *et al.* takes the form:

$$U_P = \frac{1}{2}\int_0^L EA\,(\varepsilon_{0,xx})^2\,dx \tag{14}$$

with the axial strain given by the *nonlinear* Eq. (11).

At this point, it is appropriate to comment on the form of the strain energy employed by Hanagud and Sarkar in their formulation (Eq. (8) in Ref. 6). Their expression can be

derived by using Eq. (14) in conjunction with the following simplified expression for the axial strain:

$$\varepsilon_{0,xx} \approx \frac{\partial u_1}{\partial x} + \frac{1}{2}\left[\left(\frac{\partial u_2}{\partial x}\right)^2 + \left(\frac{\partial u_3}{\partial x}\right)^2\right] \tag{15}$$

The above form is obtained from the exact relation (11) by omitting the $\left(\frac{\partial u_1}{\partial x}\right)^2$ term—an approximation often made in the context of moderate-deformation theories.[17] The third- and fourth-order terms in the strain energy of Hanagud and Sarkar result from the nonlinear terms $\left(\frac{\partial u_2}{\partial x}\right)^2$ and $\left(\frac{\partial u_3}{\partial x}\right)^2$ in the strain-displacement relation (15). It is these terms that lead to the geometric stiffening and foreshortening of the beam.

The development presented here demonstrates that, contrary to London's comment, the geometric nonlinearity *is* included in the strain energy expression given in Ref. 1. This is not apparent because Kane *et al.* employ the stretch s instead of the axial translation u_1 as a generalized coordinate in their formulation. As a consequence, they do not need to expand $\frac{\partial s}{\partial x}$ when evaluating U so that their strain energy function remains quadratic in the discrete generalized coordinates and does not explicitly contain higher-order terms.

4. We have found this comment by London somewhat bewildering. As stated in it, and as clearly stated in Ref. 1, the equations of motion developed by Kane *et al.* are linear in the elastic coordinates and incorporate the geometric stiffening. In fact, one of the crucial points made in Ref. 1 is that the linear motion equations in other works do not contain all of the linear terms, in particular the stiffening term. As implied by Kane *et al.*,[1] and later stated by Eke and Laskin[2] and Padilla and von Flotow,[3] this occurs because of premature linearization implicit in the "conventional approach." Contrary to this, we support Hanagud and Sarkar's view that one *can* obtain the stiffening effect with the conventional approach. This is achieved by employing the nonlinear strain-displacement relations in constructing the strain energy function. The stiffening term obtained with this approach is a *nonlinear* function of elastic coordinates.

   The reason why the final equations of Kane *et al.* are linear in elastic coordinates, but yet include the stiffening terms lies in their choice of the stretch s as a generalized coordinate and the fact that it includes the nonlinear contribution from transverse displacements. As we had shown in the previous comment, it is through the use of stretch instead of axial displacement, that Kane *et al.* incorporate foreshortening in their formulation, without introducing nonlinearity explicitly into the motion equations.

3 References Contended by London

London's comments on the literature review and Kane *et al.*'s reply to them[18], motivated us to investigate the material presented in several of the references in question. The main objective of this section is to summarize these findings.

3.1 Likins *et al.*, Vigneron, and Kaza and Kvaternik

Our choice to discuss the contributions of Likins *et al.*,[7] Vigneron,[8] and Kaza and Kvaternik[9] in the same section is based on several reasons. First, these articles appeared within a time-span

of four years and therefore belong to the same "era." Second, all three publications contain formulations of the dynamics equations for a flexible beam spinning in a plane at a constant speed. Finally, there is a logical relationship between these works, since that by Vigneron is a comment on Likins *et al.*, while Kaza and Kvaternik extend Vigneron's approach to obtain the nonlinear equations of motion of the aforementioned system.

Our presentation is not a plain copy of Refs. 7, 8 and 9, as it is structured to make apparent the key features of the approaches taken in the three works, establish a relationship between them, as well as identify the particular contributions of each one. Towards this end, we present the results of these works in a common notation, which will also be employed throughout the rest of the paper. This notation is similar to that used in Ref. 1 with one major difference. We choose to denote the three orthogonal elastic displacements with symbols u, v and w. Thus, u now represents the axial elastic displacement measured along the x-axis of the reference frame, while v and w are the transverse elastic displacements. Furthermore, these symbols are not restricted to the elastic axis, but represent elastic displacements of any point in the beam. This convention follows that used by Kaza and Kvaternik.[9]

As already mentioned, the system considered in all three publications[1] is a uniform elastic beam, with a symmetric cross-section, spinning at a constant angular speed Ω about the z-axis of the reference frame. This system represents a special case of that treated by Kane *et al.*, defined with: $\omega_3 = \Omega$, $\omega_1 = \omega_2 = v_1 = v_2 = v_3 = 0$, $e_2 = e_3 = 0$ as well as constant geometric and material properties. The common features of the formulations[7,8,9] are listed below.

(i) The dynamics equations are constructed *via* Hamilton's principle.

(ii) The position of a generic point located at $[x, y, z]^T$ in the undeformed beam is given by $[x + u, y + v, z + w]^T$, where v and w are functions of x and time only, that is $u(x, y, z, t) = u(x, t)$ and similarly for v and w. As well, the transverse displacements are assumed constant in a cross-section, thus precluding torsional deformation.

(iii) The kinetic energy is calculated with:

$$T = \tfrac{1}{2} \iiint \Big[\dot{u}^2 + \dot{v}^2 + \dot{w}^2 + \Omega^2 (x + u)^2 + \Omega^2 (y + v)^2 + 2\Omega (x + u)\dot{v} - 2\Omega (y + v)\dot{u} \Big] \frac{\rho}{A}\, dx\, dy\, dz \tag{16}$$

where, like in Ref. 1, the symbol ρ denotes the mass per unit length of the beam and A is the cross-sectional area. The above equation is identical to equation (6) of Likins *et al.* and applies to any elastic body spinning as specified before. It can be expanded and simplified for a beam with a symmetric cross-section to yield:

$$T = \tfrac{1}{2}\rho \int \left(\dot{u}^2 + \dot{v}^2 + \dot{w}^2 \right) dx + \tfrac{1}{2}\Omega^2 \rho \int \left(x^2 + u^2 + v^2 + 2xu \right) dx + \Omega\rho \int (x\dot{v} + u\dot{v} - v\dot{u})\, dx + \frac{1}{2A}\rho\Omega^2 L I_z \tag{17}$$

where we have used the standard definition $I_z = \iint y^2\, dy\, dz$.

(iv) The potential energy is calculated with

$$U = \frac{E}{2} \iiint (\varepsilon_{xx})^2\, dx\, dy\, dz \tag{18}$$

[1] Likins *et al.* also consider "Axial Beams."

where ε_{xx} denotes the axial strain at any point in the beam, and is given by a nonlinear expression:

$$\varepsilon_{xx} = \frac{\partial u}{\partial x} + \tfrac{1}{2}\left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial w}{\partial x}\right)^2\right] \tag{19}$$

(v) The dynamics equations are formulated for the *continuous* displacement variables, and accordingly take the form of partial differential equations.

There are two major differences between formulations presented in Refs. 7, 8 and 9. The first one relates to the form of the assumed *axial* displacement field. Likins *et al.* "expand" the axial displacement u with:

$$u = u_0 - y\frac{\partial v}{\partial x} - z\frac{\partial w}{\partial x} \tag{20}$$

Vigneron, followed by Kaza and Kvaternik adopt a different form. Their axial displacement is given by Eqs. (2) and (1a) in Refs. 8 and 9 respectively, which we write as:

$$u = u_s - y\frac{\partial v}{\partial x} - z\frac{\partial w}{\partial x} - u_f \tag{21}$$

In the above, u_f is the "displacement associated with the foreshortening effect".[8] In both references, this component of the axial displacement is specified as an explicit function of the transverse displacements:

$$u_f = u_f(x,t) = \tfrac{1}{2}\int_0^x \left[\left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial w}{\partial x}\right)^2\right] dx \tag{22}$$

We note that expression (22) is a second-order approximation for the foreshortening of the beam. Clearly, u_f corresponds to the axial displacement which results from the transverse deformation. Comparing Eqs. (20) and (21), we observe that u_0 must equal $u_s - u_f$ and hence u_0 and u_s represent different physical quantities. We have employed the subscript s in u_s to signify that it corresponds to axial displacement resulting strictly from *extension* or *stretch* of the elastic axis. By comparison, u_0 represents the total axial displacement on the elastic axis which may be due to *both* stretch and bending.

The second difference between the derivations of Likins *et al.*, Vigneron, and Kaza and Kvaternik lies in the approximations made in deriving the final equations of motion for the elastic coordinates u_0 or u_s, v and w. To be specific, these are approximations made in formulating kinetic and strain energies that are subsequently used to construct the Lagrangian of the system.

Likins *et al.* substitute their expansion for u from Eq. (20) into the kinetic and strain energy expressions (17) and (18), the latter combined with (19), thereby reformulating T and U in terms of u_0, v and w. They simplify the results by making the following assumptions.

I. The in-plane deformation is ignored based on the argument that it is present only because of the Poisson effect.

   (a) This amounts to dropping all terms in kinetic energy T which involve v or its derivatives.

(b) The strain energy U is derived by omitting $\left(\frac{\partial v}{\partial x}\right)^2$ in the definition (19) for the strain.

II. Likins *et al.* neglect all terms that involve u_0 and $\dot{u}_0$ in T, thus effectively eliminating the axial equation of motion from the model.

III. They retain only one term in the strain energy U among the additional third- and fourth-order terms which arise from the nonlinearities in the strain-displacement relation.

With the above assumptions, the kinetic and potential energy functions take the form:

$$T = \tfrac{1}{2}\rho \int_0^L \dot{w}^2\, dx + \rho\Omega^2 L\left(\frac{1}{2A} I_z + \frac{1}{6}L^2\right) \tag{23}$$

$$U = \frac{EI_z}{2}\int_0^L \left(\frac{\partial^2 w}{\partial x^2}\right)^2 dx + \frac{EA}{2}\int_0^L \frac{\partial u_0}{\partial x}\left(\frac{\partial w}{\partial x}\right)^2 dx \tag{24}$$

where in accordance with the assumption II above, we have dropped the term in the strain energy which involves u_0 only. These correspond to Eqs. (39) and (45) in Ref. 7. Note, that the second term in (23) includes the rotary inertia contribution. Being constant, it does not contribute to the motion equation for w. Also, the second term in (24) is the additional term mentioned in III and is of third order. It represents the coupling between axial and transverse displacements which leads to the stiffening of the beam in bending. This term is identical to the third-order term in the strain energy function of Hanagud and Sarkar.

At this point in their development, Likins *et al.* establish a connection between their approach, as we have just outlined, and "the textbook derivation for the transverse vibrations of beams subject to an external axial force P." As noted by them, the axial load to first approximation is given by:

$$P = EA\frac{\partial u_0}{\partial x} \tag{25}$$

so that Eq. (24) can be rewritten as:

$$U = \frac{EI_z}{2}\int_0^L \left(\frac{\partial^2 w}{\partial x^2}\right)^2 dx + \tfrac{1}{2}\int_0^L P\left(\frac{\partial w}{\partial x}\right)^2 dx \tag{26}$$

To proceed with the application of Hamilton's variational principle, Likins *et al.* assume that P is time-independent and can be approximated by its steady-state value. In fact, for the particular problem of a beam rotating at a constant speed, the axial load P is the centrifugal load on the beam. Furthermore, since the latter is a known function of the prescribed Ω, P can be calculated with:

$$P(x) = \tfrac{1}{2}\rho\Omega^2(L^2 - x^2) \tag{27}$$

With equations (23), (26) and (27), Likins *et al.* derive the motion equation for the elastic displacement w. Due to the assumptions made in evaluating the strain energy and the axial load P (Eqs. (24), (25) and (27)), geometric stiffening appears as a *linear* term in this equation.

As a final comment on Ref. 7, we note that in section titled "Finite-Element Model" Likins *et al.* explicitly include a term that represents "modifications of structural stiffness due to spin-induced loads on the structure in its steady state (the so-called "geometric stiffness")."

Let us now proceed with the developments presented by Vigneron, and Kaza and Kvaternik. As we had already mentioned, both derivations are premised on expansion (21) for the axial displacement. As well, the kinetic energy is constructed by ignoring the effects of rotary inertia. This can be achieved by neglecting the "off-elastic-axis" contribution, $(-y\frac{\partial v}{\partial x} - z\frac{\partial w}{\partial x})$, in the expression for u and dropping the last (I_z-) term in Eq. (17). Then, kinetic energy can be written as a function of u_s, v, w and u_f with:

$$\begin{aligned} T = \quad & \tfrac{1}{2}\rho \int_0^L \left(\dot{u}_s^2 + \dot{v}^2 + \dot{w}^2 - \underline{2\dot{u}_s\dot{u}_f} + \underline{\dot{u}_f^2}\right) dx \\ & + \tfrac{\Omega^2}{2}\rho \int_0^L \left(x^2 + u_s^2 + v^2 + \underline{\underline{u_f^2}} + 2xu_s - [2xu_f] - \underline{2u_su_f}\right) dx \\ & + \rho\Omega \int_0^L \left(-\dot{u}_sv + v\dot{u}_f + x\dot{v} + \dot{v}u_s - \underline{u_f\dot{v}}\right) dx \end{aligned} \tag{28}$$

Recall that according to (22), the foreshortening u_f is a quadratic function of the transverse displacement gradients and, therefore, the above expression for T includes terms of third and fourth order. These terms are underlined in (28) with single and double lines, respectively. We have also singled out by enclosing in square brackets the term which is *linear* in u_f. This term gives rise to the stiffening effect in the motion equation. We also emphasize that Eq. (28) originates from the same expression for the kinetic energy (Eq. (17)) as used by Likins *et al.* The higher order terms in it are a consequence of dividing the axial displacement u_0 into two components, u_s and u_f, and explicitly assuming a second-order form for the latter.

Vigneron approximates T by keeping only second-order terms in (28) (including the term in the square brackets) as well as, setting the axial displacement u_s and its derivatives to zero. The latter approximation corresponds to *inextensibility* assumption, which implies that the beam is modelled as axially rigid.[9] Note, that dropping the u_s-terms in Vigneron's formulation is *not* equivalent to dropping the u_0- terms in Likins *et al.*'s formulation, although in both cases this achieves elimination of the independent axial equation of motion. Vigneron's approximation yields:

$$T = \tfrac{1}{2}\rho \int_0^L \left(\dot{v}^2 + \dot{w}^2\right) dx + \frac{\Omega^2}{2}\rho \int_0^L \left(x^2 + v^2 - [2xu_f]\right) dx + \rho\Omega \int_0^L (x\dot{v})\, dx \tag{29}$$

which, without the last term, corresponds to equation (9) of his paper. Unlike Vigneron, Kaza and Kvaternik do not make the inextensibility assumption and therefore keep the axial equation of motion in their final model. They also retain third-order terms in (28). The result is given by Eq. (7) in Ref. 9 which is identical to our Eq. (28) without the double-underlined terms.

To determine the strain energy according to (18), Vigneron and Kaza and Kvaternik first evaluate the strain ε_{xx} of (19) by substituting for the axial displacement u from (21) in conjunction with (22). Upon dropping the third- and fourth-order terms, this produces the following second-order expression for the axial strain:

$$\varepsilon_{xx} = \frac{\partial\left(u_s - y\frac{\partial v}{\partial x} - z\frac{\partial w}{\partial x}\right)}{\partial x} - \frac{\partial u_f}{\partial x} + \tfrac{1}{2}\left[\left(\frac{\partial\left(u_s - y\frac{\partial v}{\partial x} - z\frac{\partial w}{\partial x}\right)}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial w}{\partial x}\right)^2\right] \tag{30}$$

where we have intentionally separated the contribution of u_f. However, according to (22), we

have

$$\frac{\partial u_f}{\partial x} = \frac{1}{2}\left[\left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial w}{\partial x}\right)^2\right] \tag{31}$$

so that (30) is simplified to:

$$\varepsilon_{xx} = \frac{\partial\left(u_s - y\frac{\partial v}{\partial x} - z\frac{\partial w}{\partial x}\right)}{\partial x} + \frac{1}{2}\left(\frac{\partial\left(u_s - y\frac{\partial v}{\partial x} - z\frac{\partial w}{\partial x}\right)}{\partial x}\right)^2 \tag{32}$$

or

$$\varepsilon_{xx} = \frac{\partial u_s}{\partial x} - y\frac{\partial^2 v}{\partial x^2} - z\frac{\partial^2 w}{\partial x^2} + \frac{1}{2}\left[\frac{\partial u_s}{\partial x} - y\frac{\partial^2 v}{\partial x^2} - z\frac{\partial^2 w}{\partial x^2}\right]^2 \tag{33}$$

The above is equivalent to Eq. (5) in Ref. 8 and Eq. (4) in Ref. 9, although neither publication includes a description of the intermediate step (30). It is interesting to note that ε_{xx} evaluated with (32) without the second-order term, can be derived from the axial strain of (19) without the $\left(\frac{\partial u}{\partial x}\right)^2$ term, but with the "transverse" nonlinear terms. As is demonstrated by Eq. (30), the latter is cancelled when foreshortening u_f is explicitly defined in the axial displacement u, which results in a linear expression for the strain. Vigneron's strain energy can be constructed by using this linearized form of (32), since he retains only quadratic terms in the strain energy. In fact, the strain energy function given in Ref. 8 has a standard form used in the linear theory of elasticity. Kaza and Kvaternik formulate their strain energy with the axial strain as given by (32), but drop the resultant fourth-order terms.

Because of the differences in the approximations made, Vigneron and Kaza and Kvaternik derive different sets of motion equations. Vigneron obtains two *linear* equations of motion for the elastic variables v and w (Eqs. (10) and (11) in Ref. 8), the latter being identical to the equation derived by Likins *et al.* Kaza and Kvaternik present a set of nonlinear dynamics equations for the three elastic deformations, u_s, v and w (Eqs. 8(a,b,c) in Ref. 9). They are nonlinear, in particular second-order, because Kaza and Kvaternik retain third-order terms in their kinetic and strain energies. It is important to emphasize, that the term responsible for the stiffening of the beam derived in Refs. 8 and 9 is a first-order term and appears *via* kinetic energy. This occurs because of the particular form assumed for the axial displacement (Eq. (21)) as well as, the expression adopted for the foreshortening (Eq. (22).) If, as is done by Likins *et al.*,[7] one does not explicitly identify u_f in the axial displacement, then the stiffening appears in the motion equations through the strain energy and is fundamentally a nonlinear term.

To conclude this section, we draw attention to some of the observations made by Kaza and Kvaternik. They identify four different approaches for deriving linear or nonlinear equations of motion. They are: (1) "the effective applied load artifice;" (2) the use of Newton's second law applied to the deformed configuration; (3) an approach in which nonlinear strain-displacement relations and a first-degree displacement field are used; (4) Vigneron's approach which uses nonlinear strain-displacement relations and a second-degree displacement field. Kaza and Kvaternik show that all four appraches "make use of geometric nonlinear theory of elasticity either implicitly or explicitly." They state that "for developing the equations of motion for a rotating beam, "the geometric nonlinear theory is necessary to obtain even the correct linear equations." In their paper, Kaza and Kvaternik also discuss whether foreshortening must be explicitly included in the axial displacement field. They conclude that although it is not necessary, the approaches where the foreshortening effect is accounted for otherwise, require special considerations.

3.2 Lips and Modi, Hughes and Fung

Lips and Modi investigate the dynamics of satellite systems composed of a central rigid body with flexible appendages. The base body is allowed to undergo general rotational motion and therefore, the dynamics model includes the rotational rigid-body equations. To illustrate the procedure for modelling deformation of appendages, Lips and Modi present an explicit form of the elastic equations for a rotating beam. In their treatment of this system, they take into account: (i) the offset between the attachment point of the appendage and the center of mass of the rigid body; (ii) variable flexural rigidity which subsumes variable modulus of elasticity and cross-sectional area; (iii) variable density. The basic assumptions made in deriving the equations for elastic displacements are similar to those made by Kaza and Kvaternik.[9] In particular, Lips and Modi explicitly separate the foreshortening component from what they refer to as the oscillation component in the assumed form of the axial displacement. As well, the kinetic and potential energy functions employed to construct the Lagrangian contain terms up to third order.

The main subject of the work by Hughes and Fung[11] is the stability of spinning satellites with long flexible appendages. Therefore, they formulate the dynamics equations with a view to addressing this issue. The system is modelled as a spinning rigid body with appended beams. The rigid-body equations are formulated for small perturbations from the nominal spin configuration. The development of the elastic motion equations presented in Ref. 11 is different from the previously considered works in two respects. First, Hughes and Fung employ *deformed* coordinates to describe the kinematics of the deformed beam, a fact which they do not state explicitly. Thus, the position of a point on the elastic axis is defined by $[X, v, w]^T$ where $X = X(t)$, $v = v(X,t)$ and $w = w(X,t)$. (Note, the corresponding undeformed description is $[x + u, v, w]^T$ where $u = u(x,t)$, etc.) Another distinct feature of Hughes and Fung's formulation is that they evaluate the kinetic and potential energies by integrating the respective appropriate integrands over the volume, which in the case of a slender beam reduces to the arc length $\hat{s}$, of the *deformed* configuration. Thus, the energy functions are defined by means of the line integral $\int(\cdot)d\hat{s}$. In general, the integration necessary to determine kinetic and potential energies of an elastic body can be performed over the deformed or undeformed configurations. However, the former is the standard choice when the kinematic description is given in terms of the deformed coordinates. (This representation corresponds to the Eulerian or spatial description of the problem.)

Hughes and Fung incorporate the geometric nonlinearity into their formulation by expressing the differential arc length $d\hat{s}$ with

$$d\hat{s} = \sqrt{1 + v'^2 + w'^2}\, dX \tag{34}$$

The prime in the above denotes differentiation with respect to the deformed coordinate X. We note that Eq. (34) is equivalent to Eq. (4) of Hanagud and Sarkar.[6] It defines the distance along the beam as a function of the transverse displacement gradients, when these are expressed in terms of the deformed coordinates. Hence, Eq. (34) is also equivalent to Eq. (1) of this paper, with $\hat{s} = x + s$. To simplify the derivation, Hughes and Fung use a second-order approximation of $d\hat{s}$ so that

$$\int_0^L (\cdot)\, d\hat{s} = \int_0^{L^\star} (\cdot)\, dX + \tfrac{1}{2} \int_0^{L^\star} (\cdot) \left(v'^2 + w'^2\right) dX \tag{35}$$

where $L^\star$ denotes the projection of the tip on the axial coordinate axis. We also note that by using L in the upper limit of the integral $\int(\cdot)d\hat{s}$, Hughes and Fung implicitly assume that the beam is inextensible.

The kinetic and potential energies derived in Ref. 11 contain second-order terms only. That, combined with the inextensibility assumption makes the formulation of Hughes and Fung similar to Vigneron's. In fact, the kinetic energy of a single beam rotating at a nominal speed, deduced from Eq. (9) of their paper, is equivalent to Vigneron's kinetic energy given by our Eq. (29). The form of the strain energy function is also the same.

This completes our overview of the publications disputed in London's and Kane *et al.*'s commentaries.[5,18] It is clear that these works incorporate geometric stiffening into the dynamics equations of a rotating beam, although through different approaches. A discussion of these will be given in §6 of the paper. At this point, we only note that they all account for the coupling between the transverse and axial deformations. Indeed, it is exactly this phenomenon that causes stiffening of an elastic body under certain conditions. The differences between the approaches lie in what we view as the *mechanism* for introducing the coupling effect into the formulation and accordingly, the stage in the derivation at which it is introduced. In the following two sections of the paper we discuss some of the other approaches that have been employed to account for the geometric stiffening in the dynamics equations of multibody systems.

4 Laskin *et al.*, Meirovitch, Banerjee *et al.*

4.1 Laskin *et al.*

Similar to Vigneron, Laskin *et al.*[12] assume a form for the axial displacement u in which the displacement of points along the elastic axis, u_0, is divided into two parts. We rewrite their Eq. (6) as:

$$u = u_{qs} + u_t - y\frac{\partial v}{\partial x} - z\frac{\partial w}{\partial x} \tag{36}$$

where they refer to u_{qs} (v_0 in Ref. 12) as a quasi-steady component and u_t (their $v^\star$) as a transient component that accounts for longitudinal vibrations. They justify this arrangement by arguing that it allows one to consider u_t as a small, more precisely, infinitesimally small displacement which is of the order of the transverse displacements v and w. The quasi-steady component u_{qs} may be comparatively large.

To discretize the elastic deformation field, Laskin *et al.* use modal expansion similar in form to (3) but written for the transient axial displacement u_t rather then u_0 (u_1 in (3)). By doing so, they implicitly choose u_t as a generalized coordinate in their formulation. Laskin *et al.* explain this choice by saying that because u_t is small, its modal coordinates can also be regarded as small. By constrast, if one discretizes $u_0 = u_{qs} + u_t$, its modal expansion cannot adequately account for both large (u_{qs}) and small (u_t) components without having to include a large number of terms in the expansion.

Laskin *et al.* employ Kane's method to derive the motion equation of the beam. Thus, the next step in the formulation is to develop the generalized inertia and active forces. Starting with the latter, they consider two contributions—the elastic forces and the controller forces. In keeping with the subject of this paper, discussion is limited to Laskin *et al.*'s derivation of the generalized *elastic* forces.

Let us recall that Kane *et al.*, and Hanagud and Sarkar derive the elastic force from the strain energy function. Laskin *et al.* take a different approach and express the elastic force $\mathbf{F}$ as a function of the stresses in the beam. This is actually a traditional way of writing the

elastic force contribution to the equilibrium equations of the theory of elasticity. As pointed out,[12] one can express $\mathbf{F}$ in *linear* elasticity with:

$$\mathbf{F} = \nabla \cdot \sigma \tag{37}$$

where σ is the stress tensor. We note that this form is also valid in nonlinear theory, provided the divergence operator is defined with respect to the deformed coordinates and the strains are small. If ∇ is defined with respect to the undeformed coordinates, then Eq. (37) must be modified to take into accout the nonlinear nature of the deformation gradient. The resultant expresssion for $\mathbf{F}$ is given by Eq. (21) in Ref. 12 which we do not repeat here. In agreement with Kaza and Kvaternik's comments on the Newton's second law approach, we point out that this form allows for arbitrary rotations and implicitly assumes nonlinear strain-displacement relations.

Since the generalized elastic forces given by (22)-(25)[12] are expressed in terms of the generalized coordinates, Laskin *et al.* must take an intermediate step of substituting for the stresses in their (21) from stress-strain and then strain-displacement relations, in order to reformulate $\mathbf{F}$ in terms of u_{qs}, u_t, v and w. What is most notable about their approach is that, by using the "nonlinear" relation for $\mathbf{F}$ in terms of the stresses, they are able to derive the geometric stiffening component of elastic forces with a *linear* form of the strain-displacement relations. Laskin *et al.* emphasize that if one were to use strain energy to derive the elastic forces $\mathbf{F}$, one would have to retain nonlinear terms in the strain-displacement relations. To add to their interpretation of this "paradoxical situation", we offer this observation. The nonlinear terms in the strain-displacement relations give rise to third- and fourth-order terms in the strain energy. As was shown in §3.1, it is the third-order terms that are responsible for what is usually referred to as geometric stiffening. (In fact, the fourth-order terms also contribute to stiffening of the beam in bending.) Since the "nonlinear" formulation of $\mathbf{F}$ in terms of stresses is one-order higher then the "linear" one, it essentially provides a mechanism for incorporating only third-order terms in the strain energy.

We note that geometric stiffening is represented by G_{ij}- and a_{i0}- terms in Eqs. (23)-(25) of Ref. 12. Although these are linear in the discrete generalized coordinates, they are also dependent on the spatial derivative of the quasi-steady axial displacement u_{qs} (see the definitions of G_{ij} and a_{i0} on p. 515 of Ref. 12, with $v_0 = u_{qs}$).

Derivation of the generalized inertia forces[12] is similar to the procedure in Ref. 1 with the main difference being the addition of six rigid generalized inertia forces. These appear because the rigid-body motion of the beam is not prescribed, but is unknown. Accordingly, the final equation of motion include six equations for the position and orientation of the floating reference frame and the differential equations for the discrete elastic coordinates. They are explicitly dependent on u_{qs}, and therefore require this axial displacement as an input.

Laskin *et al.* apply their dynamics equations to a number of special cases. They specify u_{qs} as the stretch that would occur if the beam were executing its nominal or intended motion. This stretch is defined as a solution of a linear second-order differential equation which we deduce from examples discussed in Ref. 12 to be:

$$EA\frac{\partial^2 u_{qs}}{\partial t^2} = -\frac{\partial P}{\partial x} = p \tag{38}$$

Here, P is nominal axial load on the beam and we have introduced the bymbol p to denote the axial load density. To generalize this scheme for the case of general *rotational* motions, Laskin *et al.* propose to approximate p by the axial component of the *centripetal* acceleration

(multiplied by an appropriate inertia) of a point on the elastic axis of the beam. This yields, in accordance with Eq. (78) in Ref. 12:

$$p = \rho(x + u_{qs}) \left(\omega_2^2 + \omega_3^2\right) \tag{39}$$

It is suggested that the above should provide a good approximation for the steady-state axial load in the case of rotational motions at low angular acceleration rates. Moreover, it allows for a closed-form solution of (38) for u_{qs} which can then be used as an input to their dynamics model.

To complete this section, we observe that Eq. (36) is analogous to (21) employed by Vigneron and Kaza and Kvaternik in their formulations, with the correspondence $u_f = -u_{qs}$ and $u_s = u_t$. The term "quasi-steady" used by Laskin *et al.* to qualify foreshortening can be interpreted to reflect the fact that this axial displacement is present even when there is no axial vibration, and furthermore, it exists even under static loading. Unlike what is done in Refs. 8 and 9, Laskin *et al.* do not substitute for u_{qs} in terms of v and w as in (22), nor any other expression. As a consequence, u_{qs} appears in both generalized inertia and elastic forces.

4.2 Meirovitch

In his 1967 book, Meirovitch[13] includes a section on the effect of axial forces in the bending vibration of a bar, which as he states, cannot be ignored in some cases. In this section, Meirovitch derives an equation of motion for the transverse displacement of the beam by means of extended Hamilton's principle. Thus, expressions for kinetic energy T and work function W are developed. The former takes the simple form used in planar bending vibration problems without the axial force. Rewritten in our notation, T specified in Ref. 13 is:

$$T = \tfrac{1}{2} \int_0^L \rho(x) \left(\frac{\partial v(x,t)}{\partial t}\right)^2 dx \tag{40}$$

In evaluating the work function, he proposes to include the effect of the bending moment, the transverse (external) load and axial force. The first two are formulated in the same way as for the case without the axial force. To determine the "axial" work, the change in the horizontal projection of an element $d\hat{s}$ is calculated. This differential of the foreshortening is expressed with:

$$d\hat{s} - dx = \sqrt{1 + \left(\frac{\partial v}{\partial x}\right)^2}\, dx - dx \approx \tfrac{1}{2} \left(\frac{\partial v}{\partial x}\right)^2 dx \tag{41}$$

where the approximation results from retaining two terms in the binomial expansion of $\sqrt{1 + \left(\frac{\partial v}{\partial x}\right)^2}$. Then, the work done by the axial force is:

$$W_P = -\tfrac{1}{2} \int_0^L P(x,t) \left(\frac{\partial v}{\partial x}\right)^2 dx \tag{42}$$

It is worth to point out that in adopting the above formulation, Meirovitch makes a tacit assumption that the axial force is given as a known function of x and t.

In the more recent publication,[14] Meirovitch derives a set of motion equations for a general flexible body in general motion. These equations are written in terms of the rigid-body quasi-coordinates and the continuous elastic coordinates u, v and w. Their application

to a system made up of a rigid hub and a beam-like flexible appendage is illustrated. He begins by assuming that the axial displacement can be ignored and therefore sets $u = 0$ apriori. The kinetic energy is derived in the standard manner and contains terms that are of second degree in the elastic variables. The strain energy includes the standard second-order contributions due to bending in two directions as well as the contribution due to "shortening of the projection." The latter is expressed as

$$U_P = \tfrac{1}{2}\int_0^L \left[\int_x^L p(\zeta,t)\,d\zeta\right]\left[\left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial w}{\partial x}\right)^2\right] dx \tag{43}$$

where $p(x,t)$ is the axial component of the internal force density. We note that (43) can be directly compared to (42).

Meirovitch proposes to determine the force density p from the motion equation for the axial displacement u, which as we had mentioned is excluded from the dynamics model. Thus, he defines p as a sum of the terms in the u-equation, omitting terms that involve elastic displacements, as well as the control force density. The resultant expression for p is given by Eq. (29) in Ref. 14, which we rewrite here as:

$$p = \rho\left[-\dot{v}_1 - \omega_2 v_3 + \omega_3 v_2 + x\left(\omega_2^2 + \omega_3^2\right)\right] \tag{44}$$

At this point, let us compare the above expression for the internal axial force density which is employed by Meirovitch to evaluate the strain energy due to the shortening of the beam with the corresponding expression of Laskin *et al.*, which they use to determine the quasi-steady component of the axial displacement. We recall that the expression for p proposed in Ref. 12, as given by Eq. (39) of this paper, originates from inertial acceleration of a mass element on the beam's elastic axis. In fact, it is defined as the centripetal component of the acceleration evaluated with $u = u_{qs}$. It can be shown that Meirovitch's expression for p can also be derived from the inertial acceleration distribution, but neglecting the elastic contributions. Indeed, this is not surprising since a motion equation in the absence of external forces is essentially a linear homogeneous relation for the inertial acceleration. Thus, the "centripetal component" of p, given by $\rho x\left(\omega_2^2 + \omega_3^2\right)$ in Eq. (44), is identical to p of Eq. (39) if one drops u_{qs}. Finally, by comparing (44) to (39), we observe that Laskin *et al.*'s approximation for the axial force density should apply when the translational velocity and acceleration are small relative to the angular velocity, in addition to the conditions stated in Ref. 12.

4.3 Banerjee *et al.*

Banerjee and Dickens[4] present a formulation for the dynamics of an *arbitrary* flexible body in large rigid-body motion. The formulation is based on the standard description of the deformation field where the elastic translations u, v and w are discretized by means of a modal expansion. The equations of motion are derived by employing the same general methodology as in Refs. 1 and 6. Therefore, as was done in reviewing other works, the following analysis focuses on the procedure to derive the stiffening terms only.

Banerjee and Dickens introduce the notion of "motion stiffness" as a special case of the geometric stiffness caused by the inertia loading on the body due to its large rigid motion. As is noted by Banerjee and Lemak,[15] this motion-induced stiffness has its origin in the strain energy term,

$$U_{NL} = \int \boldsymbol{\varepsilon}_{NL}{}^T \boldsymbol{\sigma}\, dV \tag{45}$$

where σ and ϵ_{NL} are 6x1 columns. One can view U_{NL} as the "nonlinear" part of U, since it arises from the nonlinear terms in the strain-displacement relations. These are represented in (45) by ϵ_{NL}, while σ represents the state stress in the body.

The key to the method proposed Ref. 4 is the observation that the stress state of the deformable body undergoing large translations and rotations results from the inertia loading on it. With that, they determine the distributions of the inertia force and torque in the body from the inertial linear and angular accelerations, respectively, in accordance with Newton's Second Law and Euler equations. In applying this procedure, they neglect elastic terms in the velocity and acceleration distributions. Thus, the resultant inertia loads are expressed as functions of rigid velocities and accelerations.

In the next step of their development, Banerjee and Dickens rewrite the inertia load as a product of a particular matrix and a column vector. For the inertia *force*, the matrix is 3x12 and is dependent on the spatial coordinates, x, y and z. The column vector contains 12 parameters A_i, $i = 1, \ldots, 12$, which are dependent strictly on the velocities and accelerations of the body. The inertia *torque* is factorized into a constant 3x9 matrix and a time-dependent 9x1 column vector of A_i, $i = 13, \ldots, 21$, which are also functions of rigid velocities and accelerations. Although not stated in Ref. 4, the motivation for this "factorization" is to separate the time-dependent component of the inertia forces from the space-dependent or constant part. By doing so, Banerjee and Dickens are able to construct the geometric stiffness term in the motion equations in two stages. The first stage produces 21 geometric stiffness matrices denoted by $S^{(i)}$.[4] These can be assembled with the standard finite-element procedure from the constant or space-dependent matrix factors of the inertia loads, prior to evaluation of the motion equations. The second stage involves calculating the geometric stiffness term from $S^{(i)}$, A_i and the discrete elastic coordinates.

Let us now comment on the relationship between the approach of Banerjee and Dickens and the other methods. First, we observe that the "nonlinear" strain energy of Eq. (45) is "exact" in the context of small strain deformation. Moreover, it can be rewritten in terms of displacement variables if one expresses stresses in terms of strains and then substitutes for strains in terms of displacements. Following this procedure, one will obtain U_{NL} in the form of third- and fourth-order terms in the displacement gradients. These were mentioned in our discussion of Hanagud and Sarkar's work.

It can be shown that Banerjee and Dickens' expression for the *axial* inertia force is equivalent to Meirovitch's axial component of the internal force density. The latter is given by our Eq. (44) and the former can be obtained from Eqs. (28) and (29) of Ref. 4. Substituting for $x_1 = x$, $x_2 = y = 0$, $x_3 = z = 0$ and $u_i = \mathrm{v}_i$, $u_{i+3} = \omega_i$ for $i = 1, \ldots, 3$, the axial inertia force $f_1^\star$ of Banerjee and Dickens takes the form:

$$\begin{aligned} f_1^\star &= -[A_1 + xA_4]\,dm \\ &= -\left[\dot{\mathrm{v}}_1 + \omega_2 \mathrm{v}_3 - \omega_3 \mathrm{v}_2 - x\left(\omega_2^2 + \omega_3^2\right)\right] dm \end{aligned} \tag{46}$$

Since Meirovitch's internal force density must be interpreted as the internal force per unit length, the equivalence between (46) and (44) follows if one substitutes for dm with $\rho\, dx$. Then, we have

$$f_1^\star = p\, dx \tag{47}$$

where p is as defined by Meirovitch (Eq. (44)). Furthermore, the axial stress in the beam is

$$\sigma_{xx}(x) = \frac{1}{A}\int_x^L p\, dx \tag{48}$$

and if the nonlinear part of the axial strain (19) is simplified to

$$\varepsilon_{NL,xx} = \tfrac{1}{2}\left[\left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial w}{\partial x}\right)^2\right], \tag{49}$$

Eq. (45) applied to the beam reduces to:

$$\begin{aligned} U_{NL} &= \int \varepsilon_{NL,xx}\sigma_{xx}\, dV \\ &= \tfrac{1}{2}\int_0^L \int_x^L p\, d\zeta \left[\left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial w}{\partial x}\right)^2\right] dx \end{aligned} \tag{50}$$

The above, which we derived from U_{NL} of Ref. 4 is identical to Meirovitch's strain energy due to shortening of the projection. In this light we propose to interpret Meirovitch's formulation of the strain energy due to shortening of an elastic beam as a particular application of Banerjee and Dickens' formulation.

To conclude this section, we would like to draw attention to a fundamental approximation made in the development of Banerjee and Dickens. It is that the stresses which contribute to stiffening of the body are only those that are caused by the inertia loading. We believe that the stress in (45) must represent the *complete* stress state in the body which arises from the total loads—inertia and external. Indeed, only then can Eq. (45) be consistent with the general formulation of the strain energy. The approximation for σ suggested in Ref. 4 is certainly valid in the absence of external forces on the body. However, it may not be appropriate for multibody systems in which each body is acted upon by the "external" (to it) interbody forces, even in the absence of external forces on the whole system.

5 Discussion

Based on our review, we propose classification of the different approaches to model geometric stiffening according to these two criteria: (i) kinematic description of the deformation field (criterion DF); (ii) the formulation of the strain energy function (criterion SE). We feel that these represent two most general criteria that can fully characterize a particular approach. The first one is related to the assumed displacement field, which in turn determines the generalized coordinates employed in deriving the motion equations. The second criterion defines the form of the elastic force term in the motion equations.

According to each criterion, we have identified the following possible cases.

Criterion DF. It is suggested that the deformation field can be described by *any* of the following sets of variables:

- DF-1: Three independent elastic translations $\{u, v, w\}$. This description is employed by Likins *et al.*, Hanagud and Sarkar, Meirovitch, and Banerjee *et al.*
- DF-2: Three independent elastic translations $\{u_s, v, w\}$, where recall u_s is the axial displacement which results strictly from the stretching of the deformable body. This description is used by Vigneron, Kaza and Kvaternik, Lips and Modi, and Laskin *et al.*

DF-3: Three independent but non-orthogonal elastic deformations $\{s, v, w\}$ where s denotes the stretch. These coordinates are employed by Kane *et al.*

Criterion SE. We distinguish two basic formulations of the strain energy:

SE-1: The total strain energy is formulated as a function of strains (or displacements) only, where strains are defined with the *nonlinear* strain-displacement relations. This formulation is employed by Likins *et al.*, Vigneron, Kaza and Kvaternik, Lips and Modi and Hanagud and Sarkar as well as, Hughes and Fung, and Kane *et al.* It can be viewed as a *displacement* formulation.

SE-2: The strain energy is subdivided into a "standard linear" contribution and a "nonlinear" part, where the latter is constructed from the stresses or forces in the body and the nonlinear part of strain. This formulation is used by Meirovitch, and Banerjee *et al.* and can be considered as a *force* formulation.

Note that Laskin *et al.* do not compute the strain energy, but derive the elastic forces directly from the stress state. However, since they eventually reformulate these in terms of strains, their method is fundamentally a displacement formulation.

Let us comment on the three options for the kinematic description of the deformation field.

DF-1. The set $\{u, v, w\}$ is the standard set of elastic displacements used in both linear and nonlinear theories of elasticity as well as, structural analysis. It requires no apriori decisions on the foreshortening part of u and in that sense is general.

Since u, v and w represent deformations along three orthogonal axes, they clearly represent *independent* degrees of freedom. However, contrary to what has been previously stated, that does not preclude the fact that part of u, in particular, the foreshortening component, is dependent on v and w. The total axial displacement will remain an independent variable as long as it includes the axial displacement u_s. Furthermore, it is not necessary to either separate u into these two components, or to explicitly account for the coupling between the axial and transverse displacements. In addition, we believe that discretizing u with a standard expansion does not imply "premature linearization." To support this point, let us consider the foreshortening part of u, which for the present purpose we simplify to:

$$u_f = \tfrac{1}{2} \int_0^x \left(\frac{\partial v}{\partial x} \right)^2 dx \tag{51}$$

If the transverse deformation v is expanded as in (3), then upon substitution, u_f becomes:

$$u_f = \tfrac{1}{2} \sum_{j=1}^{\nu} \sum_{k=1}^{\nu} \left[\int_0^x \left(\frac{\partial \phi_{2j}(x)}{\partial x} \right) \left(\frac{\partial \phi_{2k}(x)}{\partial x} \right) dx \right] q_j(t)\, q_k(t) \tag{52}$$

The above clearly represents a summation of terms $\phi_{f,i}(x)\, q_{f,i}(t)$ where $q_{f,i}(t) = q_j(t)\, q_k(t)$ and the space-dependent basis functions are defined accordingly. This summation is of the same form as the standard expansion for u.

As exemplified by the formulations of Likins *et al.* and Hanagud and Sarkar, geometric stiffening *can* be modelled with this option, provided one incorporates the nonlinear strain-displacement relations in the evaluation of strain energy and, hence, the elastic forces.

DF-2. In this case, the foreshortening component of u is explicitly separated from u_s in the axial displacement field. It may be specified in terms of v and w or left as a parameter to be defined by the user. Either case, however, involves making an approximation for u_f, although in the first option it is "known" and quantified prior to deriving the motion equations. Recall that Laskin *et al.* have argued in favor of this option based on their claim that displacements u_s, v and w are of the same order, while u_f is comparatively large. In response to this statement, we would like to draw attention to the results presented in Ref. 6, where Hanagud and Sarkar display the time-histories of axial and transverse displacements from a simulation of a beam spin-up problem. As can be seen from Figures 2a and 2b in Ref. 6, the axial deformation $u_0 = u_s - u_f$ is almost two orders of magnitude smaller than the transverse displacement v. Therefore, u_s cannot be of the same order as v, nor can u_f be significantly larger than v.

DF-3. Employment of the stretch variable is unconventional, and certainly is not a common choice, if made at all. The main advantage of using this variable instead of u is that the strain energy retains its ("linear") quadratic form. However, the resulting expression for kinetic energy (or generalized inertia forces) is more complicated than that based on DF-1 and DF-2 descriptions.

As was demonstrated in the previous sections (at least we hope it was), it is a particular combination of the coordinates and the strain energy formulation that determines how the geometric stiffening is incorporated into the motion equations as well as, what form it appears in. Thus, we will now comment on the two strain energy formulations taken in combination with the different DF options and point out some of the advantages and/or disadvantages of the resulting approaches.

The main advantage of the SE-1 formulation of the strain energy for any description of the deformation field is that one is not required to make any additional assumptions or approximations.

SE-1/DF-1. In this approach, taken by Hanagud and Sarkar, the stiffening term appears in the motion equations through the strain energy and is a nonlinear function of elastic coordinates. In particular, it can be factored into an (elastic)coordinate-dependent second-order stiffness matrix and a column vector of elastic coordinates.

As pointed out by Banerjee and Lemak, evaluation of this stiffness term requires that the stiffness matrix be updated at each time step in the simulation, which may be computationally costly. However, the update does not involve iterations, but is a simple functional evaluation. It should also be pointed out that this approach requires that the axial elastic equation be included in the dynamics model. This is likely to have an adverse effect on the computational efficiency of the simulation, since the axial deformation is usually a high-frequency component.

Finally, we note that the present method corresponds to the third approach identified by Kaza and Kvaternik and as they comment is "the one usually employed for a general three-dimensional rotating body." However, contrary to Kaza and Kvaternik's conclusion, we believe that it does *not* require special consideration, but is the most general. Moreover, this approach can be employed to extend the existing "small" deflection dynamics formulations to incorporate "large" deflection theories.

SE-1/DF-2. With this approach, the foreshortening term is always present in the kinetic energy expression and may or may not appear in the strain energy, depending on the additional

approximations made. In the simplest case, as in Vigneron's formulation, the resultant stiffening term takes a linear form.

This method of determining the stiffening term allows one the option of dropping the axial deformation from the model—not a trivial advantage from the computational point of view. However, it may be less accurate than the previous method, because of the approximation made in assuming a form for the foreshortening displacement.

SE-1/DF-3. This approach produces a linear geometric stiffening term *via* kinetic energy (or generalized inertia force). Compared to the SE-1/DF-1 option, it can be just as accurate, but not as general since the stretch variable can be defined only for a particular type of elastic bodies.

As was shown in §4.3, the SE-2 formulation of the strain energy requires an approximation for the stress field in the body, and hence, in contrast to SE-1, is inherently approximate. Among the works presented in this paper, this formulation has only been used with the DF-1 description of the deformation field. In this case, the geometric stiffening term results from the "nonlinear" strain energy, as in Refs. 4, 15 and 14. With appropriate assumptions, it is linear in elastic variables, but involves rigid-body accelerations and velocities. As for its efficiency, the algorithm presented in Ref. 14 also requires an update of the geometric stiffness matrix (see Eq. (37) in Ref. 4), because of the time-dependent quantities A_i. Moreover, the final motion equations no longer have a symmetric coefficient matrix (because of acceleration dependency of the stiffening term), thereby making evaluation of accelerations computationally more costly.

To summarize, we believe that a description of the deformation field in terms of u, v, w combined with the displacement formulation of the strain energy is the most accurate and general approach. It does not require an approximation of foreshortening, nor the stress state of the body—two critical advantages for applications to multibody systems. A definitive statement on the efficiency of this approach and how it compares to, for instance, Banerjee and Dickens' procedure can only be made through implementation of both methods. Moreover, we would expect the relative efficiency of the two formulations to vary depending on the complexity of the system and the number of elastic degrees of freedom in the model.

6 Concluding Remarks

In this paper, we presented an exposition of several approaches to model the geometric stiffening effect for dynamics simulation of flexible-body systems. Our review included 11 papers published in the period from 1973 to 1991. Although it does not represent a complete literature review of the works that have addressed this issue, it covers a wide range of formulations developed for the problem.

In reviewing these works, we have idenitified two key characteristics of the different methods which allowed us to put forward a general classification for them. We have also established the interrelationships between the various approaches, provided a number of clarifications and new interpretations and offered our opinions on their benefits. It is hoped that this work will contribute to a better understanding of the origin of geometric stiffening and how this effect can be incorporated into the dynamics model of a flexible body undergoing large rigid-body motion.

Acknowledgements

I would like to thank Dr. C. J. Damaren of Royal Roads Military College and Dr. J. Haddow for the help they have given me in tackling the subject of this paper. This research has been funded by the Natural Sciences and Engineering Research Council of Canada.

References

[1] Kane, T. R., Ryan, R. R., and Banerjee, A. K., "Dynamics of a Cantilever Beam Attached to a Moving Base," *J. Guidance, Control, and Dynamics*, Vol. 10, No. 2, 1987, pp. 139-151.

[2] Eke, F. O., and Laskin, R. A., "On the Inadequacies of Current Multi-Flexible Body Simulation Codes," AIAA Paper No. 87-2248, 1987, pp. 79-89.

[3] Padilla, C. E., and von Flotow, A. H., "Nonlinear Strain-Displacement Relations and Flexible Multibody Dynamics," *J. Guidance, Control, and Dynamics*, Vol. 15, No. 1, 1992, pp. 128-136.

[4] Banerjee, A. K., and Dickens, J. M., "Dynamics of an Arbitrary Flexible Body in Large Rotation and Translation," *J. Guidance, Control, and Dynamics*, Vol. 13, No. 2, 1990, pp. 221-227.

[5] London, K. W., "Comment on 'Dynamics of a Cantilever Beam Attached to a Moving Base'," *J. Guidance, Control, and Dynamics*, Vol. 12, No. 2, 1989, pp. 284-286.

[6] Hanagud, S., and Sarkar, S., "Problem of the Dynamics of a Cantilever Beam Attached to a Moving Base'," *J. Guidance, Control, and Dynamics*, Vol. 12, No. 3, 1989, pp. 438-441.

[7] Likins, P. W., Barbera, F. J., and Baddeley, V., "Mathematical Modeling of Spinning Elastic Bodies for Modal Analysis," *AIAA Journal*, Vol. 11, 1973, pp. 1251-1258.

[8] Vigneron, F. R., "Comment on 'Mathematical Modeling of Spinning Elastic Bodies for Modal Analysis," *AIAA Journal*, Vol. 13, 1975, pp. 126-127.

[9] Kaza, K. R., and Kvaternik, R. G., "Nonlinear Flap-Lag-Axial Equations of a Rotating Beam," *Acta Astronautica*, Vol. 15, No. 6, 1977, pp. 1349-1360.

[10] Lips, K. W. and Modi, V. J., "General Dynamics of a Large Class of Flexible Satellite Systems," *Acta Astronautica*, Vol. 17, No. 12, 1980, pp. 1349-1360.

[11] Hughes, P. C. and Fung, J. C., "Lyapunov Stability of Spinning Satellites with Long Flexible Appendages," *Journal of Celestial Mechanics*, Vol. 4, 1971, pp. 295-308.

[12] Laskin, R. A., Likins, P. W., and Longman, R. W., "Dynamical Equations of a Free-Free Beam subject to Large Overall Motions," *Journal of the Astronautical Society*, Vol. XXXI, 1983, pp. 507-528.

[13] Meirovitch, L., *Analytical Methods in Vibrations*, Macmillan, New York, 1967, pp. 440-453.

[14] Meirovitch, L., "Hybrid State Equations of Motion for Flexible Bodies in Terms of Quasi-Coordinates," *J. Guidance, Control, and Dynamics*, Vol. 14, No. 5, 1991, pp. 1008-1013.

[15] Banerjee, A. K., and Lemak, J. M., "Multi-Flexible Body Dynamics Capturing Motion-Induced Stiffness," *ASME J. Applied Mechanics*, Vol. 58, 1991, pp. 766-775.

[16] Kane, T. R., and Levinson, D. A., *Dynamics, Theory and Applications*, McGraw-Hill, New York, 1985.

[17] Brush, D. O., and Almroth, B. O., *Buckling of Bars, Plates, and Shells*, McGraw-Hill, New York, 1975.

[18] Kane, T. R., Ryan, R. R., and Banerjee, A. K., "Reply by Authors to K. W. London," *J. Guidance, Control, and Dynamics*, Vol. 12, No. 2, 1989, pp. 286-287.

Stress Stiffening and Approximate Equations in Flexible Multibody Dynamics

Carlos E. Padilla*
Photon Research Associates, Inc.
Cambridge, Massachusetts 02138
and
Andreas H. von Flotow†
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Abstract

A useful model for open chains of flexible bodies undergoing large rigid body motions, but small elastic deformations, is one in which the equations of motion are linearized in the small elastic deformations and deformation rates. For slow rigid body motions, the correctly linearized, or consistent, set of equations can be compared to prematurely linearized, or inconsistent, equations and to "oversimplified," or ruthless, equations through the use of open loop dynamic simulations. It has been shown that the inconsistent model should never be used, while the ruthless model should be used whenever possible. The consistent and inconsistent models differ by stress stiffening terms. These are due to zeroth-order stresses effecting virtual work via nonlinear strain-displacement terms. In this paper we examine in detail the nature of these stress stiffening terms and conclude that they are significant only when the associated zeroth-order stresses approach "buckling" stresses. Finally it is emphasized that when the stress stiffening terms are negligible the ruthlessly linearized equations should be used.

I. Introduction

In a previous paper[1] it was suggested that a useful model for open chains of flexible bodies undergoing large rigid body motions, but small elastic deformations, would be one in which the equations of motion are linearized in the small elastic deformations and deformation rates. In that paper, it was pointed out that in order to obtain a consistently linearized set of equations of motion it is necessary to make use of nonlinear strain-

---

* Staff Scientist, Dynamics and Control Division.

† Associate Professor, Aeronautics and Astronautics.

displacement relations,[2,3] nonlinear kinematic constraints,[4,5] or several nonlinear geometric or motion stiffness terms appended to the incorrectly linearized equations of motion.[6,7] It is easy to verify that equations of this type, linearized in only some of the states, do not conserve energy.

A presumed goal of simplified equations of motion is to obtain accurate simulation of actual system behavior with a minimum of effort. At present, there is no standardized yardstick with which to determine how various simplified equations measure up. It has been suggested[14] that those simplified equations which come closest to conserving energy should be considered most appropriate. In that paper, it is shown how equations that contain second order terms in the flexible states can be obtained from prematurely linearized velocity expressions.[4,5] But these equations, while perfectly conserving energy, still lack the first order terms (stress stiffening terms) that correctly model the physical nature of the system! In light of this example it becomes clear that conservation of energy, while desirable for numerical reasons, is not a good measure of the adequacy of a simplified set of equations of motion.

Equations in which all nonlinear terms involving the flexible generalized coordinates and their time rates of change are ignored have been termed "ruthlessly linearized".[1] The correctly linearized, or consistent, set of equations was compared to prematurely linearized, or inconsistent, equations and to the ruthless equations by means of open loop dynamics simulations for the case of slow rigid body motions. It was concluded that the inconsistent model should never be used, while the ruthless model should be used whenever possible. We point out that the ruthless model conserves energy. This stems from the fact that the ruthless model can be derived from a set of velocity expressions without any further simplifications, i.e., no terms are dropped after forming the velocity expressions. This ensures that the mass matrix remains positive definite for all possible trajectories.

The inconsistent model results from neglecting certain kinematic relationships between the elastic deformation variables, or what is equivalent, to using linear strain-displacement relations. This results in the absence of certain terms linear in the elastic coordinates and rates. The missing terms have been identified as the so-called geometric stiffness terms of certain rotational dynamics problems. These terms are also known as stress or motion stiffness terms and are not limited to rotational forces. They result from a combination of zeroth-order stresses and nonlinear strain-displacement relations in the virtual energy expressions that result in terms linear in the strain (displacement) variables.[10,7,9]

In the rest of the paper we consider the nature of the stress stiffening terms and their role in the determination of simplified equations of motion. In section II we present the general form of equations of motion for open kinematic chains of flexible bodies. The ruthless model is then defined and it is shown how this model can be derived directly. In section III we consider stress stiffness terms: their derivation; the forces that give rise to them; and their general form. An analytical example is provided using a two link flexible manipulator. Finally, in section IV we consider the importance of these stress stiffening terms for the formulation of equations of motion.

II. Equations of Motion for Chains of Elastic Bodies

The equations of motion of an open chain of elastic bodies can be expressed quite generally as:[8]

$$\begin{bmatrix} M_{RR}(x,q) & M_{RE}(x,q) \\ M_{ER}(x,q) & M_{EE}(x,q) \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & K_{EE} \end{bmatrix} \begin{bmatrix} x \\ q \end{bmatrix} = \begin{bmatrix} T_{ext,R} \\ T_{ext,E} \end{bmatrix} + \begin{bmatrix} F_R(x,q,\dot{x},u) \\ F_E(x,q,\dot{x},u) \end{bmatrix} \tag{1a}$$

$$u = \dot{q} \tag{1b}$$

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_{ext} is a vector of generalized external forces; K_{EE} is a constant stiffness matrix and F is a vector of nonlinear inertial (coriolis and centripetal) forces. Subscripts R and E denote rigid and elastic, respectively.

We are often interested in the important class of systems for which the elastic deformations remain small so we can ignore terms of second and higher order in q and u. In this case, Eq. (1) can be expanded to show more explicitly the form of the nonlinear terms:

$$\begin{bmatrix} M_{RR}(x,q) & M_{RE}(x) \\ M_{ER}(x) & M_{EE}(x) \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & K_{EE} \end{bmatrix} \begin{bmatrix} x \\ q \end{bmatrix} = \begin{bmatrix} T_{ext,R} \\ T_{ext,E} \end{bmatrix} + \begin{bmatrix} F_{R0}(x,\dot{x}) \\ F_{E0}(x,\dot{x}) \end{bmatrix}$$

$$+ \begin{bmatrix} F_{R1}(x,\dot{x}) \\ F^*_{E1}(x,\dot{x},\dot{x}) \end{bmatrix} q + \begin{bmatrix} F_{R2}(x,\dot{x}) \\ F_{E2}(x,\dot{x}) \end{bmatrix} \dot{q} \tag{2}$$

Notice in the above equation that we have lumped the terms in M_{ER} that depend on q together with the F_{E1} term and have denoted the new term by F_{E1}^*.

In Eq. (2), the superscript * terms are the only terms that could contain foreshortening terms, i.e., terms arising from the enforcement of kinematic constraints among the flexible degrees of freedom of the elastic bodies. These terms, which can be obtained equivalently by the use of nonlinear strain-displacement relations in the formation of the inertial velocities of the bodies, are missing when linear strain-displacement relations are used instead, i.e., when the equations are linearized prematurely.[5,6] In the case of a single flexible body, the incorrectly linearized equations can then be fixed through the use of "motion stiffness" matrices.[7]

II.A The Ruthlessly Linearized Model

In a "ruthlessly linearized" model,[1] we simplify the equations of motion for a chain of flexible bodies by ignoring all nonlinear terms that involve the elastic generalized coordinates and rates (q and u), including those terms in the mass matrix which depend on elastic coordinates. In this case Eq. (2) becomes:

$$\begin{bmatrix} M_{RR}(x) & M_{RE}(x) \\ M_{ER}(x) & M_{EE}(x) \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & K_{EE} \end{bmatrix} \begin{bmatrix} x \\ q \end{bmatrix} = \begin{bmatrix} T_{ext,R} \\ T_{ext,E} \end{bmatrix} + \begin{bmatrix} F_{R0}(x,\dot{x}) \\ F_{E0}(x,\dot{x}) \end{bmatrix} \tag{3}$$

As explained in Ref. 1, the ruthless model is a simplification of the consistently linearized equations of motion motivated in part by the so-called rate-linear assumption. This assumption consists in neglecting (coriolis and centripetal) terms nonlinear in the generalized rates, dx/dt and u, for slow motion of the system. In the case of n-link rigid manipulators it has been pointed out[12] that the velocity and acceleration terms of the dynamic equations have the same relative significance at any speed of movement. This indicates that the rate-linear assumption might not be a good one. On the other hand, terms that are nonlinear in the rigid generalized rates and linear in q and u might be negligible for small q and u when compared to similar terms that are constant in q and u.

Whereas the consistently linearized equations of motion are theoretically valid for chains of flexible bodies undergoing fast rigid body motions, but small elastic deformations, the ruthless equations can be said to be valid for chains of flexible bodies undergoing slow rigid body motions and small elastic deformations. How slow these rigid body motions must be is the subject of section IV. Notice that both models can accommodate large configuration changes (kinematic nonlinearities), and that the distinction being made concerns only the magnitudes of the time rates of change of the rigid body generalized coordinates. In the next section, we examine how the ruthless equations of motion can be obtained without having to start with the consistent equations. Before proceeding, however, let us make some clarifications.

Up to this point, we have made little distinction between "rigid body motions" and rigid generalized coordinates and speeds. In fact, what is meant by rigid generalized speeds is the collection of generalized speeds that for each body characterize the motion of a frame of reference from which the small elastic deformations are defined. The requirement of small elastic deformations implicitly defines the frame.[2] More explicit frame definitions can be made and the reader is referred to Ref. 13 for some examples. The point being made here is that for chains of flexible bodies, the so-called rigid generalized speeds do not necessarily correspond to rigid body motions, in the sense of a rigid body mode; and further whether there is a correspondence or not will depend on the choice of reference frame from which the small elastic deformations are described. This will be investigated further in section V when we look at further simplifications of the ruthless model.

II.B Direct Derivation of Ruthlessly Linearized Motion Equations

We are interested in finding a way to obtain ruthless equations without having to start with consistent equations. From the form of Eq. (3), we see that what is needed is some way of obtaining the mass matrix and the vector of coriolis and centripetal terms such that there is no q-dependence or dq/dt -dependence. Clearly, this could be done formally by first obtaining the consistent equations and then dropping all terms containing q and dq/dt. Due to the simplicity of the resulting equations, however, it seems that there should be an easier way of obtaining Eq. (3). There is such a way, and this is in fact one of the advantages of using ruthlessly linearized equations of motion. In the following we illustrate how to attain ruthless equations for a collection of rigid and flexible bodies in a tree topology with no closed loops.

The method is very straightforward and enables the dynamicist or control designer to obtain the ruthless equations of motion for modest chains (as in, say, manipulator applications) analytically. First, form the inertial velocities of the mass centers of the rigid bodies, of points of application of external loads, and of characteristic material particles of flexible bodies. These velocities should be constant in flexible generalized coordinates, though they must be linear in generalized speeds if there are to be any flexible equations of motion. Next, form the partial velocities with respect to the generalized speeds.[11] Note that the term *partial velocities*, coined by Kane, is a convenient way to describe partial differentiation of the velocities with respect to generalized speeds, but its use here in no way indicates that these results are exclusive to Kane's method.[11]

The partial velocities should now be constant in both q and dq/dt, and we are ready to form the mass matrix:

$$M_{ij}(x) = \sum_{k=1}^{N} \int_{k} \mathbf{v}_i^k(x) \cdot \mathbf{v}_j^k(x)\, dm \tag{4}$$

where the subscripts i,j range over all degrees of freedom, rigid and elastic; N is the number of bodies in the chain; V_k is the volume of the k-th body; and $\mathbf{v}_i^k(x)$ represents the i-th partial velocity of a material particle in body k.

In order to easily obtain the nonlinear velocity terms, we first define a pseudo-acceleration. Form the acceleration for each of the inertial velocities previously defined. Note that for purposes of forming the acceleration, the inertial velocities can also be made constant in dq/dt. Once these accelerations are formed, further simplify them by dropping all terms that contain the second time derivative of any generalized coordinate or the first time derivative of the generalized speeds (since these terms have already been included in the mass matrix above). The coriolis and centripetal terms are now obtained by dot multiplying the partial velocities with the corresponding pseudo-accelerations, $\hat{\mathbf{a}}^k$:

$$F_r(x, \dot{x}) = \sum_{k=1}^{N} \int_{V_k} \mathbf{v}_r^k \cdot \mathbf{a}^k \, dm \tag{5}$$

where the subscript r ranges over all degrees of freedom, and F is the vector of nonlinear forces in Eq. (3).

The stiffness matrix can be obtained in a variety of standard ways.[5,10] Determination of the load distribution matrix is achieved as in Kane's method by dot multiplying external loads by the partial velocities (constant in q and dq/dt) of their points of application.

III. Nonlinear Strain-Displacement and Stress Stiffness

Enforcing geometric constraints on the generalized coordinates[5] is equivalent to retaining nonlinear strain-displacement terms in a continuum mechanics formulation of the equations of equilibrium.[9] It is for this reason that appending geometric stiffness matrices[6,7] to the incorrectly linearized equations of motion yields identical results, if done consistently, to proper linearization through consideration of nonlinear geometric constraints on flexible degrees of freedom.

As defined by Ref. 10, geometric stiffness terms are used to account for second order terms in the energy expressions that result in first order effects in the motion equations but which are ignored in a premature linearization formulation. In the context of the finite element method, stresses resulting from all applied external forces are found and the virtual work they effect through the nonlinear terms of the strain expression is found to contribute linear stiffness terms that are critical in stability analysis. This has been recognized by users of finite element codes for some time, in particular as it relates to the inertial forces impressed on rotor blades by "constant" spin rates. The fact that *all* external, internal, and inertial forces should be taken into account when obtaining the stress resultants has perhaps not been made clear in all engineering applications. While some or all of these forces might be unimportant for particular applications, it remains true that for general, particularly dynamics, applications all forces should be considered.

In their paper, for the case of the free motion of a single flexible body, Banerjee and Dickens (Ref. 7) consider only forces due to impressed motions, or inertial forces. They extend the concept of geometric or stress stiffness matrices by considering time-varying inertial forces. This is achieved (see also Ref. 9) by obtaining the geometric stiffness matrices assuming unit values of a set of 21 inertia loads taken one at a time and using a standard finite element code. Invoking linearity, they then proceed to multiply each *motion* stiffness matrix by the instantaneous value of the associated inertia load and to add the resulting matrices together at each instant in time.

The above procedure results in an additional stiffness term being added to the flexible partition of the inconsistently linearized equations (Eq. (2) with the superscript * terms lacking foreshortening terms):

$$K_g q = -\sum_{k=1}^{21} S^{(k)} A_k q \tag{6}$$

K_g above is the generalized motion stiffness for a single flexible body and $S^{(k)}$ is the geometric stiffness matrix corresponding to unit values of the k-th inertia loading, A_k, with the other inertia loadings set to zero. The A_k in Eq. (6) are defined as follows:[7]

$$\begin{aligned}
A_1 &= \dot{u}_1 + u_5 u_3 - u_6 u_2 & A_4 &= -\left(u_5^2 + u_6^2\right) & A_7 &= -\dot{u}_6 + u_4 u_5 & A_{10} &= \dot{u}_5 + u_6 u_4 \\
A_2 &= \dot{u}_2 + u_6 u_1 - u_4 u_3 & A_5 &= \dot{u}_6 + u_4 u_5 & A_8 &= -\left(u_6^2 + u_4^2\right) & A_{11} &= -\dot{u}_4 + u_5 u_6 \\
A_3 &= \dot{u}_3 + u_4 u_2 - u_5 u_1 & A_6 &= -\dot{u}_5 + u_6 u_4 & A_9 &= \dot{u}_4 + u_5 u_6 & A_{12} &= -\left(u_5^2 + u_4^2\right)
\end{aligned} \tag{7}$$

where for simplicity we have ignored rotary inertia effects in the flexible body, thus setting A_k for k=13 to 21 equal to zero (see Ref.7). The u_i (i=1,...,6) are the generalized speeds characterizing the motion of a frame of reference attached to the body and are defined as:[11]

$$\begin{aligned}
u_i &= \mathbf{v}^O \cdot \mathbf{b}_i, \quad i = 1,2,3 \\
u_{i+3} &= \boldsymbol{\omega}^B \cdot \mathbf{b}_i, \quad i = 1,2,3
\end{aligned} \tag{8}$$

where O is the origin of the frame B ($\mathbf{b}_1$,$\mathbf{b}_2$,$\mathbf{b}_3$) attached to the flexible body; $\mathbf{v}^O$ is the inertial velocity of O; and $\boldsymbol{\omega}^B$ is the inertial angular velocity of B.

The above methods rely on a finite element analysis in which the displacement interpolation functions are nonlinear.[19] This is equivalent to considering nonlinear strain-displacement relations in a continuum formulation. Notice that a proper finite element code is capable of generating the correct results (i.e., a formulation of the equations of motion that contains all the motion stiffness terms) without any tampering, but at a high computational cost (see, e.g., Ref. 15). The method of Banerjee and Dickens can be likened to a preprocessing of the equations so that motion stiffness matrices are computed only once. This yields large savings in computational cost.

It is clear that a similar process is viable for all loadings that generate zeroth-order stresses, including joint interconnection forces for chains of flexible bodies. Wallrap and Schwertassek[15] show how this can be done for each body in a chain. Their method requires knowledge of interconnection forces at each instant in time, however, and these depend in general on acceleration terms. The motion equations thus can become implicit in the accelerations. Kim[16] proposes an iterative solution approach for the interconnection forces which they claim converges in very few iteration steps at each integration time step. The

iteration is started with some *a priori* guesses for the accelerations. Their method does not require updating the mass matrix at each iteration step.

III.A Possible Stress Stiffness Matrices

As mentioned earlier, all zeroth-order forces (which will result in zeroth-order stresses) can potentially generate stress stiffness matrices. Consequently, the following forces must be considered:

1) applied loads,

2) gravity loads,

3) motion-induced forces: both body inertial forces and interconnection forces.

The general form of the resulting stress stiffness matrices is given by:

$$[K_G]_{Body\ k} = \int_{V_k} [\Phi_{i,j}\, \Phi_{j,m}] \sigma_O \, dV_k$$

where $(.)i,j$ represents the partial derivative in the j-th direction of (.) in the i-th direction.

III.B Two-Flexible-Link Arm Example

In Ref. 3 the consistently linearized equations of motion for a planar, revolute, two-flexible-link manipulator arm are derived. Nonlinear strain-displacement relations for a bar are used in the modelling of each link as a Bernoulli-Euler beam. The resulting equations contain all terms linear in the flexible coordinates and their time rates of change. In particular, the stress stiffness terms are included in an implicit way. In this section we rewrite these terms due to the nonlinear strain-displacement relations (also dubbed foreshortening terms) and show explicitly that they indeed possess the general form shown in the previous section.

The transverse link displacements u_1 and u_2 of the shoulder and elbow links, respectively, are discretized using an assumed modes expansion:

$$u_1 = \sum_{i=1}^{n} \phi_i(x) q_i(t)$$

$$u_2 = \sum_{i=1}^{m} \psi_i(y) p_i(t)$$

The absolute shoulder angular position is given by θ, and the relative angular position of the elbow is given by β.

For the shoulder link, two zeroth-order forces yield stress stiffness matrices. The first one is due to the body distributed centripetal load:

$$[K_{G_1}]_{L_1} = \left[\frac{m_1 l_1}{2}\int_0^{l_1}\phi_i'\phi_j' dx - \int_0^{l_1}\rho_1\frac{x^2}{2}\phi_i'\phi_j' dx\right]\dot{\theta}^2$$

The second stress stiffness matrix is due to the axial component (along the neutral axis of the undeformed shoulder link) of the interconnection force at the elbow joint:

$$[K_{G_2}]_{L_1} = \left[\int_0^{l_1}\phi_i'\phi_j' d\sigma\right]\frac{1}{2}m_2 l_2[\sin(\beta)(\ddot{\theta}+\ddot{\beta})+\cos(\beta)(\dot{\theta}+\dot{\beta})^2]$$

In the above equations ρ_1 and ρ_2 are the mass per unit length of the shoulder and elbow links, respectively, and $m_1=\rho_1 l_1$, $m_2=\rho_2 l_2$ are the respective link masses.

The stress stiffness matrices for the elbow link can be considered to be due either to the inertial body forces, or to the axial component (along the neutral axis of the undeformed elbow link) of the interconnection force at the elbow joint:

$$[K_{G_1}]_{L_2} = \left[\frac{m_2 l_2}{2}\int_0^{l_2}\psi_i'\psi_j' dy - \int_0^{l_2}\rho_2\frac{y^2}{2}\psi_i'\psi_j' dy\right](\dot{\theta}+\dot{\beta})^2$$

$$[K_{G_2}]_{L_2} = \left[m_2\int_0^{l_2}\psi_i'\psi_j' dy - \int_0^{l_2}\rho_2 y\psi_i'\psi_j' dy\right]l_1[-\sin(\beta)\ddot{\theta}+\cos(\beta)\dot{\theta}^2]$$

IV. Relative Significance of Stress Stiffening Terms

Returning to the question of simplified equations of motion, it is natural to ask when, and if, these stress stiffening terms are important. In fact, whenever these terms are important, a consistent formulation of the equations of motion is mandatory in order to model the physical system. In this case, no further simplifications are possible, beyond the linearization in the flexible states. It was shown in a previous paper[17] that ruthlessly

linearized equations are valid (i.e., accurately model the system), precisely when these stress stiffness terms are negligible. Furthermore, it can be surmised from the detail term by term comparisons made in that paper that stress stiffness terms are significant only when the appropriate zeroth-order stress is comparable to the buckling stress, i.e., when the corresponding load approaches the buckling load for the structure. This can be surmised further by considering a static stability analysis and comparing work due to loading stresses with elastic energy (see for example Refs. 10 and 18).

VII. Conclusions

The form of ruthlessly linearized equations of motion for open chains of flexible bodies was presented. A method was outlined that allows for the easy determination of the ruthless equations of motion for a given system. We discussed the derivation of stress stiffness terms which result from nonlinear strain-displacement relations and zeroth-order stresses. These terms are needed to obtain a consistently linearized set of equations. The form of these terms was provided and it was shown that all zeroth-order loadings, not just body inertial loadings, need to be considered when determining the stress stiffening terms. It was pointed out that these terms are significant only when the associated stresses reach the level of buckling stresses. This coincides with the previously investigated limits on the validity of the ruthless equations of motion. It is concluded that ruthlessly linearized equations of motion should be used whenever stress stiffening terms are negligible.

References

[1]Padilla, C. E., and von Flotow, A. H., "Nonlinear Strain-Displacement Relations and Flexible Multibody Dynamics," *Journal of Guidance, Control, and Dynamics.*, Vol. 15, No. 1, January-February, 1992, pp. 128-136.

[2]Laskin, R. A., Likins, P. W., and Longman, R. W., "Dynamical Equations of a Free-Free Beam Subject to Large Overall Motions," *The Journal of the Astronautical Sciences*, Vol. 31, No. 4, October-December 1983, pp. 507-527.

[3]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. Also SSL Report # 6-89.

[4]Ryan, R. R., "Flexibility Modeling Methods in Multibody Dynamics," AAS Paper 87-431, presented at the AAS/AIAA Astrodynamics Specialist Conference, Kalispell, Montana, August, 1987.

[5]Kane, T. R., Ryan, R. R., and 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.

[6]Spanos, J., and Laskin, R. A., "Geometric Nonlinear Effects in Simple Rotating Systems," *Proceedings of the Workshop on Multibody Simulation*, JPL D-5190, Vol. 1, April 15, 1988, pp. 191-218.

[7]Banerjee, A. K., and Dickens, J. M., "Dynamics of an Arbitrary Flexible Body Undergoing Large Rotation and Translation with Small Vibration," AIAA Paper 89-1308, April 1989.

[8]Hughes, P. C., and Sincarsin, G. B., "Dynamics of an Elastic Multibody Chain: Part A - Body Motion Equations, Part B - Global Dynamics," *Dynamics and Stability of Systems*, Vol. 4, Nos. 3 & 4, 1989, pp. 209-244.

[9]Wallrapp, O., Santos, J., and Ryu, J. "Superposition Method for Stress Stiffening in Flexible Multibody Dynamics," *The Dynamics of Flexible Structures in Space: Proceedings of the International Conference on Dynamics*, 1990, pp. 233-247.

[10]Cook, R. D., *Concepts and Applications of Finite Element Analysis*, McGraw-Hill, 1985, pp. 331-341.

[11]Kane, T. R., and Levinson, D. A., *Dynamics: Theory and Applications*, McGraw -Hill Book Company, New York, 1985.

[12]Hollerbach, J. M., "Dynamic Scaling of Manipulator Trajectories," *Journal of Dynamic Systems, Measurement, and Control*, Vol. 106, March 1984, pp. 102-106.

[13]Likins, P. W., "Analytical Dynamics and Nonrigid Spacecraft Simulation," NASA Technical Report 32-1593, July, 1974.

[14]Sharf, I., and Damaren, C., "Simulation of Flexible-Link Manipulators: Basis Functions and Nonlinear Terms in the Motion Equations," *Proceedings of the IEEE Robotics and Automation Conference*, Nice, France, Vol. 3, May 1992, pp. 1956-1963.

[15]Wallrapp, O., and "Schwertassek, R., "Representation of Geometric Stiffening in Multibody System Simulation," *International Journal for Numerical Methods in Engineering*, Vol. 32, 1991, pp. 1833-1850.

[16]Ryu, J., Kim, Sung-Soo, and Kim, Sang-Sup, "An Efficient Computational Method for Dynamic Stress Analysis of Flexible Multibody Systems," *Journal of Computer and Structures*, Vol. 42, No. 6, 1992, pp. 969-977.

[17]Padilla, C.E., and von Flotow, A.H., "Further Approximations in Flexible Multibody Dynamics," *Proceedings of the 32nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, Baltimore, MD, April, 1991.

[18]Timoshenko, S.P., Young, D.H., and Weaver, W., *Vibration Problems in Engineering*, 4th ed., Wiley and Sons, 1974.

[19]Bathe, K.L., *Finite Element Procedures in Engineering Analysis*, Prentice-Hall, Englewood Cliffs, N.J., 1982.

Some Experiences with Krylov Vectors and Lanczos Vectors

Roy R. Craig, Jr.*, Tzu-Jeng Su†, and Hyoung M. Kim ‡

This paper illustrates the use of Krylov vectors and Lanczos vectors for reduced-order modeling in structural dynamics and for control of flexible structures. Krylov vectors and Lanczos vectors are defined and illustrated, and several applications that have been under study at The University of Texas at Austin are reviewed: model reduction for undamped structural dynamics systems, component mode synthesis using Krylov vectors, model reduction of damped structural dynamics systems, and one-sided and two-sided unsymmetric block-Lanczos model-reduction algorithms.

1. Introduction

In recent years extensive research has been carried out on Lanczos eigensolution algorithms (see, for example, Refs. [1, 2]). Nour-Omid and Clough [3] demonstrated the usefulness of Lanczos vectors in the analysis of the dynamic response of structures, and Frisch [4] included Lanczos vectors among the sets of Ritz vectors that are available in the DISCOS multibody code. Research has involved single-vector and block-vector methods, algorithms based on first-order equations of motion and algorithms based on second-order equations, and algorithms for unsymmetric matrices as well as for symmetric matrices.

Following a brief introduction to the physical meaning of Krylov vectors and Lanczos vectors, this paper summarizes several applications that have been under study recently at The University of Texas at Austin.

Structural dynamicists are all familiar with the fact that the modes of free vibration of an undamped structure (modeled, for example, as an n-degree-of-freedom finite element model) satisfy the algebraic eigenproblem

$$K\phi_r = \lambda_r M\phi_r \qquad r = 1, 2, \ldots, n \tag{1}$$

*John J. McKetta Energy Professor of Engineering, Department of Aerospace Engineering and Engineering Mechanics, The University of Texas at Austin, Austin, TX 78712-1085.

†National Research Council Research Associate, NASA Langley Research Center, Spacecraft Dynamics Branch, Hampton, VA 23665-5225.

‡Technical Specialist, McDonnell Douglas Space Systems Company, Space Station Division, Houston, TX 77062.

where K, M, λ_r, and ϕ_r are, respectively, the stiffness matrix, mass matrix, r-th eigenvalue, and r-th eigenvector. However, Krylov vectors and Lanczos vectors are not as well known as are eigenvectors. Note that Eq. (1) is basically an equilibrium equation relating elastic restoring forces, $K\phi_r$, to inertia forces $\omega^2 M\phi_r$, and recall that a modification of Eq. (1), namely

$$K\phi^{(j+1)} = M\phi^{(j)} \tag{2}$$

is the basis for the inverse iteration method for computing eigenvalues and eigenvectors [5]. The vector ϕ in Eq. (2) converges to the fundamental mode (eigenvector) or, with suitable orthogonalization with respect to lower-frequency modes, to a higher-frequency mode. Equation (2) states that, given a vector $\phi^{(j)}$, a new vector $\phi^{(j+1)}$ may be generated by solving for the *static deflection produced by the inertia forces associated with* $\phi^{(j)}$, that is,

$$\phi^{(j+1)} = K^{-1}\left[M\phi^{(j)}\right] \tag{3}$$

(assuming that K is nonsingular). Equation (3) provides a basis for defining a Krylov subspace.

Given a starting vector $\phi_K^{(1)}$, the vectors $\phi_K^{(j)}$ are said to form a *Krylov subspace* of order p, $1 \leq p \leq n$, given by

$$\Phi_K^{(p)} \equiv \left[\phi_K^{(1)},\ [K^{-1}M]\phi_K^{(1)},\ [K^{-1}M]^2\phi_K^{(1)},\ \ldots,\ [K^{-1}M]^{(p-1)}\phi_K^{(1)}\right] \tag{4}$$

Lanczos vectors differ from the Krylov vectors defined in Eq. (4) in that each Lanczos vector is made orthogonal to the previous two Lanczos vectors, and it can be shown that this makes the present Lanczos vector (theoretically) orthogonal to *all* prior vectors.

The following algorithm may be used to compute Lanczos vectors for an undamped system. Let $\phi_L^{(0)} = 0$, and select a starting vector $\phi_L^{(1)}$. The algorithm to compute the Lanczos vector $\phi_L^{(j+1)}$ may be expressed by the following equations:

$$\begin{aligned} \overline{\psi}^{(j)} &= K^{-1}M\phi_L^{(j)} \\ \psi^{(j)} &= \overline{\psi}^{(j)} - \alpha_j\phi_L^{(j)} - \beta_j\phi_L^{(j-1)} \end{aligned}$$

where

$$\begin{aligned} \alpha_j &= \phi_L^{(j)T} M\overline{\psi}^{(j)} \\ \phi_L^{(j+1)} &= \frac{1}{\beta_{j+1}}\psi^{(j)} \end{aligned} \tag{5}$$

and

$$\beta_{j+1} = \left(\psi^{(j)^T} M \psi^{(j)}\right)^{\frac{1}{2}}$$

Note that the only step in the algorithm, Eq. (5), that distinguishes Lanczos vectors from Krylov vectors is the orthogonalization step, and note that the mass matrix M is used in the orthonormalization step above.

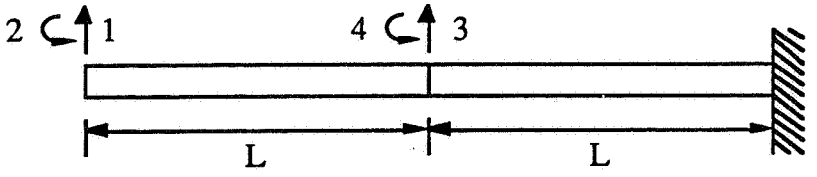


Figure 1. A 4-DOF Cantilever Beam Finite Element Model.

Figure 1 shows a four degree-of-freedom (4-DOF) finite element model for a cantilever beam that is used to illustrate Krylov vectors and Lanczos vectors. Figure 2 shows the four Krylov vectors generated from a starting vector that is the static deflection due to a unit force at the tip. Figure 3 shows the four Lanczos vectors for the cantilever beam of Fig. 1. The starting vector, $\phi_L^{(1)}$, is the same static deflection due to a unit tip force which was used as the starting Krylov vector in Fig. 2. Because the starting vector produces a shape that resembles closely the fundamental mode of the cantilever beam, the subsequent Lanczos vectors in Fig. 3 resemble the second

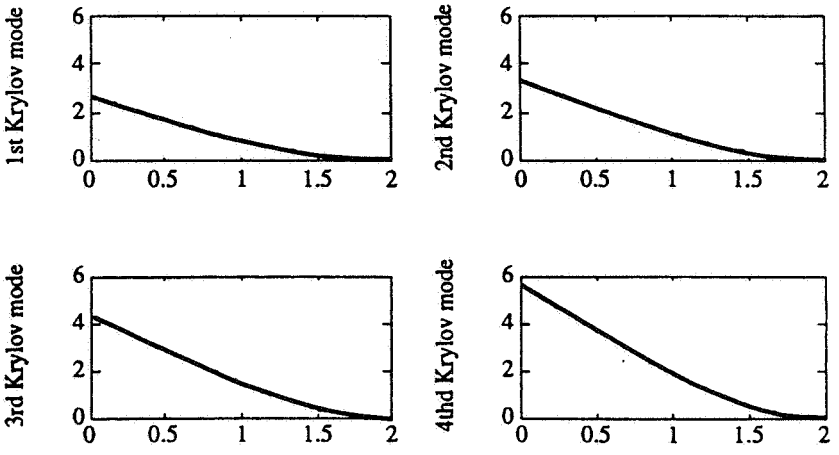


Figure 2. The Four Krylov Vectors for the 4-DOF Cantilever Beam.

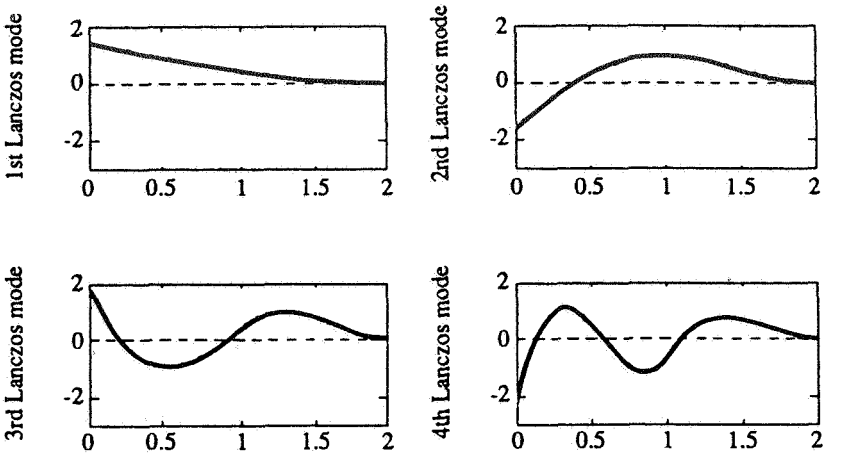


Figure 3. The Four Lanczos Vectors for the 4-DOF Cantilever Beam.

through fourth normal modes (eigenvectors).

A *Krylov subspace* of order p is a p-dimensional vector space spanned by the columns of the matrix

$$\Phi^{(p)} = \left[\phi,\ A\phi,\ A^2\phi,\ \ldots,\ A^{(p-1)}\phi\right] \tag{6}$$

where A is an $n \times n$-dimensional matrix and ϕ is any n-dimensional starting vector. Depending on the choice of A and ϕ, the basis vectors in Eq. (6) are either linearly dependent for some $p < n$, or they span the entire n-dimensional space when $p = n$. If the vector ϕ is replaced by a matrix with q linearly-independent columns rather than a single column, the subspace Φ is called a *block-Krylov subspace.*

2. Lanczos Model Reduction for Undamped Structural Dynamics Systems (Refs. [6,7])

Some studies have shown that Krylov/Lanczos-based reduced-order models provide an alternative to normal-mode (eigenvector) reduced-order models in application to structural control problems. For such applications, an undamped structural dynamics system can be described by the input-output equations

$$\begin{aligned} M\ddot{x} + Kx &= Pu \\ y &= Vx + W\dot{x} \end{aligned} \tag{7}$$

where $x \in R^n$ is the displacement vector; $u \in R^l$ is the input vector; $y \in R^m$ is the output measurement vector; M and K are the system mass and stiffness matrices;

P is the force distribution matrix; and V and W are the displacement and velocity sensor distribution matrices. In most practical cases, we can assume that l and m are much smaller than n.

Model reduction of structural dynamics systems is usually based on the Rayleigh-Ritz method of selecting an $n \times r$ transformation matrix L such that

$$x = L\bar{x} \tag{8}$$

where $\bar{x} \in R^r$ $(r < n)$ is the reduced-order vector of (physical or generalized) coordinates. Then, the reduced system equation is

$$\begin{aligned} \overline{M}\ddot{\bar{x}} + \overline{K}\bar{x} &= \overline{P}u \\ y &= \overline{V}\bar{x} + \overline{W}\dot{\bar{x}} \end{aligned} \tag{9}$$

where $\overline{M} = L^T M L$, $\overline{K} = L^T K L$, $\overline{P} = L^T P$, $\overline{V} = VL$, and $\overline{W} = WL$. The projection matrix L can be chosen arbitrarily. Here, however, we choose L to be formed by a particular set of Krylov vectors. It is shown in Ref. [7] that the resulting reduced-order model matches a set of parameters called *low-frequency moments.*

For a general linear system

$$\begin{aligned} \dot{z} &= Az + Bu \qquad z \in R^n,\ u \in R^l \\ y &= Cz \qquad\qquad y \in R^m \end{aligned} \tag{10}$$

the low-frequency moments are defined by $CA^{-i}B$, $i = 1, 2, \ldots$, which are the coefficient matrices in the Taylor series expansion of the system transfer function [8,9]. Applying the Fourier transform to Eq. (7a) yields the frequency response solution $X(\omega) = (K - \omega^2 M)^{-1} P U(\omega)$, with $X(\omega)$ and $U(\omega)$ the Fourier transforms of x and u. If the system is assumed to have no rigid-body motion, then a Taylor expansion of the frequency response around $\omega = 0$ is possible. Thus,

$$X(\omega) = (I - \omega^2 K^{-1}M)^{-1} K^{-1} P U(\omega) = \sum_{i=0}^{\infty} \omega^{2i} (K^{-1}M)^i K^{-1} P U(\omega) \tag{11}$$

Combining Eq. (7b) and Eq. (11), the system output frequency response can be expressed as

$$Y(\omega) = \sum_{i=0}^{\infty} [V(K^{-1}M)^i K^{-1}P + j\omega W(K^{-1}M)^i K^{-1}P]\omega^{2i} U(\omega) \tag{12}$$

In these expressions, $V(K^{-1}M)^i K^{-1}P$ and $W(K^{-1}M)^i K^{-1}P$ play roles similar to that of low-frequency moments in the first-order state-space formulation. To obtain the reduced-order model of Eq. (9) let

$$span\ \{L\} = span\ \{L_P\ L_V\ L_W\} \tag{13}$$

where

$$
\begin{aligned}
L_P &= \left[\; K^{-1}P \;\; (K^{-1}M)K^{-1}P \;\; \cdots \;\; (K^{-1}M)^p K^{-1}P \;\right] \\
L_V &= \left[\; K^{-1}V^T \;\; (K^{-1}M)K^{-1}V^T \;\; \cdots \;\; (K^{-1}M)^q K^{-1}V^T \;\right] \\
L_W &= \left[\; K^{-1}W^T \;\; (K^{-1}M)K^{-1}W^T \;\; \cdots \;\; (K^{-1}M)^s K^{-1}W^T \;\right]
\end{aligned}
\tag{14}
$$

for $p,\ q,\ s \geq 0$. Then the reduced system matches the low frequency moments $V(K^{-1}M)^i K^{-1}P$ for $i = 0,\ 1,\ \ldots,\ p+q+1$ and $W(K^{-1}M)^i K^{-1}P$, for $i = 0,\ 1,\ 2,\ \ldots,\ p+s+1$.

The L_P matrix above is the *generalized controllability matrix,* and the L_V and L_W matrices are the *generalized observability matrices* of the dynamic system described by Eq. (7). The vectors contained in L_P are *Krylov vectors* that are generated in block form by

$$
\begin{aligned}
Q_1 &= K^{-1}P \\
Q_{i+1} &= K^{-1}MQ_i
\end{aligned}
$$

The first vector block, $K^{-1}P$, is the system's static deflection due to the force distribution P. The vector block Q_{i+1} can be interpreted as the static deflection produced by the inertia force associated with the Q_i. If only the dynamic response simulation is concerned, we would choose $L = L_P$. In this case, the reduced model matches $p+1$ low-frequency moments. As to the vectors in L_V and L_W, a physical interpretation such as the "static deflection due to sensor distribution" may be inadequate. However, from an input-output point of view, L_V, L_W, and L_P are equally important as far as parameter-matching of the reduced-order model is concerned.

Based on Eq. (13), the following algorithm may be used to generate a Krylov basis that produces a reduced-order model with the stated parameter-matching property.

Krylov/Lanczos Algorithm

(1) *Starting block of vectors*:
- (a) $Q_0 = 0$
- (b) $R_0 = K^{-1}\tilde{P}$, $\quad \tilde{P} =$ *linearly-independent portion of* $[P\ V^T\ W^T]$
- (c) $R_0^T K R_0 = U_0 \Sigma_0 U_0^T$ (*singular-value decomposition*)
- (d) $Q_1 = R_0 U_0 \Sigma_0^{-\frac{1}{2}}$ (*normalization*)

(2) *For* $j = 1,\ 2,\ \ldots,\ k-1$, *repeat*:
- (e) $\overline{R}_j = K^{-1}MQ_j$
- (f) $R_j = \overline{R}_j - Q_j A_j - Q_{j-1}B_j$ (*orthogonalization*)

$A_j = Q_j^T K \overline{R}_j, \quad B_j = U_{j-1}\Sigma_{j-1}^{\frac{1}{2}}$

(g) $R_j^T K R_j = U_j \Sigma_j U_j^T$ (*singular-value decomposition*)

(h) $Q_{j+1} = R_j B_{j+1}^{-T} = R_j U_j \Sigma_j^{-\frac{1}{2}}$ (*normalization*)

(3) *Form the k-block projection matrix* $L = \left[\begin{array}{cccc} Q_1 & Q_2 & \cdots & Q_k \end{array} \right]$.

This algorithm is a Krylov algorithm, because the L matrix is generated by a Krylov recurrence formula (Step e). It is a Lanczos algorithm because the orthogonalization scheme is a three-term recursion scheme (Step f). Although the three-term recursion scheme is a special feature of the Lanczos algorithm, in practice, complete reorthogonalization or selective reorthogonalization is necessary to prevent the loss of orthogonality [10–12].

If the projection matrix L generated by the above algorithm is employed to perform model reduction, then the reduced-order model matches the low-frequency moments $V(K^{-1}M)^i K^{-1}P$ and $W(K^{-1}M)^i K^{-1}P$, for $i = 0, 1, 2, \ldots, 2k-1$. It can also be shown that the reduced-order model approximates the lower natural frequencies of the full-order model.

One interesting feature of the transformed system equation in Krylov/Lanczos coordinates is that it has a special form. Because of the special choice of starting vectors, K-orthogonalization, and three-term recurrence, the transformed system equation has a mass matrix in block-tridiagonal form, a stiffness matrix equal to the identity matrix, and force distribution and measurement distribution matrices with nonzero elements only in the first block. The transformed system equation has the form

$$\begin{bmatrix} \times & \times & & & & & \\ \times & \times & \times & & & & \\ & \times & \times & \cdot & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & \cdot & \cdot & \times \\ & & & & & \times & \times \end{bmatrix} \ddot{\bar{x}} + \bar{x} = \begin{Bmatrix} \times \\ 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{Bmatrix} u \tag{15}$$

$$y = [\ \times \ \ 0 \ \ 0 \ \ \cdots \ \ 0\ ]\bar{x} + [\ \times \ \ 0 \ \ 0 \ \ \cdots \ \ 0\ ]\dot{\bar{x}}$$

where $\times$ denotes the location of nonzero elements. This special form reflects the structure of a tandem system (Fig. 4), in which only subsystem S_1 is directly controlled and measured while the remaining subsystems, S_i, $i = 2, 3, \ldots$, are excited through chained dynamic coupling. In control applications, as depicted in Ref. [13], this tandem structure of the dynamic equation eliminates the control spillover and the observation spillover, but there is still dynamic spillover. For dynamic response calculations, the block-tridiagonal form can lead to an efficient time-step solution and can save storage.

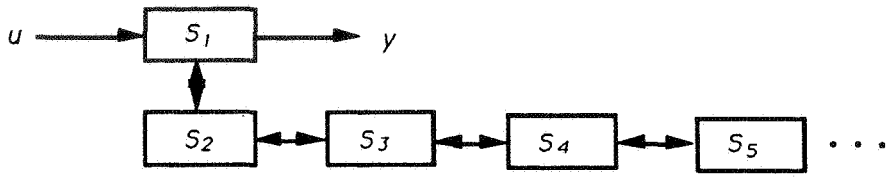


Figure 4. The Structure of a Tandem System.

3. Lanczos Model Reduction for Damped Structural Dynamics Systems (Refs. [6,13])

The previous model-reduction strategy can be extended to damped structural dynamics systems, which are described by the linear input-output equations

$$\begin{aligned} M\ddot{x} + D\dot{x} + Kx &= Pu \\ y &= Vx + W\dot{x} \end{aligned} \tag{16}$$

To arrive at an algorithm for constructing a reduced-order model that matches low-frequency moments, it is easier to start from the first-order formulation. The first-order differential equation equivalent to Eq. (16) can be expressed as

$$\begin{aligned} &\begin{bmatrix} D & M \\ M & 0 \end{bmatrix} \begin{Bmatrix} \dot{x} \\ \ddot{x} \end{Bmatrix} + \begin{bmatrix} K & 0 \\ 0 & -M \end{bmatrix} \begin{Bmatrix} x \\ \dot{x} \end{Bmatrix} = \begin{Bmatrix} P \\ 0 \end{Bmatrix} u \\ &y = [V \; W] \begin{Bmatrix} x \\ \dot{x} \end{Bmatrix} \end{aligned} \tag{17}$$

or

$$\begin{aligned} \hat{M}\dot{z} + \hat{K}z &= \hat{P}u \\ y &= \hat{V}z \end{aligned} \tag{18}$$

with

$$\hat{M} = \begin{bmatrix} D & M \\ M & 0 \end{bmatrix}, \quad \hat{K} = \begin{bmatrix} K & 0 \\ 0 & -M \end{bmatrix}, \quad \hat{P} = \begin{Bmatrix} P \\ 0 \end{Bmatrix}, \quad \hat{V} = \begin{bmatrix} V & W \end{bmatrix} \tag{19}$$

It is possible to reduce Eq. (18) to the standard first-order state-space form and to derive a projection subspace based on this standard state space form. However, as shown in Ref. [13], there are significant advantages in using the generalized first-order form of Eq. (18). This leads to the following recurrence formula for the Krylov blocks:

$$\begin{bmatrix} Q_{j+1}^d \\ Q_{j+1}^v \end{bmatrix} = \begin{bmatrix} -K^{-1}D & -K^{-1}M \\ I & 0 \end{bmatrix} \begin{bmatrix} Q_j^d \\ Q_j^v \end{bmatrix} \tag{20}$$

Superscripts d and v denote displacement and velocity portions of the vector, respectively. The matrix containing the generated vector sequence is called a Krylov matrix. It has the form

$$\begin{bmatrix} Q_1^d & Q_2^d & Q_3^d & \cdots \\ Q_1^v & Q_1^d & Q_2^d & \cdots \end{bmatrix}$$

Krylov subspaces that are generated by Eq. (20) and that have the above form produce a projection subspace L that has the desired moment-matching property [13]. Let Let

$$L_{\hat{P}} = \begin{bmatrix} Q_1^d & Q_2^d & Q_3^d & \cdots & Q_p^d \\ 0 & Q_1^d & Q_2^d & \cdots & Q_{p-1}^d \end{bmatrix} \tag{21a}$$

be the sequence of vectors generated by Eq. (20) with $\hat{K}^{-1}\hat{P}$ the starting block of vectors, i.e., $Q_1^d = K^{-1}P$, $Q_1^v = 0$, and let

$$L_{\hat{V}} = \begin{bmatrix} P_1^d & P_2^d & P_3^d & \cdots & P_q^d \\ P_1^v & P_1^d & P_2^d & \cdots & P_{q-1}^d \end{bmatrix} \tag{21b}$$

be the subspace of vectors generated by Eq. (20) with $\hat{K}^{-1}\hat{V}$ the starting block of vectors, i.e., $P_1^d = K^{-1}V^T$, $P_1^v = -M^{-1}W^T$. If the projection matrix L is chosen such that

$$span\ \{L\} = span\ \left\{ Q_1^d \ \cdots \ Q_p^d \ \ P_1^d \ \cdots \ P_q^d \ \ P_1^v \right\} \tag{22}$$

then the reduced-order model of the damped structural dynamics system matches the system parameters $\hat{V}(\hat{K}^{-1}\hat{M})^i\hat{K}^{-1}\hat{P}$, for $i = 0,\ 1,\ \ldots,\ p+q-1$. Reference [13] provides a Lanczos algorithm, similar to the above algorithm for undamped systems, that produces the desired projection matrix L. The vectors are K-normalized.

Reference [13] contains a model-reduction impulse response example and an example that illustrates the use of the Krylov/Lanczos reduced model for flexible structure control. Figure 5 shows the 48-DOF plane truss structure used in the model-reduction example, and Fig. 6 shows the impulse response based on eight Krylov vectors versus the impulse response based on the original full-order (48-DOF) model.

4. A Block-Krylov Component Synthesis Method for Structural Model Reduction (Ref. [14])

The previous discussions of Krylov/Lanczos model reduction have been applied to complete structures. Reference [14] describes a block-Krylov model reduction algorithm for structural components, providing "component modes" comparable to those that are utilized in Refs. [15–19]. The Krylov model-reduction methods described in Ref. [14] should also be applicable to flexible multibody dynamics formulations.

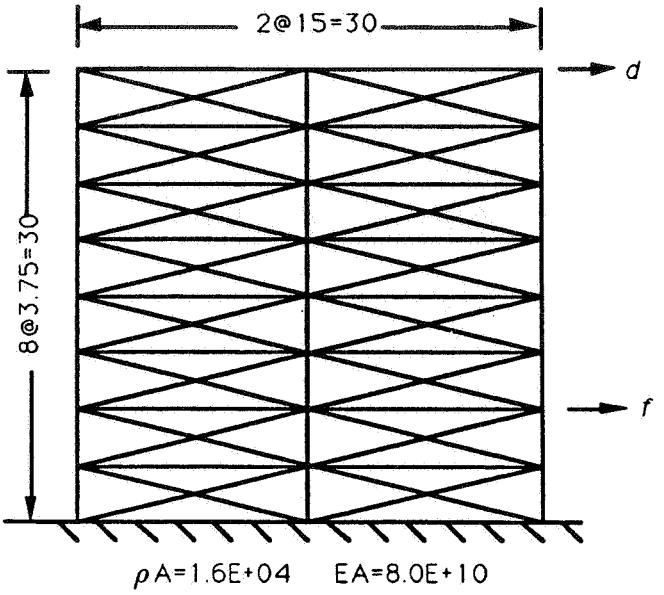


Figure 5. Details of a Plane Truss Structure for Model Reduction Example.

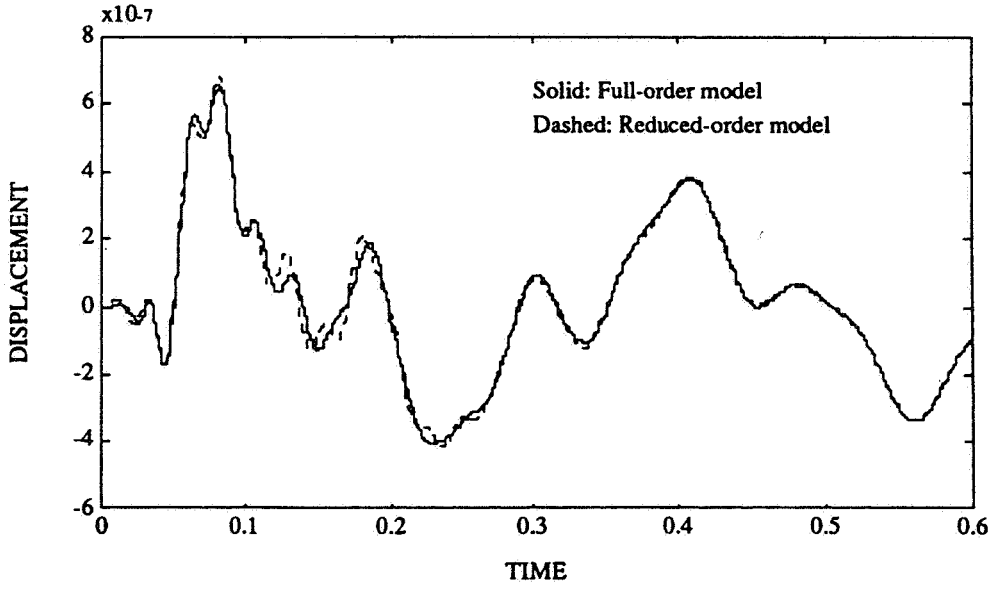
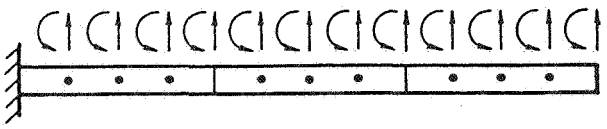
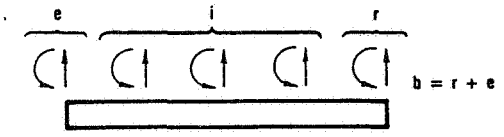


Figure 6. Impulse Response: Eight Damped Krylov Modes and Exact Solution.



a. Structural Component and the Complete System.



b. Interior (i) and Boundary (b) Coordinates of a Component.

Figure 7. A Typical Component and Coupled System.

Figure 7 shows a typical component and a corresponding system of coupled components. The equation of motion for a single undamped component can be written in the partitioned form

$$\begin{bmatrix} M_{ii} & M_{ib} \\ M_{bi} & M_{bb} \end{bmatrix} \begin{Bmatrix} \ddot{x}_i \\ \ddot{x}_b \end{Bmatrix} + \begin{bmatrix} K_{ii} & K_{ib} \\ K_{bi} & K_{bb} \end{bmatrix} \begin{Bmatrix} x_i \\ x_b \end{Bmatrix} = \begin{Bmatrix} f_i \\ f_b \end{Bmatrix} \tag{23}$$

Reference [14] describes both free-interface Krylov modes (related to the Rubin method of Refs. [17,18]) and fixed-interface Krylov modes (related to the method of Hurty [19] and Craig and Bampton [15]). Only the latter will be reviewed here.

A *constraint mode* is defined as the static deflection of a structure when a unit displacement is applied to one coordinate of a specified set of coordinates, while the remaining coordinates of that set are restrained and the remaining degrees of freedom of the structure are force-free. The starting block of vectors for the *fixed-interface Krylov component synthesis method* consists of constraint modes relative to the boundary coordinates, b. That is,

$$\begin{bmatrix} Q_1^i \\ Q_1^b \end{bmatrix} = \begin{bmatrix} -K_{ii}^{-1} K_{ib} \\ I_{bb} \end{bmatrix} [I_{bb}] = \begin{bmatrix} -K_{ii}^{-1} K_{ib} \\ I_{bb} \end{bmatrix} \tag{24}$$

Then, the fixed-interface recurrence formula

$$\begin{bmatrix} Q_{j+1}^i \\ Q_{j+1}^b \end{bmatrix} = \begin{bmatrix} \left[K_{ii}^{-1} M_{ii}\right] & \left[K_{ii}^{-1} M_{ib}\right] \\ 0 & 0 \end{bmatrix} \begin{bmatrix} Q_j^i \\ Q_j^b \end{bmatrix} \tag{25}$$

generates the successive blocks of *fixed-interface Krylov modes.*

The reduced, transformed equations of motion for a component are

$$\overline{M}\ddot{\bar{x}} + \overline{K}\bar{x} = \bar{f} \tag{26}$$

where the reduced coordinates $\bar{x}$ are related to the original coordinates x by Eq. (8) with

$$\bar{x}_b \equiv x_b \tag{27}$$

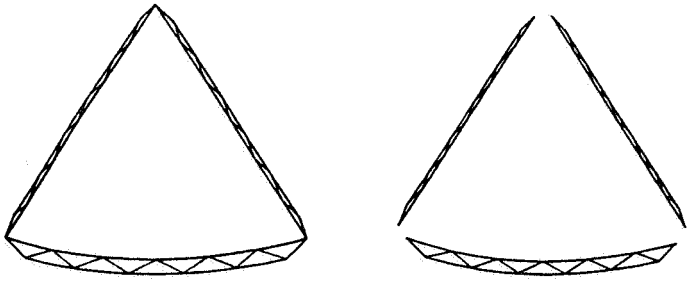
and

$$L = span\ [Q_1\ Q_2\ \cdots] \tag{28}$$

Because of the boundary coordinate identity of Eq. (27) and the form of the Q's generated by Eqs. (24) and (25), the transformed component stiffness matrix has the form

$$\overline{K} = \begin{bmatrix} \overline{K}_{11} & 0 & 0 & \cdots & 0 \\ 0 & \overline{K}_{22} & \overline{K}_{23} & & \\ 0 & \overline{K}_{32} & \overline{K}_{33} & & \\ \vdots & & & \ddots & \\ 0 & & & & \ddots \end{bmatrix} \tag{29}$$

The uncoupling of the boundary stiffness terms from the remaining partitions of the stiffness matrix is a result of the K-orthogonality between the initial constraint modes, Q_1, and all of the fixed-interface Krylov modes, Q_j, that follow.



a. Coupled Truss Structure. b. Components.

Figure 8. A Truss Used to Evaluate Block-Krylov Component Synthesis.

Table 1, from Ref. [14] compares a fixed-interface block-Krylov component-mode solution and a Craig-Bampton component-mode solution for natural frequencies of the 72 DOF plane truss shown in Fig. 8. The Craig-Bampton method produces a slightly more accurate reduced model, but at the expense of the additional computation required to produce the fixed-interface normal modes required by the Craig-Bampton method.

B-K (18 DOF)	C-B (18 DOF)	FEM (72 DOF)
0.000000E+0	0.000000E+0	0.000000E+0
0.000000E+0	0.000000E+0	0.000000E+0
0.000000E+0	0.000000E+0	0.000000E+0
1.953625E−2	1.953688E−2	1.953624E−2
2.205065E−2	2.205225E−2	2.205063E−2
4.923845E−2	4.926930E−2	4.923537E−2
7.273779E−2	7.231061E−2	7.229319E−2
7.626492E−2	7.570881E−2	7.567759E−2
1.550032E−1	1.528360E−1	1.528184E−1
1.617472E−1	1.586785E−1	1.586193E−1

Table 1. A Comparison of Block-Krylov and Craig-Bampton Component Synthesis – Natural Frequencies.

5. Unsymmetric Lanczos Algorithm for Damped Structural Dynamics Systems (Refs. [12,20,21])

Although most passive damping mechanisms yield a symmetric damping matrix, there are cases when the damping matrix is unsymmetric. For structures, unsymmetric damping may arise from active feedback control or from Coriolis forces. To deal with general unsymmetric damping, the usual approach is to write the system's dynamic equation in first-order state-space form. Then, an unsymmetric Lanczos algorithm is used to create a basis for model reduction of the first-order differential equations. References [12,20] describe a two-sided unsymmetric block Lanczos algorithm that generates a set of *left Lanczos vectors* and a set of *right Lanczos vectors* (analogous to sets of left eigenvectors and right eigenvectors). These two sets of Lanczos vectors form a basis that transforms the system equation to an unsymmetric block-tridiagonal form. The major disadvantage of a two-sided Lanczos algorithm is that the reduced-order model that is obtained may exhibit some high-frequency spurious modes or even unstable modes, although the full-order system is stable. However, the computational enhancements described in Ref. [12] produce a very robust two-sided algorithm.

A one-sided Lanczos algorithm for structures with unsymmetric damping matrix and/or stiffness matrix was recently described in Ref. [21]. Consider a linear, time-invariant system described by Eq. (10). Assume that the system is stable and completely controllable. Then, the following Lyapunov equation has a unique positive-definite solution.

$$AW_c + W_cA^T + BB^T = 0 \tag{30}$$

W_c is called the *controllability grammian* of the system. If the system's state vector is

transformed to another set of coordinates through a nonsingular projection matrix L

$$z = L\bar{z} \tag{31}$$

then the system equation becomes

$$\begin{aligned} \dot{\bar{z}} &= \bar{A}\bar{z} + \bar{B}u \\ y &= \bar{C}\bar{z} \end{aligned} \tag{32}$$

where the system matrices in the new coordinates are

$$\bar{A} = L^{-1}AL \quad , \quad \bar{B} = L^{-1}B \quad , \quad \bar{C} = CL \tag{33}$$

In Ref. [21] the transformation matrix

$$L \equiv [\ Q_1 \quad Q_2 \quad \cdots \quad Q_k\ ] \tag{34}$$

is formed by a three-term Lanczos iteration formula

$$AQ_i = Q_{i-1}\mathcal{G}_{i-1} + Q_i\mathcal{F}_i - Q_{i+1}\mathcal{G}_i^T \tag{35}$$

where

$$\begin{aligned} \mathcal{G}_{i-1} &= Q_{i-1}^T W_c^{-1} A Q_i \\ \mathcal{F}_i &= Q_i^T W_c^{-1} A Q_i \end{aligned} \tag{36}$$

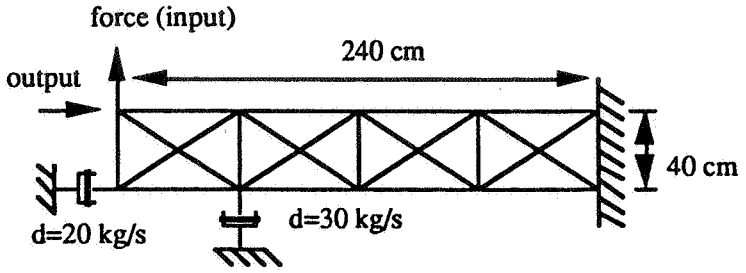
The Q_i's are orthonormalized with respect to the inverse of the controllability grammian of the system. That is

$$Q_i^T W_c^{-1} Q_j = \begin{cases} I & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{37}$$

Then, $\bar{A}$ has the almost-skew-symmetric, block-tridiagonal form

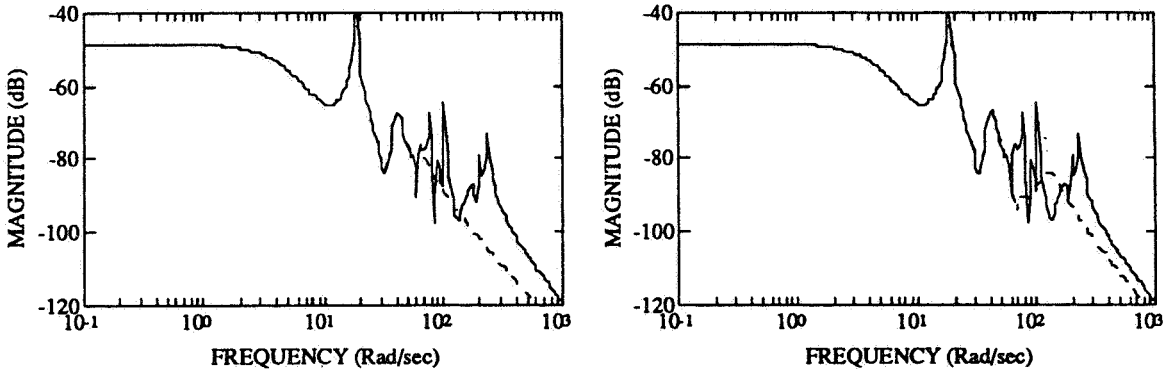
$$\bar{A} = \begin{bmatrix} \mathcal{F}_1 & \mathcal{G}_1 & & & \\ -\mathcal{G}_1^T & \mathcal{F}_2 & \mathcal{G}_2 & & \\ & -\mathcal{G}_2^T & \mathcal{F}_3 & \cdot & \\ & & \cdot & \cdot\ \cdot & \\ & & & \cdot\ \cdot\ \cdot & \\ & & & \cdot\ \cdot & \end{bmatrix} \tag{38}$$

Reference [21] lists a complete one-sided, unsymmetric block-Lanczos algorithm and also discusses special modifications to handle model reduction for unstable systems, to optimize the choice of starting vectors, to use A^{-1} as the iteration matrix, and to use the observability grammian instead of the controllability grammian. An



EA=2E+2 N ρA=2E-2 kg/cm

Figure 9. A Plane Truss Structure.



a. Full-order Model vs Twelve-state Eigenvector-reduced Model.

b. Full-order Model vs Twelve-state Lanczos-reduced Model with Starting Vector B.

Figure 10. Comparison of Frequency Response Functions.

example is provided that utilizes both controllability- and observability-grammian-based reductions. Figure 9 shows a plane truss structure (16 DOF's; 32 states), and Figs. 10a,b show frequency response plots of 12-state models based, respectively, on (complex) eigenvectors and based on (real) Lanczos vectors.

The advantages of the above-described one-sided method over the other existing unsymmetric Lanczos algorithms are: (1) the numerical breakdown problem that usually occurs in applying the two-sided unsymmetric Lanczos method is not present, (2) the Lanczos vectors that are produced lie in the controllable and observable subspace, (3) the reduced-order model is guaranteed to be stable, (4) a shifting scheme can be used for unstable systems, (5) the flexibility of the choice of starting vector can yield more accurate reduced-order models, and (6) the method is derived for general multi-input/multi-output systems.

6. Krylov/Lanczos Methods for Control of Flexible Structures

Space limitations prevent further discussion in this paper of the application of Krylov/Lanczos vectors to the control of flexible structures. Several such applications may be found in Refs. [6,13,22–24].

7. Acknowledgments

This work was supported by NASA Grant NAG9-357 with the NASA Lyndon B. Johnson Space Center. The authors wish to thank Dr. John Sunkel for his interest in this work.

References

[1] Parlett, B. N., *The Symmetric Eigenvalue Problem,* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1980.

[2] "Lanczos Method," Section 4.4.3 in *MSC NASTRAN Theoretical Manual (Preliminary),* Oct. 1985.

[3] Nour-Omid, B. and Clough, R. W., "Dynamic Analysis of Structures Using Lanczos Coordinates," *Earthquake Eng. & Struc. Dyn.,* Vol. 12, No. 4, 1984, pp. 565–577.

[4] Frisch, H. P., *IAC Program FEMDA - Theory and User's Guide,* NASA Tech Brief (Draft Copy), June 1989.

[5] Bathé, K. J. and Wilson, E. L., *Numerical Methods in Finite Element Analysis,* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1976.

[6] Su, T. J., "A Decentralized Linear Quadratic Control Design Method for Flexible Structures," Ph.D. Dissertation, The University of Texas at Austin, Austin, TX, Aug. 1989.

[7] Su, T. J. and Craig, R. R. Jr., "Krylov Model Reduction Algorithm for Undamped Structural Dynamics Systems," *J. Guidance, Control, and Dynamics,* Vol. 14, No. 6, 1991, pp. 1311–1313.

[8] Hickin, J. and Sinha, N. K., "Model Reduction for Linear Multivariable Systems," *IEEE Trans. Automat. Control,* Vol. AC-25, No. 6, 1980, pp. 1121–1127.

[9] Villemagne, C. D. and Skelton, R. E., "Model Reduction Using a Projection Formulation," *Int. J. Control,* Vol. 46, No. 6, 1987, pp. 2141–2169.

[10] Paige, C. C., "Practical Use of the Symmetric Lanczos Process with Reorthogonalization," *BIT,* Vol. 10, 1970, pp. 183–195.

[11] Parlett, B. N. and Scott, D. S., "The Lanczos Algorithm with Selective Orthogonalization," *Mathematics of Computation,* Vol. 33, No. 145, 1979, pp. 217–238.

[12] Kim, H. M. and Craig, R. R. Jr., "Computational Enhancement of an Unsymmetric Block Lanczos Algorithm," *Int. Numer. Methods in Eng.*, Vol. 30, No. 5, 1990, pp. 1083–1089.

[13] Su, T. J. and Craig, R. R. Jr., "Model Reduction and Control of Flexible Structures Using Krylov Vectors ," *J. Guidance, Control, and Dynamics,* Vol. 14, No. 2, 1991, pp. 260–267.

[14] Craig, R. R. Jr. and Hale, A. L., "The Block-Krylov Component Synthesis Method for Structural Model Reduction," *J. Guidance, Control, and Dynamics,* Vol. 11, No. 6, 1988, pp. 562–570.

[15] Craig, R. R. Jr. and Bampton, M. C. C., "Coupling of Substructures for Dynamic Analysis," *AIAA Journal,* Vol. 7, July 1968, pp. 1313–1319.

[16] MacNeal, R. H., "A Hybrid Method of Component Mode Synthesis," *Computers and Structures,* Vol. 1, 1971, pp. 581–601.

[17] Rubin, S., "Improved Component-Mode Representation for Structural Dynamic Analysis," *AIAA Journal,* Vol. 13, Aug. 1975, pp. 995–1006.

[18] Craig, R. R. Jr. and Chang, C. J., "On the Use of Attachment Modes in Substructure Coupling for Dynamic Analysis," *Proceedings of the AIAA 19th Structures, Structural Dynamics, and Materials Conference,* San Diego, CA, 1977, pp. 89–99.

[19] Hurty, W. C., "Dynamic Analysis of Structural Systems Using Component Modes," *AIAA Journal,* Vol. 3, April 1965, pp. 678–685.

[20] Kim, H. M. and Craig, R. R. Jr., "Structural Dynamics Analysis Using An Unsymmetric Block Lanczos Algorithm," *Int. J. Numer. Methods in Eng.,* Vol. 26, 1988, pp. 2305–2318.

[21] Su, T. J. and Craig, R. R. Jr., "An Unsymmetric Lanczos Algorithm for Damped Structural Dynamics Systems," Paper AIAA-92-2510, *Proc. AIAA/ASME/ ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference,* Dallas, TX, April 1992.

[22] Turner, R. M. and Craig, R. R. Jr., "Use of Lanczos Vectors in Dynamic Simulation," Report No. CAR86-3, Center for Aeronautical Research, Bureau of Engineering Research, The University of Texas at Austin, Austin, TX, May 1986.

[23] Su, T. J. and Craig, R. R. Jr., "Controller Reduction by Preserving Impulse Response Energy," AIAA Paper 89-3432, Aug. 1989.

[24] Su, T. J. and Craig, R. R. Jr., "Krylov Vector Methods for Model Reduction and Control of Flexible Structures," to appear in *Advances in Control and Dynamic Systems.*

A Component Modes Projection and Assembly Model Reduction Methodology for Articulated, Multi-Flexible Body Structures

Allan Y. Lee

Walter S. Tsuha

Jet Propulsion Laboratory

California Institute of Technology

Abstract

A two-stage model reduction methodology, combining the classical Component Mode Synthesis (CMS) method and the newly developed Enhanced Projection and Assembly (EP&A) method, is proposed in this research. The first stage of this methodology, called the COmponent Modes Projection and Assembly model REduction (COMPARE) method, involves the generation of CMS mode sets, such as the MacNeal-Rubin mode sets. These mode sets are then used to reduce the order of each component model in the Rayleigh-Ritz sense. The resultant component models are then combined to generate reduced-order system models at various system configurations. A composite mode set which retains important system modes at all system configurations is then selected from these reduced-order system models. In the second stage, the EP&A model reduction method is employed to reduce further the order of the system model generated in the first stage. The effectiveness of the COMPARE methodology has been successfully demonstrated on a high-order, finite-element model of the cruise-configured Galileo spacecraft.

1. Background and Motivation

Multibody dynamics simulation packages are gaining in popularity among dynamists for the simulation and analysis of systems of interconnected bodies (some or all of which are flexible). One such program, DISCOS,[1] was used in the development of control systems on board the Galileo spacecraft. The dual-spin Galileo spacecraft was modeled as a three-body

system, consisting of a flexible spinning rotor, a flexible stator, and a rigid scan platform.

For complex systems such as the Galileo spacecraft, practical considerations (e.g., simulation time) impose limits on the number of modes that each flexible body can retain in a given simulation. Modal truncation procedures must be used to select and retain a limited number of "important" modes which capture the salient features of the component dynamics. The Enhanced Projection and Assembly (EP&A)[6,7] technique is one way of performing this task.

The EP&A method,[6,7] is a model reduction methodology for articulated, multi-flexible body systems. In this method, a composite mode set, consisting of "important" system modes from all system configurations of interest, and not just from one particular system configuration, is first selected. It is then augmented with static correction modes before being "projected" onto the component models to generate reduced-order component models. To generate the composite mode set, eigenvalue problems concerning the full-order system models, at all configurations of interest, must be solved repetitively. This is a drawback of the EP&A method because solving large eigenvalue problems can be costly. To overcome this difficulty, a two-stage model reduction methodology, combining the classical Component Mode Synthesis (CMS) method and the Enhanced Projection and Assembly method (EP&A), is proposed in this research.

The stages involved in the proposed technique, to be called the COmponent Modes Projection and Assembly model REduction (COMPARE) method, are illustrated in Fig. 1. First, CMS mode sets, such as the MacNeal-Rubin mode sets, are generated and used to reduce the order of each component model in the Rayleigh-Ritz sense. These component mode sets are then assembled using the interface compatibility conditions to generate reduced-order system models at various system configurations. The order of these reduced-order system models is typically smaller than that of the full-order system model.

In the second stage, the newly developed EP&A model reduction method is employed to reduce further the order of the system model generated in the first stage. As described above, the composite mode set is augmented with static correction modes before being projected on the CMS-generated component models to generate the final reduced-order system model. In this way, COMPARE retains the merits of both the CMS and EP&A methods, without their demerits. The effectiveness of COMPARE will be verified using a cruise-configured Galileo spacecraft model.

2. Component Mode Synthesis Method Revisited[2,8]

The Component Mode Synthesis (CMS) method is a Rayleigh-Ritz based approximation method that is commonly used to analyze linear, high-order structural dynamics problems. To use this method, the structure is first subdivided into a number of components (or substructures), and a Ritz transformation is employed to reduce the model orders of these substructures. Many component mode sets may be used to perform this reduction but the MacNeal-Rubin (M-R) and Craig-Bampton (C-B) mode sets were shown to have good convergence properties in the sense of CMS. Once reduced, the reduced-order component models are then coupled using the interface compatibility conditions to form the reduced-order system model.

Since the Craig-Bampton and MacNeal-Rubin mode sets will be used in the present research to reduce the orders of the component models, their constructions are first briefly reviewed. To this end, consider a multi-flexible body structure as depicted in Fig. 2. The undamped motion of each component of the structure can be described by a matrix differential equation

$$M_{nn}\ddot{x}_n + K_{nn}x_n = F_n, \tag{1}$$

where x_n is an $n \times 1$ displacement vector, and M_{nn} and K_{nn} are $n \times n$ mass and stiffness matrices of the component, respectively. Note that the matrix dimensions are indicated by the matrix subscripts. The $n \times 1$ force vector acting on the component is denoted by F_n. A similar equation can also be written for component B.

To generate the C-B or M-R mode set, the last equation is partitioned as follows :

$$\begin{bmatrix} M_{ii} & M_{ij} \\ M_{ji} & M_{jj} \end{bmatrix} \begin{bmatrix} \ddot{x}_i \\ \ddot{x}_j \end{bmatrix} + \begin{bmatrix} K_{ii} & K_{ij} \\ K_{ji} & K_{jj} \end{bmatrix} \begin{bmatrix} x_i \\ x_j \end{bmatrix} = \begin{bmatrix} F_i \\ 0 \end{bmatrix}, \tag{2}$$

where x_i and x_j represent the interface and interior coordinates, respectively.

2.1 Craig-Bampton Mode Set

The Craig-Bampton mode set is generated by augmenting a low-frequency subset of the fixed interface (I/F) normal modes with a set of static-shape functions termed constraint modes. The first k fixed I/F normal modes Φ_{jk}, and the ordered eigenvalue matrix Λ_{kk} are related by the following relation :

$$-M_{jj}\Phi_{jk}\Lambda_{kk} + K_{jj}\Phi_{jk} = 0\,. \tag{3}$$

There are ways to decide on the number of modes to be kept in Φ_{jk}. One way is to keep all modes whose frequencies are less than twice a characteristic frequency of the system (e.g., control bandwidth).[2] The above determined normal modes are then augmented with i constraint modes, where constraint modes are static-shape functions that result by imposing unit displacement on one coordinate of the i-set while holding the remaining coordinate in that set fixed. It can be shown that the interior displacement for these constraint modes is given by[8]

$$\Psi_{ji} = -K_{jj}^{-1} K_{ji} \,. \tag{4}$$

The C-B mode set is

$$\begin{bmatrix} O_{ik} & I_{ii} \\ \Phi_{jk} & \Psi_{ji} \end{bmatrix}$$

which is then used to reduce the full-order component model.

2.2 MacNeal-Rubin Mode Set

In a manner similar to generating the Craig-Bampton mode set, the MacNeal-Rubin mode set is generated by augmenting a low-frequency subset of the free I/F normal modes with a set of static force response functions termed residual modes. The first k free I/F normal modes Φ_{nk}, and the ordered eigenvalue matrix Λ_{kk}, are related by the following relation :

$$-M_{nn}\Phi_{nk}\Lambda_{kk} + K_{nn}\Phi_{nk} = 0 \,. \tag{5}$$

The kept eigenvector matrix Φ_{nk}, which has been normalized with respect to the mass matrix, may be partitioned into its rigid-body and flexible parts: $[\Phi_{nr}\ \Phi_{nf}]$. Let Λ_{ff} be the eigenvalue matrix associated with the kept flexible modes. Then, the residual modes may be determined by

$$\Psi_{na} = (P_{nn}^T\, S_{nn}\, P_{nn} - \Phi_{nf}\, \Lambda_{ff}^{-1}\, \Phi_{nf}^T)\, F_{na} \,, \tag{6}$$

where $P_{nn} = I_{nn} - M_{nn}\, \Phi_{nr}\, \Phi_{nr}^T$, S_{nn} is the "pseudo" flexibility matrix of the component,[8] defined as follows. Let the set of all physical coordinates be divided into three subsets : r, a, and w. The r set may be any statically determinate constraint set which provides restraint against rigid-body motion. The a set consists of coordinates where unit forces are to be applied to define attachment modes (i.e., at the interface coordinates, and at coordinates where external forces are applied). Finally, the w set consists of the remaining coordinates in z. Using these definitions, the component stiffness matrix K_{nn} is partitioned as follows :

$$\begin{bmatrix} k_{rr} & k_{ra} & k_{rw} \\ k_{ar} & k_{aa} & k_{aw} \\ k_{wr} & k_{wa} & k_{ww} \end{bmatrix} .$$

The pseudo-flexibility matrix S_{nn} is now given by a matrix of the form[8]

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & \begin{bmatrix} k_{aa} & k_{aw} \\ k_{wa} & k_{ww} \end{bmatrix}^{-1} \\ 0 & \end{bmatrix} .$$

Finally, the matrix F_{na} is given by $[0_{ar},\ I_{aa},\ 0_{aw}]^T$, where the identity matrix I_{aa} is associated with the a interface coordinates. The M-R mode set is $[\Phi_{nk}\ \Psi_{na}]$, which is then used to generate a reduced-order model for the component.

3. A Component Modes Projection and Assembly Model Reduction (COMPARE) Methodology

Once CMS-based reduced-order component models are generated, they are assembled using the interface compatibility conditions to produce reduced-order system models at various system configurations of interest. Since the orders of these reduced-order system models are typically smaller than those of the full-order system models, we have accomplished the first of the two model reduction steps of the COMPARE methodology. However, note that these CMS-generated reduced-order system models were obtained without using knowledge of any system-level input-output information. This drawback is remedied in the second stage, in which the EP&A methodology[6,7] is used to further reduce the order of the models generated in the first stage. Since the EP&A methodology has been described elsewhere,[6,7] it will only be briefly reviewed here.

Consider a system with two flexible components. The undamped motion of component A, as described by either its M-R or C-B mode set, is given by

$$\begin{aligned} I_{pp}\ddot{\eta}_p^A + \Lambda_{pp}^A \eta_p^A &= G_{pa}^A u_a^A , \\ y_b^A &= H_{bp}^A \eta_p^A . \end{aligned} \tag{7}$$

Here, η_p^A and Λ_{pp}^A are the generalized coordinates and the diagonal stiffness matrix of component A, respectively. The dimension of η_p^A is p. The matrix G_{pa}^A is an control distribution matrix, and u_a^A is an $a \times 1$ control vector. Similarly, the matrix H_{bp}^A is an output distribution matrix, and y_b^A is an $b \times 1$ output vector. Similar equations can also be written for component B.

The system equations of motion at a particular articulation angle α may be constructed using these component equations, and enforcing displacement compatibility conditions at the component interface. To this end, let $P(\alpha) = [P_{pe}^{A^T}(\alpha), P_{qe}^{B^T}(\alpha)]^T$ be any full-rank matrix mapping a minimal system state η_e into

$$\begin{bmatrix} \eta_p^A \\ \eta_q^B \end{bmatrix} = \begin{bmatrix} P_{pe}^A(\alpha) \\ P_{qe}^B(\alpha) \end{bmatrix} [\eta_e], \tag{8}$$

where η_e is an $e \times 1$ reduced-order system coordinate, and $e = p + q - i$ (i is the number of I/F constraint relations). For ease of notation, the dependencies of the matrices P_{pe}^A, P_{qe}^B, η_p^A, η_q^B, and η_e on α are dropped in the sequel. Substituting $\eta_p^A = P_{pe}^A \eta_e$ and $\eta_q^B = P_{qe}^B \eta_e$ into (7) and the corresponding equations for component B, pre-multiplying the resultant equations by $P_{pe}^{A^T}$ and $P_{qe}^{B^T}$, respectively, and summing the resultant equations gives

$$M_{ee}\ddot{\eta}_e + K_{ee}\eta_e = G_{ea}u_a\,, \tag{8a}$$

$$y_s = H_{se}\eta_e\,, \tag{8b}$$

where M_{ee}, K_{ee}, G_{ea}, and H_{se}, all functions of α, are given by

$$\begin{aligned} M_{ee} &= P_{pe}^{A^T} P_{pe}^A + P_{qe}^{B^T} P_{qe}^B\,, \\ K_{ee} &= P_{pe}^{A^T} \Lambda_{pp}^A P_{pe}^A + P_{qe}^{B^T} \Lambda_{qq}^B P_{qe}^B\,, \\ G_{ea} &= P_{pe}^{A^T} G_{pa}^A + P_{qe}^{B^T} G_{qa}^B\,, \\ H_{se} &= \begin{bmatrix} H_{bp}^A P_{pe}^A \\ H_{lq}^B P_{qe}^B \end{bmatrix}\,, \end{aligned}$$

where $y_s = [y_b^{A^T}\ y_l^{B^T}]^T$, and $s = b + l$. To arrive at the equation for G_{ea}, we have assumed that $u_a^A = u_a^B = u_a$. Otherwise, the term $G_{ea}\,u_a$ in (8a) should be replaced by $[P_{pe}^{A^T} G_{pa}^A,\ P_{qe}^{B^T} G_{qa}^B]\ [u_a^{A^T}\ ,\ u_a^{B^T}]^T$. Since M_{ee} is symmetric and positive-definite while K_{ee} is symmetric and positive semi-definite, a transformation Φ_{ee} that diagonalizes M_{ee} and K_{ee} simultaneously can always be found. Let ϕ_e be the corresponding generalized coordinate, i.e., $\eta_e = \Phi_{ee}\,\phi_e$. Substituting this relation into (8a), and pre-multiplying the resultant equation by Φ_{ee}^T gives

$$\begin{aligned} I_{ee}\ddot{\phi}_e + \Lambda_{ee}\phi_e &= \bar{G}_{ea}u_a\,, \\ y_s &= \bar{H}_{se}\phi_e\,, \end{aligned} \tag{9}$$

where $\Phi_{ee}^T M_{ee}\Phi_{ee} = I_{ee}$, $\bar{H}_{se} = H_{se}\Phi_{ee}$, $\bar{G}_{ea} = \Phi_{ee}^T G_{ea}$, and where $\Lambda_{ee} = \Phi_{ee}^T K_{ee}\Phi_{ee}$ contains the undamped, reduced-order system eigenvalues along its diagonal. Equation (9)

represents the reduced-order system model obtained from the first stage of the COMPARE methodology.

The EP&A method is used in the second stage of COMPARE. With the EP&A method, only k of the system's e modes are kept while the remaining t $(= e - k)$ modes are removed. The kept mode set is a composite mode set, consisting of "important" system modes from all system configurations of interest, and not just from one particular configuration. With this understanding, we have

$$\eta_e = \Phi_{ee}\phi_e = [\,\Phi_{ek} \quad \Phi_{et}\,] \begin{bmatrix} \phi_k \\ \phi_t \end{bmatrix} \doteq \Phi_{ek}\phi_k\,, \tag{10}$$

where ϕ_k and ϕ_t are generalized coordinates associated with the "kept" and "truncated" modes, respectively, and Φ_{ek} and Φ_{et} the corresponding eigenvector matrices.

The composite mode set Φ_{ek} may now be projected onto the CMS-generated component models: $\eta_p^A \doteq P_{pe}^A \Phi_{ek}\, \phi_k^A = \Psi_{pk}^A \phi_k^A$. Here, ϕ_k^A denotes reduced sets of generalized coordinates of component A. The substitution of the last relation into (7) produces the "constrained" equations of motion for component A:

$$\Psi_{pk}^{A^T} \Psi_{pk}^A\, \ddot{\phi}_k^A + \Psi_{pk}^{A^T} \Lambda_{pp}^A \Psi_{pk}^A\, \phi_k^A = \Psi_{pk}^{A^T} G_{pa}^A\, u_a\,. \tag{11}$$

A second reduced-order system model can now be constructed using (11), a similar equation for component B, and the displacement compatibility conditions at the component I/F. The order of this new reduced-order model is smaller than that obtained from the first step (cf. (9)) due to the truncation of "t" modes in (10). In addition, it has been proven that the modes retained in the composite mode set Φ_{ek} are captured exactly by the resultant reduced-order system model (with a number of extraneous modes [3,4,5,6,7]). However, the static gain of the resultant reduced-order system model is not the same as that given by (9).[7]

Two different approaches were introduced in Lee and Tsuha[7] to preserve the static gain of the system described by (9) in the second reduced-order system model. The first approach involves augmenting the k "kept" modes of the system with an additional a modes so as to create a statically complete mode set. The enlarged mode set is then "projected" onto the components, and the reduced components are assembled as usual.[7] In the second approach, the k "kept" modes of the system are first projected onto the components. After the projections, the reduced-component mode sets are each augmented with static correction

modes. The augmented mode sets are then used to reduce the components, and the resultant component models are assembled to generate the reduced-order system model. The details of these approaches are given below.

3.1 Component-level Augmentation Techniques[6,7]

In the component-level augmentation approach, the k "kept" modes are first projected onto the components. We first write

$$\begin{aligned} \eta_p^A &= \Psi_{pk}^A \, \Xi_{kk}^A \, \nu_k^A = \Upsilon_{pk}^A \, \nu_k^A = [\Upsilon_{pr}^A \; \Upsilon_{pf}^A] \, \nu_k^A \,, \\ \eta_q^B &= \Psi_{qk}^B \, \Xi_{kk}^B \, \nu_k^B = \Upsilon_{qk}^B \, \nu_k^B = [\Upsilon_{qr}^B \; \Upsilon_{qf}^B] \, \nu_k^B \,, \end{aligned} \tag{12}$$

where Ξ_{kk}^A is the eigenvector matrix associated with an eigenvalue problem of (11), and ν_k^A is the corresponding generalized coordinate. The transformation matrices Υ_{pk}^A and Υ_{qk}^B are defined in (12). Matrix partitions Υ_{pr}^A and Υ_{pf}^A denote eigenvectors associated with the rigid-body and flexible modes of the projected component A model. Similar partitions are made for the eigenvectors associated with the projected component B model.

Let Υ_{pr}^A, Υ_{pf}^A, etc. be normalized such that

$$\begin{aligned} \Upsilon_{pr}^{A^T} \, \Upsilon_{pr}^A &= I_{rr} \,, \qquad & \Upsilon_{pr}^{A^T} \, \Lambda_{pp}^A \, \Upsilon_{pr}^A &= 0_{rr} \,, \\ \Upsilon_{pf}^{A^T} \, \Upsilon_{pf}^A &= I_{ff} \,, \qquad & \Upsilon_{pf}^{A^T} \, \Lambda_{pp}^A \, \Upsilon_{pf}^A &= \bar{\Lambda}_{ff}^A \,, \end{aligned} \tag{13}$$

where $\bar{\Lambda}_{ff}^A$ is an eigenvalue matrix associated with the projected flexible modes of component A. Similar expressions can also be written for component B. Using the matrices defined in (12-13), residual modes,[8] described in Section 2.2, may once again be used to augment the projected mode sets of the components. From (6), the residual modes are given by

$$\Upsilon_{pa}^A = (\bar{P}_{pp}^{A^T} \, \bar{S}_{pp}^A \, \bar{P}_{pp}^A - \Upsilon_{pf}^A \, \bar{\Lambda}_{ff}^{A^{-1}} \, \Upsilon_{pf}^{A^T}) \, F_{pa}^A \,, \tag{14}$$

where

$$\bar{P}_{pp}^A = I_{pp}^A - \Upsilon_{pr}^A \, \Upsilon_{pr}^{A^T} \,, \tag{15}$$

and $\bar{S}_{pp}^A$ is the "pseudo" flexibility matrix of component A. If we assume that the diagonal stiffness matrix of component A (see Λ_{pp}^A in (7)) is ordered so that all the rigid-body modes (with zero frequency) are given first, then $\bar{S}_{pp}^A$ is given by

$$\begin{bmatrix} 0 & 0 \\ 0 & \Lambda_{\bar{p}\bar{p}}^{A^{-1}} \end{bmatrix} . \tag{16}$$

Here $\bar{p} = p$ - r is the total number of flexible modes in the CMS mode set of component A. In (14), the matrix F^A_{pa} is given by $[0_{ar},\ I_{aa},\ 0_{a\,p-r-a}]^T$. Similar expressions can also be written for component B.

The projected mode sets Υ^A_{pk} and Υ^B_{qk} can now be supplemented with the residual modes

$$\begin{aligned} \eta^A_p &= [\Upsilon^A_{pk}\ \Upsilon^A_{pa}] \begin{bmatrix} \nu^A_k \\ \nu^A_a \end{bmatrix} = \Upsilon^A_{pv}\,\nu^A_v\,, \\ \eta^B_q &= [\Upsilon^B_{qk}\ \Upsilon^B_{qa}] \begin{bmatrix} \nu^B_k \\ \nu^B_a \end{bmatrix} = \Upsilon^B_{qv}\,\nu^B_v\,, \end{aligned} \tag{17}$$

where ν^A_a and ν^B_a are generalized coordinates associated with the residual modes. Using the Ritz transformations suggested in (17), the new component-projected equations of motion are

$$\Upsilon^{A^T}_{pv}\Upsilon^A_{pv}\ \ddot{\nu}^A_v + \Upsilon^{A^T}_{pv}\Lambda^A_{pp}\Upsilon^A_{pv}\ \nu^A_v = \Upsilon^{A^T}_{pv}\,G^A_{pa}u_a\,, \tag{18}$$

$$\Upsilon^{B^T}_{qv}\Upsilon^B_{qv}\ \ddot{\nu}^B_v + \Upsilon^{B^T}_{qv}\Lambda^B_{qq}\Upsilon^B_{qv}\ \nu^B_v = \Upsilon^{B^T}_{qv}\,G^B_{qa}u_a\,. \tag{19}$$

Using (18-19), the reduced-order system equations of motion at a particular system configuration α can now be formed by enforcing displacement compatibility at the interface

$$C^A_{ip}(\alpha)\eta^A_p + C^B_{iq}(\alpha)\eta^B_q = 0\,, \tag{20}$$

where C^A_{ip} and C^B_{iq} are matrices that establish the constraint relations between the generalized coordinates of CMS generated component models, and i is the number of constraint relations. Using (17), (20) becomes

$$[C^A_{ip}\Upsilon^A_{pv}\ \ C^B_{iq}\Upsilon^B_{qv}] \begin{bmatrix} \nu^A_v \\ \nu^B_v \end{bmatrix} = [D_{ic}] \begin{bmatrix} \nu^A_v \\ \nu^B_v \end{bmatrix} \doteq 0\,, \tag{21}$$

where $[D_{ic}]$ is defined in (21), and $c = 2v$. To construct the reduced-order system model, we partition the "compatibility" matrix $[D_{ic}]$ using the Singular Value Decomposition (SVD) technique

$$[D_{ic}] = [C^A_{ip}\Upsilon^A_{pv}\,,\ C^B_{iq}\Upsilon^B_{qv}]\,, = [U_{ii}]\,[\Sigma_{ii}\,, O_{id}] \begin{bmatrix} V^T_{ci} \\ V^T_{cd} \end{bmatrix}\,, \tag{22}$$

where $d = c$ - i, and $[O_{id}]$ is an $i \times d$ null matrix. We can now write[3,6,7]

$$\begin{bmatrix} \nu^A_v \\ \nu^B_v \end{bmatrix} = [P_{cd}]\,\nu_d = \begin{bmatrix} P^A_{vd} \\ P^B_{vd} \end{bmatrix}\,\nu_d\,, \tag{23}$$

where $[P_{cd}] = [V_{cd}]$ (cf. (22)) is a full column rank mapping matrix, and ν_d denotes a minimum set of generalized coordinates of a statically complete reduced-order system. Substituting $\nu_v^A = [P_{vd}^A]\,\nu_d$ and $\nu_v^B = [P_{vd}^B]\,\nu_d$ into (18) and (19), pre-multiplying the resultant equations by $P_{vd}^{A^T}$ and $P_{vd}^{B^T}$, respectively, and summing the resultant equations give

$$M_{dd}\ddot{\nu}_d + K_{dd}\nu_d = G_{da}u_a\,, \tag{24}$$

$$y_s = H_{sd}\nu_d\,, \tag{25}$$

where M_{dd}, K_{dd}, G_{da}, and H_{sd}, all functions of α, are given by

$$M_{dd} = P_{vd}^{A^T}\Upsilon_{pv}^{A^T}\Upsilon_{pv}^{A}P_{vd}^{A} + P_{vd}^{B^T}\Upsilon_{qv}^{B^T}\Upsilon_{qv}^{B}P_{vd}^{B}\,, \tag{26}$$

$$K_{dd} = P_{vd}^{A^T}\Upsilon_{pv}^{A^T}\Lambda_{pp}^{A}\Upsilon_{pv}^{A}P_{vd}^{A} + P_{vd}^{B^T}\Upsilon_{qv}^{B^T}\Lambda_{qq}^{B}\Upsilon_{qv}^{B}P_{vd}^{B}\,, \tag{27}$$

$$G_{da} = P_{vd}^{A^T}\Upsilon_{pv}^{A^T}G_{pa}^{A} + P_{vd}^{B^T}\Upsilon_{qv}^{B^T}G_{qa}^{B}\,, \tag{28}$$

$$H_{sd} = \begin{bmatrix} H_{bp}^{A}\Upsilon_{pv}^{A}P_{vd}^{A} \\ H_{lq}^{B}\Upsilon_{qv}^{B}P_{vd}^{B} \end{bmatrix}. \tag{29}$$

Since M_{dd} is symmetric and positive-definite while K_{dd} is symmetric and positive semi-definite, a transformation Φ_{dd} that diagonalizes M_{dd} and K_{dd} simultaneously can always be found. Let Φ_{dd} be normalized with respect to the mass matrix, and let χ_d be the corresponding generalized coordinate, i.e.,

$$\nu_d = \Phi_{dd}\,\chi_d\,. \tag{30}$$

Substituting (30) into (24-25) and pre-multiplying the resultant equation by Φ_{dd}^T give

$$I_{dd}\ddot{\chi}_d + \Lambda_{dd}\chi_d = \bar{G}_{da}u_a\,, \tag{31}$$

$$y_s = \bar{H}_{sd}\eta_d\,, \tag{32}$$

where

$$\bar{H}_{sd} = H_{sd}\Phi_{dd}\,, \tag{33}$$

$$\bar{G}_{da} = \Phi_{dd}^T G_{da}\,, \tag{34}$$

$$\Lambda_{dd} = \Phi_{dd}^T K_{dd}\Phi_{dd}\,. \tag{35}$$

Here Λ_{dd} is a diagonal matrix with the undamped, reduced-order system squared frequencies along its diagonal. It was proven in Lee and Tsuha[7] that Φ_{ek} are captured exactly in Φ_{dd}

despite the augmentations of the projected component models with residual modes. The matrix Λ_{dd} also contains a number of extraneous modes.[6]

3.2 System Level Augmentation

The equation (9) may be decomposed into its kept and truncated parts:

$$\begin{aligned} I_{kk}\ddot{\phi}_k + \Lambda_{kk}\phi_k &= \Phi_{ek}^T G_{ea}\, u_a\,, \\ I_{tt}\ddot{\phi}_t + \Lambda_{tt}\phi_t &= \Phi_{et}^T G_{ea}\, u_a\,. \end{aligned} \tag{36}$$

The mode set Φ_{et}, which is truncated, will now be replaced by a smaller but statically equivalent mode set Φ_{ea}. To find Φ_{ea}, consider the following Ritz transformation[6]

$$\begin{aligned} \phi_t &= R_{ta}\phi_a\,, \\ \text{where } R_{ta} &= \Lambda_{tt}^{-1}\,\Phi_{et}^T G_{ea}\,. \end{aligned} \tag{37}$$

Here ϕ_a is the generalized coordinate associated with the augmented mode set Φ_{ea}. It can be shown that the static gain due to the mode set $[\Phi_{ek}\,\Phi_{ea}] = [\Phi_{ek}\;\Phi_{et}R_{ta}]$ is identical to that of the original system (cf. (9)).[6] Hence, $[\Phi_{ek}\;\Phi_{et}R_{ta}]$ is a statically complete mode set.

Several observations regarding the augmented mode set $[\Phi_{ek}\;\Phi_{et}\,R_{ta}]$ are in order. First, note that the augmented mode set contains two parts. The first part contains a selected number of eigenvectors which satisfy the eigenvalue problem defined by M_{ee} and K_{ee}. The second part is formed using the residual flexibility matrix and the distribution matrix of the external load. Hence, $\Phi_{et}\,R_{ta}$, unlike Φ_{ek}, is a function of the external load distribution.

Next, we observe that $\Phi_{et}\,R_{ta}$ is identical to the residual mode defined in Section 2.2. Also, note that this mode set is mass and stiffness orthogonal to the retained mode set Φ_{ek}. Hence, the two parts of the augmented mode set are linearly independent. This statically complete mode set is now ready for projections onto the various flexible component models to generate reduced-order component models. The reduced-order component models may then be assembled to generate the reduced-order system model using the procedures outlined in Section 3.1.

4. Applying COMPARE on A High-order Finite-element Galileo Model

The effectiveness of the proposed COMPARE methodology will now be demonstrated using a high-order finite-element model of the cruise-configured Galileo spacecraft. The

three-body topology of the dual-spin Galileo spacecraft is illustrated in Fig. 3.[7] The rotor is the largest and most flexible component represented, with 243 dof. The smaller and more rigid stator is represented with 57 dof. Lastly, the scan platform is the smallest body idealized as rigid, with 6 dof.

For the purpose of controller design, a low-order system model, accurate at all configurations of interest and over a frequency range of interest (0-10 Hz) is needed. To this end, we apply the MacNeal-Rubin version of the COMPARE methodology on the Galileo model. The first stage of COMPARE requires the generation of M-R mode sets for all the flexible components. Following standard procedures, free interface normal modes of both the rotor and stator are first determined, and then truncated at twice the frequency of interest (20 Hz). Next, these truncated normal mode sets are each augmented with residual modes to generate the needed M-R mode sets for both the stator and rotor.

Next, the MacNeal-Rubin mode sets and the interface compatibility conditions are used to construct system models at all system configurations of interest, and determine from them important system-level modes at all clock angles of interest. The selected composite mode set has 8 rigid-body and 21 flexible modes.

The next step is to augment the composite mode set with one or more static-correction modes. For the Galileo example, we augment the composite mode set with two residual modes, one for an input torque about the Z-axis on the rotor side of the rotor/stator interface, and a second equal and opposite torque on the stator side of the interface. The enlarged mode set is then projected onto the flexible components. The resultant reduced-order models of the rotor and stator have 29 (with 6 rigid-body) and 21 (with 8 rigid-body) modes, respectively. The assembled reduced-order model has 44 (with 8 rigid-body) modes. The natural frequencies of these reduced-order component models and of the system model at a clock angle of 300 degrees are tabulated in Table 1. All system flexible modes retained in the composite mode set have been captured exactly in the reduced-order system model.

Comparisons of Bode plots of full-order and reduced order models, at clock angles of 60, 180, and 300 degrees, are given in Figs. 4, 5, and 6 respectively. Actuation was done at the Spin Bearing Assembly (SBA) located at the rotor-stator interface (along the Z-axis, cf. Fig. 3), and sensing was done by a gyroscope located on the scan platform. From these comparisons, we observe that the frequency responses of the full-order models, at all

configurations of interest, have been closely captured by their reduced-order counterparts, over the frequency range of interest (0-10 Hz). Results obtained at other clock angles are similar to those depicted in Figs. 4-6.

5. Concluding Remarks

A two-stage model reduction methodology, called COMPARE, is proposed in this research. The first stage of this methodology involves the generation of CMS mode sets for the flexible components. The resultant component models are then combined to generate reduced-order system models at various system configurations. A composite mode set which retains important system modes at all system configurations is then determined from these reduced order system models. In the second stage, the EP&A model reduction method is employed to reduce further the order of the system model generated in the first stage.

The merit of the COMPARE methodology is that system models (at various system configurations) assembled using CMS-generated component models are smaller in size than the full-order system models. Hence, COMPARE alleviates the need to solve large-order eigenvalue problems repetitively. The need to generate the components' M-R or C-B mode sets is not a disadvantage because efficient software exists for their construction (see, e.g., Tsuha[9]). The effectiveness of COMPARE, using the M-R version of COMPARE, has been successfully demonstrated on a high-order, finite-element model of the cruise-configured Galileo spacecraft.

6. References

1. Bodley, C.S., Devers, A.D., Park, A.C., and Frisch, H.P., "A Digital Computer Program for the Dynamic Interaction Simulation of Controls and Structure (DISCOS)," NASA Technical Paper 1219, Vols. I and II, NASA Center for AeroSpace Information, Baltimore, Maryland, May, 1978.
2. Spanos, J.T. and Tsuha, W., "Selection of Component Modes for the Simulation of Flexible Multibody Spacecraft," Journal of Guidance, Navigation, and Control, Vol. 14, No. 2, pp. 278-286, March/April, 1991.
3. Bernard, D., "Projection and Assembly Method for Multibody Component Model Reduction," Journal of Guidance, Control, and Dynamics, Vol. 13, No. 5, September/October, 1990.

4. Eke, F.O. and Man, G.K., "Model Reduction in the Simulation of Interconnected Flexible Bodies," paper AAS 87-455, AAS/AIAA Astrodynamics Specialist Conference, Kalispell, Montana, August, 1987.
5. Tsuha, W. and Spanos, J.T., "Reduced Order Component Modes for Flexible Multibody Dynamics Simulations," paper presented at the AIAA Aerospace Sciences Meeting, Reno, Nevada, January, 1990.
6. Lee, A.Y. and Tsuha, W.S., "Applying the Enhanced Projection and Assembly Model Reduction Methodology on Articulated, Multi-flexible Body Structures," Proceedings of the IEEE Singapore International Conference on Intelligent Control and Instrumentation, February 17-21, 1992.
7. Lee, A.Y. and Tsuha, W.S., "An Enhanced Projection and Assembly Model Reduction Methodology," Proceedings of the AIAA Guidance, Navigation, and Control Conference, New Orleans, Louisiana, August 12-14, 1991.
8. Craig, R.R., Jr., Structural Dynamics: An Introduction to Computer Methods, John Wiley and Sons, Inc., New York, 1981.
9. Tsuha, W.S., The Benfield-Hruda, MacNeal-Rubin, and Craig-Bampton Mode Set Generation Procedures, JPL Publication D-7213 (internal document), Jet Propulsion Laboratory, California Institute of Technology, February, 1990.

7. Acknowledgments

The research described in this paper was conducted at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. The authors wish to thank Dr. D. Eldred, Dr. J. Spanos, and Dr. M. Wette for many helpful discussions and valuable suggestions. We also wish to thank Dr. M. Lou and Dr. G. Man for their interest and encouragement.

Table 1 Frequencies of Reduced-order Stator, Rotor,
and System (at a Clock Angle of 300 degrees) Flexible Modes

Flexible Mode	ω_{stator} (Hz)	ω_{rotor} (Hz)	ω_{system} (Hz)
1	7.105	0.143	0.127
2	9.129	0.866	0.864†
3	10.561	1.237	1.236†
4	14.560	1.483	1.479†
5	43.172	1.728	1.707†
6	50.070	2.286	1.734†
7	63.530	2.809	2.072†
8	80.867	3.647	2.351†
9	86.989	3.996	2.815†
10	96.605	5.207	3.707†
11	166.34	5.337	4.167
12	240.18	5.994	5.231†
13	254.52	6.410	5.433
14		9.503	5.467†
15		10.291	6.150
16		10.553	6.436
17		13.536	7.056†
18		15.613	8.153†
19		29.188	9.606†
20		41.181	9.613
21		58.524	10.294
22		69.003	10.420†
23		77.722	10.555†
24			13.534†
25			13.997
26			15.860
27			16.800†
28			20.591
29			21.280†
30			28.462
31			34.316
32			41.214
33			48.011
34			58.318
35			71.770†
36			82.129†

† Exactly captured system flexible modes.

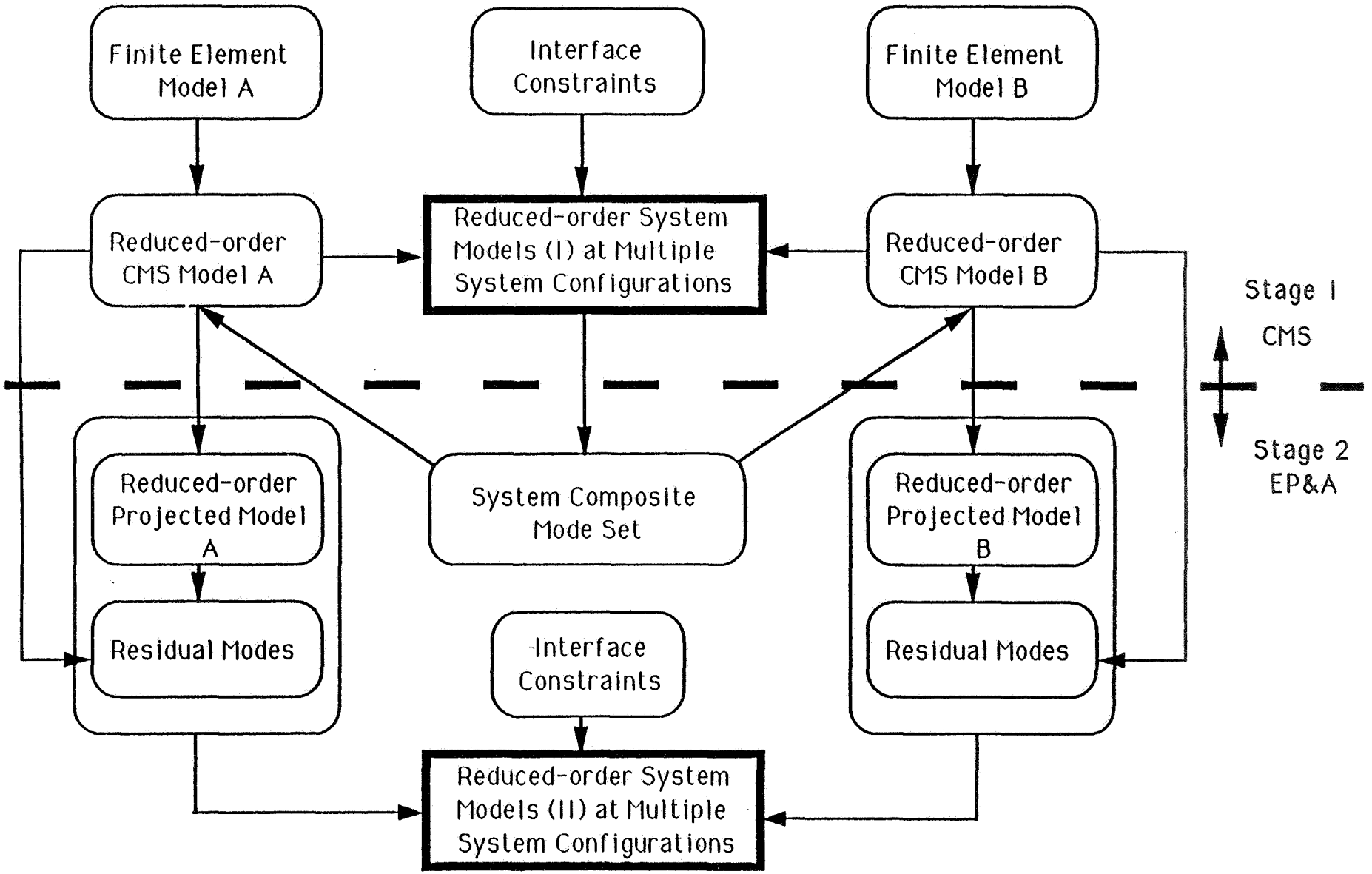


Fig. 1 A 2-stage Component Modes Projection and Assembly Model Reduction Methodology (COMPARE)

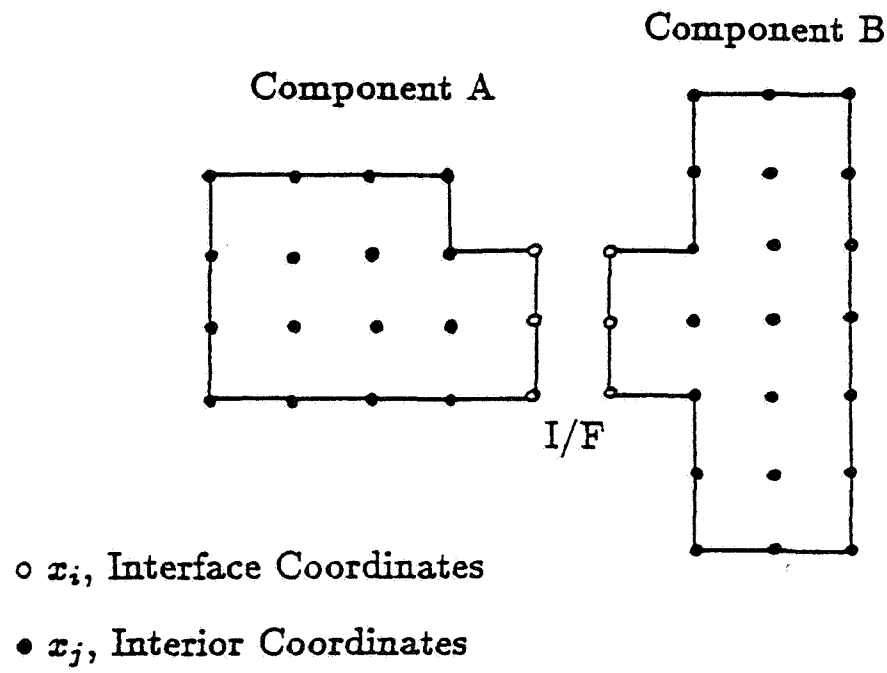


Fig. 2 Coordinate Definitions of A 2-Component System

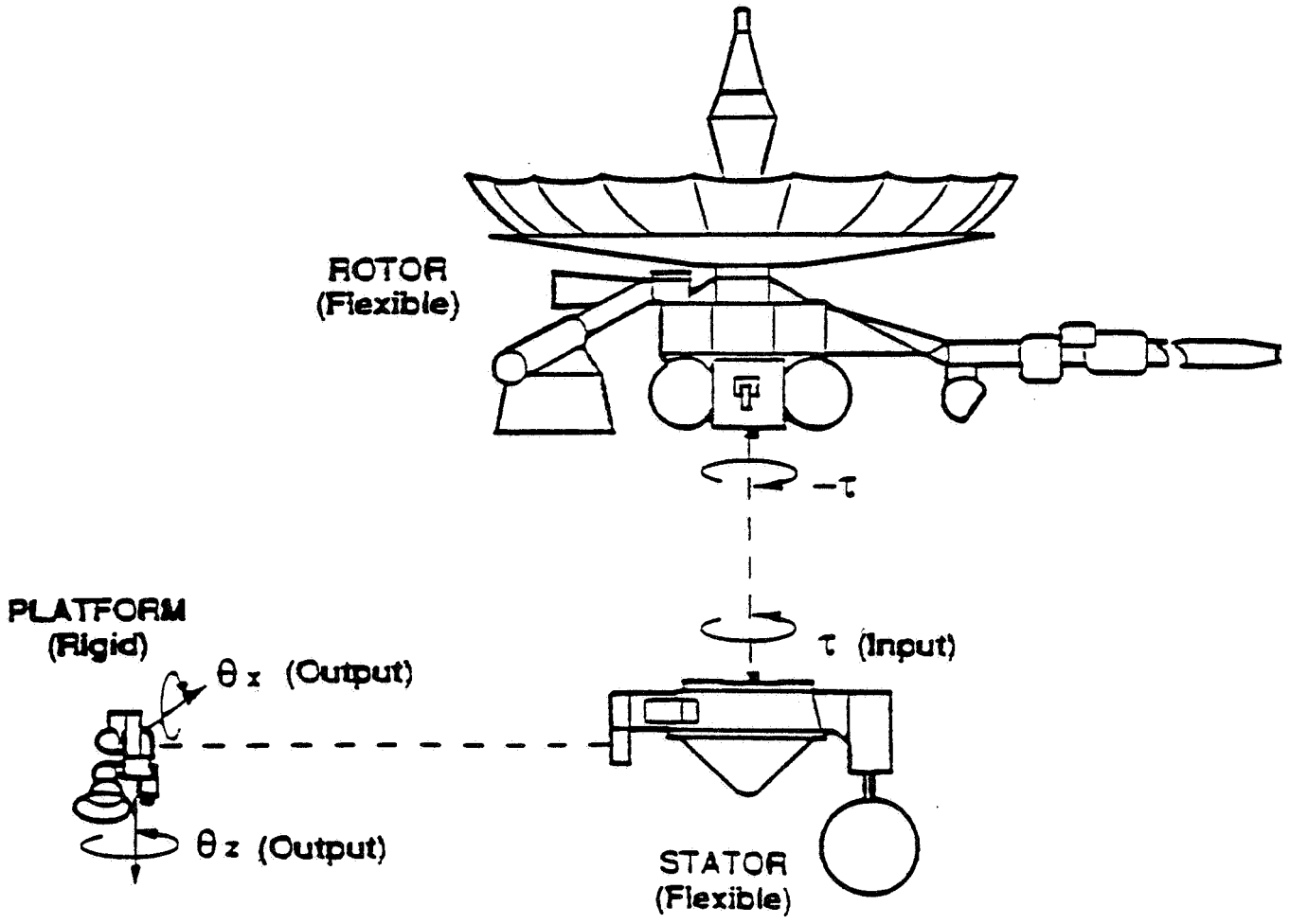


Fig. 3 Galileo Spacecraft Cruise Model[2,6,7]

Fig. 4 Bode Plot Comparison of Full- and Reduced-order Models

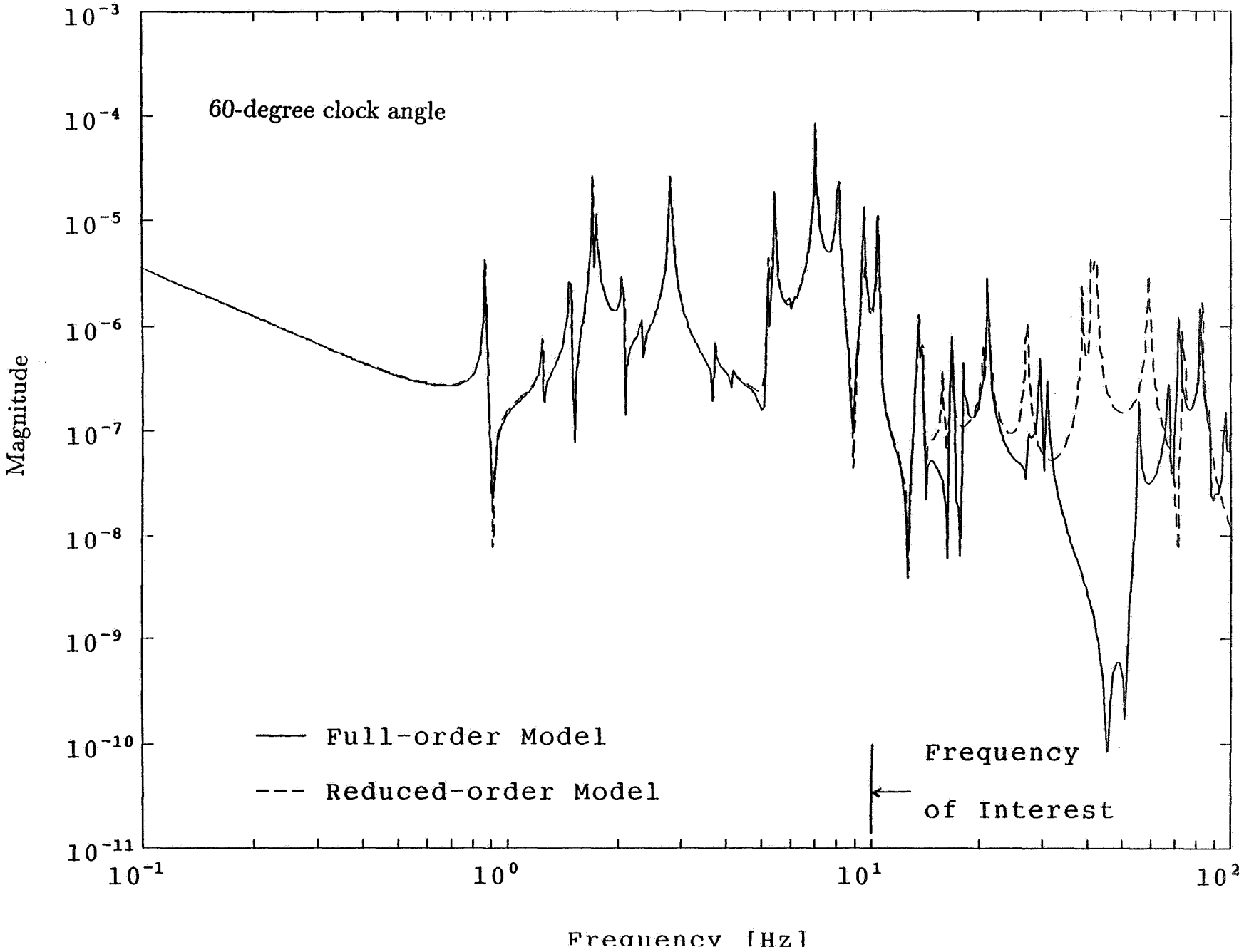


Fig. 5 Bode Plot Comparison of Full- and Reduced-order Models

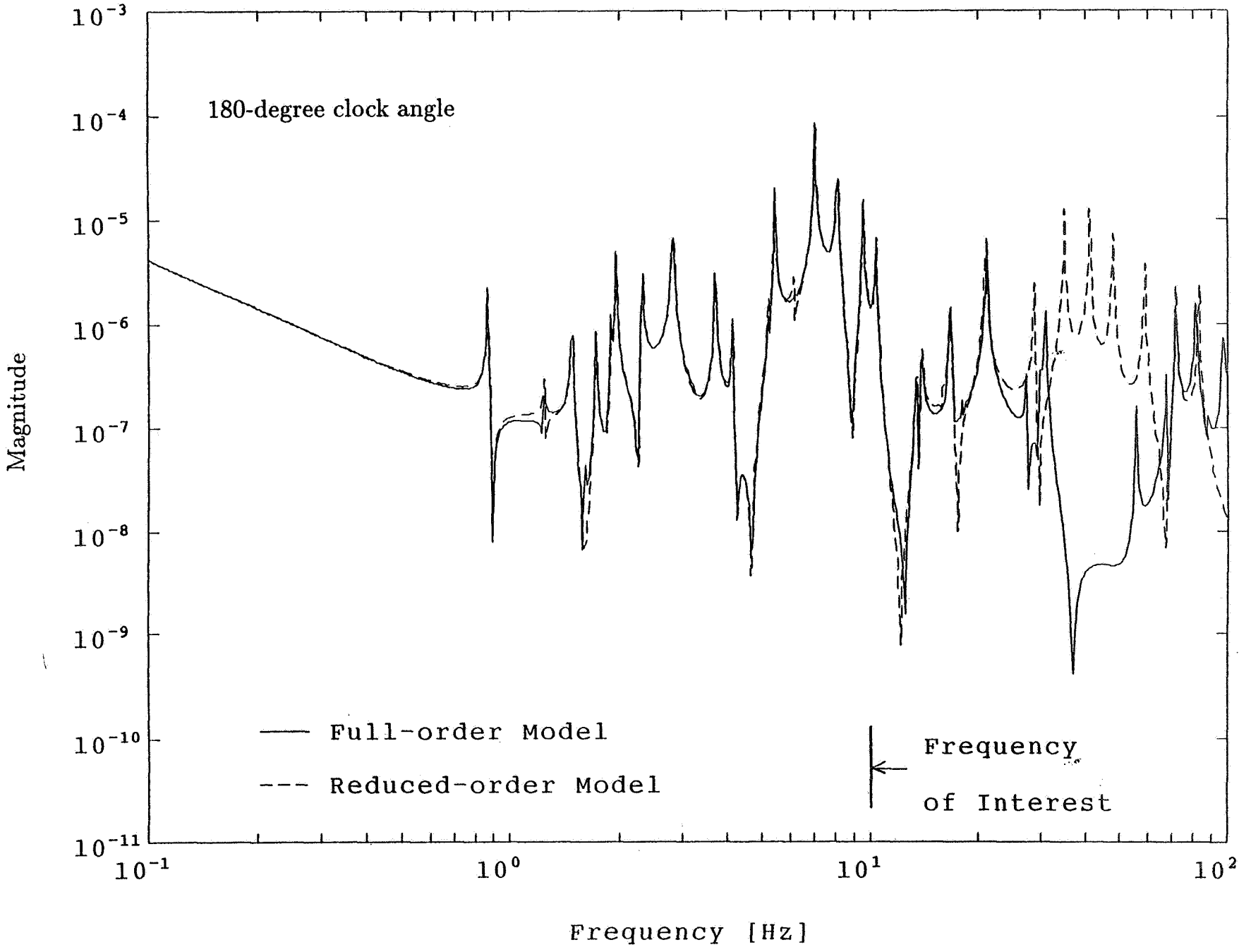
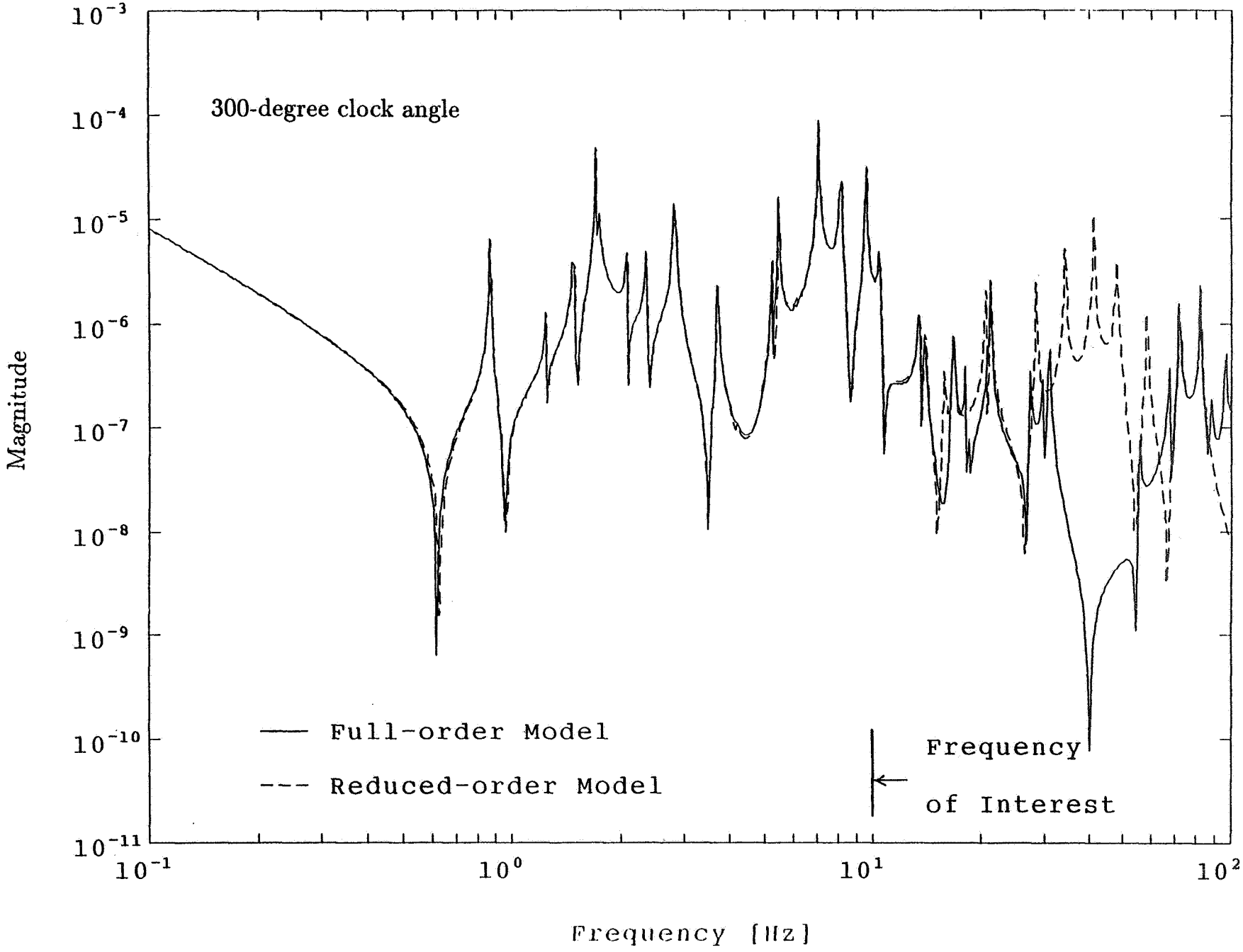


Fig. 6 Bode Plot Comparison of Full- and Reduced-order Models



Applications of Computer Algebra to Distributed Parameter Systems

by Joel A. Storch

The Charles Stark Draper Laboratory
Cambridge, Massachusetts

Abstract

In the analysis of vibrations of continuous elastic systems, one often encounters complicated transcendental equations with roots directly related to the system's natural frequencies. Typically, these equations contain system parameters whose values must be specified before a numerical solution can be obtained. The present paper presents a method whereby the fundamental frequency can be obtained in *analytical form* to any desired degree of accuracy. The method is based upon truncation of rapidly converging series involving inverse powers of the system natural frequencies. A straightforward method to developing these series and summing them in closed form is presented. It is demonstrated how Computer Algebra can be exploited to perform the intricate analytical procedures which otherwise would render the technique difficult to apply in practice. We illustrate the method by developing two analytical approximations to the fundamental frequency of a vibrating cantilever carrying a rigid tip body. The results are compared to the numerical solution of the exact (transcendental) frequency equation over a range of system parameters.

