Control and Simulation of Multibody Systems

by

Jeffrey Michael Wendlandt

B.S. (University of Illinois at Urbana-Champaign) 1991
M.S. (University of California at Berkeley) 1993
M.A. (University of California at Berkeley) 1995

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Mechanical Engineering

in the

GRADUATE DIVISION
of the
UNIVERSITY of CALIFORNIA at BERKELEY

Fall 1997

The dissertation of Jeffrey Michael Wendlandt is approved:

Cochair	Date
Cochair	Date
	Date
	Date

University of California at Berkeley

Fall 1997

Control and Simulation of Multibody Systems

Abstract

Control and Simulation of Multibody Systems

by

Jeffrey Michael Wendlandt

Doctor of Philosophy in Mechanical Engineering

University of California at Berkeley

Professor S. Shankar Sastry, Cochair
Professor Kameshwar Poolla, Cochair

Algorithms for the control and simulation of multibody systems are created in this dissertation to aid in the design of controlled mechanical systems. A multibody system is a collection of rigid bodies connected by joints and serves as a basis for many models of mechanical systems. A procedure is created to construct mechanical integrators for Lagrangian systems with holonomic constraints. Mechanical integrators are numerical integrators that respect the structure of mechanical systems and conserve energy, momentum, and / or are symplectic. The construction procedure creates symplectic-momentum integrators based on a discrete variational principle and discretizes the principles of mechanics rather than the equations of motion. The method is applied to the double spherical pendulum and the free rigid body, and numerical results are given. A 3D balancing controller is designed for a multibody model of a human biped. The controller utilizes recursive multibody dynamics algorithms, forms a workspace model, efficiently calculates kinematics and joint torques, and coordinates the degrees of freedom of the two legs. The controller is designed to serve as a basis for more complicated motions: walking, running, jumping, changing direction, and adapting to loads. Simulation results are presented of the controlled model reacting to disturbances. Recursive multibody algorithms for forward kinematics, inverse dynamics, and forward dynamics of tree-structured multibody systems are derived using Lie group notation. Components of a multibody simulator specifically designed for real-time simulation of human biped models interacting with the ground are designed. A simple contact model is developed, and the mechanical integrators with external forces are used. Simulation results of a rigid body colliding with the ground are given. Discrete-time equations of motion for tree-structured multibody systems are derived, and a method is developed to solve the equations with a computational cost that grows linearly with the number of joints. The dissertation concludes with a discussion of future work.

Professor S. Shankar Sastry
Dissertation Committee Cochair

Professor Kameshwar Poolla
Dissertation Committee Cochair

For Dalila

Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgements

I first would like to thank Shankar Sastry for all of his advice, support, encouragement, and enthusiasm over the years. I am grateful to be a part of the Intelligent Machines and Robotics Laboratory and have benefited greatly from the unique environment.

I would like to thank Jerry Marsden for his support and advice over the past several years. It was a pleasure working with him, and I have learned a great deal from the experience.

I would like to thank Kameshwar Poolla and Andy Packard for being on my committee and for giving me advice throughout graduate school.

I have enjoyed the company of many interesting graduate students and researchers. Thanks to Matt Berkemeier, Steve Burgett, Linda Bushnell, Michael Cohn, Charles Coleman, Lara Crawford, Richard Edell, Neil Getz, Datta Godbole, Raja Kadiyalla, John Lygeros, Brian Mirtich, Ed Nicolson, George Pappas, Eric Paulos, Karl Petty, Shahram Shahruz, Dawn Tilbury, Claire Tomlin, Greg Walsh, Joe Weber, and many others for all of their help, interesting discussions, and encouragement.

I would like to thank Richard Murray for his advice, encouragement, and enthusiasm over the years. I would like to thank him and Abhi Jain for their help with an initial investigation of simulation techniques.

I would like to thank Ping Hsu for his advice, friendship, and encouragement to pursue the Ph.D. degree. Thanks also to George Mulholland for introducing me to design and control as a co-op student at Fermilab.

I thank NSF (graduate student fellowship), ARO (DAAH-O4-96-1-0119 (AASERT), DAAH-04-95-1-0588), and DOE (DE-FG03095ER-25251) for financial support.

I would like to thank my family and parents for much love and support. I would like to thank my parents for encouraging me to do well in school. Little did they know that their 5 year old son would be in school for 24 years.

A big thanks to my wife, Dalila, for her love, support, and encouragement to graduate.

Chapter 1

Introduction

This dissertation is concerned with creating better algorithms for the control and simulation of multibody systems - collections of rigid bodies interconnected with joints. New algorithms are created in this dissertation to aid in designing, prototyping, and developing controlled mechanical systems by exploiting the structure of multibody systems. Specifically, a balancing controller for a 3D multibody model of a human biped is created, and mechanical integrators (preserve energy, momentum, and/or are symplectic - see (Marsden, 1992)) are created for Lagrangian systems with holonomic constraints.

Multibody systems serve as a basis for many models of mechanical systems and have uses in many areas of science and engineering. Multibody models are used to analyze biological systems and human locomotion. Controlled mechanical systems are often prototyped through computer simulation of multibody models. There are uses in medical robotics, space robotics, and space structures. There are also applications in animation systems. The specific application examined in this dissertation is the creation of predictive models of human motion. The intent of these models is to be able to predict the motion of human models in various situations and aid in training and equipment design.

For certain applications, it is important to preserve the underlying mechanical structure of the system. For example, in space applications, momentum conservation is critical, especially for long-time simulations. Energy conservation and invariance of the symplectic form (an abstract property of Hamiltonian systems to be described in Chapter 2) seems important for long-term simulation and for obtaining qualitative information from simulations. For instance, symplectic integrators have found uses in predicting the long-term motion of planets and comets in the solar system. One can imagine that symplectic-momentum or energy-momentum integration methods are useful for long-term simulation of spacecraft and space robots.

Many researchers have attacked the problem of multibody simulation and many books have been written on the subject (for a sampling, see (Schiehlen, 1993), (de Jalón and Bayo, 1994), (Shabana, 1989), and (Amirouche, 1992)). Multibody systems require special techniques to deal with the complex equations of motion and require computer algorithms for efficient simulation. However, there is a lack of mechanical integrators for multibody systems and certain applications suffer from a lack of good simulation tools. The work in this dissertation sets out to create mechanical integrators for these systems.

While there has been a significant amount of research in multibody simulation, there has been much less progress in the development of good tools for the design of multi-body control systems, especially for systems with contact. The current control methods available for these systems are difficult to apply and to customize for specific properties.

Designers of controlled multibody systems should have good tools to aid in the creation of controlled mechanical systems, especially multibody systems with many degrees of freedom. The control methods should be able to handle model changes, contact with the environment, and coordination of the degrees of freedom. The control algorithms must be able to create the desired behavior in the face of very complex, nonlinear equations of motion with discrete model changes.

1.1 Contributions

The main contributions of this dissertation are:

- *A general methodology to create symplectic-momentum mechanical integrators for Lagrangian systems with holonomic constraints.* The methodology has an elegant theoretical basis and geometric construction that is simple to apply. The construction procedure is applied to the double spherical pendulum, a rigid body, and multibody systems. The methodology provides theoretical insights into existing integration methods while also creating new integration methods.

- *The creation of a 3D balancing controller for a multibody model of a human biped.* The legged model has 18 degrees of freedom, and the controller commands 12 joint torques to respond to desired trajectories and disturbances. The controller gains are easy to tune, and the controller has a large operating region. The controller is designed to serve as a basis for more complicated motion: walking, running, jumping, changing direction, and adapting to loads.

- *The design of a multibody simulator well-suited for the development of control algorithms for human motion.* The control development platform is designed for real-time simulation of human motion and specifically designed for prototyping and testing control systems. External forces are added to the mechanical integration method, and a simple contact model is designed. The mechanical integration method is applied to tree-structured multibody systems. Sparsity can be exploited to produce a simulation method where the computational cost grows linearly with the number of degrees of freedom.

- *The derivation of existing recursive multibody kinematics and dynamics algorithms for tree-structured systems in terms of Lie group notation.* The recursive algorithms extend the formalism presented in (Murray et al., 1994) and can be used on robotic systems more complicated than serial chains.

1.2 Outline

The body of the dissertation consists of four chapters each dealing with aspects of the control and simulation of multibody systems.

- **Chapter 2 - Mechanical Integrators.** This chapter develops a method to construct symplectic-momentum integrators for Lagrangian systems with holonomic constraints. Simulations of mechanical systems are often performed by deriving the continuous-time differential equations of motion using principles of mechanics and then by applying a standard integration method to the differential equations. In this chapter, the principles of mechanics are discretized instead of the equations of motion. A version of discrete mechanics is introduced, and it is then shown how to use the theory to construct numerical integrators. The construction procedure produces symplectic-momentum integrators for Lagrangian systems with holonomic constraints. The theory is applied to the double spherical pendulum (DSP) and the rigid body (RB). Numerical results are then presented.

- **Chapter 3 - Recursive Multibody Kinematics and Dynamics.** This chapter develops recursive, linear-time algorithms to compute the forward kinematics, inverse dynamics, and forward dynamics of tree-structured multibody systems. The algorithms are derived based on the Lie group notation used in (Murray et al., 1994). The chapter primarily serves as a reference and derives existing results. Chapters 4 and 5 use the results in this chapter.

- **Chapter 4 - Recursive Workspace Control.** A model-based balancing controller for a 3D model of a human biped is designed and presented. Recursive multibody algorithms are developed to efficiently create a workspace model and to compute the joint torques and kinematics. The controller is designed to have a small number of parameters, a large operating region, to be easy to tune, and to serve as a basis for more complicated motion: walking, running, jumping, changing direction, and adapting to loads. The system is simulated in *Impulse* (Mirtich, 1996) and simulation results are presented.

- **Chapter 5 - Components of a Multibody Simulator.** This chapter presents components for a multibody simulator specifically designed for the simulation of controlled multibody models of human bipeds in contact with the ground. The methods in

Chapter 2 are first extended to include general external forces. A simple contact model is then designed to model the interaction of the feet with the ground. The contact model and a discrete-time rigid body model are combined, and simulation results of a rigid body colliding with the ground are presented. The techniques are then applied to tree-structured multibody systems to produce a symplectic-momentum mechanical integrator that can be combined with controller torques and the contact model. The discrete-time equations of motion are given and an algorithm to solve the equations in linear computational time is derived.

1.3 Mathematical Tools Required

This dissertation assumes the reader has a background in robotics, differential geometry, and geometric mechanics. The primary references used in this dissertation are (Murray et al., 1994), (Marsden and Ratiu, 1994), and (Munkres, 1991).

- **Chapter 2.** This chapter assumes a background in differential geometry (manifolds, differential forms, tangent spaces, cotangent spaces, exterior derivative, pull-backs, mappings) and geometric mechanics (Lagrangian mechanics, Hamiltonian mechanics, Noether's theorem, symplectic forms). Consult (Munkres, 1991) for an excellent background on manifolds in $\mathbb{R}^n$, differential forms, the exterior derivative, and several other useful topics. Chapter 4 in (Marsden and Ratiu, 1994) gives a more abstract background in the necessary topics in differential geometry. See also (Abraham et al., 1988) for a background in differential geometry. This dissertation uses (Marsden and Ratiu, 1994) as a primary reference for geometric mechanics. See this work for a background in Lagrangian mechanics, Hamiltonian mechanics, momentum maps, Lie group theory, and Noether's theorem. See also (Scheck, 1990) for a straightforward proof of Noether's theorem and a background in Lagrangian and Hamiltonian mechanics. The results on rigid body motion and quaternions given in Appendix A are also needed in this chapter.

- **Chapter 3.** This chapter assumes a familiarity with robotics and rigid body motion and uses (Murray et al., 1994) as a primary reference. The notation used is assembled in Appendix A for convenience.

- **Chapter 4.** This chapter builds on the work in Chapter 3 and requires a knowledge of the mathematical tools for rigid body motion and quaternions. Again, these tools are contained in Appendix A.

- **Chapter 5.** This chapter builds on the work in Chapter 2 and 3. Familiarity with rigid body motion is assumed. Quaternions are used to represent the orientation of frames, and results on quaternions are given in Appendix A.

Chapter 2

Mechanical Integrators

Goals. The goal of the research in this chapter is to create a systematic procedure to construct mechanical integrators for simulating finite dimensional mechanical systems with symmetry. The method presented in this chapter is based on a discretization of Hamilton's principle. It is desirable that the method be theoretically attractive and numerically competitive.

Mechanical Integrators. These are numerical integration methods that preserve some of the invariants of the mechanical system, such as, energy, momentum, or the symplectic form. It is known that if the energy and momentum map include all the integrals from a certain class (depending on the smoothness available), then one cannot create integrators that are symplectic, energy preserving, *and* momentum preserving unless they coincidentally integrate the equations exactly up to a time parameterization (see (Ge and Marsden, 1988) for the exact statement). Thus, mechanical integrators divide into two overall classes, *symplectic-momentum* and *energy-momentum* integrators. It is the hope that by exploiting the structure of mechanical systems, one can create mechanical integrators that are not only theoretically attractive, but are more computationally efficient and have better long term simulation properties than conventional integration schemes. The overall situation for mechanical integrators is complex, and it is still evolving. Refer to (Marsden et al., 1996) for a recent collection of papers in the area and for additional references and to (Marsden, 1992) for some additional background.

The Main Technique. The research in this chapter creates a method to construct symplectic-momentum integrators for Lagrangian systems defined on a linear space with holonomic constraints. The constraint manifold, Q, is regarded as embedded in the linear space, V. A discrete version of the Lagrangian is then formed and a discrete variational principle is applied to the discrete Lagrangian system. The resulting discrete equations define an implicit (explicit in some cases) numerical integration algorithm on $Q \times Q$ that approximates the flow of the continuous Euler-Lagrange equations on TQ. The algorithm equations are called the *discrete Euler-Lagrange (DEL)* equations. Holonomic constraints are treated through constraint functions on the linear space. The constraints are satisfied at each time step through the use of Lagrange multipliers.

The DEL equations share similarities to the continuous Euler-Lagrange equations. The DEL equations preserve a symplectic form defined in this chapter and preserve a discrete momentum derived through a discrete Noether's theorem. The discrete momentum corresponding to invariance of the continuous Lagrangian system to a linear group action is conserved, and the value of the discrete momentum approaches the value of the continuous momentum as the step size decreases. In general, the method does not preserve energy for conservative Lagrangian systems, but the numerical examples suggest that the energy varies about a constant value. The energy variations decrease and the constant value approaches the continuous energy as the step size decreases.

Accuracy. The construction method produces 2-step methods that have a second order local truncation error. The position error in the numerical examples show second order convergence. One may be able to use the ideas in (Yoshida, 1990) to increase the order of accuracy. One may also add additional points to the variational principle to create more steps or more stages.

Some of the Literature. This chapter uses the discrete variational principle presented in (Veselov, 1988) and again in (Veselov, 1991) and (Moser and Veselov, 1991). It is shown in (Veselov, 1988) that the DEL equations preserve a symplectic form. The same discrete mechanics procedure is derived in (Baez and Gilliam, 1995) using an algebraic approach, and they also show that there is a discrete Noether's theorem for infinitesimal symmetry.

Various authors have proposed versions of discrete mechanics. Some study discrete mechanics without the motivation of constructing integration schemes while this is a definite motivation for other authors. In (Maeda, 1981), the author presents a version of discrete mechanics based on the concept of a difference space. The author later shows how to derive the discrete equations from a discrete version of Hamilton's variational principle, the same discretization later used in (Veselov, 1988). The author in (Maeda, 1981) also presents a version of Noether's theorem. A different approach to discrete mechanics for point mass systems not derived from a variational principle is shown in (Labudde and

Greenspan, 1974), (Labudde and Greenspan, 1976a), and (Labudde and Greenspan, 1976b). These algorithms preserve energy and momentum. The author in (MacKay, 1992) discusses methods to approximate the action integral and to use Hamilton's principle to create numerical integrators. The authors in (Lewis and Kostelec, 1996) use Hamilton's principle and restricted function spaces to create integration algorithms. The discrete mechanics approach in (Veselov, 1988) is adopted in this dissertation.

Some authors discretize the principle of least action instead of Hamilton's principle. Algorithms that conserve the Hamiltonian are derived in (Itoh and Abe, 1988) based on difference quotients. Differentiation is not used and the action is extremized using variational difference quotients. This development presents multistep methods with variable time steps. The least action principle is discretized in a different way in (Shibberu, 1994). The resulting equations explicitly enforce energy, and it is stated that the equations preserve quadratic invariants.

Various energy-momentum integrators have been developed by Simo and his coworkers. See, for example, (Simo and Tarnow, 1992). Recently, energy-momentum integrators have been derived based on discrete directional derivatives and discrete versions of Hamiltonian mechanics in (Gonzalez, 1996a). More references on energy-momentum methods are in the reference section of (Gonzalez, 1996a) and in (Gonzalez, 1996b). Symplectic, momentum and energy conserving schemes for the rigid body are presented in (Lewis and Simo, 1995).

There is a vast amount of literature on symplectic schemes for Hamiltonian systems. The overview of symplectic integrators in (Sanz-Serna, 1991) provides background and references. See also (Channell and Scovel, 1990) for a survey of the early work and (McLachlan and Scovel, 1996) for a presentation of open problems in symplectic integration. References related to the work shown in this chapter are (Reich, 1993), (Reich, 1994), (McLachlan and Scovel, 1995), and (Jay, 1996). In (Reich, 1993), an integration method is presented for Hamiltonian systems that enforces position and velocity constraints in such a way to make the overall method symplectic. It is shown in (Reich, 1994) and in (McLachlan and Scovel, 1995) that the algorithm also conserves momentum corresponding to a linear symmetry group when the constraint manifold is embedded in a linear space. For another treatment of algorithms formed by embedding the constraint manifold in a linear space, see (Barth and Leimkuhler, 1996b). See (Leimkuhler and Patrick, 1996) for a treatment of symplectic integration on Riemannian manifolds. The algorithm presented in this chapter also embeds the constraint manifold in a linear space but only enforces position constraints.

Contributions. The dissertation work in this chapter formulates and develops a general approach to discrete mechanics by building on the work in (Maeda, 1981) and in (Veselov, 1988). These results are used to create a general method for constructing symplectic-momentum integrators for Lagrangian systems with holonomic constraints. An equivalent algorithm is presented in terms of generalized coordinates where the constraint equations are eliminated.

Outline of the Chapter. The theory for discrete mechanics is first developed and presented in a consistent notation by presenting the discrete variational principle (DVP) and by deriving the properties of the discrete Euler-Lagrange (DEL) equations. The discrete mechanics theory is then used to develop a construction procedure for mechanical integrators. A construction procedure is presented for constrained and generalized coordinates followed by a discussion of the structure of the Jacobian relevant to solving the DEL equations. It is then shown that the DEL equations have a second order local truncation error, and that the DEL equations have a solution for a small enough time step as long as the continuous Euler-Lagrange equations are solvable. The definition for the discrete momentum is then presented. The method is applied to the rigid body (RB) to produce evolution equations in terms of unit quaternions and is applied to the double spherical pendulum (DSP). For both examples, the momentum, energy, accuracy, and efficiency is examined. The DSP integrator is then compared to an energy-momentum integrator.

2.1 Discrete Variational Principle

A *discrete variational principle (DVP)* is presented in this section that leads to evolution equations that are analogous to the Euler-Lagrange equations. The evolution equations are called the *discrete Euler-Lagrange (DEL)* equations. The results in this and the next section have appeared in (Veselov, 1988), (Veselov, 1991), (Moser and Veselov, 1991) and in (Baez and Gilliam, 1995) but are rederived here in a consistent notation for completeness and clarity.

Given a configuration space, Q, a *discrete Lagrangian* is a map $\mathbb{L} : Q \times Q \to \mathbb{R}$. It is later shown in Equation (2.23) how to define a discrete Lagrangian given a continuous-time Lagrangian. A procedure that defines the evolution map for the system is now given.

For a fixed, positive integer N, the *action sum* is the map $\mathbb{S} : Q^{N+1} \to \mathbb{R}$ defined by

$$\mathbb{S} = \sum_{k=0}^{N-1} \mathbb{L}\left(q_{k+1}, q_k\right), \tag{2.1}$$

where $q_k \in Q$ and $k \in \mathbb{Z}$ is the discrete time. The action sum is a discrete analog of the action integral. The *discrete variational principle* states that the evolution equations extremize the action sum given fixed end points, q_0 and q_N. Extremizing $\mathbb{S}$ over $q_1, \cdots, q_{N-1}$ leads to the DEL equations:

$$D_2\mathbb{L}\left(q_{k+1}, q_k\right) + D_1\mathbb{L}\left(q_k, q_{k-1}\right) = 0 \quad \text{for all } k \in \{1, \cdots, N-1\} \tag{2.2}$$

or

$$D_2\mathbb{L} \circ \Phi + D_1\mathbb{L} = 0, \tag{2.3}$$

where $\Phi : Q \times Q \to Q \times Q$ is defined implicitly by $\Phi\left(q_k, q_{k-1}\right) = \left(q_{k+1}, q_k\right)$. If $D_2\mathbb{L}$ is invertible, then Equation (2.3) defines the discrete map, Φ, which evolves the system forward in discrete time.

2.2 Invariance Properties

The symplectic structure of $Q \times Q$ is defined in this section and an equation for the symplectic form on $Q \times Q$ is given. It is then shown that Φ preserves the symplectic form. A discrete Noether's theorem is then derived by showing that invariance of the discrete Lagrangian leads to a conserved quantity, a momentum map, for the iterations of Φ.

2.2.1 Symplectic Structure

First define a fiber derivative by

$$\begin{aligned} \mathbb{FL} : Q \times Q &\rightarrow T^*Q \\ (q_1, q_0) &\mapsto (q_0, D_2\mathbb{L}(q_1, q_0)) \end{aligned} \tag{2.4}$$

and define the 2-form ω on $Q \times Q$ by pulling back the canonical 2-form on T^*Q:

$$\begin{aligned} \omega &= \mathbb{FL}^*(\Omega_{\mathrm{CAN}}) \\ &= \mathbb{FL}^*(-d\Theta_{\mathrm{CAN}}) \\ &= -d(\mathbb{FL}^*(\Theta_{\mathrm{CAN}})). \end{aligned} \tag{2.5}$$

The fiber derivative is analogous to the Legendre transform in continuous-time Lagrangian mechanics. Choose coordinates, q^i, on Q and choose the canonical coordinates, (q^i, p_i), on T^*Q. In these coordinates, $\Omega_{\mathrm{CAN}} = dq^i \wedge dp_i$ and $\Theta_{\mathrm{CAN}} = p_i dq^i$. The DEL equations are

$$\frac{\partial\mathbb{L}}{\partial q_k^i} \circ \Phi(q_{k+1}, q_k) + \frac{\partial\mathbb{L}}{\partial q_{k+1}^i}(q_{k+1}, q_k) = 0 \tag{2.6}$$

or

$$\frac{\partial\mathbb{L}}{\partial q_{k+1}^i}(q_{k+2}, q_{k+1}) + \frac{\partial\mathbb{L}}{\partial q_{k+1}^i}(q_{k+1}, q_k) = 0. \tag{2.7}$$

Continuing the calculations in Equation (2.5) gives

$$\omega = -d\left(\frac{\partial\mathbb{L}}{\partial q_k^i}(q_{k+1}, q_k)\right) dq_k^i \tag{2.8}$$

$$= -\frac{\partial^2\mathbb{L}}{\partial q_k^i \partial q_{k+1}^j}(q_{k+1}, q_k)\, dq_{k+1}^j \wedge dq_k^i - \frac{\partial^2\mathbb{L}}{\partial q_k^i \partial q_k^j}(q_{k+1}, q_k)\, dq_k^j \wedge dq_k^i \tag{2.9}$$

$$= \frac{\partial^2\mathbb{L}}{\partial q_k^i \partial q_{k+1}^j}(q_{k+1}, q_k)\, dq_k^i \wedge dq_{k+1}^j, \tag{2.10}$$

since the second term in Equation (2.9) vanishes.

2.2.2 Preservation of the Symplectic Form

It is now shown that Φ preserves the symplectic form, *i.e.* $\Phi^*\omega = \omega$ where Φ^* is the pullback of Φ.

$$\begin{aligned}
\Phi^*\omega &= \Phi^*\left(-d\left(\frac{\partial \mathbb{L}}{\partial q^i_{k+1}}\left(q_{k+2}, q_{k+1}\right) dq^i_{k+1}\right)\right) && (2.11)\\
&= -d\left(\Phi^*\left(\frac{\partial \mathbb{L}}{\partial q^i_{k+1}}\left(q_{k+2}, q_{k+1}\right) dq^i_{k+1}\right)\right) && (2.12)\\
&= -d\left(\frac{\partial \mathbb{L}}{\partial q^i_{k+1}} \circ \Phi\left(q_{k+1}, q_k\right) \Phi^*\left(dq^i_{k+1}\right)\right) && (2.13)\\
&= -d\left(-\frac{\partial \mathbb{L}}{\partial q^i_{k+1}}\left(q_{k+1}, q_k\right) dq^i_{k+1}\right) && (2.14)\\
&= \frac{\partial^2 \mathbb{L}}{\partial q^j_k \partial q^i_{k+1}} dq^j_k \wedge dq^i_{k+1} && (2.15)\\
&= \omega && (2.16)
\end{aligned}$$

Equation (2.7), and the fact that $\Phi^*\left(dq_{k+1}\right) = dq_{k+1}$ is used in deriving Equation (2.14) from Equation (2.13).

2.2.3 Discrete Noether's Theorem

A discrete version of Noether's theorem is now derived. For continuous-time systems, Noether's theorem states that a symmetry of the Lagrangian leads to a conserved quantity. Let the discrete Lagrangian be invariant under the diagonal action of a Lie group G on Q, and let $\xi \in \mathfrak{g}$ where $\mathfrak{g}$ is the Lie algebra of G. Invariance of $\mathbb{L}$ implies that

$$\mathbb{L}(\exp(s\xi)q_{k+1}, \exp(s\xi)q_k) = \mathbb{L}(q_{k+1}, q_k). \tag{2.17}$$

Differentiating Equation (2.17) and setting $s = 0$ implies that

$$D_1\mathbb{L}(q_{k+1}, q_k) \cdot \xi_Q(q_{k+1}) + D_2\mathbb{L}(q_{k+1}, q_k) \cdot \xi_Q(q_k) = 0, \tag{2.18}$$

where ξ_Q is the infinitesimal generator. Consider the action sum, Equation (2.1), where $0 < i < N$ and vary q_{k+1} over $s \in \mathbb{R}$ by $q_{k+1}(s) = \exp(s\xi)\, q_{k+1}$. Since $q_{k+1}(0)$ extremizes $\mathbb{S}$, it is true that

$$\left.\frac{d\mathbb{S}}{ds}\right|_{s=0} = 0. \tag{2.19}$$

Equation (2.19) implies that

$$D_1\mathbb{L}(q_{k+1}, q_k) \cdot \xi_Q(q_{k+1}) + D_2\mathbb{L}(q_{k+2}, q_{k+1}) \cdot \xi_Q(q_{k+1}) = 0. \tag{2.20}$$

Subtracting Equation (2.18) from Equation (2.20) reveals that

$$D_2\mathbb{L}(q_{k+2}, q_{k+1}) \cdot \xi_Q(q_{k+1}) - D_2\mathbb{L}(q_{k+1}, q_k) \cdot \xi_Q(q_k) = 0. \tag{2.21}$$

If the momentum map, $\mathbb{J} : Q \times Q \to \mathfrak{g}^*$, is defined by

$$\langle \mathbb{J}(q_{k+1}, q_k), \xi \rangle \overset{\triangle}{=} \langle D_2\mathbb{L}(q_{k+1}, q_k), \xi_Q(q_k) \rangle\,, \tag{2.22}$$

then Equation (2.21) shows that the momentum map is preserved by $\Phi : Q \times Q \to Q \times Q$.

2.3 Construction of Mechanical Integrators

In this section, the discrete mechanics theory is applied to create symplectic-momentum mechanical integrators for Lagrangian systems with holonomic constraints. The first method uses Lagrange multipliers to enforce the constraints and is derived from a constrained variational problem. This formulation is called the constrained coordinate formulation. We then present a second construction procedure by choosing a set of generalized coordinates. The next section proves that the two methods are equivalent. Results are then presented on local truncation error and solvability. The discrete-time momentum map and symplectic form are then related to the continuous-time counterparts.

2.3.1 Constrained Coordinate Formulation

The construction assumes that there is a mechanical system with a constraint manifold, $Q \subset V$, where V is a real, finite dimensional vector space, and that there is an *unconstrained Lagrangian*, $L : TV \to \mathbb{R}$ which, by restriction of L to TQ, defines a *constrained Lagrangian*, $L^c : TQ \to \mathbb{R}$. The construction also assumes that there is a vector valued constraint function, $g : V \to \mathbb{R}^k$, such that $g^{-1}(0) = Q \subset V$ with 0 a regular value of g. The dimension of V is denoted n, and therefore, the dimension of Q is $m = n - k$. Also, let Λ be a real, finite dimensional vector space of Lagrange multipliers of dimension k. First define the *discrete, unconstrained Lagrangian*, $\mathbb{L} : V \times V \to \mathbb{R}$, to be

$$\mathbb{L}(y, x) = L\left(\frac{y+x}{2}, \frac{y-x}{h}\right), \tag{2.23}$$

where $h \in \mathbb{R}_+$ is the time step. The *unconstrained action sum* is defined by

$$\mathbb{S} = \sum_{k=0}^{N-1} \mathbb{L}\left(v_{k+1}, v_k\right). \tag{2.24}$$

Then extremize $\mathbb{S} : V^{N+1} \to \mathbb{R}$ subject to the constraint that $v_k \in Q \subset V$ for $k \in \{1, \cdots, N-1\}$,

$$\min_{v_k \in V, \lambda_k \in \Lambda} \left(\mathbb{S} + \sum_{k=1}^{N-1} \lambda_k^T g\left(v_k\right)\right) \tag{2.25}$$
$$\text{subject to } g(v_k) = 0 \quad \text{for all } k \in \{1, \cdots, N-1\},$$

to derive that

$$D_2\mathbb{L}\left(v_{k+1}, v_k\right) + D_1\mathbb{L}\left(v_k, v_{k-1}\right) + \lambda_k^T Dg\left(v_k\right) = 0 \quad \text{(no sum over } k\text{)} \tag{2.26}$$
$$g\left(v_k\right) = 0 \quad \text{for all } k \in \{1, \cdots, N-1\}.$$

Given v_k and v_{k-1} in $Q \subset V$, *i.e.*, $g(v_k) = 0$ and $g(v_{k-1}) = 0$, solve the following equations

$$\begin{aligned} D_2\mathbb{L}(v_{k+1}, v_k) + D_1\mathbb{L}(v_k, v_{k-1}) + \lambda_k^T Dg(v_k) &= 0 \\ g(v_{k+1}) &= 0 \end{aligned} \tag{2.27}$$

for v_{k+1} and λ_k.

In terms of the original, unconstrained Lagrangian, the symplectic-momentum integration method for Lagrangian systems with holonomic constraints given in Equation (2.27) reads as follows:

$$\boxed{\begin{gathered} \frac{1}{h}\left[\frac{\partial L}{\partial \dot{v}}\left(\frac{v_{k+1}+v_k}{2}, \frac{v_{k+1}-v_k}{h}\right) - \frac{\partial L}{\partial \dot{v}}\left(\frac{v_k+v_{k-1}}{2}, \frac{v_k-v_{k-1}}{h}\right)\right] - \\ \frac{1}{2}\left[\frac{\partial L}{\partial v}\left(\frac{v_{k+1}+v_k}{2}, \frac{v_{k+1}-v_k}{h}\right) + \frac{\partial L}{\partial v}\left(\frac{v_k+v_{k-1}}{2}, \frac{v_k-v_{k-1}}{h}\right)\right] \\ -D^T g(v_k)\lambda_k = 0 \\ g(v_{k+1}) = 0. \end{gathered}} \tag{2.28}$$

The sign of the top equation in Equation (2.27) is changed to bring Equation (2.28) in a form analogous to the continuous-time Euler-Lagrange equations with holonomic constraints.

If the continuous Lagrangian system is of the form

$$\begin{gathered} L(q,\dot{q}) = \frac{1}{2}\dot{q}^T M \dot{q} - V(q) \\ g(q) = 0, \end{gathered} \tag{2.29}$$

where M is a constant mass matrix, and V is the potential energy, then the DEL equations are

$$M\left(\frac{v_{k+1}-2v_k+v_{k-1}}{h^2}\right)+\frac{1}{2}\left(\frac{\partial V}{\partial q}\left(\frac{v_{k+1}+v_k}{2}\right)+\frac{\partial V}{\partial q}\left(\frac{v_k+v_{k-1}}{2}\right)\right)$$
$$-D^Tg\left(v_k\right)\lambda_k=0 \tag{2.30}$$
$$g\left(v_{k+1}\right)=0.$$

The constraint algorithm in Equation (2.30) is now compared to popular algorithms used in molecular dynamics simulation, and a brief background of these algorithms is given. The *Verlet* (Verlet, 1967), SHAKE (Ryckaert et al., 1977), and RATTLE (Anderson, 1983) algorithms are important for molecular dynamic simulations (Leimkuhler and Skeel, 1994). *Verlet* is the simplest of the algorithms and applies to Lagrangians in the form of Equation (2.29) with no constraints. The algorithm is a 2-step method based on a central difference of the position coordinates:

$$v_{n+1}=2v_n-v_{n-1}-h^2M^{-1}D^TV(v_n). \tag{2.31}$$

Equation (2.31) is solved for the position coordinate v_{n+1} to advance the system forward in time. SHAKE extends the *Verlet* algorithm to simulate Lagrangian systems of the form shown in Equation (2.29) with holonomic constraints. SHAKE is a 2-step method and solves for the new position coordinate v_{n+1} and Lagrange multiplier λ_n to advance the system forward in time:

$$v_{n+1}=2v_n-v_{n-1}-h^2M^{-1}D^TV(v_n)+h^2M^{-1}D^Tg(v_n)\lambda_n$$
$$g(v_{n+1})=0. \tag{2.32}$$

There are *velocity formulations* of *Verlet* and SHAKE where an equation involving the momentum and position is solved. The velocity formulation of the SHAKE algorithm is known as the leap-frog method. The velocity formulations have better numerical roundoff properties than the position formulations presented here (see (Leimkuhler and Skeel, 1994)).

The continuous-time equations of motion also satisfy the *hidden* velocity constraints created by differentiating the holonomic constraints with respect to time. SHAKE does not satisfy the hidden velocity constraint. RATTLE is an extension of SHAKE and satisfies the holonomic and hidden velocity constraints. RATTLE solves for the new position v_{n+1} and momentum p_{n+1} and also applies to Lagrangian systems in the form shown in Equation (2.29). For the RATTLE algorithm, first solve

$$v_{n+1} = v_n + hM^{-1}p_{n+\frac{1}{2}} \tag{2.33}$$

$$p_{n+\frac{1}{2}} = p_n - \frac{h}{2}D^TV(v_n) + \frac{h}{2}D^Tg(v_n)\gamma_n, \tag{2.34}$$

for v_{n+1}, $p_{n+\frac{1}{2}}$, and γ_n such that

$$g(v_{n+1}) = 0. \tag{2.35}$$

Then solve

$$p_{n+1} = p_{n+\frac{1}{2}} - \frac{h}{2}D^TV(v_{n+1}) + \frac{h}{2}D^Tg(v_{n+1})\lambda_{n+1}, \tag{2.36}$$

for p_{n+1} and λ_{n+1} such that

$$Dg(v_{n+1})M^{-1}p_{n+1} = 0. \tag{2.37}$$

The construction method developed in this chapter when applied to a Lagrangian of the form in Equation (2.29) produces an integration method similar to the SHAKE algorithm written in terms of position coordinates. However, the potential force terms differ as can be seen by comparing Equation (2.30) and Equation (2.32). The general construction procedure with the discrete Lagrangian definition in Equation (2.45) reproduces the SHAKE algorithm and gives conditions under which the SHAKE algorithm conserves momentum. The *Verlet* algorithm is recovered if the Lagrangian system has no constraints. Deriving the *Verlet* algorithm from a variational principle also appears in (Gillilan and Wilson, 1992), and the discrete variational principle they apply is similar to the principle in (Veselov, 1988). However, they do not extend the result to constraints or more general Lagrangians and do not use the discrete Lagrangian definition used here. The emphasis in (Gillilan and Wilson, 1992) is also on calculating a path given end point conditions. The procedure outlined in this chapter can handle more general Lagrangians, such as the Lagrangian for the rigid body in terms of quaternions.

