COMPUTATIONAL IMPROVEMENTS OF A MULTIBODY DYNAMIC SIMULATION ALGORITHM APPLIED TO A LANDING EVENT SIMULATION OF A FLEXIBLE LEGGED EUROPA LANDER

A Dissertation
Presented to
The Academic Faculty

By

Jacob T. Wachlin

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Mechanical Engineering

Georgia Institute of Technology

August 2018

COMPUTATIONAL IMPROVEMENTS OF A MULTIBODY DYNAMIC SIMULATION ALGORITHM APPLIED TO A LANDING EVENT SIMULATION OF A FLEXIBLE LEGGED EUROPA LANDER

Approved by:

Dr. Costello, Advisor
School of Aerospace Engineering and School of Mechanical Engineering
Georgia Institute of Technology

Dr. Rogers
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Ferri
School of Mechanical Engineering
Georgia Institute of Technology

Date Approved: May 3, 2018

The legs feed the wolf

Herb Brooks

To my parents for their constant support, and to my brothers for always being willing to discuss new ideas.

ACKNOWLEDGMENTS

I would like to thank my labmates at the Center for Advanced Machine Mobility for always being ready to answer questions or discuss ideas I have had, and for creating a fantastic lab environment.

I would like to thank my advisor, Dr. Costello, for his continual support throughout graduate school, and for always pushing me to improve my work.

I would also like to thank my numerous mentors, teachers, and coaches from throughout my life. Thank you to Raul, Roon, and the rest of my WildStang mentors. Thank you Mr. Hardman for years of fun in your classes. Thank you to Coaches Dolan, Pietro, and Marchisotto for teaching me the value of hard work and teamwork. Thank you Mike and Matt for your advice and guidance. All of your tireless work in supporting the next generation has not gone unnoticed.

Thank you to my friends in Chicago, Boston, and Atlanta for your support. You all have made my times in those places so memorable.

Finally, I would like to thank my family for their love and support. I could not have made it here without their countless hours guiding my education, supporting my interests, and encouraging me to do my best.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Multibody dynamic simulation is critical to the design and analysis of many mechanical systems. Engineers use these simulations to understand the motion and loading conditions of systems of bodies. The field of dynamic simulation has been studied for decades and many methods exist for performing multibody dynamic simulations, each with its advantages and disadvantages. For example, some methods are more computationally expensive than others, and many methods naturally eliminate inter-body loads from calculations. This thesis focuses on a constrained coordinate method for developing multibody dynamic simulations which uses nonlinear control theory techniques in the constraint stabilization task.

The constrained coordinate multibody dynamic simulation method considered in this thesis has been used to examine the performance of many systems. It has been used to model parafoil systems, articulated wing aircraft, and guided projectiles [1]. Within this method, each rigid body is simulated using a standard 6 degree-of-freedom model, with loads at connections between bodies calculated online to maintain properly constrained motion between the bodies. The method avoids the need to analytically derive a set of governing coupled differential equations for the system. In addition, it does not cancel out inter-body loads, which can be useful for engineering analysis. However, because the inter-body loads must be calculated and applied online, and constrained degrees-of-freedom are not eliminated from the simulation, this method can be computationally expensive.

This thesis makes significant computational improvements to this constrained coordinate multibody dynamic simulation algorithm. It first analyzes the algorithm to determine which sections scale most poorly with system size. It then suggests, analyzes, and tests methods to greatly reduce computation time within those problem sections. In particular, it shows how some matrix multiplication operations consist of a large number of multiplications by zero. Computation time is reduced by avoiding these trivial operations. In addition, it is shown how the joint numbering scheme determines the bandwidth of a ma-

trix corresponding to a set of linear equations that must be solved within the constraint controller. When the bandwidth is reduced, banded linear system solvers can be used to reduce computation time. The bandwidth reduction here is shown to be equivalent to the standard NP-Complete bandwidth reduction problem. Approximate bandwidth reduction methods are shown to be effective at reducing computation time. A few token systems are developed to test the methods and it is noted that computation time in some cases is reduced by more than two orders of magnitude, opening up this technique for use in trade studies of the dynamics of large systems.

Finally, these methods are applied to simulate the landing event dynamics of a proposed flexible legged lander for Europa. The reduced computation time enabled by the methods presented in this thesis allows for large Monte-Carlo simulation studies to be run in a reasonable amount of time. Systems with various levels of passive leg flexibility were modeled, as well as a system with basic active impedance control, and it was seen that flexible legs offer lower peak acceleration on impact, lower joint loads, and lower risk of rollover over a wide range of ground surface conditions, impact angles, and impact velocities. Flexible legs lowered peak lander acceleration by about 42% and 40% on simulated icy and snowy surfaces, respectively. Flexible legs were also able to virtually eliminate rollover risk when landing on those surfaces. On a simulated sandy surface with significantly higher damping, flexible legs reduced peak lander acceleration by about 31%. In addition, while landers with stiff legs rolled over in this sandy surface scenario about 35% of the time, landers with very flexible legs rolled over only 15.5% of the time.