Introduction

The general availability of computer algebra systems has resulted in analyses of complicated problems which heretofore have been regarded as analytically intractable.These tools practically eliminate the tedious error prone manipulations required by hand-derivations and allow the analyst to explore various analytical treatments which would be too costly otherwise. In the same way that digital simulation has revolutionized the numerical treatment of engineering problems, symbolic computation promises to be a powerful tool in analytical investigations.

In the area of multibody dynamics, computer algebra has been used to derive the full nonlinear equations of motion in symbolic form [1]. These equations are typically so complex that they daunt inspection. However, built in translators convert these equations into a higher programming language e.g FORTRAN which results in extremely efficient digital simulations. For sufficiently simple systems, symbolic representations can be of direct use in studying such issues as elastic stability and buckling [2]. Perturbation methods are ideally suited to treatment by symbolic computation [3].

The classical method of determining the natural frequencies of a continuous elastic system results in an eigenvalue problem and associated characteristic equation. In general, this equation is transcendental and is embedded with various system parameters. Thus recourse must be made to numerical methods to solve these equations and the dependencies of the frequencies upon the various parameters can only be revealed through exhaustive computation. It therefore appears desirable to be able to approximate the roots of these equations by analytical expressions which are relatively simple yet accurate. The principal idea behind the method presented in this paper is to find closed form expressions for infinite series, the terms of which involve inverse powers of the natural frequencies. Truncating the series after the first term gives an approximation to the fundamental frequency. By summing sufficiently high powers, this approximation can be made arbitrarily accurate; but the resulting formula increases in complexity. The methods whereby others have addressed this problem are quite varied, ranging all the way from Fourier Series to complicated contour integration and difficult procedures involving integral equations. Hughes [4] obtains numerous modal identities by expanding the Green's function in a series of eigenfunctions. The technique we present is extremely simple and appears to have been applied in a restricted form by Lord Rayleigh [5]. Our result is very general and can be applied to any vibrational system one the characteristic equation is established. The only difficulty in applying the method is the need to develop complicated transcendental functions into Taylor series and manipulating the resulting coefficients. However, with the use of a computer algebra system, this task becomes almost trivial. The method is illustrated by deriving two analytical approximations to the fundamental frequency of a vibrating cantilever carrying a rigid tip body. The accuracy of these results is verified by comparisons with numerical solution of the frequency equation over a range of

parameter values.

Approximations Based Upon Rayleigh's Principle

Before presenting the technique predicated on infinite series, let us consider a "symbolic" solution to a prototype vibration problem employing the celebrated Rayleigh Principle. The elastic system consists of an Euler-Bernoulli beam cantilevered at one end, and carrying a rigid tip body at the other (see Fig. 1.A). The beam has a constant mass density (per unit length) ρ, uniform bending stiffness EI,and nominal length ℓ. A rigid tip body of mass m and inertia J (about P) is attached to the beam tip at $x=\ell$. We denote by c the distance from P to the tip body mass center.The derivation of the partial differential equation of motion and associated eigenvalue problem is given in Appendix A. The system eigenvalues are the solutions β_k to eq.(A.9) and are seen to depend upon the three dimensionless tip body parameters

$$m^*=\frac{m}{\rho\ell},\quad c^*=\frac{c}{\ell},\quad J^*=\frac{J}{\rho\ell^3}$$

The relationship between the eigenvalues and the system natural frequencies is given by eq.(A.10).

Let us approximate the beam deflection u(x,t) with a cubic polynomial in x.

$$u(x,t)=\xi^2 q_1(t)+\xi^3 q_2(t)$$

where $\xi=x/\ell$ and the geometric boundary conditions at x=0 have been observed. Here $q_1(t),q_2(t)$ are undetermined generalized coordinates. The system kinetic energy T and potential energy V are then discretized into the respective quadratic forms (see eqs.(A.12) & (A.11))

$$T=\left[\frac{\rho\ell}{10}+\frac{m}{2}(1+2c^*)+\frac{2}{\ell^2}(J-mc^2)\right]\dot{q}_1^2+\left[\frac{\rho\ell}{14}+\frac{m}{2}(1+3c^*)^2+\frac{9}{2\ell^2}(J-mc^2)\right]\dot{q}_2^2+$$

$$\left[\frac{\rho\ell}{6}+m(1+2c^*)(1+3c^*)+\frac{6}{\ell^2}(J-mc^2)\right]\dot{q}_1\dot{q}_2$$

$$V=\frac{2EI}{\ell^3}(q_1^2+3q_2^2+3q_1q_2)$$

If we write $T=1/2\dot{\mathbf{q}}^T[M]\dot{\mathbf{q}}$ and $V=1/2\mathbf{q}^T[K]\mathbf{q}$, then the system's first two natural frequencies are approximated by the roots ω_i of the characteristic polynomial

$$\det([K]-\omega^2[M])=0$$

Expanding this determinant and solving the resultant quadratic is relatively painless if a computer algebra system is invoked.The resulting expression for the fundamental frequency can be written in the form

$$\omega_1=\sqrt{EI/\rho\ell^4}\ \beta_1^2$$

with

$$\begin{aligned}\beta_1&=\left[\frac{1260}{(630c^*+210)m^*+630J^*+t_3+51}\right]^{1/4}\\ \text{where } t_3&=2\sqrt{3}(33075J^{*2}+1260J^*+t_2+t_1+208)^{1/2}\\ \text{and } t_2&=[(66150c^*+11025)J^*+4200c^*+1680]m^*\\ t_1&=(44100c^{*2}+22050c^*+3675)m^{*2}\end{aligned}\qquad(1)$$

This result of course provides an upper bound to the true fundamental frequency.

It should be noted that this method meets with practical difficulties when one attempts to improve the accuracy by retaining additional terms in the expansion of the elastic displacement. The higher degree of the concomitant characteristic polynomial renders an analytical solution impossible. The method to be described in the next section does not have this limitation.

Approximations Based Upon Infinite Series

The current method is based upon truncation of infinite series in the frequencies ω_n such as $\sum_{n=1}^{\infty} 1/\omega_n^2$, $\sum_{n=1}^{\infty} 1/\omega_n^4$ etc. where the sum can be expressed as a relatively simple algebraic function of the system parameters. If the series convergence is sufficiently rapid, then truncating the series after the first term yields a formula which approximates the fundamental frequency ω_1. Clearly, by summing sufficiently high powers of ω_n^{-1} we can approximate the first frequency to any desired degree of accuracy and will always have a *lower* bound. As will be seen, the corresponding formulas become increasingly complex. However, it should be pointed out that the generation of these higher order results can always be carried out in practice unlike the procedure of the last section. Hughes [4] generates series like the above and expresses the sum as a volume integral containing products of the Green's function with the mass density. He notes the difficulty of performing these integrations when the powers of ω_n^{-1} increase. Our method only requires a Taylor series expansion once the transcendental frequency equation is established.

Before considering the case of a transcendental equation, we present an elementary result from the theory of polynomial equations.

Given the polynomial equation

$$1+\alpha_1 z+\alpha_2 z^2+\cdots+\alpha_n z^n=0$$

with roots $z_i (i=1,2,\cdots,n)$ (over the field of complex numbers), we can show that

$$(a)\ \sum_{i=1}^{n} \frac{1}{z_i} = -\alpha_1$$

$$(b)\ \sum_{i=1}^{n} \frac{1}{z_i^2} = \alpha_1^2 - 2\alpha_2$$

Proof

First note that if 0 is a root, it can be removed leaving a deflated polynomial with no zero roots. Hence there is no loss in generality if we assume all $z_i \neq 0$. The general polynomial with roots $z_1, z_2, \cdots, z_n$ can be written as

$$(z-z_1)(z-z_2)\cdots(z-z_n)=0$$

Expanding and dividing through by the product $(-1)^n z_1 z_2 \cdots z_n$ we obtain

$$1-\left(\frac{1}{z_1}+\frac{1}{z_2}+\cdots+\frac{1}{z_n}\right)z+\left(\frac{1}{z_1 z_2}+\frac{1}{z_1 z_3}+\cdots+\frac{1}{z_{n-1} z_n}\right)z^2+\cdots+\frac{(-1)^n}{z_1 z_2 \cdots z_n}z^n = 0$$

which is of the desired form. It follows that

$-\alpha_1$ = sum of reciprocals of roots

α_2 = sum of products of the reciprocals of the roots taken 2 at a time

Hence, $\alpha_1^2 - 2\alpha_2 = \sum_{i=1}^{n} \frac{1}{z_i^2}$.

Sums involving higher powers of the inverse roots can also be generated. Thus

$$\sum_{i=1}^{n} \frac{1}{z_i^3} = 3\alpha_1\alpha_2 - 3\alpha_3 - \alpha_1^3,$$

$$\sum_{i=1}^{n} \frac{1}{z_i^4} = \alpha_1^4 - 4\alpha_1^2\alpha_2 + 2\alpha_2^2 + 4\alpha_1\alpha_3 - 4\alpha_4 \text{ etc.}$$

In an effort to adopt these results to the case of a transcendental equation $f(z)=0$ with an infinite number of roots, it is natural to expand $f(z)$ in a power series and *formally* apply the above formulas to this "infinite degree" polynomial. It turns out that this procedure can be proven mathematically valid when f is an entire function of the complex variable z. We will not present the proof here but refer the reader to [6] for the necessary theory of entire (integral) functions.

As a means of illustration, we apply this technique to approximate the fundamental frequency of the beam-tipbody system considered above. Using power series expansions for the trigonometric and hyperbolic functions, the frequency equation (A.9) assumes the form

$$1-\frac{1}{12}(12J^*+12m^*c^*+4m^*+1)\beta^4+$$
$$\frac{1}{5040}[(420m^*+168)J^*-420m^{*2}c^{*2}+56m^*c^*+8m^*+1]\beta^8-\cdots=0 \tag{2}$$

Since the coefficients of β and β^2 are zero, we conclude that $\sum_{n=1}^{\infty}\frac{1}{\beta_n}=0$ and $\sum_{n=1}^{\infty}\frac{1}{\beta_n^2}=0$. These two results become immediately obvious, since, if $\beta_k>0$ is a root of eq.(A.9) then so are: $-\beta_k$, $i\beta_k$, $-i\beta_k$. In order to obtain series converging to a nonzero result, write eq.(2) as

$$1+\alpha_1\beta^4+\alpha_2\beta^8+\cdots=0$$

and form the auxiliary "polynomial"

$$1+\alpha_1 z+\alpha_2 z^2+\cdots=0 \tag{3}$$

If β_k is a root of eq.(2), then $z_k=\beta_k^4$ is a root of eq.(3). This artifice coalesces the quadruple of roots $\{\beta_k,-\beta_k,i\beta_k,-i\beta_k\}$ of eq.(2) into a single root of eq.(3). Applying our method to the auxiliary equation(3), we obtain

$$\sum_{n=1}^{\infty}\frac{1}{\beta_n^4}=\frac{12J^*+12m^*c^*+4m^*+1}{12} \tag{4}$$

and

$$\sum_{n=1}^{\infty}\frac{1}{\beta_n^8}=\{(5880c^{*2}+3360c^*+560)m^{*2}+[(10080c^*+2520)J^*+728c^*+264]m^*+$$
$$5040J^{*2}+504J^*+33\}/5040 \tag{5}$$

Truncating the above series after the first term leads to the respective approximations

$$\beta_1=\left(\frac{12}{12J^*+12m^*c^*+4m^*+1}\right)^{1/4} \tag{6}$$

and

$$\beta_1=\left(\frac{5040}{s_1+s_2+s_3}\right)^{1/8} \tag{7}$$

where

$s_1 = (5880c^{*2} + 3360c^* + 560)m^{*2}$

$s_2 = [(10080c^* + 2520)J^* + 728c^* + 264]m^*$

$s_3 = 504J^*(1 + 10J^*) + 33$

Numerical Results

Verification of the modal identities (4) & (5) is provided in Table 1 below. The eigenvalues β_n were generated by numerically solving the transcendental equation (A.9); the sequences of partial sums appear in the last two columns. The numbers in the last row $(n=\infty)$ were obtained from the theoretical values appearing on the right hand sides of equations (4) and (5). All values were generated with $m^*=2.0$, $J^*=0.028$, and $c^*=0.1$.

Table 1 – Partial Sums of Series:
Eigenvalues of Beam/tipbody

n	β_n	$\sum_{k=1}^{n} \beta_k^{-4}$	$\sum_{k=1}^{n} \beta_k^{-8}$
1	1.0077	.9699	.9408
2	3.4599	.9769	
3	5.9100	.9777	
4	8.5047	.9779	
5	11.3806	.9780	
∞		.9780 (eq. 4)	.9408 (eq. 5)

The two approximations to the "dimensionless frequency" β_1 based upon series truncation (eqs. (6)&(7)) are tabulated in Table 2 below for the case of a pure tip mass – $J^*=c^*=0$. The values in the second column ('β_1') were obtained by numerical solution of eq.(A.9). As the value of the tip mass increases (relative to the mass of the beam),both approximations improve. As expected,the approximation based upon eq.(7) is superior to that supplied by eq.(6).

Table 2 – Fundamental Frequency Approximations for Beam with Tip Mass

m^*	β_1	Eq. 6 (% error)	Eq. 7 (% error)
0.0	3.5160	3.4641 (1.5)	3.5154 (1.6 E-02)
5.0	0.7569	0.7559 (0.13)	0.7569 (1.4 E-04)
10.0	0.5414	0.5410 (.069)	0.5414 (3.7 E-05)
15.0	0.4437	0.4435 (.047)	0.4437 (1.7 E-05)
20.0	0.3850	0.3849 (.035)	0.3850 (9.7E-06)

Table 3 below is similar in format to Table 2 but was generated with $J^*=1$ and $c^*=0$, which represents a relatively large concentrated inertia at the tip of the beam. In this case we see that the approximations degrade with increasing m^*.

Table 3 – Fundamental Frequency Approximations for Beam with Tip Body
$c^*=0$, $J^*=1$

m^*	β_1	Eq. 6 (% error)	Eq. 7 (% error)
1.0	.8679	.8402 (3.2)	.8669 (.11)
1.3	.8406	.8120 (3.4)	.8396 (.12)
1.5	.8236	.7947(3.5)	.8225 (.13)
1.8	.7995	.7708 (3.6)	.7984 (.14)
2.0	.7845	.7559 (3.6)	.7833 (.14)

Appendix A

We shall presently analyze the free vibration of a uniform cantilevered beam carrying a rigid tip body. Expressions for the potential and kinetic energies as well as the transcendental frequency equation are established.

Fig. 1.A below depicts the system in a deformed state. A uniform beam of mass density ρ (per unit length), bending stiffness EI and length ℓ lies along the x axis when in equilibrium. A rigid body of mass m and moment of inertia J (about P) is attached to the beam tip at P. The distance between P and the rigid body mass center is c, and this directed line segment lies along the beam tip tangent direction (to prevent the tip body from exerting axial loads onto the beam). The small transverse displacement of the beam is denoted by u(x,t).

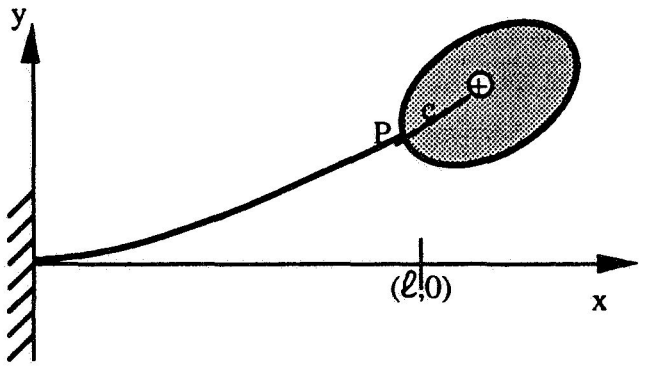


Fig. 1.A – clamped beam with tip body

We shall assume that the displacement of the beam and its slope are small quantities and therefore make the approximation that the angle θ_P between the x axis and the beam tip tangent line at P can be approximated by $u_x(\ell,t)$. Denoting the inertial velocities of P and the tip body mass center by $\mathbf{v}_P$ and $\mathbf{v}_\oplus$ respectively, we can write

$$\mathbf{v}_\oplus = \mathbf{v}_P + \boldsymbol{\omega}_P \times \mathbf{c}$$

where $\boldsymbol{\omega}_P = u_{xt}(\ell,t)\mathbf{k}$ is the angular velocity of the tip body and $\mathbf{c}$ is the vector from P to the tip body mass center. Recalling that $|\theta_P|$ is small and neglecting the term $\theta_P\dot{\theta}_P$, we find

$$\mathbf{v}_\oplus = \left[\frac{\partial u}{\partial t}(\ell,t) + c\frac{\partial^2 u}{\partial x \partial t}(\ell,t)\right]\mathbf{j} \qquad \text{(A.1)}$$

The expressions for the absolute translational and rotational accelerations of the tip body

C-2

mass center follow from the above by direct differentiation.

In order to write the boundary conditions for u(x,t) at the endpoint $x=\ell$, we consider a free body diagram of the tip body. As indicated in Fig. 2.A, the beam exerts a force *S* directed along the *y* axis and a moment *M* directed along the *z* axis upon the tip body at the point P.

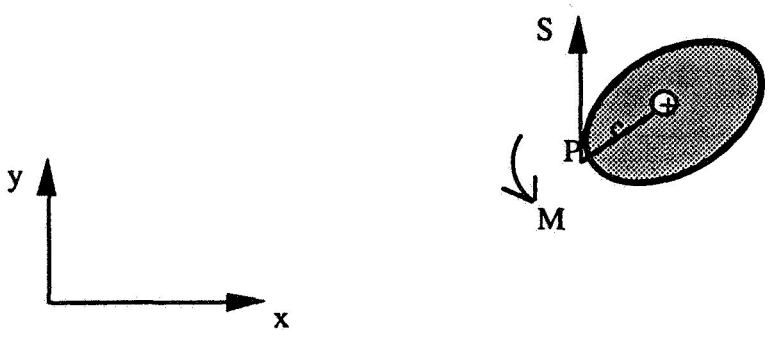


Fig. 2.A free body diagram of tip body

The equation of motion for the tip body along the *y* axis is

$$S=m\left[\frac{\partial^2 u}{\partial t^2}(\ell,t)+c\frac{\partial^3 u}{\partial x\,\partial t^2}(\ell,t)\right]$$

From elementary beam theory the shearing force in the beam at $x=\ell$ is given by $S=EI\,\partial^3 u/\partial x^3\big|_{x=\ell}$. In conjunction with the above, this supplies one of the required boundary conditions.

$$EI\frac{\partial^3 u}{\partial x^3}-m\left[\frac{\partial^2 u}{\partial t^2}+c\frac{\partial^3 u}{\partial x\,\partial t^2}\right]=0 \text{ at } x=\ell \qquad (A.2)$$

The second boundary condition at $x=\ell$ is obtained by considering the rotational motion of the tip body. If we denote by **h** the angular momentum of the tip body about its mass center, then we have the relation

$$\frac{d}{dt}\mathbf{h}=\mathbf{M}-\mathbf{c}\times\mathbf{S}$$

Taking the z component of this equation,neglecting the second order term in ω_P and using

the relation $M=-EI\partial^2u/\partial x^2\big|_{x=\ell}k$, we arrive at the result

$$EI\frac{\partial^2u}{\partial x^2}+mc\frac{\partial^2u}{\partial t^2}+J\frac{\partial^3u}{\partial x\partial t^2}=0 \quad \text{at } x=\ell \tag{A.3}$$

Since the beam is clamped at x=0, we have the two additional boundary conditions

$$u(0,t)=0 \quad \text{and} \quad \frac{\partial u}{\partial x}(0,t)=0 \tag{A.4}$$

The partial differential equation for free vibration is the well known relation

$$EI\frac{\partial^4u}{\partial x^4}+\rho\frac{\partial^2u}{\partial t^2}=0$$

We now proceed to solve the above homogeneous equation subject to the geometric boundary conditions (A.4) and natural boundary conditions (A.2) & (A.3). Seeking solutions of the form $e^{i\omega t}\varphi(x)$ we are led to the eigenvalue problem

$$\frac{d^4\varphi}{dx^4}-\lambda\varphi=0 \tag{A.5}$$

$$\varphi'''(\ell)+\frac{m\lambda}{\rho}\left[\varphi(\ell)+c\,\varphi'(\ell)\right]=0 \tag{A.6}$$

$$\varphi''(\ell)-\frac{\lambda}{\rho}\left[mc\,\varphi(\ell)+J\varphi'(\ell)\right]=0 \tag{A.7}$$

$$\varphi(0)=\varphi'(0)=0 \tag{A.8}$$

where ($'$) indicates differentiation with respect to x.

It can be shown that all the eigenvalues are positive. The general solution of eq.(A.5) is

$$\varphi(x)=c_1\sin\alpha x+c_2\cos\alpha x+c_3\sinh\alpha x+c_4\cosh\alpha x$$
$$\left(\alpha\equiv\lambda^{1/4}>0\right)$$

In order to have a nontrivial solution satisfying the boundary conditions (A.6),(A.7) & (A.8), the eigenvalues must satisfy the transcendental characteristic equation

$$m^*(J^*-m^*c^{*2})\beta^4(1-\cos\beta\cosh\beta)+m^*\beta(\cos\beta\sinh\beta-\sin\beta\cosh\beta)$$
$$-2m^*c^*\beta^2\sin\beta\sinh\beta-J^*\beta^3(\sin\beta\cosh\beta+\sinh\beta\cos\beta) \qquad (A.9)$$
$$+1+\cos\beta\cosh\beta=0$$

where we have introduced the "dimensionless frequency " $\beta\equiv\alpha\ell$ and the dimensionless tip body parameters are defined by

$$m^*=m/\rho\ell,\ c^*=c/\ell,\ J^*=J/\rho\ell^3$$

The natural frequencies are then given by

$$\omega_k=\sqrt{\frac{EI}{\rho\ell^4}}\,\beta_k^2 \qquad (A.10)$$

For purposes of reference,the systems potential energy V is in the form of strain energy stored in the beam and is given by the formula

$$V=\frac{EI}{2}\int_0^\ell\left(\frac{\partial^2u}{\partial x^2}\right)^2dx \qquad (A.11)$$

while the kinetic energy T is the sum of the translational kinetic energy of the beam and tip body with the rotational kinetic energy of the tip body. Employing eq.(A.1) we can write

$$T=\frac{1}{2}\int_0^\ell\left(\frac{\partial u}{\partial t}\right)^2\rho\,dx+\frac{m}{2}\left[\frac{\partial u}{\partial t}(\ell,t)+c\frac{\partial^2u}{\partial x\partial t}(\ell,t)\right]^2+\frac{1}{2}(J-mc^2)\left[\frac{\partial^2u}{\partial x\partial t}(\ell,t)\right]^2 \qquad (A.12)$$

References

(1) Rosenthal, D., *SD/FAST*, Symbolic Dynamics, Inc.

(2) Elishakoff,I. and Tang,J. "Buckling of Polar Orthotropic Circular Plates on Elastic Foundations by Computerized Symbolic Algebra,"*Computer Methods in Applied Mechanical Engineering*, **68**, 229-247, 1988.

(3) Rand,R.H. and Armbruster,D. *Perturbation Methods, Bifurcation theory and Computer Algebra,* Springer, New York, 1987.

(4) Hughes, P.C., "Modal Identities for Elastic Bodies, With Application to Vehicle Dynamics and Control," *Journal of Applied Mechanics,* Vol. 47, March 1980, pp.177-184.

(5) Lord Rayleigh, *Theory of Sound,* vol. 1, p.258.

(6) Titchmarsh, E.C. *The Theory of Functions,* Oxford University Press, New York 1939, Second Edition, Chapter viii.

Inverse Dynamics: Simultaneous Trajectory Tracking and Vibration Reduction with Distributed Actuators

Santosh Devasia

Eduardo Bayo

Department of Mechanical Engineering
University of California
Santa Barbara, CA 93106

ABSTRACT

This paper addresses the problem of inverse dynamics for articulated flexible structures with both lumped and distributed actuators. This problem arises, for example, in the combined vibration minimization and trajectory control of space robots and structures. A new inverse dynamics scheme for computing the nominal lumped and distributed inputs for tracking a prescribed trajectory is given.

1. Introduction

Inverse dynamics is an important problem in the control of articulated flexible structures such as space stations and manipulators. A solution for the nonredundant lumped actuator case has been provided by Bayo et. al., [1] and Book, [2]. This method produces bounded inputs which move a reference point on the structure along a desired trajectory. The inputs are necessarily non-causal when the structure dynamics are nonminimum phase. Elastic deformation which may cause vibration of the structure is also determined by the trajectory; our goal is to minimize such vibrations. The viability of distributed actuators for the control of structural vibrations, [3] , [4] and [5] , has motivated their use here for trajectory tracking.

Trajectory tracking of the structure can be accomplished by the use of the joint actuators alone [6] and in this sense the distributed actuators are redundant. We introduce the concept of using the extra actuation available through the distributed actuators in the structure to not only satisfy the trajectory tracking constraint but also minimize the accompanying elastic displacements during the motion. To obtain these new feedforward inputs, the inverse dynamics method suggested in [1] is extended to cover cases of redundantly-actuated structures. This use of distributed actuators in feedforward for end effector trajectory control is contrasted with the use of only the joint actuators in feedforward in an example of a flexible two link truss structure with distributed piezo-electric actuators to verify the efficacy of the proposed method.

The remainder of the paper is organized in the following format. The modeling of flexible structures with joint and distributed actuators, the formulation of the problem and its solution are presented in Section 2. Section 3 deals with an application of the proposed method to the example of a two link flexible truss. The discussions and conclusions are presented in section 4.

2. Formulation

The solution to the general multi-link inverse dynamics problem involves studying an individual link in the chain, coupling the equations of the individual links, and then recursively converging to the desired actuator inputs and corresponding displacements. This approach is presented below, beginning with a single link.

2.1 Equation of motion of a single link

To simplify the equations, we present the equations for a link with a revolute joint. The flexible link depicted in figure 1 forms part of a multi-link system. The link is shown with a revolute joint, however the formulation remains identical for a link with translational joint. The elastic deflections in the structure are defined with respect to a nominal position characterized by a moving frame whose origin coincides with the location of the hub of the link. The nominal motion of this frame is prespecified by its angular velocity ω_h, angular acceleration α_h and the translational motion of its origin. The above definition of the elastic displacements with respect to this nominal frame permits the linearization of the problem from the outset. Incorporating the kinematic model followed by Naganathan and Soni [7] in a finite element model (FEM), the equations of motion for a single link at any time t can be written as [1]

$$M\ddot{z} + \left[C + C_c(\omega_h)\right]\dot{z} + \left[K + K_c(\alpha_h,\omega_h)\right] z = B_T T + B_p V_p + F. \tag{2.1}$$

Where z is an R^n vector of the finite element degrees of freedom. M and K belong to $R^{n\times n}$ and are the conventional finite element mass and stiffness matrices respectively; C_c and $K_c \in R^{n\times n}$ and are the time varying Coriolis and centrifugal stiffness matrices, respectively. The $R^{n\times n}$ matrix C represents the internal viscous damping of the material. T is the unknown joint actuation. $F \in R^n$ contains the reactions at the end of the link, and the known forces produced by the rotating frame effect. The distributed actuator inputs at time t are The equivalent nodal forces at the FEM degrees of freedom due to the distributed actuators are represented by V_p, a R^{np} vector, where np is the number of distributed actuator inputs. B_T and B_p are constant matrices of dimensions R^n and $R^{n\times np}$, respectively. The set of finite element equations (2.1) may be partitioned as follows

$$M\begin{bmatrix}\ddot{\theta}_h\\ \ddot{z}_i\\ \ddot{z}_t\end{bmatrix} + \left[C + C_c(\omega_h)\right]\begin{bmatrix}\dot{\theta}_h\\ \dot{z}_i\\ \dot{z}_t\end{bmatrix} + \left[K + K_c(\alpha_h,\omega_h)\right]\begin{bmatrix}\theta_h\\ z_i\\ z_t\end{bmatrix} =$$

$$\begin{bmatrix}1\\0\\0\end{bmatrix} T + \begin{bmatrix}B_{p_h}\\ B_{p_i}\\ B_{p_t}\end{bmatrix} V_p + \begin{bmatrix}F_h\\ F_i\\ F_t\end{bmatrix} \tag{2.2}$$

where θ_h is the elastic rotation of the hub, z_t is the elastic deflection at the tip in the y direction, and the other $n-2$ finite element degrees of freedom are included in the vector z_i. The

force vector, F, and the B_p and B_T matrices are also partitioned similarly.

2.2 Minimization Objective

The requirement is to accurately track the end effector of the link along the given nominal trajectory without overshoot and residual vibrations. Additionally we also seek to minimize the ensuing structural vibrations during this motion by minimizing $J(T,V_p)$, a measure of elastic deflections in the structure defined as follows

$$J(T,V_p) = \int_{-\infty}^{\infty} z(t)^T z(t) dt. \tag{2.3}$$

Mathematically the objective can be stated as

$$\min_{(T,V_p) \in \hat{T}} J(T,V_p). \tag{2.4}$$

Where $\hat{T}$ is the set of all pairs of stable joint torque and distributed actuator inputs that when used to actuate the system defined by equation (2.2) yields $z_t(t) = 0$ for all t.

2.3 Solution Methodology

An iterative scheme is described below for each link. Equation (2.2) can be rewritten as

$$M\ddot{z} + C\dot{z} + Kz = B_T T + B_p V_p + F - C_c(\omega_h)\,\dot{z} - K_c(\alpha_h,\omega_h)\,z \tag{2.5}$$

where the time dependent Coriolis and centrifugal terms are kept on the RHS of the equation. The iteration procedure starts with the absence of the last two terms involving C_c and K_c in the right hand side. Then, the system of equations can be transformed into independent sets of simultaneous complex equations by means of the Fourier transform. For each of the evaluation frequency $\bar{\omega}$, equation (2.5) becomes

$$\left[M + \frac{1}{i\bar{\omega}}C - \frac{1}{\bar{\omega}^2}K\right] \begin{bmatrix} \overline{\ddot{z}_h} \\ \overline{\ddot{z}_i} \\ \overline{\ddot{z}_t} \end{bmatrix} = \begin{bmatrix} \overline{T} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \overline{F}_h \\ \overline{F}_i \\ \overline{F}_t \end{bmatrix} + \begin{bmatrix} B_{p_h} \\ B_{p_i} \\ B_{p_t} \end{bmatrix} \overline{V_p} \tag{2.6}$$

where the bar stands for the Fourier transform, and F represents the known forcing terms. After the first iteration it will also include the updated contributions from the Coriolis and centrifugal terms appearing in the RHS of equation (2.5). For any $\bar{\omega} \neq 0$, the matrix

$$H = \left[M + \frac{1}{i\bar{\omega}}C - \frac{1}{\bar{\omega}^2}K\right] \tag{2.7}$$

is a complex, symmetric and invertible matrix. For $\bar{\omega} = 0$ the system undergoes a rigid body motion and $H = M$ which is the positive definite invertible mass matrix. Let $G = H^{-1}$. Then

the above equation can be re-written as

$$\begin{bmatrix} \overline{\ddot{z}_h} \\ \overline{\ddot{z}_i} \\ \overline{\ddot{z}_t} \end{bmatrix} = \begin{bmatrix} G_{hh} & G_{hi} & G_{ht} \\ G_{ih} & G_{ii} & G_{it} \\ G_{th} & G_{ti} & G_{tt} \end{bmatrix} \left\{ \begin{bmatrix} \overline{T}(\varpi) \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \overline{F}_h \\ \overline{F}_i \\ \overline{F}_t \end{bmatrix} + \begin{bmatrix} B_{p_h} \\ B_{p_i} \\ B_{p_t} \end{bmatrix} \overline{V_p} \right\} . \tag{2.8}$$

The condition that the tip should follow the nominal motion is equivalent to $\overline{\ddot{z}_t} = 0$ for all ϖ. This induces a relationship between the joint actuation and the distributed actuator inputs and is obtained from the last row of the previous equation.

$$\overline{T} = - G_{th}^{-1} \begin{bmatrix} G_{th} & G_{ti} & G_{tt} \end{bmatrix} (\overline{F} + B_p \overline{V_p}) . \tag{2.9}$$

Substituting this expression for the input hub torque in equation (2.8) and using the property that $\overline{\frac{d^2 z}{dt^2}} = -\varpi^2 \overline{z}$ yields

$$\overline{z} = -\frac{1}{\varpi^2} (A\overline{V_p} + B) . \tag{2.10}$$

Where

$$A = [-G_{th}^{-1} GB_T (G_{th} \; G_{ti} \; G_{tt}) + G] \, B_p \tag{2.11}$$

and

$$B = [-G_{th}^{-1} GB_T (G_{th} \; G_{ti} \; G_{tt}) + G] \, \overline{F} . \tag{2.12}$$

Next we determine $\overline{V_p}$. Using Parseval's theorem, minimizing $J(T, V_p)$ in equation (2.4) is equivalent to minimizing $\|\overline{z}\|_2^2$ at each ϖ. This is a standard least squares approximation problem [8] and results in the following solution for the distributed actuator inputs,

$$\overline{V_p} = -U \begin{bmatrix} \Sigma^{-1} & 0 \\ 0 & 0 \end{bmatrix} V^* B \tag{2.13}$$

where Σ, U and V define the standard singular value decomposition of A as follows

$$V^* A U = \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} . \tag{2.14}$$

Where the conjugate transpose matrix operator is denoted by $*$. In addition if A has rank np, which is the number of distributed actuator inputs, then the least squares approximation yields

$$\overline{V_p} = -(A^* A)^{-1} A^* B . \tag{2.15}$$

A sufficient and necessary condition for A to have rank np is given next.

Lemma

$$\text{rank } [A] = np \quad \text{if and only if} \quad \text{rank } [B_T \mid B_p] = np+1 \tag{2.16}$$

Proof

$$\text{Rank } [B_T] = 1 \Rightarrow \text{rank } [GB_T(G_{th}\ G_{ti}\ G_{tt})] = 1$$
$$\Rightarrow \text{rank } \left[\hat{A} = [-G_{th}^{-1}\ GB_T(G_{th}\ G_{ti}\ G_{tt}) + G]\right] \geq n-1.$$

Since $B_T = [1\ 0\ 0]^*$, it is easy to see that the null space of $\hat{A}$ is the span of $[1\ 0\ 0]^*$. Hence rank $\hat{A}$ is $n-1$. Noting that $A = \hat{A}\ B_p$, the lemma follows easily. □

The above lemma requires that all the columns of the input matrices B_T and B_p be independent. This is computationally more efficient than checking the rank of A for each ϖ. Next, the corresponding joint torque component, $\overline{T}$ is then evaluated from equation (2.9). The inverse Fourier transforms for the feedforward inputs completes the first iteration and results in torques, T^1 and distributed inputs V_p^1. Then the forward dynamic analysis is carried out to compute K_c and C_c. F in the RHS of equation (2.5) is updated and the process is repeated to find the new input torques and voltages. The process is stopped at the n^{th} iteration if $\|T^n - T^{n-1}\|_2 + \|V_p{}^n - V_p{}^{n-1}\|_2 < \varepsilon$, where ε is some small positive constant. It may be noted that for slow motions the terms involving K_c and C_c are small relative to the other terms in equation (2.1) and the iterations converge in a few steps [1].

2.4 The Algorithms for the Multi-Link Cases

In the previous sub-section the procedure to evaluate the joint actuations of a single link was presented. This can be recursively extended for multi-link flexible manipulators. Algorithms are presented below for both open and closed chain multi-link mechanisms. These are similar to those proposed by Bayo et. al. [1].

Multi-Link Open Chain Case

1. Define the nominal motion (Inverse Kinematics of rigid manipulator).
2. For each link j, starting from the last one in the chain:
   a) Compute torque (or force) T^j and distributed actuator inputs P_v^j imposing $z_t^j = 0$ (Section 2).
   b) Compute the link reaction forces R^j from equilibrium.
3. Use equation (2.1) to compute the elastic displacement and joint angles.
4. Compute the inputs for the next link, $j-1$.

Multi-Link Closed Chain Case

1. Define the nominal motion (Inverse Kinematics of rigid robot).
2. Define an independent set of joint forces and reactions equal in number to the degrees of freedom of the robot.
3. For each link j, starting from the last one in the chain:
   a) Compute torque (or force) T^j and distributed actuator inputs P_v^j imposing $z_t^j = 0$ (Section 2).

b) Compute the link reaction forces R^j from equilibrium.

4. Use equation (2.1) to compute the elastic displacements and joint angles
5. Use elastic deflections to correct the nominal motion of each link.
6. Repeat steps 3 to 5 until convergence in the forces/torques is obtained.

This concludes the methodology. In the next section we present an application to a two-link flexible manipulator.

3. Example

A twolink truss experiment under development at UCSB is shown in figure 2. The trusses are made of lexan and have lumped masses (net 2 Kg for each link) distributed along their lengths. The first and the second links are tip loaded with 3.5 and 1 Kg respectively. Equivalent beam properties of the trusses used in the FEM model for simulations are Youngs modulus = 7 e^9 GPa, Link length = 1.2 m, density = 1500 Kg/m^3, cross sectional area = 4.378 e^{-5} m^2 and cross sectional area moment of inertia = 4.7244 e^{-9} m^4. Of the 10 spans in each link, two are piezo-electrically actuated. They are located at the second and ninth spans as shown in the figure 2. The piezo-electric stack actuators in those spans have the following properties. Cross sectional area, A_{cs} = 7.3 e^{-6} m^2, piezo strain to voltage constant, d_{sv} = .731 e^{-6} V^{-1}, Youngs modulus, E_p = 73 e^9 Gpa and distance of the actuator from the neutral axis of the truss, d_t = 1.27 e^{-2} m. Following the standard Bernoulli-Euler modeling for an applied voltage V_{input}, the piezo-electric actuation can be considered as two concentrated moments M acting at the two ends of the actuator [9] and [10]. Where M is given by

$$M = (d_{sv}\ N_p E_p A_{cs} d_t) V_{input} \tag{3.1}$$

and $N_p = 4$ is the number of piezos in each span. For the truss considered above $M = = 0.0198 V_{input}$. The desired trajectory is a rest to rest motion of the structure with initial conditions given by $\theta_1 = \theta_2 = 0$ and final conditions $\theta_1 = 11.25^o$ and $\theta_2 = -22.5^o$. $\theta's$ are the absolute angles of the links with respect to a frame fixed on the ground and are shown in figure 2. The nominal motion of the tip for each link are the trajectories followed by the tips of the links if the structure were rigid and followed the nominal angular motions shown in figure 3. Using the procedure in section 2.4 for open-chain mechanisms, open loop simulations were performed (1) using only the joint actuation for feedforward and (2) using the distributed piezo-electric actuators along with joint actuators in feedforward and the results are presented below.

Plots of the input piezo voltages and joint torques are presented in figures 4 and 5 respectively. To illustrate the viability of the proposed method figures 6 and 7 show the transverse structural midpoint deflections of the two links during the motion with and without the distributed actuators. Similar plots for the elastic hub rotations are shown in figures 8 and 9.

Thus the piezo-electric actuators show a significant reduction in the structural vibrations and demonstrate the viability of the proposed method. The consequent reduction in the induced strains in the structure allows the use of lighter elements and smaller actuators, especially in space structures where the loads are mainly inertial.

4. Conclusion

Typically distributed actuators like the piezo-electric ones cannot garner enough actuation to cause large motions in the structure. However they could be very effective in controlling the small structural deformations in the structure. Their use in the feedforward to aid the joint actuators for trajectory tracking is a novel idea developed in this paper. The method proposed was shown to be extremely efficient in removing structural vibrations from structures as seen in the example. Thus these feedforward actuations, obtained through the proposed inverse dynamics, augmented with joint angle feedback based closed loop controllers seem promising in the slewing control of flexible manipulators. This encouraging result motivates further work on distributed actuators in the control of flexible structures.

ACKNOWLEDGEMENT

Support from Air Force Office of Scientific Research through grant F49620-91-C-0095, the Astro Aerospace Corporation and TRW are gratefully acknowledged.

References

1. E. Bayo, M.A. Serna, P. Papadopoulos, and J. Stubbe, "Inverse Dynamics and Kinematics of Multi-Link Elastic Robots. An Iterative Frequency Domain Approach," *The International J. of Robotics Research*, vol. 8, No 6, Dec 1989.
2. D. Kwon. and W.J. Book, "An inverse dynamic method yielding flexible manipulator state trajectories," *Proc. of ACC*, pp. 186-193, 1990.
3. Fanson, J. L. and Garba, J. A., "Experimental Studies of Active members in Control of Large Space Structures," *Proc. AIAA 29th SDM Conf.*, pp. 9-17, 1988.
4. L. Meirovitch and H. Baruh, "Control of Self-Adjoint Distributed-Parameter Systems," *AIAA*, vol. 5, No 1, pp. 60-66, 1980.
5. M. J. Balas, "Active Control of Flexible Systems," *Journal of Optimization Theory and Applications*, vol. 25, No 3, pp. 415-436, 1978.
6. B. Paden, D. Chen, R. Ledesma, and E. Bayo, "Exponentially Stable Tracking Control for Multi-Joint Flexible-Link Manipulators," *ASME J. of Dynamic Systems, Measurement and Control*, vol. accepted for publication.
7. G. Naganathan and A.H. Soni, "Coupling Effects of Kinematics and Flexibility in Manipulators," *International Journal of Robotics Research*, vol. 6, No 1, pp. 75-85, 1987.
8. G.W. Stewart, *Introduction to Matrix Computations*, pp. 319-325, Academic Press, Inc., 1973.
9. E.F. Crawley and E.H. Anderson, "Detailed Models of Piezoceramic Actuation of Beams," *J. of Intelligent Material Systems and Structures*, vol. 1, pp. 4-24, 1990.
10. S. Devasia, "Modeling of Piezo Electric Actuators," *M.S. Thesis*, UCSB, 1990.

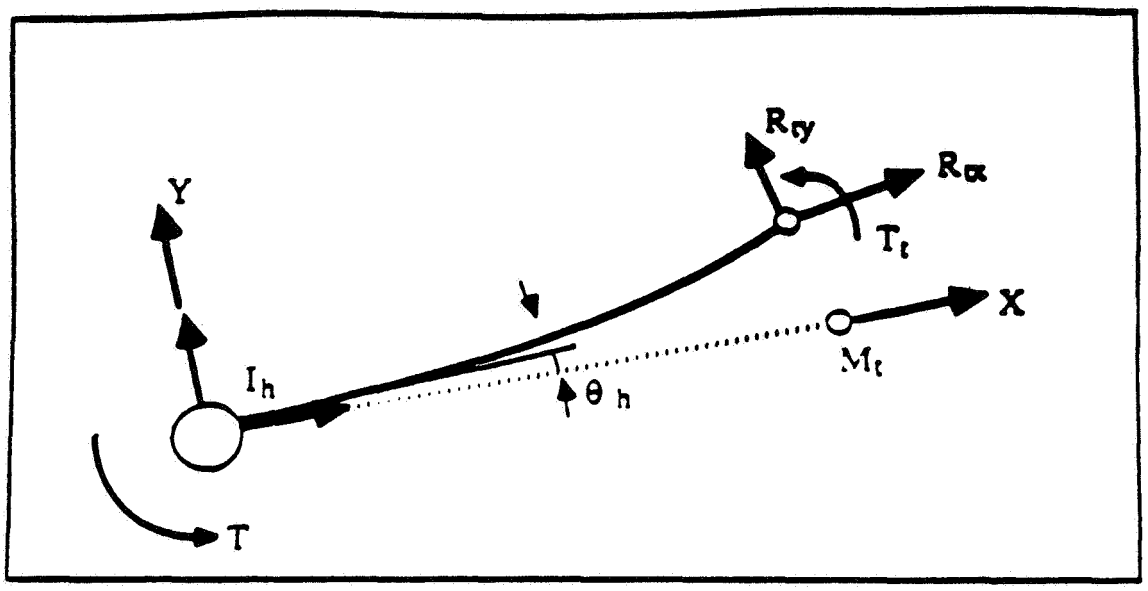


Figure 1. A Single Flexible Link

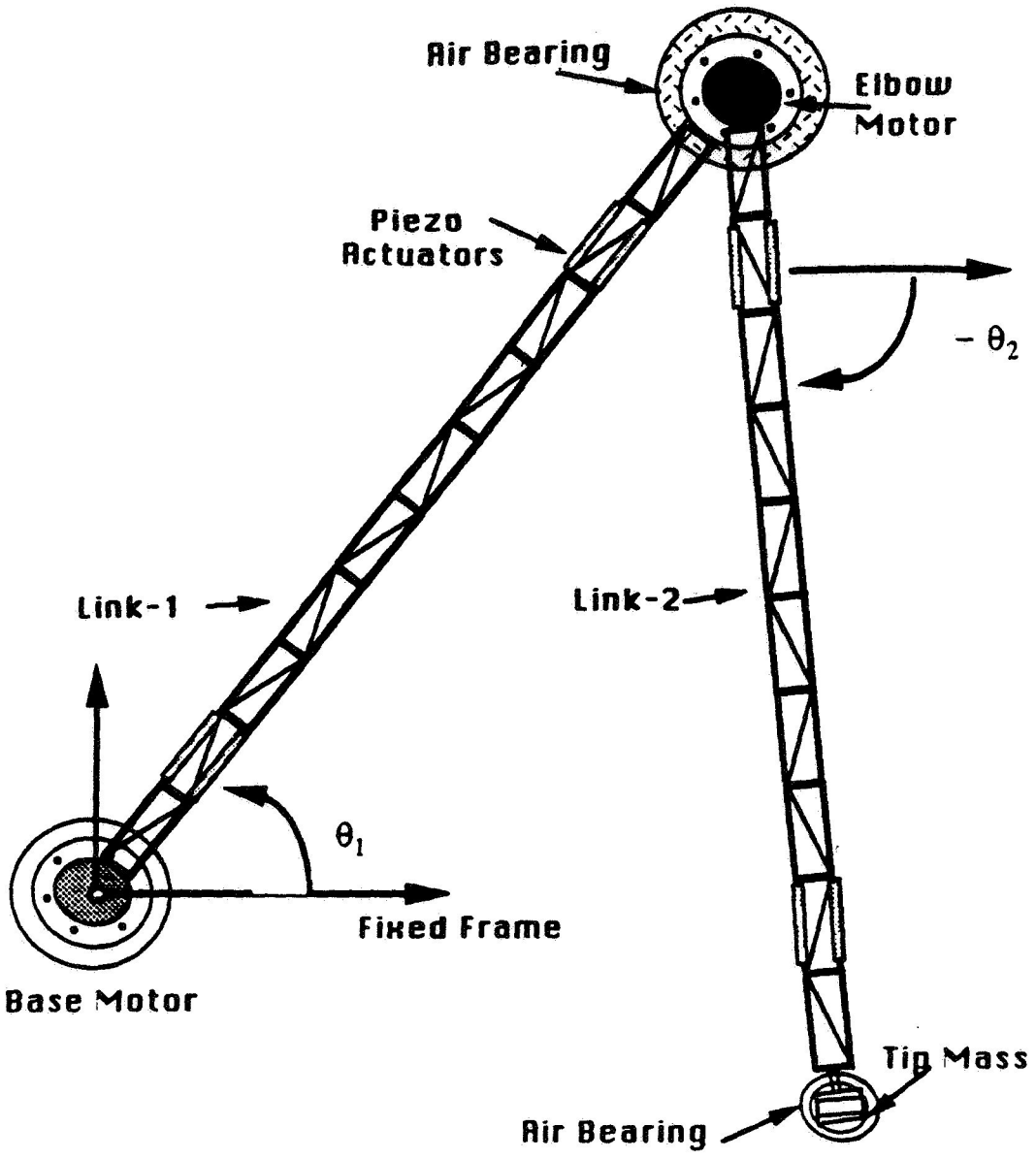


Figure 2. The Two Link Truss Structure

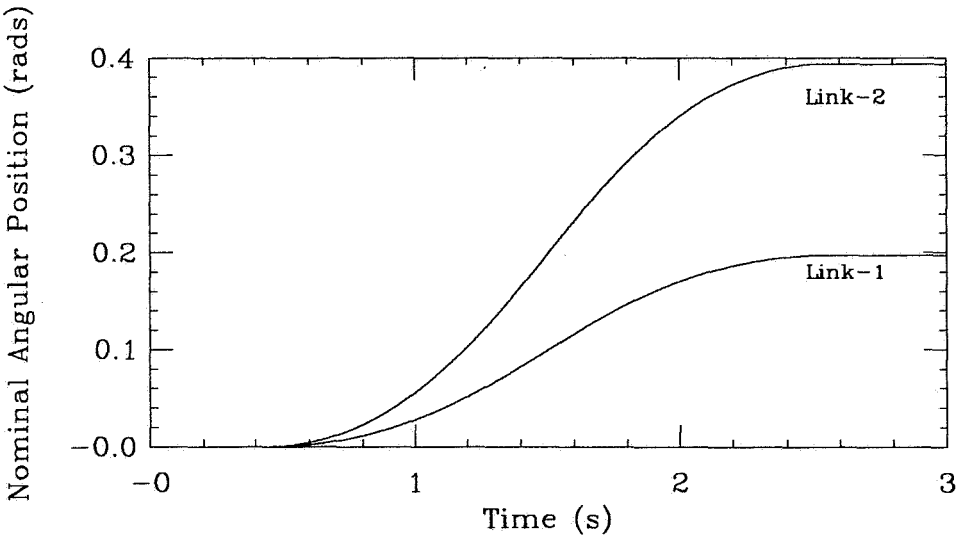


Fig.3: nominal angular positions

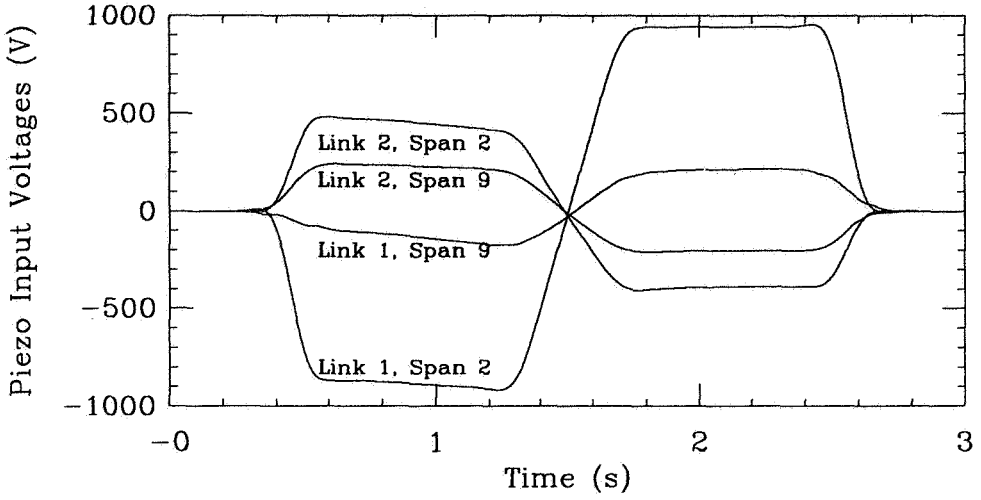


Fig.4: input piezo voltages

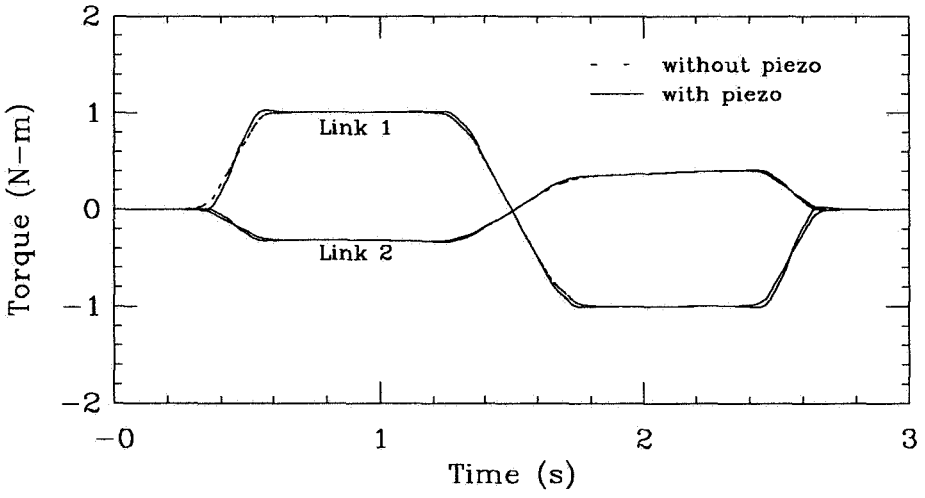


Fig.5: inverse dynamics torques

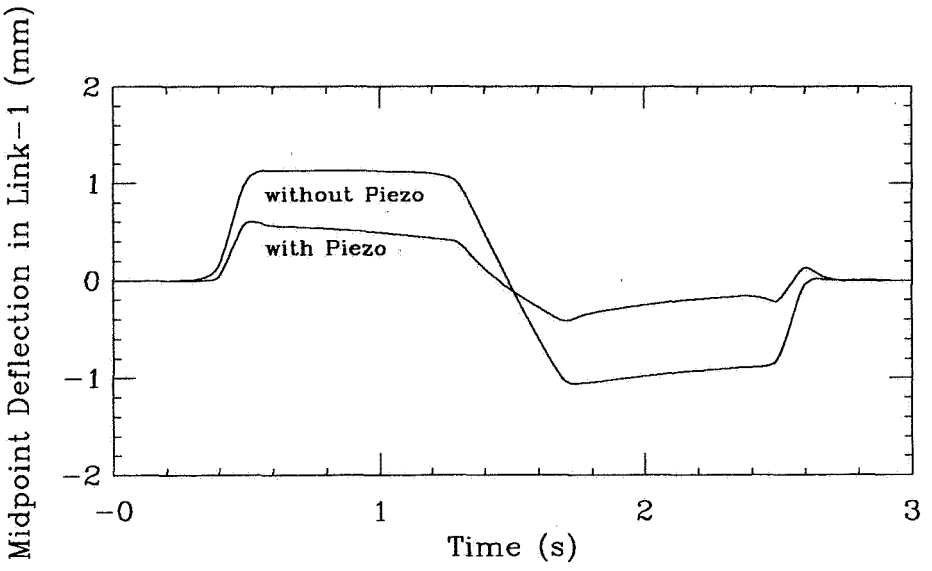


Fig.6: transverse deflection at midpoint of link 1

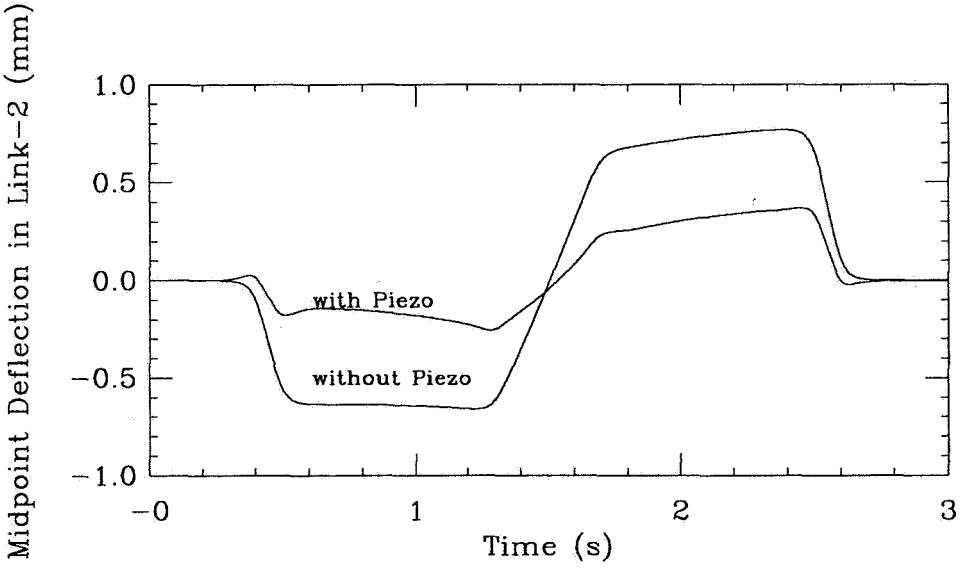


Fig.7: transverse deflection at midpoint of link 2

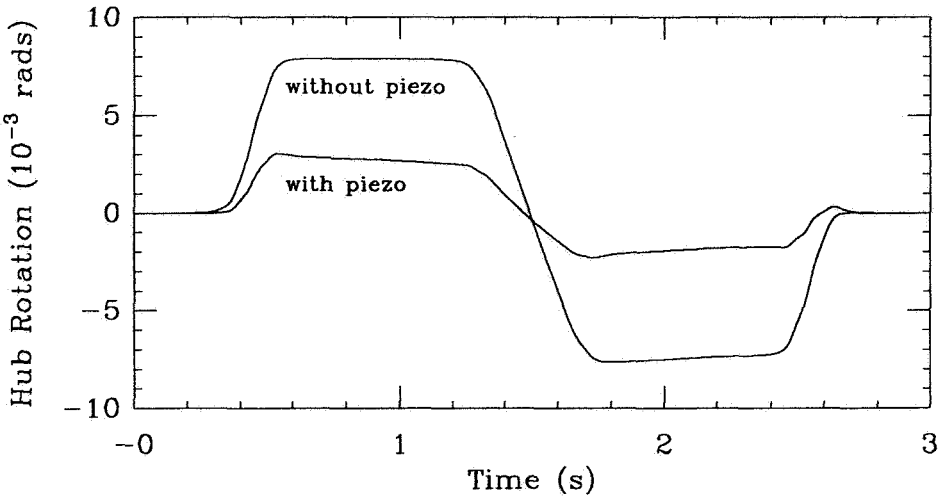


Fig.8: elastic hub rotation of link 1

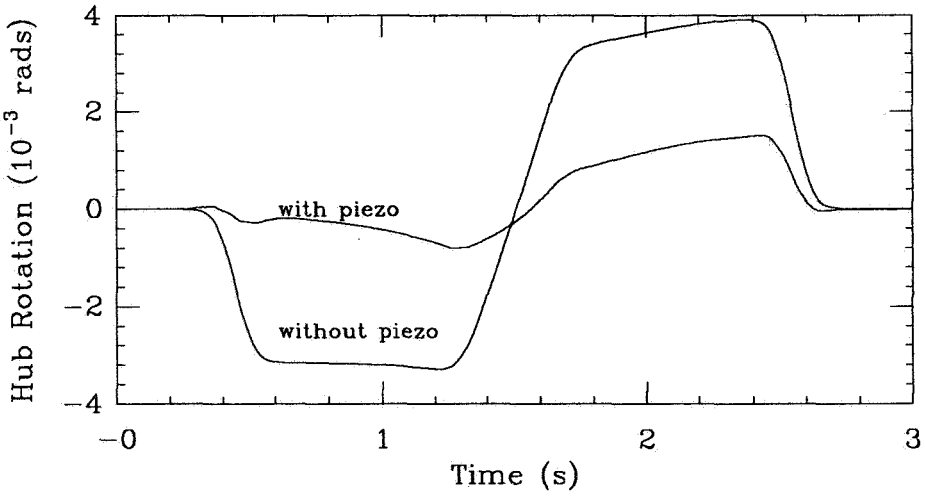


Fig.9: elastic hub rotation of link 2

Modelling Robotic Systems with DADS

L. W. Churchill*, I. Sharf*

March 30, 1992

Abstract

With the appearance of general off-the-shelf software packages for the simulation of mechanical systems, modelling and simulation of mechanisms has become an easier task. The authors have recently learned one such package, DADS, to model the dynamics of rigid and flexible-link robotic manipulators. In this paper, we present this overview of our learning experiences with DADS, in the hope that it will shorten the learning process for others interested in this software.

1 Introduction

The practice of robotic-systems simulation is presently undergoing a transition from user-customized to off-the-shelf software. Of course, the development of new methods will always require the development of new computer programs to test them. But with the appearance of general, (relatively) easy to use, simulation packages it is no longer necessary to write your own program every time you need to simulate a new mechanical system.

One such package is DADS[1], the Dynamic Analysis and Design System. The University of Victoria's department of mechanical engineering has recently acquired

*Department of Mechanical Engineering, University of Victoria, Victoria, British Columbia, V8W 2Y2 Canada.

[1]DADS is a product of Computer Aided Design Software, Incorporated, P.O. Box 203, Oakdale, Iowa 52319.

DADS for research use. As the authors' work on recursive algorithms for dynamics simulation came to fruition, we decided to use DADS as a standard in accuracy and performance tests of our own software. Thus, our primary objective was to check the results of our simulation of flexible-body open chains, in particular robot arms, against those of a completely different approach. As a secondary objective, we wished to use the package's plotting and animating capabilities, with both our own and DADS' data, to generate graphical and animated output.

We have developed a simulation implementing recursive solutions to both the *inverse* and *forward* dynamics problems for rigid-link open chains. These simulations have been extensively tested and provide highly accurate results. Our inverse dynamics code uses, as input, the angular displacements, joint rates and accelerations and generates, as output, the control forces to be applied to the actuators of the chain in order to obtain those trajectories. The forward dynamics code uses, as input, the control forces to be applied to the actuators and generates, as output, the angular displacements and rates of the joints and the resultant trajectories of the end-points of the links. The control forces can be supplied either as a prescribed set of data points or by the inverse dynamics routine. The inverse and forward algorithms differ sufficiently that a good indication of the accuracy of the simulation can be obtained from the rms error of the integrated solution vector, the set of angular displacements and joint rates for the whole chain.

We planned to use our inverse and forward dynamics programs to establish a confidence level for DADS' rigid-link dynamics, after which we would assume a "lesser than or equal" corresponding confidence level for DADS' flexible-link dynamics.

Thus, we wished to employ DADS to:

- Solve the inverse dynamics problem for a rigid-link model and compare the resultant control forces with ours.
- Solve the forward (simulation) dynamics for a rigid-link model and compare the resultant solution vector, its rms error and the trajectories of the end-points of the links to ours.

- Include link flexibility in the DADS model and simulate the motion of the system.
- Plot and animate DADS' output.

2 Getting Started

DADS comes complete with three accompanying texts: the theoretical, user's and examples manuals. The theoretical manual [1] is actually a hardcover book describing the theory underlying the program's algorithms. The user's manual [2] is really a reference manual. It includes a brief introduction to the program and some instructions in the use of the postprocessor (plotting) and the geometry and animation routines, but consists primarily of detailed outlines of the elements with which mechanical systems are modelled. The examples manual [3] consists of a large number of DADS models of varying complexity, but for the most part, the descriptions are limited to hard copies of intermediate data files, output files and plots. This set of manuals becomes quite useful once one achieves sufficient familiarity with the software.

In our version of DADS, Revision 6.1, the most commonly used portions of the program have been implemented via a graphic user interface with the remaining portions running in ordinary text windows. DADS must be run from OpenWindows (a Sun interpretation of X-Windows) and a resource/defaults file named daDS must be present in the home directory at startup.

It is worth making a few remarks on DADS' implementation. DADS can be heavy on system resources and doesn't always work smoothly with other applications. It can shut itself down if it finds another program running at startup and can interfere with other programs' operations, particulary those using color graphics. The older portions of the package work well within their text windows; though DGE, the animation routine, won't permit line editting after file input is completed. A minor problem with both the new and old command windows is that they automatically close upon job completion. This is convenient if the job ends successfully, but if the

termination was abnormal, any error messages presented flash offscreen too quickly to be read. Error messages from the routine performing the *analysis* may be recovered from an information file, but error messages from other portions of the program (*i.e.* model definition, plotting, graphics and animation) are lost.

The new command shell windows also have another fault. DADS generally uses Xterm window conventions but has preset many of the usual options. In general, these are matters of little consequence but, in combination with other factors, one of these option settings has proven a consistent source of error: DADS sets the input field to follow the mouse-pointer location. When using the data entry windows, the mouse is useful in moving quickly from one input cell to another (though it would work just as well with mouse-selected input and, for general data entry, the tab key is often more convenient anyway). However, in the new command shells, data entry is recognized only from the command line at the bottom of the window. Unfortunately, input is also allowed, prompted and echoed in the dialogue section of the window. Of course, the programs' text messages are displayed in this dialogue pane and, after a menu selection, the mouse-pointer ends up there as well. These attributes combine with the dialogue pane's larger size and central location to make it a natural site for keyboard input, even though it is completely nonfunctional. The problem is compounded by the choice of white text on dark blue and black backgrounds for the dialogue pane and command line, respectively[2]. On a black and white display, the boundary between the two fields is indistinguishable and input is very easily misdirected, resulting in corrupted data and incomplete models.

DADS is organized in program segments corresponding to the different tasks involved in building a mathematical model of a mechanical system. A model is defined and initially configured in the *preprocessor*. The simulation is then performed by the analysis package. If desired, plotting and data manipulation can be done in the *postprocessor*. Graphic representations of the model elements are developed with the geometry routines. And, finally, the simulation results and graphic representations

[2]These default settings can be changed by modifying the daDS file. Try changing *DADSMessage*background from navy to grey55 or slate blue.

are visually integrated via the animation program. In addition, DADS is equipped with a number of supplementary conversion routines which translate data and output files from one program segment for use in another. The heart of the package, however, consists of the preprocessor and analysis routines with which DADS performs simulations.

3 Simulation

Before one begins to model with DADS, it is helpful to develop an overview of the way the program works. In particular, we mention some assumptions which seem to be implicit in the program's operation and provide an outline of the different types and functions of DADS' elements.

Many of the difficulties we encountered in modelling our systems were rooted in the choice of reference frames we made for the links. DADS seems to have been originally designed to use a global reference frame in combination with local body center-of-mass frames. The system is assumed to operate with a given orientation to the ground and under the influence of gravity and possibly dissipative forces as well. These properties are undoubtedly those most appropriate for arbitrary mechanical systems but, in the analysis of robot arms, other situations are also common. The recursive algorithms we use make joint-based reference frames a natural choice for the links. And, since our main application of interest is the Canadarm, we deal with all external forces as special cases and default to a weightless, frictionless environment. DADS seemed capable of adapting to our point of view so we elected to define the DADS models in the same way we defined our recursive models. But, in places, DADS still implicitly relies on body center-of-mass reference frames, which led to several problems.

A DADS model is defined in terms of a variety of program "elements." The shear size of the program's element library can be overwhelming, leaving a new user bewildered as to which elements would be appropriate to build and drive a model.

At the highest level is the *system* element which defines the type of analysis to be performed—static, dynamic, inverse or kinematic—and sets global parameters such

as units, the run time and printing intervals. The gravitational field vector must be set here (it helps to define the units), though a scaling factor is also provided to adjust the magnitude. Tolerances for DADS assembly analysis and LU factorization are specified. Matrix operations should be set to SPARSE as the alternative, since FULL matrix operations, doesn't seem to work correctly. DADS is capable of performing a (useful) check that the model will assemble correctly to within the given assembly tolerance. Since we deal with relatively straightforward assemblies, we tend to set this tolerance value quite small (10^{-6}.) The type of reference frame to be used in the analysis is also selected here. Possible choices are global, local (body center of mass) or NCBF (non-centroidal body frames.) Contrary to our expectations, this option mainly affects the interpretation of reference points in the input data, except for reaction forces, whose coordinate frames are specified elsewhere. Output data is given in terms of the global and, sometimes, the local (body center-of-mass) frames. Finally, a debug flag may be set here. We found this useful mainly because it turns on the time echoing in the analysis window.

Each of the analysis types have their own element. We discuss only the elements relevant to the present application—inverse and dynamic. In the *inverse* element, one specifies the coordinates used to output the reaction forces. The analysis step size and solution tolerance is also determined here. We generally found the default values adequate for a first run. Only after the system was in working order did we try for greater precision. In the *dynamic* element one specifies the maximum integration step and the solution and integration tolerances. The defaults are adequate to get the system working but the tolerances had to be lowered to get the accuracies we desired.

Data elements describe the physical components of the system. In a *body* element one describes the physical properties of an individual component, defined in local (not NCBF) coordinates. One can apply external forces directly to the center of mass of a body but, for our system, we found another mechanism more convenient. If NCBF is the type of reference frame selected in the system element,

then all bodies must be associated with a corresponding *reference frame* element defining their coordinate system. Similarly, if a body is specified to be flexible it must be associated with a corresponding *flexible body* element which permits the definition of damping and/or external forces and points to a data file containing vibrational mode information from a finite elements program. Other data elements include an *initial condition* element (if absent, the associated value defaults to zero) and a *curve* element which defines a function in terms of a prescribed set of data points. We usually specified our control forces via curve elements. Curve elements can read data from a text file but, afterwards, such data cannot be edited. For this reason we suggest saving a model's configuration and curve elements separately. This greatly simplifies switching between sets of control forces, for example. We also suggest frequent saving when loading large files into curve element sets, as DADS can shut down unexpectedly during these operations.

The joint-constraint elements make up a large library of the various means for joining the bodies in the model. The name is indicative of their function—these elements are connectors with specified degrees of freedom but are *passive,* not active. Note that, despite the description in the user's manual, the order in which the bodies connected by the joint are specified can be significant. The robot arms that we have modelled with DADS use only *bracket joint* elements (connectors with no degrees of freedom) and *revolute joint* elements.

Most of the other-constraint elements enable one to model the physical consequences of the bodies' dimensions and the system's overall geometry. As our models assume an idealized geometry, we found most of these unneccesary. In the inverse dynamics analysis, joints are made to move with the *driver* element. A driver element may be specified as one of several types, driving: any of the coordinates, any component of the velocity, a distance, a relative angle or a relative translation. The driving function may be specified in a variety of ways: a curve element, a simple polynomial, a simple harmonic function, as the control output of a large set of control elements or in terms of a user-supplied subroutine. We used relative angle drivers in our inverse

dynamics models. The polynomial and harmonic functions were adequate for test cases though we will likely need to develop our own subroutines, eventually.

There are seven force elements which provide a wide variety of options for applying forces in the forward dynamics analysis. We applied control forces to our revolute joints by means of *rotational spring-damper-actuator* elements. RSDA's are, effectively, damped motors with torsion. The three types of torque contribution may have both constant and time-varying components. However, contrary to expectation, the actuator torque is applied backwards so as to be dissipative. This may be corrected in several ways. One is to change both the direction of the rotation axis or the order of the connecting bodies in the associated revolute joint element definition. Simpler, however, is adjusting the curve element to scale the applied torques by a value of -1.

To summarize, the set of elements we used in our inverse analysis consisted of a system and inverse element with a set of body and reference frame pairs connected by revolute joint-driver pairs. For dynamic analysis we used a system and dynamic element with a set of body and reference frame pairs connected by revolute joint-RSDA pairs. Initial condition elements were used and control forces were specified to the RSDA's via curve elements. For flexible-link dynamics we moved to body, reference frame, flexible body element triples.

4 Visual Presentation

As well as performing dynamical analyses, DADS is capable of plotting and/or animating the results. Plotting is accomplished with the postprocessor. Essentially any value associated with a component of the model may be plotted. DADS is also capable of combining data from different runs and data may also be read from (or saved to) text files. Plots may be displayed on screen or directed to files in a wide variety of formats.

Before animating a model, it is necessary to describe the geometries of the components. This is done with Geomake, one of the older portions of DADS. A disadvantage of this routine is that once one has created the pieces that make up an

animation, they cannot be edited. As a result, we soon learned not to build large parts files. Instead we defined command files to create the parts and saved them in separate files to be combined later as desired. DADS geometry definition is general and seems capable of generating virtually any desired object. As we were building a simple robot arm without the end effector we dealt with ordinary cylinders. In order to smooth the appearance of the arm's joints we added spheres to the proximal end of each cylinder. Also, since the geometric body description needn't correspond to its modelled dimensions, our payload was modelled as a cylinder much shorter than its actual length, enabling a clearer view of the motion of the small links at the end of the arm. These simple additions transformed the original collection of stubby cylinders into a continuous articulated arm. Geomake can create body-geometries in one or more colors but the animation routine requires one color per part. To begin with, we suggest white.

The output of Geomake must be converted to a format suitable for the animation routine. This is accomplished with a program called CONV, but this routine will not work if called from DADS' window menu (it builds an empty data file.) One shoul execute CONV from a system command line in order to get it to work properly. The other conversion routine, DADS2MOD works well.

DADS animation is impressive and easy to learn. Previously created and converted geometries are stored in .def files and may be modified and saved easily from within DADS Graphic Environment (DGE.) In fact, after one becomes familiar with DGE, the .def files become quite readable and may be simply modified with a text editor. The DGE is command driven, but a graphic interface may also be started and is to be highly recommended, though it should be called after the model has been "assembled" by viewing the first frame. DGE has several viewing modes and both the viewpoint and lighting may be changed at will. Color and shading are good and, while the animation was noticably jerky on our architecture (a SPARCstation IPC networked to a Sun 4), the speed was within acceptable limits. DGE's graphic window is small and the program will not resize it while running. However this flaw

may be overcome by executing DGE from a system command line (rather than DADS' window menu) where one may specify the starting size for the graphics window as a command option. We found DADS animations were often helpful in understanding the motions resulting from the application of arbitrary forces.

5 Results and Conclusions

Our first results with DADS were obtained for rigid-link models. We tested two systems, a single link rotating about a fixed base and a six-link system modelled after the Canadarm, with solution and integration tolerances of 10^{-6} and 10^{-8}, respectively. With the first model, we compared the joint reaction forces from DADS' inverse dynamics to the control forces generated with our software. We then used the joint reaction forces as input to DADS' forward dynamics and compared the integrated solution vector to the prescribed input trajectories. The agreement between these was good, the difference appearing only in the last decimal of DADS' single precision output. Curiously, the joint reaction forces agreed with our control forces only to an average of about four decimal digits. When DADS' forward dynamics was run with our control forces the integrated solution vector and resultant trajectories agreed with the originals to three or four digits. For the second more complicated system, even using DADS' joint reaction forces as input to the forward dynamics results in only four digits agreement between the output trajectories and the prescribed inputs. We concluded that, for a general robotic system, DADS' confidence level was about four digits.

Our results with flexible-link models began very poorly. The problem lay with an undocumented aspect of DADS' handling of flexible bodies. DADS is capable of using finite element data generated by a number of different programs. We used ANSYS and chose a body-fixed coordinate system coinciding with the NCBF coordinate system assigned to the body in DADS. However, for flexible bodies, DADS reinterprets the postion specified for the body center of mass as the origin of the flexible element coordinate system, with other complicated consequences as well. This prob-

lem was fixed by reassigning the finite element coordinate origin. A second difficulty was encountered when we modelled the flexible links as single-element cantilevered-beams. With this model, we could not obtain a solution in a reasonable amount of time. This problem was fixed by using a five-element cantilevered-beam model.

The data we were interested in comparing included the trajectories of the end-points of the links. DADS' output actually gives the trajectories of the centers of mass of the links. With flexible links, these values cannot be immediately converted to end-point trajectories, so we attempted to use a *point-of-interest* data element. This element is designed to provide information about nodes of interest in flexible bodies. Unfortunately, the output data for this element was quite incorrect, possibly a casualty of the use of two different coordinate systems, in DADS and the finite element package, for the link.

We have been able to obtain flexible-link output for a single-link system of a well-known spin-up problem. (Please refer to Kane *et al.* [4] for details.) The transverse tip deflection is shown in Figure 1 and displays the divergent behaviour previously obtained with other multibody simulation packages [4]. At present, we are involved in incorporating link flexibility into the more sophisticated six-link model of the Canadarm, for the purpose of doing detailed comparisons between DADS' results and those of our simulation.

To conclude, we have employed DADS to model a particular class of mechanical systems—robotic manipulators with rigid and/or flexible links. We have discussed the various elements available in the package for constructing the model of such a system. Also, some of the difficulties encountered in the process of using DADS were noted. A short description of DADS plotting and animation capabilities was given together with out experiences of using them. Comparison of DADS results for a rigid-link manipulator demonstrates good agreement with the results of our own inverse and forward dynamics software. Finally, results of the simulation of a flexible beam with DADS support the previous claim regarding the lack of geometric stiffening terms in the existing multibody computer programs.

References

[1] Haug, Edward J., *Computer Aided Kinematics and Dynamics of Mechanical Systems,* Allyn and Bacon, 1989.

[2] *DADS User's Manual,* Rev. **6.0**, CADSI, July 1989.

[3] *DADS Examples Manual,* Rev. **6.0**, CADSI, July 1989.

[4] Kane, T. R., Ryan, R. R. & Banerjee, A. K. (1987). "Dynamics of a Cantilever Beam Attached to a Moving Base," *J. Guidance, Control, and Dynamics*, **10**, 2, pp. 139-151.

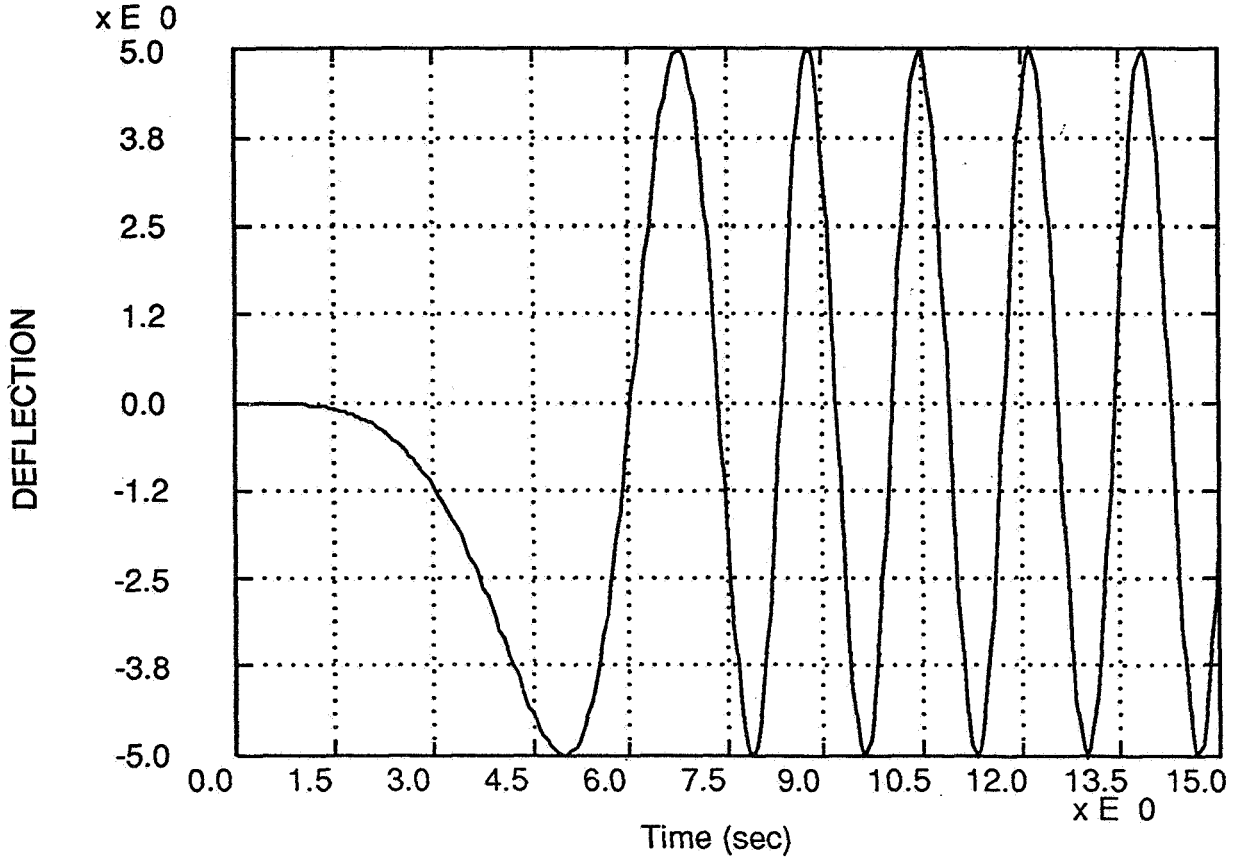


Figure 1: Transverse Deflection (m)

COMPUTATIONAL CONTROLS WORKSTATION: ALGORITHMS AND HARDWARE

R. Venugopal and M. Kumar
Dynacs Engineering Co.

ABSTRACT

The Computational Controls Workstation provides an integrated environment for the modeling, simulation and analysis of Space Station dynamics and control. Using highly efficient computational algorithms combined with a fast parallel processing architecture, the workstation makes real-time simulation of flexible body models of the Space Station possible. A consistent, user-friendly interface and state-of-the-art post-processing options are combined with powerful analysis tools and model databases to provide users with a complete environment for Space Station dynamics and control analysis. The software tools available include a solid modeler, graphical data entry tool, O(n) algorithm-based multi flexible body simulation and 2D/3D post-processors. This paper describes the architecture of the workstation while a companion paper describes performance and user perspectives.

Computational Controls Workstation: Algorithms and Hardware

The Computational Controls Workstation provides an integrated environment for the modeling, simulation and analysis of Space Station dynamics and control. Using highly efficient computational algorithms combined with a fast parallel processing architecture, the workstation makes real-time simulation of flexible body models of the Space Station possible. A consistent, user-friendly interface and state-of-the-art post-processing options are combined with powerful analysis tools and model databases to provide users with a complete environment for Space Station dynamics and control analysis. This paper describes the architecture of the workstation while a companion paper [1] describes performance and user perspectives.

Hardware Architecture

The system hardware is designed around a special-purpose parallel processing architecture based on four Intel i860 processors. The parallel processor communicates with the host system through a standard VME bus interface. All user interaction is handled by the host, a Silicon Graphics Personal Iris workstation. A dedicated graphics engine on the host is used for the animator and solid modeler. Network access (Ethernet) and disk I/O is also handled by the host. The parallel processor is capable of sustaining over 40 double precision MFLOPS when used with the simulation software. Inter-processor communication is accomplished through message passing. This architecture is shown in Figure 1.

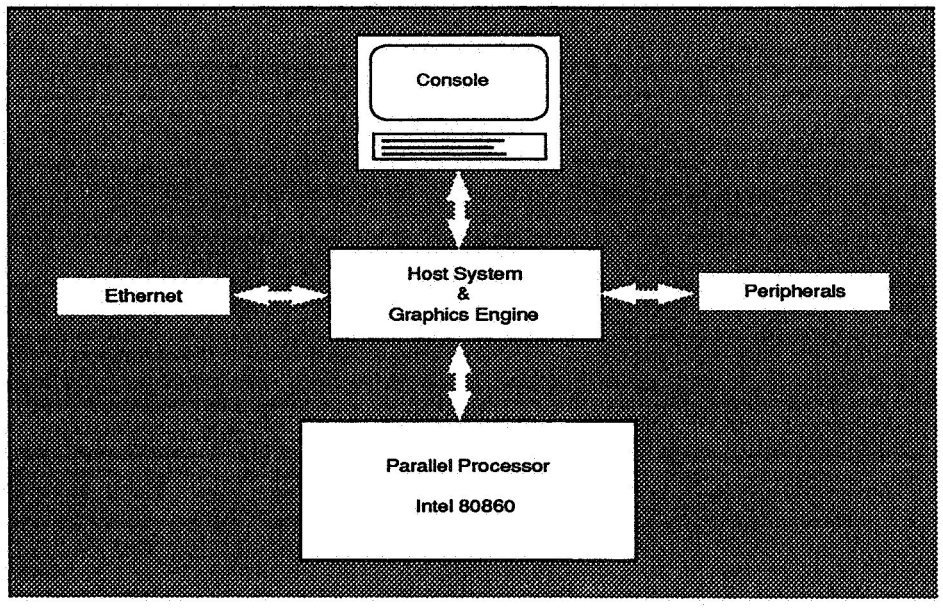


Figure 1: Workstation Hardware Architecture

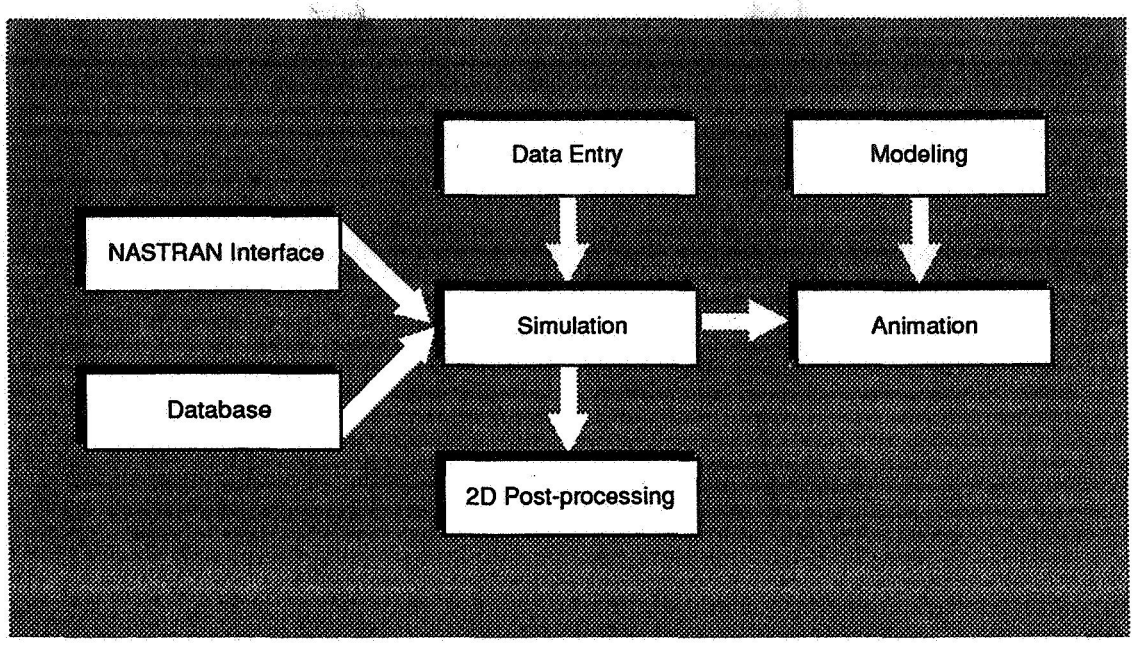


Figure 2: Workstation Software Architecture

Software Tools

The workstation prototype includes a set of integrated software tools that interact with the user through a consistent graphical user interface. They are:

- A window and mouse based interface for configuration data entry and modification
- A 3D solid modeling tool for geometry definition
- A full featured multi flexible body dynamics simulation package
- A 3D graphics animation package for the visualization of simulation results
- A 2D post-processing package for x-y plots
- An interface to NASTRAN for flexible body model data
- A database of typical Space Station configuration models

The software architecture is shown in Figure 2.

Data entry

The window and mouse based interface is built around an interactive screen editor for the creation and modification of configuration data. Up to three different data files may be edited at once, with facilities for copying and moving data elements between them. Data entry and editing is based on the concept of *forms* for each logical component (bodies, joints, sensors etc.) with full screen editing capabilities using the mouse and keyboard. Data is stored in files compatible with other tools that may require it.

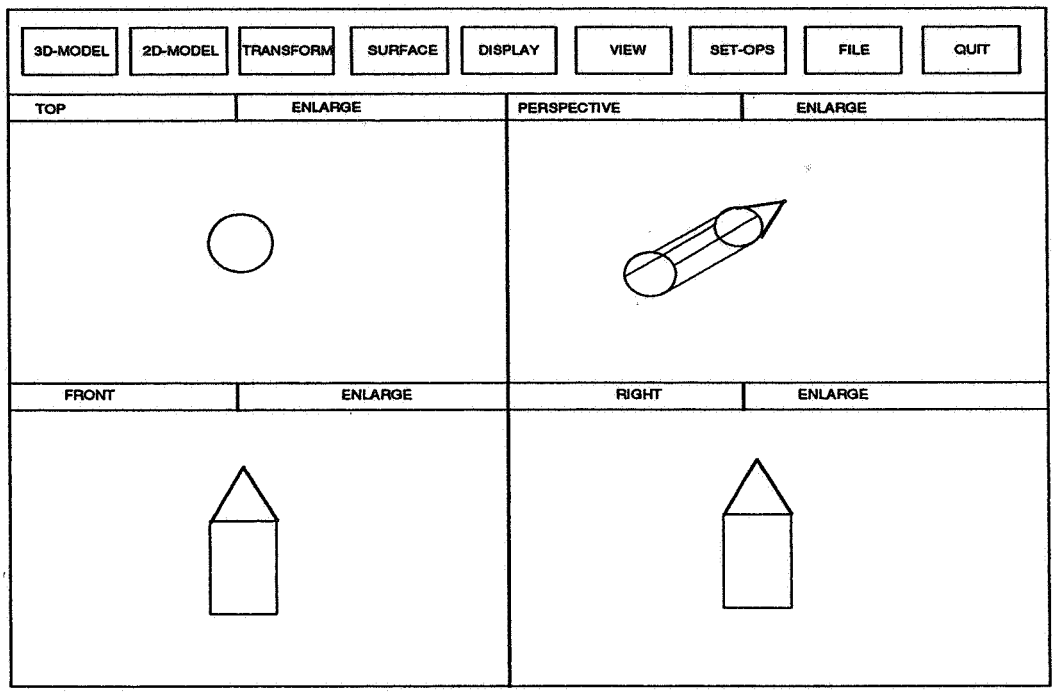


Figure 3: Solid Modeler Interface

Solid modeler

The 3D solid modeler is a simple, intuitive tool for creating, editing and storing graphical models for use in the animation facility. Using simple graphical primitives such as lines, curves, cubes, cylinders etc., the geometry of each sub-structure may be built and stored. Simple transformations (rotation, scaling) as well as complex ones (splining, extrusion, revolution, patching) help the user create highly realistic geometric representations. Wire-frame as well as solid models may be displayed, with special lighting and material effects. Several graphical objects may then be combined to form one simulation object (or body). The user interface to the modeling tool is shown in Figure 3.

Simulation

The multi flexible body simulation package represents a quantum leap in simulation technology. It uses an $O(n)$ algorithm for the solution of the dynamical equations. The equations of motion for the structure are processed by an artificial intelligence based symbolic manipulator that exploits the specific simplifications applicable to each configuration. The equations are then layered, scalarized and further simplified, producing highly optimized source code which is then compiled and linked to produce the simulation module. The simulation includes a set of built-in sensors and actuators as well as an interface for user-defined controller software. Typical RCS and CMG

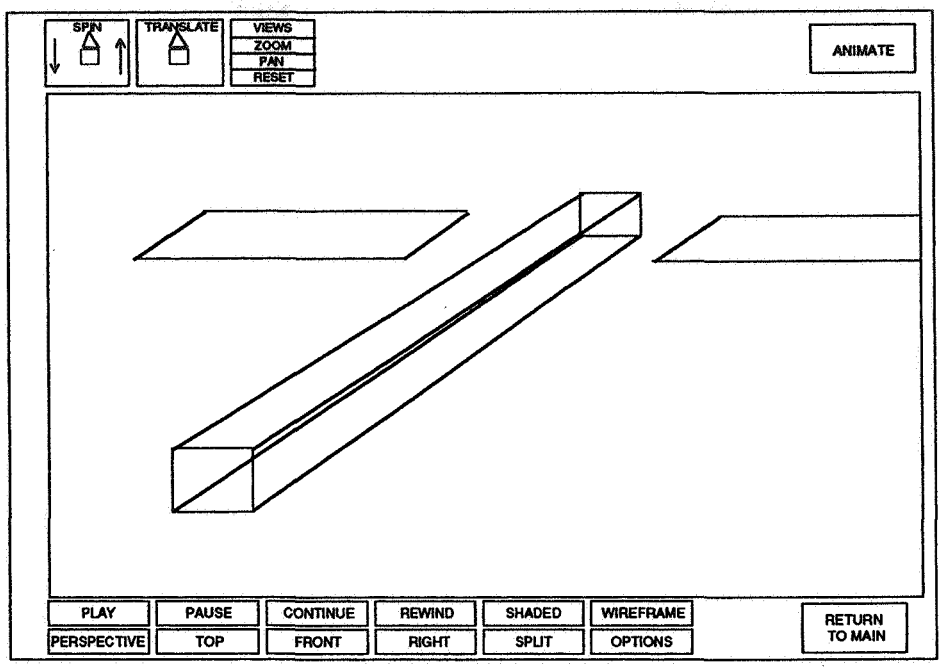


Figure 4: Animator Interface

controllers are included as part of the Space Station configuration database. Disturbances such as gravity gradient and aerodynamic drag are also modeled. The simulation software generated by the symbolic processor also includes automatic parallelization directives that split the simulation core into separate modules for concurrent execution on the four processors contained in the parallel processing hardware. Parallelization and parallel execution proceeds in a manner that is transparent to the user.

Animation

The animation tool combines the geometric data from the modeler with the simulation results to produce a 3D graphics animation of the movement of the multi-body structure. Four different viewing options (top, front right and perspective) are provided, with a choice of wire-frame or shaded drawings. The animation can be played in forward or reverse sequence, or paused at any instant for closer examination using a zoom function. The user interface to the animator is shown in Figure 4.

2D Post-processing

Standard x-y plots of simulation results can be produced using the 2D post-processor. Sophisticated features such as scaling, multiple input plots, multiple frames and phase plane plots can be

invoked. The package can be configured for hardcopies on *Postscript* and *hpgl* devices.

NASTRAN Interface

The NASTRAN interface program reads data from MSC or COSMIC NASTRAN runs and computes the integrals required in the simulation program. Using efficient computational algorithms and direct access files, the program can handle large finite element models without prohibitive memory and computational requirements.

Configuration database

A database of Space Station configurations, starting from the first element launch to assembly complete will be made available. The database will also include the applicable controllers and will be updated as the configurations evolve.

Extensions

A number of extensions and additions to the basic concept are planned. These enhancements will be made on an ongoing basis, and made available to users periodically. The solid modeler and animation facility will be extended for more complex operations. Facilities to import previously defined graphical models in industry-standard formats (IGES, etc.) will be provided. This will allow users to import solid models from several finite element pre-processors. The parallel processor will be enhanced to provide standard language constructs that can then be used to execute user-defined software concurrently.

A host of additional software tools are currently being examined for porting to the workstation. These include finite-element programs, linear analysis tools, linearization tools, model reduction tools and modern control design tools. The new tools will be implemented maintaining the smooth, transparent flow of data between them, an idea that will remain central to the workstation concept.

The Space Station configuration database will also be updated as new designs of the structure and control systems evolve. In addition, other Space Station subsystems such as the Mobile Servicing Center (MSC) may be added, along with their associated control elements. A database of sensors and actuators is also planned, that will allow the user to efficiently compare responses. Finally, the database could include an expert system interface that will provide assistance in areas such as component selection and placement, modeling, model reduction and control design. Additional modeling features are also planned, including models of the orbiter, the shuttle RMS and berthing operations. This will provide the first integrated simulation of the Space Station dynamics, attitude control, berthing and de-berthing and MSC operations.

The workstation concept is not limited to Space Station configurations, and can be used for general multi flexible body modeling, simulation and control design. All the interfaces are designed for maximum flexibility, thus minimizing migration time.

References

1. Roithmayr, C.M., Straube, T.M. and Tave, J.S., "Computational Control Workstation: Algorithms and Hardware", *Proceedings of the 5th Annual Conference on Aerospace Computational Control*, NASA-JPL, Aug. 1992.

Caesy: A Software Tool for Computer-Aided Engineering

Matt Wette
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109
mwette@csi.jpl.nasa.gov

Abstract

A new software tool, Caesy, is described. This tool provides a strongly typed programming environment for research in the development of algorithms and software for computer-aided control system design. A description of the user language and its implementation as they currently stand are presented along with a description of work in progress and areas of future work.

Introduction

Over the past few decades, control system design and analysis has become more and more dependent on computers. With the availability of more powerful hardware has come the demand for more performance. Computer-aided control system design tools such as Matlab have been used with some success in control system design since the early 1980's. However, many workers in the development of software and algorithms for control system design have recognized that these tools have limits in both flexibility and efficiency. The forces driving the development of new tools include the desire to make complex system modeling, design and analysis easier; the need for quicker turnaround time in analysis and design; the desire to make use of advanced computer architectures to help in control system design; the desire to adopt new methodologies in control; and the desire to integrate design processes (e.g., structure, control, optics). We have developed Caesy (Computer-Aided Engineering SYstem) as a means toward discovering how these desires can best be satisfied. The first Matlab-type environment for matrix manipulation, called MATLAB, was developed by Cleve Moler in the late 1970's, mostly at the University of New Mexico, with support from the National Science Foundation. Several other Matlab-based environments have been produced for control system design since then. Included in this group of control system design tools is Ctrl-C from Systems Control Technology, Matrix$_X$ from Integrated Systems, Inc. (ISI), Pro-Matlab from the MathWorks, Inc., Mat/C developed at Lawrence Livermore National Laboratory, and SFPACK developed at the University of Waterloo. In this document we will collectively refer to these MATLAB-based environments as Matlab. A new-generation Matlab package has been released from ISI. This package, called Xmath, seems to provide some features for easier use but does not offer all the features we feel are desirable. The success of Matlab as an environment for the design and application of algorithms for control system design implies that a new tool geared toward large order, complex problems should include capabilities provided by Matlab and attempt to maintain in some sense the "flavor" of Matlab.

In the development of Caesy, our goal has been to provide a more advanced environment which can provide the capabilities provided by these packages, but can also provide alternatives to better handle the problems encountered when problems become complex or the system order creates computational bottlenecks. More discussion on the requirements driving the development of Caesy can be found in [1]. Our approach to solving the complexity problem will be to develop an "object-oriented" user interface (e.g., typed variables and overloaded functions and operators). Our approach to the computational problems will be to take advantage of this interface to develop specialized software which can take advantage of any special structure in the models (e.g., symmetry, bandedness, sparseness).

In this paper we will provide an overview of the current capabilities and implementation of Caesy, the work currently in progress, and work planned for the future.

Current Implementation of Caesy

In this section, we give an overview of Caesy and some of its features. Caesy is, in a concise description, a mix between Matlab and Ada. Caesy contains many of the constructs of Ada but adds features from Matlab and whatever other modifications we felt were needed to provide the required functionality. Ada was chosen as a primary influence because it provides many of the features required and has a procedural syntax familiar to many Matlab users. In addition, the syntax used in the Caesy language is close to that being encouraged as an IFAC and IEEE standard (see [2]).

Tidbits

Caesy is an interactive shell and thus processes user input on a command by command basis. The prompt 'caesy>' shown below implies that the input is at the top level of the shell. Caesy is not case sensitive, so 'abc', 'ABC', 'AbC' and 'abC' are all equivalent. Internally, Caesy treats all symbols as lowercase. Imaginary constants, used heavily in control system analysis, are input using a real constant with a trailing 'i', 'I', 'j' or 'J' (e.g., 1.0i, 1.0J).

Types in Caesy

Probably the most striking difference between Caesy and other Matlab environments is the fact that Caesy is a typed language. That is, in Caesy all variables have type (e.g., integer, real, string) whereas in Matlab all variables typically have no type (i.e., all variables are the same type, Matrix). The addition of types to a language adds possibilities for making more powerful tools, but may place a burden on the user for declaring types.

In Caesy, we use types but put minimal requirements on users to declare variable types. The only place one is strictly required to declare variable types is when they appear as formal arguments in function and procedure declarations. Otherwise, the type of a new variable (implied by an assignment statement) can be synthesized from the type of the right hand side expression in the assignment. For example, in the statement

```
caesy> abc := 3 + 4;
```

the right hand side has type **integer** so if **abc** was not previously declared, Caesy automatically declares the variable **abc** and gives it type **integer**.

Explicit type conversion in Caesy is performed, by convention, by declaring a function with the type name. For example, a function to explicitly convert integers to real is declared as

```
function real(i: integer) return x: real;
```

and then used, for example, as

```
x := sqrt(real(2));
```

Statements

Caesy supports many of the statements and control structures found in Ada with the multiple assignment form taken from Matlab.

Assignment Statements

Assignment statements take the normal form as well as the multiple assignment form familiar to Matlab users. Here are examples of the two forms:

```
caesy> abc := 3 + 4;
caesy> { x, y, z } := fctn(abc, 3.0);
```

One of the statement terminators ';' or '?' is required in Caesy. Just hitting the return key will not do. The '?' implies that the result of the assignment should be displayed to the user:

```
caesy> abc := 3 + 4?
  abc := 7;
```

If an expression *expr* appears alone on the command line, an implied assignment of the form

'_ans_*type-name* := *expr*?'

is interpreted. For example:

```
caesy> 3 + 4;
  _ans_integer := 7;
```

Flow Control Statements

Control constructs allow the programmer to do branching and looping. In Caesy, the if-then branching is similar to many languages. An example is

```
if i < 0 then
  j := 1;
elsif i = 0 then
  j := 0;
else
  j := 1;
end if;
```

Looping in Caesy is reasonably flexible. There are for-loops, while-loops, etc. Some examples are

```
for i in 1..3 loop j := 3*j; end loop;
while j < 30 loop j := 3*j; end loop;
loop j := 3*j; exit when j >= 30; end loop;
```

User-Defined Subprograms

Subprograms in Caesy take the form of functions and procedures. (Matlab has no procedures). The following is an example of a user-defined function definition.

```
function myabs(i: integer) return j: integer is
begin
 if i < 0 then j:= -i; else j:= i; end if;
 return;
end myabs;
```

An example of a procedure definition is the following:

```
procedure swap(x, y: in out real) is
  temp: real; -- declaration optional!
begin
  temp := x; x := y; y := temp;
  return;
end swap;
```

Overloading Functions

One important feature needed in design interfaces for handling complexity is overloading functions and operators. Overloading allows a user to write functions of the same name and purpose to operate on different object types. For example, we could define a function **myabs** similar to the one above, but which operates on reals.

```
function myabs(x: real) return y: real is
begin
 if x < 0 then y := -x; else y := x; end if;
 return;
end myabs;
```

Function overloading is very handy in control system analysis. For example, one could use the expression `freq(G)` to produce a frequency response plot whether `G` represents a transfer function in state-space form, rational form, or anything else. In each case, a different function would be called depending on the argument type.

Caesy does not overload the return argument(s) of a function as Ada does. This is difficult to implement, would prohibit the ability to synthesize expression types and hence would require users to type-declare *all* variables (see above).

Overloading Operators

Caesy also provides the capability to overload most operators. Operator overloading allows users to overload operations such as '`A+B`' where `A` and `B` have user-defined types. For example, given two transfer functions `G1` and `G2` in state-space form or rational form, we could overload the operator '+' to compute a transfer function in state space form for the parallel connection of two transfer functions. The declarations of these functions in Caesy would look like

```
function "+"(G1: st_sp; G2: st_sp)
    return G: st_sp;
function "+"(G1: rat; G2: rat)
    return G: st_sp;
function "+"(G1: st_sp; G2: rat)
    return G: st_sp;
function "+"(G1: rat; G2: st_sp)
    return G: st_sp;
```

If `G1` and `G2` were two transfer functions in either state space (`st_sp`) form or rational (`rat`) form, the expression `G1+G2` would produce a state space representation of the transfer function. An alternative to the expression `G1+G2` is the explicit function call `"+"(G1,G2)`.

Default Input Arguments

Users of Matlab are accustomed to a variable number of input arguments. In the case where input arguments are not given, default arguments must be supplied in the subprogram declaration. In Caesy, a user need not specify all input arguments if default arguments are given in the subprogram declaration. For example, the following function declaration fragment contains a default input argument:

```
function abc(x: real; y: real := ydef)
```

The function could be referenced, for example, as `abc(x1, y1)` or as `abc(x1)`. In the latter case, the value of the global variable `ydef` would be used for the second argument. In Matlab, unspecified inputs are handled within the function body. In Caesy, the user can change the default input variable, and hence the behavior of the function, at any time.

Variable Number of Output Arguments

Caesy supports definition of functions with more than one output argument. In this case, if the function is used in a simple expression, only the first argument is referenced. In the case of a multiple-return assignment (see above), one or more return arguments may be referenced. In the function body, usage of return arguments can be determined via the '`argused`' operator. For example, consider the following function which echoes its input arguments:

```
function echo(i1: integer; i2: integer)
    return { j1: integer; j2: integer } is
begin
  j1 := i1;
  if argused j2 then j2 := i2; end if;
  return;
end echo;
```

Procedures

In Caesy, users may define functions and procedures. Caesy was provided with the ability to define procedures for reasons of efficiency. Procedures allow one to modify variables without creating a duplicate temporary variable. For example, suppose one wanted to write a function to perform a rank-one change on a matrix. This could be done with a function call of the form

```
A = rankfctn(A, x);
```

or with the procedure

```
rankproc(A, x);
```

The difference here is that the function `rankfctn` will create a copy of the matrix `A` to modify and return, and then `A` will be replaced with this matrix. The procedure will never create the copy of the matrix `A`. If the matrix happened to be large, the execution speed between the functional form and procedural form could be quite noticeable.

Packages

Several Matlab-type packages use the convention of collecting related script files together to form "toolboxes." Caesy formalizes this convention by using the Ada "package" construct. In Caesy, type definitions, functions, procedures and global variables can be collected together in packages. In Matlab, there is no clean way to distinguish between two different functions of the same name in two different toolboxes. This can lead to severe problems and requires that users and toolbox developers take care to avoid the "name-clash" problem. In Caesy, this problem can be avoided easily.

If a user wishes to explicitly refer to an object in one package when there may be a conflict, the user affixes the object name to the specific package name (a concept borrowed from Ada). For example, to explicitly reference the function **xyz** from the package **mypckg**, one would use the construct **mypckg.xyz(*args*)**.

Matrix Expressions

The ability to create and manipulate matrices is of fundamental importance in control system engineering. In Caesy, matrix expressions, those expressions used for constructing matrices from their parts, are supported in a unique way. An example of a matrix expression is the following:

```
A := [ 1.1, 1.2; 2.1, 2.2];
```

This is a two-by-two real matrix. In Caesy, the type for this matrix is '**ReGeMat**', for *Real General Matrix*. Complex matrices have type '**CoGeMat**'.

```
C := [ 1.0 + 0.1i, 1.2; 2.1i, 2.2];
```

In the future, we plan to have special support for symmetric and Hermitian matrices, sparse matrices, etc. In Caesy, matrices are constructed using overloaded operators. The above expression for defining '**A**' gets internally translated, in essence, to the following sequence of calls:

```
T0v1 := "["(1.1);
T0v2 := ","(T0v1, 1.2);
T0v3 := ";"(T0v2, 2.1);
T0v4 := ","(T0v3, 2.2);
A := "]"(T0v4, 2.2);
```

The variables starting with **T** are temporary variables created by Caesy. When Caesy sees the first '**1.1**' it looks for the function "[" which takes a real argument. (There are also "[" functions defined which take an integer or a complex value as the first argument.) In the Caesy **MATMATH** package, this function is defined, and returns a special type, **regematlist**, for matrix expressions. When the token '**1.2**' is seen, Caesy looks for a function "," which takes arguments of type **regematlist** and **real**. The process continues until the function "]" taking a **regematlist** and returning a **regemat** is called. By implementing matrix expressions with overloaded operators, we have made the matrix expression construct potentially usable in very creative ways.

Support of Constructors and Destructors

In Caesy, if a procedure of the form

```
procedure constructor(vble: in out Type)
```

exists for some variable of type **ObjType**, then that procedure will be called automatically when the variable comes into scope. Additionally, if a procedure of the form

```
procedure destructor(vble: in out ObjType)
```

exists for some variable of type **ObjType**, then that procedure will be called automatically when the variable comes into scope.

Implementation

Caesy is written primarily in C. The internal structure is shown in Figure 1. It includes a supervisor to prompt for input, a parser, written in YACC, a byte-code compiler, a byte-code interpreter, a C code compiler, a context (variable, types, etc.) handling module (made of CX/Sys-Ctxt blocks in the figure), an input output interface, and more. The parser outputs LTU (language transfer utility) codes which can be compiled into byte-codes or C code.

Work in Progress

In this section we discuss features which are partially implemented or are in the process of being implemented.

User Defined Structured Data Types

Caesy will have the capability for the user to define structured (i.e., record) data types. This is an essential feature needed in the reduction of complexity.

Exception Handling

Currently, Caesy has some support of exception handling. Internally, every Caesy function returns an exception code to its caller. If the function executes normally, it returns the exception code **NO_ERROR**. On the other hand if some anomaly occurs, a different exception will be raised. For example, if one tries to take the square root of a negative (real) number, a **NUMERIC_ERROR** exception will be raised. Caesy will trap machine exceptions. If a divide-by-zero occurs, a **NUMERIC_ERROR** will be raised. The exception will be passed until it gets to the supervisor level. For example,

```
caesy> a := sqrt(-9.0);
Supervisor caught NUMERIC_ERROR exception.
caesy>
```

This capability will be extended to allow exceptions to be handled in user-defined functions.

Data Files

Users often have the desire to store and retrieve data. In Caesy, data files will have special binary formats and store data in a machine-independent form. The data files will support user-defined types and hence, must be quite flexible. The stored data files will make use of the Sun XDR (external data representation) [3].

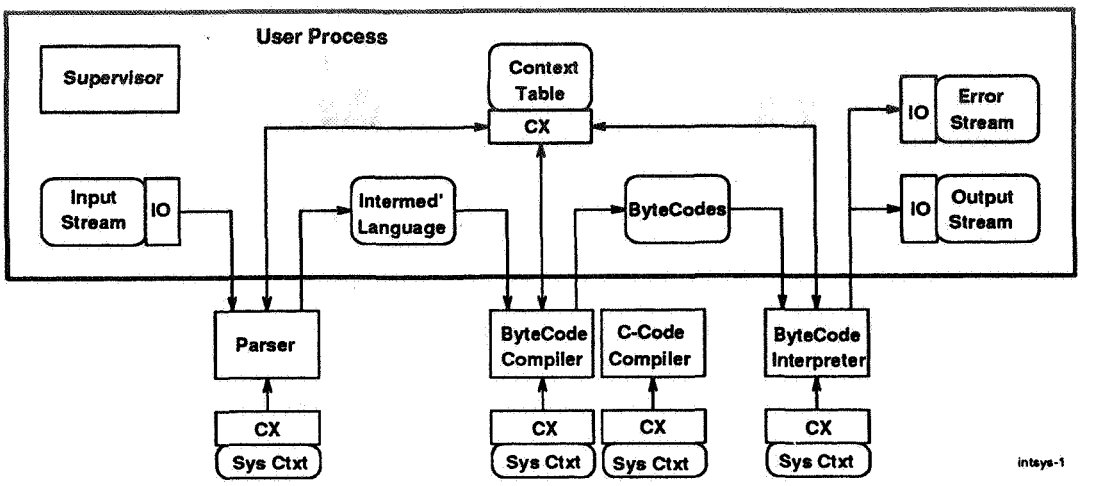


Figure 1: Implementation of Caesy

Code Generation

Caesy allows users to generate C code from their subprogram scripts. For example, for the following code segment, Caesy will produce the C code shown in Figure 2.

```
use matmath;
function test(A: regemat; B: regemat)
    return C: regemat is
begin
  for i in 1..10 loop
    C := A*B - B*A;
  end loop;
  C := [ C ; [ A, B ] ];
  return;
end test;
```

The C coding capability also makes the prospect of developing stand-alone programs attractive. An often-used application could be converted to C-code and compiled to run as a specialized program, making it more efficient. This feature also has the potential of allowing users to change a routine to make it as efficient as possible. In all, this is a very desirable feature.

Remote Computing

There is currently work in progress by another team at JPL; the team is implementing a remote computing capability in Caesy. This is being developed using Sun RPC calls [3].

Matrix Computations

Many of the matrix computations in Caesy are performed using BLAS and LAPACK [4]. These are state-of-the-art FORTRAN libraries in computing solutions to linear equations and computing eigenvalue decompositions. In fact, Caesy could be viewed as a user-friendly interface to LAPACK and, with Caesy's C code compiler, Caesy may be an attractive way to develop code based on both these routines.

Future Work

In this section we discuss features which are under consideration for future implementations. One object of the Caesy project is to determine what features in an interactive shell are needed for handling large, complex problems.

Function and Operator "Tagging"

The semantics of creating matrices using the '[', ',', ';' and ']' operators have a drawback. It may be nice to be able to use the construct to input, for example, sparse matrices. A sparse matrix could be input as

```
Asp := [ 1,1,1.1;  2,1,2.1 ];
```

where `1, 1, 1.1` indicates the element in row 1, column 1, is `1.1` and so on. This type of flexibility cannot be supported by the current semantics of the language. There is no way that Caesy knows that a sparse matrix definition is intended. One work-around could be to "tag" the first value with an explicit type cast:

```
Asp := [ spmatelt(1,1,1.1);  2,1,2.1 ];
```

Here we have a special "sparse matrix element" type that tags the first element to allow Caesy to find the function "[", which takes a sparse matrix element type.

Another work-around, which would provide more flexibility to the Caesy language overall, would be to add *tagged* functions and operators to Caesy. In this option, function and operators could have tags to indicate that special operators should be used. The tag would take the special form *function:tag* in both its declaration and use. The tagging approach would allow the user to use the following to generate a sparse matrix (of type `ReSpMat`):

```
Asp := [:sp 1 1, 1.1;  2, 1, 2.1 ];
```

The declaration for the operator [may have been

```
#include "kc.h"

kcXcode
_U1_1test(
     MATMATH_regemat* a,
     MATMATH_regemat* b,
     MATMATH_regemat* c)
{
  MATMATH_regemat T1v0;
  MATMATH_regemlist* T1v1;
  MATMATH_regemlist T1v2;
  MATMATH_regemat T1v3;
  MATMATH_regemlist* T1v4;
  MATMATH_regemlist T1v5;
  {
    int i;
    MATMATH_regemat T2v0;
    MATMATH_regemat T2v1;
    MATMATH_regemat T2v2;
    {
      i = 1;
      for (;;)
        {
          if (i>10) break;
          {
            MATMATH_1MUL(a, b, &T2v1);
            MATMATH_1MUL(b, a, &T2v2);
            MATMATH_1SUB(&T2v1, &T2v2, &T2v0);
            MATMATH_1ASSN(c, &T2v0);
          }
          i = i + 1;
        }
    }
    MATMATH_1destructor(&T2v2);
    MATMATH_1destructor(&T2v1);
    MATMATH_1destructor(&T2v0);
  }
  MATMATH_2MST(c, &T1v2);
  MATMATH_2MST(a, &T1v5);
  MATMATH_2MEL(&T1v5, b, &T1v4);
  MATMATH_1MND(T1v4, &T1v3);
  MATMATH_2MRO(&T1v2, &T1v3, &T1v1);
  MATMATH_1MND(T1v1, &T1v0);
  MATMATH_1ASSN(c, &T1v0);

  MATMATH_3destructor(&T1v5);
  MATMATH_1destructor(&T1v3);
  MATMATH_3destructor(&T1v2);
  MATMATH_1destructor(&T1v0);

  return(kc_X_NO_ERROR);
}
```

Figure 2: Generated C Code

```
function "[":sp(i: integer)
    return list: respmatlist;
```

Parallel Computation

Since Caesy has been developed primarily to explore ways in which to increase computational throughput for control design tools, it is only natural to pursue the possibilities of parallel computation. Several workstations and hardware co-processor boards with multiple processors are currently available on the market. How to best support these multiprocessor environments will most likely depend on which machine and operating system we run under. Currently, Mach derivatives and Solaris have support for *multi-threaded* programming at the C language level.

Conclusion

In this report we have provided a glimpse of a new software environment, Caesy, which will be used for the development of algorithms and software for the design of large, complex control systems. The tool provides a "next-generation" approach to control system design by taking advantage of some concepts from object-oriented programming languages. The tool should prove to provide a means for easily handling large, complex design problems. It is also being used as a "computational engine" in a tool for integrated (conceptual) design of advanced optical systems.

Acknowledgment

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

References

[1] M. Wette, "Caesy: A computer-aided engineering system," in *1992 IEEE Symposium on Computer-Aided Control System Design* (Napa, CA), pp. 232–237, March 17-19, 1992.

[2] M. Rimvall, "Command language standard for cacsd software." Control Systems Laboratory, GE-CRD, Schenectady, NY 12301, 1989.

[3] J. R. Corbin, *The Art of Distributed Applications.* New York: Springer-Verlag, 1990.

[4] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide.* SIAM, 1992.

ASTEC and MODEL: Controls Software Development at Goddard Space Flight Center

John P. Downing , Goddard Space Flight Center
Frank H. Bauer, Goddard Space Flight Center
and
Jeffrey L. Surber, Fairchild Space Company

Abstract

The ASTEC (Analysis and Simulation Tools for Engineering Controls) software is under development at the Goddard Space Flight Center (GSFC). The design goal is to provide a wide selection of controls analysis tools at the personal computer level, as well as the capability to upload compute-intensive jobs to a mainframe or super computer. In the last three years the ASTEC (Analysis and Simulation Tools for Engineering Controls) software has been under development. ASTEC is meant to be an integrated collection of controls analysis tools for use at the desktop level. MODEL (Multi-Optimal Differential Equation Language) is a translator that converts programs written in the MODEL language to FORTRAN. An upgraded version of the MODEL program will be merged into ASTEC. MODEL has not been modified since 1981 and has not kept pace with changes in computers or user interface techniques. This paper describes the changes made to MODEL in order to make it useful in the 90's, and how it relates to ASTEC.

Introduction

Several programs have been developed at NASA's Goddard Space Flight Center (GSFC) in recent years. These include the Interactive Controls Analysis (INCA) program [1] starting in 1981, and the Windowed Observation of Relative Motion (WORM) program [2] starting in 1986. An important earlier effort is MODEL (Multi-Optimal Differential Equation Language) [3] developed in the 1960's and 1970's. In the last three years the ASTEC (Analysis and Simulation Tools for Engineering Controls) [4]software has been under development. ASTEC is planned to be an integrated collection of controls analysis tools for use at the desktop level. Planned conversions of INCA and WORM to PC/Macintosh programs will be part of the ASTEC system. MODEL is a translator that converts programs written in the MODEL language to FORTRAN. An upgraded version of the MODEL program will be merged into ASTEC. MODEL has not been modified since 1981 and has not kept pace with changes in computers or user interface techniques. This paper describes the changes made to MODEL in order to make it useful in the 90's, and how it relates to ASTEC.

ASTEC

ASTEC is being written to satisfy the requirements of the GSFC Guidance and Control

Branch. As such, it must run on the computer equipment used in the branch. Currently desktop units consist of PC's, Macintoshes, and an occasional Tektronix or X-Windows terminal. Mainframe capabilities consist of VAX 8830 and IBM RS6000 computers.

ASTEC is designed to meet the continuing needs of GSFC engineers, where high order and complex systems are the rule and not the exception, and where tried and true classical methods predominate. Because spacecraft repair is very expensive if not impossible, it is important that analysis methods be exhaustive rather than quick, and that algorithms contain no shortcuts which may compromise analysis results. There is also a high demand for a modern, friendly user interface, since many of the engineers use the Macintosh or Microsoft Windows environment.

It has long been planned to port INCA and WORM from the VAX/VMS to a desktop computer. Since PC's and Macintoshes predominate in our branch, they were chosen despite relatively poor performance in floating point operations. It is hoped that there will be performance improvements in the future and that some computations can be done on the VAX and the results downloaded later.

ASTEC Architecture

ASTEC will consist of several modules. Many of the currently implemented or planned modules are described below. The capabilities of ASTEC include classical control methods, simulation both linear and non-linear, multi-variable controls and matrix methods, and new experimental capabilities -- including dynamic locus and three dimensional frequency response. The following modules are under some state of development, and more may be added. It is hoped that by the time of this conference MODEL will be available from COSMIC in VAX, PC, and Macintosh versions, and that WORM will be available in a PC version. Note that the old VAX versions of WORM and MODEL have been available for some time.

ASTEC	Manage files and launch other jobs.
(PUEBLO?)	Block diagram editing (Prototyping Utilities Emphasis is Block LayOut).
MODEL	Build systems and launch analyses.
INCA	Transfer function and state space analysis.
WORM	Plot results.

Input to some of the ASTEC modules can come from either the user or, more importantly, from other modules. Thus, for example, as user could build a block diagram model of his system using the PUEBLO program. ASTEC could use this data to generated a simulation in the MODEL language, which could be translated, compiled, linked, and executed. The results could be plotted by the WORM package.

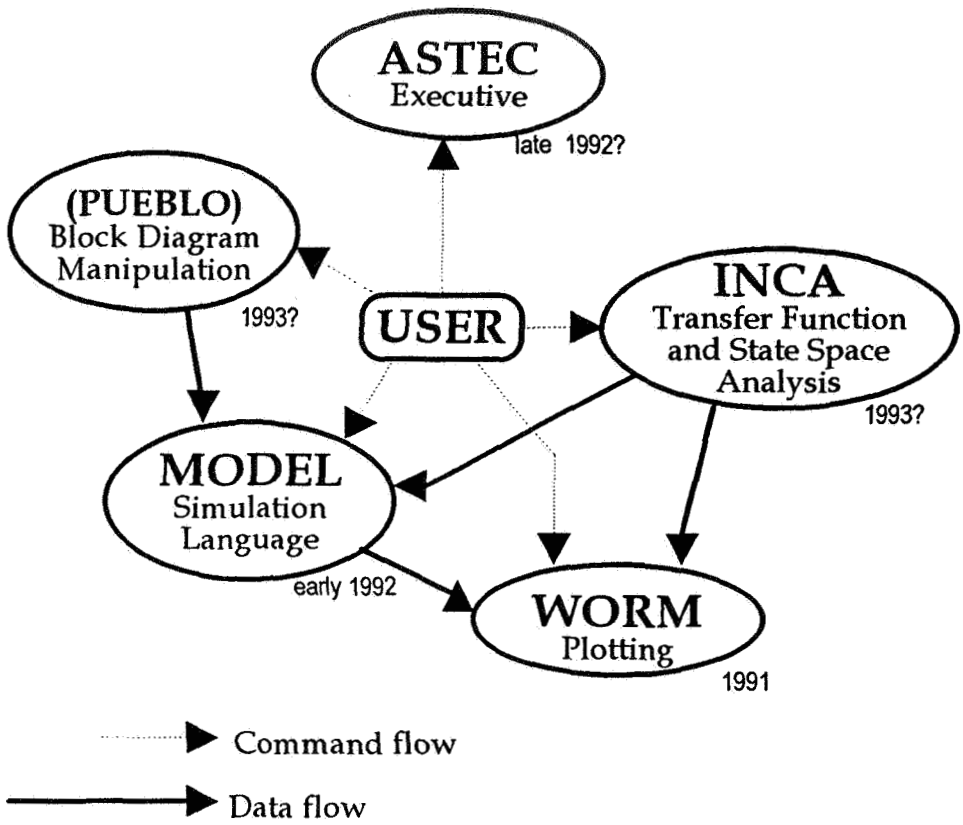


Figure 1. General ASTEC Architecture

While personal computers are quite good in the fields of graphics and user interface, they often fall short in the field of number crunching, especially if hardware floating point is not installed. For this reason a capability to transfer compute intensive jobs to a mainframe computer was deemed essential. Compute-intensive routines (such as MODEL) will be capable of dealing with text files only, allowing input and output data to be transferred between computers.

MODEL

The Multi-Optimal Differential Equation Language provides a means for generating numerical solutions to systems of differential equations using a digital computer. The notation of this language is similar to that used to describe physical systems by differential equations. Thus the learning process is simplified, programming becomes easier, and debugging is more readily accomplished. Programs written in the MODEL language are machine translated into FORTRAN-77 programs.

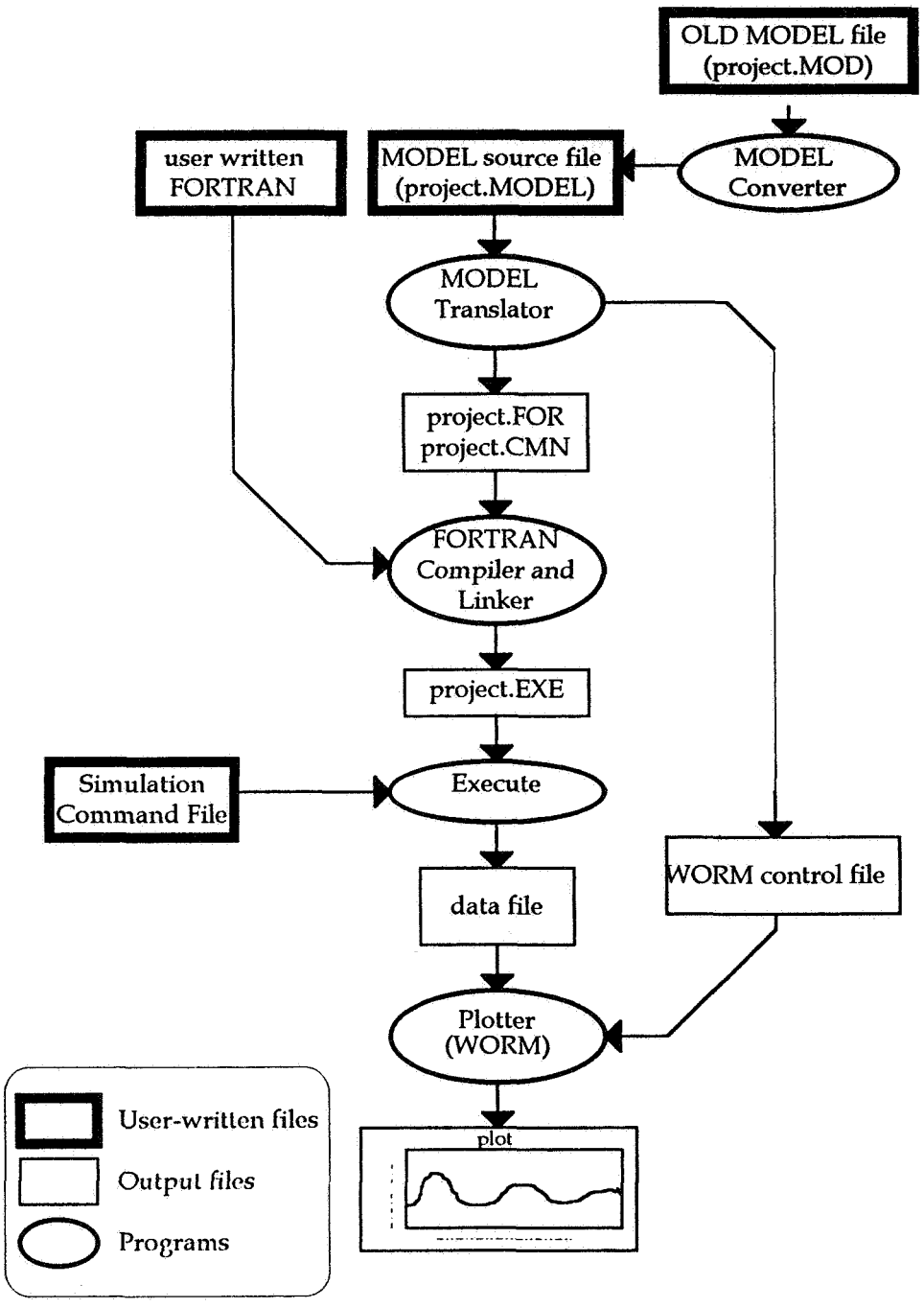


Figure 2. MODEL data flow diagram

MODEL is currently implemented on the VAX computer using VMS, on PC's under Microsoft Windows, and on the Macintosh. The VAX version is capable of automatically generating source files for the WORM plotting program. This feature will allow users to plot their data using the names assigned in MODEL.

Since the MODEL program is a translator, an additional translate step is added to the normal compile/link/run sequence. A data flow diagram is shown in Figure 2.

Language Features

A MODEL program is composed of Model statements. The basic MODEL statement is a differential equation. Equations can be entered in any order. The quotation mark (') is used to indicate a derivative, allowing the equations to be entered in a reasonably familiar way. Variables with quotation marks are derivatives of state variables. State variables can also be derivatives of other state variables. In this case multiple quotations marks are used.

Other MODEL statements include DEFINE statements to control the simulation, OPERATOR and FUNCTION statements to create an interface to user written subroutines, and comments to allow user documentation. MODEL uses a free-form line format. Multiple statements on one line are separated by semicolons (;). A statement may be continued to the next line by using ellipses (...). Two minus signs together indicate a comment--The rest of that line is ignored.

An simple MODEL program is the damped harmonic oscillator:

```
x"=z*x'+(k*x)
x(0)=10
x'(0)=0
z=-0.1
k =-0.2
WRITE (T,X",X',X)
DT=0.01; TFIN = 40.0 -- Time Step, Finish time
DEFINE FILE WRITE 0.1 ASCII TEST.OUT F14.6
```

Variables and Operators

MODEL variables must be one of seven types.

SCALARS : a single floating point number.
VECTORS : position, velocity, force, torque, magnetic fields, etc.
TENSORS : either a rotation matrix or Inertia tensors.

QUATERNIONS : are used to represent rotations.
CHARACTER STRINGS : are used to access external filenames.
MATRICES : an array of scalars, arranged in rows and columns.
SUBSCRIPT RANGES : used to create slices of matrices.

There is a simple relationship between the original program variables and the corresponding FORTRAN variables. MODEL variables are first truncated to 28 significant characters. If the variable is a state variable or derivative, an underline is appended, and then one or more 'P's to represent the order of the derivative.

x x
x' x_p
x'' x_pp
x'(IC) x_pic
x''' x_ppp
t(0) t_ic

Variables may be manipulated by using operators. Operators may be unary or binary, and unary operators may be prefix or postfix. Each operator is given a priority. In complicated expressions the rules of precedence clarify the order in which operations are performed. Operations with equal precedence are performed from left to right. Expressions within parentheses are evaluated first and independently of preceding or succeeding operators. The operators in model are grouped in order of precedence, and are listed below. Note that certain operand types may be incompatible with certain operators.

DEGREES or RADIANS	Convert scalar to angle.
ARCMINS or ARCSECS	Convert scalar to angle.
.(period)	Vector element access.
^ or ** or ^- or **-	Exponentiation.
*	Multiplication.
/ or \	Division and left division.
DOT or *	Vector dot product.
CROSS or ><	Vector cross product.
+	Addition
-	Subtraction or negation.
||	Matrix column concatenation.
//	Matrix row concatenation operator.
==	Equivalence.
<> or != or ~= or |=	Non-equivalence
<	Less than.
>	Greater than.
<=	Less than or equal.
>=	Greater than or equal.
NOT or ~	Logical negation or inverse.

AND or &	Logical and.
OR or |	Logical or.
:(colon)	Matrix subscript ranging operator.
[]	Matrix indexing operator.

MODEL is equipped with built-in functions to support many function and non-linearities required for ease in simulation. Many of these will be familiar to users of FORTRAN, Pascal, or other programming languages. The basic trigonometric functions **SIN**, **COS**, and **TAN** are also available. These take an argument which MUST be of angle type, and return a scalar. Inverse trigonometric functions **ASIN**, **ACOS** and **ATAN** take a scalar argument and return an angle.

Other functions are used to represent various operations that are used in simulations. The **RANDOM** and **RANDOM12** generate random numbers. The **IF** function allows conditional assignment, like the C language ? operator. The **QUANT** function is used to implement quantization functions. The **LIMIT** function is used to implement limiters or limit functions. Additional functions such as **DEADZONE, BANGBANG, HYSTERESIS, BACKLASH** and **TRACKSTORE** are also available.

Other Features

The MODEL preprocessor is similar to the one in the C language. The #FOR statement is followed by a list of character strings. Each line after a #FOR statement is scanned for the at-sign (@), and if one is found, all at-signs in that line are replaced in turn by each character string. Lines without any at-signs are left alone. This process continues until a #ENDFOR statement is encountered. The INCLUDE statement is used to merge text from another file. Using the SYNTAX commands, the user can use his own routines in FORTRAN or other languages.

Run-time Command Language

The run-time command file is read by the generated simulation program to control the simulation. There are four types of statements in the run-time command language. The details of using a command file are implementation dependent. A simple command file is given below:

```
RESET
tfin = 60
RUN
PAUSE AT 30.0
k = 0.3
CONTINUE
STOP
```

The **RESET** statement returns all variables to their original values. The second line is a

variable change command. The format consists of a variable name and one or more values. The RUN command starts the simulation. The first line after the RUN command is a either the keyword **STOP** or a **PAUSE AT** command. If it is a PAUSE AT [time] command, it is followed a list of variable change commands to be given at that time. This allows the user to change parameters in the middle of the simulation.

Changes from first version of MODEL

For those familiar with the initial version of MODEL developed by Benjamin Zimmerman, the following describes changes made in the new version. These include:

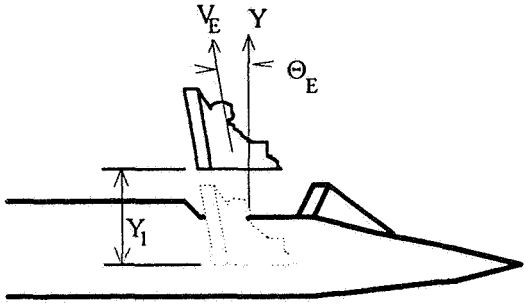
User defined functions and subroutines are now available.
Certain obscure relational operator definitions have been dropped. These are .EQ., .LT., .GT., .LE., .GE., .NE., */, /*, =/, =<, /=, =>, *=, and ><.
The IF statement has been changed to a function.
Elimination of conditional output.
The double comma (,,) may no longer be used to separate statements. Use the semicolon (;) instead.
New data types vector, tensor, quaternion character strings, matrices and subscript ranges. The old type is now called a scalar, and is now the default.
New mathematical operators have been added. These are ^, **-, ^-, \, MOD, DOT, CROSS, ||, //, ~=, !=, |=, ~, |, :, [] and others. Note that the colon has changed meaning from exponentiation to matrix ranging.
Multiple PLOT statements.
Automatic generation of WORM source files.
Several commands and that used to be abbreviated are now spelled out in full.
The END statement is no longer required.
The TAB statement and sampled variables not supported in the initial release.

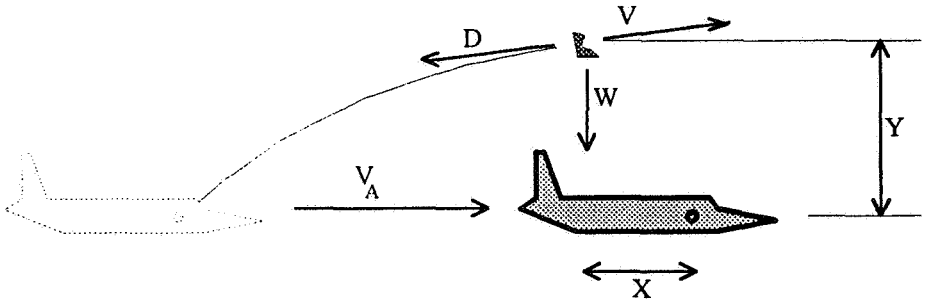
Example: Pilot Ejection Study

This study has been used as a standard of comparison for continuous simulation languages. This example is taken almost verbatim from the manual for the original Model program.

The purpose of this investigation is to determine the trajectory of a pilot ejected from a fighter aircraft and thus to ascertain whether he will strike the vertical stabilizer of the aircraft. Several combinations of aircraft speed and altitude are investigated since the drag on the pilot, which causes his relative horizontal motion with respect to the aircraft, is a function of both air density and velocity. The ejection system is so devised the pilot and his seat to travel along rails at a specified velocity V_E, at angle Q_E backward from vertical. The seat becomes disengaged from the rails at $Y = Y_1$.

Once the pilot and seat combination leaves the rails, it follows a ballistic trajectory which can be determined; however, since it is the relative motion of the pilot with respect to the aircraft (which is assumed to fly level with constant speed) that is important, we can formulate the equations so as to obtain this motion directly.





The governing equations are:

$X' = V \cos \Theta - V_A$
$Y' = V \sin \Theta$
$V' = 0$ $\quad 0 \ Y \ Y_1$
$V' = -D / m - g \sin \Theta$ $\quad Y > Y_1$
$Q' = 0$ $\quad 0 \ Y \ Y_1$
$Q' = -(g \sin \Theta) / V$ $\quad Y > Y_1$
$D' = \frac{1}{2} r C_D S V^2$

Constants (for all cases)

$m = 7$ slugs
$g = 32.2$ ft/sec^2
$C_D = 1$
$S = 10$ ft^2
$Y1 = 4$ ft

VE = 40 ft/sec
QE = 15

The initial values of V and Q (pilot's initial velocity vector at moment of leaving cockpit rails) are given by

$$V_0 = [(V_A - V_E \sin \Theta_E)^2 + (V_E \cos \Theta_E)^2]^{1/2}$$
$$Q_0 = \tan^{-1} [(V_E \cos \Theta_E) / (V_A - V_E \sin \Theta_E)]$$

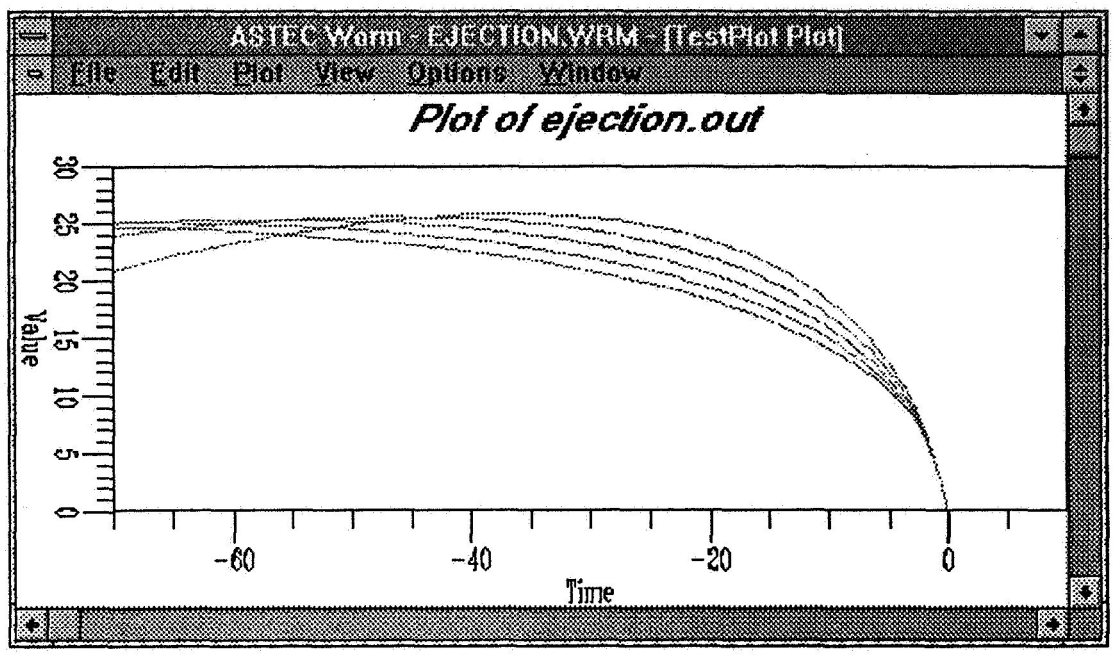
and further

$$X_0 = Y_0 = 0$$

The following quantities are to be printed every 0.002 seconds:

t, V, V', Θ, X, Y

```
------------------------------------------------
----------  PILOT EJECTION STUDY  ----------
------------------------------------------------
-----  EQUATIONS
x' = v*cos(th)-va
y' = v*sin(th)
clear = y>y1|y'<0
v' = IF(clear,-d/m-g*sin(th),0)
ANGLE th' = IF(clear,(-g*cos(th)/v),0) RADIANS
d = .5*rho*cd*s*v^2
v(IC) = sqrt((va-ve*sin(thd))^2 + (ve*cos(thd))^2)
ANGLE th(IC) = atan(ve*cos(thd)/(va-ve*sin(thd)))
x(IC) = 0; y(IC) = 0
-----  CONSTANTS
m  = 7    -- slugs
g  = 32.2  -- ft/sec^2
cd = 1
s  = 10   -- ft^2
y1 = 4    -- ft
ve = 40   -- ft/sec
ANGLE thd = 15 DEGREES
-----  DATA
va = 900    -- ft/sec
rho = 2.3769e-3  -- slugs/ft^3
-----  OUTPUT
DEFINE FILE WRITE ASCII 0.02 EJECTION.OUT 6F12.5
WRITE (t,v,v',th,x,y)
-----  MODEL PARAMETERS
DT  = 0.002; TFIN = 2.0
```



Conclusion

The new MODEL programs is an attempt to take a sixties-vintage program and updata it for the nineties. When integrated into the other modules it of ASTEC, it should prove to be an extremely useful design tool. As a standalone program, it contains some features available in no other package currently available. It will soon be submitted to COSMIC for publication.

References

1. Bauer, F. H. and Downing, J. P., Interactive Controls Analysis (INCA) User's Manuals (4 Volumes), Program Number GSC-12998, COSMIC, University of Georgia, Athens, Georgia, 1985. Updated 1988.
2. Bauer, F. H. and Downing, J. P., Windowed Observation of Relative Motion (WORM) User's Manuals (2 Volumes), Program Number GSC-13232, COSMIC, University of Georgia, Athens, Georgia, 1988.
3. Zimmerman, B. G., Multi-Optimal Differential Equation Language (MODEL) User's Manual, Program Number GSC-12830, COSMIC, University of Georgia, Athens, Georgia, 1980.
4. Downing, J. P., Bauer, F. H., and Thorpe, C. J., ASTEC -- Controls Analysis for Personal Computers, Proceeding of the Third Annual Conference on Aerospace Computational Control, Oxnard, California, pp. 600-605, 1989.

Animation of multi-flexible body systems and its use in control system design

Carl Juengst
Boeing Defense & Space Group
P.O. Box 3999, Seattle, WA 98124

Ron Stahlberg
Center for Simulation and Design Optimization
of Mechanical Systems
University of Iowa
Iowa City, IA 52242

Introduction

Animation can greatly assist the structural dynamicist and control system analyst with better understanding of how multi-flexible body systems behave. For multi-flexible body systems, the structural characteristics (mode frequencies, mode shapes, and damping) change, sometimes dramatically with large angles of rotation between bodies. With computer animation, the analyst can visualize these changes and how the system responds to active control forces and torques. Figure 1 characterizes the type of system we wish to animate.

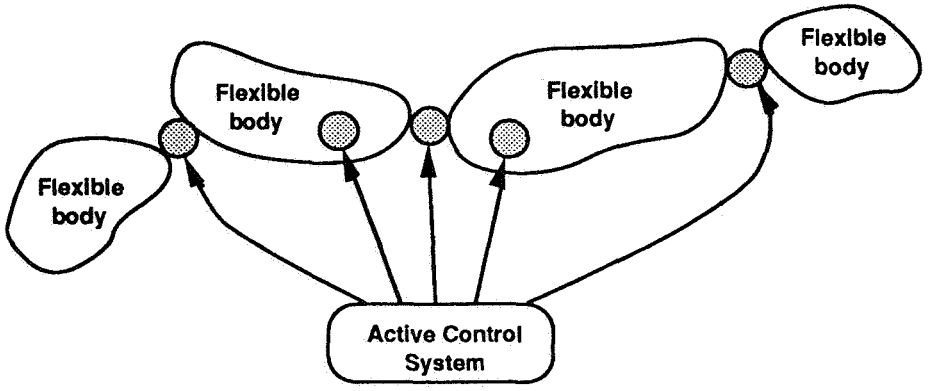


Figure 1. Multi-Flexible Body System To Be Animated

The lack of clear understanding of the above effects was a key element leading to the development of a multi-flexible body animation software package. The resulting animation software is described in some detail here, followed by its application to the control system analyst. Other applications of this software can be determined on an individual need basis.

A number of software products are currently available that make the high-speed rendering of rigid body mechanical system simulations possible. However, such options are not available for use in rendering flexible body mechanical system simulations. The desire for a high-speed flexible body visualization tool led to the development of the Flexible Or Rigid Mechanical System [1] (FORMS) software. This software was developed at the Center for Simulation and Design Optimization of Mechanical Systems at the University of Iowa. FORMS provides interactive high-speed rendering of flexible and/or rigid body mechanical system simulations, and combines geometry and motion information to produce animated output. FORMS is designed to be both portable and flexible, and supports a number of different user interfaces and graphical display devices. Additional features

have been added to FORMS that allow special visualization results related to the nature of the flexible body geometric representations.

History

The FORMS software is a direct descendent of the Visualization of Dynamic Systems[2] (VDS) software also developed at the University of Iowa. This software was designed to perform high-speed animation of rigid body simulations. VDS afforded the user a high degree of interactive control of simulation parameters, such as color, visibility, and viewing orientation. This software utilized euler vectors to specify body position and orientation, which combined rotational and translational information into one data component. The rigid nature of the bodies being rendered allowed VDS to support additional features, such as interference checking and casting of shadows. The input formats supported by VDS were somewhat limited, using an internally developed proprietary format. Translators were, however, provided between this format and a number of standards such as the Initial Graphics Exchange Specification[3] (IGES) and Movie.BYU[4]. The structure of VDS allowed for an open architecture, which enabled the software to support a large number of different user interfaces and display devices. Any combination of user interfaces and display devices could then be supported on a particular platform.

The development of FORMS incorporated many of the features utilized in the development of VDS, and also allowed for the addition of a significant number of new capabilities. FORMS is compatible with the VDS software, with the exception of some features directly related to the rigid nature of VDS objects, and can be used to produce animations from VDS input files without modification. The limitation of the proprietary input format imposed by VDS was also lifted, as FORMS supports three different input formats: the VDS proprietary format; a close modification of the VDS proprietary format intended to support flexible body information; and an ASCII input format designed to be easily edited and modified. The flexible nature of the bodies being rendered also required that the techniques for storing both geometry information and motion data be updated. Additionally, FORMS was intended to support a number of different modes of interaction, including structure editing and harmonic motion display.

Organization

The FORMS software uses a hierarchy to store information required to produce an animation sequence. Three different types of information are stored in this hierarchy. The first is what is considered essentially universal data. Data of this type influences the animation as a whole and includes such standard rendering options as view specifications, lighting, and environmental parameters. The second type of information stored in the hierarchy involves the "objects" or "parts" of the mechanical system being rendered. The nature of the motions specified in FORMS allows a linked list to be used in storing such objects. The use of a linked list rather than a tree-like structure is made possible by the fact that motions for the flexible bodies are represented as displacements on individual nodes and not motions that can be applied to a group of objects. Each object in this list contains an associated geometric representation for the object and object-specific attributes such as color and representation (i.e., solid, wireframe, point). The final type of information stored in the hierarchy is animation data. This data is stored on a per-frame basis and can be of one of two types: euler vectors, which are converted into displacement information for each node in the geometry, and straight nodal displacement data. In the context of each individual frame, the animation data is then organized according to the order specified by the object list in the hierarchy.

The functional organization of the FORMS software was designed to maximize flexibility and portability. There are three main components to the FORMS software: a communication manager, display device drivers, and user interfaces. The main FORMS executable functions as the communications manager. It handles requests from the various user interfaces and then instructs the display devices to update their representations accordingly. This organizational model is illustrated in figure 2.

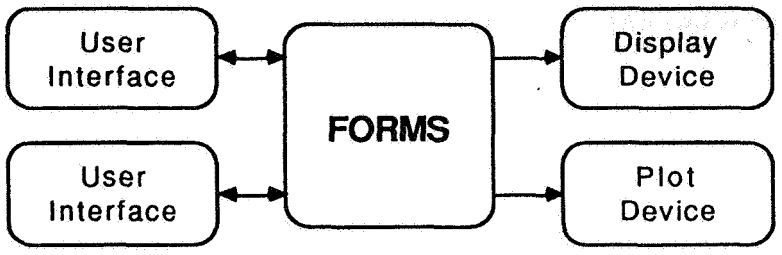


Figure 2. FORMS Communication Channels

The actual FORMS executable is isolated from any individual user interface or device driver, a design that greatly improves the portability of the FORMS software.

Data Requirements

Three types of data are required as input to the FORMS software: geometric data, motion data, and initial configuration data. Geometric data can be in the form of either rigid or flexible body representations. For rigid bodies, the data is in the form of a polygonal mesh. Flexible bodies require that geometric information be presented in the form of a finite element mesh. At the current release, seven different types of finite elements are supported by the FORMS software: point, beam, linear triangle, linear quadrilateral, linear tetrahedron, linear solid wedge, and linear solid. The representations of these elements are as presented in the IGES specification.

Motion information is specified as either euler vectors or nodal displacements. The type of motion information allowed depends on the associated geometric representation for the object. If the object is a rigid body, then only euler vector data can be used in positioning the object. For flexible objects, motion information can be specified as either nodal displacements or as a combination of euler vectors and nodal displacements. If a combination of euler vectors and nodal displacements is supplied, then the euler vectors are first converted to nodal displacements. After this conversion, the individual nodal displacements specified as input are applied. This "two-step" motion is critical in analyzing certain classes of problems.

Initial configuration information is supplied in the form of a definition or ".def" file. This file contains initial viewing, lighting, and environmental parameters, as well as initial object attributes and motion data ordering. Each object represented in the animation has a corresponding entry in the definition file. This entry contains the object's initial color, representation, shading model, and geometric representation. Geometric representations are denoted by name, which must correspond to the name of a representation supplied as input in one of the geometry files. Motion data ordering is specified in the form of an ordered list of objects. Additional objects may appear in the definition file for which no motion is specified. These objects are considered stationary.

The formats for the required data are of one of three types: VDS module format, FORMS flexible module format, or ASCII "easy" file format. The module format and flexible module formats are

proprietary file formats, which can be in either a space-reducing, machine-dependent binary format or a platform-independent ASCII representation. Both of these formats can be used to encode both motion and geometric data, but are not readily editable because they contain additional information to speed file access. In order to facilitate conversion to these formats, a number of translators from various standards are provided, including IGES, Movie.BYU, and DADS-3D[5]. The "easy" file format is an ASCII format designed to be edited using any standard text editor. The format of these files is relatively simple. No strict layout is required with regard to column location, line spacing, etc. In this way, the output from nearly any package can be manipulated into the "easy" file format with a minimum of effort.

Rendering

The major concern in rendering flexible objects is that as nodal displacements occur, faces of finite elements that were initially essentially planar lose this property. Therefore, in order to render flexible bodies with sufficient accuracy and still maintain the required throughput for animation, some pre-processing is required. In FORMS, this pre-processing takes the form of triangulation of all polygons and/or finite element faces with more than three vertices. This division of such surfaces into triangles allows each face to be decomposed into a number of planar facets. Since each three nodes will uniquely determine a plane, each triangular facet can then be rendered as planar and still give the appearance of the required flexible motion. This process is illustrated in figure 3.

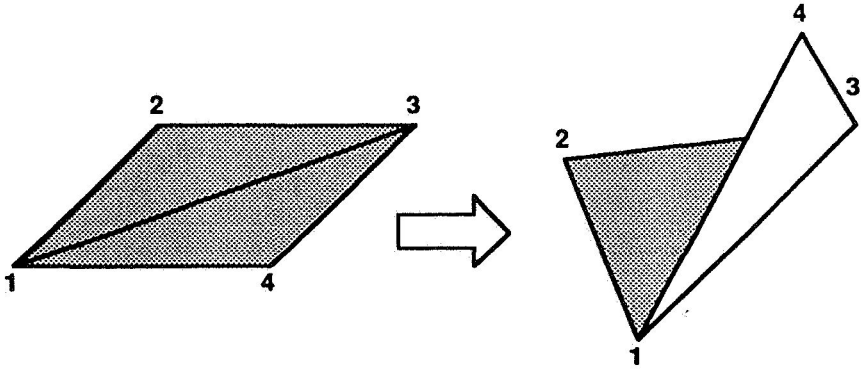


Figure 3. Triangulation into Planar Facets for Flex

Since the facets are assumed to remain contiguous, the representation will remain accurate. For simple finite element representations, the triangulation is straightforward. For more advanced polygonal representations, a modified version of the triangulation algorithm presented by Feng and Pavlidis[6] is used; the only restriction is that the polygons be simple. The resulting triangulation is used further in graphically editing the objects, since a strictly triangular representation of the scene allows for a general purpose routine to test for intersection between objects and a selected point. By computing the triangulations of these objects prior to animation and independent of the displacements involved, rendering requires only the updated positions of the individual nodes to be computed prior to drawing. The mesh generated in triangulating an object can be displayed optionally on a per-object basis.

The viewing and lighting attributes specified for each animation are generic in nature and are not of particular interest to this paper. The only aspect of viewing that will be mentioned is the fact that FORMS supports multiple views of an individual animation. Currently, up to four synchronized, independent views of any animation are permitted.

Environmental parameters are of more interest, because some values are directly related to the flexible nature of the bodies being rendered. Values that relate to the animation in general include background color and interval control. Interval control allows for strict frame-rate control, as opposed to allowing the platform to render as fast as possible. The parameters related to the flexible nature of the objects include: the color used in rendering the triangle mesh generated during preprocessing; the colors used in representing active vs. inactive and active vs. special elements during editing; and the period used in displaying harmonic motion. Each of these parameters will be discussed in more detail as they relate to individual options.

By far the greatest number of attributes are possessed by the individual objects. These parameters relate either to the general appearance of the object or to specific flexible body options. The parameters that affect general appearance include color (including diffuse, ambient, and specular color, if supported), shading model (flat or Gouraud), label status, position, and physical representation. Current representation options include solid, wireframe, point (only vertices or nodes are rendered), stick (a combination of wireframe and point representations), and disabled. Additional parameters allow for specific flexible body options. These parameters include:

superimposition	Controls display of the undeformed geometry superimposed over the current representation. Additionally, the color and representation attributes for the superimposed geometry can be independently specified.
labels	Allows the rendering of node and/or element labels.
pointsize	Controls the size of nodes rendered in point or stick representation.
beamwidth	Controls the size of the lines rendered in wireframe or stick representation.
active vs. inactive elements	Allows any element or range of elements to be made inactive, and causes them to not be displayed.
trace nodes	Causes any element or range of elements to leave a "smoke trail" as it moves from frame to frame.
inferior representations	Allows elements that do not have the representation specified for the object to be viewed. In some representations, such as solid, certain element types have no corresponding representation (i.e., beam or point elements). This option causes such elements to be rendered in their native representations.
mesh representation	Enables display of the triangle mesh generated during preprocessing of this object.

Any or all of these parameters can be manipulated interactively for any or all objects.

Animation Control

A number of controls are available with respect to the display of an animation sequence. These include the display rate, direction, sequence, and increment. Frames are normally displayed at a rate determined by the limitations of the current hardware platform, but they can also be set to appear at a pre-determined rate, as long as it does not exceed the capabilities of the hardware. The order in which frames are displayed is totally under user control, with any sequence, direction, or group of individual frames being valid for display. In the event that different timestep increments are desired for display other than those available from the input data, an increment value is available for use in displaying frame sequences. If required, linear interpolation is performed to generate intermediate frames.

It is also desirable at times to magnify the motions or displacements for a certain object. In this case, an individual motion magnification factor is maintained for each object. Separate magnification factors are kept for nodal displacement values and euler vector values. Nodal displacement values may be magnified by individual factors in any of three dimensions (x, y, z). Euler vectors can be magnified for any of seven values (tx, ty, tx, er, ex, ey, ez).

A final animation option is provided in harmonic motion display. In this mode, a deformed state of the simulation is chosen as the base frame. FORMS then uses this frame as an ending position and the undeformed geometry as an initial position. It then interpolates between the two frames, based on the number of frames specified for the period of harmonic motion. The number of frames is specified as an environment parameter. In this mode, the total number of frames available for display is updated to reflect the limitations of the harmonic motion.

Editing

In order to make the information displayed during an animation more useful, it is often desirable to change the appearance of certain elements or disable them altogether. In FORMS, these types of operations are accomplished through the use of a special rendering mode. In this mode, individual elements of a flexible body can be selected with the aid of a mouse or other pointing device. Depending upon the values set when initiating edit mode, selected elements will either be toggled between active and inactive status or active and special status. In active vs. inactive status, inactive elements are no longer displayed when returning to normal display mode. However, in edit mode, inactive elements are displayed in a distinct color to signify their modified status. In active vs. special status, individual elements may be assigned distinct color, representation, pointsize, and beamwidth values. These new values will then be used in rendering the special elements after returning to normal display mode. As with active vs. inactive status, special elements are displayed in a distinct color from other active elements while editing.

Other Information

Aside from the qualitative information provided through the animation of mechanical system motion, more quantitative information is often desired from any analysis tool. To provide such data, FORMS has two additional output options: X-Y plotting and parameter display. The X-Y plotting option is performed in a separate window from the animation. Plotting occurs in synchronization with the animation and simultaneously provides both a quantitative and qualitative view of the system under study. The range of expressions available for plotting includes any system variables, such as nodal displacements, euler vectors, and frame count. Other available

options include incremental counters, standard library functions, and any numeric constants. The range and scale of both axes can be user-specified, as can the color of the resulting plot. Plot results can be superimposed to compare various data analyses.

The "easy" file format also provides for user-specified parameter data, other than the standard nodal displacements and euler vectors. The parameter data can be entered in the motion data file and then displayed during the playback of an animation in a separate window. In this way, FORMS can inform the user of significant events or non-numeric results.

User Interfaces/Device Drivers

As was mentioned earlier, FORMS was designed with portability and flexibility in mind. To that end, two different user interfaces are currently available. The first is a standard command line keyboard interface. The second is a graphical user interface built on top of OSF Motif[7]. The graphical user interface uses a number of specialized widgets to facilitate FORMS operations.

FORMS also currently supports two different graphical display devices. Drivers currently exist to support either X11[8] or Silicon Graphics GL[9].

Multi-Flexible Body Simulations

The FORMS software requires animation data containing the rigid rotation and translation of each body and/or the displacements of the nodes for flexible bodies, as described above. This data can be generated by running multi-flexible body dynamics simulations such as TREETOPS[10], DISCOS[11], SADACS[12, 13], or any other simulation that can compute rigid body motion and/or flexible node displacements. We are currently using the Spacecraft Appendage Dynamics And Control Simulation (SADACS) as the driver for FORMS. SADACS was derived by Boeing Defense & Space Group for faster simulation of large-angle, flexible body vehicles. SADACS utilizes SD/EXACT[14] rigid body dynamics code run in parallel with the FB2 code (linearized flexible dynamics updated at user-defined increments). The dynamics are driven by a control system and detailed actuator and sensor hardware models, including friction and other nonlinear effects. A simplified SADACS block diagram is shown in figure 4.

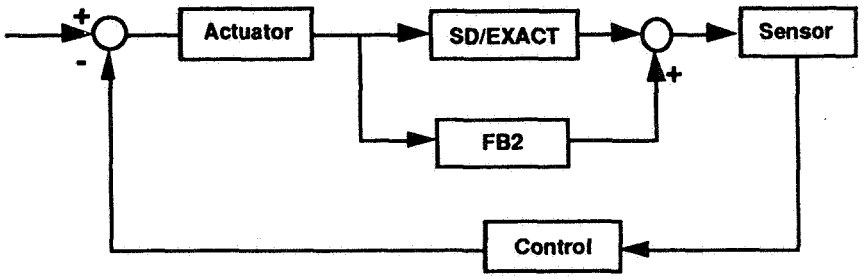


Figure 4. Simplified SADACS Block Diagram

The animation data is written to an ASCII file in the FORMS "easy" format. As SADACS is a high-speed computer simulation, many analyses and subsequent animations can be done in a short period of time.

Structural Characteristics

For multi-flexible body structures where the bodies may move through large angles with respect to each other, the structural characteristics can change dramatically. Analysis of a four-body spacecraft shows significant change in the structural modes of concern when the solar array is rotated through 90 degrees. Open-loop transfer functions in figure 5 show this result.

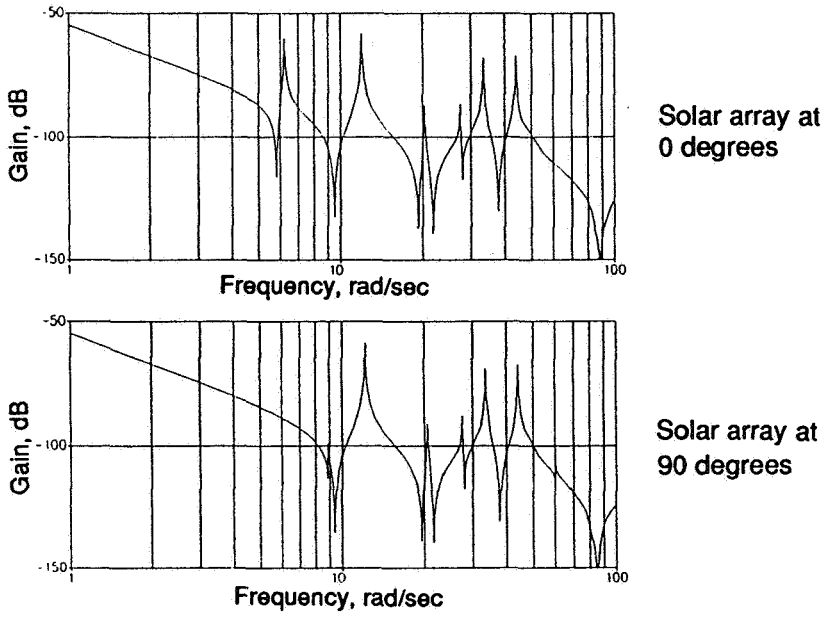


Figure 5. Open-Loop Transfer Functions for a Spacecraft with Different Solar Array Positions

The system mode frequencies and damping have been seen to change up to 50 percent. In addition to the frequency change, modes may swap with other modes or move to a different degree of freedom. The location of deformations in the structure may also change depending on the orientation of the bodies. To further complicate the picture, control forces and torques also affect how the structure behaves. The control system designer relies on knowledge of how the structural modes change in order to "tune" the control system for maximum performance. Animation of the multi-flexible body system is an extremely useful tool for visualizing all of these effects.

Animation and Control System Design

The FORMS animation software allows the control system analyst to employ visualization during the design process. Control system parameters can be changed and then animated to show whether flexibilities are suppressed or excited. This increased knowledge of the structural characteristics gives the designer more accuracy in determining, for example, optimum size and placement of sensors and actuators. Valuable insights are being realized now from current usage of animation software for control system design. Control system design is only one of many applications for multi-flexible body animation software.

[1] Copyright 1991, Center for Simulation and Design Optimization of Mechanical Systems, University of Iowa.

[2] Copyright 1989, Center for Simulation and Design Optimization, University of Iowa.

[3] Smith, B., Rinaudot, G., Reed, K., Wright, T, Initial Graphics Exchange Specification (IGES) Version 4.0, U.S. Department of Commerce, June 1988.

[4] Copyright 1988, Brigham Young University.

[5] Copyright 1991, Computer Aided Design Software, Incorporated.

[6] Feng, H. and Pavlidis, T., "Decomposition of Polygons Into Simpler Components: Feature Generation for Syntactic Pattern Recognition," IEEE Transactions on Computers, June 1975, pp. 636-650.

[7] Copyright 1990, Open Software Foundation.

[8] Copyright 1988, Massachusetts Institute of Technology.

[9] Copyright 1990, Silicon Graphics, Incorporated.

[10] Sing, R. and VanderVoort, R., "Dynamics of Flexible Bodies in Tree Topology - A Computer-Oriented Approach," J. Guidance, Vol. 8, No. 5, pp. 584-590, Sept-Oct 1983.

[11] Bodley, C., Devers, A, Park, A, and Frisch, H., "A Digital Computer Program for Dynamic Interaction and Simulation of Controls and Structures (DISCOS)," NASA Tech. Paper 1219, May 1978.

[12] Jacot, A. D., Jones, R. E., and Juengst, C., "High Speed Simulation of Flexible Multibody Dynamics," Proceedings NASA Workshop on Structural Dynamics and Critical Interactions of Flexible Structures, Huntsville, Alabama, April 1986.

[13] Jones, R. E, Morse, T. W., and Russell, W. C., "High Speed Simulation of Multi-Flexible-Body Systems with Large Rotations," Proceedings AIAA Dynamics Specialist Conference, Monterey, California, April 1987.

[14] SD/EXACT is ta rademark of Symbolic Dynamics, Inc., 928 Wright Avenue, Suite 108, Mountain View, California 94043.

The Use of Linked Lists in the Simulation of Controller – Structure Interaction

Ralph Quan[1]
Frank J. Seiler Laboratory,
U. S. Air Force Academy, Colorado Springs, CO 80840

Abstract. An algorithm for the computer simulation of large space structures under active control is considered. Linked lists are used in a matrix data structure to implement the trapezoidal rule on the system differential equations. The use of the trapezoidal rule ensures that the numerical stability is equivalent to the system stability, which is essential for this type of simulation. The sparsity of the system matrices is exploited by the linked lists, and the algorithm efficiently steps through the lists in an orderly fashion. Results of simulations on a NASA large space structure experiment are reported.

Key words. large space structures, control, linked lists, simulation, trapezoidal rule, LU factorization, LDL^T factorization, sparsity

1. Introduction. Future large space structures such as the space crane and the aerobrake will possess high flexibility and low damping. The suppression of vibrations is an important problem for this type of structure, since slewing maneuvers or disturbances can cause large amplitude vibrations over long time intervals.

Vibration suppression can be accomplished via active feedback control. However, it is possible that an unstable controller design would damage the structure. Computer simulations are therefore desirable for evaluating controller performance and for detecting instability.