2.3.2 Generalized Coordinate Formulation

For the generalized coordinate formulation, the discrete Lagrangian and the action sum is formed restricted to $Q \subset V$, and then the extremization is performed directly on Q by using a coordinate chart. The *constrained, discrete Lagrangian* is given by

$$\mathbb{L}^c : Q \times Q \to \mathbb{R}, \tag{2.38}$$

where $\mathbb{L}^c = \mathbb{L}|_{Q\times Q}$. Given a local coordinate chart, $\psi : U \subset \mathbb{R}^m \to Q \subset V$, where U is an open set in $\mathbb{R}^m$, the *constrained, discrete Lagrangian* is

$$\begin{aligned} \mathbb{L}^c\left(q_{k+1}, q_k\right) &= \mathbb{L}\left(\psi\left(q_{k+1}\right), \psi\left(q_k\right)\right) \\ &= L\left(\frac{\psi\left(q_{k+1}\right) + \psi\left(q_k\right)}{2}, \frac{\psi\left(q_{k+1}\right) - \psi\left(q_k\right)}{h}\right), \end{aligned}$$

where each q_k is in U. Notation is abused in a standard way by representing the restricted function and its representation in a coordinate chart by the same symbol. The *constrained* *action sum* is

$$\mathbb{S}^c = \sum_{k=0}^{N-1} \mathbb{L}^c\left(q_{k+1}, q_k\right). \tag{2.39}$$

Extremizing $\mathbb{S}^c : Q^{N+1} \to \mathbb{R}$ gives the discrete Euler-Lagrange (DEL) equations in terms of generalized coordinates,

$$D_2\mathbb{L}^c\left(q_{k+1}, q_k\right) + D_1\mathbb{L}^c\left(q_k, q_{k-1}\right) = 0. \tag{2.40}$$

In terms of the original, unconstrained Lagrangian, Equation (2.40) equals

$$\begin{aligned} D^T\psi\left(q_k\right)\Bigg\{\frac{1}{h}\left[\frac{\partial L}{\partial \dot{v}}\left(a_k, d_k\right) - \frac{\partial L}{\partial \dot{v}}\left(a_{k+1}, d_{k+1}\right)\right] \\ + \frac{1}{2}\left[\frac{\partial L}{\partial v}\left(a_k, d_k\right) + \frac{\partial L}{\partial v}\left(a_{k+1}, d_{k+1}\right)\right]\Bigg\} = 0, \end{aligned} \tag{2.41}$$

where

$$a_k = \frac{\psi\left(q_k\right) + \psi\left(q_{k-1}\right)}{2} \qquad \text{and} \qquad d_k = \frac{\psi\left(q_k\right) - \psi\left(q_{k-1}\right)}{h}. \tag{2.42}$$

Equations (2.41) are then solved for q_{k+1} given q_k and q_{k-1} to advance the flow one time step.

2.3.3 Equivalence of the Formulations

This section proves the equivalence between the constrained and generalized coordinate formulations.

Theorem 2.1 *Let g be the constraint function and ψ be the coordinate chart defined above. Let q_k and q_{k-1} be the two initial points in the coordinate chart and let $v_k = \psi(q_k)$ and $v_{k-1} = \psi(q_{k-1})$. Let $Dg(v_k)$ and $D\psi(q_k)$ be full rank. Then the generalized formulation, Equation (2.41), has a solution for q_{k+1} if and only if the constrained formulation, Equation (2.28), has a solution for v_{k+1} and λ_k. Furthermore, $v_{k+1} = \psi(q_{k+1})$.*

Proof.

($\Leftarrow$) Assume that there is a solution for v_{k+1} for the constrained formulation. Let $q_{k+1} = \psi^{-1}(v_{k+1})$ and it will be shown that q_{k+1} solves Equation (2.41). Multiply the top equation in Equation (2.28) on the left by $D^T\psi(q_{k+1})$. Also, substitute $v_k = \psi(q_k)$ and $v_{k-1} = \psi(q_{k-1})$ into Equation (2.28). Notice that $g(\psi(q_k)) = 0$ which implies that $Dg(\psi(q_k))D\psi(q_k) = 0$. Using the substitutions and the fact that $D^T\psi(q_k)D^Tg(\psi(q_k)) = 0$ proves that q_{k+1} is a solution for Equation (2.41).

($\Rightarrow$) To complete the proof, it is assumed that q_{k+1} is a solution for Equation (2.41), and it is then shown that there exists a Lagrange multiplier, λ_k, so that $v_{k+1} = \psi(q_{k+1})$ is a solution for Equation (2.28). Substitute the expressions for v_{k+1}, v_k, and v_{k-1} into Equation (2.28). The lower equation in Equation (2.28) is solved automatically since $v_{k+1} \in Q$. Note that $T_{v_k}V = \mathcal{R}(D\psi(q_k)) \oplus \mathcal{N}(D^T\psi(q_k))$ and that $\mathcal{R}(D^Tg(v_k)) \subset \mathcal{N}(D^T\psi(q_k))$. Since $D^Tg(v_k)$ is full rank and $\dim(\mathcal{R}(D^Tg(v_k))) = \dim(\mathcal{N}(D^T\psi(q_k)))$, $\mathcal{R}(D^Tg(v_k))$ equals $\mathcal{N}(D^T\psi(q_k))$. Then split the left-hand side in Equation (2.28) into a component in $\mathcal{R}(D\psi(q_k))$ and an orthogonal component in $\mathcal{N}(D^T\psi(q_k))$. The component in $\mathcal{R}(D\psi(q_k))$ is zero by Equation (2.41) and the fact that $\mathcal{R}(D^Tg(v_k)) = \mathcal{N}(D^T\psi(q_k))$. A Lagrange multiplier, λ_k, can then be found to make the component in $\mathcal{N}(D^T\psi(q_k))$ equal to zero since $\mathcal{R}(D^Tg(v_k)) = \mathcal{N}(D^T\psi(q_k))$. Therefore, there exists a λ_k so that $v_{k+1} = \psi(q_{k+1})$ solves Equation (2.28). $\square$

The relationship between constrained and generalized coordinate formulations for discrete-time mechanics as well as continuous-time mechanics is shown in Figure 2.1. The figure also points out where the discrete-time equations approximate the flow of the

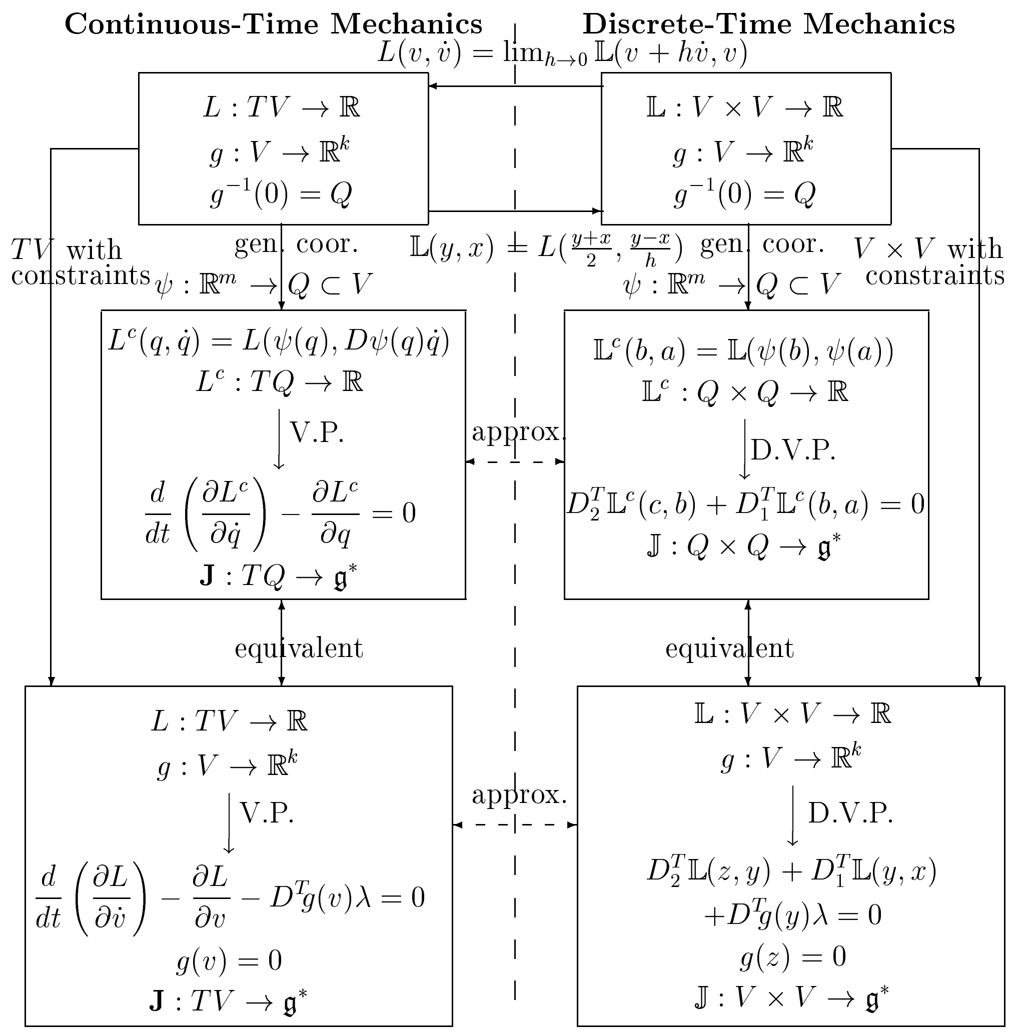
Continuous-Time Mechanics
Discrete-Time Mechanics
$L(v,\dot v) = \lim_{h\to 0} \mathbb{L}(v + h\dot v, v)$
$L : TV \to \mathbb{R}$
$g : V \to \mathbb{R}^k$
$g^{-1}(0) = Q$
$\mathbb{L} : V \times V \to \mathbb{R}$
$g : V \to \mathbb{R}^k$
$g^{-1}(0) = Q$
TV with constraints
gen. coor. $\psi : \mathbb{R}^m \to Q \subset V$
$\mathbb{L}(y,x) \doteq L(\frac{y+x}{2}, \frac{y-x}{h})$
gen. coor. $\psi : \mathbb{R}^m \to Q \subset V$
$V \times V$ with constraints
$L^c(q,\dot q) = L(\psi(q), D\psi(q)\dot q)$
$L^c : TQ \to \mathbb{R}$
V.P.
$\frac{d}{dt}\left(\frac{\partial L^c}{\partial \dot q}\right) - \frac{\partial L^c}{\partial q} = 0$
$\mathbf{J} : TQ \to \mathfrak{g}^*$
approx.
$\mathbb{L}^c(b,a) = \mathbb{L}(\psi(b), \psi(a))$
$\mathbb{L}^c : Q \times Q \to \mathbb{R}$
D.V.P.
$D_2^T \mathbb{L}^c(c,b) + D_1^T \mathbb{L}^c(b,a) = 0$
$\mathbb{J} : Q \times Q \to \mathfrak{g}^*$
equivalent
equivalent
$L : TV \to \mathbb{R}$
$g : V \to \mathbb{R}^k$
V.P.
$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot v}\right) - \frac{\partial L}{\partial v} - D^T g(v)\lambda = 0$
$g(v) = 0$
$\mathbf{J} : TV \to \mathfrak{g}^*$
approx.
$\mathbb{L} : V \times V \to \mathbb{R}$
$g : V \to \mathbb{R}^k$
D.V.P.
$D_2^T \mathbb{L}(z,y) + D_1^T \mathbb{L}(y,x)$
$+ D^T g(y)\lambda = 0$
$g(z) = 0$
$\mathbb{J} : V \times V \to \mathfrak{g}^*$

Figure 2.1: Comparison of Continuous and Discrete Formulations of Mechanics

continuous-time equations. The results for continuous-time mechanics are summarized on the left side of the figure. It is assumed that there is an unconstrained Lagrangian with constraint functions as shown in the upper left corner. One can use generalized coordinates and apply Hamilton's principle to produce the Euler-Lagrange equations or one can use constrained coordinates and enforce the constraints through Lagrange multipliers. The right side of the figure summarizes the results for discrete-time mechanics. Given the continuous, unconstrained Lagrangian, one can form the discrete, unconstrained Lagrangian. One can proceed analogously to continuous-time mechanics by using generalized or constrained coordinates. In Section 2.3.5, it is discussed how the discrete equations approximate the continuous-time equations.

2.3.4 Jacobian Structure

The DEL equations, Equation (2.27), are solved using Newton-Raphson equation solvers. These solvers require the construction of a Jacobian formed by differentiating Equation (2.27) with respect to v_{k+1} and λ_k to get

$$J(v_{k+1}, v_k, h) = \begin{bmatrix} D_{12}\mathbb{L}(v_{k+1}, v_k) & D^T g(v_k) \\ Dg(v_{k+1}) & 0 \end{bmatrix}, \tag{2.43}$$

where

$$[D_{12}\mathbb{L}(v_{k+1}, v_k)]_{ij} = \frac{\partial^2 \mathbb{L}}{\partial v_k^i \partial v_{k+1}^j}(v_{k+1}, v_k).$$

For many applications, the nearly symmetric Jacobian, Equation (2.43), is a sparse matrix and sparse matrix techniques can be used in the Newton-Raphson steps to increase the simulation efficiency. For tree-structured multibody systems, it is shown in Section 5.4 that the linear equations involving the Jacobian can be solved in linear time. The authors in (Barth and Leimkuhler, 1996b) particulate the rigid bodies in a multibody system with point masses. They then use symplectic-momentum integrators with constraints and general sparse matrix techniques to simulate multibody systems. The author in (Reich, 1996b) uses the methods in (Reich, 1996a) to create symplectic-momentum integrators for multibody systems.

2.3.5 Local Truncation Error and Solvability

Results on truncation error and solvability are presented in this section. The definition of local truncation error in (Lambert, 1991)(page 56) is followed. To calculate the truncation error, first insert an exact solution of the differential equations into the algorithm equations in Equation (2.41), and then expand the resulting equation in terms of the step size h. To calculate the expansion, it is easier to first expand Equation (2.41) about

$$v_k^i = \psi^i(q_k) \qquad \text{and} \qquad \dot{v}_k^i = \frac{\partial \psi^i}{\partial q_k^j}\dot{q}_k^j, \tag{2.44}$$

and then expand the result into powers of h. This lengthy calculation which is not reproduced here reveals that the local truncation error of the method is second order. The first term, h^0, is zero since $q, \dot{q}$ satisfy the continuous Euler-Lagrange equations. The second term, h^1, is zero through a cancellation of terms. The h^2 term is non-zero, and the coefficient is a lengthy expression involving second, third, and fourth partial derivatives of $L : TV \to \mathbb{R}$.

If one uses the following definition for the discrete Lagrangian:

$$\mathbb{L}(y, x) = L(y, \frac{y - x}{h}), \tag{2.45}$$

then the resulting DEL equations will only be first order accurate for a general Lagrangian. There is no cancellation of terms in the h^1 term as there is with the definition in Equation (2.23). However, in some cases, the resulting DEL equations may be explicit while the DEL equations from the definition in Equation (2.23) are implicit. An example of this occurring is if the continuous Lagrangian is in the form in Equation (2.29), and there are no constraints.

The existence of a solution for the continuous-time equations is related to the solvability of the generalized coordinate discrete equations. One can show that if $D_{22}L$ is non-singular and if the Jacobian of the constraints is full rank, then for a sufficiently small time step, the generalized coordinate DEL equations are solvable for q_{k+1}. This is proved by showing that the DEL equations have a solution for $h = 0$ by taking the limit and then by using the implicit function theorem to conclude that there is a solution in a neighborhood of $h = 0$. Theorem 2.1 then implies that there is also a solution for the DEL equations with Lagrange multipliers.

2.3.6 Symplectic Form and Discrete Momentum Map

The numerical integration methods created through the construction procedure are symplectic-momentum integrators; however, this statement requires clarification which is presented in this section. The integrators are symplectic in that the map produced on T^*V or T^*Q is a symplectic map. Also, if the Lie group acts linearly on V, then the continuous flow of the Euler-Lagrange equations and the discrete map produced from the DEL equations preserve the same momentum map on T^*Q.

However, if one integrates the continuous equations exactly or accurately and uses the result to initialize the discrete equations, one will notice that the *value* of the momentum map will differ from the value of the momentum map for the continuous system. The difference arises from the difference in the assignment of the momentum coordinate in T^*V through the fiber derivative. In the continuous case, the momentum is D_2L while in the discrete case, $-hD_2\mathbb{L}$ is used. A $-h$ is used from the definitions given in Equation (2.4) because $-hD_2\mathbb{L}$ converges to D_2L as $h \to 0$.

If the Lagrangian of a continuous system is invariant to the action of a group, and if the constraints are also invariant under the group action, *i.e.*

$$L : TV \to \mathbb{R}$$

$$L\left(G\cdot v, G\cdot \dot{v}\right) = L\left(v, \dot{v}\right)$$

$$g\left(G\cdot v\right) = g\left(v\right),$$

where the action of G on $v \in V$ is represented as $G\cdot v$, then the flow of the Euler-Lagrange equations preserve the momentum map,

$$\mathbf{J} : TV \to \mathfrak{g}^*,$$

where

$$\left\langle \mathbf{J}\left(v, \dot{v}\right), \xi\right\rangle \triangleq \left\langle \frac{\partial L}{\partial \dot{v}}\left(v, \dot{v}\right), \xi_V(v)\right\rangle.$$

If the group G also acts linearly on V, then the discrete Lagrangian is also invariant to the group action through the following calculation:

$$\begin{aligned}
\mathbb{L}(G\cdot v_{k+1}, G\cdot v_k) &= L\left(\frac{G\cdot v_{k+1} + G\cdot v_k}{2}, \frac{G\cdot v_{k+1} - G\cdot v_k}{h}\right) \\
&= L\left(G\cdot\left(\frac{v_{k+1}+v_k}{2}\right), G\cdot\left(\frac{v_{k+1}-v_k}{h}\right)\right) \\
&= L\left(\frac{v_{k+1}+v_k}{2}, \frac{v_{k+1}-v_k}{h}\right) \\
&= \mathbb{L}(v_{k+1}, v_k).
\end{aligned}$$

From a similar derivation to the derivation in Section (2.2.3), one can show that the following momentum map

$$\mathbb{J} : V \times V \to \mathfrak{g}^*$$

defined by the relation

$$\langle \mathbb{J}(v_{k+1}, v_k), \xi \rangle \overset{\triangle}{=} \langle D_2\mathbb{L}(v_{k+1}, v_k), \xi_V(v_k) \rangle$$

is conserved by the flow of the DEL equations.

Calculate $-hD_2\mathbb{L}$ and notice that

$$-hD_2\mathbb{L}(v_{k+1}, v_k) = -h\frac{\partial}{\partial v_k}\left(L\left(\frac{v_{k+1}+v_k}{2}, \frac{v_{k+1}-v_k}{h}\right)\right)$$
$$= \frac{\partial L}{\partial \dot{v}}\left(\frac{v_{k+1}+v_k}{2}, \frac{v_{k+1}-v_k}{h}\right) - \frac{h}{2}\frac{\partial L}{\partial v}\left(\frac{v_{k+1}+v_k}{2}, \frac{v_{k+1}-v_k}{h}\right).$$

As $h \to 0$, the discrete momentum value, $-hD_2\mathbb{L}$, converges to the continuous momentum value, D_2L. Therefore, the quantities that depend on the discrete momentum value, such as the *discrete momentum map* defined to be $-h\mathbb{J}$, converge to their continuous counterparts as $h \to 0$.

2.4 Numerical Examples

The construction procedure is now applied to produce mechanical integrators for the rigid body (RB) and the double spherical pendulum (DSP). The constrained coordinate formulation is used instead of generalized coordinates to avoid coordinate singularities and coordinate patching. Unit quaternions are used to create the rigid body algorithm, and the position of the two masses are used for the double spherical pendulum. The double spherical pendulum algorithm is then compared to an energy-momentum algorithm presented in (Wendlandt, 1995) based on the work in (Gonzalez, 1996a).

In the simulations, energy is used as a monitor to catch any obvious problems, as in (Channell and Scovel, 1990) and (Simo and Gonzalez, 1993). For a discussion of circumstances under which this can be justified, see (Sanz-Serna, 1991)(page 277–278), (Sanz-Serna and Calvo, 1994)(page 139–140), and (Benettin and Giorgilli, 1994). In general, energy conservation alone does not imply good performance as is shown in (Ortiz, 1986). In the DSP example, energy is observed to oscillate around a constant value, which is taken as a good indication.

When energy-momentum and symplectic-momentum integration methods are compared, it should be kept in mind that energy-momentum methods should be monitored on how well they conserve the symplectic form. This is of course not so straightforward as monitoring using the energy, since the symplectic condition involves computing the derivative of the flow map (*e.g.*, using a cloud of initial conditions). This point is not directly addressed in this dissertation, but it is important to keep them in mind.

2.4.1 Rigid Body

The algorithm presented here updates quaternion variables based on the previous two quaternion variables. The configuration manifold is taken to be $Q = S^3 \subset V$ where $V = \mathbb{R}^4$. Quaternions were used instead of using $V = \mathbb{R}^9$ with the six orthogonal constraints of $SO(3)$ primarily to avoid a large number of Lagrange multipliers. The constraint function is $g(v) = v \cdot v - 1$ and is enforced with a Lagrange multiplier. The use of generalized coordinates to eliminate the use of Lagrange multipliers introduces the problem of coordinate switching.

Rigid body integrators that preserve certain mechanical properties have been created by several researchers. A symplectic integrator which preserves the momentum and energy is presented in (Lewis and Simo, 1995). An energy-momentum integrator is presented in (Simo and Wong, 1991). A symplectic-momentum integrator is presented in (McLachlan and Scovel, 1995). A rigid body integrator based on a discrete variational principle and in terms of 3×3 matrices with constraints is presented in (Moser and Veselov, 1991). It would be interesting to compare in more detail the integrator in (Moser and Veselov, 1991) to the quaternion-based integrator in this chapter.

First attach a body frame to the rigid body and represent the frame as a matrix, $R \in SO(3)$, which maps vectors in the body frame, $\mathcal{B}$, to vectors in the spatial (inertial) frame, $\mathcal{S}$. The rotation matrix is then thought of as a mapping, $R : \mathcal{B} \to \mathcal{S}$.

The continuous-time Lagrangian, $L : TV \to \mathbb{R}$, is constructed using the quaternion relationships shown in Appendix A for the body angular velocity:

$$L(q, \dot{q}) = \frac{1}{2}(2\overline{q} \star \dot{q})^T \begin{bmatrix} 0 & 0 \\ 0 & \mathbb{I} \end{bmatrix} (2\overline{q} \star \dot{q}), \tag{2.46}$$

where $\mathbb{I}$ is the inertia matrix. The constraint is the unit norm constraint for quaternions, $q_s^2 + q_v \cdot q_v - 1 = 0$.

The Lagrangian in Equation (2.46) is invariant under left quaternionic multiplication, *i.e.*

$$L(r \star q, r \star \dot{q}) = L(q, \dot{q}),$$

where r is a unit quaternion. The invariance leads to conservation of angular momentum.

The discrete Lagrangian, $\mathbb{L} : TV \to \mathbb{R}$, is chosen to be

$$\mathbb{L}(y, x) = L\left(\frac{y + x}{2}, \frac{y - x}{h}\right). \tag{2.47}$$

The body angular velocity term is first simplified to get

$$2\overline{q} \star \dot{q} \mapsto \left(\overline{\frac{y+x}{2}}\right) \star \left(\frac{y-x}{h}\right) \tag{2.48}$$

$$= \frac{1}{h}(\overline{y} \star y - \overline{y} \star x + \overline{x} \star y - \overline{x} \star x). \tag{2.49}$$

Restricted to Q, $\overline{y} \star y = \overline{x} \star x = (1,0,0,0)$. Simplifying restricted to Q gives

$$2\overline{q} \star \dot{q} \mapsto \frac{1}{h}(\overline{x} \star y - \overline{y} \star x). \tag{2.50}$$

Equation (2.50) is an approximation to the body angular velocity, $(0, \omega_b)$. The simplified discrete Lagrangian restricted to Q is then

$$\mathbb{L}(y,x) = \frac{1}{2h^2}(\overline{x} \star y - \overline{y} \star x)^T \begin{bmatrix} 0 & 0 \\ 0 & \mathbb{I} \end{bmatrix} (\overline{x} \star y - \overline{y} \star x), \tag{2.51}$$

and the discrete Lagrangian on all of $V \times V$ is then taken to be equal to Equation (2.51). Since $\mathbb{S}$ restricted to Q is extremized, the extension of $\mathbb{L}$ to $V \setminus Q$ is arbitrary.

The discrete Lagrangian in Equation (2.51) is also invariant under left quaternionic multiplication, *i.e.*

$$\mathbb{L}\left(r \star y, r \star x\right) = \mathbb{L}\left(y, x\right),$$

where r is a unit quaternion, and the invariance leads to conservation of discrete momentum which converges to the continuous momentum as the step size decreases.

The DEL equations for the RB and relevant Jacobian are created in Mathematica (Wolfram, 1991) and exported to C-code for simulation. The initial conditions and RB

Table 2.1: Simulation Results for the Rigid Body Simulation

h (s)	CPU time (s)	Quat. Error	Energy Error	Mom. Error
0.0001	97.620	0.0	6.256e-7	1.671e-7
0.001	9.905	3.960e-6	6.274e-5	1.687e-5
0.01	1.397	3.997e-4	6.274e-3	1.687e-3
0.1	0.301	3.648e-2	6.217e-1	1.665e-1

parameters are

$$q_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad \omega_b = \begin{bmatrix} 0 \\ 3 \\ 4 \end{bmatrix} \qquad \mathbb{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}. \tag{2.52}$$

The rigid body integrator is first initialized by choosing two initial quaternion values. An Euler step with $\dot{q} = q \star (0, \omega_b/2)$ with $h = 10^{-5}$s initializes the system. The DVP integrator with $h = 10^{-5}$s is then used to set the second initial point for $h = 10^{-4}$s, 10^{-3}s, 10^{-2}s, and 10^{-1}s. The system is simulated for 30 seconds. To calculate errors in energy, momentum, and position, a standard value is chosen. The energy and momentum given initially after the first Euler step at $h = 10^{-5}$s is used for the standard energy and momentum values. The results of the 30s simulation with $h = 10^{-4}$s is used for the standard position variables. The following formula is used to calculate errors for each simulation:

$$\text{error} = \frac{1}{Nm} \sum_{i=1}^{N} \| v_i - v_i^s \|_2, \tag{2.53}$$

where m is the length of the vector v_i, v_i^s is the standard value at the ith sample, and N is the number of samples. The results of the simulations are tabulated in Table 2.4.1. The table lists CPU time on a SGI Indy (1 100 MHZ IP22 Processor, FPU: MIPS R4610 Floating

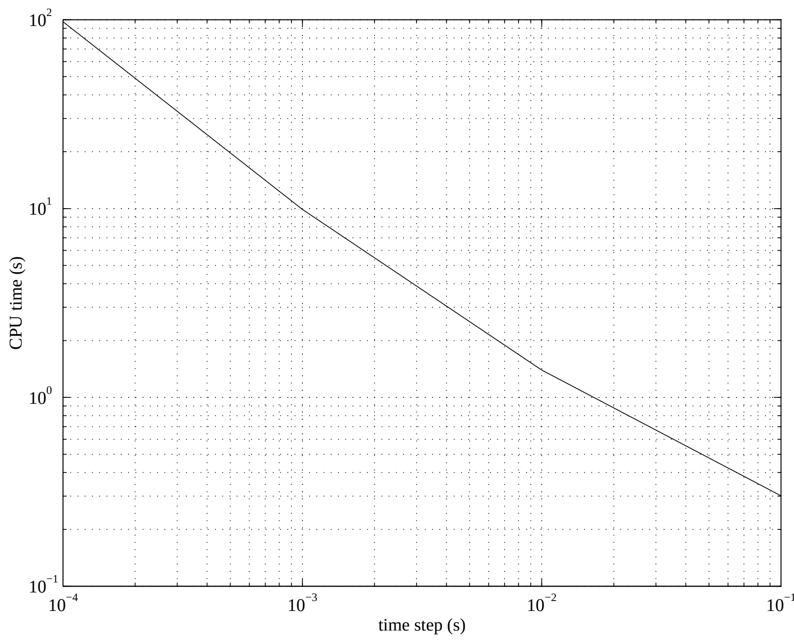

Figure 2.2: CPU Time Versus Time Step for the Rigid Body Simulation

Point, CPU:MIPS R4600 Processor), quaternion error, energy error, and momentum error.

Figure 2.2 is a log-log plot of CPU time in seconds versus time step in seconds. The CPU time drops off nearly linearly as the time step increases. The CPU time is corrected for the time it takes to initialize each simulation with the $h = 10^{-5}$s simulation.

The quaternion error versus time step is shown in Figure 2.3. The plot shows a second order relationship between error and time step.

Figure 2.4 compares the plot of the quaternion, q_y, versus time for the simulations at $h = 10^{-4}$s and $h = 10^{-1}$s. The trajectory for the large time step exhibits the same qualitative behavior as the small time step, but the deviations increase for longer simulation times.

The energy error versus time step is shown in Figure 2.5. The figure reveals a

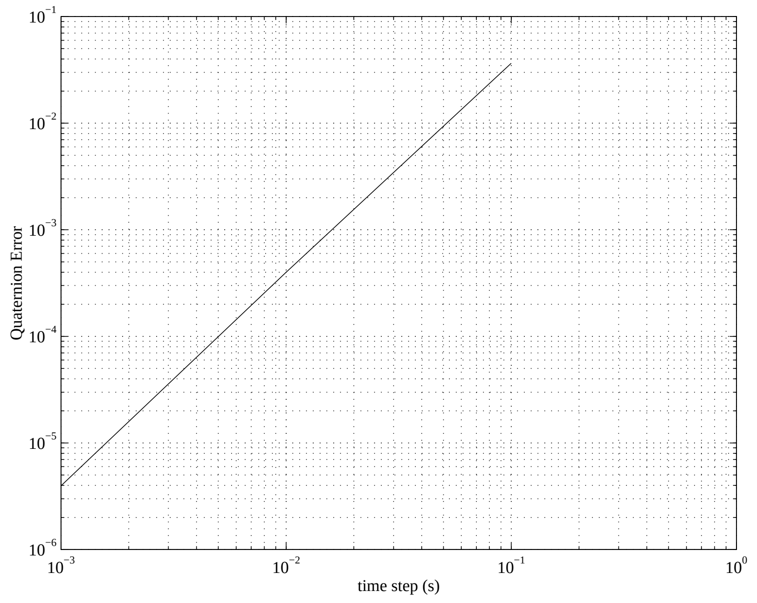

Figure 2.3: Quaternion Error Versus Time Step

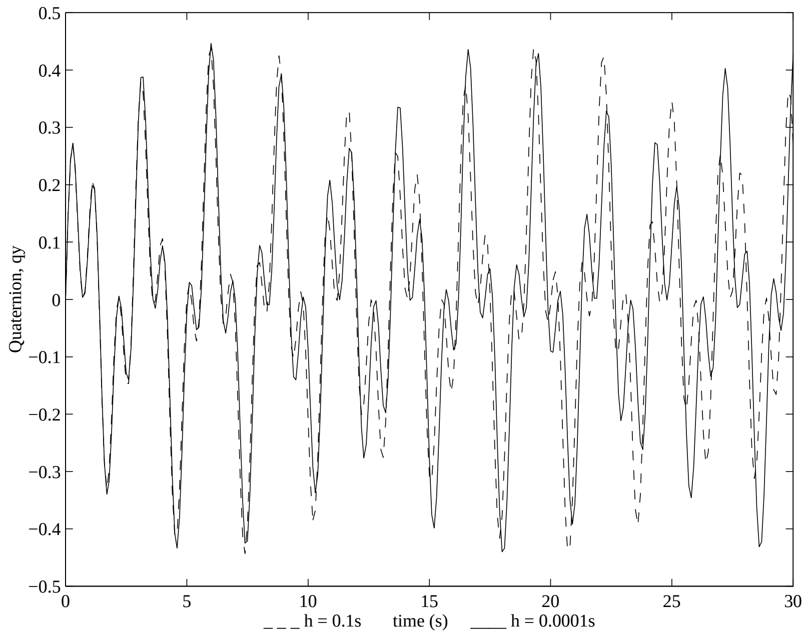

Figure 2.4: Quaternion Coordinate Versus Time

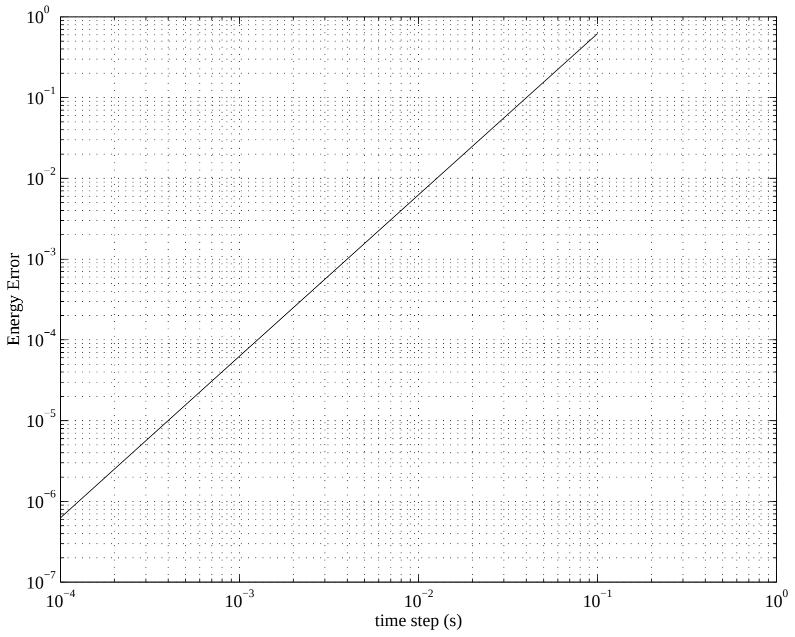

Figure 2.5: Energy Error Versus Time Step for the Rigid Body Simulation

second order relationship between energy error and time step. The energy for the $h = 10^{-4}$s simulation deviates between $32.9999993\underline{59}$J and $32.9999993\underline{49}$J. The energy for the simulation at $h = 10^{-3}$s deviates between $32.99993723\underline{6}$J and $32.99993723\underline{5}$J. There is no deviation in energy for the $h = 10^{-2}$s and $h = 10^{-1}$s simulations.

For each time step, the constant value of the discrete momentum map is conserved; however, as explained in Section 2.3.6, the value converges to the continuous momentum value as the step size decreases. The convergence of the discrete momentum is shown in Figure 2.6. The figure reveals a second order relationship between momentum error and time step. The angular momentum for each simulation should remain constant, but there are small deviations ($\pm 10^{-9}$) in the data for the $h = 10^{-4}$s simulation. There are no deviations in the momentum value for the other simulations.

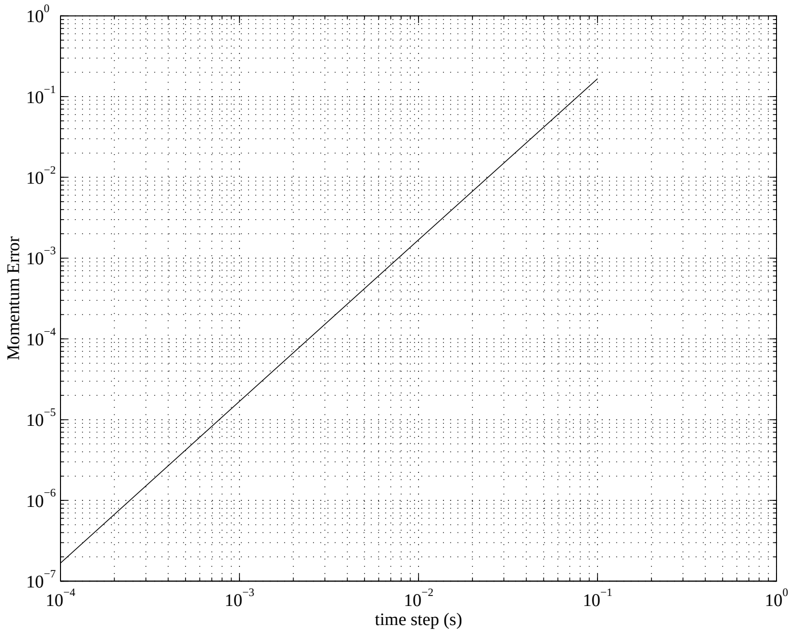

Figure 2.6: Momentum Error Versus Time Step for the Rigid Body Simulation

2.4.2 Double Spherical Pendulum

The double spherical pendulum consists of two constrained point masses. The configuration space is $Q = S^2 \times S^2$ and the linear space is $V = \mathbb{R}^3 \times \mathbb{R}^3$. The position of the first mass is $q_1 = (x_1, y_1, z_1)$, and the position of the second mass is $q_2 = (x_2, y_2, z_2)$. The constraint equation, given by the pendulum length constraints, is

$$g(v) = \begin{bmatrix} q_1 \cdot q_1 - l_1^2 \\ \\ (q_2 - q_1) \cdot (q_2 - q_1) - l_2^2 \end{bmatrix}. \tag{2.54}$$

The DSP Lagrangian system is of the form in Equation (2.29), and the DEL equations for this system are of the form in Equation (2.30). The DVP algorithm for the

DSP is the SHAKE algorithm:

$$\frac{1}{h}M\left[q^{n+1}-2q^{n}+q^{n-1}\right]+h\begin{bmatrix}0\\0\\m_1\mathrm{g}\\0\\0\\m_2\mathrm{g}\end{bmatrix}-hD^{T}g\left(q^{n}\right)\lambda=0$$

$$g\left(q^{n+1}\right)=0,$$

where

$$M=\begin{bmatrix}m_1I & 0\\ 0 & m_2I\end{bmatrix},\qquad q=\begin{bmatrix}q_1\\ q_2\end{bmatrix},\tag{2.55}$$

and m_1 and m_2 are the masses.

