PLANNING AND EXECUTING MOTIONS FOR MULTIBODY SYSTEMS IN FREE–FALL

Ph. D. Dissertation Proposal

Jonathan M. Cameron

January 31, 1991

CONTENTS

(Continued)

APPENDICES

Abstract

How do multibody systems move in free-fall? For instance, when a cat falls, it flips over before it reaches the ground. How does it do that? Multibody systems in free-fall move very differently than robots which are bolted to the ground. Once a robot with a fixed base stops moving, the link positions can be determined by kinematics alone. This is not true for a robot or multibody system in free-fall. The final link positions of a robot in space depend on the link trajectories during the motion as well as its kinematics. Kinematics and dynamics are tightly coupled for multibody systems in free-fall. Given these difficulties, how can we plan motions for multibody systems in free-fall?

The proposed research will center on several issues necessary to plan and execute motions for multibody systems in free-fall.

1. *What motions are possible for a multibody system in free-fall?* Mathematical techniques from nonlinear control theory will be used to study the nature of the system dynamics and its possible motions.
2. *How can we plan the link motions and joint torques necessary to move from one configuration to another?* Optimization techniques will be applied to plan motions.
3. *How can we store precomputed motion plans efficiently?* Since it is unlikely that motion plans can be computed in real time, precomputation will be necessary. Image compression techniques are proposed to compress the precomputed motion data for storage.
4. *Once a motion is planned, how can the system execute the motion faithfully?* A linearized controller will be devised to control the system while it executes preplanned trajectories.

Symbolic manipulation techniques will be used in the research (where practical) to reduce chances for algebraic errors and to make the approach easier to apply to new multibody systems in free-fall.

The proposed research applies to a number of activities. Most obviously, it can be used to plan motions for robots in space. It can be used to plan limb motions to reorient astronauts. The research may also be useful to plan the movements of airborne divers, gymnasts, and jumpers.

1 Introduction

1.1 Purpose

The motion of ground-based robots is reasonably well understood. The underlying kinematic and dynamic analysis relies on the fact that robots are bolted to the floor—which will not move appreciably despite the motions of the robots. This is not true for robots in space. More generally, this is not true for multibody systems in free-fall. When a multibody system is not attached to the earth, its motions are considerably more complex than its ground-based counterparts. The dynamics and kinematics become inextricably coupled.

Planning motions for multibody systems in free-fall is more difficult than for fixed-base robots largely because of the interaction of the kinematics and dynamics. Planning feasible or optimal trajectories will require extensive off-line computation and cannot be done in real time. A way is needed to precompute and store the possible motions so that the possibilities can be retrieved later and used quickly for real-time planning and execution purposes.

The purpose of this research is to develop an end-to-end system that can be applied to a multibody system in free-fall to analyze its possible motions, save those motions in a database, and design a controller that can execute those motions. A goal is for the process to be highly automated and involve little human intervention. Ideally, the output of the system would be data and algorithms that could be put in ROM to control the multibody system in free-fall.

This research applies to more than just robots in space. It applies to any multibody system in free-fall. This includes astronauts in space, falling mechanisms, athletes in jumps or dives, and airborne gymnasts.

1.2 Motivation

To illustrate the complexities and potential of the types of motion that will be addressed by this research, consider the following examples.

Falling cat problem

When a cat is held upside down and dropped, it manages to turn itself rightside up before it lands. This well-known phenomenon has long intrigued children and scientists. Thomas Kane resolved the question in 1969 with a dynamic simulation that showed the motions necessary to turn the cat over during its fall [35]. His dynamic simulation involved only two bodies but duplicated the complex motion of the cat during the maneuver.

Astronaut attitude change maneuvers

Astronauts are taught a series of maneuvers that allow them to change their orientation while in free-fall. Again, it was Kane who developed these maneuvers [36, 37]. For each of the desired rotations (pitch, roll, and yaw), he developed simplified equations of motion and then analyzed them to determine what cycles of limb motions were necessary to give the desired orientation change. The result was several cycles of simple limb motions that produce orientation changes about the desired axis.

Divers, Gymnasts, and Jumpers

Some of the most complex motions of systems in free-fall occur when springboard divers are in the air [21, 132, 133, 134, 135]. Gymnasts and athletes perform similar maneuvers while in the air during their activities. The movements of high jumpers while off the ground are also complex. The "Fosbury Flop" revolutionized high jumping by improved maneuvers while in the air[125].

Robot servicing in space

Using robots to service satellites in orbit is a goal of NASA [56, 101]. Several robot designs have been proposed which involve complex arms attached to large bodies with control-moment gyros and thrusters for maneuvering and station-keeping [8, 33, 57, 58]. These approaches to motion control have their drawbacks. Control-moment gyros are complex and expensive. Thrusters

produce plumes which may impinge on delicate equipment. If techniques could be developed to plan and execute attitude and configuration changes via limb motions, simpler, safer, and less expensive servicing systems could be designed.

2 Current Approaches (Literature Review)

Many researchers have addressed various aspects of this research. Relevant research is reviewed below.

2.1 Motion Planning and Control for Robots in free-fall

Only a few researchers have directly addressed the problem of how to plan and control motions of a robot or multibody system in free-fall.

Kane's work in astronaut maneuvering has already been mentioned. This research was reported in 1970-72 [36, 37]. He worked out the equations of motion for the human body and simplified them for the desired rotations of the body trunk. The resulting cyclic motions are interesting and useful but very different from the types of motion desired in this research. In any case, his work was specific to the human body and did not address the question of general configuration change of multibody systems in free-fall.

Longman, Lindberg, and Zedd considered a robot arm mounted to a satellite and developed special kinematics that dealt with the dynamics-kinematics interaction problem [57, 54, 58]. They assume the satellite base body contains reaction wheels to keep it from rotating when the arm moves. Their kinematics compensate for the translational movement of the base during movements of the arm. Their kinematic simplifications depend on the absence of base rotation and do not generalize to multibody systems without reaction wheels.

Vafa and Dubowsky developed the *virtual manipulator* technique for analyzing the kinematics and dynamics of robots in space [117, 118, 119, 120, 121, 80]. They devised a way to construct an imaginary manipulator with dimensions and inertial characteristics related to the actual system. The motion of the imaginary system and the real system are closely related and exactly the same for one point of the actual manipulator (such as the end effector). Once the position of this common point is determined, the necessary virtual manipulator configuration is easily computed and then the corresponding joint positions of the actual system can be determined. The dynamics of

virtual manipulators also have the advantage that the conservation of linear momentum is implicitly integrated and eliminated from the equations of motion. Unfortunately, this approach does not provide any solutions for how to move from one configuration to another. In other words, the virtual manipulator approach can be used to determine the final joint positions to accomplish some task but is not very helpful in determining the necessary joint motions to move from the starting configuration to the final configuration. They did apply this technique to manipulator motion planning by using small cyclic motions to produce small motions of the manipulator base. Although this approach is useful for planning the motion of space manipulator end-effectors, it has limited usefulness for planning large motions in which the entire final configuration is specified (such as in gymnastics).

Given an end effector position that can be reached, the virtual manipulator approach can be used to determine the necessary joint angles of the space manipulator to acheive that position.

Umetani and Yoshida studied continuous path control of manipulators. They devised an extended Jacobian for kinematic analysis of the motion of the end-effector [116]. The generalized Jacobian enforces the conservation of linear and angular momentum for the space manipulator. They use the new Jacobian to plan continuous motions of the end-effector in space and simulate them for a OMV (Orbital Maneuvering Vehicle.) This approach has similar benefits and limitations as Vafa's work because it concentrates on end-effector motion.

Nakamura and Mukherjee also addressed the problem of planning motions for space robots [71]. Their work deals specifically with the nonholonomic nature of the conservation of angular momentum. They devise kinematics which incorporate the linear and angular momentum and of the space manipulator. They then devise a controller based on a Lyapunov function. Their approach is promising but initial results were disappointing because the controller could get stuck during the motion. They solved this problem in their later work [72, 73] by designing a bi-directional control algorithm. Unfortunately, the resulting motions involve unusual cyclic motions and other peculiarities which indicate that the motion is not very general.

Sreenath and Krishnaprasad (among others) use mathematical approaches to attack the problem of controlling the motion of multibody systems in

free-fall [102, 103, 45, 131]. While these methods are difficult to understand, they do appear to offer promising techniques. Unfortunately, their research is not very useful to the proposed research because they currently consider only planar systems. In "Nonlinear Control of Multibody Systems in Shape Space," N. Sreenath indicates that extending their results to three space is "definitely non-trivial." [102, p. 1780].

In a recent paper[67], Murray and Sastry use Chow's theory and Lie brackets to determine whether a motion is possible for a system subject to a nonholonomic constraint linear in velocities. If the motion is possible, they construct a path using sinusoidal path segments. They apply their technique to the motion of planar systems in free fall and to the nonholonomic vehicle problem. Their approach has useful insights and may be applicable to the proposed research but has not yet been applied to non-planar motion.

2.2 Path Planning for Mobile Vehicles

An area related to the current research is path planning for mobile robots. Much research has been devoted to planning land vehicle motion for indoor and outdoor vehicles. Works which considers mobile vehicles as points are not considered here since they are irrelevant to the proposed research. There are a few researchers who address the nonholonomic nature of vehicles with limited steering capabilities.

Laumond considers a nonholonomic vehicle and proves that it is possible to plan collision free paths through a cluttered area by combining sets of small cyclic motions [46]. In later work he showed that whenever it is possible to plan a jagged path, it is also possible to plan a smooth path for the same motion [47]. Barraquand and Latombe addressed similar issues and devised planning techniques based on potential field techniques [5]. They apply their approach to difficult problems such as parallel parking a vehicle with several trailers. They also applied their approach to robot arms with many degrees of freedom. Similar research is covered by Jacobs and Canny [30, 31]. This type of research has many insights to offer for nonholonomic systems. Unfortunately, the nonholonomic constraints due to limited steering angles are simpler than the nonholonomic constraints due to the conservation of angular momentum. These approaches have not been applied to multibody

systems in free-fall and it is not clear how applicable they are.

2.3 Trajectory Planning and Control for Fixed–Base Robots

An extensive amount of research has been done on planning motions for fixed–base robots. A sampling of this research is given in the references [6, 17, 34, 40, 22, 23, 78, 64, 74, 75, 91, 89, 97, 98, 100, 105, 114, 126, 127].

One of the most promising approaches is presented by Tan and Potts [106, 107, 108]. Their approach does everything. Their technique is intended for fixed–base robots but is general enough to handle multibody systems in free-fall. Their technique handles full dynamic nonlinearities, actuator limitations, joint constraints (position, velocity, and jerk), avoids obstacles, and incorporates an energy objective as well. This approach has not been adapted to multibody systems in free-fall but appears useful for this research.

Another promising approach to planning optimal motions is given by Luus in recent research on controlling chemical processes [61]. His approach is based on dynamic programming and may be useful for the problem at hand. Luus has applied his technique to problems with up to eight nonlinear ordinary differential equations and determined optimal controls with limits on input variables.

2.4 Tabular Planning and Control

Another area relevant to this research is precomputing motion data and storing for later use in planning and control. This is sometimes called a *tabular* approach since motion data are precomputed and stored in tables for later retrieval in planning or control. Very little has been done in this area.