Various approaches have been taken to simulate controller – structure interaction (CSI), such as those described in [1,2,3,4] and [5,p.3]. These approaches have accuracy and / or numerical stability limitations inherent in them. The method to be described in this paper overcomes these limitations. For instance, accuracy is maximized through the use of the finite element model of the large space structure, instead of a reduced order model. In addition, the numerical stability of the simulation is made equivalent to the stability of the physical system. This has been difficult to implement in the past, because of the differences between large space structure models and control system models. Finite element models of large space structures contain large, sparse, and symmetric matrices; compensator models tend to be small, dense, and unsymmet-

[1]National Research Council Fellow, Aerospace Sciences Division

ric. An implicit integration scheme such as the trapezoidal rule has the property that the numerical stability of the simulation is equivalent to the stability of the physical system, but it also combines the matrices of the structure and the compensator; this destroys sparsity and symmetry. Computer memory requirements can thereby become impractically large, if finite element data structures are used (banded matrix, skyline matrix, etc.). Therefore, a linked list data structure is developed below which recovers the sparsity of the computer simulation. Linked list algebra and factorization are also developed, so that an implicit integration scheme can be implemented.

2. Closed Loop System Equations. Consider the following linear finite element model of a multi – input, multi – output structure:

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{B}\mathbf{u} \tag{1}$$

$$\mathbf{y} = \mathbf{C_y} \begin{pmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{pmatrix} \tag{2}$$

where the displacement $\mathbf{q} \in \Re^{n\times 1}$, the velocity $\dot{\mathbf{q}} \in \Re^{n\times 1}$, the acceleration $\ddot{\mathbf{q}} \in \Re^{n\times 1}$, the input force $\mathbf{u} \in \Re^{m\times 1}$, the output vector $\mathbf{y} \in \Re^{3n\times 1}$, and $\mathbf{M}$, $\mathbf{C}$, $\mathbf{K}$, $\mathbf{B}$, $\mathbf{C_y}$ are the mass, damping, stiffness, input, and output matrices, respectively.

The mass matrix is assumed to be positive definite; the damping and stiffness matrices are assumed to be positive semidefinite. An additional n differential equations are associated with equation (1), because the velocities are the derivatives of the displacements.

A linear compensator is assumed, with dynamics as follows:

$$\begin{pmatrix} \dot{\mathbf{x}} \\ \mathbf{u} \end{pmatrix} = \mathbf{L} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \tag{3}$$

where the compensator state vector $\mathbf{x} \in \Re^{r\times 1}$, and the matrix $\mathbf{L} \in \Re^{(r+m)\times(r+3n)}$.

The differential equations above can be appended together to form one set of differential equations:

$$\dot{\mathbf{z}} = \mathbf{V}\mathbf{z} \tag{4}$$

If we have the state $\mathbf{z}$ at the Nth time step (time t), then we can obtain the state

at the N+1 time step (time t+h) by using the trapezoidal rule:

$$\mathbf{z}_{N+1} = \mathbf{z}_N + \frac{\mathbf{h}}{\mathbf{2}}(\dot{\mathbf{z}}_{N+1} + \dot{\mathbf{z}}_N) \tag{5}$$

A linear system is stable if and only if its eigenvalues are in the left half complex plane. It is desirable that this system stability region coincide with the numerical stability region. Unexpected damage would result if the computer simulation of a controlled large space structure were to show stability, when in fact the controller were to destabilize the structure. The numerical stability region of the trapezoidal rule does coincide with the left half complex plane [6,7]; therefore this algorithm is a logical choice for the simulation of controller – structure interaction.

3. Sparse Matrix Storage. Consider the following simplified control problem, which illustrates the computer storage difficulties associated with the simulation of controller – structure interaction:

If the structure is in a steady state condition, equation (1) simplifies to:

$$\mathbf{Kq} = \mathbf{Bu} \tag{6}$$

As an example, consider the case where the structure has a tridiagonal stiffness matrix (the 'x' entries represent nonzero elements):

$$\mathbf{K} = \begin{pmatrix} x & x & 0 & 0 & 0 \\ x & x & x & 0 & 0 \\ 0 & x & x & x & 0 \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \end{pmatrix} \tag{7}$$

The nonzero elements of this matrix exhibit a certain pattern, which makes it possible to store the three diagonals of the matrix as three arrays. Let us place a force actuator at the first degree of freedom:

$$\mathbf{B} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{8}$$

Now place a displacement sensor at the last degree of freedom:

$$y = \mathbf{C_q q} \tag{9}$$

$$\mathbf{C_q} = (0 \quad 0 \quad 0 \quad 0 \quad 1) \tag{10}$$

and establish output feedback from the displacement sensor to the actuator:

$$u = -gy \tag{11}$$

Then the closed loop system equation becomes:

$$\tilde{\mathbf{K}}\mathbf{q} = \mathbf{0} \tag{12}$$

$$\tilde{\mathbf{K}} = \begin{pmatrix} x & x & 0 & 0 & g \\ x & x & x & 0 & 0 \\ 0 & x & x & x & 0 \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \end{pmatrix} \tag{13}$$

The control has affected the sparsity pattern, and it is not clear if renumbering the degrees of freedom would help considerably. When implementing the trapezoidal rule on equations (1-3), the same situation is encountered. If all elements of such matrices were stored, then computer memory would be wasted on the storage of zero–valued matrix elements. This would be a serious problem for large space structure type problems. Therefore, a linked list data structure is developed below which stores only the nonzero elements.

4. Linked List Matrices. Figure 1 displays the data structure for a linked list matrix. The leftmost array in the figure contains the number of nonzero columns in each row. Adjacent to this array is another array which points into linked lists for each row. Each record in the linked list contains a field for a floating point matrix data element, and a field for the column number.

As it will be shown later, addition and multiplication of matrices can be done if the linked lists are traversed in one direction. However, factorization of matrices will require the deletion of matrix elements. A matrix element deletion requires that the two surrounding elements be connected together; knowledge of the locations of the two

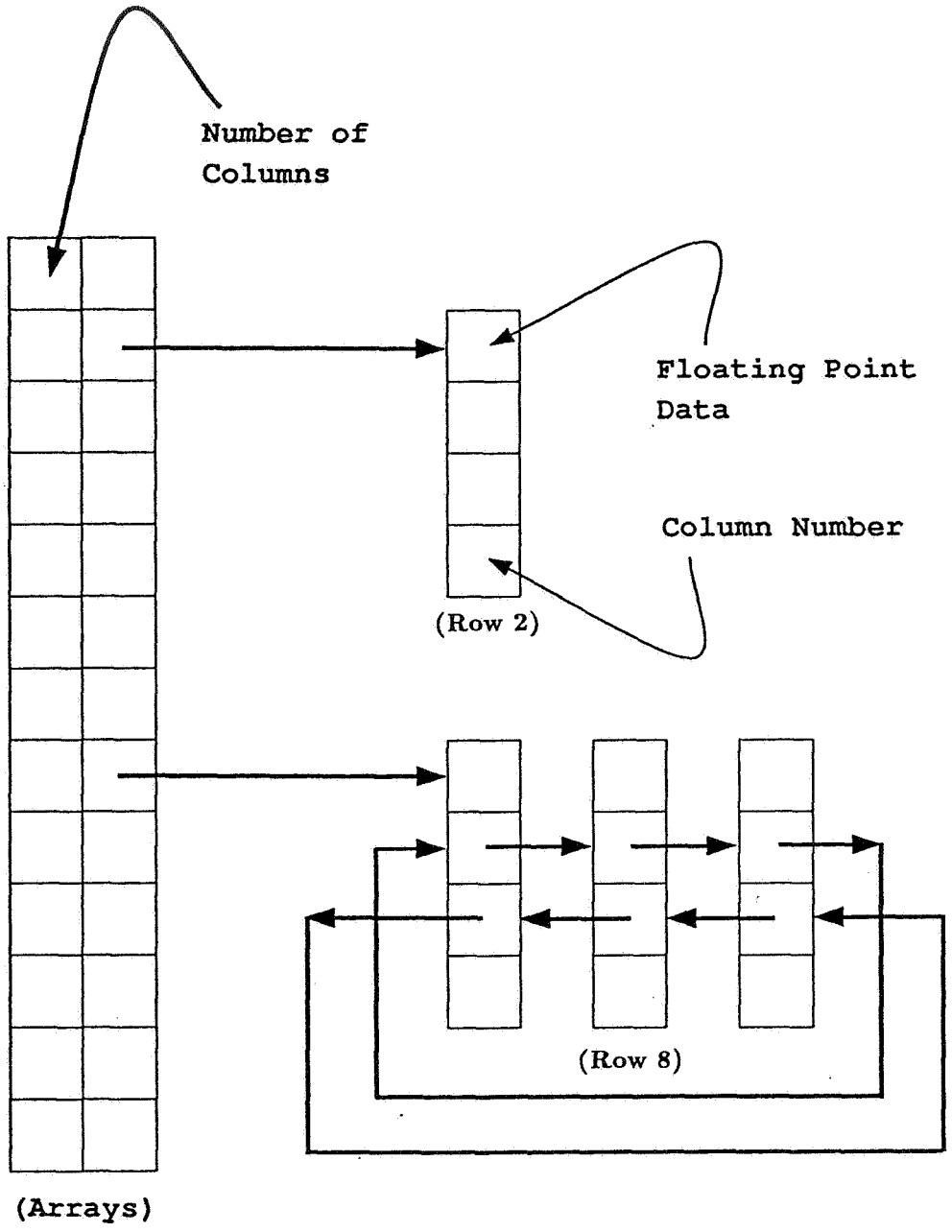


Figure 1: Linked List Matrix

surrounding elements is needed. This information can be quickly obtained if the linked list can be traversed in both directions. To establish this "double linking", there are two pointers in each linked list record which point to the two surrounding elements. If there is only one element in a linked list, then the two pointers point at that element.

Although it is possible to store and recall elements in any order (random access), it is more efficient to store and recall matrices by rows. The next section demonstrates that such row access can be used to implement the algebra needed for a controller – structure interaction simulation.

5. Sparse Matrix Algebra

5.1 Sparse Matrix Addition and Multiplication

The addition of matrices is necessary for the assembly of the finite element model from the element mass and stiffness matrices. Matrix addition, multiplication, and factorization is also required for the implementation of the trapezoidal rule (see [5] for details). The addition of two linked list matrices can be accomplished by stepping through the linked lists of the first matrix by rows. At the same time, the corresponding element in the second matrix is recalled. The two elements from the two matrices are added together and stored in the second matrix. Thus the final result appears in the second matrix.

The multiplication of two matrices is often computed by using inner products:

$$\textit{Given } \mathbf{A} \in \Re^{\mathbf{m} \times \mathbf{n}}, \ \mathbf{B} \in \Re^{\mathbf{n} \times \mathbf{p}}$$

$$\mathbf{C} = \mathbf{A} * \mathbf{B}, \quad \mathbf{C} \in \Re^{\mathbf{m} \times \mathbf{p}}, \quad \mathbf{C_{ij}} = \sum_{\mathbf{k=1}}^{\mathbf{n}} \mathbf{A_{ik}} \mathbf{B_{kj}} \tag{14}$$

In order to perform the multiplication efficiently by stepping through the linked lists in order, it is necessary to rearrange formula (14) into a form which resembles an outer product:

$$\mathbf{C} = \sum_{\mathbf{i=1}}^{\mathbf{m}} \sum_{\mathbf{j=1}}^{\mathbf{n}} \mathbf{A_{ij}} \beta^{\mathbf{ij}} \tag{15}$$

where β^{ij} is the zero matrix of dimension $(m \times p)$ with the ith row replaced by the jth row of the **B** matrix. Note that the **A** matrix is stepped through once, and that the **B** matrix is stepped through at most m times. This multiplication procedure can be illustrated for two dimensional matrices:

$$\begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

$$= \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

$$= a_{11} \begin{pmatrix} b_{11} & b_{12} \\ 0 & 0 \end{pmatrix} + a_{12} \begin{pmatrix} b_{21} & b_{22} \\ 0 & 0 \end{pmatrix}$$

$$+ a_{21} \begin{pmatrix} 0 & 0 \\ b_{11} & b_{12} \end{pmatrix} + a_{22} \begin{pmatrix} 0 & 0 \\ b_{21} & b_{22} \end{pmatrix}$$

5.2 LU factorization of a Sparse Matrix

It is known that a matrix $\mathbf{A}$ can be factorized into a product of a lower triangular matrix $\mathbf{L}$ and an upper triangular matrix $\mathbf{U}$. If the matrix equation $\mathbf{Ax} = \mathbf{b}$ must be solved repeatedly for different $\mathbf{b}$ vectors, then the LU factorization leads to computational efficiency [8]. This situation exists during the simulation of controller – structure interaction (see [5]). A linked list matrix is to be LU factorized, and therefore the usual procedure for LU factorization needs to be modified. The new procedure is given as follows:

a) Form the transpose of the matrix $\mathbf{A}$ ($\mathbf{A^T}$).

b) For all of the rows which have not been selected as a pivot row:

1. Select the sparsest row as the pivot row.

   This idea was used in [9]. However, the procedure described in [9] assumed that the nonzero elements of the sparse matrix to be factorized are packed into arrays. Deletions of matrix elements are necessary in the factorization, which is cumbersome if the data is stored in arrays. With linked lists, deletions are simple in that pointers in the matrix elements surrounding the deleted element are redirected at each other. The insertion of matrix elements is also relatively simple. This is important for the assembly of the finite element model via the addition of element mass and stiffness matrices into the global mass and stiffness matrices [10]. Thus a single data structure (the linked list matrix) can be used for the assembly of the finite element model and for the implementation of the trapezoidal rule; this avoids the need for conversions between data structures.

2. Find the element with the maximum magnitude within the pivot row. This element will be referred to as the pivot element, and the column of this element will be referred to as the pivot column. Later below, division by the pivot element will be performed, which is why the maximum magnitude element is chosen as the pivot element. If the matrix is nonsingular, a nonzero element will be found in the pivot row.

3. Delete the pivot row from $\mathbf{A^T}$.

4. Save a copy of the pivot element, and delete it from $\mathbf{A}$.

5. The nonzero elements in the pivot column are referred to as "target elements". These target elements appear in a row in $\mathbf{A^T}$, allowing for quick access. For each target element,

   i) multiplier = – target element / pivot element

   ii) Store the multiplier in the matrix $\mathbf{L}$, and delete the target element from $\mathbf{A}$ and $\mathbf{A^T}$.

   iii) Multiply the pivot row by the multiplier and add it to the target row of $\mathbf{A}$ and $\mathbf{A^T}$. Since we chose the sparsest row as the pivot row, we retain as much sparsity as possible.

(End of factorization: The matrix $\mathbf{U}$ now appears in place of matrix $\mathbf{A}$)

The matrix $\mathbf{A}$ is stored by rows. Since access by columns is needed above, $\mathbf{A^T}$ is stored. It might appear that this would double the storage requirements. However, $\mathbf{A}$ is sparse. The LU factorization process creates "fill in" (some of the zero elements become nonzero) within the $\mathbf{U}$ matrix. After each step of the factorization process, another column of $\mathbf{A^T}$ is no longer needed. The storage for this column goes towards the storage of the "fill in" elements.

After the factorization has been completed, the linked list matrix $\mathbf{A}$ is left with the upper triangular part of the factorization, and linked list matrix $\mathbf{L}$ is the lower triangular part.

5.3 $\mathrm{LDL^T}$ Factorization of a Sparse Matrix

A symmetric linked list matrix equation needs to be solved at every time step in the simulation of controller – structure interaction (see [5]). Another problem where a symmetric linked list matrix equation has to be repeatedly solved occurs during the computation of the lowest frequency modes of a structure (see [5]). The $\mathrm{LDL^T}$ factorization can be employed to efficiently solve these matrix equations. $\mathbf{L}$ is a lower triangular

matrix with ones on the diagonal, and **D** is a diagonal matrix. The LDL^T factorization is described and analyzed in Golub and Van Loan [8]. The algorithm is listed below in the style which Golub and Van Loan use in their text. The notation 1:j signifies all of the integers from 1 to j.

LDL^T Algorithm: If $\mathbf{A} \epsilon \Re^{(\mathbf{n}\mathbf{x}\mathbf{n})}$ is symmetric then the following algorithm computes a unit lower triangular matrix **L** and a diagonal matrix **D** so that $\mathbf{A} = \mathbf{LDL^T}$. It is assumed that only the lower half of the matrix **A** is stored, because of the symmetry. The matrix **A** is overwritten with the matrices **L** and **D** by this algorithm.

for $j = 1 : n$
  for $i = 1 : (j-1)$
    $v(i) = A(j,i)A(i,i)$
  end
  $v(j) = A(j,j) - A(j,(1:j-1))\,v(1:(j-1))$
  $A(j,j) = v(j)$
  $A((j+1):n,j) =$
    $(A((j+1):n,j) - A((j+1):n,1:(j-1))\,v(1:(j-1)))\,/\,v(j)$
end

The LDL^T algorithm can be modified to handle sparse symmetric matrices of the linked list storage type:

The formation of the v vector on lines $(2-5)$ is done by stepping through linked lists, instead of looping over the entire range from 1 to $(j-1)$. Because of the sparsity of the matrix, the v vector is also sparse.

The algorithm yields one column of the **L** matrix at a time, as shown in lines $(7-8)$ of the algorithm. In line (8), a vector is formed by multiplying a submatrix by the v vector $(\mathbf{A((j+1):n,1:(j-1))v(1:(j-1))})$. A small example will be useful for illustrating the sparsity:

$$\mathbf{v_r} = \mathbf{M}\mathbf{v_i} \tag{16}$$

$$\begin{pmatrix} x \\ x \\ 0 \\ 0 \\ x \end{pmatrix} = \begin{pmatrix} x & x & 0 & 0 & 0 \\ 0 & x & 0 & 0 & 0 \\ 0 & 0 & x & x & 0 \\ 0 & 0 & 0 & x & 0 \\ x & 0 & 0 & 0 & x \end{pmatrix} \begin{pmatrix} x \\ x \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{17}$$

The symbol "x" in the above equation signifies a nonzero element which is in a linked list. There is no need to perform the computations for rows 3 and 4 of the result vector $\mathbf{v_r}$, because all of the terms for those rows are zero. Since the linked list storage scheme is being used, the zero part of $\mathbf{v_i}$ does not actually exist:

$$\mathbf{v_r} = \tilde{\mathbf{M}}\tilde{\mathbf{v}}_\mathbf{i} \tag{18}$$

$$\begin{pmatrix} x \\ x \\ 0 \\ 0 \\ x \end{pmatrix} = \begin{pmatrix} x & x \\ 0 & x \\ 0 & 0 \\ 0 & 0 \\ x & 0 \end{pmatrix} \begin{pmatrix} x \\ x \end{pmatrix} \tag{19}$$

The matrix $\mathbf{M}$ is stored by rows. If the transpose of $\mathbf{M}$ is stored, then it is efficient to traverse the columns of $\mathbf{M}$ corresponding to the nonzero rows in $\mathbf{v_i}$. The union of the nonzero rows in those columns forms the set of nonzero rows in the result $\mathbf{v_r}$. We will refer to this set of rows which need to be handled as the set $\bigcup$.

It might appear that the storage of $\mathbf{A^T}$ would double the storage requirements. However, $\mathbf{A}$ is sparse. The $\mathrm{LDL^T}$ factorization process creates "fill in" within the matrix. However, after each step of the factorization process, another column of $\mathbf{A^T}$ is no longer needed. The storage for this column goes towards the storage of the "fill in" elements.

The new algorithm is given as follows:

a) Form the transpose of the matrix $\mathbf{A}$.

b) For each row p of $\mathbf{A}$:

1. Delete the row from the $\mathbf{A^T}$.
2. Determine the set $\bigcup$ of rows which need to handled.

3. Recall the diagonal element $\mathbf{A_{pp}}$ for that row, and delete it from $\mathbf{A}$.

4. Compute the v vector described above.

5. For all the rows q in the set $\bigcup$ which need to be handled:

   i) Recall the section of the row q of $\mathbf{A}$ up to column p.

   ii) Compute the dot product of that section with the v vector.

   iii) Subtract this dot product from $\mathbf{A_{qp}}$, and divide by $\mathbf{v_p}$. Store this result in $\mathbf{A_{qp}}$ and in $\mathbf{A^T_{pq}}$.

(End of Procedure)

6. The Mini–Mast Truss. Mini–Mast is an active control experiment being maintained at the NASA Langley Research Center [11]. A linear finite element model having 714 degrees of freedom was developed for this truss. Two of the accelerometers were used as sensors, and all three of the torque wheels were used for control.

The Rayleigh damping coefficients were tuned to provide 2 percent damping in the first two modes and 1 percent damping in the next three modes. These first five modes and the dynamics of the torque wheels were combined to form a reduced order model of the structure, and linear quadratic regulator theory was used to design a controller. The total number of states in the compensator state vector is 16. Figures 2 and 3 show the open loop response and the closed loop response, respectively. In both cases, a one second triangular pulse was applied at one of the torque wheels. The simulations were done on a Sun-4 workstation; total memory requirements were less than 4 megabytes. About 6 minutes of computer time was used to produce the 280 time steps shown in the simulation.

7. Conclusion. An alternative approach towards the simulation of controller – structure interaction (CSI) has been described in this paper. Linked lists were used to implement the trapezoidal rule, which enforces an equivalence between numerical stability and system stability. This characteristic is essential for CSI analysis, and has not been demonstrated by previous CSI simulation methods.

Matrix storage has been implemented with linked lists, which required the development of linked list matrix algebra for the implementation of the trapezoidal rule. Thus methods for linked list matrix addition, multiplication, LU factorization, and $\mathrm{LDL^T}$ factorization were developed. The linked lists are stepped through in order in these methods, which minimizes the number of computations required. Memory requirements

Figure 2: Mini-Mast Open Loop Response

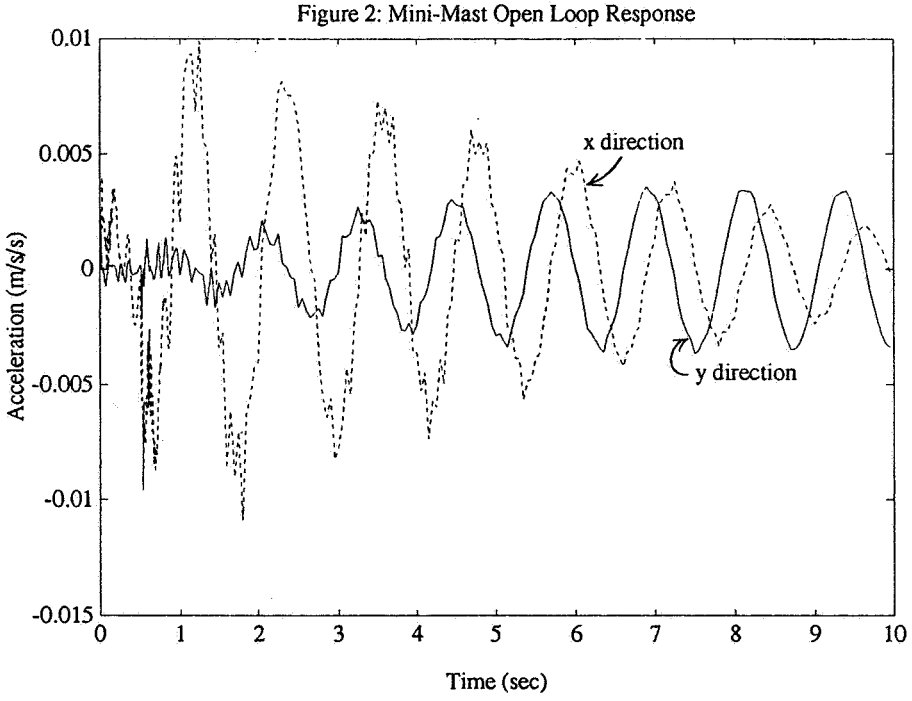
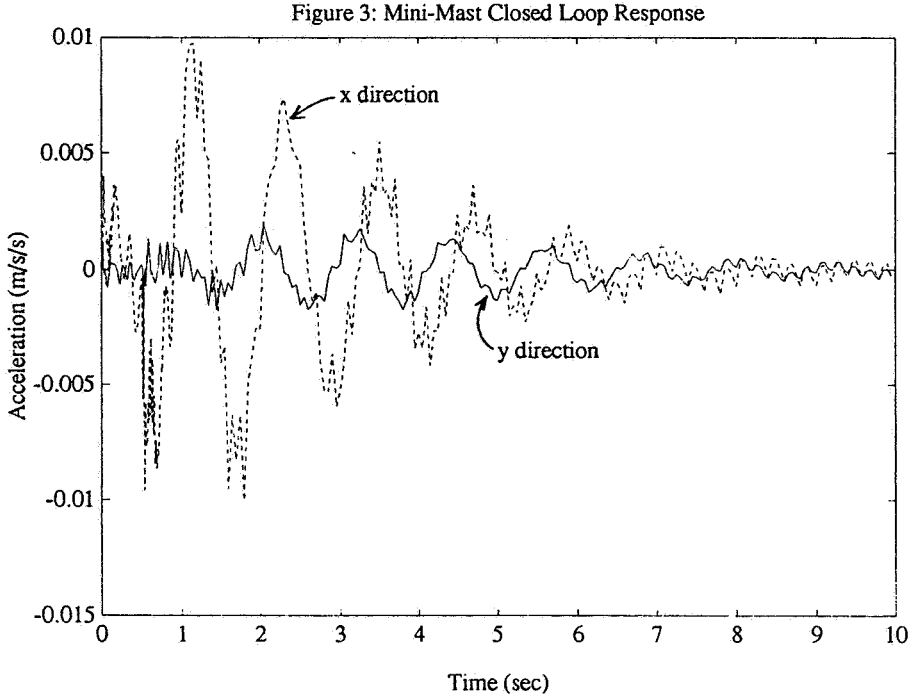


Figure 3: Mini-Mast Closed Loop Response



are also minimized because storage is dedicated only to nonzero matrix elements.

The feasability of this approach was demonstrated on a computer workstation for a large space structure experiment (the Mini–Mast truss). These linked list matrix methods show that it is possible to simulate the control of some large space structures without the use of a supercomputer. In the case of a supercomputer, it is not certain how effective linked list methods would be. A linked list is not in the form of a vector, which suggests that methods based on it would not take advantage of the special capabilities of vector processing machines. In the case of parallel processing supercomputers, research needs to be done to determine the effectiveness of these methods on such machines.

References

[1] W. K. Belvin and K.C. Park, *Computational Architecture for Integrated Controls and Structures Design*, Third Annual NASA/DOD CSI Conference, San Diego, CA (January 29 - February 2, 1989).

[2] K.C. Park and W. K. Belvin, *Stability and Implementation of Partitioned CSI Solution Procedures*, 30th Structures, Structural Dynamics and Materials Conference, Mobile, Alabama (April 3-5, 1989).

[3] W. K. Belvin, *Simulation and Interdisciplinary Design Methodology for Control–Structure Interaction Systems*, Ph.D. thesis, University of Colorado at Boulder (August 1989).

[4] K.C. Park and W.K. Belvin, *Discrete Integration of Continuous Kalman Filtering Equations for Time Invariant Second–Order Structural Systems*, AIAA Guidance, Control and Navigation Conference, AIAA 90-3387, Portland, Or., Aug. 1990.

[5] R. Quan, *Numerical Simulation of Large Actively Controlled Space Structures*, Ph.D. thesis, University of Colorado at Boulder (May 1991).

[6] M. Geradin, M. Hogge, and G. Robert, *Time Integration of Linear and Nonlinear Dynamic Problems*, Aerospace Laboratory of the University of Liege, Liege, Belgium, Report VA–38 (January 1984).

[7] C.W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall (1971).

[8] G. Golub and C.F. Van Loan, *Matrix Computations*, Johns Hopkins University Press (1989).

[9] K. Schaumburg, J. Wasniewski, and Z. Zlatev, *The Use of Sparse Matrix Technique in the Numerical Integration of Stiff Systems of Linear Ordinary Differential Equations*, Computers and Chemistry, Vol. 4, p. 1-12, (1980).

[10] O.C. Zienkiewicz, *The Finite Element Method (Fourth Edition)*, McGraw – Hill (1989).

[11] Richard Pappa, *CSI Testbed User's Guide*, Spacecraft Dynamics Branch, Structural Dynamics Division, NASA Langley Research Center, Hampton, VA (March 1989).

[12] R. Quan and M. Balas, *Numerical Simulation of Actively Controlled Space Structures*, The Proceedings of The NASA–UCLA Workshop on Computational Techniques in Identification and Control of Flexible Flight Structures, Lake Arrowhead, Ca., Nov. 2-4, 1989.

Acknowledgments

This work was done under Professor Mark J. Balas at the University of Colorado at Boulder. Thanks goes out to the Center for Space Structures and Controls, for the use of their computer facilities. In addition, the financial support through the NASA Center for Space Construction at the University of Colorado – Boulder and through the Century XXI fellowship program is gratefully acknowledged.

Spatial Operator Algebra for Flexible Multibody Dynamics

A. Jain and G. Rodriguez
Jet Propulsion Laboratory/California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA 91109

Abstract

This paper presents an approach to modeling the dynamics of flexible multibody systems such as flexible spacecraft and limber space robotic systems. A large number of degrees of freedom and complex dynamic interactions are typical in these systems. This paper uses spatial operators to develop efficient recursive algorithms for the dynamics of these systems. This approach very efficiently manages complexity by means of a hierarchy of mathematical operations.

1. Introduction

A wide range of complex mechanical systems can be modeled as a set of hinge-connected flexible and rigid bodies. This paper presents an approach to modeling the dynamics of such systems that uses spatial operators. This approach very efficiently manages complexity by means of a hierarchy of mathematical operations. The highest level in this hierarchy consists of spatial operators which relate velocities, accelerations, and forces between distinct points in the system. At lower levels, each spatial operator is decomposed easily into detailed spatially recursive algorithms to do computation. The recursive algorithms are cast within the highly developed framework of filtering and smoothing theory. Algorithms which are quite popular in state estimation theory for discrete-time systems can now be applied to conduct spatially recursive operations essential in multibody dynamics. The main focus is on serial chains, but extensions to general topologies are also described. Comparison of computational costs illustrates the efficiency of the recursive algorithms.

This paper uses spatial operators [1,2] to develop efficient recursive algorithms for flexible multibody systems for such applications as flexible spacecraft and limber space robotic systems. A large number of degrees of freedom and complex dynamic interactions are typical in these systems. The main contributions of the paper are: (1) high-level architectural understanding of the mass matrix and its inverse, (2) high-level expressions which can be easily implemented with spatial Kalman filtering and smoothing, (3) efficient inverse and forward dynamics recursive algorithms, and (4) analysis of computational cost of the new algorithms. This adds to the rapidly developing body of research in the recursive dynamics of flexible multibody systems [3–5].

2. Equations of Motion

Equations of motion are developed for a serial system formed by N articulated flexible bodies. Recursive relationships between the modal velocities, accelerations and forces are developed. Spatial operators express these relationships compactly to obtain what is referred to here as the Newton-Euler mass matrix factorization.

Each flexible body has a *lumped mass* model formed by a set of nodal rigid bodies. Such models are typically developed using standard finite element analysis. The k^{th} body has $n_s(k)$ nodes. The j^{th} node on the k^{th} body is called the j_k^{th} node. There is a *body reference frame* $\mathcal{F}_k$ for the k^{th} body. Deformation of the nodes on the body is described with respect to this body reference frame, while the rigid–body motion of the k^{th} body is characterized by the motion of frame $\mathcal{F}_k$.

The 6-dimensional *spatial deformation* (slope plus translational) of node j_k (with respect to frame $\mathcal{F}_k$) is $u(j_k) \in \Re^6$. The overall deformation field for the k^{th} body is the vector $u(k) = \text{col}\{u(j_k)\} \in \Re^{6n_s(k)}$. The vector from frame $\mathcal{F}_k$ to the reference frame on node j_k is $l(k, j_k) \in \Re^3$.

The spatial inertia of the j^{th} node is

$$M_s(j_k) = \begin{pmatrix} \mathcal{J}(j_k) & m(j_k)\tilde{p}(j_k) \\ -m(j_k)\tilde{p}(j_k) & m(j_k)I \end{pmatrix} \in \Re^{6\times 6} \tag{1}$$

where $\mathcal{J}(j_k)$, $p(j_k)$ and $m(j_k)$ are the inertia tensor about the node reference frame, the vector from the node reference frame to its center of mass, and the mass, respectively, for the j^{th} node on the k^{th} body. The *structural mass matrix* for the k^{th} body $M_s(k)$ is the block diagnal matrix

$$M_s(k) = \text{diag}\Big\{M_s(j_k)\Big\} \in \Re^{6n_s(k)\times 6n_s(k)} \tag{2}$$

The *structural stiffness matrix* is denoted $K_s(k) \in \Re^{6n_s(k)\times 6n_s(k)}$. For a 3-vector x, there is a corresponding cross product matrix $\tilde{x}$. Both $M_s(k)$ and $K_s(k)$ are typically generated using finite element analysis.

As in Figure 1, the bodies in the serial chain are numbered in increasing order from tip to base. The

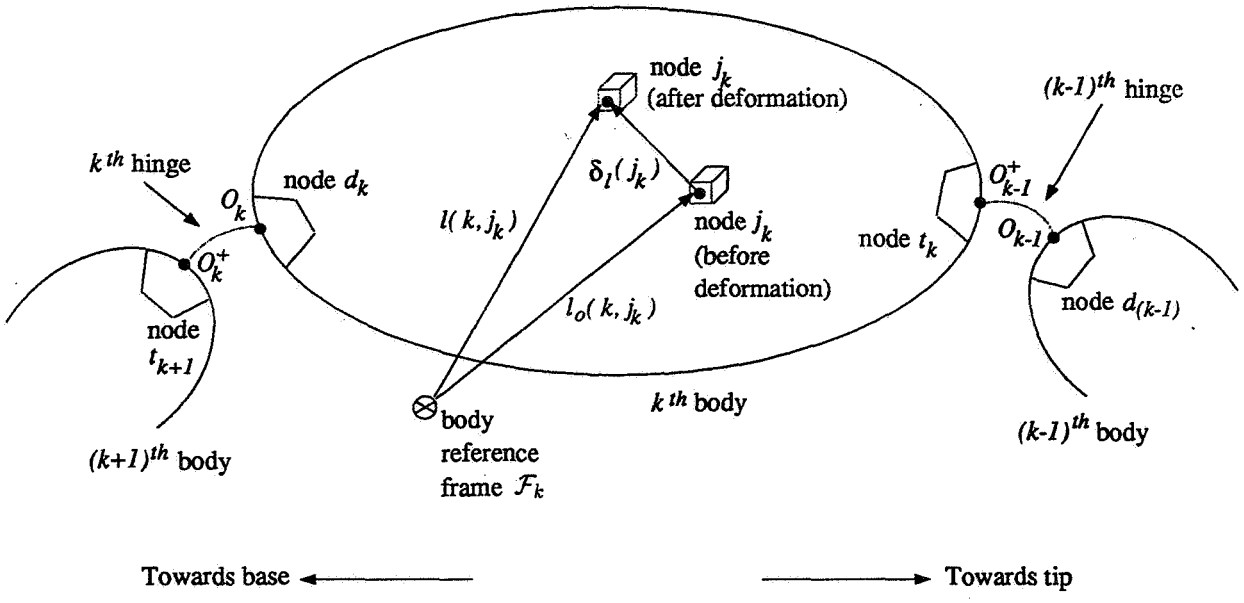


Figure 1: Illustration of links and hinges in a flexible serial multi-body system

terms *inboard (outboard)* denote the direction along the serial chain towards (away from) the base body. The k^{th} body is attached on the inboard side to the $(k+1)^{th}$ body by the k^{th} hinge, and on the outboard side to the $(k-1)^{th}$ body by the $(k-1)^{th}$ hinge. On the k^{th} body, the node to which the outboard hinge (the $(k-1)^{th}$ hinge) is attached is node t_k, while the node to which the inboard hinge (the k^{th} hinge) is attached is node d_k. The k^{th} hinge couples nodes d_k and t_{k+1}. Attached to each of these nodes are the k^{th} hinge reference frames $\mathcal{O}_k$ and $\mathcal{O}_k^+$ respectively. The number of degrees of freedom (dofs) for the k^{th} hinge is $n_r(k)$. The vector of configuration variables for the k^{th} hinge is $\theta(k) \in \Re^{n_r(k)}$, while the hinge's vector of generalized speeds is $\beta(k) \in \Re^{n_r(k)}$. In general, when there are nonholonomic hinge constraints, the dimensionality of $\beta(k)$ may be less than that of $\theta(k)$. For convenience, and without any loss in generality, it is assumed here that the dimensions of the vectors $\theta(k)$ and $\beta(k)$ are equal. In most situations, $\beta(k)$ is simply $\dot{\theta}$. However there are many cases where the use of quasi-coordinates simplifies the dynamical equations of motion, and there

may be a better choice for $\beta(k)$. The relative spatial velocity $\Delta_V(k)$ across the hinge is $H^*(k)\beta(k)$, where $H^*(k)$ is the joint map matrix for the k^{th} hinge.

A set of $n_m(k)$ assumed modes is chosen for the k^{th} body. Let $\Pi_r^j(k) \in \Re^6$ be the *modal spatial displacement vector* at the j_k^{th} node for the r^{th} mode. The *modal spatial displacement influence vector* $\Pi^j(k) \in \Re^{6\times n_m(k)}$ for the j_k^{th} node and the *modal matrix* $\Pi(k) \in \Re^{6n_s(k)\times n_m(k)}$ for the k^{th} body are

$$\Pi^j(k) = \left[\Pi_1^j(k), \cdots, \Pi_{n_m(k)}^j(k)\right] \quad \text{and} \quad \Pi(k) = \text{col}\left\{\Pi^j(k)\right\}$$

The r^{th} column of $\Pi(k)$ is $\Pi_r(k)$, which defines the mode shape for the r^{th} assumed mode for the k^{th} body. Let $\eta(k) \in \Re^{n_m(k)}$ be the vector of modal deformation variables for the k^{th} body. The spatial deformation of node j_k and the spatial deformation field $u(k)$ for the k^{th} body are

$$u(j_k) = \Pi^j(k)\eta(k) \quad \text{and} \quad u(k) = \Pi(k)\eta(k) \tag{3}$$

Note that for cantilever modes

$$\Pi_r^d(k) = 0 \quad \text{and} \quad r = 1 \cdots n_m(k) \tag{4}$$

The vector of *generalized configuration variables* $\vartheta(k)$ and the vector of *generalized speeds* $\chi(k)$ for the k^{th} body are

$$\vartheta(k) \triangleq \begin{pmatrix} \eta(k) \\ \theta(k) \end{pmatrix} \in \Re^{\mathcal{N}(k)} \quad \text{and} \quad \chi(k) \triangleq \begin{pmatrix} \dot{\eta}(k) \\ \beta(k) \end{pmatrix} \in \Re^{\mathcal{N}(k)} \tag{5}$$

where $\mathcal{N}(k) \triangleq n_m(k) + n_r(k)$. The overall vectors of *generalized configuration variables* ϑ and *generalized speeds* χ for the serial multibody system are

$$\vartheta \triangleq \text{col}\left\{\vartheta(k)\right\} \in \Re^{\mathcal{N}} \quad \text{and} \quad \chi \triangleq \text{col}\left\{\chi(k)\right\} \in \Re^{\mathcal{N}} \tag{6}$$

where $\mathcal{N} \triangleq \sum_{k=1}^{N} \mathcal{N}(k)$. The number of overall degrees of freedom for the multibody system is $\mathcal{N}$. The *state* of the multibody system is defined by the pair of vectors $\{\vartheta, \chi\}$. For a given system state $\{\vartheta, \chi\}$, the equations of motion relate the *generalized accelerations* $\dot{\chi}$ and *generalized forces* $T \in \Re^{\mathcal{N}}$. The *inverse dynamics* problem is to compute the generalized forces T for a prescribed set of *generalized accelerations* $\dot{\chi}$. Conversely, the *forward dynamics* problem is to compute the generalized accelerations $\dot{\chi}$ from the generalized forces T.

2.1 Recursive Propagation of Velocities

Let $V(k)$ be the spatial velocity of the k^{th} body reference frame $\mathcal{F}_k$, i.e.,

$$V(k) = \begin{pmatrix} \omega(k) \\ v(k) \end{pmatrix} \in \Re^6$$

where $\omega(k)$ and $v(k)$ are the angular and linear velocities respectively of $\mathcal{F}_k$. The spatial velocity $V_s(t_{k+1}) \in \Re^6$ of node t_{k+1} (on the inboard side of the k^{th} hinge) is related to the spatial velocity $V(k+1)$ of the $(k+1)^{th}$ body reference frame $\mathcal{F}_{k+1}$, and the modal deformation variable rates $\dot{\eta}(k+1)$:

$$V_s(t_{k+1}) \;=\; \phi^*(k+1, t_{k+1})V(k+1) + \Pi^t(k+1)\dot{\eta}(k+1) \tag{7}$$

The spatial transformation operator $\phi(x,y) \in \Re^{6\times 6}$ is

$$\phi(x,y) = \begin{pmatrix} I & \tilde{l}(x,y) \\ 0 & I \end{pmatrix} \tag{8}$$

where $l(x,y) \in \Re^3$ is the vector between the points x and y. Note the group property

$$\phi(x,y)\phi(y,z) = \phi(x,z)$$

for arbitrary points x, y and z. As in Eq. (7), and all through this paper, the index k will be used to refer to both the k^{th} body as well as to the k^{th} body reference frame $\mathcal{F}_k$ with the specific usage coming from the context. For instance, $V(k)$ and $\phi(k,t_k)$ are the same as $V(\mathcal{F}_k)$ and $\phi(\mathcal{F}_k,t_k)$ respectively.

The spatial velocity $V(\mathcal{O}_k^+)$ of frame $\mathcal{O}_k^+$ (on the inboard side of the k^{th} hinge) is related to $V_s(t_{k+1})$ by

$$V(\mathcal{O}_k^+) = \phi^*(t_{k+1},\mathcal{O}_k)V_s(t_{k+1}) \tag{9}$$

The relative spatial velocity $\Delta_V(k)$ across the k^{th} hinge is $H^*(k)\beta(k)$, and so the spatial velocity $V(\mathcal{O}_k)$ of frame $\mathcal{O}_k$ on the outboard side of the k^{th} hinge is

$$V(\mathcal{O}_k) = V(\mathcal{O}_k^+) + H^*(k)\beta(k) \tag{10}$$

The spatial velocity $V(k)$ of the k^{th} body reference frame is

$$V(k) = \phi^*(\mathcal{O}_k,k)V(\mathcal{O}_k) - \mathring{u}(d_k) = \phi^*(\mathcal{O}_k,k)\left[V(\mathcal{O}_k) - \Pi^d(k)\dot{\eta}(k)\right] \tag{11}$$

Eq. (7), Eq. (9), Eq. (10) and Eq. (11) together imply

$$\begin{aligned} V(k) &= \phi^*(k+1,k)V(k+1) + \phi^*(t_{k+1},k)\Pi^t(k+1)\dot{\eta}(k+1) \\ &\quad +\phi^*(\mathcal{O}_k,k)\left[H^*(k)\beta(k) - \Pi^d(k)\dot{\eta}(k)\right] \end{aligned} \tag{12}$$

Thus, with $\overline{\mathcal{N}}(k) \triangleq n_m(k) + 6$ and Eq. (12), the *modal spatial velocity* $V_m(k) \in \Re^{\overline{\mathcal{N}}(k)}$ for the k^{th} body is

$$V_m(k) \triangleq \begin{pmatrix} \dot{\eta}(k) \\ V(k) \end{pmatrix} = \Phi^*(k+1,k)V_m(k+1) + \mathcal{H}^*(k)\chi(k) \in \Re^{\overline{\mathcal{N}}(k)} \tag{13}$$

where the *interbody transformation operator* $\Phi(.,.)$ and the *modal joint map matrix* $\mathcal{H}(k)$ are

$$\Phi(k+1,k) \triangleq \begin{pmatrix} 0 & [\Pi^t(k+1)]^*\phi(t_{k+1},k) \\ 0 & \phi(k+1,k) \end{pmatrix} \in \Re^{\overline{\mathcal{N}}(k+1)\times\overline{\mathcal{N}}(k)} \tag{14}$$

$$\mathcal{H}(k) \triangleq \begin{pmatrix} I & -[\Pi^d_{\mathcal{F}}(k)]^* \\ 0 & H_{\mathcal{F}}(k) \end{pmatrix} \in \Re^{\mathcal{N}(k)\times\overline{\mathcal{N}}(k)} \tag{15}$$

where

$$H_{\mathcal{F}}(k) \triangleq H(k)\phi(\mathcal{O}_k,k) \in \Re^{n_r(k)\times 6}, \quad \text{and} \quad \Pi^d_{\mathcal{F}}(k) \triangleq \phi^*(\mathcal{O}_k,k)\Pi^d(k) \in \Re^{6\times\mathcal{N}(k)}$$

Note that

$$\Phi(k+1,k) = \mathcal{A}(k+1)\mathcal{B}(k+1,k) \tag{16}$$

where

$$\mathcal{A}(k) \triangleq \begin{pmatrix} [\Pi^t(k)]^* \\ \phi(k,t_k) \end{pmatrix} \in \Re^{\overline{\mathcal{N}}(k)\times 6} \quad \text{and} \quad \mathcal{B}(k+1,k) \triangleq [0, \quad \phi(t_{k+1},k)] \in \Re^{6\times\overline{\mathcal{N}}(k)} \tag{17}$$

Also, the modal joint map matrix $\mathcal{H}(k)$ can be partitioned as

$$\mathcal{H}(k) = \begin{pmatrix} \mathcal{H}_f(k) \\ \mathcal{H}_r(k) \end{pmatrix} \in \Re^{\mathcal{N}(k)\times\overline{\mathcal{N}}(k)} \tag{18}$$

where

$$\mathcal{H}_f(k) \triangleq \left[I, \ -[\Pi_{\mathcal{F}}^d(k)]^*\right] \in \Re^{n_m(k)\times\overline{\mathcal{N}}(k)} \quad \text{and} \quad \mathcal{H}_r(k) \triangleq [0, \ H(k)\phi(\mathcal{O}_k,k)] \in \Re^{n_r(k)\times\overline{\mathcal{N}}(k)} \tag{19}$$

With $\overline{\mathcal{N}} = \sum_{k=1}^{N}\overline{\mathcal{N}}(k)$, the spatial operator $\mathcal{E}_\Phi$ is defined as

$$\mathcal{E}_\Phi \triangleq \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ \Phi(2,1) & 0 & \dots & 0 & 0 \\ 0 & \Phi(3,2) & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \Phi(N,N-1) & 0 \end{pmatrix} \in \Re^{\overline{\mathcal{N}}\times\overline{\mathcal{N}}} \tag{20}$$

Note that $\mathcal{E}_\Phi$ is nilpotent (i.e. $\mathcal{E}_\Phi^N = 0$) and define the spatial operator Φ as

$$\Phi \ \triangleq \ [I-\mathcal{E}_\Phi]^{-1} = I + \mathcal{E}_\Phi + \cdots + \mathcal{E}_\Phi^{N-1} = \begin{pmatrix} I & 0 & \dots & 0 \\ \Phi(2,1) & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \Phi(N,1) & \Phi(N,2) & \dots & I \end{pmatrix} \in \Re^{\overline{\mathcal{N}}\times\overline{\mathcal{N}}} \tag{21}$$

where

$$\Phi(i,j) \triangleq \Phi(i,i-1)\ \cdots\ \Phi(j+1,j) \ \text{ for } \ i>j$$

Also define the spatial operator $\mathcal{H} \triangleq \text{diag}\{\mathcal{H}(k)\} \in \Re^{\mathcal{N}\times\overline{\mathcal{N}}}$. It follows that

$$V_m = \Phi^*\mathcal{H}^*\chi \tag{22}$$

where $V_m \triangleq \text{col}\{V_m(k)\} \in \Re^{\overline{\mathcal{N}}}$.

2.2 Modal Mass Matrix for a Single Body

An expression for the modal mass matrix of the k^{th} body is derived. Denote by $V_s(j_k) \in \Re^6$ the spatial velocity of node j_k. $V_s(k) \triangleq \text{col}\{V_s(j_k)\} \in \Re^{6n_s(k)}$ is the vector of all nodal spatial velocities for the k^{th} body. It follows (as in Eq. (7)) that

$$V_s(k) = B^*(k)V(k) + \mathring{u}(k) = [\Pi(k), \ B^*(k)]V_m(k) \tag{23}$$

where

$$B(k) \triangleq [\phi(k,1_k), \phi(k,2_k), \cdots, \phi(k,n_s(k))] \in \Re^{6\times 6n_s(k)} \tag{24}$$

Since $M_s(k)$ is the *structural mass matrix* of the k^{th} body, the kinetic energy of the k^{th} body is

$$\frac{1}{2}V_s^*(k)M_s(k)V_s(k) = \frac{1}{2}V_m^*(k)M_m(k)V_m(k)$$

where

$$M_m(k) \ \triangleq \ \begin{pmatrix} \Pi^*(k) \\ B(k) \end{pmatrix} M_s(k)[\Pi(k), \ B^*(k)] = \begin{pmatrix} M_m^{ff}(k) & M_m^{fr}(k) \\ M_m^{rf}(k) & M_m^{rr}(k) \end{pmatrix} \in \Re^{\overline{\mathcal{N}}(k)\times\overline{\mathcal{N}}(k)} \tag{25}$$

Corresponding to the generalized speed vector $\chi(k)$, $M_m(k)$ is the *modal mass matrix* of the k^{th} body. In the block partitioning in Eq. (25), the superscripts f and r denote the *flexible* and *rigid* blocks respectively. Thus $M_m^{ff}(k)$ represents the flex/flex coupling block, while $M_m^{fr}(k)$ the flex/rigid coupling block, of $M_m(k)$. Note that $M_m^{rr}(k)$ is precisely the rigid body spatial inertia of the k^{th} body. Indeed, $M_m(k)$ reduces to the rigid body spatial inertia when the body flexibility is ignored, i.e., no modes are used, since in this case $n_m(k) = 0$ (and $\Pi(k)$ is null).

Since the vector $l(k, j_k)$ from $\mathcal{F}_k$ to node j_k depends on the deformation of the node, the operator $B(k)$ is also deformation dependent. From Eq. (25) it follows that while the block $\mathcal{M}_m^{ff}(k)$ is deformation independent, both the blocks $M_m^{fr}(k)$ and $M_m^{rr}(k)$ are deformation dependent. The detailed expression for the modal mass matrix can be defined using *modal integrals* which are computed as a part of the finite–element structural analysis of the flexible bodies. These expressions for the modal integrals and the modal mass matrix of the k^{th} body can be found in [6]. Often the deformation dependent parts of the modal mass matrix are ignored, and free-free eigen–modes are used for the assumed modes $\Pi(k)$. When this is the case, $M_m^{fr}(k)$ is zero and $M_m^{ff}(k)$ is block diagonal.

2.3 Recursive Propagation of Accelerations

Differentiation of the velocity recursion equation, Eq. (13), results in the following recursive expression for the *modal spatial acceleration* $\alpha_m(k) \in \Re^{\overline{\mathcal{N}}(k)}$ for the k^{th} body:

$$\alpha_m(k) \triangleq \dot{V}_m(k) = \begin{pmatrix} \ddot{\eta}(k) \\ \alpha(k) \end{pmatrix} = \Phi^*(k+1,k)\alpha_m(k+1) + \mathcal{H}^*(k)\dot{\chi}(k) + a_m(k) \tag{26}$$

where $\alpha(k) = \dot{V}(k)$, and the Coriolis and centrifugal acceleration term $a_m(k) \in \Re^{\overline{\mathcal{N}}(k)}$ is

$$a_m(k) = \frac{\mathrm{d}\Phi^*(k+1,k)}{\mathrm{d}t} V_m(k+1) + \frac{\mathrm{d}\mathcal{H}^*(k)}{\mathrm{d}t}\chi(k) \tag{27}$$

The detailed expressions for $a_m(k)$ can be found in [6]. Defining $a_m = \mathrm{col}\{a_m(k)\} \in \Re^{\overline{\mathcal{N}}}$ and $\alpha_m = \mathrm{col}\{\alpha_m(k)\} \in \Re^{\overline{\mathcal{N}}}$, Eq. (26) can be reexpressed using spatial operators in the form

$$\alpha_m = \Phi^*(\mathcal{H}^*\dot{\chi} + a_m) \tag{28}$$

The vector of spatial accelerations of all the nodes for the k^{th} body, $\alpha_s(k) \triangleq \mathrm{col}\{\alpha_s(j_k)\}$ $\in \Re^{6n_s(k)}$, is obtained by differentiating Eq. (23):

$$\alpha_s(k) = \dot{V}_s(k) = [\Pi(k), \quad B^*(k)]\alpha_m(k) + a(k) \tag{29}$$

where

$$a(k) \triangleq \mathrm{col}\{a(j_k)\} = \frac{\mathrm{d}[\Pi(k), \quad B^*(k)]}{\mathrm{d}t} V_m(k) \in \Re^{6n_s(k)} \tag{30}$$

2.4 Recursive Propagation of Forces

The equations of motion for the k^{th} body are now developed. Let $f(k-1) \in \Re^6$ denote the effective spatial force of interaction, referred to frame $\mathcal{F}_{k-1}$, between the k^{th} and $(k-1)^{th}$ bodies across the $(k-1)^{th}$ hinge. Recall that the $(k-1)^{th}$ hinge is between node t_k on the k^{th} body and node d_{k-1} on the $(k-1)^{th}$ body. With $f_s(j_k) \in \Re^6$ denoting the spatial force at a node j_k, the force balance equation for node t_k is

$$f_s(t_k) \;=\; \phi(t_k, k-1)f(k-1) + M_s(t_k)\alpha_s(t_k) + b(t_k) + f_K(t_k) \tag{31}$$

For all nodes other than node t_k on the k^{th} body, the force balance equation is

$$f_s(j_k) = M_s(j_k)\alpha_s(j_k) + b(j_k) + f_K(j_k) \tag{32}$$

In the above, $f_K(k) = K_s(k)u(k) \in \Re^{6n_s(k)}$ is the vector of spatial elastic strain forces for the nodes on the k^{th} body, while $b(j_k) \in \Re^6$ is the spatial gyroscopic force for node j_k and is given by

$$b(j_k) = \begin{pmatrix} \tilde{\omega}(j_k)\mathcal{J}(j_k)\omega(j_k) \\ m(j_k)\tilde{\omega}(j_k)\tilde{\omega}(j_k)p(j_k) \end{pmatrix} \in \Re^6 \tag{33}$$

where $\omega(j_k) \in \Re^3$ denotes the angular velocity of node j_k. Define

$$C(k,k-1) \triangleq \begin{pmatrix} 0 \\ \vdots \\ \phi(t_k,k-1) \\ \vdots \\ 0 \end{pmatrix} \in \Re^{6n_s(k)\times 6} \quad \text{and} \quad b(k) \triangleq \text{col}\Big\{b(j_k)\Big\} \in \Re^{6n_s(k)} \tag{34}$$

Eq. (31) and Eq. (32) imply

$$f_s(k) \;=\; C(k,k-1)f(k-1) + M_s(k)\alpha_s(k) + b(k) + K_s(k)u(k) \tag{35}$$

where $f_s(k) \triangleq \text{col}\Big\{f_s(j_k)\Big\} \in \Re^{6n_s(k)}$. Noting that

$$f(k) = B(k)f_s(k) \tag{36}$$

and using the principle of virtual work, it follows from Eq. (23) that the *modal spatial forces* $f_m(k) \in \Re^{\overline{N}(k)}$ for the k^{th} body are

$$f_m(k) \;\triangleq\; \begin{pmatrix} \Pi^*(k) \\ B(k) \end{pmatrix} f_s(k) = \begin{pmatrix} \Pi^*(k)f_s(k) \\ f(k) \end{pmatrix} \tag{37}$$

Premultiplying Eq. (35) by $\begin{pmatrix} \Pi^*(k) \\ B(k) \end{pmatrix}$ and using Eq. (25), Eq. (29), and Eq. (37) leads to the following recursive relationship for the modal spatial forces:

$$f_m(k) \;=\; \Phi(k,k-1)f_m(k-1) + M_m(k)\alpha_m(k) + b_m(k) + K_m(k)\vartheta(k) \tag{38}$$

where

$$b_m(k) \triangleq \begin{pmatrix} \Pi^*(k) \\ B(k) \end{pmatrix} [b(k) + M_s(k)a(k)] \in \Re^{\overline{N}(k)} \tag{39}$$

and the *modal stiffness matrix*

$$K_m(k) \triangleq \begin{pmatrix} \Pi^*(k)K_s(k)\Pi(k) & 0 \\ 0 & 0 \end{pmatrix} \in \Re^{\overline{N}(k)\times\overline{N}(k)} \tag{40}$$

The expression for $K_m(k)$ in Eq. (40) uses the fact that the columns of $B^*(k)$ are the *deformation dependent* rigid body modes for the k^{th} body, and hence they do not contribute to its elastic strain energy. When a deformation dependent structural stiffness matrix $K_s(k)$ is used,

$$K_s(k)B^*(k) = 0 \tag{41}$$

However, common practice (followed in this paper) uses a constant, deformation–independent structural stiffness matrix. This leads to the apparently anomalous situation wherein Eq. (41) does not hold exactly. All these fictitious extra terms on the left–hand side of Eq. (41) are commonly ignored.

The velocity–dependent bias term $b_m(k)$ is formed using modal integrals generated by standard finite–element programs, and a detailed expression for it is given in [6]. From Eq. (38), the operator expression for the modal spatial forces $f_m \triangleq \text{col}\{f_m(k)\} \in \Re^{\overline{N}}$ for all the bodies in the chain is

$$f_m = \Phi(M_m \alpha_m + b_m + K_m \vartheta) \tag{42}$$

where

$$\mathcal{M}_m \triangleq \text{diag}\{M_m(k)\} \in \Re^{\overline{N}\times\overline{N}},\ K_m \triangleq \text{diag}\{K_m(k)\} \in \Re^{\overline{N}\times\overline{N}},\ \text{and}\ b_m \triangleq \text{col}\{b_m(k)\} \in \Re^{\overline{N}}$$

From the principle of virtual work, the *generalized forces* vector $T \in \Re^{\mathcal{N}}$ for the multibody system is

$$T = \mathcal{H} f_m \tag{43}$$

2.5 Operator Expression for the System Mass Matrix

Collection of the operator expressions in Eq. (22), Eq. (28), Eq. (42) and Eq. (43) leads to:

$$\begin{aligned} V_m &= \Phi^* \mathcal{H}^* \chi \\ \alpha_m &= \Phi^*(\mathcal{H}^* \dot{\chi} + a_m) \\ f_m &= \Phi(M_m \alpha_m + b_m + K_m \vartheta) = \Phi M_m \Phi^* \mathcal{H}^* \dot{\chi} + \Phi(M_m \Phi^* a_m + b_m + K_m \vartheta) \\ T &= \mathcal{H} f_m = \mathcal{H}\Phi M_m \Phi^* \mathcal{H}^* \dot{\chi} + \mathcal{H}\Phi(M_m \Phi^* a_m + b_m) \\ &= \mathcal{M}\dot{\chi} + \mathcal{C} \end{aligned} \tag{44}$$

where

$$\mathcal{M} \triangleq \mathcal{H}\Phi M_m \Phi^* \mathcal{H}^* \in \Re^{\mathcal{N}\times\mathcal{N}} \quad \text{and} \quad \mathcal{C} \triangleq \mathcal{H}\Phi(M_m \Phi^* a_m + b_m + K_m \vartheta) \in \Re^{\mathcal{N}} \tag{45}$$

Here $\mathcal{M}$ is the system mass matrix. The expression $\mathcal{H}\Phi M_m \Phi^* \mathcal{H}^*$ is referred to as the *Newton-Euler Operator Factorization* of the mass matrix. The term $\mathcal{C}$ is the vector of Coriolis, centrifugal, and elastic forces for the system.

The operator expressions for $\mathcal{M}$ and $\mathcal{C}$ are identical in form to those for rigid multibody systems (see [1, 7]). This similarity is extremely useful in the extension of recursive algorithms from rigid multibody systems to flexible multibody systems.

3. Composite Body Forward Dynamics Algorithm

The forward dynamics problem for a multibody system requires computing the generalized accelerations $\dot{\chi}$ for a given vector of generalized forces T and state of the system $\{\vartheta, \chi\}$. The *composite body forward dynamics algorithm* described below consists of (a) computing the system mass matrix $\mathcal{M}$, (b) computing the bias vector $\mathcal{C}$, and (c) solving the linear matrix equation for $\dot{\chi}$:

$$\mathcal{M}\dot{\chi} = T - \mathcal{C} \tag{46}$$

Section 4 describes the recursive articulated body forward dynamics algorithm that does not require the explicit computation of either $\mathcal{M}$ or $\mathcal{C}$.

Lemma 3.1: Define the *composite body inertias* $R(k) \in \Re^{\overline{N}(k)\times\overline{N}(k)}$ recursively for all the bodies in the serial chain as follows:

$$\left\{ \begin{array}{l} \qquad R(0) = 0 \\ \textbf{for k} = \mathbf{1 \cdots N} \\ \quad R(k) = \Phi(k, k-1) R(k-1) \Phi^*(k, k-1) + M_m(k) \\ \textbf{end loop} \end{array} \right. \tag{47}$$

Also define $R \triangleq \text{diag}\{R(k)\} \in \Re^{\overline{N}\times\overline{N}}$. Then, observe the following spatial operator decomposition where $\tilde{\Phi} \triangleq \Phi - I$:

$$\Phi M_m \Phi^* = R + \tilde{\Phi} R + R\tilde{\Phi}^* \tag{48}$$

Physically, $R(k)$ is the modal mass matrix of the composite body formed from all the bodies outboard of the k^{th} hinge by freezing all their (deformation plus hinge) degrees of freedom. It follows from Eq. (45) and Lemma 3.1 that

$$\mathcal{M} = \mathcal{H}\Phi M_m \Phi^* \mathcal{H}^* = \mathcal{H}R\mathcal{H}^* + \mathcal{H}\tilde{\Phi}R\mathcal{H}^* + \mathcal{H}R\tilde{\Phi}^*\mathcal{H}^* \tag{49}$$

Note that the three terms on the right of Eq. (49) are block diagonal, block lower triangular and block upper triangular respectively. The algorithm for computing the mass matrix $\mathcal{M}$ computes these terms recursively. The main recursion proceeds from tip to base, and computes the blocks along the diagonal of $\mathcal{M}$. As each such diagonal element is computed, a new recursion to compute the off–diagonal elements is spawned. Its structure is similar to that of the composite body algorithm for computing the mass matrix of rigid multibody systems (see [8,9]), and is as follows:

$$\left\{\begin{array}{l} R(0) = 0 \\ \textbf{for k} = 1\cdots\textbf{N} \\ \quad R(k) = \Phi(k,k-1)R(k-1)\Phi^*(k,k-1) + M_m(k) \\ \quad\quad = \mathcal{A}(k)\phi(t_k,k-1)R^r(k-1)\phi^*(t_k,k-1)\mathcal{A}^*(k) + M_m(k) \\ \quad X(k) = R(k)\mathcal{H}^*(k) \\ \quad \mathcal{M}_s(k,k) = \mathcal{H}(k)X(k) \\ \qquad \left\{\begin{array}{l} \textbf{for j} = (\textbf{k}+1)\cdots\textbf{N} \\ \qquad X(j) = \Phi(j,j-1)X(j-1) = \mathcal{A}(j)\phi(t_j,j-1)X^r(j-1) \\ \qquad \mathcal{M}(j,k) = \mathcal{M}^*(k,j) = \mathcal{H}(j)X(j) \\ \textbf{end loop} \end{array}\right. \\ \textbf{end loop} \end{array}\right. \tag{50}$$

The structure of the above algorithm for computing the mass matrix closely resembles the composite rigid body algorithm for computing the mass matrix of rigid multibody systems [8,9]. Like the latter, it is also highly efficient. Additional computational simplifications of the algorithm arising from the sparsity of both $\mathcal{H}_f(k)$ and $\mathcal{H}_r(k)$ are easy to incorporate.

4. Factorization and Inversion of the Mass Matrix

An operator factorization of the system mass matrix $\mathcal{M}$, referred to as the *Innovations Operator Factorization*, is derived. This factorization is an alternative to the Newton–Euler factorization in Eq. (45). In contrast with the latter, the factors in the Innovations factorization are square and invertible. Operator expressions for the inverse of these factors lead to an operator expression for the inverse of the mass matrix. Use of further operator identities results in the recursive articulated body forward dynamics algorithm in Section 5. The operator factorization and inversion results here closely resemble those for rigid multibody systems (see [1]).

The following recursive algorithm defines some required articulated body quantities. This algorithm

has the structure of the Riccati equation of Kalman filtering theory [9]:

$$\left\{\begin{array}{rcl} P^+(0) = 0 & & \\ \textbf{for k} = 1\cdots\textbf{N} & & \\ P(k) & = & \Phi(k,k-1)P^+(k-1)\Phi^*(k,k-1) + M_m(k) \in \Re^{\overline{\mathcal{N}}(k)\times\overline{\mathcal{N}}(k)} \\ D(k) & = & \mathcal{H}(k)P(k)\mathcal{H}^*(k) \in \Re^{\mathcal{N}(k)\times\mathcal{N}(k)} \\ G(k) & = & P(k)\mathcal{H}^*(k)D^{-1}(k) \in \Re^{\overline{\mathcal{N}}(k)\times\mathcal{N}(k)} \\ K(k+1,k) & = & \Phi(k+1,k)G(k) \in \Re^{\overline{\mathcal{N}}(k)\times\mathcal{N}(k)} \\ \overline{\tau}(k) & = & I - G(k)\mathcal{H}(k) \in \Re^{\overline{\mathcal{N}}(k)\times\overline{\mathcal{N}}(k)} \\ P^+(k) & = & \overline{\tau}(k)P(k) \in \Re^{\overline{\mathcal{N}}(k)\times\overline{\mathcal{N}}(k)} \\ \Psi(k+1,k) & = & \Phi(k+1,k)\overline{\tau}(k) \in \Re^{\overline{\mathcal{N}}(k)\times\overline{\mathcal{N}}(k)} \\ \textbf{end loop} & & \end{array}\right. \tag{51}$$

The operator $P \in \Re^{\overline{\mathcal{N}}\times\overline{\mathcal{N}}}$ is defined as the block diagonal matrix with the k^{th} diagonal element being $P(k)$. The quantities defined in Eq. (51) form the component elements of the following spatial operators:

$$\begin{aligned} D &\triangleq \mathcal{H}P\mathcal{H}^* = \mathrm{diag}\Big\{D(k)\Big\} \in \Re^{\mathcal{N}\times\mathcal{N}} \\ G &\triangleq P\mathcal{H}^*D^{-1} = \mathrm{diag}\Big\{G(k)\Big\} \in \Re^{\overline{\mathcal{N}}\times\mathcal{N}} \\ K &\triangleq \mathcal{E}_\Phi G \in \Re^{\overline{\mathcal{N}}\times\mathcal{N}} \\ \overline{\tau} &\triangleq I - G\mathcal{H} = \mathrm{diag}\Big\{\overline{\tau}(k)\Big\} \in \Re^{\overline{\mathcal{N}}\times\overline{\mathcal{N}}} \\ \mathcal{E}_\Psi &\triangleq \mathcal{E}_\Phi\overline{\tau} \in \Re^{\overline{\mathcal{N}}\times\overline{\mathcal{N}}} \end{aligned} \tag{52}$$

The only nonzero block elements of K and $\mathcal{E}_\Psi$ are the elements $K(k+1,k)$ and $\Psi(k+1,k)$ respectively along the first sub-diagonal. The spatial operator G is formed by a set of spatial Kalman gains [9] for a spatially recursive Kalman filter.

As in the case for $\mathcal{E}_\Phi$, $\mathcal{E}_\Psi$ is nilpotent, so the operator Ψ can be defined as

$$\Psi \triangleq (I - \mathcal{E}_\Psi)^{-1} = \begin{pmatrix} I & 0 & \cdots & 0 \\ \Psi(2,1) & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \Psi(N,1) & \Psi(N,2) & \cdots & I \end{pmatrix} \in \Re^{\overline{\mathcal{N}}\times\overline{\mathcal{N}}} \tag{53}$$

where

$$\Psi(i,j) \triangleq \Psi(i,i-1)\ \cdots\ \Psi(j+1,j) \quad \text{for} \quad i > j$$

The structure of the operators $\mathcal{E}_\Psi$ and Ψ is identical to that of the operators $\mathcal{E}_\Phi$ and Φ respectively except that the component elements are now $\Psi(i,j)$ rather than $\Phi(i,j)$. Also, the elements of Ψ have the same semigroup properties as the elements of the operator Φ, and as a consequence, high-level operator expressions involving them can be directly mapped into recursive algorithms, and the explicit computation of the elements of the operator Ψ is not required.

The Innovations Operator Factorization of the mass matrix is defined in the following lemmas established in [1].

Lemma 4.1:

$$\mathcal{M} = [I + \mathcal{H}\Phi K]D[I + \mathcal{H}\Phi K]^* \tag{54}$$

Note that the factor $[I + \mathcal{H}\Phi K] \in \Re^{\mathcal{N}x\mathcal{N}}$ is square, block lower triangular and nonsingular, while D is a block diagonal matrix. This factorization may be regarded as a block LDL^* decomposition of $\mathcal{M}$. The following lemma gives the closed form operator expression for the inverse of the factor $[I + \mathcal{H}\Phi K]$.

Lemma 4.2:

$$[I + \mathcal{H}\Phi K]^{-1} = [I - \mathcal{H}\Psi K] \tag{55}$$

Lemma 4.3:

$$\mathcal{M}^{-1} = [I - \mathcal{H}\Psi K]^* D^{-1} [I - \mathcal{H}\Psi K] \tag{56}$$

Once again, note that the factor $[I - \mathcal{H}\Psi K]$ is square, block lower triangular and nonsingular and so Lemma 4.3 may be regarded as providing a block LDL^* decomposition of $\mathcal{M}^{-1}$. This decomposition however is model-based, in the sense that the physical model of the system is used to conduct computations. This means that every step in the decomposition has a corresponding physical interpretation which adds a substantial amount of insight into the decomposition.

5. Articulated Body Forward Dynamics Algorithm

The operator-based mass matrix inverse leads to a recursive forward dynamics algorithm. The structure of this algorithm is completely identical in form to the articulated body algorithm for serial *rigid* multibody systems. Its structure is that of a Kalman filter and a Bryson-Frazier smoother [9].

The following lemma, established in [1], describes the operator expression for the generalized accelerations $\dot{\chi}$ in terms of the generalized forces T.

Lemma 5.1:

$$\dot{\chi} = [I - \mathcal{H}\Psi K]^* D^{-1} \Big[T - \mathcal{H}\Psi \{ KT + Pa_m + b_m + K_m \vartheta \} \Big] - K^* \Psi^* a_m \tag{57}$$

As in the case of rigid multibody systems [1,10], the direct recursive implementation of Eq. (57) leads to the following recursive forward dynamics algorithm:

$$\left\{ \begin{array}{rcl} & z^+(0) = 0 & \\ \textbf{for k} & = & \mathbf{1 \cdots n} \\ z(k) & = & \Phi(k,k-1) z^+(k-1) + P(k) a_m(k) + b_m(k) + K_m(k) \vartheta(k) \\ \epsilon(k) & = & T(k) - \mathcal{H}(k) z(k) \\ \nu(k) & = & D^{-1}(k) \epsilon(k) \\ z^+(k) & = & z(k) + G(k) \epsilon(k) \\ \textbf{end loop} & & \end{array} \right. \tag{58}$$

$$\left\{ \begin{array}{rcl} & \alpha_m(n+1) = 0 & \\ \textbf{for k} & = & \mathbf{n \cdots 1} \\ \alpha_m^+(k) & = & \Phi^*(k+1,k) \alpha_m(k+1) \\ \dot{\chi}(k) & = & \nu(k) - G^*(k) \alpha_m^+(k) \\ \alpha_m(k) & = & \alpha_m^+(k) + \mathcal{H}^*(k) \dot{\chi}(k) + a_m(k) \\ \textbf{end loop} & & \end{array} \right.$$

All the degrees of freedom for each body are characterized by its joint map matrix $\mathcal{H}^*(.)$ and are processed together at each recursion step in this algorithm. However, by taking advantage of the

sparsity and special structure of the joint map matrix, additional reduction in computational cost is obtained by processing the flexible dofs and the hinge degrees of freedom separately. These simplifications are described in the following sections.

Instead of giving detail, the conceptual approach to separating modal and hinge degrees of freedom is described. First, recall the velocity recursion equation in Eq. (13)

$$V_m(k) = \Phi^*(k+1,k)V_m(k+1) + \mathcal{H}^*(k)\chi(k) \tag{59}$$

and the partitioned form of $\mathcal{H}(k)$ in Eq. (15)

$$\mathcal{H}(k) = \begin{pmatrix} \mathcal{H}_f(k) \\ \mathcal{H}_r(k) \end{pmatrix} \tag{60}$$

Introducing a dummy variable k', rewrite Eq. (59) as

$$\begin{aligned} V_m(k') &= \Phi^*(k+1,k')V_m(k+1) + \mathcal{H}_f^*(k)\dot{\eta}(k) \\ V_m(k) &= \Phi^*(k',k)V_m(k') + \mathcal{H}_r^*(k)\beta(k) \end{aligned} \tag{61}$$

where

$$\Phi(k+1,k') \triangleq \Phi(k+1,k) \quad \text{and} \quad \Phi(k',k) \triangleq I$$

Conceptually, each flexible body is now associated with two bodies. The first one has the same kinematical and mass/inertia properties as the real body and is associated with the flexible degrees of freedom. The second body is a fictitious body, and it is massless and has zero extent. It is associated with the hinge degrees of freedom. The serial chain now contains twice the number of bodies as the original one, with half the new bodies being fictitious. The new $\mathcal{H}^*$ operator has the same number of columns but twice the number of rows as the original $\mathcal{H}^*$ operator. The new Φ operator has twice the number of rows as well as twice the number of columns as the original. An analysis similar to those of the previous sections leads to an operator expression similar to Eq. (57). This implies a recursive forward dynamics algorithm like Eq. (58). However each sweep in the algorithm now contains twice as many steps as the original algorithm. But since each step now processes a smaller number of degrees of freedom, the overall computational cost is reduced.

5.1 Simplified Articulated Body Forward Dynamics Algorithm

The complete recursive *articulated body forward dynamics algorithm* for a serial flexible multibody system follows from recursive implementation of Eq. (57). The algorithm has the following steps: (a) compute the articulated body quantities, (b) do a base-to-tip recursion for the modal spatial velocities $V_m(k)$ and the bias terms $a_m(k)$, $b_m(k)$, and (c) do a tip-to-base recursion followed by a

base-to-tip recursion for the joint accelerations $\dot{\chi}$.

$$
\left\{
\begin{array}{rcl}
 & P_R^+(0) = 0 & \\
\textbf{for k} & = & \mathbf{1 \cdots N} \\
\Gamma(k) & = & \phi(t_k, k-1) P_R^+(k-1) \phi^*(t_k, k-1) \\
P(k) & = & \mathcal{A}(k) \Gamma(k) \mathcal{A}^*(k) + M_m(k) \\
D_f(k) & = & \mathcal{H}_f(k) P(k) \mathcal{H}_f^*(k) \\
\mu(k) & = & [P^{rf}(k), \;\; P^{rr}(k)] \mathcal{H}_f^*(k) \\
g(k) & = & \mu(k) D_f^{-1}(k) \\
P_R(k) & = & P^{rr}(k) - g(k) \mu^*(k) \\
D_R(k) & = & H_{\mathcal{F}}(k) P_R(k) H_{\mathcal{F}}^*(k) \\
G_R(k) & = & P_R(k) H_{\mathcal{F}}^*(k) D_R^{-1}(k) \\
\overline{\tau}_R(k) & = & I - G_R(k) H_{\mathcal{F}}(k) \\
P_R^+(k) & = & \overline{\tau}_R(k) P_R(k) \\
\textbf{end loop} & &
\end{array}
\right.
\tag{62}
$$

$$
\left\{
\begin{array}{rcl}
 & z_R^+(0) = 0 & \\
\textbf{for k} & = & \mathbf{1 \cdots N} \\
z(k) & = & \begin{pmatrix} z_f(k) \\ z_r(k) \end{pmatrix} \\
 & = & \mathcal{A}(k) \phi(t_k, k-1) z_R^+(k-1) + b_m(k) + K_m(k) \vartheta(k) \in \Re^{\overline{\mathcal{N}}(k)} \\
\epsilon_f(k) & = & T_f(k) - z_f(k) + [\Pi_{\mathcal{F}}^d(k)]^* z_r(k) \in \Re^{n_m(k)} \\
\nu_f(k) & = & D_f^{-1}(k) \epsilon_f(k) \in \Re^{n_m(k)} \\
 & & \\
z_R(k) & = & z_r(k) + g(k) \epsilon_f(k) + P_R(k) a_{mR}(k) \in \Re^6 \\
\epsilon_R(k) & = & T_R(k) - H_{\mathcal{F}}(k) z_R(k) \in \Re^{n_r(k)} \\
\nu_R(k) & = & D_R^{-1}(k) \epsilon_R(k) \in \Re^{n_r(k)} \\
z_R^+(k) & = & z_R(k) + G_R(k) \epsilon_R(k) \in \Re^6 \\
\textbf{end loop} & &
\end{array}
\right.
$$

$$
\left\{
\begin{array}{rcl}
 & \alpha_m(N+1) = 0 & \\
\textbf{for k} & = & \mathbf{N \cdots 1} \\
\alpha_R^+(k) & = & \phi^*(t_{k+1}, k) \mathcal{A}^*(k+1) \alpha_m(k+1) \in \Re^6 \\
\dot{\beta}(k) & = & \nu_R(k) - G_R^*(k) \alpha_R^+(k) \in \Re^{n_r(k)} \\
\alpha_R(k) & = & \alpha_R^+(k) + H_{\mathcal{F}}^*(k) \dot{\beta}(k) + a_{mR}(k) \in \Re^6 \\
\ddot{\eta}(k) & = & \nu_f(k) - g^*(k) \alpha_R(k) \in \Re^{n_m(k)} \\
\alpha_m(k) & = & \begin{pmatrix} \ddot{\eta}(k) \\ \alpha_R(k) - \Pi_{\mathcal{F}}^d(k) \ddot{\eta}(k) \end{pmatrix} \in \Re^{\overline{\mathcal{N}}(k)} \\
\textbf{end loop} & &
\end{array}
\right.
\tag{63}
$$

The recursion in Eq. (63) is obtained by carrying out simplifications of the recursions in Eq. (58) in the same manner as described in the previous section for the articulated body quantities.

In contrast with the composite body forward dynamics algorithm of Section 3, this algorithm does not explicitly compute either $\mathcal{M}$ or $\mathcal{C}$. This algorithm is similar to those for rigid multibody systems [1,11].

C-3

6. Computational Cost

The computational costs of the composite body and the articulated body forward dynamics algorithms are compared. For low-spin multibody systems, it has been suggested in [12] that using *ruthlessly linearized models* for each flexible body can lead to significant computational reduction without sacrificing fidelity. These linearized models are considerably less complex than the full nonlinear models and do not require much of the data on modal integrals for the individual flexible bodies. All computational costs given below are based on the use of ruthlessly linearized models and the computationally simplified steps described in [6].

6.1 Computational Cost of the Composite Body Forward Dynamics Algorithm

The composite body forward dynamics algorithm described in Section 3 is based on solving the linear matrix equation

$$\mathcal{M}\dot{\chi} = T - C$$

The computational cost of this forward dynamics algorithm is as follows:

1. The cost of computing $R(k)$ for the k^{th} body by using the algorithm in Eq. (50) is $[48n_m(k) + 90]\text{M} + [n_m^2(k) + \frac{97}{2}n_m(k) + 116]\text{A}$.

2. The contribution of the k^{th} body to the cost of computing $\mathcal{M}$ (excluding cost of $R(k)$'s) using the algorithm in Eq. (50) is $\{k[12n_m^2(k) + 34n_m(k) + 13]\}\ \text{M} + \{k[11n_m^2(k) + 24n_m(k) + 13]\}\ \text{A}$.

3. Setting the generalized accelerations $\dot{\chi} = 0$, the vector C can be obtained by using an inverse dynamics algorithm for computing the generalized forces T. The contribution of the k^{th} body to the computational cost for $C(k)$ is $\{2n_m^2(k) + 54n_m(k) + 206\}\ \text{M} + \{2n_m^2(k) + 50n_m(k) + 143\}\ \text{A}$.

4. The cost of computing $T - C$ is $\{\mathcal{N}\}\ \text{A}$.

5. The cost of solving the linear equation in Eq. (46) for the accelerations $\dot{\chi}$ is $\left\{\frac{1}{6}\mathcal{N}^3 + \frac{3}{2}\mathcal{N}^2 - \frac{2}{3}\mathcal{N}\right\}\text{M} + \left\{\frac{1}{6}\mathcal{N}^3 + \mathcal{N}^2 - \frac{7}{6}\mathcal{N}\right\}\text{A}$.

The overall complexity of the composite body forward dynamics algorithm is $O(\mathcal{N}^3)$.

6.2 Computational Cost of the Articulated Body Forward Dynamics Algorithm

The articulated body forward dynamics algorithm is based on the recursions described in Eq. (62) and Eq. (63). Since the computations in Eq. (47) can be done prior to the dynamics simulation, the cost of this recursion is not included in the cost of the overall forward dynamics algorithm described below:

1. The algorithm for the computation of the articulated body quantities is given in Eq. (62). The step involving the computation of $D^{-1}(k)$ can be carried out either by an explicit inversion of $D(k)$ with $O(n_m^3(k))$ cost, or by the indirect procedure described in Eq. (62) with $O(n_m^2(k))$ cost. The first method is more efficient than the second one for $n_m(k) \leq 7$.

   - Cost of Eq. (62) for the k^{th} body, based on the explicit inversion of $D(k)$ (used when $n_m(k) \leq 7$), is $\left\{\frac{5}{6}n_m^3(k) + \frac{25}{2}n_m^2(k) + \frac{764}{3}n_m(k) + 180\right\}\text{M} + \left\{\frac{5}{6}n_m^3(k) + \frac{21}{2}n_m^2(k) + \frac{548}{3}n_m(k) + 161\right\}\text{A}$.
   - Cost of Eq. (62) for the k^{th} body based on the indirect computation of $D^{-1}(k)$ (used when $n_m(k) \geq 8$) is $\{12n_m^2(k) + 255n_m(k) + 572\}\ \text{M} + \{13n_m^2(k) + 182n_m(k) + 445\}\ \text{A}$.

2. The cost for the tip-to-base recursion sweep in Eq. (63) for the k^{th} body is $\{n_m^2(k) + 25n_m(k) + 49\}$ M + $\{n_m^2(k) + 24n_m(k) + 50\}$ A.

3. The cost for the base-to-tip recursion sweep in Eq. (63) for the k^{th} body is $\{18n_m(k) + 52\}$ M+ $\{19n_m(k) + 42\}$ A.

The overall complexity of this algorithm is $O(Nn_m^2)$, where n_m is an upper bound on the number of modes per body in the system.

The articulated body algorithm is more efficient than the composite body algorithm as the number of modes and bodies in the multibody system increases. Figure 2 contains a plot of the computa-

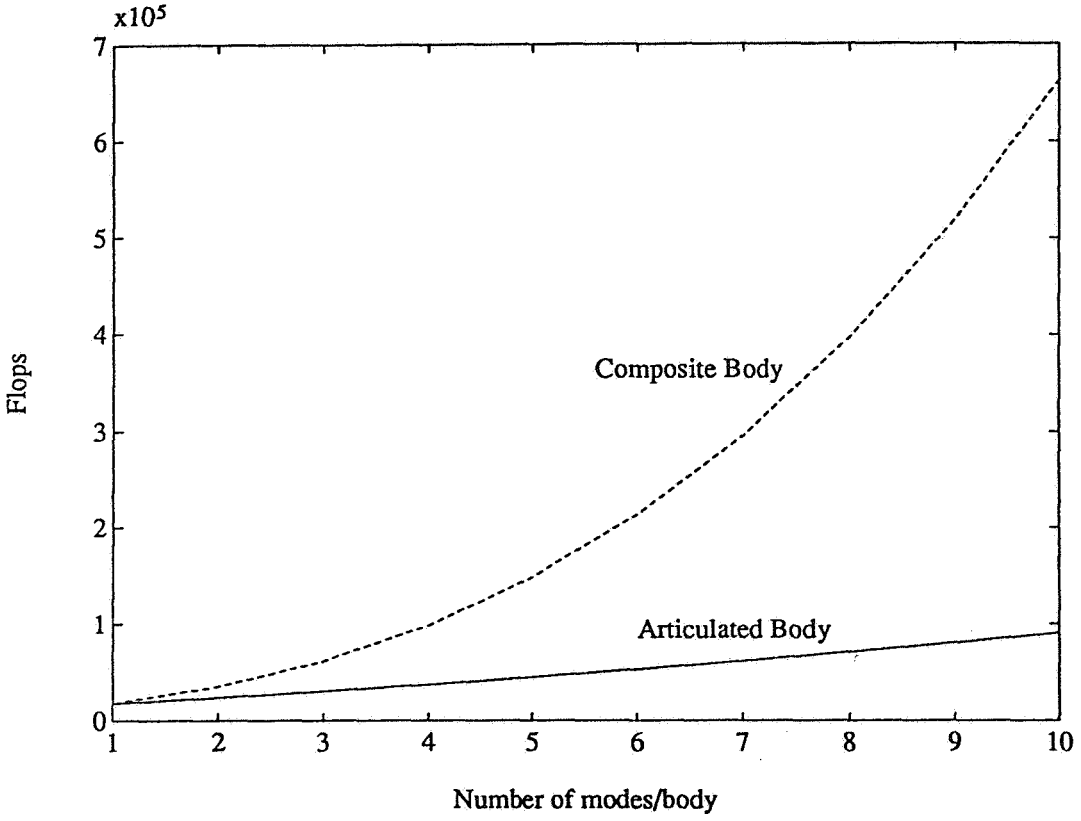


Figure 2: A comparison of computational costs for the forward dynamics algorithms for a flexible multibody serial chain system with 10 flexible bodies.

tional cost (in floating point operations) versus the number of modes per body for a serial chain with ten flexible bodies. The articulated body algorithm is faster by over a factor of 3 for 5 modes per body, and by over a factor of 7 for 10 modes per body. The divergence between the costs for the two algorithms becomes even more rapid as the number of bodies is increased.

7. Extensions to General Topology Flexible Multibody Systems

Extension to general tree and closed-chain systems is similar to methods given in prior results for rigid body configurations [7]. The key is that the operator description does not change as the

topology changes. Extending the serial chain results of this paper to tree topologies takes the following steps:

1. For any outward base to tip(s) recursion, at each body, the outward recursion must be continued along each outgoing branch emanating from the current body.
2. For an inward tip(s) to base recursion, at each body, the recursion must be continued inward only after summing up contributions from each of the incoming branches of the body.

A closed-chain flexible multibody system can be regarded as a tree topology system with additional closure constraints [7].

8. Conclusions

This paper uses spatial operator methods to develop a new dynamics formulation and spatially recursive algorithms for flexible multibody systems. The operator description of the flexible system dynamics is identical in form to the corresponding operator description of the dynamics of rigid multibody systems. A significant advantage of this unified approach is that it allows ideas and techniques for rigid multibody systems to be easily applied to flexible multibody systems. All of the computations are mechanized within a spatially recursive Kalman filtering and smoothing architecture. An extension of this algorithm to handle prescribed motion is described in reference [13].

The computational efficiency of the dynamics algorithms described in this paper makes it possible to implement real–time, high–fidelity, hardware–in–the–loop simulation of complex multibody systems such as spacecraft, robot manipulators, vehicles etc. Such simulations are essential during the design and testing of control and fault recovery algorithms. The articulated body forward dynamics algorithm is currently being used to simulate the dynamics of planetary spacecraft. One application is a spacecraft currently being assembled for a comet and asteroid rendezvous mission [14]. The multibody model for the spacecraft is of tree topology, and consists of a flexible central bus with 9 articulated appendages and 22 hinges' degrees of freedom. The simulation software provides a new capability for high speed simulation of the spacecraft. A real–time version has also been developed. Validation of this software was carried out by running independent simulations of the spacecraft using a standard flexible multibody simulation package [15]. Results from the two independent simulations show complete agreement.

9. Acknowledgement

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

References

[1] Rodriguez, G., Kreutz-Delgado, K., and Jain, A., "A Spatial Operator Algebra for Manipulator Modeling and Control," *The International Journal of Robotics Research*, vol. 10, pp. 371–381, Aug. 1991.

[2] Jain, A. and Rodriguez, G., "Recursive Flexible Multibody System Dynamics Using Spatial Operators," *Journal of Guidance, Control and Dynamics*, vol. 15, pp. 1453–1466, Nov. 1992.

[3] Kim, S.S. and Haug, E.J., "A Recursive Formulation for Flexible Multibody Dynamics, Part I: Open–Loop Systems," *Computer Methods in Applied Mechanics and Engineering*, vol. 71, no. 3, pp. 293–314, 1988.

[4] Changizi, K. and Shabana, A.A., "A Recursive Formulation for the Dynamic Analysis of Open Loop Deformable Multibody Systems," *ASME Jl. of Applied Mechanics*, vol. 55, pp. 687–693, Sept. 1988.

[5] Keat, J.E., "Multibody System Order n Dynamics Formulation Based on Velocity Transform Method," *Journal of Guidance, Control and Dynamics*, vol. 13, March-April 1990.

[6] Jain, A. and Rodriguez, G., "Recursive Dynamics for Flexible Multibody Systems Using Spatial Operators," JPL Publication 90-26, Jet Propulsion Laboratory, Pasadena, CA, Dec. 1990.

[7] Rodriguez, G., Jain, A., and Kreutz-Delgado, K., "Spatial Operator Algebra for Multibody System Dynamics," *Journal of the Astronautical Sciences*, vol. 40, pp. 27–50, Jan.–March 1992.

[8] Walker, M.W. and Orin, D.E., "Efficient Dynamic Computer Simulation of Robotic Mechanisms," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 104, pp. 205–211, Sept. 1982.

[9] Rodriguez, G., "Kalman Filtering, Smoothing and Recursive Robot Arm Forward and Inverse Dynamics," *IEEE Journal of Robotics and Automation*, vol. 3, pp. 624–639, Dec. 1987.

[10] Jain, A., "Unified Formulation of Dynamics for Serial Rigid Multibody Systems," *Journal of Guidance, Control and Dynamics*, vol. 14, pp. 531–542, May–June 1991.

[11] Featherstone, R., "The Calculation of Robot Dynamics using Articulated-Body Inertias," *The International Journal of Robotics Research*, vol. 2, pp. 13–30, Spring 1983.

[12] Padilla, C. E. and von Flotow, A. H., "Nonlinear Strain–Displacement Relations and Flexible Multibody Dynamics," in *Proceedings of the 3rd Annual Conference on Aerospace Computational Control, Vol. 1*, (Oxnard, CA), pp. 230–245, Aug. 1989. (JPL Publication 89–45, Jet Propulsion Laboratory, Pasadena, CA, 1989).

[13] Jain, A. and Rodriguez, G., "Recursive Dynamics Algorithm for Multibody Systems with Prescribed Motion," *Journal of Guidance, Control and Dynamics*, 1992. In press.

[14] Bell, C.E., Bernard, D.E., and Rasmussen, R.D., "Attitude and Articulation Control for CRAF/Cassini," in *First ESA International Conference on Spacecraft Guidance, Navigation and Control Systems*, (Noordwijk, The Netherlands), June 1991.

[15] Bodley, C. S., Devers, A. D., Park, A. C., and Frisch, H. P., "A Digital Computer Program for the Dynamic Interaction Simulation of Controls and Structure (DISCOS)," NASA Technical Paper 1219, NASA, Goddard Space Flight Center, May 1978.

NONRECURSIVE FORMULATIONS OF MULTIBODY DYNAMICS AND CONCURRENT MULTIPROCESSING

Andrew J. Kurdila
Ramesh Menon
Center for Mechanics and Control
Department of Aerospace Engineering
Texas A&M University
College Station, Texas 77843

ABSTRACT

Since the late 1980's, research in recursive formulations of multibody dynamics has flourished. Historically, much of this research can be traced to applications of low dimensionality in mechanism and vehicle dynamics. Indeed, there is little doubt that recursive order N methods are the method of choice for this class of systems. This approach has the advantage that a minimal number of coordinates are utilized, parallelism can be induced for certain system topologies, and the method is of order N computational cost for systems of N rigid bodies.

Despite the fact that many authors have dismissed redundant coordinate formulations as being of order N^3, and hence less attractive than recursive formulations, we present recent research that demonstrates that at least three distinct classes of redundant, nonrecursive multibody formulations consistently achieve order N computational cost for systems of rigid and/or flexible bodies. These formulations are the

(i) preconditioned range space formulation,
(ii) penalty methods, and
(iii) augmented Lagrangian methods

for nonlinear multibody dynamics. The first method can be traced to its foundation in equality constrained quadratic optimization, while the last two methods have been studied extensively in the context of coercive variational boundary value problems in computational mechanics. Until recently, however, they have not be investigated in the contrext of multibody simulation, and present theoretical questions unique to nonlinear dynamics. All of these nonrecursive methods have additional advantages with respect to recursive order N methods: (1) The formalisms retain the highly desirable order N computational cost, (2) the techniques are amenable to concurrent simulation strategies, (3) the approaches do not depend upon system topology to induce concurrency, (4) and the methods can be derived to balance the computational load automatically on concurrent multiprocessors. In addition to the presentation of the fundamental formulations, this paper presents new theoretical results regarding the rate of convergence of order N constraint stabilization schemes associated with the newly introduced class of methods.

Introduction

It is well known that the nonlinear equations governing the motion of complex multibody systems give rise to differential-algebraic equations that present unique and interesting problems in their solution [Gear, Petzold]. The mathematical study of differential-algebraic equationsis justifiably a field of research unto itself. However, in multibody applications, it appears that most researchers prefer to eliminate the troublesome multipliers, and deal exclusively with the problems associated with the solution of sets of ordinary differential equations. The point in the analysis at which the multipliers are eliminated has become one criteria for distinguishing among the various formulations. During the last 1970's and early 1980's, numerous publications appeared in which the constraint multipliers are eliminated at computation time by numerically constructing a basis for the instantaneous nullspace of the constraint Jacobian. Some representative work from this class includes [Singh, Wampler, Wehage, Amirouche,...], and are referred to as the nullspace methods. These algorithms have been so named due to their similarity to the nullspace methods in quadratic, equality constrained optimizations [Gill]. Despite their elegance, numerical calculation of the nullspace basis and its use to eliminate the multipliers leads to a dense system coefficient matrix. This matrix is of order (N-M) X (N-M) where N is the number of redundant coordinates and M is the number of constraints. Consequently, the nullspace methods are of $O(N-M)^3$ at each time step.

As opposed to techniques that eliminate the multipliers numerically at computation time, a number of authors have derived elegant techniques for eliminating the constraints a priori; that is during the derivation of the equations. The works [Rodriguez, June 1987; Rodriguez, 1987] are representative of this class. These methods have a number of distinct and well-known advantages over the cubic order nullspace methods:

(i) They employ a minimal coordinate set.
(ii) They attain an O(N) computational cost for systems of rigid bodies.
(iii) They can be employed in concurrent architectures for classes of problems.

These methods have come to be known as the order N or recursive methods of formulating multibody dynamics.

Despite their numerous advantages, there are several aspects of the recursive order N methods that can cause difficulties for classes of problems. Foremost among these problems is that concurrency in the recursive methods is induced, at present, by assigning topologically independent branches of the structure to computationally independent processors. One need only consider the problem of modelling a tethered satellite system to realize that not all structures exhibit such structural parallelism. Furthermore, most systems exhibit low levels of inherent structural parallelism appropriate for the concurrent implementation of recursive order N methods as in [Bae]. Typical space structures such as the space station or the CSI Evolutionary model at NASA Langley have only on the order of 6-12 "independent branches." Gross underutilization of moderately parallel architectures, with 16 to 128 processors, can result when mapping the recursive algorithm onto such a computer. Finally, computational load balancing of the problem among processors in most implementation is carried out by hand, which can be a tedious and time consuming task.

Nonrecursive Formulations

This paper surveys the results of several recent approaches presented in [Kurdila 1,2,3] and described in detail in the dissertation by [Menon]. Collectively, the authors refer to these methods as nonrecursive formulations of multibody dynamics. In point of fact, the nonrecursive formulations described in this paper represent a synthesis of a family of formulations and constraint stabilization procedures that are closely related. These methods can be employed simultaneously, independently, or in combination selected by an analyst to meet accuracy and computational speed requirements in a highly predictable manner. This class includes

(i) The PCG/Range Space Method : As with the nullspace method, this approach can be traced historically to equality constrained quadratic optimization [Gill]. Analysis of the application of the approach to multibody simulation has been carried out recently in [Kurdila] and [Menon] using an iterative technique, and earlier noniterative versions appear in [Placek] and [Wittenburg]. The primary advantage of the range space method is that is is the most numerically efficient of the three methods presented in this paper.

(ii) The Penalty Method : Of course, the penalty method has been studied in detail in application to optimization [Luenberger] and coercive variational boundary value problems [Oden]. Still, it has only recently been studied in detail in the context of nonlinear multibody simulation [Bayo, Park, Kurdila]. An advantage of this approach is that explicit bounds on the constraint violation can be achieved in terms of the penalty parameter, even in the nonlinear case. [Kurdila] One disadvantage of the method that has been noted in the literature [Kurdila] , [Bayo] is that prohibitively large values of the penalty parameter are required to achieve the analytical guarantees of accuracy, but result in numerical ill-conditioning.

(iii) The Augmented Lagrangian Method : As in the case of the penalty method, the augmented Lagrangian formulations have been studied extensively for use in the solution of variational boundary value problems [Glowinski]. Again, as in the case of the penalty method, this approach has only recently been studied within the field of nonlinear multibody dynamics [Bayo]. Empirical evidence in [Bayo1,2] suggests that the method is superior to the penalty method from a computational viewpoint in that required accuracy can be achieved without large penalty parameters. The analytical study of Augmented Lagrangian formulations of nonlinear multibody dynamics remains an open and interesting field of research.

To derive the equations employed in this paper, one must first consider the exact governing system of differential-algebraic equations

$$\begin{aligned} M(q)\ddot{q} &= f(q, \dot{q}, t) + C^T\lambda \\ \Phi(q, t) &= 0 \end{aligned} \tag{1}$$

and differentiated form of the constraints

$$\left[\frac{\partial \Phi}{\partial q}\right]\dot{q} + \Phi_t = 0 \tag{2}$$

$$C(q,t) \equiv \left[\frac{\partial \Phi}{\partial q}\right] \tag{3}$$

In these equations, q are the generalized coordinates selected for the problem at hand, and $\Phi(q)$ are the functional relations defining the holonomic constraints acting on the system.While the interested reader is referred to [Menon] for the details of their derivation, the equations governing the simulation of multibody systems considered in this paper can be written

$$\lambda_0 = 0 \tag{4}$$

or

$$(CM^{-1}C^T)\lambda_0 = -CM^{-1}f - \dot{C}\dot{q} - \dot{\Phi}_t \tag{5}$$

$$M(q)\ddot{q}_n = f(q,\dot{q},t) - C^T\frac{1}{\varepsilon}(\ddot{\Phi}_n + 2\xi\omega\dot{\Phi} + \omega^2\Phi) - C^T\lambda_n \tag{6}$$

$$\lambda_{n+1} = \lambda_n + \frac{1}{\varepsilon}(\ddot{\Phi}_n + 2\xi\omega\dot{\Phi} + \omega^2\Phi) \tag{7}$$

The terms λ are unknown Lagrange multipliers, λ_{n+1} are iterative corrections the the multipliers, and $\ddot{\Phi}_n$ are iterative values of the constraint equations. The relationship of the above system to the aforementioned range space, penalty and Augmented Lagrangian formulations can be summarized as follows:

(i) By employing equation (5) to initiate the integration procedure and letting

$$\frac{1}{\varepsilon} = 0 \tag{8}$$

the above equations reduce to the range space equations, and provide the most computationally efficient version of the above set of equations when a judiciously selected preconditioned conjugate gradient method is employed.

(ii) By employing only equation (6), and by setting

$$\lambda_n \equiv \lambda_{n+1} \equiv 0 \tag{9}$$

the equations above reduce to the inertial penalty method introduced in [Bayo] and studied in further detail in [Kurdila]. This form of the governing equations provides some guarantees on accuracy, but is more expensive computationally.

(iii) By using equation (4) for the initialization procedure, and choosing equations (6) and (7) iteratively, these equations reduce to the augmented Lagrangian approach introduce in [Bayo] and studied further in [Kurdila] and [Menon]. While this version of the governing equations are the most computationally expensive, they also provide the most control over the accuracy attained in the simulation.

At this point it should be noted that both the penalty and augmented Lagrangian methods can be viewed as stabilizaton procedures for the range space method. In the following discussion, the range space formulation will be referred to as the "baseline method". In fact, it will be demonstrated later that this is a useful interpretation and leads to hybrid simulation methods that combine the computational efficiency of the range space method and the accuracty of the penalty or augmented Lagrangian methods.

In the following sections it will be emphasized that the simulation of transient response of multibody systems using various versions of the nonrecursive formulation above has many advantages:

(i) The method achieves an order N computational cost while employing a redundant formulation.

(ii) Consequently, the often complicated relative reference frame kinematics of the recursive methods can be avoided.

(iii) The method is amenable to concurrent implementation on a wide variety of forthcoming parallel architectures.

(iv) The technique is essentially automatically, computationally load balancing

(v) The approach does not depend upon system topology to induce parallelism, and does not result in gross underutilization of available processors.

Order N Computational Cost of the Baseline Algorithm

The order N computational cost of the baseline algorithm when the governing equations are selected to be simplified versions of equations (5) and (6)

$$\begin{aligned} (CM^{-1}C^T)\lambda &= -CM^{-1}f - \dot{C}\dot{q} - \dot{\Phi}_t \\ M(q)\ddot{q} &= f(q,\dot{q},t) - C^T\lambda \end{aligned} \tag{10}$$

, otherwise denoted as the preconditioned range space formulation in [Kurdila] and [Menon], is well documented. While the details of the convergence properties of the algorithm exceed the scope of this paper the reader is referred to [Menon] for a complete discussion. In summary, the computational performance of the algorithm can be attributed to

(i) the derivation of a rapidly convergent block Jacobi preconditioner based on the directed connectivity graph of the multibody system,

(ii) the exploitation of the block-diagonal structure of the system coefficient matrices in the formulation, and

(iii) the parallel implementation of the algorithm based upon subdomain decomposition techniques that are only weakly coupled to the system topology.

Results extracted from [Kurdila] and [Menon] in figures (1) through (4) illustrate results typical of the preconditioned conjugate gradient / range space formulation.

Accuracy and Order N Performance with Constraint Stabilization

The motivation for deriving additional variants of the preconditioned conjugate gradient / range space formulation arises from the well-known need for stabilization in redundant formulations as well as the documented conditioning problems that can occur in the range space method of optimization [Gill]. Essentially, the accuracy of the simulation relies on the condition number of the constraint metric

$$CM^{-1}C^T \tag{11}$$

appearing in equation (5) and (10). As noted in [Gill], this condition number is bounded above by

$$\kappa(CM^{-1}C^T) \leq \kappa^2(C)\,\kappa(M) \tag{12}$$

and may become large when the constraints become nearly redundant. However, inasmuch as great efforts have been made to ensure the order N computational cost of the baseline formulation, it has been a central goal in this work to derive stabilization methods that retain the order N com-

Figure (1) : Order N Timing Results for Z- Truss Prototype Model

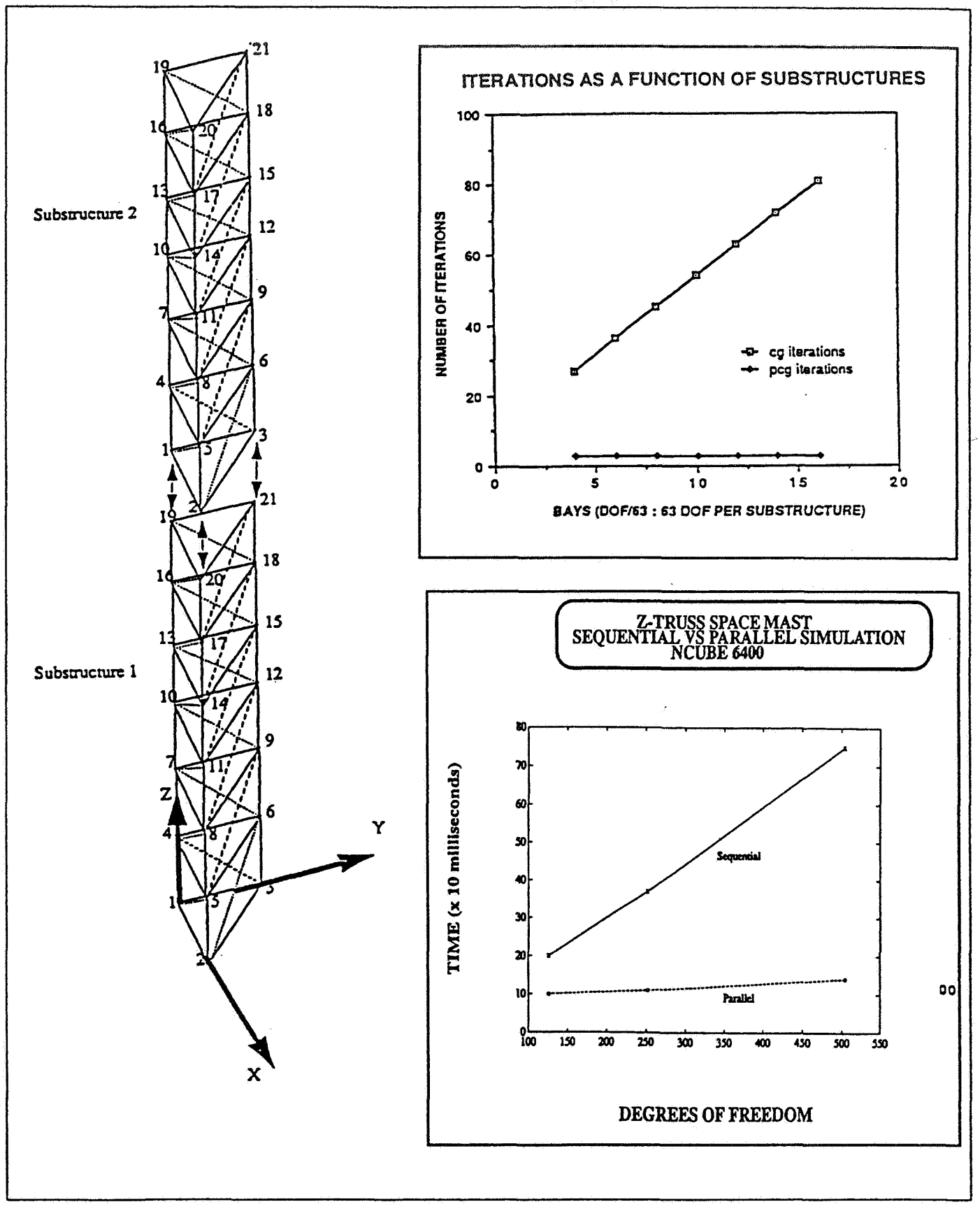
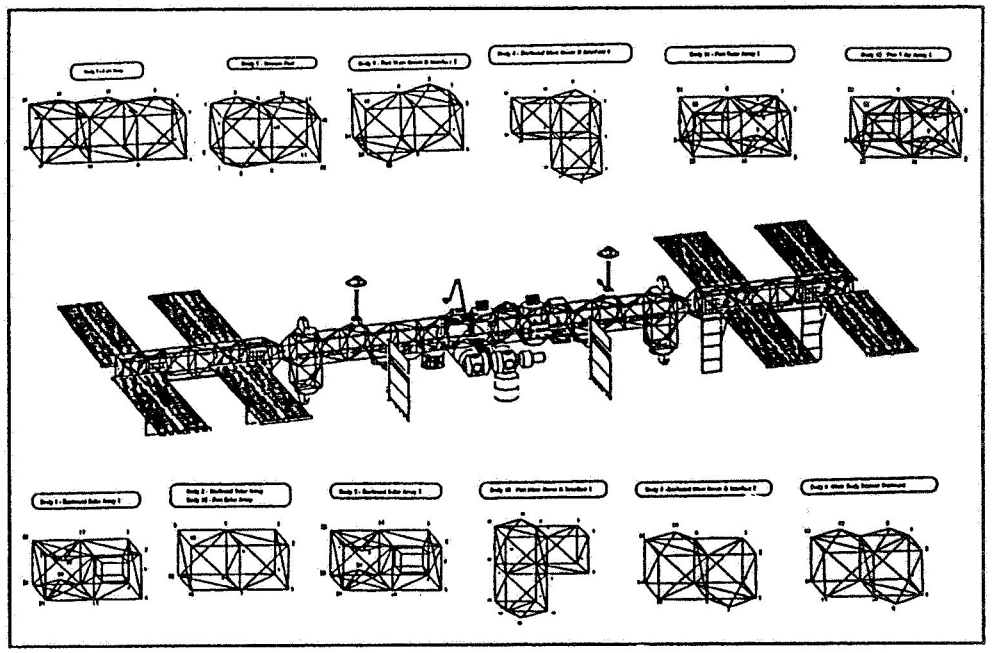


Figure (2) : Order N Timing Results for Space Station Prototype Model



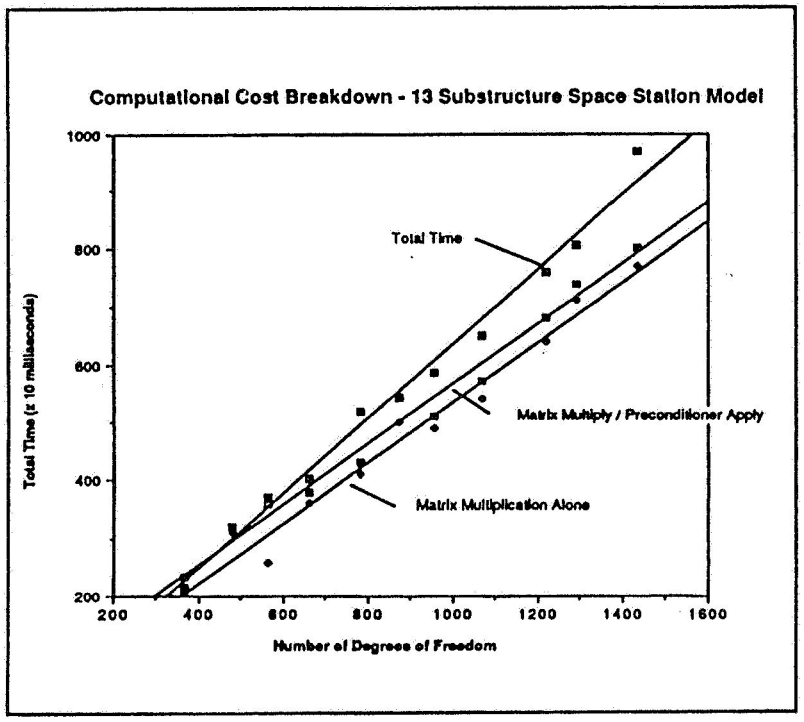
1434 DEGREES OF FREEDOM

300 CONSTRAINTS

72 TO 150 DOF/SUBSTRUCTURE

ORDER N PERFORMANCE

ENERGY RATE MATCHING INTEGRATION



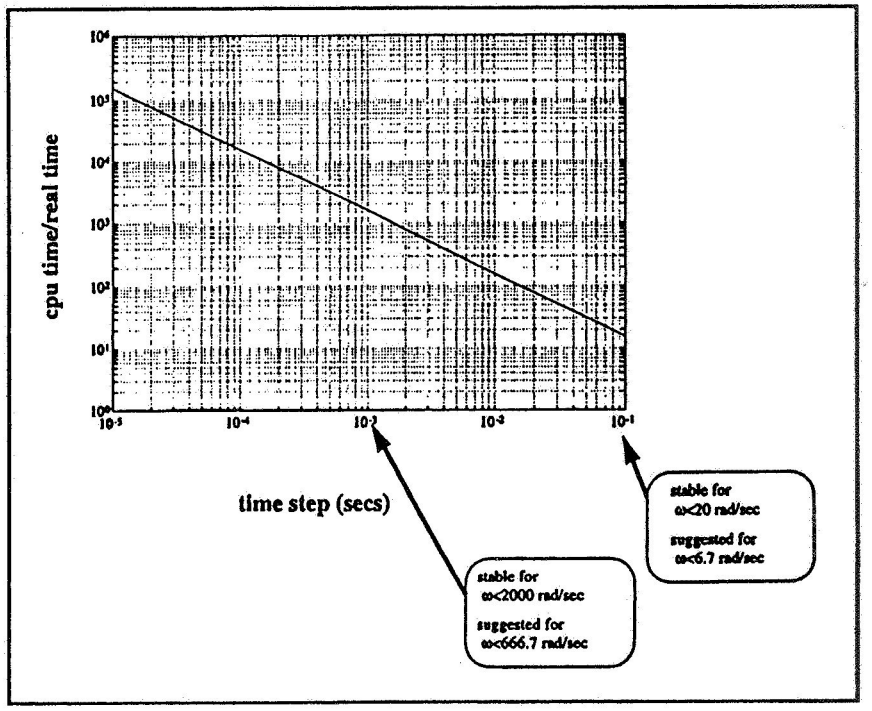


Figure (3) : Order N Timing Results for CSI Evolutionary

13 SUBSTRUCTURES

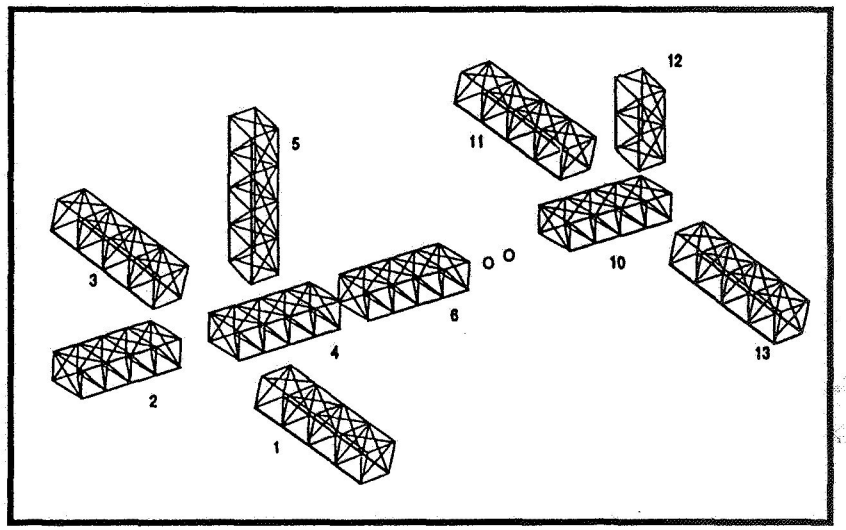
1656 DEGREES OF FREEDOM

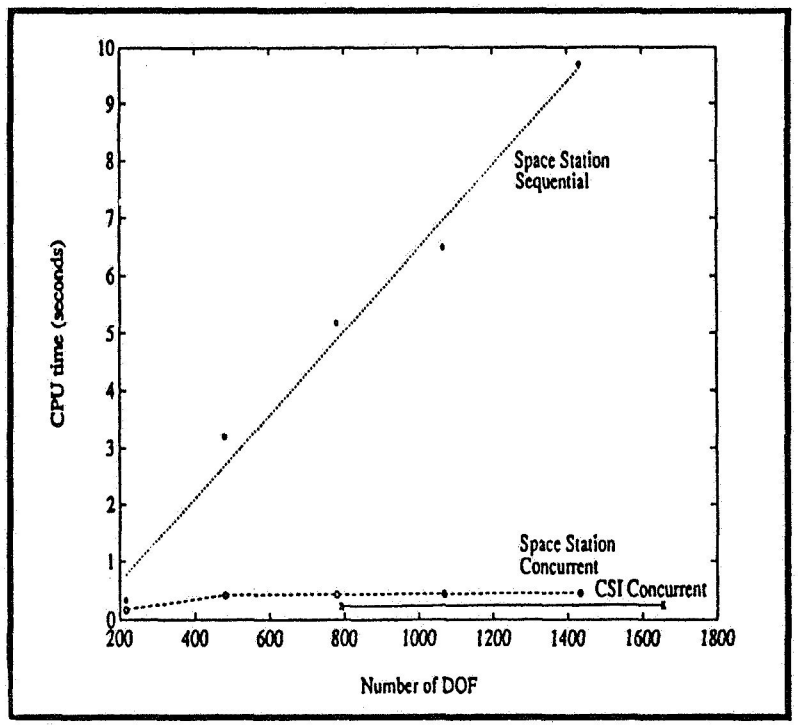
144 CONSTRAINTS

72 TO 132 DOF/SUBSTRUCTURE

ORDER N PERFORMANCE

ENERGY RATE MATCHING INTEGRATION





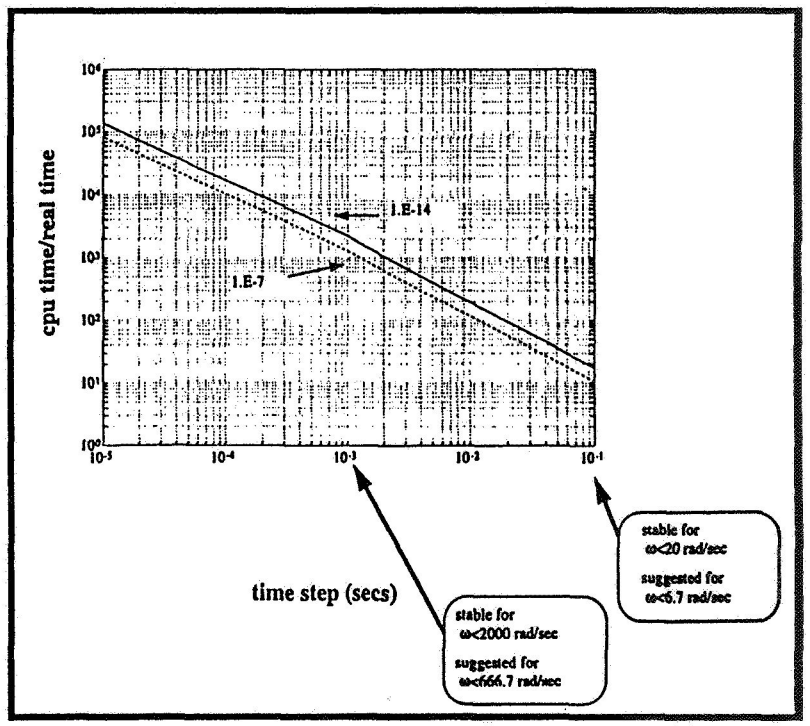


Figure (4) : Order N Timing Results for Nonlinear Tracking Prototype Model

NONRECURSIVE FORMULATION

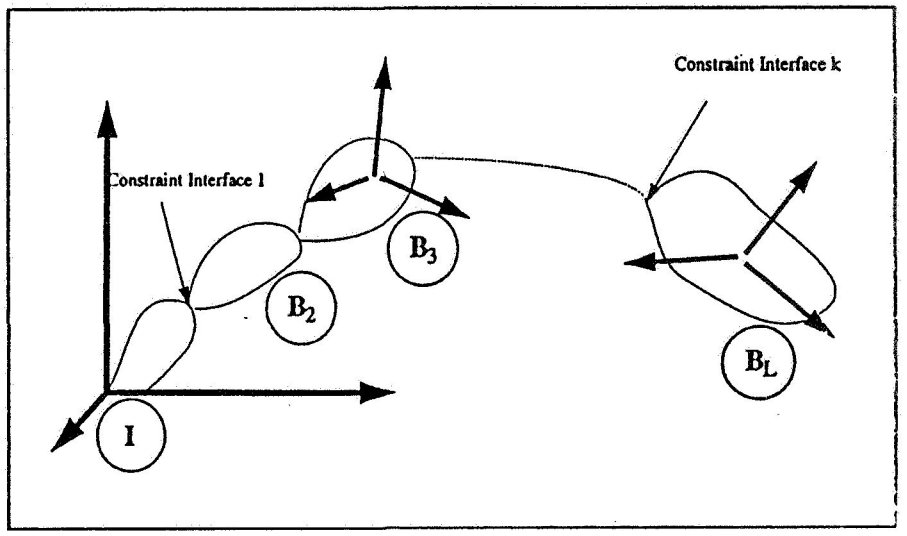
SYMBOLIC CONTRAINT JACOBIAN

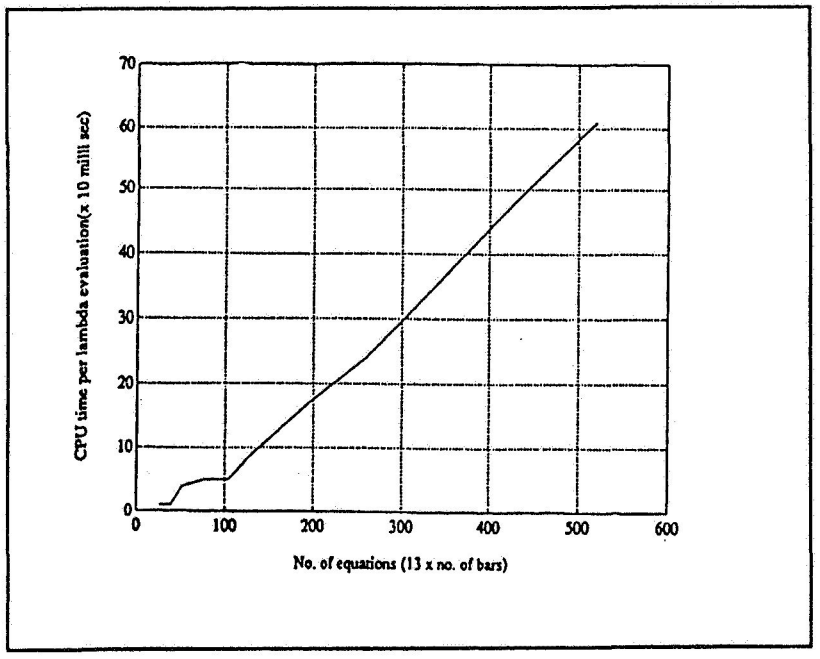
ORDER N PERFORMANCE

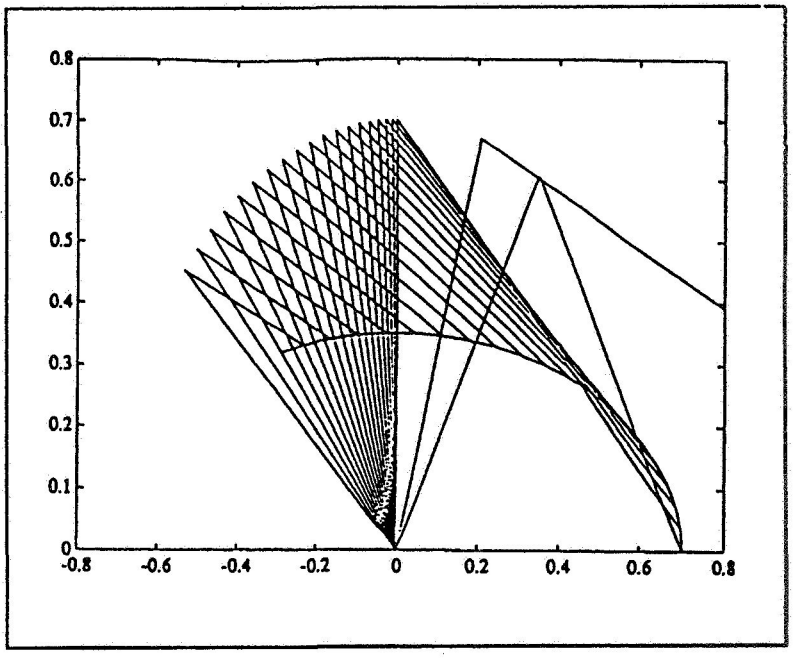
1 TO 40 RIGID BODIES

520 DEGREES OF FREEDOM

HAMILTONIAN TRACKING LAW







putational cost.

This goal has in fact been achieved and is depicted in figures (5) and (6). In figure (5), the timing results for a benchmark linear structural problem as a function of the number of degrees of freedom is depicted. Two critical observations should be made upon inspection of this graph

(i) All variants of the class of nonrecursive algorithms under investigation exhibit an order N compuational cost.

(ii) The baseline preconditioned conjugate gradient / range space algorithm is the most efficient simulation.

(iii) The computational cost of the nonrecursive methods with stabilization increases with the number of iterative corrections per time step. Hence, the penalty stabilization is more expensive thant the preconditioned conjugate gradient / ranges space method. But, the augmented Lagrangian formulation is more computationally expensive than the penalty method. In fact, the slope of the performance curve simply rotates counterclockwise as the number of fixed iterations per time step increases.

In view of the measures of accuracy depicted in figures (7) and (8), it is not surprising that the iterative corrections of the augmented Lagrangian are more costly. It is precisely the addition of the stabilization methods that yields the desired improvement in accuracy. As shown in [Kurdila], for certain classes of multibody systems the constraint violation can be bounded by

$$\left\| \dot{\Phi}_\varepsilon \right\|^2 + \left\| \Phi_\varepsilon \right\|^2 \leq \frac{2E(0)}{min\left(\sigma^2_{min}(\alpha), \sigma^2_{min}(\beta)\right)}$$

As noted earlier, a major criticism of achieving stringent tolerances using this bound is that large penalty factors must be used in the simulation which can lead to poor conditioning in practice. This result has been extended rigorously to the augmented Lagrangian stabilization methods in [Bustimante] and [Menon] in the form of the equation

$$\left| \ddot{\Phi} + 2\xi\omega\dot{\Phi} + \omega^2\Phi \right|_{i+1} \leq \frac{\varepsilon}{\upsilon} \left| \ddot{\Phi} + 2\xi\omega\dot{\Phi} + \omega^2\Phi \right|_i$$

and is depicted graphically in figure (7). Furthermore, empirical results suggest the stronger result that

$$\left| \dot{\Phi}(t)^2 + \omega^2\Phi(t)^2 \right|_{i+k} \leq \left(\frac{\varepsilon}{\upsilon}\right)^{2k} \left| \dot{\Phi}(t)^2 + \omega^2\Phi(t)^2 \right|_i$$

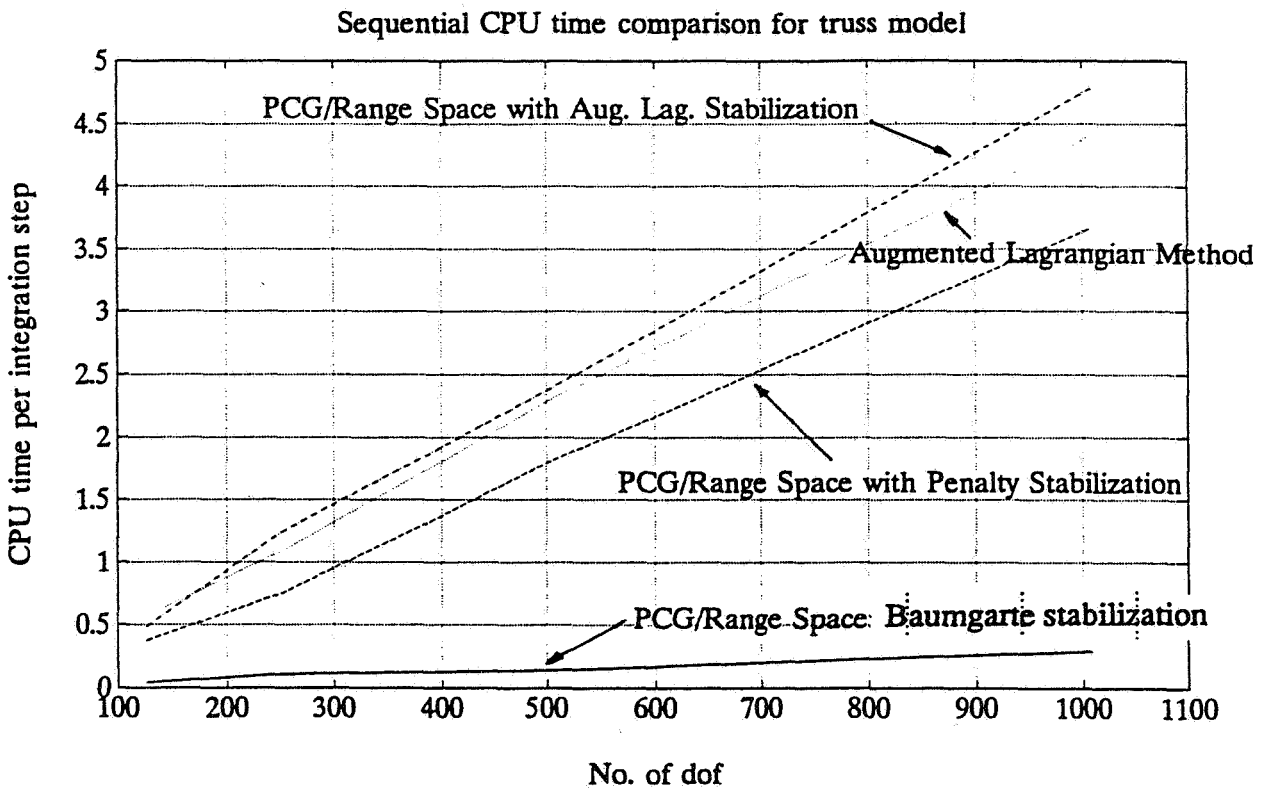
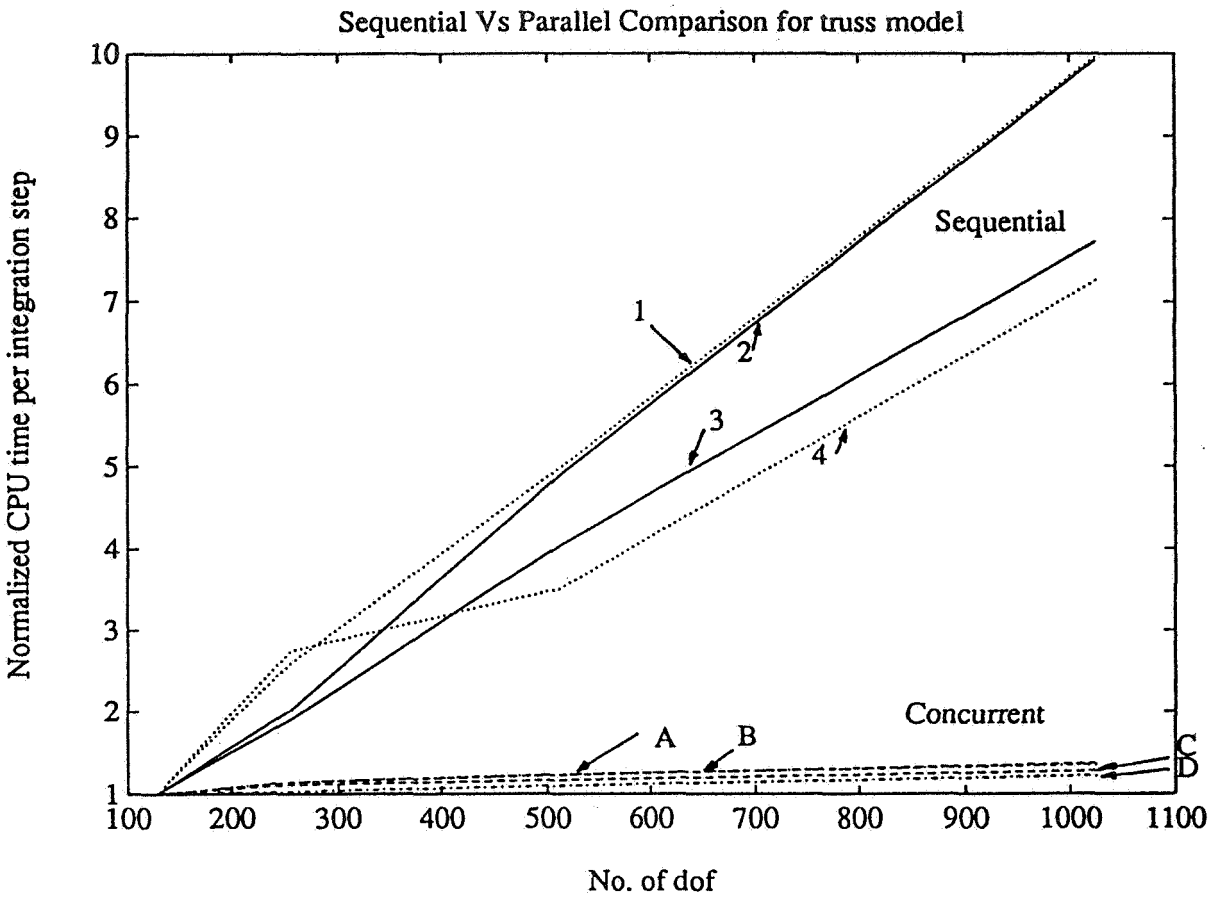


Figure (5) : Order N Computational Cost for Nonrecursive Formulation Class



LEGEND: PCG with Augmented Lagrangian, 1 and A
Augmented Lagrangian Solution, 2 and B
PCG with Penalty, 3 and C
PCG with Baumgarte's, 4 and D

Figure (6) : Normalized Order N Computational Cost for Nonrecursive Formulation Class

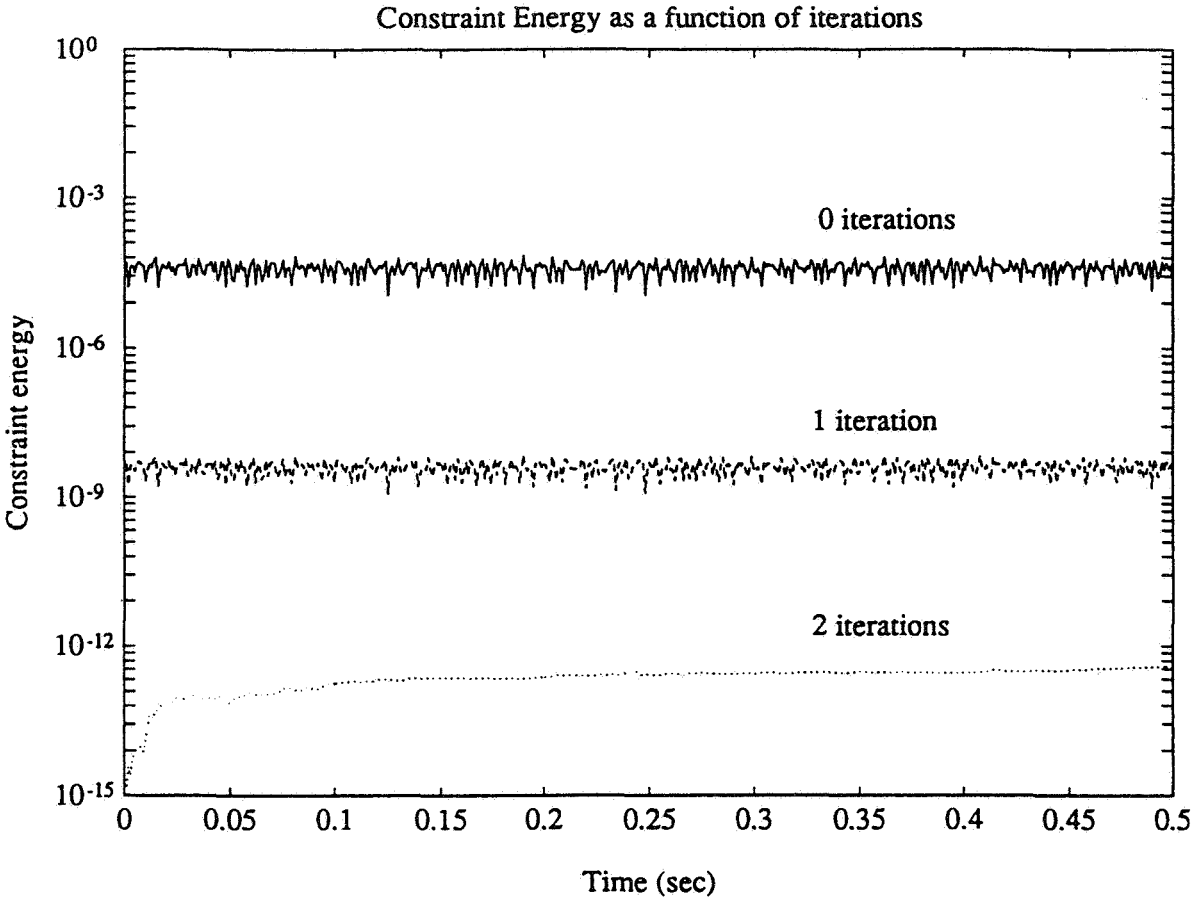


Figure (7) : Constraint Violation and Correction Steps

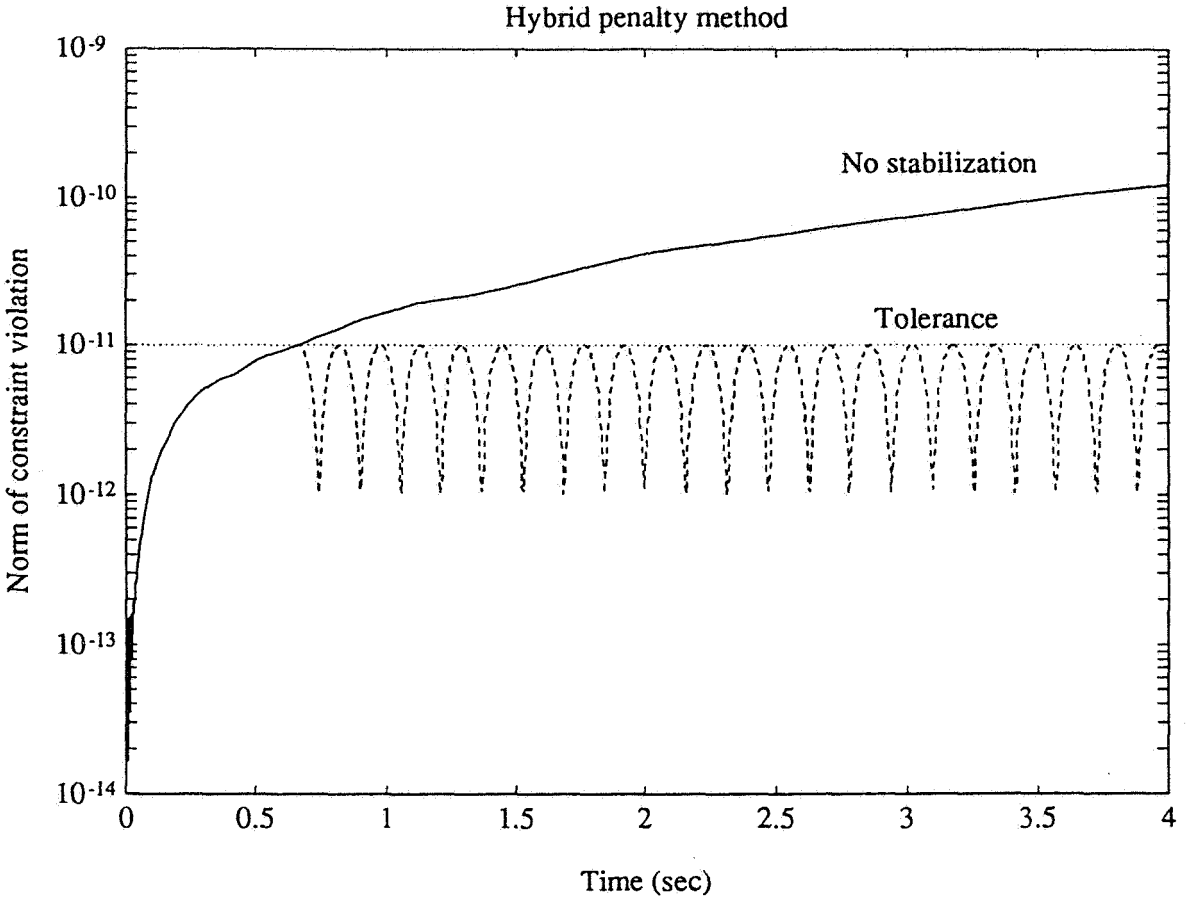


Figure (8) : Hybrid Range Space Simulation and Augmented Lagrangian Stabilization

Simply put, the "energy of constraint violation shrinks by a factor of e for each iteration" employed in the augmented Lagrangian formulation.

Thus, figures (5) and (7), illustrate the fundamental tradeoff between execution time and accuracy. Another natural consequence of this tradeoff is the design of hybrid methods such as depicted in figure (8). In these methods, the preconditioned conjugate gradient / range space method is empolyed until a specific pre-selected tolerance is met, at which point the augmented Lagrangian method is employed until another more stringent tolerance is reached. In this fashion, the computational efficiency of the range space method is retained as long as possible, at which point the iterative augmented Lagrangian method is used to guarantee the desired accuracy.

Conclusions

This paper has presented a class of nonrecursive formulations of multibody dynamics, all of which exhibit an order N computational cost. The methods are complementary to the existing class of recursive order N methods in that they seem most appropriate for large dimensional models without inherent structural parallelism. Figures (9) and (10) graphically summarize the contribution of this paper. Figure (9) shows that as of 1988 only one order N method had been derived. Figure (10) depicts the state of affairs currently, and shows that at least three distinct approaches can achieve order N computational cost, as well as numerous combinations of these algorithms.

References

Bae, D., and Haug, E.J., "A Recursive Formulation for Constrained Mechanical System Dynamics: Part I. Open Loop Systems", Mechanics of Structures and Machines, Vol. 15 No. 3 pp. 359-382, 1987.

Bae, D., and Haug, E.J., "A Recursive Formulation for Constrained Mechanical System Dynamics: Part II. Closed Loop Systems", Mechanics of Structures and Machines.

Bae, D., and Haug, E.J., "A Recursive Formulation for Constrained Mechanical System Dynamics: Part III. Parallel Processor Implementation", Mechanics of Structures and Machines, Vol. 16 No.2 pp. 249-269, 1988.

Banerjee, A.K. and Kane, T.R., "Extrusion of a Beam from a Rotating Base", Journal of Guidance, Control and Dynamics, Vol. 12, No. 2, pp. 140-146, February, 1989.

Banerjee, A.K., "Order-n formulation of Extrusion of a Beam with Large Bending and Rotation", Journal of Guidance, Control and Dynamics, Vol. 15, No. 1, pp. 121-127, January, 1992.

Baumgarte, J.W., "Stabilization of Constraints and Integrals of Motion in Dynamical Systems", Computer Methods in Applied Mechanics and Engineering, Vol. 1, pp. 1-16, 1972.

Baumgarte, J.W., "A New Method of Stabilization for Holonomic Constraints", Journal of Applied Mechanics, Vol. 50, pp. 869-870, 1983.

Bayo, E. and Serna, M.A.,"Penalty Formulation for the Dynamic Analysis of Elastic Mechanisms", Journal of Mechanisms, Transmissions and Automation in Design, vol.111, pp. 321-327, 1989.

Bayo, E. *et al*,"A Modified Lagrangian Formulation for the Dynamic Analysis of Constrained Mechanical Systems", Computer Methods in Applied Mechanics and Engineering, vol.71, pp. 183-195, 1988.

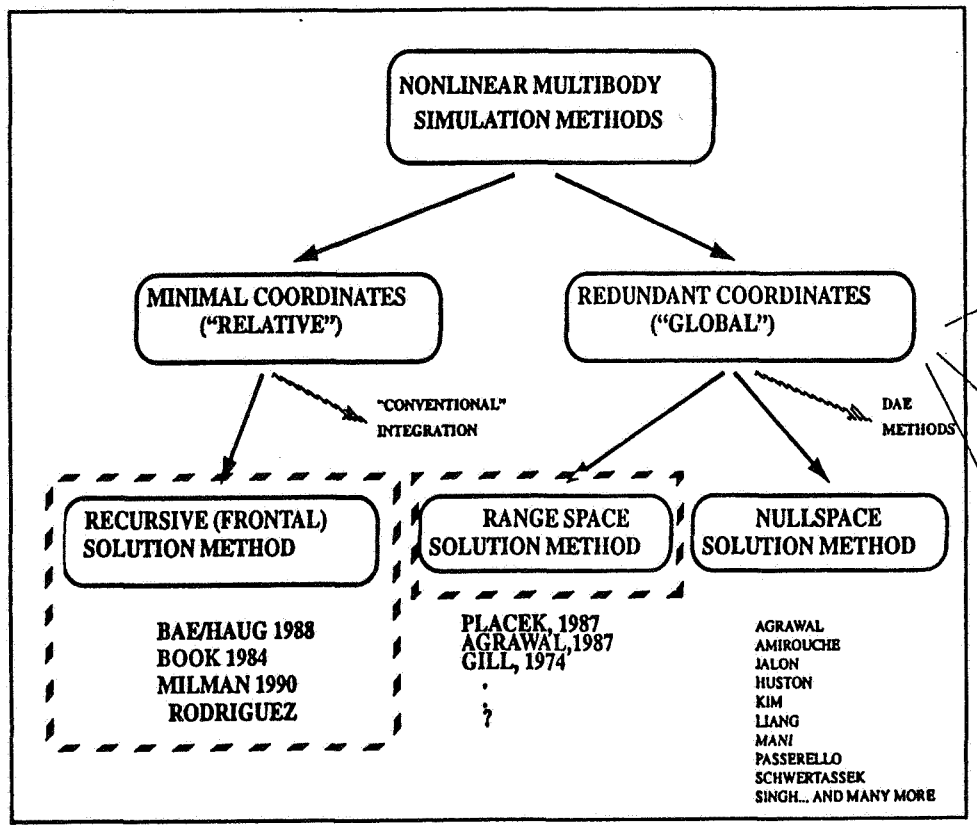


Figure (9) : Order N Methods, <=1988

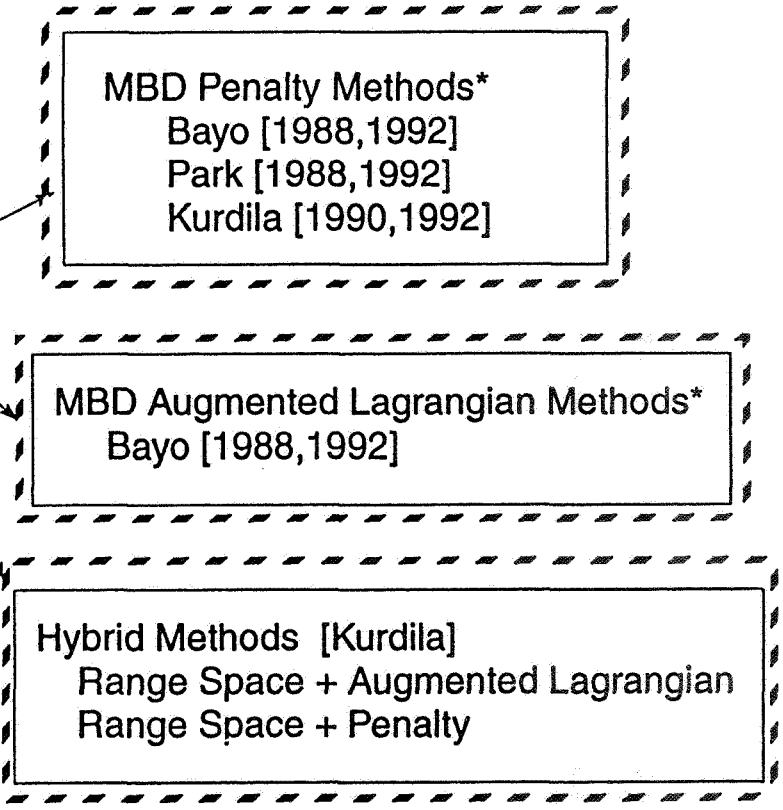


Figure (10) : Order N Methods, <= 1992

* DENOTES THE FACT THAT PENALTY AND AUGMENTED LAGRANGIAN METHODS HAVE A LONG HISTORY IN OPTIMIZATION, COMPUTATIONAL MECHANICS...

Bayo, E., Garcia de Jalon, J., Avello, A., and Cuadrado, J.,"An Efficient Computational Method for Real Time Multibody Dynamic Simulation in Fully Cartesian Coordinates", Computer Methods in Applied Mechanics and Engineering, vol.92, pp. 377-395, 1991.

Bayo, E., and Avello, A., "Singularity-Free Augmented Lagrangian Algorithms for Constrained Multibody Dynamics", preprint, to be published in Nonlinear Dynamics.

Belvin, W.K. *et. al.*, "Langley's CSI Evolutionary Model: Phase 0", NASA Technical Memorandum 104165, September, 1991.

Chiou, J.C, "Constraint Treatment Techniques and Parallel Algorithms for Multibody Dynamic Analysis", Ph.D. Dissertation, University of Colorado, November 1990.

Clark, H.T., and Kang, D.S., "Application of Penalty Constraints for Multibody Dynamics of Large Space Structures", Paper No. AAS 92-157, AAS/AIAA Spaceflight Mechanics Meeting, Colorada Springs, CO, February 1992.

Fortin, M., and Glowinski, R., "Augmented Lagrangian Methods", North-Holland, AmsterGear, C.W., "Numerical Initial Value Problems in Ordinary Differential Equations", Prentice-Hall, Englewood Cliffs, NJ, 1971.

Gear, C.W., "The Simultaneous Numerical Solution of Differential-Algebraic Equations", IEEE Transactions on Circuit Theory, TC-18, No. 1, pp. 89-95, 1971.

Gear C.W., "The Numerical Solution of Problems which may have High Frequency Components", in Computer Aided Analysis and Optimization of Mechanical System Dynamics, Haug, E.J., Editor, NATO ASI Series, Vol. F9, pp. 335-348, Springer-Verlag, Berlin, 1984.

Gear, C.W., and Petzold, L.R., "ODE Methods for the Solution of Differential/Algebraic Equations", SIAM Journal of Numerical Analysis, Vol. 21, No. 4, August 1984.

Gear, C.W., "Differential ALgebraic Equations", in Computer Aided Analysis and Optimization of Mechanical System Dynamics, Haug, E.J., Editor, NATO ASI Series, Vol. F9, pp. 323-334, Springer-Verlag, Berlin, 1984.

Glowinski, R. and Le Tallec, P., Augmented Lagrangian and Operator-Splitting Methods in Nonlinear Mechanics, Society for Industrial and Applied Mathematics, Philadelphia, 1989.

Gill, P.E., Murray, W., and Wright, M.H., Practical Optimization, Academic Press, London, 1981.

Hollerbach, J.M., "A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity", IEEE Transactions on Systems Man and Cybernetics, Vol. 10 pp. 730-736, 1980.

Featherstone, R., "Robot Dynamics Algorithms", Kluwer Academic Publishers, Boston, MA, 1987.

Keat, J.E., "Multibody System Order n Dynamics Formulation Based on Velocity Transform Method", Journal of Guidance, Control and Dynamics, Vol. 13, No. 2, pp. 207-212, 1990.

Kim, S-S., and Haug, E.J., "A Recursive Formulation for Flexible Multibody Dynamics: Part I. Open Loop Systems", Computer Methods in Applied Mechanics and Engineering, Vol. 71 pp. 293-314, 1988.

Kim, S-S, and Haug, E.J., "A Recursive Formulation for Flexible Multibody Dynamics: Part II. Closed Loop Systems", Computer Methods in Applied Mechanics and Engineering, Vol. 74 pp. 251-269, 1989.

Kim, S-S., and Vanderploeg, M.J., "QR Decomposition for State Space Representation of Constrained Mechanical Dynamic Systems", Journal of Mechanisms, Transmissions and Automation in Design, Vol. 108, pp. 108-118, June

1986.

Kurdila, A.J., and Kamat, M.P., "Concurrent Multiprocessing Methods for Calculating Null Space and Range Space Bases for Multi-body Simulation", AIAA Journal, Vol. 28 No. 7, July 1990.

Kurdila A.J., "Concurrent Multiprocessing in Computational Mechanics for Constrained Dynamical Systems", Ph.D. Dissertation, Georgia Institute of Technology, December, 1988.

Kurdila, A.J., Papastavridis, J.G., Kamat, M.P., "On the Role of Maggi's Equations in Computational Methods for Constrained Multibody Systems", Journal of Guidance, Control and Dynamics, Vol. 13, No. 1, pp. 113-120, 1990.

Kurdila, A.J., Junkins, J.L. and Menon, R.G., "Linear Substructure Synthesis via Lyapunov Stable Penalty methods", Finite Elements in Analysis and Design, no. 10, pp. 101-123, December 1991.

Kurdila, A.J., "Lyapunov Stable Penalty Methods for Imposing Holonomic Constraints in Multibody System Dynamics," International Conference on the Dynamics of Flexible Structures in Space, Cranfield Institute of Technology, May, 1990.
Kurdila, A.J., Junkins, J.L., and Hsu, S.T., "On the Existence of Limit Cycles for a Class of Penalty Formulations of Dissipative Multibody Systems," Southeastern Congress on Theoretical and Applied Mechanics SECTAM XV, Georgia Institute of Technology, Atlanta, Georgia, March 22-23, 1990.

Kurdila, A.J. and Narcowich, F.J., "Sufficient Conditions for Penalty Formulation Methods in Analytical Dynamics", to appear in Nonlinear Dynamics.

Luh, J., Walker, M., and Paul, R., "On-line Computational Scheme for Mechanical Manipulators," presented at the 2nd IFAC/IFIP Symposium on Information Control Problems, Manufacturing Technology, Stuttgart, Germany, October, 1979.

Oden, J.T., Qualitative Methods in Nonlinear Mechanics, John Wiley & Sons, Inc., New York, 1990.

Orin, D.E., McGhee, R.B., Vukobratovic, M. and Hartoch G., "Kinematic and Kinetic Analysis of Open-chain linkages utilizing Newton-Euler Methods", Mathematics of Biosciences, Vol. 43, pp. 107-130, 1979.

Park, K.C., Chiou, J.C., Downer, J.D., "Staggered Solution Procedures for Multibody Dynamics Simulation", paper No. AIAA-90-1246-CP, 1991.

Park, K.C. and Chiou, J.C., "Stabilization of Computational Procedures for Constrained Dynamical Systems", Journal of Guidance Control and Dynamics, vol. 11, No. 4, pp. 365-370, 1988.

Park, K.C., Chiou, J.C. and Downer, J.D., "Explicit-Implicit Staggered Procedure for Multibody Dynamics Analysis", Journal of Guidance Control and Dynamics, Vol. 13, No. 3, May-June, 1990.

Petzold, L., "Differential/Algebraic Equations are not ODE's", SIAM Journal of Scientific and Statistical Computing, Vol. 3, No. 3, September 1982.

Placek, B., "Contribution to the Solution of Equations of Motion of the Discrete Dynamical System with Holonomic Constraints", in The Theory of Machines and Mechanisms, pp. 379-382, Pergamon Press, Oxford, 1987.

Rosenthal, D.E., "An Order-N Formulation for Robotic Systems", Journal of Astronautical Sciences, pp. 511-530, October-December, 1990.

Rodriguez, G., "Spatially Recursive Filtering and Smoothing Methods for Multibody Dynamics", in Proceedings of the SDIO/NASA Workshop on Multibody Simulations, G. Man and R. Laskin, editors, pp. 1094-1121, Jet Propulsion Lab, Pasadena, CA, 1987.

Rodriguez, G., "Recursive Multirigid Body Topological Tree Dynamics via Kalman Filtering and Bryson-Frazier Smoothing", Proceedings of the Sixth VPI & SU Symposium on Control of Large Structures, pp. 45-60, June 1987.

Serna, M.A. and Bayo, E., "Numerical Implementation of Penalty Methods for the Analysis of Elastic Mechanisms", Trends and Developments in Mechanisms, Machines and Robotics - 1988, pp. 449-456, ASME, New York.

Singh, R.P., and Likins, P.W., "Singular Value Decomposition for Constrained Dynamical Systems", Journal of Applied Mechanics, Vol. 52, pp. 943-948, December 1985.

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

Wehage, R.A., and Haug, E.J., "Generalized Coordinate Partitioning for Dimension Reduction in Analysis of Constrained Dynamic Systems", ASME Journal of Mechanical Design, Vol. 104, pp. 247-255, January 1982.

Won, C.C., Sparks, D., Blevin, K., Sulla, J., "Application of Piezoelectric Devices to Vibration Suppression: From Modeling and Controller Designs to Implementation", preprint of paper No. 92-4610, to be presented at the AIAA Guidance, Navigation and Control Conference, Hilton Head, SC, August, 1992.

Non-recursive Augmented Lagrangian Algorithms for the Forward and Inverse Dynamics of Constrained Flexible Multibodies

Eduardo Bayo

Ragnar Ledesma

Department of Mechanical Engineering
University of California
Santa Barbara, CA 93106

ABSTRACT

A technique is presented for solving the inverse dynamics of flexible planar multibody systems. This technique yields the non-causal joint efforts (inverse dynamics) as well as the internal states (inverse kinematics) that produce a prescribed nominal trajectory of the end effector. A non-recursive global Lagrangian approach is used in formulating the equations of motion as well as in solving the inverse dynamics equations. Contrary to the recursive method previously presented, the proposed method solves the inverse problem in a systematic and direct manner for both open-chain as well as closed-chain configurations. Numerical simulation shows that the proposed procedure provides an excellent tracking of the desired end effector trajectory.

1. Introduction

Accurate positioning and vibration minimization of flexible multibody systems have generated considerable interest from the computational dynamics and controls communities. The advent of the new generation of very fast, lightweight robots and flexible articulated space structures has made the control of structural vibrations an important practical problem in the manufacturing and space industries.

There is a large body of literature dealing with the forward dynamic analysis of flexible multibody systems, *i.e.*, determining the resulting motion when the joint forces and external forces are given. Numerous approaches have been proposed that are either based on the moving frame method or the inertial frame counterpart (see reference [1] and references therein.) Similarly, numerous control approaches have also been proposed for position and vibration control of flexible articulated structures (see reference [2] and references therein.)

Solutions to the non-collocated control of flexible articulated structures have been presented in [3-6]. The so-called inverse dynamics joint actuation are non-causal or time-delayed joint torques (applied in negative time and future time) that are capable of positioning the end effector according to a desired trajectory. The importance of using the inverse dynamics approach to vibration control has been demonstrated recently in reference [7] where passive feedback and feedforward of the inverse dynamics torque were used to achieve an exponentially stable tracking control law that yields excellent end-point tracking of flexible articulated structures. In this paper, present a global Lagrangian approach to the solution of vibration minimization and end-point trajectory tracking.

2. Mathematical Formulation

In order to describe the dynamic modeling let us consider a generic flexible body (Fig. 1) representing a component of a flexible articulated structure. The configuration of the multi-body system can be described by two sets of coordinates: the first set corresponds to the rigid body coordinates representing the location and orientation of the body axes, with respect to the inertial frame; and the second set corresponds to the so-called deformation coordinates or nodal deformations representing the deformation of the body with respect to the body axes. Using the aforementioned choice of coordinates, the location of an arbitrary point P in a planar deformable body i is given by

$$\mathbf{r}^i = \mathbf{R}^i + \mathbf{A}^i \, \mathbf{u}^i \tag{1}$$

where $\mathbf{R}^i$ is the location of the origin of the body axes with respect to the inertial frame, $\mathbf{u}^i$ is the location of point P with respect to the body axes, and $\mathbf{A}^i$ is the rotation transformation matrix from the body axes to the inertial frame. In the three-dimensional case, the rotation transformation matrix is given by

$$\mathbf{A}^i = \begin{bmatrix} 2(\theta_0^2 + \theta_1^2) - 1 & 2(\theta_1\theta_2 - \theta_0\theta_3) & 2(\theta_1\theta_3 + \theta_0\theta_2) \\ 2(\theta_1\theta_2 + \theta_0\theta_3) & 2(\theta_0^2 + \theta_2^2) - 1 & 2(\theta_2\theta_3 - \theta_0\theta_1) \\ 2(\theta_1\theta_3 - \theta_0\theta_2) & 2(\theta_2\theta_3 + \theta_0\theta_1) & 2(\theta_0^2 + \theta_3^2) - 1 \end{bmatrix}^i \tag{2}$$

where the orientation coordinates are represented by four Euler parameters θ_0^i, θ_1^i, θ_2^i, and θ_3^i which satisfy the following identity:

$$\sum_{k=0}^{3} (\theta_k^i)^2 = 1$$

The vector $\mathbf{u}^i$ can be decomposed into

$$\mathbf{u}^i = \mathbf{u}_r^i + \mathbf{u}_f^i \tag{3}$$

where $\mathbf{u}_r^i$ is the position vector of point P in the undeformed state with respect to the body axes, and $\mathbf{u}_f^i$ is the deformation vector of point P with respect to the body axes. Differentiating Eq. (1) with respect to time yields the velocity vector of point P

$$\dot{\mathbf{r}}^i = \dot{\mathbf{R}}^i + \dot{\mathbf{A}}^i \, \mathbf{u}^i + \mathbf{A}^i \, \dot{\mathbf{u}}_f^i \tag{4}$$

where $\dot{(\,)}$ represents differentiation with respect to time. To separate the generalized coordinates, the second term on the right hand side of Eq. (4) may be written as

$$\dot{\mathbf{A}}^i \, \mathbf{u}^i = -2 \, \mathbf{A}^i \, \tilde{\mathbf{u}}^i \, \mathbf{E}^i \, \dot{\boldsymbol{\theta}}^i \tag{5}$$

where $\mathbf{E}^i$ is a matrix that depends linearly on the Euler parameters and is given by

$$\mathbf{E}^i = \begin{bmatrix} -\theta_1 & \theta_0 & \theta_3 & -\theta_2 \\ -\theta_2 & -\theta_3 & \theta_0 & \theta_1 \\ -\theta_3 & \theta_2 & -\theta_1 & \theta_0 \end{bmatrix}^i \tag{6}$$

and $\tilde{\mathbf{u}}^i$ is a 3 x 3 skew-symmetric matrix given by

$$\tilde{\mathbf{u}}^i = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix}^i \tag{7}$$

where u_x, u_y, and u_z are the coordinates of the generic point P with respect to the body axes, in the deformed configuration.

The deformation vector $\mathbf{u}_f^i$ can be expressed in terms of the nodal deformations by using a finite element discretization scheme

$$\mathbf{u}_f^i = \mathbf{N}^i \ \mathbf{q}_f^i \tag{8}$$

where $\mathbf{N}^i$ is the shape function matrix and $\mathbf{q}_f^i$ is the nodal deformation vector. Since the shape function matrix is time-invariant, the time derivative of the deformation vector becomes

$$\dot{\mathbf{u}}_f^i = \mathbf{N}^i \ \dot{\mathbf{q}}_f^i \tag{9}$$

where $\dot{\mathbf{q}}_f^i$ is the time derivative of the vector of nodal deformations. Substituting Eqs. (5) and (9) into Eq. (4), we obtain the following expression for the velocity vector in terms of the rigid body coordinates and nodal deformation coordinates:

$$\dot{\mathbf{r}}^i = \dot{\mathbf{R}}^i - 2\,\mathbf{A}^i\,\tilde{\mathbf{u}}^i\,\mathbf{E}^i\,\dot{\boldsymbol{\theta}}^i + \mathbf{A}^i\,\mathbf{N}^i\,\dot{\mathbf{q}}_f^i \tag{10}$$

Using Eq. (10) to describe the velocity vector of an arbitrary point P, the kinetic energy of body i can be expressed in the following quadratic form in velocities

$$K.E.^i = \frac{1}{2}\left[\dot{\mathbf{R}}^T \dot{\boldsymbol{\theta}}^T \dot{\mathbf{q}}_f^T\right]^i \begin{bmatrix} \mathbf{m}_{RR} & \mathbf{m}_{R\theta} & \mathbf{m}_{Rf} \\ \mathbf{m}_{\theta R} & m_{\theta\theta} & \mathbf{m}_{\theta f} \\ \mathbf{m}_{fR} & \mathbf{m}_{f\theta} & \mathbf{m}_{ff} \end{bmatrix}^i \begin{bmatrix} \dot{\mathbf{R}} \\ \dot{\boldsymbol{\theta}} \\ \dot{\mathbf{q}}_f \end{bmatrix}^i \tag{11}$$

where the constant submatrices $\mathbf{m}_{RR}$ and $\mathbf{m}_{ff}$ represent the total mass of the body and the consistent finite element mass matrix, respectively. The submatrix $m_{\theta\theta}$ represents the moment of inertia of the deformable body about the origin of the body axes, and the submatrix $\mathbf{m}_{\theta R}$ represents the first moment of mass of the deformable body about the body axes. The submatrices $\mathbf{m}_{fR}$ and $\mathbf{m}_{f\theta}$ represent the kinematic coupling between the rigid body coordinates and the nodal deformation coordinates.

The potential energy due to linear elastic strains in the material can be expressed in the following quadratic form in rigid body coordinates and nodal deformation coordinates

$$P.E.^i = \frac{1}{2}\left[\mathbf{R}^T \boldsymbol{\theta}^T \mathbf{q}_f^T\right]^i \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \mathbf{k}_{ff} \end{bmatrix}^i \begin{bmatrix} \mathbf{R} \\ \boldsymbol{\theta} \\ \mathbf{q}_f \end{bmatrix}^i \tag{12}$$

where $\mathbf{k}_{ff}$ is the conventional finite element stiffness matrix. The singularity of the stiffness matrix can be eliminated by imposing appropriate boundary conditions or by choosing vibration modes that are consistent with the boundary conditions.

2.1. Equations of Motion

In order to unify the equations formulated in rigid body dynamics and structural dynamics, we make use of generalized coordinates which include rigid body coordinates and deformation coordinates, hence

$$\mathbf{q}^i = \begin{bmatrix} \mathbf{R} \\ \boldsymbol{\theta} \\ \mathbf{q}_f \end{bmatrix}^i = \begin{bmatrix} \mathbf{q}_r \\ \mathbf{q}_f \end{bmatrix}^i \tag{13}$$

where $\mathbf{q}^i$ is the vector of generalized coordinates for body i, $\mathbf{q}_r^i$ and $\mathbf{q}_f^i$ are the rigid body coordinates and nodal deformation coordinates of body i, respectively. The kinetic energy of the body can therefore be expressed by

$$K.E.^i = \frac{1}{2}\,\dot{\mathbf{q}}^{iT}\,\mathbf{M}^i\,\dot{\mathbf{q}}^i \tag{14}$$

where $\mathbf{M}^i$ is the overall mass matrix of body i. Similarly, the potential energy of the body due to linear elastic deformation can be expressed by

$$P.E.^i = \frac{1}{2}\,\mathbf{q}^{iT}\,\mathbf{K}^i\,\mathbf{q}^i \tag{15}$$

where $\mathbf{K}^i$ is the overall stiffness matrix of body i.

When reference coordinates such as those described in this paper are employed in multibody systems, the generalized coordinates are not independent because the motion of specific points in different bodies are related according to the type of mechanical joint that connects the contiguous bodies. Moreover, in flexible mechanical systems, the deformation of a component affects the configuration of adjacent components. The interdependence of the generalized coordinates are expressed by a vector of kinematic constraint equations, such as

$$\mathbf{\Phi}(\mathbf{q},t) = 0 \tag{16}$$

where $\mathbf{q}$ is the total vector of system generalized coordinates, t is time, and $\mathbf{\Phi}$ is the vector of linearly independent holonomic constraint equations. For example, if a revolute joint connects two flexible planar bodies i and j at points P and Q shown in Fig. 2, two constraint equations corresponding to the constraint condition that requires points P and Q to be coincident can be written as

$$\left[\mathbf{R}^i + \mathbf{A}^i\,\mathbf{u}_P^i\right] - \left[\mathbf{R}^j + \mathbf{A}^j\,\mathbf{u}_Q^j\right] = 0 \tag{17}$$

In general, holonomic constraints can also be explicit functions of time as well as generalized coordinates, as in the case of imposing the coordinates of the end-effector to follow a desired trajectory.

Using Lagrange's equations for a system with constrained coordinates, the system equations of motion will take the form

$$\mathbf{M}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{C}\,\dot{\mathbf{q}} + \mathbf{K}\,\mathbf{q} + \mathbf{\Phi}_\mathbf{q}^\mathrm{T}\,\boldsymbol{\lambda} = \mathbf{Q}_e + \mathbf{Q}_v(\mathbf{q},\dot{\mathbf{q}}) \tag{18}$$

where $\mathbf{M}$, $\mathbf{C}$ and $\mathbf{K}$ are the system mass, system damping and system stiffness matrices, respectively, $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers associated with the constraints, $\mathbf{\Phi}_\mathbf{q}$ is the constraint Jacobian matrix, $\mathbf{Q}_e$ is the vector of applied external forces, and $\mathbf{Q}_v$ is the quadratic velocity vector. The quadratic velocity vector contains the centrifugal forces and Coriolis forces that result from the differentiation of the kinetic energy expression with respect to the generalized coordinates.

In a forward dynamic analysis, *i.e.*, finding the resulting motion given the applied joint forces and external forces, Eqs. (16) and (18) form a mixed system of differential-algebraic equations (DAE's) that have to be solved simultaneously. The solution to the inverse dynamics problem requires a forward dynamic analysis within an iteration process. We solve the forward dynamics problem by using the augmented Lagrangian penalty formulation [8]. The augmented Lagrangian penalty formulation obviates the need to solve a mixed set of differential-

algebraic equations and does not increase the number of equations to account for the constraints. Applying the augmented Lagrangian penalty formulation to Eqs. (16) and (18) will result in the following equation:

$$M(q)\,\ddot{q} + C\,q + K\,q + \Phi_q^T\,\alpha\left[\ddot{\Phi} + 2\,\mu\,\omega\,\dot{\Phi} + \omega^2\,\Phi\right] = Q_e + Q_v(\dot{q},q) - \Phi_q^T\,\lambda^* \quad (19)$$

where α is a diagonal matrix of penalty factors whose elements are large real numbers that will assure the satisfaction of constraints, ω and μ are diagonal matrices representing the natural frequencies and damping characteristics of the dynamic penalty system associated with the constraints. The augmented Lagrangian method requires an iteration for the correct value of the Lagrange multipliers. The recursive equation for the Lagrange multipliers is given by

$$\lambda_{i+1}^* = \lambda_i^* + \alpha\left[\ddot{\Phi} + 2\,\mu\,\omega\,\dot{\Phi} + \omega^2\,\Phi\right] \quad (20)$$

2.2. Inverse Kinematics and Inverse Dynamics

In the context of end-point motion and vibration control, the inverse dynamics refers to the problem of finding the actuating forces or torques that will cause the end-point of a flexible articulated structure to follow a desired trajectory. The study done in reference [9] showed that because of the non-minimum phase character of the inverse problem, the stable solution has to be non-causal, *i.e.*, actuation is required before the end-point has started to move as well as after the the end-point has stopped. In this paper, we use a global Lagrangian approach to solve the planar inverse dynamics problem. The equations of motion are partitioned to yield explicit expressions for the joint actuations and linearized elastic equations of motion that are readily suitable for non-causal inversion.