The simulation from the discrete variational principle (DVP) construction is now compared to an energy-momentum (EM) formulation based on the construction procedure in (Gonzalez, 1996a), and applied to the DSP in (Wendlandt, 1995). The EM algorithm for the DSP is

$$q_1^{n+1}-q_1^{n}-h\frac{1}{m_1}p_1^{n+\frac{1}{2}}=0$$

$$q_2^{n+1}-q_2^{n}-h\frac{1}{m_2}p_2^{n+\frac{1}{2}}=0$$

$$p^{n+1} - p^{n} + h \left[\begin{bmatrix} 0 \\ 0 \\ m_1 \mathrm{g} \\ 0 \\ 0 \\ m_2 \mathrm{g} \end{bmatrix} + \lambda_1 \begin{bmatrix} q_1^{n+1} + q_1^{n} \\ 0 \end{bmatrix} + \lambda_2 \begin{bmatrix} q_1^{n+1} + q_1^{n} - q_2^{n+1} - q_2^{n} \\ q_2^{n+1} + q_2^{n} - q_1^{n+1} - q_1^{n} \end{bmatrix} \right] = 0$$

$$\left(q_1^{n+1}\right) \cdot \left(q_1^{n+1}\right) - l_1^2 = 0$$

$$\left(q_2^{n+1} - q_1^{n+1}\right) \cdot \left(q_2^{n+1} - q_1^{n+1}\right) - l_2^2 = 0,$$

where p_i is the momentum for the ith mass, p is the six vector of momentum formed by stacking p_1 and p_2, and

$$p_i^{n+\frac{1}{2}} = \frac{1}{2}\left(p_i^{n+1} + p_i^{n}\right).$$

The following parameters are used for the DSP: $m_1 = 2.0$Kg, $m_2 = 3.5$Kg, $l_1 = 4.0$m, $l_2 = 3.0$m, and $\mathrm{g} = 9.81\mathrm{m/s}^2$. The initial conditions are $x_1 = 2.820$m, $y_1 = 0.025$m, $x_2 = 5.085$m, $y_2 = 0.105$m, $\dot{x}_1 = 3.381$m/s, $\dot{y}_1 = 2.506$m/s, $\dot{x}_2 = 2.497$m/s, and $\dot{y}_2 = 10.495$m/s. The position and velocity of the z-coordinate is determined from the constraints, and the z-coordinate for both masses is taken to be negative.

To calculate errors in energy, momentum, and position, a standard value is chosen. The following formula is then used to calculate errors for each simulation:

$$\text{error} = \frac{1}{Nm} \sum_{i=1}^{N} \parallel v_i - v_i^s \parallel_2, \tag{2.56}$$

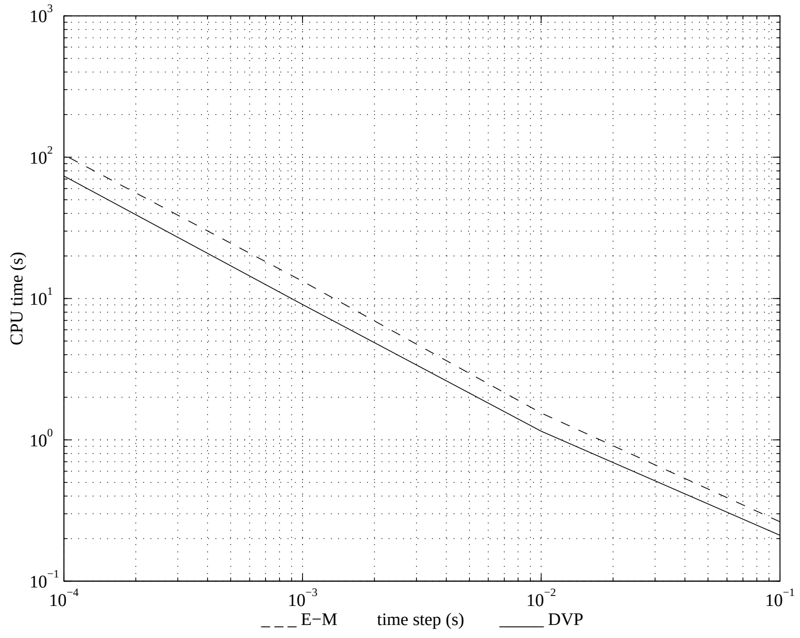

Figure 2.7: CPU Time Versus Time Step for the DSP Simulation

where m is the length of the vector v_i, v_i^s is the standard value at the ith sample, and N is the number of samples.

The output of the EM simulation at a time step of 0.0001s is used as the standard and initializes the second step in the DVP simulations. The results of the EM simulations and the DVP simulations are summarized in Table 2.4.2. The table contains the CPU time, position error, energy error and momentum error for the EM and the DVP simulations. The energy and momentum error for the EM simulations are zero. Equation (2.56) is used to calculate the errors for the DSP simulations.

Figure 2.7 is a plot of CPU time versus time step for the EM and DVP simulations. The DVP simulations are slightly faster for each time step and both CPU times drop off nearly linearly with increasing time step.

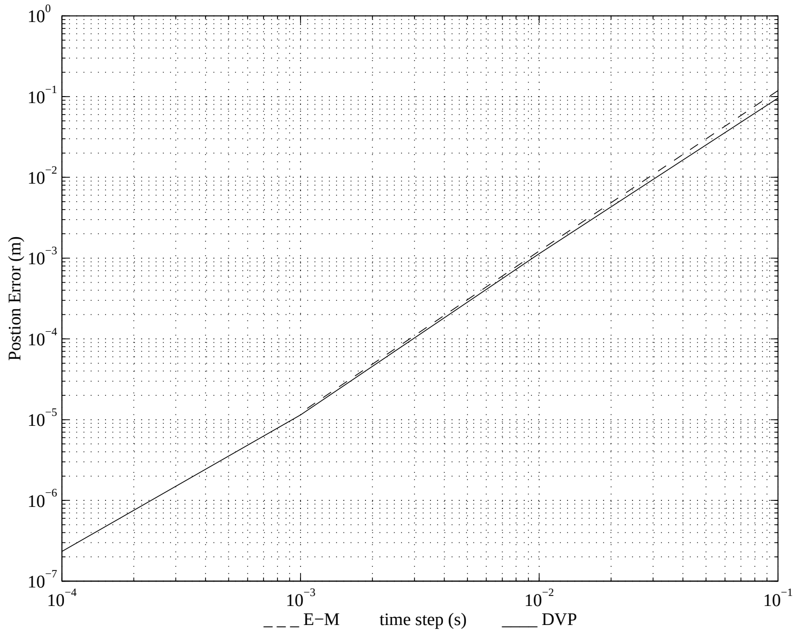

Figure 2.8: Position Error Versus Time Step for the DSP Simulation

The position error for the EM and DVP simulations is shown in Figure 2.8. Both simulations show a second order relationship between position error and time step. The error for the EM simulation is slightly greater than the error for the DVP simulation for $h \geq 10^{-3}$s.

The y position of the second mass is shown in Figure 2.9 for the EM and DVP simulations for $h = 0.0001$s and $h = 0.1$s. Both the EM and DVP simulations at $h = 0.0001$s overlap and cannot be distinguished when plotted on the same graph. For both the EM and DVP simulations, reasonably accurate and fast trajectories are produced at large time steps, $h = 0.1$s. Both simulation methods may have uses in interactive simulation applications, such as design and animation, where real-time, reasonably accurate simulations are important.

Table 2.2: Simulation Results for the DSP Simulation

h (s)	Method	CPU time (s)	Pos. Error	Energy Error	Mom. Error
0.0001	DVP	73.648	2.329e-7	6.475e-6	2.547e-6
	EM	103.871	0.0	0.0	0.0
0.001	DVP	9.065	1.146e-5	3.269e-4	1.707e-4
	EM	13.250	1.214e-5	0.0	0.0
0.01	DVP	1.152	1.135e-3	3.224e-2	1.696e-2
	EM	1.549	1.225e-3	0.0	0.0
0.1	DVP	0.211	9.576e-2	2.665	1.560
	EM	0.263	1.184e-1	0.0	0.0

The error in energy versus time step is shown in Figure 2.10. The DVP energy error appears to drop off as the square of the time step, at least for the large time steps. The energy error is zero for all time steps for the EM simulation. The energy for the DVP simulation at $h = 0.0001$s deviates between $24.94449\underline{5109}$J and $24.94449\underline{9828}$J and deviates between 20.910805793J and 25.583335766J for $h = 0.1$s.

The error in the momentum about the z-axis is shown in Figure 2.11. The momentum error for the EM simulation is zero for all time steps. The DVP algorithm should preserve momentum but for the smallest time step, $h = 0.0001$s, the momentum varies between $199.8254671\underline{70}\text{m}^2/\text{s}$ to $199.8254671\underline{84}\text{m}^2/\text{s}$. The variation may be due to numerical errors. The momentum is constant for the other time steps. Again, the constant discrete momentum value approaches the value of the continuous momentum as the step size decreases.

Figure 2.12 shows the energy for the DVP simulations versus time for $h = 0.1$s and 0.01s in the lower graph. The upper graph shows energy versus time for $h = 0.001$s and 0.0001s. The energy oscillates about a constant value, and the constant value approaches the true energy. The amplitude of the oscillations decrease as the step size decreases. The

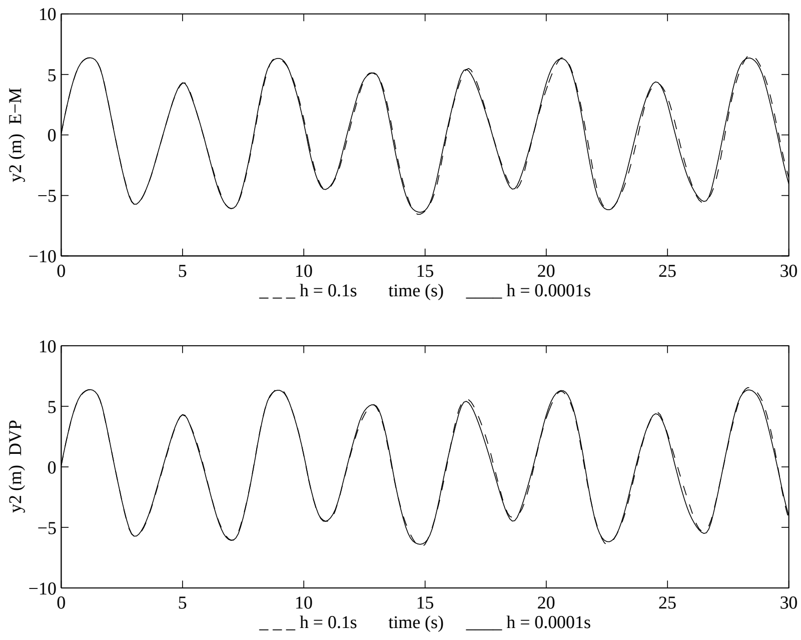

Figure 2.9: Position Coordinate Versus Time for the DSP Simulation

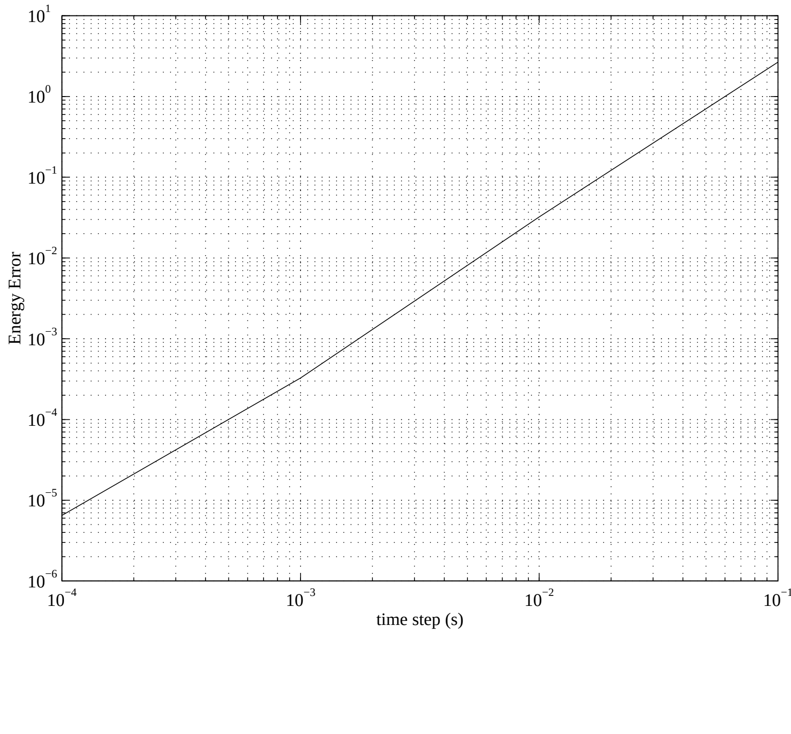

Figure 2.10: Energy Error Versus Time Step for the DSP Simulation

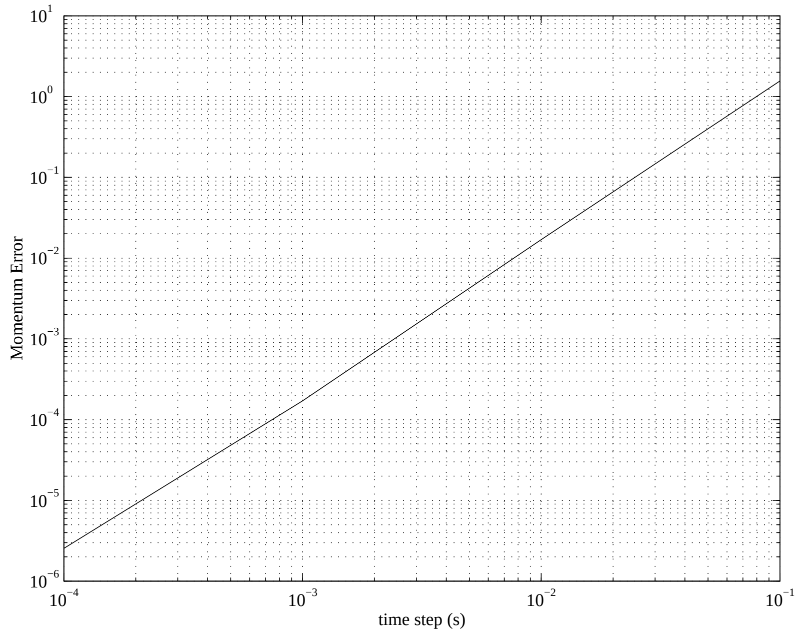

Figure 2.11: Momentum Error Versus Time Step for the DSP Simulation

fluctuations in energy appear to be related to the constraint forces. The middle graph is a plot of the multipliers versus time, and the fluctuations in the multipliers are correlated to the fluctuations in energy. This relationship has also been noticed in (Barth and Leimkuhler, 1996a), and they use variable step size to decrease the energy oscillation.

2.5 Chapter Summary

In this chapter, a general approach to discrete mechanics was formulated and developed. A general method to construct symplectic-momentum mechanical integrators for Lagrangian systems with holonomic constraints was then presented. The method was then applied to the rigid body and the double spherical pendulum. The discrete Euler-Lagrange (DEL) equations share many similarities to the continuous-time equations of motion and

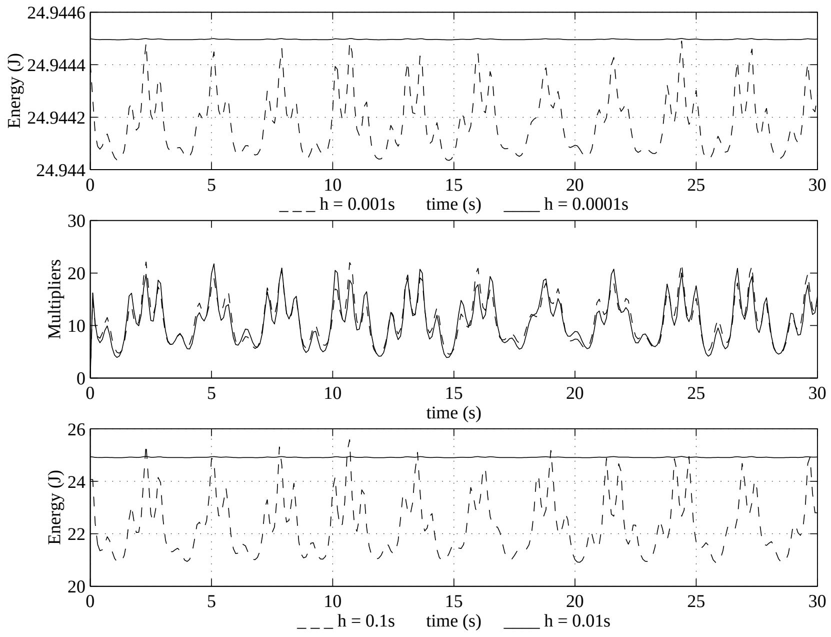

Figure 2.12: Energy and Multipliers Versus Time for the DSP Simulation

preserve a symplectic form and invariants resulting from group invariance of the Lagrangian.

There are many areas of future work and development. A few of these are listed here.

Energy-Momentum Integrators. One may proceed analogously to the derivation in this paper to create energy-momentum integrators possibly based on discretizing the principle of least action.

Nonholonomic Systems. The method presented in this paper treats holonomic constraints and one would like to generalize the method to treat nonholonomic constraints, as in (Bloch et al., 1996). For nonholonomic systems, the standard symplectic form is not preserved, and there are momentum equations and not conservation laws. Also, energy can be conserved in these systems. One has to develop algorithms taking into account these effects.

Multistep Methods, Stages, and Time Step Control. It seems possible to modify the method to construct multistep mechanical integrators or mechanical integrators with multiple stages to increase the accuracy of the method. One would also like to modify the method to allow variable time steps to improve efficiency.

External Forces. It would also be desirable to generalize the method to include external forces. This should be straightforward since they can be included in Hamilton's principle in standard fashion. One would also like to add control forces and dissipative forces to simulate controlled mechanical systems. The first author is currently using the techniques presented in this paper to develop a multibody simulator to simulate control systems for human models (see (Wendlandt and Sastry, 1996)).

Spacetime Integrators. Since the method here is variational by nature and focuses on the temporal behavior, it should be helpful in the development of spacetime integrators by synthesis with existing finite element methods.

Chapter 3

Recursive Multibody Kinematics and Dynamics

The kinematics and dynamics for tree-structured multibody systems with general joints are developed in this chapter as well as efficient, linear-time, recursive algorithms to compute both the dynamics and the kinematics. The primary purpose of this chapter is to serve as a general reference (especially for robotics researchers familiar with (Murray et al., 1994)) and to provide the necessary background for Chapters 4 and 5.

The development in this chapter is influenced by several sources. The Lie group kinematic formulation in (Murray et al., 1994) serves as a basis for the derivations in this chapter. The notation succinctly represents configurations of frames, mappings between frames, body velocities, and relative velocities. The emphasis in (Murray et al., 1994) is on the spatial representation of twists, while the scalable recursive methods favor the use of relative mappings between body frames. The emphasis in this dissertation is to develop computational methods for multibody systems; therefore, the Lie group formalism is used with an emphasis on relative transformations. The methodology in (Murray et al., 1994) is also extended from serial chains to the larger class of tree-structured multibody systems.

In the forward kinematics problem, the pose, velocity, and acceleration of each rigid body in the multibody system is calculated. In this chapter, recursive algorithms for calculating the forward kinematics are derived in a straightforward manner based on relative mappings between body frames.

The forward kinematics algorithms are then extended to calculate the inverse dynamics. In the inverse dynamics problem, the joint torques are calculated given the positions, velocities, accelerations and external forces. The recursive algorithms for the inverse dynamics problem are derived in this chapter followed by a factorization of the equations of motion.

In the forward dynamics problem, the joint accelerations are determined given the positions, velocities, torques, and external forces. Recursive algorithms are derived in this dissertation using Lie group notation and the *crucial* concepts of articulated body (AB) inertias and bias forces introduced in (Featherstone, 1983). The AB inertia and bias force concepts are introduced followed by a derivation of the recursive algorithms. The development in this chapter is influenced by the work in (Featherstone, 1983), (Jain, 1991), (Lilly, 1993), and (Mirtich, 1996). Effectively, the recursive forward dynamics algorithms solve the linear system involving the mass matrix in linear-time as opposed to the cubic-time required for solving linear systems with LU decomposition.

The chapter first presents the notation used for tree-structured systems and then discusses kinematics for multibody systems. The inverse dynamics problem for multibody systems is derived followed by a derivation of the forward dynamics.

3.1 Tree-Structured Systems

This chapter is concerned with tree-structured multibody systems. These multibody systems have a single rigid body called the root. The root is attached to the inertial frame through a joint. Each rigid body in the multibody structure has a unique, non-overlapping path from itself to the root rigid body. A graph can be created for a tree-structured multibody system by assigning nodes to each rigid body in the structure and by assigning edges to the joints connecting the rigid bodies. The inertial frame is assigned link number 0. The root node is labeled with the number 1, and the remaining rigid bodies are labeled in a depth-first manner. The joints are labeled so that the joint index is the same as the unique rigid body outboard to the joint. An example of a tree-structured multibody system and the corresponding graph is shown Figure 3.1. The rigid bodies are labeled in bold and the joints are labeled in italics.

The recursive algorithms in this chapter use a joint-centric approach as is done in (Mirtich, 1996) instead of a link-centric approach. The algorithms are iterated over the joint index, and the joint-centric approach provides an elegant method to bookkeep the link to link calculations. Outboard recursion or iteration proceeds from the root to the leaves and the joint index increases. Inboard recursion or iteration proceeds from the leaves to the root and the joint index decreases. The recursive algorithms shown below use the expression, $k.o$, to refer to the link index for the rigid body outboard to joint k, and the expression, $k.i$,

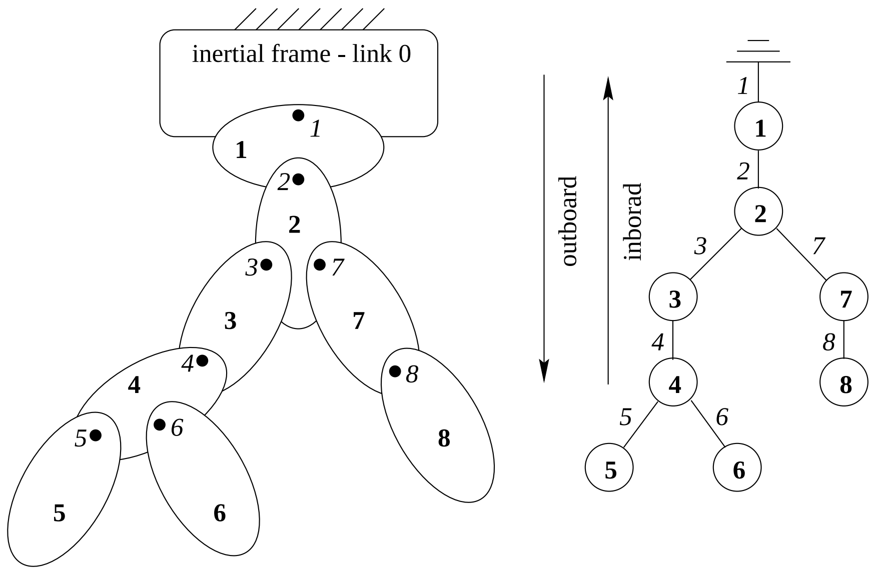

Figure 3.1: Tree-Structured Multibody Example

to refer to the link index of the rigid body inboard to joint k. The maximum joint index is denoted n.

3.2 Multibody Kinematics

The forward kinematic algorithms are developed in this section for tree-structured multibody systems with general joints (*e.g.*, spherical, revolute, prismatic, floating, etc.). The section first presents the derivation of the forward kinematics. The forward kinematic calculations provide the pose of each link in the multibody structure with respect to an inertial frame, and the calculations are performed recursively. Recursive calculations for velocities and accelerations are then presented followed by a presentation of joint kinematics for general, revolute, prismatic, and floating joints. The complete forward kinematics algorithm is then summarized.

3.2.1 Forward Kinematics

The recursive relationship for calculating the pose of each rigid body in the multi-body system with respect to an inertial frame is presented followed by a derivation of the body velocity and the acceleration.

Each rigid body has a frame attached to the center of mass and aligned with the principal axes of inertia. The pose of a rigid body, l, with respect to the inertial frame is denoted, $g_{0,l} \in SE(3)$, and maps points and vectors in the body frame to points and vectors in the inertial frame. For a joint k, the pose of the inboard body is $g_{0,k.i}$. The outboard pose, $g_{0,k.o}$, is calculated by knowing the pose of the inboard body and the pose between the outboard and the inboard link. The outboard pose based on the pose of the inboard link is given by

$$\boxed{g_{0,k.o} = g_{0,k.i}\; g_{k.i,k.o}.} \tag{3.1}$$

Equation (3.1) is then iterated in an outboard recursion to calculate the pose of each rigid body. The algorithm is initialized with $g_{0,0} = I_{4\times4}$, where $I_{4\times4}$ is the 4×4 identity matrix.

The body velocity of the outboard link with respect to the inertial frame is a 4×4

twist matrix given by

$$\begin{aligned}\hat{V}^b_{0,k.o} &= g^{-1}_{0,k.o}\dot{g}_{0,k.o}\\ &= (g_{0,k.i}g_{k.i,k.o})^{-1}\frac{d}{dt}(g_{0,k.i}g_{k.i,k.o})\\ &= g^{-1}_{k.i,k.o}g^{-1}_{0,k.i}(\dot{g}_{0,k.i}g_{k.i,k.o} + g_{0,k.i}\dot{g}_{k.i,k.o})\\ &= g^{-1}_{k.i,k.o}(g^{-1}_{0,k.i}\dot{g}_{0,k.i})g_{k.i,k.o} + g^{-1}_{k.i,k.o}\dot{g}_{k.i,k.o}\\ &= g_{k.o,k.i}\hat{V}^b_{0,k.i}g^{-1}_{k.o,k.i} + \hat{V}^b_{k.i,k.o}. \qquad (3.2)\end{aligned}$$

Writing Equation (3.2) in the form of a 6×1 twist gives

$$V^b_{0,k.o} = \mathrm{Ad}_{g_{k.o,k.i}}V^b_{0,k.i} + V^b_{k.i,k.o}. \qquad (3.3)$$

Let $A^l_k \triangleq \mathrm{Ad}_{g_{l,k}}$ and $V^b_k \triangleq V^b_{0,k}$ to simplify notation. Equation (3.3) is then rewritten to give that

$$V^b_{k.o} = \overbrace{A^{k.o}_{k.i}V^b_{k.i}}^{\text{transformed inboard velocity}} + \underbrace{V^b_{k.i,k.o}.}_{\text{relative velocity}} \qquad (3.4)$$

Equation (3.4) gives a simple expression to recursively compute the body velocity of the outboard link based on the body velocity of the inboard link and the relative velocity of the two bodies. The notation correctly takes care of coordinate changes and succinctly presents the velocity relationships.

The accelerations used in the dynamics calculations are formed by taking the time derivative of Equation (3.4) to get

$$\dot{V}^b_{k.o} = A^{k.o}_{k.i}\dot{V}^b_{k.i} + \dot{V}^b_{k.i,k.o} + \dot{A}^{k.o}_{k.i}V^b_{k.i}. \qquad (3.5)$$

3.2.2 Joint Kinematics

The kinematic properties for joints are derived in this section. The notation used for general joints is presented followed by specific relationships for revolute, prismatic, and floating joints.

General Joints

For a general joint k, there are configuration variables, θ_k, and velocity variables, β_k. The time derivative of the configuration variables are related to the velocity variables by the following equation:

$$\dot{\theta}_k = B_k(\theta_k)\beta_k, \tag{3.6}$$

where B_k is a configuration dependent matrix. The number of degrees of freedom of joint k is denoted d_k, and $0 \leq d_k \leq 6$. The dimension of β_k is then d_k. The dimension of θ_k is e_k, and $d_k \leq e_k \leq 6$. For example, if joint k is a spherical joint and unit quaternions are used to represent the configuration of the joint, then $d_k = 3$, $e_k = 4$, and $B(\theta_k)$ is a 4×3 matrix.

The relative pose between the inboard link, $k.i$, and the outboard link, $k.o$, is a function of the joint configuration variables and is given by

$$g_{k.i,k.o} = g_{k.i,k.o}(\theta_k). \tag{3.7}$$

The relative body velocity between the inboard link, $k.i$, and the outboard link, $k.o$, is related through the joint map, $H_k(\theta_k) \in \mathbb{R}^{6\times d_k}$, through the following equation:

$$V^b_{k.i,k.o} = H_k(\theta_k)\beta_k, \tag{3.8}$$

where H_k is a full rank matrix.

The body velocities, $V^b_{k.o}$, with respect to the inertial frame, link 0, are then

$$\boxed{V^b_{k.o} = A^{k.o}_{k.i} V^b_{k.i} + H_k \beta_k,} \tag{3.9}$$

and the accelerations are

$$\boxed{\dot{V}^b_{k.o} = A^{k.o}_{k.i} \dot{V}^b_{k.i} + H_k \dot{\beta}_k + a_k,} \tag{3.10}$$

where

$$\boxed{a_k = \dot{A}^{k.o}_{k.i} V^b_{k.i} + \dot{H}_k \beta_k.} \tag{3.11}$$

Revolute and Prismatic Joints

For revolute and prismatic joints, $\dot{\theta}_k = \beta_k$, and

$$g_{k.i,k.o} = e^{\widehat{\xi}_{k.i,k.o}\theta_k} g_{k.i,k.o}(0), \tag{3.12}$$

where $\xi_{k.i,k.o}$ is the twist for joint k written with respect to frame $k.i$.

The relative velocity for a revolute or prismatic joint is

$$\begin{aligned}
\widehat{V}^b_{k.i,k.o} &= g^{-1}_{k.i,k.o} \dot{g}_{k.i,k.o} \\
&= g^{-1}_{k.i,k.o}(0) e^{-\widehat{\xi}_{k.i,k.o}\theta_k} e^{\widehat{\xi}_{k.i,k.o}\theta_k} \widehat{\xi}_{k.i,k.o} \dot{\theta}_k g_{k.i,k.o}(0) \\
&= g^{-1}_{k.i,k.o}(0) \widehat{\xi}_{k.i,k.o} \dot{\theta}_k g_{k.i,k.o}(0) \\
&= g_{k.o,k.i}(0) \widehat{\xi}_{k.i,k.o} \dot{\theta}_k g^{-1}_{k.o,k.i}(0).
\end{aligned} \tag{3.13}$$

Therefore,

$$\begin{aligned} V^b_{k.i,k.o} &= \mathrm{Ad}_{g_{k.o,k.i}(0)}\xi_{k.i,k.o}\dot{\theta}_k \\ &= H_k\beta_k. \end{aligned} \tag{3.14}$$

For revolute and prismatic joints, H_k, is a constant column vector and is the twist for joint k written with respect to frame $k.o$.

Floating Joints

For floating joints, also known as free joints, $d_k = 6$, and θ_k can be formed from a unit quaternion and a position vector. Also, $\beta_k = V^b_{k.i,k.o}$ and $H_k = I_{6\times 6}$.

3.2.3 Forward Kinematics Algorithm

The previous results on forward kinematics are now summarized, and the complete forward kinematics algorithm is presented in Algorithm 3.1. The algorithm is given the positions, velocities, and accelerations of each joint and then calculates the pose, body velocity, and acceleration of each link in the tree-structured multibody system.

3.3 Inverse Multibody Dynamics

The inverse dynamics problem is developed in this section. The inverse dynamics problem calculates the joint torques given the positions, velocities, accelerations, and external forces acting on a tree-structured multibody system. The dynamic equations for rigid body motion are first given followed by a description of how to calculate the joint

Algorithm 3.1 Forward Kinematics for Tree-Structured Multibody Systems

```
given: $\theta_k$, $\beta_k$, and $\dot{\beta}_k$ for all $k \in \{1, ..., n\}$; $g_{0,0} = I_{4\times 4}$; $V_0^b = 0_{6\times 1}$
for $k = 1$ to $n$ do
    $g_{0,k.o} = g_{0,k.i}\, g_{k.i,k.o}(\theta_k)$;
    $A_{k.i}^{k.o} = \mathrm{Ad}_{g_{k.o,k.i}}$;
    $V_{k.o}^b = A_{k.i}^{k.o} V_{k.i}^b + H_k \beta_k$;
    $a_k = \dot{A}_{k.i}^{k.o} V_{k.i}^b + \dot{H}_k \beta_k$;
    $\dot{V}_{k.o}^b = A_{k.i}^{k.o} \dot{V}_{k.i}^b + H_k \dot{\beta}_k + a_k$;
end for
```

torques. The complete inverse dynamics algorithm for tree-structured multibody systems is then presented followed by a description of the factorization of the equations of motion.

3.3.1 Rigid Body Dynamics

Consider a rigid body with index l. Attach a body frame to the center of mass and align the axes with the principal axes of inertia. The equations of motion are

$$F_l^{bt} = M_l^b \dot{V}_l^b + \Omega_l^b M_l^b V_l^b, \tag{3.15}$$

where

$$M_l^b = \begin{bmatrix} m_l I & 0 \\ 0 & \mathbb{I}_l \end{bmatrix}, \qquad \Omega_l^b = \begin{bmatrix} \widehat{\omega}_l^b & 0 \\ 0 & \widehat{\omega}_l^b \end{bmatrix}, \tag{3.16}$$

V_l^b is the body velocity with respect to the inertial frame, F_l^{bt} is the total body wrench acting on the rigid body, m_l is the total mass of the rigid body, I is the 3×3 identity matrix, ω_l^b is the body angular velocity, and $\mathbb{I}_l$ is the diagonal inertia matrix. See page 167 in (Murray et al., 1994) for a derivation of the rigid body equations.

The equations of motion for a single rigid body in the multibody system are now derived. Let $F^b_{k.o}$ denote the body wrench acting on link $k.o$ from link $k.i$ through joint k. The wrench is represented in frame $k.o$. The total external body wrench (*e.g.* gravity) acting at the center of mass of link $k.o$ is denoted $F^{be}_{k.o}$. The dynamic equations for a rigid body in the multibody system are then

$$F^b_{k.o} = \sum_{j \in \mathcal{O}_{k.o}} (A^{j.o}_{j.i})^T F^b_{j.o} + M^b_{k.o} \dot{V}^b_{k.o} + b^b_{k.o}, \tag{3.17}$$

where $\mathcal{O}_{k.o}$ is the set of joints connecting the outboard rigid bodies to link $k.o$, and

$$\boxed{b^b_{k.o} = -F^{be}_{k.o} + \Omega^b_{k.o} M^b_{k.o} V^b_{k.o}.} \tag{3.18}$$

The first summation term on the right in Equation (3.17) is the reaction forces of outboard links acting on link $k.o$.

3.3.2 Joint Torque

The joint torque, τ_k, for joint k is now calculated given the wrench, $F^b_{k.o}$, acting across the joint. The joint torque is calculated by knowing that the work of the joint torque equals the work of the body wrench acting across the joint:

$$\begin{aligned} \beta_k^T \tau_k &= (V^b_{k.i,k.o})^T F^b_{k.o} \\ &= (H_k \beta_k)^T F^b_{k.o} \\ &= \beta_k^T H_k^T F^b_{k.o}. \end{aligned} \tag{3.19}$$

Algorithm 3.2 Inverse Dynamics for Tree-Structured Multibody Systems

```
given: $V^b_{k.o}$, $\dot{V}^b_{k.o}$, and $F^{be}_{k.o}$ for all $k \in \{1, ..., n\}$

for $k = 1$ to $n$ do
    $b^b_{k.o} = -F^{be}_{k.o} + \Omega^b_{k.o} M^b_{k.o} V^b_{k.o}$;
    $F^b_{k.o} = M^b_{k.o} \dot{V}^b_{k.o} + b^b_{k.o}$;
end for

for $k = n$ to 1 do
    $F^b_{k.i} = F^b_{k.i} + (A^{k.o}_{k.i})^T F^b_{k.o}$;
    $\tau_k = H^T_k F^b_{k.o}$;
end for
```

Equation (3.19) is true for all β_k which implies that

$$\boxed{\tau_k = H^T_k F^b_{k.o}.} \tag{3.20}$$

3.3.3 Inverse Dynamics Algorithm

The recursive algorithm for calculating the inverse dynamics is given in Algorithm 3.2. The forward kinematics algorithm provides the necessary data for the inverse dynamics algorithm. The body forces are first initialized for each rigid body in the first for loop. The next for loop calculates the body forces due to the outboard links and also calculates the joint torques through the projection with the joint map.

3.3.4 Factorization of the Equations of Motion

This section provides a factorization of the equations of motion for tree-structured multibody systems in terms of the Lie group notation used in this dissertation. The factorization is based on the factorization for serial chains in (Jain, 1991) and differs from the factorization of tree-structured systems in (Rodriguez et al., 1992).

The block adjoint matrix, $\mathbf{A}$, is composed of block matrices $\mathbf{A}_{kl} \in \mathbb{R}^{6\times 6}$ where

$$\mathbf{A}_{kl} = \begin{cases} A_l^k & \text{if } k.i = l.o \\ 0_{6\times 6} & \text{otherwise.} \end{cases}$$

Recall that $A_l^k = \mathrm{Ad}_{g_{k,l}}$. Column l has a non-zero element in row k if body k is connected to body l and body k is outboard to body l. By the number scheme introduced in the beginning of this chapter, the joint index l equals the body index for body $l.o$. For the example in Figure (3.1), the block adjoint matrix is

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A_1^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & A_2^3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & A_3^4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & A_4^5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & A_4^6 & 0 & 0 & 0 & 0 \\ 0 & A_2^7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & A_7^8 & 0 \end{bmatrix}.$$

The block adjoint matrix is lower triangular and has zeros along the diagonal.

Let $\mathbf{V}^b$ be formed by stacking V_k^b in numerical order. Similarly, form $\mathbf{H}, \beta, \mathbf{b}, \mathbf{a}, \tau,$

and $\mathbf{F}^b$ by stacking H_k, β_k, b_k, a_k, θ_k, and F_k^b, respectively. Let $\mathbf{M}^b$ and $\mathbf{B}$ be block diagonal matrices with M_k^b and B_k along the diagonal, respectively. The dynamic equations in matrix form are then

$$\mathbf{V}^b = \mathbf{A}\mathbf{V}^b + \mathbf{H}\beta \tag{3.21}$$

$$\dot{\mathbf{V}}^b = \mathbf{A}\dot{\mathbf{V}}^b + \mathbf{H}\dot{\beta} + \mathbf{a} \tag{3.22}$$

$$\mathbf{F}^b = \mathbf{A}^T\mathbf{F}^b + \mathbf{M}^b\dot{\mathbf{V}}^b + \mathbf{b} \tag{3.23}$$

$$\tau = \mathbf{H}^T\mathbf{F}^b \tag{3.24}$$

Notice that $I - \mathbf{A}$ is invertible since it has identity matrices along the diagonal and the upper triangle has all zero entries. Define ϕ to be $(I - \mathbf{A})^{-1}$. The inverse is then

$$\phi \overset{\triangle}{=} (I - \mathbf{A})^{-1} = I + \mathbf{A} + \mathbf{A}^2 + \cdots + \mathbf{A}^{n-1}$$

since $A^n = 0$. For the example in Figure (3.1),

$$\phi = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A_1^2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ A_1^3 & A_2^3 & 1 & 0 & 0 & 0 & 0 & 0 \\ A_1^4 & A_2^4 & A_3^4 & 1 & 0 & 0 & 0 & 0 \\ A_1^5 & A_2^5 & A_3^5 & A_4^5 & 1 & 0 & 0 & 0 \\ A_1^6 & A_2^6 & A_3^6 & A_4^6 & 0 & 1 & 0 & 0 \\ A_1^7 & A_2^7 & 0 & 0 & 0 & 0 & 1 & 0 \\ A_1^8 & A_2^8 & 0 & 0 & 0 & 0 & A_7^8 & 1 \end{bmatrix}$$

since $A_l^k A_m^l = A_m^k$.