Raibert does use tabular techniques with some success for control of the cyclic parts of motion of his one–legged hopping machine [81, 82, 83]. Tabular techniques were also proposed by Albus [1, 2]. Hollerbach criticizes tabular approaches in general (and these in particular) when he concludes that dynamics simualtion codes can be made fast enough to run in real time [27]. This criticism is not relevant to the proposed research. It may be pos-

sible to simulate the motion of multibody systems in free-fall faster that real time if the torques or forces to apply at each actuator during the motion are known before hand. The proposed research is to determine these actuator inputs. This cannot be done in real-time using known techniques even on super computers.

2.5 Symbolic Manipulation

Symbolic manipulation offers researchers many opportunities to improve the quality of their work by producing results much faster than is possible by hand, reducing the chance of mathematical errors, and allowing handling of more difficult problems. Applying symbolic manipulation to robot kinematics and dynamics is not new.

Hussain and Noble used symbolic computation for forward and inverse kinematic analysis of specific robot geometries which assisted the user but still required considerable interaction[28]. Direls developed a system for manipulation of matrices with symbolic entries and used this to analyze robot kinematics [16]. Kircanski and Vukobratovic constructed a system using FORTRAN-77 to symbolically generate the forward kinematics and Jacobian of a robot but not the inverse kinematics solution [41]. Lloyd and Hayward applied MACSYMA to the same problem and derive kinematics for several common robot architectures [55]. Tunstel and Vira also use MACSYMA to construct robot kinematics symbolically as an educational and design aid [113]. They also introduce a number of rules (symbolically implemented) that simplify the results.

Many researchers have also developed dynamic equations of motion for multibody systems symbolically. Liegois and company developed PL/1 software to derive equations of motion using a Lagrangian formulation. Others have written FORTRAN programs for symbolic generation of equations of motion for multibody systems using various approaches: Newton–Euler [43, 44] and Kane's equations [18]. Other similar work has been done by various researchers [7, 11, 24, 26, 29, 49, 50, 62, 66, 76, 77, 69, 88, 90, 94, 95, 96, 109, 110, 111, 112, 122, 128, 129, 136, 137] Others have applied similar techniques to systems with flexible components [12, 59, 115]. Many of these systems generate the equations of motion encoded in a FORTRAN or C program suited

to compiling and running for simulation purposes. Symbolic manipulation has also been applied to control applications [94, 104, 107].

2.6 Multibody Dynamics

Multibody Dynamics is a huge field. Many people have developed widely varying approaches to the problem of simulating and controlling multibody systems. Several references cover Multibody dynamics in detail [4, 19, 93, 85, 130]. Others, too numerous to mention, deal with dynamics in general and are applicable to multibody dynamics. Although serious multibody dynamics research was done more than 80 years ago [20], the field is not exhausted. Recent developments include many recursive techniques for inverse dynamics with operations counts proportional to the number of elements [3, 15, 19, 25, 27, 38, 39, 53, 60, 65, 86, 87, 124, 123]. (Most of these are based on recursive Newton-Euler approaches; some are based on operation space approaches [19, 53, 86, 87].) The most efficient of these approaches is given by He and Goldenberg [25]. Their recursive technique requires $91(n-1)-6$ multiplications and $86(n-1)-10$ additions, where n is the number of bodies. The efficiency of these recursive techniques allows the computation of joint torques necessary to produce desired motions in real-time for reasonably complex systems. Forward dynamics algorithms are not quite as efficient yet [48].

3 Proposed Research

Before analyzing the proposed research in detail, an overview may be helpful to orient the reader.

3.1 Overview

The goal of the research is to develop and test a system which can precompute, save, and execute motions for multibody systems in free-fall. The basic components of the research are listed below.

1. Analyze motion possibilities
2. Implement simulation system
3. Implement symbolic construction of equations of motion
4. Design optimal controls to accomplish motions
5. Implement symbolic generation of optimal control scheme
6. Precompute motions between selected configurations
7. Adapt compression techniques to compress motion data
8. Design linearized motion tracking control scheme
9. Implement symbolic generation of linearized controller
10. Use simulation to verify linearized controller
11. Apply system to example multibody systems

3.2 Proposed Approach

3.2.1 Definitions

Several terms are used in specific ways in this proposal and are defined here. The terms appear in italics in the following definitions.

Configuration (or *pose*) refers to the shape of the body as determined by the joint positions. *Orientation* refers to the attitude of the system with respect to some global reference frame. More precisely, orientation refers to the attitude of some reference link of the body with respect to a global reference frame. If a robot is bolted to the floor, there is no reason to make the

distinction between configuration and orientation. Once the base of a robot or multibody system is free to move with respect to the global reference frame, this distinction becomes useful and important.

Typical Configurations are configurations of the multibody system that occur often during motions of the system and are useful in studying and planning its motions. For instance, a tuck is a typical configuration for divers. For more detail, see Appendix A, page 32.

Motions refer to movement from one combination of configuration and orientation to another combination of configuration and orientation. In this research, this will be accomplished strictly by joint motions.

3.2.2 Analyze motion possibilities

What motions are possible for multibody systems in free-fall? That question is central to this research. The possible motions depend on the nature of the mechanism, the initial configuration and orientation and the final configuration and orientation. For instance, a mechanism with one revolute joint (like a hinge) has a limited range of motion. It can open and close but the axis of the hinge cannot be tilted by opening and closing the hinge. This is because its motion is holonomic. A nonholonomic system has more potential motions. Consider a vehicle on the plane with a limited steering angle. The front wheel imposes a motion constraint that is nonholonomic. The vehicle has three degrees of freedom in the large but only two degrees of freedom at any instant. Yet, by careful maneuvering, any position in the plane can be reached. Multibody systems in free-fall must conserve angular momentum because they have no external torques acting on them. The conservation of angular momentum can be thought of as a nonholonomic constraint on the motion of the system in free-fall. Depending on the character of the angular momentum, a mechanism in free-fall may be able to move from any combination of configuration and orientation to any other combination of configuration and orientation: or it may not—as in the case of a hinge in free-fall. Obviously, since no external forces are used, the system center of mass will not move either case.

This research will investigate this issue further and devise tests to be applied to determine if each of the desired motions is possible. For example, Frobe-

nius' theorem can be applied using Lie brackets to evaluate the nonholonomic nature of the angular momentum (whether it is integrable) [51, 5, 71, 99, 70]. This can be done symbolically[42] since the angular momentum can be generated symbolically. Research will also address the general controllability and reachability for these systems. It should be noted that it is very difficult to perform this type of research without symbolic manipulation due to the complexity of the equations of motion and angular momentum.

3.2.3 Implement simulation system

A basic part of the proposed research is a simulation environment in which the various components of the research will be implemented and tested. This simulation system will allow the user to construct robots from links and joints and then simulate kinematics and dynamics of the robots. The simulation environment will be used to verify the resulting motion libraries and control schemes. An extended description of how the simulation environment can be used is included in Appendix A.

The initial implementation of the simulation environment will handle multibody systems composed of rigid bodies since that is the focus of this research. To be even more useful, the simulation environment should also be able to handle flexible members. The software design and implementation will make provisions for future expansion in this direction.

The software approach will be object-oriented and the code will be written in an appropriate computer language such as C++, object-oriented Pascal, or Ada. An important component of such a system is the graphical display. These considerations and the goal of source-code portability indicate that C++ and X-Windows might be a good choice.

3.2.4 Implement symbolic construction of equations of motion

A number of systems exist for studying the motion of multibody systems. These include SD/Fast, SD/Exact, Autolev, DADS, and ADAMS. Others are mentioned in the literature review. These systems simulate multibody motions, and some generate C or FORTRAN code for simulation and control purposes. Unfortunately, the output of most of these systems is not

directly suitable for further symbolic manipulation (for controls analysis, for instance.)

The proposed system will generate equations of motion in symbolic form suitable for further symbolic manipulation. (A few of the systems mentioned in the literature review do this.) The resulting symbolic form of the equations of motion will be used in three ways. First, the equations of motion will be used to generate executable code for simulation and planning purposes. Second, the equations of motion will be used to analyze system controllability. Third, the equations of motion will be used to construct a linearized controller for trajectory tracking purposes. The last two will be done symbolically and the resulting symbolic material will be converted to appropriate code as necessary.

The dynamical formulation that will be used has not been determined yet. A significant part of the research will involve comparing the various approaches and choosing the most appropriate one to implement symbolically. Approaches to be compared include Newton-Euler, Lagrange equations (with Routh's extensions), Hamilton's canonical equations, Kane's equations, and spatial algebra/screw theory approaches.

To be suitable, the chosen technique of generating equations of motion must be suitable for symbolic implementation, and suitable for efficient simulations. The symbolic implementation should also apply typical methods to improve the efficiency of the code by doing such things as computing common subexpressions only once and by precomputing time–invariant terms.

An issue to be addressed is how to adapt existing recursive approaches to multibody dynamics in free-fall. These formulations are satisfactory for symbolic manipulations for the systems under consideration. Unfortunately, the numerical implementations generally depend on the angular velocity of the base remaining zero. In free-fall, this is not true. In fact, the angular velocity and position of the base depends on the motion of all the joints (due to the conservation of angular momentum.) Techniques to handle these unknown quantities during the recursions have not been described in the literature and will be studied in this research. This may lead to recursive formulations for the angular momentum of multibody systems in free-fall. (Note that this is not a problem for the symbolic use of recursive formulations since the unknown values are carried along as symbolic values in any case.)

3.2.5 Design optimal controls to accomplish motions

Since the idea is to precompute motions, it makes sense to compute the best possible motions. Therefore, optimal controls approaches will be used to plan the motions. It should be noted that motions produced by this approach will be optimal in some sense but that the main goal is to plan motions that are feasible and avoid extensive cyclic motions.

There are several possible approaches to be considered. An optimal control scheme based on variational analysis and the maximum principle is a logical candidate for computing the multibody motions. Appendix B gives a sample of the type of analysis proposed. The analysis actually used must satisfy several requirements. It must be simple enough and predictable to implement via symbolic manipulation. It must produce the system of executable equations (for example, state and costate equations) which are reasonably efficient. The analysis in the appendix is given to illustrate the type of approach proposed.

Appendix C illustrates the application of the optimal control scheme from Appendix B to an example.

Other approaches were mentioned in the literature survey. The technique described by Tan and Potts in "A Discrete Path/Trajectory Planner for Robotic Arms" is intended for fixed-base arms but is adaptable to this current problem [107]. It involves constructing a discrete non-linear model of the robot dynamics[68] which can then be used to construct a large non-linear programming problem. The approach is very flexible since it allows constraints on joint positions, velocities, jerks, and actuator limitations. It can avoid obstacles and will minimize a user specifiable cost function over the path.

Luus has devised another technique based on dynamic programming which is also applicable. In the recent paper "Application of Dynamic Programming to High-Dimensional Non-Linear Optimal Control Problems," Luus used dynamic programming to optimize several non-linear problems subject to input limitations. In one example, he studied a complex system with eight non-linear ordinary differential equations and determined optimum control input histories.

3.2.6 Implement symbolic generation of optimal control scheme

Once the optimal control scheme is designed, it must be implemented in terms of symbolic manipulations. An example of the type implementation to accomplish this is given in Appendix C. In this example, the optimal controls scheme outlined in Appendix B is implemented. Sample results in terms of state and costate equations are given for example systems. Example code is also shown which has been generated from these state and costate equations.

3.2.7 Precompute motions between selected configurations

In order to prepare the system for movement between configurations in various orientations, the necessary motions must be precomputed. The optimal control scheme must be applied to produce the movement data necessary for each motion. This will be implemented in the simulation environment.