In partitioned form, Eq. (18) can be written as

$$\begin{bmatrix} m_{RR} & m_{R\theta} & m_{Rf} \\ m_{\theta R} & m_{\theta\theta} & m_{\theta f} \\ m_{fR} & m_{f\theta} & m_{ff} \end{bmatrix} \begin{bmatrix} \ddot{R} \\ \ddot{\theta} \\ \ddot{q}_f \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & c_{ff} \end{bmatrix} \begin{bmatrix} \dot{R} \\ \dot{\theta} \\ \dot{q}_f \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & k_{ff} \end{bmatrix} \begin{bmatrix} R \\ \theta \\ q_f \end{bmatrix} + \begin{bmatrix} \Phi_R^T \\ \Phi_\theta^T \\ \Phi_{q_f}^T \end{bmatrix} \lambda = \begin{bmatrix} Q_{eR} \\ Q_{e\theta} \\ Q_{ef} \end{bmatrix} + \begin{bmatrix} Q_{vR} \\ Q_{v\theta} \\ Q_{vf} \end{bmatrix} \quad (21)$$

The second set of equations in Eq. (21) can be rearranged to express the externally applied joint forces as

$$Q_{e\theta} = m_{\theta R}\,\ddot{R} + m_{\theta\theta}\,\ddot{\theta} + m_{\theta f}\,\ddot{q}_f + \Phi_\theta^T\,\lambda - Q_{v\theta} \quad (22)$$

Eq. (22) is the inverse dynamics equation that yields the joint forces (torques) necessary for the end-point to follow a prescribed trajectory. In order to obtain $Q_{e\theta}$, the nodal acceleration vector $\ddot{q}_f$ is needed. This vector can be obtained from the third set of equations in Eq. (21), which can be written as

$$m_{ff}\,\ddot{q}_f + c_{ff}\,\dot{q}_f + k_{ff}\,q_f = Q_{ef} + Q_{vf} - \Phi_{q_f}^T\,\lambda - m_{fR}\,\ddot{R} - m_{f\theta}\,\ddot{\theta} \quad (23)$$

The linearized form of Eq. (23) makes the nonlinear inversion problem amenable to successive linear inversion techniques. The vector of applied nodal forces Q_{ef} can be expressed in terms of the externally applied torques through the following mapping:

$$Q_{ef} = G_f\,Q_{e\theta} \quad (24)$$

where in the planar case, the matrix G_f is a constant matrix which maps the externally applied torques to the vector of externally applied nodal forces. Substituting Eqs. (22) and (24) into

Eq. (23) results in

$$\mathbf{m}_{ff}\,\ddot{\mathbf{q}}_f + \mathbf{c}_{ff}\,\dot{\mathbf{q}}_f + \mathbf{k}_{ff}\,\mathbf{q}_f = \mathbf{G}_f\,\mathbf{m}_{\theta f}\,\ddot{\mathbf{q}}_f + \mathbf{F}_1(\lambda,\mathbf{q}_r,\dot{\mathbf{q}}_r,\ddot{\mathbf{q}}_r,\mathbf{q}_f,\dot{\mathbf{q}}_f) \tag{25}$$

where $\mathbf{F}_1$ is a force vector that includes the inertial terms, reaction terms between contiguous bodies, and quadratic velocity terms.

The inertial coupling submatrix $\mathbf{m}_{\theta f}$ can be decomposed into a sum of a time-invariant matrix and a time-varying matrix

$$\mathbf{m}_{\theta f} = \mathbf{m}^c_{\theta f} + \mathbf{m}^t_{\theta f} \tag{26}$$

where $\mathbf{m}^c_{\theta f}$ and $\mathbf{m}^t_{\theta f}$ are the time-invariant part and time-varying part of $\mathbf{m}_{\theta f}$, respectively. This decomposition is essential for the iteration process needed to obtain the solution as explained below. Substituting Eq. (26) into Eq. (25), we obtain the inverse kinematics equation of motion for the nodal displacements $\mathbf{q}_f$:

$$\mathbf{m}^*_{ff}\,\ddot{\mathbf{q}}_f + \mathbf{c}_{ff}\,\dot{\mathbf{q}}_f + \mathbf{k}_{ff}\,\mathbf{q}_f = \mathbf{F}(\lambda,\mathbf{q}_r,\dot{\mathbf{q}}_r,\ddot{\mathbf{q}}_r,\mathbf{q}_f,\dot{\mathbf{q}}_f,\ddot{\mathbf{q}}_f) \tag{27}$$

where

$$\mathbf{m}^*_{ff} = \mathbf{m}_{ff} - \mathbf{G}_f\,\mathbf{m}^c_{\theta f} \tag{28}$$

The problem statement for the inverse kinematics is that of finding the internal states $\mathbf{q}_f$ so that the end-point coordinates characterized by a subset of the rigid body coordinates $\mathbf{q}_r$ follow a prescribed trajectory. The mass matrix $\mathbf{m}^*_{ff}$ is nonsymmetric and it is precisely the nonsymmetry of the mass matrix that produces internal states which are non-causal with respect to the end-point motion. Eq. (27) is a nonlinear differential equation in the variable $\mathbf{q}_f$. As explained below, Eq. (27) is solved iteratively in the frequency domain to yield the nodal deformation vector $\mathbf{q}_f$ that is non-causal with respect to the end-point motion.

In the frequency domain, Eq. (27) can be written as a set of complex equations for a particular frequency $\bar{\omega}$

$$\left[\mathbf{m}^*_{ff} + \frac{1}{i\bar{\omega}}\,\mathbf{c}_{ff} - \frac{1}{\bar{\omega}^2}\,\mathbf{k}_{ff}\right]\hat{\ddot{\mathbf{q}}}_f(\bar{\omega}) = \hat{\mathbf{F}}(\bar{\omega}) \tag{29}$$

where $\hat{\ddot{\mathbf{q}}}_f(\bar{\omega})$ is the Fourier transform of $\ddot{\mathbf{q}}_f(t)$ and $\hat{\mathbf{F}}(\bar{\omega})$ is the Fourier transform of $\mathbf{F}(t)$. Eq. (29) is based on the assumption that $\ddot{\mathbf{q}}_f(t)$ and $\mathbf{F}(t)$ are Fourier transformable. This assumption is valid for slewing motions which are from rest to rest. The nodal acceleration vector $\ddot{\mathbf{q}}_f(\bar{\omega})$ can be obtained directly from Eq. (29) for each frequency $\bar{\omega}$. The leading matrix of Eq. (29) is a complex regular matrix that is invertible for all frequencies except for $\bar{\omega} = 0$. However, for $\bar{\omega} = 0$, the system undergoes a rigid body motion determined only by the invertible mass matrix $\mathbf{m}^*_{ff}$. The nodal accelerations in the time domain may be obtained through the application of the inverse Fourier transform, *i.e.*,

$$\ddot{\mathbf{q}}_f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{\ddot{\mathbf{q}}}_f(\bar{\omega})\,e^{i\bar{\omega}t}\,d\bar{\omega} \tag{30}$$

Once the non-causal nodal accelerations are known, Eq. (22) can be used to explicitly compute the non-causal inverse dynamics joint efforts that will move the end effector according to a desired trajectory. We note, however, that the inverse dynamics torque and internal

states given by Eqs. (22) and (27), respectively, depend on the Lagrange multipliers and rigid body coordinates, which in turn depend on the internal states and the applied torque. Moreover, the rigid body coordinates and Lagrange multipliers are different from their nominal values when the components of the multibody system are flexible. Therefore, a forward dynamic analysis is required to obtain an improved estimate of the generalized coordinates and Lagrange multipliers given the torques computed previously using nominal values of rigid body coordinates and Lagrange multipliers. In order to ensure that the iteration process converges to obtain the joint efforts that will cause the end-effector to follow the desired trajectory, the forward dynamics analysis is carried out with the additional constraint that the coordinates of the end-point follow the desired trajectory. These additional constraints have corresponding Lagrange multipliers which act as correcting terms to the joint efforts that have been previously calculated.

To summarize, the procedure for obtaining the inverse dynamics solution for flexible multibody systems involve the following steps:

Algorithm:

1. Perform a rigid body inverse dynamic analysis to obtain the nominal values of the rigid body coordinates $\mathbf{q}_r$ and Lagrange multipliers $\boldsymbol{\lambda}$.
2. Solve the inverse kinematics equation represented by Eq. (27) to obtain the time-delayed nodal accelerations $\ddot{\mathbf{q}}_f$.
3. Compute the inverse dynamics joint efforts $\mathbf{Q}_{e\theta}$ using Eq. (22).
4. Perform a forward dynamic analysis using Eq. (19) to obtain new values for the generalized coordinates and Lagrange multipliers.
5. Repeat steps 2 through 4 until convergence in the inverse dynamics torques is achieved.

3. Simulation Results

We present in this section the results of numerical simulations that verify the procedure discussed above. First, we apply the global Lagrangian approach to an open-chain flexible multibody system and compare the results with those obtained by the recursive Newton-Euler method [5] to test the validity of the proposed procedure. Next, we present the results of the application of the global Lagrangian approach to a closed-chain flexible multibody system to determine the inverse dynamics torque that will produce the desired motion at the end effector.

3.1. Open-Chain Multibody System

Fig. 3 shows a two-link flexible multibody system in the horizontal plane. The end-point of the second link is specified to move along the x-axis according to the acceleration profile described by Fig. 5, which corresponds to an end-point displacement of 0.483 meters along the x-axis. The geometric and material properties of the links are:

First Link:

Length: 0.66 m
Cross-section area: $1.2 \times 10^{-4}\ m^2$
Cross-section moment of inertia: $2.3 \times 10^{-10}\ m^4$

Second Link:

Length: 0.66m
Cross-section area: $4.0 \times 10^{-5}\ m^2$
Cross-section moment of inertia: $8.5 \times 10^{-12}\ m^4$

The two links share the following properties:

Young's modulus: 14 GPa
Mass density: 2715 kg/m^3

In Fig. 6, the inverse dynamics torque profile for the base motor using the global Lagrangian method is superimposed on the inverse dynamics torque profile determined by the recursive Newton-Euler method. The inverse dynamics torque profiles for the elbow motor computed by the two aforementioned methods are superimposed in Fig. 7. Both the recursive and global formulations yield the same result and superimpose to each other, thus validating the proposed method. The corresponding rigid body torques are also shown in Figs. 6 and 7 to illustrate the pre-actuation and post-actuation present in the inverse dynamics torque profiles.

3.2. Closed-Chain Multibody System

Fig. 4 shows a closed-chain flexible multibody system made up of four flexible links with two joints which are fixed against translation relative to the ground. As in the open-chain case, the multibody system is assumed to lie on a horizontal plane so that gravity effects are neglected. The desired trajectory of joint 5 is a straight line 45 degrees with respect to the x and y axes. The x and y-components of the acceleration of joint 5 are specified to follow the acceleration profile shown in Fig. 8. The four links share the following geometric and material properties:

Length: 0.60 m
Cross-section area: $4.0 \times 10^{-5}\ m^2$
Cross-section moment of inertia: $8.5 \times 10^{-12}\ m^4$
Young's modulus: 14 GPa
Mass density: 2715 kg/m^3

Fig. 9 shows the inverse dynamics torque profile at joint 1 obtained by the global Lagrangian method. The rigid body inverse dynamics torque profile is superimposed for comparison. The figure shows the noncausal character of the solution for the inverse problem. Fig. 10 shows the inverse dynamic torque profile at joint 3 superimposed with the corresponding rigid body inverse dynamics torque profile. Again, the time delay due to the noncausality of the solution is seen in this figure.

Fig. 11 shows the elastic angular rotation at the base of the first link obtained by a feedforward of the inverse dynamics torque. Superimposed in the same figure is the corresponding elastic angular rotation obtained by a feedforward of the rigid body torque. Whereas the rigid body torque produces residual angular rotations, the inverse dynamic torque does not show

residual angular rotations. As a matter of fact, it has been observed in the simulations that the rigid body torques produced residual vibration in all the nodal deformations while the inverse dynamics torques eliminated the residual oscillation. Furthermore, the inverse dynamics torques produced nodal deformations which exhibit non-causal characteristics with respect to the end-point motion. Fig. 12 shows a comparison of the vertical tip error at joint 5 obtained by a feedforward of the inverse dynamics torque with the tip error resulting from a feedforward of the rigid body torque. This figure shows that the inverse dynamics torque provides an excellent tracking of the desired end effector trajectory whereas the rigid body torque again induces substantial vibration on the end-point motion.

4. Conclusion

A global Lagrangian approach for the inverse dynamics of flexible multibody systems has been presented. The procedure is capable of solving for the inverse dynamics torque profiles of both open-chain and closed-chain flexible multibody systems in a unified and systematic manner. The method is found to produce an excellent tracking of the desired trajectory of the end effector. In a future paper, we will address the inverse dynamics problem for flexible multibody systems undergoing motion in three dimensions. New problems arise in the three-dimensional case, since the actuating torque vectors have directions which are time-varying and nonlinear functions of the rigid body coordinates, as contrasted with the planar case where the applied torque vectors have directions fixed perpendicular to the plane of the multibody system. In addition, to be able to perform the inverse kinematics and inverse dynamics analyses, additional actuation at the joints may be necessary.

Acknowledgements

The support of this work by the Air Force Office of Scientific Research under contract no. F49620-91-C-0095 and by TRW is gratefully acknowledged.

References

1. A. A. Shabana, *Dynamics of Multibody Systems*, John Wiley & Sons, Inc., 1989.
2. M. C. Oakley and R. H. Cannon, "Anatomy of an experimental two-link flexible manipulator under end-point control," *Proceedings, 29th IEEE Conference on Decision and Control*, Honolulu, Hawaii, December 1990.
3. E. Bayo, "A finite-element approach to control the end-point motion of a single-link flexible robot," *Journal of Robotic Systems*, vol. 4, no. 1, pp. 63-75, 1987.
4. E. Bayo and H. Moulin, "An efficient computation of the inverse dynamics of flexible manipulators in the time domain," *Proceedings, IEEE Conference on Robotics and Automation*, pp. 710-715, 1989.
5. E. Bayo, P. Papadopoulos, J. Stubbe and M. Serna, "Inverse kinematics and dynamics of a multi-link elastic robot: an iterative frequency domain," *International Journal of Robotics Research*, vol. 8, no. 6, pp. 49-62, 1989.
6. D. S. Kwon and W. J. Book, "An inverse dynamic method yielding flexible manipulator state trajectories," *Proceedings, American Control Conference* , San Diego, California, 1990.

7. B. Paden, D. Chen, R. Ledesma and E. Bayo, "Exponentially stable tracking control for multi-joint flexible-link manipulators," *ASME Journal of Dynamic Systems, Measurement and Control.* Accepted for publication.

8. E. Bayo, J. Garcia de Jalon and M. Serna, "A modified lagrangian formulation for the dynamic analysis of constrained mechanical systems," *Computer Methods in Applied Mechanics and Engineering*, vol. 71, pp. 183-195, Nov. 1988.

9. H. Moulin, "Problems in the inverse dynamics solution for flexible manipulators," *Ph.D. Thesis*, University of California, Santa Barbara, 1989.

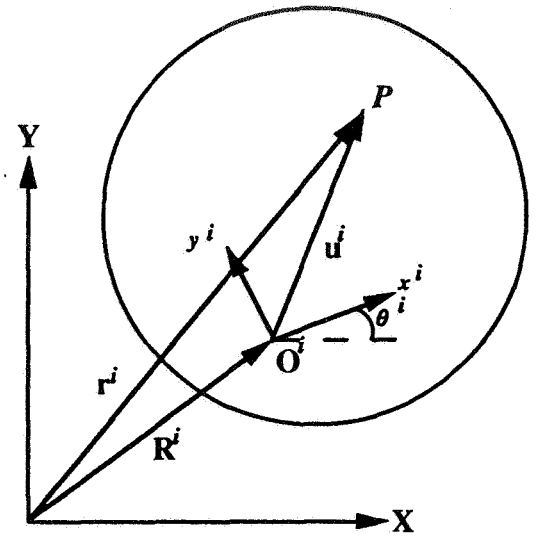


Fig. 1: Reference coordinates for planar body i

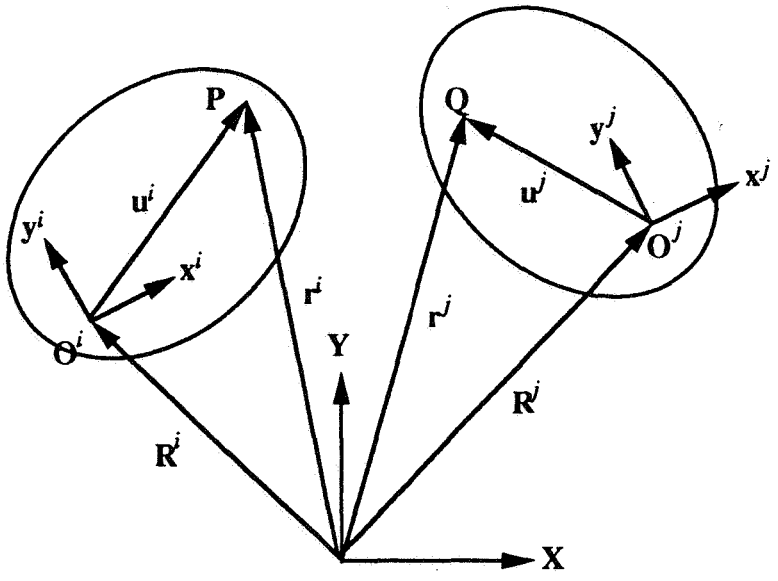


Fig. 2: A pair of flexible planar bodies

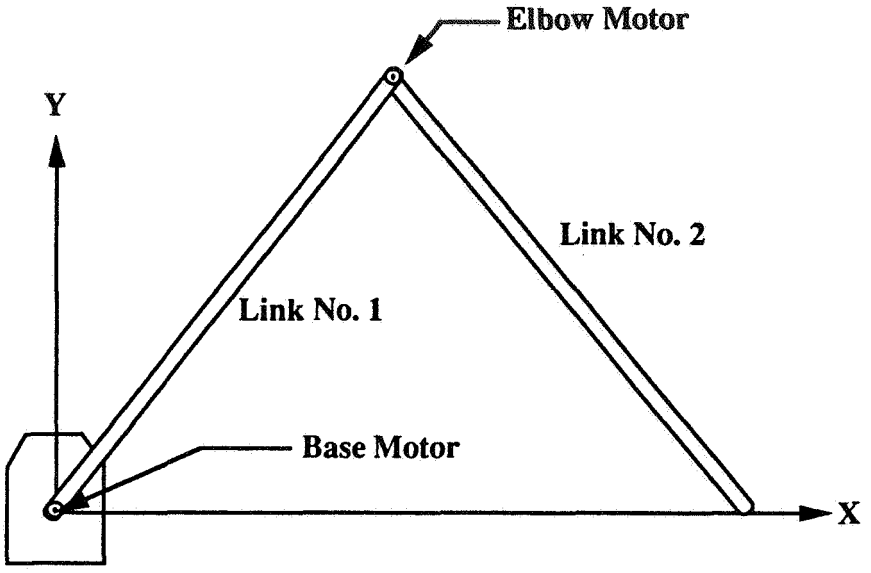


Fig. 3: Two-Link Open-Chain Flexible Multibody System

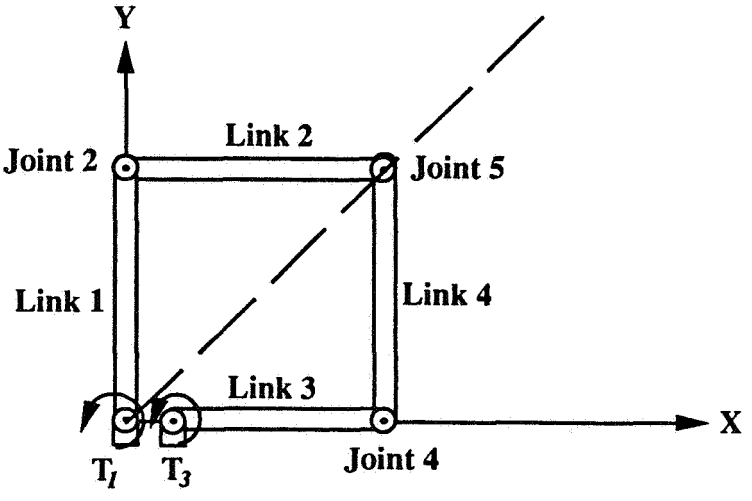


Fig.4: Closed-Chain Flexible Multibody System

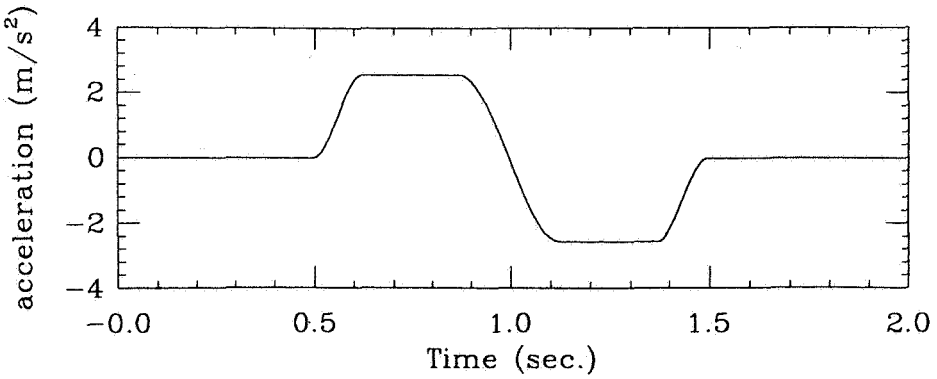


Fig. 5: end−point acceleration along the x−axis

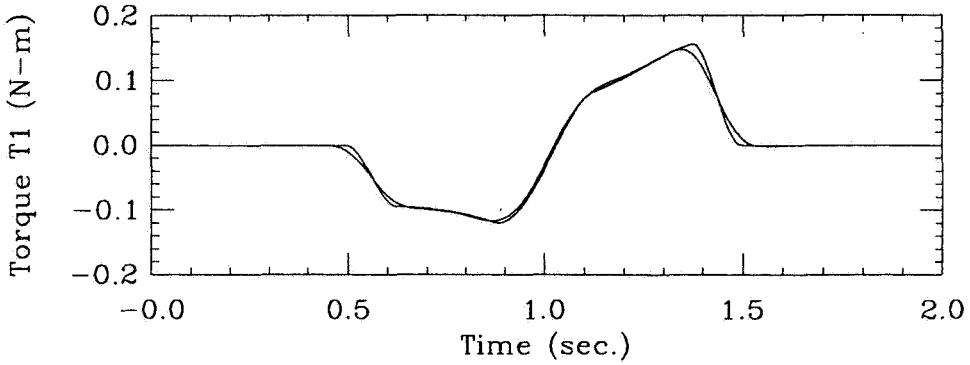


Fig. 6: inverse dynamics and rigid torque at base motor

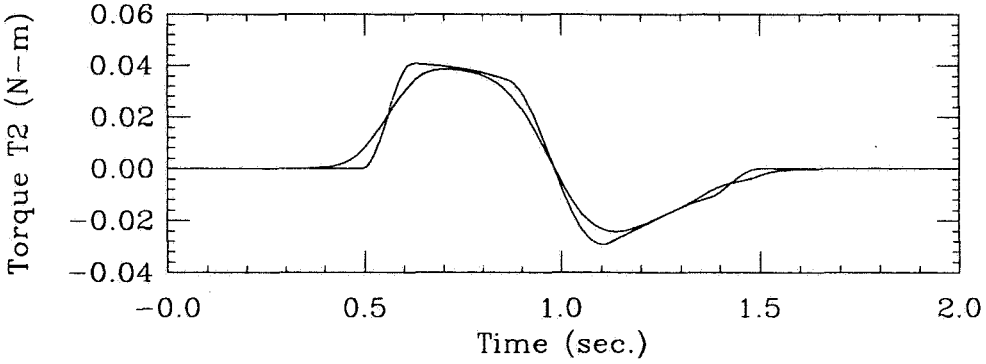


Fig. 7: inverse dynamics and rigid torque at elbow motor

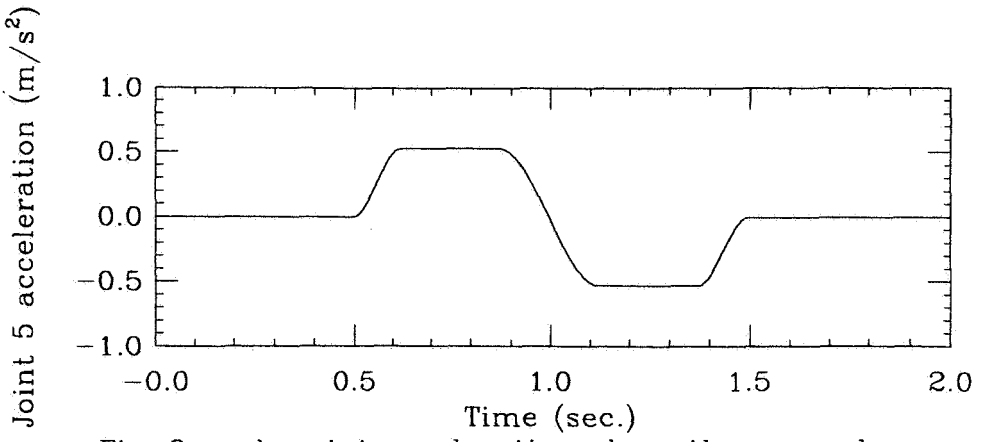


Fig. 8: end−point acceleration along the x− and y− axes

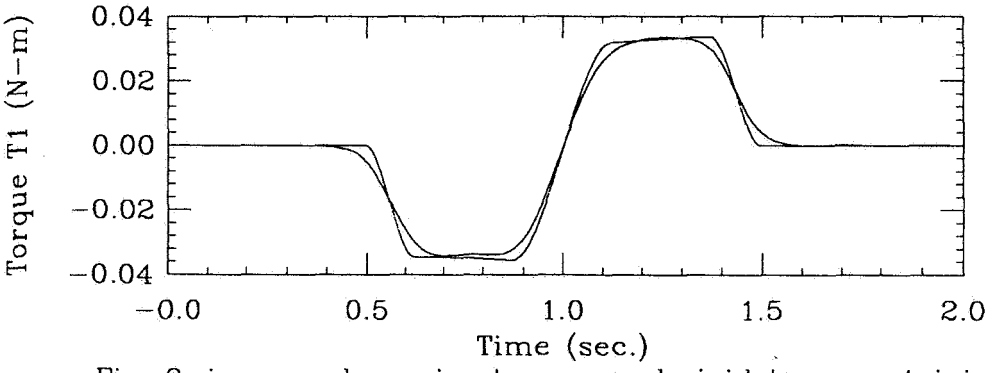


Fig. 9: inverse dynamics torque and rigid torque at joint 1

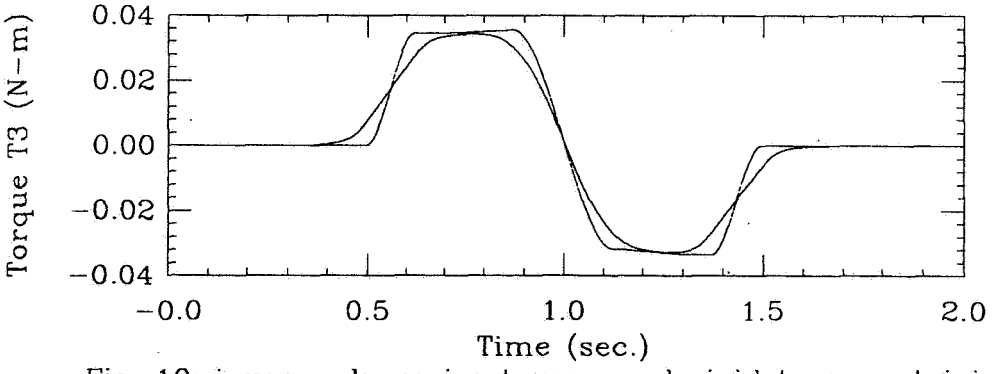


Fig. 10: inverse dynamics torque and rigid torque at joint 3

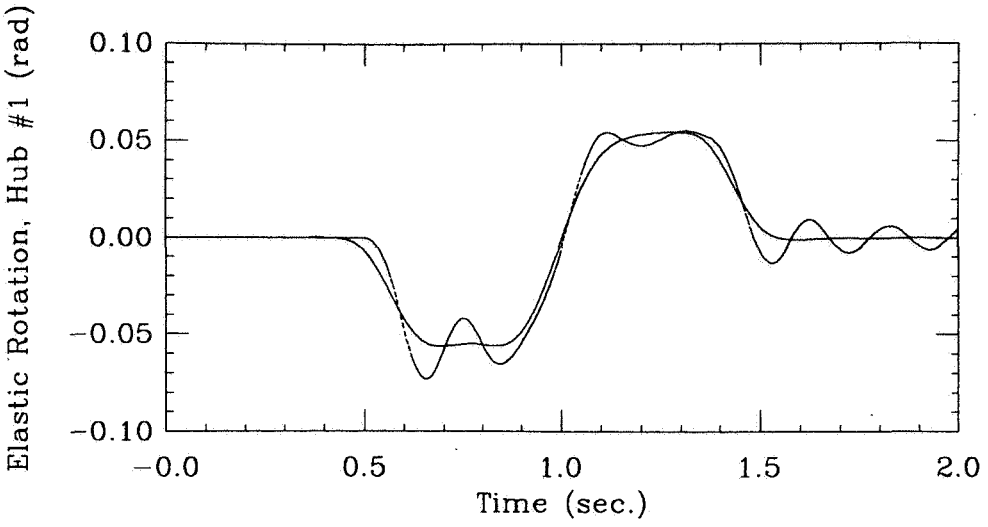


Fig. 11: elastic rotation: inverse dynamics vs. rigid torques

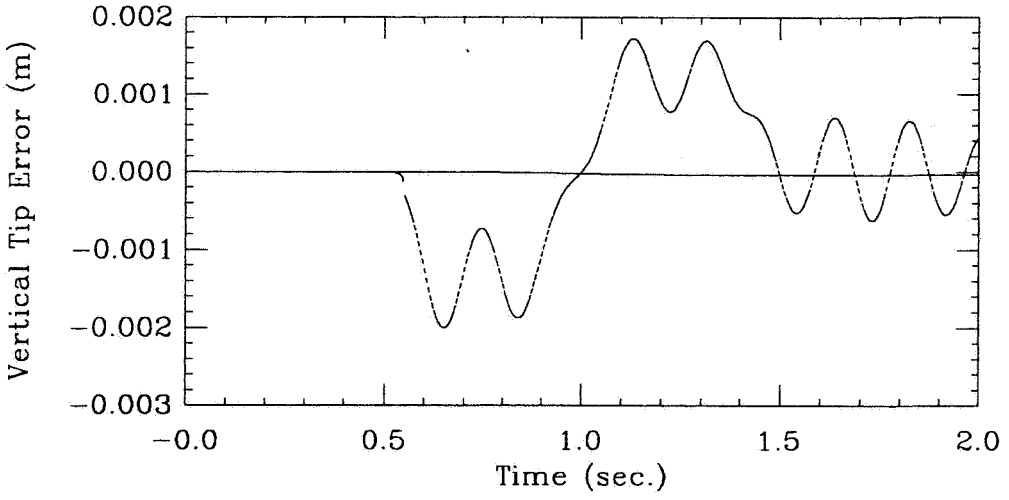


Fig. 12: vertical tip error: inverse dynamics vs. rigid torques

A Neural-Network Approach to Robotic Control

D P W Graham and G M T D'Eleuterio

Institute for Aerospace Studies
University of Toronto

Abstract

An artificial neural-network paradigm for the control of robotic systems is presented. The approach is based on the Cerebellar Model Articulation Controller created by James Albus and incorporates several extensions. First, recognizing the essential structure of multibody equations of motion, two parallel modules are used that directly reflect the dynamical characteristics of multibody systems. Second, the architecture of the proposed network is imbued with a self-organizational capability which improves efficiency and accuracy. Also, the networks can be arranged in hierarchical fashion with each subsequent network providing finer and finer resolution.

1. Introduction

The brain possesses a remarkable ability to learn and perform motor control functions without the apparent need to write out elaborate differential equations. Researchers have long been intrigued by this cerebral calculus and have made various attempts at replicating it. In an age of robotics, not only has this goal been pursued with more vigor than ever before, but the issue has also become of decidedly greater practical import.

After having had initially lost popularity, the concept of *artificial neural networks* has experienced a renaissance and the field is now flourishing. Although many of the proposed architectures bear little resemblance to the biological structures that motivated them, they have been successfully implemented in myriad applications from pattern recognition to real-time control.

In the field of robotics, one must deal with highly nonlinear systems which are in general not easily amenable to analysis. While the basic dynamics of a system can be had with a modicum of effort, accurate modeling of motor dynamics, joint friction, link damping and structural flexibility is, at the very least, an arduous task. Yet *model-based* control relies on a model and when that model is suspect one must rely on the robustness of the controller.

Artificial neural networks offer another approach to control, a *nonmodel-based* approach. Neural nets in particular have been demonstrated to be quite a viable strategy for robotic control. One of the main attractive features of neural nets is that they can 'learn.' In the present context, this means the ability to infer, through training, the dynamics of a robotic system including nonlinear characteristics that may be difficult to model by even modern techniques. Neural nets are furthermore inherently parallel, robust, fault tolerant and less susceptible to noise. Surveys of

neural-network architectures for robotic applications have been presented by Kung and Hwang (1989), Freeman and Kosko (1990) and Lee and Bekey (1991).

In debating model-based control *vs.* nonmodel-based control, it is important to observe that the choice is far from binary. Rather, these two broadly described approaches can be considered opposite ends of the same spectrum, a 'control' spectrum which is virtually continuous. Even many control approaches traditionally considered model-based depend on system identification, for example, which itself introduces an element of learning. Indeed, in many ways a neural-network approach may be regarded ultimately as an extensive form of system identification. But here too, there is much to gain by making a few simple and yet general observations about the 'model.'

This paper presents a new artificial neural-network (ANN) paradigm for robotic control. The architecture of the proposed network is founded on the work of Albus (1975, 1981) and it attempts to encode very basic knowledge about the dynamics of robotic manipulators. Several extensions to Albus's work are made including the development of a modular architecture of cooperative networks specifically tailored for mechanical systems. Our network is moreover given a self-organizing structure using the technique of Kohonen (1989). Finally, a hierarchy of these networks can be established to provide progressively more accurate representations of the system at hand. The theoretical development is complemented by simulation results showing improvements in the control of robotic systems over existing comparable methods.

2. Foundations

Our aim in the present work is to develop an ANN schema for the modeling of the inverse dynamics of a (rigid) multibody system that can be used in a computed-torque control procedure. In essence, we seek an ANN representation of motion equations of the form

$$\mathcal{M}(q)\ddot{q} + \eta(q, \dot{q}) = f_c \tag{1}$$

where q represents the system degrees of freedom and f_c is the column of (joint) control forces and/or torques. The mass matrix $\mathcal{M}(q)$ is configuration-dependent and $\eta(q, \dot{q})$ accounts for nonlinear inertial forces such as Coriolis and centrifugal forces. Gravity effects and friction are also assumed to be contained in η.

Let us, however, begin by considering the more generic mapping $f : x \mapsto y$ where $x = \text{col}\{x_1, \ldots, x_N\}$. A relatively crude representation of f can be had by creating a 'look-up table,' directly reminiscent of 'trig' and 'log' tables. Michie and Chambers (1968) used essentially such a technique in their scheme, called BOXES, to control a broom-carriage system (an inverted pendulum on a cart). The value in each 'box,' *i.e.*, table element, was learned from the response of the system to various force stimuli. The direct table look-up method, however, possesses several problems. First, the number of table elements grows exponentially as the number of input states increases. Second, information is not shared; that is, similar inputs should produce similar responses, but with BOXES and other table look-up strategies the response for each set of inputs must be learned separately.

Albus's CMAC

James Albus sought to remedy these shortcomings by developing his CMAC—Cerebellar Model Articulation Controller. This architecture was motivated by the biological motor control functions of the human cerebellum (Albus 1975, 1981). The basic concept can be best represented diagrammatically for a two-dimensional problem (with inputs x_1 and x_2), as in Figure 1. Instead of using one layer of finely divided cells, essentially the structure of BOXES, the CMAC employs several overlapping layers of coarser cells with each layer progressively shifted relative to the previous one. The example shown in Figure 1 displays four layers of coarse cells. To evaluate the function $f(x_1, x_2)$ in this case requires 'activating' four cells and summing their 'encoded values.' This quantization process is an example of what are generally called coarse-coding techniques (Rumelhart and McClelland 1988).

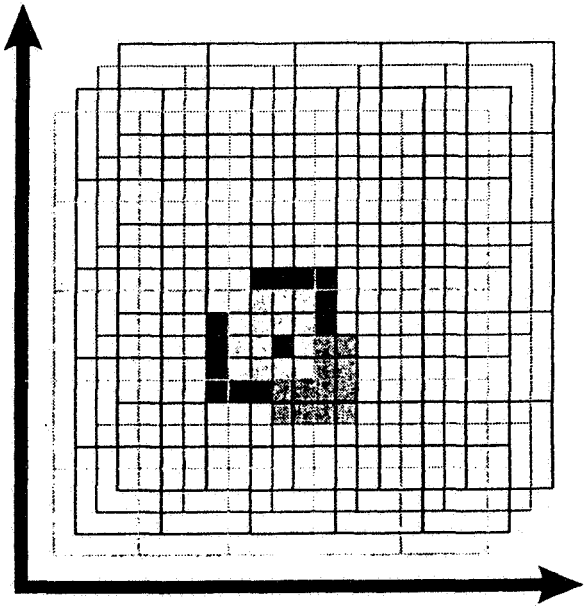


Figure 1: Two-dimensional example of CMAC discretization. Shaded areas represent the activated coarse cells for a point in the small black square.

As can be seen from Figure 1, the resolution attainable in this procedure is substantially better (depending on the number of coarse-cell layers used) than the actual size of the coarse cells. Indeed, with four layers, the CMAC can achieve the same resolution as BOXES but with only about one-third of the total number of cells. For larger problems, the savings factor could be orders of magnitude with, of course, a comparable saving in required memory space.

Albus moreover recognized that the entire input space is typically not necessary for applications to robotics. Hence, the coarse cells can be 'hashed' (randomly assigned) to a smaller

number of units, called 'granule' cells. The sparseness of the active input space will ensure (in probabilistic terms) that the number of collisions in hashing will be negligible. This hashing procedure makes more efficient use of the input space and further reduces memory requirements.

In addition, the overlapping layers in the CMAC permit the sharing of information. Returning briefly to the example of Figure 1, the value of f at any point must be reconstructed by the information (encoded value) contained in four coarse cells. However, evaluating f at a new point in a neighboring fine cell will activate three of the previous four coarse cells. Thus, some knowledge at the new point is already known from the learning done at the previous point. This process is called 'generalization' and enables one to acquire knowledge over large regions of the input domain by learning at only a relatively few points.

Although not done by Albus, the CMAC can be cast in the familiar ANN-like architecture as shown in Figure 2. Each coordinate is finely discretized into 'input units' and then coarsely discretized, according to the shifted layers or grids of coarse cells, into 'receptive units.' The receptive units activate a single coarse cell ('coarse cell unit') in each grid, which in turn can be hashed to a reduced set of 'granule cell units.' Outputs from these units are weighted and summed by the 'output unit' to yield the final output value. A complete description of this architecture can be found in Graham and D'Eleuterio (1991*a*).

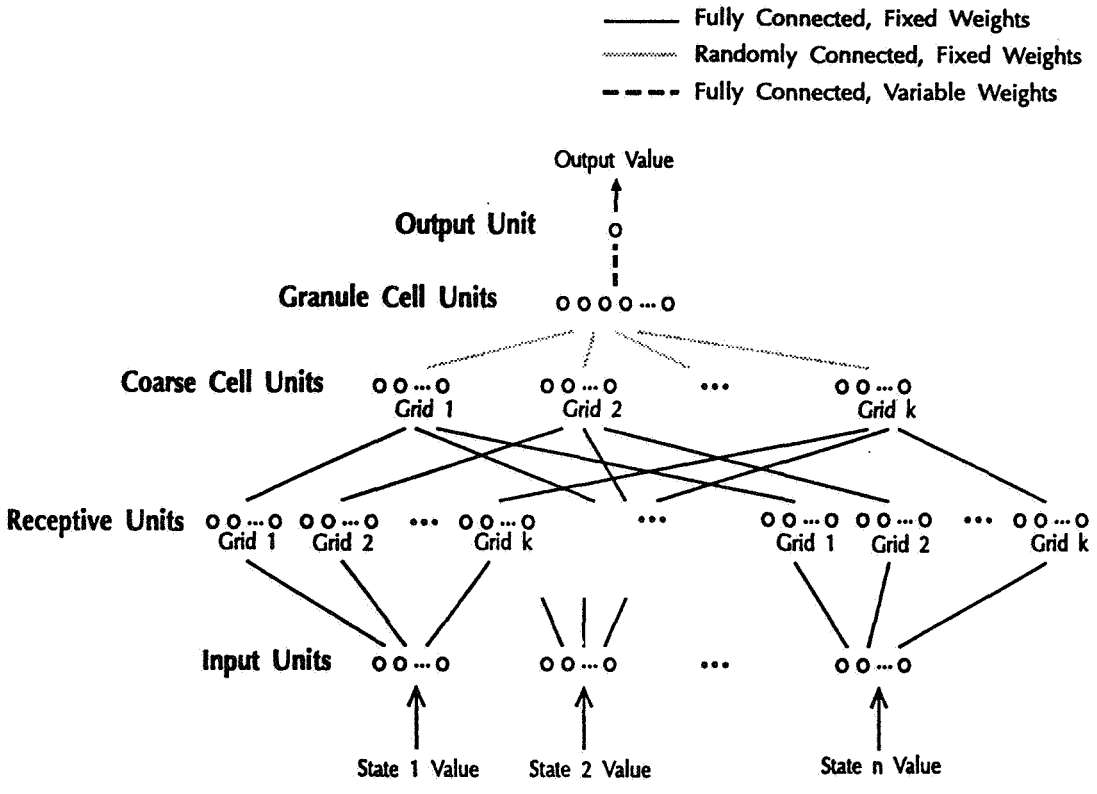


Figure 2: Architecture of the CMAC cast in the form of an ANN.

It is important to note that it is only the last layer which contains variable weights (*i.e.*, the encoded values) that must be learned. The rest of the network can be said to be 'hard-wired' or more precisely 'hard-coded.' Learning is therefore quite fast, typically orders of magnitude faster than comparable backpropagation networks.

Mathematical Formulation

The CMAC concept can in general be represented mathematically as follows:

$$\widehat{f}(\boldsymbol{x}) = \sum_{\alpha} w_\alpha \psi_\alpha(\boldsymbol{x}) \tag{2}$$

where w_α are variable weights (encoded values) and $\psi_\alpha(\boldsymbol{x})$ may be viewed as basis functions or 'receptive fields.' Also, the notation $\widehat{f}$ recognizes that the expansion (2) is in general only an approximation to f. In a CMAC, $\psi_\alpha(\boldsymbol{x})$ may be described as 'hyperbox' functions, *i.e.*, they would delineate the coarse cells:

$$\psi_\alpha(\boldsymbol{x}) \triangleq \begin{cases} 1, & \boldsymbol{x}_\alpha < \boldsymbol{x} < \boldsymbol{x}_{\text{next}(\alpha)} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where $\boldsymbol{x}_\alpha < \boldsymbol{x} < \boldsymbol{x}_{\text{next}(\alpha)}$ is to be read as $x_{1,\alpha} < x_1 < x_{1,\text{next}(\alpha)} \cdots x_{N,\alpha} < x_N < x_{N,\text{next}(\alpha)}$. Note that we must write $\boldsymbol{x}_{\text{next}(\alpha)}$ instead of $\boldsymbol{x}_{\alpha+1}$, say, since we cannot index cellular divisions consecutively in N-space; however, a functional relation, 'next(α),' can be defined. Another example of possible basis functions are the Gaussian fields used by Moody and Darken (1989).

The learning rule, the well known Delta Rule, for w_α can be expressed, for general $\psi_\alpha(\boldsymbol{x})$, as

$$\Delta w_\alpha = \gamma \frac{\psi_\alpha(\boldsymbol{x})}{\sum_\beta \psi_\beta(\boldsymbol{x})} \Delta f(\boldsymbol{x}) \tag{4}$$

where

$$\Delta f(\boldsymbol{x}) \triangleq f(\boldsymbol{x}) - \sum_{\beta} w_\beta \psi_\beta(\boldsymbol{x})$$

is the error in the mapping. For the basis functions defined by (3), the denominator

$$k \triangleq \sum_{\beta} \psi_\beta(\boldsymbol{x}) \tag{5}$$

is the number of 'activated' cells and is furthermore constant; in fact, k is the number of coarse-cell layers. Thus, we can define the 'learning rate' or 'learning coefficient' as

$$\nu \triangleq \frac{\gamma}{k} \tag{6}$$

Note that in the CMAC architecture, the error is distributed uniformly among layers.

Application to Robotics

The CMAC was designed with manipulator control specifically in mind. An implementation of the CMAC scheme in the control of a two-link planar manipulator has been performed by Miller *et al.* (1987). The input variables in this application were the joint angles, rates and accelerations ($\theta = \text{col}\{\theta_1, \theta_2\}, \dot{\theta}, \ddot{\theta}$). Two CMAC networks are used, each requiring all six inputs. The outputs from networks are the two joint control torques.

The CMAC controller measures the state of the system (θ and $\dot{\theta}$) and a trajectory planner determines the required acceleration ($\ddot{\theta}$) to drive the actual trajectory towards the desired trajectory in a prescribed number of time steps. A position-error controller is superimposed onto the CMAC controller to deal with any residual error as may arise from the CMAC discretization. The position-error controller also provides nominal control during the initial learning phase. The control system is displayed in Figure 3.

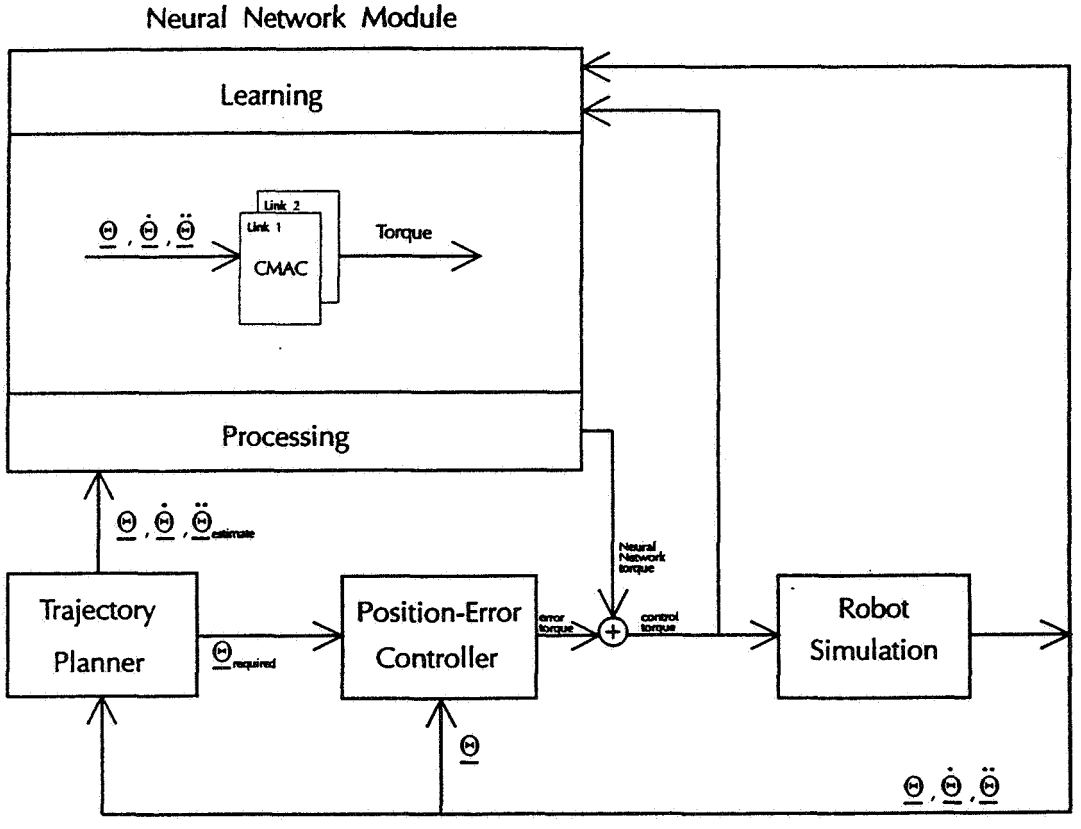


Figure 3: Robotic control system that includes a single CMAC module.

The CMAC is taught by presenting data on input torques and the corresponding measured state (plus joint accelerations) of the system. The CMAC is thus able to learn the inverse dynamics of the manipulator without direct supervised learning. Miller *et al.* show that the CMAC can learn to follow a path when presented with it only a few times. It was also found that the CMAC architecture was able to handle multiple paths, noise, different cell sizes and learning rates and it was able to adapt readily to changes in the manipulator's mass properties. Miller *et al.* (1990) have also successfully implemented the CMAC controller on a five degree-of-freedom manipulator.

3. Modular Architecture

The CMAC approach as developed by Albus and as implemented by Miller *et al.* does not assume anything about a robotic system apart from the selection of the input variables. A significant enhancement, however, can be achieved by making the merest note of the motion equations. Rewriting (1) slightly, we have

$$\boldsymbol{\mu}(\boldsymbol{q}, \ddot{\boldsymbol{q}}) + \boldsymbol{\eta}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{f}_c \tag{7}$$

(The purpose of doing this is to emphasize the fact that the structure of the first term is unimportant in the following.) It should be underscored that (7) still represents the most general form of motion equations for multibody systems.

Written as (7), it is clear that the equation of motion can be parsed into two distinct parts: One being a function of only position and acceleration, and another of only position and rate. This structure suggests a modular architecture consisting of two CMAC subnetworks, each defined on a different subset of the augmented state ($\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}}$) and, hence, each smaller than a single CMAC. Unlike other modular networks (*e.g.*, Jacobs *et al.* 1991), each subnetwork here operates cooperatively and on a different set of inputs. In addition to reducing the total memory requirements when compared to a single-CMAC implementation, this 'divide and conquer' approach possesses the very attractive feature that it captures the dynamical structure of a multibody system without explicitly encoding the motion equations. In fact, setting $\boldsymbol{\eta} = \mathbf{0}$ would yield the linearized equations of motion.

Learning Procedure

The problem in this architecture arises in learning. In application to a real robotic system, we cannot assume the separate parts of the motion equation are available to us to enable the modules to learn separately. Rather we would only have the total error produced by the network (Graham and D'Eleuterio 1991*b*). Thus, a technique to distribute the error between the two modules is needed.

In general, we can represent a modular architecture of cooperating CMACs as

$$\hat{f}(\boldsymbol{x}) = \sum_{\kappa} \sum_{\alpha} w_{\kappa,\alpha} \psi_{\kappa,\alpha}(\boldsymbol{x}_{\kappa}) \tag{8}$$

where κ is the modular index and $\boldsymbol{x}_\kappa \in \text{span}\, \boldsymbol{x}$. For the problem at hand, there are only two modules which can be identified by $\kappa = \mu$ for the first (rate-linear) module and $\kappa = \eta$ for the second (rate-nonlinear) one. The proposed learning rule may be expressed as

$$\Delta w_{\kappa,\alpha} = \nu \rho_\kappa(\boldsymbol{x}) \psi_{\kappa,\alpha}(\boldsymbol{x}_\kappa) \Delta f(\boldsymbol{x}) \tag{9}$$

where $\Delta f(\boldsymbol{x})$ is the error given by the mapping (8) relative to the desired value and the 'gating coefficients' $\rho_\kappa > 0$ satisfy

$$\sum_\kappa \rho_\kappa(\boldsymbol{x}) = 1$$

which assures that all the error is distributed although ν can be adjusted to set the learning rate separately.

For robotic systems, we propose the following heuristic for determining the gating coefficients:

$$\rho_\mu = \frac{r_{\ddot{q}} \|\ddot{\boldsymbol{q}}\|}{r_{\ddot{q}} \|\ddot{\boldsymbol{q}}\| + r_{\dot{q}} \|\dot{\boldsymbol{q}}\|}, \qquad \rho_\eta = \frac{r_{\dot{q}} \|\dot{\boldsymbol{q}}\|}{r_{\ddot{q}} \|\ddot{\boldsymbol{q}}\| + r_{\dot{q}} \|\dot{\boldsymbol{q}}\|} \tag{10}$$

where $r_{\ddot{q}}$ and $r_{\dot{q}}$ are fixed weighting constant. The ratio of these parameters is set here as the ratio of the expected input limits of the joint rates and accelerations. An algorithm to determine the gating coefficients using reinforcement learning is under development (McGuire and D'Eleuterio 1992).

4. Self-Organized Hierarchical Architecture

Thus far, we have implied that the size and spacing of the coarse cells as well as the number of coarse-cell layers are fixed and moreover regular. However, there is ample reason to investigate the choice of these parameters and indeed the manner in which they may be changed. A trade-off exists, for example in the selection of the size of cells: Smaller cells may increase resolution at a cost of generalization; larger cells may overgeneralize and reduce resolution. A delicate balance must be struck. Miller *et al.* (1990) suggest that a broad range for these parameters exists that permits successful learning. In the spirit of artificial neural networks, it would make for an effective approach if, for example, the cell size and position could be automatically organized according to the input training data.

Several self-organizing neural networks have attempted to capture and exploit the spatial distribution of input data. The 'locally tuned network' by Moody and Darken (1989) and the 'self-organizing network' by Kohonen (1989) are two such approaches. Both are statistically based and organizes the neurons only; learning is a separate step.

One technique that uses differently sized cells is offer by Moody (1989). In this approach, levels of progressively finer CMACs are employed. The first CMAC uses coarse grids and is allowed to learn over the entire input space. Once this learning has achieved a precision within a prescribed tolerance level, the weights of this CMAC are fixed. A second CMAC with a finer grid is then added and learning continues, adjusting only the weights of the second CMAC. This procedure is repeated as required until the resulting hierarchy provides the desired resolution.

A disadvantage of the technique, however, is that subsequent grids must span the entire input space. As a consequence, the number of coarse cell units needed grows exponentially which in turn increases the number of granule cell units to prevent interference that may occur because of hashing. Also, the cell sizes of subsequent layers must be specified *a priori*.

Cell Organization Based on Kohonen's Network

Motivated by these efforts, we now present a concept for a self-organized hierarchical architecture compatible with the modular architecture described earlier and based on the CMAC network and Kohonen's self-organizing network (Graham and D'Eleuterio 1991*c*). Kohonen's network is well-suited to this application because of its simplicity and its nonoverlapping cell structure.

For explanatory purposes, let us consider a one-dimensional case. The Kohonen cells are precisely the fine cells which result from the overlapping coarse cells. Each cell, designated $\square_n$, has associated with it a real-valued weight v_n which is the position of the cell as measured from some reference point to the center of the cell. Without loss in generality, the weights can be ordered such that $v_n < v_{n+1}$. Our objective is to change gradually the position of the cells to reflect the probabilistic distribution of the input variable and thereby render the structure of the CMAC more efficient, providing greater accuracy in regions of the input space which is likely to display more activity.

Now consider a sample input value x. The cell that is 'activated' $\square_p$ is, in general, the one whose weight most closely matches to the input value. In this case, it is the cell which is closest in distance to x. This 'winner take all' activation can be represented by

$$|x - v_p| = \min_n |x - v_n| \tag{11}$$

where v_p is the weight of the activated cell. A neighborhood of cells $\mathcal{N}_p$, centered on $\square_p$, that is,

$$\mathcal{N}_p = \{\square_{p-r}, \ldots, \square_p, \ldots, \square_{p+r}\} \tag{12}$$

where the index r is a 'radius of activation,' is selected for weight adjustment. This adjustment is accomplished as follows:

$$\Delta v_n = \begin{cases} \lambda(x - v_n), & \square_n \in \mathcal{N}_p \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

where the learning rate λ is chosen between 0 and 1. Both the learning rate and the radius of activation are gradually decreased to zero which allows the Kohonen network to converge to a stable, ordered distribution of cells.

As desired, the effect of (13) is to redistribute the cells incrementally with each input datum. Each new input value essentially acts as a magnet drawing the cells in a given neighborhood slightly towards itself. Repeated over a set of input data, the resulting distribution will bear the statistical signature of the input space.

An example of the result of Kohonen self-organization is given in Figure 4. This is a cross-section of the fine (joint angle) discretization used for a two-link robotic arm. The training data was distributed normally.

Multiresolution

Following Moody (1989), a further enhancement that can be made to the present technique is to stack subsequent CMAC modules with progressively finer grids. With each additional CMAC, the weights of the previous one would after sufficient learning be fixed. But instead of having to span the entire input space with these subsequent CMACs, the self-organizing results afforded by Kohonen's network permit us to identify those regions of greatest activity in the input space and thereby restrict subsequent CMACs to selected areas. Of course, Kohonen self-organization can be implemented with each CMAC.

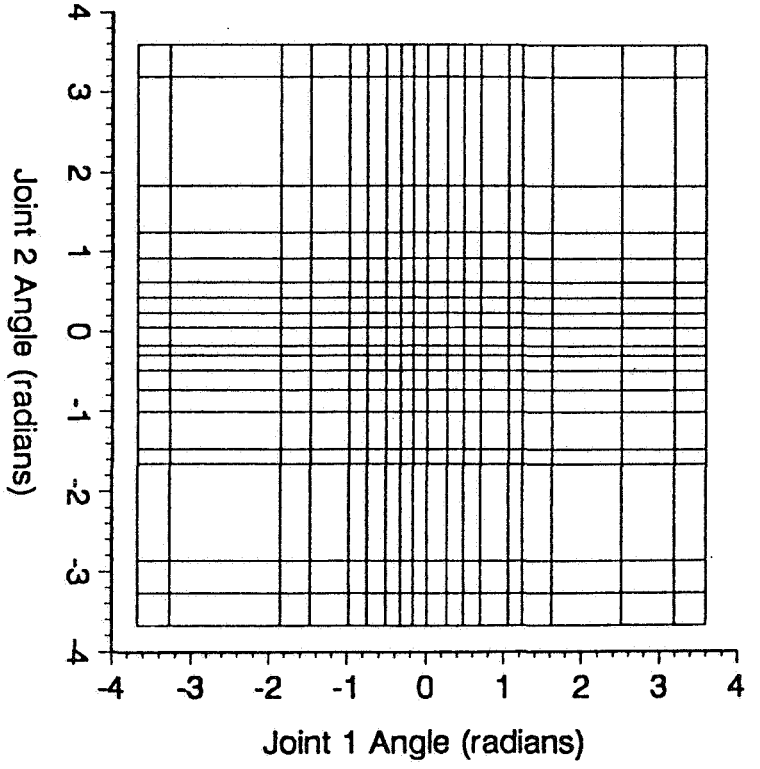


Figure 4: Cross-section of redistributed (fine) cells for a two-link manipulator after training on 1,000 normally distributed random samples (Mean: 0, Standard Deviation: $\frac{1}{6}$ of input range).

5. Simulation Results

We refer to the architecture resulting from these enhancements to the CMAC architecture as MOVE—Manipulator Operation using Value Encoding. We now present computer simulation results demonstrating the performance of MOVE and comparing it to the CMAC. The strawman system investigated here is the two-link manipulator used by Miller *et al.* (1987).

Modular Architecture. We begin by comparing MOVE, consisting only of the modular-architecture enhancement, with a single CMAC. Each variable is evenly discretized into 100 units and, to promote a reasonable amount of generalization, 30 layers (grids) of coarse cells are used. The single CMAC possesses 18,000 granule cells for each joint while each module in MOVE has 9,000 granule cells. Thus, the total memory requirements for each system are the same. A learning factor of $\gamma = 0.6$ was used.

For the first test, 100,000 uniformly distributed random sets of input data (θ, $\dot{\theta}$, $\ddot{\theta}$ and corresponding f_c). served as training data. For every set of 100 input samples, the networks were permitted to learn at only that sample which produced the worst error in the joint torques. Thus, actual learning was done on only 1,000 input samples. The average RMS error in the torques, however, was computed on each set of 100 samples. The results are plotted in Figure 5. As can be seen, not only does 'modular' MOVE learn significantly faster but it yields a lower final error.

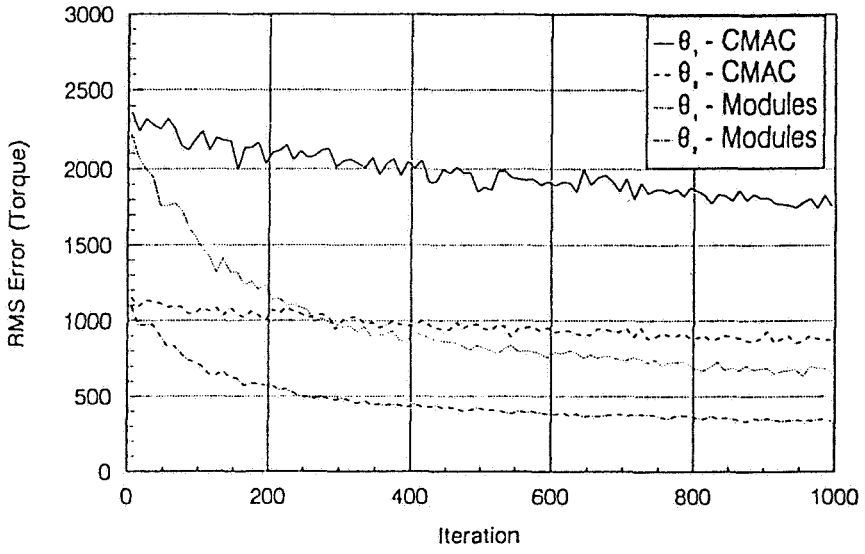


Figure 5: Comparison of learning trends of the CMAC and the modular version of MOVE.

The second test compares the CMAC and modular MOVE in a simulated control environment. For MOVE, the neural-network module in the control system of Figure 3 is replaced by the

modular architecture of MOVE represented diagrammatically in Figure 6. Representative control results, based on the random-sample learning explained above, are shown in Figure 7. The RMS error, as computed over the length of the trajectory, were 0.32 rad for the CMAC and 0.10 rad for modular MOVE.

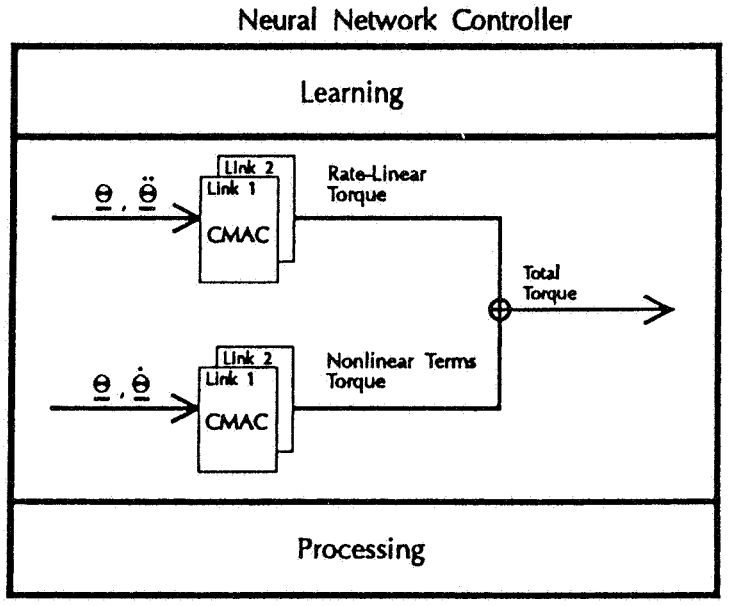


Figure 6: Neural-network module using modular MOVE

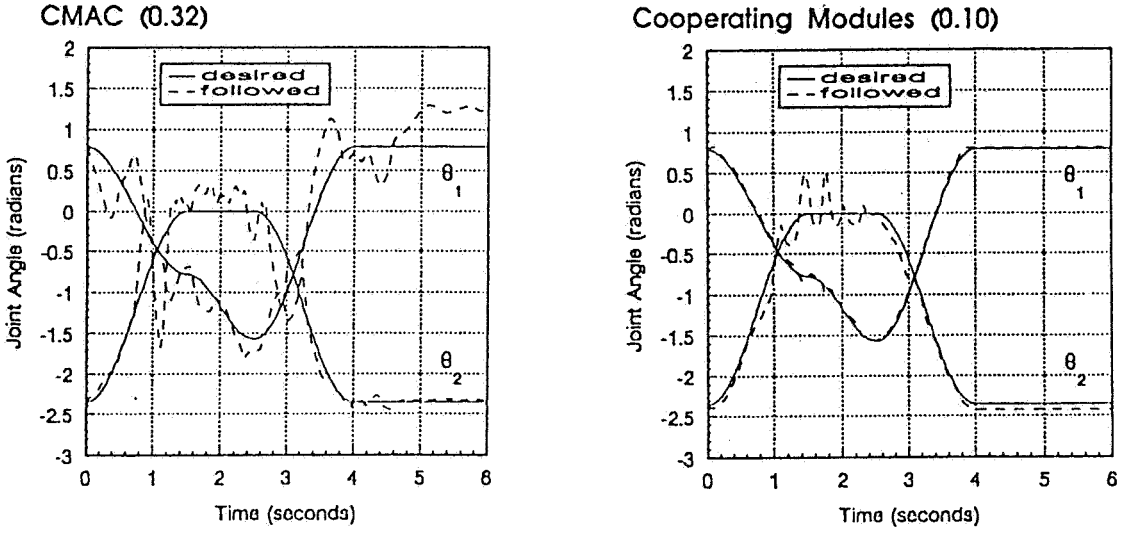


Figure 7: Comparison of control for CMAC and the modular version of MOVE

Self-Organized, Multiresolution Hierarchical Architecture. The hierarchical architecture of MOVE for self-organization and multiresolution was demonstrated and evaluated independently of the modularity enhancement. The neural-network module in the control system of Figure 3 in now replaced by the module in Figure 8. Two levels of CMACs are used in this example. The first level is trained on 1,000 normally distributed input samples (with zero mean and standard deviation of one-sixth of the input range). The weights of this level are then fixed and the second level is trained on a further set of 1,000 input samples.

Figure 9 shows the control results for a representative trajectory. The single CMAC, whose results are shown for comparison, was trained on all 2,000 input samples. The plots show the absolute errors in tracking for the two joints separately. The RMS error over the entire trajectory for both joints was 0.016 rad for the single CMAC and 0.011 rad for 'multiresolution' MOVE.

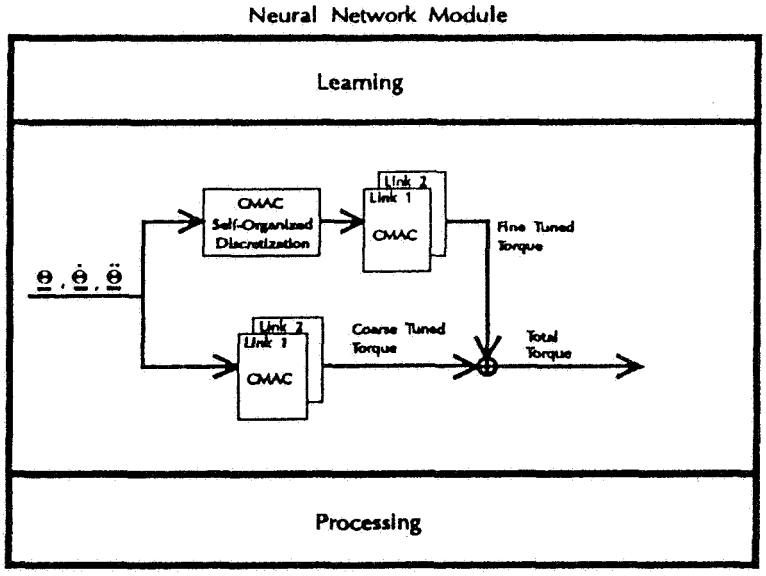


Figure 8: Neural-network module using self-organized, multiresolution hierarchical architecture of MOVE

6. Concluding Remarks

The basic concepts introduced by Albus in his CMAC provide a sound foundation for an artificial neural-network approach to the control of robotic systems. The enhancements incorporated in MOVE significantly improve on the performance of a CMAC robotic controller. The modular architecture of MOVE anticipates the form of the dynamical equations. By recognizing that all mechanical systems share this simple yet basic form, an appropriate structure can be

imposed on MOVE without compromising its applicability to robotics. This 'divide and conquer' technique results in faster learning and more efficient use of memory space.

The generalization property intrinsic to the CMAC has been enhanced by implementing a self-organization scheme based on the Kohonen network. This self-organization enables the cells in a CMAC to arrange themselves according to the statistical distribution of the training data. Furthermore, by creating a hierarchy of self-organized, multiresolution CMACs, one can also improve accuracy. This hierarchical architecture has been successfully employed in the modeling of chaotic systems as well.

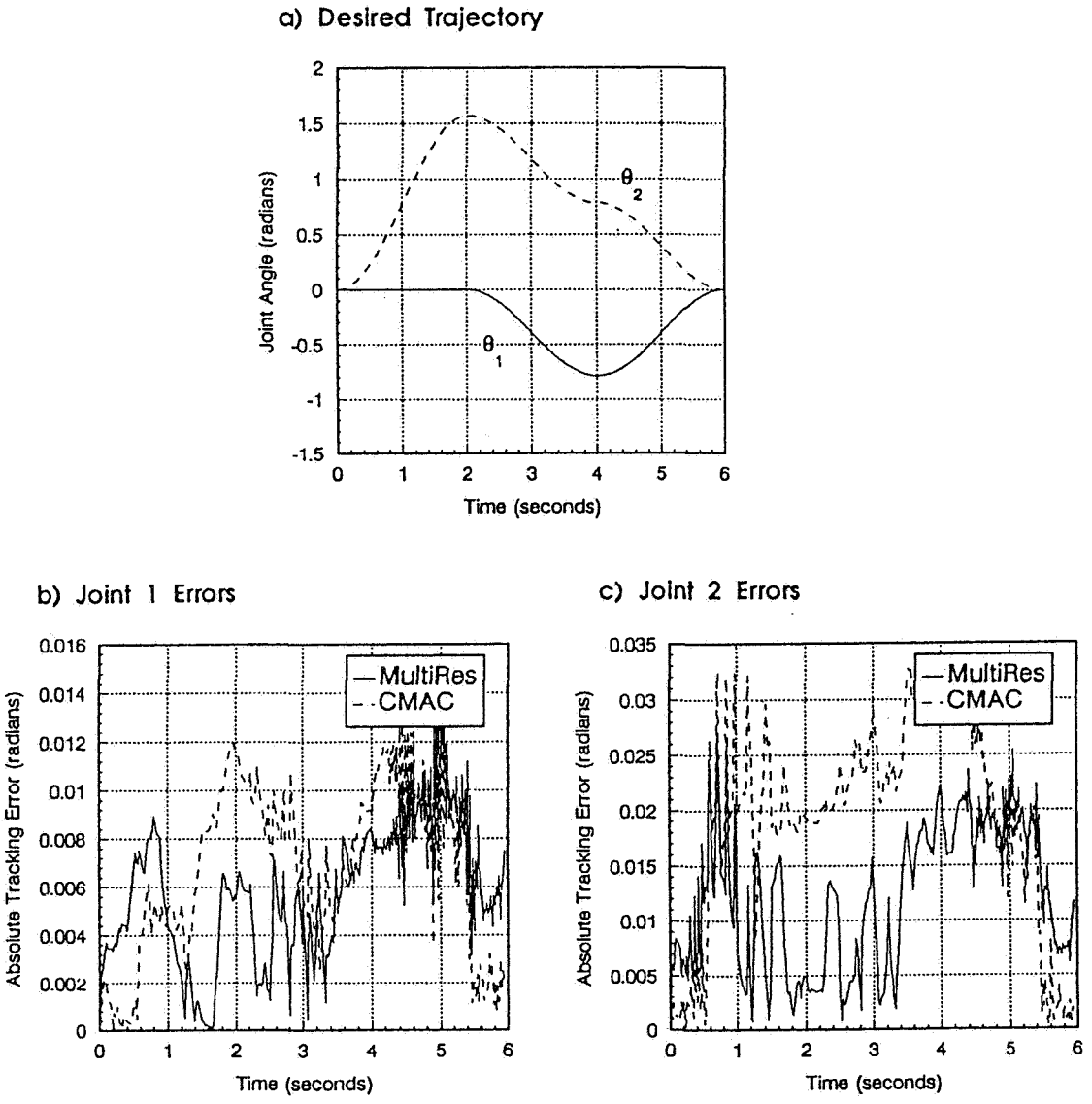


Figure 9: Comparison of a single CMAC and the self-organized, multiresolution hierarchical version of MOVE for a given trajectory.

It is evident that the concept of artificial neural networks holds considerable promise in the field of robotics. Neural networks allow us to dispense, at to least some extent, with carefully constructed system models. They moreover possess a characteristic highly desirable in the industrial workplace, that of being able to adapt to gradually deteriorating systems. For example, the dynamics of a new robotic manipulator will not be the same as the dynamics of that same manipulator when it is older. But neural networks continually learn, continually adapt.

The re-emergence of artificial neural networks also resonates with another current trend—that of parallel processing in computer technology. Like the brain, neural networks are inherently parallel which is of course a very desirable feature, particularly when considering real-time implementation. A hardware version of the the CMAC architecture is already commercially available and DiNardo and Graham (1992) have investigated the performance of MOVE on a parallel transputer platform.

The MOVE artificial neural-network architecture also exhibits considerable potential in other robotic application areas such vision, pattern recognition and analysis, sensor fusion, and flexible manipulators as well as a multitude of nonrobotic applications.

Acknowledgements

This research was supported by the Natural Science and Engineering Research Council of Canada, the Institute for Robotics and Intelligent Systems, and the Institute for Space and Terrestrial Science.

References

ALBUS J.S., "A New Approach to Manipulator Control: Cerebellar Model Articulation Controller (CMAC)," *ASME J. Dynamic Systems, Measurement, and Control*, September 1975, pp. 220–233.

ALBUS J.S., *Brains, Behavior, and Robotics*, BYTE Publications, 1981.

DINARDO G.D.M. & GRAHAM D.P.W., "The Efficiency of the MOVE Artificial Neural Network Architecture on a Multi-Transputer Platform," Seventh CASI Conference on Astronautics, Ottawa, ON, 4-6 November 1992.

FREEMAN W. & KOSKO B., "CHAIRs," 1990 International Joint Conference on Neural Networks, San Diego, CA, June 1990.

GRAHAM D.P.W. & D'ELEUTERIO G.M.T., "MOVE—A Neural-Network Paradigm for Robotic Control," *Canadian Aeronautics and Space Journal*, Vol. 37, No. 1, March 1991*a*, pp. 17–26.

GRAHAM D.P.W. & D'ELEUTERIO G.M.T., "Robotic Control Using a Modular Architecture of Co-operative Artificial Neural Networks," 1991 International Conference on Artificial Neural Networks, Helsinki, Finland, 22–26 June 1991*b*.

GRAHAM D.P.W. & D'ELEUTERIO G.M.T., "A Hierarchy of Self-Organized Multiresolution Artificial Neural Networks for Robotic Control," 1991 International Joint Conference on Neural Networks, Seattle, WA, 8-12 July 1991*c*.

JACOBS R.A., JORDAN M.I., NOWLAN S.J. & HINTON G.E., "Adaptive Mixtures of Local Experts," *Neural Computation*, Vol. 3, No. 1, 1991.

KOHONEN T., *Self-Organization and Associative Memory*, 3rd Edition, Springer-Verlag, 1989.

KUNG S-Y. AND HWANG J-N., "Neural Network Architectures for Robotic Applications," *IEEE Transactions on Robotics and Automation*, Vol. 5, No. 5, October 1989, pp. 641-657.

LEE S. & BEKEY G.A., "Applications of Neural Networks to Robotics," *Control and Dynamic Systems* (C.T. Leondes, ed), Volume 39: Advances in Robotic Systems, Academic Press, 1991.

MCGUIRE P.F. & D'ELEUTERIO G.M.T., "Active Control of Interference in CMAC/MOVE Neural Networks for Robotic Applications," Seventh CASI Conference on Astronautics, Ottawa, ON, 4-6 November 1992.

MICHIE D. & CHAMBERS R.A., "BOXES: An Experiment in Adaptive Control," *Machine Intelligence 2*, (Dale E. & Michie D., eds), Oliver and Boyd, Edinburgh, 1968, pp. 137-152.

MILLER III W.T., GLANZ F.H. & KRAFT III L.G., "Application of a General Learning Algorithm to the Control of Robotic Manipulators," *International J. Robotics Research*, Vol. 6, No. 2, 1987, pp. 84-98.

MILLER III W.T., HEWES R.P., GLANZ F.H. & KRAFT III L.G., "Real-Time Dynamic Control of an Industrial Manipulator Using a Neural-Network-Based Learning Controller," *IEEE Transactions on Robotics and Automation*, Vol. 6, No. 1, February 1990.

MOODY J., "Fast Learning in Multi-Resolution Hierarchies," Research Report YALEU/DCS/RR-691, February 1989.

MOODY J. & DARKEN C., "Fast Learning Networks of Locally-Tuned Processing Units," Research Report YALEU/DCS/RR-654, March 1989.

RUMMELHART D.E. & MCCELLAND J.L., *Parallel Distributing Processing*, MIT Press, 1988.

Computational Strategies in the Dynamic Simulation of Constrained Flexible MBS

F.M.L. Amirouche, Professor
M. Xie, Ph.D Research Assistant
Department of Mechanical Engineering
University of Illinois at Chicago

ABSTRACT

This research focuses on the computational dynamics of flexible constrained multibody systems. At first a recursive mapping formulation of the kinematical expressions in a minimum dimension as well as the matrix representation of the equations of motion are presented. The method employs Kane's equation, FEM and concepts of continuum mechanics. The generalized active forces are extended to include the effects of high temperature conditions, such as creep, thermal stress and elastic-plastic deformation. The time variant constraint relations for rolling/contact conditions between two flexible bodies are also studied. The constraints for validation of MBS simulation of gear meshing contact using a modified Timoshenko beam theory are also presented. The last part deals with minimization of vibration/deformation of the elastic beam in multibody systems making use of time variant boundary conditions. The above methodologies and computational procedures developed are being implemented in a program called DYAMUS.

KINEMATICS OF FLEXIBLE TREE-LIKE SYSTEMS

An explicit matrix representation of the partial velocities and partial angular velocities for tree-like structures is given below. Consider a flexible body in a MBS discretized into P elements. Let the position vector to an arbitrary element i of body j w.r.t. a fixed reference frame R be given by

$$\overline{p}_{ji} = \{\sum_{k=0}^{H(j)} \{q_l\}^T [S^{l-1,0}] + \sum_{k=0}^{H(j)} \{\zeta_l\}^T [S^{l-1,0}]$$

$$+ (\{r_{ji}\}^T + \{\rho_{ji}\}^T [N]^T)[S^{j0}]\}\{\overline{n}\} \qquad (1)$$

where S denotes the shift matrix, q, ζ and r represent the body vector, the translation vector between adjacent bodies, and the position vector from the local reference frame of body j to element i, respectively. N is the shape function matrix, ρ denote the nodal coordinates, and $\overline{n}$ a set of unit vector fixed in R (see reference [1]-[2] for more detail).

The velocity of element i of body j found by differentiation of the above equation can be expressed as

$$\overline{v}_{ji} = \{\dot{x}\}^T\{V^{ji}\} + \{\dot{q}\}^T\{V_s^j\} + \{\dot{\zeta}\}^T\{V_s^j\} + \{\dot{\rho}\}^T\{V_e^{ji}\} \qquad (2)$$

Four arrays are identified in the velocity expression and found to take a special form. Note that x represent the rigid body rotation between adjacent bodies. The partial derivative of the element velocity yield the following

$$[V^{ji}] = [W] \begin{bmatrix} ([S_{q_2}] + [S_{\zeta_2}])[S^{10}] \\ ([S_{q_3}] + [S_{\zeta_3}])[S^{20}] \\ \vdots \\ ([S_{q_j}] + [S_{\zeta_j}])[S^{j-1,0}] \\ ([S_{r_{ji}}] + [S_{ji}])[S^{j0}] \\ \vdots \\ 0 \end{bmatrix} \qquad (3)$$

where W is a transformation matrix used to isolate the generalized coordinate derivatives from the generalized speeds. S_q, S_ζ, S_r and S_{ji} are skew matrices corresponding to q, ζ, r and ρ, respectively. The partial velocity array associated with element deformation is given by

$$[V_e^{ji}] = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ [N]^T[S^{j0}] \\ \vdots \\ 0 \end{bmatrix} \qquad (4)$$

The partial velocity array associated with $\dot{q}$ is expressed as

$$[V_s^j] = \begin{bmatrix} [I] \\ [S^{10}] \\ \vdots \\ [S^{j-1,0}] \\ \vdots \\ 0 \end{bmatrix} \tag{5}$$

The bodies of the above block matrices could be achieved through a budgeting procedure where a master block is first developed then the rest of the arrays are formed through a partition and mapping technique see Table 1.

EQUATIONS OF MOTION

The governing equations of motion for flexible multibody systems can be expressed as

$$[M]\{\ddot{y}\} + [C]\{\dot{y}\} + [K]\{y\} = \{F\} \tag{6}$$

where $[M]$ denotes the generalized mass matrix composed of 9 submatrices of the form

$$[M_{11}] = \sum_j \sum_i \int_{v_{ji}} (m_{ji}[V^{ji}][V^{ji}]^T + [\omega^j][I_{ji}][\omega^j]^T) dv \tag{7}$$

The generalized mass is symmetric and the components of M_{ij} come directly from the kinematic bank of partial velocities and angular velocities of elements. The other mass components have similar expressions. Similarly, we can write the dynamic damping matrix, generalized stiffness matrix and force vector in a partition form with its components expressed as

$$[C_{11}] = \sum_j \sum_i \int_{v_{ji}} \{m_{ji}[V^{ji}][\dot{V}^{ji}]^T + [\omega^j]([I_{ji}][\dot{\omega}^j]^T + [\Omega_x^{0j}][I_{ji}][\omega^j]^T)\} dv \tag{8}$$

and

$$\{F_1\} = \sum_j \sum_i \int_{s_{ji}} ([V^{ji}]\{f_{ji}\} + [\omega^j]\{\mathcal{M}_{ji}\}) ds + \sum_j \sum_i \int_{v_{ji}} [V^{ji}]\{b_{ji}\} dv \tag{9}$$

It is important to note at this stage how the kinematical expression form the bulk of all computations. In the above equations m_{ji} denotes the mass of element i in body j, I_{ji} the tensor dyadic, f_{ji} the force vector array acting on element i of body j, $\mathcal{M}_{ji}$ the corresponding moment array and b_{ji} the surface traction contribution vector.

CONSIDERATION OF HIGH TEMPERATURE, CREEP AND ELASTIC-PLASTIC DEFORMATIONS

The modeling of time-dependent forces resulting from deformable bodies when subjected high temperature conditions can be of interest in many engineering applications, which include creep, thermal stress, thermal shock, etc.. Many researchers studied material nonlinearities, in which some problems are solved, other still remain to be issues of concern.

The effects of temperature, creep and thermal stress and thermal shock can be included in the third term of the generalized force (see reference [2])

$$\{F_3\} = \sum_j \sum_i \int_{s_{ji}} ([V_e^{ji}]\{f_{ji}\} + [\omega^{ji}]\{\mathcal{M}_{ji}\}) ds + \sum_j \sum_i \int_{v_{ji}} [V_e^{ji}]\{b_{ji}\} dv + \{F_T\} \tag{10}$$

where the last part $\{F_T\}$ brings in the contribution from the effects of temperatures and material nonlinearities

$$\{F_T\} = \sum_j \sum_i \int_{v_{ji}} \{[B]^T[D][\{\varepsilon_c\} + \alpha(\{T\} + \sigma^*\{T_0\})] + \alpha[N]^T[D]\{T'\}\} dv \tag{11}$$

The nonlinearities including geometric nonlinearity and material nonlinearity can be considered in the stiffness matrix. So does the elastic-plastic deformation. The material property matrix is given by

$$[D] = [D_e] + [D_p]$$

where $[D_e]$ denotes the elastic part. The second part $[D_p]$ is the contribution from the plastic deformation

$$[D_p] = [D_e] -$$

$$[D_e]\{\frac{\partial F}{\partial \sigma}\}\{\frac{\partial F}{\partial \sigma}\}^T[D_e](A + \{\frac{\partial F}{\partial \sigma}\}^T[D_e]\{\frac{\partial F}{\partial \sigma}\})^{-1} \quad (12)$$

TIME VARIANT BOUNDARY CONSTRAINT CONDITIONS

For the time variant boundary conditions, finite difference method can be used to account for the rate of change of mode shape. Consider the modal transformation

$$\{\eta\} = [\Phi]\{\rho\} \quad (13)$$

Differentiation of the above equation yields

$$\{\dot{\eta}\} = [\dot{\Phi}]\{\rho\} + [\Phi]\{\dot{\rho}\} \quad (14)$$

When substituting the nodal displacement with the nodal coordinates and taking into consideration the effects of $[\dot{\Phi}]$, then at $t = t_l$ the new terms coming from the previous and newly computed mode shapes at $t = t_l$ are seen in $[C]$ and $[K]$ as[2],[3]

$$[C_{l33}] = [C_{33}] + \frac{2}{\Delta t}[M_{33}]([\Phi_l] - [\Phi_{l-1}]) \quad (15)$$

and

$$[K_{l33}] = [K_{33}] + \frac{1}{\Delta t^2}[M_{33}]([\Phi_l]$$

$$- 2[\Phi_{l-1}] + [\Phi_{l-2}]) + \frac{1}{\Delta t}[C_{33}]([\Phi_l] - [\Phi_{l-1}]) \quad (16)$$

The method developed above has a wide range of applications for which one can easily see and analyze its dynamics.

While the time variant contact conditions can be considered as a set of constraints which can be holonomic or nonholonomic. Some constraint equations which do not contain prescribed motion terms can be factorized to minimize the dimension of the equations of the system. For the case of two flexible bodies with one rolling without slipping on the other as shown in Figure 1, we can write in R the following position vector[2]

$$\bar{p}_{ni} = \bar{p}_{n-1} + \bar{q}_n + \bar{r}_{ni} \quad (17)$$

where

$$\bar{q}_n = \bar{r}_{n-1,c} + \bar{u}_{n-1,c} - (\bar{r}_{nc} + \bar{u}_{nc}) \quad (18)$$

Differentiation of equation (17) yields the constraint equations at the velocity level

$$[J]\{\dot{y}\} = \{g\} \quad (19)$$

where

$$\{y\} = [\ x^T \ \ \zeta^T \ \ q_n^T \ \ \rho^T\ ]^T \quad (20)$$

and

$$\{g\} = [S^{n-1,n}][\Omega^{n-1,n}](\{r_n\} + [N_n]\{\rho_{nc}\}) \quad (21)$$

$[J]$ is a Jacobi matrix and a function of generalized coordinates and velocities.

In the dynamics of MBS for the case when one flexible body is rolling on another, equations (19) and (6) extended with λJ^T are solved together. The time history of the system allows us to systematically update the contact position and the reevaluation of the Jacobi matrix J.

DYNAMICS OF GEAR MESHING TEETH

For validation of the results obtained by multibody dynamics code which utilize FEM, a modified Timoshenko beam theory is presented to analyze the dynamics of gear meshing teeth in rotorcraft systems. The acting position, direction and magnitude of the

external forces are assumed to time variant. The meshing tooth is considered as a cantilever beam, as shown in Figure 2, where the inertia force due to the large rotation of the tooth base, as well as the external equivalent axial force and moment are all included in the equation of motion.[2]

$$\frac{\partial^2}{\partial x^2}\{EI(x)\{\frac{\partial^2 w}{\partial x^2} - \frac{1}{kA(x)G}[\varrho A(x)\frac{\partial^2 w}{\partial t^2} + f(x,t) + P(x,t)\frac{\partial^2 w}{\partial x^2}]\} + m(x,t)\} - \varrho I(x)\frac{\partial^4 w}{\partial x^2 \partial t^2} + \frac{\varrho I(x)}{kA(x)G}\frac{\partial^2}{\partial t^2}[\varrho A(x)\frac{\partial^2 w}{\partial t^2} + f(x,t) + P(x,t)\frac{\partial^2 w}{\partial x^2}] + \varrho A(x)\frac{\partial^2 w}{\partial t^2} + f(x,t) + P(x,t)\frac{\partial^2 w}{\partial x^2} = 0 \quad (22)$$

For the assumed model, the boundary conditions are given by:
At the fixed end $x = 0$,

$$\psi(0,t) = w(0,t) = 0 \quad (23)$$

At the free end $x = l$,

$$V(l,t) = kA(l)G[\frac{\partial w(l,t)}{\partial x} - \psi(l,t)] = 0 \quad (24)$$

$$M(l,t) = [EI(l)\frac{\partial \psi(l,t)}{\partial x} + m(l,t)] = 0 \quad (25)$$

A solution to the above proposed model will result in prediction of contact forces or dynamic loading on gear teeth.

MINIMIZATION OF VIBRATION IN ELASTIC BEAMS

The minimization of vibration (deformation) of flexible bodies in mechanical systems is a major concern in dynamics and control. What follows are procedures used to minimize vibration in elastic beams. The elastic beam is modeled in two ways: one has a movable support not to exceed the lower tip, whereas the other treats the body as a hollow beam with a moving mass.

Equation of motion for the model used to minimize vibration of the flexible beam, as shown in Figure 3, is given by[2]

$$EI\frac{\partial^4 y}{\partial x^4} + m(\frac{\partial^2 y}{\partial t^2} - y\dot{\theta}^2 + g\cos\theta + a_O^t + x\ddot{\theta}) = 0 \quad (26)$$

Laplace transform gives

$$Y = \sum_{i=1}^{4} c_i e^{r_i x} + \frac{1}{ms^2}\{L(x,s) + m[sf_1(x) + f_2(x)]\} \quad (27)$$

The functional used to minimize vibration of the beam is

$$J(x_0) = \int_0^t F(x, x_0, t)dt \quad (28)$$

Euler-Lagrange equation

$$\frac{\partial F}{\partial x_0} - \frac{d}{dt}(\frac{\partial F}{\partial \dot{x}_0}) = 0 \quad (29)$$

is used to solve for the problem at hand.

The solution for optimum positioning conditions is time variant and yields minimum deflection at the proposed location of the beam.

*

References

[1] Amirouche, F.M.L.: *Computational Method in Multibody Dynamics*, Prentice Hall, 1992.

[2] Xie, M.: *Flexible Rotorcraft System Dynamics with Time Variant Contact Conditions*, Ph.D Dissertation, University of Illinois at Chicago, Chicago, IL, March 1992.

[3] Amirouche, F.M.L. and Xie, M.: Dynamic analysis of flexible multibody systems with time variant mode shapes, *The 13th Biennial ASME Conference, Mechanical Vibration and Noise*, Miami, Florida, Sept. 1991. ASME DE *Structural Vibration and Acoustics*, 34:261-267.

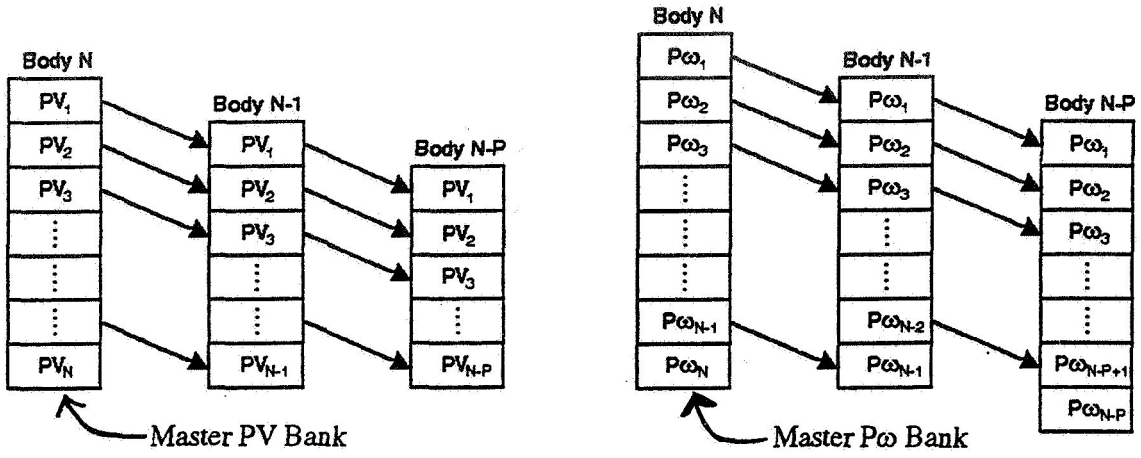


Table 1: Mapping technique used in MBS

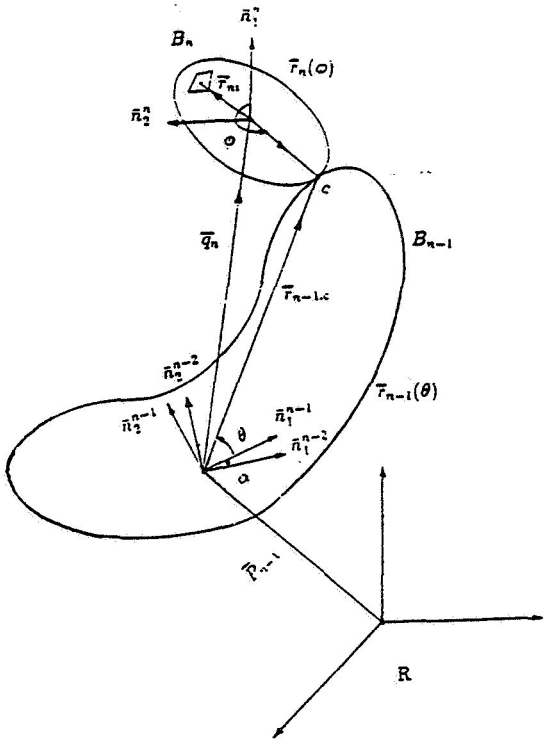


Figure 1: Model for time variant contact conditions

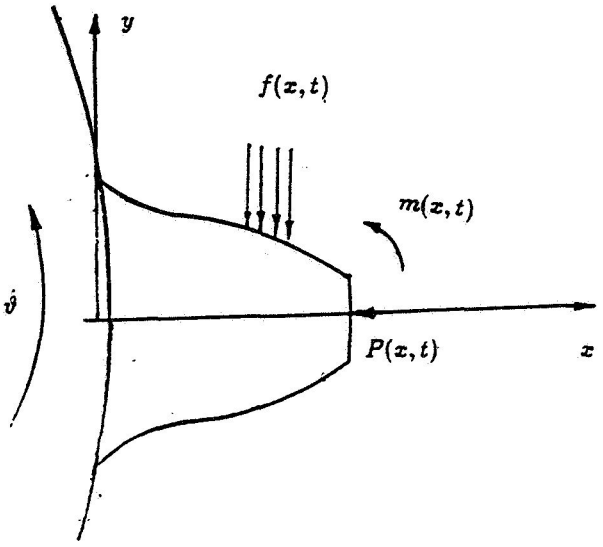


Figure 2: Model for gear meshing teeth

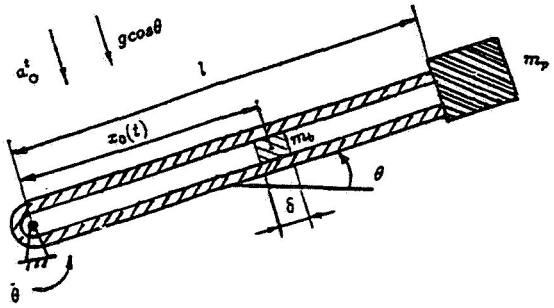


Figure 3: Models for vibration minimization

Parallel O(Log n) Algorithms for Open- and Closed-Chain Rigid Multibody Systems Based on a New Mass Matrix Factorization Technique

Amir Fijany
Jet Propulsion Laboratory, California Institute of Technology
Pasadena, CA 91109

Abstract

In this paper, parallel $O(\log n)$ algorithms for computation of rigid multibody dynamics are developed. These parallel algorithms are derived by parallelization of new $O(n)$ algorithms for the problem. The underlying feature of these $O(n)$ algorithms is a drastically different strategy for decomposition of interbody force which leads to a new factorization of the mass matrix ($\mathcal{M}$). Specifically, it is shown that a factorization of the inverse of the mass matrix in the form of the *Schur Complement* is derived as $\mathcal{M}^{-1} = \mathcal{C} - \mathcal{B}^{*}\mathcal{A}^{-1}\mathcal{B}$, wherein matrices $\mathcal{C}$, $\mathcal{A}$, and $\mathcal{B}$ are block tridiagonal matrices. The new $O(n)$ algorithm is then derived as a recursive implementation of this factorization of $\mathcal{M}^{-1}$. For the closed-chain systems, similar factorizations and $O(n)$ algorithms for computation of Operational Space Mass Matrix Λ and its inverse Λ^{-1} are also derived. It is shown that these $O(n)$ algorithms are *strictly parallel*, that is, they are less efficient than other algorithms for serial computation of the problem. But, to our knowledge, they are the only known algorithms that can be parallelized and that lead to both time- and processor-optimal parallel algorithms for the problem, i.e., parallel $O(\log n)$ algorithms with $O(n)$ processors. The developed parallel algorithms, in addition to their theoretical significance, are also practical from an implementation point of view due to their simple architectural requirements.

n	Total number of Degrees-Of-Freedom (DOF) of the system
$P_{i,j}$	Position vector from point O_j to point O_i, $P_{i+1,i} = P_i$
m_i	Mass of body i
h_i, k_i	First and Second Moment of mass of body i about point O_i
$I_{i,j}$	Spatial Inertia of body i about point O_j, $I_{i,i} = I_i = \begin{bmatrix} k_i & \tilde{h}_i \\ \tilde{h}_i^* & m_i U \end{bmatrix} \in \mathbb{R}^{6x6}$ (* denotes the transpose)
$\mathcal{M} \in \mathbb{R}^{nxn}$	Symmetric Positive Definite (SPD) mass matrix
$\theta \triangleq \text{col}\{\theta_i\} \in \mathbb{R}^{nx1}$	Vector of joint positions
$Q \triangleq \text{col}\{Q_i\} \in \mathbb{R}^{nx1}$	Vector of joint velocities
$\dot{Q} \triangleq \text{col}\{\dot{Q}_i\} \in \mathbb{R}^{nx1}$	Vector of joint accelerations
$\mathcal{T} \triangleq \text{col}\{\tau_i\} \in \mathbb{R}^{nx1}$	Vector of applied (control) joint forces/torques
ω_i, $\dot{\omega}_i$	Angular and linear acceleration of body i (frame i+1)
v_i, $\dot{v}_i$	Linear velocity and acceleration of body i (point O_i)
f_i, n_i	Force and moment of interaction between body i-1 and body i
$\dot{V}_i = \begin{bmatrix} \dot{\omega}_i \\ \dot{v}_i \end{bmatrix} \in \mathbb{R}^{6x1}$	Spatial acceleration of body i
$F_i = \begin{bmatrix} n_i \\ f_i \end{bmatrix} \in \mathbb{R}^{6x1}$	Spatial force of interaction between body i-1 and body i
$H_i \in \mathbb{R}^{6xn_i}$	Spatial axis (map matrix) of joint i

Table I. Notation

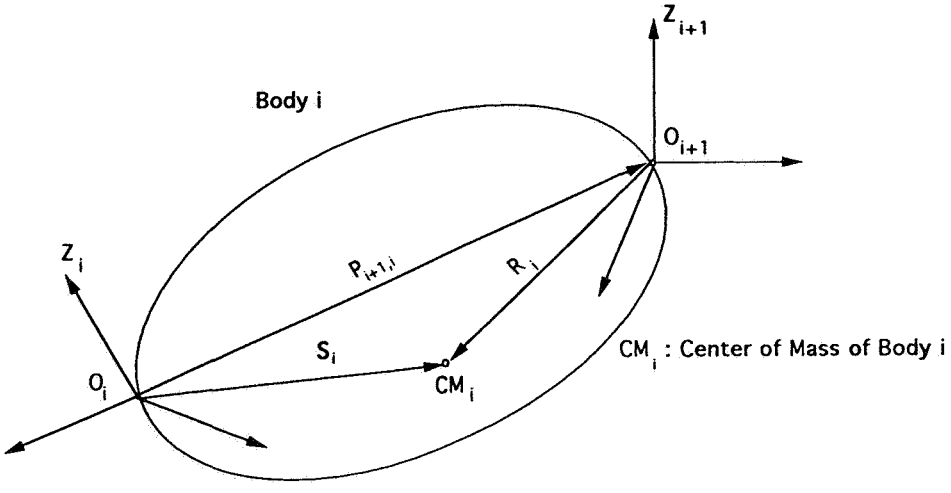


Figure 1. Body, Frames, and Position Vectors

I. Introduction

The multibody dynamics problem concerns the determination of the motion of the mechanical system, resulting from the application of a set of control forces. In the context of robotics, the dynamic simulation problem is better known as the forward dynamics problem.

From a computational point of view, the multibody dynamics problem can be stated as the solution of a linear system as

$$\mathcal{M}\dot{Q} = \mathcal{T} - b(\theta, Q) = \mathcal{F}_T, \text{ or} \tag{1}$$

$$\dot{Q} = \mathcal{M}^{-1}\mathcal{F}_T \tag{2}$$

where the vector $b(\theta, Q)$ represents the contribution of nonlinear terms and can be computed by using the recursive Newton-Euler (N-E) algorithm [3] by setting the joint accelerations to zero. Hence, in Eqs. (1)-(2), $\mathcal{F}_T \triangleq \text{col}\{F_{Ti}\} \varepsilon \mathbb{R}^{nx1}$ represents the acceleration-dependent component of the control force.

The developed serial algorithms for the problem can be classified as the $O(n^3)$ algorithm [4], the $O(n^2)$ algorithm [5], and the $O(n)$ algorithms [6-13]. See also [14] for more complete references as well as an extensive analysis and comparison of these algorithms. In addition to these algorithms, which are based on rather direct methods, there is also another class of indirect (or, iterative) algorithms for solution of Eq. (1) which include the $O(n^2)$ conjugate gradient algorithms [4,15,16].

It seems that the development of serial algorithms for the problem has reached a certain level of maturity. Asymptotically, the $O(n)$ algorithms represent the fastest possible serial method for the problem, since, given the n-component input (vector of control force), the evaluation of the n-component output (joint accelerations) requires at least $O(n)$ distinct steps in the computation. Hence, any further improvement in computational efficiency of the $O(n)$ algorithms can only be achieved by reducing the coefficients (see for example [10,14] wherein this reduction has been achieved by avoiding explicit computation of the term $b(\theta, Q)$).

The relationship among the different direct algorithms is also well understood, and two fundamental results have been established [14]. The first is that, at a conceptual level, the $O(n)$ algorithms can be essentially considered as a procedure for recursive factorization and inversion of mass matrix, i.e., recursive computation of $\mathcal{M}^{-1}\mathcal{F}_T$ [9,10,11,14]. The second result is that, at a computational level, the $O(n)$ algorithms lead to the computation of the articulated-body inertia [7]. The reader is referred to [14] for an

extensive analysis of commonalities in the computation of the $O(n)$ algorithms. It should be emphasized that our analysis of the parallel computation efficiency of different algorithms relies on these two results.

Despite the significant improvement in the efficiency of serial algorithms, even the fastest algorithm is still far from providing real-time or faster-than-real-time simulation capability. With the maturity of serial algorithms, any further significant improvement in computational efficiency can be achieved only through exploitation of parallelism. This is further motivated by advances in VLSI technology that have made parallel computation a practical and low-cost alternative for achieving significant computational efficiency. However, unlike serial computation, there are few reports on the development of parallel algorithms for the problem.

The development of efficient parallel algorithms for multibody dynamics is a rather challenging problem. It represents an interesting example for which the analysis of the efficiency of a given algorithm for parallel computation is far different and more complex than that for serial computation. In fact, our previous analysis [1,2,17] and the results of this paper clearly show that those algorithms that are less efficient (in terms of either asymptotic complexity or number of operations) for serial computation provide a higher degree of parallelism and hence are more efficient for parallel computation.

A preliminary investigation of parallelism in the computation of forward dynamics, analyzing the efficiency of existing algorithms for parallel computation, is reported in [2]. The main result of this investigation was that the $O(n^3)$ algorithms provide the highest degree of parallelism and are the most efficient for parallel computation. Specifically, it was shown that

1. Theoretically, the time lower bound of $O(\log^2 n)$ can be achieved by parallelization of the $O(n^3)$ algorithms by using $O(n^3)$ processors.

2. Practically, the best parallel algorithm for the problem is of $O(n)$ which results from parallelization of the $O(n^3)$ algorithms on a two-dimensional array of $O(n^2)$ processors. This parallel algorithm, although of $O(n)$, achieves a significant speedup over the best serial $O(n)$ algorithms by reducing the coefficient of the n-dependent term on polynomial complexity by more than two orders-of-magnitude. Different approaches for parallelization of the $O(n^3)$ algorithms have also been proposed in [18,19].

The analysis in [1] also led to two additional important conclusions. The first was that, if indeed there can be a parallel algorithm achieving the time

lower bound of $O(\log n)$ with an optimal number of $O(n)$ processors, then this parallel algorithm can only be derived by parallelization of an $O(n)$ serial algorithm. However, the analysis in [1] showed that the parallelism in the existing $O(n)$ algorithms was bounded, that is, at best only a constant speedup in the computation can be achieved, leading to the parallel $O(n)$ algorithms. More specifically, it was shown that the recurrence for computation of the articulated-body inertia is strictly serial and cannot be parallelized (see Sec. II.D). Hence, the second conclusion in [1] was that if the forward dynamics problem is to have the time lower bound of $O(\log n)$ for its computation, it can only result from a totally different serial $O(n)$ algorithm. Such an algorithm can only be derived by a global reformulation of the problem and not an algebraic transformation in the computation of existing $O(n)$ algorithms.

Physically, a given algorithm for multibody dynamics can be classified based on its force decomposition strategy. Mathematically, the algorithm can be classified based on the resulting factorization of the mass matrix which corresponds to the specific force decomposition (see Sec. II.B and C). A new algorithm based on a global reformulation of the problem is, then, the one that starts with a different and new force decomposition strategy and results in a new factorization of mass matrix.

Interestingly, a recently developed $O(n)$ algorithm in [21-24] for a single serial chain represents such a global reformulation of the problem. It differs from the existing $O(n)$ algorithms in the sense that it is based on a different strategy for force decomposition (see Sec. III). We will show that this strategy leads to a new and completely different factorization of M^{-1}. This factorization, in turn, results in a new $O(n)$ algorithm for the problem which is strictly efficient for parallel computation, that is, it is less efficient than other $O(n)$ algorithms for serial computation but, as will be shown, it can be parallelized to achieve the time lower bound of $O(\log n)$ with $O(n)$ processors. We show that this factorization of M^{-1} also directly leads to new factorizations and $O(n)$ algorithms for closed-chain systems. Again, these new $O(n)$ algorithms for closed-chain systems can be parallelized to derive both time- and processor-optimal parallel algorithms for the problem, i.e., $O(\log n)$ parallel algorithms with $O(n)$ processors. Furthermore, the new factorizations for both open- and closed-chain systems can be uniformly described in terms of the *Schur Complement* and provide different and deeper physical insights into the problem.

This paper is organized as follows. In Sec. II, the $O(n)$ algorithms, i.e, the Articulated-Body Inertia algorithm and recursive factorization and inversion of mass matrix, are briefly reviewed. In Sec. III, the Constraint Force algorithm and the new factorization of mass matrix are derived. In Sec. IV, new factorizations and $O(n)$ algorithms for closed-chain systems are presented. In Sec. V, parallel $O(\log n)$ algorithms for both open- and closed-chain systems are briefly presented. Finally, some concluding remarks are made in Sec. VI.

II. The $O(n)$ Algorithms: Recursive Factorization and Inversion of Mass Matrix

A. Notation and Preliminaries

In our discussion of the $O(n)$ algorithms, a set of spatial notations is used which, though slightly different from those in [8-11, 21-24], allows a clear understanding and comparison of the algorithms (see also Table I and Fig. 1). For the sake of clarity, the spatial quantities are shown with upper-case italic letters. Here, only joints with one revolute DOF are considered. However, all the results can be extended to the systems with joints having different and more DOF's.

With any vector V, a tensor $\tilde{V}$ can be associated whose representation in any frame is a skew symmetric matrix as

$$\tilde{V} = \begin{bmatrix} 0 & -V_{(z)} & V_{(y)} \\ V_{(z)} & 0 & -V_{(x)} \\ -V_{(y)} & V_{(x)} & 0 \end{bmatrix}$$

where $V_{(x)}$, $V_{(y)}$, and $V_{(z)}$ are the components of V in the considered frame. The tensor $\tilde{V}$ has the properties that $\tilde{V}^* = -\tilde{V}$ and $\tilde{V}_1 V_2 = V_1 \times V_2$. A matrix $\hat{V}$ associated to the vector V is defined as

$$\hat{V} = \begin{bmatrix} U & \tilde{V} \\ 0 & U \end{bmatrix} \text{ and } \hat{V}^* = \begin{bmatrix} U & 0 \\ -\tilde{V} & U \end{bmatrix} \in \mathbb{R}^{6\times 6}$$

where here (as well as through the rest of the paper) U and 0 stand for unit and zero matrices of appropriate size. The spatial forces acting on two rigidly connected points A and B are related as

$$F_B = \hat{P}_{A,B} F_A$$

where $P_{A,B}$ denotes the position vector from B to A. If the linear and angular velocities of point A are zero then

$$\dot{V}_A = (\hat{P}_{A,B})^* \dot{V}_B$$

The matrix $\hat{P}_{A,B}$ has the properties $\hat{P}_{A,B}\hat{P}_{B,C} = \hat{P}_{A,C}$ and $(\hat{P}_{A,B})^{-1} = \hat{P}_{B,A}$.

In derivation of equations of motion, it is assumed that the nonlinear term $b(\theta,Q)$ is explicitly computed by using the recursive N-E algorithm. For both the articulated-body algorithm (as shown in [14]) and the new algorithm, this explicit computation can be avoided. However, this does not affect the efficiency of the algorithms for parallel computation. In fact, as for the $O(n^3)$ and $O(n^2)$ algorithms [16,17], the explicit computation of $b(\theta,Q)$ provides additional parallelism which can be exploited to further increase the speedup in the computation.

By computing the term $b(\theta,Q)$ and subtracting it from $\mathcal{T}$ (Eq. 1), i.e., by explicitly computing $\mathcal{F}_T$, the multibody system can be assumed to be a system at rest which upon the application of the control force $\mathcal{F}_T$ accelerates in space. The equation of motion for body i, as a single rigid body, is given as

$$F_i = I_i \dot{V}_i$$

and as an interconnected member of the serial chain is given as (Fig. 1)

$$\dot{V}_i = \hat{P}^*_{i-1}\dot{V}_{i-1} + H_i\ddot{Q}_i \tag{3}$$

$$F_i = I_i\dot{V}_i + \hat{P}_i F_{i+1} \tag{4}$$

Eqs. (3)-(4) represent the simplified N-E algorithm (with nonlinear terms excluded) for the serial chain.

Equation (4) represents the interbody force-decomposition strategy of the N-E formulation. As shown in [9-11], this force-decomposition strategy leads to a specific factorization of $\mathcal{M}$. To see this, let us rewrite Eqs. (3)-(4) as

$$\dot{V}_i - \hat{P}^*_{i-1}\dot{V}_{i-1} = H_i\ddot{Q}_i \tag{5}$$

$$F_i - \hat{P}_i F_{i+1} = I_i\dot{V}_i \tag{6}$$

and define

$$\mathcal{H} \triangleq \text{diag}\{H_i\} \in \mathbb{R}^{6n\times n}$$

$$\mathcal{I} \triangleq \text{diag}\{I_i\} \in \mathbb{R}^{6n\times 6n}$$

$$\dot{\mathcal{V}} \triangleq \text{col}\{\dot{V}_i\} \in \mathbb{R}^{6n\times 1}$$

$$\mathcal{F} \triangleq \text{col}\{F_i\} \in \mathbb{R}^{6n\times 1}$$

$$\mathcal{P} = \begin{bmatrix} U & & & & \\ -\hat{P}_{n,n-1} & U & & & \\ 0 & -\hat{P}_{n-1,n-2} & U & & \\ 0 & 0 & & & \\ 0 & 0 & & -\hat{P}_{2,1} & U \end{bmatrix} \in \mathbb{R}^{6nx6n}$$

$$\mathcal{P}^{-1} = \begin{bmatrix} U & & & & \\ \hat{P}_{n,n-1} & U & & & \\ \hat{P}_{n,n-2} & \hat{P}_{n-1,n-2} & U & & \\ & & & & \\ \hat{P}_{n,1} & \hat{P}_{n-1,1} & & \hat{P}_{n,1} & U \end{bmatrix} \in \mathbb{R}^{6nx6n}$$

Eqs. (5)-(6) can now be rewritten in a global form as

$$\mathcal{P}^{*}\dot{V} = \mathcal{H}\dot{Q} \tag{7}$$

$$\mathcal{P}\mathcal{F} = \mathcal{J}\dot{V} \tag{8}$$

A factorization of mass matrix, associated with the force decomposition in Eq. (4), can now be derived as

$$\mathcal{F}_T = \mathcal{H}^{*}\mathcal{F} = \mathcal{H}^{*}\mathcal{P}^{-1}\mathcal{J}\dot{V} = \mathcal{H}^{*}\mathcal{P}^{-1}\mathcal{J}(\mathcal{P}^{*})^{-1}\mathcal{H}\dot{Q} \tag{9}$$

which, in comparison with Eq. (1), represents a factorization of $\mathcal{M}$ as

$$\mathcal{M} = \mathcal{H}^{*}\mathcal{P}^{-1}\mathcal{J}(\mathcal{P}^{*})^{-1}\mathcal{H} \tag{10}$$

Although the matrices $\mathcal{P}^{-1}$, $\mathcal{J}$, and $(\mathcal{P}^{*})^{-1}$ are square and have trivial inverses, the matrices $\mathcal{H}^{*}$ and $\mathcal{H}$ are not square. This prevents the computation of $\mathcal{M}^{-1}$ from the above factorization.

B. The Articulated-Body Inertia (A-BI) Algorithm

The Articulated-Body Inertia (A-BI) algorithm is based on a decomposition of F_i as [8]

$$F_i = I_i^A \dot{V}_i + T_i^A \tag{11}$$

where I_i^A is the articulated-body inertia of body i. The force T_i^A is a function of I_j^A and F_{Tj} for j = n to i+1. If I_j^A (and hence T_j^A), for j = n to i, is computed, then the projection of Eq. (6) along the joint axis i leads to a new equation with $\dot{V}_i$ as the only unknown

$$F_{Ti} = H_i^{*} F_i = H_i^{*} I_i^A \dot{V}_i + H_i^{*} T_i^A \tag{12}$$

Starting from i = 1, the joint accelerations can then be recursively computed from Eq. (12). This clearly explains the motivation behind the specific force decomposition in Eq. (11), which, unlike the one in Eq. (4), leads to the solution for joint accelerations.

The computational steps of the A-BI algorithm are given as [8]

For i = n to 1

$$I_i^A = I_i + \hat{P}_i\left(I_{i+1}^A - I_{i+1}^A H_{i+1}(H_{i+1}^* I_{i+1}^A H_{i+1})^{-1} H_{i+1}^* I_{i+1}^A\right)\hat{P}_i^* \qquad I_n^A = I_n \qquad (13)$$

$$T_i^A = \hat{P}_i\left(T_{i+1}^A - I_{i+1}^A H_{i+1}(H_{i+1}^* I_{i+1}^A H_{i+1})^{-1}(F_{Ti+1} - H_{i+1}^* T_{i+1}^A)\right) \qquad T_n^A = 0 \qquad (14)$$

For i = 1 to n

$$\dot{Q}_i = (F_{Ti} - H_i^* I_i^A \hat{P}_{i-1}^* \dot{V}_{i-1} - H_i^* T_i^A)(H_{i+1}^* I_{i+1}^A H_{i+1})^{-1} \qquad \dot{V}_0 = 0 \qquad (15)$$

$$\dot{V}_i = \hat{P}_{i-1}^* \dot{V}_{i-1} + H_i \dot{Q}_i \qquad (16)$$

C. Recursive Factorization and Inversion of Mass Matrix

In [9-11], starting with the factorization in Eq. (10), an alternate factorization of the mass matrix in terms of square factors is derived as

$$\mathcal{M} = (U + \mathcal{H}^* \mathcal{P}^{-1} \mathcal{K})\mathcal{D}(U + \mathcal{H}^* \mathcal{P}^{-1} \mathcal{K})^* \qquad (17)$$

$$\mathcal{E}_P \triangleq U - \mathcal{P} \in \mathbb{R}^{6nx6n}$$

$$\mathcal{I}^A \triangleq \text{diag}\{I_i^A\} \in \mathbb{R}^{6nx6n}$$

$$\mathcal{D} \triangleq \text{diag}\{D_i\} = \text{diag}\{H_i^* I_i^A H_i\} \in \mathbb{R}^{nxn} \qquad (18)$$

$$\mathcal{G} \triangleq \text{diag}\{G_i\} = \mathcal{I}^A \mathcal{H} \mathcal{D}^{-1} \in \mathbb{R}^{6nxn} \qquad (19)$$

$$\mathcal{K} \triangleq \mathcal{E}_P \mathcal{G} \in \mathbb{R}^{6nxn} \qquad (20)$$

The nxn matrices $(U + \mathcal{H}^* \mathcal{P}^{-1} \mathcal{K})$, $\mathcal{D}$, and $(U + \mathcal{H}^* \mathcal{P}^{-1} \mathcal{K})^*$ are, respectively, lower triangular, diagonal, and upper triangular. The factorization in Eq. (17) represents the LDL^* factorization of the SPD mass matrix (which is unique) in an analytical form. Furthermore, due to the positive definiteness of $\mathcal{M}$, the matrix $\mathcal{D}$ is nonsingular, that is, $D_i \neq 0$ (this is also proved in [8]).

In [9-11] it is shown that the inverse of the factor $(U + \mathcal{H}^* \mathcal{P}^{-1} \mathcal{K})$ can be derived in an analytical form as

$$(U + \mathcal{H}^* \mathcal{P}^{-1} \mathcal{K})^{-1} = (U - \mathcal{H}^* \psi \mathcal{K}) \qquad (21)$$

where $\Psi = \{\psi_{i,j}\} \in \mathbb{R}^{6nx6n}$ is a lower triangular matrix with

$\psi_{i,i} = U$ and $\psi_{i,j} = \hat{P}_{i,j}(U - G_i H_i^*) \in \mathbb{R}^{6x6}$, i = n to 1 and j = i-1 to 1 (22)

From Eqs. (17) and (21), a factorization of $\mathcal{M}^{-1}$ is derived as

$$\mathcal{M}^{-1} = (U - \mathcal{H}^* \Psi \mathcal{K})^* \mathcal{D}^{-1} (U - \mathcal{H}^* \Psi \mathcal{K}) \quad (23)$$

The significant contribution of the work in [9-11] is to exploit further structure of the mass matrix (in addition to the symmetry and positive-definiteness) and explicitly obtain the above factorization of $\mathcal{M}^{-1}$. It also demonstrates that the force decomposition in Eq. (11) corresponds to this factorization of $\mathcal{M}^{-1}$. If the articulated-body inertia is computed from Eq. (13) and the terms $\mathcal{K}$ and Ψ are computed according to Eqs. (18)-(20) and (22), then from Eqs. (2) and (23) the solution for $\dot{Q}$ is obtained as

$$\dot{Q} = (U - \mathcal{H}^* \Psi \mathcal{K})^* \mathcal{D}^{-1} (U - \mathcal{H}^* \Psi \mathcal{K}) \mathcal{F}_T \quad (24)$$

In [9-11,14] it is shown that the recursive implementation of Eq. (24) results in an $O(n)$ algorithm whose computational steps (with some minor modifications) correspond to those in Eqs. (13)-(16).

D. Parallelism in the $O(n)$ Algorithms

The main bottleneck in parallel computation of the A-BI algorithm is the computation of I_i^A from Eq. (13), which can be represented, at an abstract level, as the solution of a set of first-order nonlinear recurrences

$$X_i = C_i + \phi_2(X_{i+1})/\phi_1(X_{i+1}) = C_i + \phi(X_{i+1})$$

where C_i is a 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 the parallelism in computation of nonlinear recurrences of the above form and with deg ϕ>1 is bounded [25,26], that is, regardless of the number of processors used, their computation can be speeded up only by a constant factor. This is due to the fact that the data dependency in nonlinear recurrences and particularly those containing division is stronger than in linear recurrences [26]. Hence, the parallelism in the $O(n)$ articulated-body based algorithms is bounded and their parallelization leads to parallel $O(n)$ algorithms which are faster than the serial algorithms only by a constant factor. Note that a rather simple model was used to describe the nonlinear recurrences for computation of the articulated-body inertia, while they are far more complex than those usually studied in the literature, e.g., in [25,26].

However, the computations in Eqs. (14)-(16) can be fully parallelized since they can be transformed into a set of first-order linear recurrences (here,

due to the lack of space, we do not discuss these transformations). This clearly indicates that the main obstacle in parallelization of the $O(n)$ articulated-body based algorithms is the computation of the articulated-body inertia. It should also be mentioned that the $O(n)$ algorithm in [7], which was originally developed for serial chains with 3-DOF spherical joints, involves nonlinear recurrences which are even more complex than those for computation of articulated-body inertia.

III. The Constraint Force Algorithm

A. Basic Force Decomposition and Algorithm

The algorithm in [21-24] is based on a decomposition of interbody force as

$$F_i = H_i F_{Ti} + W_i F_{Si} \tag{25}$$

where F_{Si} is the constraint force and W_i is the orthogonal complement of H_i which is defined [27,28] by

$$H_i H_i^* + W_i W_i^* = U \tag{26}$$

The matrix H_i is a projection matrix and hence

$$H_i^* H_i = U \tag{27}$$

It then follows that the matrix W_i is also a projection matrix and that [27]

$$H_i^* W_i = W_i^* H_i = 0 \text{ and } W_i^* W_i = U \tag{28}$$

For a joint i with ni DOF's (ni<6), it follows that $H_i \varepsilon \mathbb{R}^{6xni}$ and $W_i \varepsilon \mathbb{R}^{6x(6-ni)}$. For a more detailed discussion on these projection matrices see [27,28].

The decomposition in Eq. (25) seems to be more natural (and perhaps more physically intuitive) than those in Eqs. (4) and (11) since it expresses the interbody force in terms of two physically more basic components: the control (or, working) force and the constraint (or, nonworking) force. In fact, as stated in [21], the basic idea of the algorithm was first presented in [29] for a system of particles, and later in [30] it was extended to rigid body systems. However, both works were concerned with the *constraint stabilization* problem and the algorithm had not been used as an alternative procedure for the dynamic simulation problem. Also, the independent derivation of the algorithm in [21-24] was mainly motivated by its suitability for parallel iterative solution of the dynamic simulation problem.

It is not surprising that the algorithm has not been considered as a viable alternative for direct serial and parallel solution of the multibody dynamics problem. The decomposition in Eq. (25) naturally leads to the explicit

computation of the constraint (and interbody) forces, which has also motivated the designation of the algorithm as the Constraint Force (CF) algorithm. In fact, researchers have always argued that since the constraint forces are nonworking forces, their computation is not needed and leads to computational inefficiency. Consequently, the elimination of the constraint forces from the equations of motion has always been considered as a necessary first step in the derivation of efficient algorithms.

Here, for the sake of clarity and self-completeness, we first redrive the algorithm as presented in [21-24]. We then show that the force decomposition in Eq. (25) leads to a new factorization of $\mathcal{M}^{-1}$. This allows a better understanding of the algorithm as well as its comparison with other algorithms, particularly the recursive factorization and inversion of mass matrix.

Equation (25) can be written in global form as

$$\mathcal{F} = \mathcal{H}\mathcal{F}_T + \mathcal{W}\mathcal{F}_S \tag{29}$$

with $\mathcal{W} \triangleq \text{diag}\{W_i\} \in \mathbb{R}^{6n\times 5n}$ and $\mathcal{F}_S \triangleq \text{col}\{F_{Si}\} \in \mathbb{R}^{5n\times 1}$. For global matrices $\mathcal{H}$ and $\mathcal{W}$, Eqs. (26)-(28) are written as

$$\mathcal{H}\mathcal{H}^* + \mathcal{W}\mathcal{W}^* = U, \quad \mathcal{H}^*\mathcal{W} = \mathcal{W}^*\mathcal{H} = 0, \text{ and } \mathcal{H}^*\mathcal{H} = \mathcal{W}^*\mathcal{W} = U \tag{30}$$

From Eqs. (7)-(8) and (30), it follows that

$$\dot{\mathcal{V}} = \mathcal{J}^{-1}\mathcal{P}\mathcal{F} \tag{31}$$

$$\mathcal{W}^*\mathcal{P}^*\dot{\mathcal{V}} = \mathcal{W}^*\mathcal{H}\dot{Q} = 0 \tag{32}$$

$$\mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{F} = 0 \tag{33}$$

and substituting Eq. (29) into Eq. (33) yields

$$\mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}(\mathcal{H}\mathcal{F}_T + \mathcal{W}\mathcal{F}_S) = 0 \Rightarrow \mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{W}\mathcal{F}_S = -\mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H}\mathcal{F}_T, \text{ or} \tag{34}$$

$$\mathcal{A}\mathcal{F}_S = -\mathcal{B}\mathcal{F}_T \tag{35}$$

where $\mathcal{A} \triangleq \mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{W} \in \mathbb{R}^{5n\times 5n}$ and $\mathcal{B} \triangleq \mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H} \in \mathbb{R}^{5n\times n}$. The global constraint force, $\mathcal{F}_S$, is computed as the solution of the linear system in Eq. (35), where $\mathcal{A}$ is a symmetric, positive-definite, block tridiagonal matrix. The global interbody force ($\mathcal{F}$) and acceleration ($\dot{\mathcal{V}}$) are then computed from Eqs. (29) and (31). Finally, the joint accelerations are computed from Eqs. (7) and (30) as

$$\mathcal{H}^*\mathcal{H}\dot{Q} = \mathcal{H}^*\mathcal{P}^*\dot{\mathcal{V}} \Rightarrow \dot{Q} = \mathcal{H}^*\mathcal{P}^*\dot{\mathcal{V}} \tag{36}$$

The solution of the linear system in Eq. (35) represents the most computationally intensive part of the algorithm. In [21-24], exploiting the structure of matrix $\mathcal{A}$ (i.e., symmetry, positive-definiteness, and block

tridiagonal form), a set of iterative algorithms for solution of Eq. (35) is developed. It is shown that these iterative algorithms can be efficiently parallelized and implemented on a simple architecture with n processors while the rest of the computation is performed serially. Although the computational complexity of the developed parallel iterative algorithms is still of $O(n)$, the extensive simulation in [21-24] has shown that the algorithms achieve speedup over the serial A-BI algorithm.

B. A New Factorization of $\mathcal{M}^{-1}$

Here, we extend the work in [21-24] by first deriving an operator form of the algorithm and then showing that the force decomposition in Eq. (25) indeed leads to a new and interesting factorization of $\mathcal{M}^{-1}$. From Eqs. (29) and (34) the global interbody force can be computed as

$$\mathcal{F} = \left(\mathcal{H} - \mathcal{W}(\mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{W})^{-1}\mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H}\right)\mathcal{F}_T \tag{37}$$

From Eqs. (8) and (37), $\dot{\mathcal{V}}$ is computed as

$$\dot{\mathcal{V}} = \mathcal{J}^{-1}\mathcal{P}\left(\mathcal{H} - \mathcal{W}(\mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{W})^{-1}\mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H}\right)\mathcal{F}_T \tag{38}$$

and finally from Eqs. (36) and (38), $\dot{\mathcal{Q}}$ can be computed as

$$\dot{\mathcal{Q}} = \left(\mathcal{H}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H} - \mathcal{H}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{W}(\mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{W})^{-1}\mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H}\right)\mathcal{F}_T \tag{39}$$

which represents a compact operator form of the algorithm. In comparison with Eq. (2), an operator form of $\mathcal{M}^{-1}$, in terms of its decomposition into a set of simpler operators, is given as

$$\mathcal{M}^{-1} = \mathcal{H}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H} - \mathcal{H}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{W}(\mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{W})^{-1}\mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H} \tag{40}$$

C. Alternate Approach for Factorization of $\mathcal{M}^{-1}$ based on the Schur Complement

The operator form of $\mathcal{M}^{-1}$ given by Eq. (40) represents an interesting mathematical construct. To see this, let

$$\mathcal{C} \triangleq \mathcal{H}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H} \in \mathbb{R}^{nxn}$$

$\mathcal{M}^{-1}$ is now written as

$$\mathcal{M}^{-1} = \mathcal{C} - \mathcal{B}^*\mathcal{A}^{-1}\mathcal{B} \tag{41}$$

Consider a matrix $\mathcal{L}$ defined as

$$\mathcal{L} \triangleq \begin{bmatrix} \mathcal{A} & \mathcal{B} \\ \mathcal{B}^* & \mathcal{C} \end{bmatrix} \in \mathbb{R}^{6nx6n} \tag{42}$$

then $\mathcal{C} - \mathcal{B}^* \mathcal{A}^{-1} \mathcal{B}$ is the *Schur Complement* of $\mathcal{A}$ in $\mathcal{L}$ [31] and is designated as $(\mathcal{L}/\mathcal{A})$. The structure of matrix $\mathcal{L}$ motivates a different and simpler approach for derivation of the algorithm. Assume that the spatial acceleration of body i is written in terms of two components: one generated by acceleration of DOF's ($\dot{Q}_i$), and the other generated by acceleration of Degrees-Of-Constraint (DOC's), denoted as $\dot{\sigma}_i \in \mathbb{R}^{5x1}$ (of course, by definition $\dot{\sigma}_i = 0$). Then rewrite Eqs. (5) and (7) as

$$\dot{V}_i - \hat{P}^*_{i-1} \dot{V}_{i-1} = H_i \dot{Q}_i + W_i \dot{\sigma}_i \tag{43}$$

$$\mathcal{P}^* \dot{V} = \mathcal{H} \dot{Q} + \mathcal{W} \dot{\Sigma} \tag{44}$$

with $\dot{\Sigma} \triangleq \text{col}\{\dot{\sigma}\} \in \mathbb{R}^{5nx1}$. From Eqs. (8), (29), and (44), it then follows that

$$\mathcal{P}^* \mathcal{J}^{-1} \mathcal{P} \mathcal{W} \mathcal{F}_S + \mathcal{P}^* \mathcal{J}^{-1} \mathcal{P} \mathcal{H} \mathcal{F}_T = \mathcal{H} \dot{Q} + \mathcal{W} \dot{\Sigma} \tag{45}$$

$\dot{\Sigma}$ and $\dot{Q}$ can be obtained by multiplying both sides of Eq. (45) first by $\mathcal{W}^*$ and then by $\mathcal{H}^*$ as

$$\mathcal{W}^* \mathcal{P}^* \mathcal{J}^{-1} \mathcal{P} \mathcal{W} \mathcal{F}_S + \mathcal{W}^* \mathcal{P}^* \mathcal{J}^{-1} \mathcal{P} \mathcal{H} \mathcal{F}_T = \dot{\Sigma} \tag{46}$$

$$\mathcal{H}^* \mathcal{P}^* \mathcal{J}^{-1} \mathcal{P} \mathcal{W} \mathcal{F}_S + \mathcal{H}^* \mathcal{P}^* \mathcal{J}^{-1} \mathcal{P} \mathcal{H} \mathcal{F}_T = \dot{Q}, \text{ or} \tag{47}$$

$$\mathcal{A} \mathcal{F}_S + \mathcal{B} \mathcal{F}_T = \dot{\Sigma} \tag{48}$$

$$\mathcal{B}^* \mathcal{F}_S + \mathcal{C} \mathcal{F}_T = \dot{Q} \tag{49}$$

Eq. (39) is then obtained by setting $\dot{\Sigma} = 0$ and using the Guassian elimination for solving for $\mathcal{F}_T$. If the vectors of total acceleration ($\dot{a}_G$) and total force ($\mathcal{F}_G$) are defined as

$$a_G \triangleq \begin{bmatrix} \dot{\Sigma} \\ \dot{Q} \end{bmatrix} \in \mathbb{R}^{6nx1} \text{ and } \mathcal{F}_T \triangleq \begin{bmatrix} \mathcal{F}_S \\ \mathcal{F}_T \end{bmatrix} \in \mathbb{R}^{6nx1}$$

then Eqs. (48)-(49) can be written as

$$\mathcal{L} \mathcal{F}_G = \dot{a}_G \tag{50}$$

The matrix $\mathcal{L}$ can be interpreted as the inverse of the augmented mass matrix; it relates the total force and acceleration. $\mathcal{M}^{-1}$ is then the Schur Complement of $\mathcal{A}$ in $\mathcal{L}$, that is,

$$\mathcal{M}^{-1} = (\mathcal{L}/\mathcal{A}) \tag{51}$$

It should be mentioned that an even simpler physical interpretation of the algorithm along with an alternate direct approach for derivation of Eqs. (48)-(49) can be given by noting the physical interpretation of the operators $\mathcal{H}$, $\mathcal{P}$,

$\mathcal{G}^{-1}$, $\mathcal{W}$, etc. and the matrices $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ [32].

It should also be pointed out that by using the matrix identity

$$(C - XDY)^{-1} = C^{-1} + C^{-1}X(D^{-1} - YC^{-1}X)YC^{-1} \tag{52}$$

in Eqs. (40)-(41) an operator expression of $\mathcal{M}$ can be obtained as

$$\begin{aligned}\mathcal{M} &= \mathcal{C}^{-1} + \mathcal{C}^{-1}\mathcal{B}^{*}(\mathcal{A} - \mathcal{B}\mathcal{C}^{-1}\mathcal{B}^{*})^{-1}\mathcal{B}\mathcal{C}^{-1} \\ &= (\mathcal{H}^{*}\mathcal{P}^{*}\mathcal{G}^{-1}\mathcal{P}\mathcal{H})^{-1} + (\mathcal{H}^{*}\mathcal{P}^{*}\mathcal{G}^{-1}\mathcal{P}\mathcal{H})^{-1}(\mathcal{H}^{*}\mathcal{P}^{*}\mathcal{G}^{-1}\mathcal{P}\mathcal{W})\Big[(\mathcal{W}^{*}\mathcal{P}^{*}\mathcal{G}^{-1}\mathcal{P}\mathcal{W}) - (\mathcal{W}^{*}\mathcal{P}^{*}\mathcal{G}^{-1}\mathcal{P}\mathcal{H}) \\ &\quad (\mathcal{H}^{*}\mathcal{P}^{*}\mathcal{G}^{-1}\mathcal{P}\mathcal{H})^{-1}(\mathcal{H}^{*}\mathcal{P}^{*}\mathcal{G}^{-1}\mathcal{P}\mathcal{W})\Big]^{-1}(\mathcal{W}^{*}\mathcal{P}^{*}\mathcal{G}^{-1}\mathcal{P}\mathcal{H})(\mathcal{H}^{*}\mathcal{P}^{*}\mathcal{G}^{-1}\mathcal{P}\mathcal{H})^{-1}\end{aligned} \tag{53}$$

An $O(n^2)$ algorithm for computation of the mass matrix can be derived based on the above expression of $\mathcal{M}$ (which is asymptotically as fast as the best serial algorithms). However, this operator expression of $\mathcal{M}$ is significantly more complex and its associated algorithm is less efficient than other operator expressions and their associated algorithms in Eqs. (10) and (17).

D. Serial Computation of the Constraint Force Algorithm

An efficient serial implementation of the $O(n)$ CF algorithm is based on rewriting Eq. (39) as

$$\dot{Q} = \mathcal{H}^{*}\mathcal{P}^{*}\Big[U - \mathcal{G}^{-1}\mathcal{P}\mathcal{W}(\mathcal{W}^{*}\mathcal{P}^{*}\mathcal{G}^{-1}\mathcal{P}\mathcal{W})^{-1}\mathcal{W}^{*}\mathcal{P}^{*}\Big]\mathcal{G}^{-1}\mathcal{P}\mathcal{H}\mathcal{F}_{T} \tag{54}$$

Here, the key to achieving greater efficiency for both serial and parallel computation is to simply perform, as much as possible, the matrix-vector multiplication instead of matrix-matrix multiplication. In this regard, the matrices $\mathcal{B}$, $\mathcal{B}^{*}$, and $\mathcal{C}$ do not need to be computed explicitly and only the explicit computation of $\mathcal{A}$ is needed. Given $\mathcal{F}_{T}$, the computational steps of the algorithm consist of a sequence of matrix-vector multiplications and vector additions where the matrices, except for $\mathcal{A}^{-1}$, are either bidiagonal or diagonal block matrices. Multiplication of a vector by $\mathcal{A}^{-1}$ is equivalent to the solution of a symmetric, positive-definite, block tridiagonal system.

The vector $\mathcal{F}_{T}$ can be computed in $O(n)$ steps by using the N-E algorithm [3]. The matrix-vector multiplications with diagonal or bidiagonal block matrices can be performed in $O(n)$ steps. The solution of the block tridiagonal system can also be obtained in $O(n)$ steps by using block LDL^{*} factorization [33] in $O(n)$ steps. Therefore, the computational complexity of the serial CF algorithm is of $O(n)$.

Note that, however, Eq. (54) is presented in a coordinate-free form. Hence, before its implementation the tensors and vectors involved in its computation

should be projected onto a suitable frame. The choice of optimal frame for projection of equations and other issues regarding efficient serial implementation of the CF algorithm are extensively discussed in [1], wherein the computation cost in terms of number of operations is also evaluated (see Table II). However, as can be seen, even with the most efficient schemes for serial implementation, the CF algorithm is significantly less efficient than the other algorithms for serial computation (see Table II). For large n, the A-BI algorithm is more efficient than the CF algorithm by a factor of about 2.5 (in terms of the total number of operations) for serial computation. Obviously, for smaller n (say n<12), the CF algorithm is also significantly less efficient than the other $O(n^2)$ or $O(n^3)$ algorithms.

It should be mentioned that the explicit computation of $\mathcal{M}^{-1}$ (though it is not usually needed) can be performed in $O(n^2)$ steps. To see this, note that in Eq. (41) the computation of the term $\mathcal{A}^{-1}\mathcal{B}$ is equivalent to the solution of a block tridiagonal system with n right-hand sides which can be computed in $O(n^2)$ steps. $\mathcal{M}^{-1}$ can then be explicitly computed by performing a matrix-matrix multiplication and a matrix-matrix addition, each in $O(n^2)$ steps. This leads to a total computational complexity of $O(n^2)$ for explicit evaluation of $\mathcal{M}^{-1}$.

IV. New Mass Matrix Factorizations for Computation of Closed-Chain Multibody Dynamic Systems

In this section we briefly discuss the application of the new factorization of $\mathcal{M}^{-1}$ to the computation of dynamics of closed-chain systems. Our discussion follows the treatment of the problem as presented in [34-37], wherein it is shown that the main computational problems are the evaluation of the Operational Space Mass Matrix Λ [38,39] and its inverse Λ^{-1}. Note that the computation of Λ is also required for the task space dynamic control of single robot arms [38,39]. The matrices Λ^{-1} and Λ are defined as

$$\Lambda = (\mathcal{J}\mathcal{M}^{-1}\mathcal{J}^*)^{-1} \in \mathbb{R}^{6x6} \text{ and } \Lambda^{-1} = \mathcal{J}\mathcal{M}^{-1}\mathcal{J}^* \in \mathbb{R}^{6x6} \tag{55}$$

where $\mathcal{J} \in \mathbb{R}^{6x6n}$ is the Jacobian matrix. In [34-37] recursive $O(n)$ algorithms are developed for computation of Λ^{-1}, and the matrix Λ is then computed by explicit inversion of Λ^{-1}. The main computational step in these algorithms is the computation of the articulated-body inertia as in Eq. (13). Therefore, as discussed in Sec. II.D, these $O(n)$ algorithms are also strictly serial.

Here, we show that the new factorization of $\mathcal{M}^{-1}$ directly leads to new factorizations of both Λ^{-1} and Λ as well as new $O(n)$ algorithms for their

computation. These new factorizations are similar to that of M^{-1} since they can be described in terms of the Schur Complement and thus provide simple physical interpretation and a different and deeper insight into the problem. More importantly, however, the resulting algorithms can be parallelized to derive both time- and processor-optimal parallel algorithms, i.e., $O(\log n)$ parallel algorithms with $O(n)$ processors, for computation of Λ^{-1} and Λ.

A. A New Factorization of the Inverse of Operational Space Mass Matrix Λ^{-1}

An operator expression of $\mathcal{J}$ is derived in [34,35]. Using the notation of this paper, this operator expression is given as

$$\mathcal{J} = \beta(\mathcal{P}^*)^{-1}\mathcal{H} \tag{56}$$

where $\beta = [\hat{P}_n^* \; 0 \; 0 \; \ldots \; 0] \varepsilon \mathbb{R}^{6x6n}$. From Eqs. (40) and (56), an operator expression of Λ^{-1} is then derived as

$$\begin{aligned}\Lambda^{-1} &= \beta(\mathcal{P}^*)^{-1}\mathcal{H}\{\mathcal{H}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H} - \mathcal{H}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}W(W^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}W)^{-1}W^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H}\}\mathcal{H}^*(\mathcal{P})^{-1}\beta^* \\ &= \beta\{(\mathcal{P}^*)^{-1}(\mathcal{H}\mathcal{H}^*)\mathcal{P}^*(\mathcal{J}^{-1} - \mathcal{J}^{-1}\mathcal{P}W(W^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}W)^{-1}W^*\mathcal{P}^*\mathcal{J}^{-1})\mathcal{P}(\mathcal{H}\mathcal{H}^*)(\mathcal{P})^{-1}\}\beta^*\end{aligned} \tag{57}$$

The above expression can be simplified by noting that from Eq. (30), we have

$$\mathcal{H}\mathcal{H}^* = U - WW^* \tag{58}$$

By inserting Eq. (58) into Eq. (57) and after some involved algebraic manipulations, a simple operator expression of Λ^{-1} is derived as

$$\Lambda^{-1} = \beta\mathcal{J}^{-1}\beta^* - \beta\mathcal{J}^{-1}\mathcal{P}W(W^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}W)^{-1}W^*\mathcal{P}^*\mathcal{J}^{-1}\beta^* \tag{59}$$

This expression can be further simplified since

$$\mathcal{D} = \beta\mathcal{J}^{-1}\beta^* = \hat{P}_n^* I_n^{-1}\hat{P}_n = I_{n,n+1}^{-1} \tag{60}$$

$$\mathcal{E}^* = \beta\mathcal{J}^{-1}\mathcal{P}W = [\hat{P}_n^* I_n^{-1} W_n \; 0 \; 0 \; \ldots \; 0] \varepsilon \mathbb{R}^{6x5n} \tag{61}$$

This factorization of Λ^{-1} is then written in the form of the Schur Complement

$$\Lambda^{-1} = \mathcal{D} - \mathcal{E}^*\mathcal{A}^{-1}\mathcal{E} \tag{62}$$

Note that the matrix $\mathcal{A}$ is the same as in Eq. (41). Let us define a matrix $\mathcal{L}'$

$$\mathcal{L}' \triangleq \begin{bmatrix} \mathcal{A} & \mathcal{E} \\ \mathcal{E}^* & \mathcal{D} \end{bmatrix} \varepsilon \mathbb{R}^{6nx6n} \tag{63}$$

Λ^{-1} is then the Schur Complement of $\mathcal{A}$ in $\mathcal{L}'$, i.e., $\Lambda^{-1} = (\mathcal{L}'/\mathcal{A})$. As in the previous section, based on the Schur Complement factorization of Λ^{-1} and the structure of matrix $\mathcal{L}'$, a set of linear equations can be formed leading to both simple physical interpretation and alternative derivation of this factorization of Λ^{-1} (see [40] for a more detailed discussion).

From Eq. (62), Λ^{-1} can be explicitly computed in $O(n)$ steps as follows. The term $\mathcal{A}^{-1}\mathcal{E}$ can be computed in $O(n)$ steps since it is equivalent to the solution of a block tridiagonal system with six right-hand sides. Λ^{-1} is then computed by performing a matrix-matrix multiplication and a matrix-matrix addition, each with a cost of $O(1)$, leading to a total computation complexity of $O(n)$. However, usually the multiplication of Λ^{-1} by a vector, say F_{n+1}, rather than the explicit computation of Λ^{-1} is needed. In this case, it is significantly more efficient to directly compute $\Lambda^{-1}F_{n+1}$ rather than first explicitly compute Λ^{-1} and then perform the matrix-vector multiplication. Note that the computation of $\Lambda^{-1}F_{n+1}$ is also done in $O(n)$ steps.

B. A New Factorization of Operational Space Mass Matrix Λ

The operator expression of Λ is derived by using the matrix identity in Eq. (52) for inverting the matrix Λ^{-1} in Eq. (62) as

$$\Lambda = (\mathcal{D} - \mathcal{E}^*\mathcal{A}^{-1}\mathcal{E})^{-1} = \mathcal{D}^{-1} - \mathcal{D}^{-1}\mathcal{E}^*(\mathcal{E}\mathcal{D}^{-1}\mathcal{E}^* - \mathcal{A})^{-1}\mathcal{E}\mathcal{D}^{-1} = (\beta\mathcal{J}^{-1}\beta^*)^{-1} - \tag{64}$$

$$(\beta\mathcal{J}^{-1}\beta^*)^{-1}\beta\mathcal{J}^{-1}\mathcal{P}W\{W^*\mathcal{P}^*(\mathcal{J}^{-1}\beta^*(\beta\mathcal{J}^{-1}\beta^*)^{-1}\beta\mathcal{J}^{-1} - \mathcal{J}^{-1})\mathcal{P}W\}^{-1}W^*\mathcal{P}^*\mathcal{J}^{-1}\beta^*(\beta\mathcal{J}^{-1}\beta^*)^{-1}$$

The factorization of Λ can be further simplified by noting that

$$\mathcal{G} = \mathcal{D}^{-1} = (\beta\mathcal{J}^{-1}\beta^*)^{-1} = (I^{-1}_{n,n+1})^{-1} = I_{n,n+1} \tag{65}$$

$$\beta\mathcal{J}^{-1}\mathcal{P}W = [\hat{P}^*_n I^{-1}_n W_n \; 0 \; 0 \; \ldots \; 0] \tag{66}$$

$$\mathcal{R}^* = (\beta\mathcal{J}^{-1}\beta^*)^{-1}\beta\mathcal{J}^{-1}\mathcal{P}W = [(\hat{P}_n)^{-1}I_n(\hat{P}^*_n)^{-1}\hat{P}^*_n I^{-1}_n W_n \; 0 \; 0 \; \ldots \; 0] \tag{67}$$

$$= [\hat{P}_{n,n+1} W_n \; 0 \; 0 \; \ldots \; 0]$$

$$\mathcal{J}'^{-1} = \mathcal{J}^{-1}\beta^*(\beta\mathcal{J}^{-1}\beta^*)^{-1}\beta\mathcal{J}^{-1} - \mathcal{J}^{-1} = \mathrm{Diag}\{I'^{-1}_i\} \in \mathbb{R}^{6n\times 6n} \tag{68}$$

with $I'^{-1}_n = 0$ and $I'^{-1}_i = -I^{-1}_i$, $i = n-1, n-2, \ldots, 1$

$$\mathcal{S} = W^*\mathcal{P}^*\mathcal{J}'^{-1}\mathcal{P}W \tag{69}$$

Note that the matrix $\mathcal{S}$ is a rank one modification of matrix $\mathcal{A}$ in Eq. (41). The factorization of Λ is then written in terms of the Schur Complement as

$$\Lambda = \mathcal{G} - \mathcal{R}^*\mathcal{S}^{-1}\mathcal{R} \tag{70}$$

Let us define a matrix $\mathcal{L}''$ as

$$\mathcal{L}'' \triangleq \begin{bmatrix} \mathcal{S} & \mathcal{R} \\ \mathcal{R}^* & \mathcal{G} \end{bmatrix} \in \mathbb{R}^{6n\times 6n} \tag{71}$$

Λ is then the Schur Complement of $\mathcal{S}$ in $\mathcal{L}''$, i.e., $\Lambda = (\mathcal{L}''/\mathcal{S})$. Again, based on the Schur Complement factorization of Λ^{-1} and the structure of matrix $\mathcal{L}'$, a set of linear equations can be formed leading to both simple physical

interpretation and alternative derivation of this factorization of Λ^{-1} (see [40] for a more detailed discussion).

As for Λ^{-1}, from Eq. (70) Λ can be explicitly computed in $O(n)$ steps since the term $\mathcal{S}^{-1}\mathcal{R}$ can be computed as the solution of a block tridiagonal system with six right-hand sides. Λ is then computed by performing a matrix-matrix multiplication and a matrix-matrix addition, each with a cost of $O(1)$, leading to a total computation complexity of $O(n)$. However, usually the multiplication of Λ by a vector, say $\dot{V}_{n+1}$, rather than the explicit computation of Λ is needed. In this case, it is significantly more efficient to directly compute $\Lambda\dot{V}_{n+1}$ rather than first explicitly compute Λ^{-1} and then perform the matrix-vector multiplication. Again, note that the computation of $\Lambda\dot{V}_{n+1}$ is done in $O(n)$ steps.

V. Parallel $O(\log n)$ Algorithms for Computation of Open- and Closed-Chain Rigid Multibody Systems

The parallel implementation of the CF algorithm for a serial open-chain system is extensively discussed in [1]. Here, we briefly present the results of [1] and their extension to the computation of closed-chain systems.

A. Parallel $O(\log n)$ Algorithms for Open-Chain Rigid Multibody Systems

The computation of the parallel CF algorithm is performed as follows.

Step I. Projection and Computation of Matrix $\mathcal{A}$

The projection of vectors and tensors and the explicit computation of matrix $\mathcal{A}$ is performed in $O(1)$ steps with n processors.

Step II. Computation of $\mathcal{F}_T$

By using the algorithm in [20], $\mathcal{F}_T$ is computed in $O(\log n)+O(1)$ steps with n processors.

Step III. Computation of $\dot{\Sigma}_T = \mathcal{B}^*\mathcal{F}_T = \mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H}\mathcal{F}_T$ and $\dot{Q}_T = \mathcal{C}\mathcal{F}_T = \mathcal{H}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H}\mathcal{F}_T$

The computation of $\dot{\Sigma}_T$ and $\dot{Q}_S$ involves two sequences of matrix-vector multiplication wherein the matrices are bidiagonal or diagonal block matrices and is performed in $O(1)$ steps with n processors.

Step IV. Computation of $\mathcal{F}_S = \mathcal{A}^{-1}\dot{\Sigma}_T$

The SPD block tridiagonal system is solved in $O(\log n)+O(1)$ steps with n processors and by using the Odd-Even Elimination (OEE) variant of the cyclic reduction algorithm [41].

Step V. Computation of $\dot{Q}_S = \mathcal{B}\mathcal{F}_S = \mathcal{H}^*\mathcal{P}^*\mathcal{I}^{-1}\mathcal{P}\mathcal{W}\mathcal{F}_S$ and $\dot{Q} = \dot{Q}_S + \dot{Q}_T$

The computation of $\dot{Q}_S$ is similar to that of $\dot{Q}_T$ in Step III and is performed in $O(1)$ steps with n processors.

As can be seen, the overall computational complexity of the parallel CF algorithm is of $O(\log n)+O(1)$ by using n processors. In [1] it is shown that the algorithm can be efficiently implemented on an SIMD parallel architecture with n processors and with a Shuffle-Exchange augmented with Nearest-Neighbor (SENN) interconnection. The SENN interconnection allows a perfect mapping, i.e., with no topological variation, of the parallel algorithm since it perfectly matches the inherent communication structure of different steps of the algorithm and thus leads to minimum communication cost. In [1] the computation and communication cost of the parallel CF are evaluated as $(732m+653a)\lceil\log_2 n\rceil+(542m+439a)$ and $(134\lceil\log_2 n\rceil+49)c$, where m, a, and c stand for the cost of multiplication, addition, and communicating a single datum ($\lceil x\rceil$ is the smallest integer greater than or equal to x).

Note that the parallel algorithm, while achieving the time lower bound in the computation, remains highly compute-bound. The ratio of the computation cost over communication cost is greater than 10, which indicates that the parallel algorithm has a rather large grain size and thus can be efficiently implemented on commercially available MIMD parallel architectures. In fact, the parallel CF algorithm is currently being implemented on an MIMD Hypercube parallel architecture. Furthermore, the parallel CF algorithm allows the exploitation of parallelism at several computational levels. In [1] it is shown that a greater speedup in the computation can be achieved by exploiting a multilevel parallelism and implementing the algorithm on an architecture with 3n processors.

B. Parallel $O(\log n)$ Algorithms for Closed-Chain Rigid Multibody Systems

1. Computation of Λ^{-1} and $\Lambda^{-1}F_{n+1}$

The explicit evaluation of matrix $\mathcal{A}$, similar to Step I of Sec. V.A, can be performed in $O(1)$ steps with n processors. The term $\mathcal{A}^{-1}\mathcal{E}$ in Eq. (62) represents the solution of an SPD block tridiagonal system with 6 right-hand sides and can be computed in $O(\log n)+ O(1)$ steps with n processors by using the OEE variant of the cyclic reduction algorithm. Λ^{-1} is then computed by performing a matrix-matrix multiplication and a matrix-matrix addition wherein each operation can be performed in $O(1)$ steps with $O(1)$ processors. This leads to a computational complexity of $O(\log n)+O(1)$ with n processors, which

indicates a both time- and processor-optimal parallel algorithm for evaluating Λ^{-1}. Similar results with greater computation efficiency can be obtained when evaluating $\Lambda^{-1}F_{n+1}$ since the solution of an SPD block tridiagonal system with a single right-hand side is needed.

2. Computation of Λ^{-1} and $\Lambda^{-1}F_{n+1}$

The explicit evaluation of matrix $\mathcal{P}$, similar to that of $\mathcal{A}$, can be performed in $O(1)$ steps with n processors. The rest of the computation in Eq. (70) is similar to that in Eq. (62). Therefore, both Λ^{-1} and $\Lambda^{-1}F_{n+1}$ can be computed in $O(\log n)+O(1)$ steps with n processors, which indicates both time- and processor-optimal parallel algorithms for the computations. Note, however, that, unlike the matrix $\mathcal{A}$, which is always SPD, for some configurations, the matrix $\mathcal{P}$ can become singular [34-37], and thus special care should be taken in computation of Eq. (70). However, it should be mentioned that the new and simple factorization of both Λ^{-1} and Λ provides much better insight for the analysis of the singularity in the computation of Λ [40].

VI. Discussion and Conclusion

In this paper, we presented parallel $O(\log n)$ algorithms for computation of open- and closed-chain rigid multibody dynamics. These parallel algorithms were derived from new $O(n)$ algorithms for the problem. These $O(n)$ algorithms are based on a new force-decomposition strategy which results in new factorizations of $\mathcal{M}^{-1}$, Λ^{-1}, and Λ, presented in Eqs. (40), (62), and (70).

Some important conceptual features of these new algorithms and their underlying factorizations can be summarized as follows.

1. The factorizations of $\mathcal{M}^{-1}$, Λ^{-1}, and Λ are very similar and can be described in terms of the Schur Complement. Due to this similarity, both serial and parallel algorithms involve similar computational steps.

2. Compared to the previous factorization of mass matrix, which is based on a multiplicative decomposition of $\mathcal{M}^{-1}$ (see Eq. (23)), the new factorization leads to an additive decomposition of $\mathcal{M}^{-1}$ which involves simpler matrices, i.e., block tridiagonal matrices.

3. Unlike the previous approaches, wherein Λ^{-1} is first recursively computed and then Λ is obtained by explicit inversion of Λ^{-1}, independent factorizations for both Λ^{-1} and Λ are derived, which allows direct computation of either of them.

From a computational point of view, the main feature of the new algorithms is that they are *strictly* parallel algorithms. That is, as was shown through

Algorithm		Computation Cost	Number of Processors
Serial	A-BI	$586n - 371$	-
	CF	$1500n - 755$	-
Parallel	SL CF	$1385\lceil \log_2 n \rceil + 981$	n
	ML CF	$780\lceil \log_2 n \rceil + 595$	3n
	$O(n^3)$	$6n+69\lceil \log_2 n \rceil + 340$	$n(n+1)/2$

A-BI: Articulated-Body Inertia Algorithm. CF: Constraint Force Algorithm
SL CF: Single Level Parallel CF. ML CF: Multilevel Parallel CF
$O(n^3)$: Parallel $O(n^3)$ algorithm in [2].

Table II. Computation Costs of Serial and Parallel Algorithms

an extensive analysis in [1] for a single serial chain, the algorithm is less efficient than other $O(n)$ algorithms for serial computation (see Table II). However, based on the analysis in Sec. II.D, they are the only known algorithms that can be parallelized and lead to both time- and processor-optimal parallel algorithms for the problem.

The computation costs of different serial and parallel algorithms for the problem are presented in Table II, wherein it is assumed that $m = a$. As can be seen, for small n, the parallel algorithm resulting from parallelization of the $O(n^3)$ algorithm with $O(n^2)$ processors is the most efficient. However, as n increases, so does the efficiency of the parallel $O(\log n)$ algorithms.

As the last point, it should be emphasized that the parallel algorithms developed in this paper--in addition to being theoretically significant by proving, for the first time, the existence of both time- and processor-optimal parallel algorithms for the problem--are also highly practical from an implementation point of view. This is due to their large grain size and simple communication and processor interconnection requirements. In fact, these algorithms are currently being implemented on a Hypercube parallel architecture.

Acknowledgments
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 (NASA). The author gratefully acknowledges many insightful discussions with Dr. G. Rodriguez of JPL regarding different aspects of this research work.

REFERENCES

1. A. Fijany,"Parallel *O*(Log N) Algorithms for Rigid Multibody Dynamics," JPL Eng. Memorandum (Internal Document) EM 343-92-1258, August 1992.
2. A. Fijany and A.K. Bejczy,"Techniques for Parallel Computation of Mechanical Manipulator Dynamics. Part II: Forward Dynamics," in *Advances in Control and Dynamic Systems, Vol. 40: Advances in Robotic Systems Dynamics and Control,* C.T. Leondes (Ed.), pp. 357-410, Academic Press, March 1991.
3. J.Y.S. Luh, M.W. Walker, and R.P.C. Paul,"On-Line Computational Scheme for Mechanical Manipulator," ASME J. Dynamic Syst., Meas., Control, Vol. 102, pp. 69-76, June 1980.
4. M.W. Walker and D.E. Orin,"Efficient Dynamic Computer Simulation of Robotic Mechanism," ASME J. Dynamic Systems, Measurement, and Control, Vol. 104, pp. 205-211, 1982.
5. D.E. Rosental,"Triangularization of Equations of Motion for Robotic Systems," J. Guidance, Control, and Dynamics, Vol. 11, pp. 278-281, 1988.
6. A.F. Vereshchagin,"Computer Simulation of the Dynamics of Complicated Mechanism of Robot Manipulators," Engineering Cybernetics, Vol. 6, pp. 65-70, 1974.
7. W.W. Armstrong,"Recursive Solution to the Equation of Motion of an N-Link Manipulator," Proc. 5th World Congress on Theory of Machines and Mechanisms, pp. 1343-1346, 1979.
8. R. Featherstone,"The Calculation of Robot Dynamics Using Articulated-Body Inertia," Int. J. Robotics Research, Vol. 2(1), pp. 13-30, 1983.
9. G. Rodriguez,"Kalman Filtering, Smoothing and Recursive Robot Arm Forward and Inverse Dynamics," IEEE J. Robotics and Automation, Vol. RA-3(6), pp. 624-639, Dec. 1987.
10. G. Rodriguez and K. Kreutz-Delgado,"Spatial Operator Factorization and Inversion of the Manipulator Mass Matrix," IEEE Trans. Robotics and Automation, Vol. RA-8(1), pp. 65-76, Feb. 1992.
11. G. Rodriguez, K. Kreutz, and A. Jain,"A Spatial Operator Algebra for Manipulator Modeling and Control," Int. J. Robotics Research, vol. 10(4), pp. 371-381, Aug. 1991.
12. D. Rosental,"Order N Formulation for Equations of Motion of Multibody Systems," Proc. SDIO/NASA Workshop on Multibody Simulation, pp. 1122-1150, Sept. 1987.
13. D.S. Bae and E.J. Haug,"A Recursive Formulation for Constraint Mechanical System Dynamics: Part I. Open Loop Systems," Mech. Struct. & Mach., Vol. 15(3), pp. 359-382, 1987.
14. A. Jain,"Unified Formulation of Dynamics for Serial Rigid Multibody Systems," J. Guidance, Control, and Dynamics, Vol. 14(3), pp. 531-542, May/June 1991.
15. A. Fijany and R.E. Scheid,"Efficient Conjugate Gradient Algorithms for Computation of the Manipulator Forward Dynamics," Proc. NASA Conf. Space Telerobotics, Vol. IV, pp. 329-340, Jan. 1989.
16. A. Fijany and R.E. Scheid,"Fast Parallel Preconditioned Conjugate Gradient Algorithms for Robot Manipulator Dynamics Simulation," To appear in J. of Intelligent & Robotic Systems: Theory & Applications, 1992. Also, in JPL Eng. Memorandum (Internal Document) EM 343-1196, Aug. 1991.
17. A. Fijany and A.K. Bejczy,"Parallel Algorithms and Architecture for Computation of Robot Manipulator Forward Dynamics," Proc. IEEE Int. Conf. Robotics and Automation, pp. 1156-1162, April 1991, Sacramento, CA.
18. H. Kasahara, H. Fujii, and M. Iwata,"Parallel Processing of Robot Motion Simulation," Proc. 10th IFAC World Congress, July 1987.
19. C.S.G. Lee and P.R. Chang,"Efficient Parallel Algorithms for Robot Forward Dynamics Computation," IEEE Trans. Syst., Man, and Cybern., Vol. 18(2), pp. 238-251, March/April 1988.

20. C.S.G. Lee and P.R. Chang,"Efficient Parallel Algorithms for Robot Inverse Dynamics Computation," IEEE Trans. Syst., Man, and Cybern., Vol. 16(4), pp. 532-542, July/August 1986.
21. I. Sharf, *Parallel Simulation Dynamics for Open Multibody Chains*, Ph.D. Diss., Univ. of Toronto, Canada, Nov. 1990.
22. I. Sharf and G.M.T. D'Eleuterio,"Parallel Simulation Dynamics for Rigid Multibody Chains," Proc. 12th Biennial ASME Conf. on Mechanical Vibration and Noise, Montreal, Canada, Sept. 1989.
23. I. Sharf and G.M.T. D'Eleuterio,"Computer Simulation of Elastic Chains Using a Recursive Formulation," Proc. IEEE Int. Conf. Robotics and Automation, pp. 1539-1545, Philadelphia, PA, 1988.
24. I. Sharf and G.M.T. D'Eleuterio,"Parallel Simulation Dynamics for Elastic Multibody Chains," Proc. IEEE Int. Conf. Robotics and Automation, pp. 740-747, Cincinnati, OH, 1990.
25. J. Miklosko and V.E. Kotov (Eds.), *Algorithms, Software, and Hardware of Parallel Computers*. Springer-Verlag, 1984.
26. L. Hyafil and H.T. Kung,"The Complexity of Parallel Evaluation of Linear Recurrences," J. ACM, Vol. 24(3), pp. 513-521, July 1977.
27. P.C. Hughes and G.B. Sincarsin,"Dynamics of an Elastic Multibody Chain: Part B - Global Dynamics," Dynamics and Stability of Systems, Vol. 4(3&4), pp. 227-244, 1989.
28. C.J. Damaren and G.M.T. D'Eleuterio,"On the Relationship between Discrete-Time Optimal Control and Recursive Dynamics for Elastic Multibody Chains," Contemporary Mathematics, Vol. 97, pp. 61-77, 1989.
29. J. Baumgarte,"Stabilization of Constraints and Integrals of Motion in Dynamical Systems," Computer Methods in Applied Mechanics and Engineering, Vol. (1), pp. 1-16, 1972.
30. R.E. Roberson,"Constraint Stabilization for Rigid Bodies: An Extension of Baumgarte's Method," Proc. IUTAM Symp. Dynamics of Multibody Systems, pp. 274-289, Munich, 1978.
31. R.W. Cottle,"Manifestation of Schur Complement," Linear Algebra and its Application, Vol. 8, pp. 189-211, 1974.
32. A. Fijany, I. Sharf, and G.M.T. D'Eleuterio,"Parallel O(Log N) Algorithms for Computation of Manipulator Forward Dynamics," Submitted to IEEE Trans. Robot. Automat.
33. G.H. Golub and C.F. Van Loan, *Matrix Computations*, 2nd Edition, The Johns Hopkins Univ. Press, 1989.
34. G.Rodriguez,"Recursive Forward Dynamics for Multiple Robot Arms Moving a Common Task Object," IEEE Trans. Robot. Automat., Vol. 5(4), Aug. 1989.
35. 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, March 1988.
36. K.W. Lilly and D.E. Orin,"Efficient O(n) Computation of the Operational Space Inertia Matrix," Proc. IEEE Int. Conf. Robotics & Automation, pp. 1014-1019, Cincinnati, OH, May 1990.
37. S. McMillan, P. Sadayappan, D.E. Orin,"Efficient Dynamic Simulation of Multiple-Manipulator Systems with Singular Configurations," Proc. IEEE Int. Conf. Robotics & Automation, May 1992.
38. O. Khatib,"The Operational Space Formulation in the Analysis, Design, and Control of Manipulators," 3rd Int. Symp. Robotics Research, 1985.
39. O. Khatib,"A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation," IEEE J. Robot. Automat., Vol. RA-3, pp. 43-53, Feb. 1987.
40. A. Fijany,"New Factorization Techniques and O(n) Algorithms for Computation of Operational Space Mass Matrix and its Inverse," In preparation.
41. R.W. Hockney and C.R. Jesshope, *Parallel Computers*, Adam Hilger Ltd., 1981.

Parallel Methods for Dynamic Simulation of Multiple Manipulator Systems

Scott McMillan P. Sadayappan† David E. Orin

Department of Electrical Engineering
†Department of Computer and Information Science
The Ohio State University
Columbus, OH 43210

Abstract

In this paper, efficient dynamic simulation algorithms for a system of m *manipulators, cooperating to manipulate a large load, are developed; and their performance, using two possible forms of parallelism on a general-purpose parallel computer, is investigated. One form, temporal parallelism, is obtained with the use of parallel numerical integration methods [1]. A speedup of 3.78 on four processors of a CRAY Y-MP8 was achieved with a parallel four-point block predictor-corrector method for the simulation of a four manipulator system. These multi-point methods suffer from reduced accuracy, and when comparing these runs with a serial integration method, the speedup can be as low as 1.83 for simulations with the same accuracy. To regain the performance lost due to accuracy problems, a second form of parallelism is employed. Spatial parallelism allows most of the dynamics of each manipulator chain to be computed simultaneously. Used exclusively in the four processor case, this form of parallelism in conjunction with a serial integration method results in a speedup of 3.1 on four processors over the best serial method. In cases where there are either more processors available or fewer chains in the system, the multi-point parallel integration methods are still advantageous despite the reduced accuracy because both forms of parallelism can then combine to generate more parallel tasks and achieve greater effective speedups. This paper also includes results for these cases.*

1. Introduction

With increases in the structural complexities of today's systems, such as multiple manipulators, multilegged vehicles, and flexible robots, parallelization of the dynamic algorithms for these systems must be considered in an effort to improve computational rates. With significant speedups over previous implementations, real-time performance of graphic animation would make man-in-the-loop remote control of these systems feasible [2]. And with super-real-time simulation (computing seconds of motion in milliseconds), an entirely new approach to on-line robotic control using predictive simulation for planning is within range [3]. One promising area of research which is striving to achieve these computational rates focuses on the use of parallel algorithms.

In previous work, fine-grain parallel algorithms have been developed for robot dynamics computations. Some require the use of special-purpose architectures to implement the fine-grain parallelism of the computations required for a single-chain system [4, 5, 6, 7]. Others decompose the algorithm into groups of concurrent tasks that are scheduled on a number of

tightly coupled parallel processors to produce speedup [8, 9]. All of these algorithms, while designed to perform well on specific parallel architectures, are usually much less efficient when implemented on available general-purpose parallel machines with a limited number of processors. These systems were not designed to handle the large amount of interprocessor communication and synchronization that is required by the fine-grain tasks.

In our work, parallel algorithms which run efficiently on existing general-purpose parallel computers are investigated. In the initial stage of this research, the dynamic algorithms required to simulate a single, open-chain robotic manipulator were effectively parallelized [1]. The approach used a parallel block predictor-corrector integration method to achieve a temporal parallelism which enabled the forward dynamics problem to be computed for multiple points in time simultaneously. In this paper, the work is extended to simulate systems of multiple manipulators that are cooperating to manipulate a large load. In this case, a second form of parallelism which corresponds to the system's structural parallelism is explored. Called spatial parallelism, it allows most of the dynamics of the individual chains to be computed in parallel as well.

This work shows that there are various costs associated with the two forms of parallelism which affect their efficiency. With temporal parallelism, the accuracy of the parallel integration methods is lower than the corresponding serial methods. As a result, smaller stepsizes must be used, and hence, more computation must be performed with the temporal parallel methods to achieve the same accuracy. With spatial parallelism, most of the dynamics may be computed in parallel; however, a serial portion of the algorithm which computes the dynamics of the common load reduces the overall efficiency of this parallelism as well. The advantage of these methods is that they may be combined by implementing the spatial parallelism *within* each parallel integration task to gain even greater speedups. Our results show the effects of reduced efficiency of both methods and how they are combined to gain the greatest speedups on varying numbers of processors.

In the following section, the algorithm to simulate the multiple manipulator system is developed. In this development, the dynamics used in the previous work for a single manipulator are extended and the spatial parallelism in this algorithm is presented. In Section 3, the block predictor-corrector integration methods are presented which provide various amounts of temporal parallelism in the simulation problem. Section 4 discusses how both forms of parallelism are combined and implemented on a general-purpose parallel computer. This algorithm is then implemented on the CRAY Y-MP8/864, and the speedup results are given in Section 5 for various configurations of the simulation system including spatial parallelism only, temporal parallelism only, and a combination on various numbers of the CRAY's processors. Finally, a summary and conclusions are presented in Section 6.

2. Parallel Algorithm for Multiple Manipulator Dynamic Equations

In this section, an efficient dynamic simulation algorithm for a multiple, closed-chain manipulator system is developed, and the spatial parallelism in the algorithm is investigated. The system contains m manipulators, each with N degrees of freedom, that are

rigidly grasping a common load, called the reference member. Each simple, closed chain (manipulator) is governed by the dynamic equations of motion for a single chain. Numbering the chains in the system 1 through m, the equation for each chain is given as follows:

$$\tau_k = \mathbf{H}_k(\mathbf{q}_k)\ddot{\mathbf{q}}_k + \mathbf{C}_k(\mathbf{q}_k, \dot{\mathbf{q}}_k) + \mathbf{G}_k(\mathbf{q}_k) + \mathbf{J}_k^T(\mathbf{q}_k)\mathbf{f}_k \quad \text{for all } k = 1, \ldots, m, \tag{1}$$

where

τ_k	$N \times 1$ vector of torques/forces of the joint actuators,
$\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_k$	$N \times 1$ vectors of joint positions, rates, and accelerations,
$\mathbf{H}_k$	$N \times N$ symmetric, positive definite inertia matrix,
$\mathbf{C}_k$	$N \times 1$ vector specifying centrifugal and coriolis effects,
$\mathbf{G}_k$	$N \times 1$ vector specifying the effects due to gravity,
$\mathbf{J}_k$	$6 \times N$ Jacobian matrix, and
$\mathbf{f}_k$	6×1 force exerted by the tip of chain k on the reference member.

The "for all" in Eq. (1) indicates that the equation may be computed for all chains in parallel provided the required quantities are known. Since we are interested in the solution to the Forward Dynamics (or simulation) problem, the state of the system, consisting of joint positions and rates, and the input joint torques/forces are given. The joint accelerations for each chain must then be computed. Lilly and Orin [10] present an efficient serial algorithm for Forward Dynamics which is the basis for the parallel implementation used in this paper.

By grouping some terms from Eq. (1) and solving for the joint accelerations, we obtain the following equation:

$$\ddot{\mathbf{q}}_k = \ddot{\mathbf{q}}_{k_{open}} - \mathbf{H}_k^{-1}\mathbf{J}_k^T\mathbf{f}_k \quad \text{for all } k = 1, \ldots, m, \tag{2}$$

where $\ddot{\mathbf{q}}_{k_{open}}$ is the vector of joint accelerations for chain k if it were not contacting the reference member causing the tip force to be zero. This is called its open-chain solution which was implemented in [1]. Note that this quantity, as well as $\mathbf{H}_k^{-1}\mathbf{J}_k^T$, depends only on the system state and input joint torques/forces and may be computed from known quantities.