The dynamic equations can then be rewritten in the following form:

$$\mathbf{V}^b = \phi \mathbf{H} \beta \tag{3.25}$$

$$\dot{\mathbf{V}}^b = \phi \mathbf{H} \dot{\beta} + \phi \mathbf{a} \tag{3.26}$$

$$\mathbf{F}^b = \phi^T \mathbf{M}^b \dot{\mathbf{V}}^b + \phi^T \mathbf{b} \tag{3.27}$$

$$\tau = \mathbf{H}^T \mathbf{F}^b \tag{3.28}$$

which implies that

$$\tau = \mathbf{H}^T \phi^T \mathbf{M}^b \dot{\mathbf{V}}^b + \mathbf{H}^T \phi^T \mathbf{b} \tag{3.29}$$

$$= \mathbf{H}^T \phi^T \mathbf{M}^b \phi \mathbf{H} \dot{\beta} + \mathbf{H}^T \phi^T \mathbf{M}^b \phi \mathbf{a} + \mathbf{H}^T \phi^T \mathbf{b} \tag{3.30}$$

$$= \mathbb{M} \dot{\beta} + \mathbb{C}, \tag{3.31}$$

where

$$\mathbb{M} \stackrel{\triangle}{=} \mathbf{H}^T \phi^T \mathbf{M}^b \phi \mathbf{H} \tag{3.32}$$

and

$$\mathbb{C} \stackrel{\triangle}{=} \mathbf{H}^T \phi^T (\mathbf{M}^b \phi \mathbf{a} + \mathbf{b}). \tag{3.33}$$

The Coriolis, centrifugal, and external forces are contained in $\mathbb{C}$. The mass matrix is $\mathbb{M}$. The matrix $\mathbf{H}$ is formed from the joint maps for each joint. The term $\mathbf{H}\beta$ gives the relative body velocities from the inboard joint to the outboard joint. The matrix ϕ maps relative body velocities to the body velocities of each link. The matrix ϕ^T maps body wrenches into the wrenches across each joint. The matrix $\mathbf{H}^T$ maps body wrenches across the joints to joint torques.

Again, the factorization presented here is based on the derivations for serial chains in (Jain, 1991). The factorization presented in this dissertation is for tree-structured multibody systems and is derived using the Lie group notation used in this dissertation. A similar factorization to the one presented here is given in (Park et al., 1995). The authors use Lie group notation to compute the inverse dynamics and factor the equations of motion. They do not derive the forward dynamics algorithm which is derived here in the next section.

The first order differential equations of motion are

$$\dot{\theta} = \mathbf{B}(\theta)\beta \tag{3.34}$$

$$\dot{\beta} = \mathbb{M}^{-1}(\theta)(\tau - \mathbb{C}(\theta, \dot{\theta})). \tag{3.35}$$

In a typical multibody simulator, the accelerations are calculated through a recursive forward dynamics algorithm. The joint velocities and accelerations are then passed to a standard integrator to flow the system forward in time.

3.4 Forward Multibody Dynamics

Recursive and linear-time algorithms are derived in this section to compute the forward dynamics for tree-structured multibody systems. The forward dynamics algorithm calculates the joint accelerations given the joint torques, external forces, positions, and velocities. The algorithm is based on the concept of articulated body (AB) inertias and bias forces presented in (Featherstone, 1983). The derivation is primarily based on the work in (Featherstone, 1983), (Jain, 1991), (Lilly, 1993), and (Mirtich, 1996). This section presents the recursive algorithms in a consistent and concise manner with the notation used in this dissertation.

3.4.1 Articulated Body Inertias and Bias Forces

The *important* concept in the recursive forward dynamics algorithms is the concept of articulated body (AB) inertias and bias forces. The AB inertias and bias forces provide an affine relationship between the body wrench, $F^b_{k.o}$, acting on the rigid body in the multibody system, the AB inertia denoted $P_{k.o}$, the acceleration of the body denoted $\dot{V}^b_{k.o}$, and the bias forces denoted $z_{k.o}$. A recursive relationship is derived which calculates the bias forces and AB inertias of the inboard link based on the AB inertias and bias forces of the outboard links. The *crucial* relationship is

$$\boxed{F^b_{k.o} = P_{k.o}\dot{V}^b_{k.o} + z_{k.o},} \tag{3.36}$$

where $P_{k.o} = P^T_{k.o} > 0$, $P_{k.o}$ is only a function of positions, and $z_{k.o}$ is only a function of positions, velocities, and external forces. It is shown in this section that if the relation in Equation (3.36) holds for the outboard links, then the relation holds for the inboard link and there exists a recursive method to calculate the AB inertias and bias forces.

3.4.2 Derivation of the Recursive Forward Dynamics Algorithm

First note that the relation is true for each leaf in the tree-structure where $P_{k.o} = P^T_{k.o} = M^b_{k.o}$ and $z_{k.o} = b^b_{k.o}$. For an arbitrary body in the tree-structured multibody system,

$$F^b_{k.o} = \sum_{j \in \mathcal{O}_{k.o}} (A^{j.o}_{j.i})^T F^b_{j.o} + M^b_{k.o}\dot{V}^b_{k.o} + b^b_{k.o}, \tag{3.37}$$

where $\mathcal{O}_{k.o}$ is the set of joints connecting the outboard rigid bodies to link $k.o$. Apply the induction hypothesis to get that

$$F^b_{k.o} = \sum_{j\in\mathcal{O}_{k.o}} (A^{j.o}_{j.i})^T(P_{j.o}\dot{V}^b_{j.o} + z_{j.o}) + M^b_{k.o}\dot{V}^b_{k.o} + b^b_{k.o} \tag{3.38}$$

$$= \sum_{j\in\mathcal{O}_{k.o}} (A^{j.o}_{j.i})^T(P_{j.o}(A^{j.o}_{j.i}\dot{V}^b_{j.i} + H_j\dot{\beta}_j + a_j) + z_{j.o}) + M^b_{k.o}\dot{V}^b_{k.o} + b^b_{k.o}. \tag{3.39}$$

Notice that $j.i = k.o$ for $j \in \mathcal{O}_{k.o}$. Therefore,

$$\begin{aligned} F^b_{k.o} = &\left[\sum_{j\in\mathcal{O}_{k.o}} (A^{j.o}_{k.o})^T P_{j.o} A^{j.o}_{k.o} + M^b_{k.o}\right]\dot{V}^b_{k.o} \\ &+ \sum_{j\in\mathcal{O}_{k.o}} (A^{j.o}_{k.o})^T P_{j.o} H_j\dot{\beta}_j + \sum_{j\in\mathcal{O}_{k.o}} (A^{j.o}_{k.o})^T(P_{j.o}a_j + z_{j.o}) + b^b_{k.o}. \end{aligned} \tag{3.40}$$

The variable $\dot{\beta}_k$ needs to be eliminated to get Equation (3.40) in the form in Equation (3.36). Equation (3.20) implies that

$$\tau_j = H_j^T F^b_{j.o} = H_j^T(P_{j.o}\dot{V}^b_{j.o} + z_{j.o}) \tag{3.41}$$

$$= H_j^T(P_{j.o}(A^{j.o}_{j.i}\dot{V}^b_{j.i} + H_j\dot{\beta}_j + a_j) + z_{j.o}) \tag{3.42}$$

$$= H_j^T P_{j.o} H_j\dot{\beta}_j + H_j^T P_{j.o} A^{j.o}_{j.i}\dot{V}^b_{j.i} + H_j^T(P_{j.o}a_j + z_{j.o}). \tag{3.43}$$

Define $D_{j.o}$ to be $H_j^T P_{j.o} H_j$ and note again that $j.i = k.o$ for $j \in \mathcal{O}_{k.o}$. Also, $D_{j.o}$ is invertible since $P_{j.o}$ is assumed to be positive definite and H_j has full rank. Therefore,

$$\dot{\beta}_j = D^{-1}_{j.o}\left[\tau_j - H_j^T P_{j.o} A^{j.o}_{k.o}\dot{V}^b_{k.o} - H_j^T(P_{j.o}a_j + z_{j.o})\right]. \tag{3.44}$$

First examine the term $\sum_{j\in\mathcal{O}_{k.o}}(A^{j.o}_{k.o})^T P_{j.o}H_j\dot{\beta}_j$ and then substitute the results back into Equation (3.40).

$$\begin{aligned}
&\sum_{j\in\mathcal{O}_{k.o}}(A^{j.o}_{k.o})^T P_{j.o}H_j\dot{\beta}_j = \\
&\sum_{j\in\mathcal{O}_{k.o}}(A^{j.o}_{k.o})^T P_{j.o}H_j D^{-1}_{j.o}\left[\tau_j - H_j^T P_{j.o}A^{j.o}_{k.o}\dot{V}^b_{k.o} - H_j^T(P_{j.o}a_j + z_{j.o})\right] \\
&= \left[-\sum_{j\in\mathcal{O}_{k.o}}(A^{j.o}_{k.o})^T P_{j.o}K_{j.o}P_{j.o}A^{j.o}_{k.o}\right]\dot{V}^b_{k.o} \\
&- \sum_{j\in\mathcal{O}_{k.o}}(A^{j.o}_{k.o})^T P_{j.o}K_{j.o}(P_{j.o}a_j + z_{j.o}) + \sum_{j\in\mathcal{O}_{k.o}}(A^{j.o}_{k.o})^T G_{j.o}\tau_j,
\end{aligned} \tag{3.45}$$

where

$$K_{j.o} \triangleq H_j D^{-1}_{j.o}H_j^T \tag{3.46}$$

and

$$G_{j.o} \triangleq P_{j.o}H_j D^{-1}_{j.o}. \tag{3.47}$$

Substitute Equation (3.45) into Equation (3.40) to get that

$$\begin{aligned}
F^b_{k.o} &= \left[\sum_{j\in\mathcal{O}_{k.o}}(A^{j.o}_{k.o})^T P_{j.o}(I - K_{j.o}P_{j.o})A^{j.o}_{k.o} + M^b_{k.o}\right]\dot{V}^b_{k.o} \\
&+ \sum_{j\in\mathcal{O}_{k.o}}(A^{j.o}_{k.o})^T(I - P_{j.o}K_{j.o})(P_{j.o}a_j + z_{j.o}) + \sum_{j\in\mathcal{O}_{k.o}}(A^{j.o}_{k.o})^T G_{j.o}\tau_j + b^b_{k.o}.
\end{aligned} \tag{3.48}$$

Let

$$L^{j.o}_{k.o} \triangleq \left(I - K_{j.o}P_{j.o}\right)A^{j.o}_{k.o}. \tag{3.49}$$

Equation (3.48) then simplifies to

$$F^b_{k.o} = \left[\sum_{j \in \mathcal{O}_{k.o}} (L^{j.o}_{k.o})^T P_{j.o} A^{j.o}_{k.o} + M^b_{k.o} \right] \dot{V}^b_{k.o}$$

$$+ \sum_{j \in \mathcal{O}_{k.o}} (L^{j.o}_{k.o})^T (P_{j.o} a_j + z_{j.o}) + \sum_{j \in \mathcal{O}_{k.o}} (A^{j.o}_{k.o})^T G_{j.o} \tau_j + b^b_{k.o}. \tag{3.50}$$

Therefore,

$$P_{k.o} = M^b_{k.o} + \sum_{j \in \mathcal{O}_{k.o}} (L^{j.o}_{k.o})^T P_{j.o} A^{j.o}_{k.o} \tag{3.51}$$

and

$$\boxed{z_{k.o} = \sum_{j \in \mathcal{O}_{k.o}} (L^{j.o}_{k.o})^T (P_{j.o} a_j + z_{j.o}) + \sum_{j \in \mathcal{O}_{k.o}} (A^{j.o}_{k.o})^T G_{j.o} \tau_j + b^b_{k.o}.} \tag{3.52}$$

The bias forces in Equation (3.52) are only a function of positions, velocities, and external forces. It must now be shown that $P_{k.o}$ is positive definite and symmetric. This is shown by first deriving that

$$(L^{j.o}_{k.o})^T P_{j.o} A^{j.o}_{k.o} = (L^{j.o}_{k.o})^T P_{j.o} L^{j.o}_{k.o}. \tag{3.53}$$

First expand the left-hand side of Equation (3.53) to get that

$$\begin{aligned} (L^{j.o}_{k.o})^T P_{j.o} A^{j.o}_{k.o} &= (A^{j.o}_{k.o})^T \left[I - P_{j.o} K_{j.o} \right] P_{j.o} A^{j.o}_{k.o} \\ &= (A^{j.o}_{k.o})^T \left[P_{j.o} - P_{j.o} K_{j.o} P_{j.o} \right] A^{j.o}_{k.o} \end{aligned} \tag{3.54}$$

Expand the right-hand side of Equation (3.53) to get that

$$\begin{aligned} (L^{j.o}_{k.o})^T P_{j.o} L^{j.o}_{k.o} &= (A^{j.o}_{k.o})^T \left[I - P_{j.o} K_{j.o} \right] P_{j.o} \left[I - K_{j.o} P_{j.o} \right] A^{j.o}_{k.o} \\ &= (A^{j.o}_{k.o})^T \left[P_{j.o} - 2 P_{j.o} K_{j.o} P_{j.o} + P_{j.o} K_{j.o} P_{j.o} K_{j.o} P_{j.o} \right] A^{j.o}_{k.o}. \end{aligned} \tag{3.55}$$

Note that

$$K_{j.o}P_{j.o}K_{j.o} = H_j D_{j.o}^{-1} H_j^T P_{j.o} H_j D_{j.o}^{-1} H_j^T$$
$$= H_j D_{j.o}^{-1} D_{j.o} D_{j.o}^{-1} H_j^T = H_j D_{j.o}^{-1} H_j^T$$
$$= K_{j.o}. \tag{3.56}$$

Simplifying Equation (3.55) with Equation (3.56) reveals that

$$(L_{k.o}^{j.o})^T P_{j.o} L_{k.o}^{j.o} = (A_{k.o}^{j.o})^T \left[P_{j.o} - P_{j.o}K_{j.o}P_{j.o}\right] A_{k.o}^{j.o}$$
$$= (L_{k.o}^{j.o})^T P_{j.o} A_{k.o}^{j.o}. \tag{3.57}$$

Equation (3.57) implies that

$$\boxed{P_{k.o} = M_{k.o}^b + \sum_{j \in \mathcal{O}_{k.o}} (L_{k.o}^{j.o})^T P_{j.o} L_{k.o}^{j.o}.} \tag{3.58}$$

Equation (3.58) shows that $P_{k.o}$ is a sum of a positive definite symmetric matrix and semi-definite symmetric matrices, and this implies that $P_{k.o} = (P_{k.o})^T > 0$.

The AB inertias and bias forces are calculated in an inboard recursion. The accelerations are then calculated in an outboard recursion by using Equation (3.44) rewritten in Equation (3.59) followed by Equation (3.10) rewritten in Equation (3.60) where

$$\dot{\beta}_k = D_{k.o}^{-1}(\tau_k - H_k^T z_{k.o}) - G_{k.o}^T (A_{k.i}^{k.o} \dot{V}_{k.i}^b + a_k) \tag{3.59}$$

and

$$\dot{V}_{k.o}^b = A_{k.i}^{k.o} \dot{V}_{k.i}^b + H_k \dot{\beta}_k + a_k. \tag{3.60}$$

Algorithm 3.3 Forward Dynamics for Tree-Structured Multibody Systems

```
given: $a_k$, $b^b_k$, $A^{k.o}_{k.i}$ for all $k \in \{1, ..., n\}$
for $k = 1$ to $n$ do
    $z_{k.o} = b^b_{k.o}$;    $P_{k.o} = M^b_{k.o}$;
end for
for $k = n$ to $1$ do
    $D_{k.o} = H^T_{k.o} P_{k.o} H_{k.o}$;    $K_{k.o} = H_{k.o} D^{-1}_{k.o} H^T_{k.o}$;    $G_{k.o} = P_{k.o} H_k D^{-1}_{k.o}$;
    $L^{k.o}_{k.i} = [I - K_{k.o} P_{k.o}] A^{k.o}_{k.i}$;
    $P_{k.i} = P_{k.i} + (L^{k.o}_{k.i})^T P_{k.o} L^{k.o}_{k.i}$;
    $z_{k.i} = z_{k.i} + (L^{k.o}_{k.i})^T (P_{k.o} a_k + z_{k.o}) + (A^{k.o}_{k.i})^T G_{k.o} \tau_k$;
end for
for $k = 1$ to $n$ do
    $\dot{\beta}_k = D^{-1}_{k.o}(\tau_k - H^T_k z_{k.o}) - G^T_{k.o}(A^{k.o}_{k.i} \dot{V}^b_{k.i} + a_k)$;
    $\dot{V}^b_{k.o} = A^{k.o}_{k.i} \dot{V}^b_{k.i} + H_k \dot{\beta}_k + a_k$;
end for
```

3.4.3 Recursive Dynamics Algorithm

The full forward dynamics algorithm is now given in Algorithm 3.3. The bias forces and AB inertias are first initialized to the body forces and the body mass matrix. The recursive operators are then formed and the AB inertias and bias forces are updated in an inboard recursion. The joint accelerations are then calculated in an outboard recursion.

3.5 Chapter Summary

Recursive techniques for the forward kinematics, inverse dynamics, and forward dynamics of tree-structured multibody systems with general joints were derived in this chapter. Lie group matrix notation presented in (Murray et al., 1994) was used and this simplified the derivation and correctly provided the coordinate changes necessary to implement the algorithms. A factorization of the equations of motion was provided based on the work in (Jain, 1991). Consult (Jain, 1991) for additional factorizations, operators, and interpretations of the recursive algorithms. The development in this chapter is influenced by the work in (Featherstone, 1983), (Jain, 1991), (Lilly, 1993), and (Mirtich, 1996) and consult these references for different derivations of recursive multibody techniques. Consult (Bae and Haug, 1988) and (Rodriguez et al., 1992) for ideas on extending the tree-structured results to systems with closed loops.

Chapter 4

Recursive Workspace Control

The first stages of a research effort to create general control methods for multibody systems is presented in this chapter. A specific multibody system, a 3D model of human biped, is examined in this effort, and the goal is to create predictive, dynamic models for human locomotion. The goal of the predictive models is to create models of human bipeds that appropriately respond to outside disturbances and that also allow one to perform computer experiments that normally need to be performed with human subjects. Towards this end, a balancing controller for a 3D model of a human biped is created and described in this chapter. The balancing controller is designed to serve as a basis for more sophisticated controllers for walking, running, jumping, changing direction, and adapting to loads.

General control techniques for multibody systems have many applications. General control techniques will allow engineers to easily create behaviors in complicated robotic systems for space, underwater, and hazardous environment applications. Predictive models of human locomotion have uses in human equipment design, sports equipment design, human training, human injury reduction, and animation. A predictive model is useful for researchers and designers to allow them to easily perform "what-if" experiments through computer simulation.

Creating predictive controllers for human motion is difficult to achieve. The differential equations of motion are complicated, lengthy, coupled, and nonlinear. The multibody models of humans possess a large number of degrees of freedom for a robotic system, and the controller needs to coordinate these degrees of freedom. The dynamics of the system significantly change as the system intermittently contacts the environment. To be truly predictive, the controller must not rely on specific experimental data such as joint trajectories. A major difficulty is that the underlining control system for human locomotion is not known or well understood.

There are many references on human balancing, biped locomotion, and human modeling in the areas of robotics, biology, and animation. A small sampling of the research performed in this area is given here. Several robotics researchers have created locomoting robots with one to many legs. The researchers in (Furusho and Sano, 1990) have created a nine-link biped robot that walks by using linear control techniques, dividing the walk into phases, and developing controllers for each phase. The volume 9, Number 2, April 1990 edition of the International Journal of Robotics Research is a special issue on legged robots and contains several useful references. In the book of (Vukobratović et al., 1990), the authors present a study of biped locomotion, analyze biped locomotion, and list some of the existing hardware implementations. In (Goddard et al., 1992), the authors analyze the heel-off to toe-off motion in locomotion and also provide a good literature review of biped locomotion research. The authors in (Stewart and Cremer, 1989) develop control algorithms for a walking robot and simulate the system in Newton (Cremer and Stewart, 1989), a general-purpose multibody simulator. Hopping robots were created in (Raibert, 1986) by first creating a hopping robot with one leg and then extending the ideas to robots with two and then four legs. State machines are used to coordinate low-level PD controllers to make the system locomote without falling over. Algorithms for foot placement, balancing, and speed control are also provided in (Raibert, 1986). Gymnastics were also performed with the hopping robot, and this work is documented in (Hodgins and Raibert, 1990). The author in (Argáez, 1993) designed and created an experimental climbing robot and introduced the notion of control points to simultaneously control force and position.

There are also numerous studies by researchers in biology on human locomotion and only a few of the many references are presented here. The book of (McMahon, 1984) has numerous experimental results, force data, and joint trajectory data of human subjects. Descriptions of muscles are also provided. In (Winter, 1989), the author studies human gait and provides numerous experimental results on joint trajectories, force plate data, and joint torques. In (Brooks, 1986), the author discusses motor control, posture, and locomotion. Also, consult (Winters and Savio, 1990) for useful information on muscle systems.

There has been recent interest in using dynamic simulators and control algorithms to produce realistic animations for computer graphics applications. The authors in (Raibert and Hodgins, 1991) produce animations of legged systems, including models of kangaroos, by simulating controlled multibody models in a dynamic simulator. The technique uses low-level PD controllers coordinated by state machines. The authors in (Hodgins et al.,

1995) produce animations of human athletics including bicycling, running, and vaulting. The technique uses joint trajectory data, PD controllers, and state machines to produce the behaviors. Three-dimensional running is documented in (Hodgins, 1996), and the motion is produced using PD controllers, specifying joint trajectories, and coordinating the PD controllers with state machines. Recently, the authors in (Kokkevis et al., 1996) have used model reference adaptive controllers (MRAC) to produce animations of human models in a dynamic multibody simulator.

The fundamental limitation of the current control techniques for biped locomotion is that there are no general, *predictive* models of human motion. The current techniques require a knowledge of existing joint trajectories and set points. The controller parameters are difficult to tune and require a lot of time consuming trial and error to produce a particular motion. Even after the motion is created, the system may only work in a small region around the created motion. The current techniques are difficult to apply to complex systems, such as the human body, and it is difficult to prove that the controller produces the desired system behavior. The major limitation is that current control techniques are difficult to apply to the nonlinear, complex systems whose dynamics change as the system intermittenly contacts the environment. Controlling systems with discrete model changes and complex, nonlinear dynamics needs to addressed by new control methods.

Since existing techniques are difficult to apply to create predictive models of human motion, new control designs are necessary. A new balancing controller is designed and presented in this chapter and is designed to serve as a basis for more complicated controllers for walking, running, jumping, changing direction, and adapting to loads. The controller is designed to overcome the limitations in current control techniques. It is designed to have a minimum number of parameters that are easy to tune, to have a large operating region, to coordinate the degrees of freedom, to be able to handle the complex equations of motion, and to provide a "good" interface to higher-level controllers. To meet these goals, the controller is designed to first create errors in the central body and not in the joint space. It is felt that the central body is more important than joint angles, and the desired motion is easier to specify in the central body coordinates. A model is created in the workspace, the space of the central body, to provide a decoupled response between the error coordinates and to provide simpler error dynamics. It is difficult, time-consuming, and prone to error to symbolically create the workspace dynamic equations by hand. Recursive multibody techniques are then used to efficiently create the workspace dynamics model. The recursive techniques can be used for more complicated structures, and the computational cost grows linearly with the number of degrees of freedom for tree-structured multibody systems. The joint torques are also calculated in a recursive manner, and this alleviates the designer from symbolically creating the necessary Jacobian. The controller presented in this paper is a 3D extension of the planar balancing controller presented in (Wendlandt and Sastry, 1996).

The chapter first describes the biped model and then provides an overview of the recursive workspace controller. The workspace controller is then described in more detail followed by simulation results performed in a 3D multibody simulator designed to handle contact with the environment called *Impulse* (Mirtich, 1996). The derivations in this chapter assume a familiarity with the results in Appendix A and Chapter 3.

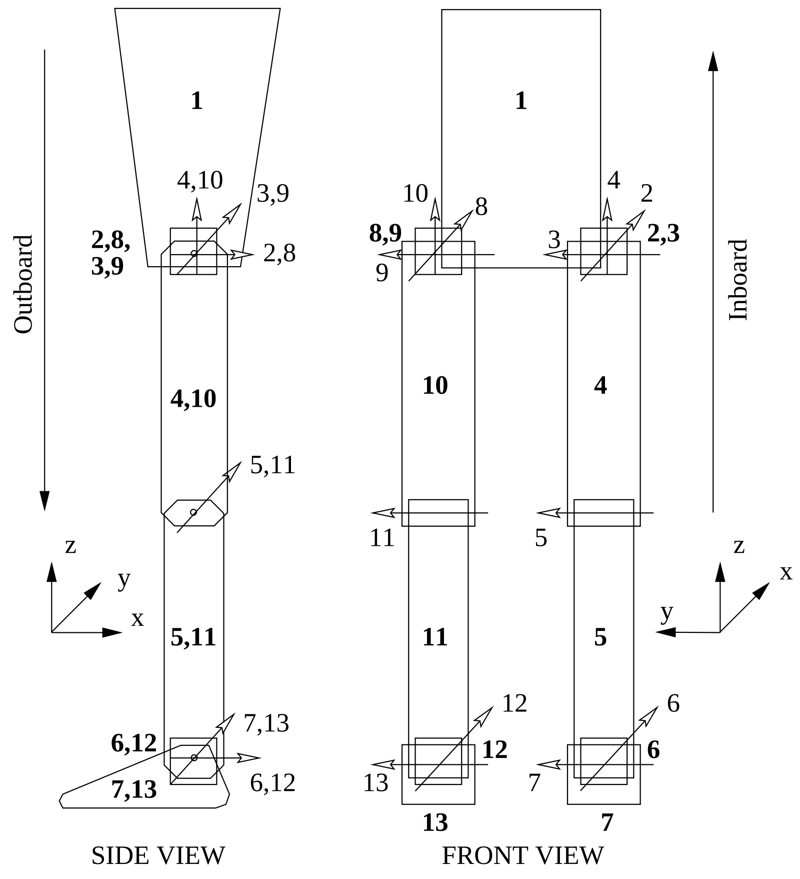

Figure 4.1: Biped Model

4.1 Description of Biped Model

The 3D multibody model of the biped consists of a central body and two legs and is shown in Figure 4.1. The dimensions of the rigid bodies and the placement of the joints are based on measurements of a human subject. Each leg consists of an upper leg, a lower leg, and a foot. The leg is attached to the central body through a 3 degree of freedom (DOF) spherical joint consisting of two intermediate rigid bodies connected with revolute joints. The knee joint has one degree of freedom and connects the upper leg to the lower leg. The foot is attached to the lower leg through a 2 DOF joint consisting of one intermediate body connected with a revolute joint. The total number of rigid bodies in the system is 13, and the total number of DOF is 18. When both feet are in contact with the ground, the contact effectively decreases the total number of DOF to 6. The rigid bodies in the system are labeled in bold text. The revolute joints are shown with arrows pointing in the positive direction and are labeled with plain text. The left leg components are labeled starting at number 2 while the right leg components start at number 8.

Consider an inertial frame placed at the floor having the same orientation as the frame shown in Figure 4.1. The body frames aligned with the principal axes of inertia for each rigid body, except the two feet, have the same orientation in the home configuration as the inertial frame. The orientation of the principal axis frame for the feet is formed from the inertial frame by rotating about the $-Y$ axis by 8.547 degrees. In the home configuration and relative to the inertial frame, the center of mass of link 1 is located at $(0, 0, 123)$ cm. The axes of inertia for link 1 are $(1.67, 1.58, 0.581)$ Kg-m^2, and the mass is 45 Kg. The mass of each rigid body is calculated based on the volume and the density of water. The center of mass of link 2 and 3 is at $(0, -15, 96)$ cm, and the center of mass of link 8 and 9 is at $(0, 15, 96)$ cm. The center of mass of link 6 is at $(0, -15, 6)$ cm, and the center of mass of link 12 is at $(0, 15, 6)$ cm. The inertias about each axis for link 2, 3, 6, 8, 9, and 12 are equal and are 0.00028 Kg-m^2, and the mass for each of these bodies is 0.343 Kg. The center of mass of link 4 is at $(0, -15, 73)$ cm, and the center of mass of link 10 is at $(0, 15, 73)$ cm. Link 4 and 10 have a mass of 7.5 Kg, and the inertias are $(0.163, 0.171, 0.0198)$ Kg-m^2. The center of mass of link 5 is at $(0, -15, 28)$ cm, and the center of mass of link 11 is at $(0, 15, 28)$ cm. The mass of link 5 and 11 is 4.79 Kg, and the inertias are $(0.0961, 0.0959, 0.00784)$

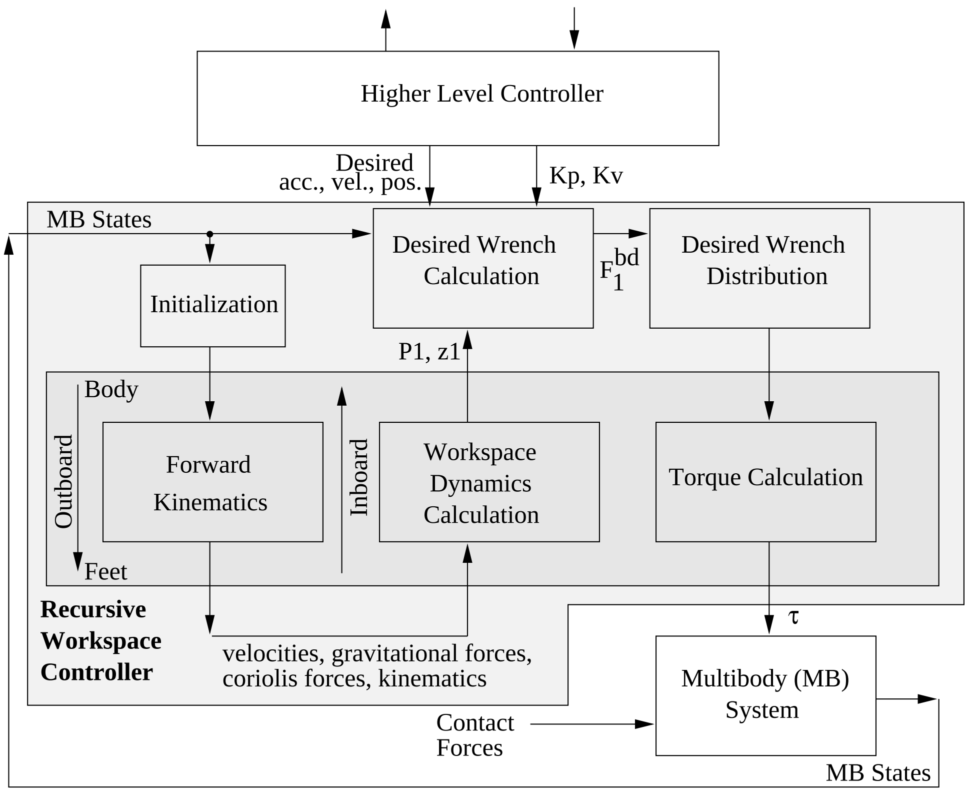

Figure 4.2: Controller Block Diagram

Kg-m^2. The center of mass of link 7 is at $(-5.325, -15, 3.473)$ cm, and the center of mass of link 13 is at $(-5.325, 15, 3.473)$ cm. The mass of link 7 and 13 is 1.455 Kg, and the inertias are $(0.00189, 0.00608, 0.00662)$ Kg-m^2. Joints 2, 3 and 4 are located at $(0, -15, 96)$ cm, and joints 8, 9, and 10 are located at $(0, 15, 96)$ cm. Joint 5 is located at $(0, -15, 50)$ cm, and joint 11 is located at $(0, 15, 50)$ cm. Joints 6 and 7 are located at $(0, -15, 6)$ cm, and joint 12 and 13 are located at $(0, 15, 6)$ cm. The total mass of the system is 74.558 Kg (approximately 164.4 lbs.).

4.2 Recursive Workspace Controller

The block diagram for the recursive workspace controller is shown in Figure 4.2. The terms recursive and iterative refer to successive link to link calculations. The calculations performed for the recursive workspace controller are contained in the outer shaded box. The inboard and outboard recursive steps are indicated in the interior, darker box. The inputs to the controller are the multibody (MB) states (central body position and orientation, center of mass velocity, central body angular velocity, joint angles, and joint velocities); desired central body acceleration, velocity, and position; and position and velocity gains. The desired values and gains are provided by a higher level controller. The controller outputs joint torques.

The basic idea of the control design is to form a model of the dynamics in terms of the workspace, the coordinates of the central body. The workspace dynamics model is created efficiently and easily through multibody recursive techniques. A desired force acting at the central body is then created based on the workspace model, the desired trajectory, and the controller gains. The desired force is then realized through the joint torques in the two legs.

The MB states are first used to initialize controller variables and then used in the forward kinematics stage. In an outboard recursion, the body velocities and pose of each rigid body in the multibody system are calculated. Forces and torques due to Coriolis and gravitational forces are also calculated.

The information calculated in the forward kinematics block is used to calculate the workspace dynamics that consist of the workspace inertia, gravitational forces, and Coriolis forces. The workspace dynamics calculation consists of an inboard recursion from the feet to the central body and produces the approximate workspace dynamics for the central body. This stage calculates the articulated body(AB) inertia (Featherstone, 1983) and bias forces of the inboard link based on the AB inertia and bias forces of the outboard link. The AB inertia and bias forces relate an external force applied to a link to the acceleration of that link and take into account the links outboard to the link. The AB inertia and bias force for the central body provide the approximate workspace dynamics. AB inertias and bias forces are discussed in many references including (Featherstone, 1983), (Lilly, 1993), (Jain, 1991), (Mirtich, 1996), and in Chapter 3 of this dissertation. The recursive workspace controller assumes that the feet are fixed to the floor. We approximate this condition from a *big base* assumption presented in (Lilly, 1993). The approximation assigns the feet a large mass and inertia (approximating the Earth) and sets the feet bias forces to zero. Exact algorithms for calculating the workspace dynamics recursively for serial chains is provided in (Kreutz-Delgado et al., 1992).

The desired wrench acting at the center of mass of the central body is calculated based on the approximate workspace dynamics. The desired trajectory of the central body, the workspace dynamics, the current MB states, trajectory errors, and control gains are combined in this stage to create a desired wrench acting at the center of mass of the central body. The wrench is written with respect to the frame of link 1. The desired wrench cannot be realized directly but is realized through the actuator torques.

The desired wrench is provided through the joint torques of the left and right legs. The desired contribution from each leg is calculated based on the current pose of each foot.

Algorithm 4.1 Controller Initialization

```
**given:** $V_1^b$, $g_{0,1}$;
$a_1^b = [0\ 0\ 0\ 0\ 0\ 0]^T$;
$b_1^b = -F_1^{be} + \Omega_1^b M_1^b V_1^b$;
```

The desired wrench distribution is calculated to minimize the reaction wrenches of the feet with the floor and is calculated with a pseudo-inverse to be described below.

Finally, the joint torques are calculated in an outboard recursion along each leg based on the desired wrench contribution from the left and right legs. The joint torques are created from a Jacobian transpose calculation which is realized in a recursive manner.

4.2.1 Initialization

During the initialization stage, the body velocity and pose of the central body are used to initialize variables for the central body. The calculations during the initialization stage are given in Algorithm 4.1. a_1^b is a kinematic acceleration term and is zero for the free joint between link 1 and the inertial frame. b_1^b consists of gravitational and Coriolis terms for link 1. The term, F_i^{be}, is a 6-vector of external forces acting at the origin of the body frame of link i; V_i^b is the 6-vector body velocity of link i with respect to link 0, the inertial frame;

$$M_i^b = \begin{bmatrix} m_i I & 0 \\ 0 & \mathbb{I}_i \end{bmatrix}; \ \Omega_i^b = \begin{bmatrix} \hat{\omega}_i^b & 0 \\ 0 & \hat{\omega}_i^b \end{bmatrix}; \tag{4.1}$$

m_i is the mass of link i; I is the 3x3 identity matrix; and $\mathbb{I}_i$ is the diagonal inertia matrix of link i. The external forces, F_i^{be}, in this system are the gravitational forces transformed to the body frame of link i.