The motion simulations will involve an extensive amount of computation and may require assistance from fast mainframe computers. One advantage of using X-Windows in a networked environment it is quite possible for the simulation environment to generate C code for the motion simulation, move this to a remote computer (perhaps a super computer), compile the code on that computer, run it on that computer, and return the data to the simulation environment without user interaction.

One premise of this research is that the amount of data generated by the motion simulations will not be over-whelming. To validate that assumption, it is necessary determine how much data will be generated for various situations. Appendix D contains a derivation of the number of data points that must be stored as a function of the various parameters. The resulting equation is:

$$N_{DP} = 3 N_O N_J N_D N_P^2 \tag{1}$$

where N_{DP} = Total number of data points required
N_O = Number of relative rotations
N_J = Number of joints or internal DOF
N_D = Number of data points per variable
(Number of time steps + 1)
N_P = Number of poses (or configurations)

To give some feel for the amount of data indicated by this equation, consider a few examples. Using two configurations and moderately optimistic values for the parameters in Equation 1, the amount of storage required for several cases are given in Table 1 (see Appendix D for details). The first example

Number of Joints, N_J	N_{DATA} (KB)
3	45.8
6	91.6
14	213.8

Table 1: Amount of Data Necessary to Store Motions

uses $N_J = 3$ and corresponds to a relatively simple mechanism. The second example uses $N_J = 6$ and corresponds to a six degree-of-freedom robot. The last example uses $N_J = 14$ and corresponds (roughly) to a human [134].

Although this is a large amount of data, it is within reason. Storing this amount of data on hard disks is quite feasible. Storing this amount of data in ROM is possible but N_P cannot be very large.

It may be possible to reduce the amount of data to be stored by only storing the joint positions during each motion. Joint velocities and joint torques can be computed on the fly by using recursive inverse dynamics formulations.

3.2.8 Adapt compression techniques to compress motion data

For each starting configuration and final configuration there will be three degrees of freedom in orientation that will be simulated. This can be thought of as a vector from the center of a sphere to some point on its surface plus an angle about that vector. One of the issues to be examined is how fine to subdivide these angles. The grid points must be close enough together so that interpolation between nearby motions will produce nearly correct results. This will be discussed further in the next section on the motion tracking control system. Unfortunately, increasing the number of divisions will tremendously affect the amount of number crunching necessary and the quantity of resulting data.

Since the motion simulation will produce a tremendous amount of data, an important component of this work will be how to compress it into a motion database (or library). Consider the plot for one joint position (or control input) over a motion. This is a single simple plot. Now consider a set of these for one of the degrees of freedom in orientation. Each plot of the joint position can be treated like a scan line of an image so that the set of plots can be thought of as an image. There are three degrees of freedom so the resulting data can be thought of as a two dimensional array of images. Since the motion data can be thought of as images, one approach to compressing this data is to apply image compression techniques. The current state of the art in image compression for exact reproduction is roughly 3:1 for typical images. In this case, exact reproduction is not necessary; techniques exist which give nearly lossless single image compressions of roughly 10:1 to 40:1 [32, 10]. When a number of similar images are compressed, further compression is possible by exploiting the similarity between the images—resulting in compression ratios of up to roughly 100:1. With this type of compression, it is possible to compress an extensive set of data into a reasonable amount of space. It is expected that the set of pseudo-images will be relatively similar so that compression techniques will be effective.

The image compression techniques described typically depend on the image being composed of integer data with limited range, for instance, 0-255. The joint position and control input data will typically be floating point. An issue to be addressed is what level of quantization will allow acceptable reconstruction of the joint and control profiles.

There may be a relationship between the type of compression scheme implemented and how the system will be used. If the motions are needed often and quickly (as it might be for planning), then retreival speed becomes an issue. The most effective image compression techniques depend on reconstructing the entire image at once. All that will be needed in this case is the equivalent of one scan line from several different images. Some compression techniques may be more efficient for retreiving one scan line at a time from an image (or set of images).

The computations for compression will be extensive. This is not necessarily a significant problem for an actual application since video compression hardware exists today which can do the necessary compression at video frame rates.

3.2.9 Design linearized motion tracking control scheme

The process of compression means the reconstructed joint position and control input profiles will not be exactly what they should be. Also, there will be uncertainties in the parameters of the actual system. Given the joint position and control input profiles necessary to accomplish some configuration and orientation change how can we persuade the system to actually complete the desired motion? Obviously some type of trajectory tracking controller will be necessary. There are a number of possibilities here. One is a time-varying linearized control system. Another approach is feedback linearization. In the research, various options will be examined.

The controllability of the time-varying linearized system is an important issue that will be addressed. In a sense, the linearized control system controls the motions in the small at any instant. It will not be fully controllable (in general) since it cannot command motions that violate the angular momentum constraint.

A resonable approach (if the mechanism is suitable) is to reduce the order of the system used to determine the planned motion (for instance, by freezing some of the joints). Then, during the motion tracking phase, the linearized controller can use those joints to keep the system close to the desired motion.

3.2.10 Implement symbolic generation of linearized controller

Once the form of the time-varying linearized controller is designed, it should not be difficult to use symbolic manipulation to apply it to the equations of motion. In this way, two implementation problems can be addressed via symbolic manipulation. First, the system will generate C code to implement the linearized feedback control law. This code will not vary during the motion. Second, the system will generate C code to compute the time-varying data necessary for the linearized controller. This code will be run as necessary to update the data in the linearized feedback control code.

3.2.11 Use simulation to verify linearized controller

To test the motion data libraries and linearized controller, the system will use perform simulations. The system will use standard multibody simulation techniques with joint actuator inputs from the linearized controller and motion libraries. These simulations will test many phases of the research. They will also give a feel for what kind of accuracy and resolution is necessary in the motion database to give adequate control with the linearized controller.

3.2.12 Apply system to example multibody systems

To illustrate use of the system and to test it, it will be applied to several example multibody systems in free-fall. Useful examples include two body systems, typical space robots, and simplified human models. Although human motion in free-fall is a desirable application, it may be too ambitious for initial applications due to its large number of degrees of freedom.

3.3 Expected Results and Contributions

It should be noted that no single piece of this research is revolutionary. At most small extensions from the state of the art are proposed. What makes this research unique is the way the components are put together. Nobody has yet successfully addressed the end-to-end problem of how to control multibody systems in free-fall in real time. This will be the primary contribution of this research.

Other contributions will include:

- Extending recursive multibody formulations for numerical simulations of systems in free-fall.
- Embedding the multibody tree structure in the software objects created to model it.
- Construction of a flexible, powerful, and portable simulation environment which can be applied to real motion problems.
- Design and implementation of optimal control for configuration change. Using symbolic manipulation to construct the optimal controller.
- Using image compression techniques to compress motion data.
- Storing precomputed motions for complex systems for later use.
- Using symbolic manipulation to implement the time-varying linearized motion tracking controller.
- Use of symbolic manipulation for dynamics and controls in one integrated system.

References

[1] J. S. Albus, "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)," *Journal of Dynamic Systems, Measurement, and Control*, Vol. 97, 1975, pp. 1975.

[2] J. S. Albus, "Data Storage in the Cerebellar Model Articulation Controller (CMAC)," *Journal of Dynamic Systems, Measurement, and Control*, Vol. 97, 1975, pp. 228–233.

[3] C.A. Balafoutis, P. Misrta, and R.V. Patel, "A Cartesian Tensor Approach for Fast Computation of Manipulator Dynamics," *Proceedings of the IEEE International Conference on Robotics and Automation*, April 1988, pp. 1348–1353.

[4] R.S. Ball, *A Treatise on the Theory of Screws*, Cambridge University Press, London, 1900.

[5] Jérome Barraquand and Jean-Claude Latombe, "On Nonholonomic Mobile Robots and Optimal Maneuvering," *Proceedings of the IEEE International Symposium on Intelligent Control 1989*, Albany, NY, September, 1989, pp. 340–347.

[6] J.E. Bobrow, S. Dubowsky, and J.S. Gibson, "Time-Optimal Control of Robotic Manipulators Along Specified Paths," *International Journal of Robotics Research*, Vol. 4, 1985, pp. 3–17.

[7] J.W. Burdick, "An Algorithm for Generation of Efficient Manipulator Dynamics," *Proceedings of the IEEE International Conference on Robotics and Automation*, 1986, pp. 212–218.

[8] Mark A. Bronez, Margaret M. Clarke, Alberta Quinn, "Requirements Development for a Free-Flying Robot—the "Robin"," *Proceedings of the IEEE International Conference of Robotics and Automation*, San Francisco, California, 1986.

[9] A.E. Bryson and Y.-C. Ho, *Applied Optimal Control*, John Wiley and Sons, New York, 1975.

[10] Jonathan M. Cameron, *Survey of the State of the Art in Image Compression for Low Data-Rate Remote Driving*, EM 347-90-279, Jet Propulsion Laboratory, Pasadena, California, September 12, 1990.

[11] G. Cesareo, F. Nicolo, and S. Nicosia, "DYMIR: A Code for Generating Dynamic Models of Robots," *Proceedings of the IEEE International Conference on Robotics*, Atlanta, Georgia, 1984, pp. 115–120.

[12] S. Cetinkunt, and W. J. Book, "Symbolic Modeling of Flexible Manipulators," *Proceedings of the IEEE International Conference on Robotics and Automation*, Raleigh, North Carolina, April 1987, pp. 2074–2081.

[13] C.H. Chen and A.C. Kak, "A Robot Vision System for recognizing 3-D Objects in Low-Order Polynomial Time," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 19, No. 6, Nov./Dec. 1989, p. 1535–1563.

[14] John J. Craig, *Introduction to Robotics, Mechanics and Control*, Second Edition,

Addison-Wesley, 1986, 1989.

[15] X. Cyril, J. Angeles, and A. Misra, "Efficient Inverse Dynamics of General N-Axis Robotic Manipulators," *Proceedings of the 9th Symposium on Engineering Applications in Mechanics*, May 1988, pp. 595–602.

[16] Morris R. Direls, "Symbolic Matrix Manipulation Package for the Kinematic Analysis of Robot Manipulators," *Computers in Mechanical Engineering*, Vol. 5, No. 2, September 1986, pp. 38–46.

[17] S. Dubowsky, M.A. Norris, and Z. Shiller, "Time-Optimal Trajectory Planning for Robotic Manipulators with Obstacle Avoidance: a CAD Approach," *Proceedings of the IEEE International Conference on Robotics and Automation*, 1986, pp. 1906–1912.

[18] H. Faessler, "Computer-Assisted Generation of Dynamical Equations for Multibody Systems," *International Journal of Robotics Research*, Vol. 5, No. 3, Fall 1986, pp. 129–141.

[19] Roy Featherstone, *Robot Dynamics Algorithms*, Kluwer Academic Publishers, Boston, 1987.

[20] O. Fisher, *Einführung in die mechanik lebender mechanismen (Introduction to the Mechanics of Living Organisms)*, Leipzig, Germany, 1906.

[21] Cliff Frohlich, "Do Springboard Divers Violate Angular Momentum Conservation?" *American Journal of Physics*, Vol. 47, No. 7, July 1979.

[22] H.P. Gearing et al., "Time-Optimal Motions of Robots in Assembly Tasks," *IEEE Transactions on Automatic Control*, Vol. 31, 1986, pp. 512–518.