An equivalent operational-space formulation for the dynamics of a closed-chain manipulator can be found by using the following relationship:

$$\dot{\mathbf{x}}_k = \mathbf{J}_k\dot{\mathbf{q}}_k, \tag{3}$$

where $\dot{\mathbf{x}}_k$ is the Cartesian velocity of the tip of the manipulator. Taking the time derivative of both sides yields the equation for the closed-chain acceleration of the tip:

$$\ddot{\mathbf{x}}_k = \dot{\mathbf{J}}_k\dot{\mathbf{q}}_k + \mathbf{J}_k\ddot{\mathbf{q}}_k, \tag{4}$$

and substituting from Eq. (2) yields the following:

$$\ddot{\mathbf{x}}_k = \ddot{\mathbf{x}}_{k_{open}} - \mathbf{\Lambda}_k^{-1}\mathbf{f}_k, \quad \text{for all } k = 1, \ldots, m, \tag{5}$$

where

$$\ddot{\mathbf{x}}_{k_{open}} = \dot{\mathbf{J}}_k\dot{\mathbf{q}}_k + \mathbf{J}_k\ddot{\mathbf{q}}_{k_{open}}, \text{ and} \tag{6}$$

$$\Lambda_k = \left(\mathbf{J}_k \mathbf{H}_k^{-1} \mathbf{J}_k^T\right)^{-1}. \tag{7}$$

The quantity, $\ddot{\mathbf{x}}_{k_{open}}$, is the acceleration of the tip of chain k if it were open, and Λ_k is the operational space inertia matrix [11]. Both quantities may be computed given the current state of the system and the input. The physical interpretation of Λ_k is an inertial quantity which combines the dynamic properties of the chain and projects them to its tip. It is the physical resistance that is "felt" when a force is exerted on the tip, and it defines the relationship between the force that is applied to the tip, and the resulting acceleration of the tip of the chain, $\ddot{\mathbf{x}}_k$. The advantage to using these operational space quantities is that the matrix sizes are constant regardless of the number of degrees of freedom in the chain.

With the chains attached with a fixed grip to the reference member, the accelerations of the tips of each chain, at an appropriate point attached to the end-effector, are the same and are equal to the reference member acceleration, $\mathbf{a}_r$. As a result, Eq. (5) can be rewritten as follows:

$$\mathbf{a}_r = \ddot{\mathbf{x}}_{k_{open}} - \Lambda_k^{-1} \mathbf{f}_k \qquad \text{for all } k = 1, \ldots, m. \tag{8}$$

Now the dynamic behavior of the reference member can be determined using a spatial force balance equation. This states that the sum of the spatial forces exerted by the tips of the chains and any other external forces (including gravity) is related to the acceleration of the reference member through its inertia. This may be written as follows:

$$\sum_{k=1}^{m} \mathbf{f}_k + \mathbf{g}_r = \mathbf{I}_r \mathbf{a}_r + \mathbf{b}_r, \tag{9}$$

where

$\mathbf{g}_r$ 6×1 vector specifying the force of gravity exerted on the reference member,
$\mathbf{a}_r$ 6×1 acceleration vector of the reference member,
$\mathbf{I}_r$ 6×6 spatial inertia of the reference member [10],

and the last term, $\mathbf{b}_r$, is a spatial vector specifying the bias force due to the spatial velocity of the reference member.

In this work, we also assume that the chains are not in singular configurations, and consequently, all of the Λ_k^{-1} matrices are non-singular. Therefore, the unknown force terms, $\mathbf{f}_k$, may be isolated in Eq. (8) and substituted into Eq. (9). After collecting terms, the following equation results:

$$\left[\mathbf{I}_r + \sum_{k=1}^{m} \Lambda_k\right] \mathbf{a}_r = \left[\mathbf{g}_r - \mathbf{b}_r + \sum_{k=1}^{m} \Lambda_k \ddot{\mathbf{x}}_{k_{open}}\right]. \tag{10}$$

This equation states that the sum of all the inertias of the system multiplied by the reference member acceleration is equal to the sum of all the forces that are exerted on the reference member. The acceleration term, $\mathbf{a}_r$, may now be determined from this linear system of

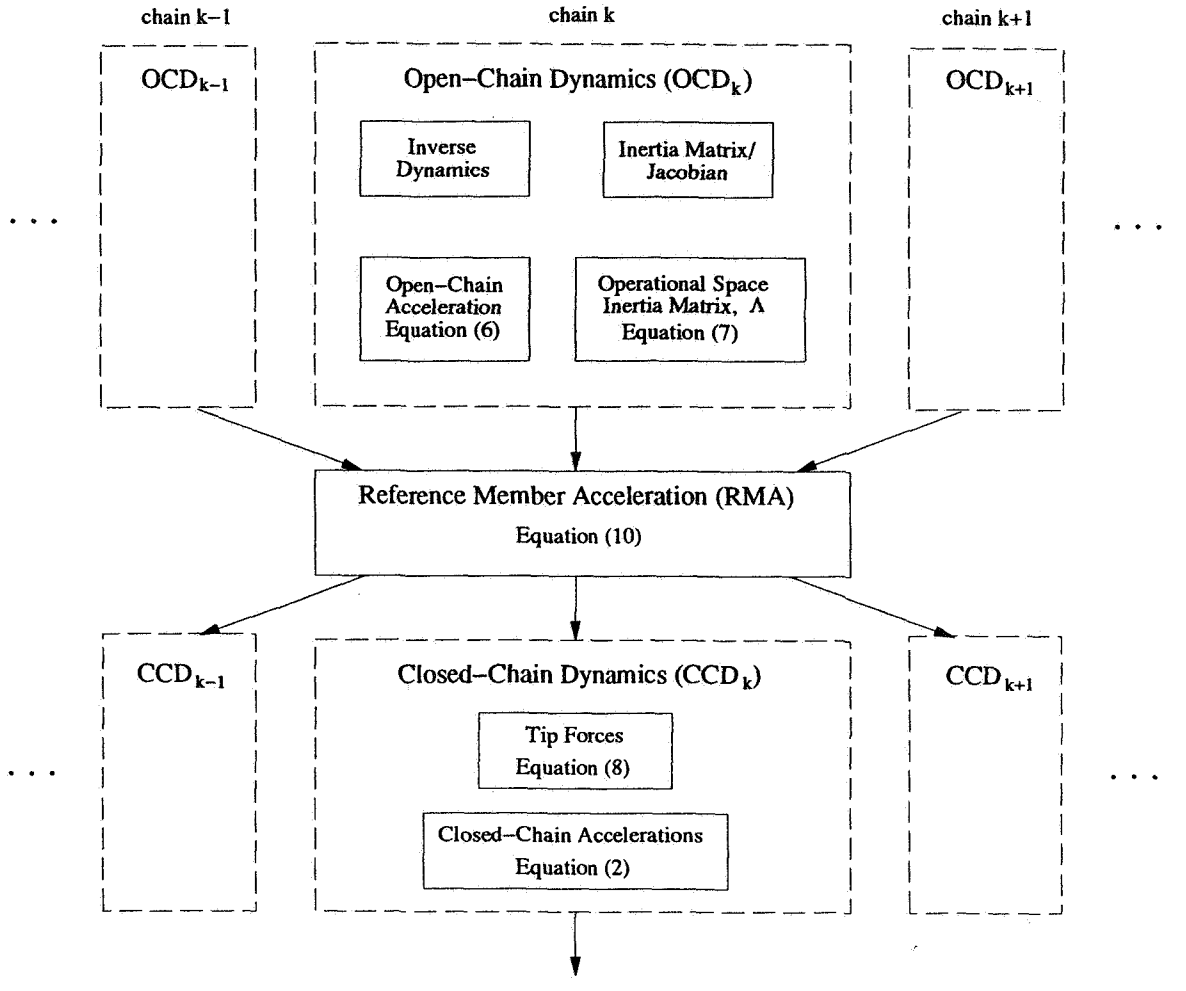


Figure 1: Closed-Chain Dynamics Algorithm.

equations using any linear system solver (in this case Cholesky decomposition can be used). Finally $\mathbf{a}_r$ is substituted back into Eq. (8) to determine the tip forces, $\mathbf{f}_k$, on each chain and this can then be used to determine the joint accelerations from Eq. (2).

The flowchart in Figure 1, outlines the steps in the algorithm to compute the desired accelerations. This constitutes the "derivative computation" which uses the state information that is provided by a numerical integration routine. The additional computation required for the closed chain dynamics that were not present in the open-chain algorithm used in [1] is indicated with the appropriate equation numbers except for the computation of the manipulator Jacobian. With slight modification to the inertia matrix routine that was used in [1], this matrix may be computed as well. The algorithm for this computation can also be found in [12]. Note that the Open-Chain Dynamics (OCD) and Closed-Chain Dynam-

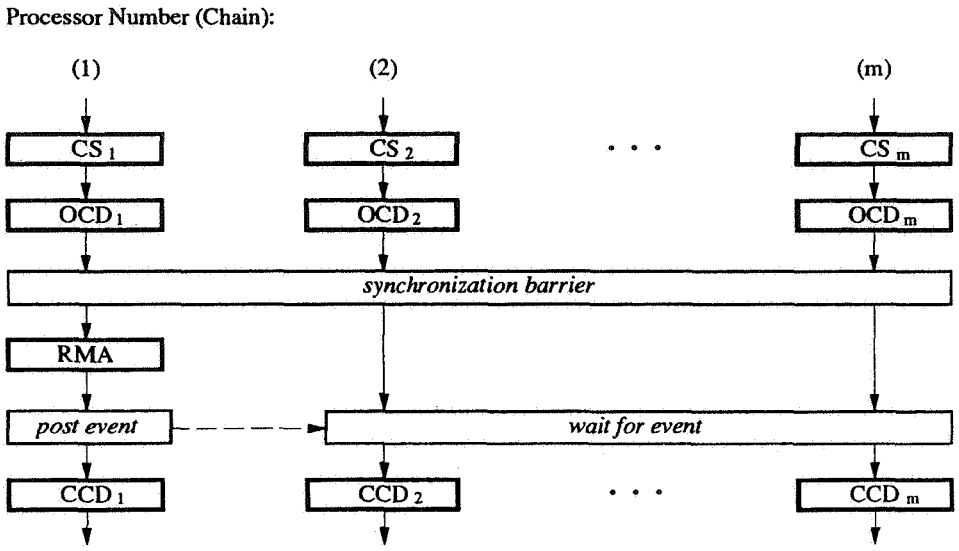


Figure 2: Spatial Parallelism in Multiple-Chain Dynamic Equations (Serial RMA Computation).

ics (CCD) blocks are repeated for each chain in the system, while the Reference Member Acceleration (RMA) is performed once for a given derivative evaluation. This implies that the OCD and CCD computations for each chain may be executed in parallel.

Figure 2 shows how this algorithm, along with a numerical integration method, would be implemented on a general-purpose parallel computer. The CS (Compute State) block consists of the integration method that is used to compute the state of the required chain, and will be discussed in the next section. A processor synchronization is required after the parallel OCD computation in order to collect the chains' operational space inertia matrices and open-chain accelerations before the serial RMA computation is performed. In Figure 2 this is shown as a *barrier* which holds the parallel tasks which have completed the OCD section until all m of them have finished. After the serial computation is completed by one processor, it signals the other processors to continue with the parallel CCD computations. This is accomplished by posting an *event* signal for which the other processors are waiting.

This computation can be modified to remove the event synchronization as shown in Figure 3. This is accomplished by allowing each task to maintain its own "copy" of the reference member information and perform the same computation on all of them. While this introduces redundant computation, it can actually reduce the wallclock time because a costly synchronization has been removed and the same time that was spent waiting for one processor to perform the RMA calculation is now used to perform it on all of the processors. This algorithm is then used within the framework of various parallel integration methods that are described in the next section.

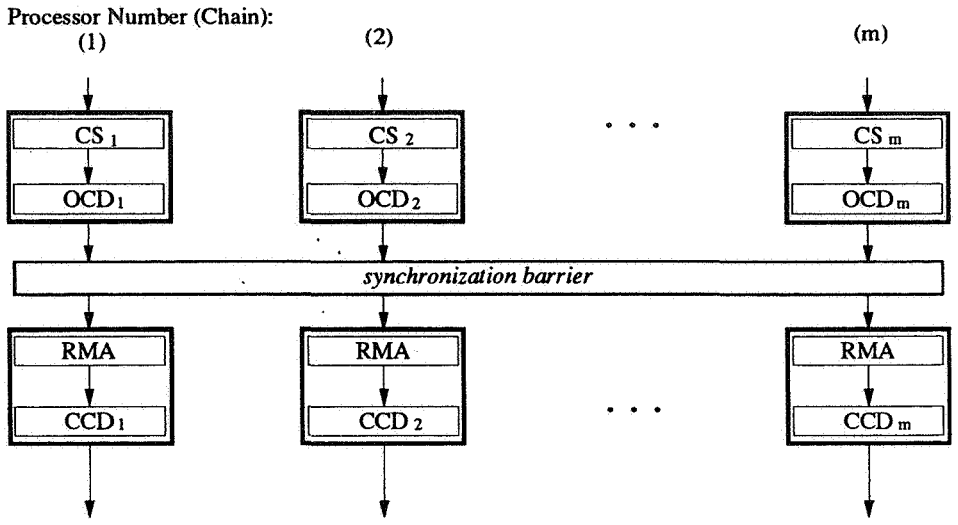


Figure 3: Spatial Parallelism in Multiple-Chain Dynamic Equations (Redundant Parallel RMA Computation).

3. Parallel Integration Methods

Many different algorithms exist to perform numerical integration. One set of algorithms which has shown in the past to be readily parallelizable are predictor-corrector methods. The standard serial algorithm, commonly abbreviated PECE, consists of two pairs of steps which correspond to the letters of the abbreviation. The steps consist of Predicting the next state and Evaluating the derivative based on this prediction, and then Correcting the state with a corresponding derivative Evaluation. The derivative evaluations required in both steps to determine the accelerations were presented in the last section, and the methods for predicting and correcting the states are described here.

In this paper, fifth-order methods are used because they provide an adequate tradeoff between accuracy, which tends to increase with order, and stability, which tends to decrease with order. Figure 4 shows the quantities needed and computed in both steps of the serial method. This method, usually called PECE5, will be referred to as B1PC5 in the rest of this paper to indicate a one-point block method which utilizes fifth-order predictor-corrector formulas. The predictor step, shown in Figure 4(a), uses a linear combination of five past derivative values, $\mathbf{f}_{-4}, \ldots, \mathbf{f}_0$ (hence it is fifth-order) and the state at the most recent point in time, $\mathbf{y}_0$ to predict the state of the system (joint and reference member positions and rates) at the next point in time, $\mathbf{y}_1^p$. With this, the algorithm in the previous section can be used to compute the derivative (joint and reference member accelerations) at this point, $\mathbf{f}_1^p$, which is based on the predicted state. To correct the state in the second step of the method, the quantities shown in Figure 4(b) are used. In this step, the "oldest" derivative value is dropped and the linear combination used to compute the corrected state, $\mathbf{y}_1^c$, now includes the new predicted derivative value. The same state quantity, $\mathbf{y}_0$, is used rather

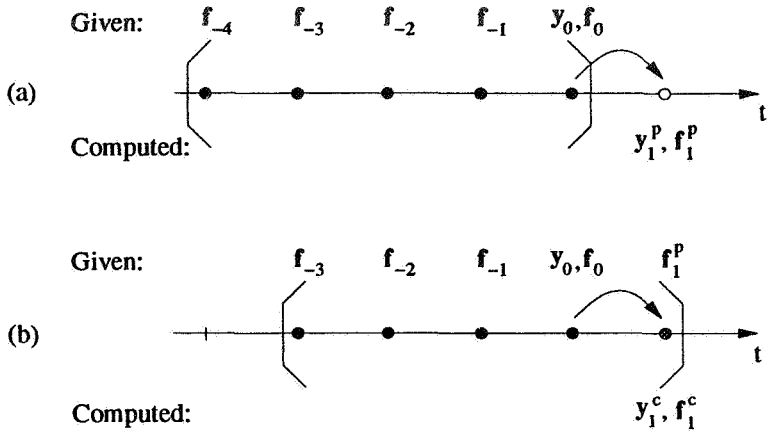


Figure 4: Serial B1PC5 Integration: (a) predictor step, (b) corrector step.

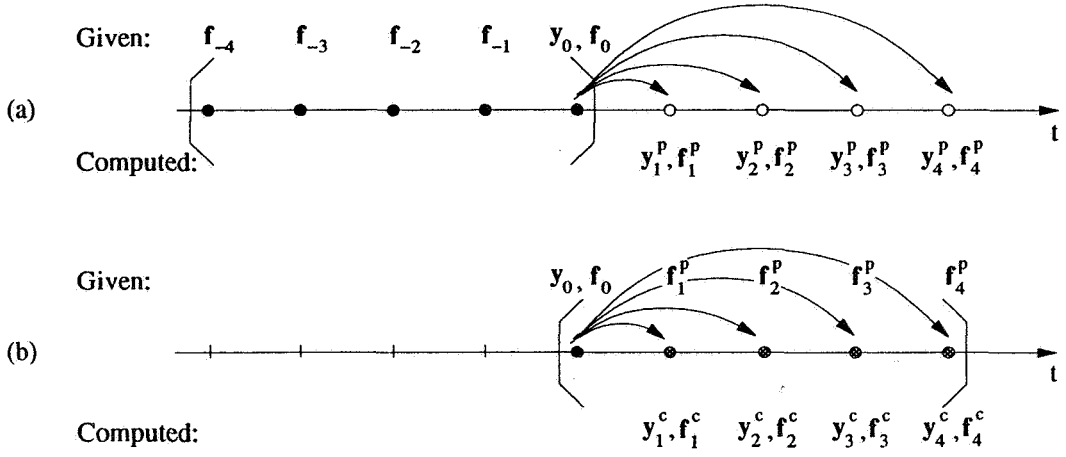


Figure 5: Four-Point Parallel B4PC5 Integration: (a) predictor step, (b) corrector step.

than the predicted one for reasons of stability and accuracy of the method. Finally, the corrected derivative value is computed using $\mathbf{y}_1^c$. Then in the next iteration of the method, the values are shifted and four old derivative values and the new corrected derivative and state values are used.

A parallel version of this method called the block predictor-corrector method was first formulated by Birta and Abou-Rabia [13] and used in our previous work [1]. The fifth-order version of this method, B4PC5, is shown in Figure 5. In this method, a block of four points are computed in parallel during a single iteration. Each processor is responsible for computing the required quantities at a single point in the block. In the predictor step shown in Figure 5(a), each processor computes the predicted value for its point using the same information (but with a different linear combination) in parallel. Then each uses

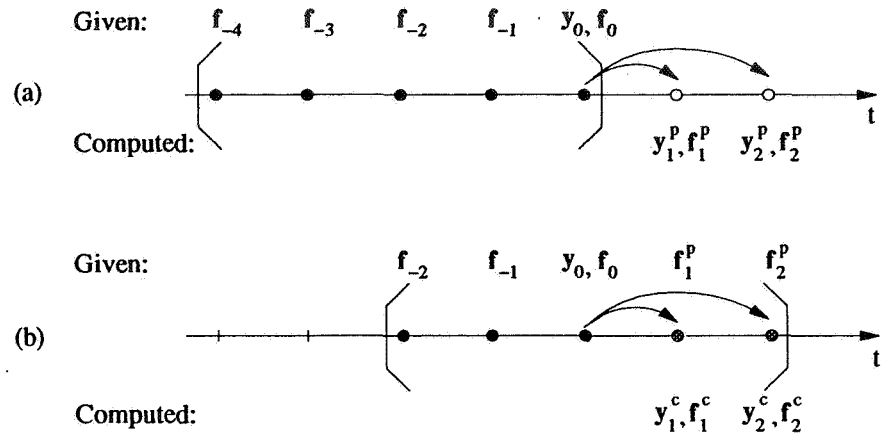


Figure 6: Two-Point Parallel B2PC5 Integration: (a) predictor step, (b) corrector step.

the dynamics algorithm to compute the derivative at its point in parallel as well. In the corrector step, an entire block of old derivative values are discarded in favor of the predicted ones as shown in Figure 5(b). Once the processors have swapped the derivative information from the predictor step, correction and subsequent derivative evaluation can again be done in parallel.

The B4PC5 method offers a way to simulate the system using four processors in parallel. Because the method extrapolates farther from the known values in the predictor step, it is subject to larger errors for a given stepsize. Also, when both forms of parallelism are combined in the next section, the number of parallel tasks may exceed the number of processors available. For these reasons, we developed a two-point parallel method called B2PC5 which lies between the two methods described thus far in terms of parallelism and accuracy. This method is shown in Figure 6. Instead of predicting four points as in the B4PC5, it only predicts two new points and thus can be implemented in parallel on two processors. The coefficients for both steps of this method were computed using the method of undetermined coefficients so that the resulting method is fifth-order as well.

The structure of the temporal parallelism for a single block of the parallel integration methods presented in this section is shown in Figure 7. The PE (compute Predicted state and Evaluate the derivative) and CE (compute Corrected state and Evaluate the derivative) blocks in this figure make up the predictor and corrector steps in these methods along with the derivative evaluations described in the previous section. There are p parallel tasks which correspond to the number of block points computed by the method during a single iteration. A single column of blocks represents the computation of the solution of the entire m-chain system for a specific point in time on one of p processors. Barriers are used to synchronize these parallel tasks after each derivative evaluation so that state and derivative information may be exchanged. An event is also used so that a small amount of serial processing may be performed by a single processor before starting the next integration iteration.

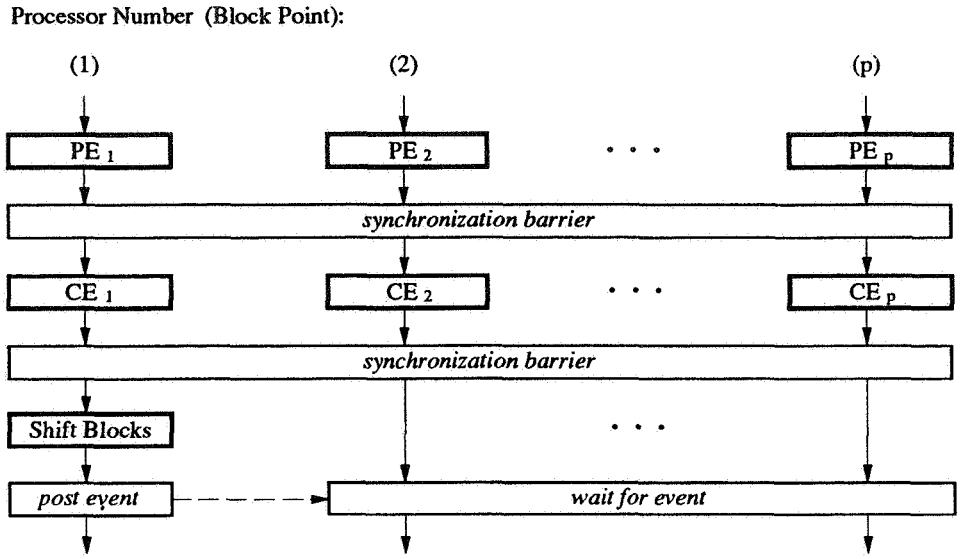


Figure 7: Temporal Parallelism in Dynamic Simulation.

4. Implementation of Combined Parallelism

Both forms of parallelism, spatial and temporal, can be combined by introducing the spatial parallelization of the derivative evaluations in Figure 3 into the individual PE/CE computational blocks of Figure 7. In this case, the CS (Compute State) blocks of Figure 3 become the predictor or corrector equations of the desired integration method. A modification is then made to decrease the required synchronization between all of the tasks to improve the performance. This modification is made to the barriers and events associated with the temporal parallelism. Because the integration of the state of each chain depends only on its own values at all of the block points, the temporal barriers are broken apart so that each one only synchronizes with the processors associated with the same chain. In some sense, these temporal synchronization points have been spatially parallelized. In addition, the serial Shift Blocks task can also be spatially parallelized.

Figure 8 shows the resulting implementation which consists of as many as mp parallel tasks. Also included in this figure are the synchronization commands that are used in the implementation on the CRAY Y-MP8/864. The simulation code was written in FORTRAN (Version 5.0 running under UNICOS Release 6.0) using macrotasking commands to implement the parallelism. The BARSYNC commands correspond to the synchronization barriers and the argument supplied to this command corresponds to the number of parallel tasks that must be synchronized by it. Finally, the EVPOST-EVWAIT commands provide the mechanism by which certain processors are suspended while others perform serial operations.

In order to examine the effects of various configurations on the parallelism (taking the

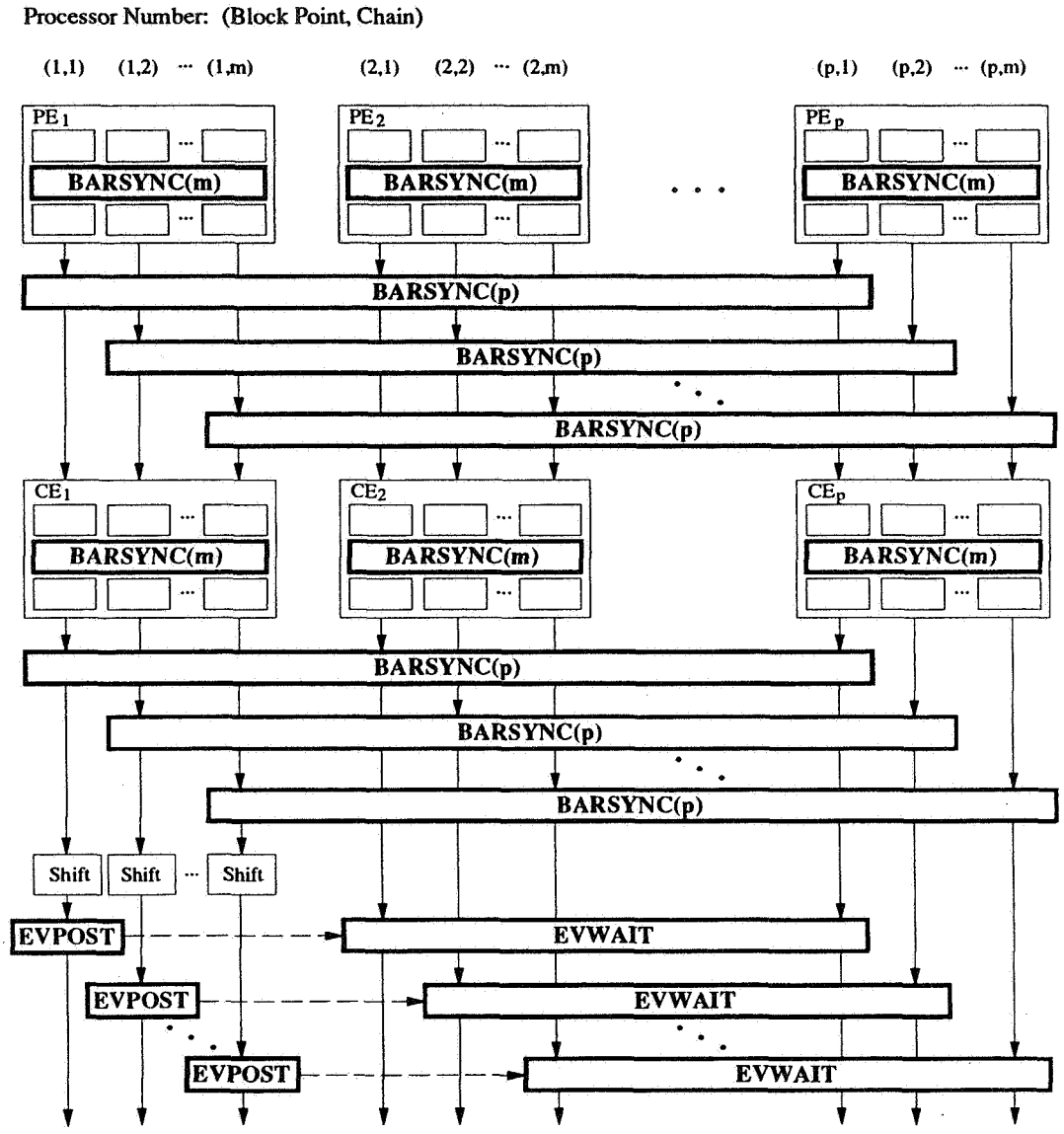


Figure 8: Structure of Combined Spatial and Temporal Parallelism.

number of integration block points, number of chains, and number of physical processors into account), a mechanism was included in the implementation which allows the user to specify the number of desired parallel tasks along both the temporal and spatial dimensions of the simulation regardless of the number of chains and block points. These numbers are denoted by $\tilde{m}$ and $\tilde{p}$, respectively, and are used in place of m and p in Figure 8 when referring to the number of actual parallel computational blocks. With this mechanism, for example, the required computation could be paired off and a single processor could be

Table 1: Speedup Results for B4PC5.

# Processors Requested	Chain Tasks, $\tilde{m}$	Integration Tasks, $\tilde{p}$	T (sec.)	S_p
1	1	1	0.241	1.00
4	1	4	0.0637	3.78
	2	2	0.0668	3.61
	4	1	0.0731	3.30
8	2	4	0.0390	6.18
	4	2	0.0417	5.78

responsible for the computations of two chains (or two block points, or both) instead of just one. This would allow the computation of a larger system, in which mp exceeded the number of processors, to be efficiently mapped to smaller parallel machines. In order to maintain the load balance, however, the specified partitions in both dimensions should be integer divisors of the total number of block points and chains, respectively. Results for various partitions of the computation using this method are given in the next section.

5. Results

With the parallel algorithm described in the last section, a system consisting of four PUMA 560 manipulators was simulated. A test trajectory with a duration of one second was generated which consisted of lifting a 4kg object 0.8 meters straight up. Then an appropriate joint torque profile was computed which, when applied to the joints of the manipulators, would produce the desired motion. These torques were used as input into the simulator, and the error in the final position of the reference member was used as a measure of accuracy for the various integration methods. Only this value was reported for brevity and also its accuracy was representative of the accuracy of the rest of the states in the system.

The simulation was then executed using the B4PC5 integration method on 1, 4, and 8 of the Y-MP's eight processors. Using the mechanism for partitioning the computation among existing processors, the block point and chain computations were partitioned singly (one chain and one block point per processor), in pairs (two chains and two blocks points), all together, or any useful combination thereof. A fixed integration stepsize was used and varied over a number of runs so that a profile of execution time versus error was produced. To get a fair comparison of relative performance between the different integration methods in later experiments, an estimate of the execution time, T, required to achieve an error of 10^{-6} meters was reported in Table 1.

Examining these execution times for a given number of processors, it can be seen that they increased as the number of temporal parallel tasks, $\tilde{p}$, decreased. When $\tilde{p}$ is less than four, the full amount of temporal parallelism provided by this integration method

Table 2: Accuracy Performance for Various Configurations.

# Processors Requested	Chain Tasks, $\tilde{m}$	Integration Tasks, $\tilde{p}$	Block Size	T (sec.)	S_p	S_a	S_T
1	1	1	4	0.241	1.00	0.485	0.485
			2	0.152	1.00	0.770	0.770
			1	0.117	1.00	1.00	1.00
4	1	4	4	0.0637	3.78	0.485	1.83
	2	2	4	0.0668	3.61	0.485	1.75
			2	0.0444	3.42	0.770	2.63
	4	1	4	0.0731	3.30	0.485	1.60
			2	0.0463	3.28	0.770	2.53
			1	0.0377	3.10	1.00	3.10
8	2	4	4	0.0390	6.18	0.485	3.00
	4	2	4	0.0417	5.78	0.485	2.80
			2	0.0295	5.15	0.770	3.97

is not being fully utilized. As a result, increased numbers of redundant reference member acceleration (RMA) calculations are being performed by each processor in an effort to avoid the extra synchronization point that was discussed at the end of Section 2. Consequently, the best speedups due to parallelization, S_p, were achieved by using the greatest amount of temporal parallelism which was as high as 3.78 on four processors. Runs were also made while requesting all eight of the Y-MP's processors and a speedup of 6.18 was still achieved despite an inability to gain dedicated use of these processors in our multi-user environment.

When $\tilde{p}$ was less than four, simulations using an integration method with a smaller block size, such as B2PC5 or serial B1PC5, were also tried. Because these methods compute fewer points per iteration, the cost that is incurred is an increased number of iterations (and hence, parallel task synchronizations) than the B4PC5 method for a given integration stepsize. However, the overall efficiency of these methods increased because they were more accurate. This resulted in fewer iterations and less execution time for a given amount of error. The complete set of results using the various integration block sizes as well as parallel configurations is shown in Table 2.

In this table, S_p is the speedup for a given method over the serial runtime using the same integration method. Since there is an accuracy loss associated with the larger block methods a speedup due to algorithmic changes, S_a, is also reported. This corresponds to the amount of time relative to the B1PC5 method that is required by the methods to compute the trajectory with a given error. Based on the serial results for the three integration methods, the traditional serial method, B1PC5, took less than half the time to simulate the trajectory to the desired accuracy as the B4PC5, and the performance of B2PC5 fell in between. The last column shows the total effective speedup of the various configurations. This is based on the time required for the method to simulate the trajectory to the desired accuracy as compared to the best serial time which was exhibited by the B1PC5 method,

and it is equal to the product of S_p and S_a. When compared with the best integration methods the S_p speedups obtained with the B4PC5 method must therefore be cut in about one-half, and the B2PC5 speedups are reduced by approximately 23%.

Therefore, the best effective speedups were obtained using the integration method with the smallest block size while partitioning the computation so that the number of parallel tasks was equal to the number of processors. Since the system had four chains, the serial B1PC5 method was the best on four processors and the resulting speedup was 3.1. On eight processors, where the serial method could not be used and still have eight parallel tasks, the B2PC5 method showed the greatest speedup at 3.97. From these results it would appear that the B4PC5 has no advantage. However, if a system with a smaller number of chains is to be simulated, this method would allow a greater number of parallel tasks to be generated than the other methods and would most likely exhibit the greatest speedup.

6. Summary and Conclusions

In this paper, two approaches for achieving effective parallelization for dynamic simulation on a general-purpose parallel computer were presented. One approach that was discussed in [1] for parallel simulation of a single chain system was based on temporal parallelism achieved with the use of a parallel numerical integration method. In this paper, the work has been extended to include multiple chain systems which introduce a second form of parallelism. Called spatial parallelism, the form comes from the ability to compute the dynamics of individual chains in the system simultaneously.

Various ways to use both forms of parallelism to the greatest advantage were investigated. The greatest effective speedup from these methods was gained by partitioning the computation into as many load balanced parallel tasks as possible while using the integration method with the smallest block size. This implies that the greatest amount of spatial parallelism, and the most accurate integration methods should be employed.

With the general rule in mind, our results for the simulation of a four chain system showed that the greatest speedup on four processors of the CRAY Y-MP8 was 3.1. This was achieved with spatial parallelism only, and the use of the serial predictor-corrector integration method. Even greater speedup was achieved on eight processors when full spatial parallelism was used. In this case, a two-point parallel integration method was used to achieve the desired amount of parallel tasks. And it appears that the four-point integration method would be beneficial if sixteen processors are available.

An additional benefit to these forms of parallelism is that they do not preclude any of the previous work mentioned in the introduction that dealt with the fine-grain parallel algorithms for computation of the robot dynamics quantities. The special-purpose architectures required could be set up in parallel and used in conjunction with the methods discussed in this paper. The resulting combination of parallel computation could be thought of as occurring in three dimensions, and shows promise for even greater speedups.

7. References

[1] S. McMillan, D. E. Orin, and P. Sadayappan, "Towards Super-Real-Time Simulation of Robotic Mechanisms Using a Parallel Integration Method," *IEEE Transactions on Systems, Man, and Cybernetics*, January/February 1992.

[2] L. Conway, R. Volz, and M. Walker, "Tele-Autonomous Systems: Methods and Architectures for Intermingling Autonomous and Telerobotic Technology," in *Proc. of 1987 IEEE Int. Conf. on Robotics and Automation*, (Raleigh, NC), pp. 1121–1130, 1987.

[3] D. E. Orin, "Dynamical Modelling of Coordinated Multiple Robot Systems," in *Report of Workshop on Coordinated Multiple Robot Manipulators* (A. J. Koivo and G. A. Bekey, eds.), (San Diego, CA), Jan. 1987.

[4] M. Amin-Javaheri and D. E. Orin, "Systolic Architectures for the Manipulator Inertia Matrix," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 18, pp. 939–951, November/December 1988.

[5] C. S. G. Lee and P. R. Chang, "Efficient Parallel Algorithms for Robot Forward Dynamics Computation," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 18, pp. 238–251, March/April 1988.

[6] P. Sadayappan, Y. L. C. Ling, K. W. Olson, and D. E. Orin, "A Restructurable VLSI Robotics Vector Processor Architecture for Real-Time Control," *IEEE Trans. on Robotics and Automation*, vol. 5, pp. 583 – 599, October 1989.

[7] A. Fijany and A. K. Bejczy, "Techniques for Parallel Computation of Mechanical Manipulator Dynamics. Part II: Forward Dynamics," *Control and Dynamic Systems*, vol. 40, pp. 357–410, 1991.

[8] C. L. Chen, C. S. G. Lee, and E. S. H. Hou, "Efficient Scheduling Algorithms for Robot Inverse Dynamics Computation on a Multiprocessor System," in *IEEE Int. Conf. on Robotics and Automation*, (Philadelphia, PA), pp. 1146–1151, 1988.

[9] 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, pp. 214 – 234, March/April 1982.

[10] K. W. Lilly and D. E. Orin, "Efficient Dynamic Simulation for Multiple Chain Robotic Mechanisms," in *Proc. 3rd Annual Conf. on Aerospace Computational Control* (D. E. Bernard and G. K. Man, ed.), vol. 1, (Pasadena, CA), pp. 73–87, December 1989.

[11] 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, pp. 43–53, February 1987.

[12] K. W. Lilly and D. E. Orin, "Alternate Formulations for the Manipulator Inertia Matrix," *The International Journal of Robotics Research*, The MIT Press, vol. 10, pp. 64–74, February 1991.

[13] L. G. Birta and O. Abou-Rabia, "Parallel Block Predictor-Corrector Methods for ODE's," *IEEE Trans. on Computers*, vol. C-36, pp. 299–311, March 1987.

OVERVIEW OF MULTIBODY DYNAMICS ANALYSIS CAPABILITIES

Hon M. Chun and James D. Turner

Fifth Annual
NASA/NSF/DoD Workshop
on
Aerospace Computational Control

Santa Barbara, California
August 17-19, 1992

PRA
Photon Research Associates, Inc.
1033 Massachusetts Avenue
Cambridge, MA 02138

PRA
Photon Research Associates, Inc.

PRESENTATION OVERVIEW

- Historical Overview
- Dynamic Interactions
- Recent Advances
- Current Capabilities
- Unresolved Issues

Photon Research Associates, Inc.

HISTORICAL OVERVIEW

MECHANISMS	SPACECRAFT
Loop Topology	Tree Topology
2D Motion	3D Motion
Lagrangian Equations	Newton/Euler Equations
Joint Libraries	General Joints

Tree & Loop Topologies
2D & 3D Motion
Various Formulations
Joint Libraries and General Joints

Fast Algorithms
Parallel Computers
Symbolic Equations

Photon Research Associates, Inc.

SPACECRAFT COMPLEXITY

SPACECRAFT

GRAVITY GRADIENT BOOMS

NUTATION DAMPERS

SPINNING PARTS OF MOMENTUM EXCHANGE DEVICES

SOLAR ARRAYS

ARTICULATED SUBSTRUCTURES

INCREASING COMPLEXITY

Aerospace interest in stabilization and control of s/c attitude, using gravity-gradient and momentum exchange devices fostered the development of equations of motion for multibody systems.

Photon Research Associates, Inc.

DYNAMIC INTERACTIONS

- Environmental - Thermal Solar Magnetic Gravity
  Plumes Atmosphere
- Active Control - Attitude (Thrusters, Wheels, Gas Jets)
  Structural Optical Maneuver
  Vibration Thermal Deployment
  Joint motion (e.g. manipulator arm, radiometer rotation)
- Rigid/Flex Coupling
- On-Board Disturbances - Fluid Slosh Spinning Devices
  Thermal Astronaut Motion
  Docking Cryogenic Coolers

DYNAMIC INTERACTIONS (CONT'D)

Photon Research Associates, Inc.

- Changing System Parameters -
  - Mass Flow
  - Deployment (Length and Elasticity Change)
  - Aging/Damage
  - Depletion of Consumables

Photon Research Associates, Inc.

RECENT ADVANCES

- ORDER (N) ALGORITHMS, REAL-TIME ALGORITHMS & OTHER ALGORITHMIC SPEED-UPS
- PARALLEL PROCESSING
- SYMBOLIC FORMULATION
- USER-FRIENDLY GRAPHICAL INTERFACES
- HIGH-LEVEL SIMULATION ENVIRONMENT
- JOINT MODELS - CLEARANCES, JOINT FLEXIBILITY, MULTIPLE CONTACT
- SIMPLE IMPACT MODELS
- SENSITIVITY ANALYSIS - RECURSIVE LINEARIZATION

Photon Research Associates, Inc.

RECENT ADVANCES (CONT'D)

- GEOMETRIC STIFFENING (FIRST ORDER CORRECTIONS)
- COMPONENT MODE SHAPES
- VERIFICATION

Photon Research Associates, Inc.

CURRENT CAPABILITIES

ANALYSIS	CAPABILITY	COMMENTS
NUMBER OF BODIES	MANY	• FAST ALGORITHMS • PARALLEL PROCESSING • SYMBOLIC CODE GENERATION
STRUCTURAL MODELS	FROM FINITE ELEMENT, ANALYTICAL, OR CAD	• SMALL ELASTIC DEFLECTIONS
SYSTEM PARAMETERS	CONSTANT	• NO TIME-VARYING MODE SHAPES AND FREQUENCIES • NO MASS AND INERTIA CHANGES DUE TO EXPENDABLE FUELS
DOCKING IMPACT	RIGID - LIMITED FLEXIBLE - NONE	• RESEARCH PROBLEM • IMPULSE FORMULATION
DEPLOYMENT	LIMITED	• SEQUENTIAL ALGORITHMS
CONTROL	GENERAL ALGORITHMS	• INTERFACES WITH CONTROL DESIGN/ANALYSIS SOFTWARE • GENERAL INPUTS BY USER
THERMAL	QUASI-STATIC	• NO RAPID-TRANSIENT HEATING • NEEDED FOR ENVIRONMENTAL, ON-BOARD, AND THREAT MODELING

Photon Research Associates, Inc.

CURRENT CAPABILITIES (CONT'D)

ENVIRONMENTAL DISTURBANCES	GRAVITY GRADIENT MAGNETIC FIELD SIMPLE ATMOSPHERE MODELS	• NO SOLAR RADIATION MODELS • NO PLUME IMPINGEMENT MODELS
DISTRIBUTED PARAMETER SENSING AND CONTROL	NONE	• RESEARCH TOPIC
NONLINEAR JOINT MODELS	LIMITED	• MUCH WORK DONE RECENTLY • JOINT-DOMINATED STRUCTURES DIFFICULT TO DEAL WITH
WAVE MOTION MODELS	NONE	• PROHIBITIVELY LARGE MODELS REQUIRED • PROBLEM MAY GO AWAY WITH ADVANCES IN ALGORITHMS AND COMPUTER HARDWARE • PDE MODELING BEST
PASSIVE DAMPING	NONE (IN-HOUSE PROPRIETARY CODES PERHAPS)	• FINITE ELEMENT MODELING FOR VISCOELASTIC MATERIAL IS A CURRENT RESEARCH TOPIC • PDE MODELS PROVIDE A MORE DIRECT MODELING • FREQUENCY, STRAIN AND TEMPERATURE DEPENDENCE DIFFICULT TO MODEL

Photon Research Associates, Inc.

CURRENT CAPABILITIES (CONT'D)

FLUID-MECHANICAL INTERACTION	SIMPLE PENDULUM MODELS	• FLUID SLOSH
SPECIAL APPLICATIONS	BIOMECHANICS RAIL VEHICLES	• SPECIALIZED USER-FRIENDLY INTERFACES • SPECIALIZED KINEMATICS
HIGH LOAD SYSTEMS	GEOMETRIC STIFFENING	• HIGH ANGULAR RATES • LARGE LOADS DUE TO INERTIAL, APPLIED, OR CONSTRAINT FORCES
DESIGN SENSITIVITY	HELP IN OPTIMIZING SYSTEM PARAMETERS	
LINEARIZATION	COMPUTED ABOUT SPECIFIED OPERATING POINT	• CONTROL DESIGN • SYSTEM EIGENVALUES • TRANSFER FUNCTIONS

PRA Photon Research Associates, Inc.

UNRESOLVED ISSUES/FUTURE WORK

- EXPERIMENTAL VALIDATION AND CROSS VERIFICATION
- BETTER JOINT CHARACTERIZATION
- COMPUTATIONAL EFFICIENCY
- USER INTERFACE (GRAPHICS, VIRTUAL REALITY)
- INTERCONNECTIONS WITH OTHER CODES - FLUIDS, THERMAL
- DEPLOYMENT DYNAMICS WITH MASS FLOW, CHANGING MODE SHAPES AND FREQUENCIES
- IMPACT DYNAMICS, PLASTIC DEFORMATION AND BUCKLING MODELS

Photon Research Associates, Inc.

UNRESOLVED ISSUES/FUTURE WORK (CONT'D)

- SENSITIVITY ANALYSIS FOR OPTIMIZATION OF CONTROL, LINKAGE DIMENSIONS, MASS, INERTIA, ETC.
- SYSTEM IDENTIFICATION
- CONTROL OF FLEXIBLE MULTIBODY SYSTEMS
- KINEMATIC SYNTHESIS - DESIGN LINKAGES TO FOLLOW A CERTAIN TRAJECTORY

Real–Time Dynamics Simulation of the Cassini Spacecraft Using DARTS

Part I. Functional Capabilities and the Spatial Algebra Algorithm

A. Jain and G.K. Man
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA 91109

Abstract

This paper describes the Dynamics Algorithms for Real–Time Simulation (DARTS) real–time hardware–in–the–loop dynamics simulator for the National Aeronautics and Space Administration's Cassini spacecraft. The spacecraft model consists of a central flexible body with a number of articulated rigid–body appendages. The demanding performance requirements from the spacecraft control system require the use of a high fidelity simulator for control system design and testing. The DARTS algorithm provides a new algorithmic and hardware approach to the solution of this hardware–in–the–loop simulation problem. It is based upon the efficient spatial algebra dynamics for flexible multibody systems. A parallel and vectorized version of this algorithm is implemented on a multiprocessor low–cost computer to meet the simulation timing requirements.

1. Introduction

The Cassini mission will be the first to conduct an in–depth study of the Saturnian system by sending a spacecraft and a probe to the planet. The planned launch date is in the mid 1990s with an arrival date in mid 2004. The major scientific goals of the mission are to obtain fundamental new information about the origin and evolution of the solar system, molecular evolution in space and its possible role in the origin of life, and astrophysical plasma dynamics and processes. There are twelve instruments on board and they can be grouped into three categories:

- Scanning–platform–mounted high–precision pointing and scanning instruments for imaging and spectroscopy in visual and infrared.
- Turntable–mounted instruments for plasma, charged particles, and magnetospheric imaging and dust detection.
- Basebody–mounted instruments such as the magnetometer and plasma/radio wave sensors on extended booms, radar mapper for imaging of Titan's surface using the spacecraft high gain antenna.

The spacecraft also carries the Huygens Probe, supplied by the European Space Agency, which contains six instruments to take science measurements as it enters Titan's atmosphere.

Figure 1 shows the deployed Cassini spacecraft. The probe is shown off to the left

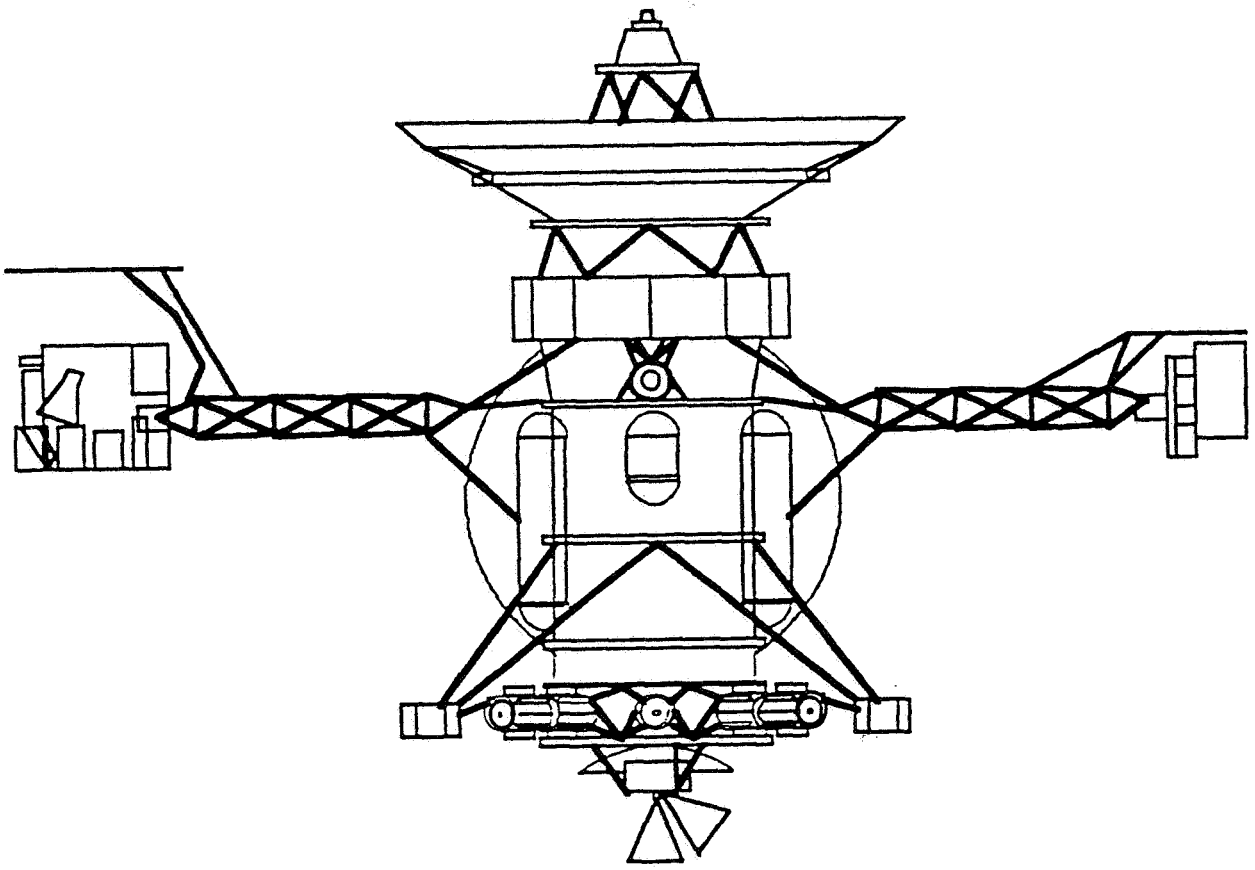


Figure 1: A schematic of the Cassini spacecraft

exposing the spin/eject device. At the top of the spacecraft is the high and low gain antenna. Three booms are attached to the upper equipment module. They carry the high-precision scanning platform (HPSP), the magnetometer and the 10-m plasma/radio wave antenna, and the turntable. The middle spacecraft structure contains the propulsion tanks carrying 68% of the spacecraft mass. At the bottom of the propulsion module is the lower equipment module which supports three radioisotope thermoelectric generators for spacecraft power, four reaction wheels, the articulated probe relay antenna, and the articulable main engine.

The HPSP articulates in two directions, one about the boom axis and one about an orthogonal intermediate axis. The turntable rotates continuously about the boom axis at 0.1, 1.0, or 3.0 rpm. The probe relay antenna has one degree of freedom (dof) about an axis

parallel to the turntable boom. The main engine can be articulated about two axes allowing velocity control along the high gain antenna beam direction.

There are a number of key attitude control functions the spacecraft must perform. The spacecraft must acquire the Sun and certain stars for inertial reference shortly after launch. Then it will maintain Earth/Sun pointed for ground communication and thermal control. There are a number of propulsive maneuvers for plane changes and orbit insertion. During the science phase of the mission when the Huygen Probe will be released into Titan, the data from the probe will be collected and relayed by the spacecraft back to Earth. The spacecraft will spend the next 4 years conducting intensive scientific investigations of the Saturnian system.

The primary attitude control sensors are the star sensors and the gyroscopes located on the HPSP. The key actuators are: (1) the electro-mechanical actuators for the HPSP, the turntable, the main engines, and the reaction wheels; and (2) the chemical propulsion thrusters for attitude control, and the main engines. During main engine firing, the gyroscopes are used as the control sensor and they are separated from the main engine gimbal actuators by the spacecraft bus and boom, which are nonrigid. Furthermore, the bus carries a large amount of liquid propellant. This sensor and actuator noncollocation problem is one that requires high fidelity dynamics simulation of the spacecraft for control design and testing. This simulation must also be used to develop all the control loops including the high precision control loop.

The simulation requirement is most stringent for real–time hardware-in-the-loop testing when flight hardware and software are integrated with a simulation of the spacecraft. In the mid 1980s, technology precluded the use of hardware-in-the-loop simulation for all but the simplest dynamic systems (typically single–axis rigid–body equations). Since the late 1980s, technology has advanced to a point that fairly complex spacecraft can be simulated in real–time but the cost of the computer hardware is very high. Part I of this paper presents a new algorithmic and hardware approach to this important hardware-in-the-loop simulation problem.

The Cassini spacecraft was simplified in June 1992 to cut mission costs. The new baseline eliminated the articulated HPSP and the turntable. The dynamics of the spacecraft were simplified and so were the simulation requirements. We look at the pre–June 1992 Cassini simulation needs and present the DARTS (Dynamics Algorithms for Real–Time Simulation) solution to this technically more challenging problem.

The DARTS algorithm is based upon the spatial algebra flexible dynamics algorithm [1]. Part I of this paper describes the functional capabilities and requirements of the DARTS simulator. It also contains an overview of the spatial algebra algorithm for flexible multibody dynamics. A parallel and vectorized version of this algorithm has been developed and implemented on a multiprocessor real–time computer consisting of a pair of SKYbolt *i*860 vector processors. These processors are high performance computers and are relatively inexpensive. Part II of this paper describes the parallel/vectorization of this algorithm and the real–time computing hardware and implementation.

2. Functional Capabilities of DARTS

The multibody model for the Cassini spacecraft is a *star–topology* dynamics model consisting of a central flexible "extended" bus body (denoted BUS in Figure 2) to which a number of articulated rigid–body appendages are attached. One such model for the spacecraft is shown in Figure 2. In this model, the bus body encompasses all bodies with significant structural

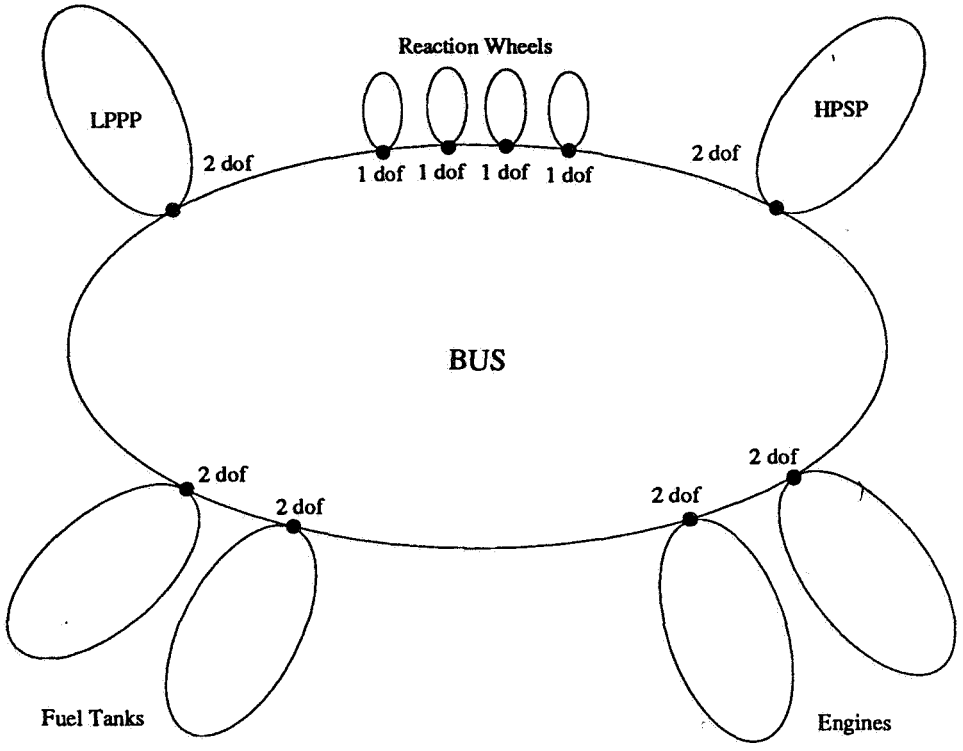


Figure 2: A star–topology multibody model for the Cassini spacecraft

flexibility such as the magnetometer boom and the platform truss structures. The articulated rigid–body appendages in the model include the high–precision scanning platform (HPSP) and the low–precision pointing platform (LPPP), four reaction wheels, two engine assemblies, and a pair of pendulum models for the fuel tanks.

The DARTS simulator has been designed to handle the flexible multibody dynamics of such a model for both the real–time and non–real–time simulation needs for the Cassini Project at the Jet Propulsion Laboratory. Some of the important features of this star–topology implementation of DARTS are described below.

The real–time DARTS was required to compute the generalized accelerations for the spacecraft within 5 ms. This timing requirement was for a model consisting of ten articulated appendages and five flexible assumed modes for the bus with overall 25 degrees of freedom.

During initialization, DARTS reads the data defining the spacecraft star–topology

model from a user specified model file (referred to herein as the MODEL file). Changes to the spacecraft model do not require any additional changes or recompilation of the DARTS software. Thus, spacecraft model changes, such as in the number of appendages, the mass and inertia properties, kinematical properties, the type of hinges, etc., require only the updating of the MODEL file. Evolutionary changes in the spacecraft model, as well as multiple models for early and late mission scenarios, can thus be easily handled using DARTS.

There are no restrictions within DARTS on the number of appendages or on the number of assumed modes used to model the bus flexibility in the spacecraft model. The assumed modes' data for bus flexibility is also provided in the MODEL file. This data includes the modal vectors for the various nodes which serve as attachment points for the appendages, actuators, and sensors on the bus. The modes are assumed to be eigen-modes and, therefore, the stiffness and damping matrices for the bus are diagonal matrices. Spacecraft models with fidelity ranging from rigid-body models to high-fidelity models with a large number of modes can be used by simply changing the MODEL file.

Pin, universal and gimbal rotational hinges, and one degree of freedom prismatic hinges between the appendages and the bus have been implemented in DARTS.

For the most part, during spacecraft simulations, the generalized force for the hinges is provided as an input, while the corresponding generalized accelerations are computed by DARTS. These hinges are referred to as "regular" hinges. DARTS also allows "prescribed motion" hinges - i.e., hinges for whom the generalized acceleration is provided as an input while the corresponding generalized forces are computed by DARTS. Prescribed motion hinge models are required for engine thrust vector control and for testing fault-recovery algorithms. DARTS allows multiple-degree-of-freedom hinges to be a hybrid combination of regular and prescribed motion degrees of freedom. The degrees of freedom can also be switched from regular to prescribed motion models (and vice versa) during run time. This feature is useful for simulating fault events such as hinge lockup as well as the dynamics of probe release.

DARTS allows for an arbitrary number of sources of external forces and moments on the spacecraft. Within DARTS, these external forces and moments are assumed to be applied by actuators located on the spacecraft. The actuators can include physical actuators such as thrusters, as well as pseudo-actuators, used to model disturbance forces from misalignment, comet dust, gravity gradients, etc. These actuators differ from the hinge actuators in that the latter only contribute to the generalized forces for the spacecraft. The external force actuators can be at arbitrary locations on the bus and the appendages. Data regarding the location and number of these actuators is provided in the MODEL file. The external force data is assumed to be defined in the actuator's reference frame.

DARTS allows for an arbitrary number of sensors on the spacecraft. Sensors can include accelerometers, cameras, gyroscopes, sun sensors, etc., and can be at arbitrary locations on the bus and the appendages. DARTS computes the orientation and location of each sensor with respect to the bus frame. The velocity and the acceleration of each sensor are also computed in its own reference frame. The number and location of the sensors are

once again specified through the MODEL file.

DARTS allows a limited number of changes to the spacecraft model during run time. It is possible to switch a hinge degree of freedom between prescribed to regular motion status during run time. There is also a limited (quasistatic) capability for handling changes to the mass and inertia properties of the appendages. This feature can be used to handle changes in fuel mass due to fuel depletion during engine burns.

The kinetic energy, the deformation potential energy, and the linear and angular momenta of the spacecraft are computed by DARTS. Also computed are the location and velocity of the center of mass of the spacecraft.

The structure of the DARTS software is in the form of a subroutine. It is completely portable across different computing platforms and can be used as a part of off–line simulations for control subsystem design as well as for hardware–in–the–loop real–time simulations.

3. The DARTS Spatial Algebra Algorithm

The DARTS dynamics algorithm is based upon the high–speed spatial algebra algorithm for flexible multibody dynamics described in reference [1]. The algorithm also incorporates the new techniques for handling prescribed motion described in reference [2]. A brief overview of the main developments that forms the basis of the DARTS algorithm is presented.

For the sake of brevity, a description of the nomenclature used here is omitted and, instead, details are described in reference [1]. The equations of motion of a (tree-topology) flexible multibody system can be written in the form

$$T = \mathcal{M}\dot{\chi} + \mathcal{C} \tag{3.1}$$

where the mass matrix $\mathcal{M}$ and the vector of Coriolis and centrifugal forces $\mathcal{C}$ are given as follows:

$$\mathcal{M} \triangleq \mathcal{H}\Phi M_m \Phi^* \mathcal{H}^* \in \Re^{N_{dofs} \times N_{dofs}} \quad \text{and} \quad \mathcal{C} \triangleq \mathcal{H}\Phi(M_m \Phi^* a_m + b_m + K_m \vartheta) \in \Re^{N_{dofs}} \tag{3.2}$$

χ and T denote the vectors of generalized velocities and forces for the system. The definition of the spatial operators, such as $\mathcal{H}$, Φ etc., in Eq. (3.2) can be found in reference [1]. The expression for the mass matrix $\mathcal{M}$ in Eq. (3.2) is referred to as its *Newton–Euler Operator Factorization.*

The following equation describes an alternative factorization, known as the *Innovations Operator Factorization* of the mass matrix:

$$\mathcal{M} = [I + \mathcal{H}\Phi K] D [I + \mathcal{H}\Phi K]^* \tag{3.3}$$

In this factorization, the factor $[I + \mathcal{H}\Phi K]$ is square, block lower triangular and nonsingular, while D is a block diagonal matrix. This factorization may be regarded as providing a

closed–form expression for the block LDL^* decomposition of $\mathcal{M}$. The following equation gives the closed–form operator expression for the inverse of the factor $[I + \mathcal{H}\Phi K]$.

$$[I + \mathcal{H}\Phi K]^{-1} = [I - \mathcal{H}\Psi K] \tag{3.4}$$

It follows from Eqs. 3.3 and 3.4 that the operator expression for the inverse of the mass matrix is given by:

$$\mathcal{M}^{-1} = [I - \mathcal{H}\Psi K]^* D^{-1} [I - \mathcal{H}\Psi K] \tag{3.5}$$

Once again, note that the factor $[I - \mathcal{H}\Psi K]$ is square, block lower triangular and nonsingular and so Eq. (3.5) may be regarded as providing a closed–form expression for the block LDL^* decomposition of $\mathcal{M}^{-1}$. The operator expression for the mass matrix inverse in Eq. (3.5) leads to the following operator expression for the generalized accelerations $\dot{\chi}$:

$$\dot{\chi} = [I - \mathcal{H}\Psi K]^* D^{-1} \Big[T - \mathcal{H}\Psi \{ KT + Pa_m + b_m + K_m \vartheta \} \Big] - K^* \Psi^* a_m \tag{3.6}$$

This expression for the generalized accelerations directly leads to a recursive algorithm for computing the dynamics of the system. The structure of this algorithm is very similar in form to the articulated body algorithm for *rigid* multibody systems. The computational cost of this algorithm is reduced by separately processing the flexible and hinge degrees of freedom at each step in the recursion.

Based upon the unique features of the star–topology dynamics model for the Cassini spacecraft, the general spatial algebra algorithm has been simplified to obtain the DARTS algorithm. The primary simplifying features of the model are: (a) only the central bus body is a flexible body; (b) the model has only single body appendages; (c) all of the appendages are rigid bodies; and (d) the spacecraft is a free–flying multibody system. The computational cost of the algorithm for a 10–body spacecraft model is shown in Figure 3.

3.1 Algorithm Structure

Details of the computational steps of the DARTS algorithm can be found in reference [1]. It consists of three recursive sweeps whose structure is shown in Figure 4. Sweep 1 consists of an outward recursion (from the bus to the appendages) to compute the kinematics, velocities, and nonlinear Coriolis and centrifugal terms for all the appendages. This is followed by Sweep 2 which is an inward recursion from each appendage towards the bus. During this recursion, the articulated inertias and residual forces for each of the appendages are computed. The computation of the residual forces takes into account the external forces from any actuators on the appendage as well as the hinge torques for the appendage. The computations for each of the appendages are carried out independently of each other.

When these appendage computations are complete, the results from each appendage are used to compute the overall articulated body inertia and residual force for the bus. Unlike the case of the appendages, the computations here are more complex because of the structural flexibility of the bus. In addition to the rigid–body component, additional modal

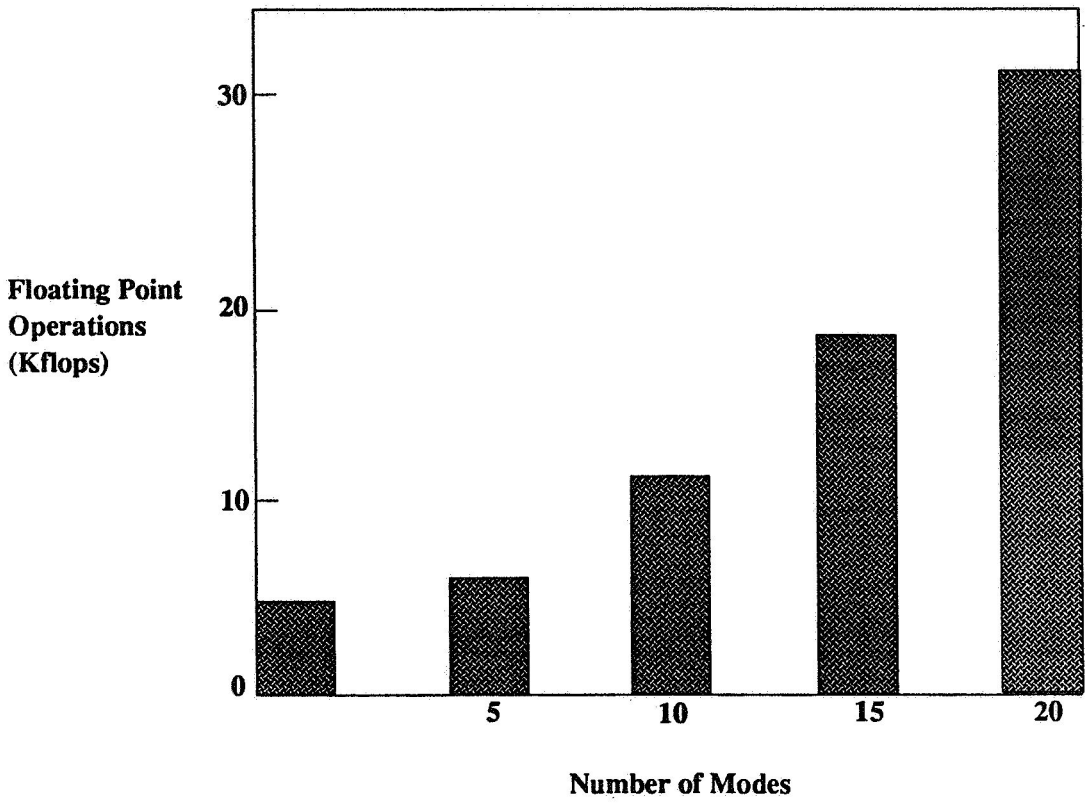


Figure 3: Computational cost versus the number of bus modes for the DARTS algorithm

components have to be computed. The bus articulated body inertia and residual force are used to compute the rigid–body acceleration and modal acceleration for the bus.

This step is followed by Sweep 3, which consists of independent outward recursions from the bus to each of the appendages. During this recursion, the hinge accelerations for each of the appendages are computed. Also as each body is processed, the attitude and velocity information for each sensor is also computed. This completes a single evaluation of the spacecraft dynamics.

3.2 Prescribed Motion

The above algorithm has been modified in order to handle prescribed motion hinges. While most of the hinge degrees of freedom on the spacecraft are "regular," there are some hinge degrees of freedom, such as the engine gimbal assembly, that are modeled as undergoing prescribed motion. Moreover, fault–recovery algorithms are tested by simulating actuator

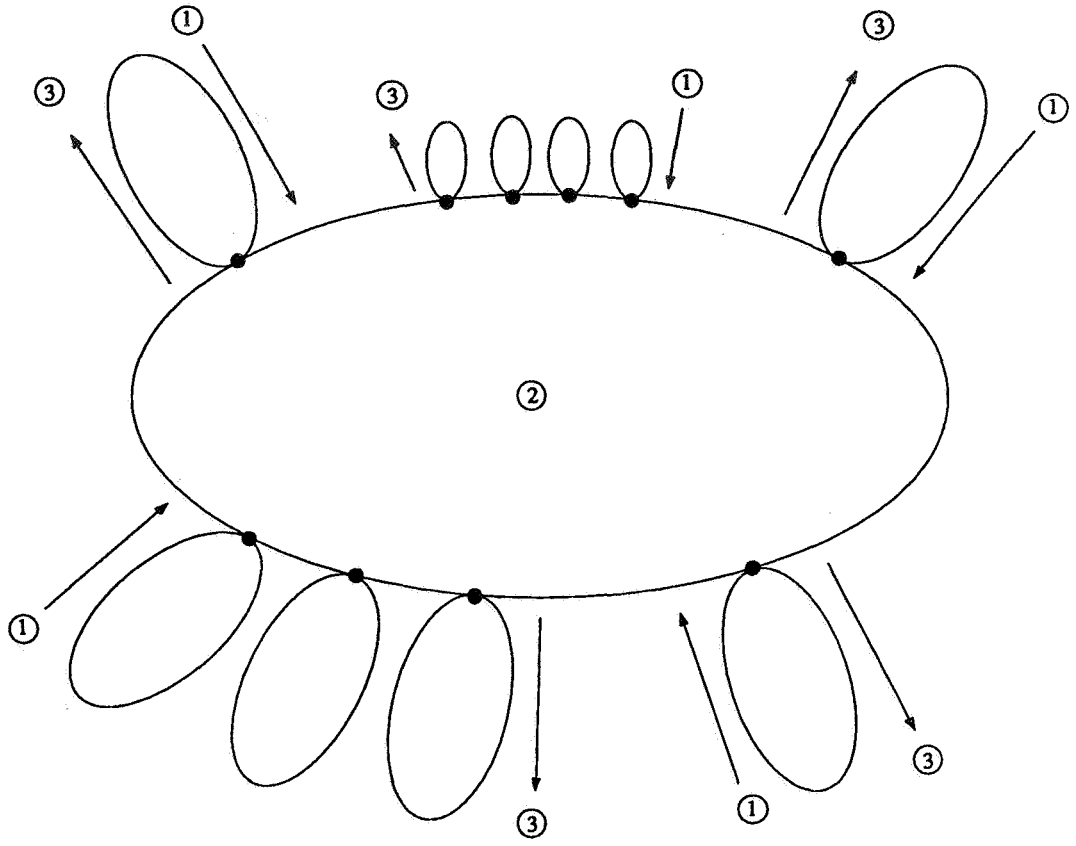


Figure 4: Structure of the star–topology DARTS algorithm

lock–up faults using prescribed motion models. This requires changing the status of the degree of freedom from regular to prescribed motion mode during run time.

With prescribed motion degrees of freedom, the dynamics problem is a "mixed" problem in that for each hinge degree of freedom either the generalized force or the generalized acceleration is known, and the complementary information for each degree of freedom needs to be computed. The prescribed motion features are implemented in DARTS using a recently developed spatial algebra algorithm for prescribed motion [2]. The structure of this algorithm turns out to be a simple variant of the regular dynamics algorithm and retains the same recursive structure. In the prescribed motion algorithm, during the articulated body recursion of Sweep 2, each hinge is checked for its regular/prescribed motion status. Depending on the status, the articulated body inertia and the residual force are computed differently using whichever of the generalized force or the generalized acceleration is known for the degree of freedom. The changes required to handle prescribed motion are entirely local to the hinge and do not affect the computations for any other hinge. The outward recursion of Sweep 3 is similar to Sweep 2 as well. Depending on the regular/prescribed motion status of a hinge, the unknown – the generalized acceleration or the generalized force – for the hinge is computed. Once again, the changes required to do the computational steps are completely local to the hinge and do not affect the computations at any other hinge. This algorithm allows the component degrees of freedom of multiple–degree–of–freedom hinges to

have arbitrary regular/prescribed motion status. This is especially useful for simulating a variety of actuator faults.

The local nature of the changes to the computations for a prescribed motion makes it simple to implement. Moreover, there is little computational overhead when the regular/prescribed motion status of a hinge is changed during run time. This is in contrast with the conventional prescribed motion algorithms which treat the prescribed motion as a global constraint on the dynamics.

4. Conclusions

Part I of this paper describes the functional capabilities of the DARTS flexible dynamics simulator for the Cassini spacecraft as well as the high–speed spatial algebra computational algorithms. The DARTS software is being used throughout the Cassini Project for control algorithm design and analysis, flight software integration and testing, and for real–time hardware–in–the–loop simulation. DARTS has been designed to be completely portable to run on different computing platforms. DARTS has also been designed to be data–driven. Thus DARTS can handle different spacecraft models to meet the various design and testing scenarios without any software modification. This feature considerably simplifies software maintenance since the same software is being used across the whole project. Due to the data–driven feature, the recent significant design changes to the spacecraft have not required any modifications to the DARTS software.

5. Acknowledgment

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] A. Jain and G. Rodriguez, "Recursive Flexible Multibody System Dynamics Using Spatial Operators," *Journal of Guidance, Control and Dynamics*, Nov. 1992. In press.

[2] A. Jain and G. Rodriguez, "Recursive Dynamics Algorithm for Multibody Systems with Prescribed Motion," *Journal of Guidance, Control and Dynamics*, 1992. In press.

Real-Time Dynamic Simulation of the Cassini Spacecraft Using DARTS

Part II: Parallel/Vectorized Real-Time Implementation

A. Fijany, J.A. Roberts, A. Jain, and G.K. Man
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA 91109

1. Introduction

Part I of this paper presented the requirements for the real-time simulation of Cassini spacecraft, along with some discussion of DARTS algorithm. Here, in Part II we discuss the development and implementation of parallel/vectorized DARTS algorithm and architecture for real-time simulation. Development of the fast algorithms and architecture for real-time hardware-in-the-loop simulation of spacecraft dynamics is motivated by the fact that it represents a *hard real-time problem*, in the sense that the correctness of the simulation depends on both the numerical accuracy and the exact timing of the computation. For a given model fidelity, the computation should be completed within a predefined time period. Further reduction in computation time allows increasing the fidelity of the model (i.e., inclusion of more flexible modes) and the integration routine.

An analysis based on the computational structure of DARTS and the specific dynamic model of the spacecraft is made to determine efficient algorithmic/ architectural techniques for achieving real-time simulation capability. This analysis indicates that a combined parallel/vector algorithmic technique along with a multiple vector processors architecture represents the most efficient and cost-effective approach.

The most important (and the new) issue in this paper is the development of the vectorized algorithms for spacecraft dynamic simulation. Until recently, only the users of vector supercomputers for non-real-time applications were concerned about the vectorization issue. Usually, the vectorization was limited to the use of the automatic vectorizers, provided by the vector supercomputers vendors, using an already developed software code. This represents a suboptimal use of the vector supercomputers computing power since the automatic vectorizers have a very limited capability and are efficient only for low level vectorization. For most problems, a significant speedup can be achieved by developing a new algorithm or restructuring the old algorithm by global vectorization of the computation. However, due to the non-real-time nature of the applications, the fact that the vector supercomputers even in scalar mode (for serial computation) were faster than any other serial processor, the time and effort required for the development of new vectorized algorithms and software codes, the users were, most often, satisfied by the suboptimal performance. As a result, the development of the vectorized algorithms has been studied for

very few and mostly regular problems, e.g., matrix-vector operations, direct and indirect (iterative) linear system solution, etc. To our knowledge, the vectorization of multibody dynamics has been only recently studied for a rather simple case of a serial chain of rigid multibody (a robot manipulator) on Cray supercomputers [5,6].

However, this situation is rapidly changing and more effort is being made on the analysis and design of vectorized algorithms and software. This is motivated by the advent of a new generation of low-cost single-board vector processors with computational powers previously offered by the vector supercomputers. These new vector processors provide, for the first time, the opportunity for the design and implementation of high performance embedded computer architecture for real-time applications. However, in order to meet the real-time constraint, efficient vectorized algorithms need to be developed for exploitation of computing power of these new vector processors.

The hardware and software considered in Part II of this paper represent the first generation of such low-cost vector supercomputers. As such, they lacked some important features for efficient vectorization and parallelization. However, since the development of this work, significant improvement has been made on hardware and software of new generations. The approach developed in Part II of this paper and the lessons learned through practical hardware and software implementation, along with these advanced new generations of multiple vector processors, indicate that the real-time simulation capability for more complex systems such as the Space Station is now achievable.

Part II of this paper is organized as follows: Section 2 reviews the different algorithmic and architectural techniques for fast implementation of DARTS, and discusses the features of selected target architecture; Section 3 discusses techniques for global vectorization and efficiency of vector algorithms; Section 4 discusses some implementation issues and several aspects of the implemented algorithms through examples; and finally, Section 5 contains some discussion and concluding remarks.

2. Algorithm and Architecture Selection For Real-Time Simulation

A. AN ANALYSIS OF ALGORITHMIC / ARCHITECTURAL TECHNIQUES FOR FAST IMPLEMENTATION OF DARTS

Generally, there are three algorithmic/architectural techniques that can be used to speed up the computation of a given problem: symbolic manipulation, parallelization, and vectorization. The choice of one or a combination of these techniques depends on: (1) the structure of computational problem, and (2) the availability and cost effectiveness of the required computer architecture.

Symbolic manipulation is a rather straightforward technique that is widely used in multibody dynamics community (see, for example, [12]). Using this technique, a greater computational efficiency can be achieved by eliminating the redundant operations and

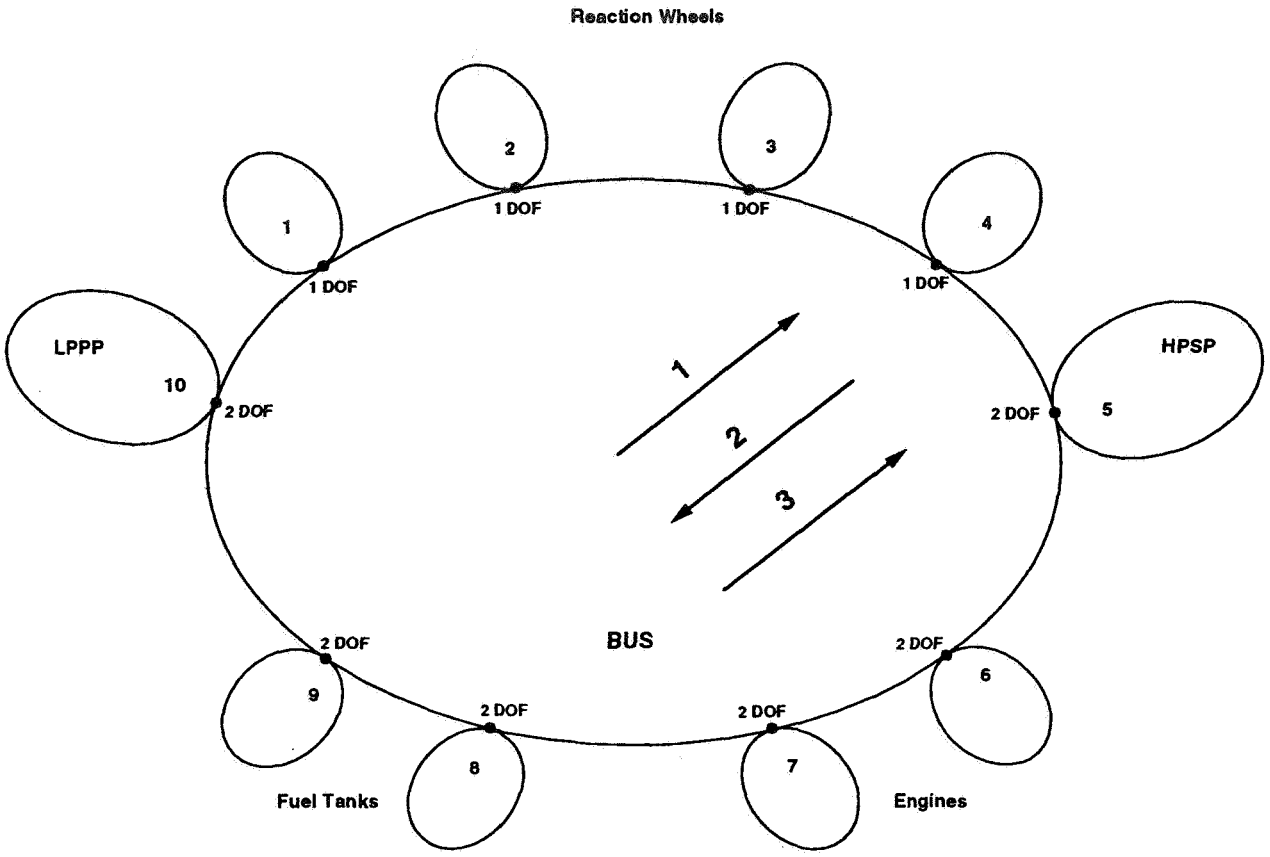


Figure 1. Cassini Spacecraft Star-Topology Dynamics Model and Computational Steps of DARTS

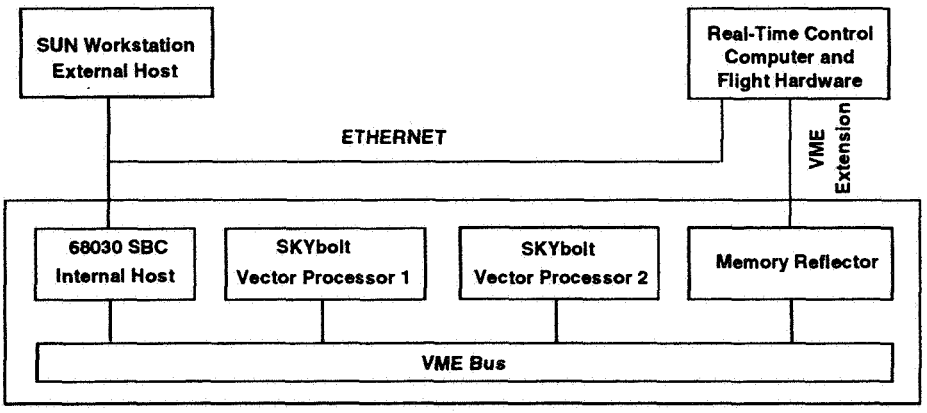


Figure 2. Dedicated Parallel/Vector Computer Architecture for Real-Time Dynamic Simulation of Cassini Spacecraft

generating the symbolic expressions for the equations. However, two issues regarding the application of the symbolic manipulation technique need to be considered. First, the speedup due to the symbolic manipulation should be analyzed in a relative context and not as an absolute one. That is, if the original algorithm has a compact, efficient, and recursive structure–which is the case for DARTS–then the use of symbolic manipulation will not result in a noticeable speed-up. Second, the evaluation of the symbolic expressions is a strictly serial computation. Hence, if symbolic manipulation is used, then it would be difficult to further reduce the computation time by parallelization and/or vectorization. In this case, the only way to reduce the computation time is to use a faster serial processor.

However, both the structure of DARTS and the specific model (star topology and flexible bus) of the Cassini spacecraft make the computation highly suitable for parallelization and/or vectorization. For parallel computation, at first glance it may seem that the computation can be fully parallelized by assigning one processor per body. However, as discussed below, this will lead to a limited speed-up. For vector computation, a large part of the computation can be described in terms of two basic operations: scatter and gather operations, which are highly suitable for vectorization since they involve operations on large matrices and vectors. Furthermore, the size of matrices and vectors increases with both the number of flexible modes and the number of appendages. In order to better assess the suitability of the computation for parallel and/or vector computation and analyze the resulting algorithmic/architectural trade-offs, a more careful study of the structure of the computation is needed. Note that in this study we are only interested in the coarse grain parallelism since it can be exploited by low-cost, commercially available, multiprocessor architecture.

The basic computational steps of the DARTS for the Cassini spacecraft (Figure 1) can be summarized as follows (see [1,3] for a more detailed discussion):

Step I:

1. **Propagate the linear and angular velocity from the bus to appendages.**

   This step is suitable for both parallelization and vectorization. It can be done in parallel for all appendages. It also represents a scatter operation and can be done by performing a single, large matrix-matrix multiplication (see Section 4).

2. **Compute the gyroscopic accelerations and forces of the appendages.**

   This computation is more suitable for parallelization since it can be done in parallel for all appendages. It involves matrix-vector operations with rather small vectors and matrices which makes it less efficient for vectorization (see also Section 4).

Step II:

1. **Propagate Articulated-Body Inertia from appendages to the bus and compute the Articulated-Body Inertia of the bus.**

   The propagation of the Articulated-Body Inertia from appendages to the bus can be performed in parallel for all appendages. But the computation of the Articulated-Body Inertia of the bus remains serial and also involves many-to-one type of interprocessor communication. However, both the propagation of the Articulated-Body Inertia from appendages and the computation of Articulated-Body Inertia of the bus can be described in terms of gather operations, which involve matrix-matrix multiplications with very large matrices and, hence, they are highly efficient for vector computation.

2. **Propagate the residual forces from appendages to the bus and compute the effective residual forces of the bus.**

   Again, the propagation of the residual forces from appendages to the bus can be performed in parallel for all appendages. But the computation of residual force of the bus remains serial and also involves many-to-one type of inter-processor communication. However, both the propagation of the residual forces and the computation of residual force of the bus can be described in terms of gather operations, which involve matrix-vector multiplications of large matrices and vectors and, hence, are highly efficient for vector computation.

Step III:

1. **Compute the acceleration of bus.**

   The computation of acceleration of bus involves the solution of a symmetric, positive definite, linear system which is more suitable for vectorization than for coarse grain parallelization.

2. **Propagate acceleration of bus to appendage.**

   As in Step I.1, this propagation can be performed in parallel but it also involves one type to many types of interprocessor communication. It also represents a scatter operation and can be done by performing a single, large matrix-vector operation.

3. **Compute hinges acceleration.**

   Similar to Step I.2, this computation is more suitable for parallelization since it can be done in parallel for all hinges. It involves matrix-vector operations with rather small vectors and matrices which makes it less efficient for vectorization.

The above analysis clearly suggests that the computation of DARTS for the Cassini

spacecraft can be speeded up by both parallelization and vectorization. Furthermore, a combined parallelization/vectorization algorithmic approach can lead to a speed-up greater than that achievable by either parallelization or vectorization alone. This combined algorithmic approach is further motivated by the emergence of low-cost multiprocessor architectures that employ vector processors, such as Intel i860, as the node processor.

There are, however, two issues that need to be considered in applying this combined algorithmic approach, which also can affect the choice of an optimal target architecture for its implementation. The first issue is that a limited speed-up can be achieved by assigning one processor per body since the ratio of fully parallelizable computations over strictly serial computations isn't very large. This is mainly due to the specific model of Cassini spacecraft; that is, rigidity of the appendages and high flexibility of the bus. In fact, most of the fully parallelizable parts of the algorithm involve the computations for the rigid appendages; e.g., Steps I.2 and III.3, which are less intensive than the strictly serial parts that involve the computations for flexible bus, and also the computation of Articulated-Body Inertia (Step II.1) and acceleration (in Step II.1) of the bus.

Another important factor for efficient parallelization of the computation is the processors interconnection. As stated before, the parallelization of DARTS for Cassini spacecraft involves many-to-one and one-to-many types of processors communication. Therefore, without an interconnection structure that can handle fast processors communication of these types, the communication overhead can degrade the achievable speed-up.

The second issue is that there is a trade-off between the degree of parallelization (i.e., number of processors), and the degree of vectorization. To see this, let us consider those steps that are suitable for both parallelization and vectorization (Steps I.1, II.2, III.2, etc.). For example, in Step I.1., with one processor per appendage, the propagation can still be done by performing matrix-vector operations with small matrices. However, if the number of processors is reduced–which also reduces the speedup due to the parallelization–then the propagation for more than one appendage is done by each processor which implies that the size of matrices and hence the speedup due to vectorization increases.

B. THE CHOICE OF TARGET ARCHITECTURE

Based on the above analysis and given the possible options on the commercially available low-cost multiprocessor architectures (at the time of this project) and other constraints on cost, hardware and software development time and effort, we chose a two-vector processors architecture [2]. This choice was based on our conclusion that, in order to speed up the computation, it was more efficient (both from an algorithmic and architectural point of view) to exploit a limited parallelism but attempt to exploit maximum vectorization.

Figure 2 shows the dynamic simulation system [2]. It consists of a SUN workstation and a VME subsystem. The SUN workstation is the host of system, which is used for software development, and is interconnected through ETHERNET to the VME subsystem. The VME subsystem includes a general-purpose single-board computer based on 68030

processor, which is the local host of VME subsystem, two single board vector processors, and a memory reflector board for high speed interface with another VME system, the real-time control computer.

Each vector processor is a SKYbolt VME bus compatible board with an i860 as the vector processor and an i960 as the communication processor. The choice of the SKYbolt over other commercially available i860-based boards was mainly due to a faster main memory [2]. The SKYbolt was the only one that provided a SRAM main memory while the others had a DRAM main memory with a memory bandwidth of half of that of SRAM main memory. As will be discussed in Section 5, our practical implementation showed that the choice of the SRAM main memory resulted in a very decisive factor in meeting the real-time constraint.

The VME compatibility was basically required for the purpose of integration with and the interface to the rest of the spacecraft hardware-in-the-loop simulation hardware. Based on the vendor's specification, the SKYbolt board provided three communication channels through the VME port, VSB port, and AUX (a fast and private I/O) port [13]. Therefore, it was originally assumed that the communication between the two SKYbolt boards would be performed by using the fast AUX port. However, neither AUX port nor VSB port were functional at the time of our implementation. This forced us to use the VME bus as the communication bus between the two SKYbolt boards. However, the VME interface chips on the SKYbolts were not fully functional. This resulted in a significant loss in the communication speed between the two SKYbolts compared to the nominal speed of the VME bus. As a result, our system was highly imbalanced for parallel computation since the processors' computation speed (particularly in full vector mode) was much greater than the bus' communication speed. This implied that only very coarse grain parallelism with minimum communication requirement could be efficiently exploited by the system. Note that, even by using a fully functional AUX channel the system would have still remained imbalanced. This clearly indicates that, without using an extremely fast communication structure, efficient parallel computation with multiple vector processors such as i860 would not be possible (see Section 5).

3. Vectorization Strategy

The SKYbolt can be used as an accelerator, i.e., simply as a fast serial processor, to speed up the serial computation. According to the vendor's claim, in accelerator mode the SKYbolt can provide a speed-up of about 2 over the SUN Sparc II. Our double precision implementation of serial DARTS algorithm on both the SUN Sparc II and the SKYbolt showed a similar speedup (Table I). This result indicated that, in order to meet the real-time constraint, a greater speedup through vectorization of the algorithm was needed.

The i860 has *peak computational power* of 80 and 64 MFLOPS for single- and double-precision computation. It has most of the functional units of vector supercomputers. However, the vector supercomputers, such as the Cray series, in addition to a fast vector

processing unit, also have a fast (usually the fastest available) scalar processor for serial computation [8,10]. For i860, both vector and serial parts of the computation are performed by the same units. As a result, the i860 has a poor ratio of speed of serial over vector computation because the speed of serial computation (as can be seen by the above comparison with SUN Sparc II) is much less than that of vector computation. Thus, a higher degree of vectorization, even more than that for vector supercomputers, is required for i860 to achieve a satisfactory *sustained computation power.*

The SKYbolt also provides a software environment almost similar to that offered by the vector supercomputers. However, the i860 and the SKYbolt represent the first generation of such low-cost vector processors and, therefore, lack many important features needed for efficient vector computation (see Section 5). Nevertheless, the strategy for vectorization on the SKYbolt is basically similar to that for other vector processors. In the following, we briefly discuss some of the key issues that have been considered in the design, analysis, and implementation of the vectorized version of a DARTS algorithm.

A. LOCAL AND GLOBAL VECTORIZATION TECHNIQUES

There are two techniques for vectorization of a given computation [9,10]: local and global vectorization. The SKYbolt, like vector supercomputers, provides two tools for local vectorization. The first tool is a library of highly optimized routines for matrix-vector and other operations that can be used by subroutine calls. The second tool is an automatic vectorizer that analyzes the data dependency and then vectorizes the computation of innermost Do-loops (i.e., scalar loops) of the overall computation [7,8-10]. Another widely used technique not in the above computation is chaining of the operations [8-10].

However, if the matrices and vectors are small, then the use of optimized routines does not significantly increase the performance. Also, if a code is already developed and optimized for serial computation, then it may have strong data dependency in which even the most advanced automatic vectorizers cannot vectorize. For most problems a greater speedup can be achieved by recasting the algorithm in a form suitable for vector computation, i.e., by a global vectorization. This is more difficult than the local vectorization and can be only done by the algorithm designer [9,10] as it may require major restructuring of the data and computation of the algorithm. In Section 4, we discuss some examples of such global vectorization. It should be also mentioned that our practical implementation indicated that even efficient use of library routines may require restructuring of the computation. There are not well-defined techniques for global vectorization and it is indeed highly problem dependent [9]. Nevertheless, there are several key issues regarding efficient vectorization that need to be taken into account in the design and analysis of vectorized algorithms. These issues are briefly reviewed here. The reader is referred to [7, 9-11] for a more detailed discussion.

B. THE EFFICIENCY OF VECTOR ALGORITHMS

The speedup of vectorized algorithms, like parallel algorithms, is measured according

to the Amdahl's Law.

Let f represent the vectorized fraction of the computation, k the speed of vector operations relative to the speed of scalar operations, and SP the speedup of the vectorized algorithm over serial algorithm. It follows then that

$$SP = \frac{1}{(1-f)+f/k} \tag{1}$$

In order to increase the speedup, f and k should be increased. k is a function of the size of vectors and matrices as well as the type of matrix-vector operation. The computation time, T, of a vector operation is given as:

$$T = \tau + nt \tag{2}$$

where n and t stand for the size of the vectors and the clock time of the vector processor. τ represents the overhead of vector operation due to the loop setup, load and store operations, etc. If n is large enough so $nt >> \tau$ then the computation of vector operation is dominated by nt. That is, k is maximized and the vector processor performs one operation per clock cycle.

There is a vector size below which vector computation becomes less efficient than scalar computation. This size is called breakeven point [7] and is designated as n_B . The value of n_B depends on the type of operation. There is no information on n for the i860. Although originally we suspected n_B to be rather small [3], in practical implementation and for various matrix-vector operations we found n_B to be quite large (several times that for Cray series), which indicates that only operations on very large matrices and vectors can be efficiently implemented on the i860.

As a conclusion, in vectorizing the algorithm an attempt should be made to:

(1) increase the number of matrix-vector operations, and hence increase f; and

(2) increase the size of the vectors and matrices, n, so that $n >> n_B$, and hence increase k.

C. MEMORY BANDWIDTH AND DATA ORGANIZATION

For vector processing, the data movement may sometimes take more time than the computation (see the example in Section 4). Therefore, the second issue in analyzing the performance of vector supercomputers is the data structure. To efficiently use the high speed floating-point units, data should be fed with adequate speed. In the pipelined mode, the i860 can initiate two floating-point operations (one add and one multiply) per clock. This requires fetching four operands and storing two results per clock which indeed requires a very high bandwidth memory. To achieve such a high bandwidth, the vector

supercomputers use a hierarchical memory organization. However, in addition to the memory organization, the data structuring is also needed for achieving and maintaining the high bandwidth. For example, while the i860 can perform two floating-point operations per clock cycle, fetching an operand from an arbitrary location in the main memory can take several clock cycles.

The memory organization of vector supercomputers usually includes a set of registers, as a fast and limited size memory; a cache memory, as medium-size medium-speed memory; and a main memory, as a large and slow memory. The i860 has a set of thirty two 32-bit data registers and 8 Kbyte (1 K double-precision) data cache. The selected SKYbolt board provides a 2-Mbyte fast SRAM memory as the main memory. Unlike the register-oriented vector supercomputers, such as Cray series, which utilize a larger size register (in the order of Kbyte), the i860 has a rather small size set of registers. However, it is claimed that the cache memory can be used with the same performance as the registers for vector operations [4].

To minimize the data movement overhead, the following issues need to be considered:

1. Data Contiguity:

   The related data should be located, as much as possible, in the contiguous locations in the cache and main memory. Obviously for vector operation the elements of the vectors (and matrices) should be stored in contiguous locations, i.e., with unit vector stride. The vector instructions that access memory have a known pattern and if the elements of vectors (matrices) are all adjacent, then the maximum speed in data access is achieved by pipelining.

2. Data Locality:

   Given the slow speed of the main memory, the access to the main memory should be minimized. This implies that the intermediate data should be kept in the registers and cache memory. Also, once data is fetched from the main memory and loaded into the cache, all of the operations that require the data should be performed before the data is returned to the main memory, i.e., the vector touch should be minimized. Given the limited size of the cache, this may even require reordering the computation.

As a conclusion on the design of efficient vector algorithms, we would like to quote from [11, p. 47], *"We have shown that the efficiency of a vector- pipeline matrix computation depends upon the vector length, the vector stride, the vector touch, and the data re-use properties of the algorithm. Optimizing with respect to all these attributes is very complicated and something of an art. A good compiler can of course do some of the thinking for us, but do not count on it!"*

Note that in [11] general matrix computations are considered which are much simpler than a rather complex algorithm such as DARTS.

4. Development and Implementation of Vectorized / Parallel DARTS

Based on the analysis of Sections 2 and 3, we first developed a parallel/vectorized version of DARTS [3]. However, the practical implementation of this algorithm resulted in an *interactive vectorization* process. Detailed timing was used to measure the computation time of each subroutine and the overall computation. The data structure and operations of the algorithm were then constantly modified to minimize the computation time. As a result, the final implementation was different from the original algorithm in [3]. Two issues made these modifications necessary. First, the algorithm in [3] was based on general and theoretical assumptions regarding vector processing. Given the fact that this was our first experimentation, many lessons were learned on detailed practical issues through actual implementation. The second, and more important, issue was due to the shortcomings of both hardware and software of the SKYbolt. Some of the necessary routines either were not provided or were not functional. Also, no means was provided to control the cache memory (see also Section 5). As a result, we were forced to develop our own subroutines or to change the computation. Here, we discuss some of the implementation issues. Due to the lack of space, only a few representative examples are given.

A. SCATTER OPERATIONS: VELOCITY PROPAGATION

The propagation of velocities is a simple, but representative, example that shows how the topology of the spacecraft allows efficient global vectorization of computation, which follows m and n that stand for the number of bus flexible mode and the number of appendages. Here, the main computation is the evaluation of the deformation variables for all the appendages:

For $i = 1$ to n,

$$\delta_r(i) = \sum_{j=1}^{m} \lambda_{ij}\eta_j = \lambda(i)\eta \tag{3}$$

$$\delta_\ell(i) = \sum_{j=1}^{m} \gamma_{ij}\eta_j = \gamma(i)\eta \tag{4}$$

$$\delta_\omega(i) = \sum_{j=1}^{m} \lambda_{ij}\dot{\eta}_j = \lambda(i)\dot{\eta} \tag{5}$$

$$\delta_v(i) = \sum_{j=1}^{m} \gamma_{ij}\dot{\eta}_j = \gamma(i)\dot{\eta} \tag{6}$$

where $\eta \triangleq \text{col}\{\eta_j\}$ and $\dot{\eta} \triangleq \text{col}\{\dot{\eta}\}\varepsilon\Re^{m\times 1}$ are the vectors of modal deformation coordinates of the bus. λ_{ij} and $\gamma_{ij}\varepsilon\Re^{3\times 1}$ are the rotational and translational displacement vectors of the jth mode for the ith appendage, $\lambda(i) \triangleq \text{row}\{\lambda_{ij}\}$ and $\gamma(i) \triangleq \text{row}\{\gamma_{ij}\}\varepsilon\Re^{3\times m}$. $\delta_r(i), \delta_\ell(i), \delta_\omega(i)$, and $\delta_v(i)\varepsilon\Re^{3\times 1}$ are the translational and rotational deformation and the linear and angular deformation velocities of appendage i. Due to the star topology of the spacecraft, the propagation for all appendages can be done simultaneously, i.e., the computation in Eqs. (3)-(6) can be done in parallel for all $i = 1$ to n.

For serial computation, the two forms in Eqs. (3)-(6) have the same cost while the second form is more efficient for vector computation. This efficiency for vectorization can be further increased as follows. Define

$$\hat{\eta} = \{\eta \quad \dot{\eta}\}\varepsilon\Re^{m\times 2};\ \lambda \triangleq \text{col}\{\lambda(i)\} \text{ and } \gamma \triangleq \text{col}\{\gamma(i)\}\varepsilon\Re^{3n\times m};\ \hat{\Pi} \triangleq \begin{Bmatrix} \lambda \\ \gamma \end{Bmatrix} \varepsilon\Re^{6n\times m};$$

$$\delta_r \triangleq \text{col}\{\delta_r(i)\}, \delta_\ell = \text{col}\{\delta_\ell(i)\}, \delta_\omega = \text{col}\{\delta_\omega(i)\}, \text{ and } \delta_v = \text{col}\{\delta_v(i)\}\varepsilon\Re^{3n\times 1};$$

$$\delta_{r\ell} = \begin{pmatrix} \delta_r \\ \delta_\ell \end{pmatrix} \text{ and } \delta_{\omega v} = \begin{pmatrix} \delta_\omega \\ \delta_v \end{pmatrix} \varepsilon\Re^{6n\times 1};\ \text{and } \delta = \{\delta_{\omega v}\delta_{r\ell}\}\varepsilon\Re^{6n\times 2}.$$

The computation in Eqs. (3)-(6) can then be performed by a simple matrix-matrix multiplication as:

$$\delta = \hat{\Pi}\hat{\eta} \tag{7}$$

Note that the matrix $\hat{\Pi}$ is constant and can be precomputed. Also, the above computation results in a certain arrangement of the vectors $\delta_r(i), \delta_\ell(i), \delta_\omega(i)$, and $\delta_v(i)$, which affects the rest of the computation and should be taken into account. However, because of the structure of the matrix $\hat{\Pi}$ and the vector $\hat{\eta}$ is not efficient to use, the regular matrix-matrix multiplication routine is based on the vector-dot operation (see below).

B. SCATTER AND GATHER OPERATIONS: FORCE AND ACCELERATION PROPAGATION

Using similar technique as for the velocity propagation, the propagation of force and acceleration can be globally vectorized and represented in terms of large matrix-vector multiplications as [3]:

$$Z(B) = \begin{pmatrix} \Pi^* \\ \phi \end{pmatrix} Z_1^+ + K = (\Pi^*\phi)Z_1^+ + K \tag{8}$$

$$\alpha_1^+ = (\Pi\phi^*)\,\alpha(B) = (\Pi\phi^*)\alpha(B) \tag{9}$$

where $Z(B)$ and $\alpha(B)\varepsilon\Re^{(m+6)\times 1}$ are the vectors of residual force and acceleration of the bus. $Z_1^+(i)$ and $\alpha_1^+(i)\varepsilon\Re^{6\times 1}$ are the residual force and acceleration of appendage i, and $Z_1^+ \triangleq \text{col}\,\{Z_1^+(i)\}$ and $\alpha_1^+ \triangleq \text{col}\,\{\alpha_1^+(i)\}\varepsilon\Re^{6n\times 1}$. $\Pi\varepsilon\Re^{6n\times m}$ is an appropriate combination of $\lambda(i)$ and $\gamma(i)$ and can be precomputed. $\phi\varepsilon\Re^{6\times 6n}$ is a sparse matrix that needs to be formed in real time. In Eqs. (8)-(9), $\Pi^*\phi\varepsilon\Re^{(m+6)\times 6n}$ and $\Pi\phi^*\varepsilon\Re^{6n\times(m+6)}$.

The matrix-vector multiplication routine provided by the SKYbolts (and other vector supercomputers) is based on the vector-dot operation. Consider a matrix-vector multiplication as $V = MU$ where M is a $P \times Q$ matrix and let $M^{(i)}$ and $M_{(i)}$ denote the rows and columns of matrix M, respectively. The vector-dot based routine is given:

For $i = 1$ to P,

$$V(i) = M^{(i)} \cdot U \tag{10}$$

However, another possible algorithm for matrix-vector multiplication is based on the SAXPY (scalar-vector multiply plus vector) operation [11]:

For $i = 1$ to Q,

$$V^i = V^{i-1} + M_{(i)}U(i) \tag{11}$$

Both the vector-dot and SAXPY operations are highly suitable for vector computation. The operation in Eq. (10) requires P vector-dot operations on vectors of dimension Q while that in Eq. (11) requires Q SAXPY operations on vectors of dimension P. Based on our discussion in Section 3.B, it then follows that the $P < Q$, the vector-dot based routine, and the $P > Q$, the SAXPY based routine, are more efficient.

For our implementation, the values of n and m were $n = 13$ and $m = 10$. Thus, the vector-dot routine is highly optimal for matrix-vector multiplication in Eq. (8) because it requires 16 vector-dot operations on vectors of dimension 78. However, it is highly inefficient for Eq. (9) as it requires 78 vector-dot operations on vectors of dimension 16. If the SAXPY based routine is used for Eq. (9), then it requires only 16 SAXPY operations on the vectors of dimension 78.

The C language was used for the development of our vectorized code, which implied that the matrices are stored by rows. However, for efficient implementation of the SAXPY based routine, matrices need to be stored and fetched by columns. For Eq. (9), the need for transposing the matrix $\Pi\phi^*$ can be simply eliminated by rewriting it as:

$$(\alpha_1^+)^* = (\alpha(B))^*(\Pi\phi^*)^* = (\alpha(B))^*(\Pi^*\phi) \tag{12}$$

Another advantage of Eq. (12) is that for both Eqs. (8) and (12), only the matrix $\Pi^*\phi$ needs to be formed.

Our SAXPY based routine, though developed in C language, significantly increased the computational efficiency and was used very frequently. Obviously, if this routine is provided by the vendors and developed in assembly language, it can offer an even greater computational efficiency. Note that, for the matrix-matrix multiplication in Eq. (7), we also used a SAXPY based matrix-matrix multiplication routine that is more efficient than the vector-dot based routine.

C. COMPUTATION OF ARTICULATED-BODY INERTIA AND ACCELERATION OF BUS

The computation of the articulated-body inertia, $P(B)\varepsilon\Re^{(m+6)\times(m+6)}$, and acceleration, $\alpha(B)$, of the bus represents the major computation-intensive parts of the vectorized algorithm (over 30% of the total computation time). As stated before, the computation of $P(B)$ represents a gather operation and was globally vectorized in a similar fashion as the computation of $Z(B)$. However, significant reduction (more than 50%) in computation time was achieved by several changes in the data structure and the type of operations to find the most optimal way for computation of $P(B)$. In the final form, the symmetry of $P(B)$ was exploited and only the diagonal and lower triangular parts of $P(B)$ were computed. $\alpha(B)$ is computed as the solution of the system.

$$P(B)\alpha(B) = \varepsilon(B) \tag{13}$$

We first used a Cholesky-based routine provided by the SKYbolt's library for the solution of the symmetric, positive definite, system in Eq. (13). Later, we developed a routine based on the LDL^* decomposition [11] that did not require square-root operation. Although our routine was developed in C and was not vectorized, it was significantly faster than the SKYbolt's routine. However, the main motivation for and the advantage of this routine was that, given the way the matrix $P(B)$ was computed, it could easily be used for solution of Eq. (13) without any need for data movement. Again, if this routine is developed by the vendor in assembly language and in fully vectorized form, it can offer an even greater computational efficiency.

D. DATA MOVEMENT MINIMIZATION

Major improvement in the efficiency of the vectorized algorithm was achieved by minimizing the data movement overhead through modification of the data structure and operations of the algorithm. Here, a few examples are discussed.

1. Matrix-Matrix Multiplication

The computation of vectorized DARTS involves many matrix-matrix multiplications as $A = BC$ for both small and very large matrices. A vector-dot based matrix-matrix multiplication routine requires that first the matrix C be transposed. However, if matrix C is symmetric, then it does not need to be transposed. The SKYbolts provided two matrix-matrix multiplication routines for the general (nonsymmetric matrix) case and the special case (symmetric matrix). However, even for the general case, whenever possible we eliminated the need to transpose the matrix C by either forming C^* (if it could be precomputed) or directly computing C^*.

Another frequently used operation was chained matrix-matrix multiplication, as $A = BCB^*$, for both small and very large matrices and with C being a symmetric matrix. For example, this type of matrix-matrix multiplication occurs in projection of mass

matrices of the appendages onto the bus frame or in computation of $P(B)$. This operation can be performed without any need for matrix transposition by simply rewriting it as $A = B(BC)^*$. This simple modification resulted in a significant reduction of the data movement overhead particularly for computation of $P(B)$ wherein the matrices involved in the computation were very large.

2. Vector Touch Minimization

One of the widely used operations in the vector processing is a GAXPY (matrix-vector multiply plus vector) operation [11] as $V_1 = MV_2 + V_3$ wherein M is a matrix and V_1, V_2, and V_3 are vectors. In addition to the computational efficiency, a GAXPY routine reduces the vector touch since the vector $V_4 = MV_2$ does not need to be explicitly computed, stored, and reloaded. However, the SKYbolt's library did not provide such a routine and we had to develop our own routine. Several other routines were also developed for other operations with the purpose of minimizing the vector touch.

3. Data Structure Modification

Our major effort in reducing the data movement overhead was based on modifying the data structure of the algorithm to find the most optimal form. Here, we give a simple example that underlines the importance of the data movement overhead minimization. The computation times of evaluating angular, a_{wi}, and linear, a_{vi}, gyroscopic acceleration of appendages for $i = 1$ to n, were measured as 137 μs and 121 μs. For the rest of the computation, it was then required to merge the vectors a_{wi} and a_{vi} and form a vector $a_i = \begin{Bmatrix} a_{wi} \\ a_{vi} \end{Bmatrix}$. However, it took 143 μs to form the vectors a_i for $i = 1$ to n which was greater than the computation time for either a_{wi} or a_{vi}. The algorithm was then modified to directly compute and form the vectors a_i without any data movement. This simple example clearly shows that for vector processing the data movement time can be even greater than the computation time.

E. GLOBAL VECTORIZATION OF SMALL MATRIX-VECTOR OPERATION LOOPS

A rather significant part of the DARTS algorithm, which seemed to be unvectorizable, involved many Do-loops with small vectors and matrices. An example of such frequently occurring Do-loops is:

For $i = 1$ to n,

$$V_{1i} = M_i V_{2i} + V_{3i} \tag{14}$$

where V_{1i}, V_{2i}, and V_{3i} are 3×1 vectors and M_i is 3×3 matrix. Due to the small dimension of vectors and matrix, it is more efficient to use scalar (serial) routines for such Do-loops. However, we developed a technique for global vectorization of such Do-loops.

Algorithm	Computation Time (in ms)	Speedup
Serial DARTS on SUN SPARC II	24.39 ms	-- (Reference Time)
Serial DARTS on 1 SKY Bolts	12.37 ms	2 Faster Hardware
Vectorized DARTS on one SKY Bolt	7.29 ms	1.7 Vectorization
Parallel/Vectorized DARTS on two SKY Bolts	4.82 ms	1.5 Parallelization & Vectorization

Table I. Comparison of different algorithms/architecture computational efficiency

To see this, let us define

$$V_1 \triangleq \text{col}\,\{V_{1i}\}, V_2 \triangleq \text{col}\,\{V_{2i}\}, V_3 \triangleq \text{col}\,\{V_{3i}\} \varepsilon \Re^{3n\times 1}, \text{ and } M \triangleq \text{diag}\,\{M_i\} \varepsilon \Re^{3n\times 3n}$$

The above loop can then be replaced by a single matrix-vector multiplication:

$$V_1 = MV_2 + V_3 \tag{15}$$

Of course, due to the sparse structure of matrix M, it is highly inefficient to compute Eq. (15) by performing a general matrix-vector multiplication. However, the matrix M is a banded matrix with the nonzero elements only on its five leading diagonals. The computation in Eq. (15) can be efficiently done by performing the matrix-vector multiplication by diagonals. To see this, let M^j, $j = -2$ to 2, denote the diagonals of matrix M, where M^0 is the main diagonal.

The computation in Eq. (15) can then be done:

Set $V_1^{-3} = V_3$

For $j = -2$ to 2,

$$V_1^j = M^j \odot V_2 + V_1^{j-1} \tag{16}$$

where $\odot$ indicates element-by-element multiplication of two vectors. The element-by-element vector multiplication plus vector operation in Eq. (16) is highly efficient for vector computation and it is also provided by the SKYbolt's library. The efficiency of this technique for global vectorization results from the fact that it involves five such vector operations on large vectors. However, we did not implement this technique since the routine provided by the SKYbolt's library was not functional for double precision. Furthermore, its efficient implementation requires the minimization of the data movement overhead which may occur in forming the diagonals of matrix M. This requires a direct control of cache memory, which was not possible on the SKYbolt. Nevertheless, for future applications, this technique is very promising as it allows the seemingly strictly serial Do-loops to be vectorized.

5. Discussion and Conclusion

Table I shows a comparison of the different implementations of DARTS for a 13-body and 10-flexible modes model of Cassini spacecraft. As stated before, the speedup of the vectorized algorithm increases with the increase in the number of flexible modes and/or the number of bodies. For a 13-body and 20-flexible modes model, the vectorized algorithm achieves a speedup of 2 over serial DARTS on one SKYbolt. We did not discuss the algorithm's parallelization in detail. Suffice it to mention that, despite using several strategies to overlap the computation and communication as much as possible, the communication overhead from the slow VME bus remained a major bottleneck, which explains the rather poor speedup of parallelization.

Here, we would like to summarize some of the shortcomings of the SKYbolt and to discuss some desired features.

A. DOUBLE-PRECISION PERFORMANCE

The i860 is claimed to be a 64-bit vector processor [4]. However, it has a 128-bit wide data path, which means that only two double-precision operands can be simultaneously loaded from or stored to the cache memory. This significantly reduces the speed of the processor for those vector operations that involve three operands. We implemented our vectorized algorithm with double precision. Although, we did not try the single-precision implementation of the algorithm, given the high vectorization degree of our algorithm, a much greater speedup can be expected for single-precision implementation.

B. LIMITED CACHE MEMORY AND LACK OF CACHE MANAGEMENT

The SKYbolt did not provide a means for managing the cache memory. Thus, we could not further reduce overhead caused by the data movement between the cache and main memory by explicitly defining the physical location of data in the cache memory and, hence, increasing the data re-use. As a result, most of the computation was performed on the data located in the main memory which, in addition to increasing the overhead, significantly reduced the computation speed of both scalar and vector operations. Given

this extensive use of the main memory, a DRAM main memory with a slower speed over an SRAM main memory, would have increased the overhead by a factor of 2.

For double precision, the size of i860 cache is 1 K. However, the vectorized algorithm involved the operations on matrices larger than the size of the cache. For example, the matrix $\Pi^*\phi$ in Eqs. (8) and (12) is of dimension 16 $\times$ 78 and the computation of $P(B)$ involves even bigger matrices. An efficient technique for handling such cases is the segmentation of the computation [10]. However, this requires a direct control of the cache memory which, as stated before, was not possible.

C. SKYBOLT'S LIBRARY

As stated before, the SKYbolt's library did not provide some of the useful routines that were frequently used in our implementation. Also, some of the routines provided were not functional either at all or for double precision.

Despite all the above shortcomings, the SKYbolt was highly cost effective and allowed us to meet our goal (see Table I) with a relatively short development time. As stated before, the SKYbolt represents the first generation of low-cost vector processors. The new generations not only provide a drastic reduction in the cost over performance ratio, but also significant improvements in both hardware and software. The size of cache memory in the new versions of i860 has increased by a factor of 2. Single board multiple i860 processor-based architectures [13] are now offered that present a much more balanced system for parallel computation since the communication between processors can be performed on board and via a fast interconnection network. The library routines are also improved. In particular, based on our suggestions to the vendors, new routines including some of the routines developed by us, e.g., the SAXPY-based matrix-vector and matrix-matrix multiplication routines and LDL^* routine for linear system solution,were added to the library.

The results of our work, along with the significant improvements in both the price and performance of these architectures, clearly suggest that the parallel/vector algorithms and architectures present a highly efficient and low-cost approach for achieving real-time simulation capability for even more complex and computationally demanding multibody systems, such as the Space Station. In particular, it should be mentioned that the Space Station has a star topology that allows the application of a similar global vectorization strategy as for Cassini spacecraft. Also, due to the flexibility of Space Station appendages, not only the computation for appendages can be vectorized but also, based on our analysis in Section 2, more vector processors can be used to increase the speedup to the parallelization.

Acknowledgments

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. The authors gratefully acknowledge the assistance of R. Graves, K. Pendergast, and J. Hernandez in implementing the hardware and software discussed in Part II of this paper.

References

1. A. Jain and G. Rodriguez,"Recursive Flexible Multibody System Dynamics using Spatial Operators," *J. Guidance, Control, and Dynamics*, Nov. 1992. In Press.

2. A. Fijany and J.A. Roberts,"Dedicated Computer Architecture for Real-Time Dynamic Simulation of CRAF/Cassini Spacecraft: Requirements and Specifications," JPL Engineering Memorandum EM 343-91-236 (internal document), Jet Propulsion Laboratory, Pasadena, California, March 1991.

3. A. Fijany,"Parallel/Vectorized Algorithms for Real-Time Dynamic Simulation of CRAF/Cassini Spacecraft," JPL Engineering Memorandum EM 343-91-1230 (internal document), Jet Propulsion Laboratory, Pasadena, California, June 1991.

4. L. Kohn and N. Margulis,"The i860 64-Bit Supercomputing Microprocessor," *Proc. Supercomputing 89*, pp. 450-546, 1989.

5. H. Cheng and K.C. Gupta,"Vectorization of Robot Dynamics on a Pipelined Vector Processor," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 96-101, April 1991.

6. S. McMillan, D.E. Orin, and P. Sadayappan,"Real-Time Robot Dynamic Simulation on a Vector/Parallel Supercomputer," *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 1836-1841, April 1991.

7. B.L. Buzbee,"A Strategy for Vectorization," *Parallel Computing*, Vol. 3, pp. 187-192, 1986.

8. K. Hwang and F.A. Briggs, *Computer Architecture and Parallel Processing*, McGraw-Hill Inc., New York, New York, 1984.

9. K. Hwang,"Advanced Parallel Processing with Supercomputer Architectures," *Proc. of IEEE*, Vol. 75(10), pp. 1348-1379, 1987.

10. K. Hwang and S-P Su,"Vector Computer Architecture and Processing Techniques," *Advances in Computers*, Vol. 20, pp. 115-197, 1981.

11. G.H. Golub and C.F. Van Loan, *Matrix Computation*, second edition, Johns Hopkins University Press, Baltimore, Maryland, 1989.

12. D.E. Rosental and M.A. Sherman,"High Performance Multibody Simulation via Sym-

bolic Equation Manipulation and Kane's Method," *J. Astro. Sciences*, Vol. 34(3), pp. 223-239, 1986.

13. *SKY Computer Product Summary*, Sky Computers Inc,. Chelmsford, Massachusetts, April 1992.

USE OF HARDWARE-IN-THE-LOOP SIMULATION FOR SPACECRAFT MISSION PLANNING/PREPARATION/SUPPORT

L. SLAFER

FIFTH ANNUAL
NASA/NSF/DOD WORKSHOP ON
AEROSPACE COMPUTATIONAL CONTROL

SANTA BARBARA, AUGUST 1992

HUGHES

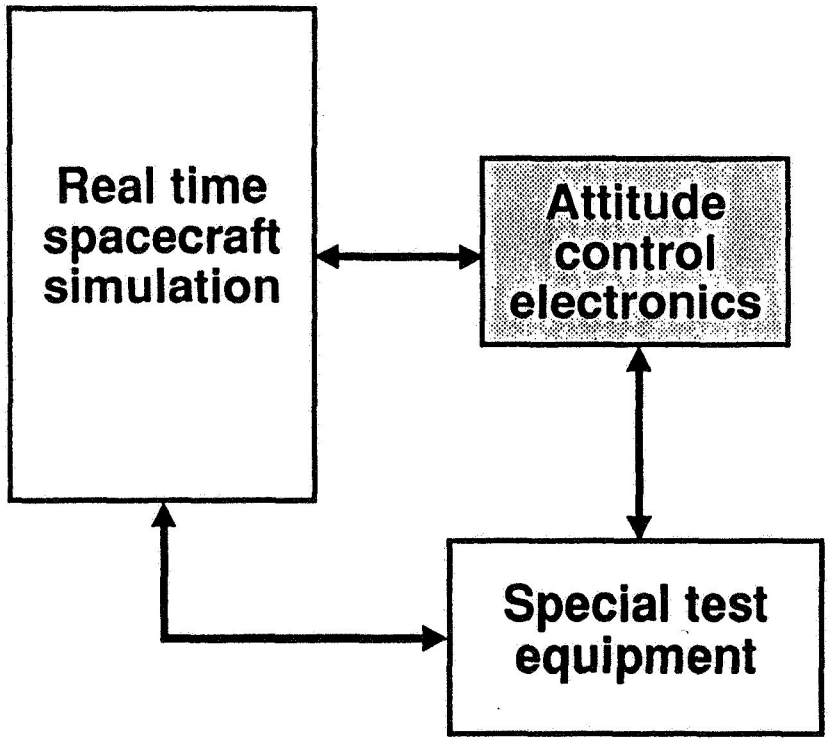
Overview

HUGHES

- 'Mixed simulation' testing at Hughes SCG
- 'Ground station MST' concept
- Ground station MST implementation - 'the whole is greater than the sum of the parts'
  - System 100
  - Dynamic satellite simulator
  - Ground station H/W and S/W
- Ground station MST current/future applications
- HS 601 mission support system facility
- Concluding remarks

Mixed Simulation Testing Is Key to ACS Development/Validation

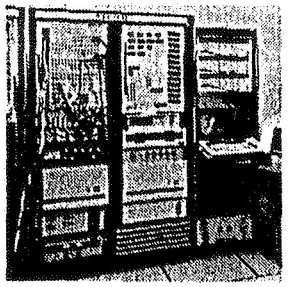
HUGHES



- MST combines breadboard and/or flight control electronics with detailed spacecraft, sensor, actuator simulations
- MST is real time, simulated on-orbit environment for
  - ACS design and performance validation
  - Flight S/W development and qual testing
  - ACE qual/acceptance testing

Spacecraft Control System MST Creates Simulated On-orbit Environment

HUGHES

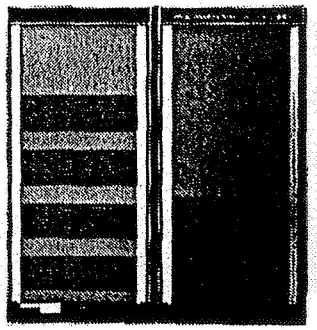


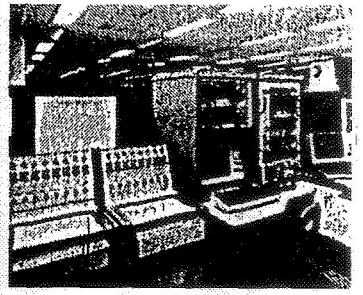
AD100 real time simulation system

Spacecraft dynamics model

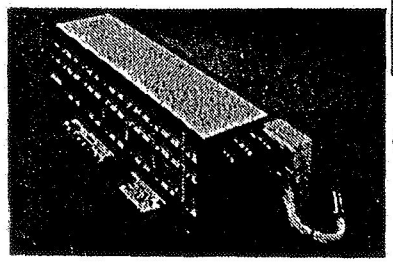
Input

Output





Control electronics



Telemetry And Control Signals

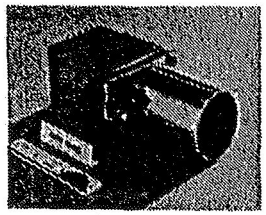
Telemetry And Control Signals

Special purpose interface unit

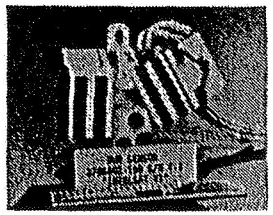
Sensor simulator

VAX computer system

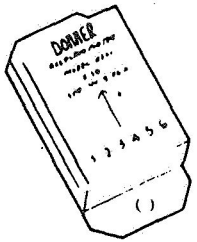
Simulation control



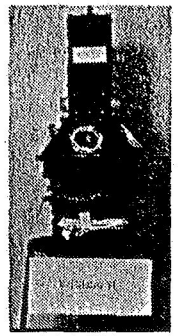
Earth sensor



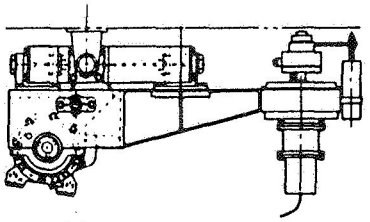
Sun sensor



Nutation accelerometer

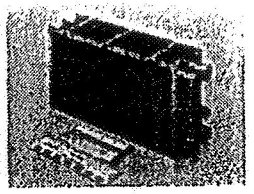


BAPTA



Reflector deployment and positioner mechanism

Attitude control thrusters



Valve driver unit

MST Activities at Hughes SCG

HUGHES

MST at Hughes has evolved over 25 yrs to become essential element of spacecraft development

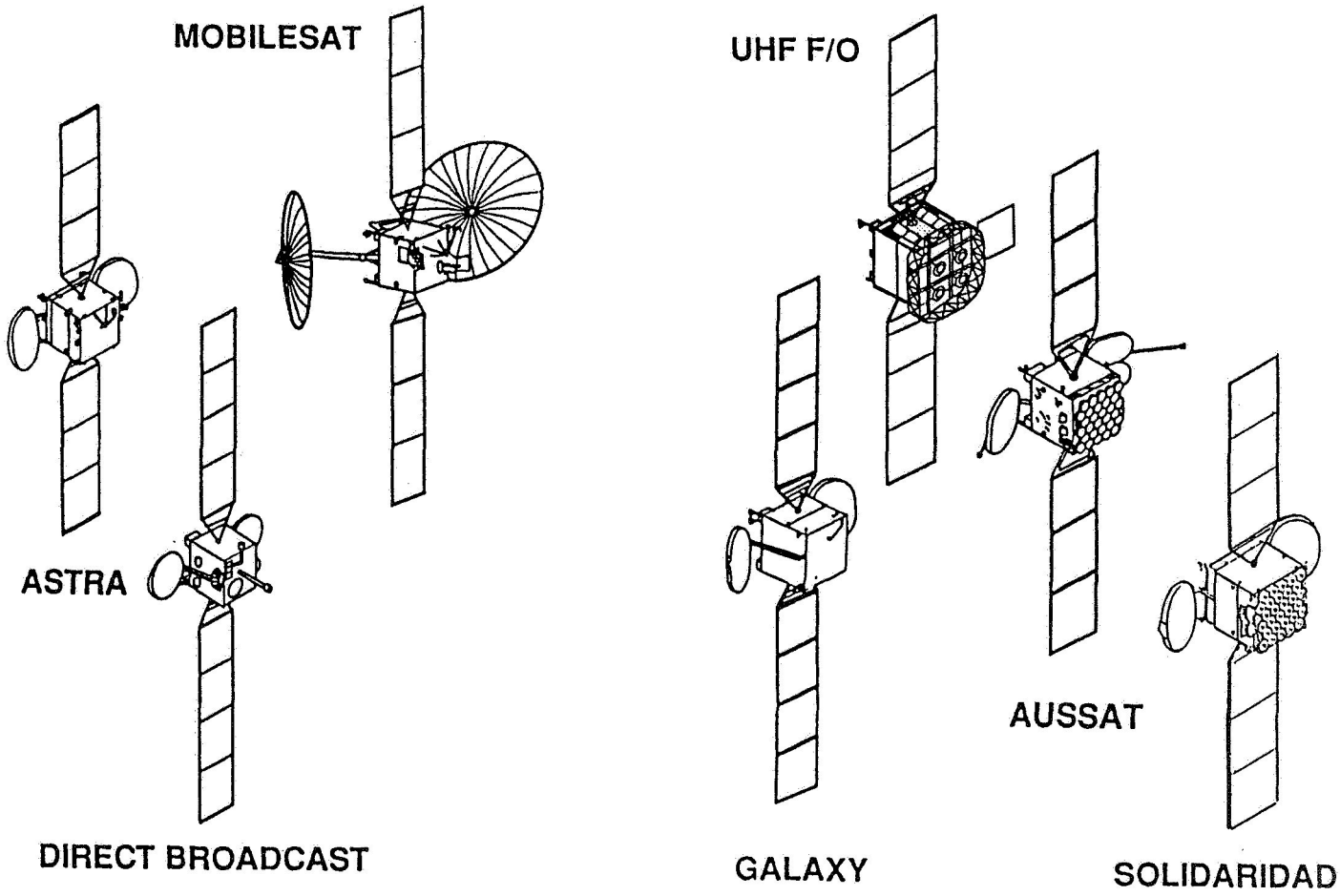
MST used in many areas

- ACS performance predictions/verification
- Developing insight into on-orbit response characteristics
- Hardware and software design verification/validation
- Unit and spacecraft testing
- Mission planning
- On-orbit anomaly investigations

In many instances, MST has been key tool in understanding and solving problems on ground and in-orbit

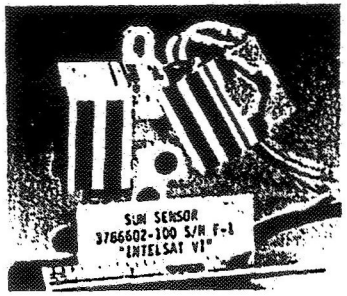
HS601 FAMILY OF SPACECRAFT REPRESENTS VERY COMPLEX NEW SYSTEM FOR HUGHES

HUGHES

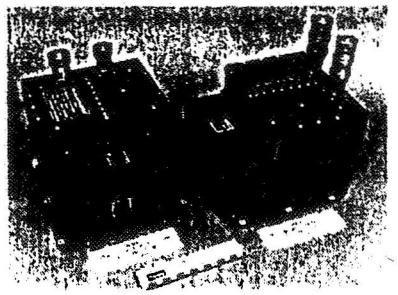


HS 601 Attitude Control Subsystem Components

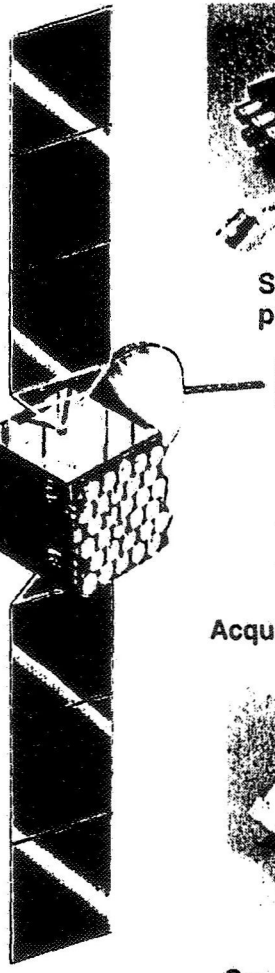
HUGHES

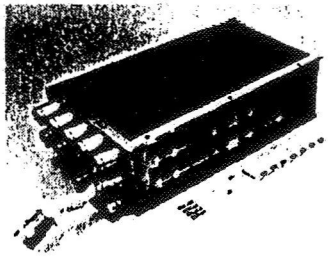


Transfer orbit sun sensors (2)

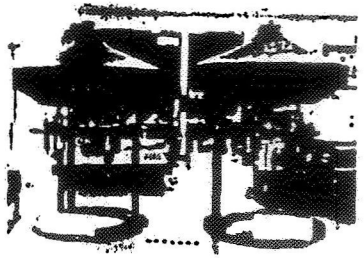


Inertial reference unit (2)

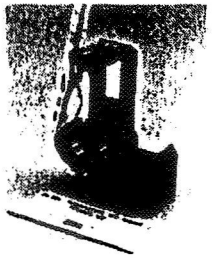




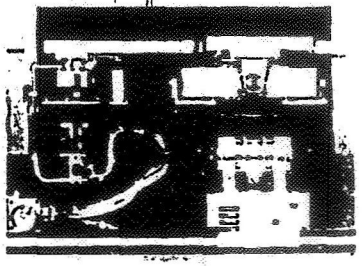
Spacecraft control processor (2)



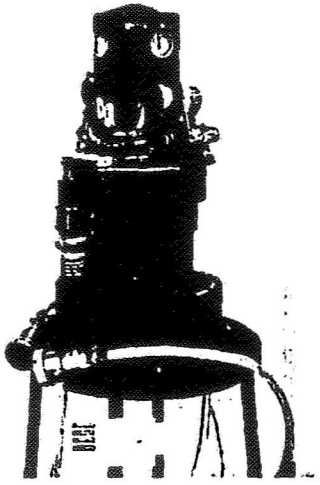
Momentum wheel and wheel platform



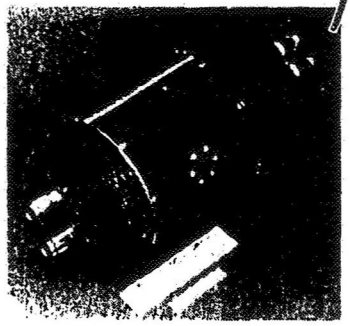
Acquisition sun sensors (2)



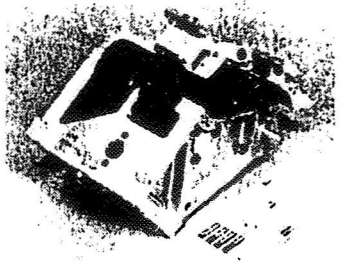
Reflector positioner and deployment activator



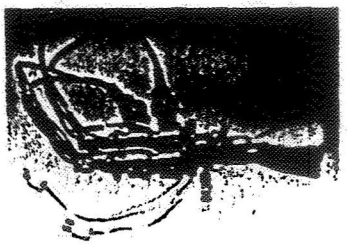
Solar wing drive (2)



Earth sensor assembly



Omni deployment actuator



Attitude control thrusters

M921561-106

HUGHES PRODUCTS FOR OUR SATELLITE CUSTOMERS

HUGHES

- ***REAL-TIME GROUND STATION SYSTEM***
  - RECEIVE/PROCESS/DISPLAY SPACECRAFT TELEMETRY
  - FORMAT COMMANDS TO THE SPACECRAFT
  - PROVIDE OPERATOR INTERFACE TO SPACECRAFT
  - SOFTWARE TO MONITOR HEALTH AND STATUS OF SPACECRAFT (DATA PLOTTING, CONFIGURATION BLOCK DIAGRAMS, LONG-TERM TREND ANALYSIS)
  - AUTOMATIC HEALTH/STATUS MONITORING (ALARMS TO OPERATORS)

- ***DYNAMIC SATELLITE SIMULATOR***
  - REAL-TIME COMPUTER SIMULATION OF SATELLITE BUS AND PAYLOAD
  - FUNCTIONAL REPRESENTATION OF ALL ELEMENTS
  - FORMATS/TRANSMITS TELEMETRY FROM SIMULATOR
  - RECEIVES REAL-TIME COMMANDS
  - PROVIDES TLM AND CMD INTERFACE TO GROUND STATION COMPUTER SYSTEM
  - OPERATOR TRAINING AID

Ground Station MST Heritage

HUGHES

Ground station MST developed with recognition by ACS, systems engineering, and mission operations senior staff that method of validating critical HS 601 operating procedures and mission software did not exist ⇒

- Significantly increased system and operational complexity (over HS 376) required high fidelity hardware-in-the-loop simulator
- Multiple HS 601 programs with different system designs required low cost techniques to perform operations design and validation
- High fidelity simulator essential as training/rehearsal tool

Experience with first program (Aussat) led to utilization on other HS 601 programs (UHF, Galaxy, MSAT...)

Ground Station Mixed Simulation Testing for HS 601

HUGHES

Ground station MST integrates

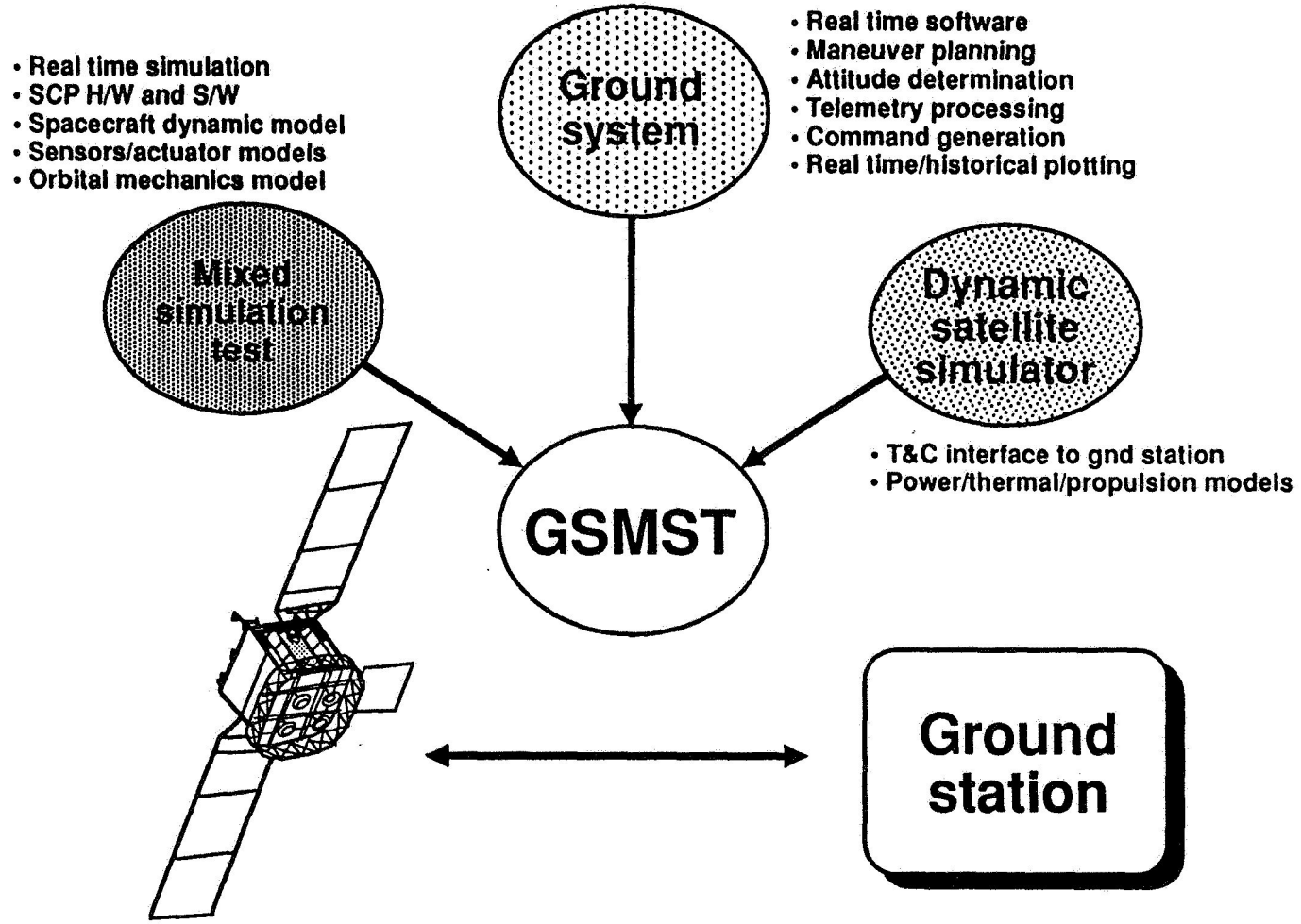
- ACS MST (with breadboard or flight control electronics)
- Core of spacecraft dynamic satellite simulator (DSS)
- Ground station real time software (tlm and cmd)
- Ground station command generation S/W
- Ground station telemetry processing S/W
- Ground station attitude determination and mission planning software

To create complete, high fidelity, real time simulated on-orbit environment to

'Fly the spacecraft on the ground'

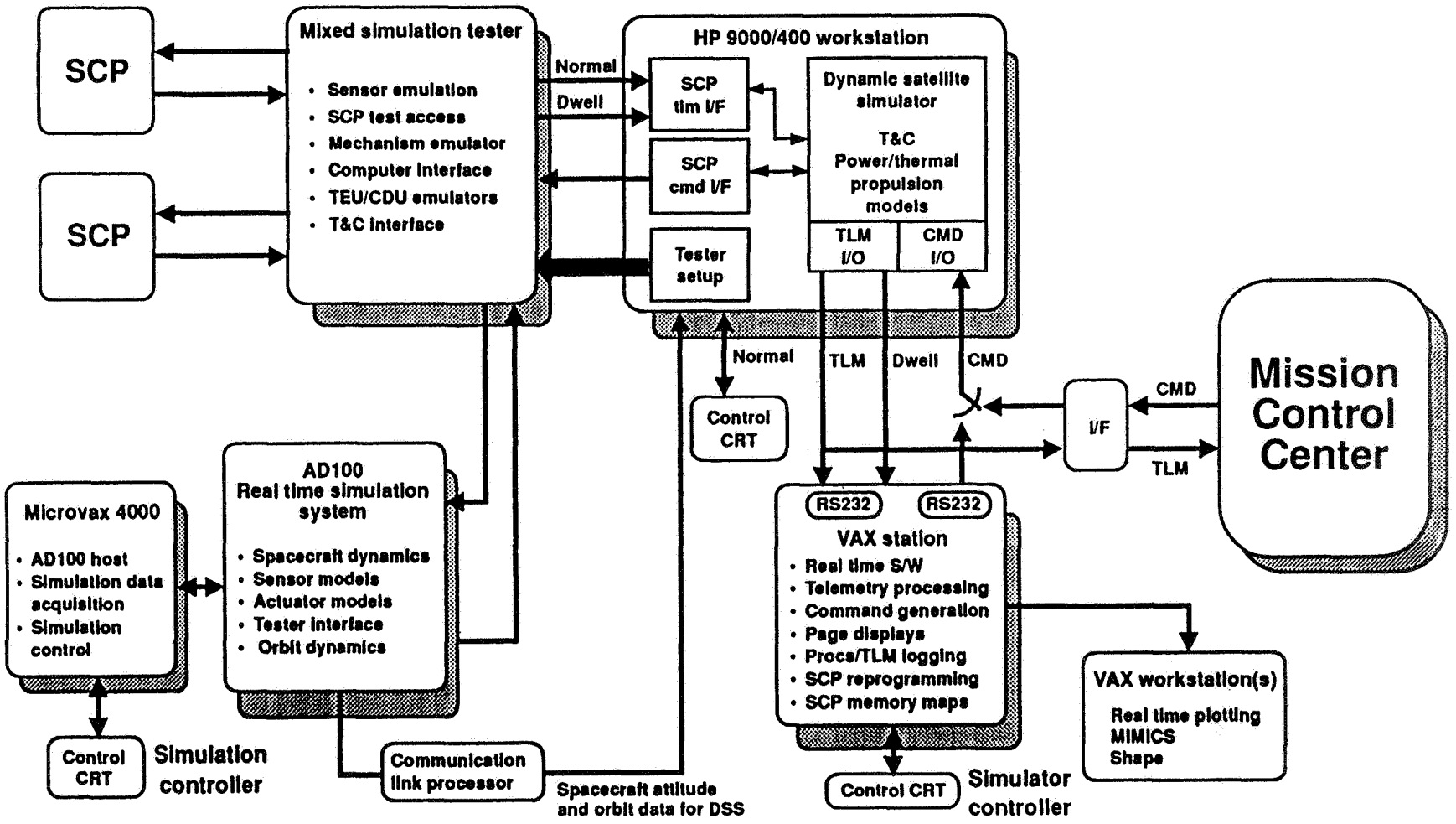
GSMST Integrates 3 Existing Elements

HUGHES



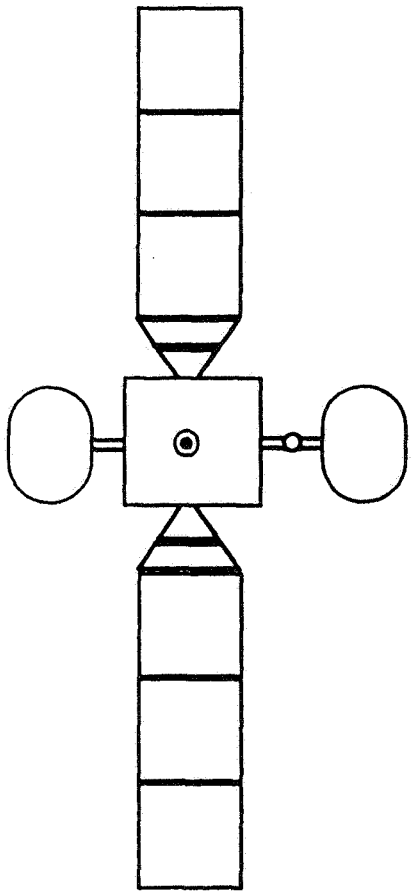
GSMST Configuration

HUGHES



AD100 Simulation Features

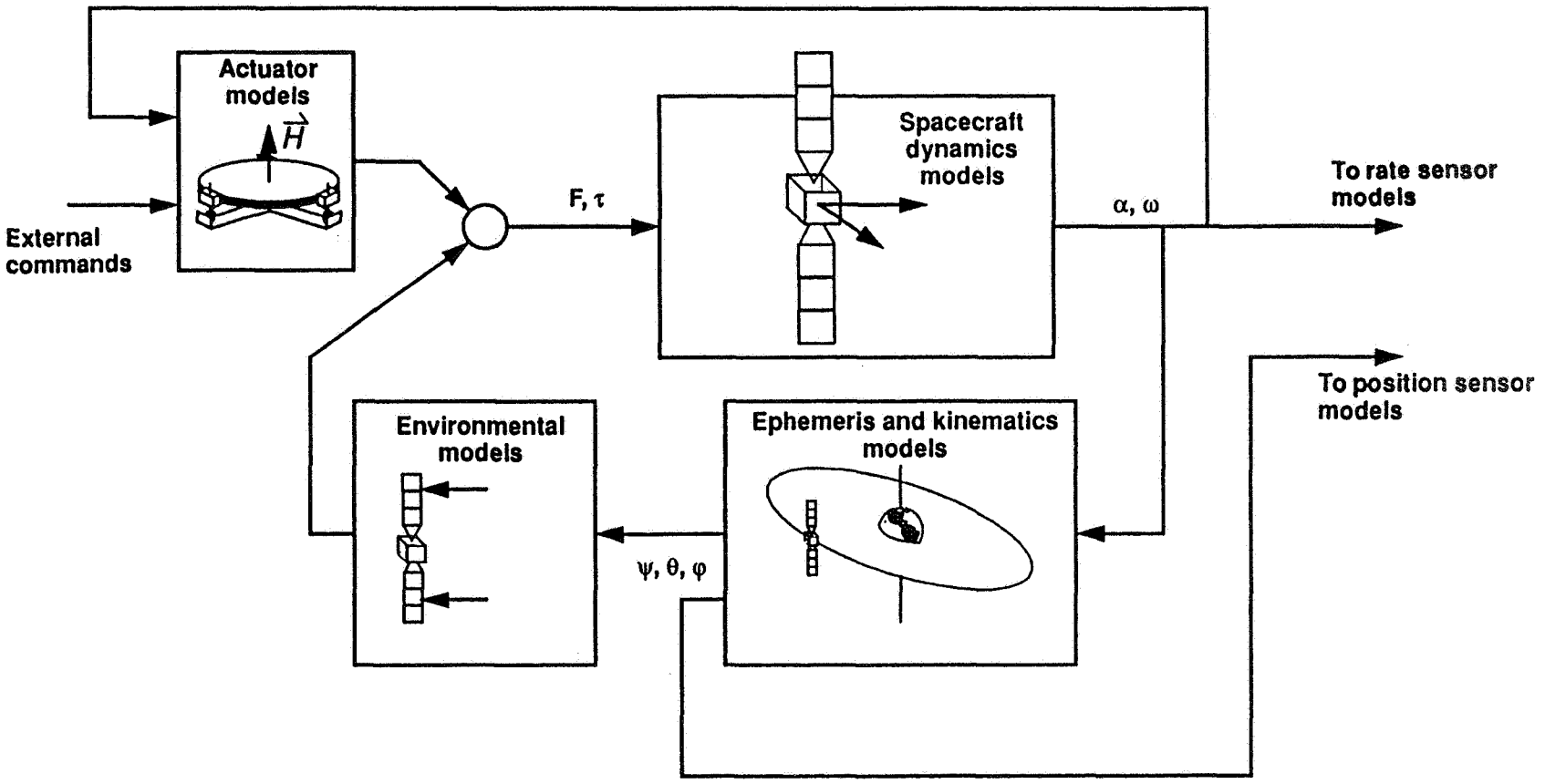
HUGHES



- Rigid dynamics with 16 DOF
- Flexible modeling with 12 modes/wing, 4 modes/reflector
- Large angle kinematics modeling
- Selectable orbits (low earth to geosync) and ephemeris models
- Selectable mass property and appendage configurations
- Solar and magnetic environmental torque models
- On-orbit maneuver fuel slosh modeling
- Eclipse thermal shock modeling
- High fidelity actuator and sensor modeling
- Selectable anomaly/failure mode modeling
- I/O interface to command and telemetry subsystem

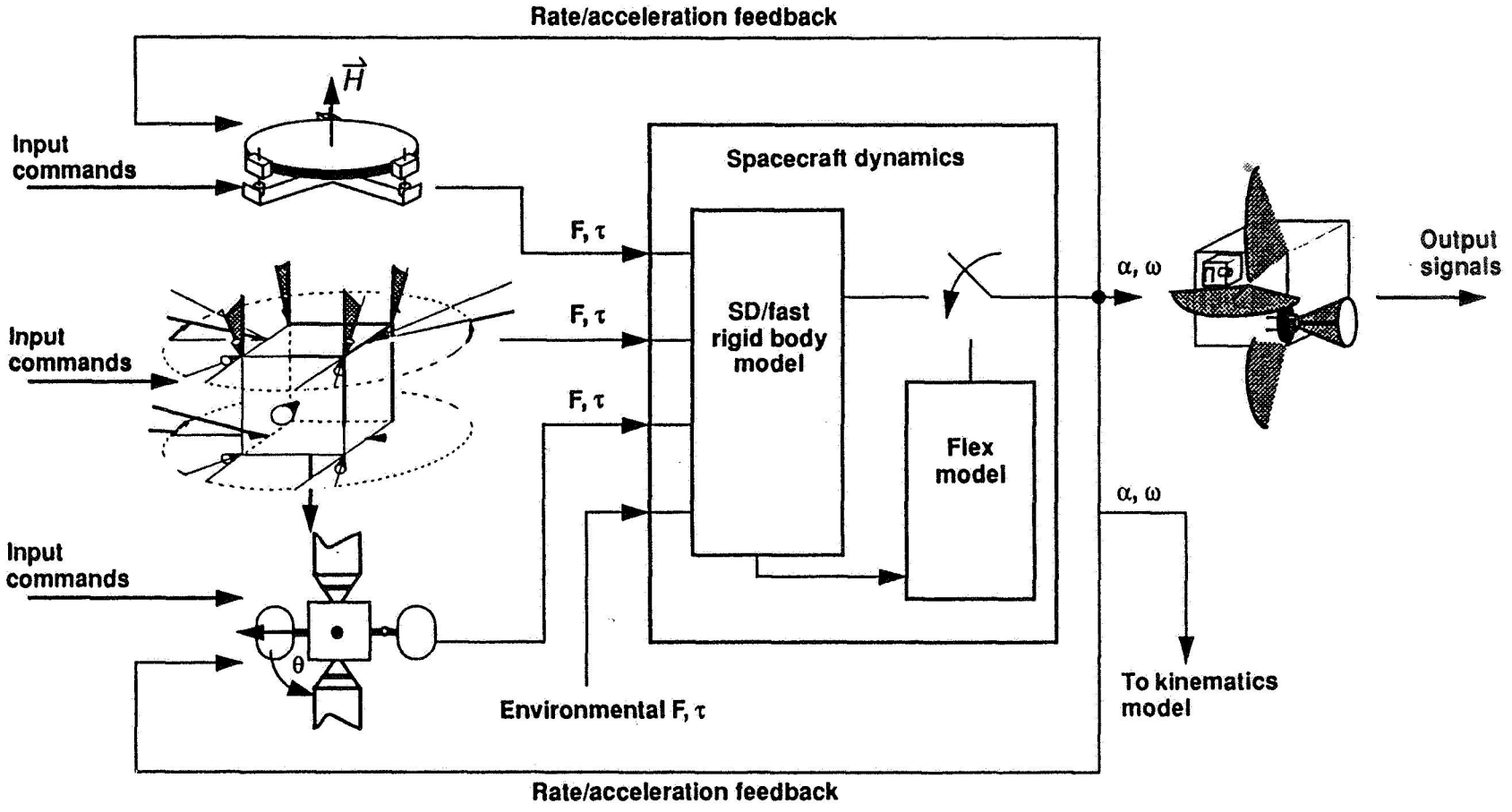
HS 601 AD100 Model Elements

HUGHES



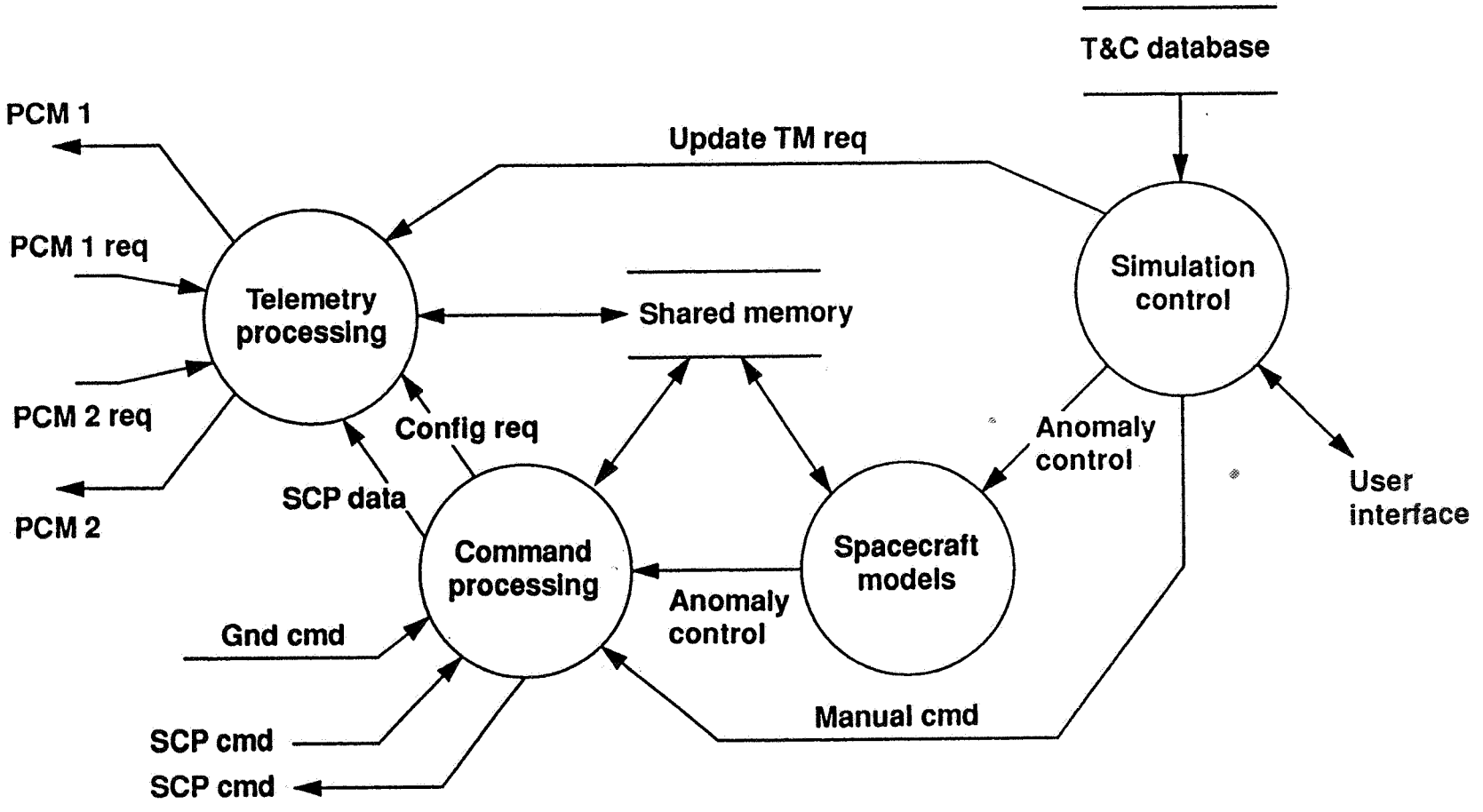
Dynamics Modeling

HUGHES



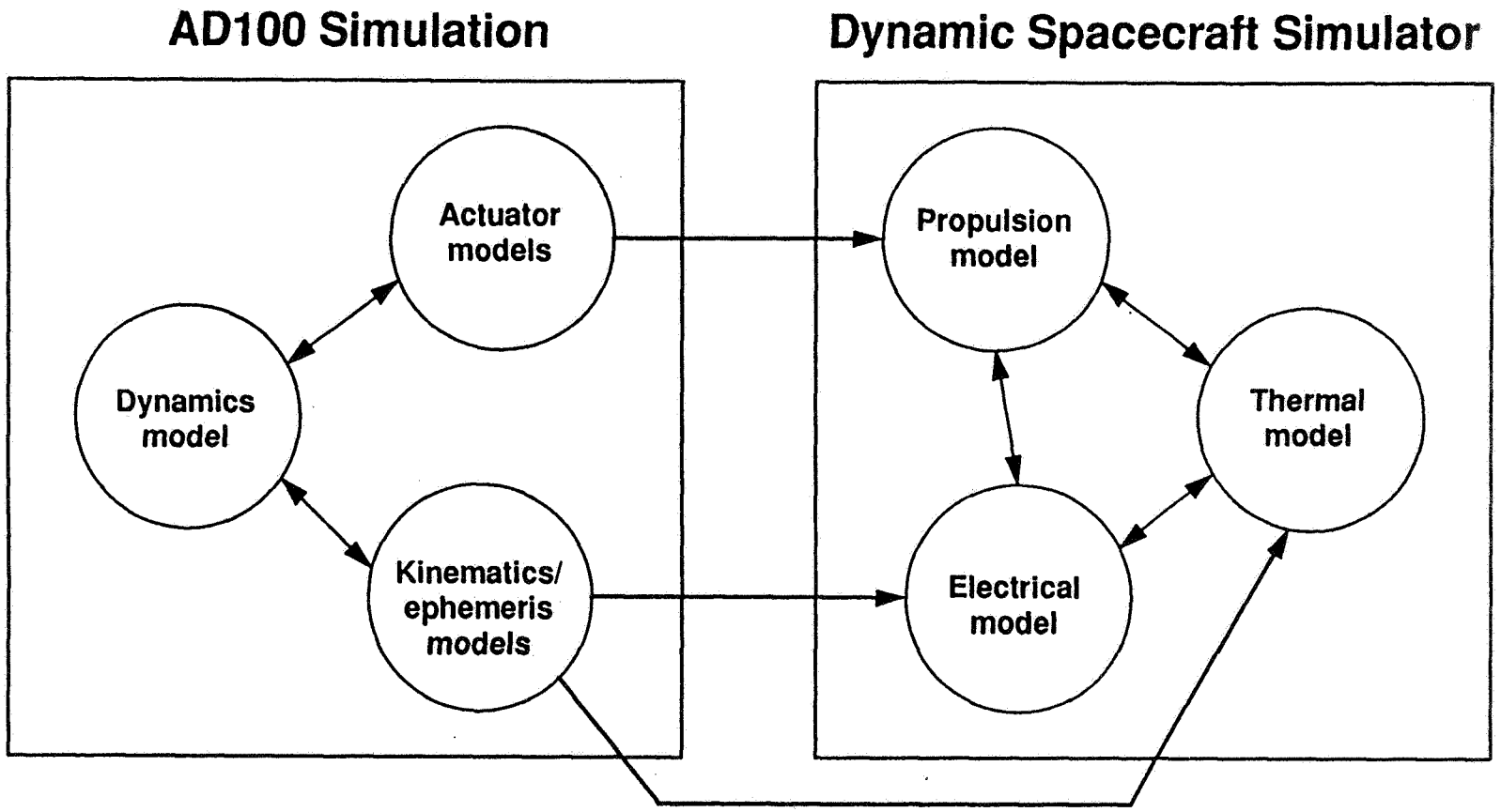
Dynamic Spacecraft Simulator

HUGHES



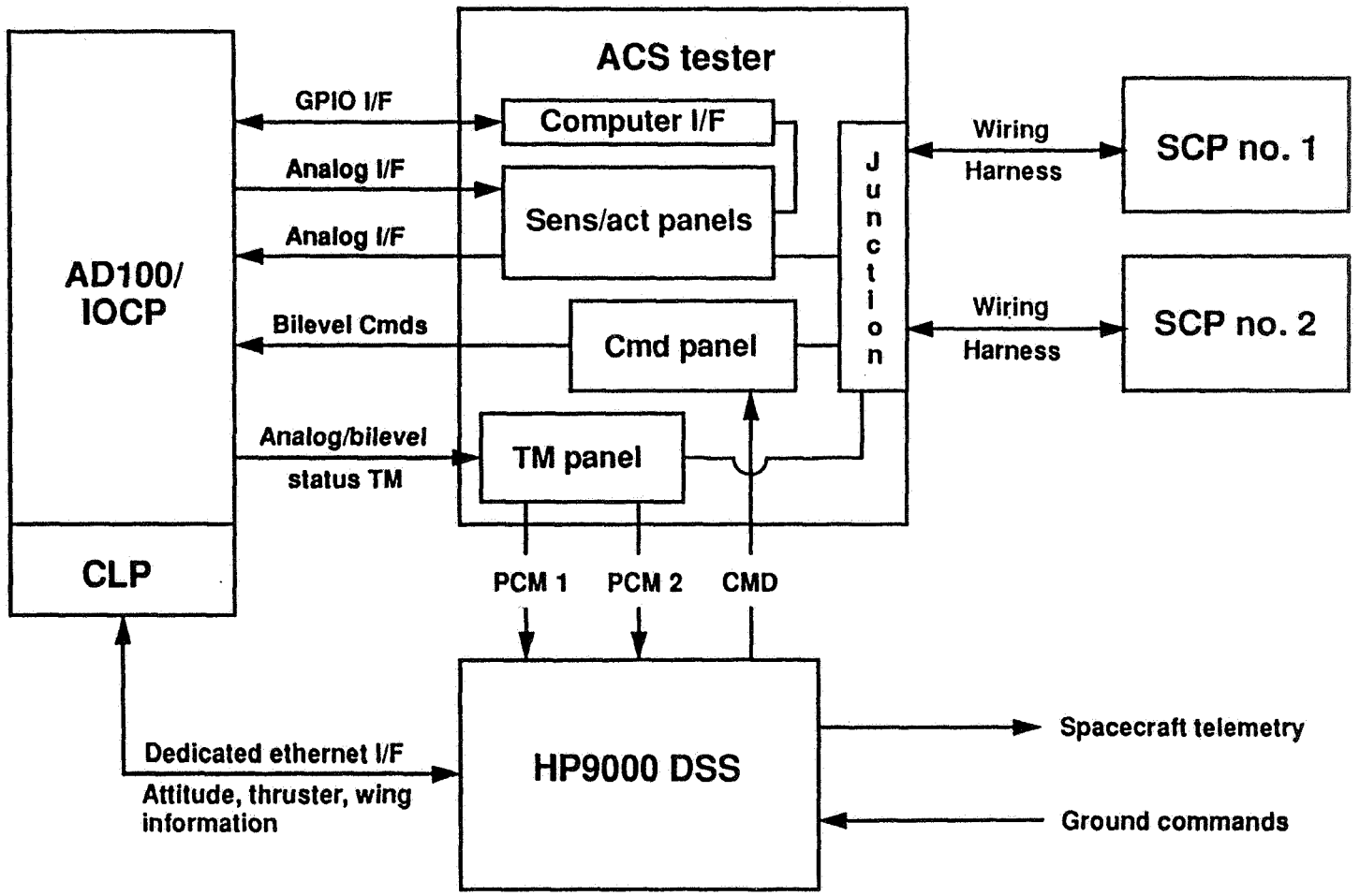
AD100 Simulation - DSS Information Flow

HUGHES



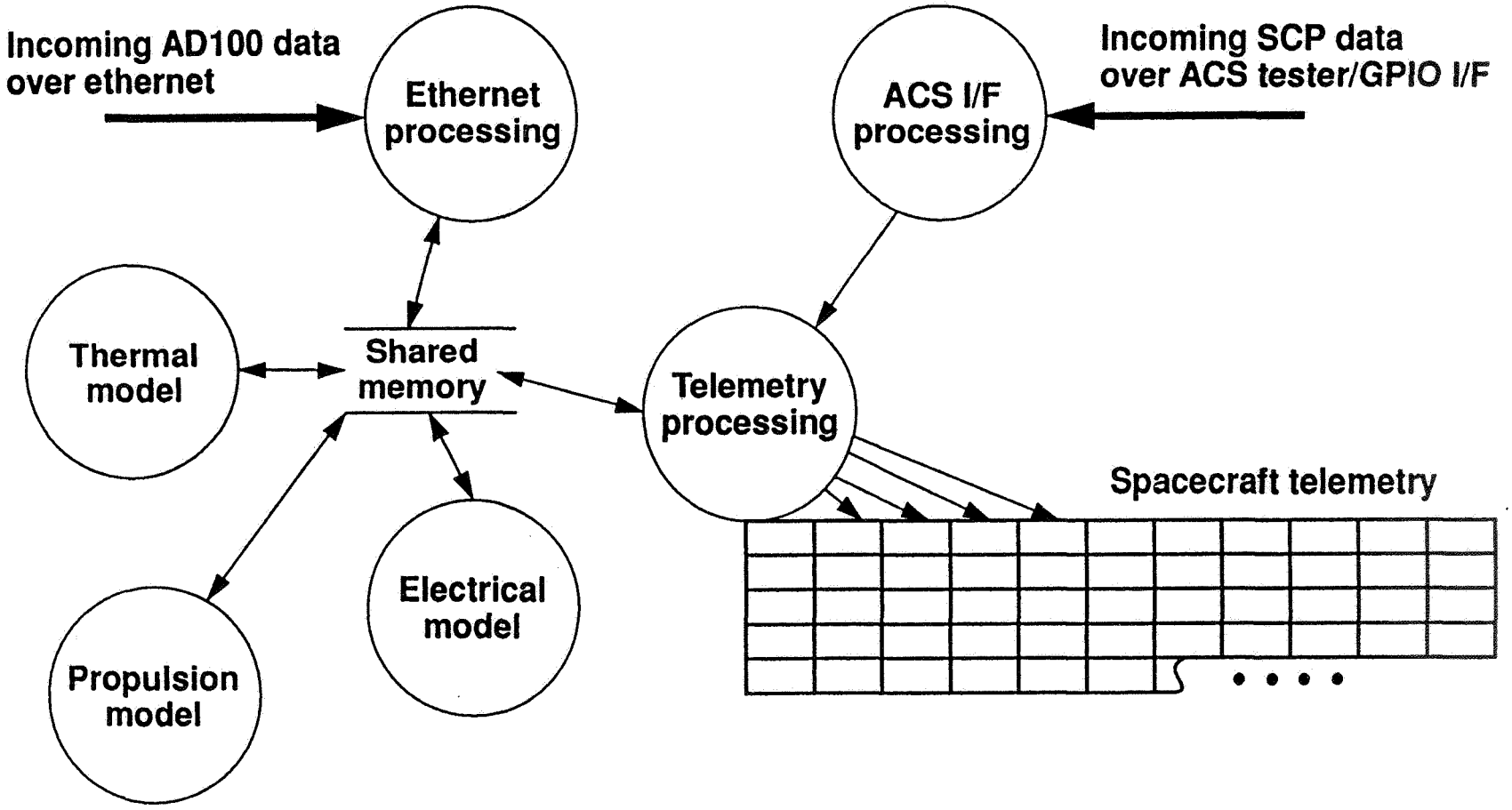
AD100 - DSS Interfaces

HUGHES



Building Spacecraft TM Frame

HUGHES



HS 601 GSMST Provides Extensive Range of Capabilities

HUGHES

Mission planning/preparation
- Development/validation of operational procedures
- Development/checkout of on-orbit test procedures
- Verification of mission sequence of events

Groundstation software
- Groundstation software development/validation with actual flight hardware and software
- Validation of telemetry and command data bases/processing
- Develop/validate groundstation attitude determination/maneuver software
- Design/test of command procedure software

Training
- High fidelity environment for Hughes mission rehearsals (sim lab or HMCC)
- Mission rehearsals from customer groundstation
- Customer operator training

On-orbit support
- Real time mission support for pretesting of procedure mods
- On-orbit anomaly investigations
- On-orbit RAM patch development/checkout/preuse validation
- Provides a mission oriented environment for additional flight software testing (system stress testing)

Other Hughes customers consider this capability essential to their mission operations

Current and Future GSMST Applications

HUGHES

GSMST has become a 'best seller'

It is considered essential by Hughes management for mission planning/preparation/validation

On-going applications

- Aussat B-1 (first HS 601 built for Australia; controlled from Sydney groundstation)
- UHF Follow-on (Navy satellites which use Air Force CSOC facility)
- Galaxy
- MSAT
- DBS

Because of importance of this system, Group authorized funds to develop dedicated facility solely for support of mission activities

HS 601 'Mission Support System'

HUGHES

GSMST system dedicated to support HS 601 spacecraft mission operations

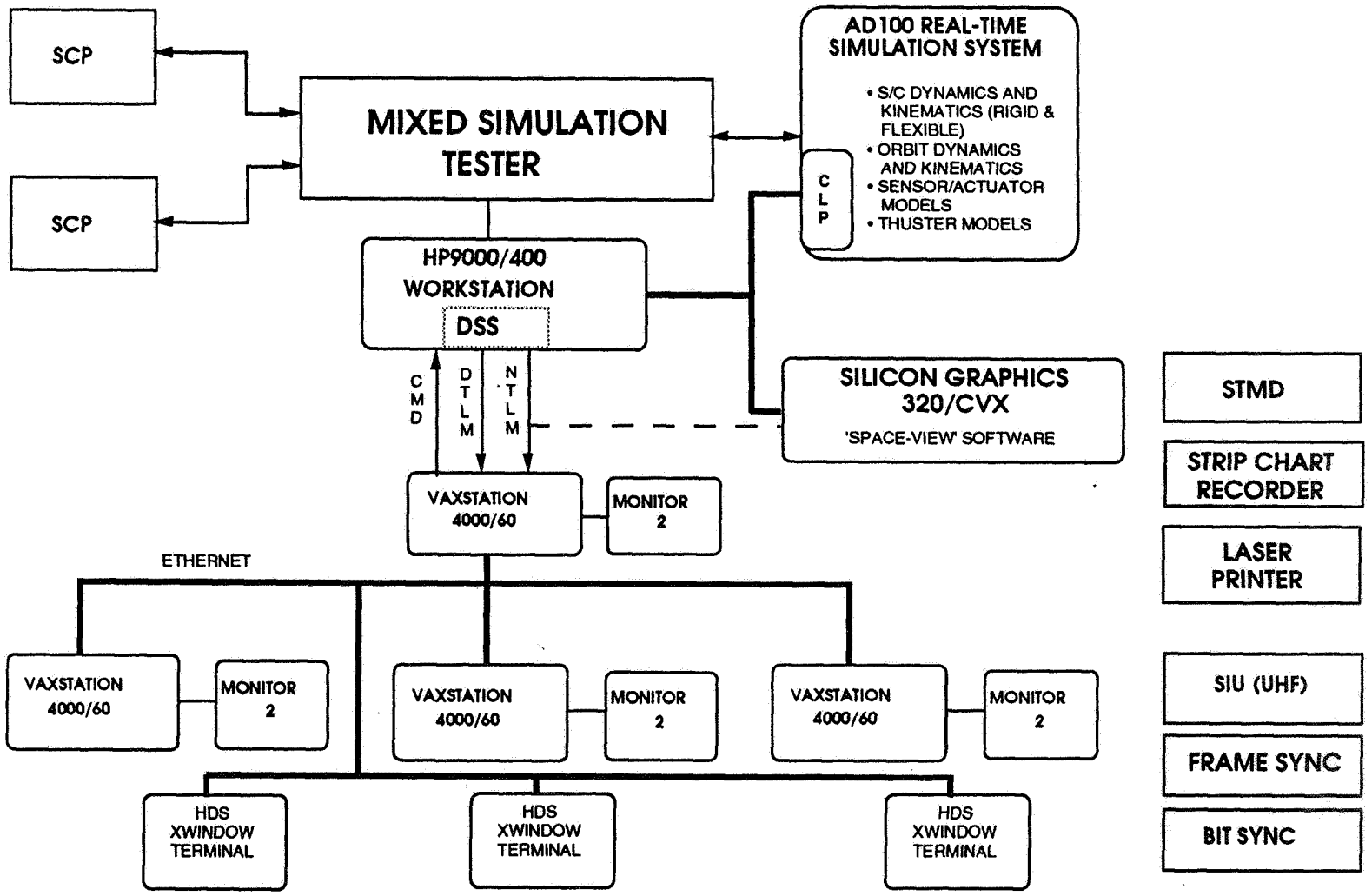
Located within control and dynamics simulation lab (now in building E4)

Provides capability for

- Local, internal rehearsals or procedure/software checkout
- Communications link to Hughes mission control center (in building S67) for full rehearsals
- Remote operation from customer groundstation (Australia, Colorado Springs already demonstrated)
- Receipt and processing of mission/on-orbit telemetry data for analysis

HS601 MISSION SUPPORT GSMST SYSTEM CONFIGURATION

HUGHES



Concluding Remarks

HUGHES

- Hardware in loop MST essential to development of our satellite attitude control subsystems
- Extending MST to include rest of spacecraft and ground segment elements was natural addition to the MST capability
- GSMST proven itself to be essential and cost effective method of supporting mission operations
- Future use of system will provide high-fidelity training aid for our customers

Space Station Common Berthing Mechanism, a Multi-Body Simulation Application

Ian Searle
Boeing Defense and Space Group

July 30, 1992

1 Introduction

This paper discusses an application of multi-body dynamic analysis conducted at the Boeing Company in connection with the Space Station (SS) Common Berthing Mechanism (CBM). After introducing the hardware and analytical objectives we will focus on some of the day-to-day computational issues associated with this type of analysis.