Algorithm 4.2 Forward Kinematics

```
for $k = 2$ to $13$ do
    $g_{k.o,0} = g_{k.o,k.i}(\theta_k) g_{k.i,0}$
    $A_{k.i}^{k.o} = \mathrm{Ad}_{g_{k.o,k.i}(\theta_k)}$
    $V_{k.o}^b = A_{k.i}^{k.o} V_{k.i}^b + H_k \dot{\theta}_k$
    $a_k^b = \dot{A}_{k.i}^{k.o} V_{k.i}^b + \dot{H}_k \dot{\theta}_k$
    $b_{k.o}^b = -F_{k.o}^{be} + \Omega_{k.o}^b M_{k.o}^b V_{k.o}^b$
end for
```

4.2.2 Forward Kinematics

During the forward kinematics stage, the body velocities, gravitational forces, rigid body poses, and Coriolis forces are calculated in an outboard recursion as shown in Algorithm 4.2. The map $g_{j,k} \in SE(3)$ takes coordinates of a point or vector in link k and gives the corresponding coordinates in link j. The symbol, $k.o$, gives the link index of the link outboard to joint k. The symbol, $k.i$, gives the link index of the link inboard to joint k. The adjoint appropriately transforms velocities and forces in different frames. The joint map is given by H_k and is a vector that represents the twist of joint k written in the link frame outboard to joint k, the frame of link $k.o$. The term $H_k \dot{\theta}_k$ is the relative body velocity between link $k.i$ and link $k.o$. The term $A_{k.i}^{k.o} V_{k.i}^b$ transforms the body velocity of link $k.i$ to link $k.o$ coordinates. The Coriolis terms are contained in a_i^b and in the last term of $b_{k.o}^b$. The first term in $b_{k.o}^b$ is the gravitational force written in the coordinates of link $k.o$.

Algorithm 4.3 Workspace Dynamics Calculation

```
for $k = 1$ to 13 do
    $z_k = b_k^b;$
    $P_k = M_k^b;$
end for
Apply fixed base approximation
    $P_7 = P_{13} = 10^6 M_1^b;$
    $z_7 = z_{13} = [0\ 0\ 0\ 0\ 0\ 0]^T;$
for $k = 13$ to 2 do
    $D_k = H_k^T P_{k.o} H_k;$
    $K_k = H_k (D_k)^{-1} H_k^T;$
    $L_k = [I - K_k P_{k.o}] A_{k.i}^{k.o};$
    $P_{k.i} = P_{k.i} + L_k^T P_{k.o} L_k;$
    $z_{k.i} = z_{k.i} + L_k^T (P_{k.o} a_k^b + z_{k.o});$
end for
```

4.2.3 Workspace Dynamics Calculation

The *approximate* workspace dynamics are calculated through a recursion from the feet to the body. The algorithm calculates the articulated body (AB) inertia (Featherstone, 1983) and bias forces of the inboard link based on the AB inertia and bias force of the outboard link. The AB inertia and bias forces for the central body are calculated in the last step in the iteration. The calculations are given in Algorithm 4.3.

The bias forces, z_k, and AB inertias, P_k, are first initialized as shown in Algorithm 4.3. The fixed base approximation is then applied to the two feet. The inboard recursion calculates the AB inertia for each link based on the AB inertia of the outboard link. The approximate workspace inertia is P_1, and the approximate Coriolis and gravitational forces are given in z_1. Given an external wrench, F_1, acting at the center of mass of the central body, the approximate workspace dynamics are $F_1 = P_1 V_1^b + z_1$.

The approximate calculation of the workspace inertia is given in (Lilly, 1993) based on the *big base* assumption. An exact calculation of the workspace dynamics for serial chains is given in (Kreutz-Delgado et al., 1992) in terms of the spatial operator algebra.

4.2.4 Desired Wrench Calculation

In this section, the algorithm to calculate the desired wrench acting at the center of mass and written with respect to the body frame is given. The desired wrench is denoted by F_1^{bd} and is realized through the joint torques that are calculated in Section 4.2.6.

If the feet are fixed to the floor and the legs are non-singular, then the equation relating the external body wrench, F_1^b, acting at the center of mass of link 1; the exact workspace inertia, $\tilde{P}_1$; and the exact workspace Coriolis and gravitational forces, $\tilde{z}_1$, is

$$F_1^b = \tilde{P}_1 \dot{V}_1^b + \tilde{z}_1, \tag{4.2}$$

where $\dot{V}_1^b$ is the time derivative of the body velocity (the body acceleration) of link 1. Algorithm 4.3 produces P_1 and z_1 which approximates $\tilde{P}_1$ and $\tilde{z}_1$, respectively. The desired force is designed to be in the form

$$F_1^{bd} = P_1 X^b + z_1, \tag{4.3}$$

where X^b is a term that is designed below. If F_1^{bd} is exactly realized through the joint torques ($F_1^b = F_1^{bd}$), $\tilde{P}_1 = P_1$, and $\tilde{z}_1 = z_1$, then

$$\tilde{P}_1 \dot{V}_1^b + \tilde{z}_1 = P_1 X^b + z_1, \tag{4.4}$$

implying that

$$P_1(\dot{V}_1^b - X^b) = 0. \tag{4.5}$$

Since P_1 is positive definite, then

$$\dot{V}_1^b = X^b. \tag{4.6}$$

The term X^b is now designed with the assumption that the body acceleration is directly controlled through X^b. This is not realized in a real system due to actuator limits, unmodeled dynamics, sliding feet, and the small difference between the exact and approximate workspace dynamics. The term X^b is designed to drive the position of the center of mass of link 1 with respect to the inertial frame to track a desired trajectory and to drive the body orientation to track a desired orientation with respect to the inertial frame. X^b is also designed to decouple the error dynamics of the center of mass from the orientation error dynamics. X^b is also designed to produce error dynamics relative to the spatial frame and to provide an intuitive relationship between gains, set points, trajectories, and the resulting motion.

The body acceleration is composed of two parts, and X^b is divided into two corresponding parts:

$$\dot{V}_1^b = \begin{bmatrix} \ddot{p}_{0,1}^b \\ \dot{\omega}_{0,1}^b \end{bmatrix} \tag{4.7}$$

and

$$X^b = \begin{bmatrix} X_c^b \\ \\ X_R^b \end{bmatrix}. \tag{4.8}$$

The kinematic equations are then

$$\ddot{p}_{0,1}^b = X_c^b \tag{4.9}$$

and

$$\dot{\omega}_{0,1}^b = X_R^b. \tag{4.10}$$

The controller for the center of mass is first designed by designing X_c^b. First note that $\dot{p}_{0,1}^b = R_{0,1}^T \dot{p}_{0,1}^s$ where $\dot{p}_{0,1}^s$ is the velocity of the center of mass of link 1 with respect to the inertial frame, and $R_{0,1}$ is a rotation matrix which maps body coordinates to inertial coordinates . It follows that

$$R_{0,1}\ddot{p}_{0,1}^b = -\omega_{0,1}^s \times \dot{p}_{0,1}^s + \ddot{p}_{0,1}^s \tag{4.11}$$

from a straightforward calculation. Let

$$X_c^b = R_{0,1}^T (X_c^s - \omega_{0,1}^s \times \dot{p}_{0,1}^s). \tag{4.12}$$

Equations (4.9), (4.11), and (4.12) then imply that

$$\ddot{p}_{0,1}^s = X_c^s. \tag{4.13}$$

Let $e_c^p = p_{0,1}^s - p_{0,1}^{sd}$ where $p_{0,1}^{sd}$ is the desired position of the center of mass with respect to the inertial frame. Also, let $X_c^s = \ddot{p}_{0,1}^{sd} - K_c^v \dot{e}_c^p - K_c^p e_c^p$ where K_c^v and K_c^p are positive definite, symmetric, 3×3 gain matrices. Equation (4.13) then implies that

$$\ddot{e}_c^p + K_c^v \dot{e}_c^p + K_c^p e_c^p = 0. \tag{4.14}$$

If Equation (4.14) holds, then $e_c^p \to 0$ exponentially in time. See (Murray et al., 1994) (page 192) for a proof. The position error will only converge to zero if Equation (4.6) holds, and this relies on the assumptions that the feet are fixed to the ground and that the dynamic terms of the model and the controller model are equal.

Creating a tracking controller for the orientation is more difficult than creating a controller for the position terms. A tracking orientation controller is now created by designing X_R^b. Define the term X_R^s to be $R_{0,1}X_R^b$. Since $R_{0,1}\dot{\omega}_{0,1}^b = \dot{\omega}_{0,1}^s$,

$$\dot{\omega}_{0,1}^s = X_R^s. \tag{4.15}$$

X_R^s is then designed to produce error dynamics with terms expressed relative to the spatial frame.

A difficulty in designing a tracking controller for controlling orientation is determining how to create the proportional action term. In other words, determining the difference between two rotation matrices. Consult (Bullo and Murray, 1997) for a recent reference on orientation error and more general measurements of errors on manifolds.

The error chosen for the orientation controller is based on the matrix R_e given by

$$R_e = R_{0,1}R_{0,d}^T, \tag{4.16}$$

where $R_{0,d}$ is the desired orientation of the body. Equation (4.16) implies that $R_e R_{0,d} = R_{0,1}$. This relation provides an interpretation of the meaning of R_e. Since successive spatial rotations multiply on the left, R_e is interpreted as the matrix represented relative to the spatial frame which rotates the desired frame into the body frame, $R_{0,1}$. Also, since $R_{0,d} = R_e^T R_{0,1}$, R_e^T rotates the body frame into the desired frame. Again, R_e^T is represented relative to the spatial frame.

A proportional action term is created from the matrix R_e. Let $q_{0,1}$ be a quaternion representing $R_{0,1}$ and let $q_{0,d}$ represent $R_{0,d}$. A quaternion representing R_e is

$$q_{et} = q_{0,1} \star \bar{q}_{0,d} = (q_{et}^s, q_{et}^v). \tag{4.17}$$

The quaternion $-q_{et}$ also represents R_e. The quaternion chosen to represent the error matrix is the one having the axis-angle interpretation with the angle in $[0, \pi]$ radians. This error quaternion, $q_e = (q_e^s, q_e^v)$, is the one with a non-negative scalar component (see Appendix A for more details). The vector component of the error quaternion is used to generate the proportional action. The orientation error term, e_R^p, is then given by

$$e_R^p = \sin(\frac{\theta_e}{2})\omega_e = q_e^v, \tag{4.18}$$

where $\theta_e \in [0, \pi]$, $\| \omega_e \| = 1$, and ω_e is a vector represented in the spatial frame. Rotating the desired frame about the spatial axis ω_e by an angle θ_e brings the desired frame to the body frame. Rotating the body frame about axis ω_e by $-\theta_e$ radians brings the body frame to the desired frame.

The velocity error term is chosen to be the spatial angular velocity of matrix R_e given by

$$\hat{\omega}_e^s = \dot{R}_e R_e^T. \tag{4.19}$$

One can show by expanding Equation (4.19) that

$$\omega_e^s = \omega_{0,1}^s - R_e \omega_{0,d}^s, \tag{4.20}$$

and

$$\dot{\omega}_e^s = \dot{\omega}_{0,1}^s - R_e \dot{\omega}_{0,d}^s - \omega_{0,1}^s \times (R_e \omega_{0,d}^s). \tag{4.21}$$

Also,

$$\begin{aligned}\dot{q}_e &= (0, \frac{\omega_e^s}{2}) \star q_e \\ &= (-q_e^v \cdot \frac{\omega_e^s}{2}, q_e^s \frac{\omega_e^s}{2} + \frac{\omega_e^s}{2} \times q_e^v).\end{aligned} \tag{4.22}$$

Since q_e is a unit quaternion, $q_e^s = \pm\sqrt{1-(q_e^v)^T q_e^v}$. To represent the orientation error, the quaternion with the positive scalar component is used, and therefore, $q_e^s = +\sqrt{1-(q_e^v)^T q_e^v}$. Using Equation (4.22), one derives that

$$\dot{q}_e^v = \frac{1}{2}\left[I\sqrt{1-(q_e^v)^T q_e^v} - \hat{q}_e^v\right]\omega_e^s \overset{\triangle}{=} Q(q_e^v)\omega_e^s. \tag{4.23}$$

Given gain matrices K_R^v and K_R^p, X_R^s is designed to be

$$X_R^s = R_e\dot{\omega}_d^s + \omega_{0,1}^s \times (R_e\omega_{0,d}^s) - K_R^v\omega_e^s - K_R^p q_e^v. \tag{4.24}$$

Equation (4.24) and (4.15) imply that

$$\dot{\omega}_{0,1}^s - X_R^s = \dot{\omega}_e^s + K_R^v\omega_e^s + K_R^p q_e^v = 0. \tag{4.25}$$

The orientation error dynamics are then

$$\begin{bmatrix}\dot{q}_e^v \\ \dot{\omega}_e^s\end{bmatrix} = \begin{bmatrix}Q(q_e^v)\omega_e^s \\ -K_R^p q_e^v - K_R^v\omega_e^s\end{bmatrix}, \tag{4.26}$$

and the Jacobian for the linearization evaluated at the equilibrium, $(q_e^v, \omega_e^s) = (0,0)$, is

$$\begin{bmatrix}0 & \frac{1}{2}I \\ -K_R^p & -K_R^v\end{bmatrix}. \tag{4.27}$$

If the gain matrices are positive definite and symmetric, then as is proved in (Murray et al., 1994) (page 192), the eigenvalues of the Jacobian matrix lie in the left hand plane. This proves locally that the orientation error, q_e^v, converges to zero exponentially. Various orientation trajectories and many initial conditions have been simulated with this orientation controller, and the error has always converged to zero. This suggests that there may exist a stronger theorem for this controller with global convergence properties.

The orientation controller developed here is very similar to the left error orientation controller presented in (Bullo and Murray, 1997). The same error matrix is used, but the proportional action term differs. The velocity action is also different. The development of equation (4.23) is also influenced by the compatibility error condition presented in (Bullo and Murray, 1997). Consult (Bullo and Murray, 1997) for more information on tracking controllers for mechanical systems on manifolds.

The algorithm to calculate the desired wrench is shown in Algorithm 4.4. The components of X^b are calculated and then used to calculate the desired body wrench.

4.2.5 Desired Wrench Distribution

In this section, the desired body wrench, F_1^{bd}, is divided into two contributions: one from the left leg and one from the right leg. Distributing the wrench needs to be done carefully to avoid creating unnecessary and detrimental internal forces and torques.

Let F_r be the wrench acting on the right foot from the ground and represented in the body frame of the right foot (link 13). Let F_l denote the corresponding wrench for the left foot (link 7). The desired body wrench is given by

$$F_1^{bd} = \mathrm{Ad}^T_{g_{1,7}^{-1}} F_l + \mathrm{Ad}^T_{g_{1,13}^{-1}} F_r = \begin{bmatrix} \mathrm{Ad}^T_{g_{1,7}^{-1}} & \mathrm{Ad}^T_{g_{1,13}^{-1}} \end{bmatrix} \begin{bmatrix} F_l \\ F_r \end{bmatrix} \triangleq G \begin{bmatrix} F_l \\ F_r \end{bmatrix} \tag{4.28}$$

and is a result of transforming F_l and F_r to the frame of link 1 and adding. First note that

Algorithm 4.4 Desired Wrench Calculation

```
**given:** trajectory data, MB states, workspace dynamics, and gains.
$e_c^p = p_{0,1}^s - p_{0,1}^{sd}$; $\dot{e}_c^p = \dot{p}_{0,1}^s - \dot{p}_{0,1}^{sd}$;
$X_c^s = \ddot{p}_{0,1}^{sd} - K_c^v \dot{e}_c^p - K_c^p e_c^p$;
$X_c^b = R_{0,1}^T (X_c^s - \omega_{0,1}^s \times p_{0,1}^s)$;
$q_{et} = q_{0,1} \star \bar{q}_{0,d} = (q_{et}^s, q_{et}^v)$;
**if** $q_{et}^s \geq 0$ **then**
    $q_e = (q_e^s, q_e^v) = (q_{et}^s, q_{et}^v)$;
**else**
    $q_e = (q_e^s, q_e^v) = (-q_{et}^s, -q_{et}^v)$; {$R_e$ *is represented by* $\pm q_{et}$}
**end if**
$\omega_e^s = \omega_{0,1}^s - R_e \omega_{0,d}^s$;
$X_R^s = R_e \dot{\omega}_{0,d}^s + \omega_{0,1}^s \times (R_e \omega_{0,d}^s) - K_R^v \omega_e^s - K_R^p q_e^v$;
$X_R^b = R_{0,1}^T X_c^s$;
$X^b = \begin{bmatrix} X_c^b \\ X_R^b \end{bmatrix}$;
$F_1^{bd} = P_1 X^b + z_1$;
```

the matrix G has full row rank and then solve Equation (4.28) for F_l and F_r by computing a minimum norm solution in the following way:

$$\begin{bmatrix} F_l \\ F_r \end{bmatrix} = G^T(GG^T)^{-1}F_1^{bd}. \tag{4.29}$$

Let $\tilde{F}_k^{bd}$ be the desired wrench acting on body $k.i$ from body $k.o$ and written with respect to frame $k.i$. Let F_k^{bd} be the desired wrench acting on link $k.o$ from link $k.i$ and written with respect to frame $k.o$. The wrenches acting on the central body, link 1, from the left and right legs are then

$$\tilde{F}_2^{bd} = \mathrm{Ad}^T_{g_{1,7}^{-1}} F_l \tag{4.30}$$

and

$$\tilde{F}_8^{bd} = \mathrm{Ad}^T_{g_{1,13}^{-1}} F_r. \tag{4.31}$$

In this way,

$$F_1^{bd} = \tilde{F}_2^{bd} + \tilde{F}_8^{bd}, \tag{4.32}$$

and the desired wrench is divided into a contribution from the left and right legs. The calculation is given and summarized in Algorithm 4.5.

4.2.6 Torque Calculation

The joint torques are now calculated in an outboard recursion given the wrench contributions from the left and right legs.

Let τ_l be the joint torques for the left leg and τ_r be the joint torques for the right leg. Let J_l be the body Jacobian for the left leg and J_r be the body Jacobian for the right leg.

Algorithm 4.5 Desired Wrench Distribution

```
given: $F_1^{bd}, g_{1,7}, g_{1,13}$
$G = [\mathrm{Ad}^T_{g_{1,7}^{-1}} \;\; \mathrm{Ad}^T_{g_{1,7}^{-1}}]$; {*G has full row rank*}
$\begin{bmatrix} F_l \\ F_r \end{bmatrix} = G^T(GG^T)^{-1}F_1^{bd};$
$\tilde{F}_2^{bd} = \mathrm{Ad}^T_{g_{1,7}^{-1}} F_l;$
$\tilde{F}_8^{bd} = \mathrm{Ad}^T_{g_{1,13}^{-1}} F_r$; $\{F_1^{bd} = \tilde{F}_2^{bd} + \tilde{F}_8^{bd}\}$
```

If the feet are fixed, J_l maps joint velocities for the left leg to the body velocity of link 1, and J_r maps right leg joint velocities to the body velocity of link 1. The joint torques are then $\tau_l = J_l^T \tilde{F}_2^{bd}$ and $\tau_r = J_r^T \tilde{F}_8^{bd}$. The Jacobian transpose relationship is a static equilibrium calculation and can be calculated by finding the equilibrium forces on each link as though each link were at rest. The recursive algorithm to calculate the joint torques is given in Algorithm 4.6. The joint torques are calculated in an outboard iteration over the joints. First create F_k^{bd} by using equal and opposite forces across joint k: $F_k^{bd} = -\mathrm{Ad}^T_{g_{k.o,k.i}^{-1}} \tilde{F}_k^{bd}$. Then project F_k^{bd} across the joint axis to get the joint torque: $\tau_k = H_k^T F_k^{bd}$. Then update $\tilde{F}_{k+1}^{bd}$ by noting that link $k.o$ is in equilibrium for the Jacobian calculation: $\tilde{F}_{k+1}^{bd} = -F_k^{bd}$.

4.3 Simulation Results

The controlled multibody system is created and simulated in *Impulse* (Mirtich, 1996) and simulation results are presented in this section. *Impulse* is a multibody simulator that handles contact through impulses. The controller is written in C and interfaced to

Algorithm 4.6 Torque Calculation

```
given: $\tilde{F}_2^{bd}$ and $\tilde{F}_8^{bd}$; $g_{k.o,k.i}$ and $H_k$ for all $k \in \{2, \cdots, 13\}$;
for $k = 2$ to 13 do
  $F_k^{bd} = -\mathrm{Ad}^T_{g^{-1}_{k.o,k.i}} \tilde{F}_k^{bd}$; {equal and opposite forces across joint $k$}
  $\tau_k = H_k^T F_k^{bd}$;
  if link $k.o$ is not a leaf then
    $\tilde{F}_{k+1}^{bd} = -F_k^{bd}$; {body $k.o$ is in equilibrium}
  end if
end for
```

the simulator. Two experiments are presented, and animations of the experiments can be found at http://robotics.eecs.berkeley.edu/~wents. For the two experiments, the coefficient of friction is 0.3, the coefficient of restitution is 0.2, and the gravitational acceleration is $9.81\frac{\mathrm{m}}{\mathrm{s}^2}$. The multibody system initially has zero velocity. The center of mass of the central body is initially placed at $(0, 0, 103.5)$ cm, and the central body is initially rotated about the $-y$ axis by 5 degrees. The system is initially slightly above the ground and collides with the ground right after the start of the simulation. The initial joint angles are $\{\theta_2, \cdots, \theta_{13}\} = \{0, 40, 0, -77.4595, 0, 42.4595, 0, 40, 0, -77.4595, 0, 42.4595\}$ degrees. The controller gains are

$$K_c^p = \begin{bmatrix} 50 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 200 \end{bmatrix}, K_c^v = \begin{bmatrix} 7.07 & 0 & 0 \\ 0 & 7.07 & 0 \\ 0 & 0 & 14.142 \end{bmatrix},$$

$$K_R^p = \begin{bmatrix} 50 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 50 \end{bmatrix}, \text{ and } K_R^v = \begin{bmatrix} 7.07 & 0 & 0 \\ 0 & 7.07 & 0 \\ 0 & 0 & 7.07 \end{bmatrix}.$$

The vertical gain in the Z position is chosen based on the experiments on astronauts in (Newman et al., 1996). The remaining gains are chosen arbitrarily with a damping ratio of 0.5. The controller is called at 500 Hz. On an SGI Indigo (R4000, 100 MHz) workstation, each control loop takes 7 milliseconds to compute. For experiment 2, a three second simulation takes 2066 seconds (34 minutes 26 seconds) to compute.

Hand tweaking of the controller gains and set points were not required to produce the simulations presented in this section. The fact that the controller is model-based and that the errors are formed in the central body provides an intuitive relationship between the error dynamics, set points, and gains. The 12 joint torques are coordinated by the design of the control system.

For Experiment 1, the set point for the center of mass is $(-1, 0, 110)$ cm, and the desired orientation is $q_{0,d} = (0.99144486, -0.13052619, 0.0, 0.0)$ corresponding to a 15 degree rotation about the $-X$ axis. The system moves to the desired pose. At approximately 0.18 seconds, the front part of the left foot raises off of the floor. This disturbance is felt in the system, but the controller recovers and drives the errors to zero.

The joint angles over time are shown in Figure 4.3. The evidence of the left foot raising is shown in the trajectory of joint 7. There is a slight drifting in the joint angles as time progresses. This is believed to be caused by the feet sliding relative to the ground as can been seen in the 3D graphical display of the multibody system. The sliding feet is

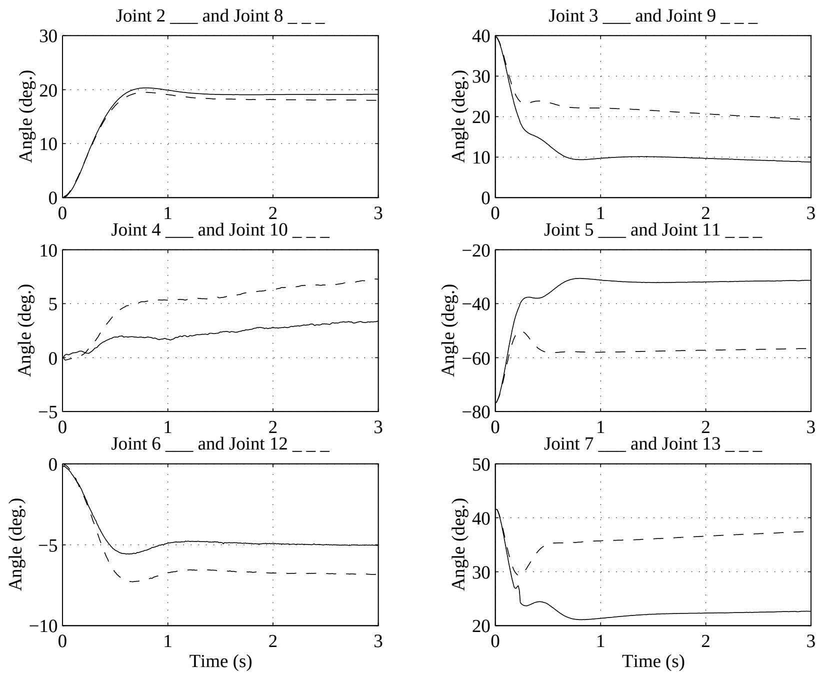

Figure 4.3: Joint Angles for Experiment 1

believed to be an artifact of the impulsive contact model. The controller corrects for the errors caused by the feet sliding.

The joint torques versus time are shown in Figure 4.4. The maximum joint torque occurs in the right knee at 0 seconds and is 192 N-m. The joint torques for joints 2, 3, 4, 6, 8, 9, 10, and 12 are substantially smaller. Evidence of the feet sliding are seen in the joint torques for joints 6 and 12.

The position of the center of mass of link 1 is shown in Figure 4.5. The position of the center of mass converges to a neighborhood of the desired value. The convergence in the Z coordinate is faster than in the other two coordinates by the choice of the position gains. The desired position in the Y coordinate deviates from zero initially and then recovers.

The orientation over time is shown in Figure 4.6 and is given by $q_{0,1}$. The orienta-

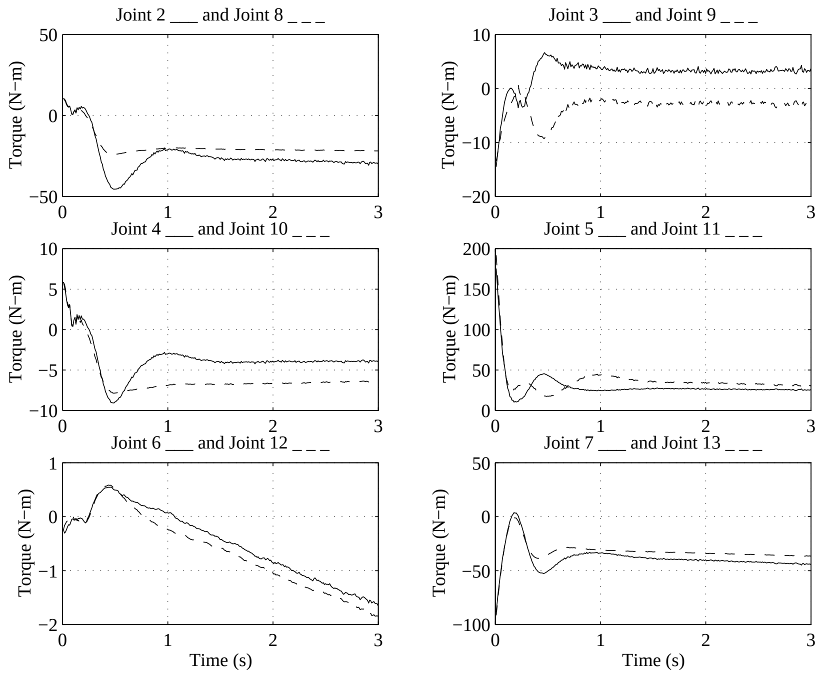

Figure 4.4: Joint Torques for Experiment 1

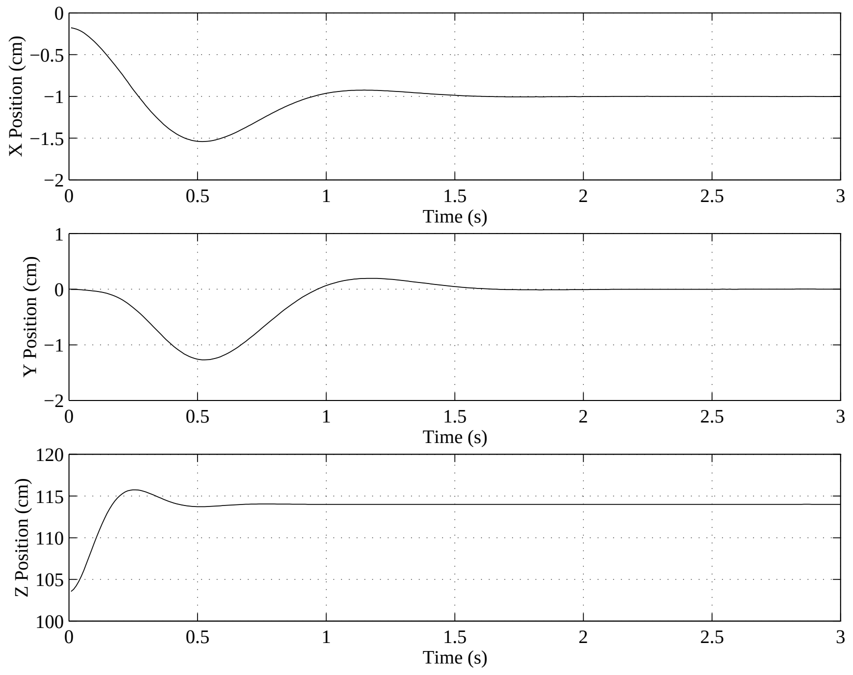

Figure 4.5: Body Position for Experiment 1

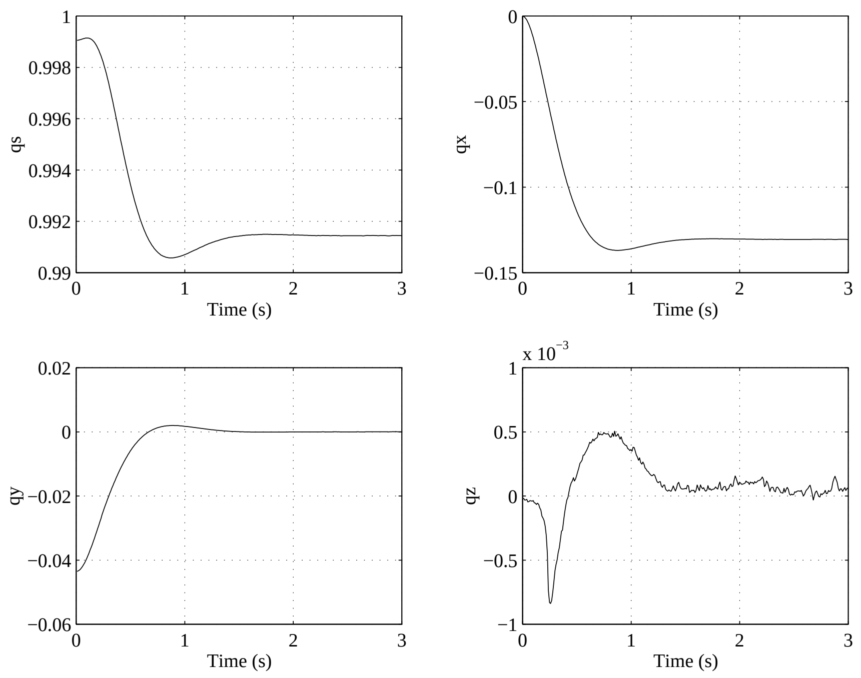

Figure 4.6: Body Orientation for Experiment 1

tion converges to a neighborhood of the desired orientation. There are disturbances caused by the collisions in impulse as well as the fact that the feet slide over time.

The errors in position and orientation are shown in Figure 4.7 and in Figure 4.8. The position error converges to a neighborhood of zero and the quaternion error, q_{et}, converges to the identity quaternion. The controlled system experiences the disturbances caused by the collisions as well as the feet sliding.

The desired body forces over time are shown in Figure 4.9. The desired force in the Z direction is a maximum of 1687 N at 0 seconds and converges to approximately 594 N at 3 seconds.

The desired body torque is shown in Figure 4.10. The largest torque is in the body Y axis and is -49 N-m. The desired torque about the Z axis is between -2 and 2 N-m. The

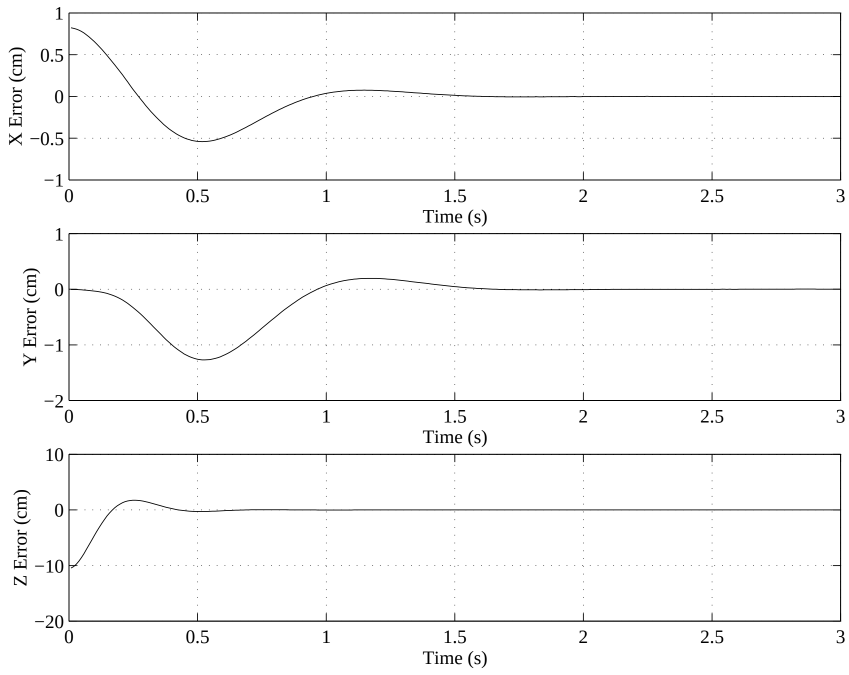

Figure 4.7: Body Position Error for Experiment 1

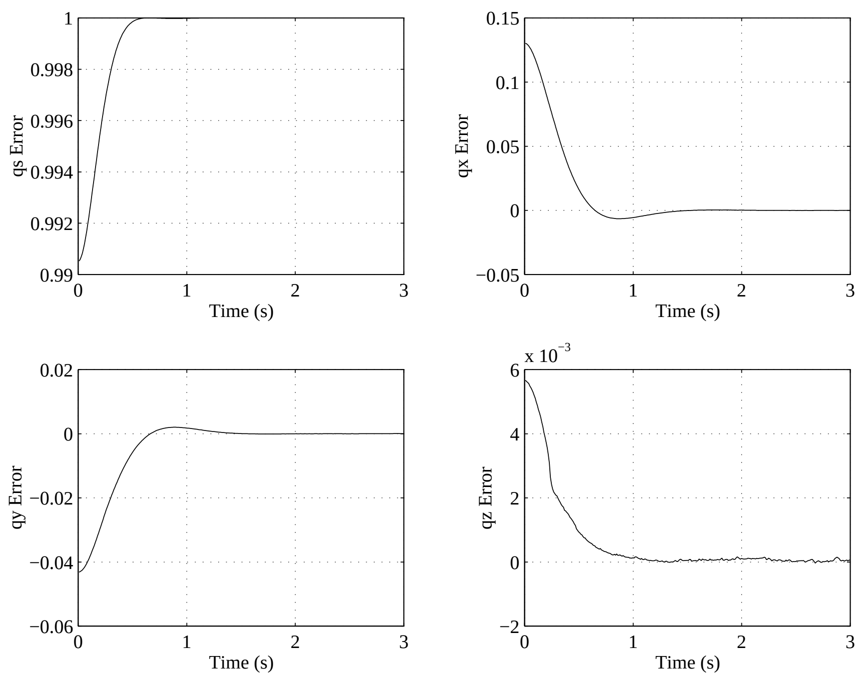

Figure 4.8: Body Orientation Error for Experiment 1

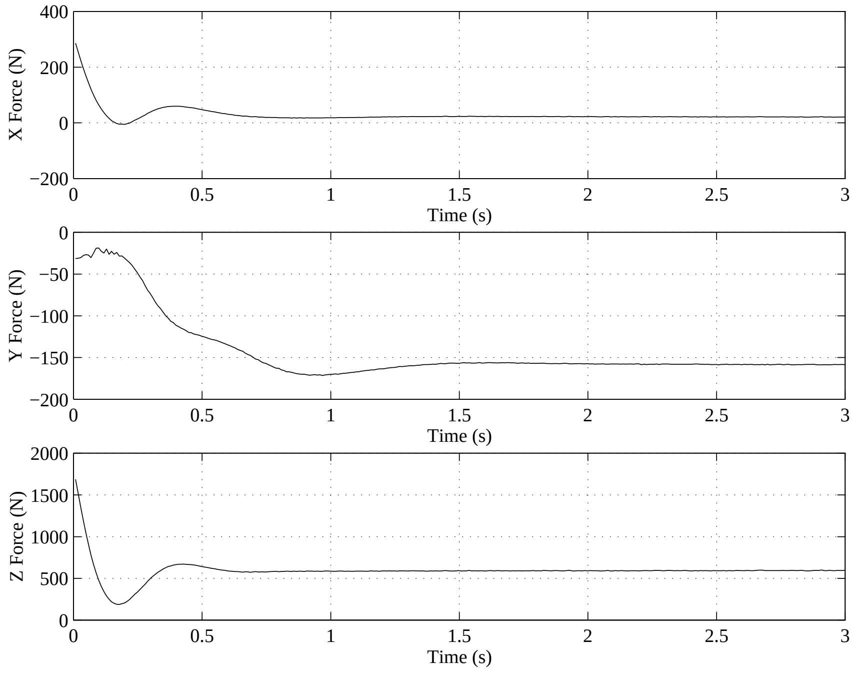

Figure 4.9: Desired Body Force for Experiment 1

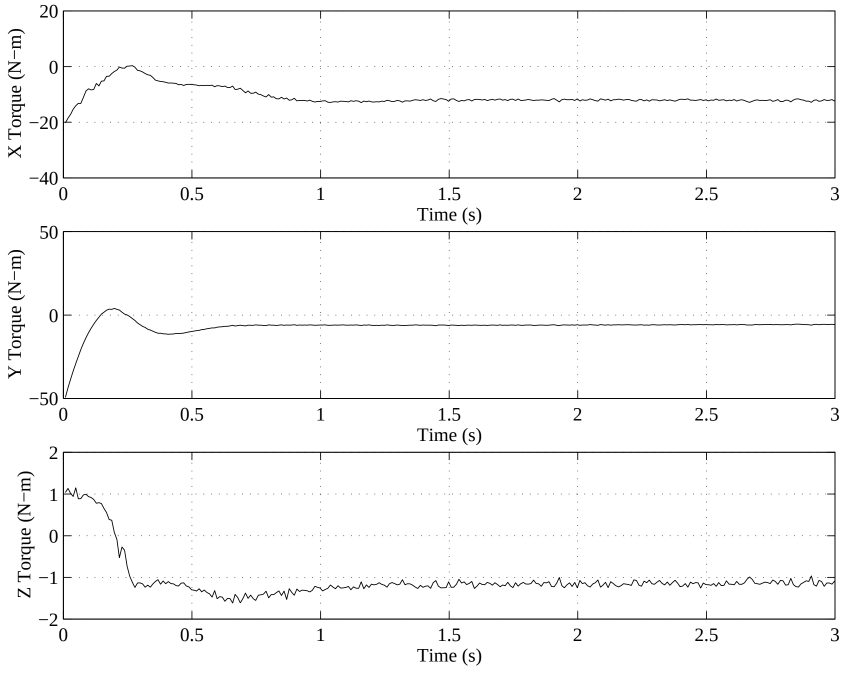

Figure 4.10: Desired Body Torque for Experiment 1

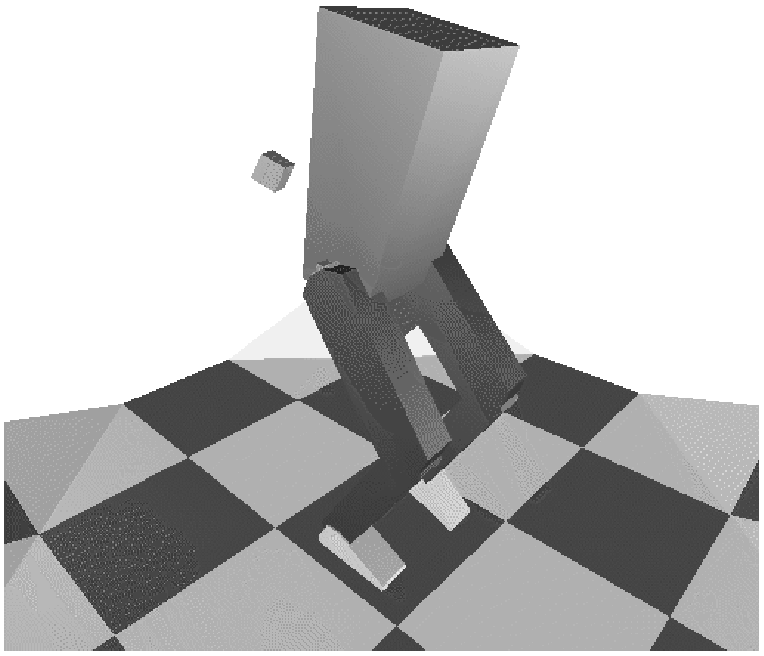

Figure 4.11: Screen Capture of Experiment 2

desired torque about the X axis is between -20 and 0 N-m.