[23] E.G. Gilbert and D.W. Johnson, "Distance Functions and Their Applications to Robot Path Planning in the Presence of Obstacles," *IEEE Journal of Robotics and Automation*, Vol. 1, 1985, pp. 21–30.

[24] Arthur L. Hale and Leonard Meirovitch, "Derivation of the Equations of Motion for Complex Structures by Symbolic Manipulation," *Computers and Structures*, Vol. 9, December 1978, pp. 639–649.

[25] Xiaogeng He and A.A. Goldenberg, "An Algorithm for Efficient Computation of Dynamics of Robotic Manipulators," *Journal of Robotic Systems*, Vol. 7, No. 5, 1990, pp. 689–702.

[26] Wolfgang Hirschberg and Dieter Schramm, "Application of NEWEUL in Robotic Dynamics," *Journal of Symbolic Computation*, Vol. 7, 1989, pp. 199–204.

[27] John M. Hollerbach, "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, November 1980, pp. 730–736.

[28] M.A. Hussain and B. Noble, "Application of Symbolic Computation to the Analysis of Mechanical Systems, Including Robot Arms," *NATO ASI Series on Computer*

Aided Analysis and Optimization of Mechanical System Dynamics. Iowa City, Iowa, 1984, pp. 283–304.

[29] A. Izaguirre and R. Paul. "Automatic Generation of Dynamic Equations of the Robot Manipulators Using a LISP Program." *Proceedings of the IEEE International Conference on Robotics and Automation*. San Francisco, California, 1986, pp. 220–226.

[30] Paul Jacobs and John Canny. "Planning Smooth Paths for Mobile Robots." *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*. May 1989, pp. 2–7.

[31] Paul Jacobs and John Canny. "Robust Motion Planning for Mobile Robots." *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*. 1990, pp. 2–7.

[32] A. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, 1989.

[33] Lyle M. Jenkins. "Telerobotic Work System--Space Robotics Application." *Proceedings of the IEEE International Conference of Robotics and Automation*. 1986.

[34] M.E. Kahn and B. Roth. "The Near-Minimum-Time Control of Open-Loop Articulated Kinematic Chain." *Journal of Dynamic Systems, Measurement, and Control (Transactions ASME)*. Vol. 93, 1971, pp. 164–172.

[35] T.R. Kane and M.P. Scher. "A Dynamical Explanation of the Falling Cat Phenomenon." *International Journal of Solids and Structures*. Vol. 5, 1969, pp. 663–670.

[36] T.R. Kane and M.P. Scher. "Human Self-Rotation by Means of Limb Motions." *Journal of Biomechanics*. Vol. 3, 1970, pp. 39–49.

[37] T.R. Kane, M.R. Headrick, and J.D. Yatteau. "Experimental Investigation of an Astronaut Maneuvering Scheme." *Journal of Biomechanics*. Vol. 5, 1972, pp. 313–320.

[38] K. Kazerounian and K.C. Gupta. "Manipulator Dynamics Using the Extended Zero Reference Position Description." *IEEE Transactions of Robotics and Automation*. Vol. RA-2, 1986, pp. 221–224.

[39] W. Khalil, J.F. Kleinfinger, and M. Gautier. "Reducing the Computational Burden for the Dynamical Models for Robots." *Proceedings of the IEEE International Conference on Robotics and Automation*. 1986, pp. 525–531.

[40] Byung Kook Kim and Kang G. Shin. "Minimum-Time Path Planning for Robot Arms and Their Dynamics." *IEEE Transactions on Systems, Man, and Cybernetics*. Vol. SMC-15, No. 2, March/April 1985, pp. 213–223.

[41] M. Kircanski and M. Vukobratovic. "Computer-Aided Generation of Manipulator Kinematic Models in Symbolic Form." *15th International Symposium on Industrial Robots*. 1985, pp. 1043–1049

[42] D.M. Klimov, V.M. Rudenko, and V.F. Zhuravlev. "Application of Lie Group and

Computer Algebra to Nonlinear Mechanics." *Proceedings of the European Conference of Computer Algebra. EUROCAL '87*. Leipzig, Germany, June 1978. Lecture Notes in Computer Science No. 378. Springer-Verlag. 1987

[43] E.J. Kreuzer, "Dynamic Analysis of Mechanisms Using Symbolic Equation Manipulation," *Proceedings of the Fifth World Congress on Theory of Machines and Mechanisms (ASME)*. 1979, pp. 599–602.

[44] Edwin J. Kreuzer and Werner O. Schiehlen, "Generation of Symbolic Equations of Motion for Complex Spacecraft using Formalism NEWEUL," *Advances in the Astronautical Sciences*, Vol. 54, Part 1, 1983, pp. 21–36.

[45] P.S. Krishnaprasad, "Geometric Phases and Optimal Reconfiguration for Multibody Systems," *Proceedings of the 1990 American Control Conference*, 1990, pp. 2440–2444.

[46] Jean-Paul Laumond, "Feasible Trajectories for Mobile Robots with Kinematic and Environment Constraints," *Proceedings of the IEEE International Conference on Intelligent Autonomous Systems*, December 1987, Amsterdam, pp. 346–354.

[47] Jean-Paul Laumond, "Finding Collision-Free Smooth Trajectories for a Non-Holonomic Mobile Robot," *10th International Joint Conference on Artificial Intelligence*, Milano, Italy, 1987, pp. 1120–1123.

[48] C.S.G. Lee and P.R. Chang, "Efficient Parallel Algorithms for Robot Forward Dynamics Computations," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 18, No. 2, March/April 1988.

[49] M.C. Leu and N. Hemati, "Automated Symbolic Derivation of Dynamic Equations of Motion for Robotic Manipulators," *Journal of Dynamic Systems, Measurement and Control, Transactions ASME*, Vol. 108, No. 3, September 1986, pp. 172–179.

[50] D. A. Levinson, "Equations of Motion for Multiple Rigid-Body Systems via Symbolic Manipulation," *Journal of Spacecraft and Rockets*, Vol. 14, No. 8, August 1977, pp. 479–87.

[51] Z. Li and J.F. Canny, *Robot Motion Planning with Non-Holonomic Constraints*, Memo UCB/ERL M89/13, Electronics Research Laboratory, University of California, Berkeley, California, 1989.

[52] A. Liegois, W. Khalil, J. Dumas, and M. Renaud, "Mathematical and Computer Models of Interconnected Mechanical Systems," *Proceedings of the Second International CISM-IFToMM Symposium*, Warsaw, September 1976, pp. 5–17.

[53] Kathryn W. Lilly and David E. Orin, "Efficient O(N) Computation of the Operational Space Inertia Matrix," *Proceedings of the IEEE International Conference on Robotics and Automation*, 1990, pp. 1014–1019.

[54] R.E. Lindberg, R.W. Longman, M.F. Zedd, "Kinematics and Reaction Moment Compensation for a Space-Borne Elbow Manipulator," *Proceedings, 24th AIAA Aerospace Sciences Meeting*, Reno, Nevada, 1986.

[55] John Lloyd and Vincent Hayward, "Kinematics of Common Industrial Robots,"

Robotics 4, 1988, pp. 169–191.

[56] B.A. Logan, Jr., "Space Station Remote Manipulator Requirements Definition," *23rd AIAA Aerospace Sciences Meeting*., Reno, Nevada, January, 1985.

[57] Richard W. Longman, Robert E. Lindberg, Michael F. Zedd, "Satellite Mounted Robot Manipulators — New Kinematics and Reaction Moment Compensation," *Proceedings, AIAA Guidance, Navigation, and Control Conference*, New York, NY, 1985.

[58] Richard W. Longman, Robert E. Lindberg, and Michael F. Zedd, "Satellite Mounted Robot Manipulators — New Kinematics and Reaction Moment Compensation," *International Journal of Robotics Research*, Vol. 6, No. 3, Fall 1987, pp. 87–103.

[59] P. Lucibello, F. Nicolò and R. Pimpinelli, "Automatic Symbolic Modeling of Robots with a Deformable Link," *IFAC Theory of Robots*, 1986, pp. 131–135.

[60] J.Y.S. Luh, M.W. Walker, and R.P.C. Paul, "On-Line Computational Scheme for Mechanical Manipulators," *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 102, 1980, pp. 69–76.

[61] Rein Luus, "Application of Dynamic Programming to High-Dimensional Non-Linear Optimal Control Problems," *International Journal of Control*, Vol. 52, No. 1, 1990, pp. 239–250.

[62] G. A. Macala, "SYMBOD: A Computer Program for the Automatic Generation of Symbolic Equations of Motion for Systems of Hinge-Connected Rigid Bodies," *AIAA Paper No. 83-0013*, Presented at the 21st AIAA Aerospace Sciences Meeting, Reno, Nevada, January 1983.

[63] K. Magnus, *Dynamics of Multibody Systems*, Springer-Verlag, Berlin, 1978.

[64] R.V. Mayorga and A.K.C. Wong, "A Global Approach for the Optimal Path Generation of Redundant Robot Manipulators," *Journal of Robotic Systems*, Vol. 7, No. 1, 1990, pp. 107–128.

[65] B.C. McInnis and C.K.F. Liu, "Kinematics and Dynamics of Robotics: A Tutorial Based on Classical Concepts of Vectorial Mechanics," *IEEE Transactions of Robotics and Automation*, Vol. RA-2, 1986, pp. 181–187.

[66] J.J. Murray and C.P. Neumann, "ARM: An Algebraic Robot Dynamic Modeling Program," *Proceedings of the IEEE International Conference on Robotics and Automation*, Atlanta, Georgia, 1984, pp. 103–114.

[67] Richard M. Murray and S. Shankar Sastry, "Steering Nonholonomic Systems Using Sinusoids," *Proceedings of the 20th Conference on Decision and Control*, Honolulu, Hawaii, December 1990, pp. 2097–2101.

[68] Charles P. Neuman and Vassilios D. Tourassis, "Discrete Dynamic Robot Models," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-15, No. 2, March/April 1985, pp. 193–204.

[69] P.E. Nielan and T.R. Kane, "Symbolic Generation of Simulation/Control Routines for Multibody Systems," *Dynamics of Multibody Systems*, IUTAM/IFToMM Symposium, Udine, Italy, 1986, pp. 153-164.

[70] H. Nijmeijer and A.J. Van der Schaft, *Nonlinear Dynamical Control Systems*, Springer-Verlag, New York, 1990.

[71] Yoshihiko Nakamura and Ranjan Mukherjee, "Nonholonomic Path Planning of Space Robots," *Proceedings of the IEEE International Conference on Robotics and Automation*, 1989, pp. 1050-1055.

[72] Yoshihiko Nakamura and Ranjan Mukherjee, "Nonholonomic Path Planning of Space Robots via Bi-directional Approach," *Proceedings of the IEEE International Conference on Robotics and Automation*, 1990, pp. 1764-1769.

[73] Yoshihiko Nakamura and Ranjan Mukherjee, "Bi-directional Approach for Nonholonomic Path Planning of Space Robots," *Proceedings, 5th International Symposium of Robotics Research*, August 28-31, 1989, Tokyo, Japan, pp. 101-112.

[74] Yoshihiko Nakamura, Hideo Hanafusa, and Tsuneo Yoshikawa, "Task-Priority Based Redundance Control of Robot Manipulators," *International Journal of Robotics Research*., Vol. 6, No. 2, Summer 1987, pp. 3-15.

[75] Yoshihiko Nakamura and Hideo Hanafusa, "Optimal Redundancy Control of Robot Manipulators," *International Journal of Robotics Research*, Spring 1987, Vol. 6, No. 1, pp. 32-42.