1.1 Hardware

The major components of the CBM are the four 5-bar mechanisms, two berthing port contact rings, and the alignment guides. The function of the CBM is to complete the attachment of two modules once the modules are in close proximity. The modules are initially placed with either the Space-Station Remote Manipulator System (SSRMS) or the Shuttle Remote Manipulator System (SRMS). Once the modules are in position the RMS is put in *limp-mode*, which signifies that all RMS actuators are disabled and the arm can be driven by external forces to a new position. Moving the RMS while it is in the limp-mode means that some of the RMS actuators must be back-driven. The amount of force required to back-drive an RMS is a function of the arm position, which determines what combination of RMS joints must be driven.

After the RMS has been placed in limp-mode, and any residual motion has ceased[1], the final berthing sequence is initiated. The four capture latch mechanisms extend, then retract in an effort to grab the passive berthing

[1] New requirements specify berthing operations shall tolerate small residual motions.

module's latches. As the two modules are being pulled together, the alignment guides force the two modules into precise alignment through contact. Once the modules are in contact, a set of powered bolts[2] fastens the two modules together.

Figure 1 shows the active and passive ports of the CBM[3]. The active port contains eight alignment guides and the four capture-latch mechanisms. The passive port contains four alignment guides, which fit tightly between the active port guides when the two ports are berthed. Typical module and station weights can range from 30,000 lbf to 450,000 lbf.

Figure 2 depicts the capture-latch mechanism. The mechanism consists of four links: the two drive-arms, the idler-arm, and the capture-arm. Only one of the drive-arms is actually driven; we shall call this the primary drive-arm. The other pivots freely about the drive shaft; we shall call this the secondary drive-arm. The primary drive-arm is powered by a DC motor with a purely mechanical clutch, which limits the applied torque. The motor control attempts to drive the motor at a constant speed of 2 rpm. At the time the analysis was performed the motor-clutch design was not final. It was assumed that the rate sensor for the motor would be placed on the output shaft of the clutch. The capture-latch mechanism has a total weight of approximately 5 lbf.

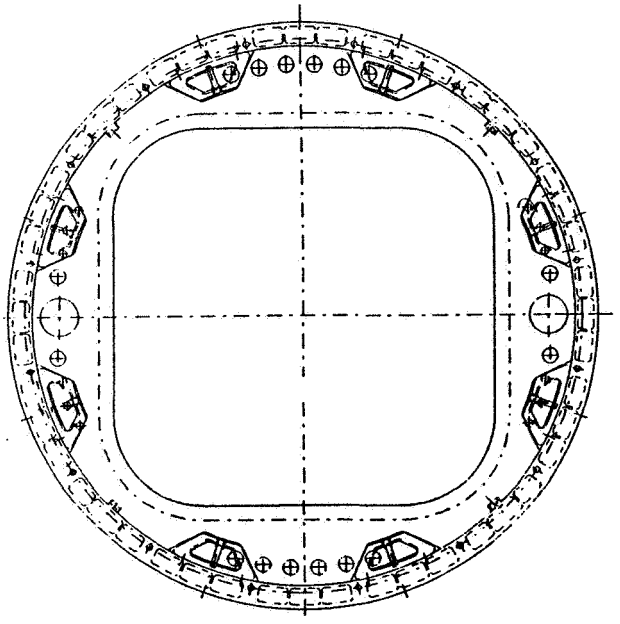
Figure 3 depicts the capture-latch mechanism motion for a typical berthing. Initially the mechanism is in the open state. When the mechanism is commanded to close, the capture-arm swings into the passive port and travels down the edge until it engages the passive port capture-latch fitting. The mechanism drive arm continues to rotate at 2 rpm until the drive-arm is over-center and the ports are in contact.

2 Objective

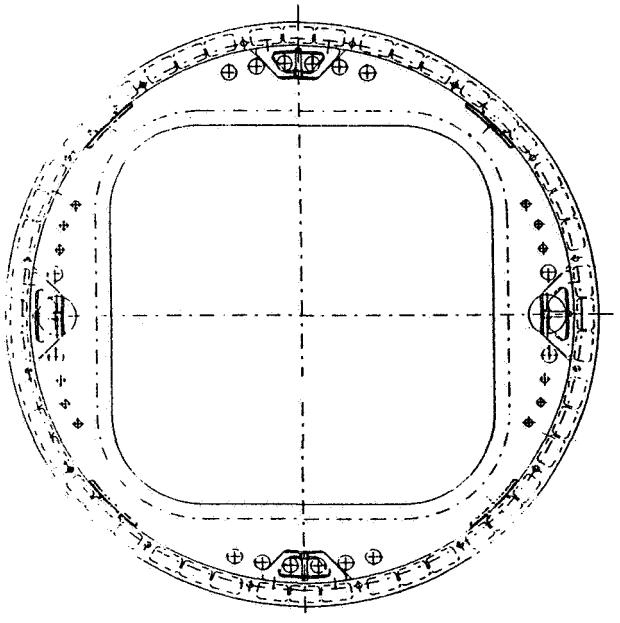
The principle objective of the analysis effort was to assess the capability of the CBM to function in the presence of SSRMS back-drive forces, friction, and worst-case port-to-port misalignment. The capture-latch mechanism drive motor and clutch have a slip-torque-limit of 40 in-lbf. The definition of worst-case SSRMS back-drive forces had not been finalized. In order to

[2]For the rest of this paper will will ignore the powered bolts since they are not a part of this analysis.

[3]The CBM alignment guide configuration was not finalized at the time the analysis was performed



Active Alignment Guides
PDR-I configuration.



Passive Alignment Guides
PDR-I configuration.

Figure 1: Common Berthing Mechanism, Active and Passive Ports

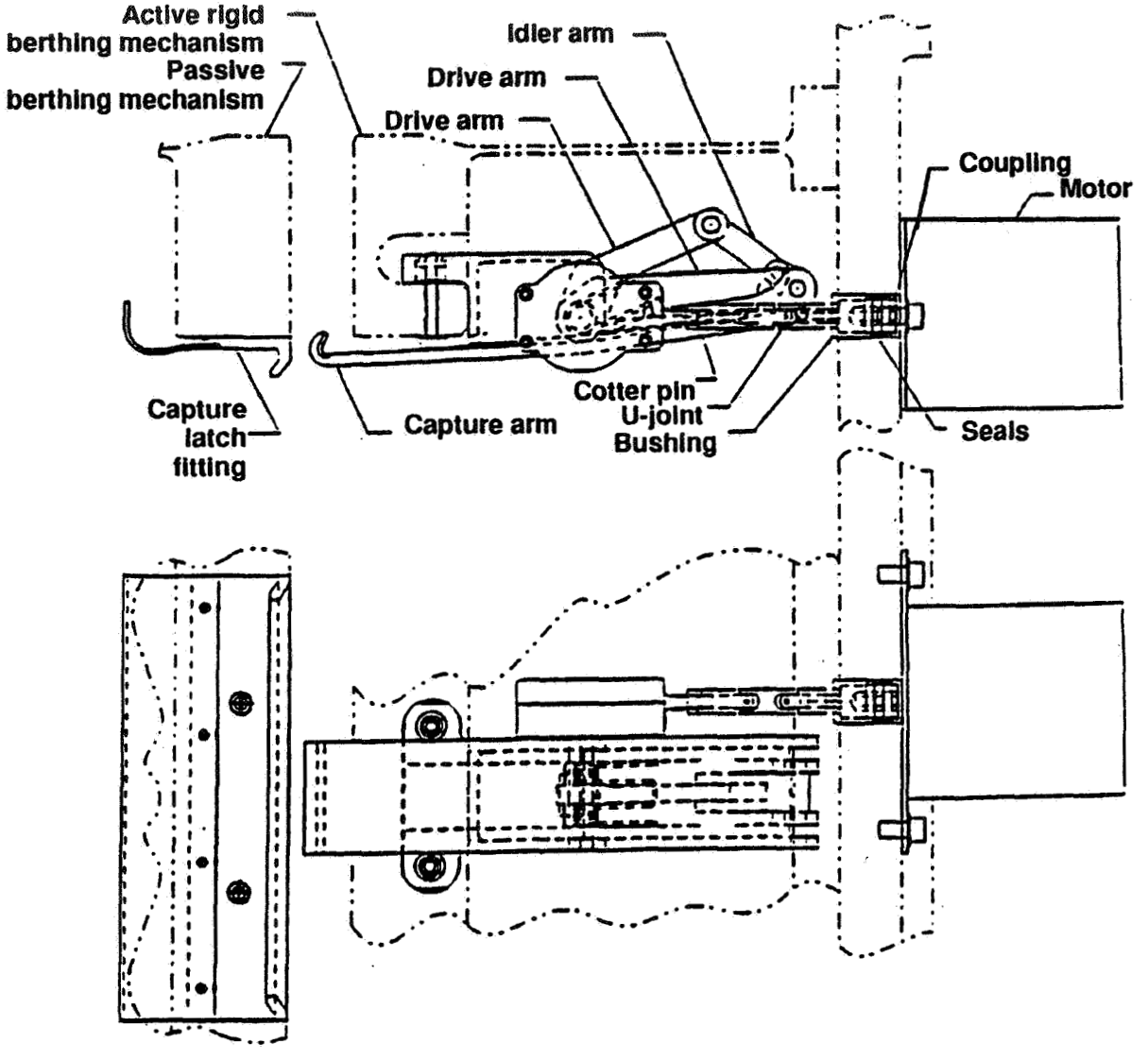


Figure 2: Common Berthing Mechanism, Capture-Latch Mechanism

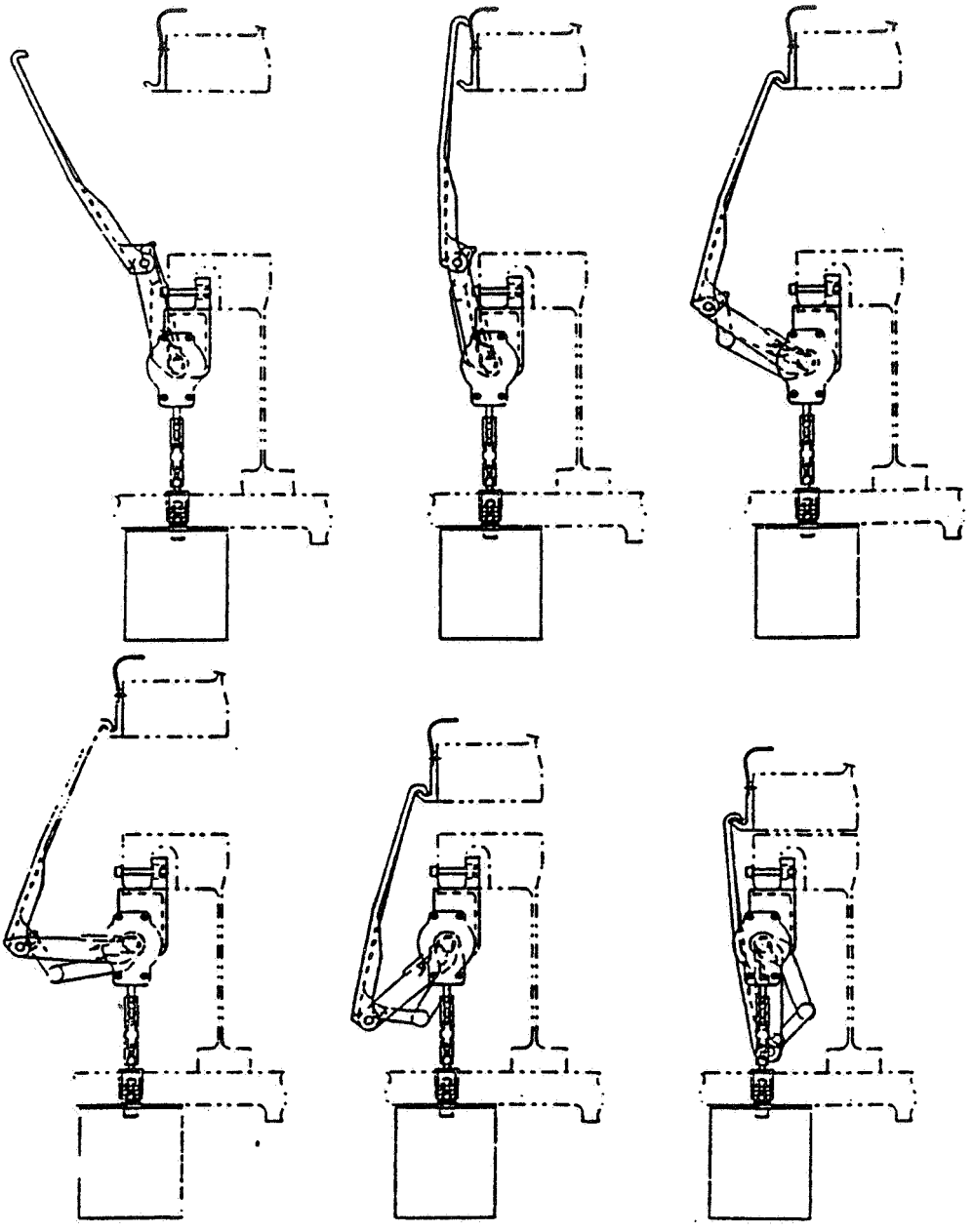


Figure 3: Common Berthing Mechanism, Capture-Latch Mechanism Motion

finalize the design, an analytical capability to assess the design in the face of changing requirements was necessary.

3 Decisions

At this point we had to decide how best to construct a simulation capability in a short period of time. Listed below are most of the options that were considered for this task, and a few words explaining each option's strong and/or weak points.

1. A Boeing in-house FORTRAN code exists that is well developed for transient dynamics problems, but has very little constraint capability. All degrees of freedom (d.o.f.) must be expressed in a common global coordinate system. This means that a mechanism, whether it is in a plane of the global coordinate system or not, must have more degrees of freedom than are required. For this particular problem we have four 5-bar mechanisms, thus we will have to have 96 d.o.f. just for the mechanisms. Furthermore, we will have to add "soft-constraints" to remove the d.o.f that we did not want in the first place. This program is very good at some problems, but this does not appear to be one of them; both the size of the problem, and the 80 plus extra opportunities to make a mistake were significant factors in the decision not to use this program.

2. There is always the option to *roll-your-own.* In this case an integrator and functions to evaluate a set of state-space equations are all that is needed. However, this is no small task. Writing all the support functions is a considerable effort. Experience has shown that 3-4 months could easily be spent writing all the code for a rigid-body analysis. The problem is there is then no room left for the errors or changes that will surely be made. However, if the time had been available this method may have been chosen. The resulting program, specifically tailored to this problem, may have been significantly faster than a program that utilizes a general formulation. This is a very import consideration if the program is going to be used on flight hardware. When flight hardware, and people are involved, safety is an important issue. There can be a tremendous amount of what-if games played; the result is the analyst gets to run hundreds, maybe thousands of simulations.

3. A commercially available tool such as ADAMS, SD-FAST, or DADS could be used. ADAMS has been used in the past for this type of analysis, and has proven quite useful. The fact that the analysis would be performed in two cities, and on several different computers was a severe disadvantage for the commercial codes. It would be a very expensive proposition to purchase 3-4 commercial licenses for 1 analysis task.

4. TREETOPS was considered for several reasons: TREETOPS uses a formulation based upon Kane's method, and therefore does not need to use additional equations for constraints between bodies such as pin-joints. See References [1] through [2]. One draw-back to this method is the assembly of a system mass matrix. As the size of the simulation grows the inversion of $[M]$[4] grows like N^3.

   TREETOPS has a menu-driven preprocessor which performs some error checking on the inputs. In many cases, the model is completely defined in the input file. The point being that an analyst can usually get a faster start if he/she does not have to write and debug code at first.

   TREETOPS is freely-available. Marshall Space Flight Center (MSFC) will supply interested parties with a tape of the FORTRAN source code. The recipient has the freedom to compile and use the software on any available computer. There are no restrictions to number of users or installations. The only restriction is that you do not re-distribute it; this is to help MSFC keep track of usage, and versions of the code. Additionally TREETOPS capabilities put it in the same class as the commercially available codes as far as capability (it is lacking in the graphical-user-interface dept.).

For this project TREETOPS was chosen. We knew we would probably have integration problems with almost any simulation tool we choose due to the small masses of the capture-latch mechanisms, and the high frequencies that contact forces usually introduce.

4 Computational Issues

There is not enough space here to discuss all aspects of the analysis, or even cover it in a chronological overview. Instead we will focus on those aspects

[4] An N-by-N matrix.

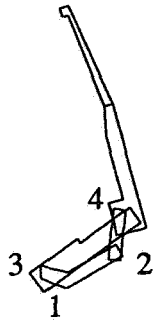


Figure 4: Capture Latch Mechanism, TREETOPS Model

that were either a source of difficulty, or required a large amount of the analyst's time.

4.1 Results Visualization

Interpretation of the simulation results, especially for a 3-dimensional analytical simulation, is often an overlooked issue. A great portion of time is occupied with interpreting simulation results. Not only does a good visualization tool speed up the process, but also reduces the likelihood of errors. To illustrate the point ... Figure 4 contains a simple representation of the TREETOPS mechanism model with the joints numbered. Figure 5 shows time history results of the joint relative euler angles. With this data alone it is difficult, and time consuming to verify that the output is correct. Especially since all the joint angles are not directly related to a either common frame, or a common body. Now add graphical animation output (see Figure 6) to the information at hand. As most will agree, the same conclusion can be arrived at quicker, and with more confidence if Figure 6 is available.

4.2 Numerical Integration

Throughout this analysis there were constant problems with the integration[5] step-size. Every time a new contact force model or a new simulation condition was introduced, the trial and error process of determining an adequate

[5]The standard TREETOPS integrator is a 4-th order Runge-Kutta

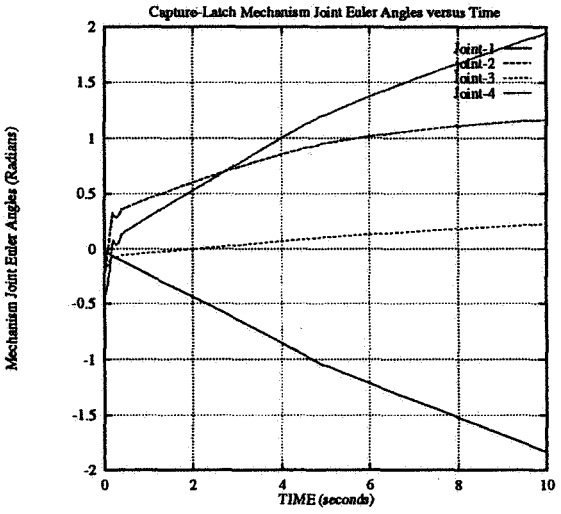


Figure 5: Capture-Latch Euler Angles

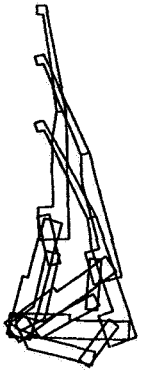


Figure 6: Capture-Latch Animation

step-size was repeated. An initial step-size of $\frac{1}{10}$ the period of the highest frequency component in the model was used repeatedly. However, this method often left us with a step-size that was too small. How can a step-size be too small? When the required simulation duration is 15-25 seconds (real time) a very small step-size can result in hours of cpu-time; this is unacceptable in the development phase of the analysis.

An additional factor interfered with our efforts to find a reasonable step-size. The difference between largest and smallest mass values was eight orders of magnitude. Since the mass matrix is inverted at every integration step, an ill-conditioned mass matrix can cause run-time problems that exhibit symptoms similar to step-size problems. When the simulation is unstable TREETOPS will often expire with the message "Mass matrix not positive definite". Since the mass matrix is configuration dependent, this implies that the system has reached a numerically impossible state.

Since there were two significant contributors to our integration problems, we decided to eliminate one of the sources. Since the mechanism velocities are small we decided to test the assumption that the mechanism component mass properties played a small role in the overall system dynamics by increasing the mechanisms mass properties by two orders of magnitude. Simulation comparisons with earlier simple mechanism models showed that this assumption was reasonable for some conditions. However, we must be careful and use this assumption only as a *stop-gap*, and not a permanent fix.

4.2.1 Contact Modeling

The contact modeling effort deserves some attention because at least 50% of the total effort was spent in the pursuit of accurate contact models. Not only did the contact models take a significant amount of man-power to produce; they also require a significant amount of computer time to simulate. For this problem there are 5 plausible types of contact:

1. Latch-arm / capture-hook.
2. Passive port ring / active port ring.
3. Passive port alignment guide / active port alignment guide.
4. Passive port alignment guide / active port ring.
5. Active port alignment guide / passive port ring.

For each type of contact, specific geometric calculations are performed to check for interference, or constraint violation on a case-by-case basis. If interference is detected a reaction force is computed. The reaction force is a function of the interference and the interference rate. We will look briefly at the alignment guide/guide contact model as an example.

The guide/guide contact is modeled with an edge/edge contact model. Figure 7 contains a picture of the model nodes, local coordinate systems and vectors used in the contact force computation. The procedure uses the cross product of two vectors, each representing an edge to facilitate computation of contact forces. The logic used follows:

1. Compute $\boldsymbol{C} = \boldsymbol{R}_a \times \boldsymbol{R}_b$ where $\boldsymbol{R}_a = \boldsymbol{A1} - \boldsymbol{A2}$ and $\boldsymbol{R}_b = \boldsymbol{B1} - \boldsymbol{B2}$. The shortest distance between the two lines will have the same direction as $\boldsymbol{C}$.

2. Transform the position and velocity vectors of the edge endpoints into the new coordinate system defined by the direction cosine matrix $\boldsymbol{A}$, formed as follows:

$$\boldsymbol{A} = \begin{bmatrix} \dfrac{\boldsymbol{C} = \boldsymbol{R}_a \times \boldsymbol{R}_b}{} \\ \dfrac{\boldsymbol{R}_a}{} \\ \dfrac{\boldsymbol{C} \times \boldsymbol{R}_a}{} \end{bmatrix}$$

3. The shortest distance between $\boldsymbol{R}_a$ and $\boldsymbol{R}_b$ is the distance between $\boldsymbol{A1}$ and $\boldsymbol{B1}$ along the $\boldsymbol{C}$ axis. If that distance is less than zero:

   (a) Compute point of intersection:

$$intx = -y_1 \left(\frac{x_2 - x_1}{y_2 - y_1} \right) + x_1$$

   (b) Calculate offsets:

$$\boldsymbol{aoff} = intx \cdot \boldsymbol{R}_a$$

$$\boldsymbol{boff} = \left((intx - B1(2))^2 + (B1(3))^2 \right)^{1/2} \cdot \boldsymbol{R}_b$$

   (c) Calculate relative velocity at point of intersection:

$$\boldsymbol{V}_a = \boldsymbol{V}_{a1} + \boldsymbol{\omega}_a \times \boldsymbol{aoff}$$

$$\boldsymbol{V}_b = \boldsymbol{V}_{b1} + \boldsymbol{\omega}_b \times \boldsymbol{boff}$$

   where $\boldsymbol{V}_{a1}$, $\boldsymbol{V}_{b1}$, $\boldsymbol{\omega}_a$, and $\boldsymbol{\omega}_b$ are the velocities at points A_1 and B_1. These velocities are supplied by TREETOPS.

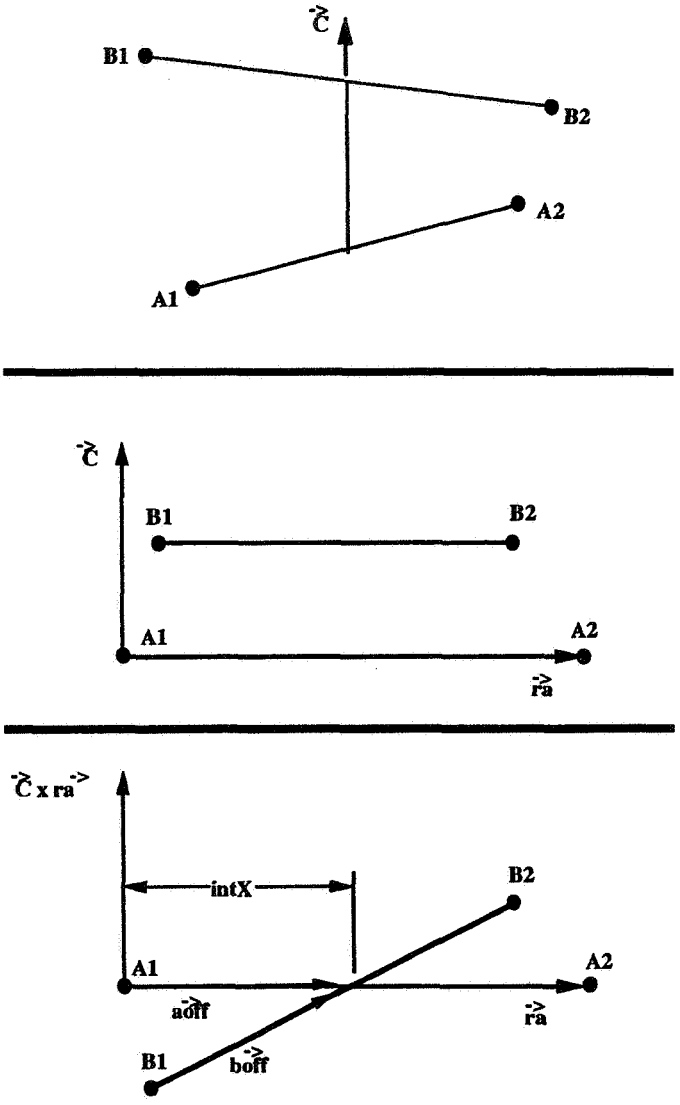


Figure 7: Guide - Guide Contact Geometry

The above procedure calculates the relative distance and velocity between the two edges. A FORTRAN subroutine that performs this task can run anywhere from two to four pages (including comments). Once the contact geometry is defined the remaining task is to define the physical properties of the contact.

For elastic contact at low velocities it may be acceptable to use a simple linear[6] spring and damper to model the impact effect. Calculation of the proper stiffness can usually be accomplished via careful static analysis of the involved components. One can take the local stiffness, and combine it in series with the global stiffness of the contacting parts. In this case the load path we are concerned with is the in-plane loading of the edges of an alignment guide. Like the integration step size, calculated contact stiffness values are only a starting point. Once the initial value is calculated it must be tested. Insight into the appropriateness of the initial value can be gained by observing the measured deflections at the point of contact.

4.3 Back-Driving the RMS

Modeling and simulating the RMS back-drive forces was one of the simpler aspects of the analysis. Why is such a complex piece of hardware simple? Neither the SSRMS or the SRMS was modeled in any detail. Instead a *requirement-model* was assembled. A requirement-model does not necessarily behave like the physical component it is supposed to represent; instead its behavior is representative of the worst-case events or conditions as spelled out in the requirements document. In this case the requirement-model of the RMS was an arm that supplied a constant 150 lbf resistance in all directions.

Simulations showed that the existing capture-latch mechanisms could not successfully close the gap between the active and passive ports. Indeed simulations showed that a mechanism with 3 times the authority was required to pull the two ports together under extreme misalignment conditions and worst case contact friction and RMS back-drive forces.

5 Summary

We have discussed various aspects and problems associated with a 3-4 month analysis effort. Unfortunately there is not room enough to discuss all of the problems involved. From a computational point of interest it is important to note:

[6]linear in the sense that the spring has a constant stiffness when engaged

1. Data visualization tools (VDS, BPLOT) are crucial to performing a timely analysis. As simulations get more and more complex, so must the output visualization capability.

2. The method used herein for calculating contact forces is much too tedious, time consuming, and error prone. If this method proves to be the most computationally efficient, then some way of speeding up the algorithm development and debugging is in order.

3. It is important to select the software tool best suited for a particular job. Therefore, a collection of specialized tools like TREETOPS, that solve a certain class of problems well, is essential. Our experience shows that software that tries to do everthing, ends up doing nothing well.

6 Acknowledgements

Thanks to Bill Gustafson for always providing interesting work. We would like to thank Phil Williams and Pat Tobby from Control Dynamics Company, who have, and continue to develop a more detailed simulation capability for Boeing, and MSFC at the Marshall 6-dof facility. They provided invaluable assistance during the analysis.

References

[1] *User's Manual for TREETOPS, Revision 8*, Singh Likins, et al., NASA contract no. NAS8-34588.

[2] *Dynamics of Flexible Bodies in Tree Topology*, R. P. Singh and R. J. VanderVoort, Journal of Guidance, Control, and Dynamics, Sept-Oct 1985.

Simulation of Linear Mechanical Systems

S. W. Sirlin
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

INTRODUCTION

A dynamics and controls analyst is typically presented with a structural dynamics model and must perform various input/output tests and design control laws. The required time/frequency simulations need to be done many times as models change and control designs evolve.

This paper examines some simple ways that open and closed loop frequency and time domain simulations can be done using the special structure of the system equations usually available. Routines were developed to run under Pro-Matlab [1] in a mixture of the Pro-Matlab interpreter and Fortran (using the .mex facility). These routines are often orders of magnitude faster than trying the typical "brute force" approach of using built-in Pro-Matlab routines such as *bode*. This makes the analyst's job easier since not only does an individual run take less time, but much larger models can be attacked, often allowing the whole model reduction step to be eliminated.

I will first briefly discuss the standard model forms, then address each of the simulation cases separately.

LINEAR MECHANICAL SYSTEM MODELS

Linear mechanical system models have a special second order differential form. In general the various structural dynamics codes generate equations in the form:

$$M\ddot{q} + N\dot{q} + Kq = Bu + Gf, \tag{1}$$

$$y = C_p q + C_v \dot{q} + Du,$$

where $q \epsilon R^n$ is the modal state, $u \epsilon R^m$ is the control input, $y \epsilon R^p$ is the output, and $f \epsilon R^q$ is the disturbance input. Systems of the above form can almost always (generically) be put into modal form, and this modal form is typically what is given to the dynamics and controls analyst by the structural modellers (e.g. NASTRAN output):

$$\ddot{\eta} + 2\Sigma\dot{\eta} + \Lambda\eta = Bu + Gf, \tag{2}$$
$$y = C_p\eta + C_v\dot{\eta} + Du,$$

where $\eta\epsilon R^n$ is the modal state, Σ and Λ are diagonal matrices, u and f are the control input and disturbance force respectively, and y is the output which can be position, velocity, input force, or some combination of these. The transformation to the form (2) is not easy for general damping, (N) but typically damping is so poorly known that simple modal damping models suffice. I will assume that (2) is available for the rest of the paper.

The standard analysis tools of Pro-Matlab require first order form, which can be done as follows:

$$\dot{x} = \bar{A}x + \bar{B}u + \bar{G}f, \tag{3}$$
$$y = \bar{C}x,$$
$$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}, \qquad \bar{A} = \begin{bmatrix} 0 & I \\ -\Lambda & -2\Sigma \end{bmatrix},$$
$$\bar{B} = \begin{bmatrix} 0 \\ B \end{bmatrix}, \qquad \bar{G} = \begin{bmatrix} 0 \\ G \end{bmatrix}, \qquad \bar{C} = [\,C_p \quad C_v\,].$$

Note that the special structure does not fit any of the standard forms (block diagonal, triangular, banded, or Hessenberg).

To the above plant equations must be added typical control dynamics:

$$\dot{w} = A_c w + B_c(r - y), \tag{4}$$
$$u = C_c w + D_c(r - y),$$

where r is some reference command input. Again resorting to brute force we have the whole system:

$$\dot{z} = \hat{A}z + \hat{B}r + \hat{G}f, \tag{5}$$
$$y = \hat{C}z,$$
$$z = \begin{bmatrix} q \\ \dot{q} \\ w \end{bmatrix}, \qquad \hat{A} = \begin{bmatrix} \bar{A} - \bar{B}D_c\bar{C} & \bar{B}C_c \\ -B_c\bar{C} & A_c \end{bmatrix},$$
$$\hat{B} = \begin{bmatrix} BD_c \\ B_c \end{bmatrix}, \qquad \hat{G} = \begin{bmatrix} \bar{G} \\ 0 \end{bmatrix}, \qquad \hat{C} = [\,\bar{C} \quad 0\,].$$

Note that now even the special structure of (3) is not present (in general) even in the upper left block of the $\hat{A}$ matrix. Due to the feedback we have lost all information regarding the eigenstructure. The system may not even be very sparse anymore.

FREQUENCY DOMAIN RESPONSE - OPEN LOOP

If we desire to compute the open loop frequency response for (2,3)

$$H(s) = \bar{C}(Is - \bar{A})^{-1}\bar{B},$$

there is the built-in *bode* command of Pro-Matlab, that uses the well known method of Laub [2]. While the method is efficient for the general case, given the special structure of (2) we can write down a much simpler solution:

$$H_{ij}(s_k) = \Sigma_{l=1}^{n} \frac{(C_{pil} + C_{vil}s)B_{lj}}{s_k^2 + 2\zeta_l\omega_l s_k + \omega_l^2}.$$

This can be easily implemented using just a few lines of the Pro-Matlab interpreter, the only trick being to come up with some way of representing a rank 3 array. The header of a routine that does this, *freqm2d*, is listed in the Appendix. Timing results are as one would expect. From Figure 1 (results obtained on a VAX 11/780), the brute force computation time increases roughly quadratically, while the direct solution requires only a linear cost with system order.

FREQUENCY DOMAIN RESPONSE - CLOSED LOOP

Once the system of (2,4) is put into the form (5) then the same standard tools apply as for the open loop case. As was pointed out above, the feedback destroys all of the open loop eigenstructure, hence there is no direct solution as in the open loop case. On the other hand consider Figure 2. Taking an "analog" approach to the analysis, given the individual transfer functions for the plant and the control one can combine them to get the overall transfer function via the usual algebra in the frequency domain:

$$H(s) = GF(I + GF)^{-1} = G(I + GF)^{-1}F = (I + GF)^{-1}GF.$$

In comparison to the brute force approach, there are a few issues:

- We can take advantage of any special forms for the individual blocks - this is clearly advantageous.

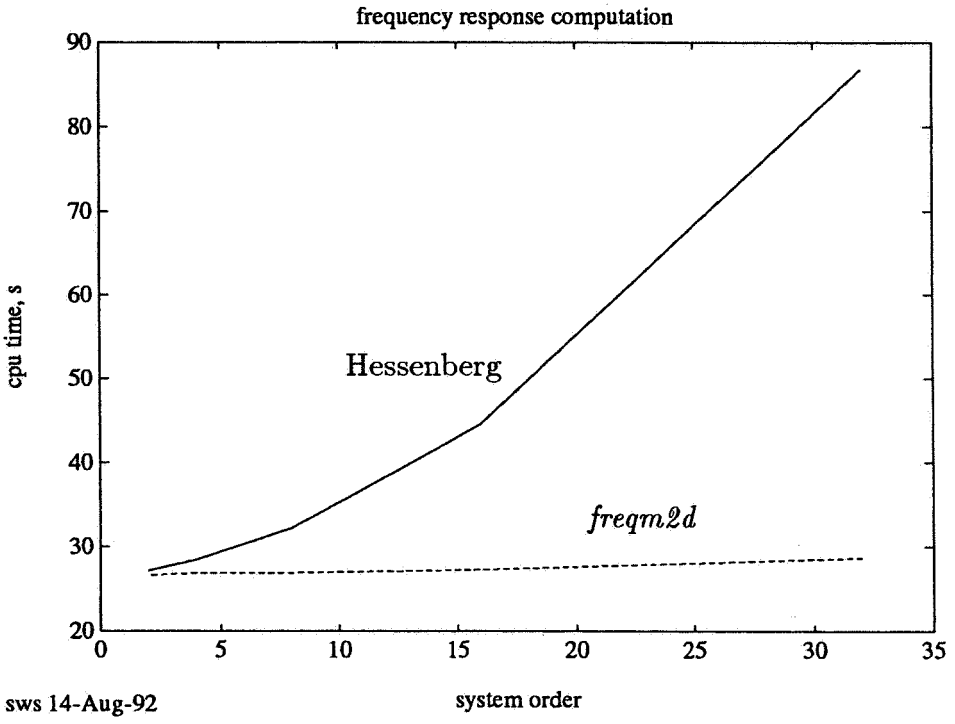


Figure 1. Computation time for open loop frequency response: Hessenberg versus *freqm2d* (specialized for mechanical systems). The test case is a second order system with a variable number of modes.

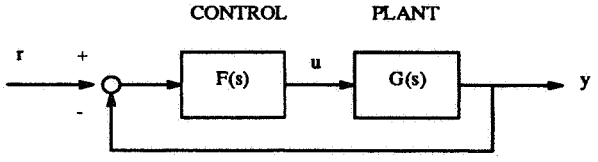


Figure 2. A simple closed loop system.

- We require matrix inversion of the order of the minimal input (to the plant or the control) - this is not clearly good or bad computationally. We're trading the usual n^2 cost for m^3, which is likely to be advantageous for large plants with only a few feedback loops, but a loss for full state feedback.

- The technique allows work directly with experimentally determined transfer

C-5

functions - there is no need for system identification to proceed all the way to a state space model.

- We require storage of the intermediate results - this is an implementation issue. Currently the routines I have [3] first generate the individual transfer functions then combine the results. In some cases the memory to hold all the individual transfer functions can be quite high. This cost can be eliminated by moving the whole routine to Fortran however. The header of a routine that implements this, *feedbackmtf*, is listed in the Appendix.

As a simple example, consider a plant with 64 modes, 4 outputs, m inputs, and m pseudo-derivative feedback controllers. Timing results (again on a VAX 11/780) for a modest test case are shown in Figure 3.

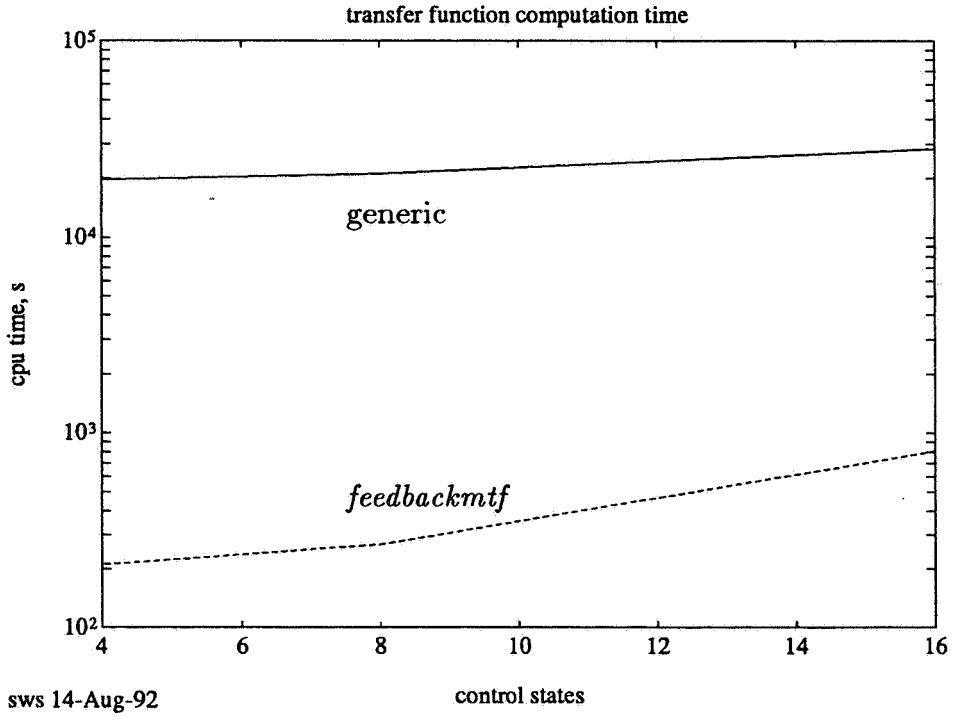


Figure 3. Computation time for closed loop frequency response: generic system approach versus subsystem approach (*feedbackmtf*). A 64 mode example with a variable amount of feedback.

A more stressing test, but one of practical interest was the NASA CSI (Control Structure Interaction) Focus Mission Interferometer (FMI) structural control [4]. The FMI is a $30m$ baseline interferometer, with the goal of controlling optical pathlength to the nanometer level. The plant model has 527 modes, with 17 rigid body modes and modal frequencies from 3.9 to $40000Hz$. To this we want to add 25 9-state controllers with displacement and force measurements, and force output. This gives a total system order of 1279. Use of the Hessenberg routines on the whole system is completely unrealistic in this case. Model reduction is a sensible approach, especially as modes above 100 Hz are not believable. On the other hand it is clear that model reduction is only required due to the numerical inefficiency of dealing with the problem as a whole block. Calculations of the individual components can be easily carried out using the full model without having to worry about truncation issues such as residual flexibility or the need for augmenting the reduced system with Ritz vectors [5]. The computation (for 1068 frequency points) took $4640s$ inverting the 25×25 input force matrix at every frequency. If this is to be done many times for the same system, then clearly model reduction is desirable, but if done only a few times (as was the case here), then model reduction isn't worthwhile. In this case only a 1×1 inverse is really required as the structural loops are all uncoupled, and so some time is certainly wasted in the (LINPACK) routine checking zeroes. Trying to construct more efficient routines I've run into some of the limitations of Pro-Matlab:

- Using an external (Fortran) routine many times (say for sequentially closing many loops) can lead to enormous allocation of unnecessary memory,

- Describing a system as separate blocks requires a data object much more complex than a matrix. What is needed is the concept of a data structure, for example that of the C language, or even better something that could be created, changed, and destroyed interactively such as the generalized arrays of APL2 or J[6]. Without this function, input lists must be long and specialized to particular cases.

In spite of these limitations, the tool is quite useful, enabling analysis for large systems and being fast for more modest systems.

TIME DOMAIN RESPONSE - OPEN LOOP

While the general solution to (3) is given by the well known matrix exponential, this is very difficult and expensive to compute in practice for general systems [7], often requiring a model reduction step.

On the other hand, the solution to (2) above is quite obvious since we just have a set of uncoupled second order equations. The solution is

$$\xi_k(t+\Delta t) = \Phi_k \xi_k(t) + \Gamma_k f_{Tk}(t), \qquad k = 1, ...n,$$

$$\xi_k(t) = \begin{bmatrix} \eta_k(t) \\ \dot{\eta}_k(t) \end{bmatrix}$$

$$f_T = Bu + Gf,$$

where the Φ_k and Γ_k matrices are easily calculated given the k^{th} eigenvalue (with various special cases depending on whether it is 0, real, imaginary, or complex). The computational cost increases linearly with state order. In addition the propagation from one time to the next can be done with a $2n \times 2$ matrix (n Φ_k's) versus the usual $2n \times 2n$ exponential matrix, saving on storage space.

Results for the Galileo spacecraft present another practical case of interest. The model at hand was built for investigating the possibility of finding ways to shake the spacecraft to free the stuck high gain antenna. The model had 142 modes, with 8 rigid body modes (6 for the whole plus the dual spin main bearing and a scan platform bearing), and with structural modal frequencies from 0.143 to $144Hz$. In the case shown the response of the system to the deployment and stow of a movable low gain antenna (LGA) was investigated. We've also looked at the system response to thruster firings and to torques at the spin bearing and scan platform bearing.

The routines were coded in Fortran, linked to Pro-Matlab, and run on a Sun SPARCstation 2 (Figure 4). The header for the main routine, *lsim2*, is listed in the Appendix. The computational gain in comparison to the standard built in Pro-Matlab c2d and lsim on the whole system is evident.

TIME DOMAIN RESPONSE - CLOSED LOOP

To see how to close the loop we look first at a simple coupled system with no external inputs:

$$\dot{x}_1 = A_1 x_1 + A_2 x_2,$$

$$\dot{x}_2 = B_1 x_1 + B_2 x_2.$$

The exact solution is again the matrix exponential

$$x(t) = \Phi(t) x_0 = e^{At} x_0, \qquad A = \begin{bmatrix} A_1 & A_2 \\ B_1 & B_2 \end{bmatrix},$$

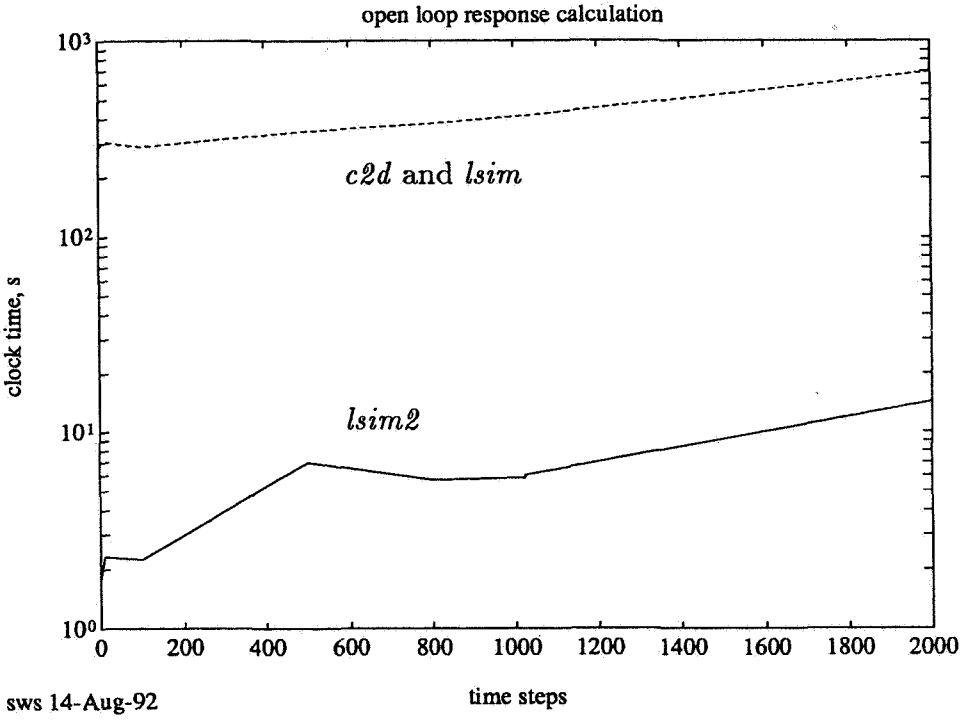


Figure 4. Galileo LGA deployment response calculation cost. Standard approach (*c2d* and *lsim*) versus approach specialized for mechanical systems (*lsim2*).

$$\Phi(t) = I + \begin{bmatrix} A_t & A_2 \\ B_1 & B_2 \end{bmatrix} t + \begin{bmatrix} A_1^2 + A_2 B_1 & A_1 A_2 + A_2 B_2 \\ B_1 A_1 + B_2 B_1 & B_1 A_2 + B_2^2 \end{bmatrix} \frac{t^2}{2} + \ldots .$$

Unlike the frequency domain case, we can't easily find the response of the two systems separately and then combine algebraically (this would require a convolution in time). On the other hand if x_2 were controller states in a discrete controller, we would be perfectly justified in generating the responses for each system separately assuming a zero order hold for the other system, as this is exactly what really happens. Arguing that this must still be reasonable for low pass analog controllers, we have an approximate solution:

$$\hat{x}(t) = \hat{\Phi}(t) x_0 = \begin{bmatrix} \phi_1 & \gamma_1 \\ \gamma_2 & \phi_2 \end{bmatrix} x_0,$$

$$\phi_1 = e^{A_1 t}, \qquad \gamma_1 = \int_0^t e^{A_1(t-\tau)} d\tau A_2,$$

$$\phi_2 = e^{B_2 t}, \qquad \gamma_2 = \int_0^t e^{B_2(t-\tau)} d\tau B_1,$$

$$\hat{\Phi}(t) = I + \begin{bmatrix} A_1 & A_2 \\ B_1 & B_2 \end{bmatrix} t + \begin{bmatrix} A_1^2 & A_1 A_2 \\ B_2 B_1 & B_2^2 \end{bmatrix} \frac{t^2}{2} + \dots .$$

Comparing the approximate to the true solution, we see that

$$\Phi - \hat{\Phi} = \begin{bmatrix} A_2 B_1 & A_2 B_2 \\ B_1 A_1 & B_1 A_2 \end{bmatrix} \frac{t^2}{2} + \dots ,$$

so that the error $e(t) = ||x - \hat{x}||$ is of order t^2, so this method is of order 1. With this method we can use any special structure present in the individual subsystems, which can save considerable computational cost. On the other hand we must make sure to take a small enough time step. Currently a Fortran/Pro-Matlab implementation just uses a fixed step size. Local error estimates could be used to warn of possible trouble or change the step size. The routine *lsim2fb* (the header is listed in the Appendix) is the current implementation of these ideas.

Returning to the Galileo example, the previous section mentioned 8 rigid body modes, but the spin bearing and scan platform bearing (these are called the Clock and Cone degrees of freedom for the scan platform) are likely to be in a "caged" mode, actively controlled to have about $0.25Hz$ natural frequency with 70% damping ratio. A 2-state controller was added to implement this. Figure 5 shows the results of a Fortran/Pro-Matlab implementation, again comparing with the standard generic path in Pro-Matlab on a Sun SPARCstation 2. Note that the simulation times are very close to the open loop case. The difference in the system state was less than 4% for this case, in which the time step used was $11msec$. The time step was chosen based on the minimum discretization for thruster pulses rather than to minimize simulation error.

SUMMARY AND CONCLUSIONS

A set of tools has been built specifically for linear mechanical systems, open and closed loop, for use with Pro-Matlab. These tools use the pre-existing open loop eigenstructure typically available for structural dynamics models, which can save orders of magnitude in cpu time for typical problems.

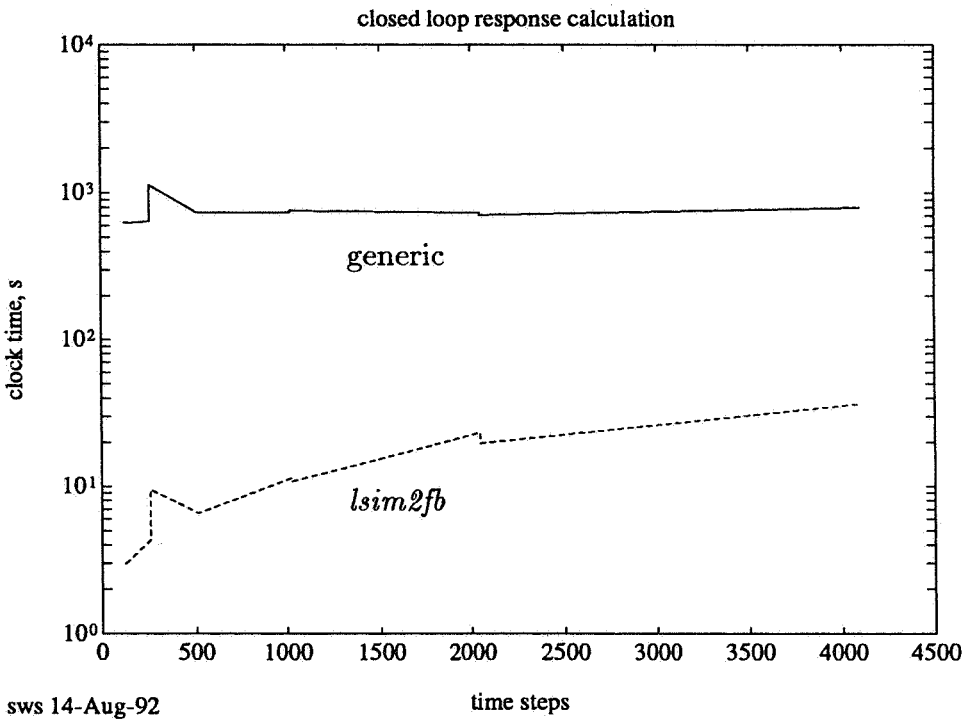


Figure 5. Galileo LGA response calculation cost given closed loop clock and cone control. Standard system approach versus approximate subsystem approach (*lsim2fb*).

For closed loop systems, treating the analysis of each block separately allows analysis of problems that might otherwise be too large, reducing or eliminating the model reduction step in the analysis.

ACKNOWLEDGEMENT

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

REFERENCES

1. Pro-Matlab User's Guide and Control System Toolbox, The MathWorks, Inc., Natick, MA, 1990.

2. A. J. Laub, "Efficient Multivariable Frequency Response Computations," IEEE Transactions on Automatic Control, Vol. AC-26, No. 2, April 1981.

3. S. W. Sirlin, "Pro-Matlab Functions for Frequency Response," JPL EM 343-1163, December 1989 (JPL internal document).

4. S. W. Sirlin, "Active Structural Control for Damping Augmentation and Compensation of Thermal Distortion," Second Joint Japan-U.S.A. Conference on Adaptive Structures, Nagoya, Japan, November 1991.

5. C. C. Chu, M. H. Milman, "Computational Issues in Optimal Tuning and Placement of Passive Dampers," this proceedings.

6. R.K.W. Hui, K.E. Iverson, E.E. McDonnell, A. Whitney, "APL \ ?," APL Quote Quad, Volume 20, Number 4, July, 1990, pp. 192-200.

7. C. Moler, C. van Loan, "Nineteen Dubious Ways to Compute the Exponential of a Matrix," SIAM Review, Vol. 20, No. 4, October 1978.

APPENDIX
SOME SELECTED Pro-Matlab ROUTINES FOR SIMULATION

Below are included the selections from headers for the main routines used in the above examples.

I. Frequency domain

A. Open loop

```
function [ht] = freqm2d(sigma2,omega2,b,c,d,np, w)
% [h] = freqm2d(sigma2,omega2,b,c,d,np, w)
%
% MIMO tf calculation, given second order modal form
%
%    s^ 2 x(i) + sigma2(i) s x(i) +omega2(i) x(i) = b(i,:) u
%
%    y = [yp]
%        [yv]
%
%         yp = cp x   + dp u
%         yv = cv s x + dv s u
%
%              c = [cp]    d = [dp]  size np
%                  [cv]        [dv]
%
% The results are returned in a matrix
%
%       h = [h11, h12, ..., h1m, ]
%           [h21, ...            ]
%           [...
%
% given m inputs and q outputs,
% where each hij is an npts x 1 vector.
%
```

B. Closed loop

```
function ht=feedbackmtf(g,mg,ng,fg, h,mh,nh,fh, np)
% function ht=feedbackmtf(g,mg,ng,fg, h,mh,nh,fh, np)
%
%   Combine the frequency response of two systems in a feedback
```

```
loop.
% Two vector inputs and two vector outputs are allowed for:
%
%      r   +      ug
%      --Mg-->o------>G-----------Ng--> yg
%           - |                 |
%             |                 |
%             Fh                Fg
%             |                 |
%             |                 |+
%      <--Nh----------H<--------o<--Mh-- d
%       yh                 uh      +
%
```

II. Time domain

A. Open loop

```
% function [x, dx]= lsim2(sigma,lambda,b,u,dt,x0, dx0);
%   Generate the time response of the system:
%
%      (d^2)x + 2 sigma dx + lambda x = b*u
%
```

B. Closed loop

```
%function [y,u,x,dx,maxle]= lsim2fb(sigma,lambda,b,g,f,cp,cv,...
%                           Ac,Bc,Cc,Dc, r,dt,x0, dx0);
%
% Generate the time response of a second order system with feedback:
%
%      (d/dt)^2 x + 2 sigma (d/dt)x + lambda x = b u + g f
%      y = cp x + cv (d/dt) x
%
%      (d/dt) w = Ac w + Bc (r-y)
%      u = Cc w + Dc (r-y)
%
```

Optimization-based Controller Design for Rotorcraft

N.-K. Tsing[1], M.K.H. Fan[2], J. Barlow[3], A.L. Tits[1], M.B. Tischler[4]

Abstract

An optimization-based methodology for linear control system design is outlined by considering the design of a controller for a UH-60 rotorcraft in hover. A wide range of design specifications is accounted for: internal stability, decoupling between longitudinal and lateral motions, handling qualities and rejection of wind gusts, while taking into account physical limitations in the swashplate displacements and rates of displacement. The methodology crucially relies on user-machine interaction for tradeoff exploration.

1. Introduction

Airframe concepts for future rotorcraft such as the light attack helicopter (LHX) include high effective hinge offset rotors and multiple control effectors to maximize maneuverability. This and other possible features lead to a high level of bare airframe instability, control cross coupling, and control redundancy. At the same time there is a need to include explicitly in a design methodology numerous specifications on control limits, actuator capabilities, cross coupling limits, and handling qualities. The classical control techniques which have been the primary tools of the industry are not well suited for such complex combinations. There is a great need for improved techniques. Numerous modern control methodologies have recently been proposed for application to these types of problems(e.g., [1–3]). However these studies have not adequately accounted for the many practical implementation problems of rotorcraft [4]. Key concerns in evaluating prospective modern control designs are the interaction of the rotor system dynamics [5] and the explicit inclusion of the complex design specifications(Mil-Specs [6]) in the design process.

A methodology is proposed that can account for various types of concurrent specifications: stability, decoupling between longitudinal and lateral motions, handling qualities, and physical limitations of the swashplate motions. This is achieved by synergistic use of analytical techniques (Q-parametrization of all stabilizing controllers, transfer function interpolation) and advanced numerical optimization techniques. The proposed methodology includes a two-parameter controller with separate consideration of the feedforward and the feedback portions. Preliminary results on the input/output part of the design were reported in [7], where a 21-state model was used for the aircraft. Here overall results are reported on

[1]Department of Electrical Engineering and Systems Research Center, University of Maryland, College Park, MD 20742.

[2]School of Electrical Engineering, Georgia Institute of Technology, Altanta, GA 30332

[3] Department of Aerospace Engineering, University of Maryland, College Park, MD 20742

[4]Ames Research Center, Moffett Field, CA 94035

a 37-state model generated by the UM-GenHel [8] nonlinear simulator. Nonlinear validation is also presented.

A central tool in this study was the interactive optimization-based package CONSOLE [9,10]. CONSOLE handles multiple (competing) objective functions, and constraints can be "soft" (moderate violations are allowed) or "hard". Specifications can be "functional", i.e., involve an entire time- or frequency-response rather than a single time or frequency point. For each specification, the designer provides a "good value" and a "bad value" (a "good curve" and a "bad curve" in case of a functional specification) and these can be interactively adjusted as tradeoffs are identified.

In Section 2 below, we briefly describe a simplified model of a rotorcraft in hover (based on the UH-60). In Section 3, a set of design specifications is proposed. In Section 4, the design methodology is outlined. In Section 5, some results are presented, including validation on the nonlinear model. Section 6 is devoted to some final remarks.

2. A Linearized Model for an UH-60 in Hover

The model we used for controller design, denoted by $P_0(s)$, is a 39-state linear model generated from UM-GenHel. UM-GenHel, developed at the Aerospace Engineering Department of the University of Maryland, is a nonlinear flight simulation program for helicopters [8]. It models the high order dynamics (such as main rotor blade motions and main rotor inflow) of the helicopter, and can generate linearized models of the helicopter at various flight conditions. The last six states of this linearized model correspond to hidden modes associated with the engine dynamics. These six states were removed from the model, leaving a 33-state minimal realization of $P_0(s)$. This realization has one pole at zero and a pair of real poles in the open right-half plane. It is non-minimum phase and strictly proper. In each of the four channels, the control system actuator is modeled by a first order Padé approximation corresponding to a 0.050 second delay. This yields a final count of 37 states.

An overall block diagram of the closed-loop system is represented in Figure 1. The four variables of the input $\delta_s = (\delta_{s\phi}, \delta_{s\theta}, \delta_{sc}, \delta_{s\psi})$ of $P_0(s)$ represent respectively the lateral, longitudinal, and collective displacements of the swashplate, and the position of the tail rotor actuator. The output is $y = (u, v, w, p, q, r, \phi, \theta, \psi)$, the variables of which stand for longitudinal velocity (u), lateral velocity (v), vertical velocity (w), roll rate (p), pitch rate (q), yaw rate (r), roll angle (ϕ), pitch angle (θ), and yaw angle (ψ). The delay block $D_e(s)$ has 4 states, 4 inputs, and 4 outputs. The command input $\delta = (\delta_\phi, \delta_\theta, \delta_c, \delta_\psi)$ consists of

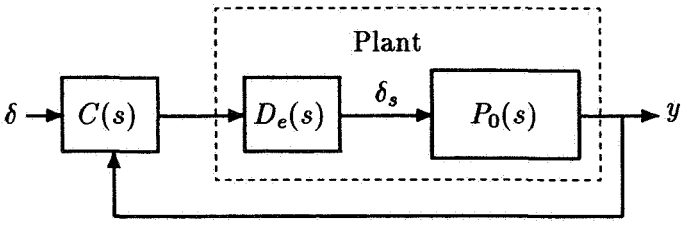


Figure 1: Control system configuration

roll command (δ_ϕ), pitch command (δ_θ), collective command (δ_c), and yaw command (δ_ψ). The rotorcraft is to be controlled by a two-parameter controller $C(s)$.

3. Design Specifications

A wide range of specifications, in both the time- and frequency domains, are to be satisfied. First, the closed-loop system must be internally stable. Second, to the extent possible, it is desired that the various longitudinal and lateral modes be decoupled and approximate specified step responses (handling quality specifications). Specifically, the pitch command should mostly affect the pitch angle (and pitch rate) and the longitudinal velocity; the collective command should mostly affect vertical velocity; the roll command should mostly affect the roll angle (and roll rate) and the lateral velocity; and the yaw command should affect mostly the yaw angle (and yaw rate). The "diagonal" responses should exhibit desirable characteristics as given in [6]. Third, the displacement and rate of displacement of the swashplate may not violate some physical limitations. Lastly, the closed-loop system should exhibit good performance in gust rejection. The mathematical formulation of these specifications is described below. (Note: all inputs are expressed in inches of stick.)

Spec 1 The closed-loop system is internally stable.

Spec 2 When the input $\delta = [0 \;\; 5 \;\; 0 \;\; 0]^T$:

Spec 2.1 Pitch response $\theta(t)$ should lie between the two curves $C_1(t)$ and $C_2(t)$ of Figure 2.

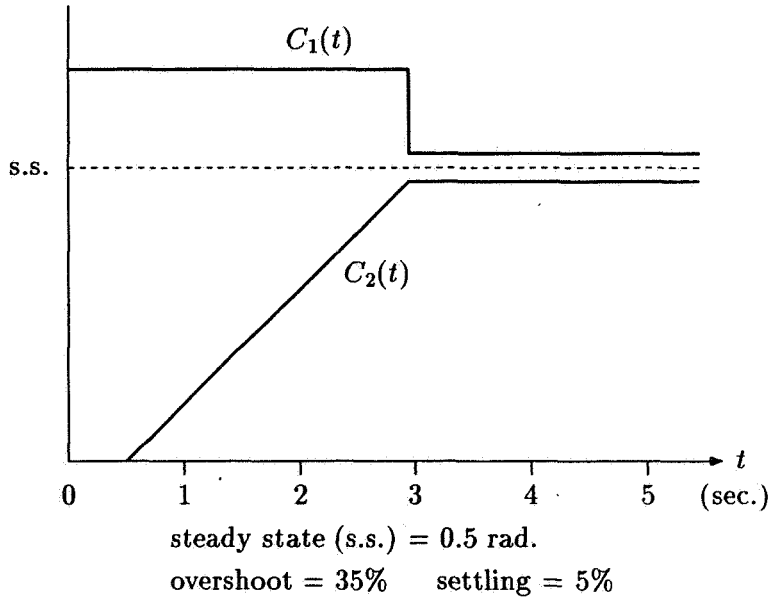


Figure 2: Boundaries for pitch response

Spec 2.2 Decoupling: (i) $\phi(t)$, $\psi(t)$ should have the absolute values less than 5% of $\theta(5)$ for $t \in [0, 5]$; (ii) $v(t)$, $w(t)$ should have the absolute values less than 5% of $u(5)$ for $t \in [0, 5]$.

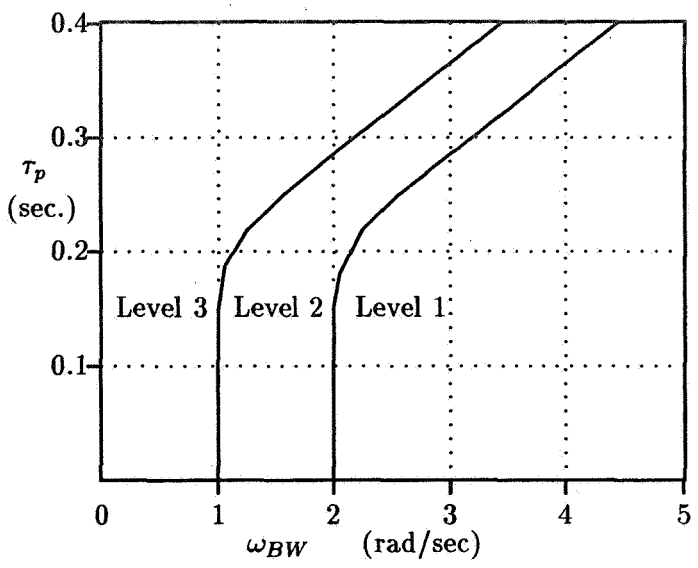


Figure 3: Handling quality specification

Spec 2.3 Physical limitations on swashplate displacements: the absolute values of $\delta_{s\theta}(t) + \delta_{sc}(t), \delta_{s\theta}(t) + \delta_{s\phi}(t), \delta_{s\phi}(t) + \delta_{sc}(t), \delta_{s\phi}(t), \delta_{s\theta}(t), \delta_{sc}(t)$, and $\delta_{s\psi}(t)$ should all be less than 10 inches for $t \in [0, 5]$.

Spec 2.4 Limitations on the velocities of the swashplate motion: the absolute values of $\dot{\delta}_{s\theta}(t) + \dot{\delta}_{sc}(t), \dot{\delta}_{s\theta}(t) + \dot{\delta}_{s\phi}(t), \dot{\delta}_{s\phi}(t) + \dot{\delta}_{sc}(t), \dot{\delta}_{s\phi}(t), \dot{\delta}_{s\theta}(t), \dot{\delta}_{sc}(t)$, and $\dot{\delta}_{s\theta}(t)$ should all be less than 10 inches per second for $t \in [0, 5]$.

Spec 3 Let H_θ be the transfer function from δ_θ to θ. Suppose ω_{BW} (bandwidth) and τ_p (phase delay) are defined based on the Bode plot (phase part) of H_θ, such that ω_{BW} is the frequency corresponding to 45 degree phase margin, and phase delay τ_p is defined as

$$\tau_p = -\frac{\Phi_{2\omega_{180}} + 180}{\frac{360}{2\pi} \times 2\omega_{180}}$$

where ω_{180} is the frequency at which the phase angle attains -180 degree, and $\Phi_{2\omega_{180}}$ is the phase angle at frequency $2\omega_{180}$ (see [4]). Then the graph of (ω_{BW}, τ_p) should lie within region Level 1 in Figure 3.

Spec 4 When the input $\delta = [5 \ \ 0 \ \ 0 \ \ 0]^T$:

Spec 4.1 Roll response $\phi(t)$ should lie between the two curves $C_1(t)$ and $C_2(t)$ of Figure 2, with s.s.= 1 rad.

Spec 4.2 Decoupling: (i) $\theta(t)$, $\psi(t)$ should have the absolute values less than 5% of $\phi(5)$ for $t \in [0, 5]$; (ii) $u(t)$, $w(t)$ should have the absolute values less than 5% of $v(5)$ for $t \in [0, 5]$.

Spec 4.3 Physical limitations on swashplate displacements: same as spec 2.3.

Spec 4.4 Limitations on the speed of swashplate motion: same as spec 2.4.

Spec 5 Same as Spec 3 with H_θ replaced by H_ϕ, the transfer function from δ_ϕ to ϕ.

Spec 6 When input = [0 0 0 2.5] :
Spec 6.1 Yaw rate $r(t)$ should lie between the two curves $C_1(t)$ and $C_2(t)$ of Figure 2, with s.s.= 0.4 rad/sec.
Spec 6.2 Decoupling: $p(t)$, $q(t)$ should have the absolute values less than 5% of $r(5)$ for $t \in [0, 5]$.
Spec 6.3 Physical limitations on swashplate displacements: same as spec 2.3.
Spec 6.4 Limitations on the speed of swashplate motion: same as spec 2.4.

Spec 7 Same as Spec 3 with H_θ replaced by H_ψ, the transfer function from δ_ψ to ψ.

Spec 8 When input = [0 0 5 0] :
Spec 8.1 Vertical velocity $w(t)$ should satisfy

$$\mathcal{L}^{-1}\left[\frac{5Ke^{-\tau s}}{s(Ts+1)}\right](t) \le w(t) \le 1.1\,\mathcal{L}^{-1}\left[\frac{5Ke^{-\tau s}}{s(Ts+1)}\right](t) + 2$$

for all $t \in [0, 5]$, where $K = 10$, $\tau = 0.2$, $T = 5$, and $\mathcal{L}^{-1}$ denotes the inverse Laplace transform.
Spec 8.2 Decoupling: $u(t)$, $v(t)$ should have the absolute values less than 5% of $w(5)$ for $t \in [0, 5]$.
Spec 8.3 Physical limitations on swashplate displacements: same as spec 2.3.
Spec 8.4 Limitations on the speed of swashplate motion: same as spec 2.4.

Spec 9 A wind gust, which is modeled by the step response of

$$g(s) = \frac{0.44s}{(s+1)^2}, \qquad \text{(gust model)} \tag{1}$$

is to be injected at the output nodes ϕ, θ, and ψ in turn.
Spec 9.1 The closed-loop transient response at the corresponding node, when the wind gust is injected, should be kept within the envelope of the step response of

$$h_{\text{allow}}(s) = \frac{0.155s}{s^2 + 1.77s + 0.462}. \tag{2}$$

Step responses of $g(s)$ and $h_{\text{allow}}(s)$ are shown in Figure 4.
Spec 9.2 Physical limitations on swashplate displacements, same as spec 2.3, but time interval is changed to $[0, \infty)$.
Spec 9.3 Limitations on the speed of swashplate motion, same as spec 2.4, but time interval is changed to $[0, \infty)$.

We remark that, given the highly coupled dynamics of the plant ($P_0(s)$), these specifications are very stringent, and to our knowledge no controller has been built so far that satisfies all these specifications.

The asymmetry of the rotorcraft places fundamental limitations on the amount of decoupling that can be achieved between the various channels. The most striking instance is the impossibility of maintaining a constant nonzero pitch angle without steady state banking. For the linear model used in this study, it can be verified (see [11,12] for details) that the minimum achievable ratio between steady state roll and pitch angles is slightly over

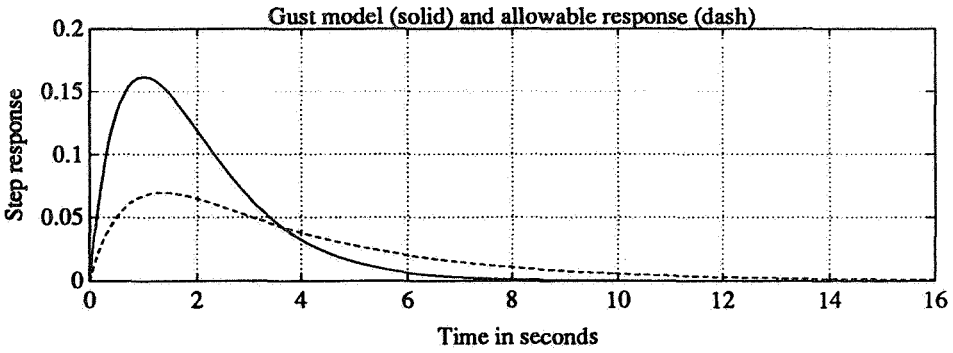


Figure 4: Gust model and allowable closed-loop response

20%. Note however that this does not preclude the possibility of meeting specification 2.2 as it requires significant decoupling over the first 5 seconds only. Yet, this limitation ought to be kept in mind when interactively exploring possible tradeoff designs.

4. Methodology

Design of a two-parameter controller proceeds as follows [13]. First the overall aircraft transfer matrix (rotor + airframe dynamics + delay) $P(s) = P_0(s)D_e(s)$ is factorized over the ring of stable transfer functions, i.e.,

$$P(s) = N(s)D^{-1}(s) = \tilde{D}^{-1}(s)\tilde{N}(s)$$

where $N(s)$, $D(s)$, $\tilde{N}(s)$ and $\tilde{D}(s)$ are stable transfer matrices, $(N(s), D(s))$ are right coprime and $(\tilde{N}(s), \tilde{D}(s))$ are left coprime. Next let $X(s)$ and $Y(s)$ be stable transfer matrices satisfying the Bezout identity

$$X(s)N(s) + Y(s)D(s) = I$$

Then all stabilizing controllers can be obtained according to the block diagram of Figure 5 where both $Q(s)$ and $R(s)$ range over the class of all stable transfer matrices of appropriate dimension (with the condition that $\det(Y(s) - R(s)\tilde{N}(s))$ does not vanish identically).

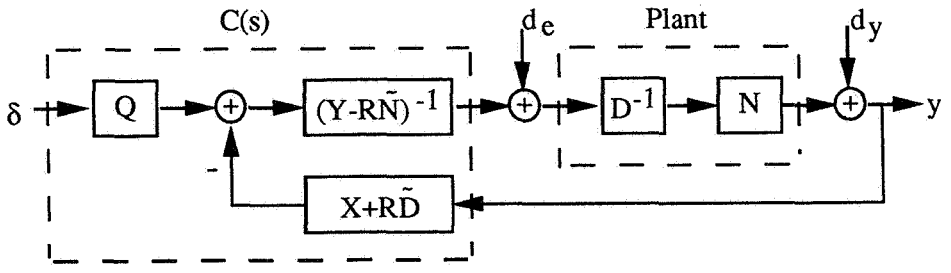


Figure 5: Two-parameter controller

It is easily verified that the overall transfer function $T(s)$ from δ to y is given by

$$T(s) = N(s)Q(s) \tag{4.1}$$

while the transfer function from δ to the swashplate displacements δ_s by

$$S(s) = D_e(s)D(s)Q(s) \tag{4.2}$$

Thus both are independent of the R parameter. On the other hand the transfer function from disturbance inputs d_e and d_y to those same outputs y and δ_s is given by

$$\begin{bmatrix} D(s)(Y(s) - R(s)\tilde{N}(s)) & -D(s)(X(s) + R(s)\tilde{D}(s)) \\ N(s)(Y(s) - R(s)\tilde{N}(s)) & -N(s)(X(s) + R(s)\tilde{D}(s)) + I \end{bmatrix}$$

which is independent of the Q parameter. By adopting the two-parameter controller structure, the control designer can then separate the design problem into two independent subproblems of input/output performance (to be solved by choosing a "good" Q: Q-design) and of disturbance rejection (to be solved by choosing a "good" R: R-design), respectively. At the same time, internal stability will be automatically guaranteed if the chosen $Q(s)$ and $R(s)$ are stable.

Q-design

We just saw that any stable $Q(s)$ yields a stable $T(s)$ and any stable $T(s)$ corresponds to a stable $Q(s)$, so that this "Q-parameterization" automatically takes care of specification 1. Concerning the other input/output specifications, the structure of the problem allows another major simplification. Note that each one of the specifications 2-8 involved only one of the four input channels and (see (4.1-2)) each column of $Q(s)$ affects the I/O transfer function corresponding to exactly one input channel (i.e., exactly one column of $Q(s)$). As a result the problem can be decomposed into four optimization problems, each one involving a single column of $Q(s)$ and the specifications involving inputs corresponding to that column. Thus, interactive optimization will be used, for each input channel, to try to satisfy the corresponding input/output specifications; this will involve a single column of $Q(s)$, which we will denote by $q(s)$.

A possible parametrization for $q(s)$, once its McMillan degree n has been chosen, is by writing

$$q(s) = C(sI - A)^{-1}B + D \tag{4.3}$$

with

$$A = \begin{bmatrix} 0 & 1 & & & & & \\ p_1 & p_2 & & & & & \\ & & 0 & 1 & & & \\ & & p_3 & p_4 & & & \\ & & & & \ddots & & \\ & & & & & 0 & 1 \\ & & & & & p_{2j-1} & p_{2j} \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix} \tag{4.4a}$$

if $n = 2j$ is even, and

$$A = \begin{bmatrix} 0 & 1 & & & & \\ p_1 & p_2 & & & & \\ & & \ddots & & & \\ & & & 0 & 1 & \\ & & & p_{2j-1} & p_{2j} & \\ & & & & & p_{2j+1} \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \tag{4.4b}$$

if $n = 2j + 1$ is odd, and using the p_i's as well as all entries of C and D as free parameters. Indeed, it is clear that by constraining the p_i's to be negative, only stable $Q(s)$ will be generated; it is also easy to show that the family of matrices $Q(s)$ thus generated is open and dense in the set of stable matrices of McMillan degree n.

One difficulty with the parametrization just proposed is the ensuing large number of design parameters, resulting in (i) large computation times and (ii) likeliness that a local rather than global optimum be reached. The following heuristics drastically reduces the number of design parameters by eliminating a large portion of "clearly nonoptimal" $q(s)$ and likely retaining enough of the "good" ones to allow a "close to optimal" $q(s)$ to be identified. For given A and B (i.e., given p_i's) and given "sampling frequencies" $\omega_1, \ldots, \omega_\ell$ (the p_i's and the ω_i's will be the design parameters) corresponding C and D are selected by solving, in a least square sense,

$$N(j\omega_i)(C(j\omega_i I - A)^{-1}B + D) = T(j\omega_i), i = 1, \ldots, \ell \tag{4.5}$$

where $T(s)$ is an approximate "desired" transfer function constructed from the specifications: good "diagonal" responses and "off diagonal" responses set to zero. (The method that was actually used is somewhat more sophisticated in that it involves solving two successive linear least squares problems: see [11,12].)

To summarize, the numerical optimization process will proceed as follows. Given values of the p_i's and ω_i's (design parameters), $q(s)$ will be obtained by solving for C and D the linear least square problem; time and frequency responses of the closed-loop system will be computed (by invoking MATLAB); the values of the specifications and their gradients (with respect to the design parameters) will be determined; the optimization algorithm (the heart of CONSOLE) will use this information to select new values of the design parameters; and the cycle will start again until the designer decides to interrupt it. The designer can then relax or tighten selected specifications in order to explore various tradeoffs.

In addition to the specifications outlined in Section 3, it may be desired to achieve or approach certain steady state values when unit steps are fed into individual channels. In view of the Final Value theorem, these values are determined by $Q(0)$. The best value Q_* for $Q(0)$ can be determined, again, by solving a linear least squares problem. Let q_* be a column of Q_*. Then, in view of (4.3), we may require that

$$C(-A)^{-1}B + D = q_*,$$

i.e.,

$$D = q_* + CA^{-1}B.$$

Substituting this expression for D in (4.5) we obtain a linear least squares problem in C only. Note that, in the design results reported below, there was no such steady state requirement.

R-design

A major difference with the Q-design is that the columns of $R(s)$ cannot be designed independently, as each of these columns may affect all disturbance rejection specifications. Thus a parametrization of the form (4.3-4), where C and D are obtained by solving a linear system of the form (4.6) in the least squares sense, is used for the entire matrix $R(s)$.

5. Results and Validation

Based on our Q-design results, it is found that in the lateral, longitudinal, and tail collective channels, the swashplate rate specifications (Specs 2.4, 4.4, 6.4) are competing with the handling quality (based on frequency response) specifications (Specs 3, 5, 7). Specifically, for the handling quality indicator (ω_{BW}, τ_p) to lie within region Level 1 in Figure 3, the swashplate rates will attain magnitude above 10 inch/sec within the first 0.1 second. For example, Figure 6 and Figure 7 show the results of a design of the second column of Q (the longitudinal channel) on Specs 2.4 and 3. For this design, Spec 2.4 is satisfied but Spec 3 is not. Figure 8 and Figure 9 show the results of another design of the same column of Q on Specs 2.4 and 3. For this design, Spec 3 is satisfied but Spec 2.4 is not.

Our R-design results are that Specs 9.2 and 9.3 are satisfied, but Spec 9.1 is not. Figure 10 shows the response of the roll angle $\phi(t)$ (or, respectively, the pitch angle $\theta(t)$, or the yaw angle $\psi(t)$) when a wind-gust, modeled by the step response of $0.44s/(s+1)^2$, is injected at the roll angle node (or, respectively, the pitch angle node or the yaw angle node). From the figure, one sees that the graphs of the response do not lie within the allowable envelope given in Spec 9.1. Hence Spec 9.1 is not satisfied. The figure also suggests that the helicopter cannot act fast enough to respond to the gust, and can counteract the gust only after 0.3 second.

We finally choose a design of the controller for which the specifications on swashplate rates (Specs 2.4, 4.4, and 6.4) are satisfied but the handling quality specifications (Specs 3, 5, and 7) are not. This controller has 152 states. It is then reduced to 82 states via the balanced realization method. It has been examined, by checking against the specifications, that the performance of the order-reduced controller on the linearized model is as good as the unreduced, 152-state controller. This order-reduced controller is tested on both the UM-GENHEL linear and nonlinear simulation program. Based on the comparison between the open-loop time responses (to a step input) of the nonlinear and the linearized model, it is found that the nonlinearity of the helicopter model is very high.

Figure 11 shows some of the relevant time responses of the closed-loop (linear and nonlinear) systems, when a 5 unit (in terms of inches of stick command) step command input is put on the lateral channel (δ_ϕ) of the controller. One notices from this figure that, after 2.5 seconds, the closed-loop responses of the nonlinear model do not match those of the linear model very well. However, given the high nonlinearity of the dynamics of the nonlinear model, this phenomenon should be expected. For example, Figure 11 shows that at 2.5 seconds after the control command has been issued, the roll angle of the nonlinear model is about 1 radian. This flight condition is very different from hover, and hence the dynamics of the nonlinear model at this point may be very different from that of the

linearized model for hover.

One would expect the discrepancies between the response of the closed-loop nonlinear system and that of the linear system should be less, if the magnitude of the control input is lowered. Figure 12 gives the comparison between the relevant responses of the nonlinear and linearized model, when the magnitude of the command input is lowered to 1. This clearly confirms that the response of the nonlinear model matches better with that of the linear model than before.

Finally, closed-loop responses of both the nonlinear and linear model, due to gust disturbance, are examined. For example, Figure 13 shows the roll angle response when gust is injected at the roll angle node. Notice that there are some discrepacies between the roll angle responses of the linear and the nonlinear model.

6. Discussion

The controller discussed in Section 5 is but one of the many sub-optimal controllers obtained when running CONSOLE. Indeed, a key advantage of an interactive package such as CONSOLE is that it allows the designer to explore alternative solutions by fine tuning the various target responses (this is especially so with CONSOLE's not yet released graphical interface).

It turns out that even the stability specification is amenable to tuning. Indeed, as discussed in [13], the Q-parametrization approach can be extended to generalized stability, i.e., confinement of closed loop poles in a more general region Π of the complex plane. This can be achieved by (i) factoring $P(s)$ in the ring of Π-stable (rather than Hurwitz stable) transfer functions and (ii) take for $Q(s)$ and $R(s)$ any matrices with all their poles in Π. The latter can be accomplished by suitably modifying parameterization (4.4). E.g., if Π is as in Figure 14, it suffices to replace in (4.4) 2×2 blocks of the form

$$\begin{bmatrix} 0 & 1 \\ p_1 & p_2 \end{bmatrix}$$

by

$$\begin{bmatrix} 0 & 1 \\ (1-y)(b-x)^2 - x^2(yk^2+1) & 2x \end{bmatrix}$$

and let x and y vary over $[(a+b)/2, b]$ and $[0,1]$, respectively (see [11,12] for details).

Finally, the merits of the approach discussed in this paper should be compared to those of the convex optimization approach proposed by S.P. Boyd and C.H. Barratt [14]. The main advantage of the latter is that it always yields the globally optimal design for the given specifications (for $Q(s)$ and $R(s)$ ranging over a finite dimensional subspace of the space of stable transfer matrices). A crucial requirement, however, is that all specifications be convex (as functions of the closed loop transfer matrix). It turns out that some of the specifications considered in the present study (in particular handling quality specifications such as Spec 3) do not satisfy this requirement (see [11,12] for details). Compared to the Ritz parameterization used in [14], while parameterization (4.3-4) would destroy any existing convexity, it has the advantage to cover all (in fact, "almost all") stabilizing controllers of a given degree.

Acknowledgement

This work was supported in part by NASA-Ames University Consortium Interchange No. NCA2-309, NSF Engineering Research Center Program No. NSFD-CDR-88-03012, and by NSF Grant No. DMC-84-51515.

References

[1] R.D. Holdridge, W.S. Hindson & A.E. Bryson, "LQG-Design and Flight Test of a Velocity-Command System for a Helicopter," *AIAA Guidance and Control Conference* (1985).

[2] W.L. Garrard & B.S. Liebst, "Design of a Multivariable Helicopter Flight Control System for Handling Qualities Enhancement," *Proceedings of the 43rd Annual Forum, American Helicopter Society*, Alexandria, VA (1987).

[3] D.F. Enns, "Multivariable Flight Control for an Attack Helicopter," *IEEE Control Systems Magazine* (April 1987).

[4] M.B. Tischler, "Digital Control of Highly Augmented Combat Rotorcraft ," NASA TM 88346, 1987.

[5] D.G. Miller & F. White, "A Treatment of the Impact of Rotor-Fuselage Coupling on the Helicopter Handling Qualities," *43rd Annual National Forum of the American Helicopter Society* (1987).

[6] R.H. Hoh, D.G. Mitchell & B.L. Aponso, "Proposed Specification for Handling Qualities of Military Rotorcraft. Vol. 1 – Requirements," USA AVSCOM Technical Report 89-A-4, May 1988.

[7] M.K.H. Fan, A.L. Tits, J. Barlow, N.K. Tsing, M. Tischler & M. Takahashi, "On the Design of Decoupling Controllers for Advanced Rotorcraft in the Hover Case," paper AIAA-91-0421, presented at the AIAA 29th Aerospace Sciences Meeting, Reno, Nevada, January 7-10, 1991.

[8] F.D. Kim, R. Celi & M.B. Tischler, "High-Order State-Space Simulation Models of Helicopter Flight Mechanics," *Proceedings of the 16th European Rotorcraft Forum*, Glasgow, UK (1990).

[9] M.K.H. Fan, L.-S. Wang, J. Koninckx & A.L. Tits, "Software Package for Optimization-Based Design with User-Supplied Simulators," *IEEE Control System Magazine* 9 (1989).

[10] M.K.H. Fan, A.L. Tits, J.L. Zhou, L.S. Wang & J. Koninckx, "CONSOL-OPTCAD User's Manual (Version 1.1, Released 9/1990)," Systems Research Center, University of Maryland, Technical Report TR-87-212r3, College Park, Maryland 20742, 1991.

[11] N.-K. Tsing, *Computer-Based Techniques for Control System Design, with Applications to Rotorcraft Control*, Ph.D. Dissertation, Dept. of Electrical Engineering, University of Maryland, College Park, 1992.

[12] "Rotorcraft Flight Control System Methodologies (part II)," Final report for NASA-Ames University Consortium Interchange #NCA2-510, 1992.

[13] M. Vidyasagar, *Control System Synthesis – A Factorization Approach*, MIT Press, 1985.

[14] S.P. Boyd & C.H. Barratt, *Linear Controller Design: Limits of Performance*, Prentice-Hall, Englewood Cliffs, NJ, 1991.

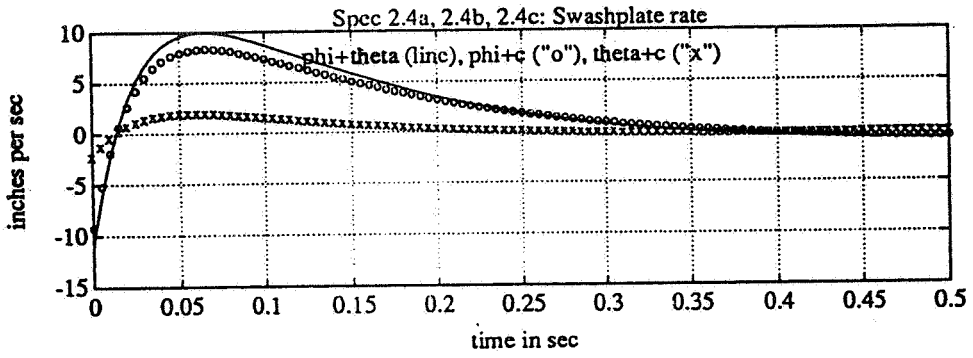


Figure 6. Pitch command: swashplate rates

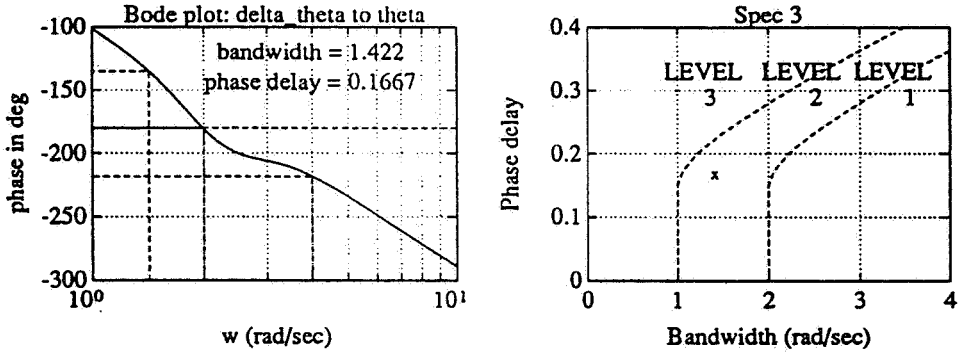


Figure 7. Bandwidth and phase delay for pitch channel

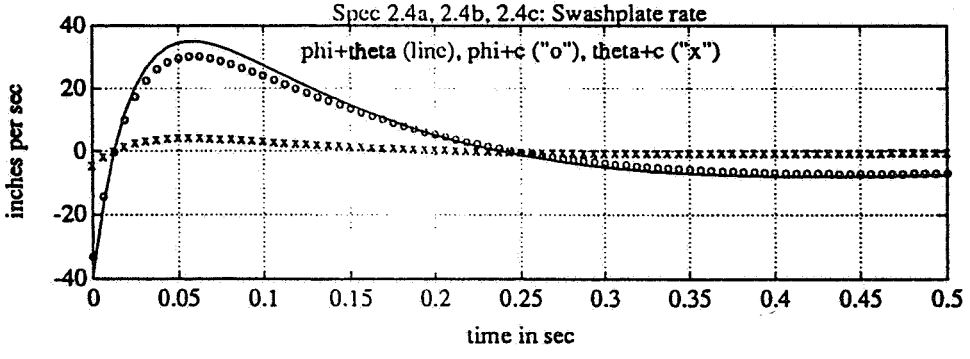


Figure 8. Pitch command: swashplate rates (alternate design)

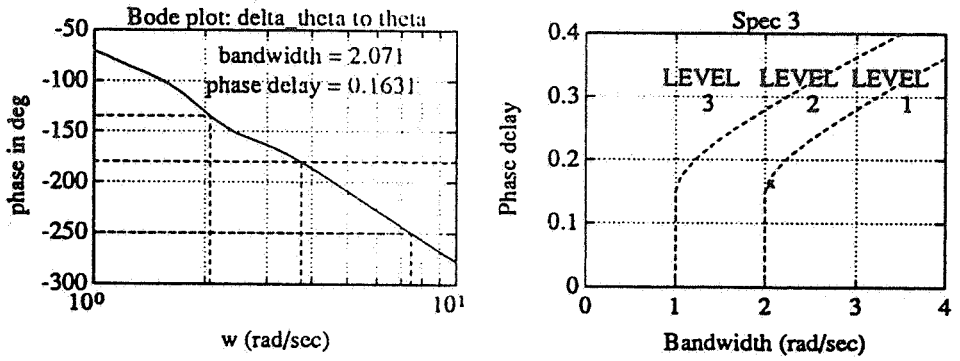


Figure 9. Bandwidth and phase delay for pitch command (alternate design)

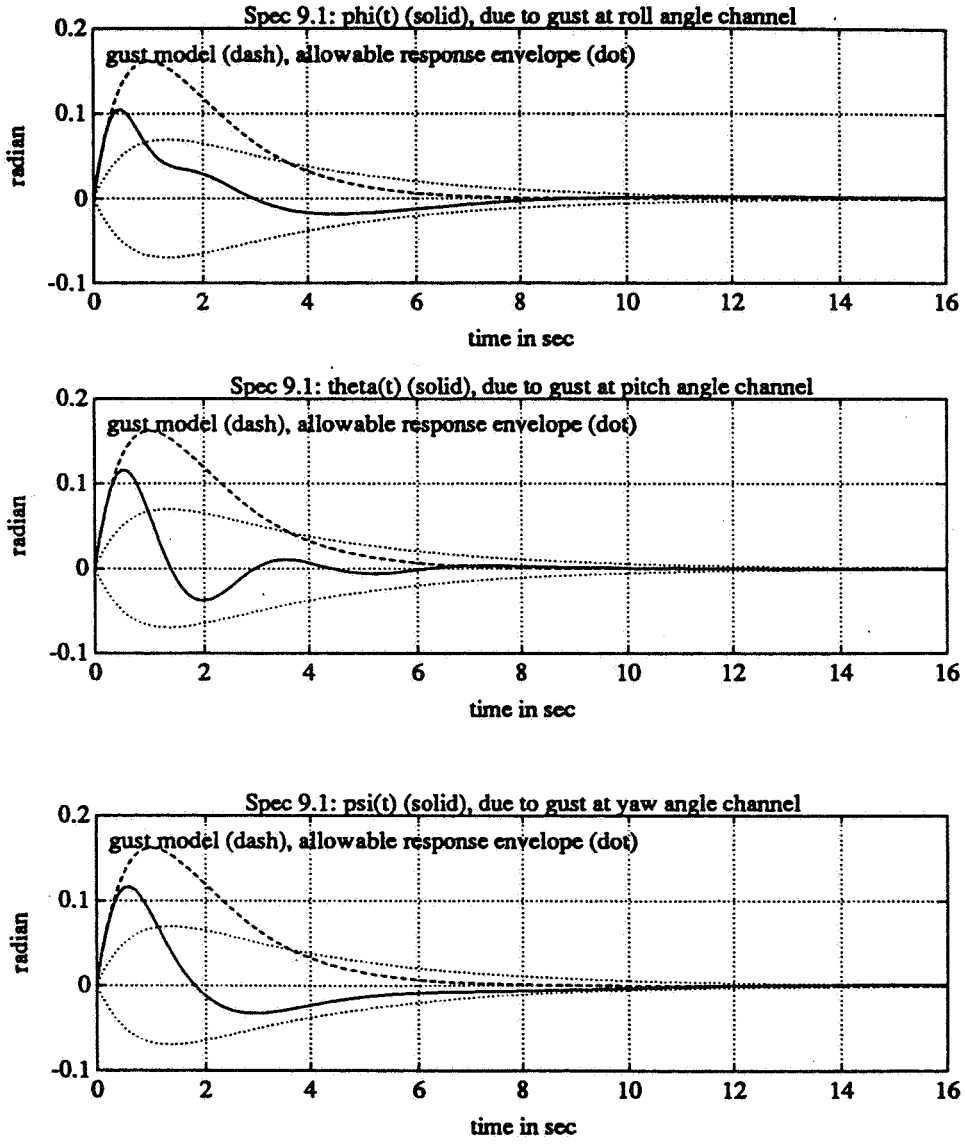


Figure 10. Time-responses to wind gust

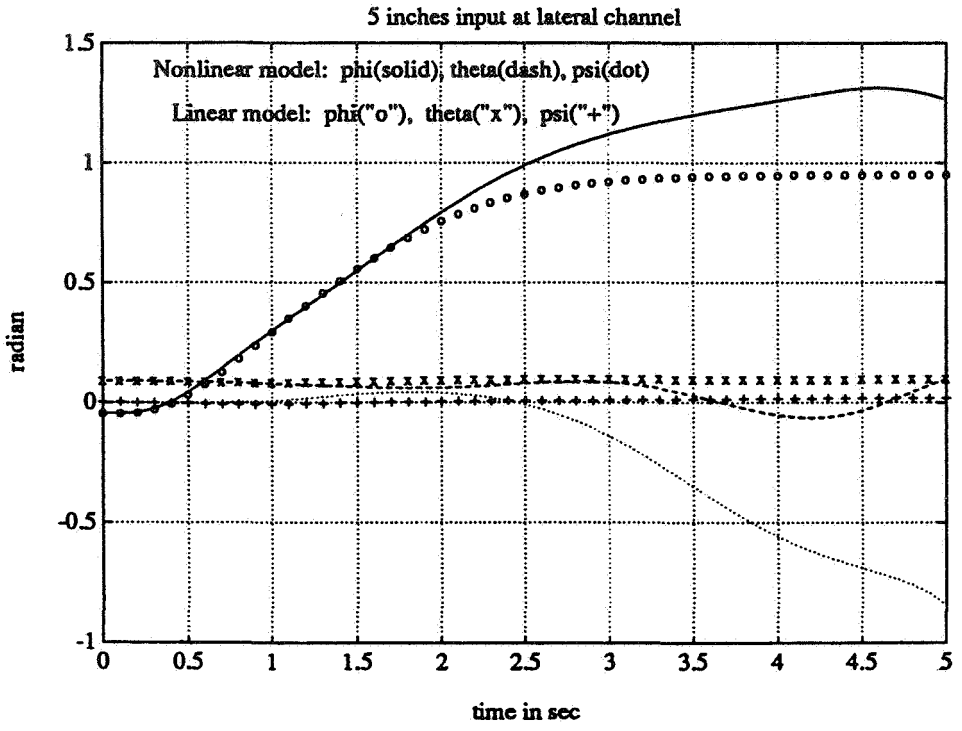


Figure 11. Nonlinear validation: large signals

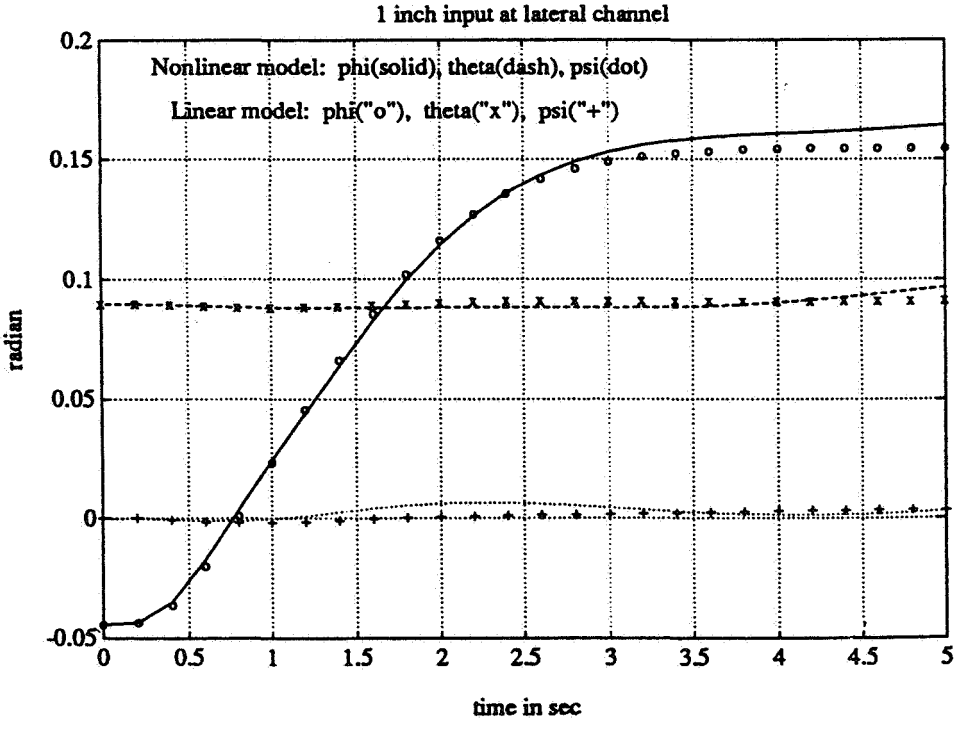


Figure 12. Nonlinear validation: average size signals

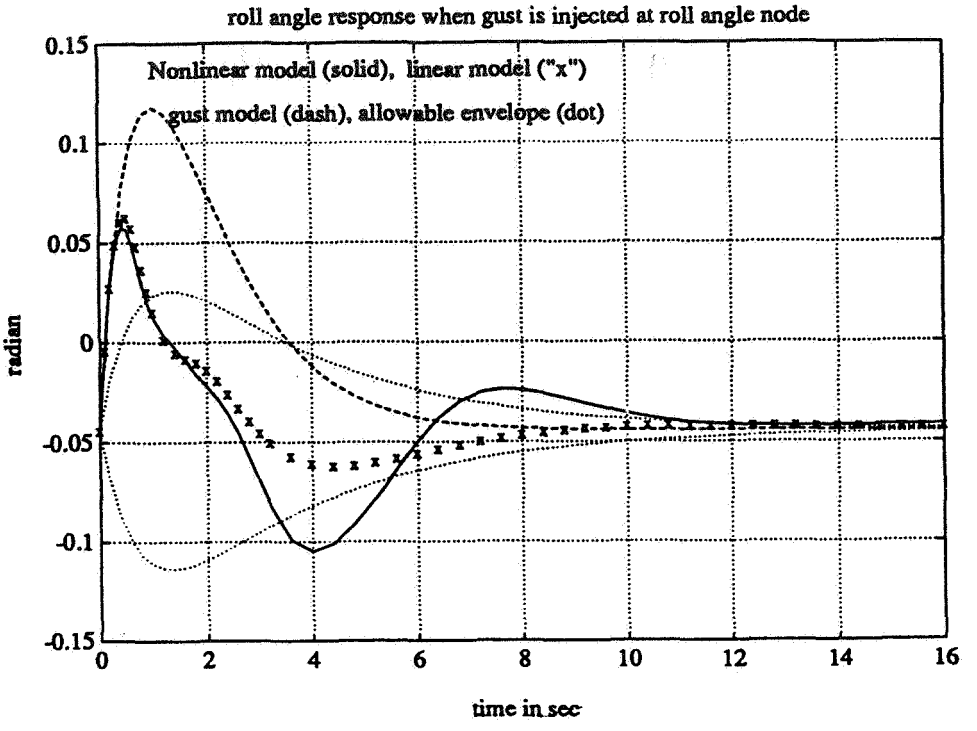


Figure 13. Nonlinear validation: wind gust rejection

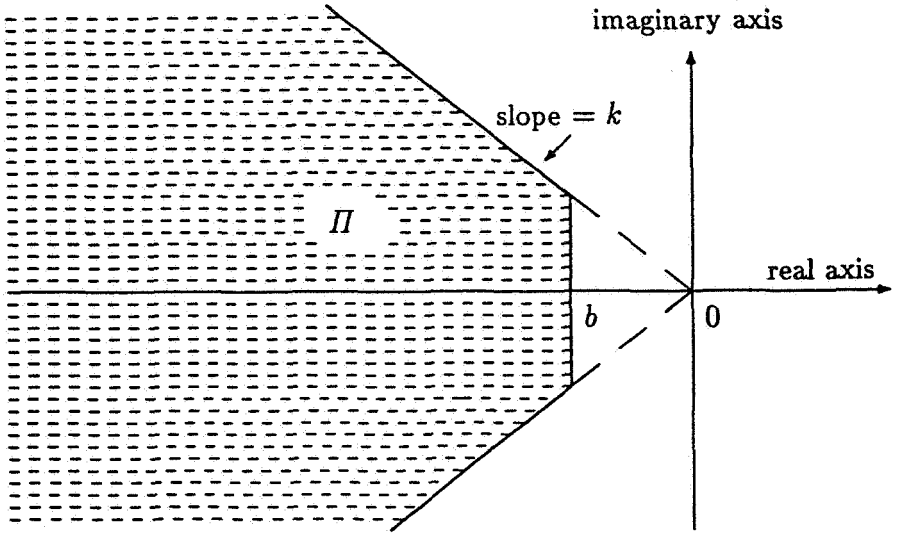


Figure 14. Π-stability region

On l^1 - Optimal Decentralized Performance

Dennis Sourlas, Vasilios Manousiouthakis *
5531 Boelter Hall, Chemical Engineering Department
University of California, Los Angeles, CA 90024

Abstract: In this paper, the Manousiouthakis parametrization of all decentralized stabilizing controllers is employed in mathematically formulating the l^1 optimal decentralized controller synthesis problem. The resulting optimization problem is infinite dimensional and therefore not directly amenable to computations. It is shown that finite dimensional optimization problems that have value arbitrarily close to the infinite dimensional one can be constructed. Based on this result, an algorithm that solves the l^1 decentralized performance problems is presented. A global optimization approach to the solution of the finite dimensional approximating problems is also discussed.

*Author to whom correspondence should be addressed. Tel (310)-825 9385, FAX (310)-206 4107.

1. Introduction

Consider a feedback control loop with its inputs and its outputs partitioned in a compatible way: $u_1=(u_{11}^T,u_{12}^T,\cdots,u_{1n}^T)^T$, $u_2=(u_{21}^T,u_{22}^T,\cdots,u_{2n}^T)^T$, $y_1=(y_{11}^T,y_{12}^T,\cdots,y_{1n}^T)^T$, and $y_2=(y_{21}^T,y_{22}^T,\cdots,y_{2n}^T)^T$. The controller C is decentralized iff it is block diagonal i.e. the $i-th$ subvector, y_{1i}, of the manipulated variable vector is only affected by the $i-th$ subvector, y_{2i}, of the measured output:

$$y_{1i}=C_{ii}(u_{1i}-y_{2i})$$

Since the early 70's significant research effort has been expended on the subject of decentralized control. Nevertheless, two unanswered questions remain :

(a) given the set of measurements and manipulations, how does one select the appropriate pairings ?

(b) How can one assess fundamental limitations to decentralized control system performance ?

The first question is referred to as the decentralized control structure synthesis problem while the second can be unequivocally addressed only through the optimal decentralized controller synthesis problem. Given the set of measurements and manipulations, the solution of the decentralized controller structure synthesis problem determines the flow of information in the control loop, or equivalently the pairings between the measurements and the manipulations. The solution to the second problem determines the best achievable closed-loop dynamic performance for the given decentralized control structure.

It has been established, that given a plant and a decentralized structure there may not exist any decentralized stabilizing controllers with that structure. Aoki (1972) [2], demonstrated that there may exist decentralized control structures that prevent stabilization of the closed loop. Wang & Davison (1973) [16], introduced the notion of *decentralized fixed eigenvalues* also called as *fixed modes* of a given system. Algebraic characterizations of the notion of decentralized controllability, which is related to the fixed mode concept, for the two input vector case are given in Morse (1973) [11], Corfmat & Morse (1976) [3], [4], and Potter, Anderson & Morse (1979) [12]. Anderson & Clements (1981) [1], employed algebraic concepts and characterized the decentralized fixed eigenvalues of a system and presented computational tests for the existence of fixed modes.

Recently, the issue of stability of decentralized control systems has been addressed within the fractional representation approach to control theory. For linear time invariant processes, Manousiouthakis (1989) [9] presented a parametrization of all decentralized stabilizing controllers for a given process and a fixed decentralized control structure. Within this framework, any decentralized stabilizing controller is parametrized in terms of a stable transfer function matrix that has to satisfy a finite number of quadratic equality constraints. For the same class of processes (LTI plants) Desoer and Gundes presented an equivalent parametrization where the stable parameter satisfies a unimodularity condition (Desoer & Gundes, 1990, p.122, 165) [7].

In this paper, the Manousiouthakis parametrization is employed in mathematically formulating the optimal controller synthesis problem. The decentralized performance problem is formulated as an infinite dimensional l^1 optimization problem. Performing appropriate truncations a finite dimensional optimization problem is obtained. Theorems that establish the connection between the two problems are presented. It is shown that iterative solution of the finite dimensional problem creates a sequence of values that converges to the values of the infinite

dimensional problem. Based on these conergence results a computational procedure that yields ε-optimal solutions to l^1 optimization problem is outlined. Locally optimal solutions to the intermediate finite dimensional problems can be obtained through existing nonlinear optimization algorithms (MINOS, GINO etc.). Global solution of the intermediate finite dimensional approximations guarantees that the limit of the sequence that is being created corresponds to the best performance that can be obtained by the given decentralized structure. Feasibility (or infeasibility) of the optimization problem is equivalent to existence (or nonexistence) of decentralized controllers with the given structure.

2. Mathematical Preliminaries

2.1. Fractional Representations of Linear systems

Let G be the set of all proper, rational transfer functions and $M(G)$ be the set of matrices with entries that belong to G. Also let S be the set that includes only the stable members of G and let $M(S)$ be the set of matrices with entries that belong to S.

In this work, theoretical results related to the notion of *doubly coprime fractional representations* and the *parametrization of all stabilizing compensators* are used. These results and a complete exposition of the underlying theory can be found in Vidyasagar (1985; pp. 79, 83, 108, 110) [14]. The notation used in the present work is compatible with the notation in the aforementioned reference.

2.2. Input - Output Linear Operators

One of the frameworks developed to describe the stability and performance of dynamical systems is the input - output approach. Although the theory has been developed for both continuous and discrete systems, in this work the focus is on discrete systems.

In the sequel the fact that every linear BIBO operator can be represented by an l^1 sequence will be utilized. For such operators the $l^\infty - l^\infty$ induced norm is equal to the l^1 norm of the corresponding l^1 sequence. The results that are used can be found in Desoer and Vidyasagar (1975; pp. 23-24, 100, 239) [5].

2.3. Elements from Real and Functional Analysis

The notion of denseness will be used in the proofs in Section 4. The fact that the set ϕ_o of all sequences with finitely many nonzero elements is dense in l^1 will also be used. Properties of the compact sets will be used in Lemma 2 in Section 4. The related theory is given by Wheeden & Zygmud (1977, pp. 4, 8-9, 134) [17].

The properties of point-to-set mappings are also used. All relevant results can be found in Fiacco (1983; pp. 12, 14) [6].

3. Parametrization of Decentralized Stabilizing Controllers

In this section, the results presented by Manousiouthakis (1989) [9] are outlined. The 2-channel case is outlined in section 3.1. In section 3.2, the result corresponing to the general l-channel case is presented.

3.1. 2-Channel Decentralized Control

Consider a feedback control system shown with plant P and controller C:

$$C=\begin{bmatrix} C_{1,1} & C_{1,1} \\ C_{2,1} & C_{2,1} \end{bmatrix} \in G^{n\times m}\ ,\ C_{1,1}\in G^{n_1\times m_1}\ ,\ C_{2,2}\in G^{(n-n_1)\times(m-m_1)}\ ,$$

$$P=\begin{bmatrix} P_{1,1} & P_{1,1} \\ P_{2,1} & P_{2,1} \end{bmatrix} \in G^{m\times n}\ ,\ P_{1,1}\in G^{m_1\times n_1}\ ,\ P_{2,2}\in G^{(m-m_1)\times(n-n_1)}$$

Manousiouthakis (1989) [9], demonstrated that based on the YJB parametrization of all stabilizing compensators for a given plant, the set of all 2-channel decentralized stabilizing compensators for the given plant can be described as:

$$S_d(P)=\Big\{ C=(\tilde{X}+D_P Q)(\tilde{Y}-N_P Q)^{-1}\ ,\ \det(\tilde{Y}-N_P Q)\neq 0\ ,\ Q\in M(S)\ ;\ S_1+Q S_2+S_3 Q+Q S_4 Q=0 \Big\} \tag{1}$$

where,

$$\left.\begin{aligned} S_1&=Y L_{n_1}\tilde{X}-X L_{m_1}\tilde{Y} & S_3&=Y L_{n_1} D_P+X L_{m_1} N_P \\ S_2&=\tilde{N}_P L_{n_1}\tilde{X}+\tilde{D}_P L_{m_1}\tilde{Y} & S_4&=\tilde{N}_P L_{n_1} D_P-\tilde{D}_P L_{m_1} N_P \end{aligned}\right\} \tag{1a}$$

and

$$L_{n_1}=\begin{bmatrix} I_{n_1} & 0 \\ 0 & -I_{n-n_1} \end{bmatrix}=L_{n_1}^{-1}\ ,\ L_{m_1}=\begin{bmatrix} I_{m_1} & 0 \\ 0 & -I_{m-m_1} \end{bmatrix}=L_{m_1}^{-1}$$

3.2. l-Channel Decentralized Control

The parametrization of all $l\times l$ block diagonal stabilizing controllers is based on the results of the previous section, namely relations (1), and (1a). It has been established that the set of all l-channel decentralized stabilizing controllers can be parametrized as (Manousiouthakis, 1989) [9]:

$$S_d(P)=\Big\{ C=(\tilde{X}+D_P Q)(\tilde{Y}-N_P Q)^{-1}\ ,\ \det(\tilde{Y}-N_P Q)\neq 0\ ,\ Q\in M(S)\ ;\ S_{1j}+Q S_{2j}+S_{3j} Q+Q S_{4j} Q=0\ ,\ j=1,\cdots,l-1 \Big\} \tag{4}$$

The transfer matrices S_{ij} , $i=1,2,3,4$, $j=1,\cdots,l-1$ are given by relations similar to (1a).

4. Optimal Decentralized Control System Performance

4.1. l^1 Performance Results

The performance problem is often posed as follows : *determine whether the output of the system remains within specified bounds for all bounded external disturbances and for all times.* The idea of considering disturbances bounded in magnituded was introduced by Vidyasagar (1986) [15] and led to the l^1-optimal control problem.

The mathematical formulation of the problem is performed as follows:

(a) *Disturbance, d, is bounded*

$$0 \le |d(k)| \le w_1 \ , \ \forall\, k \ge 0 <=> \|w_1^{-1} d\|_\infty \le 1.$$

(b) *Output, y, is bounded*

$$0 \le |y(k)| \le w_2^{-1} \ , \ \forall\, k \ge 0 <=> \|w_2 y\|_\infty \le 1.$$

(c) *Satisfy bounds on y for all allowable d*

$$\Phi(P,C) = \sup_{\|w_1^{-1} d\|_\infty \le 1} \|w_2 y\|_\infty \le 1.$$

Let $H(P,C)$ be the closed loop map between the disturbance (d) and the output (y). Then, $y = H(P,C)\, d$. According to (c) the output of the system remains within the desirable bounds iff:

$$\Phi(P,C) \le 1 <=> \sup_{\|w_1^{-1} d\|_\infty \le 1} \|w_2 y\|_\infty = \sup_{\|w_1^{-1} d\|_\infty \le 1} \|w_2 H(P,C)\, w_1 (w_1^{-1} d)\|_\infty =$$

$$= \|w_2 H(P,C)\, w_1\|_{\infty\infty} \le 1$$

To determine what is the best performance that a decentralized controller can deliver, the value of the following optimization problem should be identified :

$$\nu = \inf_{C \in S_d(P)} \|w_2 H(P,C) w_1\|_{\infty\infty}$$

If the value of this problem is less than 1 then there exist controllers with the given structure such that the output of the system satisfies the performance requirements. For simplicity in notation, w_1 and w_2 will be augmented in the map $H(P,C)$.

Employing the parametrization of all stabilizing decentralized controllers the last optimization problem is expressed in terms of the stable map $Q \in M(S)$:

$$\nu = \inf_{Q \in M(S)} \|H(P,Q)\|_{\infty\infty} \inf_{Q \in M(S)} \|T_1 - T_2\, Q\, T_3\|_{\infty\infty} \qquad \text{(DPP)}$$

subject to,

$$S_1 + Q\, S_2 + S_3\, Q + Q\, S_4\, Q = 0$$

In the last problem, the optimization variable belongs to an infinite dimensional linear space. In addition, the closed loop map $H(P,Q)$ is affine in Q : $H(P,Q)=T_1-T_2 Q T_3$ where $T_1,T_2,T_3 \in M(S)$, are known and depend on the factorization of P that is employed (Vidyasagar,1985, p.110) [14].

Let $h(p,q)$ be the impulse response sequence that corresponds to $H(P,Q)$. Let also $t_i=\{ t_i(k) \}_{k=0}^{\infty}$, $i=1, 2, 3$ and $q=\{ q(k) \}_{k=0}^{\infty}$, where $t_i(k)$ and $q(k)$ are real matrices of appropriate dimensions. Then, the sequence $h(p,q)=\{ h(k) \}_{k=0}^{\infty}$ is given by :

$$h(k)\hat{=}\begin{bmatrix} h_{11}(k) & \cdots & h_{1n}(k) \\ \vdots & & \vdots \\ \vdots & & \vdots \\ h_{m1}(k) & \cdots & h_{mn}(k) \end{bmatrix} = t_1(k)-\sum_{j=0}^{k}\left[t_2(k-j)\sum_{\lambda=0}^{j} q(\lambda)t_3(j-\lambda)\right] \qquad (1)$$

Similarly, let s_1,s_2,s_3,s_4 be the impulse response sequences, members of $M(l^1)$, that correspond to S_1,S_2,S_3,S_4 respectively and $s_i=\{ s_i(k) \}_{k=0}^{\infty}$ $i=1,2,3,4$. Then, the quadratic constraint is satisfied iff all the elements of the impulse response sequence that coresponds to the LHS are equal to zero. Thus the following infinite set of quadratic equality constraints is obtained:

$$f_k(q)\hat{=}s_1(k)+\sum_{j=0}^{k}\left\{ q(j)s_2(k-j)+s_3(k-j)q(j)+q(k-j)\left[\sum_{\lambda=0}^{j} s_4(j-\lambda)q(j)\right]\right\}=0\,,\ \forall\, k\geq 0 \qquad (2)$$

The objective function of (DPP) becomes:

$$\|H(P,Q)\|_{\infty\infty}=\|h(p,q)\|_{l^1}=\max_{i=1,\cdots,m}\sum_{j=1}^{n}\sum_{k=0}^{\infty}\left|\left\{ t_1(k)-\sum_{j=0}^{k}\left[t_2(k-j)\sum_{\lambda=0}^{j} q(\lambda)t_3(j-\lambda)\right]\right\}_{i,j}\right|$$

where

$$h_{i,j}(k)\hat{=}\left\{ t_1(k)-\sum_{j=0}^{k}\left[t_2(k-j)\sum_{\lambda=0}^{j} q(\lambda)t_3(j-\lambda)\right]\right\}_{i,j}$$

Based on (1) and (2) (DPP) is transformed into the following constrained l^1-optimization problem:

$$\nu=\inf_{q\in l^1_{m\times n}}\|h(p,q)\|_{l^1} \qquad \text{(DPPs)}$$

subject to,

$$f_k(q)=0\ ,\ \forall\, k\geq 0$$

For a given sequence q and a value of k, $f_k(q)$ is an $m\times n$ matrix with entries: $f_k^{i,j}(q)$, $i=1,...,m$, $j=1,...,n$. Then (DPPs) can be reformulated as:

$$\nu = \inf_{q \in l^1_{m\times n}} \|h(p,q)\|_{l^1} \qquad \text{(DPPs-a)}$$

subject to,

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=0}^{\infty} | f_k^{i,j}(q) | = 0$$

These optimization problem are infinite dimensional; the optimization variable (q) lies in an infinite dimensional linear space and the constraint is also infinite dimensional. In the following, it will be shown that one can obtain solutions, arbitrarily close to the solution of these problems, by solving appropriately constructed finite dimensional optimization problems.

To reduce (DPPs) to a finite dimensional optimization problem two types of truncations are performed:

Truncate the constraint

$$\nu^M = \inf_{q \in l^1_{m\times n}} \|h(p,q)\|_{l^1} \qquad \text{(DT1)}$$

subject to,

$$f_k(q) = 0 \ , \quad k = 0, \cdots, M$$

and

Truncate the variable q and the constraint

$$\nu_N^M = \inf_{q \in \phi_{o,N}} \|h(p,q)\|_{l^1} \qquad \text{(DT2)}$$

subject to,

$$f_k(q) = 0 \ , \quad k = 0, \cdots, M$$

where $\phi_{o,N} \subset \phi_o$ is the set of all $l^1_{m\times n}$ sequences with their first $N+1$ elements nonzero.

In the following, it is assumed that (DPPs) is feasible, thus feasibility of (DT1) is guaranteed. Under this condition, feasibility of (DT2) is guaranteed provided that $N > M$. The following theorem establishes the relationship between (DT1) and (DT2).

Theorem 1

$$\lim_{N\to\infty} \nu_N^M = \nu^M$$

Before proceeding with the proof of the theorem, let us consider the following sets:

$$G_M = \left\{ q \in l^1_{m\times n} \mid f_i(q) = 0 \ , \quad i = 0, \cdots, M \right\}$$

and

$$G_{o,M} = \left\{ q_1 \in \phi_o \mid f_i(q_1) = 0 \ , \quad i = 0, \cdots, M \right\} \subset G_M .$$

The first set is the feasible set of (DT1). The feasible set of (DT2) is a subset of the second set. Then the following lemma establishes a result that will be used in the proof of theorem 1.

Lemma 1

The set $G_{o,M}$ is dense in G_M.

Proof of Lemma 1

To prove that the first set is dense in the second it suffices to show that : given $\delta > 0$ and $q \in G_M$ there exists $q_1 \in G_{o,M}$ such that $\|q - q_1\|_{l^1} < \delta$.

The set of $M+1$ constraints $f_i(q) = 0$, $i = 0, \cdots, M$ involves only the first $M+1$ elements of the sequence q. Any other sequence with the same $M+1$ first elements satisfies the constraints. Therefore, q can be partitioned as : $q = [\, q_a \mid q_b \,] \in G_M$, where $q_a \in \mathrm{R}^{M+1}$ is the vector of elements of q that appear in the constraints. Clearly, $q_b \in l^1$ and there exists a sequence $q_{1,b} \in \phi_o$ such that for given $\delta > 0 \Rightarrow \|q_b - q_{1,b}\|_{l^1} < \delta$. The sequence $q_1 = [\, q_a \mid q_{1,b} \,]$ is a member of $G_{o,M}$ and :

$$\|q - q_1\|_{l^1} = \|q_a - q_a\|_1 + \|q_b - q_{1,b}\|_{l^1} < \delta .$$

□

Now, the proof of theorem 1 follows.

Proof of Theorem 1

We want to prove that $\lim_{N\to\infty} \nu_N^M = \nu^M$.

Equivalently, for a given $\varepsilon > 0$ we want to prove that there exists N such that $|\nu^M - \nu_N^M| < \varepsilon$. Since, $\nu^M = \inf_{q \in G_M} \|h(p,q)\|_{l^1}$, for a given $\varepsilon > 0$ there exists a $q \in G_M$ such that :

$$\left| \nu^M - \|h(p,q)\|_{l^1} \right| < \frac{\varepsilon}{2} \tag{2}$$

Since $\|h(p,\cdot)\|_{l^1}$ is continuous on G_M then, given $q \in G_M$ there exists $\delta > 0$ such that for all $q' \in G_M$,

$$\|q - q'\|_{l^1} < \delta \ \Rightarrow \ \left| \|h(p,q)\|_{l^1} - \|h(p,q')\|_{l^1} \right| < \frac{\varepsilon}{2}$$

Since $G_{o,M}$ is dense in $G_M \Rightarrow$ there exists $q_1 \in G_{o,M} : \|q - q_1\|_{l^1} < \delta$. Let N be the smallest

number such that $q_1 \in \phi_{o,N}$ and q_1 satisfies the last relationship.

Consequently,

$$\left| \|h(p,q)\|_{l^1} - \|h(p,q_1)\|_{l^1} \right| < \frac{\varepsilon}{2} \tag{3}$$

From (2) and (3) the following relation is obtained :

$$\left| \nu^M - \|h(p,q_1)\|_{l^1} \right| < \varepsilon$$

Since $\nu^M \le \nu_N^M$ and $0 \le \nu_N^M \le \|h(p,q_1)\|_{l^1}$ we finally obtain : $|\nu^M - \nu_N^M| < \varepsilon$.

□

This theorem establishes the first approximation result. However, its application assumes that for any given N and M the value ν_N^M is known, or can be computed. The solution of (DT2) involves the minimization of an infinite sum.

Given an $l^1_{m\times n}$ sequence ξ, its $l^1(L)$ sum is defined as :

$$\|\xi\|_{l^1(L)} = \max_{i=1,\cdots,m} \sum_{j=1}^{n} \|\xi_{i,j}\|_{l^1(L)} = \max_{i=1,\cdots,m} \sum_{j=1}^{n} \sum_{k=0}^{L} |\xi_{i,j}(k)|$$

Define the set of sequences $q \in \phi_{o,N}$ that are norm bounded by some positive number B :

$$q \in \phi_{o,N}^B = \left\{ q \in \phi_{o,N} \; ; \; \|q\|_{l^1} \le B \right\}$$

The set $\phi_{o,N}^B$ is finite dimensional (so is $\phi_{o,N}$), closed, bounded and therefore by the Heine-Borel theorem it is compact. Consider a formulation of (DT2) where the variable is norm bounded:

$$\nu_N^M(B) = \inf_{q \in \phi_{o,N}^B} \|h(p,q)\|_{l^1} \tag{DT2a}$$

subject to,

$$f_i(q) = 0 \;, \quad i = 0, \cdots, M$$

Clearly, $\nu_N^M \le \nu_N^M(B)$ for all values of B.

Let $\bar{q} \in \phi_{o,N}$ be a suboptimal solution of (DT2), i.e.:

$$\forall\, \varepsilon > 0 \,, \; \exists \bar{q} \in \phi_{o,N} \; ; \; 0 \le \|h(p,\bar{q})\|_{l^1} - \nu_N^M < \varepsilon$$

The norm of $\bar{q}$ is finite. Let $\|\bar{q}\|_{l^1} = B(\varepsilon)$. Existence of the solution of (DT2a) is guaranteed by the compactness of its feasible set (see proof of lemma 2). Let $\bar{q}'$ be the solution of (DT2a). Then, $\bar{q}'$ satisfies the relation:

$$\nu_N^M \le \nu_N^M(B(\varepsilon)) = \|h(p,\bar{q}')\|_{l^1} \le \|h(p,\bar{q})\|_{l^1}$$

Then, combining these statements the following is obtained:

$$\forall\, \varepsilon > 0\,,\ \exists B(\varepsilon)\,;\ 0 \le \nu_N^M(B(\varepsilon)) - \nu_N^M < \varepsilon$$

If B is selected to be sufficiently large, the solution of (DT2a) is arbitrarily close to the solution of (DT2). In subsequent sections it will be demonstrated that the calculation of bounds for the optimization variables is feasible.

Based on this discussion, the following optimization problem is formulated:

$$\nu_{N,L}^M = \inf_{q \in \phi_{o,N}^B} \| h(p,q) \|_{l^1(L)} \qquad \text{(DT3)}$$

subject to,

$$f_k(q) = 0\ ,\quad k = 0, \cdots, M$$

The following lemma establishes the fact that the solution to (DT2) can be obtained through iterative solution of (DT3) for increasing values of L.

Lemma 2

$$\lim_{L \to \infty} \nu_{N,L}^M = \nu_N^M(B)$$

Proof

The sequence $\{ \nu_{N,L}^M \}_{L=0}^{\infty}$ is nondecreasing and bounded:

$$\forall\, L \ge 0 :\ 0 \le \nu_{N,L}^M \le \nu_N^M(B)\,.$$

As a result : $\exists \alpha > 0\ ;\ \lim_{L \to \infty} \nu_{N,L}^M = \alpha.$

This limit cannot be greater than $\nu_N^M(B)$. Assume that $\alpha < \nu_N^M(B)$.

The feasible set of (DT3) is compact. Indeed, it can be written as the intersection of a finite dimensional, closed and bounded set ($\phi_{o,N}^B$) with a finite dimensional closed set:

$$\phi_{o,N}^B \cap G_{o,M} = \phi_{o,N}^B \cap \left\{ q_1 \in \phi_{o,N} \ \middle|\ f_i(q_1) = 0\ ,\quad i = 0, \cdots, M \right\}$$

As a result the feasible set is finite dimensional, closed and bounded.

From the continuity of the $l^1(L)$ norm, and the compactness of the feasible set it follows that (Luenberger, 1969; p.14) [8]:

$$\forall\, L > 0\,, \exists q_L \in \phi_{o,N}^B \cap G_{o,M}\ ;\ \nu_{N,L}^M = \| h(p,q_L) \|_{l^1(L)}$$

Compactness of the feasible set implies that the sequence $\{ q_L \}_{L=0}^{\infty}$ has a subsequence $\{ q_{Lk} \}_{k=0}^{\infty}$ that converges in $\phi_{o,N}^B \cap G_{o,M}$:

$$\lim_{k \to \infty} q_{Lk} = \bar{q} \in \phi_{o,N}^B \cap G_{o,M}$$

Then,

$$\lim_{L\to\infty} \nu_{N,L}^{M} = \lim_{L\to\infty} \|h(p,q_L)\|_{l^1(L)} = \lim_{k\to\infty} \|h(p,q_{L_k})\|_{l^1(L_k)} = \|h(p,\bar{q})\|_{l^1} = \alpha$$

This contradicts the assumption that $\alpha < \nu_N^M(B)$. Hence, $\alpha = \nu_N^M(B)$.

□

In summary, lemma 2 establishes that solution of (DT3) for increasing values of L identifies the solution of (DT2a). Based on the "equivalence" of (DT2a) and (DT2), it can be said that this iterative procedure identifies ε - optimal solutions of (DT2).

Remark : In the case where the coprime factors of P have been constructed to be FIR's, for each $q \in \phi_{o,N}$ the sequence $h(p,q)$ will have a finite number of nonzero elements. This number is known and depends only on N. Therefore the norm of $h(p,q)$ can be exactly calculated by a finite sum. In this case, the exact solution of (DT2) is obtained in one step.

The relation between the two different types of truncated problems has been established through Theorem 1. In the remaining of this section the connection between (DT1) and (DPPs) is shown. For a non-negative real number δ consider the following sets:

$$G(\delta) \hat{=} \left\{ q \in l^1_{m\times n} : \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=0}^{\infty} |f_k^{i,j}(q)| \le \delta \right\}, \quad G^M(\delta) \hat{=} \left\{ q \in l^1_{m\times n} : \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=0}^{M} |f_k^{i,j}(q)| \le \delta \right\}$$

Then, the following theorem holds:

Theorem 2

Assume that $G(\cdot)$ and $G^M(\cdot)$ represent upper semicontinuous mappings from the non-negatives reals to subsets of l^1. Then,

$$\lim_{M\to\infty} \nu^M = \nu.$$

Before we proceed with the proof of Theorem 2, two intermediate results will be presented. Consider the two optimization problems:

$$\nu(\omega) = \inf_{q\in l^1_{m\times n}} \left\{ \|h(p,q)\|_{l^1} + \omega \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=0}^{\infty} |f_k^{i,j}(q)| \right\} \qquad \text{(P1)}$$

and

$$\nu^M(\omega) = \inf_{q\in l^1_{m\times n}} \left\{ \|h(p,q)\|_{l^1} + \omega \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=0}^{M} |f_k^{i,j}(q)| \right\} \qquad \text{(P2)}$$

It is easily verified that (P1) is the penalty function formulation of (DPPs-a) (Luenberger, 1969, pp.302-305) [8]. For this type of problems the following lemma can be shown to hold:

Lemma 3

Let $\{\omega^r\}_{r=0}^{\infty}$ be an icreasing sequence such that $\lim_{r\to\infty} \omega^r = \infty$. Let also $G(\cdot)$ be an upper semicontinuous mapping from the non-negative reals to subsets of $l^1_{m\times n}$. Then, the following statements hold:

1. $\nu(\omega^{r+1}) \geq \nu(\omega^r)$
2. $\nu \geq \nu(\omega^r)$
3. $\lim_{r\to\infty} \nu(\omega^r) = \nu$.

Proof

Part 1 & 2

The proof of Part 1 & 2 is performed in a similar way as the proof of statements 1 & 2 in Lemma 1 in Luenberger (1969, p.305) [8].

Part 3

From Part 1 & 2 it follows that the limit exists and is less than or equal to ν. The proof is similar to the proof of Part 3 in Lemma 4.2 in Sourlas and Manousiouthakis (1992) [13]. □ Employing the same line of arguments the following corollary can be shown to hold:

Corollary 1

Let $G^M(\cdot)$ be an upper semicontinuous mapping from the non-negative reals to subsets of $l^1_{m\times n}$. Then,

1. $\nu^M(\omega^{r+1}) \geq \nu^M(\omega^r)$
2. $\nu^M \geq \nu^M(\omega^r)$
1. $\lim_{r\to\infty} \nu^M(\omega^r) = \nu^M$.

For a given value of ω, Lemma 4 establishes the relation between $\nu^M(\omega)$ and $\nu(\omega)$.

Lemma 4

$$\lim_{M\to\infty} \nu^M(\omega) = \nu(\omega).$$

Proof

The proof is based on the fact that the sequence $\{ f_k(q) \}_{k=0}^{\infty}$ belongs to $l^1_{m\times n}$ thus allowing the approximation of the infinite sum in the penalty term in (P1) with a finite sum. The detailed proof can be found is similar to the proof of Lemma 4.3 in Sourlas and Manousiouthakis (1992) [13]. □

Now the proof of Theorem 2 follows.

Proof of Theorem 2

The proof consists of two parts. First existence of the limit will be shown and then convergence to ν will be demonstrated. The sequence ν^M is nondecreasing, and bounded above by ν. As a result, it converges to a real number:

$$\alpha \hat{=} \lim_{M\to\infty} \nu^M \leq \nu.$$

Using Corollary 1, Lemma 3 and 4 the last relation can be rewritten as:

$$\alpha = \lim_{M\to\infty} v^M = \lim_{M\to\infty}\lim_{\omega\to\infty} v^M(\omega) \le v = \lim_{\omega\to\infty}\lim_{M\to\infty} v^M(\omega) \qquad (4)$$

In the remaining of the proof it will be shown that the two iterated limits are equal.

Using the proof technique of Lemma 3 it can be established that:

$$v^M(\omega) \le \lim_{\omega\to\infty} v^M(\omega) \ , \ \forall M \ , \ \forall \omega$$

This implies that:

$$v(\omega) = \lim_{M\to\infty} v^M(\omega) \le \lim_{M\to\infty}\lim_{\omega\to\infty} v^M(\omega) \ , \ \forall \omega \Rightarrow$$

$$v = \lim_{\omega\to\infty}\lim_{M\to\infty} v^M(\omega) \le \lim_{M\to\infty}\lim_{\omega\to\infty} v^M(\omega) = \alpha$$

In view of (4), the last relation implies that $\alpha = v$.
As a result, the following statement has been proven:

$$\alpha = \lim_{M\to\infty} v_M = \lim_{M\to\infty}\lim_{N\to\infty} v_{N,M} = v$$

□

4.2. Computational Procedure

In view of theorems 1,2 and lemma 2, the value v is obtained through a limiting procedure, which involves solution of (DT3) for increasing values of L, N and M. The finite dimensional optimization problem can be formulated as a nonlinear program, which can be solved by finite dimensional optimization techniques. For a particular example, the structure of the nonlinear program is given in appendix A.

The computationally intensive part of the procedure is the solution of (DT3), which is a nonlinear programming problem, for different values of L, N and M. The resulting nonlinear programs are nonconvex, due to the existence of the quadratic equality constraints. Nevertheless, global solution of these problems determines the globally optimum performance that can be achieved by a certain decentralized structure. If (DT3) is solved locally, then the limit of the resulting sequence will identify an upper bound to the l^1 optimal decentralized performance.

4.3. Global Optimization Approach

Global optimization approaches that solve the general nonconvex optimization problem have recently been developed. Manousiouthakis & Sourlas (1992) [10], presented a procedure that is based on the transformation of the original nonlinear optimization problem into one that has convex constraints and objective with an additional separable, quadratic, reverse convex constraint. Employing this transformation procedure, (DT3) becomes a convex programming problem with an additional reverse convex, quadratic and separable constraint. This problem

can then be solved and its global optimum can be identified through the use of a branch and bound type of algorithm.

The implementation of this global optimization algorithm requires the existence of valid upper and lower bounds for all elements of the the sequence q that appear in the quadratic equality constraints in (DT3). Bounds on these variables can always be obtained from local minima information, namely the value a local minimum of (DT2) for fixed values of N and M. One can always obtain such information if the solution of (DT3) for fixed N, M and increasing L is performed using local optimization algorithms. The generated sequence of values converges to the value of (DT2) at a local minimum. Let δ_{local} be this value. This is an upper bound to the globally optimal value of (DT2). Thus, for $q \in \phi_{o,N}$ and for each value of L, it holds that:

$$\|h(p,q)\|_{l^1(L)} \le \delta_{local} <=> \max_{i=1,\cdots,m} \sum_{j=1}^{n} \sum_{k=0}^{L} |h_{i,j}(k)| \le \delta_{local} =>$$

$$h_{i,j}(k) \in \left[-\delta_{local}, \delta_{local}\right] \quad ,i=1,\cdots,m \ , \ j=1,\cdots,n \ , \ k=0,\cdots,L \tag{5}$$

Then, one can obtain lower (upper) bounds on all elements of the sequence q through the solution of a minimization (maximization) problem with objective the corresponding element of the sequence and constraints the inclusion realtions that appear in (5). These optimization problems are linear.

5. Example

In this section the computational procedure introduced in section 4.2 is applied to the following 2×2 example.

$$P(z) = \begin{bmatrix} \dfrac{3z+1}{z} & \dfrac{0.5z+2}{z} \\ \dfrac{z^2+2.5z+1}{z^2} & \dfrac{3z+4.5}{z} \end{bmatrix}$$

The objective of our analysis is to determine the best possible achievable performance for a decentralized control system featuring the pairings $\{(y_1,u_1);(y_2,u_2)\}$.

Parametrization of all Decentralized Stabilizing Controllers

The process P is stable. Therefore one can select coprime factors as follows:

$$N_P = \tilde{N}_P = P \ , \ D_P = Y = I \ , \ \tilde{D}_P = \tilde{Y} = I \ , \ X = \tilde{X} = 0$$

According to (2), sec.3.1, any stabilizing controller for P is parametrized as:

$$C = Q(I-PQ)^{-1} \ , \ Q = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \in M(S) \tag{1}$$

The controller C is decentralized iff:

$$Q_{21}P_{12} = P_{21}Q_{12}$$
$$Q_{12} = -P_{12}(Q_{11}Q_{22} - Q_{12}Q_{21})$$

The Input - Output map between disturbances and output is given by:

$$H(P,C): u_2 \rightarrow y_2 \; ; \; H(P,C) = (I + PC)^{-1} P = (I - PQ) P$$

Problem Formulation

The disturbance rejection problem is formulated according to the guidelines introduced in section 4.1. First, the known bounds on the disturbance and the desired bounds on the objective are defined:

Disturbance: $u_2^T(t) \in [-1,1] \times [-1,1] \; , \; \forall t$

Output: $y_2^T(t) \in [-1,1] \times [-1,1] \; , \; \forall t$

Then, (DPPs) is readily formulated and transformed to (DT3) according to the procedure presented in Section 4. Based on Appendix A, (DT3) is in turn transformed into a nonlinear programming problem. Since the coprime factors are F.I.R. then the solution to (DT2) is obtained in one step. For $N=M=0,1,2$ the globally optimum value of the corresponding optimization problem has been identified. The complete sequence of values that converges to the value of ν for this particular problem is shown in the following table:

N	M	ν_N^M	N	M	ν_N^M
0	0	8.00	5	5	5.20
1	1	6.70	6	6	5.16
2	2	5.85	7	7	5.13
3	3	5.51	8	8	5.13
4	4	5.32	9	9	5.13

Hence, with $\varepsilon = 10^{-2}$ the value of the l^1 - optimal decentralized performance problem was found to be 5.13. When only the linear constraints are considered the value of the global lower bound to the l^1 decentralized performance is 4.98. The l^1 - optimal centralized performance, for the same set of specifications was to found to be equal to 4.72.

6. Conclusions

Based on the Manousiouthakis parametrization of all decentralized stabilizing controllers, the l^1 optimal decentralized performance problem has been formulated as an infinite dimensional optimization problem. This problem was transformed into a finite dimensional one through the introduction of appropriate truncations. Theorems that establish the equivalence (in the limit) of the original problems to their finite dimensional approximations were proven, and a computational procedure was proposed. It has been established that solution to the optimal decentralized performance problem amounts to global solution of a series of quadratically constrained programming problems. If locally optimal solutions are identified for each of the finite dimensional problems, the limit of the corresponding sequence of values will be an upper bound to the optimal l^1 decentralized performance. Based on this work, one can actually evaluate the best performance achievable by a given decentralized structure.

References

1. B. D. O. Anderson and D. J. Clements, "Algebraic Characterization of Fixed Modes in Decentralized Control," *Automatica*, vol. 17, No 5, pp. 703-712, 1981.
2. M. Aoki, "On Feedback Stabilizability of Decentralized Dynamic Systems," *Automatica*, vol. 8, pp. 163-173, 1972.
3. J. P. Corfmat and A. S. Morse, "Decentralized Control of Linear Multivariable Systems," *Automatica*, vol. 12, pp. 479-495, 1976.
4. J.P. Corfmat and A.S. Morse, "Control of Linear Systems Through Specified Input Channels," *SIAM J. Control and Opt*, vol. 14, pp. 163-175, 1976.
5. C. A. Desoer and M. Vidyasagar, *Feedback Systems : Input - Output Properties*, Academic Press, New York, 1975.
6. A.V. Fiacco, *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*, Academic Press, New York, 1983.
7. A. N. Gundes and C. A. Desoer, *Algebraic Theory of Linear Feedback Systems with Full and Decentralized Compensators*, Springer-Verlag, Heidelberg, 1990.
8. David G. Luenberger, *Optimization by Vector Space Methods*, John Wiley & Sons Inc., New York, 1969.
9. V. Manousiouthakis, "On the Parametrization of all Decentralized Stabilizing Controllers," *Proc. American Control Conf.*, vol. 3, pp. 2108-2111, Pittsburgh, PA, June 1989.
10. V. Manousiouthakis and D. Sourlas, "A Global Optimization Approach to Rationally Constrained Rational Programming," *Chem. Eng. Comm*, vol. 115, pp. 127-147, 1992.
11. A. S. Morse, "Structural Invariants of Linear Multivariable Systems," *SIAM J. Control*, vol. 11, pp. 446-465, 1973.
12. J. M. Potter, B. D. O. Anderson and A. S. Morse, "Single Channel Control of a Two Channel System," *IEEE Trans. Automat. Contr.*, vol. AC-24, pp. 491-492, 1979.
13. D. Sourlas and V. Manousiouthakis, "On $l^1 - l^\infty$ Simultaneously Optimal Control," *submitted IEEE Trans. Automat. Contr.*, 1992.
14. M. Vidyasagar, *Control System Synthesis. A factorization Approach*, MIT Press, Cambridge, MA, 1985.
15. M. Vidyasagar, "Optimal Rejection of Persistent Bounded Disturbances," *IEEE Trans. Automat. Contr.*, vol. AC-31, pp. 527-534, 1986.
16. S. Wang and E. J. Davison, "On the Stabilization of Decentralized Control Systems," *IEEE Trans. Automat. Contr.*, vol. AC-18, No 5, pp. 473-478, 1973.
17. Richard L. Wheeden and Antoni Zygmund, *Measure and Integral. An Introduction to Real Analysis*, Marcel Dekker Inc., New York, 1977.

Appendix A

The optimization problem (DT3) can be formulated as a nonlinear programming problem. The steps that make this transformation feasible follow. For illustration purposes the 2×2 case, with stable plant, the same as the one in the example, is considered. From the definition of the $l^1(L)$ sum :

$$h(p,q) \in l^1_{2\times 2} \Rightarrow \|h(p,q)\|_{l^1(L)} = \delta = \max_{i=1,2} \sum_{k=0}^{L} |h_{i1}(k)| + |h_{i2}(k)|$$

Using the definition of the maximum as the least upper bound, (DT3) is finally transformed into:

$$\nu^M_{N,L} = \inf_{\substack{q_{ij}(k),\delta,\delta_{ij}(k) \\ i,j=1,2 \\ k=0,\cdots,L}} \delta \qquad \text{(A.1)}$$

subject to,

$$\left.\begin{aligned} f_k(q) &= \sum_{i=0}^{k} \left[q_{12}(i) p_{21}(k-i) - q_{21}(i) p_{12}(k-i) \right] = 0 \\ g_k(q) &= q_{12}(k) + \sum_{i=0}^{k} p_{12}(k-i) \sum_{j=0}^{i} \left[q_{11}(j) q_{22}(i-j) - q_{12}(j) q_{21}(i-j) \right] = 0 \end{aligned}\right\} k = 0, \cdots, M$$

$$q_{ij}(k) = 0 \ , \ i, j = 1, 2 \ , \ N+1 \le k \le L$$

$$-\delta_{ij}(k) \le h_{ij}(k) \le \delta_{ij}(k) \ , \ i, j = 1, 2 \ , \ k = 0, \cdots, L$$

$$\sum_{k=0}^{L} \left[\delta_{i1}(k) + \delta_{i2}(k) \right] \le \delta \ , \ i = 1, 2$$

where $f_k(q)$ and $g_k(q)$ are scalar constraints resulting from the application of the parametrization to this particular case (2×2 controller, stable plants).

A Rational Interpolation Method to Compute Frequency Response *

Charles Kenney, Stephen Stubberud, and Alan J. Laub
Department of Electrical and Computer Engineering
University of California
Santa Barbara, CA 93106-9560

Abstract

A rational interpolation method for approximating a frequency response is presented. The method is based on a product formulation of finite differences, thereby avoiding the numerical problems incurred by near-equal-valued subtraction. Also, resonant pole and zero cancellation schemes are developed that increase the accuracy and efficiency of the interpolation method. Selection techniques of interpolation points are also discussed.

1 Introduction

Consider the linear time-invariant system given by the state-space model

$$\dot{x} = Ax + Bu \tag{1}$$
$$y = Cx \tag{2}$$

where $A \in \mathbb{R}^{n_A \times n_A}$, $B \in \mathbb{R}^{n_A \times n_B}$, $C \in \mathbb{R}^{n_C \times n_A}$, and the state vector, input vector, and output vector, x, u, and y, respectively, are properly dimensioned. We shall refer to the matrices, A, B, and C, as the state coupling matrix, the input coupling matrix, and the output coupling matrix, respectively.

The frequency response of such a modeled system is defined as the Laplace transform of the input-output relationship evaluated along the $j\omega$-axis,

$$G(\omega) = C(j\omega I - A)^{-1}B \tag{3}$$

where

$$0 < \omega < \infty.$$

*This research was supported in part by the Air Force Office of Scientific Research under Contract No. AFOSR91-0240.

In this paper, a fast and reliable interpolation method to compute frequency response is presented. The basic idea of this method is based on the simple Taylor series approximation

$$G(\omega + h) = T_0 + T_1 h + \cdots + T_k h^k + E_k \tag{4}$$

but considered in the general interpolation form with $k+1$ interpolation points $h_0, h_1, \ldots, h_k$:

$$G(\omega + h) = G_0 + G_1(h - h_0) + \cdots + G_k(h - h_0)(h - h_1)\cdots(h - h_{k-1}) + E_k. \tag{5}$$

The coefficient matrices, $G_0, G_1, \ldots, G_k$ are of size $n_C \times n_B$ as is the truncation error E_k. Therefore, the cost of evaluating the matrix polynomial approximation

$$P_k(h) = G_0 + G_1(h - h_0) + \cdots + G_k(h - h_0)(h - h_1)\cdots(h - h_{k-1}) \tag{6}$$

is just $kn_B n_C$ floating-point operations (flops). The cost of computing each coefficient matrix is approximately the same as evaluating G by the method that would normally be preferred.

The polynomial interpolation scheme works well as long as ω is not near a resonant pole or zero of the system. In order to avoid this problem, we introduce methods of preliminary pole and zero cancellation. These greatly increase the accuracy of the interpolation scheme while causing only a negligible increase in the cost of computing the coefficient matrices.

We shall also discuss the implementation of this algorithm including ideas on the selection of interpolation points.

2 Existing Frequency Response Methods

2.1 Straightforward Computation

An obvious method for computing frequency response for a system modeled in state-space form is first to perform an LU decomposition in order to solve the linear system

$$(j\omega I - A)X = B, \tag{7}$$

followed by a matrix multiplication involving the solution to (7),

$$G(\omega) = CX.$$

This method does not exploit any special structure, e.g., sparse or banded, and therefore would only be used for general systems. To compute a frequency response implementing this method, for just one value of ω, approximately $\frac{1}{3}n_A^3 + \frac{1}{2}(n_B + n_C)n_A^2 + n_A n_B n_C$ flops are required. As the number of desired frequency points becomes large, the calculation of the entire frequency response becomes computationally intensive.

2.2 The Principal Vector Algorithm

In order to reduce computation cost, several methods have been developed in order to reduce the cost of solving the linear system (7) either by exploiting the structure of the state coupling matrix or by implementing a similarity transformation to put the matrix A into an exploitable form. One method of the latter variety is the Principal Vector Algorithm (PVA) [10].

The idea of the PVA is to initially transform the state coupling matrix into a Jordan Canonical Form (JCF). The algorithm uses the principal vectors to compute the JCF in a more accurate way than previous such algorithms. Let

$$A = M^{-1}JM \tag{8}$$

where J is in Jordan form. If we substitute this identity into (3), the frequency response becomes

$$\begin{aligned} G(\omega) &= CMM^{-1}(j\omega I - A)^{-1}MM^{-1}B \\ &= \tilde{C}(j\omega I - J)^{-1}\tilde{B}. \end{aligned} \tag{9}$$

The initial transformation using the PVA to compute the JCF requires only $O(n_A^3)$ flops if the state coupling matrix is not defective while $O(n_A^4)$ flops are required if the matrix A is defective. Note that this transformation only occurs once, thus the cost is only incurred once. The advantage occurs in computations at each frequency point where the cost is reduced to $O(n_A + n_A n_B n_C)$ flops in the nondefective case and $O(\frac{3}{2}n_A + n_A n_B n_C)$ flops in the defective case. So the computational saving occurs after the computation of one frequency point in the former case and n_A frequency points in the latter case.

Although this algorithm produces significant savings in the computational cost of a frequency response, it can also frequently encounter numerical instabilities. First, the JCF is extremely unstable. The slightest perturbation can change a defective matrix into a nondefective matrix. Another problem is that the similarity transform may be ill-conditioned with respect to inversion depending on the basis of eigenvectors. If they are guaranteed to form a matrix which is well-conditioned with respect to inversion as would occur if the matrix were normal, the algorithm is very effective.

2.3 The Hessenberg Method

Another algorithm which uses similarity transformations to put the state coupling matrix into an exploitable form is the Hessenberg Method [4]. This algorithm is the current standard for computing the frequency response for generic dense systems. The Hessenberg Method, as its name implies, performs an initial transformation on the state coupling matrix to reduce it to upper Hessenberg form. So in this case, we use the identity

$$A = Q^{-1}HQ,$$

where H is in upper Hessenberg form, instead of the JCF identity (8), in the frequency response (3).

As with PVA, this initial transformation is performed only once at the start of the algorithm at a cost of $O(n_A^3)$ flops. When this transformation is used, the cost of computing the frequency response at each value of ω becomes $O(n_A^2(n_B+1)+n_An_Bn_C)$ flops. Usually, $n_B \ll n_A$ so a significant reduction in computation can be realized.

Fortunately, there always exists an orthogonal transformation to reduce the state coupling matrix into an upper Hessenberg form. This prevents ill-conditioning from being introduced into the calculations by the similarity transformation as can occur with the Principal Vector Algorithm.

2.4 Sparse Systems

Many of today's large ordered systems are sparse systems. A sparse system is one whose modeling matrices have relatively few nonzero entries when compared to the total number of entries. In such cases the Hessenberg Method should not be used. Instead of maintaining sparsity, the initial transformation will create a large dense system which then must be solved. There exist many storage techniques for sparse matrices which require a significantly smaller amount of memory allocation than a full matrix of the same order would require. Also, sparse matrix algorithms have been developed to exploit sparsity in order to reduce the computational costs in comparison to their dense counterparts. (See [6], [9], and [7].) These algorithms attempt to prevent the cost of solving the linear system (7) from growing to $O(n_A^3)$ flops.

2.5 Frequency Selection Routines

The cost of computing an entire frequency response can also be reduced by eliminating needless recalculations or overcalculations in attempts to get a desired resolution in the solution. When the frequency mesh is too coarse to give the required information, usually the user recomputes the entire frequency response. Often, the response from the previously computed frequency values either is recalculated or just ignored in the new calculation. Also, many times the user creates a fine frequency point mesh across the entire frequency range. Usually, only in small subregions is the finer mesh needed. A coarser mesh would suffice over the rest of the frequency range.

In an effort to eliminate these unnecessary calculations but still give the required accuracy, so-called adaptive routines have been developed. These routines adapt the frequency point's selection to the characteristics of the system being analyzed.

One such adaptive scheme is similar in nature to the QUANC8 adaptive integration routine [1]. The basic idea is first to select the endpoints of an interval in the desired frequency

region. Then the frequency responses of the two points are compared. If the difference between their magnitudes or their phases is greater than specified tolerances, the interval is divided in half. Then the three points are compared. If their differences are outside the tolerances, the subintervals are again halved. This subinterval halving continues until the tolerances are met across the entire interval or until a specified number of frequency points has been calculated. A single-input single-output variation of a method based on subinterval halving has been implemented commercially [5].

The use of *a priori* information, e.g., the locations of poles and zeros of a system, can also be used in the choice of frequency locations. More points are placed in the areas where the poles and zeros of a given system have an effect. Fewer points are placed outside these areas. Such a method is now being implemented in a linear system package [2] to automatically choose the frequency range over which the frequency response is computed as well as to determine the number of points needed to be calculated.

These adaptive schemes also can be combined to form hybrid routines. This would permit an initial placement of points with the *a priori* method and then create the frequency mesh to join the regions between the areas of the initial placement.

3 Polynomial Interpolation

In order to compute the coefficient matrices, $G_1, \ldots, G_k$, of the interpolation equation

$$P_k(h) = G_0 + G_1(h - h_0) + \cdots + G_k(h - h_0)(h - h_1)\cdots(h - h_{k-1}) \tag{10}$$

finite differences will be employed. The first-order difference is defined as

$$M[h_0, h_1] = \frac{M(h_1) - M(h_0)}{h_1 - h_0} \tag{11}$$

while higher-order differences are defined as

$$M[h_0, h_1, ..., h_n] = \frac{M[h_1, ..., h_n] - M[h_0, ..., h_{n-1}]}{h_n - h_0}. \tag{12}$$

If we let

$$M(h) = (jhI - A_0)^{-1} \tag{13}$$

where

$$A_0 = -jwI + A, \tag{14}$$

the k^{th}-order interpolation approximation can be written as

$$\begin{aligned} P_k(h) \;=\; & C(M(h_0) + M[h_0, h_1](h - h_0) + \cdots \\ & + M[h_0, h_1, ..., h_k](h - h_0)(h - h_1)\cdots(h - h_{k-1}))B \end{aligned} \tag{15}$$

with the interpolation error

$$\begin{aligned} E_k &= G(\omega + h) - P_k(h) \\ &= C(M[h_0, h_1, ..., h_k, h] \prod_{i=0}^{k} (h - h_i))B. \end{aligned} \tag{16}$$

Now, for convenience, define

$$\begin{aligned} \tilde{P}_k(h) &= M(h_0) + M[h_0, h_1](h - h_0) + \cdots \\ &\quad + M[h_0, h_1, ..., h_k](h - h_0)(h - h_1) \cdots (h - h_{k-1}) \end{aligned} \tag{17}$$

and

$$\begin{aligned} \tilde{E}_k &= M(h) - \tilde{P}_k(h) \\ &= M[h_0, h_1, ..., h_k, h] \prod_{i=0}^{k} (h - h_i). \end{aligned} \tag{18}$$

Although finite differences have a certain elegance to their formulation, they can encounter numerical inaccuracies due to the subtraction of near-equal-valued quantities. An extreme example of this is the case in which all of the interpolation points are the same. In theory, the first-order difference is exactly the first derivative of M, but numerically it is useless. Fortunately, the differences of the resolvent function (13), can be expressed in matrix product forms which avoid these cancellation problems as the following theorem shows.

Theorem 1 *For the resolvent function, the matrix difference functions in (12) satisfy*

$$M[h_0, h_1, \ldots, h_m] = (-j)^m M(h_0)M(h_1) \cdots M(h_m). \tag{19}$$

Proof: Using (13)

$$\begin{aligned} M(h_1) - M(h_0) &= (jh_1 I - A_0)^{-1} - (jh_0 I - A_0)^{-1} \\ &= (jh_0 I - A_0)^{-1} \{jh_0 I - A_0 - (jh_1 I - A_0)\} (jh_1 I - A_0)^{-1} \\ &= (-j)(h_1 - h_0)M(h_0)M(h_1). \end{aligned}$$

Thus the first finite difference becomes

$$M[h_0, h_1] = -jM(h_0)M(h_1)$$

which proves (19) for $m = 1$. Now suppose that (19) is true for $m-1$. Since $M(h_0), \ldots, M(h_m)$ all commute with each other, we find that

$$M[h_0, h_1, \ldots, h_m]$$

$$
\begin{aligned}
&= (M[h_1,\ldots,h_m] - M[h_0,\ldots,h_{m-1}])/(h_m - h_0) \\
&= (-j)^{m-1}\left(M(h_1)\cdots M(h_m) - M(h_0)\cdots M(h_{m-1})\right)/(h_m - h_0) \\
&= (-j)^{m-1}\left(M(h_1)\cdots M(h_{m-1})\right)\left(M(h_m) - M(h_0)\right)/(h_m - h_0) \\
&= (-j)^{m}\left(M(h_1)\cdots M(h_{m-1})\right)M(h_0)M(h_m) \\
&= (-j)^{m}M(h_0)M(h_1)\cdots M(h_m).
\end{aligned}
$$

Thus (19) is true for m and thus, by induction, the theorem is true. □

If we now substitute the resolvent identity (19) into (17) and (18) and use the commutative property of the resolvent functions, the interpolation approximation becomes

$$
\begin{aligned}
\tilde{P}_k(h) &= M(h_0) + (-j)M(h_1)M(h_0)(h - h_0) + \cdots \\
&\quad +(-j)^k M(h_k)M(h_{k-1})\cdots M(h_0)(h - h_0)\cdots(h - h_{k-1}) \qquad (20)
\end{aligned}
$$

with the error formula

$$
\begin{aligned}
\tilde{E}_k &= M(h) - \tilde{P}_k(h) \\
&= (-j)^{k+1}\prod_{i=0}^{k} M(h_i)\prod_{i=0}^{k}(h - h_i)M(h). \qquad (21)
\end{aligned}
$$

The next lemma gives an interpolation series for the resolvent using the original $k+1$ interpolation points and setting all of the higher-order interpolation points equal to zero. For convenience we shall use the notation $M(0) = M_0$. Note that if all of the interpolation points are set equal to zero the analysis would be that of the Taylor series.

Lemma 2 *Let $h_0,\ldots,h_k$ be given and set $h_m = 0$ for all $m > k$. For*

$$
|h| < \min_{\lambda\in\Lambda(A)} |j\omega - \lambda|, \qquad (22)
$$

M may be expanded as

$$
M(h) = \sum_{m=0}^{+\infty}(-j)^m \prod_{i=0}^{m} M(h_i)\prod_{i=0}^{m-1}(h - h_i). \qquad (23)
$$

Proof: Let $\ell > k$. By (21),

$$
\begin{aligned}
&M(h) - \sum_{m=0}^{\ell}(-j)^m \prod_{i=0}^{m} M(h_i)\prod_{i=0}^{m-1}(h - h_i) \\
&\quad = (-j)^{\ell+1}\prod_{i=0}^{\ell} M(h_i)\prod_{i=0}^{\ell}(h - h_i)M(h) \\
&\quad = (-j)^{\ell+1}\prod_{i=0}^{k} M(h_i)\prod_{i=0}^{k}(h - h_i)M(h)\prod_{i=k+1}^{\ell} M_0 h \\
&\quad = \left\{(-j)^{\ell+1}\prod_{i=0}^{k} M(h_i)\prod_{i=0}^{k}(h - h_i)M(h)\right\}(hM_0)^{\ell-k}.
\end{aligned}
$$

But $(hM_0)^{\ell-k} \to 0$ as $\ell \to +\infty$ if and only if $\rho(hM_0) < 1$, which is the well-known convergence requirement for a geometric series. From the definition of M_0, we have $M_0 = (j\omega I - A)^{-1}$. Hence,

$$\begin{aligned} \rho(jhM_0) &= |h| / \min_{\lambda \in \Lambda(A)} |j\omega - \lambda| \\ &= |h|/r \ . \end{aligned}$$

where

$$r = \min_{\lambda \in \Lambda(A)} |j\omega - \lambda|. \tag{24}$$

□

This lemma is also important in the development of a pole and zero cancelling routine.

4 Pole and Zero Cancellation

Polynomial interpolation approximation works well unless the LTI system being analyzed has poles or zeros near the imaginary axis. Such poles and zeros are called resonant poles and resonant zeros. The following examples provide the general idea of the effect.

Example: The deleterious effect of poles and zeros can be illustrated by means of a scalar rational function example. Consider

$$f(x) = \frac{1 + 2x + 3x^2}{1 - x} = 1 + 3x + 6x^2 + \ldots \tag{25}$$

We can use a polynomial approximation to evaluate this function at various values of x. Suppose that we choose a second-order polynomial approximation:

$$\tilde{f}(x) = 1 + 3x + 6x^2.$$

If we evaluate $\tilde{f}$ for $x = 0.01$ and $x = 0.99$, we get the approximations

$$\begin{aligned} \tilde{f}(0.01) &= 1.0306, \\ \text{and} & \\ \tilde{f}(0.99) &= 9.8506, \end{aligned}$$

respectively. If we compare these to the actual values,

$$\begin{aligned} f(0.01) &= 1.0306061 \\ \text{and} & \\ f(0.99) &= 592.03, \end{aligned}$$

we can see that as we approach a pole, a much higher-order approximation is required in order to get even modest accuracy.

However, if initially we eliminate the pole before we make our calculations for values near $x = 1$, the accuracy of the method increases dramatically. Again, use a second-order approximation with pole cancellation, and we get

$$\bar{f}(x) = (1-x)\frac{1+2x+3x^2}{1-x} = (1+2x+3x^2).$$

After evaluating $\bar{f}$, we let

$$\tilde{f} = \frac{\bar{f}}{(1-x)}.$$

As can be seen in this case, the second-order interpolation is exact. In most cases, however, only a marked increase in accuracy is realized.

In order to cancel a pole in our frequency response, we write

$$M(h) = \frac{(jh+j\omega-\lambda)M(h)}{(jh+j\omega-\lambda)}$$

and then find a polynomial approximation of $(jh+j\omega-\lambda)M(h)$. Therefore, our interpolation becomes

$$G(\omega+h) = \frac{G_0 + G_1(h-h_0) + \cdots + G_k(h-h_0)\cdots(h-h_{k-1})}{(jh+j\omega-\lambda)}, \tag{26}$$

where the coefficient matrices are for a system devoid of the resonance problem. The following lemma shows how to compute the new coefficient matrices while preserving the form of the interpolating series.

Lemma 3 *Let $h_0, \ldots, h_k$ be given and set $h_m = 0$ for all $m > k$. For $|h| < r$, where r is defined in (24), define the coefficient matrices $F_m^{(n)}$ implicitly via*

$$\prod_{\ell=1}^{n}(jh+j\omega-\lambda_\ell)M(h) = \sum_{m=0}^{+\infty} F_m^{(n)} \prod_{i=0}^{m-1}(h-h_i) \quad . \tag{27}$$

Then

$$F_m^{(0)} = (-j)^m \prod_{i=0}^{m} M(h_i) \quad , \tag{28}$$

and

$$F_m^{(n)} = (jh_m + j\omega - \lambda_n)F_m^{(n-1)} + jF_{m-1}^{(n-1)}, \qquad m = 0, 1, \ldots \tag{29}$$

where we define $F_{-1}^{(\ell)} = 0$ for all ℓ.

Proof: Equation (28) is immediate from (23). By (27),

$$(jh + j\omega - \lambda_n) \sum_{m=0}^{+\infty} F_m^{(n-1)} \prod_{i=0}^{m-1} (h - h_i) = \sum_{m=0}^{+\infty} F_m^{(n)} \prod_{i=0}^{m-1} (h - h_i). \tag{30}$$

The assumptions that $h_m = 0$ for all $m > k$ and $|h| < r$ ensure that the series in (30) are absolutely convergent. We may thus rearrange the left-hand summation as follows:

$$\begin{aligned}
&(jh + j\omega - \lambda_n) \sum_{m=0}^{+\infty} F_m^{(n-1)} \prod_{i=0}^{m-1} (h - h_i) \\
&= \sum_{m=0}^{+\infty} (jh_m + j\omega - \lambda_n + j(h - h_m)) F_m^{(n-1)} \prod_{i=0}^{m-1} (h - h_i) \\
&= \sum_{m=0}^{+\infty} (jh_m + j\omega - \lambda_n) F_m^{(n-1)} \prod_{i=0}^{m-1} (h - h_i) \\
&\quad + \sum_{m=0}^{+\infty} j F_m^{(n-1)} \prod_{i=0}^{m} (h - h_i) \\
&= \sum_{m=0}^{+\infty} (jh_m + j\omega - \lambda_n) F_m^{(n-1)} \prod_{i=0}^{m-1} (h - h_i) \\
&\quad + \sum_{m=0}^{+\infty} j F_{m-1}^{(n-1)} \prod_{i=0}^{m-1} (h - h_i) \\
&= \sum_{m=0}^{+\infty} \left((jh_m + j\omega - \lambda_n) F_m^{(n-1)} + j F_{m-1}^{(n-1)} \right) \prod_{i=0}^{m-1} (h - h_i).
\end{aligned}$$

Comparison with (30) gives (29). □

If we need to cancel resonant zeros, we then need to find a polynomial approximation of $\frac{M(h)}{(jh+j\omega-z)}$. The following lemma illustrates how this is done.

Lemma 4 *Let $h_0, h_1, \ldots, h_k$ be given and set $h_m = 0$ for all $m > k$. For $|h| < r$, where r is defined in (24), define the coefficient matrices $D_m^{(n)}$ implicitly via*

$$M(h) = \prod_{\ell=1}^{n} (jh + j\omega - z_\ell) \sum_{m=0}^{+\infty} D_m^{(n)} \prod_{i=0}^{m-1} (h - h_i). \tag{31}$$

Then

$$D_m^{(0)} = (-j)^m \prod_{i=0}^{m} M(h_i), \tag{32}$$

and

$$D_m^{(n)} = (D_m^{(n-1)} - j D_{m-1}^{(n)})/(jh + j\omega - z_n), \qquad m = 0, 1, \ldots \tag{33}$$

where we define $D_{-1}^{(\ell)} = 0$ for all ℓ.

Proof: The proof is similar to that of the preceding lemma except that we start with the identity

$$(jh + j\omega - \lambda_n) \sum_{m=0}^{+\infty} D_m^{(n)} \prod_{i=0}^{m-1} (h - h_i) = \sum_{m=0}^{+\infty} D_m^{(n-1)} \prod_{i=0}^{m-1} (h - h_i) \tag{34}$$

and continue from there. □

5 Frequency Response Interpolation Algorithm

Step 1 Solve for X_0 in

$$(j(h_0 + \omega)I - A)X_0 = B \ , \tag{35}$$

and then solve recursively for $X_1, \ldots, X_k$ in

$$(j(h_m + \omega)I - A)X_m = -jX_{m-1} \ . \tag{36}$$

Step 2 Let $X_m^{(0)} = X_m$ and define

$$X_m^{(\ell)} = (jh_m + j\omega - \lambda_\ell)X_m^{(\ell-1)} + jX_{m-1}^{(\ell-1)} \ , \quad 0 \le m \le k \ , \tag{37}$$

with $X_{-1}^{\ell} \equiv 0$ for $0 \le \ell \le n$.

Step 3 Let $X_m^{(n)(0)} = X_m^{(n)}$ and define

$$X_m^{(n)(\ell)} = (X_m^{(n)(\ell-1)} - jX_{m-1}^{(n)(\ell)})/(jh_m + j\omega - z_\ell) \ , \quad 0 \le m \le k \ , \tag{38}$$

with $X_{-1}^{\ell} \equiv 0$ for $0 \le \ell \le l$.

Step 4 Form the coefficient matrices $G_0, \ldots, G_k$ via

$$G = CX_m^{(n)(l)} \ . \tag{39}$$

Step 5

$$\begin{aligned} G(\omega + h) \ &= \ (G_0 + G_1(h - h_0) + \cdots + G_k(h - h_0)\cdots(h - h_{(k-1)})) \\ &\quad \cdot \frac{\prod_{i=1}^{l}(j\omega + jh - z_i)}{\prod_{m=1}^{n}(j\omega + jh - \lambda_m)} \end{aligned} \tag{40}$$

Remark
The method used to solve the recursive linear systems in the first step of the algorithm depends on the initial structure of the LTI system being investigated. If the system has an exploitable structure such as sparsity, an algorithm that exploits that particular structure will be used. If no such structure exists, an initial similarity transformation, most likely to upper Hessenberg form, will be applied to the system.

6 Interpolation Point Selection

The placement of the interpolation points is of great importance in getting a good approximation to the frequency response. We have tested three simple methods to place the interpolation points: linear, loglinear, and Chebyshev. We have also tested placement using the *a priori* information of the pole locations.

Since frequency response is usually plotted against frequency on a log scale, the use of linearly spaced interpolation points does not usually perform well. It places too many points at the end of an interval. Both the loglinear and the Chebyshev interpolation point placements have shown promise. The loglinear placement technique usually gives an excellent approximation in the beginning to the middle of an interval, but sometimes can fail miserably at the end of an interval. The Chebyshev interpolation points (see [8]) spread the approximation error fairly evenly across the interval. However, several times the error of the Chebychev selection, although acceptable, is larger than that of the acceptable range of a loglinear interpolation of the same size. Currently, we are investigating possible hybrid techniques to exploit the best of both placement schemes.

In the cases where we have tried placing interpolation points with the knowledge of the poles and zeros of the system the results have been mixed in comparison to the two previously mentioned techniques. What has been learned is that under no circumstances should the interpolation points be the same as the resonant frequency of a resonant pole or zero. However, placing an interpolation point near the resonant frequency improves the approximation significantly.