For Experiment 2, the set point for the center of mass of link 1 is $(-1, 0, 114)$ cm, and the desired orientation is $q_{0,d} = (0.99144486, -0.13052619, 0, 0)$ corresponding to a 15 degree rotation about the $-Y$ axis. The system begins rising to the desired pose while a 1.8 Kg block is thrown at the body from the position of $(50, 10, 120)$ cm with a velocity of $(-450, 0, 20)$ cm/s (approximately 4 lbs. thrown at 10 mph). The block collides with the body at approximately 0.08 seconds. The two feet begin to rise at approximately 0.13 seconds, and the system rests on the front of the two feet as the heels rise. The right foot makes contact with the ground at approximately 0.36 seconds, and the left foot makes contact at approximately 0.38 seconds. A snapshot of the simulation in *Impulse* at 0.32 seconds is shown in Figure 4.11. During this simulation as well, the feet drift relative to the

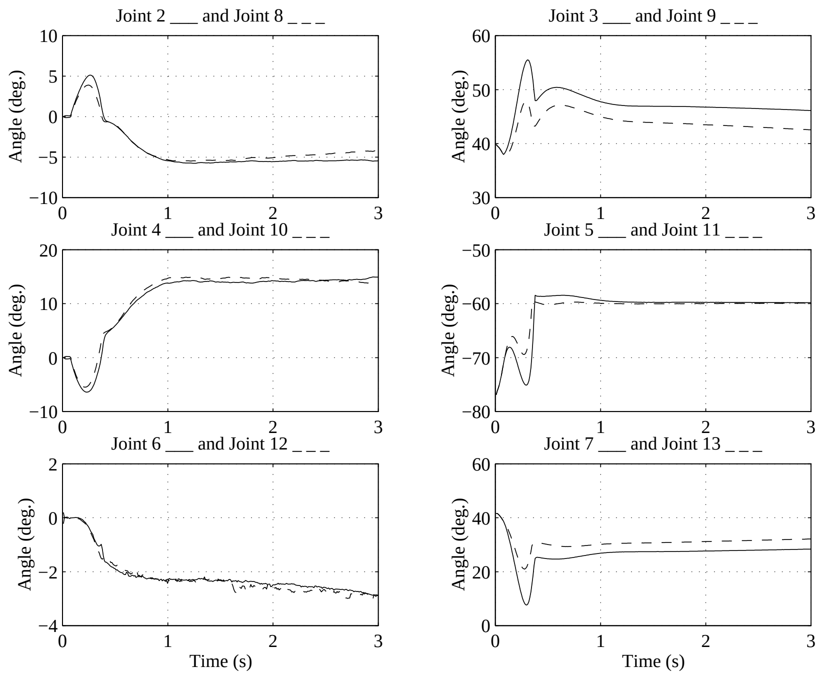

Figure 4.12: Joint Angles for Experiment 2

ground as the simulation proceeds. The controller compensates for the collisions, the feet leaving contact with the ground, the impulsive contact forces, and the sliding feet to drive the error to a neighborhood of zero.

The joint angles over time for Experiment 2 are shown in Figure 4.12. The sharp change in joint angles due to the collision can be seen at 0.08 seconds. The feet making contact with the ground at 0.36 and 0.38 seconds is also seen in the data. After these disturbances, the joint angles settle with a small drifting due to the sliding feet.

The joint torques over time are shown in Figure 4.13. The torques fluctuate after the collision and have sharp changes after the changes in the contact of the feet with the ground.

The position of the center of mass relative to the inertial frame is shown in Fig-

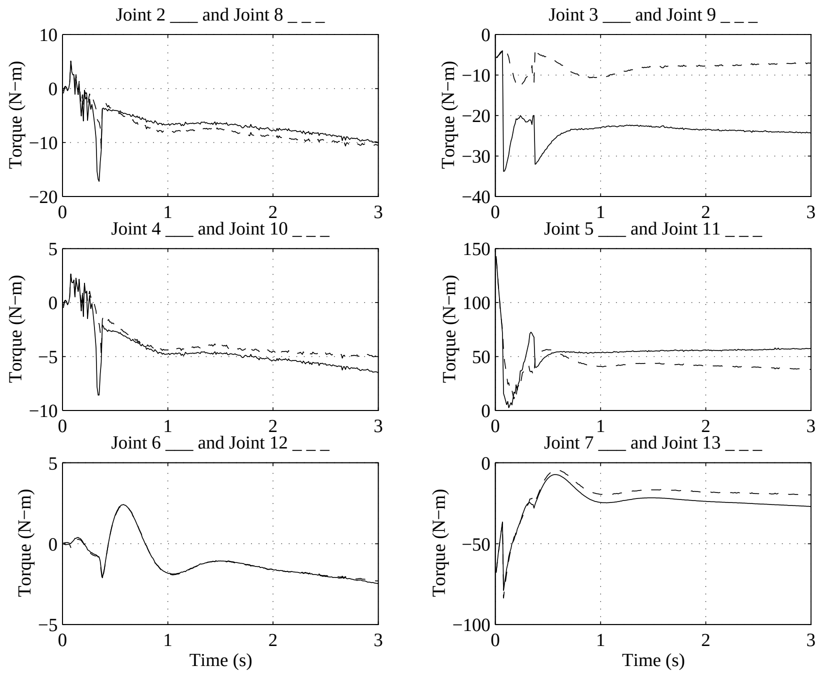

Figure 4.13: Joint Torques for Experiment 2

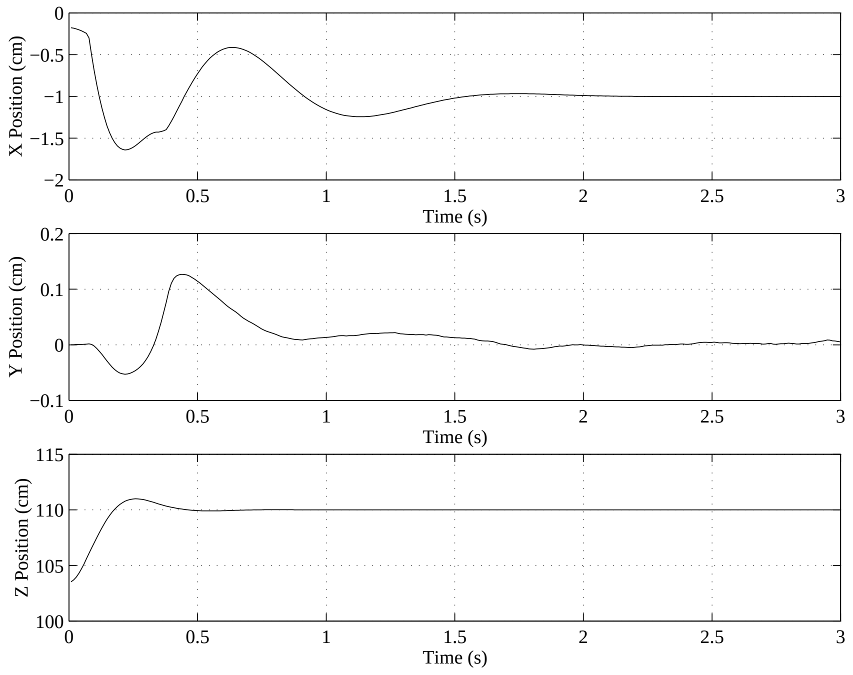

Figure 4.14: Body Position for Experiment 2

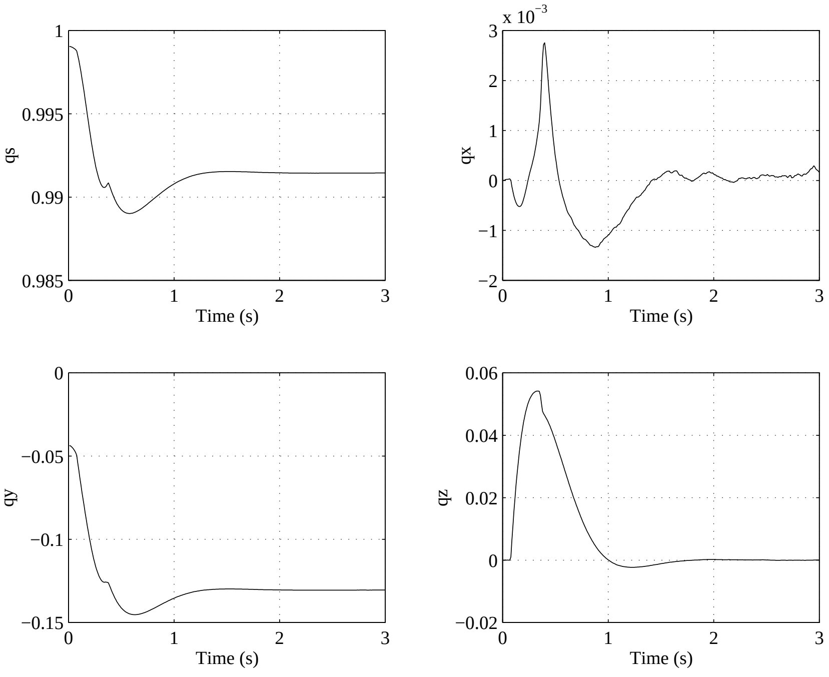

Figure 4.15: Body Orientation for Experiment 2

ure 4.14. The errors converge to a neighborhood of zero. The convergence is faster in the Z coordinate. The Y position is seen to drift about zero, and this is believed to be due to the sliding feet. The collision disturbance is seen to push the body in the $-X$ direction and then the controller compensates for the disturbance.

The orientation of the central body is shown in Figure 4.15. The collision causes the body to rotate about the $-Y$ axis and to rotate about the Z axis as can be seen in the plot of q_y and q_z. The controller compensates for the disturbances and drives the error to a neighborhood of zero.

The position errors are shown in Figure 4.16. The position errors converge to a neighborhood of zero. The controller compensates for the disturbances to the system.

The orientation error is shown in Figure 4.17. The orientation error converges to

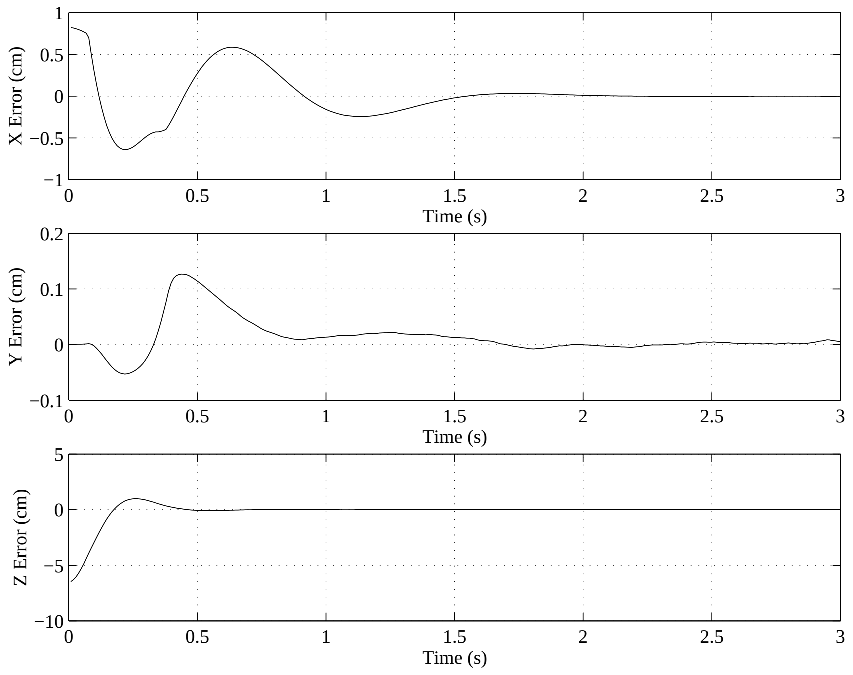

Figure 4.16: Body Position Error for Experiment 2

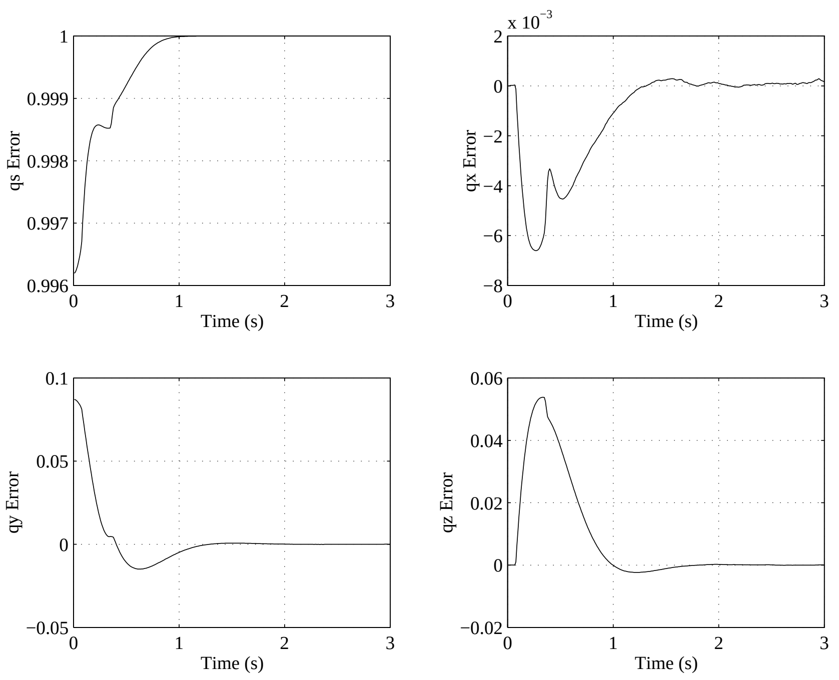

Figure 4.17: Body Orientation Error for Experiment 2

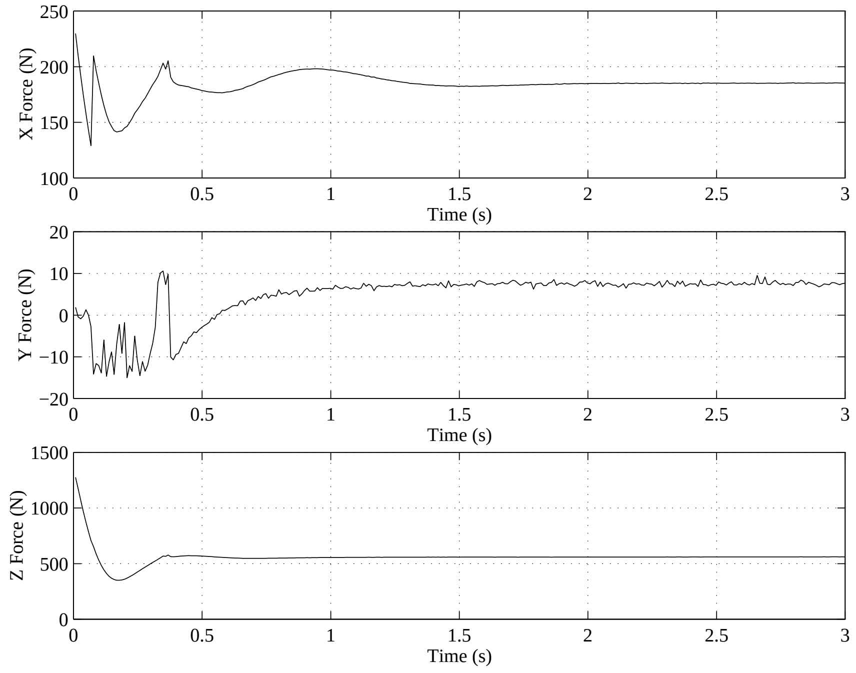

Figure 4.18: Desired Body Force for Experiment 2

a neighborhood of the identity quaternion. The disturbances caused by the collision and the contact changes of the feet with the floor can be seen in this figure.

The desired body force is shown in Figure 4.18. The sharp changes in the desired force caused by the disturbances are seen in the data. The small peaks shown in the Y force data may be caused by the impulsive contact model as well as the sliding feet.

The desired body torque is shown in Figure 4.19. The desired torque compensates for the disturbances and contact changes in the feet with the ground. The small peaks in the desired torques may be caused by the impulsive contact model and the sliding feet.

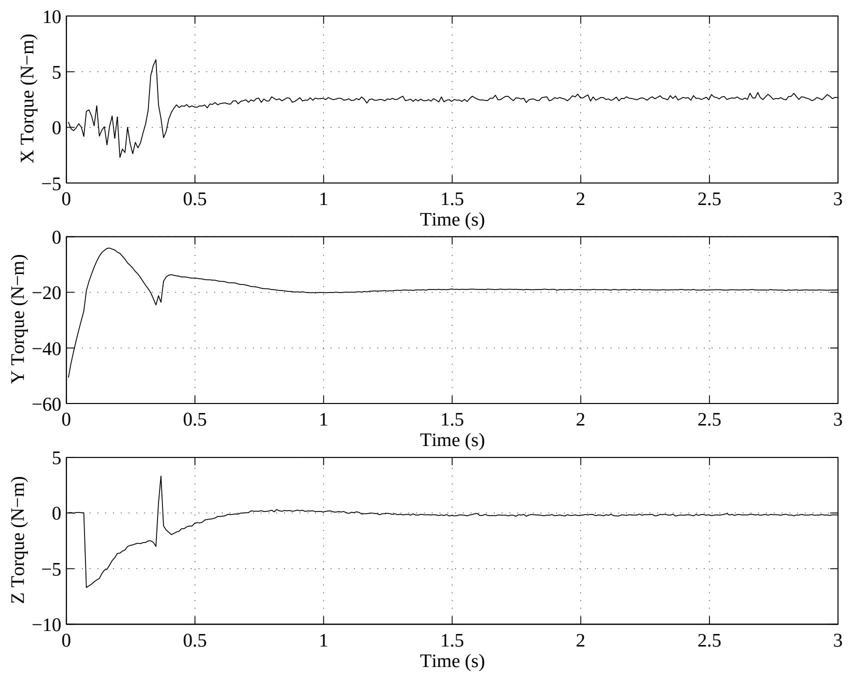

Figure 4.19: Desired Body Torque for Experiment 2

4.4 Chapter Summary

The balancing controller presented in this chapter extends the planar results in (Wendlandt and Sastry, 1996) and applies the control method to produce a workspace balancing controller for a 3D multibody model of a biped. A model-based controller is formed in the workspace of the 3D biped, and the model is efficiently calculated through link to link iterations in the multibody structure. The recursive calculations free the control designer from having to symbolically create the complicated, nonlinear model.

The balancing controller has desirable properties observed through the simulation results. It is easy to choose gains and set points. The simpler error dynamics are easier to tune for a desired response. A sensitivity of the dynamic response to small changes in the parameters was not observed. The system also has a large operating region for the gains simulated. The joint torques are automatically coordinated by design and one does not tune individual joint gains. The controlled system is also seen to react to disturbances as is seen in the simulation of the collision with a projectile. Hand tweaking of various parameters in the controller was not necessary. However, the controlled system can not recover from the projectile disturbance if the velocity of the projectile is too large. The balancing controller may serve well as a basis for more complicated controllers for walking, running, jumping, changing direction, and adapting to loads.

Improvements can be made to the balancing controller. The current controller is model-based and does not adapt to changes in the model parameters. Adaptation may be used to react to loads as well as correct for modeling errors. See (Berghuis, 1993) for a study of model-based robot controllers with and without adaptation. This controller is not designed to operate near singularities. For example, the leg is at a singularity when the knee joint angle is at zero degrees. A modification to the existing controller or the design of a new controller is necessary to handle singularities. The controller is not designed to handle redundancy in the legs. If one toe joint is added, then each leg will have 7 degrees of freedom. The controller needs to be modified or re-designed to handle redundancy. New controllers need to be designed to produce walking, running, jumping, and adapting to loads. The controller has been designed to not rely on specific joint trajectories, but it has not been shown to be predictive. One needs to perform more detailed comparisons with experiments on human subjects. The hope then is to modify the controller to better predict human motion. The controller must extract principles from the experiments and must not rely on very specific data from individual experiments. The current balancing controller relies on the tree-structure of the multibody system. Many engineering applications of multibody systems have closed-loops, and creating multibody controllers for these systems is an area of future work. Also, the workspace dynamics are calculated approximately and rely on the *big base* assumption. One may use algorithms to calculate the exact workspace dynamics although it is not clear that much is to be gained by this improvement.

The simulation method can also be improved. *Impulse* handles contact through many impulses, and this method of modeling contact does not handle continuous planar contact very efficiently. The problem experienced with the feet drifting slowly on the floor is an artifact of the impulsive contact model. The creators of *Impulse* are currently creating the ability to switch between an impulsive contact model and a constraint-based contact model, and this should eliminate the problem of the sliding feet. One may also choose to use a different contact model, and this is the motivation for the contact modeling work in Chapter 5. *Impulse* provides a nice interface to simulate the controlled multibody system and to simulate the system in many different situations. Utilizing a general multibody simulator designed to handle contact has been found to be a useful tool for developing control systems for these multibody models.

Chapter 5

Components of a Multibody Simulator

This chapter presents components of a multibody simulator specifically designed for modeling biped models in contact with the ground in real-time. The emphasis is on creating a control development platform for designing discrete-time control algorithms for human motion. In this effort, external forces are added to the methodology developed in Chapter 2. A simple model of contact is designed, and simulation results of a rigid body colliding with the ground are presented. Symplectic-momentum multibody integrators are then created whose computational cost grows linearly with the number of joints for tree-structured systems. It is shown how to combine the multibody integrator with the contact model and controller actuator torques to create a control development platform.

Researchers have been studying multibody simulation techniques for many years and have created several techniques for computer simulation. Recursive algorithms are one class of techniques that calculate the joint accelerations of tree-structured systems with a computational cost that grows linearly with the number of joints (see Chapter 3, (Featherstone, 1983), (Jain, 1991), and (Bae and Haug, 1987)). These algorithms are well-suited for tree-structured systems, and the techniques have been extended to systems with closed loops (see (Bae and Haug, 1988) and (Rodriguez et al., 1992)). Recursive techniques with closed-loops often use Baumgarte's stabilization technique (PD controllers enforcing the constraints) (Baumgarte, 1972) to counteract drifting in the loop closure constraints.

Differential-algebraic-equation (DAE) techniques treat multibody systems as a collection of differential equations and algebraic constraints. Several DAE integrators have been created (for example, MEXX (Lubich et al., 1993), DASSL (Petzold, 1982), (Brenan et al., 1989), and MBSSIM (Andrzejewski et al., 1993). Also see (Schiehlen, 1993) for additional references). These methods seem natural for systems with closed-loops and by exploiting sparsity in matrices can also achieve linear-time computational cost for tree-structured multibody systems.

Researchers have also developed contact models and have combined them with multibody simulation techniques to create general purpose simulators. The authors in (Cremer and Stewart, 1989) created *Newton* as a general purpose multibody simulator with contact. *Impulse* (Mirtich, 1996) is a recent multibody simulator that uses recursive algorithms and models contact with impulses. The author in (Baraff, 1996) created a multibody simulator with contact by utilizing DAE algorithms and by exploiting sparsity to obtain linear-time complexity for tree-structured multibody systems. These simulators have applications in animation, computer graphics, and engineering design.

Modeling contact efficiently and robustly is difficult to achieve for general purpose multibody simulators. Part of the difficulty is the fact that the physical processes involved in two materials contacting is extremely complicated and many simplifying assumptions need to be made, including the coulomb friction model. One must also detect contact with collision detection algorithms, and evolve the system while in contact. The main approaches to date are impulsive contact models (Mirtich, 1996), solving a linear complementarity problem (LCP) (Trinkle et al., 1997), (Baraff, 1996), (Pfeiffer and Glocker, 1996), and using a spring-based method (Goyal et al., 1994a), (Goyal et al., 1994b). See (Mirtich, 1996) and (Brogliato, 1996) for a more in depth discussion of contact modeling. In some applications, curvature of the contact interaction may be important as is discussed in (Rimon and Burdick, 1994). Contact simulation is important in some finite element applications, *e.g.*, car crash simulations. Contact simulation with finite element analysis is presented in (Simo and Laursen, 1992), (Peric and Owen, 1992), (Laursen and Simo, 1993), (Kulak and Schwer, 1991), and (Zhong, 1993).

Recently, a few researchers have created mechanical integrators for multibody systems. Mechanical integrators respect the structure of mechanical systems and preserve energy, momentum, and/or are symplectic. It is important in some applications that these properties are obeyed by the numerical integration scheme. For example, in space applications, momentum conservation is critical. For long-term simulations of Hamiltonian systems, symplectic integration seems important. The authors in (Barth and Leimkuhler, 1996b) particulate the rigid bodies in a multibody system with point masses. They then use symplectic-momentum integrators with constraints and general sparse matrix techniques to simulate multibody systems. The author in (Reich, 1996b) uses the methods in (Reich, 1996a) to create symplectic-momentum integrators for multibody systems. The integrator in (Reich, 1996b) is an explicit integrator which satisfies the position and velocity constraints.

For real-time simulation, one is less concerned with numerical accuracy and more concerned with getting fairly accurate results with a predictable computational cost. More accurate simulations can be performed when computational times are not critical. Implicit integrators with a fixed number of iterations have been recommended for real-time applications (de Jalón and Bayo, 1994) (pages 266-267) since implicit integrators can retain reasonable accuracies with relatively large time-steps, can remain stable for stiff systems, and can have a fixed computational cost. However, with implicit integrators, a set of non-linear equations are solved at each time-step, and it is difficult to guarantee that a solution will be found. Whether to use an implicit or explicit method really depends on the particular application. The ability to use large time steps has also been a stated advantage of mechanical integrators, but more experimentation needs to be done to provide evidence for this claim.

The chapter first presents a method to add external forces to the mechanical integrators developed in Chapter 2. A simple contact model is then designed, and simulation results of a rigid body colliding with the ground are presented. A symplectic-momentum integrator for multibody systems is developed, and it is shown how to add actuator torques and contact forces to the discrete-time equations of motion. The multibody simulation algorithms produced in this dissertation: 1) integrate DAE systems 2) handle closed-loops

3) achieve linear-time computational cost for tree-structures 4) are symplectic-momentum mechanical integrators 5) have a simple contact model and 6) are implicit, second-order accurate, and 2-step methods.

5.1 External Forces

This section provides a method to include general external forces into the integration methods described in Chapter 2. Including general external forces is necessary to simulate systems with actuator torques and contact forces. External forces that are a function of positions and velocities in the linear space V are considered.

Let $F(v, \dot{v})$ be the continuous-time representation of the external force. The external force needs to be included in the discrete-time approximation of the system. This is accomplished by including the external forces in the same manner that the potential forces are included. In the integration equations given in Equation (2.28), the potential forces are given by

$$\frac{1}{2}\left[\frac{\partial L}{\partial v}\left(\frac{v_{k+1}+v_k}{2}, \frac{v_{k+1}-v_k}{h}\right)+\frac{\partial L}{\partial v}\left(\frac{v_k+v_{k-1}}{2}, \frac{v_k-v_{k-1}}{h}\right)\right]. \tag{5.1}$$

The discrete-time approximation of the general, external force, $F(v, \dot{v})$, is then chosen to be

$$\frac{1}{2}\left[F\left(\frac{v_{k+1}+v_k}{2}, \frac{v_{k+1}-v_k}{h}\right)+F\left(\frac{v_k+v_{k-1}}{2}, \frac{v_k-v_{k-1}}{h}\right)\right], \tag{5.2}$$

and the integration equations with external forces are then

$$\begin{aligned}
&\frac{1}{h}\left[\frac{\partial L}{\partial \dot{v}}\left(\frac{v_{k+1}+v_k}{2},\frac{v_{k+1}-v_k}{h}\right)-\frac{\partial L}{\partial \dot{v}}\left(\frac{v_k+v_{k-1}}{2},\frac{v_k-v_{k-1}}{h}\right)\right]- \\
&\frac{1}{2}\left[\frac{\partial L}{\partial v}\left(\frac{v_{k+1}+v_k}{2},\frac{v_{k+1}-v_k}{h}\right)+\frac{\partial L}{\partial v}\left(\frac{v_k+v_{k-1}}{2},\frac{v_k-v_{k-1}}{h}\right)\right] \\
&-D^T g\left(v_k\right)\lambda_k = \\
&\frac{1}{2}\left[F\left(\frac{v_{k+1}+v_k}{2},\frac{v_{k+1}-v_k}{h}\right)+F\left(\frac{v_k+v_{k-1}}{2},\frac{v_k-v_{k-1}}{h}\right)\right] \\
&g\left(v_{k+1}\right)=0.
\end{aligned} \tag{5.3}$$

By including the external forces in this way, a potential force will be recovered if the external force is really a potential force. Also, the overall discrete equations have second order local truncation error.

In applications, it is more convenient to multiply the top equation in Equation (5.3) by h and redefine λ_k to $h\lambda_k$. With this modification, the discrete equations with external forces become

$$\boxed{\begin{aligned}
&\left[\frac{\partial L}{\partial \dot{v}}\left(\frac{v_{k+1}+v_k}{2},\frac{v_{k+1}-v_k}{h}\right)-\frac{\partial L}{\partial \dot{v}}\left(\frac{v_k+v_{k-1}}{2},\frac{v_k-v_{k-1}}{h}\right)\right]- \\
&\frac{h}{2}\left[\frac{\partial L}{\partial v}\left(\frac{v_{k+1}+v_k}{2},\frac{v_{k+1}-v_k}{h}\right)+\frac{\partial L}{\partial v}\left(\frac{v_k+v_{k-1}}{2},\frac{v_k-v_{k-1}}{h}\right)\right] \\
&-D^T g\left(v_k\right)\lambda_k = \\
&\frac{h}{2}\left[F\left(\frac{v_{k+1}+v_k}{2},\frac{v_{k+1}-v_k}{h}\right)+F\left(\frac{v_k+v_{k-1}}{2},\frac{v_k-v_{k-1}}{h}\right)\right] \\
&g\left(v_{k+1}\right)=0.
\end{aligned}} \tag{5.4}$$

Equation (5.4) is an important equation in this chapter and is used throughout.

5.2 Simple Contact Model

In this section, a model of a point contacting, sliding, and sticking on a horizontal floor is presented. The point contact model is then used to model rigid bodies in contact with a horizontal floor. The motivation for this work is to provide a simple contact model for a shoe colliding, resting, and interacting with a horizontal floor.

The model is designed to be simple to allow for real-time simulation of multibody systems interacting with the environment. The contact model should model no contact, sliding friction, sticking, and switching between the modes. A complicated finite element model of a shoe interacting with a stiff floor is one option, but this requires too much computation.

The contact model is a state machine consisting of three steady states and three transition states. The state machine contact model is shown in Figure 5.1. The steady states (1, 2, and 4) can persist over integration steps while the transition states (3, 5, and 6) are only entered at the end of an integration step. The system then begins in state 1, 2, or 4 at the beginning of the next integration step. When there is more than one transition out of a state, the priority of the transitions are labeled a, b, and c with a having the highest priority.

Consider a frame attached to the top of the floor with the Z axis in a direction normal to the floor and pointing outward from the floor. Let p_n be the distance of the contact point to the horizontal floor in the direction of the z axis. If $p_n > 0$, then the point is not in

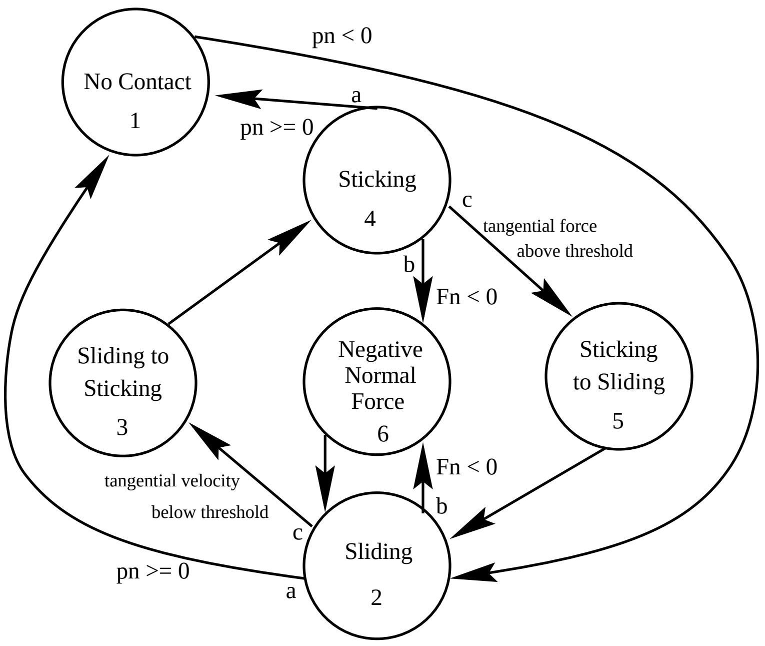

Figure 5.1: Point Contact Model

contact with the floor. Let $p_t \in \mathbb{R}^2$ be the tangential coordinates of the contact point. The force on the contact point in the positive Z direction is F_n, and the force in the tangential direction is F_t. Let k_n be a normal spring constant, b_n be a normal damping constant, k_t a tangential spring constant, and b_t a tangential damping constant. The static friction coefficient is μ_s, and μ_d is the dynamic friction coefficient. The zero velocity threshold is zvt and is a tangential velocity which indicates that the system is moving from sliding to sticking. An anchor point is $a_t \in \mathbb{R}^2$ and is specified in the tangential coordinates.

State 1 is the simplest state and is the no contact state. The contact point is in state 1 if $p_n \geq 0$. In this state, F_n and F_t are zero.

If the vertical distance drops below zero, then the friction state of the contact point transitions from state 1 to state 2, the sliding state. In this mode, the vertical force is $F_n = -(k_n p_n + b_n \dot{p}_n)$. If $F_n < 0$, then the state is sent to transition state 6, the negative normal force state. If $F_n \geq 0$ and $\dot{p}_t \leq zvt$, then the tangential force is $F_t = -\mu_d F_n \dot{p}_t / zvt$, and the friction state moves to state 3, the sliding to sticking state. If $F_n \geq 0$ and $\dot{p}_t > zvt$, then $F_t = -\mu_d F_n \dot{p}_t / \parallel \dot{p}_t \parallel$, and the system remains in the sliding state.

In the negative normal force state, state 6, F_n and F_t are set to zero. The system transitions to state 2, the sliding state, before the start of the next integration step.