[76] José Lopes de Siqueira Neto, Antonio Eduardo Costa Pereira, and João Bosco da Mota Alves, "Symbolic Computation Applied to Robot Dynamic Modeling," *Proceedings of the 16th International Symposium on Industrial Robotics*, 1986, pp. 389-400.

[77] C.P. Neumann and J. Murray, " Symbolically Efficient Formulations for Computational Robot Dynamics," *Journal of Robotic Systems*, Vol. 4, No. 4, 1987, pp. 503-526.

[78] M. Niv and D.M. Auslander, "Optimal Control of a Robot with Obstacles," *Proceedings of the American Control Conference*, 1984, pp. 280-287.

[79] J. M. Ortega, *Matrix Theory*, Plenum Press, New York, 1987.

[80] E. Papadopoulos and S. Dubowsky, "On the Dynamic Singularities in the Control of Free-Floating Space Manipulators," *Dynamics and Control of Multibody/Robotic Systems with Space Applications*, Ed. S.M. Joshi, L. Silverberg, and T.E. Alberts, Presented at the Winter Annual Meeting of the American Society of Mechanical Engineers, San Francisco, California, December, 1989, pp. 45-52.

[81] Marc H. Raibert, "Analytical Equations vs. Table Look-Up for Manipulation: A Unifying Concept," *Proceedings of the IEEE Conference on Decision and Control*, New Orleans, December 1977, pp. 576-579.

[82] Marc H. Raibert, et al., *Dynamically Stable Legged Motion*, Technical Report CMU-RI-TR-83-1, Robotics Institute, Carnegie-Mellon University, January 1983.

[83] Marc H. Raibert, Francis C. Wimberly, "Tabular Control of Balance in a Dynamic Legged System," *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, 1983.

[84] V.T. Ranjan, "Minimum Time Trajectory Planning," *Proceedings of the IEEE International Conference on Robotics and Automation*, 1985, pp. 759–764.

[85] R.E. Roberson and R. Schwertassek, *Dynamics of Multibody Systems*, Springer-Verlag, Berlin, 1988.

[86] G. Rodriguez, K. Kreutz, and A. Jain, "A Spatial Operator Algebra for Manipulator Modeling and Control," *Proceedings of the IEEE International Conference on Robotics and Automation*, 1989, pp. 1374–1379.

[87] G. Rodriguez and K. Kreutz, *Recursive Mass Matrix Factorization and Inversion: An Operator Approach to Open- and Closed- Chain Multibody Dynamics*, Jet Propulsion Laboratory Publication, 88–11, March 1988.

[88] D.E. Rosenthal and M.A. Sherman, "High Performance Multibody Simulations via Symbolic Equation Manipulation and Kane's Method," *Journal of the Astronautical Sciences*, Vol. 34, No. 3 July–September, 1986, pp. 223–239.

[89] G. Sahar and J.M. Hollerbach, "Planning of Minimum-Time Trajectories for Robot Arms," *International Journal of Robotics Research*, Vol. 5, 1986, pp. 90–100.

[90] David B. Schaechter and David A. Levinson, "Interactive Computerized Symbolic Dynamics for the Dynamicist," *The Journal of the Astronautical Sciences*, Vol. 36, No. 4, October–December 1988, pp. 365–388.

[91] V. Scheinman and B. Roth, "On the Optimal Selection and Placement of Manipulators," *Proceedings, 5th International CISM-IFToMM Symposium*, 1984, pp. 39–46.

[92] W.O. Schiehlen and E.J. Kreuzer, "Symbolic Computerized Derivations of Equations of Motion," *Proceedings of the IUTAM Symposium on Dynamics of Multibody Systems*, Munich, 1977, Springer-Verlag, 1978, pp. 290–305.

[93] Ahmed A. Shabana, *Dynamics of Multibody Systems*, John Wiley and Sons, New York, 1989.

[94] Michael Sherman, "Control System Verification Using Real Time Execution of Symbolically-Generated Multibody Equations of Motion," *Proceedings of the American Control Conference*, 1988, pp. 595–601.

[95] M. A. Sherman, "The Practical Application of Symbolic Manipulation to Multibody Dynamics," *SDIO/NASA Workshop on Multibody Simulations*, Jet Propulsion Laboratory, Pasadena, California, September, 1987.

[96] M. A. Sherman, "SD/EXACT and SD/FAST Symbolic Multibody Codes," *SDIO/NASA Workshop on Multibody Simulations*, Jet Propulsion Laboratory, Pasadena, California, September, 1987.

[97] K.G. Shin and N.D. McKay, "Minimum-Time Control of Robotic Manipulators with Geometric Path Constraints," *IEEE Transactions on Automatic Control*, Vol. 30,

No. 6, June 1985, pp. 531–541.

[98] K.G. Shin and N.D. McKay, "Selection of Near-Minimum Time Geometric Paths for Robotic Manipulators," *Proceedings of the American Control Conference*, 1985, pp. 346–355.

[99] Jean-Jacques E. Slotine and Weiping Li, *Applied Nonlinear Control*, Prentice Hall, Englewood Cliffs, New Jersy, 1991.

[100] E.D. Sontag and H.J. Sussman, "Time-Optimal Control of Manipulator," *Proceedings of the IEEE International Conference on Robotics and Automation*, 1985, pp. 1643–1652

[101] *Spacecraft Servicing Demonstration Plan*, National Aeronautics and Space Administration, Marshall Space Flight Center, MCR 84-1866, NAS8-35496, July, 1984.

[102] N. Sreenath, "Nonlinear Control of Multibody Systems in Shape Space," *IEEE International Conference on Robotics and Automation*, 1990, pp. 1776–1781.

[103] N. Sreenath, "Phase Space Analysis of Multibody Systems Using Lie-Transforms Approach," *Proceedings of the 1990 American Control Conference*, 1990, pp. 2446-2447.

[104] David Stoutemyer, *Computer Algebraic Manipulation for the Calculus of Variations, the Maximum Principle, and Automatic Control*, ALOHA Project Technical Report, University of Hawaii, 1979.

[105] Suk-Hwan Suh and Kang. G. Shin, "Robot Path Planning with Distance-Safety Criterion," *Proceedings of the 20th IEEE Conference on Decision and Control*, 1987, pp. 634–641.

[106] H.H. Tan and R.B. Potts, "Minimum Time Trajectory Planner for the Discrete Dynamic Robot Model with Dynamic Constraints," *IEEE Journal of Robotics and Automation*, Vol. 4, No. 2, 1988, pp. 174–185.

[107] H.H. Tan and R.B. Potts, "A Discrete Path/Trajectory Planner for Robotic Arms," *Journal of the Australian Mathematical Society*, Series B, Vol. 31, 1989, pp. 1–28.

[108] H.H. Tan and R.B. Potts, "A Discrete Trajectory Planner for Robotic Arms with Six Degrees of Freedom," *IEEE Transactions on Robotics and Automation*, Vol. 5, No. 5, October 1989, pp. 681–690.

[109] R.W. Toogood, *Symbolic Generation of Robot Dynamics Equations, Part I: The DYNIM/CLEAR System*, ACMIR TR-87-05, University of Alberta, 1987.

[110] R.W. Toogood, *Symbolic Generation of Robot Dynamics Equations, Part II: Case Studies Using the DYNIM/CLEAR System*, ACMIR TR-87-06, University of Alberta, 1987.

[111] R.W. Toogood, *Robot Direct Dynamics Algorithms Using Symbolic Generation*, ACMIR TR 88-??, University of Alberta.

[112] R.W. Toogood, "Efficient Robot Inverse and Direct Dynamics Algorithms Using Micro-Computer Based Symbolic Generation," *Proceedings of the IEEE Conference*

on Robotics and Automation. 1989, pp. 1827–1832.

[113] E. Tunstel and N. Vira, "Mechanization of Manipulator Kinematic Equations via MACSYMA," *Proceedings of the IEEE International Computers in Engineering Conference and Exhibit (ASME)*, 1989, pp. 649–655.

[114] L.I. Tyves and S.V. Markevich, "Robot-Motion Path-Planning Algorithm with Dynamic Properties of Actuator Models," *Soviet Machine Science (Trans. Mashinovedenie)*, No. 4, 1987, pp. 27–34.

[115] Anthony P. Tzes, Stephan Yurkocich, and F. Dieter Langer, "A Symbolic Manipulation Package for Modeling of Rigid or Flexible Manipulators," *Proceedings of the IEEE International Conference on Robotics and Automation*, 1988, pp.1526–1531.

[116] Yoji Umetani and Kazuya Yoshida, "Continuous Path Control of Space Manipulators Mounted on OMV," *Acta Astronautica*, Vol. 15, No. 12, pp. 981–986, 1987.

[117] Ziaeddin Vafa, *The Kinematics, Dynamics and Control of Space Manipulators: The Virtual Manipulator Concept*, Ph.D. Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, November 1987.

[118] Z. Vafa and S. Dubowsky, "Kinematic and Dynamic Models of Manipulators for Use in Space: The concept of the Virtual Manipulator," *Proceedings of the 7th World Congress on Theory of Machines and Mechanisms*, 1987, Vol. 2, pp. 1233–1236.

[119] Z. Vafa and S. Dubowsky, "On the Dynamics of Manipulators in Space Using the Virtual Manipulator Approach," *Proceedings of the IEEE International Conference on Robotics and Automation*, 1987, pp. 579–585.

[120] Z. Vafa, "Space Manipulator Motions with No Satellite Attitude Disturbances," *Proceedings of the IEEE International Conference on Robotics and Automation*, 1990, pp. 1770–1775

[121] Z. Vafa and S. Dubowsky, "The Kinematics and Dynamics of Space Manipulators: the Virtual Manipulator Approach," *International Journal of Robotics Research*, Vol. 9, No. 4, August, 1990.

[122] L. Vecchio, S. Nicosia, F. Nicolò, and D. Lentini, "Automation Generation of Dynamical Models of Manipulators," *Proceedings of the 10th International Symposium on Industrial Robots*, Milan, Italy, March 1980, pp. 293–301.

[123] L.T. Wang and B. Ravani, "Recursive Computations of Kinematic and Dynamics Equations for Mechanical Manipulators," *IEEE Journal of Robotics and Automation*, Vol. RA-1, 1985, pp. 124–131.

[124] M.W. Walker and D. Orin, "Efficient Dynamic Computer Simulation of Robotic Mechanisms," *ASME Journal of Dynamic Systems, Measurement and Control*, Vol. 104, 1982, pp. 205–211.

[125] Robert S. Welch, "The Fosbury Flop is Still a Big Hit," *Sports Illustrated*, Vol. 69, No. 11, Sept. 12, 1988, pp. 12–15.

[126] John Wen and Alan A. Desrochers, "An Algorithm for Obtaining Bang-Bang Con-

trol Laws." *Journal of Dynamic Systems, Measurement, and Control*, June 1987, Vol. 109, pp. 171–175.

[127] John Wen, "Existence of the Time Optimal Control for Robotic Manipulators," *Proceedings of the American Control Conference*, 1986, pp. 109–113.

[128] J. Wittenburg and U. Wolz, "MESA VERDE: A Symbolic Program for Nonlinear Articulated Rigid Body Dynamics," *ASME Paper No. 85-DET-151*, 1985.