7 Conclusion

In this paper we have presented a rational interpolation method for computing the frequency response of a system. A significant computational savings can be achieved over several of the current methods for computing a frequency response. An error analysis for the method, together with other details, can be found in [3].

The method presented in this paper avoids the numerical problem of subtraction of near equal quantities in the difference terms by using the resolvent identity of Theorem 1. Also, simple pole and zero cancellation techniques significantly increase the accuracy of the algorithm.

We are currently writing a software package to implement the algorithm in this paper. In addition, we are extending this algorithm for use with descriptor systems.

References

[1] Forsythe, G.E., M.A. Malcolm, and C.B. Moler, *Computer Methods for Mathematical Computations*, Prentice-Hall, Englewood Cliffs, NJ, 1977.

[2] Grace, A., A.J. Laub, J.N. Little, and C. Thompson, *Control System Toolbox, User's Guide*, The MathWorks, Natick, MA, Oct. 1990.

[3] Kenney, C.S, S.C. Stubberud, and A.J. Laub, "Frequency Response Computation Via Rational Interpolation," *Proceedings of the 1992 IEEE Symposium on Computer-Aided Control System Design*, pp. 188–195, Napa, Calif, March 1992, IEEE Control Systems Society.

[4] Laub, A.J., "Efficient Multivariable Frequency Response Computations," *IEEE Trans. Auto. Control*, AC-26 (1981), pp. 407–408.

[5] Lee, E.A., *Control Analysis Program for Linear Systems for MS-DOS Personal Computers, User Manuals*, CAPLIN Software, Redondo Beach, Calif., April 1990.

[6] Meier, W.A., "Sparse Matrix Applications in Computer-Aided Control System Design," *Proceedings of the 1992 IEEE Symposium on Computer-Aided Control System Design*, pp. 196–203, Napa, Calif, March 1992, IEEE Control Systems Society.

[7] Osterby, O. and Z. Zlatev, *Direct Methods for Sparse Matrices*, Springer-Verlag, Berlin, 1983.

[8] Rivlin, T.J, *The Chebyshev Polynomials*, John Wiley and Sons, New York, 1974.

[9] Schendel, U., *Sparse Matrices: Numerical Aspects with Applications for Scientists and Engineers*, Ellis Horwood, Chichester, 1989.

[10] Walker, R.A., *Computing the Jordan Form for Control of Dynamic Systems*, PhD thesis, Stanford University, Department of Aeronautics and Astronautics, March 1981.

An Application of the IMC Software to Controller Design for the JPL LSCL Experiment Facility

Guoming Zhu and Robert E. Skelton

Space Systems Control Laboratory
1293 Potter Engineering Center
Purdue University
West Lafayette, IN 47907

ABSTRACT

A software package which Integrates Model reduction and Controller design (The IMC software) is applied to design controllers for the JPL Large Spacecraft Control Laboratory Experiment Facility. Modal Cost Analysis is used for the model reduction, and various Output Covariance Constraints are guaranteed by the controller design. The main motivation is to find the controller with the "best" performance with respect to output variances. Indeed it is shown that by iterating on the reduced order design model, the controller designed does have better performance than that obtained with the first model reduction.

1. INTRODUCTION

The objective of this research is to develop controller design software IMC (Integrated Modeling and Control) for a realistic flexible space structure control problem. The main interests are two-fold: i) the design of high performance fixed order dynamic controllers for this complex structure, and ii) to test the efficacy of the IMC software for the search of the controller with the "best" performance, among all model based controllers.

Almost all available controller design techniques are based upon a given model of the physical plant. In general, perfect models are impossible to construct. Modeling error exists in every mathematical model used for control design. There are three ways to deal with modeling error in a controller design procedure. First, one may use robust control theory. The controller designed with robust control theory is tolerant to a specified set of modeling errors. But a poor model may lead to a poor controller even if the controller is robust with respect to the given model. Second, one may treat the modeling and controller design as a combined problem, and try to refine the design model to find one that is "appropriate" for controller design in the sense of best closed loop operation. The third method is adaptive control which intends to adjust the controller in real-time to compensate for modeling errors.

From our experience a *nominal* controller design procedure based on an "appropriate" model may yield better performance than a *robust* controller that is based on an poor model (say, given by finite element modeling or identification). Hence, we use the second method to obtain a design model that is more compatible to the particular controller design than the other two methods.

In this research the integrated design procedure is applied to design controllers for the LSCL Experiment Facility. Assuming that a "true enough" high order mathematical model can be obtained by some modeling method (analytical or by identification), our procedure reduces the "true enough" model (we shall call this the "evaluation model") to an order appropriate for full order controller design based on the reduced order model. Repeating the model reduction and controller design by using *closed* loop information such that the process is convergent, the integrated procedure produces a design model "appropriate" to the corresponding controller.

The model reduction technique used in this experiment is the Modal Cost Analysis (MCA) which calculates each modal contribution V_i to a weighted quadratic cost function [7-9].

$$V \triangleq \lim_{k \to \infty} \mathcal{E} y^T(k) Q y(k) = \sum_{i=1}^{N} V_i \ , \qquad (1.1)$$

where N is the number of modes in the model. The smallest contribution (smallest V_i) indicates the modes to be deleted in the reduced model. Closed form analytical expression of V_i are available, see [8].

Two controller design methods (BOCC and EOL_∞) were applied to this experiment. The BOCC algorithm [1-4] designs controllers minimizing the control effort subject to output covariance constraints (for zero mean white noise input). The BOCC algorithm can be also used to satisfy the output ℓ_∞ constraints when the input is an ℓ_2 disturbance. The EOL_∞ algorithm [5] is an extension of the deterministic interpretation of the BOCC. The EOL_∞ designs controllers to satisfy given output ℓ_∞ constraints when the input ℓ_2 disturbances have an outer product matrix upper bound. The main difference between those two design algorithms is that the BOCC algorithm only iterates on the feedback gain, but the EOL_∞ algorithm iterates on both estimator and control feedback gains. We only present the BOCC results in this paper. The definition and solution of the BOCC and EOL_∞ can be found in [3-5].

There are two iteration loops in the IMC software, one inner loop and one outer, used to realize the integration of model reduction and controller design. The inner loop, called the α-loop, intends to obtain the controller for "best" performance (with respect to the *evaluation model*) with the given reduced order model (called the *design model*) by gradually increasing the required performance (smaller variance constraints). The outer loop iterates on the design model to make the design model be "appropriate" to the corresponding controller with the "best" performance.

The paper is organized as follows. Section 2 combines model reduction and controller design techniques which is the main philosophy of the IMC software presented in Section 3. The controller design and test results are presented in Section 4. The last section adds some conclusions.

2. INTEGRATION OF MODEL REDUCTION AND CONTROLLER DESIGN

It is well known that finding a good model for control design is a difficult problem because of uncertain parameters, nonlinearity and neglected dynamics of the physical system. It is impossible to separate the modeling and controller design

problems. For example, considering a linear system with a nonlinear actuator, one may apply linear control theory to design a controller. In this case the nonlinear actuator should be linearized at some nominal point, but the nominal point is related to the control signal level of the controller which will be designed after linearization of the actuator model. Consequently, the modeling and controller design problems become an iterative process, see the examples in [6].

In this section we mainly consider the effect of the neglected dynamics of the physical system. We are trying to obtain the "best" performance for a high order given physical system with a fixed order controller. There are at least three ways to find a fixed order controller for a given linear system. The first way is to design a fixed order controller directly. The second is to design a full order controller first and then reduce the controller to the required order. The last one is to reduce the model first and then do the full order control design based on the reduced order model. The advantage of the first method is that the performance of the closed loop system with the designed controller is guaranteed. But unfortunately there exists no closed form for the design of such controllers. Since full order controller design methods are available for most control theories, H_∞, LQG and so on, we will use a variation of the third method, we call *the integration of model reduction and controller design*, to design reduced order controllers.

The integrated design procedure, utilizing Modal Cost Analysis for model reduction and the BOCC or EOL_∞ for controller design, is shown in Figure 1. The design procedure searches for the controller with the "best" performance by tuning the design model until the design model corresponds to the controller with the "best" performance. This procedure is developed under the following basic assumption that the only modeling errors existing in the design model are from the model reduction, i.e., the evaluation model is assumed to be "true enough". This assumption allows us to evaluate the designed controller based on the evaluation model, prior to hardware testing in the lab. Of course, we also compare these analytical results with the experimental results.

The evaluation model in Figure 1 can be obtained either from system identification or from mathematical modeling, e.g., the finite element model combining with the sensor and actuator dynamics. Generally, the size of the evaluation model is too large for controller design. Hence, the model reduction is necessary.

The cost function defined in (1.1) used in the model reduction is the summation of the weighted output variance with respect to the white noise input. Note that the modal cost is very much dependent upon the input and output weighting matrices $\overline{W} = \text{diag}[W, R]$ and Q, where the input weighting matrix $\overline{W}$ is used to compute the output covariance. Hence, the choice of those two matrices will directly effect the model reduction. How to choose Q and $\overline{W}$ is a major subject of this paper. For the first iteration of this experiment, matrix W is the input white noise covariance matrix W_p, and Q and R are diagonal matrices whose elements are the inversed square of the hard limitation on inputs and outputs, respectively.

The main philosophy of our α-loop in Figure 2 is to obtain a sequence of controllers from low control effort to high. Here α denotes the controller number. The controller sequence is obtained by reducing the required performance specification during controller design.

The main purpose of the BOCC α-loop is to obtain the "best" performance with the given (reduced order) *design model* (obtained from MCA model reduction of the *evaluation model*), which is expressed in the following form

$$\left.\begin{aligned} x_p(k+1) &= A_p x_p(k) + B_p u(k) + D_p w_p(k) \\ y_p(k) &= C_p x_p(k) \\ z(k) &= M_p x_p(k) + v(k) \end{aligned}\right\} . \qquad (2.1)$$

The BOCC α-loop starts with the evaluation and design models. Suppose that the output y_p can be divided into m output groups $\hat{y}_i$. Let $Y_i(0)$ (i = 1, 2, ..., m) denote the open loop output covariance of the *evaluation model* for output group $\hat{y}_i$, assuming that the open loop system is asymptotically stable. Define L_i (i = 1, 2, ..., m) to be a lower bound of the output covariance of the closed loop system with any full order controller. Hence, any specification which is less than or equal to L_i is unachievable with respect to the design model. Then the specification matrix $\overline{Y}_i(\alpha)$ (i = 1, 2, ..., m) can be generated by the following equation

$$\overline{Y}_i(\alpha) = [Y_i(0) - L_i](1 - \beta)^{\alpha} + L_i \ , \ \alpha = 1, 2, ..., \alpha_m \ , \qquad (2.2)$$

where $0 < \beta < 1$ is a design parameter and α is the integer counter (iteration number for the α-loop). Note that the specifications are gradually reduced as α increases. The main reason to use (2.2) to produce specification $\overline{Y}_i(\alpha)$ is to make the *change* of specification small (from one iteration to the next) when it is close to its lower bound L_i.

With each set of design specifications $\bar{Y}_i(\alpha)$, the BOCC algorithm will produce a controller with index α, called the αth controller, using the *design model* (2.1). The closed loop system with the *design model* and the αth controller is asymptotically stable because the BOCC controller is an LQG controller with a special choice of the output weighting matrix. But the closed loop system with the *evaluation model* may not be stable. If the closed loop system with respect to the *evaluation model* is unstable, the α-loop will be terminated, according to the BOCC α-loop diagram in Figure 2, otherwise the output covariance matrices $Y_i^e(\alpha)$ and $Y_i^d(\alpha)$ with respect to the *evaluation* and *design models* will be computed for future use.

Since the open loop system is asymptotically stable, the closed loop system will be asymptotically stable if the controller gain is small enough. As the control gains increase, i.e., α increases, the closed loop system with respect to the *evaluation model* may become unstable. Hence, a plot similar to Figure 3 can be generated for analysis. We use α_b to denote the point with the "best" performance with respect to the evaluation model. The information on the α_bth controller will be used for the new model reduction because we want the design model to be "appropriate" to the controller with the "best" performance. The new output and input weighting matrices Q and R will be computed in the following way

$$Q_i = \alpha_q Q_i(\alpha_b) + (1 - \alpha_q) \left| \bar{\sigma}[Y_i^e(\alpha_b)] - \bar{\sigma}[Y_i^d(\alpha_b)] \right| I_{m_i} \tag{2.3a}$$

$$Q = \text{block diag}[Q_1, Q_2, ..., Q_m] \tag{2.3b}$$

and

$$R = \alpha_r R(0) + (1 - \alpha_r)\text{diag}([..., \left| U_j^e(\alpha_b) - U_j^d(\alpha_b) \right|, ...]) \tag{2.4}$$

where $0 \le \alpha_q \le 1$ and $0 \le \alpha_r \le 1$ are design parameters. R(0) is the controller channel input weighting matrix used in the first MCA model reduction. $U_j^e(\alpha_b)$ and $U_j^d(\alpha_b)$ ($j = 1, 2, ..., n_u$) are the closed loop input variances of the α_bth controller with respect to the evaluation and design model respectively. Similarly, $Y_i^e(\alpha_b)$ and $Y_i^d(\alpha_b)$ are the output covariances. The main reason to add these items to correct the input and output weighting matrices is to reduce the differences between the evaluation and design models for the α_bth controller.

$Q_i(\alpha_b)$ is the convergent output weighting matrix for the ith block during the design of the α_bth controller. The importance of $Q_i(\alpha_b)$ can be clearly observed in the OVC problem (a special case of the BOCC problem when each block has dimension

equal to one). It is noted that during the OVC design iteration procedure the output weighting matrix Q is adjusted so that if a particular output specification $\overline{Y}_i$ is not achieved, the corresponding Q_i will be increased according to the discrepancy between the current output variance Y_i and the specification $\overline{Y}_i$. Consequently, those outputs with hard-to-achieve specifications (indicated by $Y_i = \overline{Y}_i$) will end up with large Q_i's, and those with easy-to-achieve specifications ($Y_i < \overline{Y}_i$) will have the small Q_i's. In fact, for those outputs that end up with variances smaller than the corresponding $\overline{Y}_i$'s the final convergent Q_i's will be zero. This implies that these output constraints are not important and can be disregarded during design. However, at the beginning, this information is unknown. As a result, the convergent Q appropriately reflects the importance of each output with respect to the given specification. This property is very helpful for the model reduction using Modal Cost Analysis, because MCA calculates the contribution of each mode to a weighted output cost $\mathcal{E}_\infty y^T Q y$ and deletes the least important modes accordingly. Hence, if the weighting matrix can appropriately reflect the importance of each output, then the reduced model using MCA will keep the information which is important to the required performance.

The controller evaluation part mainly evaluates the designed controllers in the α-loop study to see whether the performance is satisfactory or not. The evaluation (plot in Figure 3) will provide the information to adjust these design parameters, e.g., α_q, α_r and so on, in the α-loop study.

As a result, it is clear now that in the integration of model reduction and controller design there are two iterative loops, the Q-loop and α-loop. The Q-loop is used to *combine the model reduction and the controller design process* such that at convergence the design model corresponds to the controller with the "best" performance. The α-loop intends to *search for the controller of the "best" performance with respect to the evaluation model, and a given design model.*

3. THE IMC SOFTWARE

An IMC (Integration of Model reduction and Controller design) software has been developed to integrate the model reduction and controller design process presented in the last section. The IMC software makes it possible to obtain the rapid redesign capability in a workstation environment using MATLAB.

The idea of the integrated procedure of model reduction and controller design was first applied to design controllers for NASA's Minimast at Langley Research Center [10]. The realization of this integrated idea needs a certain amount of computation, and some expert is needed to manage the whole integrated design process. Some parameters must be chosen, and if changed, the whole process must be repeated. In order to reduce the repeated work during the integrated controller design process, we are motivated to put all the independent software modules, e.g., MCA model reduction, OVC, BOCC and EOL_∞ controller design software, together to form a software package IMC. If some information of the physical system (like pulse responses), or a mathematical model is available, the software will go through the whole integrated process automatically such that a person who has no knowledge of MATLAB can design controllers using this software. This software is programmed in MATLAB which is available in most workstations.

The main idea of this software is shown in Figure 1. For a physical system, the mathematical model of the given system can be obtained by identification or by mathematical modeling. Then the software starts either with the signals which are necessary for identification or with the given mathematical model. Based on the given model or identified model, the integrated process will produce controllers for evaluation. If the requirements of the evaluation are satisfied, the controllers can be implemented in the hardware equipment for testing.

For this experiment, we used the finite element model plus sensor and actuator dynamics as our evaluation model. The IMC controller design process is shown in Figure 2. The IMC software (Version imc_g03) has seven modules as follows.

i) Constructing a continuous and discrete evaluation model from the given finite element model.

ii) Constructing a design model by MCA model reduction.

iii) Constructing a discrete evaluation model by identification (not available).

iv) α-loop study — discrete OVC controller design.

v) α-loop study — discrete BOCC controller design.

vi) α-loop study — discrete EOL_∞ controller design.

vii) Evaluation Tool.

To design a controller from the finite element model, one can use modules i) and ii) to form the discrete state space evaluation and design models. Choosing an α-loop controller design module, (for example, the BOCC α-loop study), one can iterate on the modules ii) and iv) to carry on the Q-loop. After the Q-loop has converged, one can evaluate designed controllers using module vii). Now let us introduce each module in detail.

Using frequencies and mode shape vectors obtained from the finite element analysis, the first module combines the finite element model with sensor and actuator dynamics to form a continuous time state space model. By choosing a proper sampling rate, the discrete evaluation model can be obtained by discretizing the continuous time model. In the case that the order of the finite element model is relatively high, an additional (optional) MCA model reduction can be applied to obtain a lower order evaluation model.

The MCA model reduction module includes two kinds of MCA model reduction routines, continuous and discrete versions. The discrete reduced order model can be obtained from the discretized high order model by both continuous and discrete MCA model reductions, because both MCA results provide the contribution of each mode to the total cost, which can be used to decide which mode should remain in the design model. Also a modal cost analysis table will be generated.

Using the pulse responses or white noise responses, the identification module (not yet available) will produce an identified evaluation model by the q-Markov COVER method in [11-13].

The α-loop study modules for the OVC, BOCC and EOL_∞ controller design are similar. Here we only discuss the BOCC α-loop study module. The block diagram of the BOCC α-loop study is shown in Figure 2. The main philosophy of the α-loop study is to obtain a sequence of the controllers from low control effort to high. As a result, the controller of the "best" performance can be obtained among those controllers.

The evaluation tool box module includes seven blocks described as follows.

i) Plotting pole locations.

ii) Discrete simulation of pulse responses.

iii) Plotting output variances with respect to α.

iv) Simulation with arbitrary input functions.

v) Continuous simulation of pulse responses.

vi) Transferring MATLAB data file to ASCII code data files.

vii) Plotting FORTRAN simulation responses.

4. CONTROLLER DESIGN AND EXPERIMENTAL RESULTS

4.1 System Description and State Space Model

The JPL Large Space Control Laboratory Experiment Facility [14] is shown in the Figure 4. The main component of the apparatus consists of a central hub to which 12 ribs are attached. The diameter of the dish-like structure is slightly less than about 19 feet, the large size being necessary to achieve the low frequencies desired. The ribs are coupled together by two rings of wires which are maintained under nearly constant tension. Functionally, the wires provide coupling of motion in the circumferential direction which would otherwise occur only through the hub. The ribs, being quite flexible and unable to support their own weight without excessive droop, are each supported at two locations along their free length by levitators. A levitator assembly consists of a pulley, a counterweight, and a wire attached to the counterweight which passes over the pulley and attaches to the rib. The hub is mounted to the backup structure through a gimbal arrangement so that it is free to rotate about two perpendicular axes in the horizontal plane. A flexible boom is attached to the hub and hangs below it, and a weight, simulating the feed horn of an antenna, is attached at the bottom end of the boom. A 3 foot long boom is used for this experiment.

Actuation of the structure is as follows. Each rib can be individually manipulated by a rib-root actuator mounted on that rib near the hub. A rib root actuator reacts against a mount which is rigidly attached to the hub. In addition, two actuators are provided which torque the hub about its two gimbal axes. The hub torquers do not provide torque directly but rather are linear force actuators which produce torque by pushing or pulling at the outer circumference of the hub. The placement of these actuators guarantees good controllability of all of the flexible modes of motion. The locations of the actuators are shown in Figure 5. Two hub actuators are used for control in x and y directions. They are denoted by HA1 and HA10 respectively. The transfer

function from command torque to net torque is shown as follows.

$$\frac{T(s)}{T_c(s)} = \frac{3947.8}{s^2 + 44.43s + 3947.8} \quad . \tag{4.1}$$

Only four rib root actuators are used in this experiment. They are rib root actuators on ribs 1, 4, 7 and 10, denoted by RA1, RA4, RA7 and RA10. The transfer function from the command force to the net force is

$$\frac{F(s)}{F_c(s)} = \frac{24674}{s^2 + 111.1s + 24674} \quad . \tag{4.2}$$

The sensor locations are also shown in Figure 5. First, each of the 24 levitators is equipped with a sensor which measures the relative angle of the levitator pulley. The levitator sensors thus provide, in an indirect manner, the measurement of the vertical position of the corresponding ribs at the points where the levitators are attached. Four position sensors measure rib displacement at the rib-root actuator locations. Sensing for the hub consists of two rotation sensors which are mounted directly at the gimbal bearing. There are a total of 24 levitator sensors used for measurements. They are denoted by LS1 to LS24. The transfer function from the physical output to the measurement is assumed to be one because the optical sensor has pretty wide bandwidth. Two hub optical angle sensors, HS1 and HS10, are used to measure the hub angle in x and y directions. Similarly, the transfer function is assumed to be one. Only four rib root sensors, RS1, RS4, RS7 and RS10, are available for measurements. The dynamics are omitted (the transfer function is assumed to be one). Since the hub and rib root sensors are very noisy, a first order filter is applied for each of those six sensors. The transfer function of the filter is

$$H(s) = \frac{502.65}{s + 502.65} \quad . \tag{4.3}$$

A summary of outputs and inputs is contained in Table 1.

JPL created two finite element models with 30 and 84 modes respectively. The 30 mode finite element model is used in this experiment. All modes with natural frequencies less than 10 Hz are given in Table 2. Let the structure be described in its modal coordinates by the following

$$\left.\begin{aligned} \ddot{\eta}_i + 2\xi_i\omega_i\dot{\eta}_i + \omega_i^2\eta_i &= b_i^T u_a \ , \ i = 1, 2, ..., 30 \\ y &= \sum_{i=1}^{30} p_i\eta_i \end{aligned}\right\} , \tag{4.4}$$

where u_a is the actuator output signal and y is the displacement vector co-located with the sensor inputs. JPL provided 30 frequencies (ω_i, i = 1, 2, ..., 30) and 30 mode shapes (p_i, i = 1, 2, ..., 30) obtained from a finite element analysis.

The actuator output signal u_a is now filtered by hub actuator and rib root actuator dynamics modeled by the following

$$\left.\begin{aligned} \dot{x}_a &= A_a x_a + B_a u \\ u_a &= C_a x_a + w_p \end{aligned}\right\} , \tag{4.5}$$

where u is composed of the command signals to the hub and rib root actuators, and w_p is the actuator noise with intensity $\overline{W}_p$. The measurement output z now can be presented by

$$\left.\begin{aligned} \dot{x}_s &= A_s x_s + B_s y \\ z &= C_s x_s + D_s y + v \end{aligned}\right\} , \tag{4.6}$$

where v is the sensor noise with intensity $\overline{V}$. Combining models (4.4-4.6), we can obtain a continuous time full order model. Since the frequencies of all modes in our model are less than 5 Hz, we discretize the continuous model at 25 Hz which is the computer sample rate. The discrete *evaluation model* is as follows.

$$\left.\begin{aligned} x_e(k+1) &= A_e x_e(k) + B_e u(k) + D_e w_p(k) \\ y_p(k) &= C_e x_e(k) \\ z(k) &= M_e x_e(k) + v(k) \end{aligned}\right\} , \tag{4.7}$$

where w_p and v are white noise with covariance matrix $W_p = \overline{W}_p/25$ and $V = \overline{V}/25$ respectively.

4.2 The BOCC Controller Design and Experimental Results

The design strategy used here is the integration of model reduction and controller design introduced in the last section. Using the open loop experimental results at JPL, we adjusted some frequencies, damping coefficients and input/output magnitudes such that the responses of the finite element model combining with the sensor and actuator dynamics were closer to the experimental pulse responses. The adjusted frequencies and damping coefficients are shown in Table 2. The magnitude coefficients vary in different designs.

4.2.1 The OVC Design and Experimental Results

The OVC Controller Design

We start controller design with the OVC algorithm because the OVC problem is a special case of the BOCC. Note that in this case the constraints on the output covariance matrices reduce to those on output variances. Hence, all the constraints are scalars. Some errors in the finite element model of the structure are found. The errors result from the sign convention on the hub sensors. Also the units used in the finite element model and those used in the real-time control computer are different. The units used in the measurement and output are meter and radian in the finite element model but those in real-time control computer are milli-meter and milli-radian. We use input/output scaling matrices to overcome unit differences and finite element modeling errors. The input scaling matrix is

$$S_u = \text{diag}[0.5, -1, 1, 1, 1, 1]e{+}3 \ , \tag{4.8}$$

and the output scaling matrix S_y is a diagonal matrix with unity diagonal entries except the 26th diagonal element which is negative unity. The finite element model provided by JPL is modified by redefining the input vector $S_u u_a$ and output vector $S_y y$ as u_a and y in (4.4) respectively. The evaluation model used in this design is obtained by combining the modified finite element model, sensor and actuator dynamics in (4.4-4.6). The evaluation model is discretized at a sampling frequency 25 Hz. The state space realization of this model is in the form (4.7), where A_e, B_e, D_e, C_e and M_e are the system matrices respectively of dimension 78×78, 78×6, 78×6, 30×78 and 30×78, and u, y_p and z are input, output and measurement vectors, respectively, as described in Table 1. Vector w_p is the system noise from hub and rib root actuators with the following variances,

$$W_p = \text{diag}[0.04, 0.04, 0.04, 0.04, 0.04, 0.04] \ . \qquad (4.9)$$

Vector v is the measurement noise of the levitator, hub and rib root sensors with the following variance,

$$V = \text{block diag}\ [1.5625I_{22}, 3.0500I_2, 0.2500I_4] \ , \qquad (4.10)$$

where the suffix of matrix I indicates the dimension of the identity matrix. All the variances are taken from signal to noise ratios.

In order to decide the order of the controller to be used, we designed 12th, 16th, 20th and 24th order controllers for the first Q-loop. It turns out that 16th, 20th and 24th order controllers have close performances with respect to the evaluation model. But the 12th order controller yields poor performance. Hence, we choose controller order to be 16. The 16th order controller is designed by using the design methodology presented in the last section.

The design parameters used in this design for the Q–loop are

$$\beta = 0.2 \ ; \ \alpha_q = 0.5 \ ; \ \alpha_w = 0.5 \ . \qquad (4.11)$$

We compute the open loop output variance $Y_i(0)$ with respect to the evaluation model, and the lower bound L_i for the design model.

The Modal Cost Analysis results for the different Q–loops 1 and 3 can be found in Table 3. The first 8 dominant modes in the Q-loop 1 and 3 are the same. Hence, in this case the Q-loop will not converge but oscillate between two models which are obtained in Q-loop 1 and 2. Since the "best" performance with respect to the evaluation model is obtained in Q-loop 2, we use the reduced order model of Q-loop 2 which keeps modes 2, 1, 14, 13, 27, 28, 4 and 6 as the final design model. The iteration on the Q-loop is terminated at Q-loop 3.

Note that for each Q-loop the OVC α-loop algorithm produces a number of controllers from low to high control effort. The input/output variance curves of Q-loop 2 for the 16th order controller design are shown in Figure 6. The solid curve with "o" is the performance of the controllers obtained from the OVC algorithm evaluated with the design model. The dashed line with "*" evaluates these controllers with the evaluation model. In the α-loop study, 13 controllers are produced. The first 12 controllers stabilize the evaluation model. The output variances of the closed loop system with respect to the evaluation model are plotted in Figure 6. The α-loop iterations terminate because the 13th controller destabilizes the evaluation model.

From Figure 6 it can be observed that the "best" performance for outputs 4 and 5 is provided by the closed loop systems obtained by evaluating controllers 8 and 9 with the evaluation model. Similar input/output variance curves of Q-loop 0 are obtained. In order to show the improvement of iterating the design model, we compute the differences between the output variances of Q-loop 0 and those of Q-loop 2 for each output. Let $Y_2(i,j)$ and $Y_0(i,j)$ denote the ith output variance obtained by evaluating the jth controllers of Q-loop 2 and 0 with the evaluation model respectively. Plots $[Y_2(i,8) - Y_0(i,8)]/Y_0(i,8)$ and $[Y_2(i,9) - Y_0(i,9)]/Y_0(i,9)$ can be found in Figure 7. Since plots for controllers 8 and 9 are negative for almost all outputs, it is clear that the Q-loop improves the model reduction and controller design process, i.e., a better controller with respect to the evaluation model can be obtained by integration of model reduction and controller design.

The OVC Controller Experiment

Controllers 1, 3, 5, 7, 9, 11 and 13 of Q-loop 2 were tested on the JPL LSCL Experiment Facility. It is expected that the responses of controller 1 are pretty close to the open loop ones due to low control effort. The sequence of controllers allows one to do lab tests easily with little risk of damaging the system. Starting with low control effort controller, we can test controllers one by one with increased control effort, and stop the test when some controller destabilizes the system, or the oscillations become unacceptable. Because the control effort is increased gradually, the test facility will not be damaged. This is a nice feature of the integrated controller design strategy.

Since the system is highly damped, a pulse input with the width equal to a sample period (0.04 second) does not excite the system much. Hence, it is difficult to compute all the output variances by experimental data. We did the pulse experiments for each controller obtained in Q-loop 2 with pulse input on HA1 and HA10 respectively, where the magnitude of the pulse is 2 Newton-meters, and the width is 4 seconds (100 sample periods). We computed the input and output ℓ_2 norms in the following way

$$\|u(\cdot)\|_2^2 \triangleq \Delta^2 \sum_{k=101}^{p} u^T(k)u(k) \; ; \tag{4.12a}$$

$$\|y_i(\cdot)\|_2^2 \triangleq \Delta^2 \sum_{k=101}^{p} y_i^2(k) \; , \tag{4.12b}$$

where $\Delta = 0.04$ second is the sample period and $p = 1001$ is the number of sample

periods used for the test. Using (4.12) Figure 8 presents plots of input/output ℓ_2 norms for outputs 4, 5, 16 and 17, where the dotted line with "+" is associated with experimental data, dashed line with "*" is obtained from simulated data with the evaluation model, and the solid line with "o" is also from simulated data but with the design model. Note that we did not test every controller designed in the α-loop study of Q-loop 2. Hence, the "+" signs on Figure 5.4 are the ℓ_2 norms of the open loop responses and closed loop responses related to controllers 1, 3, 5, 7, 9, 11 and 13 from left to right. Due to noisy data and the difference between the finite element model and the real structure, lab tested ℓ_2 norm curves stay above the simulated ones. It is obvious that the 9th controller in the α-loop study of Q-loop 2 provides the "best" closed loop ℓ_2 response, which is consistent to the α-loop study result. In the α-loop study of Q-loop 2, the 13th controller destabilizes the evaluation model. It turns out that the closed loop system with that controller is unstable, too. It is clear from Figure 8 that the ℓ_2 norms blow up for controller 13. Hence, the test result agrees with the analytic one. The controller yielding the best performance experimentally is the best controller from the analytical designs.

Because the control experiment facility has no special channels to apply disturbances, the test has been done in such a way that the system is open loop at $t = 0$, when exciting signals are applied to the structure through control actuators. When the open loop command signals vanish, the control loop will be closed to conduct the closed loop experiment. Hence, the exciting signals applied through the actuator channels provide the initial condition for the structure.

The pulse responses of HA1 with controller 9 of Q-loop 2 are shown in Figure 9, where all input pulses are with the magnitude 2 Newton-meters and period 4 seconds. Hence, the closed loop control started at the 4th second, and open and closed loop responses are supposed to be the same for the first 4 seconds. It is obvious that the first two modes with frequency 0.0902 Hz are excited. From those responses, it is clear that the controller improves the performance of the system.

4.2.2 The BOCC Design and Experimental Results

The BOCC Controller Design

From the experience of the OVC controller design we feel that it is not necessary to use all outputs for controller design because of the symmetrical property of the structure. Hence, we choose to reduce the output number for the model reduction and control design process but still use all 30 measurements for the control design purpose. Outputs used for the BOCC design are

$$y_p = [y_1\ y_4\ y_{13}\ y_{16}\ y_{25}\ y_{26}\ y_{27}\ y_{28}]^T \ . \tag{4.13}$$

We group outputs in the following way

$$\hat{y}_1 \triangleq \begin{bmatrix} y_1 \\ y_4 \end{bmatrix} \ ; \ \hat{y}_2 \triangleq \begin{bmatrix} y_{13} \\ y_{16} \end{bmatrix} \ ; \ \hat{y}_3 \triangleq \begin{bmatrix} y_{25} \\ y_{26} \end{bmatrix} \ ; \ \hat{y}_4 \triangleq \begin{bmatrix} y_{27} \\ y_{28} \end{bmatrix} \ . \tag{4.14}$$

Hence, in this case constraints of the BOCC problem are 2×2 matrices for all output groups. Physical interpretation of this design is clear. Consider the output group $\hat{y}_3$ which is hub angle in x and y directions. Suppose that the maximal singular value of the constraint matrix is σ_3. Then the design will guarantee that the hub angle at any direction of x–y plane will be less than or equal to the square root of σ_3 times the input ℓ_2 norm.

According to the lab test of the OVC controllers, the output scaling matrix is changed as follows

$$S_y = \text{block diag}[I_{24},\ 1,\ -1,\ 0.1I_4] \ , \tag{4.15}$$

where the subscript of matrix I denotes the dimension of the identity matrix. The input scaling matrix remains unchanged.

The noise covariance matrix W_p is changed to a non-diagonal one to allow correlated noise on rib root actuators, where

$$W_p = \begin{bmatrix} 0.040 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.040 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.720 & 0.360 & 0.360 & 0.360 \\ 0.000 & 0.000 & 0.360 & 0.720 & 0.360 & 0.360 \\ 0.000 & 0.000 & 0.360 & 0.360 & 0.720 & 0.360 \\ 0.000 & 0.000 & 0.360 & 0.360 & 0.360 & 0.720 \end{bmatrix} . \quad (4.16)$$

We choose to design the 20th order controller for the BOCC problem. The design parameters used in this design for the Q-loop are

$$\beta = 0.2 \ ; \ \alpha_q = 0.5 \ ; \ \alpha_r = 0.5 \ . \quad (4.17)$$

The open loop output covariance matrix and lower bound of the output covariance matrix are computed in same way as in the OVC design.

The MCA model reduction results of the BOCC design is quite similar to the OVC Case. The Q-loop does not converge but oscillates between two design models. We plot the closed loop output maximal singular value curves with respect to the summation of input variances, where the maximal singular values are computed with respect to the design and evaluation models. The plot for Q-loop 2 is shown in Figure 10, where all symbols have the same meaning as those in the OVC design. It is observed that the "best" performance of output group 1, which is difficult to be achieved by the design, is provided by the 12th controller designed in Q-loop 2. Those controllers designed in Q-loop 2 were tested in the lab. In the α-loop, fifteen controllers are designed. All controllers stabilize the evaluation model. The 12th controller provides the "best" performance for output group 2.

The BOCC controller Experiment

The controllers 1, 3, 5, 7, 9, 11 and 13 of Q-loop 2 were tested on the JPL LSCL Experiment Facility. We define the ℓ_2 norm for each output group as

$$\|\hat{y}_i(\cdot)\|_2^2 \triangleq \Delta^2 \sum_{k=101}^{P} \hat{y}_i^T(k)\hat{y}_i(k) \ ; \ i = 1, 2, ..., 4 \ , \quad (4.18)$$

with the same definition on the input ℓ_2 norm as in the OVC case. The input/output ℓ_2 norm curves of the BOCC test are shown in Figure 11. From the input/output ℓ_2 norm plots in Figure 5.11, the "best" ℓ_2 performances are obtained by the 11th controller of Q-loop 2. The analytical ℓ_2 responses of output group 4 are quite different from the

test. We have no definitive answer, but we attribute such difference to nonlinearity and friction. All experiments have been done in the same way as the OVC design. The pulse responses of controller 11 with the same exciting signals as the OVC test can be found in Figure 12.

5. CONCLUSIONS

A reduced order controller design methodology with an integration of model reduction and controller design has been applied to the JPL LSCL Experiment Facility. This design strategy is an extension of that in [10,15]. The design strategy has provided a practical method for large space structure controller synthesis. The application of this strategy to the JPL LSCL Experiment Facility has met our high expectation.

From this experiment, we see that iterating between modeling and control (selecting an "appropriate" design model) plays an important role in the controller design. For two different design objectives (the OVC and BOCC designs), the iteration in the Q-loop improves the design model, which indicates that the integration of model reduction and controller design does improve the controller synthesis.

This is the first BOCC controller design tested in lab. The BOCC design algorithm, which is a generalization of the OVC and OCC algorithms, works well for this project. The difference in the performance between the OVC and BOCC design is attributed to the difference in the type of design specifications, rather than any preference for one method over the other. The BOCC is much more general, including the OVC as a special case.

Finally, a MATLAB software package IMC has been produced to integrate modeling and controller design for flexible structures. This is the first experimental test of this software.

6. LIST OF REFERENCES

[1] G. Zhu and R.E. Skelton, "Mixed $\mathcal{L}_2$ and $\mathcal{L}_\infty$ problems by weight selection in quadratic optimal control," *Int. J. Control, Vol. 53, No. 5, 1991.*

[2] C. Hsieh and R.E. Skelton, "Minimum energy controllers satisfying inequality output variance constraints," *Appl. Methods, Vol. 10, No. 4, 347-366, 1989.*

[3] G. Zhu and R. E. Skelton, "Controller Design to Achieve Covariance Constraints," *IFAC Symposium on Design Methods of Control System, Zurich, Switzerland, Sep., 1991.*

[4] G. Zhu, "$\mathcal{L}_2$ and $\mathcal{L}_\infty$ Multiobjective Control for Linear Systems," *PhD Dissertation, Purdue University, May, 1992.*

[5] G. Zhu and R. E. Skelton, "A Two-Riccati Feasible Algorithm for Guaranteeing Output $\mathcal{L}_\infty$ Constraints," *Proceedings of Control and Decision Conference, 1991.*

[6] R. E. Skelton, *Dynamics System Control*, Wiley, New York, 1988.

[7] R. E. Skelton, R. Singh and J. Ramakrishnan, "Component Model Reduction by Component Cost Analysis," *Proc. AIAA Guid. and Contr. Conf., Minneapolis, MN, 1988.*

[8] J. H. Kim and R. E. Skelton, "Model Reduction by Weighted Component Cost Analysis," *AIAA Dynamics Specialists Conf., Long Beach, CA, 1990.*

[9] R. E. Skelton, P. C. Hughes and H. B. Hablani, "Order Reduction for Models of Space Structure Using Modal Cost Analysis," *J. Guid., Contr. and Dyn., Vol. 5, No. 4, 1982.*

[10] C. Hsieh, J. H. Kim, G. Zhu, K. Liu and R.E. Skelton, "An iterative algorithm combining model reduction and control design," *Proceedings of American Control Conference, pp 2120-2125, 1990.*

[11] R. E. Skelton and B. D. O. Anderson, "Q-Markov Covariance Equivalent Realization," *Int. J. Contr., Vol. 44, No. 5, 1986.*

[12] D. A. Wagie and R. E. Skelton, "A Projection Approach to Covariance Equivalent Realizations of Discrete Systems," *IEEE Trans. Auto. Contr., AC-31, 1986.*

[13] R. E. Skelton and E. G. Collins, "Set of Q-Markov Covariance Equivalent Model of Discrete Systems," *Int. J. Contr., Vol. 46, No. 1, 1987.*

[14] H. C. Vivian, P. E. Blaire, D. B. Eldred, G. E. Fleischer, C.-H. C.Ih, N. M. Nerheim, R. E. Scheid and J. T. Wen, "Flexible Structure Control Laboratory Development Technology Demonstration," *Jet Propulsion Laboratory, California Institute of Technology, 1987.*

[15] C. Hsieh, J. H. Kim and R. E. Skelton, "NASA GI Project-Purdue Annual Report," *NASA GI Project Annual Report Meeting, Hampton, VA, Jan., 1990.*

Table 1 Inputs, Outputs and Their Limits

Inputs		Outputs	
Hub Actuator		Hub Sensor	
Notation	Limit	Notation	Limit
HA10 (u_1)	2 (N-M)	HS1 (y_{25})	69.8 (mrad)
HA1 (u_2)	2 (N-M)	HS10 (y_{26})	69.8 (mrad)
Rib Root Actuator		Rib Root Sensor	
Notation	Limit	Notation	Limit
RA1 (u_3)	2 (N)	RS1 (y_{27})	10 (mm)
RA4 (u_4)	2 (N)	RS4 (y_{28})	10 (mm)
RA7 (u_5)	2 (N)	RS7 (y_{29})	10 (mm)
RA10 (u_6)	2 (N)	RS10 (y_{30})	10 (mm)
		Levitator Sensor	
		Notation	Limit
		LS1–LS24 ($y_1 - y_{24}$)	114.3 (mm)

Table 2 Frequencies and Damping Coefficients

Mode No.	Frequency (Hz)		Damping Coeff.	
	(Original)	(Modified)	(Original)	(Modified)
1	0.0902	0.0975	0.0100	0.1225
2	0.0902	0.0917	0.0100	0.2500
3	0.2089	0.2089	0.0263	0.0263
4	0.2527	0.2527	0.0100	0.0100
5	0.2527	0.2527	0.0100	0.0100
6	0.2894	0.2894	0.0100	0.0100
7	0.2894	0.2894	0.0100	0.0100
8	0.3218	0.3218	0.0100	0.0100
9	0.3218	0.3218	0.0100	0.0100
10	0.3435	0.3435	0.0100	0.0100
11	0.3435	0.3435	0.0100	0.0100
12	0.3509	0.3509	0.0100	0.0100
13	0.6150	0.6250	0.0200	0.0200
14	0.6150	0.6200	0.0300	0.0300
15	1.5083	1.5083	0.0100	0.0100
16	1.5295	1.5295	0.0100	0.0100
17	1.5295	1.5295	0.0100	0.0100
18	1.5461	1.5461	0.0100	0.0100
19	1.5461	1.5461	0.0100	0.0100
20	1.5625	1.5625	0.0100	0.0100
21	1.5625	1.5625	0.0100	0.0100
22	1.5744	1.5744	0.0100	0.0100
23	1.5744	1.5744	0.0100	0.0100
24	1.5746	1.5746	0.0100	0.0100
25	1.6842	1.6842	0.0100	0.0100
26	1.6842	1.6842	0.0100	0.0100
27	2.5771	2.5771	0.0100	0.0100
28	2.5771	2.5771	0.0100	0.0100
29	4.8576	4.8576	0.0100	0.0100
30	4.8576	4.8576	0.0100	0.0100

Table 3 Modal Cost Analysis of the OVC design for Q-loop 1 and 3

Index	Q-loop 1		Q-loop 3	
	$\alpha_b = 9$		$\alpha_b = 9$	
	Modal Cost	Mode No.	Modal Cost	Mode No.
1	6.6723e+1	2	6.7071e+1	2
2	2.5040e+1	1	2.3619e+1	1
3	4.2676e+0	14	4.8554e+0	14
4	2.2173e+0	13	2.6104e+0	13
5	4.7986e-1	4	4.7182e-1	4
6	2.0138e-1	8	1.9765e-1	8
7	1.9129e-1	3	1.8981e-1	7
8	1.7050e-1	7	1.7689e-1	3
9	1.4221e-1	6	9.9888e-2	6
10	8.7302e-2	11	7.9285e-2	16
11	7.9527e-2	12	7.8692e-2	11
12	7.8317e-2	10	7.5395e-2	10
13	5.2735e-2	16	7.3416e-2	20
14	4.8844e-2	20	7.3290e-2	12
15	3.3937e-2	27	5.8980e-2	28
16	3.2116e-2	28	4.9655e-2	27
17	2.6763e-2	15	3.8368e-2	19
18	2.5116e-2	19	3.8337e-2	15
19	2.3742e-2	24	3.4985e-2	23
20	2.3351e-2	23	3.4597e-2	24

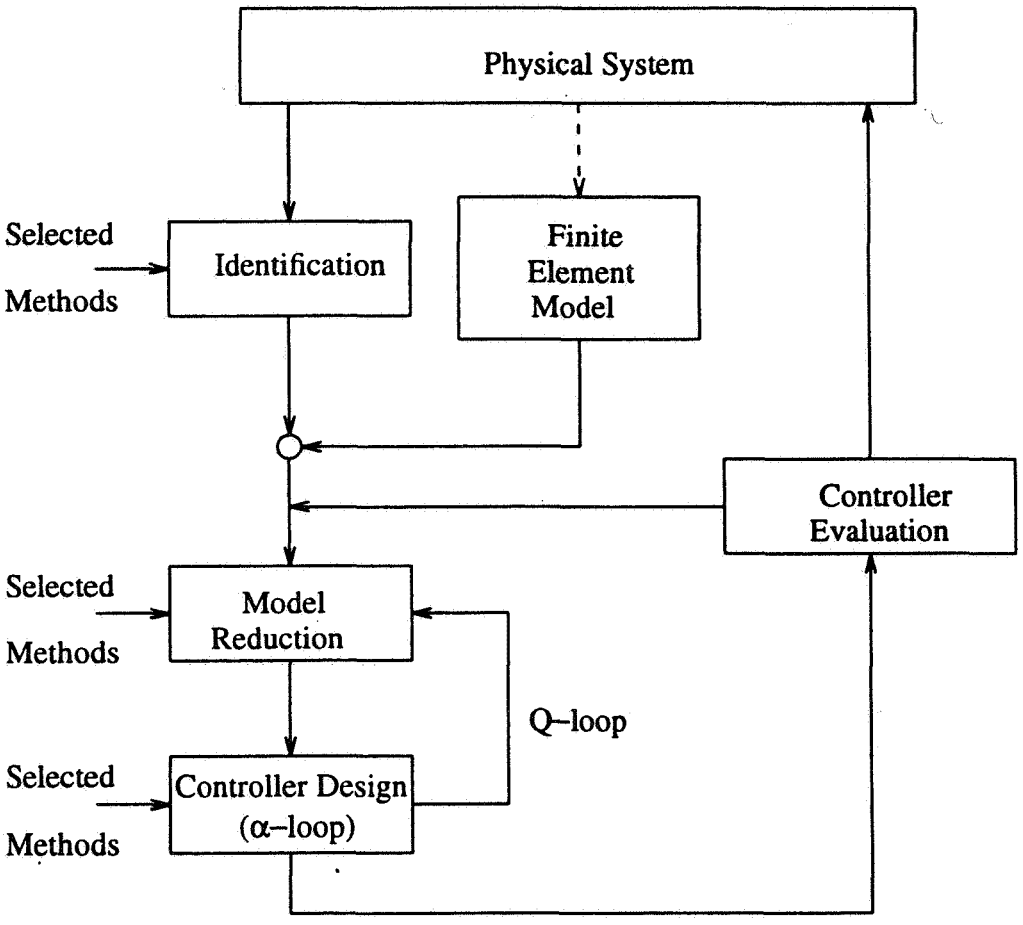


Figure 1 Flowchart of the IMC Software

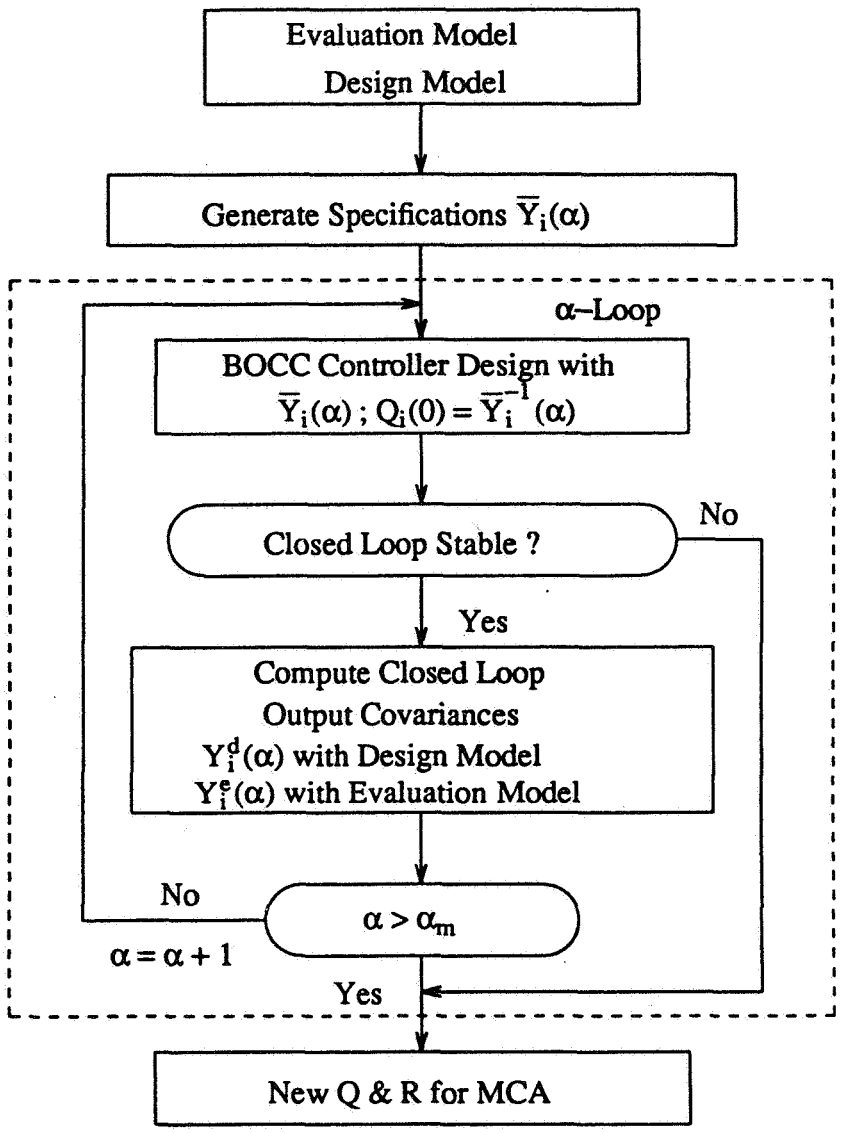


Figure 2 The BOCC α-loop Study

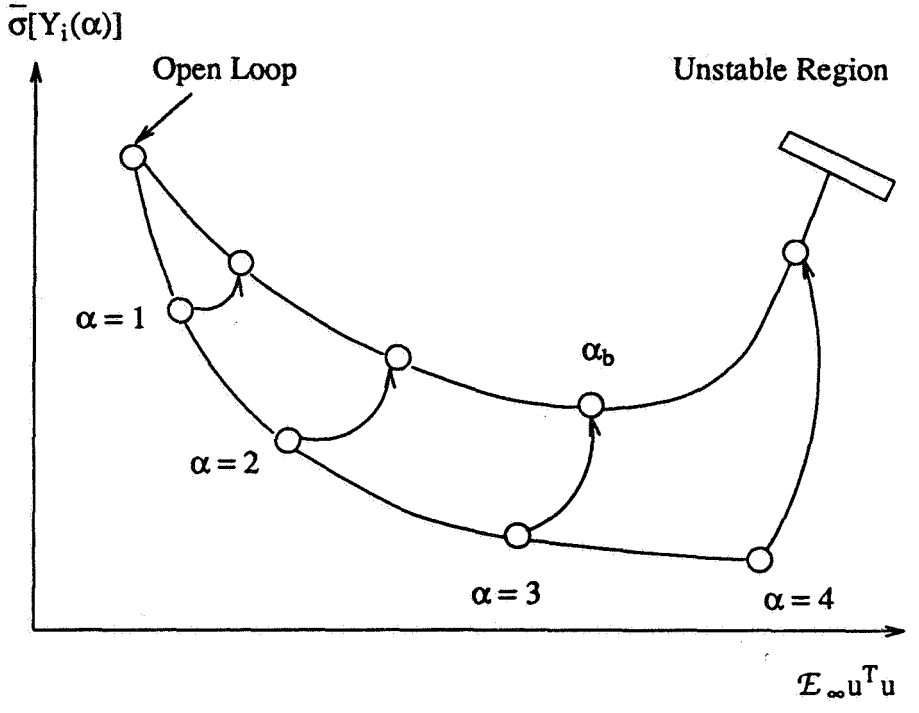


Figure 3 Tradeoff of Output Covariance and Input Variances

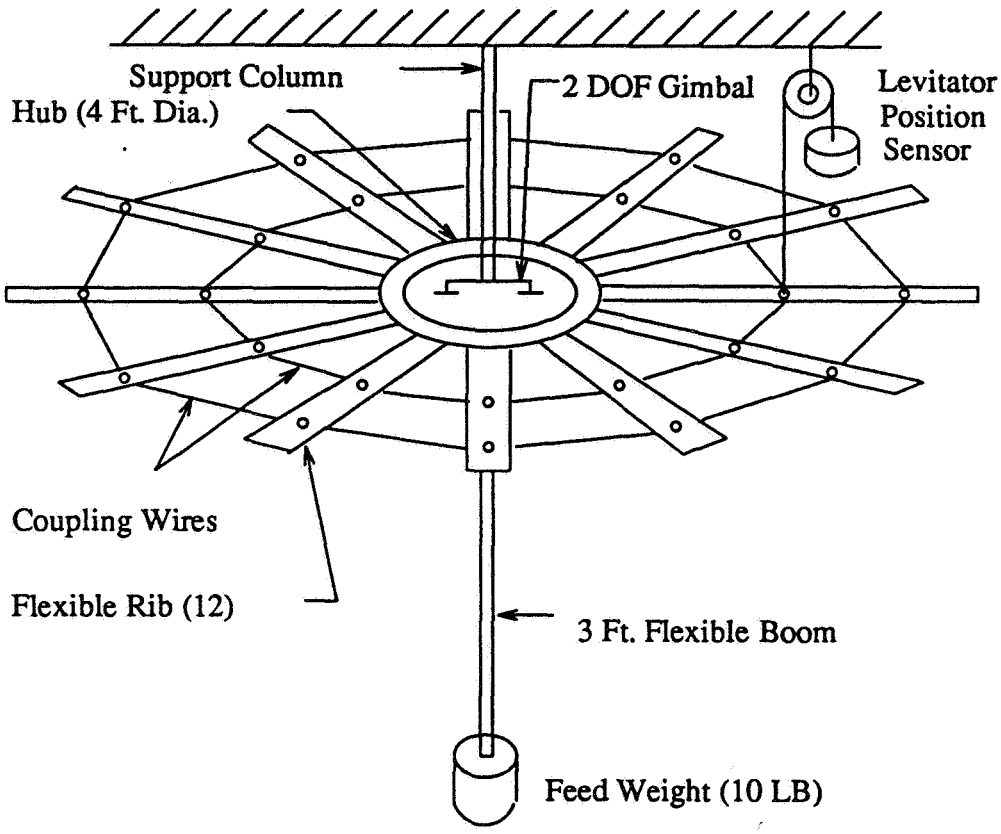


Figure 4 Experiment Structure

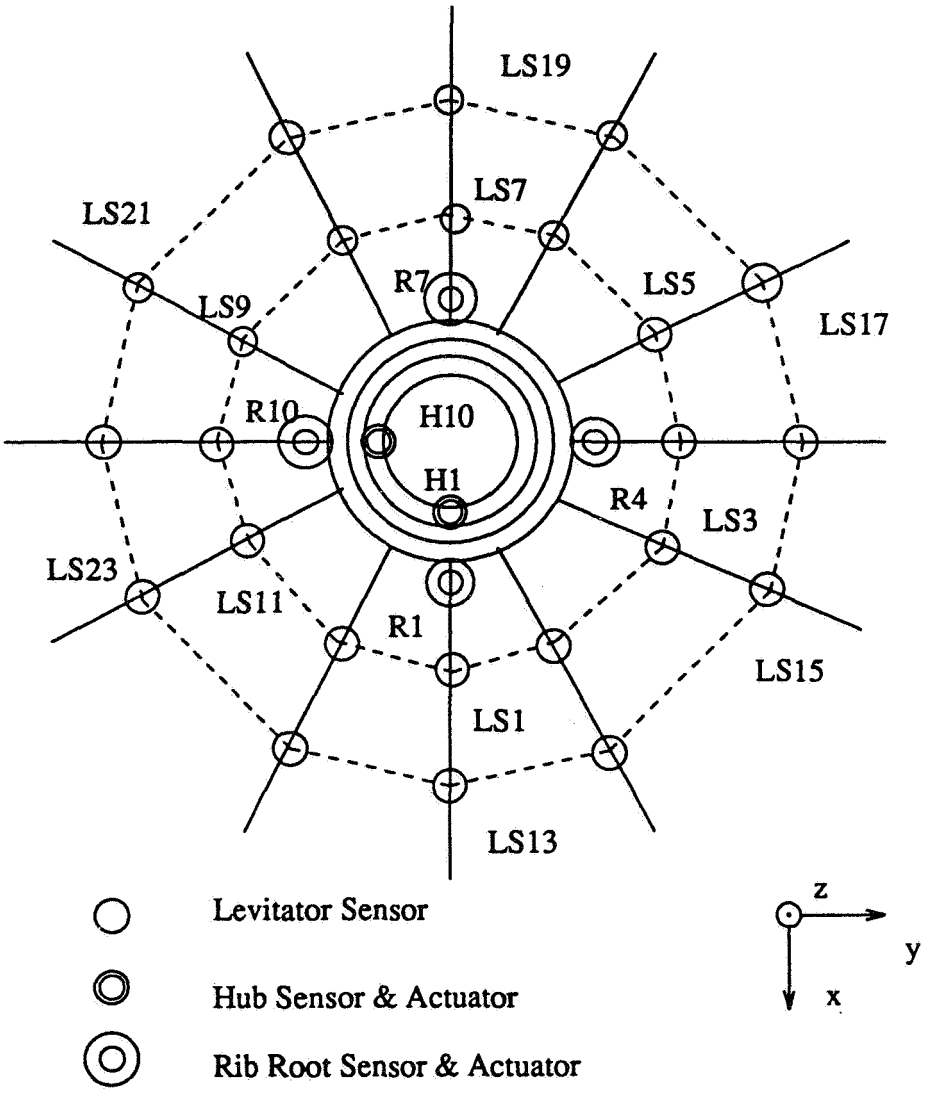


Figure 5 Transducer Locations and Labeling

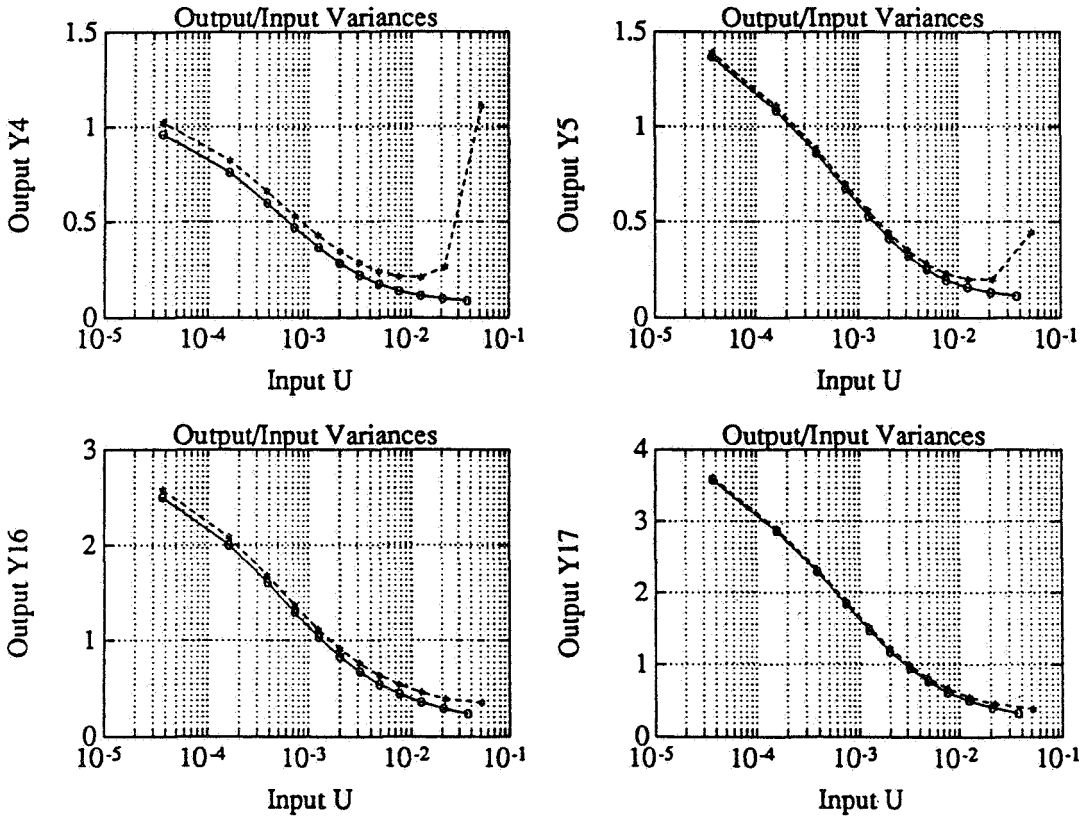


Figure 6 Input/Output Variance Curves for the OVC Design

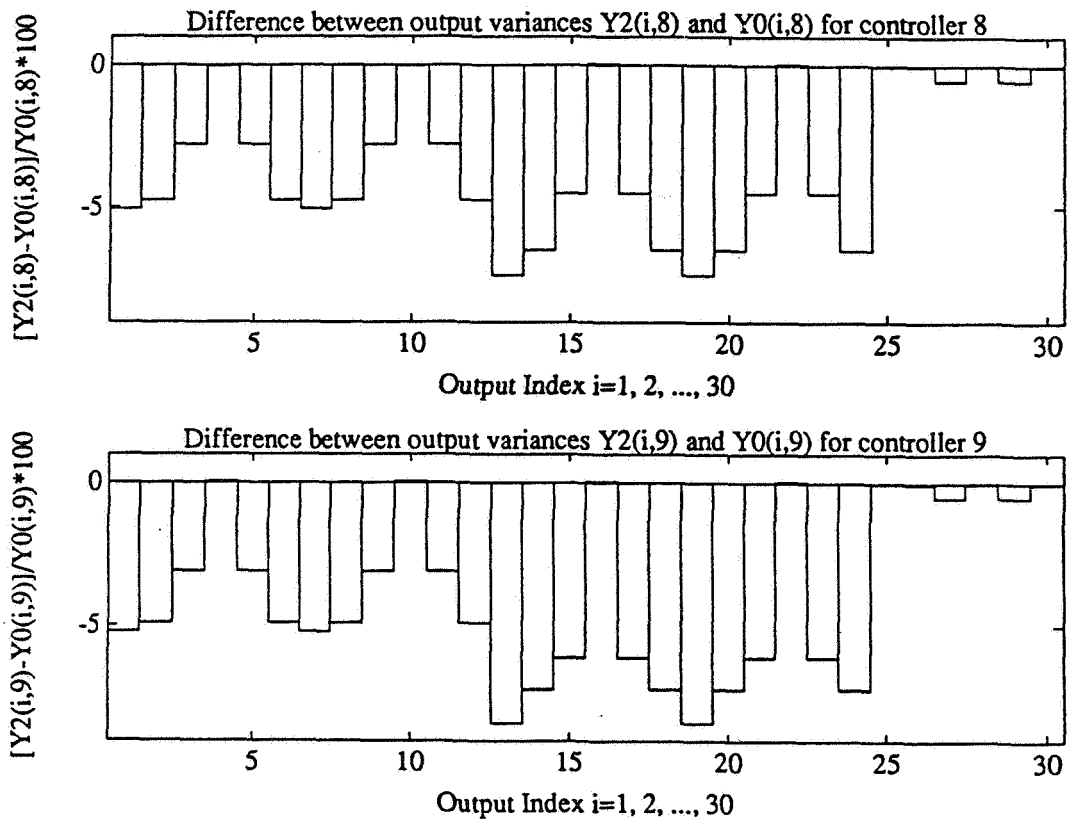


Figure 7 Output Variance Differences of Q-loop 0 and 2

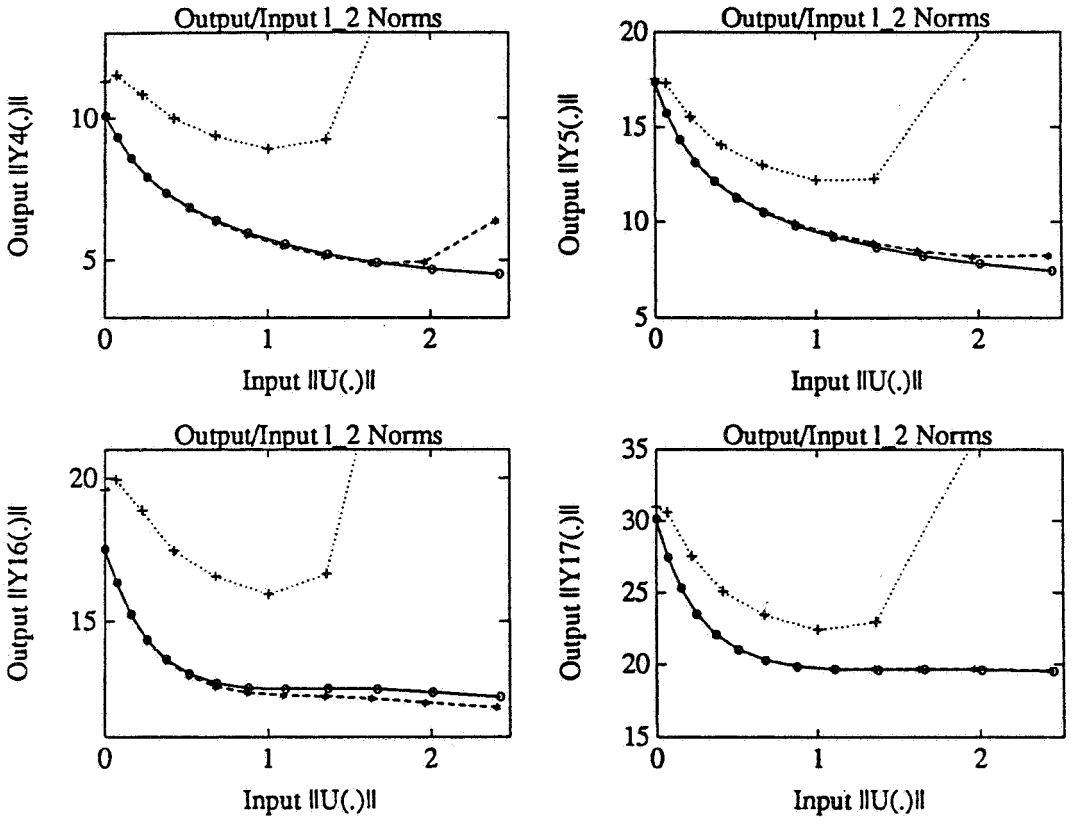


Figure 8 Input/Output ℓ_2 Norm Curves for the OVC Design

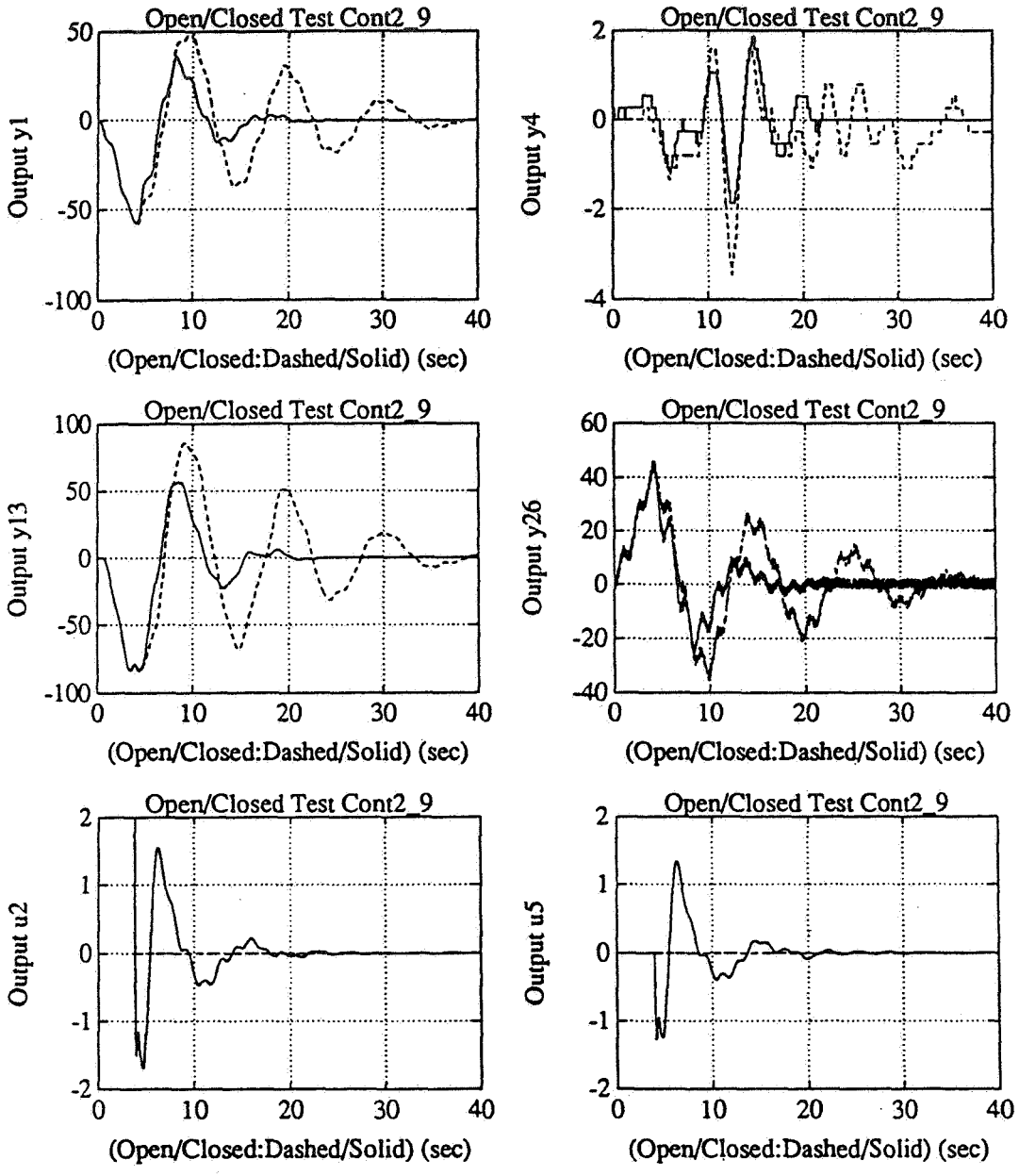


Figure 9 Y Direction Pulse Responses of the OVC Design

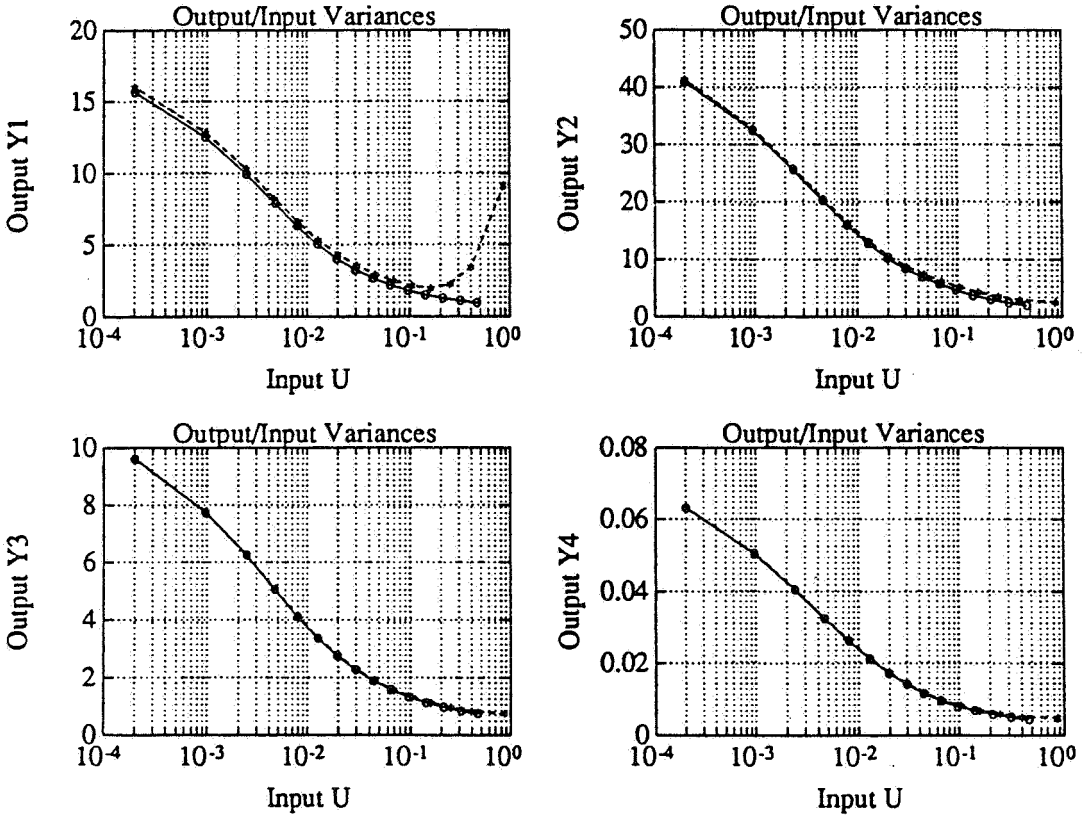


Figure 10 Input/Output Variance Curves for the BOCC Design

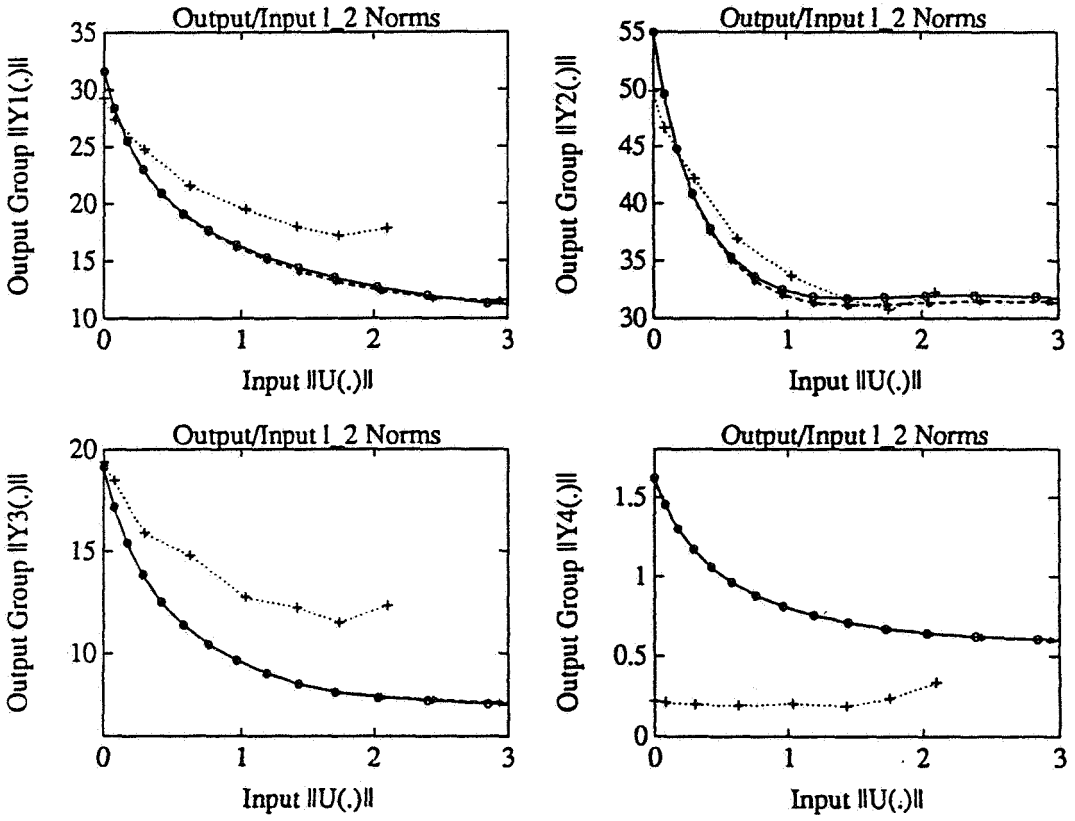


Figure 11 Input/Output ℓ_2 Norm Curves for the BOCC Design

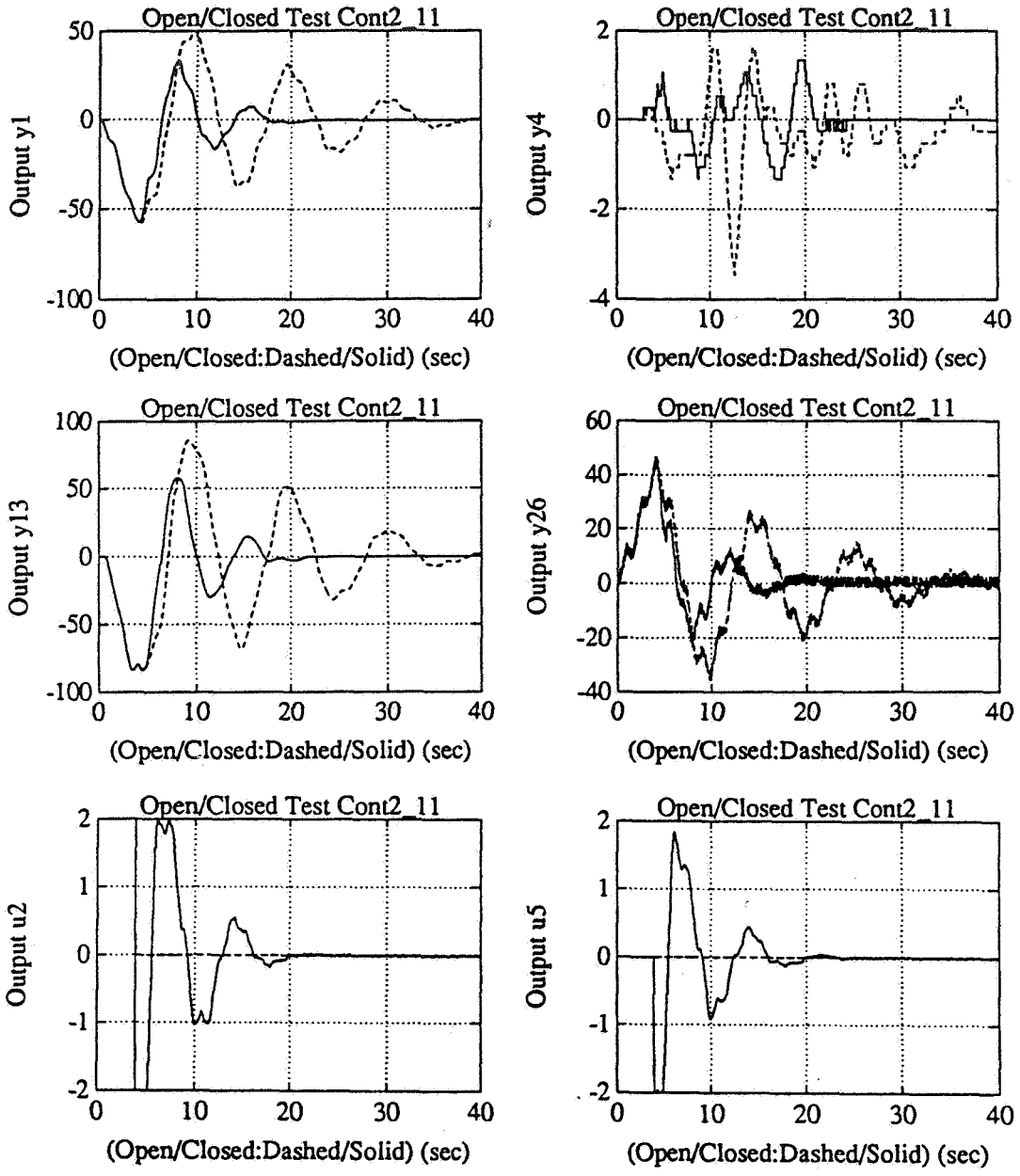


Figure 12 Y Direction Pulse Responses of the BOCC Design

Control System Design for Flexible Structures Using Data Models

R. Dennis Irwin, W. Garth Frazier
Jerrel R. Mitchell, Enrique A. Medina

Department of Electrical & Computer Engineering
Ohio University, Athens, Ohio

Angelia P. Bukley
NASA Marshall Space Flight Center, MSFC, Alabama

Abstract

The dynamics and control of flexible aerospace structures exercises many of the engineering disciplines. In recent years there has been considerable research in the developing and tailoring of control system design techniques for these structures. This problem involves designing a control system for a multi-input, multi-output (MIMO) system that satisfies various performance criteria, such as vibration suppression, disturbance and noise rejection, attitude control and slewing control. Considerable progress has been made and demonstrated in control system design techniques for these structures. The key to designing control systems for these structures that meet stringent performance requirements is an accurate model. It has become apparent that theoretically and finite-element generated models do not provide the needed accuracy; almost all successful demonstrations of control system design techniques have involved using test results for fine-tuning a model or for extracting a model using system ID techniques.

This paper describes past and ongoing efforts at Ohio University and NASA Marshall Space Flight Center (MSFC) to design controllers using "data models". The basic philosophy of this approach is to start with a stabilizing controller and frequency response data that describes the plant; then, iteratively vary the free parameters of the controller so that performance measures become closer to satisfying design specifications. The frequency response data can be either experimentally derived or analytically derived. One "design-with-data" algorithm presented in this paper is called the Compensator Improvement Program (CIP). The current CIP designs controllers for MIMO systems so that classical gain, phase, and attenuation margins are achieved. The center-piece of the CIP algorithm is the constraint improvement technique which is used to calculate a parameter change vector that guarantees an improvement in all unsatisfied, feasible performance metrics from iteration to iteration. The paper also presents a recently demonstrated CIP-type algorithm, called the Model and Data Oriented Computer-Aided Design System (MADCADS), developed for achieving H_∞ type design specifications using data models. Control system designs for the NASA/MSFC Single Structure Control Facility are demonstrated for both CIP and MADCADS. Advantages of design-with-data algorithms over techniques that require analytical plant models are also presented.

Control System Design for Flexible Structures Using Data Models

R. Dennis Irwin, W. Garth Frazier
Jerrel R. Mitchell, Enrique A. Medina

Department of Electrical & Computer Engineering
Ohio University, Athens, Ohio

Angelia P. Bukley
NASA Marshall Space Flight Center, MSFC, Alabama

Introduction

The performance objectives in the design of controllers for flexible structures (FS's) include vibration suppression, disturbance rejection, and attitude control. FS's are characterized by having many low frequency, closely spaced, lightly damped structural modes. For controller designs to meet specifications, several structural modes must lie within the control system bandwidth. Because the structural modes of FS's are inherently lightly damped, they can cause vibrations problems once excited, and they provide paths of propagation between disturbances and quantities being controlled or regulated. The controller design process must either dampen or suppress (notch) these modes.

Because the modes for many FS's are closely spaced in frequency, the design process, e.g., LQG, H_∞, loop-at-the time, μ-synthesis, etc., used to dampen and/or suppress these modes, typically produces controllers with lightly damped characteristics in the frequency range of those modes inside the control system bandwidth. This produces significant problems of robustness to model inaccuracy. Experience has shown that models developed either from physical laws or finite element methods (FEM's) do not provide sufficient accuracy for controller designs for FS's with stringent vibration/disturbance/attitude performance specifications. It is not anticipated that significant breakthroughs will occur in control system model development from either physical laws or FEM's in the next decade.

An alternative is to develop control system design models from test results. The normal approach is to fabricate the FS, perform testing, and extract an analytical control system design model from the test data. The last step is called system identification (ID), and the results can either be a time domain or frequency domain model. This process is not trivial and is greatly complicated by FS's being inherently multi-input, multi-output (MIMO) in nature. In fact, system ID for FS's is still more of an art than a science and is time consuming and numerically intensive. Furthermore, for the MIMO case the order of the resulting model of the system can easily exceed one hundred. Numerical techniques used to design controllers cannot normally handle orders of this magnitude. To circumvent the order problem, the model is reduced, using model reduction schemes. The model reduction schemes "throw things away", and some produce models with different modes and mode shapes than were produced by system ID. The consequence is a design model that is significantly different from that identified. A controller

design based on the reduced model may or may not produce a closed loop system satisfying design specifications. If the design does not meet specifications, the designer must either find a better model or fine-tune the design.

Alternate approaches are obviously needed. Approaches that directly utilize data models, i.e., test data or frequency response data obtained by operating upon test data by an FFT, to design controllers or fine-tune reduced order controllers, can avoid or circumvent the pitfalls of the system ID, model reduction, controller design process.

The philosophy of designing controllers using data models is not new. One of the most successful ventures in the development of an automated approach to the design of controllers for complex aerospace vehicles using frequency response data models is the Compensator Improvement Program (CIP) developed for NASA/MSFC in the 1970's for aiding in the design of controllers for the ascent flight control systems of the Saturn V and the Space Shuttle [1,2].

In this paper the description of the control system design problem as an abstract mathematical programming problem is presented. This is followed by a brief description of a straightforward algorithm used to find the solution to this problem using data models. Next, the basic features of two software programs, the CIP and the Model and Data Oriented Computer-Aided Design System (MADCADS) that implement variations of this algorithm are described. The application of these programs to the design of controllers for a flexible structure are then presented. Finally, plans for future enhancements to CIP and MADCADS are described, followed by concluding remarks.

An Algorithm for Design with Data Models

This section illustrates how the control systems design problem can be cast as a mathematical programming problem and presents a viable, iterative algorithm for its solution. Two software programs using this iterative algorithm are then described.

Problem Statement and Solution

The problem of designing a controller to meet various specifications can be stated abstractly as a mathematical programming problem of the form: Find $x \in \mathbf{R}^n$ to satisfy

$$f_i(x) \geq 0, \quad i = 1,2,\ldots,N_f \tag{1}$$

where each f_i is a function corresponding to a design specification and x is a vector of design variables that correspond to the free parameters of a controller representation. An approach to solving this problem using data models that has been proven effective is the Constraint

Improvement Technique (CIT). The CIT is based upon the fundamental principles of optimization in finite dimensional spaces under the assumption that the constraint functions are differentiable functions of the controller's parameters. The CIT has the following algorithmic structure. Let $x^{(k)}$ denote the value of the parameter vector at the k^{th} iteration. Set $k = 1$.

Step 1: [Test for convergence.] If all the constraints are satisfied, stop. Set the solution equal to $x^{(k)}$.

Step 2: [Calculate a search direction.] Compute a nonzero $d^{(k)} \in \mathbf{R}^n$ that has the possibility of improving some function of the constraints.

Step 3: [Calculate a step length.] Compute a nonnegative $\alpha^{(k)}$ such that when the constraint functions are evaluated at $x^{(k)} + \alpha^{(k)}d^{(k)}$ a measure of algorithm progress is improved.

Step 4: [Update parameters.] Set $x^{(k+1)} = x^{(k)} + \alpha^{(k)}d^{(k)}$ and $k \leftarrow k + 1$. Go to Step 1.

The key step in determining performance for this algorithm structure is the calculation of the search direction. The search direction as determined by CIT is calculated by finding $d^{(k)}$ such that

$$\nabla f_i^T(x^{(k)})d^{(k)} = c_i^{(k)}, \quad \forall\, i \in T^{(k)} \tag{2}$$

where ∇f_i is the gradient of the i^{th} constraint function, $c_i^{(k)}$ is a positive scalar and $T^{(k)}$ is a set denoting the constraints violated at the k^{th} iteration. If the number of elements in $T^{(k)}$ is less than or equal to the dimension of the parameter space and all the $\nabla f_i(x^{(k)})$ are linearly independent then there exists a $d^{(k)}$ to satisfy Equation 2. The requirement that each $c_i^{(k)} > 0$ insures that all the violated constraints can be "theoretically" improved at each iteration. Equation 2 is actually an underdetermined system of linear equations that can be written in the form

$$J^{(k)}d^{(k)} = c^{(k)} \tag{3}$$

where $J^{(k)}$ is a matrix with rows formed from the gradients and $c^{(k)}$ is a vector formed from the $c_i^{(k)}$'s. The angle between the search direction and the i^{th} gradient is given by

$$\theta_i^{(k)} = \text{acos} \frac{c_i^{(k)}}{\| \nabla f_i(x^{(k)}) \|_2 \| d^{(k)} \|_2} \cdot \tag{4}$$

Minimizing the 2-norm solution for $d^{(k)}$ in Equation 3 keeps this angle as small as possible for each i. Experience has shown that choosing

$$c_i^{(k)} = \| \nabla f_i(x^{(k)}) \|_2$$

(which causes $\theta_i^{(k)} = \theta_j^{(k)} \quad \forall\, i,j \in T^{(k)}$) provides good algorithm performance.

CIP Overview

The first software program to employ CIT has been CIP; a viable candidate for improving or augmenting control system designs for FS's. It can be used to recover lost performance caused by spillover in state space or transfer function designs or to fine-tune loop-at-the-time designs. The essence of CIP is to start with an initial stabilizing design and iteratively increment the design parameters so as to improve open loop performance measures. The initial version of CIP was developed to improve designs of controllers for single input, multiple output systems [1]. Later CIP was extended to handle true MIMO systems [2].

CIP views the connection of the controller/plant as a multiple loop system. The general block diagram for which CIP has been tailored is shown in Figure 1. The design philosophy implemented in CIP is to iteratively increment the parameters of the controller so that *simultaneous* improvement of the open loop frequency responses of each loop occurs with all other loops closed. For complex systems, such as FS's, this task is pragmatically impossible by manual design techniques. In regard to Figure 1 the loops are broken between the controller and the plant.

The prominent features and characteristics of CIP are described as follows:

(1) The plant or system is assumed to be described in the form of a transfer function matrix. CIP requires frequency response data of each element of this matrix for a system description. By using frequency response data as a system model, numerical problems in handling large order systems are eliminated, and experimentally determined frequency responses can be directly accommodated.

(2) Performance specifications can be made frequency dependent. This accommodates different specifications for phase and gain stabilization regions.

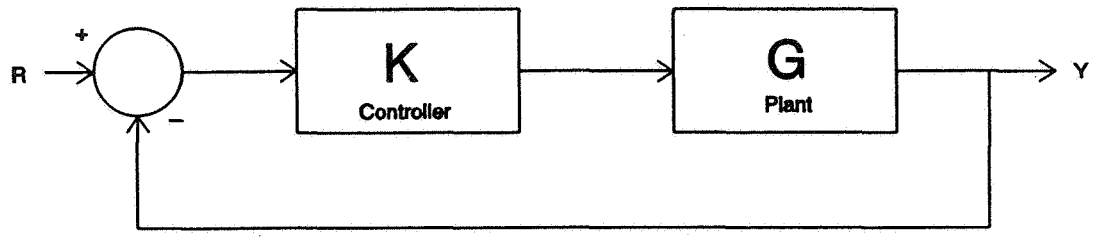


Figure 1: General Block Diagram for which CIP is Designed

(3) The controller is described as a transfer function matrix in which each element is represented as a ratio of first and second order factors. For continuous controllers these are s-plane functions, whereas for digital controllers these are w-plane functions. The coefficients of these factors are varied by CIP to improve the system performance. By constraining the variations in certain coefficients, restrictions can be placed on a controller element. For example, the D.C. gain of an element can be held constant to assure steady-state error performance, the coefficients of first order factors can be constrained to be positive in order to avoid first order right-half plane poles and zeros, or the damping ratios of second order factors can be specified to be above minimum values in order to assure robustness of the controller.

(4) CIP can test for system stability on each iteration.

(5) The coefficient change vector computed by CIP assures from iteration to iteration that an improved design results.

The code is started by specifying the system with frequency response data for each element of the transfer function matrix, the initial compensation, the desired design specifications, etc. At each iteration the performance measurements of the system are evaluated by opening each feedback loop, with all other loops closed, and determining stability and attenuation margins. The performance measurements are compared with respect to the design specifications; if all specifications are satisfied, the design is complete, and the process is terminated. Otherwise, the controller coefficient change vector (search direction) is computed using the gradients of the unsatisfied performance measurements with respect to the free parameters of the controller. A step is then taken along the search direction to compute a new compensator such that an improved solution is assured. If a step cannot be found that improves the performance measures above some user specified minimum, convergence is assumed. Otherwise, the iterative process is repeated.

As discussed previously, a property of the search direction computed by CIT is that it has a positive inner product with all gradient vectors; as a consequence, it is theoretically

possible to simultaneously improve all unsatisfied performance measurements. However, from a practical point of view small degradations in some performance measurements are greatly outweighed by large improvements in one or more of the others. Such a philosophy has been incorporated into CIP.

MADCADS Overview

Recently, MADCADS, a code similar to CIP and based upon CIT, has been developed to use frequency domain data models to design controllers to meet H_∞-type multivariable control system design constraints [3]. An example of a typical constraint is the shape of the frequency response of the maximum singular value of the sensitivity function which can be written as

$$\sigma_{max}\left[(I + G(\omega)K(\omega))^{-1}\right] \leq c(\omega) \quad \forall\, \omega \in [\omega_0, \omega_f] \tag{5}$$

where c is a function defined so as to achieve desired closed loop specifications.

In contrast to CIP, MADCADS uses a state-space realization to parameterize the controller in order to provide more flexibility in the controller's structure. Since the number of parameters in an arbitrary state-space realization is rather large (more than n^2 for an n^{th} order controller), it is necessary for computer memory limitations and algorithm performance to limit the number of parameters that are free to change at each iteration. In the current MADCADS the number of free parameters is limited by restricting the "A" matrix of the realization to be in upper-Hessenberg form. This does not pose any serious limitations on the structure of the controller. MADCADS also differs from CIP in that controllers for sampled-data systems are designed directly in the z-plane rather than in the w-plane. The current version of MADCADS does not assume any particular block diagram; rather the user must code subroutine modules to calculate constraints and gradients as they are needed. A large library of these modules has been developed for frequency dependent singular value constraints for various control system configurations.

Applications to the SSC Facility

This section describes the application of CIP and MADCADS to a flexible aerospace structure ground test facility. The details of the controller design procedures and experimental results of the implementations are also presented.

Description of the SSC Facility

A schematic of the NASA Marshall Space Flight Center Single Structure Control (SSC) Facility is shown in Figure 2. The SSC Facility is suitable for the study of line-of-sight (LOS)

and vibration suppression control issues as pertaining to flexible aerospace structures. The primary element of the SSC Facility, a spare Voyager magnetometer boom, is a lightly damped beam measuring approximately 45 feet in length and weighing about 5 pounds.

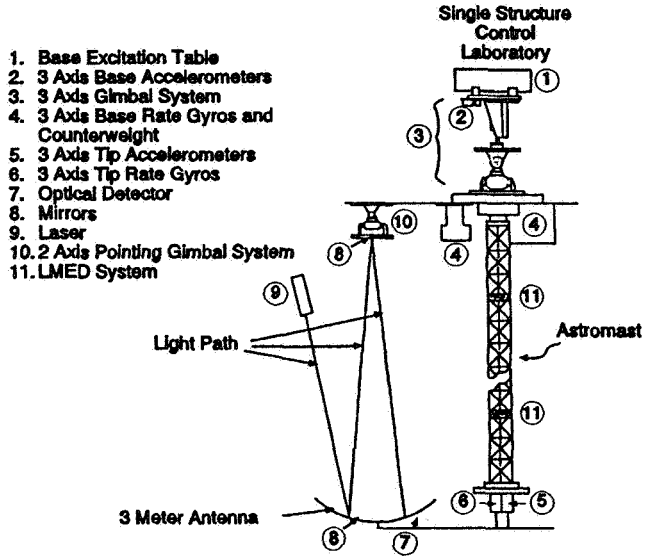


Figure 2: Schematic of the ACES Structure

The goal of the control system design is to maintain the reflected laser beam in the center of the antenna (location of the detector) in the presence of disturbances introduced by the base excitation table (BET). The digital controller is to be implemented on the HP9000 computer located at the facility using the fixed sampling rate of 50 Hertz and a fixed, one sample period computational delay. The results of other controller designs for the SSC Facility have been reported in the literature [4].

The experimental open loop frequency response from the y-axis of the Image Motion Compensation (IMC) gimbals to the x-axis LOS error is shown in Figure 3. The effect of the computational delay is quite apparent from analysis of the phase characteristic. The frequency responses of the other axes of the IMC-to-LOS are similar, although the cross-axis terms have less gain. The open loop frequency response from the y-axis Advanced Gimbal System (AGS) gimbal to the y-axis base gyro is shown in Figure 4. This response reveals the numerous lightly damped modes of the structure. The frequency responses of other elements of the AGS-to-base gyros transfer function matrix are similar. Mathematical modeling of the structure does not provide a model with sufficient fidelity to accomplish the above stated design goal. The frequency responses shown in Figures 3 and 4 were obtained by first exciting the system with pseudo-random inputs applied by the IMC and AGS, respectively, collecting the time response

data and then using FFT techniques to compute the frequency response data. Averaging techniques were

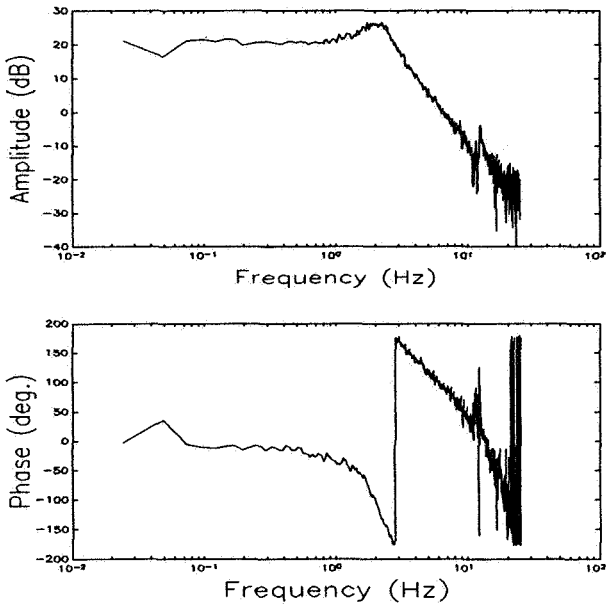


Figure 3: Experimental Frequency Response from y-Axis IMC to x-Axis LOS Error

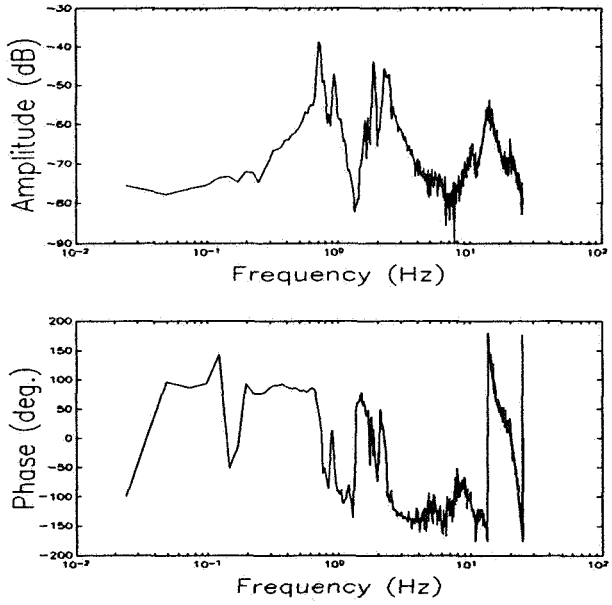


Figure 4: Experimental Frequency Response from y-Axis AGS to y-Axis Base Gyro

used to minimize the effects of noise and environmental disturbances. The error in the AGS magnitude data is estimated to be 10 dB or less for frequencies above 0.5 Hz.

There is a pendulum mode in the axis represented by Figure 4 with a frequency of roughly 0.15 Hz. As can be seen, the data does not show a lightly damped mode at this frequency. This is a resolution problem caused by a limitation in the data acquisition system hardware preventing the storage of a sufficient number of data points for accurately computing the frequency response characteristics of this mode. In fact, from other studies it is known that the frequency response data should show a peak of roughly 40 dB above the value shown.

Application of CIP

Since it is felt by the authors that an important factor in achieving a successful design for the system is the attainment of increased damping of the modes of the structure, the design using CIP has been limited to this goal. From Figure 4 it is seen that all the modes up to 5 Hz are reasonably phase stabilized. The first 180 degree crossing is roughly at 8 Hz. The design strategy is to gain stabilize all modes above 3 Hz, leave the modes below 3 Hz phase stabilized, and close the loop with a D.C. compensator gain of at least 60 dB. It is expected that significant damping will be added to all modes less than 3 Hz. The major difficulty in carrying out the design strategy is to roll-off (gain stabilize) the modes above 3 Hz while not adversely affecting the phase of the modes below 3 Hz, viz., minimizing phase lag spillover.

As a start, a low pass elliptic filter is designed with a break frequency of 6 Hz. The elliptic filter is chosen in order to minimize the phase lag spillover in the frequency range less than 3 Hz and to provide satisfactory attenuation of modes above 6 Hz. The stability and attenuation margins with this compensation are shown in Table 1.

CIP was run for 18 iterations and converged without satisfying all the design constraints, although significant improvements had been made. A third order factor with unity frequency response with poles near the frequencies where the constraints were not satisfied was placed in cascade with the resulting compensator. CIP then ran for 24 more iterations and satisfied the design constraints. The resulting margins are given in Table 2.

The resulting compensator was transformed to the time domain and down-loaded to the control computer of the SSC Facility. As expected significant improvements in performance were observed. In fact, test results showed that the damping of the pendulum mode more than tripled. It was assessed that more damping could be obtained if the D.C. gain could be raised; it was also realized that the loop gain could be doubled without jeopardizing the stability of the loop. The gain was raised conservatively to 1500, and a two cycle sine pulse at 0.15 Hz was applied. The results are shown in Figure 5. If this is closely compared to the open loop response with the same excitation, shown in Figure 6, several observations can be made. The pendulum mode damping has been increased by a factor of three, as estimated from comparison of the logarithmic decrements. The modes with frequencies of roughly 0.58, 0.76 and 1.9 Hz

are not detectable. A closed loop mode with a frequency of roughly 2.4 Hz is observed. This results from a mode not being sufficiently phase stabilized with the increase in gain. This could not be predicted due to the fact that there is roughly 10 dB and 25 degrees uncertainty in the frequency response at this frequency. Further design work would involve performing more extensive identification studies at the appropriate frequency, possibly utilizing the closed loop response.

Minimum Gain Margin		
Frequency (Hz)	Margin Value	Specification
5.38	0.1 dB	8 dB
Minimum Phase Margin		
2.76	15.2°	45.0°
Maximum Attenuation Margin		
13.05	0.0317	0.5

Table 1: Design Constraints at Iteration 1

Minimum Gain Margin		
Frequency (Hz)	Margin Value	Specification
2.82	8.6 dB	8 dB
Minimum Phase Margin		
0.91	47.2°	45.0°
Maximum Attenuation Margin		
7.48	0.2529	0.5

Table 2: Design Constraints at Iteration 42

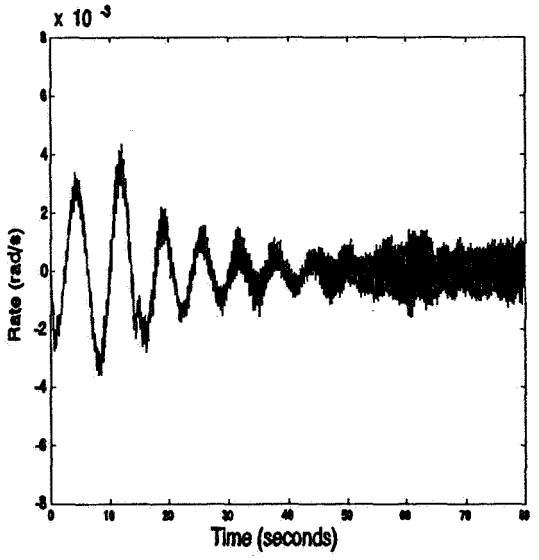


Figure 5: Closed Loop Response to a Two-Cycle Sine Pulse Input

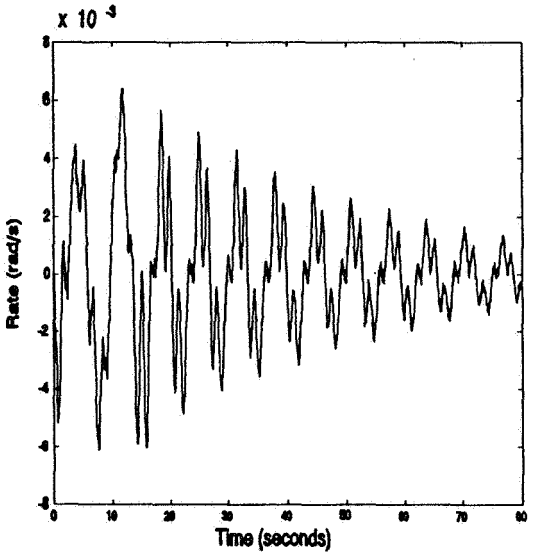


Figure 6: Open Loop Response to a Two-Cycle Sine Pulse Input

Application of MADCADS

In the application of MADCADS, a complete multivariable design is performed. The basic design philosophy is to dampen the pendulum modes and the bending modes of the structure by using feedback from the base gyros to the AGS while using the IMC gimbals with feedback from the detector to maintain the laser beam at the center of the detector. Due to sufficient decoupling, each two-input, two-output subsystem (AGS and IMC) is designed separately. One concern is the impact of disturbances that reach the IMC gimbals through the connecting arm that is attached to the base (as opposed to disturbances impacting the detector). Due to the inherently high optical gain from the IMC to the detector these disturbances can have a significant impact on the LOS error. To compensate for the effects of these disturbances it is not only necessary to maintain high loop gain over the frequency band of interest but to also maintain high IMC controller gain as well. Analysis of Figure 3 reveals that achieving high controller gain while also maintaining acceptable stability margins is difficult because of the combination of the high optical gain and the additional phase lag introduced by the computational delay. Fortunately, the impact of these disturbances can also be reduced by increasing the damping of the modes of the structure using the AGS; thereby reducing the motion of the base and the arm supporting the IMC gimbals. It is also desired to maintain reasonable levels of stability robustness.

The first step of the design procedure is the determination of a set of closed loop constraints consistent with the design philosophy such as those given in the first column of Table 3. Next, initial controllers are designed for the IMC-to-LOS and AGS-to-base gyro subsystems using a classical one-loop-at-a-time technique with the experimental frequency response data. Although the attempt was made to satisfy the constraints when designing the initial controllers, they are not satisfied as can be observed by comparing the first and second columns in Table 3. The initial controller for each subsystem is 10^{th} order. It should be noted that recently developed high fidelity models are 60^{th} order for the AGS-to-base gyro loops alone [5]. Design techniques such as LQG and H_∞ would yield controllers of at least this order (excluding weighting).

The multivariable design (i.e., taking cross-axis coupling within each subsystem into account) is then performed for each subsystem using MADCADS. The code is started with the initial 10^{th} order controllers described above, with no restrictions other than stability placed on the structure of the controllers. After approximately 100 iterations on each subsystem MADCADS converges without satisfying all the constraints. The final values of all the constraint functions are provided in the third column of Table 3.

Rather than trying to improve the design further, it was decided to implement the resulting 20^{th} order controller. The open loop x-axis LOS error due to an x-axis BET pulse disturbance intended to simulate the effect of spacecraft crew motion is shown in Figure 7. The dominant behavior in the response is the lightly damped 0.15 Hz pendulum mode. As shown in Figure 8, closing the loop with the resulting controller considerably reduces the impact of the

pendulum mode and the first bending mode. The y-axis LOS error was negligible. The crew motion disturbance was applied to the y-axis of the BET yielding similar results.

Constraint	Initial Value	Final Value
$\sigma_{min}[I+GK(z)]_{IMC} \geq 0.5, f\in(0,25)$	0.2289	0.5090
$\sigma_{min}[I+KG(z)]_{IMC} \geq 0.5, f\in(0,25)$	0.2276	0.5056
$\sigma_{min}[I+(GK(z))^{-1}]_{IMC} \geq 0.6, f\in(0,25)$	0.2827	0.6072
$\sigma_{min}[I+(KG(z))^{-1}]_{IMC} \geq 0.6, f\in(0,25)$	0.2805	0.6112
$\sigma_{min}[I+GK(z)]_{IMC} \geq 18, f\in[0.14,0.16]$	10.0020	14.1000
$\sigma_{min}[I+GK(z)]_{AGS} \geq 0.6, f\in(0,25)$	0.3649	0.5996
$\sigma_{min}[I+KG(z)]_{AGS} \geq 0.6, f\in(0,25)$	0.3585	0.5988
$\sigma_{min}[I+(GK(z))^{-1}]_{AGS} \geq 0.7, f\in(0,25)$	0.3600	0.6719
$\sigma_{min}[I+(KG(z))^{-1}]_{AGS} \geq 0.7, f\in(0,25)$	0.3589	0.6712
IMC represents IMC subsystem; AGS represents AGS subsystem G represents plant; K represents controller $z = e^{j2\pi fT}$, T = 0.02 sec		

Table 3: Summary of MADCADS Design Constraints and Results

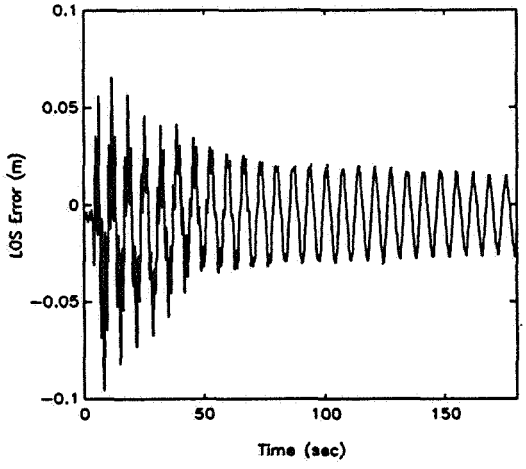


Figure 7: Experimental Open Loop x-axis LOS Error due to Crew Motion Disturbance

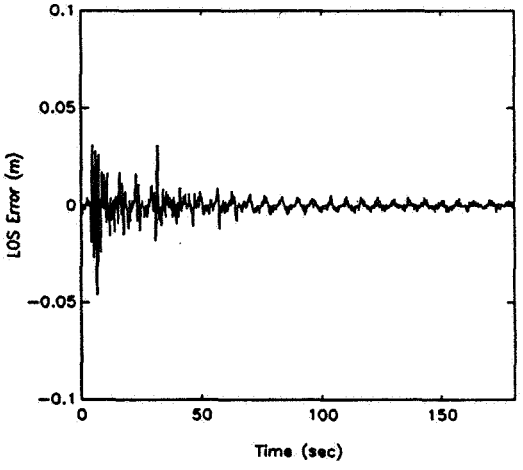


Figure 8: Experimental Closed Loop x-axis LOS Error due to Crew Motion Disturbance

Future Enhancements Planned for CIP and MADCADS

Both CIP and MADCADS have been demonstrated to be viable control system candidates for large space structures. There are ongoing programs at Ohio University and NASA/MSFC to refine and enhance these codes. This section briefly discusses these refinements and enhancements. These modifications fall into two basic categories: algorithm improvements and user interface improvements.

Algorithm Improvements

In order to increase the utility of CIP and MADCADS it is planned to increase the variety of constraints that can be handled. Current plans for CIP include the incorporation of constraints on the shapes of closed loop frequency responses and the use of state-space realizations to parameterize the controller. The ability to perform designs directly in the z-plane is currently being incorporated into CIP. Current plans for MADCADS include incorporation of constraints on the shapes of frequency responses of individual I/O pairs, operator 2-norm constraints (H_2-type constraints), as well as constraints on the damping ratios of the controller's poles and the zeros of individual I/O pairs of the controller. Other improvements for both CIP and MADCADS include better methods for calculating search directions and the application to the more general block diagram shown in Figure 9.

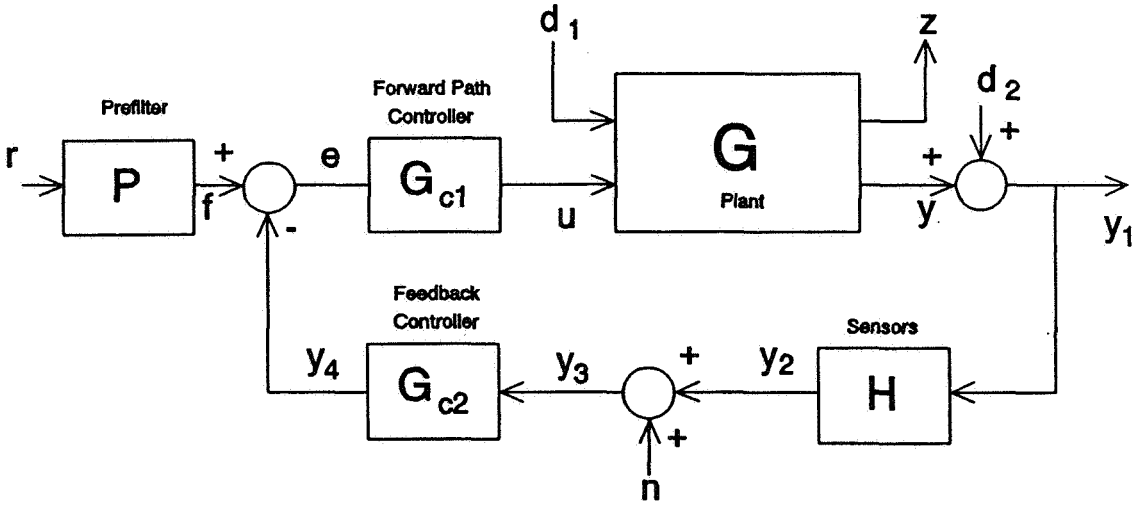


Figure 9: Proposed Block Diagram for CIP and MADCADS

User Interface Improvements

Current versions of CIP and MADCADS run in batch environments; hence user interaction is very limited and tedious. Experience has shown that the ability for the user to monitor algorithm performance and make design tradeoffs during code execution is needed. Therefore, work is in progress to develop versions of CIP and MADCADS that operate in a professional graphics workstation environment. It is planned to include in the codes the ability to monitor the progress of the design constraints in real-time by automatically updating graphics windows containing plots of the constraint functions. Other planned features include the ability to specify constraints graphically through the use of a mouse (this is especially convenient for specifying frequency domain constraints), single-step execution, and the ability to change the design constraints between iterations.

Conclusions

A review of ongoing research efforts in the area of multivariable controller design using data models at Ohio University and NASA Marshall Space Flight Center has been presented. The results of the application of two software programs, the Compensator Improvement Program and the Model and Data Computer Aided Design System, to the design of controllers for a flexible aerospace structure ground test facility was also presented. Both applications provided promising results. Future plans for expanded versions of CIP and MADCADS were also discussed.

References

1. J. R. Mitchell, "An Innovative Approach to Compensator Design," NASA Contractor Report, CR-2248, May 1973.

2. J. R. Mitchell, W. L. McDaniel, Jr., and L. L. Gresham, "Compensator Improvement for Multivariable Control Systems," Final Report, Contract No. NAS8-31568, NASA/MSFC, August 1977.

3. W. G. Frazier and R. D. Irwin, "A Numerical Approach to Controller Design with an Application to a Space Structure Test Facility," *Proceedings of the American Control Conference*, June 1992.

4. E. G. Collins, D. J. Phillips and D. C. Hyland, "Robust Decentralized Control Laws for the ACES Structure," *IEEE Control Systems Magazine*, vol. 11, no. 3, pp. 62-70, April 1991.

5. E. A. Medina, *Multi-input, Multi-output System Identification from Frequency Response Samples with Applications to the Modeling of Large Space Structures*, M. S. Thesis, Ohio University, November 1991.

Computational Issues in Optimal Tuning and Placement of Passive Dampers

C.C. Chu M.H. Milman
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

Abstract

The effectiveness of viscous elements in introducing damping in a structure is a function of several variables, including their number, their location in the structure, and their physical properties. In this paper, the optimal damper placement and tuning problem is posed to optimize these variables. Both discrete and continuous optimization problems are formulated and solved, corresponding, respectively, to the problems of placement of passive elements and to the tuning of their parameters. The paper particularly emphasizes the critical computational issues resulting from the optimization formulations. Numerical results involving a lightly damped testbed structure are presented.

1. Introduction

A problem of considerable importance in the development of technology for future space structures is the analysis and optimization of passive elements placed in these structures. Passive damping introduced by these devices is an effective mechanism for reducing peak responses in the vicinity of resonant frequencies for lightly damped systems. This not only enhances the stability of the open-loop system, but also allows for the implementation of more aggressive control strategies to achieve greater performance. This philosophy is being pursued on a series of Control Structure Interaction (CSI) testbeds at the Jet Propulsion Laboratory.

The effectiveness of viscous elements in introducing damping is a function of several variables, including their number, their location in the structure, and their physical parameters, namely damping and stiffness coefficients. This paper is concerned with the optimal placement and tuning problem for the passive viscous dampers with emphasis on its computational aspects.

Two qualitatively different optimization problems arise in this context: a combinatorial optimization problem which determines the placement of elements, and a mathematical programming problem which optimizes (tunes) the damper parameters. In our approach, a simulated annealing strategy [4] is used for the combinatorial optimization problem, while a sequential quadratic programming algorithm (SQP) [2] is applied to the damper parameter optimization problem. One of the most important ingredients in any optimization problem is the cost functional evaluation, regardless of the performance metric that is used. This is particularly true for the optimal damper placement and tuning problem due to the

complexity of the system. The performance metric chosen here is the $\mathcal{H}_2$-norm of selected transfer functions of interest. An excellent candidate is the transfer matrix between external disturbance inputs and the controlled outputs.

It is well known that the computation of the $\mathcal{H}_2$-norm requires solving a Lyapunov equation. However, due to the high-dimensionality of the system model, it is unrealistic to use the full-order model in any computation. A reduced-order model must be generated to make the computation involved more manageable. The Ritz reduction method that has been studied in [1] is employed to reduce the numerical bottleneck created by solving large systems of this type.

The paper is organized as follows. Section 2 presents the dynamic model of a viscously damped structure. The general optimal damper placement and tuning problem is formulated in Section 3 with a review on the computation of the $\mathcal{H}_2$-norm of the particular transfer matrix which is chosen to be our performance metric. Section 4 addresses the computation issues involved in our optimization problem. In particular, the Ritz reduction method will be described in detail. A number of numerical examples involving the JPL testbed structure are presented in Section 5. Finally, concluding remarks on future work are given in Section 6.

2. Dynamic Modeling for Viscously Damped Structures

Throughout this paper, it is assumed that the dynamics of the undamped structures can be described by a linear, second-order matrix differential equation of the form:

$$M\ddot{z} + Kz = B_d d \,. \tag{1}$$

Here z denotes the n-dimensional vector of generalized coordinates, d is an l-dimensional external forcing input vector, M is the $n \times n$ symmetric, positive definite mass matrix, K is the $n \times n$ symmetric, positive definite stiffness matrix, and B_d is the $n \times l$ forcing input influence matrix.

Assume that a discrete passive damper is placed between two nodal points in the structure, replacing the original structural element. The passive damper is modelled as a device that applies a force at the nodal points with equal magnitude but in opposite directions and proportional to the relative displacement and velocity between the nodal points.

The dynamic structural model incorporating the damper actuator force, u, is written as

$$M\ddot{z} + Kz = bu + B_d d \tag{2}$$

where the vector b represents the influence vector associated with u. The force u generated by the damper is modelled as a constant linear combination of collocated position and

velocity feedback so that

$$u = -(k_p y_p + k_v y_v) \tag{3}$$

with $y_p = b^T z$ and $y_v = b^T \dot{z}$ where y_p and y_v denote the position and velocity "measurements," respectively, and k_v denotes the damping rate, which is always taken as a nonnegative quantity to ensure stability. The parameter k_p is only required to be greater than or equal to the value -k_e, where k_e denotes the stiffness of the structural element that has been replaced by the damper. When $-k_e \leq k_p < 0$, the structure is softened, while $k_p > 0$ causes the structure to be stiffened.

Hence, the dynamic structural model with the inclusion of a passive damper can be represented as

$$M\ddot{z} + (K + k_p bb^T)z = b(-k_p b^T z - k_v b^T \dot{z}) + B_d d \ , \tag{4}$$

or

$$M\ddot{z} + (k_v bb^T)\dot{z} + (K + k_p bb^T)z = B_d d \ . \tag{5}$$

A more general model including multiple passive dampers can be written as

$$M\ddot{z} + (\sum_{i=1}^{n_p} k_{v_i} b_i b_i^T)\dot{z} + (K + \sum_{i=1}^{n_p} k_{p_i} b_i b_i^T)z = B_d d \tag{6}$$

where n_p is the number of passive dampers in the structure.

3. Optimal Placement and Tuning Problem for Passive Dampers

The general optimal placement/tuning problem of passive dampers can be posed as

$$\min_{K_p \in \mathcal{K}_\mathcal{P}, K_v \in \mathcal{K}_\mathcal{V}} \ \min_{B_p \in \mathcal{B}_\mathcal{P}} \mathcal{J}_{cost}(B_p, K_p, K_v) \tag{7}$$

where

- $\mathcal{J}_{cost}(B_p, K_p, K_v)$ is defined as the performance metric for the optimization with a given damper configuration of locations corresponding to B_p and the corresponding stiffness and damping rate K_p and K_v
- $\mathcal{B}_\mathcal{P} \triangleq \{(b_{i_1}, b_{i_2}, \ldots, b_{i_{n_p}}) : i_1, i_2, \ldots, i_{n_p} \in \mathcal{N}_\mathcal{P}, \ i_\alpha \neq i_\beta, \forall \alpha, \beta = 1, 2, \ldots, n_p (\alpha \neq \beta)\}$ (b_{i_α} is the influence vector corresponding to the $i_\alpha{}^{th}$ location).
- $\mathcal{K}_\mathcal{P} \triangleq \{(k_{p_{i_1}}, k_{p_{i_2}}, \ldots, k_{p_{i_{n_p}}}) : i_1, i_2, \ldots, i_{n_p} \in \mathcal{N}_\mathcal{P}, i_\alpha \neq i_\beta, \forall \alpha, \beta = 1, 2, \ldots, n_p (\alpha \neq \beta)\}$ (k_{p_j} is the stiffness correction corresponding to the damper at j^{th} location, and

  $$k_{s_{min}} \leq k_{p_j} + k_{e_j} \leq k_{s_{max}}$$

  where k_{e_j} is the element stiffness of the undamped structure at j^{th} location, k_{smin} and k_{smax} are the lower and upper bound of the damper stiffness).

- $\mathcal{K}_{\mathcal{V}} \triangleq \{(k_{v_{i_1}}, k_{v_{i_2}}, \ldots, k_{v_{i_{n_p}}}) : i_1, i_2, \ldots, i_{n_p} \in \mathcal{N}_{\mathcal{P}}, i_\alpha \neq i_\beta, \forall \alpha, \beta = 1, 2, \ldots, n_p (\alpha \neq \beta)\}$ (k_{v_j} is the damping rate corresponding to the damper at j^{th} location, and

$$0 \leq k_{v_j} \leq k_{v_{max}}$$

where $k_{v_{max}}$ is the highest possible damping rate for the passive damper).

- $\mathcal{N}_{\mathcal{P}}$ is defined as the set of all candidate locations for placement.

It is clear that the above optimization problem is a joint "continuous+discrete" optimization problem. The selection of locations (B_p) for placement is a "discrete" combinatorial optimization problem while the selection of values for K_p and K_v (tuning) is a continuous mathematical programming problem.

Two types of performance metrics are typically considered. The first one is the structural modal damping for selected modes. The computation involved is to solve for the eigenvalues of the "A"-matrix obtained from writing (6) in first-order form for a given damper configuration with corresponding damper stiffness and damping coefficients. The second type of criterion requires both the external disturbance input vector and the controlled output vector to be specified. As discussed in [6], a meaningful and numerically tractable criterion for the associated optimization problem is to minimize the $\mathcal{H}_2$-norm of the transfer function from d to y_o. In addition, a weighting function $W_d(s)$ can be used to model the spectral property of d and a weighting function $W_p(s)$ can be used to improve the performance of y_o over a certain frequency range. In this case, the cost functional is simply

$$\mathcal{J}_{cost} = \|W_p(s) G_p(s; B_p, K_p, K_v) W_d(s)\|_2 \tag{8}$$

where $G_p(s; B_p, K_p, K_v)$ is defined as the transfer matrix from the d to y_o with a given damper configuration of locations corresponding to B_p and with the corresponding stiffness and damping coefficients, K_p and K_v. For a given damper configuration (B_p, K_p, K_v) and the weighting functions ($W_p(s)$, $W_d(s)$), the $\mathcal{H}_2$-norm can be computed through the solution of a specific Lyapunov equation.

Define

$$T(s) = W_p(s) G_p(s; B_p, K_p, K_v) W_d(s)$$

and assume that $T(s)$ has the state-space realization (A, B, C) where the matrix A is asymptotically stable. Then the corresponding $\mathcal{H}_2$-norm of $T(s)$ is simply

$$\|T(s)\|_2 = [trace(CPC^T)]^{1/2} = [trace(B^T Q B)]^{1/2}$$

where P and Q are the positive semi-definite solutions of the following two Lyapunov equations:

$$AP + PA^T + BB^T = 0 \tag{9}$$

and

$$A^T Q + QA + C^T C = 0 \,, \tag{10}$$

respectively [3].

4. Computational Issues and Model Reduction

As discussed in the previous section, the damper placement and tuning problem includes solving a nonlinear mathematical programming problem for tuning, and a combinatorial optimization problem for placement.

In particular, the combinatorial optimization problem is known to be difficult due to the fact that the potential number of candidate locations for placement (N_p) will be large in large space flexible structures. However, relatively few passive devices (n_p) will be available. In general, $N_p >> n_p$, and the total number of combinations, $\frac{N_p!}{n_p!\,(N_p - n_p)!}$, is usually very large. Therefore, it is impractical, if not completely impossible, to try the exhaustive search.

In our approach, a sequential quadratic programming algorithm (SQP) [2] is applied to the damper parameter tuning problem while a simulated annealing strategy [4] is used for the combinatorial optimization problem. The question of developing a hybrid approach for combining these strategies into a single approach will not be dealt with here and is one of our future research topics.

Our current strategy is to solve each of these problems individually. One approach is to solve the damper parameter tuning problem for each candidate location first. These parameters will then be used to evaluate the cost functional in the simulated annealing process.

Another approach is to use a "pruning" process after each of the candidate locations is "tuned." This pruning process is simply to choose the top N_p' candidate locations according to the ranking of their respective optimized cost functional where $N_p >> N_p' > n_p$. An exhaustive combinatorial search is then conducted throughout this subset to find the "optimal" combination of elements which yields the smallest $\mathcal{H}_2$-norm cost. This *ad hoc* pruning approach has been demonstrated to be quite useful. However, it is difficult to make a general statement regarding the solutions of these sub-optimal approaches as compared to the optimal ones.

As stated in the Introduction, one of the most important ingredients in any optimization problem is the cost functional evaluation. This is particularly true for the optimal damper placement and tuning problem due to the complexity of the system. The performance metric chosen here is the $\mathcal{H}_2$-norm of selected transfer functions of interest.

The procedure to compute the $\mathcal{H}_2$-norm of a stable transfer matrix has been given in Section 3 and requires solving a Lyapunov equation. However, it is impractical, if not impossible, to use the full-order model in the computation of the $\mathcal{H}_2$-norm since the order of the model,

$2 \times n$, is typically very large. Hence, a high-fidelity, low-order, reduced model must be used to perform the required computation efficiently.

The Ritz reduction method that has been studied in [1] is employed to reduce the numerical bottleneck created by solving large systems of this type. Details of this model-reduction method will be described in the rest of this section.

The Ritz Reduction Method

To solve the optimization problem posed in the previous section, it is impractical, if not impossible, to use the full-order model in the optimization process since the order of the model, $2 \times n$, is typically very large. Hence, a high-fidelity, low-order, reduced model must be used to perform the required computation efficiently.

The model-reduction method considered here is a second order reduction technique based on reducing the number of generalized coordinates of the system via a transformation of the form $z = \mathcal{P}q$, where $q \in R^N$ with $N < n$. Applying the transformation $\mathcal{P}$ to (6) results in the reduced-order model

$$(\mathcal{P}^T M \mathcal{P})\ddot{q} + \left[\sum_{i=1}^{n_p} k_{v_i} (\mathcal{P}^T b_i)(\mathcal{P}^T b_i)^T\right] \dot{q} + \left[(\mathcal{P}^T K \mathcal{P}) + \sum_{i=1}^{n_p} k_{p_i} (\mathcal{P}^T b_i)(\mathcal{P}^T b_i)^T\right] q = (\mathcal{P}^T B_d) d \ . \tag{11}$$

The transformation matrix, $\mathcal{P}$, consists of the first m ($m << n$) eigenvectors corresponding to the first m eigenvalues, $\{\omega_1, \omega_2 \ldots\ldots, \omega_m\}$, and an additional Ritz vector to account for the static correction for each of the forcing inputs. This method will be referred to as the "Ritz reduction method." A detailed discussion on this subject can be found in [1].

Suppose that the lowest m eigenvalues and their corresponding eigenvectors are known and Φ_m is defined as the $n \times m$ matrix consisting of the m eigenvectors corresponding to $\{\omega_1, \omega_2 \ldots\ldots, \omega_m\}$. Then the desired Ritz vector corresponding to b_i (i^{th} damper) is simply the solution to the following linear equation:

$$K \psi_i = b_i \ .$$

It is desirable for the transformation matrix to preserve M-orthonormality. Therefore, ψ_i needs to be M-orthonormalized. This is done easily by first

1. making ψ_i M-orthogonalized to Φ_m

$$\tilde{\psi}_i = \psi_i - \Phi_m({\Phi_m}^T M \psi_i) \tag{12}$$

and then

2. making $\tilde{\psi}_i$ M-normalized, i.e.,

$$\phi_r = (\tilde{\psi}_i^T M \tilde{\psi}_i)^{-1/2} \tilde{\psi}_i \ . \tag{13}$$

Similarly, the desired Ritz vector corresponding to b_{d_j} (j^{th} external disturbance input) can be computed using the same procedure.

Note that for each of the forcing inputs, one Ritz vector needs to be computed. The forcing inputs could be either the force inputs corresponding to the dampers or external disturbance inputs. Let ϕ_i^r denote the M-orthonormalized Ritz vector corresponding to the i^{th} influencing input vector, b_i, and $\phi_{d_j}^r$ denote the M-orthonormalized Ritz vector corresponding to the j^{th} external disturbance influencing input vector, b_{d_j}. Note that each of the corresponding Ritz vectors is M-orthonormalized to Φ_m; however, the $(n_p + l)$ Ritz vectors may not be M-orthogonal among themselves. An additional M-orthogonalized step is required. Define

$$\tilde{\Phi}_{ritz} = [\, \phi_1^r \ \phi_2^r \ \ldots \ \phi_{n_p}^r \ \phi_{d_1}^r \ \phi_{d_2}^r \ \ldots \ \phi_{d_l}^r \,]$$

and form

$$M_{ritz} = \tilde{\Phi}_{ritz}^T M \tilde{\Phi}_{ritz}, \quad and \quad K_{ritz} = \tilde{\Phi}_{ritz}^T K \tilde{\Phi}_{ritz}$$

to find $\hat{\Phi}_{ritz}$ such that $\hat{\Phi}_{ritz}$ is M_{ritz}-orthonormalized, i.e.,

$$\hat{\Phi}_{ritz}^T M_{ritz} \hat{\Phi}_{ritz} = I_{(n_p+l)\times(n_p+l)} \ , \ and \ \hat{\Phi}_{ritz}^T K_{ritz} \hat{\Phi}_{ritz} = \Omega_{ritz}^2$$

where $\Omega_{ritz} = diag\,[\omega_{r_1} \quad \omega_{r_2} \quad \ldots \quad \omega_{r_{n_p+l}}]$.

Define $\Phi_{ritz} = \tilde{\Phi}_{ritz} * \hat{\Phi}_{ritz}$, then the M-orthonormal transformation matrix $\mathcal{P}$ is

$$\mathcal{P} = [\, \Phi_m \quad \Phi_{ritz} \,]$$

and Eq. (11) is equivalent to

$$I_{N\times N}\ddot{q} + (\sum_{i=1}^{n_p} k_{v_i} (\mathcal{P}^T b_i)(\mathcal{P}^T b_i)^T)\dot{q} + \left[\Omega_N^2 + \sum_{i=1}^{n_p} k_{p_i} (\mathcal{P}^T b_i)(\mathcal{P}^T b_i)^T\right] q = (\mathcal{P}^T B_d) d \tag{14}$$

where $N = m + n_p + l$ is the order of the reduced model, and $\Omega_N = diag\,[\Omega_m \quad \Omega_{ritz}]$.

The reduced-order model in Eq. (10) can also be rewritten in the state-space representation as

$$\dot{x} \ = \ \begin{bmatrix} 0_{N\times N} & I_{N\times N} \\ -\Omega_N^2 - \sum_{i=1}^{n_p} k_{p_i} (\mathcal{P}^T b_i)(\mathcal{P}^T b_i)^T & -\sum_{i=1}^{n_p} k_{v_i} (\mathcal{P}^T b_i)(\mathcal{P}^T b_i)^T \end{bmatrix} \dot{x} \ + \ \begin{bmatrix} 0_{N\times m} \\ \mathcal{P}^T B_d \end{bmatrix} d \tag{15}$$

where $x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$ is the state vector.

5. Numerical Examples

A detailed description of the JPL testbed can be found in [5] (see Figure 1). Briefly, the system is modeled with 249 degrees of freedom and contains 186 candidate locations to insert passive damping elements.

Because the accuracy of the cost functional evaluation methods is of paramount importance in the optimization process, Table 1 contains a comparison of eigenvalue approximations using the full-order model, the Ritz reduced model, and a modally reduced model. The second column in part (a) of the table contains the eigenvalues of the undamped nominal system. All of the other values correspond to the damped system with three viscous dampers placed at the locations 132, 140, and 142. It is assumed that the three dampers have the same damping and stiffness coefficients: 320 $lbs - sec/in$ and 8,000 lbs/in respectively.

The conclusion here is that the Ritz reduction method yields high-precision estimates with enormous reduction in computation. In this example, instead of solving a 498×498 eigenvalue problem, the results can be obtained by solving a 30×30 eigenvalue problem which results from the Ritz reduction method. However, the modally reduced model produces inaccurate results. What is of equal significance is that not only does the modally reduced model produce inaccurate results, it also leads to inaccurate trends for choosing damper parameters. Figure 2 contains damping predictions of the second system mode as a function of the damper viscous parameter coefficient. Note that the full and Ritz reduced models lead to an optimal coefficient of approximately 500 $lbs-sec/in$, while the modally reduced model leads to a significantly larger value that is far from optimal. The Ritz reduction method also leads to very accurate approximation to the $\mathcal{H}_2$-norm, with 6 digits of accuracy.

Table 2 contains the eigenvalues of the damped system where the three dampers are placed at the locations 6, 19 and 91. The three locations are the simulated annealing solution to the optimal damper placement problem. The performance metric is the $\mathcal{H}_2$-norm of the transfer function from an input disturbance located at grid point 412 between the third and fourth bays of the structure, to the outputs consisting of all of the nodal displacements directly beneath the trolley (see Fig. 1). The disturbance was generated as the output of a 6^{th}-order low-pass filter with a bandwidth of 25 Hz. This weighting function is chosen to reflect the objective of disturbance reduction in the frequency range below 25 Hz. A representative comparison of the undamped and damped frequency responses is given in Figure 3.

6. Concluding Remarks

The use of strategically placed and tuned passive elements in future large space structures will play a significant role in their design and development. The ability to analyze, predict, and ultimately optimize system performance with respect to these passive devices is critical for the application of this damper placement technology.

A comprehensive overview of the optimal damper placement and tuning problem was presented in this paper. Approaches and computational aspects of the associated optimization problems were discussed. The results of the paper indicate that significant levels of damping can be introduced into these structures in a very systematic and tailored fashion.

Although reasonably good results have been demonstrated using the approach presented here, the combined discrete plus continuous optimization problem was essentially solved for each individually. This is the major drawback of our current approach. Our future work will concentrate on the development of a hybrid approach to jointly solve the two qualitatively distinct optimization problems.

Acknowledgments

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

References

1. Chu, C.C., and Milman, M., "Eigenvalue Error Analysis of Viscously Damped Structures Using a Ritz Reduction Method," to appear, *AIAA Journal.*

2. Gill, P.E., Murray, W., and Wright, M., *Practical Optimization*, Academic Press Inc., San Diego, CA, 1989.

3. Kwakernaak, H., and Sivan, R., *Linear Optimal Control Systems*, John Wiley and Sons, New York, New York, 1972.

4. Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P., "Optimization by Simulated Annealing," *Science*, Vol. 22, pp. 671–680, May, 1983.

5. O'Neal, M., Eldred, D., Liu, D., and Redding, D., "Experimental Verification of Nanometer Level Optical Pathlength Control on a Flexible Structure," 14^{th} *Annual AAS Guidance and Control Conference*, Keystone, CO, Feb., 1991.

6. Chu, C.C., Fanson, J., Milman, M., and Eldred, D., "Optimal Active Member and Passive Damper Placement and Tuning," *Proceedings of* 4^{th} *NASA/DoD Control/ Structures Interaction Technology Conference*, Orlando, FL, Nov. 5–7, 1990.

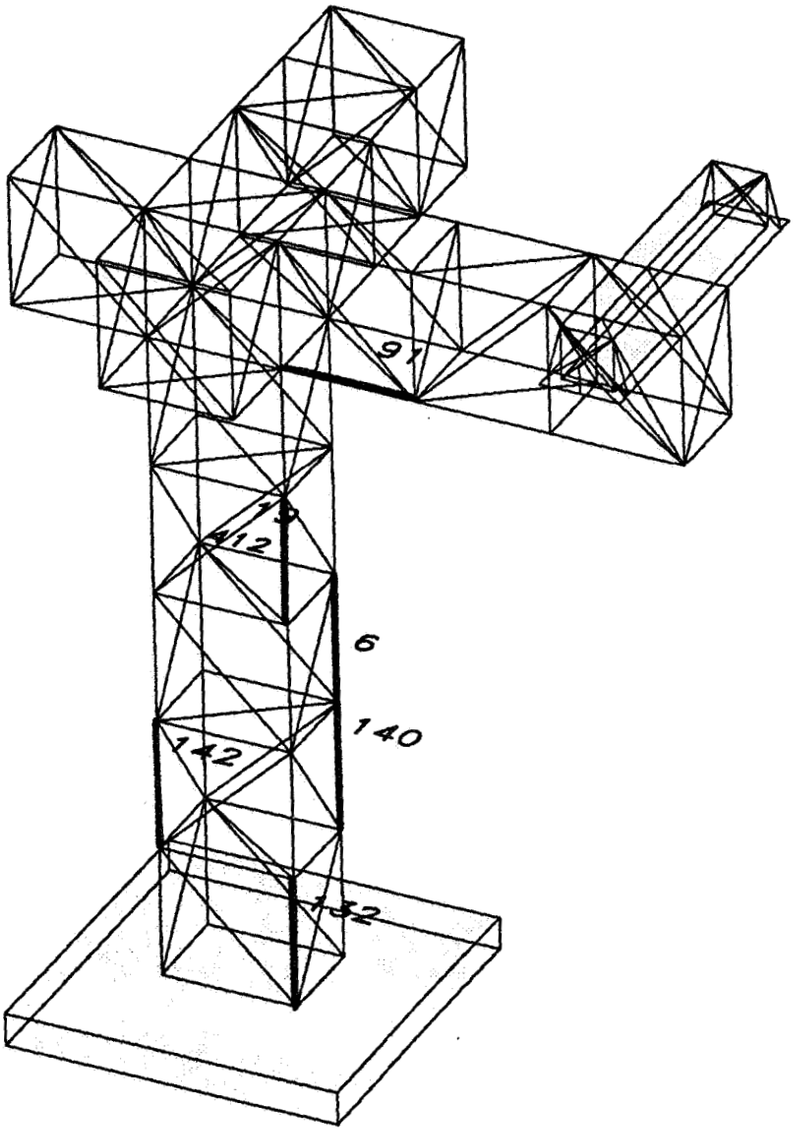


Figure 1. JPL CSI Phase B Testbed

Mode	Undamped System	Damped System		
		249 Modes *(Full order)*	12 Modes *plus* 3 Ritz vectors	15 Modes *(Truncation)*
1	0.7427	0.7420	0.7420	0.7425
2	5.4263	5.2940	5.2940	5.3262
3	7.4565	7.0376	7.0376	6.9540
4	11.6777	10.4862	10.4862	10.4493
5	17.4248	17.4386	17.4386	17.3444
6	20.8423	20.8236	20.8236	20.7055
7	31.1387	31.2231	31.2231	31.0481

(a) Frequency (in Hertz)

Mode	Damped System		
	249 Modes *(Full order)*	12 Modes *plus* 3 Ritz vectors	15 Modes *(Truncation)*
1	0.0179	0.0179	0.0012
2	4.5744	4.5744	0.6125
3	25.5358	25.5357	2.3228
4	32.6380	32.6379	5.5664
5	0.9033	0.9034	0.4066
6	1.3197	1.3197	0.5709
7	0.5013	0.5016	0.5031

(b) Damping (in %)

Table 1. Undamped and Damped Eigenvalues
(Damper Locations: 132, 140, and 142)

Mode	Frequency (Hz)	Damping (%)
1	0.7414	0.0245
2	5.0393	6.8905
3	7.1748	10.0192
4	11.4717	3.7751
5	17.5924	3.2823
6	20.9413	2.0084
7	31.1573	0.0788

Table 2. Eigenvalues of the Damped System with $\mathcal{H}_2$-Optimized Damper Locations at 6, 19, and 91.

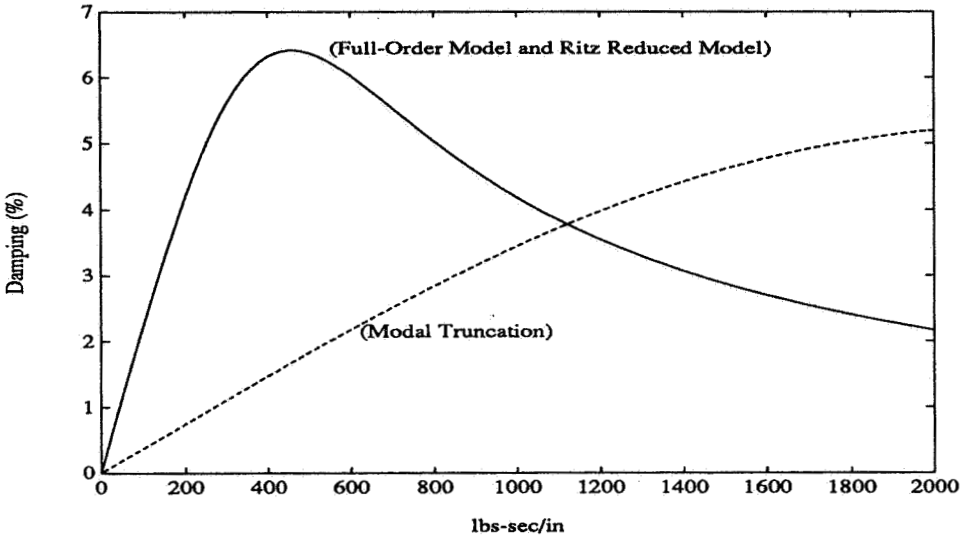


Figure 2. Damping Prediction by Reduction Methods

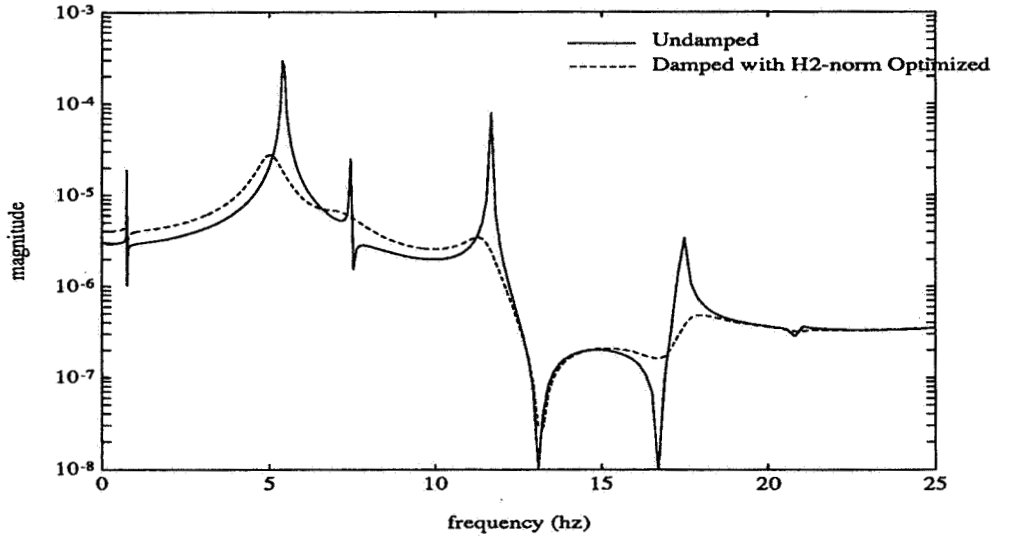


Figure 3. Disturbance Frequency Responses of Undamped and Damped Systems

Computational Control Workstation: Users' Perspectives

Carlos M. Roithmayr* and Timothy M. Straube†
NASA Johnson Space Center, Houston, Texas, 77058

Jeffrey S. Tave‡
Lockheed Engineering and Sciences Company, Houston, Texas, 77258

Abstract

A Workstation has been designed and constructed for rapidly simulating motions of rigid and elastic multibody systems. We examine the Workstation from the point of view of analysts who use the machine in an industrial setting. Two aspects of the device distinguish it from other simulation programs. First, one uses a series of windows and menus on a computer terminal, together with a keyboard and mouse, to provide a mathematical and geometrical description of the system under consideration. The second hallmark is a facility for animating simulation results. An assessment of the amount of effort required to numerically describe a system to the Workstation is made by comparing the process to that used with other multibody software. The apparatus for displaying results as a motion picture is critiqued as well. In an effort to establish confidence in the algorithms that derive, encode, and solve equations of motion, simulation results from the Workstation are compared to answers obtained with other multibody programs. Our study includes measurements of computational speed.

Introduction

A companion paper, Ref. [1], describes in detail a Workstation consisting of hardware and software designed specifically for performing numerical simulations of motions of rigid and elastic multibody systems. Through a series of windows and menus on the Workstation's console, an analyst describes a system's topography and mass distribution, provides information associated with elastic behavior of the system's bodies, and creates geometric representations of each body for animating simulation results. Dynamical equations of motion for the system of interest are derived via symbolic manipulation and an order N algorithm based on Kane's method [2], tailored to take advantage of four parallel processors, and encoded into FORTRAN subroutines. Simulation results can be displayed as numerical values, plots, or a three-dimensional animation.

We are in the process of becoming familiar with the Workstation, and suggesting ways to make it easier to use. Evaluations are presented here that are subjective and, where possible, objective in nature, based on our experiences thus far.

The manner in which multibody systems are described is addressed first. Second, we take up the means of presenting simulation results— in the form of curves, or as a motion picture.

*Aerospace Engineer, Vehicle Dynamics Section.

†Co-op Student (University of Colorado at Boulder), Vehicle Dynamics Section.

‡Engineer; Guidance, Control, and Aerosciences Dept.

Next, comparisons of results with those from other simulation programs are discussed, followed by comparisons of computational speed. Finally, a summary of suggestions for some of the more significant improvements to the Workstation is presented.

Describing a Multibody System

Before a simulation can be performed with the Workstation, one must furnish numerical information about the topography of the system of interest, the mass distribution of each body, and elastic behavior of deformable bodies. If the system's motion is to be animated, geometric representations of each body must be supplied as well.

Data-entry windows appearing on the Workstation console are the principal avenues for supplying mathematical descriptions of a system. Familiarity with a text editor is unnecessary when using the windows, which contain graphical "buttons", and well defined, labeled fields into which numerical values are typed, as shown in Fig. 1. The colors black and grey are used to distinguish relevant information from the irrelevant. For example, if a body is regarded as rigid, "Model Reduction" and all of the labels and fields related to elastic behavior become grey. Messages are produced instantly to give notification of a user's mistakes. All of the information supplied through the windows is stored in a data-entry file. The windows are intended to free analysts from having to know the format of the files; however, the goal is not altogether achieved. Moreover, the windows present a few disadvantages.

Numerical information needed to describe a system often lies scattered over several computers. Data can be "cut" from a file residing on another machine and "pasted" into a data-entry file on the Workstation when the contents of both files are displayed in separate portions of the console; however, this requires a knowledge of data-entry files' format. This kind of information exchange is not possible when data-entry windows are used— an inconvenient state of affairs.

It is easier to check a system description for errors by inspecting a single data-entry file, rather than many data-entry windows. In many cases, correcting slight errors or making minor changes is accomplished more quickly by editing a file than by using the windows. Additional motivation for working directly with data-entry files arises because the Workstation can serve (in addition to an analyst on the console) one or more remote users through a computer network, but the data-entry windows can not be displayed on remote terminals.

The data-entry windows seem most useful when a system is to be described for the first time, or when major changes are to be made. Although the files in which descriptions are stored contain many comments indicating the nature of particular numerical values, it is essential that the user's manual contain a complete explanation of valid options, data types, and proper placement.

A system composed of several rigid bodies fastened together is simpler to deal with than a system that includes deformable bodies. Therefore, the method for describing rigid bodies is reviewed first, followed by a discussion of the process by which an analyst creates

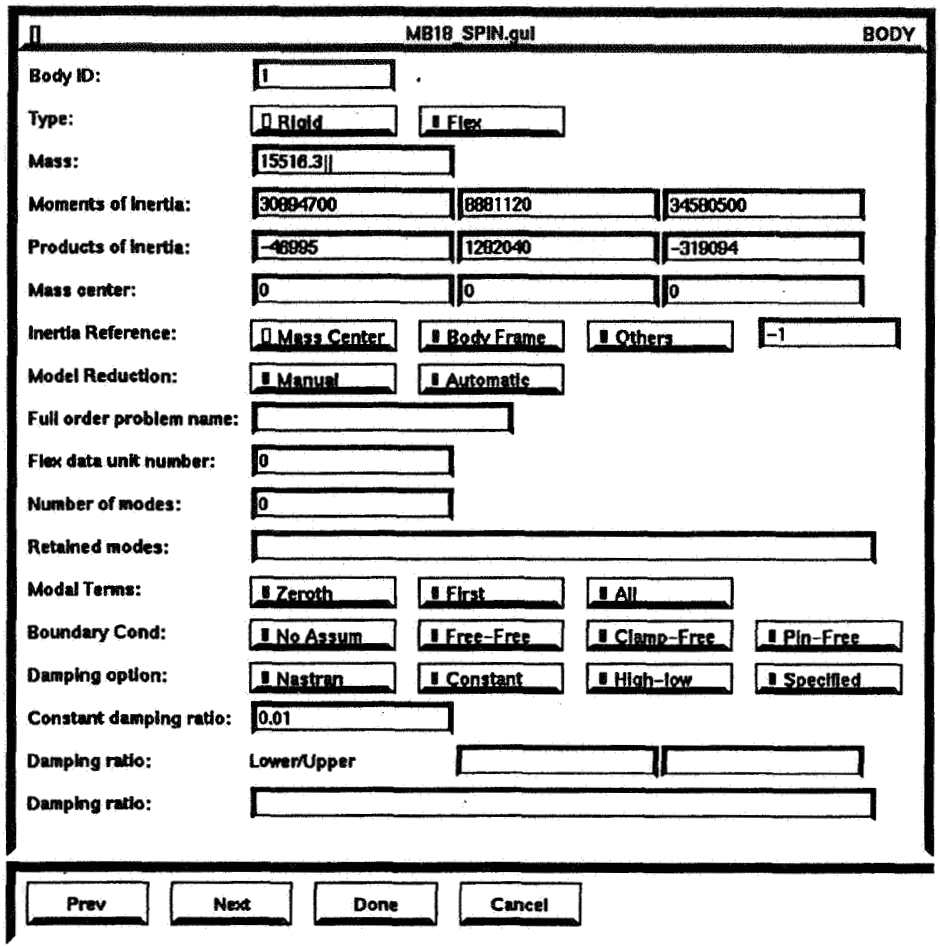


Figure 1: Data-Entry Window

geometrical representations of each body to be included in a motion picture. Finally, the procedure for providing descriptions of elastic bodies is considered.

System Topography and Mass Distribution

System topography (the manner in which bodies are connected) and mass distribution of rigid bodies are described more or less straightforwardly. One way to assess the ease or difficulty of the procedure for describing systems is to compare it to the corresponding activity performed with other multibody software, such as SD/FAST [3, p. 236]. The "free" format and use of keywords in SD/FAST system-description files circumvent many of the problems associated with the Workstation's data-entry windows. Table 1 represents a comparison of the information required to describe a rigid-body system to SD/FAST, and the Workstation; it reveals both similarities and differences.

Table 1: Information Required to Describe System

SD/FAST	Workstation	Comments
Body Name	Body Number	
Mass of Body	Mass of Body	
Three Moments of Inertia	Three Moments of Inertia	
Three Products of Inertia	Three Products of Inertia	(if necessary)
	Position vector from Q, some point on body, to $B^\star$, body mass center [default (1,1,1)].	(With SD/FAST, positions of points are measured from $B^\star$)
	"Inertia reference": Point for which inertia scalars are provided [Q (the default) or $B^\star$].	(With SD/FAST, inertia scalars for $B^\star$ must be given)
	A number to identify each node (point of interest) on each body.	Analyst can make use of predefined sensors and actuators with Workstation, but not with SD/FAST.
Inboard Joint Type (e.g. `slider`, `pin`, `u-joint`)	Inboard Joint Number. Number of • prismatic pins • revolute pins	SD/FAST has a ball joint, Workstation doesn't.
Name of Inboard Body	Numbers of *two* adjacent Bodies and *two* nodes fixed on joint.	
Position vector from $B^\star$ to J, a point fixed in adjacent bodies.	Position vector from Q to J.	
Position vector from inboard body mass center to J.	Position vector from some point on inboard body to J.	

The first item under the Workstation column that lacks a counterpart under the SD/FAST column is the position vector from Q, some point fixed in the body under consideration, to $B^\star$, the mass center of that body. With SD/FAST, the positions of all points of interest in a body are measured *from* $B^\star$; with the Workstation, the positions may be measured from any point, Q. This versatility in the Workstation can be an asset because analysts often receive data in which positions are measured from some point other than $B^\star$. However, unless Q is an interesting point in its own right, measurements from $B^\star$ are preferrable because the amount of information to be supplied is minimized. In practice, Q is often a point that does not come into play in a derivation of equations of motion. A good example of such a point is the geometric center of a space station's central truss. On the Workstation, one can provide measurements from $B^\star$, so long as the position vector from Q to $B^\star$ is made to vanish. Unfortunately, the vector from Q to $B^\star$ that is displayed when a data-entry window first appears is 1 unit in each of three directions. It is extremely unlikely that a vector of this magnitude and direction can be used in practice; one that is 0 units in each direction is more likely to result in a savings of analysts' time.

The second item without a counterpart is a pair of buttons in the data-entry window that gives analysts the option of providing inertia scalars of a body (moments and products of inertia) for either point Q or $B^\star$. *Central* inertia scalars must be used with SD/FAST; in other words, they are for $B^\star$. On the Workstation, point Q is selected when the window first appears. In practice, however, central inertia scalars are used most often, so a small amount of time can be saved if $B^\star$ is preselected by the machine.

Points that remain fixed in two adjacent bodies, points at which forces are applied, and a body's mass center, are all of special interest in connection with formulating equations of motion. The two simulation programs deal with such points in different ways. With SD/FAST, a multibody system's topography and mass distribution are described without reference to points at which forces are applied, and users do not spend time applying numeric or alphabetic labels to points that remain fixed in adjacent bodies. On the Workstation one refers to points of interest (other than mass centers of bodies) as *nodes*, and gives every node a numerical label that can serve as a shorthand for three measure numbers of a position vector. The labels provide an easy way of specifying locations of accelerometers, position sensors, reaction jets, and control-moment gyroscopes, all of which are modeled in pre-written routines available on the Workstation.

The procedure for describing joints that connect adjacent bodies is another area in which SD/FAST differs from the Workstation. Users of SD/FAST may choose a joint from a predefined set, which includes `pin`, `slider`, `u-joint`, etc. The name of an appropriate joint is registered in a system-description file. If a joint that is not a member of the set is to be used, it can be created with the joints available, and bodies without mass. On the Workstation one refers to a joint by a number; indicates whether there are 0, 1, 2, or 3 prismatic pins in the joint; and whether there are 0, 1, 2, or 3 revolute pins in the joint. The Workstation requires no more than six keystrokes: three to erase pre-existing numbers, if necessary, and three to enter the appropriate numbers. If one is constructing a system with joints unavailable in SD/FAST, the amount of labor required by SD/FAST

is obviously greater than that expended on the Workstation, but this situation is probably the exception instead of the rule. When dealing with one of SD/FAST's predefined joints, there is little difference in the level of effort put forth in this area. One joint that can be taken advantage of with version B of SD/FAST, a ball joint, is unavailable to users of the Workstation.

The different approaches to describing joints and points of interest culminates in (or follows from!) a difference in the way one indicates which bodies are connected to one another. The names of one inboard body and one inboard joint are associated with each body described to SD/FAST, with the possible exception of the base body. This information, together with two position vectors, completely describes the way in which adjacent bodies are fastened together. Instead of providing the name of an inboard body, one must, when using the Workstation, supply the numbers of *two* bodies to be connected by a joint, as well as the number of a node on *each* body. The former procedure is decidedly easier than the latter.

A description of a body's orientation is, in general, accomplished with the aid of three orthogonal, right-handed basis vectors, regarded as fixed in the body. Each basis vector associated with a body has the same direction as the corresponding vectors of every other body for the system configuration that is described to SD/FAST. All joint angles and displacements are, by definition, equal to zero in that case. Users of the Workstation do not face a similar restriction. By recording measure numbers of two orthogonal unit vectors in each of two bases fixed in adjacent bodies, an analyst can define angular displacements of a joint to be zero when both bases are not aligned. This feature can prove useful when dealing with NASTRAN information that has been produced by several people not working with a common basis.

Geometric Objects for Animations

Multibody simulations are, first and foremost, performed in order to obtain knowledge of a system's motion. Relatively large changes in position and orientation are best displayed on the Workstation in a three-dimensional animation— an extremely useful complement to a traditional two-dimensional plot, useful for depicting relatively small movements. So-called rigid body motion of a system is represented in an animation, but elastic behavior is not. The facility for animating rigid-body motion is one of the features that distinguishes the Workstation from many other simulation programs.

The apparatus for creating objects for a motion picture is not intended to take the place of a computer-aided design program. It should, however, be capable of producing simple models quickly. Plans call for the Workstation to make use of Integrated Design and Engineering Analysis Software (IDEAS) geometric models for animation, when such models are available.

Rigid geometrical objects that play the part of each body in an animation are constructed in a window that displays three orthographic projections, a perspective view, and several pull-down menus. An object appears in the orthographic projections as a wireframe. In the perspective view, an object can be shown either as a wireframe or a colored solid, and can be

seen from any angle. Each of the four views can be magnified to various levels. Models are constructed with a number of basic two-dimensional and three-dimensional objects: lines, rectangles, triangles, cubes, spheres, and cylinders. Extrusion and revolution, two laborious activities, can be used to create complex objects. The machine responds much more slowly when dealing with many objects, or objects that are complex.

New objects are placed into one of the planar views, and appear simultaneously in all views: their position, size, or orientation can be adjusted by selecting an appropriate option from one of the menus, and moving a mouse pointer to the planar view from which the changes are to be made. Adjustments can be regulated with either a mouse or arrows on the keyboard; both are usually needed because the behavior of the mouse is not easy to control, and the arrow keys act very slowly. Precise changes in an object's orientation or size are difficult to make; consequently, one is often faced with the inconvenience of repeated adjustments.

Before an object can be edited, it must first be *selected*: a procedure not easily performed on the Workstation. The most straightforward way to pick out an object would be to use a mouse to select it from one of the orthogonal projections. However, this approach has proven unsatisfactory: when several objects are close together, the wrong object is often chosen. Instead, one now works the way through three levels of menus to choose from a list of names of basic objects that have been created. The process would be speedier if the list were at a higher level. The current procedure produces lengthy delays while the computer registers a user's choice of an object. On several occasions the program has crashed while dealing with a large number of objects.

Each body is composed of one or more objects, selected and *grouped* together by a user. A body's appearance may be simple or detailed, depending on the number of objects included in the group. A number, corresponding to one of the bodies described in a data-entry window, is assigned by a user to each group. One drawback to this process is that once a group has been formally established, it can not be dissolved. Objects that are part of a group can not be edited, nor can objects be added to or removed from a body. Changing a body's appearance is cumbersome: it must be deleted entirely, and then re-created from scratch.

Up to now, the animation facility's biggest shortcoming has been the necessity for a human to perform two tasks manually. The point Q, described in the Section on system topography, had to be identified for each body. In addition, an orthogonal triad of color-coded vectors had to be correctly oriented in each body to identify the directions that were used in the descriptions in the data-entry windows. If any of this was done improperly, bodies appeared to "jump" at first, or drift apart in the course of an animation. The process involved a great deal of trial and error, and consumed much of an analyst's time. Since the computer is better suited for these jobs, its help is being enlisted in several ways.

First of all, a user assigns a scale to the grid lines in the orthogonal projections. The computer uses information from a data-entry file, together with the scale, to place graphical reference points of each body, and orient each triad of colored vectors correctly. Subsequently, an analyst adjusts the size of geometrical objects and matches up a body with

each triad. Development of automatic placement of reference points and direction vectors is continuing. Two features that might aid a user in proportioning geometric objects are a display of the mouse pointer's coordinates, and an indication of each joint's location.

Information for Elastic Bodies

In the data-entry windows discussed earlier, a body may be designated as either rigid or elastic. For each deformable body, one must provide the number of mode shapes to be used, numbers that identify the modes of interest, and a number to identify a file in which NASTRAN information for the body resides. In return, one is excused from the chore of supplying body mass, mass center position, and inertia information, all of which are contained in the NASTRAN file. In addition, the analyst indicates the nature of structural damping, and numerical values of damping ratios. The presence of *modal integrals* in equations of motion indicates coupling between rigid body motions and motions arising from a body's elasticity, as well as between motions identified with elastic modes. A user is asked to indicate whether modal integrals should contain terms in which mode shapes and slopes appear to 1st, 2nd, or 3rd order. Automatic model reduction, and boundary condition specification, which can be used to simplify equations of motion, are planned but not yet available.

Each point of interest on an elastic body must be associated with a NASTRAN grid point. In the data-entry window for each node, an analyst must record an identifying number, obtained by inspecting a table in a NASTRAN output file. This frees one from the task of furnishing a node's position, which is present in a NASTRAN file. The present Workstation user's manual does not contain a discussion of the way in which NASTRAN must be used to produce data for the Workstation. Moreover, the manual lacks information regarding the maximum number of grid points and mode shapes that can be dealt with.

Plotting and Animating Simulation Results

Time-histories of generalized coordinates, generalized speeds, and a wide variety of other simulation parameters are recorded in a file, and can be displayed as two-dimensional curves. An analyst chooses which variables to plot, and has control over details such as the number of figures on a page, and the scales of the ordinate and abscissa. A change in any *one* of these items requires that the window containing the curves be closed, and *all* of the items over which the user exercises control must be specified again. A considerable amount of time can be saved if repetitious selections are eliminated by allowing a single change to be made while the curves remain visible.

An animation is performed with a geometric model of a system and information from a numerical simulation that has been recorded in a file. Animations can be displayed in one of three orthogonal projections, the perspective view, or all four views at once. Models can be examined from any angle, and portions of any view can be magnified.

The geometric model may be displayed as a collection of wireframes or colored solids.

Animations take place considerably faster when working with the wireframes. One can control the motion picture in much the same way as one uses a video cassette recorder; playing scenes backward or forward in time, and pausing at any point. The Workstation lacks a feature that allows an animation to begin from an instant in time specified by a user. This would be useful for viewing interesting scenes without first watching preceding material.

To summarize, the animation apparatus enables one to check the reasonableness of simulation results in a way that is pleasing to the eye. It is said that a picture can be worth a thousand words: an animation can, in some cases, be worth ten or twenty plots, especially when bodies in a system undergo large changes in orientation.

Simulation Results

In order to check the simulation results given by the Workstation, and form an opinion about the ease with which the Workstation can be used, attempts have been made to re-create several simulations that have been performed by other means. Brief descriptions of those simulations, together with comparisons of the outcomes, shall be given presently. First, however, we take up two general topics related to simulation results.

Initial values of joint angles, displacements, and speeds, as well as orbital parameters, are typed into fields in the data-entry windows. Angular values must be given in units of radians. The authors, who often receive and report information in units of degrees, find this inconvenient. Plots of time histories of angular quantities are presented in radians, making it difficult to compare results to those from other simulations. It would certainly be useful to have an option for working with either unit.

Opportunities for mistakes in deriving, encoding, and numerically integrating equations of motion are legion. The use of symbol manipulation in programs like SD/FAST and AUTOLEV [4], and the Workstation's software, almost eliminate the possibility of human error. Nevertheless, it is always advisable to check the results of numerical integrations. One way of doing so is to evaluate an integral of the equations of motion, if one is available, and verify that it remains constant at every step. To this end, SD/FAST and AUTOLEV can derive and encode expressions for system central angular momentum and kinetic energy, in a Newtonian reference frame. The Workstation lacks a facility for testing results in this way, but plans call for one to be added.

Simulation of Centrifuge Operations

A Space Station Freedom centrifuge facility comprised of two rotors, each 2.5 meters in diameter, is being developed by NASA's Ames Research Center to perform experiments involving plants, rodents, and primates. The service rotor will spin up once or twice a day in order to install or remove experiments from the main rotor, which will remain spinning for about a month at a time.

Refs. [5] and [6] describe simulations carried out at NASA's Johnson Space Center to

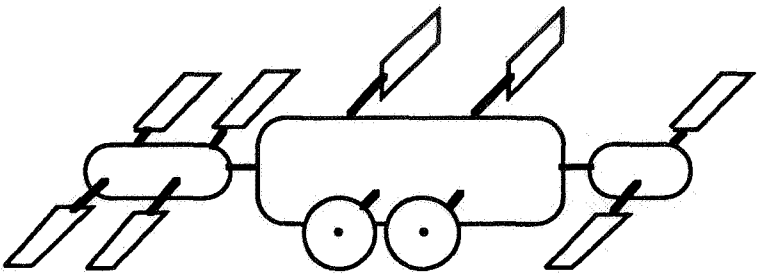


Figure 2: Space Station Topography

quantify the effects of centrifuge operations on the Station's attitude behavior and the performance of Control Moment Gyroscope (CMG) momentum management algorithms. In order to test-drive the Workstation, we attempted to duplicate a simulation very much like those mentioned in Refs. [5] and [6], which were performed with the Space Station Multi Rigid Body Simulation (SSMRBS), a collection of "user-supplied" routines written to work together with routines produced by SD/FAST [3].

The Space Station, sans the centrifuge rotors, is made up of eight bodies fastened together with simple revolute joints, as shown in Fig. 2: a core body, two truss structures immediately outboard of the core, three pairs of solar arrays attached to the outboard truss structures (two to the starboard, one to the port), and two radiator panels attached to the core body.

Each rotor is treated as a disk with uniform mass distribution, attached to the Station's core body by means of a simple revolute joint that keeps the rotor's mass center fixed in the core body. The angular displacements, speeds, and accelerations of the rotors in the core body are prescribed functions of time. As a result, several aspects of centrifuge operations are not taken into account: the change in mass distribution when experiments are moved from one rotor to another, translations of the service rotor, any mass imbalance of the rotors, and motion that results from vibration isolation mounts or centrifuge control systems. The results of interest are not likely to be altered significantly by including these details in a simulation.

The simulation, which is ten orbits in duration, is representative of much of the analysis that is done in support of the Space Station program in the area of attitude control: it involves the CMG attitude control system, controlled motion of outboard truss structures and solar arrays, and prescribed motion of appendages. The initial altitude of the Space Station is approximately 200 nautical miles, and the action of gravitational forces on each body is modeled.

The Preliminary Design Review momentum management algorithm is used to control the orientation of the Station's core body in a local-horizontal-local-vertical reference frame. The motion of each alpha and beta joint is controlled independently by means of a Proportional-Integral-Derivative (PID) feedback scheme. The Workstation subroutines associated with core body and appendage control are, as nearly as possible, identical to those

used with SSMRBS. The current Workstation handbook fails to delineate the argument list that must be present in a user-written "controller" routine. Both centrifuge rotors remain motionless for the first five orbits, after which the angular speeds of both rotors reach an absolute value of 174.3 deg/s in 130 seconds. The angular speed of the main rotor remains constant for the final five orbits, while that of the service rotor remains constant for five minutes and then returns to zero in 130 seconds.

The algorithm used to calculate gains for the PID controllers on the Workstation are slightly different from those used in SSMRBS. Furthermore, there are differences in the ephemerides used to compute the direction to the Sun. Although the simulation results from the two programs share a very strong resemblance, they are, understandably, not identical. In both cases, the system central principal axes of inertia are nearly parallel to local vertical and local horizontal, and normal to the orbit plane; the average steady-state magnitude of the sum of CMG central angular momenta is less than 3,500 ft-lb_f-sec, and the amplitude of the oscillations in the solar arrays' beta joints is approximately 1.5 deg. In neither case does the spin-up of the centrifuge rotors have a pronounced effect on the attitude motion of the core body.

Motions depicted in animations of the simulation results are entirely consistent with the information contained in two-dimensional plots. One interesting observation is that a viewer is presented with an optical illusion in which the main centrifuge rotor appears to spin at approximately the same speed as the outboard truss structures: about 0.064 deg/sec. Movement of the service rotor is barely noticeable. These phenomena are related to the frequency with which results are recorded. The main rotor should appear to spin faster, and more service rotor motion should be visible, if data are recorded more often. However, the animation will last longer. An animation of results from the simulation of centrifuge operations takes just over two minutes to watch when bodies are depicted as colored solids, and data is saved at intervals of 100 seconds. Means of speeding up and slowing down a motion picture would be useful, particularly when angular or linear speeds of bodies differ by, say, more than a factor of 10.

Other Rigid Body Simulations

Results from other simulations of Space Station motion, performed on the Workstation, have been compared to results obtained with Lockheed Engineering & Science Company's Station Control Simulator (SCS), which makes use of an algorithm based on Kane's method, and can simulate motion of systems with elastic bodies. The SCS algorithm is an order N variety [7], patterned after the material in Refs. [8], [9], [10], and [11].

In this Section we discuss three simulations of a rigid body spacecraft's motion, in which forces from a 55 lb_f reaction control jet are applied for 200 milliseconds to the Mission-Build 5 (MB-5) configuration of Space Station Freedom, shown in Fig. 3. A fixed-step, 4th order Runge-Kutta scheme is used in all of the simulations to numerically integrate equations of motion from time $t = 0$ to $t = 20$ seconds, with a step size of 0.01 seconds. Two sensors, a rate gyroscope and an accelerometer, are placed at the center of mass of the Guidance, Navigation, and Control pallet.

Case I involves a rigid body whose mass distribution is identical to that of the MB-5 Space Station. In case II, five bodies are rigidly attached to one another: a core body, a truss structure immediately outboard of the core, two solar arrays connected to the outboard truss structure, and a radiator panel fastened to the core body. The distribution of the system's mass is the same as that of the body in case I. Clearly, a program's results for case I should be identical to those for case II. Case III differs from case II in that the five bodies are fastened together with simple revolute joints; the relative angular speed of each pair of adjacent bodies is a prescribed constant, and initial angular displacement at each joint is non-zero.