In the sliding to sticking state, state 3, an anchor point, a_t, is set at the current configuration of the contact point. The system moves into the sticking state for the next integration step. In state 3, $F_t = -\mu_d F_n \dot{p}_t / zvt$.

In state 4, the sticking state, F_n is calculated as in state 2. If $F_n < 0$, then the system moves to state 6, the negative normal force state. If $F_n \geq 0$, then $F_t = -(k_t(p_t - a_t) + b_t \dot{p}_t)$. This equation models the material of the floor sticking to the contact point. If $F_t > \mu_s F_n$, then the system moves to transition state 5, the sticking to sliding state.

In the sticking to sliding state, the tangential force is clipped so that $F_t = \mu_s F_n F_t / \parallel F_t \parallel$. The anchor point is also released, and the system transitions to the sliding state before the beginning of the next integration step.

The parameters in the contact model are the normal spring constant k_n, the normal damping constant b_n, the tangential spring constant k_t, the tangential damping constant b_t, the dynamic friction coefficient μ_d, the static friction coefficient μ_s, and the zero velocity threshold zvt. The constants should be chosen based on material properties of the interacting bodies.

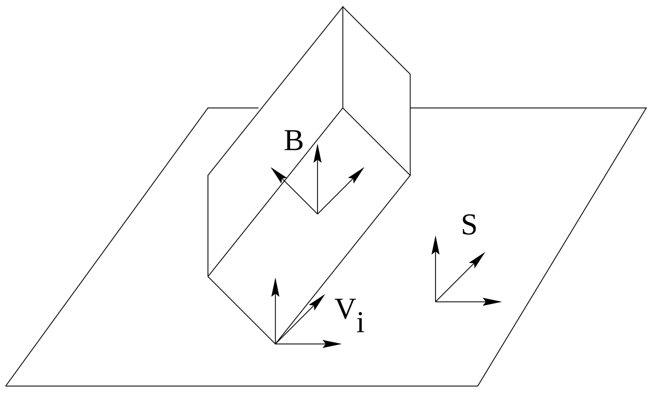

Figure 5.2: Rigid Body in Contact with the Ground

5.3 Rigid Body Motion with Contact

In this section, the contact model is combined with a rigid body integration method derived from Equation (5.4) to create a model of a rigid body colliding with the floor.

Three sets of frames shown in Figure 5.2 are important in the derivation of the discrete-time equations of motion: the body frame B, the spatial frame S fixed to the floor, and the vertex frames V_i attached to each contact point. The surface of the rigid body is divided into $n < \infty$ contact points. Each vertex frame has the same orientation as the spatial frame, but the origin of the frame is the contact point. The body frame is attached to the center of mass of the rigid body and aligned with the principal axes of inertia.

Discrete-time approximations to the continuous-time equations of motion for a rigid body in contact with the ground are created using Equation (5.4). The configuration of the rigid body is the position of the center of mass, $x_c \in \mathbb{R}^3$, with respect to the spatial frame, and the orientation of the body frame, $R_{s,b} \in SO(3)$, with respect to the spatial frame. The orientation is represented by a unit quaternion, x_q. The linear space (V given in Chapter 2) is $\mathbb{R}^3 \times \mathbb{R}^4 = \mathbb{R}^7$. $X \triangleq [x_c^T x_q^T]^T$ is used to represent a point in the linear space. The only constraint is $g(X) = x_q^T x_q - 1$ and specifies that x_q is a unit quaternion. The Lagrangian is

$$L(X, \dot{X}) = \frac{1}{2}(2\overline{x}_q \circ \dot{x}_q)^T \begin{bmatrix} 0 & 0 \\ 0 & \mathbb{I} \end{bmatrix} (2\overline{x}_q \circ \dot{x}_q) + \frac{1}{2} m \parallel \dot{x}_c \parallel^2 - mgx_{c_z}, \tag{5.5}$$

where m is the mass of the rigid body, $\mathbb{I}$ is the diagonal inertia matrix, $\overline{x}_q$ denotes the conjugate of x_q, g is the gravitational acceleration ($9.81 \frac{\text{m}}{\text{s}^2}$), and x_{c_z} is the Z coordinate of the center of mass.

The external forces, $F(X, \dot{X})$, are the accumulated contact forces. Each contact point has its own separate state machine for the contact model. The contact force at the contact point is a function of the position and velocity of the contact point, and the current state in the friction state machine. Let $p_{s,v_i}(X)$ be the position of the contact point with respect to the spatial frame . Let $\dot{p}_{s,v_i}(X, \dot{X})$ be the velocity of the contact point with respect to the spatial frame. The position of the contact point is only a function of the rigid body configuration, while the velocity of the contact point is a function of the rigid body configuration and the derivative of the configuration. The wrench written in frame V_i and due to the contact forces is denoted $F_{v_i}^c(p_{s,v_i}(X), \dot{p}_{s,v_i}(X, \dot{X}))$ and is a function of the rigid body configuration, X, and velocity, $\dot{X}$. The wrenches represented at the contact points then need to be transformed to the body frame. The body wrench due to the contact at vertex i is denoted $F_{b,v_i}^c(X, \dot{X}) = \text{Ad}_{g_{b,v_i}^{-1}(X)}^T F_{v_i}^c(p_{s,v_i}(X), \dot{p}_{s,v_i}(X, \dot{X}))$. The total wrench acting at the body due to the contact at each contact point is $F_b^{ct}(X, \dot{X}) = \sum_{i=1}^n \text{Ad}_{g_{b,v_i}^{-1}(X)}^T F_{v_i}^c(p_{s,v_i}(X), \dot{p}_{s,v_i}(X, \dot{X}))$. The body torque needs to be converted to a "force" in the space of unit quaternions. The body torque is converted to a quaternion force, F_q, by using Equation (A.1) and a conservation of work argument. Let q be a unit quaternion representing the orientation of a rigid body. Let ω^b be the body angular velocity, and let τ^b be the body torque. First note that

$$\omega^b = W^b(q)\dot{q}. \tag{5.6}$$

By conservation of work,

$$\begin{aligned} \dot{q}^T F_q &= (\omega^b)^T \tau^b \\ &= \dot{q}^T W^b(q)^T \tau^b \end{aligned} \tag{5.7}$$

implying that

$$F_q = W^b(q)^T \tau^b. \tag{5.8}$$

Let

$$K(X) = \begin{bmatrix} I & 0 \\ 0 & W^b(x_q)^T \end{bmatrix}. \tag{5.9}$$

The total external force, $F(X, \dot{X})$, due to the contact of the n contact points with the ground and written in terms of the position of the center of mass and the quaternion coordinate is then

$$F(X, \dot{X}) = K(X) \sum_{i=1}^{n} \mathrm{Ad}^T_{g^{-1}_{b,v_i}(X)} F^c_{v_i}(p_{s,v_i}(X), \dot{p}_{s,v_i}(X, \dot{X})). \tag{5.10}$$

The discrete equations in Equation (5.4) are applied to the Lagrangian system with holonomic constraints and external forces to produce the discrete-time equations of motion. The nonlinear equations in Equation (5.4) are solved to advance the system each time step. A Newton-Raphson step is used and a Jacobian is formed. The contribution of the Jacobian from the contact points are calculated based on the current iteration state in the contact model.

A box with 8 contact points at the vertices is simulated with this contact model. The parameters of the system are $m = 70$ Kg, g $= 9.81\frac{\mathrm{m}}{\mathrm{s}^2}$, $k_n = k_t = 75000$ N/m, $b_n = b_t = 2000$ N-s/m, and $zvt = 1$ mm/s. The parameters are chosen to loosely model a rubber floor. The time step is fixed and is 0.001s. The size of the box is $30 \times 40 \times 20$ cm in the X, Y, and Z directions. The contact points are labeled 1 through 8, and the coordinates of these vertices relative to the body frame are $(0.15, -0.2, -0.1)$ m, $(0.15, 0.2, -0.1)$ m, $(-0.15, 0.2, -0.1)$ m, $(-0.15, -0.2, -0.1)$ m, $(0.15, -0.2, 0.1)$ m, $(0.15, 0.2, 0.1)$ m, $(-0.15, 0.2, 0.1)$ m, and $(-0.15, -0.2, 0.1)$ m.

The initial position of the center of mass is $(0.0, 0.0, 0.3)$ m, and the initial velocity of the center of mass is $(5.0, 0.0, -0.04)$ m/s. The initial orientation in quaternions is $(0.9239, 0.2706, 0.2706, 0.0)$, and the initial angular velocity is zero. The simulation method was prototyped in Matlab and simulated on a Sun Ultra 2 - 200 MHz. The total simulation time for a 1.5 second simulation was 411 seconds (6 minutes 51 seconds).

The position of the center of mass over time is shown in Figure 5.3. The velocity in the X direction is constant until the first vertex strikes the ground at 0.102 seconds. The system settles to rest at approximately 1.3 seconds. The position in the Y axis stays constant until the collision and increases until settling at approximately 1.3 seconds. The position in the Z axis rises and falls until settling. There is a penetration of the box into the floor as the equilibrium is reached.

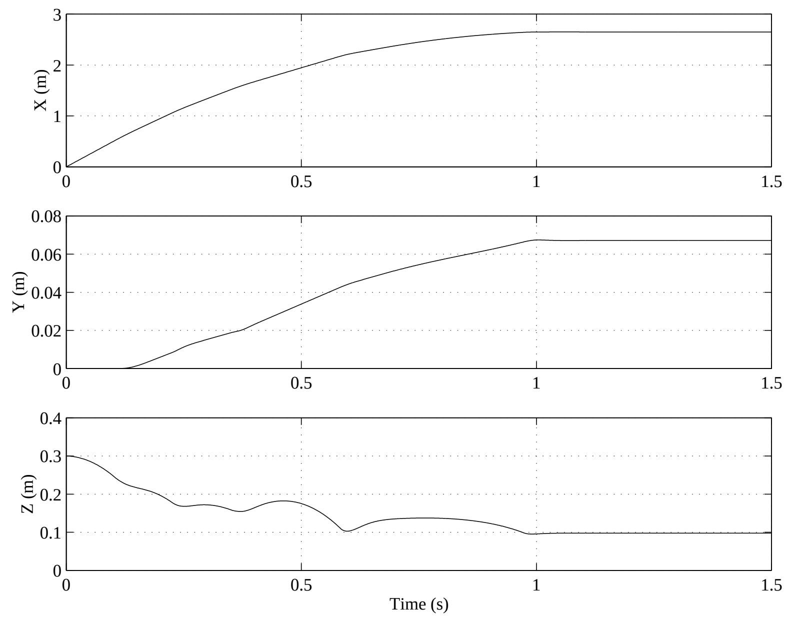

Figure 5.3: Center of Mass for the Contact Simulation

The orientation represented by the unit quaternion is shown in Figure 5.4. The orientation changes as the box interacts with the ground. After the motion reaches an equilibrium, the top of the box flips to the bottom through an approximate rotation about the Y axis. There is also a slight rotation about the X axis.

The friction states in the contact model for the 8 contact points are shown in Figure 5.5. The first vertex contacts the ground at 0.102 seconds and moves into the sliding state (state 2). Vertex 1 then moves between the negative normal force state (state 6) before the vertex leaves the surface. Vertex 1 finally moves into the no contact state (state 1). Vertex 2 contacts the floor and slides, but then finally moves into the no contact state. Vertices 3 and 4 never contact the floor. Vertices 5 through 8 move between the sliding state, the negative normal force state, the sliding to sticking transition state (state 3), and

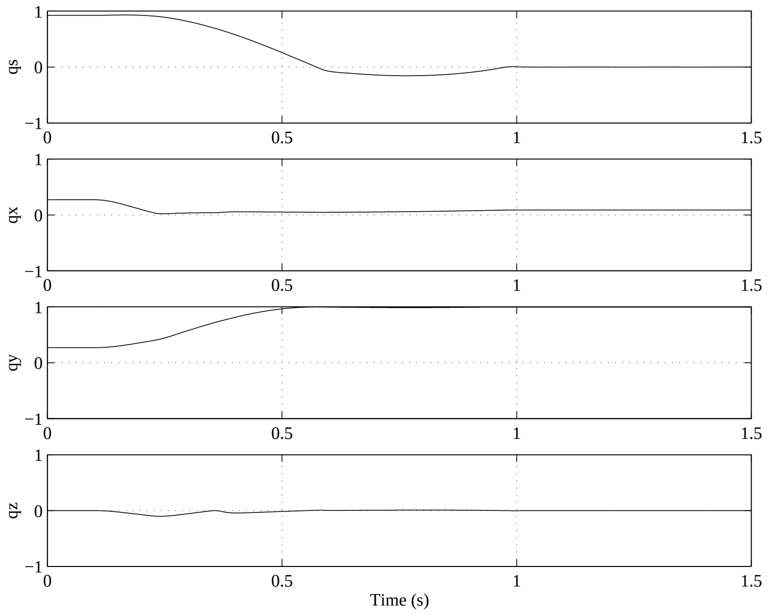

Figure 5.4: Quaternion Orientation for the Contact Simulation

the sticking state (state 4). These vertices finally settle into the sticking state.

The implicit discrete equations are solved at each time step by using Newton-Raphson steps. The number of iterations to reach a solution (determined by a tolerance) for each time step is shown in Figure 5.6. The maximum number of iterations is 5, and this occurs during the initial sliding of vertex 1. The number of iterations after the system settles to steady state is 1.

5.4 Multibody Integrator

A method of integrating multibody systems is created in this section based on the work in Chapter 2 and Equation (5.4). The multibody system is composed of many rigid bodies linked together with joints, and the joints are realized through joint constraints.

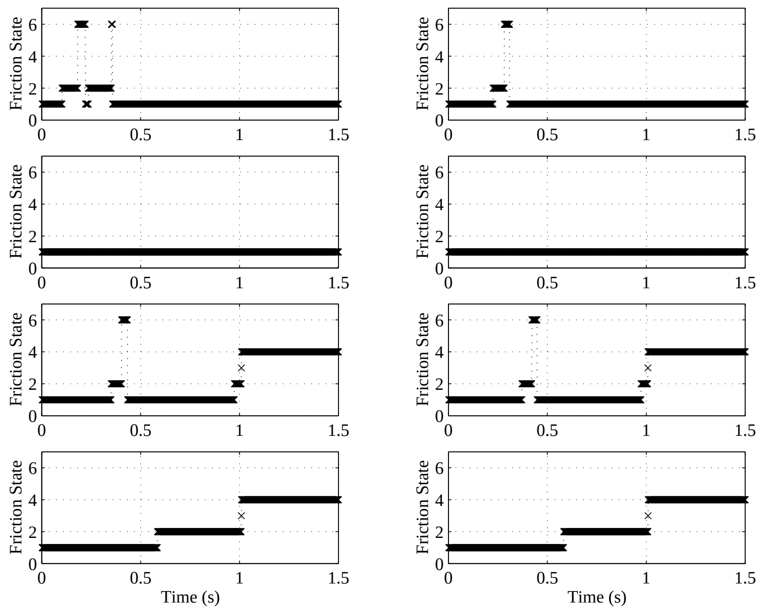

Figure 5.5: Friction States for the Contact Simulation

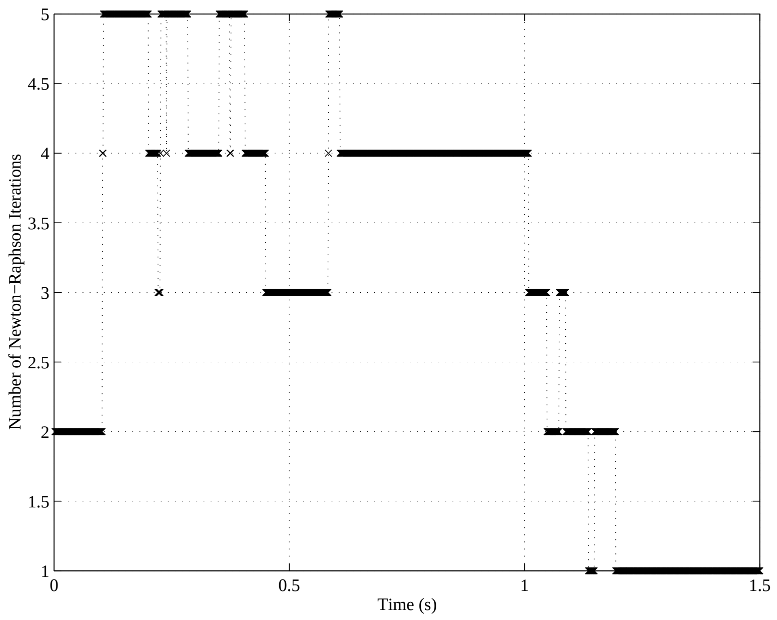

Figure 5.6: Number of Newton-Raphson Iterations for the Contact Simulation

The mechanical integration method presented in Chapter 2 is applied to the rigid body Lagrangians, unit quaternion constraints, and joint constraints to produce a mechanical integrator for multibody systems.

Multibody systems that are tree-structured (see Section 3.1) are considered in this chapter, although the mechanical integration method can be applied to multibody systems with closed-loops. Also, the discrete-time representation of the external forces are assumed to belong to the special class of external forces, $\overline{\mathrm{ext}}_{k.o}^{m}$. These external forces are only a function of the states of body $k.o$ and do not depend on the states of the surrounding bodies in the multibody system. Contact forces with the ground as well as rotary joint actuator torques from discrete-time controllers belong to this class. The tree-structure and external force assumptions are used in the sparse solving algorithm presented below.

5.4.1 Notation

The notation used in this section is given here. The notation is complicated and one needs to keep track of joint indices, rigid body indices, multibody connectivity, and discrete-time steps.

- k, j: typical labels for joint indices
- n: total number of joints in the system
- m: typical index for the discrete-time step
- $k.o$: link or body index for the body outboard to joint k
- $k.i$: link or body index for the body inboard to joint k

- $\mathcal{O}_{k.o}$: set of joints outboard to link $k.o$
- $X_{k.o} \in \mathbb{R}^7$: rigid body state for body $k.o$ consisting of the center of mass position and a unit quaternion
- X: state of the multibody system consisting of the states of every rigid body in the system
- $X^m_{k.o}$: rigid body state of link $k.o$ at discrete-time step m
- X^m: multibody state at discrete-time step m
- $\gamma^m_{k.o}$: Lagrange multiplier to enforce the unit quaternion constraint for link $k.o$ at discrete-time m
- λ^m_k: Lagrange multipliers to enforce the constraint for joint k at discrete-time m
- $b_{k.o}(X_{k.o})$: unit quaternion constraint function for body $k.o$
- $b^m_{k.o}$: equals $b_{k.o}(X^m_{k.o})$
- $g_k(X_{k.o}, X_{k.i})$: joint constraint function for joint k
- g^m_k: equals $g_k(X^m_{k.o}, X^m_{k.i})$
- $B^m_{k.o}$: equals $D_{X^m_{k.o}} b_k(X^m_{k.o})$, the derivative of the unit quaternion constraint with respect to $X^m_{k.o}$.
- $G^m_{k,o}$: equals $D_{X^m_{k.o}} g_k(X^m_{k.o}, X^m_{k.i})$
- $G^m_{k,i}$: equals $D_{X^m_{k.i}} g_k(X^m_{k.o}, X^m_{k.i})$

- $L_{k.o}(X_{k.o}, \dot{X}_{k.o})$: continuous-time rigid body Lagrangian for body $k.o$ given in Equation (5.5)

- $\mathrm{free}_{k.o}^{m+1}$: equals

$$\left[\frac{\partial L_{k.o}}{\partial \dot{X}_{k.o}}\left(\frac{X_{k.o}^{m+1}+X_{k.o}^{m}}{2}, \frac{X_{k.o}^{m+1}-X_{k.o}^{m}}{h}\right) - \frac{\partial L_{k.o}}{\partial \dot{X}_{k.o}}\left(\frac{X_{k.o}^{m}+X_{k.o}^{m-1}}{2}, \frac{X_{k.o}^{m}-X_{k.o}^{m-1}}{h}\right)\right] - \frac{h}{2}\left[\frac{\partial L_{k.o}}{\partial X_{k.o}}\left(\frac{X_{k.o}^{m+1}+X_{k.o}^{m}}{2}, \frac{X_{k.o}^{m+1}-X_{k.o}^{m}}{h}\right) + \frac{\partial L_{k.o}}{\partial X_{k.o}}\left(\frac{X_{k.o}^{m}+X_{k.o}^{m-1}}{2}, \frac{X_{k.o}^{m}-X_{k.o}^{m-1}}{h}\right)\right]$$

- $F_{k.o}(X, \dot{X})$: the continuous-time external wrenches acting on link $k.o$ and appropriately transformed for the position-quaternion state variables of link $k.o$

- $\mathrm{ext}_{k.o}^{m+1}$: discrete-time external forces equaling

$$\frac{h}{2}\left[F_{k.o}\left(\frac{X^{m+1}+X^{m}}{2}, \frac{X^{m+1}-X^{m}}{h}\right) + F_{k.o}\left(\frac{X^{m}+X^{m-1}}{2}, \frac{X^{m}-X^{m-1}}{h}\right)\right]$$

- $\overline{\mathrm{ext}}_{k.o}^{m+1}$: special class of discrete-time external forces where $F_{k.o}$ is *only* a function of $X_{k.o}$ and $\dot{X}_{k.o}$ and not a function of the other rigid body states in the multibody system. $\overline{\mathrm{ext}}_{k.o}^{m+1}$ equals

$$\frac{h}{2}\left[F_{k.o}\left(\frac{X_{k.o}^{m+1}+X_{k.o}^{m}}{2}, \frac{X_{k.o}^{m+1}-X_{k.o}^{m}}{h}\right) + F_{k.o}\left(\frac{X_{k.o}^{m}+X_{k.o}^{m-1}}{2}, \frac{X_{k.o}^{m}-X_{k.o}^{m-1}}{h}\right)\right]$$

- $M_{k.o}^{m+1}$: equals $D_{X_{k.o}^{m+1}}\mathrm{free}_{k.o}^{m+1} - D_{X_{k.o}^{m+1}}\overline{\mathrm{ext}}_{k.o}^{m+1}$

5.4.2 Discrete-Time Equations

The discrete-time equations for tree-structured multibody are given in this section. The discrete-time equations are solved to advance the system forward in time. The rigid body Lagrangians, external forces, unit quaternion constraints, and joint constraints are combined as given in Equation (5.4) to create the discrete-time equations of motion. The external forces acting on body $k.o$ are assumed to only be a function of the states of body $k.o$. More complicated external forces destroy the sparsity in the Jacobian matrix used in solving the discrete-time equations of motion.

The individual equations are organized with the joint indices as is done in Chapter 3. The multibody equations for joint k are given below in Equation (5.11).

$$\begin{aligned} f_{k.o}^{m+1} \triangleq \text{free}_{k.o}^{m+1} - B_{k.o}^{m\ T} \gamma_{k.o}^{m+1} - G_{k,o}^{m\ T} \lambda_k^{m+1} - \sum_{j \in \mathcal{O}_{k.o}} G_{j,i}^{m\ T} \lambda_j^{m+1} - \overline{\text{ext}}_{k.o}^{m+1} &= 0 \\ b_{k.o}^{m+1} &= 0 \\ g_k^{m+1} &= 0. \end{aligned} \tag{5.11}$$

The $\text{free}_{k.o}^{m+1}$ term includes the contribution from the free rigid body Lagrangian. The unit quaternion constraint for body $k.o$ is $b_{k.o}^{m+1} = b_{k.o}(X_{k.o}^{m+1})$, and the unit quaternion constraint force is $B_{k.o}^{m\ T} \gamma_{k.o}^{m+1}$. The constraint for joint k is $g_k^{m+1} = g_k\left(X_{k.o}^{m+1}, X_{k.i}^{m+1}\right)$, and the corresponding constraint force is $G_{k,o}^{m\ T} \lambda_k^{m+1}$. The constraint forces from the joints outboard to body $k.o$ are given in $\sum_{j \in \mathcal{O}_{k.o}} G_{k,i}^{m\ T} \lambda_j^{m+1}$. Remember that the joints outboard to body $k.o$ are given in $\mathcal{O}_{k.o}$. The discrete-time representation of the external forces acting on body $k.o$ that are only a function of the states for body $k.o$ are given in $\overline{\text{ext}}_{k.o}^{m+1}$. There is one set of equations for each joint in the multibody system, and the entire system needs to be solved together to advance the system forward in time. The solution is found by using Newton-Raphson steps and by exploiting sparsity in the Jacobian matrix.

5.4.3 Jacobian Structure

Newton-Raphson steps are used to solve the full set of nonlinear equations given in Equation (5.11). To solve a nonlinear function, $f(x) = 0$, $x \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}^n$, using Newton-Raphson steps, the system

$$Df(x^i)\triangle x^i = -f(x^i) \tag{5.12}$$

is solved for $\triangle x^i$. The Jacobian of $f(x)$ is $Df(x)$. The $\triangle x^i$ term is then used to update $x^{i+1} = x^i + \triangle x^i$. The Newton-Raphson step is calculated again until the steps converge to the zero of f.

The Jacobian equations for the Newton-Raphson steps for the multibody system equations given in Equation (5.11) are presented in this section. The equations need to be solved for $X_{k.o}^{m+1}$, $\gamma_{k.o}^{m+1}$, and $\lambda_{k.o}^{m+1}$ for all $k \in \{1, \cdots, n\}$. The Jacobian system in Equation (5.12) is then formed and solved for $\triangle X_{k.o}^{m+1}$, $\triangle\gamma_{k.o}^{m+1}$, and $\triangle\lambda_{k.o}^{m+1}$ for all $k \in \{1, \cdots, n\}$. The Jacobian system is created by taking the derivative of Equation (5.11) with respect to $X_{k.o}^{m+1}$, $\gamma_{k.o}^{m+1}$, and $\lambda_{k.o}^{m+1}$ for all $k \in \{1, \cdots, n\}$. The Jacobian system is then

$$\begin{aligned} M_{k.o}^{m+1}\triangle X_{k.o}^{m+1} - {B_{k.o}^{m}}^{T}\triangle\gamma_{k.o}^{m+1} - {G_{k,o}^{m}}^{T}\triangle\lambda_{k}^{m+1} - \sum_{j\in\mathcal{O}_{k.o}} {G_{j,i}^{m}}^{T}\triangle\lambda_{j}^{m+1} &= -f_{k.o}^{m+1} \\ B_{k.o}^{m+1}\triangle X_{k.o}^{m+1} &= -b_{k.o}^{m+1} \\ G_{k,o}^{m+1}\triangle X_{k.o}^{m+1} &= -g_{k}^{m+1} - G_{k,i}^{m+1}\triangle X_{k.i}^{m+1} \end{aligned} \tag{5.13}$$

for all $k \in \{1, \cdots, n\}$.

5.4.4 Sparse Solving Algorithm

An algorithm to calculate $\triangle X_{k.o}^{m+1}$, $\triangle\gamma_{k.o}^{m+1}$, and $\triangle\lambda_{k.o}^{m+1}$ for all $k \in \{1, \cdots, n\}$ in computational time that grows linearly with the number of joints is presented in this section. The algorithm relies on the tree-structure of the multibody system as well as the restriction that the external forces acting on body $k.o$ only depend on the state variables for body $k.o$. The algorithm shares similarities to the AB inertia and bias force algorithm presented in Chapter 3. The algorithm first builds terms resembling the AB inertias and bias forces in an inboard recursion. The results of the inboard recursion are then used to calculate the desired values in an outboard recursion.

The sparse algorithm shares similarities to the sparse algorithms presented in (Lubich et al., 1992) for differential algebraic equations of multibody systems. The derivations in this section are complicated by the unit quaternion constraint.

It will be shown that the equations in Equation (5.13) can be manipulated into the following form:

$$\begin{aligned} \widehat{M}_{k.o}^{m+1}\triangle X_{k.o}^{m+1} - B_{k.o}^{m\ T}\triangle\gamma_{k.o}^{m+1} - G_{k,o}^{m\ T}\triangle\lambda_{k}^{m+1} &= -\widehat{f}_{k.o}^{m+1} \\ B_{k.o}^{m+1}\triangle X_{k.o}^{m+1} &= -b_{k.o}^{m+1} \\ G_{k,o}^{m+1}\triangle X_{k.o}^{m+1} &= -g_{k}^{m+1} - G_{k,i}^{m+1}\triangle X_{k.i}^{m+1} \end{aligned} \tag{5.14}$$

if the equations for the outboard joints, $j \in \mathcal{O}_{k.o}$, can also be put in the form of this equation. Formulas for $\widehat{M}_{k.o}^{m+1}$ and $\widehat{f}_{k.o}^{m+1}$ will be developed. Note that the equations for a leaf in the multibody system, where there are no outboard joints, is in this form with $\widehat{M}_{k.o}^{m+1} = M_{k.o}^{m+1}$ and $\widehat{f}_{k.o}^{m+1} = f_{k.o}^{m+1}$.

First write Equation (5.14) in block matrix form and use j to index the outboard joints to get

$$\begin{bmatrix} \widehat{M}_{j.o}^{m+1} & -B_{j.o}^{m\,T} & -G_{j,o}^{m\,T} \\ B_{j.o}^{m+1} & 0 & 0 \\ G_{j,o}^{m+1} & 0 & 0 \end{bmatrix} \begin{bmatrix} \triangle X_{j.o}^{m+1} \\ \triangle \gamma_{j.o}^{m+1} \\ \triangle \lambda_{j}^{m+1} \end{bmatrix} = \begin{bmatrix} -\widehat{f}_{j.o}^{m+1} \\ -b_{j.o}^{m+1} \\ -g_{j}^{m+1} - G_{j,i}^{m+1} \triangle X_{j.i}^{m+1} \end{bmatrix}. \tag{5.15}$$

Let

$$\left[\begin{array}{c|c} A_j^{m+1} & C_j^{m+1} \\ \hline E_j^{m+1} & F_j^{m+1} \end{array}\right] = \left[\begin{array}{cc|c} \widehat{M}_{j.o}^{m+1} & -B_{j.o}^{m\,T} & -G_{j,o}^{m\,T} \\ B_{j.o}^{m+1} & 0 & 0 \\ \hline G_{j,o}^{m+1} & 0 & 0 \end{array}\right]^{-1}, \tag{5.16}$$

where A_j^{m+1} and F_j^{m+1} are square matrices, and the dimension of F_j^{m+1} is the same as the dimension of the 0 block in the lower right corner. This implies that

$$\left[\begin{array}{c} \triangle X_{j.o}^{m+1} \\ \triangle \gamma_{j.o}^{m+1} \\ \hline \triangle \lambda_{j}^{m+1} \end{array}\right] = \left[\begin{array}{c|c} A_j^{m+1} & C_j^{m+1} \\ \hline E_j^{m+1} & F_j^{m+1} \end{array}\right] \left[\begin{array}{c} -\widehat{f}_{j.o}^{m+1} \\ -b_{j.o}^{m+1} \\ \hline -g_{j}^{m+1} - G_{j,i}^{m+1} \triangle X_{j.i}^{m+1} \end{array}\right] \tag{5.17}$$

and that

$$\triangle \lambda_j^{m+1} = -E_j^{m+1} \begin{bmatrix} \widehat{f}_{j.o}^{m+1} \\ b_{j.o}^{m+1} \end{bmatrix} - F_j^{m+1} g_j^{m+1} - F_j^{m+1} G_{j,i}^{m+1} \triangle X_{j.i}^{m+1}. \tag{5.18}$$

Substitute Equation (5.18) into $-\sum_{j\in\mathcal{O}_{k.o}} G_{j,i}^{m\,T} \triangle \lambda_j^{m+1}$ and then later combine the result into Equation (5.13):

$$-\sum_{j\in\mathcal{O}_{k.o}} G_{j,i}^{m\,T} \triangle \lambda_j^{m+1} =$$

$$\sum_{j\in\mathcal{O}_{k.o}} G_{j,i}^{m\,T} \left[E_j^{m+1} \begin{bmatrix} \widehat{f}_{j.o}^{m+1} \\ b_{j.o}^{m+1} \end{bmatrix} + F_j^{m+1} g_j^{m+1} + F_j^{m+1} G_{j,i}^{m+1} \triangle X_{j.i}^{m+1} \right]. \tag{5.19}$$

Note that for $j \in \mathcal{O}_{k.o}$, $j.i = k.o$. Therefore,

$$-\sum_{j\in\mathcal{O}_{k.o}} {G_{j,i}^{m}}^T \triangle\lambda_j^{m+1} = \sum_{j\in\mathcal{O}_{k.o}} {G_{j,i}^{m}}^T \left[E_j^{m+1} \begin{bmatrix} \widehat{f}_{j.o}^{m+1} \\ b_{j.o}^{m+1} \end{bmatrix} + F_j^{m+1} g_j^{m+1} \right]$$
$$+ \left[\sum_{j\in\mathcal{O}_{k.o}} {G_{j,i}^{m}}^T F_j^{m+1} G_{j,i}^{m+1} \right] \triangle X_{k.o}^{m+1}. \tag{5.20}$$

Substitute the result back into the first equation in Equation (5.13) and rearrange to get

$$\left(M_{k.o}^{m+1} + \sum_{j\in\mathcal{O}_{k.o}} {G_{j,i}^{m}}^T F_j^{m+1} G_{j,i}^{m+1} \right) \triangle X_{k.o}^{m+1} - {B_{k.o}^{m}}^T \triangle\gamma_{k.o}^{m+1}$$
$$-{G_{k,o}^{m}}^T \triangle\lambda_k^{m+1} = -f_{k.o}^{m+1} - \sum_{j\in\mathcal{O}_{k.o}} {G_{j,i}^{m}}^T \left[E_j^{m+1} \begin{bmatrix} \widehat{f}_{j.o}^{m+1} \\ b_{j.o}^{m+1} \end{bmatrix} + F_j^{m+1} g_j^{m+1} \right]. \tag{5.21}$$

Choose $\widehat{M}_{k.o}^{m+1}$ to be

$$\widehat{M}_{k.o}^{m+1} = M_{k.o}^{m+1} + \sum_{j\in\mathcal{O}_{k.o}} {G_{j,i}^{m}}^T F_j^{m+1} G_{j,i}^{m+1}, \tag{5.22}$$

and $\widehat{f}_{k.o}^{m+1}$ to be

$$\widehat{f}_{k.o}^{m+1} = f_{k.o}^{m+1} + \sum_{j\in\mathcal{O}_{k.o}} {G_{j,i}^{m}}^T \left[E_j^{m+1} \begin{bmatrix} \widehat{f}_{j.o}^{m+1} \\ b_{j.o}^{m+1} \end{bmatrix} + F_j^{m+1} g_j^{m+1} \right]. \tag{5.23}$$

In this way, Equation (5.13) is transformed into the form of Equation (5.14).

For the root joint, joint 1, in the tree-structured multibody system, the joint constraint is only a function of the state of link $1.o$. Therefore, the equations for the root rigid body are

$$\begin{aligned} \widehat{M}_{1.o}^{m+1} \triangle X_{1.o}^{m+1} - {B_{1.o}^{m}}^T \triangle\gamma_{1.o}^{m+1} - {G_{1,o}^{m}}^T \triangle\lambda_1^{m+1} &= -\widehat{f}_{1.o}^{m+1} \\ B_{1.o}^{m+1} \triangle X_{1.o}^{m+1} &= -b_{1.o}^{m+1} \\ G_{1,o}^{m+1} \triangle X_{1.o}^{m+1} &= -g_1^{m+1}. \end{aligned} \tag{5.24}$$

These equations are then solved for the desired variables as is done in Equation (5.15) and by knowing that the joint constraint is only a function of the state variables for link $1.o$. The desired values $\triangle X_{k.o}^{m+1}$, $\triangle\gamma_{k.o}^{m+1}$, and $\triangle\lambda_{k.o}^{m+1}$ for all $k \in \{2, \cdots, n\}$ are then calculated in an outboard recursion. The complete algorithm is summarized in Algorithm 5.1.

5.5 Chapter Summary

Components of a multibody simulator were developed in this chapter to create a control development platform for human models in contact with the ground. The emphasis was placed on developing a real-time simulation environment for multibody systems with contact and discrete-time controllers. First, external forces were added to the methodology presented in Chapter 2. A simple contact model was then developed and demonstrated in a simulation of a rigid body in contact with the environment. The discrete-time equations of motion for tree-structured multibody systems were developed, and it was shown how to solve the equations in linear-time. For tree-structured multibody systems with no external forces, the mechanical integrator is a 2-step, implicit, linear-time, symplectic-momentum method.