[129] J. Wittenburg and U. Wolz, "A Computer Program for the Alpha-Numerical Generation of Robot Dynamics Equations," *Proceedings of the 5th CISM-IFToMM Symposium on the Theory and Practice of Robots and Manipulators*, Udine, Italy, June, 1984, MIT Press, 1985.

[130] J. Wittenburg, *Dynamics of Systems of Rigid Bodies*, B.G. Teubner, Stuttgart, 1977.

[131] Rui Yang and P.S. Krishnaprasad, "On the Dynamics of Floating Four-Bar Linkages," *Proceedings of the 28th IEEE Conference on Decision and Control*, Tampa, Florida, December 1989, pp. 1632–1637.

[132] M. R. Yeadon, "The Simulation of Aerial Movement—I. The Determination of Orientation Angles from Film Data," *Journal of Biomechanics*, Vol. 23, No. 1, 1990, pp. 59–66.

[133] M. R. Yeadon, "The Simulation of Aerial Movement—II. A Mathematical Inertia Model of the Human Body," *Journal of Biomechanics*, Vol. 23, No. 1, 1990, pp. 67–74.

[134] M. R. Yeadon, "The Simulation of Aerial Movement—III. The Determination of the Angular Momentum of the Human Body," *Journal of Biomechanics*, Vol. 23, No. 1, 1990, pp. 75–83.

[135] M. R. Yeadon, "The Simulation of Aerial Movement—IV. A Computer Simulation Model," *Journal of Biomechanics*, Vol. 23, No. 1, 1990, pp. 85–89.

[136] S. Yin and J. Yuh, "An Efficient Algorithm for Automatic Generation of Manipulator Dynamic Equations," *Proceedings of the IEEE International Conference on Robotics and Automation*, 1989, pp. 1812–1817.

[137] S. Yin and J. Yuh, "A User-Friendly PC Program for Symbolic Robot Dynamic Equations: ARDEG," *Proceedings of the International Computers in Engineering Conference and Exposition*, 1989, pp. 643–648.

Appendix A. How the Simulation Environment Might Be Used

The following description explains how the simulation environment might be used in a step-by-step manner. This sequence described here is not the only way the system can be used, but does illustrate the basic components of the simulation system.

1. **User constructs system in simulation environment.** The starting point of analyzing the motion of the multibody system is for the user to model the multibody system to be analyzed in the simulation environment. This could be done by direct manipulation (on the computer screen) or by reading an appropriate data file. To specify the multibody mechanism by direct manipulation, the user will select the links from a catalog of link shapes, specify (or modify) the link's geometric and inertia properties, indicate where any joints would be located, and what other links are attached to each joint. The system would then create software objects to model the links and joints. Note that this would automatically establish the connectivity (or tree structure) of the system. The software for each object would know how to construct the relevant transformations (symbolically and numerically) to determine the robot kinematics. Similarly, the object's code would also know how to add their components (symbolically and numerically) to recursive dynamic formulations.

2. **User chooses typical configurations** At this point, the user will choose *typical configurations or poses* for the multibody system. The typical poses will be selected to put the multibody system in various useful or desirable configurations. For example, if the system is a human, a typical configuration might be a straight body with arms extended. In [21], Cliff Frohlich gives nine different human body configurations that are commonly used by divers and trampolinists. Whether the body is upside down or rightside up is not important in specifying the configuration. The typical poses will probably include only configuration information (such as joint positions) and will not include joint velocities. Including initial and final velocities in the typical con-

figurations will increase computation and storage requirements by an unreasonable degree. The system will be able to plan individual motions with initial and final velocities but these will not be included as part of the precomputation part of the research. This may limit the usefulness of the precomputation and storage aspect of the research to human motion since large initial and final velocities are often part of athletic motions.

The typical configurations may also be selected to simplify the system dynamics. For instance, the pose might put the wrists into a neutral position. During the planned motion, these joints might not be used to reduce the dimensionality of the problem.

3. **System automatically constructs equations of motion.** After the description of the system is entered, the system will generate the equations of motion in symbolic form. The simulation system will be able to deal with the dynamics of the system in three ways. First, it will be able to simulate the multibody system dynamics directly using standard recursive approaches. Second, it will be able to generate the equations of motion in a symbolic form. Third, it will be able to simulate the multibody dynamics by using software code generated from the symbolic representations of the equations of motion.

4. **System automatically generates optimal controls.** Once the equations of motion are generated in symbolic form, the optimal control law for reconfiguration will be derived symbolically. This is why it is important to generate the equations of motion in symbolic form. This will also include generation of code to verify whether the motion is possible (from the analysis of nonlinear controllabiltiy and reachability analysis).

5. **System simulates optimal motions.** Once the equations of motion have been generated and the optimal control scheme has been constructed, these will be used to simulate optimal motions for orientation changes between the typical configurations. During each simulation the basic data of control inputs and states during the motion will be saved. The goal is to simulate the motions from any configuration to any other configuration in any other orientation where such motions

are possible. Hopefully, the number of typical configurations will be small. (If the number of typical configurations is large, the amount of computation and resulting data may be excessive.)

6. **System compresses the resulting simulation data into library of motion data.** The preceding step will generate a large amount of data. The resulting motion data will compressed using image compression techniques to construct a database of motion data (or motion library).

7. **System designs a linearized controller to execute the maneuvers in the motion library.** After the system constructs the motion library, the library can be used to plan and execute motions. However, the data compression techniques will introduce some errors into the reconstructed motion profiles due to quantization and other effects. This, along with imperfect modeling, indicate a motion tracking controller will be necessary. At this point, the simulation environment will use the symbolic version of the equations of motion to symbolically generate the necessary linearized controller to allow the system to execute a retreived motion profile. This linearized controller will be converted from symbolic form to usable software code for simulation purposes.

8. **User uses system to simulate various motions.** Once the previous steps have been completed, the simulation environment can be used to simulate motions and test the motion tracking controller. This could be the goal of the entire system. Consider how such a system could be used:

   - The user could simulate possible motions just to see what they look like and what types of control inputs are necessary.
   - The system could be used to verify the linearized controller by doing simulations with precomputed motion data and a perturbed system. The control inputs could go into a true dynamic simulation to verify the results.
   - A diving coach could use the system to construct a new dive sequence and show it to divers in a movie form. This would involve using several intermediate poses and splicing together the

necessary motions. (It might also involve generating specific new motions or new motion libraries.) The system would take the sets of joint position profiles and construct a longer sequence to show how the motion would look. Individual joint motions could be isolated to show the diver what to do and when. It is even possible that this system could be used to discover maneuvers that have never been thought of before.

- Similar techniques could be used with astronauts to train them to do combined configuration and orientation maneuvers.
- The system could be used to construct a motion library and tracking controller for an orbital servicing robot. This might involve generating new motion libraries tailored to specific tasks. The resulting data and code could be put into ROM for use on orbit.

One of the goals of this work is to reduce the amount of user interaction necessary. Ideally, steps 1 and 2 would be the only steps the user would have to supervise. In reality, some interaction will probably be necessary during some of the other steps as well.

Appendix B. Sample Optimal Control Analysis

Suppose that we have a nonlinear system such as a multibody robot in free-fall and we would like to choose a set of control inputs to move the system from one configuration to another. Start by putting the equations of motion of the system into the following form (this can be done symbolically from the equations of motion):

$$\frac{d}{dt}\mathbf{x} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u} \tag{2}$$

where $\mathbf{x}$ is a vector describing the state and velocities of the multibody system and $\mathbf{u}$ is a vector of available control inputs (such as control torques at each joint). This form was chosen since the equations of motion for multibody systems can generally be put into this form.

The initial configuration, $\mathbf{x}(t_0) = \mathbf{x}_0$, is known and the goal is to use the control inputs to move the system into the final configuration, $\mathbf{x}(t_f) = \mathbf{x}_f$, (where t_f is unspecified). The requirement that the system acheive the desired terminal state can be formulated in the terminal constraint:

$$\boldsymbol{\Psi}[\mathbf{x}(t_f), t_f] = \mathbf{x} - \mathbf{x}_f = 0 \tag{3}$$

The terminal constraint is adjoined to the terminal cost (which is zero so far) by using a vector of constant multipliers, $\boldsymbol{\nu}$:

$$\Phi[\mathbf{x}(t_f), t_f] = \boldsymbol{\nu}^T\boldsymbol{\Psi} = \boldsymbol{\nu}^T(\mathbf{x} - \mathbf{x}_f) \tag{4}$$

The motion should minimize some combination of time required for the motion and control effort of the actuators during the motion. A suitable cost function is:

$$J = \Phi[\mathbf{x}(t_f), t_f] + \int_{t_0}^{t_f} L\,dt \tag{5}$$

where:

$$L = a + \frac{1}{2}\mathbf{u}^T B\mathbf{u} \tag{6}$$

$$0 \le a < 1 \tag{7}$$

$$B_{ij} = \begin{cases} b_i(1-a) > 0 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{8}$$

The constant a is the ratio between the conflicting goals of minimizing the time required for the motion and minimizing the control effort of the actuators during the motion. If $a = 0$, then time is of no concern. If $a \approx 1$ then control effort levels are of little concern. Note that B is positive definite so B^{-1} exists. Also, since B is diagonal, $B^{-1} = B^{-T}$.

Following the typical optimal controls approach, the previous constrained problem can be converted to an unconstrained optimization. This is done by constructing a modified cost functional which enforces the equations of motion by adjoining the equations of motion with lagrange multipliers, $\boldsymbol{\lambda}$:

$$\bar{J} = \Phi[\mathbf{x}(t_f), t_f] + \int_{t_0}^{t_f} \left[L + \boldsymbol{\lambda}^T(-\dot{\mathbf{x}} + \mathbf{f} + \mathbf{g}\mathbf{u}) \right] dt \tag{9}$$

To simplify the problem, the Hamiltonian of the system at some instant in time is introduced:

$$H = L + \boldsymbol{\lambda}^T(\mathbf{f} + \mathbf{g}\mathbf{u}) \tag{10}$$

Which means the modified cost functional is:

$$\bar{J} = \Phi[\mathbf{x}(t_f), t_f] + \int_{t_0}^{t_f} \left[H - \boldsymbol{\lambda}^T \dot{\mathbf{x}} \right] dt \tag{11}$$

Taking the total variation of $\bar{J}$ and integrating by parts results in:

$$\delta\bar{J} = \left[\left(\frac{\partial \Phi}{\partial \mathbf{x}} - \boldsymbol{\lambda}^T \right) \delta\mathbf{x} \right]_{t=t_f} + \left[\boldsymbol{\lambda}^T \delta\mathbf{x} \right]_{t=t_0} + \int_{t_0}^{t_f} \left[\left(\frac{\partial H}{\partial \mathbf{x}} + \dot{\boldsymbol{\lambda}}^T \right) \delta\mathbf{x} + \frac{\partial H}{\partial \mathbf{u}} \delta\mathbf{u} \right] dt \tag{12}$$

Note that the variation with respect to $\boldsymbol{\lambda}$ has been omitted since it leads back to the equations of motion. To force the variation of the modified cost functional, $\delta\bar{J}$, to vanish, we choose:

$$\dot{\boldsymbol{\lambda}}^T = -\frac{\partial H}{\partial \mathbf{x}} \tag{13}$$

$$\boldsymbol{\lambda}^T(t_f) = \left[\frac{\partial \Phi}{\partial \mathbf{x}} \right]_{t=t_f} \tag{14}$$

$$\frac{\partial H}{\partial \mathbf{u}} = 0 \tag{15}$$

$$\delta\mathbf{x}|_{t=t_0} = 0 \tag{16}$$

$$= \frac{\partial H}{\partial \mathbf{x}}\mathbf{f} + \mathbf{f}^T\left(-\frac{\partial H}{\partial \mathbf{x}}\right)^T + 0\dot{\mathbf{u}} + \frac{\partial H}{\partial t} \tag{23}$$

$$= \frac{\partial H}{\partial t} \tag{24}$$

The Hamiltonian is autonomous so $\frac{d}{dt}H = \frac{\partial H}{\partial t} = 0$. This means that H is constant on the optimal trajectory. Its constant value must be the same as its final value. $H_f = 0$. Therefore, $H = 0$ on the optimal trajectory.