Results from simulations performed with the Workstation are in agreement with those obtained from SCS in all three cases.

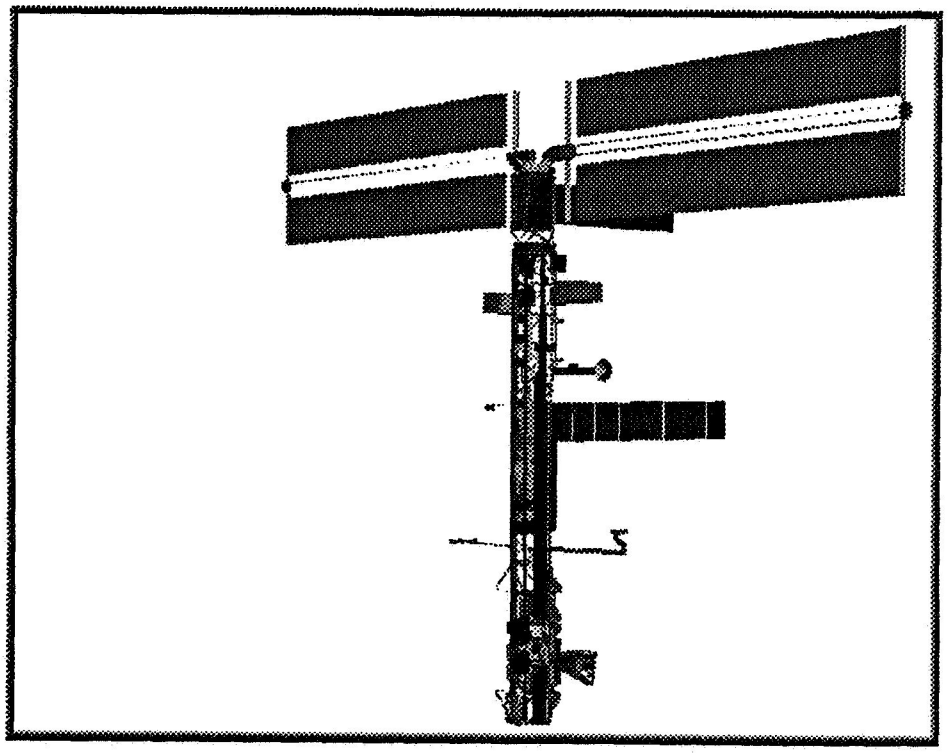


Figure 3: Space Station, Mission Build 5

Elastic Body Simulations

In this Section we compare results of simulations that are similar to those described in the preceding Section; however, the bodies are regarded as deformable.

Case IV, like case I, involves a single body. MacNeal-Schwendler Corp.'s NASTRAN has been employed to compute structure mode shapes and frequencies; 35 modes whose frequencies are below 10 Hz. have been retained. Cases V and VI are the elastic counterparts

of cases II and III, where the number of retained modes for bodies $1, \ldots, 5$ are, respectively, 12, 2, 7, 7, and 6.

The Workstation yields results that are similar to those of SCS in cases IV, V, and VI. Time histories obtained with each program are best described as oscillations superimposed on the curves obtained in the corresponding rigid-body simulations— a reassuring sign.

Computational Speed

The overriding objective in the design of Workstation hardware and software was to maximize computational speed in multibody simulations. Therefore, comparisons with other programs are once more in order. Computational tasks, with the exception of numerical integrations of equations of motion, are performed by the Workstation's Silicon Graphics Personal Iris. Integrations are handled by one or more of the four parallel processors, which are not part of the Iris' hardware.

SD/FAST uses 27.6 seconds of CPU (Central Processing Unit) time on a SUN 4/60 SPARCstation 1 to derive and encode equations of motion for the 10-body space station with centrifuge rotors. The same task is accomplished in 24.4 CPU seconds by the Personal Iris. The 10-orbit simulation is performed in 304 CPU seconds by SSMRBS (also on a SUN SPARC 1), while the Workstation takes about 100 CPU seconds. It is important to have an idea of the number of operations that can be performed over some period of time by each of the machines involved in this comparison. Our best estimate is that the SUN is capable of doing 1.4×10^6 floating point operations per second (flops). The Personal Iris accomplishes about 0.9×10^6 flops, and a Workstation processor probably carries out 10×10^6 flops for this task. The equations of motion written by our copy of SD/FAST are of order N^3, and those from the Workstation are order N, but, in this case, the Workstation makes use of only one of the four processors during the simulation. The system of interest posseses only 13 degrees of freedom, so a significant difference in the speed of the two simulations should not be expected since, as is pointed out in [8, p. 528], the order N and N^3 methods "are roughly equivalent for robots with between six to ten degrees of freedom."

Table 2 presents a comparison of computational speeds for the programs used in cases I–VI. The amount of time (in CPU seconds) required to perform the 20-second simulation is contained in the columns labeled "Sim", while the time spent prior to the simulation to process information from NASTRAN files, in cases IV–VI, is reported in the columns labeled "Flex". The machine on which the SCS resides, a Cyber 930, can perform about 1×10^6 flops. The Workstation is 2 to 3 times as fast as SCS in carrying out the rigid-body simulations, and about 4 times faster at the elastic-body tasks. It should be pointed out that the SCS does not employ symbolic manipulation to derive equations of motion; consequently, it does not enjoy the accompanying advantages in computational speed described in Ref. [3]. However, SCS is about 1.3 times faster than the Workstation at calculating modal integrals, and in case IV, 40 times as fast!

Table 2: Comparison of CPU Time, in seconds

CASE		WORKSTATION		SCS (Cyber 930)	
		Sim	Flex	Sim	Flex
I	One rigid body	87	- -	169	- -
II	One rigid body, 5 pieces	115	- -	373	- -
III	Five rigid bodies	117	- -	383	- -
IV	One elastic body	734	4792	3084	107
V	One elastic body, 5 pieces	352	314	1326	245
VI	Five elastic bodies	352	309	1331	245

Conclusions and Recommendations for Improvements

The Workstation holds a great deal of promise for expeditious simulation of motions of multibody systems: symbolic manipulation, an order N algorithm that accounts for elastic behavior, parallel processors, and a facility for animation are married together in the interest of computational speed and informative display of results. In tests performed thus far, the Workstation yields solutions that agree with those of other programs.

The advances in computational speed can be accompanied by more efficient use of analysts' time: the processes of describing a system and viewing simulation results are hampered by several aspects of Workstation software and documentation. To that end we make the following recommendations.

1. Modify certain pre-selected quantities in the data-entry windows.
2. Speed up selection of geometric objects, and make the menu more accessible.
3. Continue modifications to provide computer assistance in creating geometric models.
4. Furnish means to change motion picture speed, and to specify a time at which an animation begins.
5. Provide an option to work with angular quantities in degrees or radians.
6. Provide a facility for checking results of numerical integrations.
7. Place additional material in the User's Manual
   (a) a complete explanation of data-entry file format, and controller routine argument list.
   (b) documentation on using NASTRAN to produce data for the Workstation.
   (c) information on limitations, such as maximum number of bodies, grid points, and mode shapes.

References

[1] Kumar, M. N., and Venugopal, R., "Computational Control Workstation: Algorithms and Hardware", *Proceedings of the 5th Annual Conference on Aerospace Computational Control*, National Aeronautics and Space Administration, Jet Propulsion Laboratory, Aug. 1992.

[2] Kane, T. R., and Levinson, D. A., *Dynamics: Theory and Applications*, McGraw-Hill, New York, 1985.

[3] Rosenthal, D. E., and Sherman, M. A., "High Performance Multibody Simulations via Symbolic Equation Manipulation and Kane's Method", *The Journal of the Astronautical Sciences*, Vol. 34, No. 3, July–Sept. 1986, pp. 223–239.

[4] Schaechter, D. B., and Levinson, D. A., "Interactive Computerized Symbolic Dynamics for the Dynamicist", *The Journal of the Astronautical Sciences*, Vol. 36, No. 4, 1988, pp. 365–388.

[5] Roithmayr, C. M., "Momentum Manager Performance During Centrifuge Operations", NASA Johnson Space Center, EG2-91-051, Aug. 28, 1991.

[6] Schroeder, C. A., "Momentum Manager Performance During Centrifuge Operations", NASA Johnson Space Center, EG2-91-071, Oct. 22, 1991.

[7] Glandorf, D. R., "An Order-N Algorithm for Open Tree Structural Topologies", LESC-28665, Lockheed Engineering & Sciences Company, Houston, Texas, Jan. 1991.

[8] Rosenthal, D. E., "An Order n Formulation for Robotic Systems", *The Journal of the Astronautical Sciences*, Vol. 38, No. 4, Oct.–Dec. 1990, pp. 511–529.

[9] Rodriguez, G., "Kalman Filtering, Smoothing, and Recursive Robot-Arm Forward and Inverse Dynamics", JPL 86-48, Jet Propulsion Laboratory, Pasadena, California, Dec. 1986. Cited in Ref. [7].

[10] Rosenthal, D. E., "Order N Formulation for Equations of Motion of Multibody Systems", *Proceedings of the Workshop on Multibody Simulation*, Vol. III, National Aeronautics and Space Administration, Jet Propulsion Laboratory, April 15, 1988, pp. 1122–1150.

[11] Singh, R. P., Schubele, B., and Sunkel, J. W., "Computationally Efficient Algorithm for the Dynamics of Multi-Link Mechanisms", AIAA Paper 89-3527-CP, Guidance & Control Conference, Boston, MA, Aug. 1989. Cited in Ref. [7].

Control Structural Interaction Testbed: A Model For Multiple Flexible Body Verification

M. A. Chory, A. L. Cohen, R. A. Manning, M. L. Narigon, V. A. Spector
TRW Space and Technology Group, Redondo Beach, CA

Abstract

Conventional end-to-end ground tests for verification of control system performance become increasingly more complicated with the development of large, multiple flexible body spacecraft structures. The expense of accurately reproducing the on-orbit dynamic environment, and the attendant difficulties in reducing and accounting for ground test effects limits the value of these tests.

TRW has developed a building block approach whereby a combination of analysis, simulation, and test has replaced end-to-end performance verification by ground test. Tests are performed at the component, subsystem, and system level on engineering testbeds. These tests are aimed at authenticating models to be used in end-to-end performance verification simulations: component and subassembly engineering tests and analyses establish models and critical parameters, unit level engineering and acceptance tests refine models, and subsystem and system level tests confirm the models' overall behavior.

The Precision Control of Agile Spacecraft (PCAS) project has developed a control structural interaction testbed with a multibody flexible structure to investigate new methods of precision control. This testbed is a model for TRW's approach to verifying control system performance.

This approach has several advantages: 1) no allocation for test measurement errors is required, increasing flight hardware design allocations, 2) the approach permits greater latitude in investigating off-nominal conditions and parametric sensitivities and 3) the simulation approach is cost effective, because the investment is in understanding the root behavior of the flight hardware and not in the ground test equipment and environment.

Control Structural Interaction Testbed: A Model For Multiple Flexible Body Verification

M. A. Chory, A. L. Cohen, R. A. Manning, M. L. Narigon, V. A. Spector
TRW Space and Technology Group, Redondo Beach, CA

Introduction

Conventional end-to-end ground tests for verification of control system performance become increasingly more complicated with the development of large, multiple flexible body spacecraft structures. These future generations of NASA and DOD spacecraft will require a high level of agility and precision in line-of-sight pointing. In addition, many missions will have multiple gimbaled payloads that must maintain precise pointing despite large maneuvers of the main spacecraft and other appendages. Dynamic range and bandwidth considerations demand a dimensionally stable structure with multiple overlapping control systems. The expense of accurately reproducing the on-orbit dynamic environment, and the attendant difficulties in reducing and accounting for ground test effects limits the value of end-to-end tests.

Conventional spacecraft design techniques in the areas of structures, materials, and control systems are incapable of meeting these future space mission requirements. Improvements are required in all areas, and the new approaches need to be integrated and verified. In particular, an integrated design process is needed in order to exploit the potential synergy among the disciplines while minimizing mission risk due to harmful control/structural interactions.

TRW has developed a method of coordinated control/structural design that has been used to deal with large, structurally complicated spacecraft. This approach involves a combination of analysis, simulation, and test to coordinate the design of the control system and structure. This methodology has two effects: it leads to a truly integrated design process where required, and it reduces the reliance on end-to-end ground test for performance verification.

Instead of an end-to-end ground test, tests are performed at the component, subsystem, and system level on engineering testbeds. These tests are aimed at authenticating models to be used in end-to-end performance verification simulations: component and subassembly engineering tests and analyses establish models and critical parameters, unit level engineering and acceptance tests refine models, and subsystem and system level tests confirm the models' collective behavior.

This verification approach has several advantages: 1) no allocation for system test measurement errors is required, increasing flight hardware design allocations, 2) the approach permits greater latitude in investigating performance under off-nominal conditions and parametric sensitivities and 3) the simulation approach is cost effective, because the investment is in understanding the root behavior of the flight hardware and not in the ground test equipment and environment. In addition, the simulation is a very effective requirements allocation and verification tool.

The Precision Control of Agile Spacecraft (PCAS) Independent Research and Development project has developed a control structural interaction testbed with a multibody flexible structure to investigate new methods of precision control. The test article is an 18-foot long space truss mounted on an air bearing so that it is free to slew over a 60-degree arc. Attached to the truss is a flexible appendage system for study of multiple body interactions. The project also includes

testbed control equipment and measurement instrumentation. TRW's design and verification approach has been used on this testbed..

This paper will briefly describe TRW's approach to multiple flexible body design and verification. The coordinated design and verification methodology used in the development of the control system and the performance simulation will be discussed. The control structural interaction testbed, results obtained from design work, and tests performed to date will be summarized.

Motivation/History

TRW has been a major developer and integrator of space vehicles for over 30 years. A methodology for coordinated control/structural design and verification for complicated satellite systems (large and structurally rich) has evolved over the years that has proven to accurately predict on-orbit performance. The approach does not require end-to-end performance testing but relies on a carefully constructed performance simulation with models authenticated by appropriately defined tests. The extension of this methodology for future large space structures that require state-of-the-art structure design is possible with the use of a control structural interaction testbed.

Testbed facilities are widely used in the study of active control of flexible structures (Reference 1). TRW's testbed design is a natural evolution of years of work in the control structural interaction (CSI) arena (References 2-20). Figure 1 is a summary of TRW's technology heritage regarding CSI.

The testbed design was influenced by TRW's approach to design and verification of large space structures. The testbed embodies the characteristics of and has performance metrics traceable to future agile spacecraft missions. It can be used to verify simulation tools and models and is easily reconfigurable to specific projects.

The Coordinated Design Process

Conventional spacecraft design techniques are based on the independent design of the control and structure subsystems (see Figure 2). Usually the structure is designed with little or no regard for either adverse control/structural interactions or beneficial control/structure synergy. A control system (i.e., sensors, actuators, and control laws) is then designed for the predefined structure, using at most mode separation and mass property information. This approach has been successfully employed on many past spacecraft where mission requirements permitted generous separation between structural frequencies and control bandwidths.

The trend toward simultaneous requirements of large size, light weight, rapid slew, and precision pointing precludes designs that rely solely on control/structure frequency separation. Also, a predetermined structure may unduly restrict the type and location of control sensors and actuators. Iterating the independent structure/control design procedure may, depending on the requirements, finally lead to an acceptable design. However, for demanding missions, the independent design procedure will usually lead to a spacecraft that is far from optimal in terms of weight, size, power, performance, robustness, and design/production cost.

TRW currently implements a coordinated approach to control/structure design (see Figure 3.) This approach combines design information and requirements to blur the traditional separation between the structure and control design. The structure is designed not only to meet loads

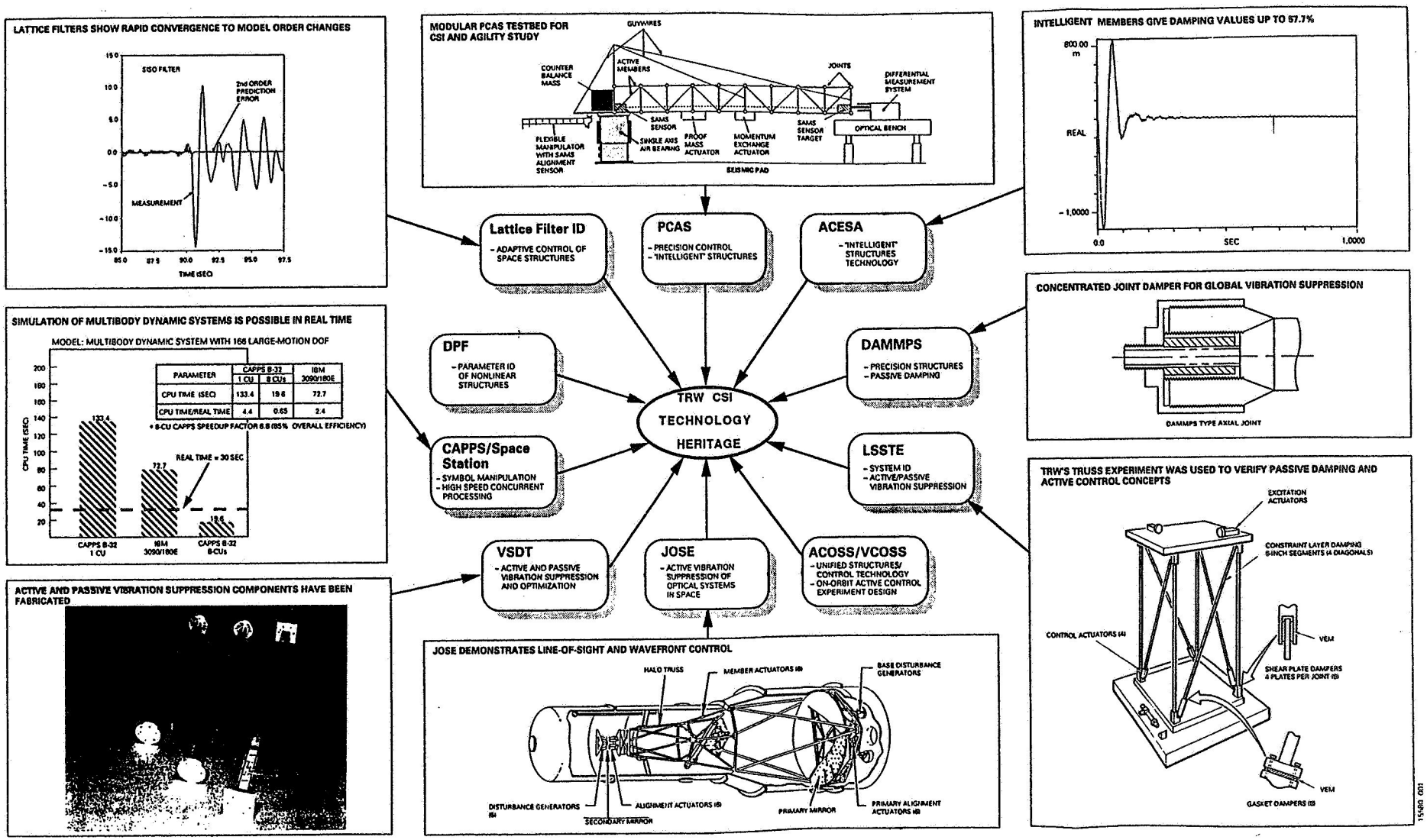


Figure 1.TRW's CSI Technology Heritage

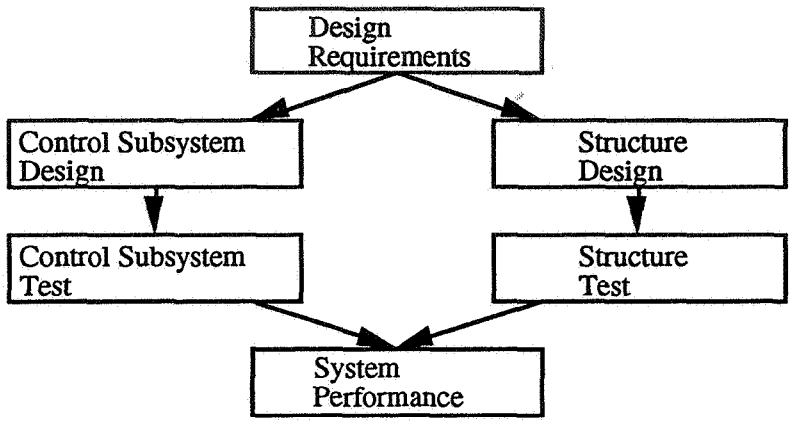


Figure 2. Independent Design

requirements (i.e., stowed loads or stowed stiffness requirements), but also stiffness requirements levied by the control subsystem so that system performance and stability is assured. This approach also is a natural step to an integrated design, where not only is the structure designed so that the control subsystem can meet its performance requirements, but also that the structure subsystem depends on control subsystem performance to meet its design requirements (see Figure 4.) Integrated design opens up a range of new options and approaches. For example, in the independent design process, the structure is often designed to achieve a specific first modal frequency, based on structure/control frequency separation. Total system weight could be reduced, however, by combining active shape control with the structural design to achieve the required total stiffness. In some cases, neither the passive structure nor the active control alone can meet the stiffness requirements, but together they do. Likewise, bandwidth and robustness of the attitude control loops can be increased by incorporating active or passive damping into the structure.

Figure 5 shows a more detailed view of the control system design process with coordinated structural design. Structure/control design iterations are performed concurrently, rather than sequentially. In this way, detrimental interactions can be identified and avoided prior to the fabrication of the flight hardware. Figure 6 shows the concurrent control/structure design process to determine structural stiffness requirements needed for acceptable pointing performance. Once these initial requirements are established, changes to the stiffness requirements resulting from detailed analysis or component tests follow the process shown in Figure 7.

Referring again to Figure 5, this coordinated design process results in the development of a performance simulation that incorporates models of all significant effects, including structural dynamics, control and sensor dynamics, control law implementation, and models of the system environment. This performance simulation is the tool used to assess requirements allocation and design changes, and to incorporate information from component and breadboard/brassboard tests.

The Verification Process

This coordinated design approach has two effects: it leads to a truly integrated design approach where required, and it reduces the reliance on end-to-end ground test for performance verification. Because both the coordinated and the integrated design approach rely on a system simulation for assessment of the design, it is a natural progression to rely on the simulation to

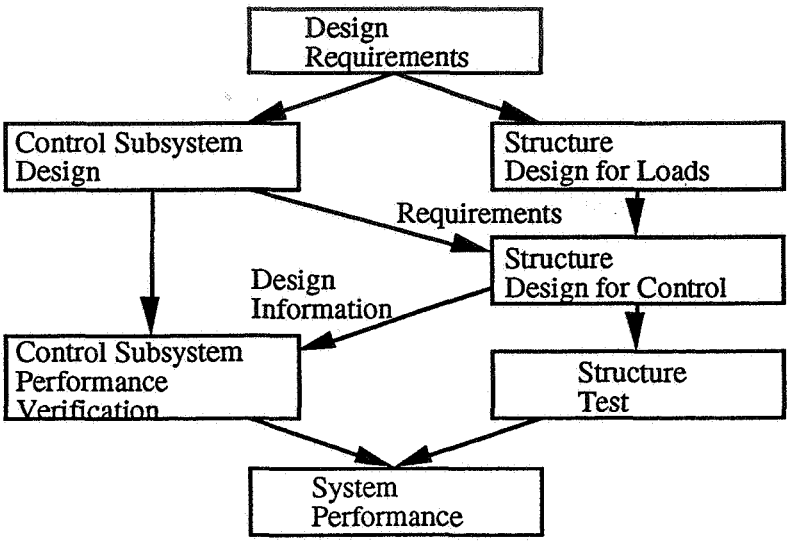


Figure 3. Coordinated Design

support formal verification of the system performance. TRW defines the verification activity as the series of steps taken to show that a design meets its requirements. The validation activity is the series of steps that show the the requirements are consistant.

Three aspects have to be considered: unit (subsystem, box, slice) interface verification, performance verification, and functional validation. Interface verification is the process of determining that the unit meets it interface requirement as specified in the unit specification. For example, this is routinely determined as part of the box acceptance test and is verified at the system level during box integration onto the spacecraft. Performance verification proves that the parameter to be verified satisfies performance specifications. Examples of such elements for an attitude control subsystem include pointing accuracy, jitter, attitude determination accuracy, etc. The final aspect is functional validation, which is the demonstration that the elements function as assumed in various verification processes.

TRW's verification philosophy and design approach are complementary. Functional validation is achieved by early integration of breadboard and engineering models into a hardware-in-the-loop testbed. This provides early checkout of hardware and hardware-software interfaces and validates the overall system model used in the performance simulation. The performance verification is provided by the end-to-end performance verification simulation whose models are anchored by component, assembly, and system level tests.

Functional operation of the system is determined with hardware-in-the-loop tests. These tests help to anchor the performance simulation and assess the implementation of the functional requirements in the flight hardware and software. Figure 8 shows the arrangement of equipment in a hardware in the loop test. In the 1960's and 1970's TRW performed Moving Base Tests to verify the operation of the spacecraft attitude control subsystem. The spacecraft sensors, actuators, and control electronics were mounted on an air bearing so that the entire assembly was free to move in response to the thrusters or reaction wheels. The spacecraft hardwired logic would respond to this motion and the response would be compared to the results predicted by

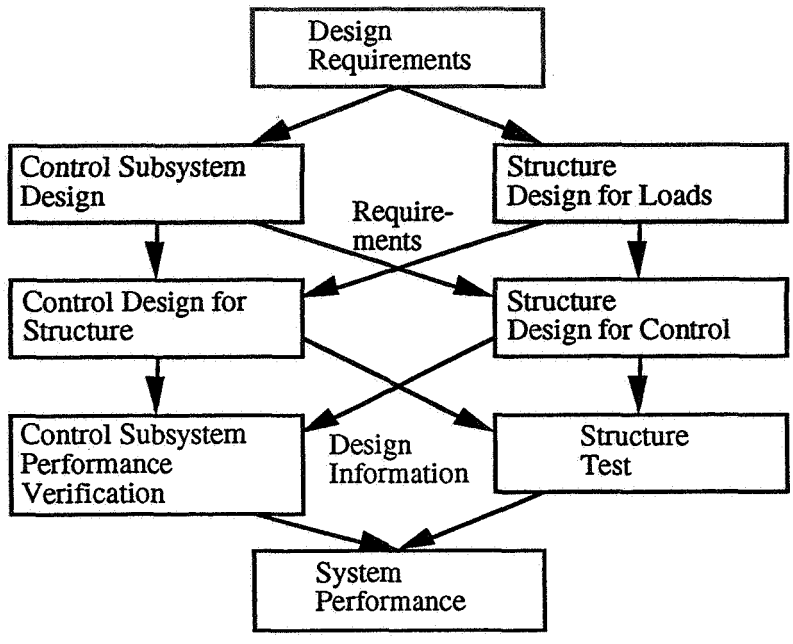


Figure 4. Integrated Design

analysis. These tests verified attitude control subsystem operation as implemented in the logic circuits and validated the mathematical modeling used to design the control laws.

Due to the expense and difficulty of accurately simulating the spacecraft motion and performance with larger spacecraft on an air bearing, Moving Base Tests were replaced by Fixed Base Tests, or hardware-in-the-loop tests. This was possible because a high degree of confidence in the mathematical modeling of rigid body dynamics exists after years of test validation and successful on-orbit performance. The hardware-in-the-loop tests use analog and digital computers together with interfacing electronics to sense control system actuator outputs and provide sensor inputs. The spacecraft components were not free to move, hence the name Fixed Base.

Sensed outputs are input to a computer where the resulting motion of the spacecraft is mathematically modeled. The simulated motion is used to determine the proper stimulation to be applied to the sensors. The simulation of the spacecraft motion was computationally demanding, resulting in the development of special hardware and software techniques to perform these functions. Increases in computer processing speed and reductions in cost have simplified the simulation process.

Because of increasingly complex attitude control subsystem performance requirements, the hardware-in-the-loop tests also changed from being a formal verification of the subsystem performance to an engineering development test. Realtime demands of the test made it impractical to put all the performance simulation fidelity into this test. The formal verification of the subsystem performance is by analysis and the performance simulation. The hardware in the

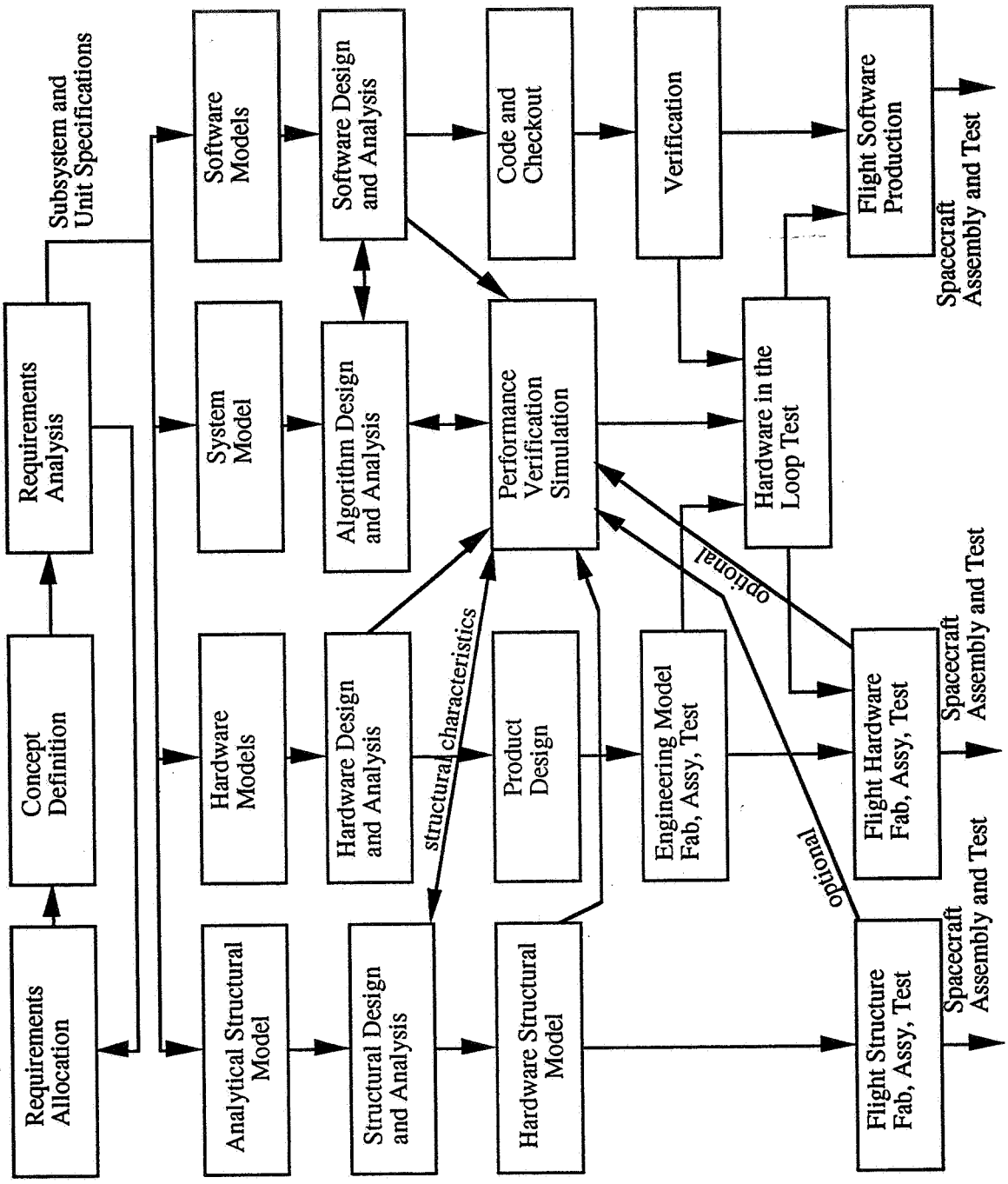


Figure 5. Control System Design Process with Coordinated Structural Design

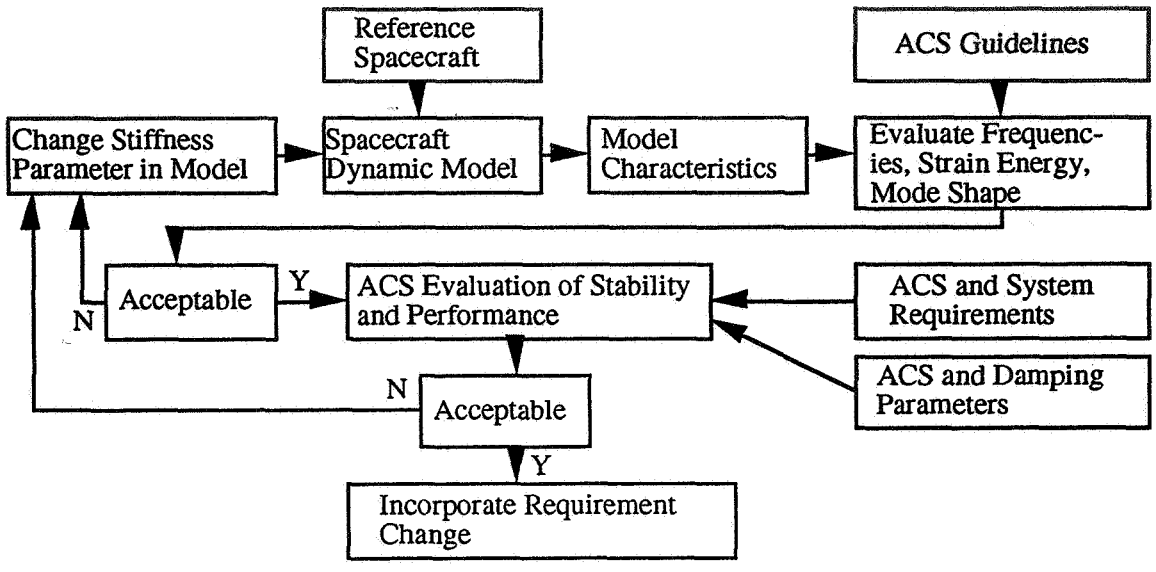


Figure 6. Initial Requirement Development Process

loop test is used to provide confidence in the functionality and compatibility of the control subsystem hardware and software and the model used in the verification simulation.

Performance verification by simulation offers several advantages over a ground based end-to-end performance verification test approach. In the test approach, allocation for test measurement errors erodes already stringent requirements, which compounds flight hardware design challenges. Additionally, if the flight hardware is state of the art, the test support requirements are potentially beyond state of the art or demand expensive precision. The simulation approach is cost effective: the investment is in understanding the root behavior of the flight hardware and its environment, and not in developing elaborate test equipment and environments. Finally, this process permits greater latitude in investigating performance under off-nominal conditions. This design and verification approach has been used on a control structural interaction testbed developed by TRW.

The Control Structural Interaction Testbed

The Precision Control of Agile Spacecraft (PCAS) project is a company funded Internal Research and Development program to investigate new design techniques to deal with future generations of NASA and DOD spacecraft (References 9, 14, 16, 17). These spacecraft will require a high level of agility and precision of line-of-sight (LOS) pointing. Simultaneous requirements on agility and pointing, as well as limits on size and weight present significant technical challenge for spacecraft designers. LOS control for these spacecraft will be needed typically over a wide range of motions (in both amplitude and frequency.) Submicroradian jitter requirements to satisfy payload performance must be balanced with large payload fields of view (i.e., 50 to 1000 microradians) and fields of regard (2 to 50 degrees.) In addition, many missions will have multiple gimbaled payloads that must maintain precise pointing despite large maneuvers of the main spacecraft and other appendages. Dynamic range and bandwidth considerations demand a dimensionally stable structure with multiple overlapping control subsystems. Many missions will require at least four levels of control: slew, attitude, shape, and vibration. The single -bay truss

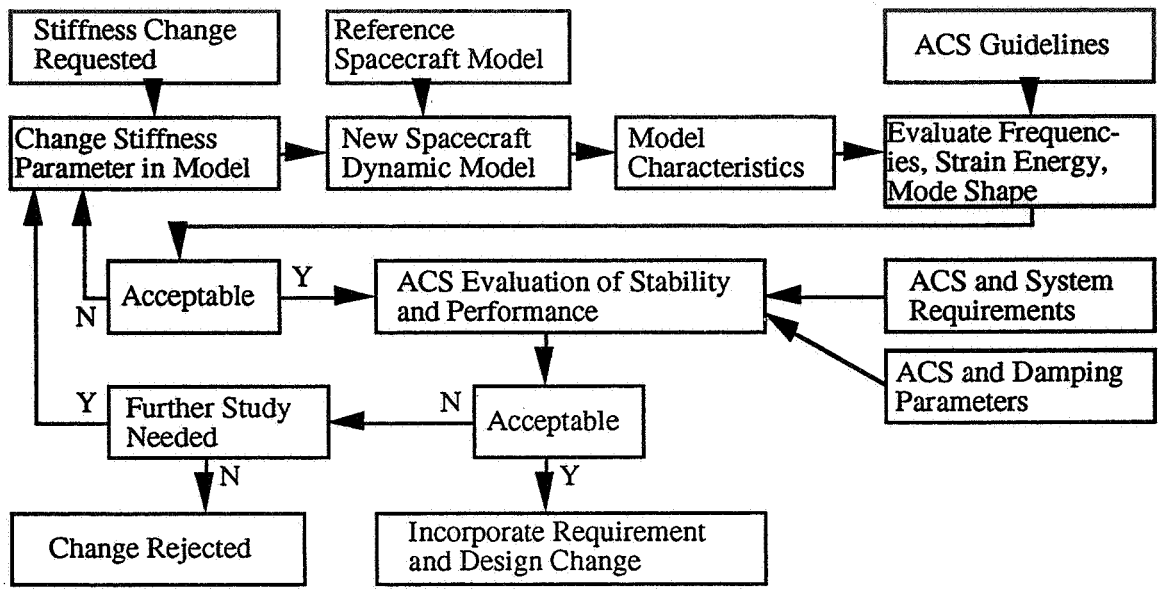


Figure 7. Stiffness Change Evaluation Process

structure (see Figure 9, Reference 4) used for many years of research needed to be recycled to provide a relevant testbed.

A mission analysis was performed to gather a set of requirements for the design of a control structural testbed test article. Candidate configurations were evaluated on the basis of proper scaling of flexible dynamics characteristics, strength, and versatility. A multi-bay truss mounted on a single axis air bearing (with provisions for mounting a flexible appendage) was selected for final detailed design. Conflicting requirements of high angular acceleration, low natural frequencies, and structural integrity were resolved through design iteration. Parameters traded included truss length and width, number of bays, member cross section, joint mass, and inclusion of gravity off-load guy wires.

Figure 10 shows the final truss configuration. The truss is long (18 feet) and narrow (10 inches) to provide low frequencies in the slew plane and higher frequencies in the vertical direction. NASTRAN modeling of the nominally loaded truss yielded a 5.5 Hz horizontal plane fundamental frequency and moment of inertia consistent with the required 4 degree per second per second slew angular acceleration. The truss has sufficient strength to support enough additional mass to reduce the horizontal plane fundamental frequency to the 4 to 5 Hz range, with some reduction in slew acceleration. Guy wires are used to raise the fundamental frequency in the vertical direction to 13.2 Hz while the backing structure provides moment balance and a platform for mounting a flexible appendage.

An evaluation of truss construction methods led to the selection of the patented STAR*NET Structures threaded hub system. This system allows easy interchange of the baseline aluminum structure with passive and active composite members. Control hardware, such as the Surface Accuracy Measurement System (SAMS) sensor targets, reaction wheels, and proof mass actuators, can also be placed at any bay of the truss. This modular design is ideally suited to

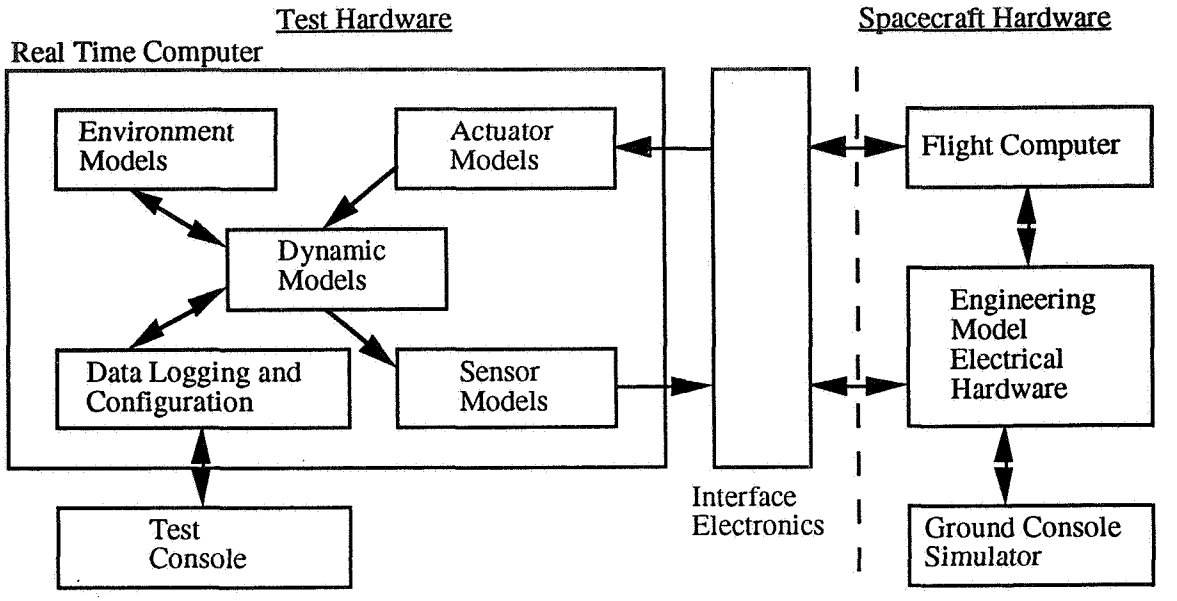


Figure 8. Hardware in the Loop Test Configuration

evaluation of a wide variety of structure and control concepts and allows realistic measures of sensitivity to structural parameters and sensor/actuator location.

Testbed control equipment, shown in Figure 12, is also configured for modularity and interchange ability. A Digital Equipment Corporation MicroVAX II computer hosts the test software and provides interfaces with the Structural Control Processor (SCP) and Numerix vector processor.

The SCP implements the very high bandwidth multi-input multi-output control laws for active vibration control using embedded piezo-ceramic actuators and sensors. The SCP is a TRW developed, flight qualifiable control computer based on 32-bit floating point digital signal processors, and is currently programmed to perform 12 parallel digital filters with sample rates of 2.8 kHz. The SCP has a maximum sample rate of 30 kHz. Adaptive active structural control capability is provided by a high speed serial link for real time update from either the MicroVAX or a PC to the SCP. Identification algorithms and control laws for the appendage and the truss actuators are implemented in the Numerix vector processor. Numerix supplied Analog to Digital (A/D) and Digital to Analog (D/A) cards are used to convert analog sensor inputs to a form suitable for the vector processor and to provide signals for the actuators. The digital input capability of the Numerix vector processor is expanded with a custom digital conditioning electronics assembly.

The truss is carried on a custom Anorad air bearing equipped with a 30 foot-pound peak torque brushless DC motor. Maximum acceleration is greater than 4 degrees per second per second with nominal truss inertia. Slews of up to 60 degrees can be accommodated with the 18 foot truss. The air bearing contains an optical encoder accurate to 1 microradian, which can be read by either the Numerix vector processor or the MicroVAX. Two SAMS sensors mounted on the truss allow relative deflection measurements of the truss during slew. A laser interferometer target is mounted on the truss endpoint to measure slew/settle residual motion.

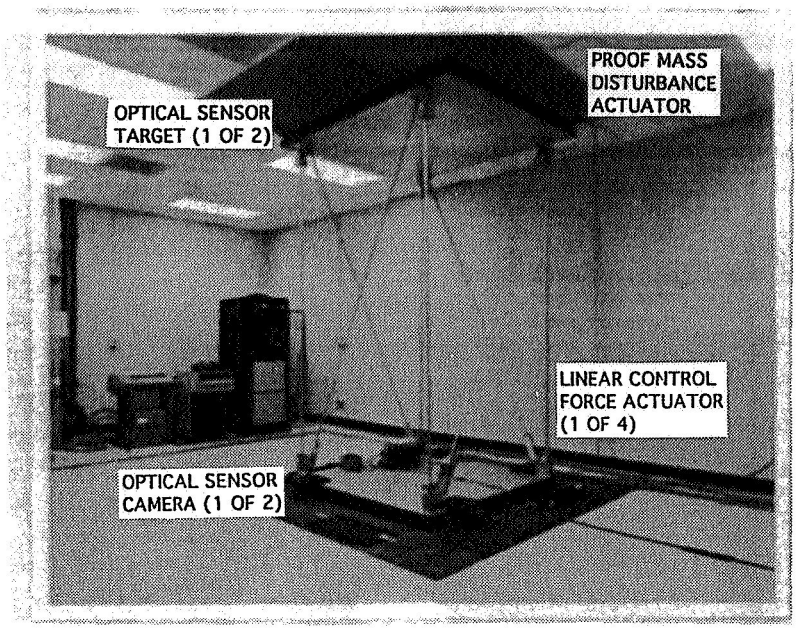


Figure 9. Single Bay Truss

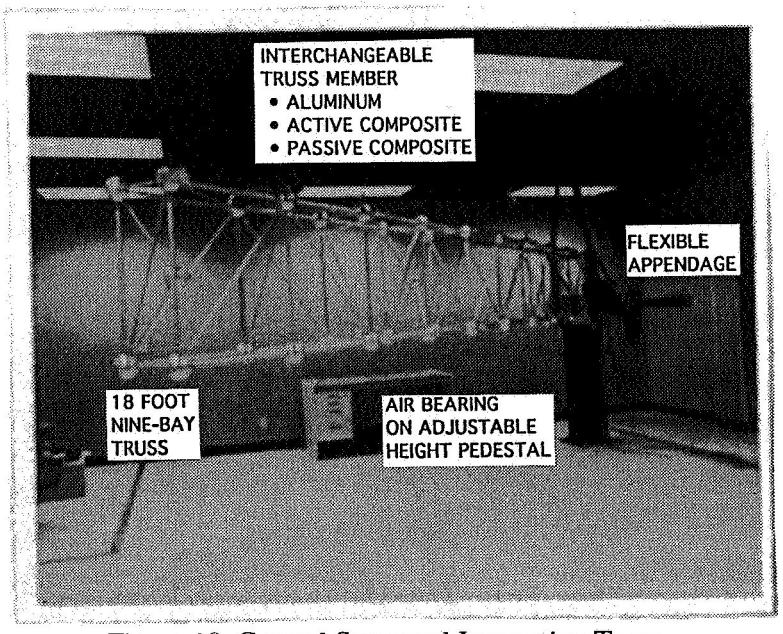


Figure 10. Control Structural Interaction Truss

The flexible appendage system, shown in Figure 12, is configured either for stand alone operation or as part of the multi-body truss. A complete drive system, including a 20 foot-pound peak torque Pacific Scientific brushless DC motor with resolver, motor controller, and computer interface was integrated in 1989. Use of a Ringfeder shaft locking assembly allows convenient interchange of the appendage structure. Presently attached to the drive is a 3 foot long flexible appendage fitted with a SAMS sensor. Four multiplexed light emitting diodes (LED) allow the

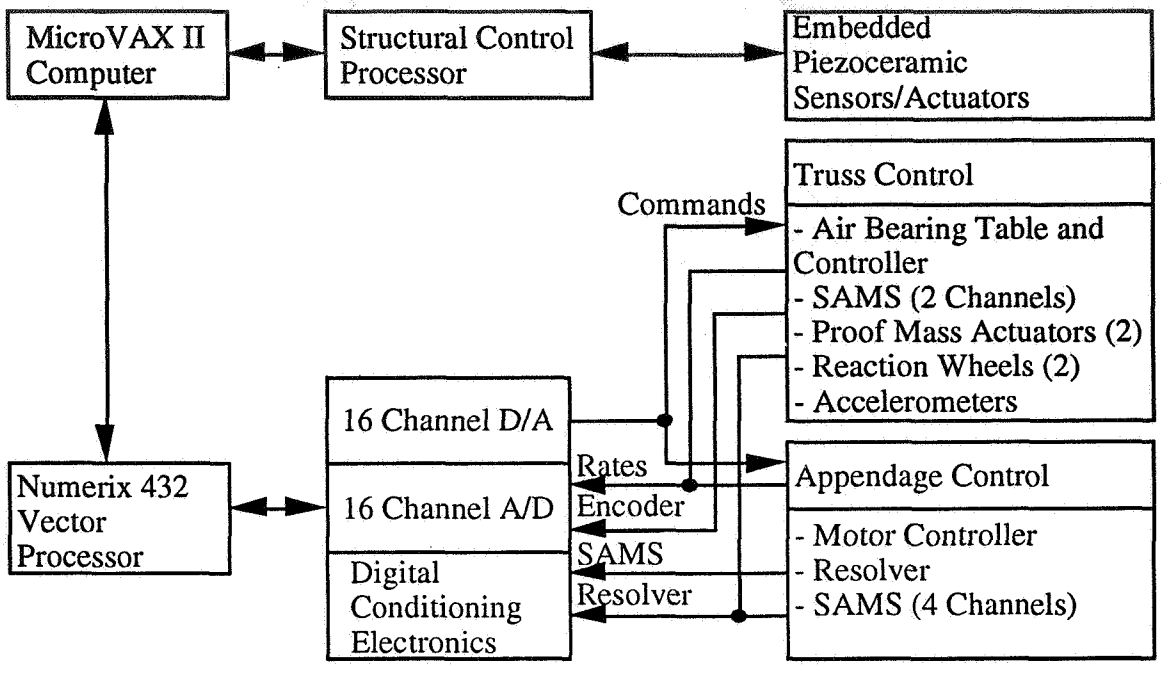


Figure 11. Control Structural Interaction Testbed Control Equipment

SAMS to measure the relative deflection of the appendage at four points along the length. A custom interface with the Numerix vector processor permits reading each LED position at a 250 Hz sample rate, for use in closed loop appendage control laws.

The completed test facility also includes testbed control equipment and measurement instrumentation. A Hewlett-Packard 16-channel modal survey system is used to provide detailed dynamic models of the testbed for either control design or identification algorithm verification. A Hewlett-Packard multi-axis laser interferometer system with 2 nanometer resolution allows structural deformation measurements during slew. A Kaman inductive position sensing system with 100-nanometer resolution provides high bandwidth measurements of residual vibration during the settle phase. A pneumatically isolated optical bench is provided to support the measurements for both truss and component tests.

To provide the test environment required for precision measurements, the facility is built around a 20 foot by 30 foot isolated seismic pad located in a dedicated test room. Adjacent to the test room is an equipment room and a computer/test operator room. Test room walls are treated to increase sound absorption and reduce stray and reflected light. Low velocity air conditioning equipment, separate from the house air conditioning plant, further reduces disturbance on the testbed. Sealed cable ports in the test room walls provide for electrical connections to the computer and equipment rooms, without introducing stray light or air currents.

Testbed Results

The testbed has been used to verify the simulation tools and models used in the precision flexible spacecraft integrated process. A system level modal survey, with multiple accelerometers, was performed to demonstrate the fidelity of the NASTRAN model and obtain accurate modal

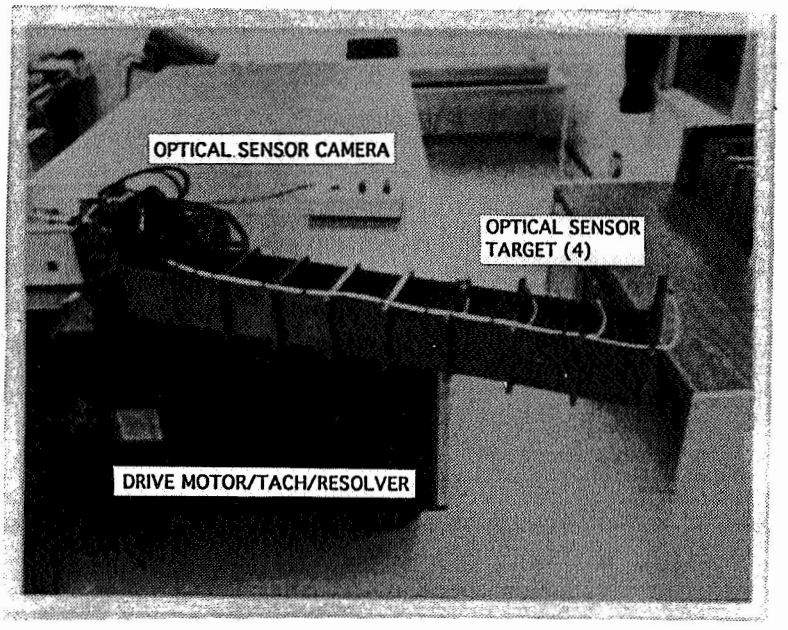


Figure 12. Flexible Appendage System

damping values. Component tests were conducted on the active members to characterize their low level creep and linearity. System level slew tests, with and without active members, were run to verify the assumptions made in the simulation and to understand the nature of the dynamic response.

Agreement between modal survey results and NASTRAN predictions was better than 10% for the first group of modes, and averaged better than 10% for the second group of modes. This level of agreement is satisfactory considering the lack of detail in the model in regard to joint stiffness and member preload from manufacturing and assembly variations in the truss members. Observed modal damping levels of a few tenths of a percent are typical of those expected from a threaded joint type of structure. Modal survey results were used to tune the NASTRAN model to reduce the errors in modal frequencies and mode shapes. Table 1 summarizes the results of the modal survey.

Active (embedded piezo-ceramic) composite member hardware models were verified by component tests. A test fixture was constructed to permit precision measurement of active member motions using the laser interferometer system. Short term and long term creep was measured by applying step changes in voltage to the active member actuators and observing the response with the laser interferometer. Very low values of creep were observed: responses were typically 95% complete in less than 0.1 seconds and 98% complete in less than 0.2 seconds, with total creep in the 1% to 2% range. Active member low level linearity was measured by applying a small sinusoidal voltage to the actuators and heavily filtering the laser interferometer measurement. Response remained linear, with no observable threshold effects, down to the 10 nanometer resolution of the laser interferometer optical configuration.

Performance verification of the testbed and the traceability of the simulation was proven by conducting system level static and slew tests. Multi-mode (horizontal plane, vertical plane, and torsion) active damping in the 10% to 20% range was demonstrated by inserting four active members in the truss at selected locations. Active damping control laws were implemented in the

Table 1. Modal Survey Test Results

Mode	Mode Description	NASTRAN	Test Data	Prediction Error (%)
		Frequency (Hz)	Frequency (Hz)	
1	Rigid Body	0.00	0.00	0.0
2	Horizontal Bending	4.79	4.64	3.1
3	Vertical Bending	5.36	5.85	9.1
4	Torsion	13.49	13.89	3.0
5	Vertical Bending	22.66	26.29	16.0
6	Horizontal Bending	38.43	34.05	11.4
7	Torsion	42.18	41.64	1.3

structural control processor sampling at 2.8 kHz. Slew control laws were implemented in the Numerix computer.

Figure 13 shows a comparison of simulation and test results for a typical truss slew case with a 3.5 degree slew in 2 seconds. A high level of correlation between simulation and test is seen during the slew phase. During the settle phase, both cases show similar level of ripple at the 6.4 Hz frequency of the first gain stabilized mode. The discrepancy in response during the early part of the settling was traced to unmodeled dynamics in the air bearing torquer, which will be corrected in the next model update. This demonstrates the value of modelling iterations, based on test results, to ensure valid simulations. However, since the airbearing is not part of the flight system, it also demonstrates the difficulty of removing all test artifacts from an end-to-end test.

Summary

TRW has developed a testbed that provides the critical capability to validate control system performance for multibody flexible structures for future NASA and DoD missions (high levels of agility, submicroradian jitter requirements, large structures). The testbed is easily reconfigurable and can be used to validate key models and simulation tools. The emphasis of the testbed design is to validate performance simulation tools required and is not intended as a complete ground-based end-to-end test.

This approach to testbed design is very cost-effective and flexible. TRW's methodology is based on a proven building block process whereby a combination of analysis, simulation and test has replaced end-end performance verification by ground test. Many advantages to this approach were described. In particular, the simulation approach grounded by appropriate hardware test data is cost-effective because the investment is in understanding the root behavior of the flight hardware and not in the ground test equipment and environment.

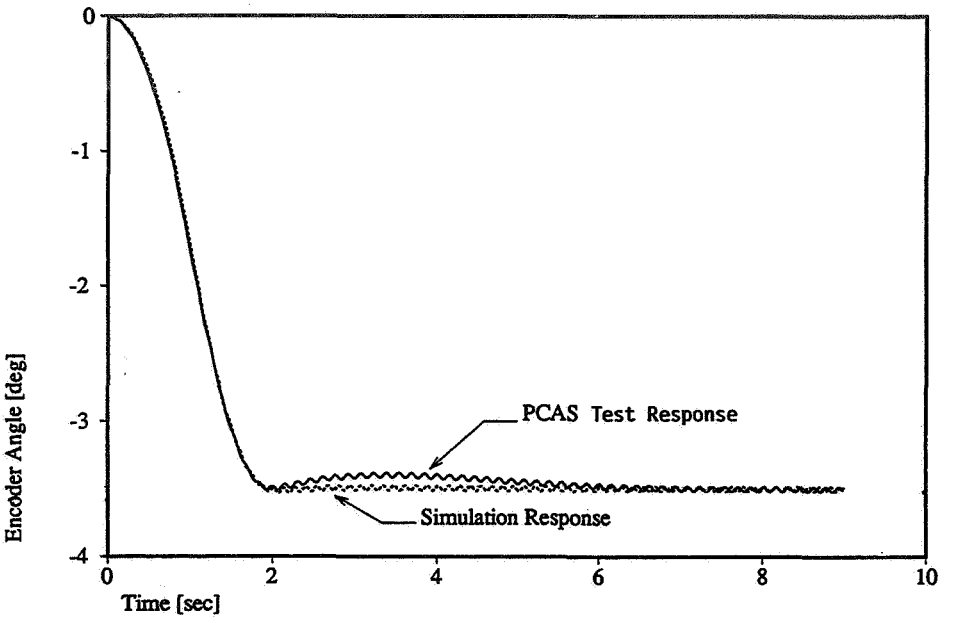


Figure 13. Response to Bang-Bang Slew Profile

Acknowledgements

The authors gratefully acknowledge A.W. Fleming and J.M. Maguire for their material on the TRW verification and validation methodology. Also, Todd Mendenhall and Maribeth Roesler have greatly contributed to the PCAS IRAD project. The work led by A.J. Bronowicki in intelligent structures technology is integral to the design process for future missions. We would also like to acknowledge Irene Curtis for her support in preparing this paper.

References

1. Sparks, Dean W., and Jer-Nan, Juang, "Survey of Experiments and Experimental Facilities for Control of Flexible Structures," Journal of Guidance, Control, and Dynamics, Vol. 15, No.4, July-August 1992, pages 801-816.

2. Bronowicki, A.J., Lukich, M.S., and Kuritz, S.P., "Application of Physical Parameter Identification to Finite Element Models," 1st NASA/DoD CSI Technology Conference, Norfolk, VA, November 18-21, 1986.

3. Bronowicki, A.J., "Structural Modification to Minimize Response of Electro-Optical Systems to Random Excitation," Structural Mechanics of Optical Systems II, Vol. 748, Proceedings of SPIE, January 13-15, 1987.

4. Lukich, M.S., "System Identification and Control of the Truss Experiment: A Retrospective," AIAA Guidance Navigation and Control Conference, Minneapolis, MN, August 15-18, 1988.

5. Lukich, M.S., Dailey, R.L., Balas, G.I., and Doyle, C.J., "Robust Control of a Truss Experiment," AIAA Guidance, Navigation and Control Conference, Minneapolis, MN, August 15-18, 1988.

6. Spector, V.A. and Flasher, H., "Sensitivity of Structural Models for Non-Collocated Control Systems," AIAA Guidance, Navigation and Control Conference, Minneapolis, MN, August 15-18, 1988.

7. Iwens, R.P. and Major, C.S., "The JOSE Preliminary Design," AIAA Guidance, Navigation and Control Conference, MN, August 15-17, 1988.

8. Spector, V.A. and Flasher, H., "Sensitivity of Structural Models for Non-Collocated Control in Flexible Systems," ASME Journal of Dynamic Systems, Measurement and Control, Vol. 111, No. 4, December 1989.

9. Spector, V.A., Narigon, M.L., Manning, R.A., Roesler, M.D., and Wise, D.W., "Development and Verification of Key Technologies for Agile Spacecraft," presented at the American Control Conference, San Diego, CA, May 1990.

10. Spector, V.A. and Flashner, H., "Modeling and Design Implications of Non-Collocated Control in Flexible Systems," ASME Journal of Dynamic Systems, Measurement and Control, Vol. 112, No. 2, June 1990.

11. Betros, R.S., and Dvorsky , G.R., "Encapsulation Technique to Enhance Actuator Performance in Composite Beams," presented at the Fourth NASA/DoD Controls-Structures Interaction Technology Conference, Orlando, FL, November 5-7, 1990.

12. Bronowicki, A.J., Mendenhall, T.L., Betros, R.S., Wyse, R.E., and Innis, J.W., "ACESA Structural Control System Design," First Joint U.S./Japan conference on Adaptive Structures, November 1990.

13. Manning, R.A., "Optimum Design of Intelligent Truss Structures," AIAA/ASME/ASCE/AHS/ASC Structural Dynamics, Materials Conference Proceedings, Baltimore, MD 1991.

14. Chory, M.A., Roesler, M.D., Spector, V.A., Bayo, E., Flashner, H., and Jabbari, F., "Control Structural Interaction Testbed: A Model for University Industry Interaction," IFAC Conference on Advances in Control Education, Boston, MA, June 24-5, 1991.

15. Betros, R.S., Bronowicki, A.J., and Manning, R.A., "Adaptive Structures Technology Programs for Space Based Optical Systems." SPIE Applied Science and Engineering Conference Proceedings, San Diego, CA July 1991.

16. Manning, R.A., Mendenhall, T.L., and Spector, V.A., "Quieting Technologies for Precision Slewing Spacecraft," The 33rd AIAA/ASME/ASCE/AHS/ASC Structures Structural Dynamics, and Materials Conference Proceedings, Dallas, TX 1992.

17. Manning, R.A., and Spector, V.A., "Precision Slew/Settle Technologies for Flexible Spacecraft," Fifth NASA/DoD Controls-Structures Interaction Technology Conference Proceedings, Lake Tahoe, NV 1992.

18. Betros, R.S., Steffen, N.R., and Triller, M.J., "ARMA Models for Real-Time System Identification of Smart Structures," Proceedings of First European conference on Smart Structures and Materials, Glasgow, Scotland, May 1992.

19. Bronowicki, A.J., Betros, R.S., Nye, T.W., McIntyre, L.J. , Miller, L.R., and Dvorsky, G.R., "Mechanical Validation of Smart Structures," Proceedings of First European conference on Smart Structures and Materials, Glasgow, Scotland, May 1992.

20. Nye, T.W., Ghaby, R.A., Dvorsky, G.R., and Honig, J.K., "The Use of Smart Structures for Spacecraft Vibration Suppression," International Astronautical Federation, Materials and Structures Symposium, Washington DC, August 1992.

PACE: A Test Bed for the Dynamics and Control of Flexible Multibody Systems

Moon K. Kwak, Monty J. Smith and Alok Das

The USAF Phillips Laboratory
OLAC PL/VTSS
Edwards AFB, CA 93523

Abstract

The Phillips Laboratory at Edwards AFB has constructed a test bed for the validation and comparison of modeling and control theories for the dynamics and control of flexible multibody systems. This project is called the Planar Articulating Controls Experiment (PACE). This paper presents the experimental apparatus for PACE and the problem formulation. An in depth analysis on DC motor dynamics was also performed.

1. Introduction

The Air Force remains active in space systems, and hardware, such as space robots, rotorcraft and spacecraft, consists of subsystems which can be described as flexible multibodies. The dynamics and control of flexible multibody systems has been of interest for many years. (Refs. 1-11). Identifying, modeling and controlling such systems using various theories with confidence has also become an important issue. At present, a real need exists for the validation and comparison of various modeling and control theories based on an actual hardware experiment. However, compared to theoretical developments and number of computer programs available, experimental verification has never been conducted. To this end, the Phillips Laboratory at Edwards AFB has constructed a flexible multibody structure which consists of 2 flexible beams connected in series with motors at both the hub and the elbow joint. Figure 1 shows the multibody body structure named the Planar Articulating Controls Experiment (PACE).

2. Experimental Apparatus

Figure 2 shows the PACE control loop which includes 2 flexible arm manipulator, various sensors and actuators, amplifiers, an AC-100 data acquisition controller and the VAX workstation. Actuators and sensors are connected to D/A and A/D interfaces of the AC-100 controller through amplifiers. Sixteen channels are available on each D/A and A/D interfaces. The VAX 3100 workstation controls the AC-100 data acquisition controller with the ISI real-time monitor software which includes MatrixX, Autocode and Interactive Animation module.

The motion of the multibody test article is constrained to horizontal motion on a 14 foot square granite table. An air compressor is used to activate an air bearing, which allows the whole system to float on air so that friction forces do not exist between the support plate and the granite table. The shoulder motor is fixed at the center of the granite slab and the elbow motor is located at the junction of two beams. Thus, the arms can be driven

through some trajectory using the DC motors.

Sensors for PACE include:

1. **Encoders** count the change of a square wave; its resolution is 2540 cycles per revolution. A differential interface converts the encoder output to the number of changes in the data acquisition time interval, which is proportional to the angular velocity of the motor shafts. Tachometers are not being used currently due to their inaccuracy in measurements, as reported by the manufacturer, PMI company.

2. **Lasers** trace the positions of two arms. JPL is designing and testing the hardware implementation.

3. **Accelerometers** will measure accelerations of arbitrary points of interest on the structure.

4. **Piezoceramic Sensors** will measure strains along beams.

Actuators for PACE include:

1. **PMI DC motors** provide primary torques to slew the arms. Both motors have a gear box.

2. **Piezoceramic actuators** will suppress vibrations of the arms.

3. **Reaction wheels** (Optional) will apply a secondary moment to suppress vibrations of the arms.

The free body diagram for PACE, seen in Fig. 3, can be divided into 4 components; 2 DC motors and 2 flexible beams with tip masses at both ends. The dynamics of DC motors should be incorporated into the dynamics of the whole system. Modeling of the PMI DC motor with a gearbox is not an easy task since the gear box of DC motor causes noise, backlash and power loss.

3. Problem Formulation

Before the integration of the components of PACE, DC motor identification was performed using the unit-step and sine-sweep response tests. The sine-sweep response of the elbow, shown in Fig. 5, illustrates the dominant effect of static friction torque on the system response. Based on these tests, the mathematical model includes mass moment of inertia, viscous and Coulomb damping and static friction. Thus, the equation of motion for the shoulder motor is given as

$$n^2 I_{sm}\ddot{\theta}_1 + C_{sv}\dot{\theta}_1 + C_{sc}\, sgn(\dot{\theta}_1) = n(T_s - T_{sf}) - T_1 \tag{1}$$

while the elbow motor is described as

$$n^2 I_{em}\ddot{\theta}_2 + C_{ev}\dot{\theta}_2 + C_{ec}\, sgn(\dot{\theta}_2) = n(T_e - T_{ef}) - T_2 \tag{2}$$

where n is the gear reduction ratio, I_{sm} and I_{em} are the mass moment of inertias of the shoulder and elbow motors, respectively, θ_1 and θ_2 are the angular displacements of each motor relative to motor housings, C_{sv} and C_{ev} represent the viscous damping coefficients of each motor, C_{sc} and C_{ec} represent the Coulomb frictions of each motor, T_s and T_e are the motor torques, T_{sf} and T_{ef} represent the static friction torques of each motor, and T_1 and T_2 represent the torques transferred to each arm of the shoulder and the forearm, respectively. Note that the subscripts s and e denote the shoulder motor and the elbow motor, respectively. The static friction torques, T_{sf} and T_{ef}, are expressed in terms of the motor torques, T_s and T_e.

$$T_{sf} = \begin{cases} -T_{sf}^* & \text{if } T_s < -T_{sf}^*; \\ T_s & \text{if } -T_{sf}^* \leq T_s \leq T_{sf}^*; \\ T_{sf}^* & \text{if } T_s > T_{sf}^*. \end{cases} \tag{3}$$

$$T_{ef} = \begin{cases} -T_{ef}^* & \text{if } T_e < -T_{ef}^*; \\ T_e & \text{if } -T_{ef}^* \leq T_e \leq T_{ef}^*; \\ T_{ef}^* & \text{if } T_e > T_{ef}^*. \end{cases}$$

Based on the circuits of the DC motors and the servo amplifiers, we can express the motor torque in terms of the input voltage to the servo amplifier

$$T_s = K_{sT}(\alpha_s V_s + \beta_s)$$

$$T_e = -K_{eT}(\alpha_e V_e + \beta_e) \tag{4}$$

where K_{sT} and K_{eT} are motor constants, α_s, α_e and β_s and β_e are constants of servo amplifiers, and V_s and V_e are the input voltage to the servo amplifiers, respectively. Equations (3) and (4) relate the output voltages of the AC-100 controller to the motor torques of DC motors. Note that the minus sign is added in Eq. (4) because of the direction. Parameters of the PMI DC motors were identified independently due to errors in the manufacturer data. The identification results for the shoulder and elbow motors are tabulated in Tables 1 and 2. The material properties of each arm, listed in Fig. 5, will be identified in the near future.

Equations of motion for PACE are derived using the Lagrangian approach (Ref. 12). Since each beam is represented by a distributed parameter system, the resulting equations have the form of hybrid equations, i.e., the combination of ordinary differential equations and partial differential equations. For the numerical simulation and control design, the elastic displacements are discretized with the assumed-mode technique, resulting in high-order nonlinear ordinary differential equations. Thus, recalling Eqs. (1) and (2), we may write the following state equation for the whole system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{F}) \tag{5}$$

where

$$\mathbf{x} = [\theta_1 \ \mathbf{q}_1^T \ \theta_2 \ \mathbf{q}_2^T \ \dot{\theta}_1 \ \dot{\mathbf{q}}_1^T \ \dot{\theta}_2 \ \dot{\mathbf{q}}_2^T \ ]^T \tag{6}$$

and

$$\mathbf{F} = [V_s \ \mathbf{v}_1^T \ V_e \ \mathbf{v}_2^T \ ]^T \tag{7}$$

in which $\mathbf{x}$ is the state, $\mathbf{F}$ is the control vector, $\mathbf{q}_1$ and $\mathbf{q}_2$ represent the generalized coordinates for the elastic vibration of each arm, and $\mathbf{v}_1$ and $\mathbf{v}_2$ are the input voltage to the amplifiers of piezoceramic actuators and reaction wheels mounted on each beam, respectively.

The nonlinear sensor equations are

$$\mathbf{y} = \mathbf{g}(\mathbf{x}, \dot{\mathbf{x}}) \tag{8}$$

where $\mathbf{y}$ consists of the encoder outputs, output voltages of piezoceramic sensors and accelerometers.

The dynamic analysis of PACE will be performed using both commercial and in-house algorithms.

4. Control Design

The VAX workstation is equipped with integrated software for data acquisition and control, the Real-Time Monitor Control. Fortran or C subroutines can be incorporated into the real-time control of PACE. The control objectives of PACE are to drive the arms to a certain position with a specified line of sight in minimum time and to suppress vibrations simultaneously based on sensor measurements. The authors are performing a survey of possible control techniques such as neural network control, nonlinear control and adaptive control techniques that can be applied to PACE. The difficulty of finding a control algorithm for PACE is in the inherent nonlinearity of the system, including the structural, sensor and actuator dynamics.

5. Summary and Conclusions

The experimental apparatus for PACE has been described in detail. The current configuration enables us to verify the numerical results of modeling and control theories for flexible multibody systems experimentally. Presently, both sensors and actuators are still being mounted to the PACE structure, and system identifications of its components must continue to be performed. However, in the near future, PACE will be the ideal test bed for researchers studying the dynamics and control of flexible multibody systems.

References

1. Meirovitch, L. and Hale, A. L., "Synthesis and Dynamic Characteristics of Large Structures with Rotating Substructures," *Proceedings of the IUTAM Symposium on the Dynamics of Multi-Body Systems*, (Editor: K. Magnus), Springer-Verlag, Berlin, 1978, pp. 231–244.

2. Hollerbach, J. M., "A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-10, No. 11, 1980, pp. 730–736.

3. Huston, R. L., "Flexibility Effects in Multibody System Dynamics," *Mechanics Research Communications*, Vol. 7, No. 4, 1980, pp. 261–268.

4. Huston, R. L., "Multi-Body Dynamics Including the Effects of Flexibility and Compliance," *Computers and Structures*, Vol. 14, No. 5–6, 1981, pp. 443–451.

5. Book, W. J., "Recursive Lagrangian Dynamics of Flexible Manipulator Arms," *The International Journal of Robotics Research*, Vol. 3, No. 3, 1984, pp. 87–101.

6. Yoo, W. S. and Haug, E. J., "Dynamics of Articulated Structures. Part I. Theory," *Journal of Structural Mechanics*, Vol. 14, No. 1, 1986, pp. 105–126.

7. Yoo, W. S. and Haug, E. J., "Dynamics of Articulated Structures. Part II. Computer Implementation and Applications," *Journal of Structural Mechanics*, Vol. 14, No. 2, 1986, pp. 177–189.

8. Changizi, K. and Shabana, A. A., "A Recursive Formulation for the Dynamic Analysis of Open Loop Deformable Multibody Systems," *Journal of Applied Mechanics*, Vol. 55, 1988, pp. 687–693.

9. Shabana, A. A., "On the Use of the Finite Element Method and Classical Approximation Techniques in the Non-linear Dynamics of Multibody Systems," *International Journal of Non-Linear Mechanics*, Vol. 25, No. 2, 1990, pp. 153–162.

10. Meirovitch, L. and Kwak, M. K., "Rayleigh-Ritz Based Substructure Synthesis for Flexible Multibody Systems," *AIAA Journal*, Vol. 29, No. 10, Oct. 1991, pp. 1709–1719.

11. Meirovitch, L. Stemple, T. and Kwak, M. K., "Dynamics and Control of Flexible Space Robots," presented at the *Symposium on Mechanical Systems with Time-Varying Topology*, CSME Mechanical Engineering Forum 1990, June 3-9, 1990, Toronto, Canada.

12. Kwak, M. K. and Meirovitch, L. "A New Approach to the Maneuvering and Control of Flexible Multibody Systems," *Journal of Guidance, Control and Dynamics* (to appear).

	Shoulder Motor	Unit
n	60	
I_{sm}	1.043×10^{-4}	$kg\ m^2$
C_{sv}	0.829	$N\ m/rad/s$
C_{sc}	1.097	$N\ m$
K_{sT}	0.084	$N\ m/A$
α_s	1.7189	A/V
β_s	0.1590	A
T^*_{sf}	0.030	$N\ m$

Table 1. Data for the Shoulder Motor (PMI S9M4II)

	Elbow Motor	Unit
n	60	
I_{em}	0.278×10^{-4}	$kg\ m^2$
C_{ev}	0.073	$N\ m/rad/s$
C_{ec}	0.423	$N\ m$
K_{eT}	0.030	$N\ m/A$
α_e	1.5479	A/V
β_e	0.0289	A
T^*_{ef}	0.012	$N\ m$

Table 2. Data for the Elbow Motor (PMI S6M4H)

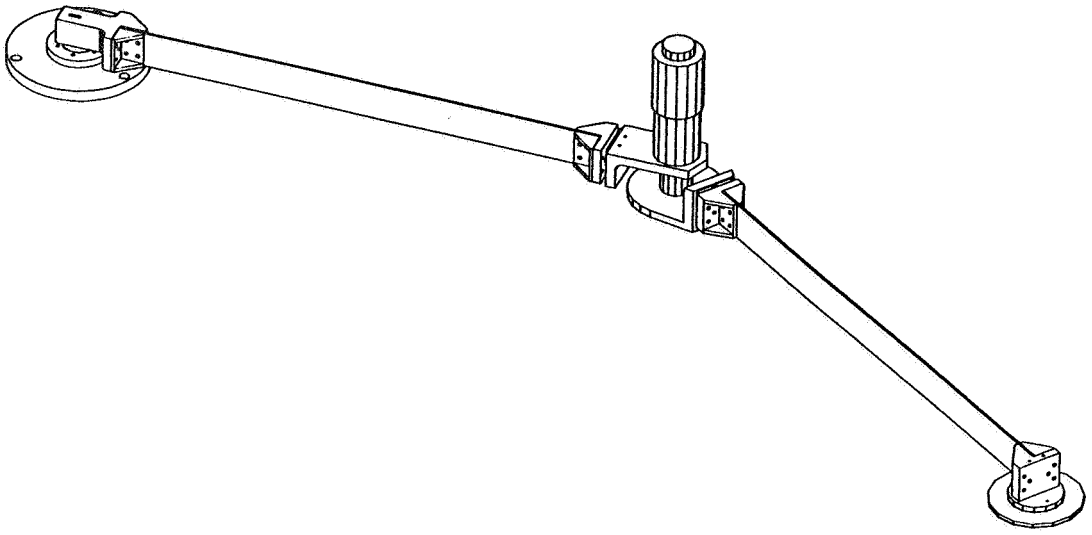


Fig. 1. The Planar Articulating Controls Experiment (PACE) Test Article

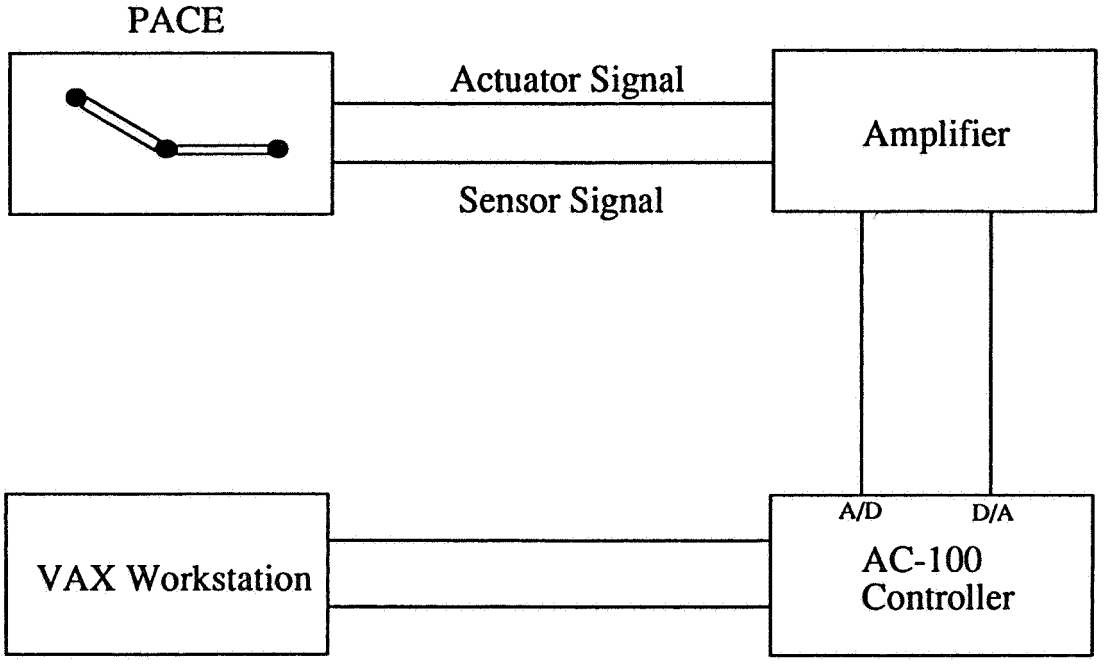


Fig. 2. The Control Loop for the PACE

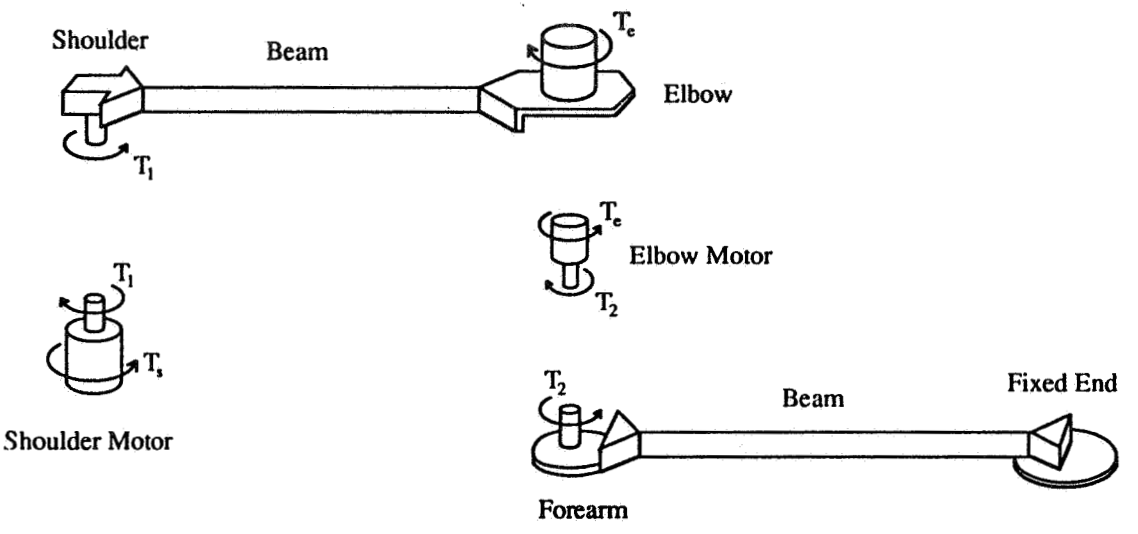


Fig. 3. The Free Body Diagram for the PACE

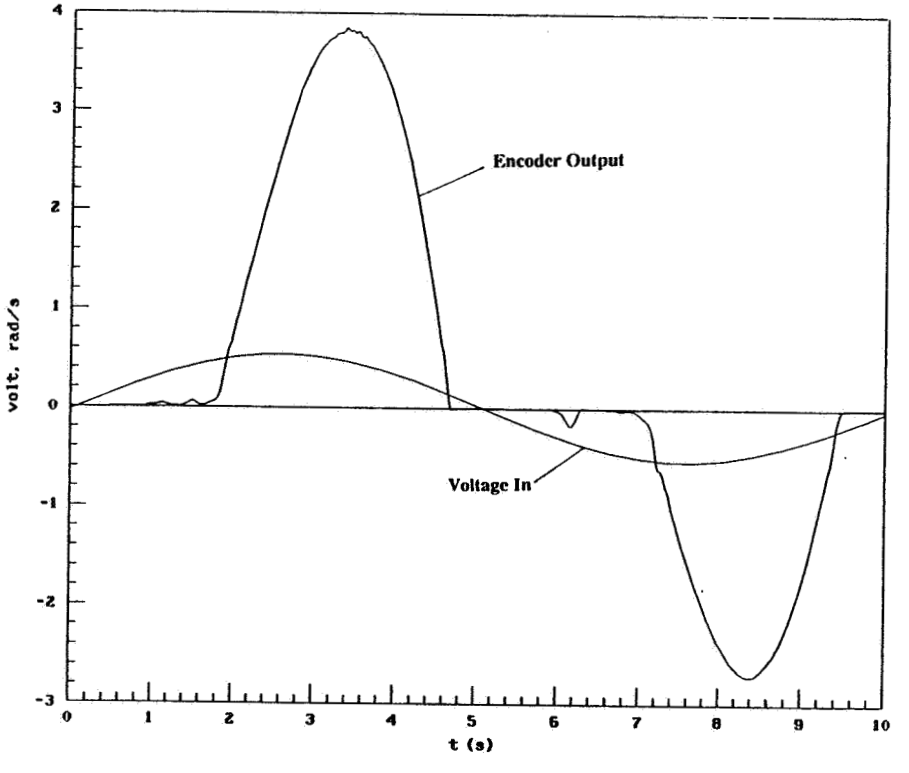


Fig. 4. The Sine-Sweep Test of the Elbow Motor

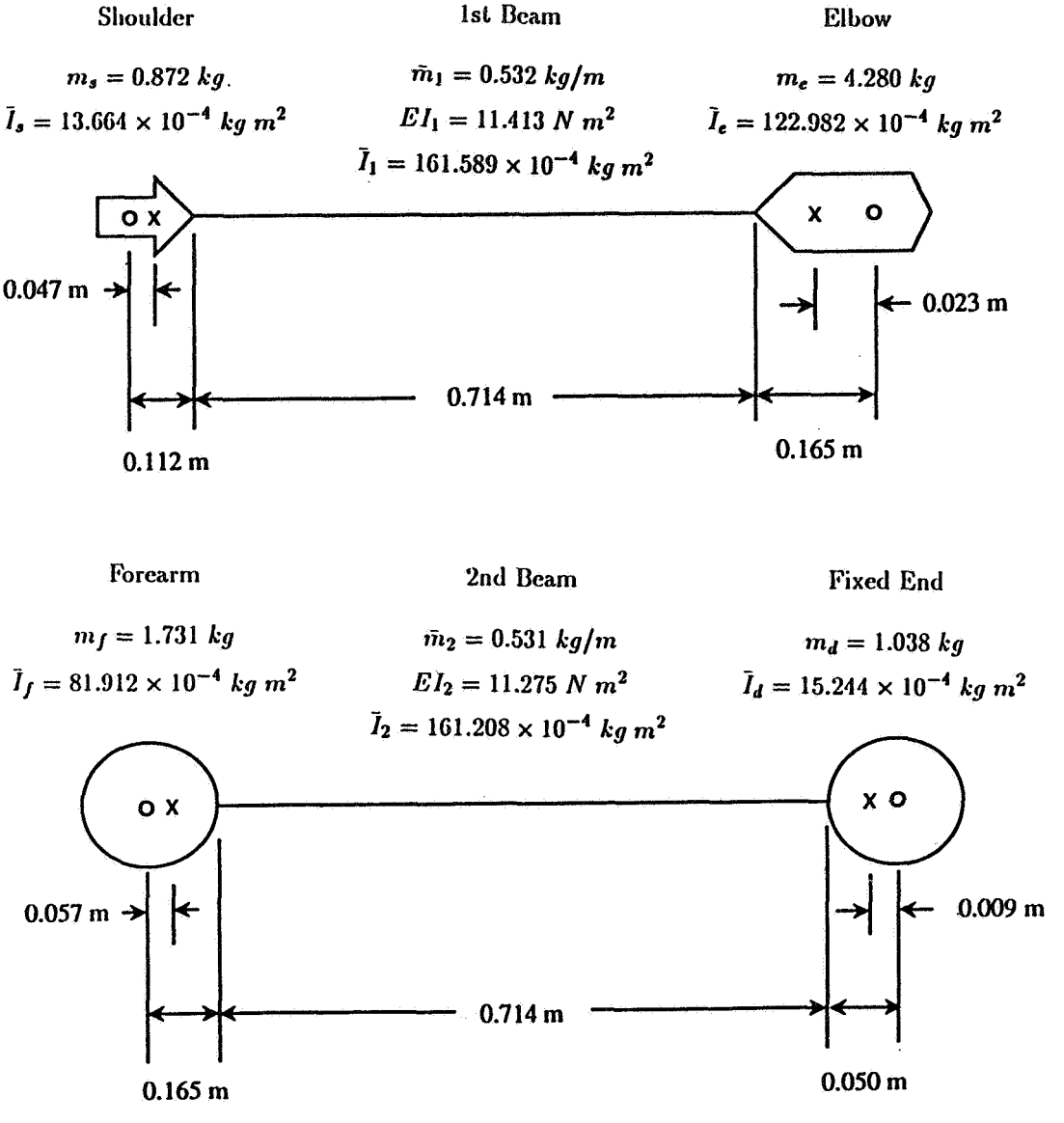


Fig. 5. The Material Properties for the PACE

Design and Control of a Macro-Micro Robot for Precise Force Applications

Yulun Wang, Amante Mangaser, Keith Laby, Steve Jordan, Jeff Wilson

Computer Motion, Inc.
Goleta, CA. 93117

Abstract

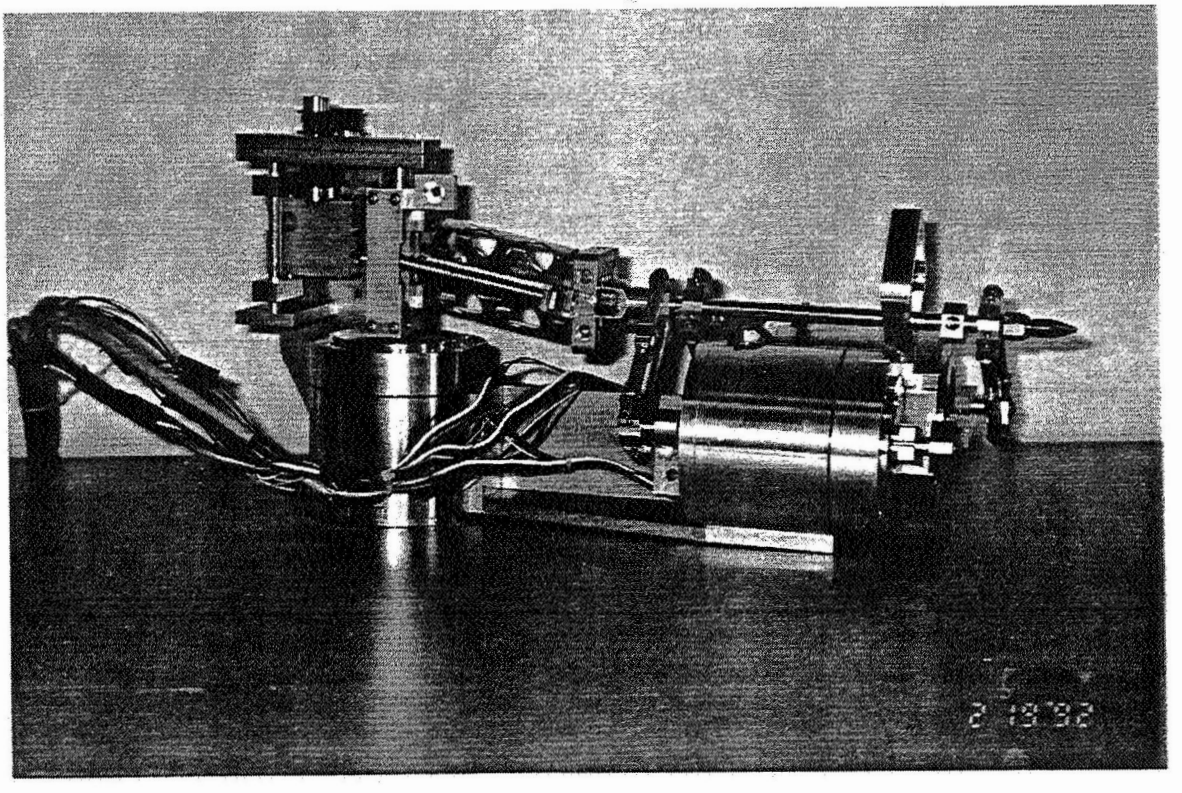
Creating a robot which can delicately interact with its environment has been the goal of much research. Primarily two difficulties have made this goal hard to attain. The execution of control strategies which enable precise force manipulations are difficult to implement in real time because such algorithms have been too computationally complex for available controllers. Also, a robot mechanism which can quickly and precisely execute a force command is difficult to design. Actuation joints must be sufficiently stiff, frictionless, and lightweight so that desired torques can be accurately applied.

This paper describes a robotic system which is capable of delicate manipulations. A modular high-performance multiprocessor control system was designed to provide sufficient compute power for executing advanced control methods. An 8 degree of freedom macro-micro mechanism was constructed to enable accurate tip forces. Control algorithms based on the impedance control method were derived, coded, and load balanced for maximum execution speed on the multiprocessor system. Delicate force tasks such as polishing, finishing, cleaning, and deburring, are the target applications of the robot.

1.0 A Force Controllable Manipulator

A manipulator capable of delicate interactions with its environment must be designed differently from today's position controlled robots. It has been shown that a high-bandwidth, low effective end-effector inertia design is helpful for precise force control [1,2]. There are two design approaches to creating such a structure. One is to design the manipulator so that the entire structure is very light. This approach can be very costly since expensive materials and tight tolerances are required. The other approach is to attach a low-inertia small manipulator to the end of another larger and heavier manipulator. This macro-micro structure results in a combined structure with the low end-effector inertia of the micro robot and the large workspace of the macro robot.

Our macro-micro design couples a 3 degree of freedom micro robot to the end of a 5 degree of freedom macro robot. A schematic and photograph of the micro design is shown in Figure 1, and a schematic of the macro design is shown in Figure 2. For the micro robot, the x and y directions are actuated with a parallel set of 5-bar-link mechanisms, one attached to each side end of the two motor shafts. The z motion is actuated by a fixed motor oriented perpendicular to the x and y motors. This motor is attached to the parallel link mechanism through a pair of universal joints. The range of motion is 2 centimeters along each axis. A fourth pneumatic motor, located furthest from the tip, rotates the tip through a series of transmissions at a constant speed for polishing, finishing, and grinding applications.



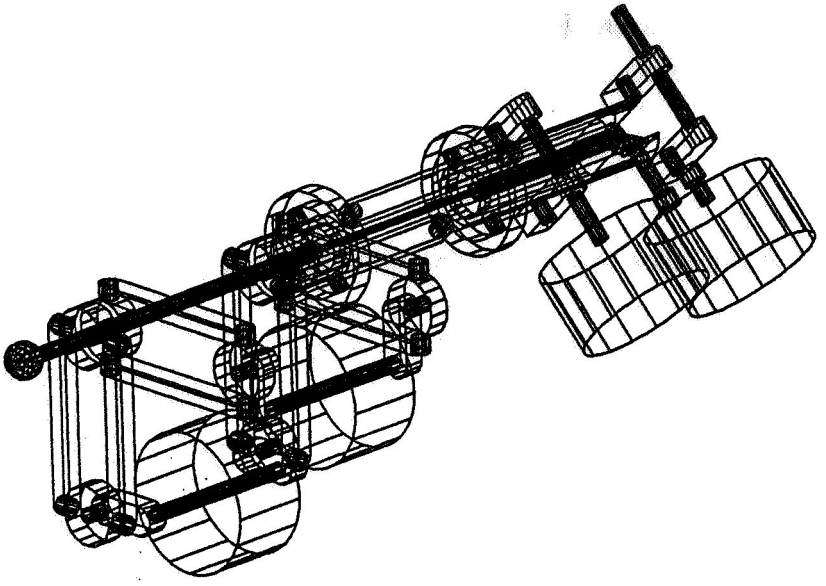


Figure 1. The Micro Manipulator

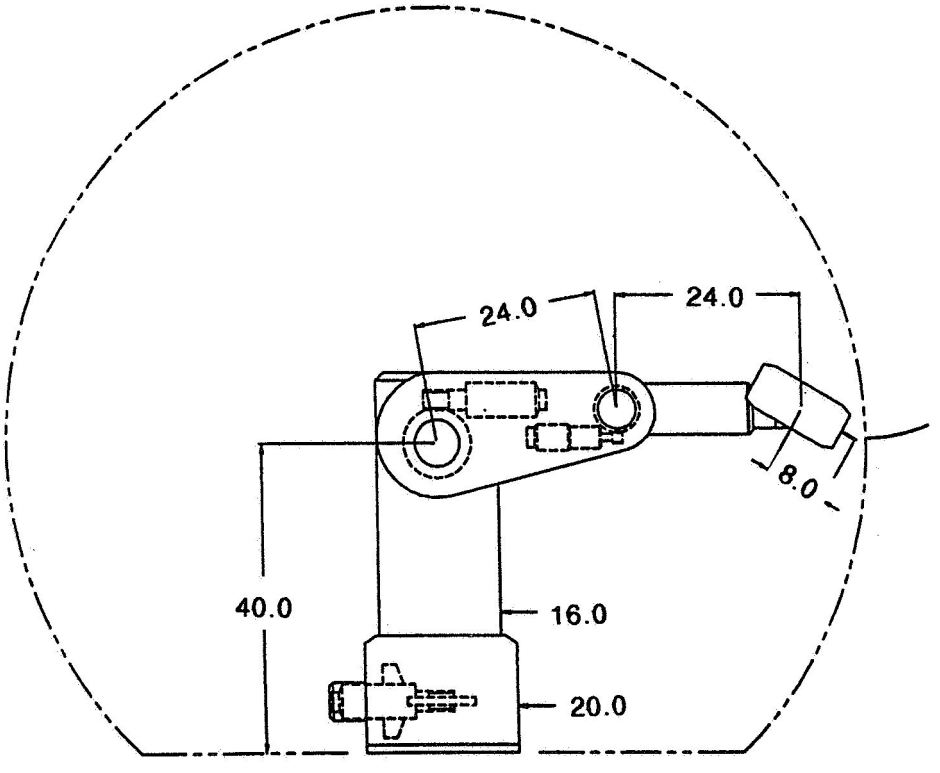


Figure 2. The Macro Manipulator

Since the macro and micro robots coordinate as a single system, many tradeoffs influence both designs. For example, the size of the micro robot's workspace influences the accuracy with which the macro robot must be able to position itself. The mass of the micro robot also influences the payload capability of the macro design. Our design strategy was to simplify the macro design by making the micro robot more capable. The main consequence of this decision is a large micro workspace, thereby allowing less accuracy and performance in the macro. However, the micro's workspace volume directly influences the overall mass and size of the design considerably. In our design, reducing travel along each dimension by a factor of two roughly reduces the size and mass of the micro robot by a factor of two.

The main objectives of the micro design were to minimize end-effector inertia, minimize joint friction, maintain tip orientation throughout the workspace, and support a maximum payload (i.e. force exertion) of 3 kilograms. The resulting tip inertia is roughly 250 gms. The joint friction was minimized by using direct-drive transmission and limited angle flex bearings at the joints. These limited-angle bearings offer virtually no friction. They do generate a spring force, however, which must be compensated for in the control law. Tip orientation is maintained by the parallel 5 bar link structure.

Secondary goals were to minimize the size and weight of the micro-manipulator. The final size is 35.5 by 19 by 17.8 centimeters, and the weight is 6.3 kilograms. Strain gages mounted on the links provide sensing for 5 degrees of freedom (as shown in Figure 1). Sensors for detecting a moment about the tip axis were not included.

1.1 Micro kinematics

The micro robot is a closed-chain kinematic structure with 3 servo actuators to produce translational motion in 3 space while maintaining constant orientation. To derive the kinematic equations we only need to consider the tip's position. This positioning mechanism can be thought of as three links which are all connected by ball joints at one end. The other ends of each link are connected to separate actuators, each through a universal joint. Since the joint angle of the actuators are always known, the end position of each link is known, and will be referred to as P1, P2, and P3 as shown in Figure 3.

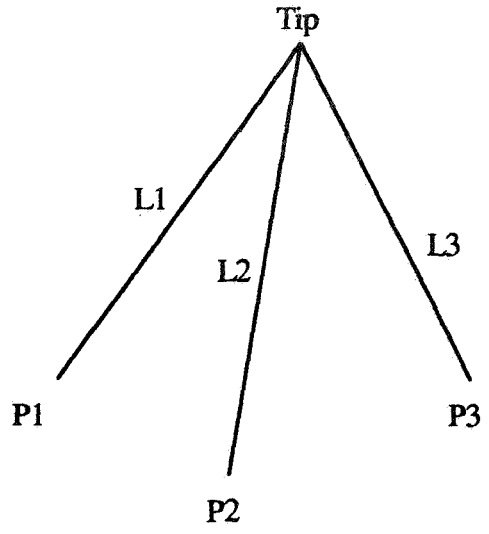


Figure 3. Kinematic Model of Micro Robot

Three simple geometric relations which can be observed are as follows:

$$| \text{Tip} - \text{P1} | = \text{L1}$$
$$| \text{Tip} - \text{P2} | = \text{L2}$$
$$| \text{Tip} - \text{P3} | = \text{L3}$$

where L1, L2, and L3 are the lengths of the 3 links. The forward kinematics problem is to solve for the Tip position given L1, L2, L3, P1, P2, and P3. P1, P2, and P3 can easily be related back to the joint angles. This solution may be visualized geometrically by imagining three spheres with radius L1, L2, and L3 centered at P1, P2, and P3 respectively. The point where all three spheres intersect is the robot's tip position.

Solving these equations for the Tip position is possible. However, the solution is a very large and complex high order polynomial. Since it is important that the micro robot is servoed at a high rate, it was necessary to develop a more computationally efficient approximate solution to the forward kinematics.

If we assume that the links rarely rotate beyond 10 degrees from the center position, we can derive a fairly accurate and simple forward kinematics solution. The angles α and β, shown in Figure 4, describe the orientation of each link and can be used to approximate the manipulator's tip position with the following equations:

$$\text{Tip}_x = A_1 \sin\theta_1 - \text{L1} \left[(1 - \cos\alpha_1) + (1 - \cos\beta_1) \right]$$
$$\text{Tip}_y = A_2 \sin\theta_2 - \text{L1} \left[(1 - \cos\alpha_2) + (1 - \cos\beta_2) \right]$$
$$\text{Tip}_z = A_3 \sin\theta_3 - \text{L1} \left[(1 - \cos\alpha_3) + (1 - \cos\beta_3) \right]$$

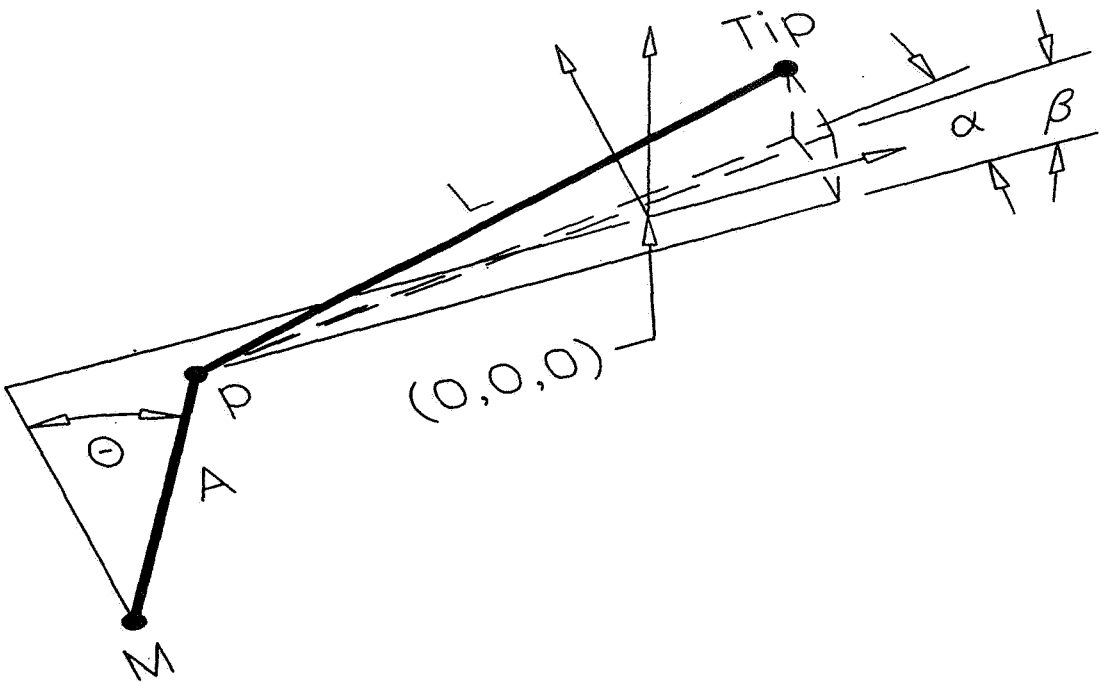


Figure 4. Geometric quantities for one branch of the micro robot

These equations are simple enough to be computed quickly so that high speed servoing can be achieved. This solution is very accurate at the center of the workspace, and error increases towards the edge of the workspace. The influence of this approximation error on the impedance control law is quantified in section 3.0.

1.2 Micro dynamics

In order to calculate the closed-form dynamic equations of the micro mechanism, we assumed a rigid body model which includes 3 independent motor inertias, and a lumped mass at the tip with different masses along each cartesian axis. Since the micro robot is carried about by the Macro robot, it is important to include the macro's motion into the dynamic equations. By using Lagrange's equation we derived the following equations of motion:

$$\tau = \dddot{\theta}_I^T \left(J_I^T M_T J_I + I_M \right) + \ddot{\theta}_I^T \left(\dot{J}_I^T + 2 J_I^T \dot{R}^T R \right) M J_I$$
$$+ \vec{X}_I^T \left(K_F + \left(\ddot{R}^T R - 2 \dot{R}^T \dot{R} \right) M_T \right) J_I + \vec{g}^T R (M_T + M_z) J_I + \vec{F}_x^T + \ddot{\vec{X}}_A^T R M_T J_I$$

where

$\vec{\theta}_I^T$ = joint angles of Micro robot

J_I^T = jacobian of Micro robot

M_T = tip mass of Micro robot

R = rotational matrix from Micro coordinate system to Macro coordinate system

$\vec{F}_x^T$ = force exerted at tip of Micro

K_F = force created by flex bearings in Micro joints

$\vec{X}_A$ = tip position of Macro robot

$\vec{X}_I$ = tip position of Micro robot

M_z = added mass felt by z-axis motor

The macro design is a 5 degree-of-freedom articulated manipulator, as shown in Figure 2. This manipulator supports the weight and continuous force exertion capability of the micro-manipulator throughout the workspace with 1g acceleration. A 1 meter reach was chosen as a reasonable workspace. The main features of this design are high mechanical rigidity, simple kinematics, large workspace volume, and cost effectiveness.

The 5 degree of freedom kinematic structure is very similar to the first five joints of a PUMA robot [3]. A 6th joint is unnecessary because the tip of the micro robot spins continuously. Link offsets, link lengths, and structural characteristics were designed to account for the size and mass constraints imposed by the micro-manipulator, however.

We considered a variety of actuation methods. Options which were considered were direct-drive, harmonic drive, spur gear, worm gear, planetary gear, and different combinations of these. The goal was to maximize accuracy, resolution, and stiffness while staying cost effective. After various optimization procedures we decided on a harmonic drive - worm gear double reduction scheme for the first three joints. The last two joints, which carry a much smaller load, use harmonic drives.

The procedure for solving for the inverse kinematics equations of this robot is very similar to that of the PUMA robot and can be found in many of different robotics textbooks [4]. The kinematics and dynamic equations used for computed torque control can also be derived very easily using of the generalized formulations which have been developed [5]. However, because of the high reduction ratios of the transmissions, independent joint control is adequate.

2.0 A Modular Multi-processor Control System

A high performance multiprocessor system is used to satisfy the significant computational demands of controlling this robot. We designed this control system as a general purpose high performance controller with both hardware and software modularity as a key feature. The ability to easily rearrange and extend hardware and software modules to support different requirements for various tasks is particularly important in experimental projects such as this. Frequently designs are unable to accomodate even minor modifications without a major impact to the existing system configuration.

A schematic of the motion control system configuration is shown in Figure 5. The four basic units are the compute unit, the global memory unit, the position, velocity and digital I/O unit, and the A-to-D D-to-A unit.

The compute unit is based on Texas Instrument's TMS320C31 floating-point digital signal processor. In our earlier generation systems [6,7], we used a novel 3D computing processor which proved to offer much higher performance than DSPs or RISC processors on kinematic and dynamic calculations. However, due to the high cost of implementing this design using discrete datapath parts we opted to used an off-the-shelf processor. At a crystal speed of 33Mhz the TMS320C31 offers 33 MFLOPS of peak power. Each unit contains 2 Mbyte of program memory, 2 Mbyte of data memory, 2 programmable timers, interrupt capabilities for both the I/O Bus and the VME bus, and bus arbitration logic for accessing the I/O Bus. The memory is directly accessible by the host computer over the VME bus. Different levels of concurrency are provided to maximize execution speed. For example, the host may access data memory while the processor continues program execution. Programs are developed in either C or C++ on the host computer and downloaded to the appropriate unit before run time. Several libraries are provided to support program development. Remote procedure calls were provided so that UNIX services, such as printf(), scanf(), open(), and close(), are available for code development. Math functions, functions for accessing sensory data, and message passing functions for multi-processing are also provided.

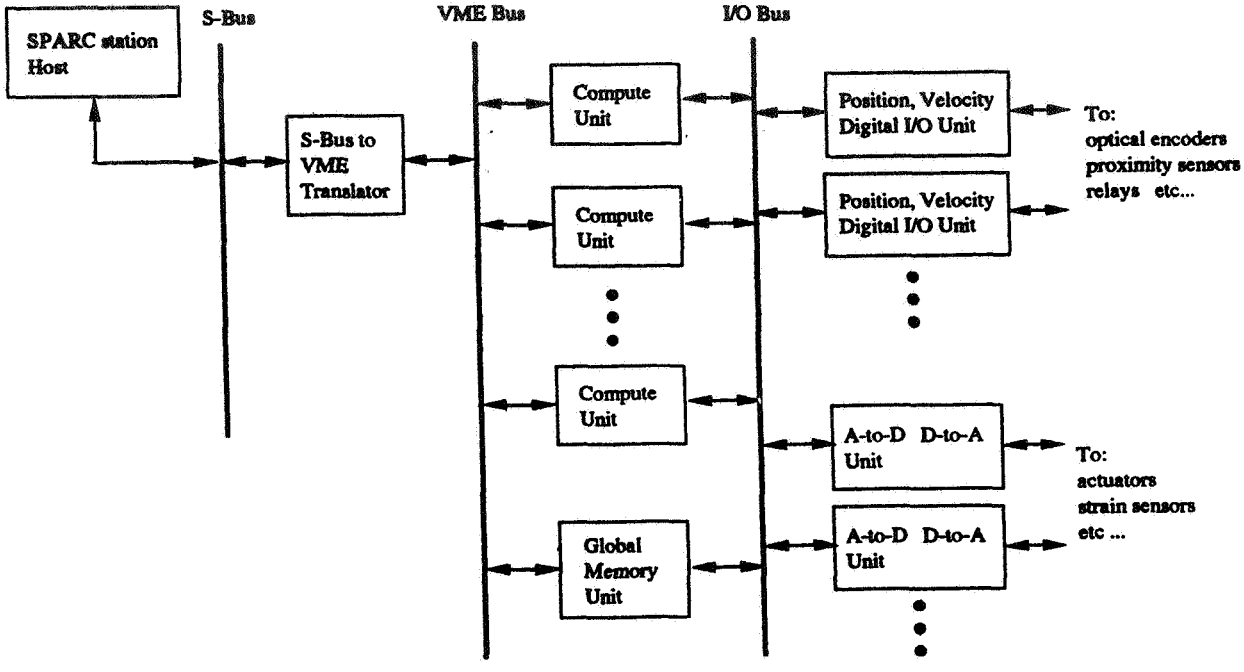


Figure 5. The Motion Control System

The global memory unit contains 2 Mbytes of memory for passing messages between compute units, to and from the host, and to store global variables shared by multiple compute units. A mailbox message passing scheme is implemented to support multiprocessor communication. Information is passed from one compute unit to another compute by first acquiring the IO Bus, then writing the message into the target compute unit's mailbox, and then interrupting the target compute unit. The target compute unit reads its mailbox, and sends an acknowledgement to the sending compute unit. Hardware interlocking and interrupt mechanisms are included to achieve high bandwidth communication. Reading or writing a message requires ~3 μs overhead and another 180ns for each 32-bit word.

The position, velocity, and digital I/O unit accepts 6 channels of 2 channel quadrature encoder input and translates that into absolute position and velocity. Each channel also supports index pulse detection, which is generally used for position homing. Position is stored to 24-bit accuracy and velocity is stored to 10-bit accuracy. Thirty-two bits of digital input and 32 bits of digital output are included for instrumenting relays, proximity sensors, or other on-off type devices.

Velocity is generated by two different schemes, depending on the velocity range. At low speeds, velocity is generated in hardware by a free running counter which measures time between successive encoder counts. At high

speeds, velocity is determined by calculating the number of encoder counts which have passed during the previous sample period. For each velocity read operation, the software automatically chooses between the two schemes by reading the velocity counter and comparing it with a threshold value. The result of this method is a more accurate velocity signal with minimized quantization effects.

Velocity is generated in hardware from the optical encoder signal by incorporating a free running counter chip which calculates the time between successive encoder pulses. Velocity is usually derived from a quadrature signal by subtracting the current position with the previous sample period's position. This subtraction may result in very quantized velocity signals especially at high sample rates, however. The hardware counter method produces a much more finely resolved velocity signal. There is still a problem, however, since at low speeds there may be significant time delay between new velocity acquisitions.

The A-to-D D-to-A unit provides 9 channels of 12-bit digital-to-analog output, and 8 channels of 12-bit analog-to-digital input. Separate digital to analog converters are provided for each output channel. A single analog-to-digital converter is multiplexed between the 8 input channels. Each channel requires 3 μs of conversion time. Software routines are provided to configure the card to only sample the channels which are in use. Conversion is performed continuously and asynchronously only on the channels being used. Therefore, the maximum delay from when the data was acquired to when it was read is 3 μs $\times$ number of selected channels.

The software structure of the operating system level software is shown in Figure 6. Note that there is a clear separation between the real-time execution environment and the non-real-time UNIX environment. The UNIX environment is used for program development, user interface, and monitoring the real-time system. Because of the UNIX front-end, the robot interface must be carefully constructed such that the integrity of the real-time system is not lost. For example, UNIX service requests by the real-time system cannot be made while servoing since a real-time response from the UNIX process cannot be guaranteed.

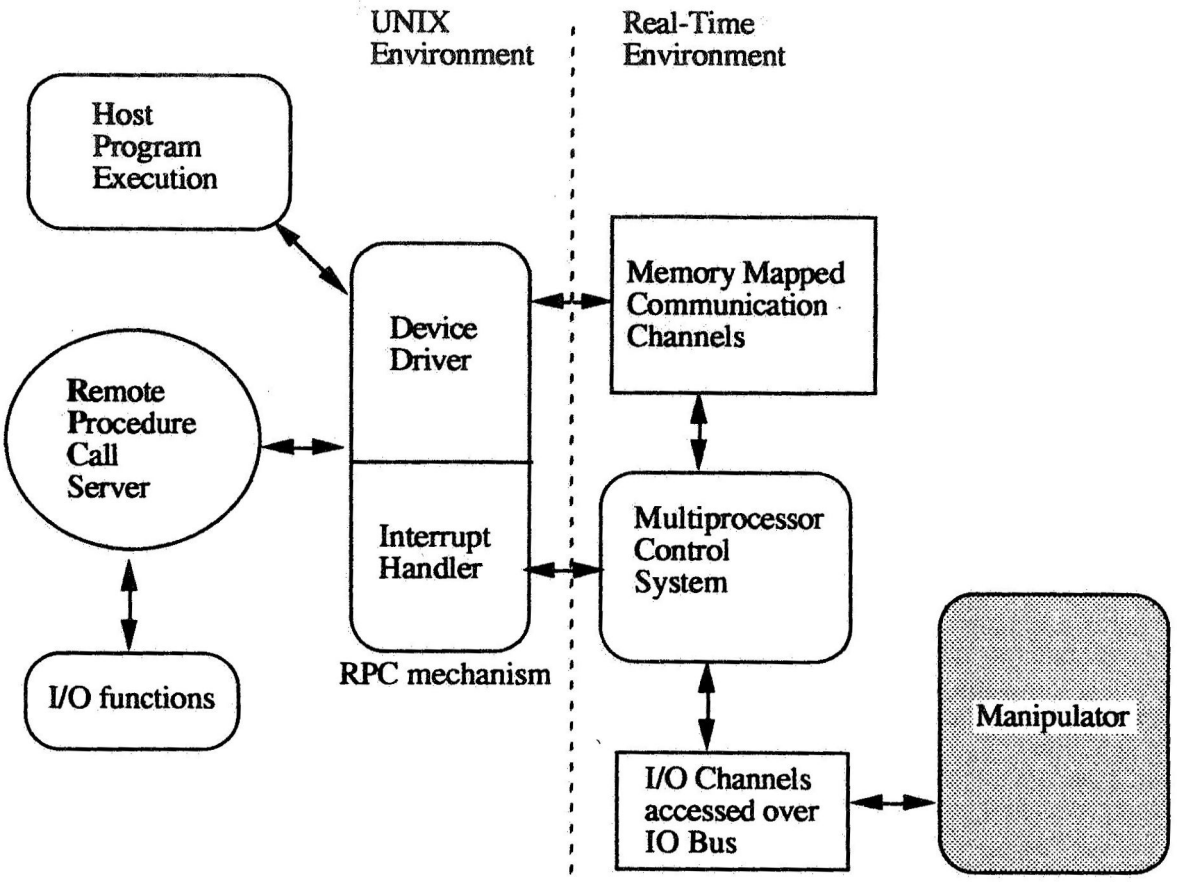


Figure 6. System Software Structure

Figure 7 shows the general hierarchy of the application software of the system. Macro calls provide fast access to the various hardware features of the system. C language routines provide the next layer, which support functions such as synchronizing multiple processes, remote procedure calls to the host, and algorithms for performing mathematic operations. At the highest level, object-oriented class libraries are supported in C++.

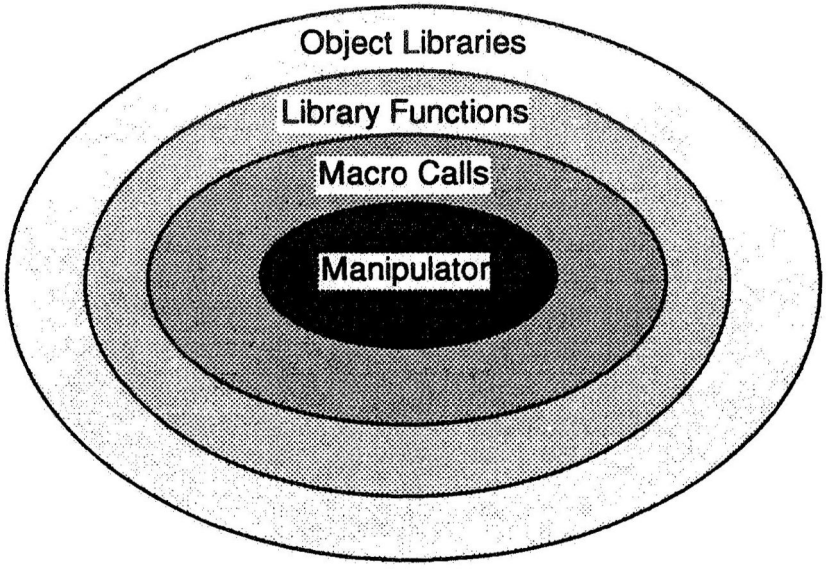


Figure 7. Software Support for Application Development

3.0 Impedance Control for a Macro-Micro Robot

The impedance control method enables a robot to interact with its environment in a well controlled and precise manner [8]. The manipulator's end-effector reacts to environmental disturbances in the same manner as a linear mass, spring, damper system. The mass, spring, and damper values are controlled electronically and can be different along different axes, and can continuously change during a trajectory.

This method is different from hybrid position/force control [9] since specific forces or positions are never specified. The control variable is the equilibrium point of the mass, spring, damper system without external forces. The advantage of this methodology is that a single control variable and control algorithm can be used to guide a robot through interactions with the environment. Hybrid position/force control, on the other hand, requires a switch in control methods and control variables whenever the robot changes the configuration in which it interacts with its environment.

Figure 8 gives an example of a trajectory specified by the equilibrium path where the manipulator comes into contact with a surface, slides across it, and then leaves the surface. Note that the nominal force exerted on the surface is proportional to the spring constant. By using the spring constant and surface location information, it is simple to calculate the equilibrium point's trajectory to produce a desired force across the surface. The force at the contact point will be influenced by contributions due to the mass and damper as well. Consequently, if precise force control is important, the smaller the mass and damper values are the better. The macro-micro design facilitates small mass values.

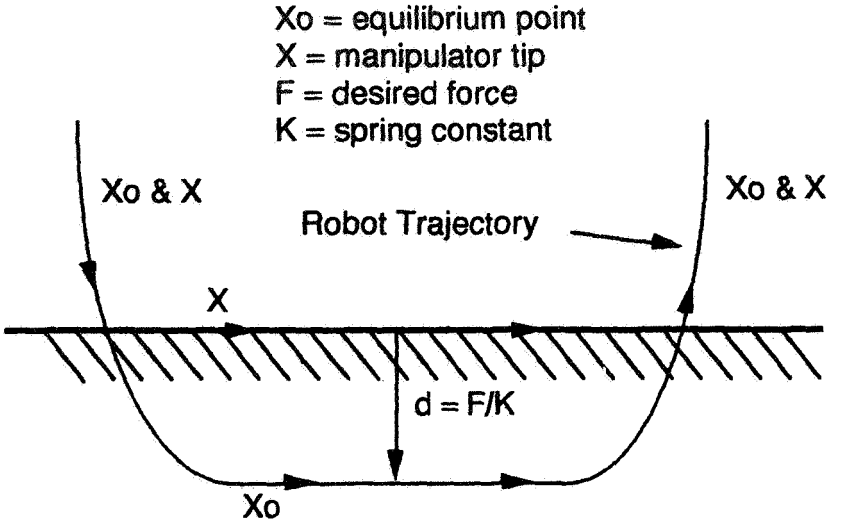


Figure 8. Manipulator trajectory specified by equilibrium point

The impedance equation can be written as follows:

$$F_{ext} = M_s (\ddot{X}_R - \ddot{X}_o) + C_s (\dot{X}_R - \dot{X}_o) + K_s (X_R - X_o)$$

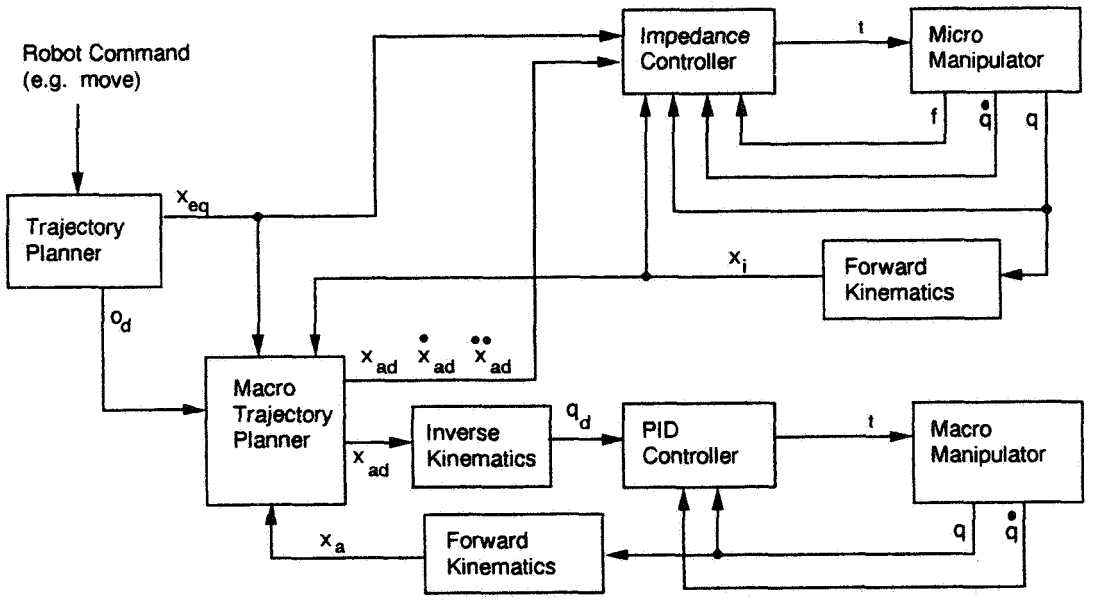
where

F_{ext}	external force applied to robot tip
X_R	tip position of macro-micro robot
X_o	desired equilibrium point of macro-micro robot
M_s	desired mass constant
C_s	desired damper constant
K_s	desired spring constant

Impedance control of a macro-micro design has the added complexity of managing the manipulator's redundancy to optimize force interactions by exploiting the micro robot's low tip inertia. In other words, the redundancy should be used to keep the micro robot from reaching its workspace limit, where one or more degrees of freedom would be lost. Our robot has 3 degrees of redundancy along the translational axes. Delicate interactions for translational motion is possible because of the micro robot. Orientation is left to the macro robot and is position controlled.

A block diagram of the control structure is shown in Figure 9. The impedance control law, which outputs torques to the micro robot, is derived by combining the desired impedance equation stated above with the equations of motion of

the micro robot presented in section 1.2. Note that the servo control law for all 5 joints of the macro robot is a simple position controller without feedback from the micro robot. However, feedback from the micro robot is input into a real-time trajectory generator for the macro robot. This trajectory generator uses the robot's redundant degrees of freedom by constantly updating the macro robot's desired position such that the micro robot is centered in its workspace, and hence far from its workspace boundary. Consequently, entire manipulator can respond to external disturbances with the quick reaction of the micro robot over the entire workspace of the macro robot.



x_i - micro robot's actual tip position

f - micro robot's force sensor readings

x_a - macro robot's actual tip position

o_d - desired orientation of macro-micro robot

x_{ad} $\dot{x}_{ad}$ $\ddot{x}_{ad}$ - desired tip position, velocity, and acceleration of macro robot

q_d - desired joint position

x_{eq} - equilibrium point

q - actual joint position

$\dot{q}$ - actual joint velocity

t - torque

Figure 9. Impedance control of macro-micro robot

The maximum distance which the micro will deviate from its center position is a relationship which includes the ratio of the maximum accelerations of the macro and micro, the magnitude and time of the maximum disturbance, and the reaction time of the servoing system. This information is important since it quantifies the critical tradeoffs between the micro's performance versus the macro's performance. We will obtain more insight into these relationships through experimentation of the robot.

With this control strategy, since the macro robot is purely position controlled it may be possible to apply this strategy to a micro connected onto the end of a commercial robot. However, the success of this approach is dependant upon the ability of the commercial robot to accept and quickly respond to new position commands. The requirements of a commercial robot used in this manner will become clearer with more experimentation on our robot.

In order to quantify the errors in the impedance control law resulting from the approximations used in section 1.1 we evaluated the control law at several different robot configurations and compared the resulting torques with torques calculated using exact equations. Nominal values of desired mass, spring, damper were chosen, and the micro's workspace center was used as the equilibrium point. To represent the results in a more intuitive framework we multiplied the final joint torques with $(J^T)^{-1}$ to produce a single force vector. The errors are quantified by the percentage different in magnitude, and the orientation different in degrees, from the force vector produced by the exact method. Figure 10a confirms that error is small at the workspace center and increases as we move towards the edge. Figure 10b shows the resulting error at different configurations and includes non-zero velocity terms. Note that in all of these cases the error in the force vector never exceeds 2 percent in magnitude and 1 degree in orientation.

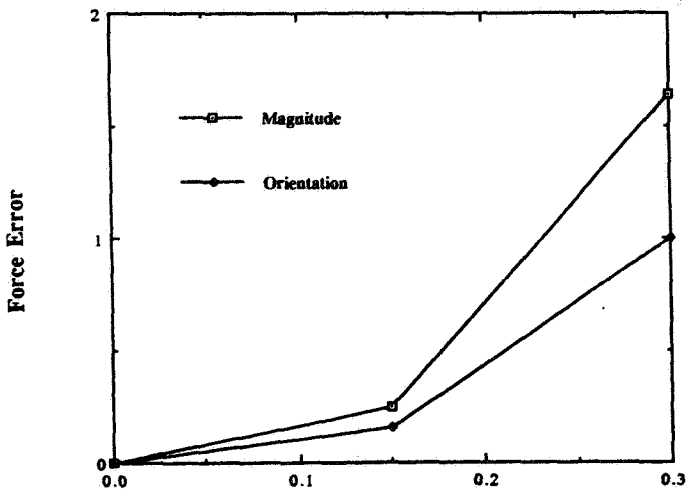


Figure 10a. Force error w.r.t. Tip offset along X-axis (velocity = 0.0, $F_x = F_y = F_z = 4$ lbs)

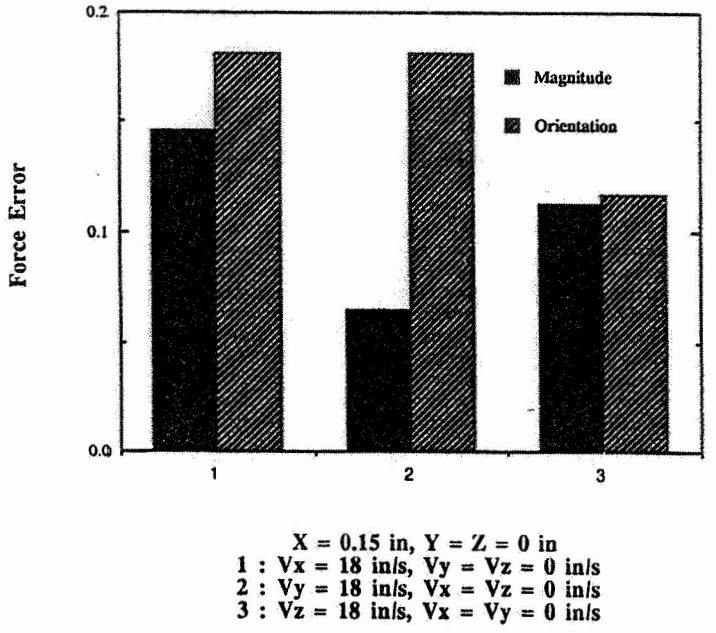


Figure 10b. Force error with non-zero tip velocity

4.0 Control Implementation on a Multiprocessor controller

The hardware control system which we assembled for executing the impedance control method includes 4 compute units, 2 position velocity units, 2 analog and digital units, and one global memory unit. One of the compute units is used for a trajectory planner. The execution of the control algorithms are mapped across the other 3 compute units to optimize the response of the system. Even though there is a great deal of research on automating the processing partitioning a task for parallel processing , our approach has been to manually partition the problem optimizing for balanced loads and meeting the various real-time constraints.

In addition to parallel processing with multiple compute units, within compute units we nested algorithms so that variables which changed faster were evaluated more frequently than variables which changed slower. For example, calculations which include force readings are grouped together and calculated at the highest update rate, and calculations which change only with position are grouped together and calculated at a slower update rate. Figure 11 shows the update hierarchy employed in our scheme.

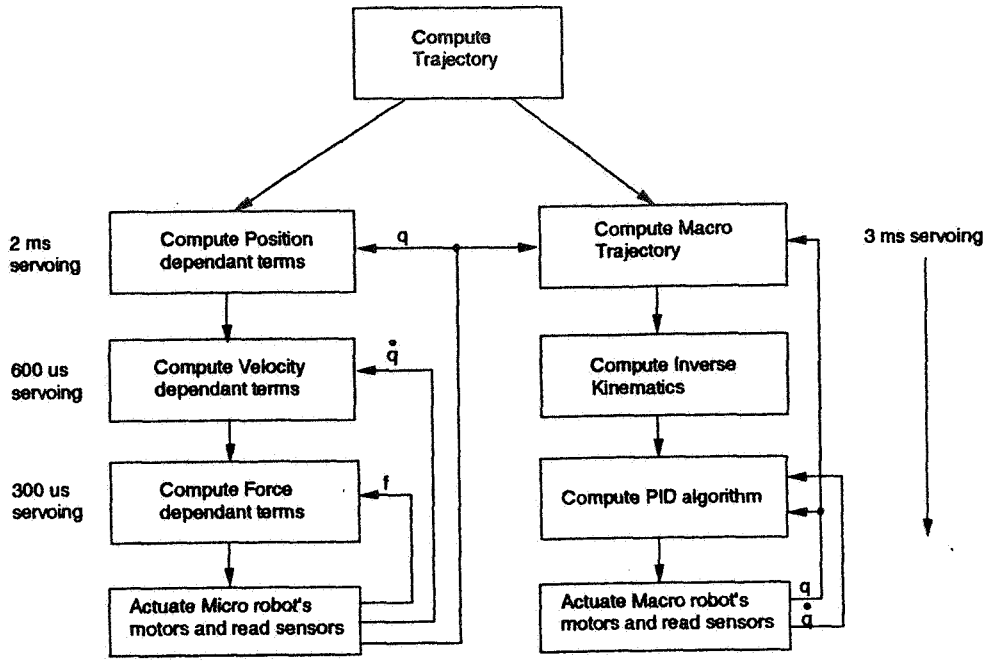


Figure 11. Hierarchical update rates

5.0 Conclusion

Building a robot which can interact delicately with its environment is a challenging task. This paper describes a robotic system designed for such tasks. An 8 degree of freedom macro-micro manipulator is controlled by an impedance-based controller, executed on a high performance multiprocessor control system. The manipulator's tip inertia is very low and can therefore react quickly to force disturbances. The control method compensates for manipulator dynamics, and can generate very precise torques. The multiprocessor offers sufficient compute power to meet the real-time demands of the control strategy. Preliminary results show that this design will be capable of precise force control. More conclusive experimental results will be available by the end of the year.

Acknowledgements

The authors would like to thank Kevin Schantz for his assistance in implementing and debugging the control software. We would also like to thank Professor Yoshihiko Nakamura for his guidance in developing the control algorithm. This work was performed under funding from the Small Business Innovative Research Program from both NASA and the National Science Foundation.

References

[1] Khatib, Oussama, "Augmented Object and Reduced Effective Inertia in Robot Systems," *Proc. of the American Control Conference*, Atlanta, Georgia, June 1988.

[2] Sharon, Andre, Neville Hogan, and David E. Hardt, "High Bandwidth Force Regulation and Inertia Reduction Using a Macro/Micro Manipulator System," *Proc. of the IEEE Conf. on Robotics and Automation*, Philadelphia, Penn., April 1988.

[3] Leahy, M.B. and et. al., "Efficient Dynamics for the PUMA-600," *Proc. of the IEEE Conf. on Robotics and Automation*, San Francisco, CA., 1986.

[4] Wolovich, William A., *Robotics: Basic Analysis and Design*, Holt, Rinehart and Winston, New York, 1987.

[5] Nakamura, Yoshihiko,"Unified Recursive Formulation of Kinematics and Dynamics of Robot Manipulators," *Proc. of the Japan - USA Symposium on Flexible Automation*, Osaka, Japan, July 14-18, 1986.

[6] Wang, Yulun, Amante Mangaser, Steven Butner, Partha Srinivasan, and Steve Jordan, "The 3DP: A Processor Architecture for 3-Dimensional Applications," *IEEE Computer*, January 1992.

[7] Wang, Yulun and Steven E. Butner, "RIPS: A Platform for Experimental Real-Time Sensory-based Robot Control", *IEEE Transactions on Systems, Man, and Cybernetics*, vol 19, no 4, July/August 1989, pp 853 - 860.

[8] Hogan, Neville, "Stable Execution of Contact Tasks Using Impedance Control," *Proc. of IEEE Int. Conf on Robotics and Automation*, Raleigh, North Carolina, 1987.

[9] Raibert, M. and J. Craig, "Hybrid Position/Force Control of Manipulators," *Journal of Dynamic Systems*, Measurement, and Control, vol. 102, pp. 126-133.

A PROGRAM FOR THE INVESTIGATION OF THE MULTIBODY MODELING, VERIFICATION, AND CONTROL LABORATORY

Patrick A. Tobbe
Logicon Control Dynamics
Huntsville, Alabama

Paul M. Christian
Dynacs Engineering Company, Inc.
Huntsville, Alabama

John M. Rakoczy and Marlon L. Bulter
Marshall Space Flight Center, Alabama

ABSTRACT

The Multibody Modeling, Verification, and Control (MMVC) Laboratory is under development at the NASA Marshall Space Flight Center in Huntsville, Alabama. The laboratory will provide a facility in which dynamic tests and analyses of multibody flexible structures representative of future space systems can be conducted. The purpose of the tests are to acquire dynamic measurements of the flexible structures undergoing large angle motions and use the data to validate the multibody modeling code, TREETOPS, developed under sponsorship of NASA. Advanced control systems design and system identification methodologies will also be implemented in the MMVC laboratory.

This paper describes the ground test facility, the real-time control system, and the experiments. A top-level description of the TREETOPS code is also included along with the validation plan for the MMVC program. Dynamic test results from component testing are also presented and discussed. A detailed discussion of the test articles, which manifest the properties of large flexible space structures, is included along with a discussion of the various candidate control methodologies to be applied in the laboratory.

INTRODUCTION

Approximate numerical methods are generally employed to solve the nonlinear partial differential equations for flexible multibody dynamics. The TREETOPS multibody modeling code is one such tool. This code uses Kane's equations and the component mode approach for multibody simulation. To date, verification of multibody tools have has been limited to the fixed point case, accomplished by comparing component and system mode results to those of the NASTRAN finite element code. Validation of the modeled nonlinear behavior can not be accomplished in this manner. Hardware experiments highlighting modeling features of interest, such as large angle slewing, are required for such validation. The Multibody Modeling, Verification, and Control (MMVC) Program at Marshall Space Flight Center (MSFC) is focused on the experimental validation of multibody modeling codes and the application of control theory to nonlinear dynamic systems.

The MMVC Program was initiated in November, 1990. The MMVC laboratory is currently under development and will provide a testbed for the execution of experiments designed specifically to validate modeling of complex systems. Modeling features under study are body flexibility, including large motions with small and large deformation; interface degree-of-freedom, including point and line interfaces undergoing translation and rotation; geometric stiffness, including gravity and foreshortening; and constraints, including prescribed motions and closed-tree topologies. The top-level design of a basic set of experiments that emphasize critical modeling features presently included in the TREETOPS simulation has been completed. Beginning with a simple single beam experiment and evolving to multiple beams, joints, and various topologies, the experiments will grow in complexity as each modeling feature is examined. The final experiment will feature a test article traceable to the Advanced X-Ray Astronomical Facility (AXAF). Figure 1 depicts the general methodology of the MMVC validation plan. Experiment hardware has been fabricated, and individual components have been tested. Detailed procedures for system-level experiments are being developed.

Critical to the experiments is the design and development of a test facility. A facility design was chosen such that an existing platform will be modified to accommodate the MMVC experiments. Additional structure will be added to the platform to provide a support base for the test articles and to raise the fundamental frequency of the platform such that it is outside the frequency range of interest for the experiments. The facility design has been finalized, and fabrication should be completed next year. An integral part of the facility is the real-time closed-loop system (RTCS). Its function is to process the sensor inputs, implement the controller, and provide the real-time output signals to the actuators. The RTCS is in place and functionally verified.

As part of the MMVC program, enhancements to the TREETOPS code are planned. The goal is to develop a Government-owned "all-in-one" tool that can be used to develop structural models of multibody systems, perform model order reduction, develop controllers, and assess controller performance in a closed-loop sense via simulation. Currently, the simulation tool is a menu-driven program used to model and analyze flexible multibody structures exhibiting either open- or closed-tree topologies. The menu program provides the means to implement gains for a standard proportional, integral, differential (PID) controller or to include a user-defined controller. The results of this effort will be the enhancement of TREETOPS to include model reduction techniques, thermal effects, optical path analysis capability, expanded controller design capability, and to improve computational efficiency.

The MMVC Program at MSFC will provide experimental validation of multibody simulations and lead to the development of a Government-owned multibody modeling and control system design and analysis tool. The results of the experiments and the enhanced TREETOPS code are and will be publicly available upon request to the Government. The following

sections contain brief descriptions of the TREETOPS code and planned enhancements, the MMVC experiments and validation plan, the MMVC facility, and highlights of the control design techniques envisioned for use in the closed-loop control experiments.

DESCRIPTION OF THE TREETOPS MODELING TOOL

Introduction

TREETOPS is a time history simulation of the motion of arbitrary complex multibody flexible structures with active control elements.[1] The name TREETOPS, which is not an acronym, refers to the class of structures whose motion can be simulated by the program, those having an open- or a closed-tree topology. The program offers the user an advanced capability for analyzing the dynamics and control-related issues of such structures.

In the simulation, the total structure is considered as an interconnected set of individual bodies, each described by its own modal characteristics with prescribed boundary conditions. An interactive set-up program creates all necessary data files. A linearization option that provides both the simplified model typically used during the initial phases of control system design and the complex model needed for final verification is also available. Thus, TREETOPS can be used throughout the life of a project, and the user is not required to learn a new simulation system as the project progresses.

In addition to multibody simulation, TREETOPS contains subroutines for control system analysis and design. Using this complete capability, the user can create and linearize complex, multibody models, import the plant model into MATLAB, design a feedback compensator in matrix form and export the results back to TREETOPS as a 'matrix controller' for final design verification.

The current version can be configured to execute on most Unix platforms as well as PC class machines. The graphics program, TREEPLOT, is customized for specific monitors and printers and is continuously updated. The PC version of TREEPLOT has yet to be developed; however, TREETOPS is completely compatible with the PC version of MATLAB and this product can be used for obtaining graphical output from TREETOPS.

Planned Enhancements for TREETOPS

A number of enhancements are planned for TREETOPS. Among these enhancements are order-N formulation for greater computational efficiency, the inclusion of inverse dynamics control and geometric nonlinearities, and an improved graphical user interface (GUI).

The multibody dynamics formulation and corresponding solution algorithm presently employed in TREETOPS is classified as an order-N-cubed approach, where N is the number of degrees of freedom. The dynamic equations of motion are formulated using Kane's Equations. The algorithm currently in use involves a matrix-vector implementation wherein a generalized NxN system mass matrix is formed and inverted

to solve for the N degree of freedom accelerations. This procedure requires N^3 operations. Research in numerical analysis has demonstrated that such problems can be solved using order-N algorithms requiring N operations. These algorithms essentially perform recursive operations to solve the equations of motion wherein the assembly and inversion of a system mass matrix is avoided. For a large system order, order-N techniques result in a substantial savings in computational time.

The increasing demand for high-operating speed, accuracy, and efficiency has led to strict requirements on the design of control systems for space-based manipulators. This requires consideration of a set of highly coupled nonlinear dynamic equations to determine the control torques and forces necessary to produce the desired motion of the manipulator. This also suggests the use of more sophisticated control schemes, such as inverse dynamics controllers. Hence, this feature will be added to TREETOPS. This enhancement is discussed in more detail in a later section.

Another planned enhancement is the inclusion of the effects of geometric nonlinearities. When properly accounted for, these terms will accurately reflect the motion induced change in stiffness of the structure. The current version of TREETOPS uses the assumed modes method to describe the elasticity in the links. The assumption in this method is that the elastic deflection is small and can be obtained as a linear superposition of the modes multiplied by their respective time-dependent amplitudes. These deflections are the axial and transverse elastic displacements, and rotations of a configuration point.

The assumed modes method is perhaps the most suitable method to describe the elasticity in any arbitrarily shaped body. Such a body can be mathematically discretized and its modal frequencies and mode shapes easily obtained using any linear finite element program. An approach is sought to compensate for the change in stiffness created by the use of the linear finite element program. One solution is the retention of the nonlinear part of the strain expression that is omitted in the linear finite element theory.

In the expression for the potential energy due to the nonlinear expression in the strain, the impressed loads (stresses) explicitly appear. Once these loads are specified, a stiffness matrix, called "the geometric stiffness matrix," which is analogous to the linear stiffness matrix, is obtained. This approach will be extended to multibody systems with arbitrarily shaped flexible bodies and included in the analysis code.

A GUI is currently under development. The goals for the GUI development are to increase learning speed and simulation implementation time, reduce errors, and encourage rapid recall for infrequent users. The desktop metaphor, with its windows, icons, and pull down menus, is very popular because it is easy to learn and requires minimal typing skills. The requirement to memorize arcane keyboard commands is also alleviated. The GUI will comprise full screen form using cursor keys and a mouse for movement from field to field. The input options will be designed as a set of icons.

TREETOPS currently lacks a unified environment in which to run the constituent programs with transparent data communications. The user must invoke each program at the command line with a problem name. The commands have a three level hierarchy. The user is constrained to sequential movement from a higher level to lower level. In addition the user must remember the exact command for each operation. Thus the user has the burden of with committing the entire command set to memory. With the new GUI, the user will be able to specify a problem name and choose any of the available options, including NASTRAN, TREESET, TREESEL, MATLAB, and others. If the option the user selects requires any interaction, then a form for that interaction is presented on the screen and the users simply provides the required input data. Communication between the different program elements will be through data files, but will be transparent to the user. The GUI will also have an extensive error checking routine executed at all stages of data entry. When an error is detected, the GUI will prompt the user to re-enter the data.

TREETOPS Modeling Features to be Verified via Laboratory Experiments

Several aspects of the flexible multibody modeling problem will be examined in the MMVC program. The primary focus will be on the evaluation of the assumed modes method when applied to multibody systems. In this technique, the structural flexibility of each body is modeled as a linear combination of spatial shape functions and generalized time coordinates. Through proper selection of the component shape functions or Ritz vectors, the system dynamic characteristics may be recovered. Several points will be addressed concerning the selection of the Ritz vectors. First, the type of Ritz vectors that should be used for various classes of multibody systems will be assessed. These vectors may be normal modes, Lanczos modes, block Krylov modes, and shape functions from substructure coupling techniques. Next, the sets of shape functions to be retained for each body will be determined as will the boundary conditions to be used in computing these shape functions. These points will be addressed through a series of increasingly complex experiments to be conducted in the MMVC laboratory. The experiments will be designed such that the flexible effects of the components dominate the time response of the system.

Experiments will also be designed to examine other aspects of multibody systems. Modeling techniques will be evaluated which account for geometric stiffening of systems described through the assumed modes method. These techniques account for changes in structural stiffness induced by motion and gravity. In particular, experiments will be performed to measure the time response of systems undergoing buckling loads and large angular velocities. These results will be compared to analytical predictions which account for the changes in stiffness. Additional studies will be performed to evaluate modeling techniques in the areas of joint friction, joint flexibility, kinematic and closed-loop constraints.

Assumed Modes Validation Plan

The MMVC validation plan consists of verification of the assumed modes hypothesis for a multibody structure and

will provide insight as to how the multibody structures should be modeled. The current procedure consists of three steps; 1) model development, 2) data collection, 3) post test analysis. The overall plan is illustrated in Figures 2, 3, and 4.

MMVC EXPERIMENTS

The proposed series of experiments for the MMVC program can be classified into three categories: 1) Open-loop topologies, 2) Closed-loop topologies, and 3) Space structures. Each of these categories have specific issues associated with them. For example, the open-loop topologies have one actuator for each joint while the closed topologies have fewer actuators than joints. Furthermore, in closed-loop topologies the component flexible links can be modeled independently, but the system imposes interdependencies between the component modes through closed-loop constraints. Space structures can belong to any of the above categories but elaborate modeling may be required and the control objectives may also differ significantly from those in the first two categories.

A set of experiments has been devised to address the modeling issues identified in the MMVC program. The first group of experiments considers open-loop topologies, the second set is for closed-loops, and the last set focuses on a representative space structure. The experiments are previewed in the following sections and the specific issues of each experiment are addressed. The experiments are ordered according to complexity. Each configuration will be used to address several modeling and dynamics issues and incorporate several control objectives.

Two control objectives will be used in virtually all configurations; pick-and-place control and trajectory control. The objective of pick-and-place is to move from one point to another without regard to the trajectory, while the second approach specifies the trajectory to be followed.

Open-Loop Topologies

The experiments designed for this class of problems are composed of single and two link systems connected through active and passive joints to a moving base. The base may be held fixed or actively controlled. The experiment configurations are based on a building block approach using interchangeable components. The designer may select from a wide variety of links with varying dynamic characteristics. There are aluminum and steel beams of varying cross sections and lengths, as well as more complex "geodesic" and "ladder" beams. Each of the beams has been modeled in NASTRAN and its component characteristics documented. There are standard mechanical interfaces to attach the beams to passive and active joints as well as tip masses and counter weights. The active joints are driven by DC torque motors and may be configured for planer or three dimensional experiments. Figures 5, 6, 7, and 8 are typical open-loop topology experiments. The objectives of the open-loop experiments are:

1) To demonstrate the coupling between rigid body and elastic motion of systems.
2) To address the issue of modal selection and types of shape functions used in the modeling process.

3) To investigate motion induced stiffness changes.

The control objectives are:

1) Pick and place control.
2) Pointing control.
3) Pendulum mode control.

Closed-Loop Topologies

This class of experiments consists of combinations of rigid and flexible links forming a closed-loop mechanism as shown in Figure 9. Typically, the number of active joints in the system is greater than the number of passive joints. These experiments are designed to validate use of kinematic and closed-loop constraints equations in multibody codes.

Space Structures

The previous beam experiments were designed to address several aspects of multibody dynamics and control through increasing levels of complexity. The Very Elastic Rotating NASA Experiment (VERNE) will incorporate the experience gained thus far into the modeling and control of a complex spacecraft. VERNE, shown in Figure 10, is composed of a moderately flexible core body, flexible pointing unit, two flexible solar arrays, and a pair of whip antennas with end masses. A rigid beam attaches the core body to the linear motion system of the facility through a ball joint. The experiment will inherently have two pendulum modes, which are rotations about the X and Y axes, and a roll mode about the Z axis. VERNE was designed such that the bending modes of the solar arrays and antenna are highly coupled with the pendulum modes. The pointing unit is connected to the core body through three linear electromechanical actuators, forming a closed-loop topology. The pointing unit has a range of motion of ± 30 degrees about the local X and Y axes. The linear actuators can generate a peak force of 200 pounds and have a throw of 18 inches. The pointing resolution of the unit computed from the accuracy of the incremental encoders on the lead screws of the actuators is .002 degrees. The point unit is two feet tall and is composed of three triangular plates connected by longerons. A generic housing was fabricated with the triangular plates to hold assorted laser or optical sensors.

The flexible solar panels are 8 feet long and 1 foot wide. The panels consist of thin aluminum struts bolted in a truss like fashion. The solar panels have 360 degrees of travel about the X axis and are powered by a direct drive D.C. motor. The drive shafts are instrumented with incremental encoders and tachometers. The encoder resolution is .35 degrees. The peak torque available from the motors is 11 foot-pounds.

The core body is composed of aluminum angle. The whip antenna are rigidly connected to the core body. Three orthogonal reaction wheels are mounted to the core body along the body axes. Each reaction wheel is driven by a D.C. torque motor equipped with a tachometer. The core body is also instrumented with a three axis rate gyro system.

The preliminary system modal characteristics are shown in Table 1. The first two bending modes at .263 and .275 Hertz are torsion modes of the solar panels about the drive shafts. The next mode is a

system pendulum mode at .366 Hertz about the Y axis. The bending mode at .484 Hertz is a combination pendulum mode about X and solar panel torsion. These modes may be shifted through the use of counter weights and adjustments to the solar panels and antenna.

Table 1. Preliminary System Modal Characteristics

Mode	Frequency (Hz)	Description
1	0	Rigid Body Rotation About Z
2	.263	Solar Panel Rotation in Phase
3	.275	Solar Panel Rotation
4	.366	Pendulum About Y
5	.484	Pendulum About X / Solar Panel Torsion
6	1.577	Antenna 1st Bending About X in Phase
7	1.640	Antenna 1st Bending About X
8	1.718	Antenna 1st Bending About Z
9	1.795	Antenna 1st Bending About Z
10	5.164	Solar Panel 1st Bending

VERNE Experiments

The objectives of the experiments proposed for VERNE are divided into dynamics and controls. The objectives of the dynamic open-loop tests are:

1) to test the validity of the generalization of modal selection issues from earlier experiments.
2) to study the pendulum modes in a multi-body context.
3) to study motion coupling through various prescribed open-loop maneuvers.

The control objectives are:

1) pointing control in the presence of base excitation.
2) pointing control in the presence of solar panel maneuvers.
3) pointing control in the presence of pendulum modes.

Three open-loop experiments have been proposed. First, the translational degree of freedom of the linear motion system will be locked and the solar panels will be driven through various slew maneuvers. Next, the solar panels will be held fixed and the system will be driven through base excitation. Finally, the solar panels will again be driven, but this time in the presence of base excitation. The effect of solar array motion and base excitation on the system pendulum modes will be studied using sensor time histories and compared to analytical results.

The controls experiments consist of accurately pointing the lower unit in the presence of solar panel motion and base excitation. The control system designer will

have access to line of sight error from a light source on the lower unit illuminating a quad detector on the ground. The designer will also have information from the rate gyros, solar panel drive shaft position and rate, and relative angle between the core body and lower pointing unit. The engineer must design the loops generating torque/force commands for the reaction wheels, solar panel drives, and linear actuators from the feedback of the various sensors.

THE MMVC LABORATORY FACILITY

The MMVC project consist of multibody modeling, verification, and control. Currently dynamic multibody systems with flexible members and large rotations and translations at the joints are modeled using TREETOPS. Information on the flexible modes is input to the code from NASTRAN models of the bodies. There are many open questions as to which modes should be input to TREETOPS - that will be addressed in the modeling experiments. TREETOPS has been widely used for many years, but its results have never been experimentally confirmed. This issue will be addressed in the verification section. Finally, new methods for control of the structures will be investigated in the control section.

Platform and Linear Motion System Design

The MMVC facility will be located in the west high bay area of building 4619 at MSFC. This facility is joined with the Flexible Space Structures (FSS) ground test facilities and is accessed via the control room. The two primary requirements for MMVC facility are experiment work volume and support structure stiffness. The desired work volume is 20' by 20' by 20'. This will allow room for large translations and rotations of the experiments, as well as for larger test articles needed for low frequency modes. The experiment support structure must withstand the static and dynamic loads from the test articles. The structure should also isolate the experiments from unwanted disturbances. Isolation will be accomplished by moving the support structure natural frequencies to a range outside of those under study. Other factors considered in designing the facility were: facility enclosure, power, lighting, ventilation, access, safety, and cost.

Currently, outside of the FSS control room in Building 4619, there is a balcony off the third floor in the high bay. Three locations for the facility were considered. First, the experiments could be hung from the existing balcony. Second, the experiments could be enclosed in a stand-alone structure below the existing balcony on the first floor. Finally, the test articles could be suspended from a fixture above the existing balcony. The last alternative was chosen because of several advantages. The primary advantage is that the real-time computer controlling the experiments will be located in the existing FSS control room. Also, test articles will be highly visible from the control room and the current platform or balcony. This location will have a high work volume and require no external lighting or ventilation. The system bending modes computed from finite element analysis are shown in Table 2. These modes were calculated assuming an 800 pound experiment located in the center of the front edge of the new platform. As expected, this is a diving board mode of the new structure at 19.7 Hertz. The frequency is well above those of interest of the experiments.

Table 2. MMVC Facility

Mode	Frequency	Description
1	19.702 Hz	Platform Bending
2	22.381 Hz	Localized Torsion
3	23.540 Hz	Localized Bending
4	25.550 Hz	Localized Torsion
5	27.872 Hz	Localized Bending

A linear motion system will be installed along the front edge of the new balcony. The motion system has a range of travel of 6 feet with a sensor resolution of .003 inches. It is a ball screw system driven by a brushless DC motor with a peak force capability of 430 pounds and can withstand loads well above 800 pounds.

MMVC Real-Time Control System

The user interface is through the Silicon Graphics Personal Iris 4D-25TG console. The real-time functions will be predominantly executed on four Mercury Computer Systems MC860VB-4 single board computers running MC/OS Version 2.0. A SPARC Engine 1E single board computer serves as a host for the MC860VBs. The host interfaces the Mercury boards to a SCSI bus and Ethernet.

The I/O boards consist of a Xycom XVME-203 Counter/Timer Board, a VME Microsystems International VMIVME-2528 128-bit Digital I/O Board, four Datel DVME-611F 14-bit Analog Input Boards, and four VME Microsystems International VMIVME-4100 Analog Output Boards. The MMVC Closed-Loop Controller will be used to provide digital control of the test articles in the MMVC Lab. The controller will be interfaced to the experiment of sensors, compute control outputs, and apply the outputs to the experiment of actuators. The closed-loop control laws will require a large amount of computational power, and must be executed at rates as high as 250 Hz.

MMVC CONTROLLER METHODS

Many control schemes have been evaluated that would not only provide adequate tracking, but also provide vibration suppression. The major problem with these linear design techniques is that the structure (plant) is a highly nonlinear system. Control design studies have showed that a linear controller, designed for the MMVC experiments may result in unstable systems for large-angle slew commands. This is because of the interactions between the control system and the nonlinear centrifugal stiffening, softening, and Coriolis effects. In the following paragraphs are presented three control schemes that may provide acceptable controllability and performance while the system is undergoing these nonlinear interactions.

Inverse Dynamics Controller

One approach to compensate for nonlinear forces is to use a technique referred to as inverse dynamics control.[2] [3] The way the inverse dynamics control law works is illustrated by considering the following equation

$$m(q)\ddot{q} = u(q,\dot{q}) = B(q)r \qquad (1)$$

where q is the n-dimensional vector of generalized coordinates, M(q) is the n x n mass matrix, u is the n-dimensional vector including the effect of centripetal, Coriolis, and gravity terms as well as all other stiffness and damping terms, r is the external torque (or force) vector of dimension m, and B(q) is the n x m torque distribution matrix.

The idea of inverse dynamics control is to seek a nonlinear control logic expression

$$r = f(q, \dot{q}) \qquad (2)$$

which, when substituted into equation (1), results in a linear closed-loop system. Here, we assume that the state vector, q, is available.

In this paper, we consider the general case where the number of external torques can be less than the number of the generalized coordinates describing the equation of motion (1). Several control logic expressions and their computational steps are developed to apply the inverse dynamics control to this case.

TREETOPS subroutine facilities are used to perform this computation. The state vector, q, is defined to be the set of the hinge angles and translations and the modal coordinates of flex modes. The non-actuator forces, i.e., forces due to gravity, stiffness, damping, etc. are summed with the inertial forces. Also, the torque distribution matrix B(q) is not directly computed.

Model Reference Adaptive Control

Another control design option for the MMVC experiments is a spin-off from the model reference adaptive control (MRAC) methodology referred to as Direct Multivariable Model Reference Adaptive Control (DMMRAC). The primary advantage DMMRAC possesses over conventional MRAC and other control techniques is that it is completely model independent. DMMRAC is a nonlinear adaptive control methodology driven only by the accumulated error between the reference model and plant outputs. The nonlinear part of the filter results from the adapting law being a function of the square of the reference model states. Unlike classical MRAC, DMMRAC does not require any knowledge of the plant. Therefore, the order of the reference model is strictly up to the designer. Conventional MRAC methods require the order of a reference model to be at least equal to that of the plant. This is a major drawback for these other methods because predicting the order of a complex nonlinear plant is essentially impossible.

Fuzzy Control

The MMVC team is currently searching for new and innovative control methods for large space structures. Fuzzy logic control holds much promise in this application.[4] [5] [6] Fuzzy logic is a rule-based control methodology based on linguistic phrases and provides control the way a human operator would. It is especially suited for the nonlinear, time varying, and ill-defined systems such as large flexible structures. Another key feature to fuzzy logic is that it is completely model independent. Typical fuzzy rules are of the form:

$$\text{If } X_1 \text{ is } A_{i,1} \text{ and } X_2 \text{ is } A_{i,2} \text{ then } U \text{ is } B_i \qquad (3)$$

where X_1 and X_2 are the inputs to the controller, U is the output, A's and B's are membership functions, and the subscript i denotes the rule number. For example, a rule for line-of-sight error control may state "If the Line Of Sight (LOS) error is negative small and the change in the LOS error is positive big, then torque is positive small". Given input values of X_1 and X_2, the DOF of rule "i" is given by the minimum of the degrees of satisfaction of the individual antecedent clauses i.e.,

$$DOF = \min \{A_{i,1}(X_1), A_{i,2}(X_2), \ldots\} \quad (4)$$

The output value is computed by

$$u = \frac{\sum_{i=1}^{N} (DOF_i) B_i^d}{\sum_{i=1}^{N} (DOF_i)} \quad (5)$$

where B_i^d is called the defuzzified value of the membership function B_i and n is the number of rules. The defuzzified value of a membership function is the single value that best represents the controls linguistic description. If a rule is active for the present conditions such that its output is "increased moderately", the defuzzified value is the centroidal value about the abscissa. In this case the defuzzified value is 3.0.

For control of highly nonlinear, time varying, and hard-to-define dynamics of large flexible structures, fuzzy logic with its model independence properties may prove to be a very practical method of control.

CURRENT EXPERIMENT ACTIVITIES

In order to develop analytical models of the system configurations, it is essential to accurately model all of the components that comprise the system. Figures 2 and 4 conceptually describe this process. In order to increase the fidelity of the system components, the first phase of experimentation involves component testing. Component testing involves beam-element modal tests, joint-element dynamic and static testing, and frequency response testing of the sensors and actuators.

Free-free modal tests were performed on the beam specimens in order to validate component mode shapes and frequencies predicted by NASTRAN, and to identify the damping ratio of each component mode. As expected, the free-free NASTRAN predictions match well with the free-free test results, within about five percent. Table 3 shows the results of the free-free modal test for one particular beam.

The next phase in the component testing plan is clamped-free modal tests. These tests will attempt to validate the clamped-free modes predicted by NASTRAN. The clamped-free and free-free component modes can then be used in assembling models. Next, system-level experiments will be performed. At this point, modal analysis will be carried out to determine which type of modes to use and what type of substructure coupling method best predicts the results.

Table 3. Free-Free Modal Test Results: First Four Modes of DYN30-254

Mode	Frequency (Hz)	Damping (%)
First Bending	3.24	3.5
Second Bending	8.59	1.4
Third Bending	17.23	0.9
Fourth Bending	30.67	0.5

SUMMARY

The MMVC program has been established at MSFC to experimentally validate multibody modeling codes and to improve the computational efficiency of such codes. Experiments have been designed to emphasize modeling features that are to be verified and validated in the effort. A laboratory facility has been designed and is under development. The RTCS is in place and has been functionally verified. Preliminary experiments that do not require the test volume to be provided when construction of the MMVC laboratory is completed are under way. Enhancements to the TREETOPS code are initiated and ongoing. This paper has presented a top-level overview of the MMVC program and its goals and methods.

REFERENCES

1. TREETOPS User's Manual, Dynacs Engineering Company, Inc., Revision 8.
2. H. Baruh and S.S.K. Tadikonda *Issues in the Dynamics and Control of Flexible Robot Manipulators,* AIAA Journal of Guidance, Control, and Dynamics, Vol.12, No.5, Sept.-Oct. 1989.
3. H. Baruh and K. Choe *Sensor Placement in Structural Control,* AIAA Journal of Guidance, Control, and Dynamics, Vol.13, No.3, May-June 1990.
4. Sugeno, Michio, *An Introductory Survey of Fuzzy Control*, Information Sciences, Vol.36, 1985.
5. Chiv Stephen, *Robustness Analysis of Fuzzy Control Systems with Application to Aircraft Roll Control*, AIAA, J. Guidance and Control, June 1991.
6. Zadeh, Lotfi A., *Making Computers Think Like People*, IEEE Spectrum, August 1984.

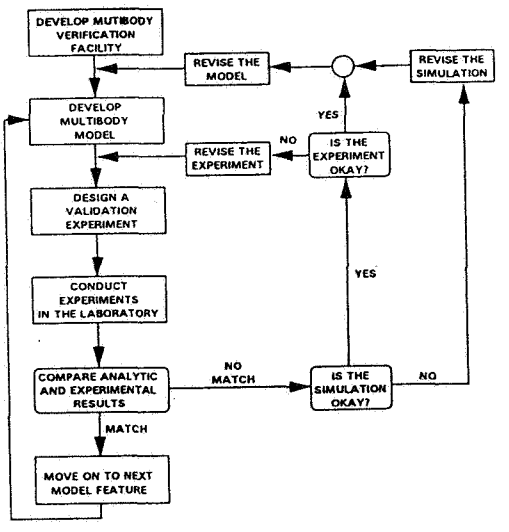


Figure 1

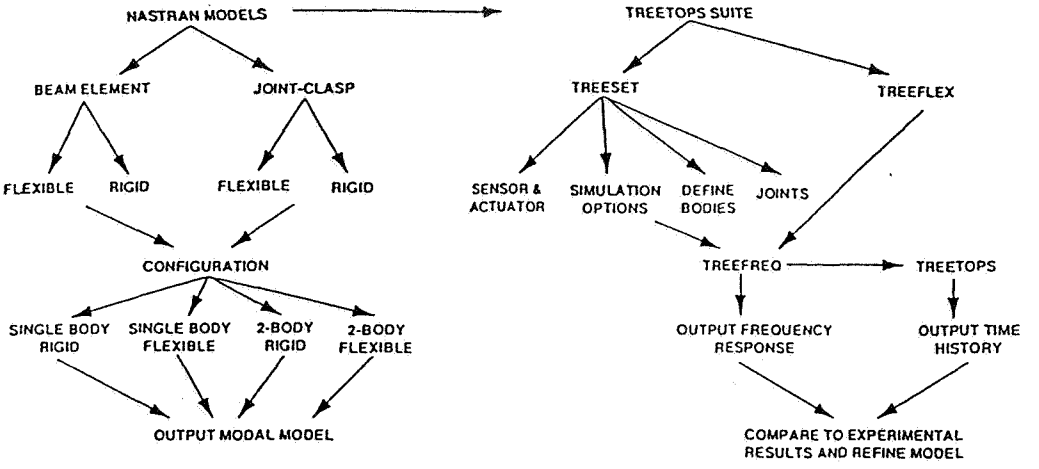


Figure 2

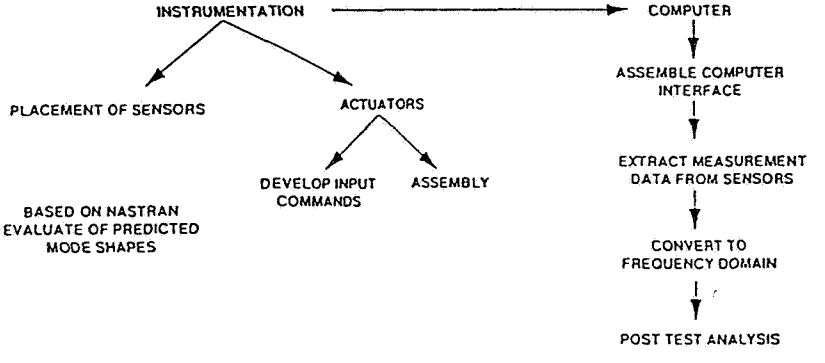


Figure 3

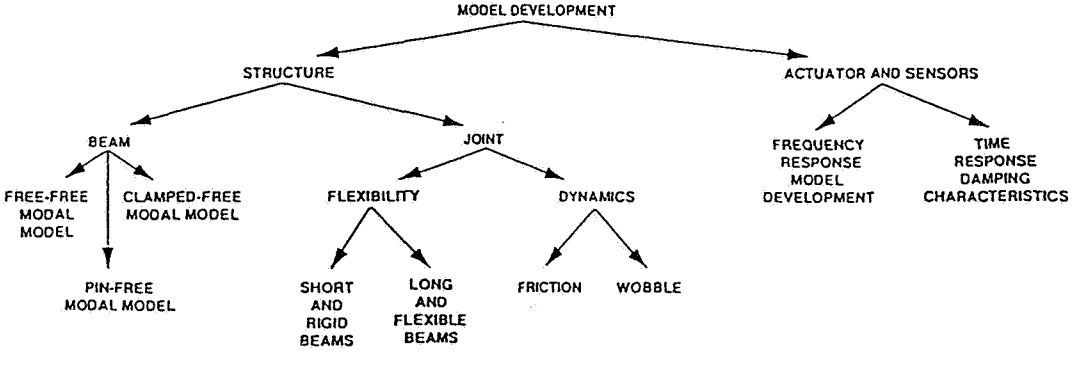


Figure 4

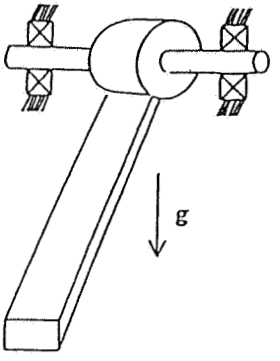


Figure 5

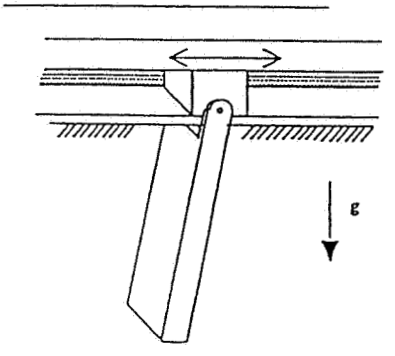


Figure 6

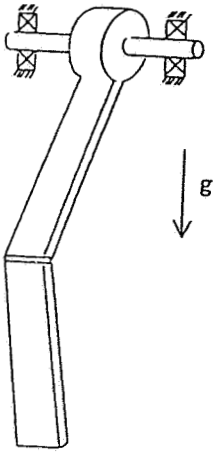


Figure 7

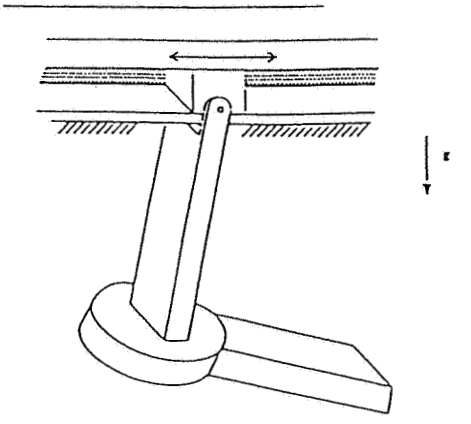


Figure 8

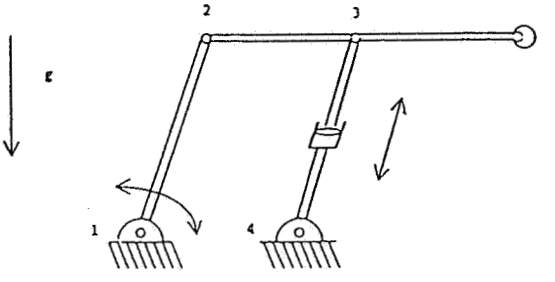


Figure 9

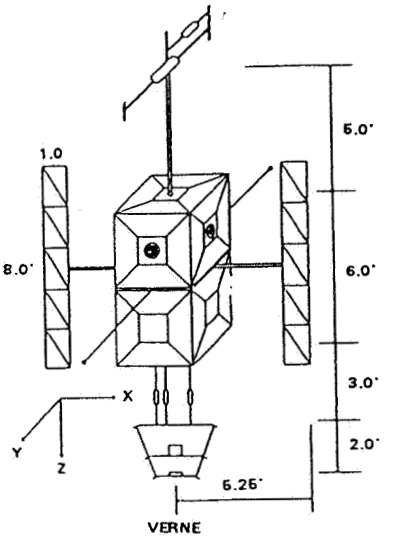


Figure 10

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No. 93-02	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Proceedings of the Fifth NASA/NSF/DOD Workshop on Aerospace Computational Control		5. Report Date February 15, 1993 6. Performing Organization Code
7. Author(s) M. Wette and G. K. Man, editors		8. Performing Organization Report No.
9. Performing Organization Name and Address JET PROPULSION LABORATORY California Institute of Technology 4800 Oak Grove Drive Pasadena, California 91109		10. Work Unit No. 11. Contract or Grant No. NAS7-918 13. Type of Report and Period Covered JPL Publication
12. Sponsoring Agency Name and Address NATIONAL AERONAUTICS AND SPACE ADMINISTRATION Washington, D.C. 20546		14. Sponsoring Agency Code RE156 BK-506-59-61-12-00

15. Supplementary Notes

16. Abstract

The Fifth Annual Workshop on Aerospace Computational Control was one in a series sponsored by NASA, NSF and the DOD. The purpose of these workshops is (1) to address computational issues in the analysis, design and testing of flexible multibody control systems for aerospace applications and (2) to bring together users, researchers and developers of computational tools in aerospace systems (spacecraft, space robotics, aerospace transportation vehicles, etc.) to exchange ideas on computational tools and techniques.

The Fifth Workshop provided attendees a window into current research, development and applications in computational issues related to the development of flexible multibody systems. The workshop objectives were to provide

- o NASA, NSF and DOD a window into current research
- o tool users and tool developers an opportunity to exchange ideas and experience on the correctness, efficiency and usability of current computational tools
- o researchers and users an opportunity to see what areas are ripe for application and what areas require more effort
- o academia and industry an opportunity to strengthen cooperation to aid the transfer of new technologies to applications

The focus of the workshop was in modeling, flexible multibody simulation, CAE issues, control system design, testing and verification, and applications.

17. Key Words (Selected by Author(s))	18. Distribution Statement
Spacecraft design, testing and performance Mechanical engineering Mathematical and computer sciences (general) Laboratories, test facilities, and test equipment	Unclassified; unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	578 plus cover	