This chapter presented a method to include external forces into the mechanical integrators of Chapter 2. The external forces are added in the same manner as the potential forces. By adding forces in this way, potential forces can be recovered, and the method still has 2nd order local truncation error. More research can be performed to either justify incorporating external forces in this way or by designing a new method of incorporating external forces. Also, methods can be designed to include special external forces, such as

Algorithm 5.1 Sparse Solving Algorithm for Tree-Structured Multibody Systems

```
given: $M_{k.o}^{m+1}$, $B_{k.o}^{m+1}$, $B_{k.o}^{m}$, $G_{k,o}^{m+1}$, $G_{k,o}^{m}$, $G_{k,i}^{m+1}$, $f_{k.o}^{m+1}$, $b_{k.o}^{m+1}$, $g_{k}^{m+1}$ for all $k \in \{1, ..., n\}$;
for $k = 1$ to $n$ do
    $\widehat{M}_{k.o}^{m+1} = M_{k.o}^{m+1}$;    $\widehat{f}_{k.o}^{m+1} = f_{k.o}^{m+1}$;
end for
for $k = n$ to $1$ do
    $\left[\begin{array}{c|c} A_k^{m+1} & C_k^{m+1} \\ \hline E_k^{m+1} & F_k^{m+1} \end{array}\right] = \left[\begin{array}{cc|c} \widehat{M}_{k.o}^{m+1} & -B_{k.o}^{m\,T} & -G_{k,o}^{m\,T} \\ B_{k.o}^{m+1} & 0 & 0 \\ \hline G_{k,o}^{m+1} & 0 & 0 \end{array}\right]^{-1}$;
    $\widehat{M}_{k.i}^{m+1} = \widehat{M}_{k.i}^{m+1} + G_{k,i}^{m\,T} F_k^{m+1} G_{k,i}^{m+1}$;
    $\widehat{f}_{k.i}^{m+1} = \widehat{f}_{k.i}^{m+1} + G_{k,i}^{m\,T} \left[ E_k^{m+1} \begin{bmatrix} \widehat{f}_{k.o}^{m+1} \\ b_{k.o}^{m+1} \end{bmatrix} + F_k^{m+1} g_k^{m+1} \right]$;
end for
$\left[\begin{array}{c} \triangle X_{1.o}^{m+1} \\ \triangle \gamma_{1.o}^{m+1} \\ \hline \triangle \lambda_1^{m+1} \end{array}\right] = \left[\begin{array}{c|c} A_1^{m+1} & C_1^{m+1} \\ \hline E_1^{m+1} & F_1^{m+1} \end{array}\right] \left[\begin{array}{c} -\widehat{f}_{1.o}^{m+1} \\ -b_{1.o}^{m+1} \\ \hline -g_1^{m+1} \end{array}\right]$;
for $k = 2$ to $n$ do
    $\left[\begin{array}{c} \triangle X_{k.o}^{m+1} \\ \triangle \gamma_{k.o}^{m+1} \\ \hline \triangle \lambda_k^{m+1} \end{array}\right] = \left[\begin{array}{c|c} A_k^{m+1} & C_k^{m+1} \\ \hline E_k^{m+1} & F_k^{m+1} \end{array}\right] \left[\begin{array}{c} -\widehat{f}_{k.o}^{m+1} \\ -b_{k.o}^{m+1} \\ \hline -g_k^{m+1} - G_{k,i}^{m+1} \triangle X_{k.i}^{m+1} \end{array}\right]$;
end for
```

dissipation.

A simple contact model was also presented based on spring damper models with a coordinating state machine. The contact model captures sliding, stiction, no contact, and the transitions between states. There are areas where improvements can be made to the contact simulation. First, the exact times of contact state changes are not found in the current implementation. The focus was on real-time simulation and fixed time steps were used. More accurate solutions may be able to be obtained by finding the switching times and restarting the integration. Second, a simple Newton-Raphson iteration was used. For some simulations, the iteration can oscillate about a solution and not converge. These situations may be corrected by using Newton-Raphson iterations with a line search.

The integration method is a 2-step, implicit method. The implicit equations allow the use of larger time steps than an explicit integrator, but nonlinear equations need to be solved at each time step. Also, it is not known the number of iterations required for convergence. In some applications, it may be desirable to use a single-step, explicit, mechanical integration method. The symplectic-momentum multibody integrator in (Reich, 1996b) may be useful for these applications.

For tree-structured multibody systems, a symplectic-momentum integrator was created. For a special class of external forces, the nonlinear equations of motion can be solved in linear-time by exploiting sparsity in a Jacobian matrix. The special class of external forces includes actuator torques from rotary joints and contact of a rigid body with the environment. By including the contact model into the external forces of the multibody simulator, a control development platform can be created to model human motion.

Chapter 6

Conclusion

This dissertation created new algorithms for controlling and simulating multibody systems. These algorithms were designed to aid in the creation and design of complex, controlled mechanical systems. The focus was on exploiting the structure of multibody systems to produce useful algorithms.

A method to construct mechanical integrators was presented based on a discrete variational principle. The method produces symplectic-momentum integrators for Lagrangian systems with holonomic constraints by discretizing the principles of mechanics rather than the equations of motion. The method is easy to apply, and the resulting discrete equations of motion share many similarities to the continuous-time equations of motion. The symplectic form is invariant, momentum is conserved, and energy is seen to oscillate about a constant value in the numerical experiments. The method can be applied in a differential-algebraic-equation (DAE) form or an equivalent generalized coordinate form. The method was applied to the double spherical pendulum and the free rigid body.

A 3D balancing controller was created for a multibody model of a human biped. The system has eighteen degrees of freedom, a central body, and two legs. A model-based workspace controller was designed. The model was created by using efficient recursive multibody algorithms. The controller design coordinated the degrees of freedom of the two legs and responds to disturbances. The gains are easy to tune, and the desired trajectories are easy to choose for a desired motion. The controller was designed to serve as a basis for more complicated controllers for walking, running, jumping, changing direction, and adapting to loads.

Recursive algorithms are important in the study of multibody systems. Recursive forward kinematics, inverse dynamics, and forward dynamics algorithms for tree-structured multibody systems were derived in terms of the Lie group notation used in (Murray et al., 1994). These algorithms and the ideas in the algorithms were used in the development of the 3D balancing controller.

This dissertation presented components of a multibody simulator specifically designed for developing control algorithms for human motion and for simulating human biped models in contact with the ground. External forces were added to the mechanical integration work presented earlier in the dissertation. A simple contact model was developed and simulation results of a rigid body colliding with the ground were presented. The discrete-time equations of motion for tree-structured multibody systems were developed, and it was shown how to solve the equations in linear-time. For tree-structure multibody systems with no external forces, the integration technique is a 2-step, implicit, linear-time, symplectic-momentum method.

6.1 Future Work

There are many areas for future research and development in the areas of multibody control, mechanical integration, and multibody simulation. More research is especially needed in creating control algorithms for complicated multibody systems interacting with the environment.

Multibody Control. The 3D balancing controller presented in Chapter 4 addresses some of the issues in the control of multibody systems, but there are more areas for further study. New algorithms need to be designed to handle redundancy and singularities in the legs. Controllers for walking, running, jumping, changing direction, and adapting to loads need to be created and designed. These algorithms must be able to handle the discrete-time model changes and the complicated continuous-time dynamics. The controller presented in Chapter 4 was designed for tree-structured multibody models. Algorithms need to be created for systems with closed-loops.

Mechanical Integration. The method presented in Chapter 2 produces symplectic-momentum integrators for Lagrangian systems with holonomic constraints. These methods and other mechanical integration methods can be improved, especially for engineering applications. A method was provided in Chapter 5 to add external forces to the methodology presented in Chapter 2, but more study needs to be done in developing mechanical integrators with external forces. Discrete models for special classes of external forces, such as dissipative forces, can be created. The order of accuracy of the mechanical integrators presented in this dissertation can be improved. The methods have second order accuracy, and the accuracy may be increased by perhaps adding additional points to the discrete variational principle to create multi-step methods or more integration stages. Adaptive and variable step size are useful features of numerical integrators, and it would be beneficial to have mechanical integrators with these properties. The method produced in Chapter 2 is a 2-step, implicit integration method. Single-step methods are easier to implement and explicit methods are better suited for some applications. Creating integration methods with these properties would be beneficial. The methods introduced in this dissertation produce symplectic-momentum integrators, and it would be interesting to find a discrete mechanics principle to create energy-momentum integrators.

Multibody Simulation. Many methods have been created to simulate multibody systems, but there are still areas for future work. One area is in creating mechanical integrators for these systems as discussed above. These algorithms may turn out to be more efficient than current techniques and can be used in applications where the underlying structure is important. Efficiently modeling contact still needs improvement. The LCP and impulse approaches available for hard contact problems are complementary (see (Mirtich, 1996)) and combining these approaches seems to be a promising area for future research. The creators of *Impulse* are currently exploring this area of work. The contact model presented in this dissertation was developed as a soft contact model of the feet interacting with the ground. Future work can be performed in better integrating the contact model with the discrete-time equations of motion. Also, using Newton-Raphson iteration with line searching should improve the solution of the nonlinear equations and may reduce the number of iterations required. An interesting area for future study is in combining DAE integration methods with impulses and contact models. DAE methods seem more natural for modeling systems with closed-loops.

Appendix A

Robotics Background

A brief background in rigid body motion, quaternions, and robotics is given in this appendix. The notation used in this dissertation for rigid body motion is also presented. Consult (Murray et al., 1994) for a more detailed treatment of many of the topics.

A.1 Rotation Matrices

Rotation matrices represent the orientation of one frame with respect to another. A rotation matrix, R, is an element of the Lie group $SO(3)$, where

$$SO(3) = \left\{ R \in \mathbb{R}^{3\times 3} : RR^T = I, \det R = +1 \right\}.$$

Consider a frame K and a frame L. The rotation matrix, $R_{l,k}$, maps vectors represented in frame K to the corresponding vectors represented in frame L. The columns of $R_{l,k}$ are the representation of the coordinate axes of frame K with respect to frame L. Another interpretation of $R_{l,k}$ is that it rotates vectors in L to new vectors in L by considering K and L to be initially coincident and by considering $R_{l,k}$ to move K to the final configuration. The inverse of $R_{l,k}$ equals $R_{l,k}^T = R_{k,l}$.

A.2 Angular Velocities

Let frame K move over time with respect to frame L so that $R_{l,k}$ is a function of time. Let

$$\widehat{\omega}_{l,k}^b = R_{l,k}^T \dot{R}_{l,k} \quad \text{and} \quad \widehat{\omega}_{l,k}^s = \dot{R}_{l,k} R_{l,k}^T.$$

The body angular velocity matrix is $\widehat{\omega}_{l,k}^b$, and the spatial angular velocity matrix is $\widehat{\omega}_{l,k}^s$. Both are elements of $so(3)$, the $3-$dimensional vector space of 3×3 skew-symmetric matrices. Given a vector $\omega = (w_x, w_y, w_z)^T$, $\widehat{\omega}$ converts the vector to an element of $so(3)$ such that

$$\omega \times b = \widehat{\omega} b = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} b,$$

where b is an arbitrary vector in $\mathbb{R}^3$. The $\vee$ operator extracts the 3-vector from an element of $so(3)$ so that $\omega = (\widehat{\omega})^\vee$.

A.3 Quaternions

Unit quaternions are a 4 parameter, singularity free, double covering of $SO(3)$ and are useful in computations with rotation matrices. The quaternion consists of a scalar value, q_s, and a vector with three components which is denoted $q_v = (q_x, q_y, q_z)$. The following formula constructs a $SO(3)$ matrix, R, from its unit quaternion representation, q:

$$R = (2q_s^2 - 1)I + 2q_s\widehat{q}_v + 2q_v q_v^T.$$

A useful property of the unit quaternion representation is that if A, B, and $C \in SO(3)$ are represented by unit quaternions, a, b, and c, respectively, then $C = AB$ if and only if $c = \pm a \star b$ where $\star$ represents quaternion multiplication. If $c = a \star b$, then $c_s = a_s b_s - a_v \cdot b_v$ and $c_v = a_s b_v + b_s a_v + a_v \times b_v$. Also, the conjugate of a denoted $\bar{a}$ is given by $\bar{a} = (a_s, -a_v)$. For unit quaternions, $\bar{a}$ is the inverse of a, in that $a \star \bar{a} = (1, 0, 0, 0)$. If $w, v \in \mathbb{R}^3$, $A \in SO(3)$, $w = Av$, and a is a unit quaternion that represents A, then $(0, w) = a \star (0, v) \star \bar{a}$ where $(0, w)$ is a quaternion formed from the vector w.

Quaternionic multiplication has a matrix representation. Let $w = q \star r$, where w, q, and r are quaternions. In vector notation,

$$w = W_L(q)r = W_R(r)q,$$

where

$$W_L(q) \triangleq \begin{bmatrix} q_s & -q_v^T \\ q_v & q_s I + \widehat{q}_v \end{bmatrix} = \begin{bmatrix} q_s & -q_x & -q_y & -q_z \\ q_x & q_s & -q_z & q_y \\ q_y & q_z & q_s & -q_x \\ q_z & -q_y & q_x & q_s \end{bmatrix}$$

and

$$W_R(q) \triangleq \begin{bmatrix} q_s & -q_v^T \\ q_v & q_s I - \widehat{q}_v \end{bmatrix} = \begin{bmatrix} q_s & -q_x & -q_y & -q_z \\ q_x & q_s & q_z & -q_y \\ q_y & -q_z & q_s & q_x \\ q_z & q_y & -q_x & q_s \end{bmatrix}.$$

Let R be the rotation matrix representing the orientation of a frame. If q is the unit quaternion representing R, then $\bar{q} \star \dot{q} = (0, \omega^b/2)$, where ω^b is the body angular velocity. Also, $\dot{q} \star \bar{q} = (0, \omega^s/2)$, where ω^s is the spatial angular velocity. In matrix notation,

$$\begin{bmatrix} 0 \\ \frac{\omega^b}{2} \end{bmatrix} = W_L(\bar{q})\dot{q}$$

and

$$\begin{bmatrix} 0 \\ \frac{\omega^s}{2} \end{bmatrix} = W_R(\bar{q})\dot{q}.$$

Therefore,

$$\omega^b = 2 \begin{bmatrix} -q_x & q_s & q_z & -q_y \\ -q_y & -q_z & q_s & q_x \\ -q_z & q_y & -q_x & q_s \end{bmatrix} \dot{q} \overset{\triangle}{=} W^b(q)\dot{q} \tag{A.1}$$

and

$$\omega^s = 2 \begin{bmatrix} -q_x & q_s & -q_z & q_y \\ -q_y & q_z & q_s & -q_x \\ -q_z & -q_y & q_x & q_s \end{bmatrix} \dot{q} \overset{\triangle}{=} W^s(q)\dot{q}. \tag{A.2}$$

The fact that $\dot{q} = q \star (0, \omega^b/2) = (0, \omega^s/2) \star q$ implies that

$$\dot{q} = W_L(q) \begin{bmatrix} 0 \\ \frac{\omega^b}{2} \end{bmatrix} = W_R(q) \begin{bmatrix} 0 \\ \frac{\omega^s}{2} \end{bmatrix}. \tag{A.3}$$

Therefore,

$$\dot{q} = \frac{1}{2} \begin{bmatrix} -q_x & -q_y & -q_z \\ q_s & -q_z & q_y \\ q_z & q_s & -q_x \\ -q_y & q_x & q_s \end{bmatrix} \omega^b \overset{\triangle}{=} B^b(q)\omega^b \tag{A.4}$$

and

$$\dot{q} = \frac{1}{2} \begin{bmatrix} -q_x & -q_y & -q_z \\ q_s & q_z & -q_y \\ -q_z & q_s & q_x \\ q_y & -q_x & q_s \end{bmatrix} \omega^s \overset{\triangle}{=} B^s(q)\omega^s. \tag{A.5}$$

Given a rotation matrix, R, one can find a unit length axis, $w \in \mathbb{R}^3$, and an angle, $\theta \in [0, 2\pi]$ radians, such that $R = e^{\hat{w}\theta}$. Also, using axis $-w$ and angle $2\pi - \theta$ produces the same rotation matrix, *i.e.*, $R = e^{-\hat{w}(2\pi-\theta)} = e^{\hat{w}\theta}$. The identity matrix is represented by an arbitrary axis, and the angle 0 or 2π.

The unit quaternion representing a rotation matrix can be created from its axis and angle representation in the following way:

$$q = (q_s, q_v) = (\cos(\frac{\theta}{2}), \sin(\frac{\theta}{2})w)$$

Unit quaternions are a double covering of $SO(3)$ and q and $-q$ represent the same rotation matrix. If q is created from axis w and angle θ, then $-q$ is created from axis $-w$ and angle $2\pi - \theta$ radians. If the scalar component of a unit quaternion is non-negative, then the quaternion can be created from an axis-angle pair where the angle is in $[0, \pi]$ radians. Given a unit quaternion, $q = (q_s, q_v)$, an axis angle representation can be recovered. If $qs = 1$, then $\theta = 0$ radians and w is an arbitrary unit vector; else if $qs = -1$, then $\theta = 2\pi$ and again w is an arbitrary unit vector; else $\theta = 2\arccos(q_s) \in (0, 2\pi)$ radians and $w = q_v / \sin(\theta/2)$.

A.4 Rigid Body Transformations

Rigid body transformations describe the pose (position and orientation) of one frame with respect to another. Consider a frame K and a frame L. The pose of frame K with respect to frame L is given by the position of the origin of frame K with respect to frame L denoted $p_{l,k}$, and the orientation of frame K with respect to frame L denoted $R_{l,k}$. The pose is given by an element in the Lie group $SE(3)$, where

$$SE(3) = \left\{ (p, R) : p \in \mathbb{R}^3, R \in SO(3) \right\}.$$

The homogeneous representation is used to represent the pose of one frame with respect to another frame. A point is represented by a 4 vector with the location of the point given by the first three coordinates and a 1 placed in the 4th position. A vector is represented by a 4 vector with the vector coordinates in the first 3 positions and a 0 placed in the 4th position. An element, $g_{l,k} \in SE(3)$, is represented by a 4×4 matrix in the following form:

$$g_{l,k} = \begin{bmatrix} R_{l,k} & p_{l,k} \\ 0 & 1 \end{bmatrix}.$$

The $SE(3)$ element, $g_{l,k}$, transforms the coordinates of vectors and points represented in frame K to the corresponding coordinates in frame L. The inverse of an $SE(3)$ element is

$$g_{l,k}^{-1} = \begin{bmatrix} R_{l,k}^T & -R_{l,k}^T p_{l,k} \\ 0 & 1 \end{bmatrix} = g_{k,l}.$$

A.5 Twists

Let frame K move over time with respect to frame L so that $g_{l,k}$ is a function of time. Let

$$\widehat{V}^b_{l,k} = g^{-1}_{l,k}\dot{g}_{l,k} \quad \text{and} \quad \widehat{V}^s_{l,k} = \dot{g}_{l,k}g^{-1}_{l,k}.$$

The 4×4 body twist matrix is $\widehat{V}^b_{l,k}$, and the spatial twist matrix is $\widehat{V}^s_{l,k}$. The form of a twist matrix, $\widehat{\xi}$, is

$$\widehat{\xi} = \begin{bmatrix} \widehat{\omega} & v \\ 0 & 0 \end{bmatrix},$$

where $\widehat{\omega} \in so(3)$. The twist matrix is 6 dimensional and one can extract a 6 dimensional twist from the twist matrix. The twist, ξ, is given by

$$\xi = (\widehat{\xi})^\vee = \begin{bmatrix} v \\ \omega \end{bmatrix}.$$

Consider a simple robot consisting of a two links connected by a revolute or a prismatic joint. Attach frame L to one of the links and frame K to the other link. Parameterize the joint angle by the variable θ. Then,

$$g_{l,k}(\theta) = e^{\widehat{\xi}^s_{l,k}\theta}g_{l,k}(0),$$

where $\xi^s_{l,k}$ is a constant spatial twist. Consult (Murray et al., 1994) for a more detailed description of revolute and prismatic joints.

A.6 Adjoints

Adjoints are 6×6 matrices which transform velocities represented in one frame to velocities represented in another frame. Let $g = (p, R) \in SE(3)$, then the adjoint for g, Ad_g, is

$$\mathrm{Ad}_g = \begin{bmatrix} R & \widehat{p}R \\ 0 & R \end{bmatrix}$$

and

$$(\mathrm{Ad}_g)^{-1} = \begin{bmatrix} R^T & -R^T\widehat{p} \\ 0 & R^T \end{bmatrix} = \mathrm{Ad}_{g^{-1}}.$$

Also, if $\widehat{\xi}$ is a twist matrix with the corresponding twist ξ, then $g\widehat{\xi}g^{-1}$ is a twist matrix with corresponding twist $\mathrm{Ad}_g\xi$. Consider frame B and frame C to be two frames fixed relative to each other but moving with respect to a spatial frame L. The body velocity of frame C and frame K are related through the following equation:

$$V^b_{l,c} = \mathrm{Ad}_{g^{-1}_{b,c}} V^b_{l,b} = \mathrm{Ad}_{g_{c,b}} V^b_{s,b}.$$

A.7 Wrenches and Wrench Transformations

A wrench, F, is a vector in $\mathbb{R}^6$ and is formed by stacking the force, f, acting at a point above the torque, τ, acting at the same point:

$$F = \begin{bmatrix} f \\ \tau \end{bmatrix}.$$

Consider frame B and frame C to be fixed relative to each other and fixed to a single rigid body moving in space. Let frame S be a fixed inertial frame. Let F_b denote the wrench applied to the origin of frame B and written with respect to frame B. Let F_c denote the wrench applied to the origin of frame C and written with respect to frame C. An equivalent wrench is a wrench which does the same amount of work on a rigid body for all rigid body motions. Wrench F_c and wrench F_b are equivalent if

$$F_c = \mathrm{Ad}^T_{g^{-1}_{c,b}} F_b.$$

Note that

$$\mathrm{Ad}^T_{g^{-1}_{c,b}} = \begin{bmatrix} R_{c,b} & 0 \\ \widehat{p}_{c,b} R_{c,b} & R_{c,b} \end{bmatrix}.$$

A.8 Robot Equations of Motion

The differential equations of motion for a serial chain robot are

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + N(\theta, \dot{\theta}) = \tau,$$

where the joint angles are given in the vector θ, and the joint torques are given in the vector τ. The mass matrix is denoted $M(\theta)$ and is symmetric and positive definite. The Coriolis matrix is $C(\theta, \dot{\theta})$ and is chosen such that $\dot{M} - 2C$ is skew-symmetric.

A.9 Computed Torque Control

The computed torque controller in joint space determines the torques by the following equation: $\tau = M(\theta)\left(\ddot{\theta}_d - K_v \dot{e} - K_p e\right) + C(\theta, \dot{\theta})\dot{\theta} + N(\theta, \dot{\theta})$, where $\theta_d(t)$ is the desired joint trajectory and $e = \theta - \theta_d$. The gain matrices, K_v and K_p, are symmetric, positive definite matrices.

Bibliography

Abraham, R., Marsden, J. E., and Ratiu, T. S. (1988). *Manifolds, Tensor Analysis, and Applications*. Springer-Verlag, New York, second edition.

Amirouche, F. M. L. (1992). *Computational methods in multibody dynamics*. Prentice Hall, Englewood Cliffs, N.J.

Anderson, H. C. (1983). Rattle: A velocity version of the shake algorithm for molecular dynamics calculations. *Journal of Computational Physics*, 52:24–34.

Andrzejewski, T., Bock, H., Eich, E., and von Schwerin, R. (1993). Recent advances in the numerical integration of multibody systems. In Schiehlen, W., editor, *Advanced multibody system dynamics: simulation and software tools*. Kluwer Academic Publishers.

Argáez, D. I. (1993). An analytical and experimental study of the simultaneous control of motion and force of a climbing robot. Master's thesis, Massachusetts Institute of Technology.

Bae, D. and Haug, E. J. (1987). A recursive formulation for constrained mechanical system dynamics: Part i. open loop systems. *Mech. Struct. and Mach.*, 15(3):359–382.

Bae, D. and Haug, E. J. (1988). A recursive formulation for constrained mechanical system dynamics: Part ii. closed loop systems. *Mech. Struct. and Mach.*, 15(4):481–506.

Baez, J. C. and Gilliam, J. W. (1995). An algebraic approach to discrete mechanics. Preprint, http://math.ucr.edu/home/baez/ca.tex.

Baraff, D. (1996). Linear-time dynamics using lagrange multipliers. In *Computer Graphics (SIGGRAPH '96)*, pages 137–146, New Orleans, LA.

Barth, E. and Leimkuhler, B. (1996a). A semi-explicit, variable-stepsize integrator for constrained dynamics. Mathematics department preprint series, University of Kansas.

Barth, E. and Leimkuhler, B. (1996b). Symplectic methods for conservative multibody systems. *Fields Institute Communications*, 10:25–43.

Baumgarte, J. (1972). Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, 1(1):1–16.

Benettin, G. and Giorgilli, A. (1994). On the Hamiltonian interpolation of near-to-the-identity symplectic mappings with application to symplectic integration algorithms. *Journal of Statistical Physics*, 74(5-6):1117–43.

Berghuis, H. (1993). *Model-based Robot Control: from Theory to Practice.* PhD thesis, University of Twente.

Bloch, A., Krishnaprasad, P., Marsden, J., and Murray, R. (1996). Nonholonomic mechanical systems with symmetry. *Archive for Rational Mechanics and Analysis.* To appear.

Brenan, K., Campbell, S., and Petzold, L. (1989). *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations.* Elsevier Science Publishers, New York : North-Holland.

Brogliato, B. (1996). *Nonsmooth impact mechanics : models, dynamics, and control.* Springer-Verlag, London.

Brooks, V. B. (1986). *The Neural Basis of Motor Control.* Oxford University Press, New York.

Bullo, F. and Murray, R. M. (1997). Tracking for fully actuated mechanical systems: A geometric framework. CIT-CDS Technical Report 97-003, California Institute of Technology.

Channell, P. and Scovel, C. (1990). Symplectic integration of Hamiltonian systems. *Nonlinearity*, 3:231–259.

Cremer, J. F. and Stewart, A. J. (1989). The architecture of Newton, a general-purpose dynamics simulator. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 1806–1811, Scottsdale, AZ.

de Jalón, J. G. and Bayo, E. (1994). *Kinematic and Dynamic Simulation of Multibody Systems: The real-time challenge.* Springer-Verlag, New York.

Featherstone, R. (1983). The calculation of robot dynamics using articulated-body inertias. *The International Journal of Robotics Research*, 2(1):13–30.

Furusho, J. and Sano, A. (1990). Sensor-based control of a nine-link biped. *The International Journal of Robotics Research*, 9(2):83–98.

Ge, Z. and Marsden, J. (1988). Lie-Poisson integrators and Lie-Poisson Hamilton-Jacobi theory. *Phys. Lett. A*, 133:134–139.

Gillilan, R. and Wilson, K. (1992). Shadowing, rare events, and rubber bands. A variational Verlet algorithm for molecular dynamics. *J. Chem. Phys.*, 97(3):1757–1772.

Goddard, R. E., Zheng, Y. F., and Hemami, H. (1992). Control of the heel-off to toe-off motion of dynamic biped gait. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(1):92–102.

Gonzalez, O. (1996a). *Design and Analysis of Conserving Integrators for Nonlinear Hamiltonian Systems with Symmetry.* Ph.D. thesis, Stanford University. Department of Mechanical Engineering.

Gonzalez, O. (1996b). Time integration and discrete Hamiltonian systems. *Journal of Nonlinear Science.* To appear.

Goyal, S., Pinson, E. N., and Sinden, F. W. (1994a). Simulation of dynamics of interacting rigid bodies including friction i: General problem and contact model. *Engineering with Computers*, 10(3):162–174.

Goyal, S., Pinson, E. N., and Sinden, F. W. (1994b). Simulation of dynamics of interacting rigid bodies including friction ii: Software system design and implementation. *Engineering with Computers*, 10(3):175–195.

Hodgins, J. K. (1996). Three-dimensional human running. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 3271–3276, Minneapolis, MN.

Hodgins, J. K. and Raibert, M. H. (1990). Biped gymnastics. *The International Journal of Robotics Research*, 9(2):115–132.

Hodgins, J. K., Wooten, W. L., Brogan, D. C., and O'Brien, J. F. (1995). Animating human athletics. In *Computer Graphics (SIGGRAPH '95)*, pages 71–78, Los Angeles, CA.

Itoh, T. and Abe, K. (1988). Hamiltonian-conserving discrete canonical equations based on variational difference quotients. *Journal of Computational Physics*, 77:85–102.

Jain, A. (1991). Unified formulation of dynamics for serial rigid multibody systems. *Journal of Guidance Control and Dynamics*, 14(3):531–542.

Jay, L. (1996). Symplectic partitioned Runge-Kutta methods for constrained Hamiltonian systems. *SIAM Journal on Numerical Analysis*, 33(1):368–87.

Kokkevis, E., Metaxas, D., and Badler, N. I. (1996). User-controlled physics-based animation for articulated figures. In *Computer Animation*, pages 16–26, Geneva, Switzerland.

Kreutz-Delgado, K., Jain, A., and Rodriguez, G. (1992). Recursive formulation of operational space control. *International Journal of Robotics Research*, 11(4):320–8.

Kulak, F. and Schwer, E. (1991). *Computational aspect of contact, impact, and penetration.* Elemepress International, Lausanne, Switzerland.

Labudde, R. A. and Greenspan, D. (1974). Discrete mechanics-a general treatment. *Journal of Computational Physics*, 15:134–167.

Labudde, R. A. and Greenspan, D. (1976a). Energy and momentum conserving methods of arbitrary order for the numerical integration of equations of motion-i. motion of a single particle. *Numer. Math.*, 25:323–346.

Labudde, R. A. and Greenspan, D. (1976b). Energy and momentum conserving methods of arbitrary order for the numerical integration of equations of motion-ii. motion of a system of particles. *Numer. Math.*, 26:1–16.

Lambert, J. (1991). *Numerical Methods for Ordinary Differential Systems.* John Wiley and Sons, New York.

Laursen, T. A. and Simo, J. C. (1993). A continuum-based finite element formulation for the implicit solution of multibody, large deformation frictional contact problems. *International Journal for Numerical Methods in Engineering*, 36(20):3451–3486.

Leimkuhler, B. and Patrick, G. (1996). Symplectic integration on Riemannian manifolds. *Journal of Nonlinear Science*, 6:367–384.

Leimkuhler, B. J. and Skeel, R. D. (1994). Symplectic numerical integrators in constrained Hamiltonian systems. *Journal of Computational Physics*, 112:117–125.

Lewis, D. and Simo, J. (1995). Conserving algorithms for the dynamics of Hamiltonian systems on Lie groups. *Journal of Nonlinear Science*, 4:253–299.

Lewis, H. and Kostelec, P. (1996). The use of Hamilton's principle to derive time-advance algorithms for ordinary differential equations. *Computer Physics Communications.* To appear.

Lilly, K. W. (1993). *Efficient Dynamic Simulation of Robotic Mechanisms.* Kluwer Academic Publishers, Boston.

Lubich, C., Nowak, U., Pöhle, U., and Engstler, C. (1992). *MEXX - Numerical Software for the Integration of Constrained Mechanical Multibody Systems.* Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB). Available through anonymous ftp at elib.zib-berlin.de in pub/elib/codelib.

Lubich, C., Nowak, U., Pöhle, U., and Engstler, C. (1993). An overview of MEXX: Numerical software for the integration of multibody systems. In Schiehlen, W., editor, *Advanced multibody system dynamics: simulation and software tools.* Kluwer Academic Publishers.

MacKay, R. (1992). Some aspects of the dynamics of Hamiltonian systems. In Broomhead, D. S. and Iserles, A., editors, *The Dynamics of numerics and the numerics of dynamics*, pages 137–193. Clarendon Press, Oxford.

Maeda, S. (1981). Lagrangian formulation of discrete systems and concept of difference space. *Math. Japonica*, 27(3):345–356.

Marsden, J. (1992). *London Mathematical Society Lecture Note Series 174: Lectures on Mechanics.* Cambridge University Press, Cambridge, England.

Marsden, J., Patrick, G., and Shadwick, W. (1996). *Integration Algorithms and Classical Mechanics.* Fields Institute Communications. American Mathematical Society. Vol. 10.

Marsden, J. and Ratiu, T. (1994). *Introduction to Mechanics and Symmetry.* Springer-Verlag, New York.

McLachlan, R. and Scovel, C. (1995). Equivariant constrained symplectic integration. *Journal of Nonlinear Science*, 5:233–256.

McLachlan, R. I. and Scovel, C. (1996). A survey of open problems in symplectic integration. *Fields Institute Communications*, 10:151–180.

McMahon, T. A. (1984). *Muscles, Reflexes, and Locomotion.* Princeton University Press, Princeton, NJ.

Mirtich, B. (1996). *Impulse-based Simulation of Rigid Body Systems.* PhD thesis, U.C. Berkeley.

Moser, J. and Veselov, A. P. (1991). Discrete versions of some classical integrable systems and factorization of matrix polynomials. *Comm. in Mathematical Physics*, 139(2):217–243.

Munkres, J. R. (1991). *Analysis on Manifolds.* Addison-Wesley, Redwood City, CA.

Murray, R., Li, Z., and Sastry, S. (1994). *A Mathematical Introduction to Robotic Manipulation.* CRC Press, Boca Raton, Fl.

Newman, D. J., Jackson, D. K., and Bloomberg, J. J. (1996). Altered astronaut performance in downward jumping following spaceflight-ii. mass center kinematics. submitted to Exp Brain Res.

Ortiz, M. (1986). A note on energy conservation and stability of nonlinear time-stepping algorithms. *Computers and Structures*, 24(1):167–168.

Park, F., Bobrow, J., and Ploen, S. (1995). A Lie group formulation of robot dynamics. *The International Journal of Robotics Research*, 14(6):609–618.

Peric, D. and Owen, D. (1992). Computational model for 3-d contact problems with friction based on the penalty method. *International Journal for Numerical Methods in Engineering*, 35(6):1289–1309.

Petzold, L. R. (1982). A description of dassl: a differential/algebraic system solver. In *10th IMACS World Congress on System Simulation and Scientific Computation*, volume 1, pages 430–432, Montreal, Canada.

Pfeiffer, F. and Glocker, C. (1996). *Multibody Dynamics with Unilateral Contacts.* John Wiley and Sons, New York.

Raibert, M. H. (1986). *Legged Robots That Balance.* MIT Press, Cambridge, MA.

Raibert, M. H. and Hodgins, J. K. (1991). Animation of dynamic legged locomotion. In *Computer Graphics (SIGGRAPH '95)*, pages 349–358, Los Angeles, CA.

Reich, S. (1993). Symplectic integration of constrained Hamiltonian systems by Runge-Kutta methods. Technical Report 93-13, University of British Columbia.

Reich, S. (1994). Momentum preserving symplectic integrators. *Physica D*, 76(4):375–383.

Reich, S. (1996a). Symplectic integration of constrained hamiltonian systems by composition methods. *SIAM J. Numer. Anal.*, 33:475–491.

Reich, S. (1996b). Symplectic integrators for systems of rigid bodies. *Fields Institute Communications*, 10:181–191.

Rimon, E. and Burdick, J. (1994). Mobility of bodies in contact-ii: How forces are generated by curvature effects? In *1994 IEEE International Conference on Robotics and Automation*, volume 3, pages 2336–2341, San Diego, CA.

Rodriguez, G., Jain, A., and Kreutz-Delgado, K. (1992). Spatial operator algebra for multibody system dynamics. *Journal of the Astronautical Science*, 40(1):27–50.

Ryckaert, J., Ciccotti, G., and Berendsen, H. (1977). Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes. *Journal of Computational Physics*, 23:327–341.

Sanz-Serna, J. M. (1991). Symplectic integrators for Hamiltonian problems: an overview. *Acta Numerica*, 1:243–286.

Sanz-Serna, J. M. and Calvo, M. P. (1994). *Numerical Hamiltonian Problems.* Chapman and Hall, London.

Scheck, F. (1990). *Mechanics: From Newton's Laws to Deterministic Chaos.* Springer-Verlag, Berlin.

Schiehlen, W., editor (1993). *Advanced multibody system dynamics : simulation and software tools.* Kluwer Academic, Dordrecht.

Shabana, A. A. (1989). *Dynamics of multibody systems.* John Wiley and Sons, New York.

Shibberu, Y. (1994). Time-discretization of Hamiltonian systems. *Computers Math. Applic.*, 28(10-12):123–145.

Simo, J. and Gonzalez, O. (1993). Assessment of energy-momentum and symplectic schemes for stiff dynamical systems. In *ASME Winter Annual Meeting*, New Orleans. Nov. 28 - Dec. 3, 1993.

Simo, J. and Wong, K. (1991). Unconditionally stable algorithms for rigid body dynamics that exactly preserve energy and momentum. *International Journal for Numerical Methods in Engineering*, 31:19–52. Also see addendum, 33:1321-1323, 1992.

Simo, J. C. and Laursen, T. A. (1992). An augmented lagrangian treatment of contact problems involving friction. *Computers and Structures*, 42(1):97–116.

Simo, J. C. and Tarnow, N. (1992). The discrete energy-momentum method. Conserving algorithms for nonlinear elastodynamics. *ZAMP*, 43:757–792.

Stewart, A. J. and Cremer, J. F. (1989). Algorithmic control of walking. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 1598–1603, Scottsdale, AZ.

Trinkle, J., Pang, J., Sudarsky, J., and Lo, G. (1997). On dynamic multi-rigid-body contact problems with coulomb friction. *Zeitschrift für Angewandte Mathematik und Mechanik*, 77(4):267–279.

Verlet, L. (1967). Computer experiments on classical fluids. *Phys. Rev*, 159:98–103.

Veselov, A. P. (1988). Integrable discrete-time systems and difference operators. *Funkts. Anal. Prilozhen.*, 22(2):1–13.

Veselov, A. P. (1991). Integrable Lagrangian correspondences and the factorization of matrix polynomials. *Funkts. Anal. Prilozhen.*, 25(2):38–49.

Vukobratović, M., Borovac, B., Surla, D., and Stokić, D. (1990). *Biped Locomotion: Dynamics, Stability, Control, and Application.* Springer-Verlag, Berlin.

Wendlandt, J. and Sastry, S. (1996). Recursive workspace control of multibody systems: A planar biped example. In *IEEE Control and Decision Conference*, Kobe, Japan. Dec. 11-13, 1996.

Wendlandt, J. M. (1995). Pattern evocation and energy-momentum integration of the double spherical pendulum. MA thesis, University of California at Berkeley. Department of Mathematics, CPAM-656.

Winter, D. A. (1989). Coordination of motor tasks in human gait. In Wallace, S. A., editor, *Perspectives on the Coordination of Movement.* Elsevier Science Publishers, North-Holland.

Winters, J. M. and Savio, L.-Y. W. (1990). *Multiple muscle systems: biomechanics and movement organization.* Springer Verlag, New York.

Wolfram, S. (1991). *Mathematica: A System for Doing Mathematics by Computer.* Addison-Wesley, RedWood City, second edition.

Yoshida, H. (1990). Construction of higher order symplectic integrators. *Phys. Lett. A*, 150:262–268.

Zhong, Z. (1993). *Finite element procedures for contact-impact problems.* Oxford University Press, New York.