The final value of the costate vector, $\boldsymbol{\lambda}$, is determined from the terminal cost:

$$\boldsymbol{\lambda}(t_f) = \boldsymbol{\lambda}_f = \left[\frac{\partial \Phi}{\partial \mathbf{x}}\right]^T_{t=t_f} \tag{25}$$

$$= \left[\frac{\partial}{\partial \mathbf{x}}(\boldsymbol{\nu}^T\boldsymbol{\Psi})\right]^T_{t=t_f} \tag{26}$$

$$= \left[\frac{\partial}{\partial \mathbf{x}}(\boldsymbol{\nu}^T(\mathbf{x}-\mathbf{x}_f))\right]^T_{t=t_f} \tag{27}$$

$$= \boldsymbol{\nu} \tag{28}$$

This indicates that the final value of the costates are unknown constants. In order to determine their values, consider the Hamiltonian at the final time. Substitute $\mathbf{u} = -B^{-1}\mathbf{g}^T$, $\boldsymbol{\lambda} = \boldsymbol{\nu}$, $\mathbf{f}_f = \mathbf{f}(\mathbf{x}(t_f))$, and $\mathbf{g}_f = \mathbf{g}(\mathbf{x}(t_f))$ into H:

$$H = \left[a + \frac{1}{2}\mathbf{u}^T B\mathbf{u} + \boldsymbol{\lambda}^T(\mathbf{f}+\mathbf{g}\mathbf{u})\right]_{t=t_f} \tag{29}$$

$$= a + \frac{1}{2}(-B^{-1}\mathbf{g}_f^t\boldsymbol{\nu})^T B(-B^{-1}\mathbf{g}_f^t\boldsymbol{\nu}) + \boldsymbol{\nu}^T(\mathbf{f}_f + \mathbf{g}_f(-B^{-1}\mathbf{g}_f^t\boldsymbol{\nu})) \tag{30}$$

$$= a + \frac{1}{2}\boldsymbol{\nu}^T\mathbf{g}_f B^{-T}BB^{-1}\mathbf{g}_f^T + \boldsymbol{\nu}^T\mathbf{f}_f - \boldsymbol{\nu}^T\mathbf{g}_f B^{-1}\mathbf{g}_f^T\boldsymbol{\nu} \tag{31}$$

$$= a + \boldsymbol{\nu}^T\mathbf{f}_f - \frac{1}{2}\boldsymbol{\nu}^T(\mathbf{g}_f B^{-1}\mathbf{g}_f^t)\boldsymbol{\nu} \tag{32}$$

At the final time, H is:

$$H = a + \boldsymbol{\nu}^T\mathbf{f}_f - \frac{1}{2}\boldsymbol{\nu}^T(\mathbf{g}_f B^{-1}\mathbf{g}_f^T)\boldsymbol{\nu} = 0 \tag{33}$$

These choices guarantee that the resulting $\boldsymbol{\lambda}$ and $\mathbf{u}$ produce a stationary value of $\bar{J}$. For $\bar{J}$ to be minimized, the second gradient must be positive definite:

$$\begin{bmatrix} H_{xx} & H_{xu} \\ H_{ux} & H_{uu} \end{bmatrix} > 0 \tag{17}$$

where the subscripts represent partial derivatives. This can be implemented symbolically. If this is satisfied, then the resulting controls will be optimal. See [9, p.50], for a detailed derivation of this requirement. Note that $H_{uu} = B$ which was chosen to be positive definite.

Applying these results to the problem at hand gives:

$$\frac{\partial H}{\partial \mathbf{x}} = -\dot{\boldsymbol{\lambda}}^T \implies \dot{\boldsymbol{\lambda}} = -\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)^T \boldsymbol{\lambda} \tag{18}$$

$$\frac{\partial H}{\partial \mathbf{u}} = 0 \implies \mathbf{u}^T B + \boldsymbol{\lambda}^T \mathbf{g} = 0 \tag{19}$$

Equation (18) gives the differential equations for the costate vector, $\boldsymbol{\lambda}$. Solving Equation (19) for $\mathbf{u}$ gives the optimal control law:

$$\mathbf{u} = -B^{-1}\mathbf{g}^T\boldsymbol{\lambda} \tag{20}$$

Since the terminal time, t_f, is not specified, it is a free parameter. Treating t_f as a free parameter produces a modified total variation of the cost functional, $\delta\bar{J}$. The previous analysis and choices force all the terms to vanish except for the variation due to possible changes in the final time:

$$\delta\bar{J} = \left[\frac{\partial \Phi}{\partial t} + H\right]_{t=t_f} \delta t_f \tag{21}$$

See [9, p.72], for a detailed derivation of this requirement. Since Φ is autonomous, $\frac{\partial \Phi}{\partial t} = 0$ and therefore $H_f = 0$.

The Hamiltonian, H, is constant as can be seen by taking its derivative with respect to time:

$$\frac{d}{dt}H = \frac{\partial H}{\partial \mathbf{x}}\dot{\mathbf{x}} + \frac{\partial H}{\partial \boldsymbol{\lambda}}\dot{\boldsymbol{\lambda}} + \frac{\partial H}{\partial \mathbf{u}}\dot{\mathbf{u}} + \frac{\partial H}{\partial t} \tag{22}$$

However, since $H = 0$ during the entire motion, a similar statement is true at the intial time:

$$H = a + \boldsymbol{\mu}^T \mathbf{f}_0 - \frac{1}{2}\boldsymbol{\mu}^T(\mathbf{g}_0 B^{-1}\mathbf{g}_0^T)\boldsymbol{\mu} = 0 \tag{34}$$

where $\boldsymbol{\mu} = \boldsymbol{\lambda}(t_0)$, $\mathbf{f}_0 = \mathbf{f}(\mathbf{x}(t_0))$, and $\mathbf{g}_0 = \mathbf{g}(\mathbf{x}(t_0))$. Note that $\boldsymbol{\mu}$ is not known but that $\mathbf{f}_0$ and $\mathbf{g}_0$ are both known. (Note that if this is satisfied at the initial time, the optimal control guarantees that the similar requirement will be satisfied at the final time.)

This is a quadratic form in $\boldsymbol{\mu}$. It describes a multi-dimensional surface. Mathematically, this equation can have either no solutions, a unique solution, or many solutions.

A Newton-Raphson style scheme can be used to find a solution for $\boldsymbol{\mu}$ if one exists. Consider the change in H to a small change in μ_i near a solution:

$$H_0(\mu_1, \mu_2, \ldots, \mu_i + \Delta\mu_i, \mu_{i+1}, \ldots, \mu_n) \;=\; H_0(\boldsymbol{\mu}) + \frac{\partial H_0}{\partial \mu_i}\Delta\mu_i + O(\mu_i^2) \tag{35}$$

If μ is near a solution, then $H_0(\mu_1, \mu_2, \ldots, \mu_i + \Delta\mu_i, \mu_{i+1}, \ldots, \mu_n) \approx 0$ and the $O(\mu_i^2)$ terms are nearly zero so the resulting equation is:

$$0 \approx H_0(\boldsymbol{\mu}) + \frac{\partial H_0}{\partial \mu_i}\Delta\mu_i \tag{36}$$

This equation can be solved for $\Delta\mu_i$:

$$\Delta\mu_i = -\frac{H_0(\mu)}{\left(\frac{\partial H_0}{\partial \mu_i}\right)} \tag{37}$$

Then μ_i can be improved:

$$(\mu_i)_{new} = (\mu_i)_{old} + \Delta\mu_i \tag{38}$$

This is a scalar equation for one μ_i. The values of $\frac{\partial H_0}{\partial \mu_i}$ can be determined as follows:

$$\frac{\partial H_0(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} = \mathbf{f}_0^T - \boldsymbol{\mu}^T(\mathbf{g}_0 B^{-1}\mathbf{g}_0^T) \tag{39}$$

For computational efficiency, the n update equations of the form of Equation (38) can be applied in parallel.

Some experimentation may be required to choose the weighting values a and b_i to produce reasonable trajectories with reasonable joint actuator levels.

In summary, the system has a set of first-order ordinary differential equations for state and costate. It is a two point boundary problem where initial and final values exist for the state. The initial values of the costate are unknown but candidates can be found using the Newton-Raphson type approach just described. This problem can be attacked by a shooting technique. Errors at the end can be used to improve the guess of the inital costate values. The only modfication needed to this is that the initial costate values must be further modified so that they satisify the quadratic form.

Appendix C. Applied Optimal Controls Example

Appendix C.1. Description

The purpose of this example is to demonstrate the type of symbolic manipulation techniques that can be applied to generate optimal controls laws. Obviously, this requires that the equations of motion are available in symbolic form.

Appendix C.2. MACSYMA Usage Descriptions and Code

The following description of the function OPTCONT is from the file OPTCONT.USAGE:

```
The function OPCONT will take an array of first order ordinary
differential equations with a cost functional and it will derive
and return the optimal control, the costate equations, the
Hamiltonian, the Hessian of the system.  To use this function, the
dynamic system must be put in the form of a list of first order
differential equations of the form:

                dx
                -- = f(x,u)
                dt

USAGE:    OPTCONT(odes,L,x_name,u_name);
-----

INPUTS:
------
  odes : list of first order ordinary differential equations, eg,
         [dx1/dt = f1(x,u), dx2/dt = f2(x,u), ...]
         describing the dynamics of the system.

     L : the cost functional of the system (in terms of x and u);
         [scalar function of x,u]
```

```
x_name : list of names of the states used in odes and L.
       . Must be in the same order and in one-to-one corrspondence
         with the variables in the left hand sides in the odes.

u_name : list of names of the control inputs used in f and L.


OUTPUTS:
-------

A list composed of:

costate : List of the costate first-order ode equations in terms
          of the original state variable names and new multiplier
          variables, Li.

costate_names : a List of the newly introduced costate names

  u_opt : List of Optimal controls

      H : The Hamiltonian of the system, H = L + LT*f [scalar
          function]

 H_hess : The hessian of H,
                              H_hess = [ Hxx Hxu ]
                                       [ Hux Huu ]
          This can be used to determine if the generated control
          inputs are optimal.  [(n+m) x (n+m) MATRIX of scalar
          functions]

NOTE: The list costate contains variables of the form L1,L2,L3...
      This function uses KILL on all of these it uses, so existing
      variables with names of this form will be destroyed.

By: Jonathan M. Cameron
```

The MACSYMA code for the function OPTCONT follows:

```
OPTCONT(odes,L,x_name,u_name) :=
BLOCK([ n : Length(odes),    /* Number of states */
        m : Length(u_name),  /* Number of controls */
        Lambda, U_Eqns, Hx, Hu, i, j,
        costate, u_opt, H, H_hess],
```

```
/* Construct list of lambdas */
Lambda : [],
for i:1 thru n do
   Lambda : append(Lambda, [concat('L,i)]),
Apply('KILL,Lambda),

/* Form the Hamiltonian */
H : L,
for i:1 thru n do
   H : H + Lambda[i]*RHS(Odes[i]),

/* Construct costate equations */
Hx : [],
for i:1 thru n do
   Hx : append(Hx, [diff(H,x_name[i])]),
costate : [],
for i:1 thru n do (
   Depends(Lambda[i],T),
   costate : append(costate, [diff(lambda[i],T) = -Hx[i]])
   ),

/* Solve for the optimal controls */
Hu : [],
for i:1 thru m do
   Hu : append(Hu, [diff(H,u_name[i])]),
U_Eqns : [],
for i:1 thru m do
   U_Eqns : append(U_Eqns, [0 = Hu[i]]),
u_opt : Solve(U_Eqns,u_name),

/* Generate the Hessian */        /* H_hess = [Hxx Hxu] */
H_hess : ZEROMATRIX(n+m,n+m),     /*          [Hux Huu] */
/* Do the Hxx block */
for i:1 thru n do
   for j:1 thru n do
      H_hess[i,j] : diff(Hx[j], x_name[i]),
/* Do the Hxu block */
for i:1 thru n do
   for j:1 thru m do
      H_hess[i,j+n] : diff(Hu[j], x_name[i]),
/* Do the Hux block */
for i:1 thru m do
   for j:1 thru n do
      H_hess[i+n,j] : diff(Hx[j], u_name[i]),
```

```
/* Do the Huu block */
for i:1 thru m do
   for j:1 thru m do
      H_hess[i+n,j+n] : diff(Hu[j], u_name[i]),

/* Return the results */
[costate, lambda, u_opt, H, H_hess]
)$
```

The following description of the function SIMPCONT is from the file SIMPCONT.USAGE:

```
SIMPCONT is a function which will take lists of state and costate
first order ordinary differential equations, the optimal control,
and other outputs of OPTCONT and will substitute in the optimal
control in the state ODEs and then solve as many of the ODEs as
possible.  The resulting system is ready to simplify via boundary
conditions.

USAGE:
-----
SIMPCONT(state_odes,costate_odes,u_opt,state_names,costate_names);


INPUTS:
------

state_odes : List of first order ordinary differential equations
             [dx1/dt = f1(x,u), dx2/dt = f2(x,u), ...] describing
             the dynamics of the system.

costate_odes : List of first order ordinary differential equations
               derived for optimal control by OPTCONT

u_opt : list of optimal controls derived by OPTCONT

state_names : list of the names of the states

costate_names : list of the names of the costates (generated by
                OPTCONT)

OUTPUTS:
-------
```

```
A list composed of:

new_system : List of the new simplified set of state and costate
             equations

new_names : List of the names of the states or costates in
            new_system

constants : List of the new constants generated by solutions of
            ODEs


By:  Jonathan M. Cameron
```

The MACSYMA code for the function SIMPCONT follows:

```
SIMPCONT(state,costate,u_opt,state_name,costate_name) :=
BLOCK([ n : Length(state),
        m : Length(u_opt),
        new_system : [],
        new_name : [],
        const_num : 0,
        constants : [],
        i, ii, soln],

    /* Substitute the optimal controls into the state equations */
    for i:1 thru m do
       for ii:1 thru n do
          state[ii] : LHS(state[ii]) = subst(u_opt, RHS(state[ii])),

    /* check each of the costate ODEs and try to solve them */
    for i:1 thru n do (
       soln : ode2(costate[i],costate_name[i],t),
       if soln # 'FALSE then (
          soln : subst(concat('C,const_num),%C,soln),
          constants : append([concat('C,const_num)], constants),
          const_num : const_num + 1,

          /* Do substitutions with soln to eliminate the costate */
          for ii:1 thru n do (
             state[ii] : LHS(state[ii]) = subst(soln,RHS(state[ii])),
             costate[ii] : LHS(costate[ii]) = subst(soln,RHS(costate[ii]))
             ),
          for ii:1 thru length(new_system) do
```

```
        new_system[ii] :
           LHS(new_system[ii]) = subst(soln,RHS(new_system[ii]))
      )
   else (
      new_system : append(new_system, [costate[i]]),
      new_name : append(new_name, [costate_name[i]])
      )
   ),

/* check each of the state ODEs and try to solve them */
for i:1 thru n do (
   soln : ode2(state[i],state_name[i],t),
   if soln # 'FALSE then (
      soln : subst(concat('C,const_num),%C,soln),
      constants : append([concat('C,const_num)], constants),
      const_num : const_num + 1,

      /* Do substitutions with soln to eliminate the state */
      for ii:1 thru n do (
         state[ii] : LHS(state[ii]) = subst(soln,RHS(state[ii])),
         costate[ii] : LHS(costate[ii]) = subst(soln,RHS(costate[ii]))
         ),
      for ii:1 thru length(new_system) do
         new_system[ii] :
            LHS(new_system[ii]) = subst(soln,RHS(new_system[ii])),
      /* Add this solution to the system */
      new_system : append(new_system, [soln]),
      new_name : append(new_name, [state_name[i]])
      )
   else (
      new_system : append(new_system, [state[i]]),
      new_name : append(new_name, [state_name[i]])
      )
   ),

/* Return the results */
Declare(constants, constant),
[new_system, new_name, constants]
)$
```

Appendix C.3. Sample MACSYMA Session Output

The following output is from a MACSYMA session using the functions OPTCONT and SIMPCONT on a simple problem.

```
(C3) load(optcont);

Batching the file USERD1:[CAMERON.PROP]optcont.mac;60
Batchload done.

(D3)                    USERD1:[CAMERON.PROP]optcont.mac;60


(C4) load(simpcont);

Batching the file USERD1:[CAMERON.PROP]simpcont.mac;33
Batchload done.

(D4)                    USERD1:[CAMERON.PROP]simpcont.mac;33


(C5) kill(x,v,u);

(D5)                                    DONE


(C6) depends([x,v,u],t);

(D6)                              [X(T), V(T), U(T)]


(C7) state_eqns : [diff(x,t)=v,diff(v,t)=u];

                                    dX      dV
(D7)                               [-- = V, -- = U]
                                    dT      dT


(C8) state_names : ['x,'v];

(D8)                                    [X, V]
```

```
(C9) control_names : ['u];

(D9)                                  [U]


(C10) L : 0.5*u^2;

                                        2
(D10)                                0.5 U


(C11) /* Find the optimal control and costate equations */
results : optcont(state_eqns,L,state_names,control_names)$


(C12) costate_eqns : results[1];

                                 dL1      dL2
(D12)                           [--- = 0, --- = - L1]
                                 dT       dT


(C13) costate_names : results[2];

(D13)                               [L1, L2]


(C14) opt_control : results[3];

(D14)                               [U = - L2]


(C15) H : results[4];

                                          2
(D15)                       L1 V + 0.5 U  + L2 U


(C16) Hessian : results[5];

                                    [ 0  0  0 ]
                                    [         ]
(D16)                               [ 0  0  0 ]
                                    [         ]
                                    [ 0  0  1 ]
```

```
(C17) /* Solve as many of the ODEs as possible */
results :
  simpcont(state_eqns,costate_eqns,opt_control,state_names,costate_names)$
DUBO:[MACSYMA_412.ODE]ode2.fas;1 being loaded.


(C18) system : results[1];

                          2                                      2
                      C0 T                                   C0 T
(D18)        [X = T (----- - C1 T + C3) + C2, V = ----- - C1 T + C3]
                        2                                      2


(C19) names : results[2];

(D19)                                  [X, V]


(C20) constants : results[3];

(D20)                             [C3, C2, C1, C0]


(D21)                                  DONE
```

The next step depends on the problem to be solved. In this case, it is not hard to apply inital and final state values to resolve the resulting constants. In this example, the costate and state differential equations were solved completely. This will not happen with the type of systems to be considered in this research. In general, some differential equations will be produced. In any case, it is not difficult to take the results of SIMPCONT and use MACSYMA to convert it to C or FORTRAN code for simulation purposes. Symbolic manipulation systems such as MACSYMA and Mathematica have powerful capabilities to generate program code. A function to take the results of SIMPCONT and generate code could also perform various optimizations such as computing common terms only once.

Appendix D. Movement Library Size Requirements

In order to validate the premise that the amount of data that will be saved is not too excessive, an estimate is presented in this appendix.

The motions will move the system from one combination of orientation and configuration to another combination of orientation and configuration. Suppose the goal is to move from one typical configuration to another typical configuration. Since the system is in free-fall, the final configuration has three degrees of attitude freedom with respect to the starting configuration. Think of this as the points on a unit sphere and another degree of freedom about a line from the center of the unit sphere to the points on the surface of the sphere. In order to tesselate the unit sphere, a procedure based on constructing geodesics from icosahedrons can be used [13]. The degree of the tesselation is Q. To tesselate the unit sphere so that there is an angle of approximately α between vertices, and:

$$Q = \text{int}[\arctan(2)/\alpha] \tag{40}$$

$$\text{where} \quad Q = \text{degree of tesselation} \tag{41}$$
$$\alpha = \text{approximate angle between verteces}$$

Therefore, the total number relative orientations that must be considered for moving from one configuration to another is:

$$N_O = \text{Number of relative orientations} \tag{42}$$

$$= \frac{2\pi}{\alpha}\left[10Q^2 + 2\right] \tag{43}$$

For each orientation, the joint positions, velocities, torques must be saved over the motion. So the number of data points for one motion is:

$$N_M = \text{Number of data points per motion} \tag{44}$$

$$= 3N_J N_D \tag{45}$$

$$\text{where} \quad N_J = \text{Number of joints} \tag{46}$$
$$N_D = \text{Number of data points per variable}$$

Multiplying N_O and N_M gives the number of data points necessary to store the motions from one configuration to another.

$$\begin{aligned} N_{MP} &= \text{Number of data points to move from} && (47)\\ &\quad \text{one configuration to another} && (48)\\ &= N_O.N_M && (49)\\ &= 3.N_O.N_J.N_D && (50) \end{aligned}$$

Now assume there are N_P typical configurations. The total number data points to for motions from any configuration to any other is:

$$\begin{aligned} N_{DP} &= \text{Total number of data points} && (51)\\ &= N_{MP}\left[2\binom{N_P}{2} + N_P\right] && (52)\\ &= 3.N_O.N_J.N_D\left[N_P(N_P-1)+N_P\right] && (53)\\ &= 3.N_O.N_J.N_D.N_P^2 && (54) \end{aligned}$$

where the term $\binom{N_P}{2}$ in Equation (52) gives the number of pairs of configurations. The factor of 2 is necessary because the motions for each pair of configurations could considered in either direction. The next term, N_P, accounts for motions from one configuration back to the same configuration in a different orientation.

Given a data compression ratio of C_R to 1 and assuming that the floating point value for each data point can be quantized to 8 bits (or a byte), the amount of data (in KB or kilobytes) is:

$$\begin{aligned} N_{DATA} &= \text{Total amount of data in K Bytes} && (55)\\ &= \frac{N_{DP}}{1024 C_R}\ \text{KB} && (56) \end{aligned}$$

To give some feel for the amount of data indicated by these equations, consider a few examples. For 10° orientation resolution, $Q \approx 6$ and $N_O = 13.032$. For each example $N_D = 10$, $N_P = 2$, and $C_R = 100$. The results are given in Table 1 on page 17.