CHAPTER 1
INTRODUCTION

Rigid body dynamic simulation is a critical step in developing many mechanical systems. Before a system is fabricated, its dynamic response can be analyzed. This can be used to develop controllers, analyze structures, test actuation methods, and generally ensure proper functionality.

There are a variety of methods used to develop the equations of motion needed to perform simulations. These methods include the Newton-Euler equations, Lagrange's equations, Kane's equations, and others. This thesis focuses on one method for performing multibody dynamic simulations, called the nonlinear control theory (NLCT) constrained coordinate method, which offers a few significant benefits over other multibody dynamic simulation methods. This method is derived in detail in Chapter 2.

The first benefit of this method is that it is based on the rigid body dynamics of a single body. Analytic differential equations need not be derived for the coupled system to be modeled. Rather, it is only necessary to generate a consistent set of initial conditions for all of the bodies and define the connections between bodies.

Second, this method does not eliminate inter-body forces and moments. Instead, constraint loads are a visible feature of this method. These loads can be useful for analyzing the structural strength of a system. Without these loads, the simulation may provide information about how the system moves, but not about the internal loading conditions. Because inter-body loads are visible when using the NLCT method, it combines nicely with FEM structural analysis methods.

This method for performing multi-body dynamic simulation has been used to simulate robotic landing gear for rotorcraft, guided parafoil systems, guided munitions, articulated-wing aircraft, and more [1, 2].

While this simulation method is convenient and effective, it comes with computational challenges. Specifically, computational complexity of sections within this algorithm scale poorly as the number of bodies and inter-body connections increase. The first part of this thesis analyzes the constrained coordinate multibody dynamic simulation method, proposes methods for computational improvements, and tests these methods on some example multibody systems. In particular, the analysis shows how some matrix multiplication operations consist of a large number of multiplications by zero. Computation time is reduced by avoiding these trivial operations. In addition, it is shown how the joint numbering scheme determines the bandwidth of a matrix corresponding to a set of linear equations that must be solved within the constraint controller. When the bandwidth is reduced, banded linear system solvers can be used to reduce computation time. The bandwidth reduction here is shown to be equivalent to the standard NP-Complete bandwidth reduction problem. Approximate bandwidth reduction methods are shown to be effective at reducing computation time. For some example systems, computation time was reduced by more than two orders of magnitude, with no loss in accuracy.

Previous work by Gross, Rogers and Costello improved computation time of this dynamic simulation method by reducing the number of times the constraint controller was called within a simulation, at the cost of some accuracy [3]. Their paper recognized that the constraint controller consumes a significant portion of total computation time. This is true even for systems which are fairly small, and in fact they only examined systems consisting of five bodies or fewer. Critically, the methods in this thesis reduce computation time within each step of the constraint controller without loss of accuracy. Therefore, total computation time is vastly reduced without sacrificing accuracy. If accuracy is not critical, the methods developed here could be combined with the methods developed by Gross et. al. to reduce computation time even further.

The second part of this thesis focuses on the simulation of a proposed legged lander design for a mission to Europa. Europa is scientifically intriguing, as its icy surface and

sub-surface thermal activity may provide the correct conditions for life. However, our best images of Europa do not provide information about the surface at the scale of a lander. Therefore, any lander must be able to safely land on a wide variety of surface conditions. The unknown conditions also imply that a large range of Monte-Carlo landing event simulations should be performed. The proposed lander model consists of 13 bodies and 12 connections. The resulting differential algebraic equation has 169 states and 60 constraints to satisfy. Using a naive implementation of the constrained coordinate method, this system is extremely slow to simulate, with simulation times about 1400 times slower than real-time. A 4800-case trade study of this lander system would take over two CPU-years to complete on the author's workstation with an Intel Xeon processor. However, this study was feasible when the methods proposed in this thesis are used, and was run in about two weeks. In fact, the methods proposed here result in a computation time reduction of about 24-fold for this lander system, with no loss in accuracy.

This thesis compares legged lander designs with various levels of passive leg flexibility, ranging from stiff legs to conforming legs which cannot support the weight of the lander system. In addition, one system with basic active impedance control was examined. Studies of these systems over a range of ground surface parameters, impact velocities, and impact angles showed that a lander with flexible legs experiences lower peak acceleration upon impact, lower peak joint loads, and a lower risk of rollover compared to a lander with stiff legs under the same conditions. Flexible legs lowered peak lander acceleration by about 42% and 40% on simulated icy and snowy surfaces, respectively. Flexible legs were also able to virtually eliminate rollover risk when landing on those surfaces. On a simulated sandy surface with significantly higher damping, flexible legs reduced peak lander acceleration by about 31%. In addition, while landers with stiff legs rolled over in this sandy surface scenario about 35% of the time, landers with very flexible legs rolled over only 15.5% of the time.

CHAPTER 2
NONLINEAR CONTROL THEORY CONSTRAINED COORDINATE MULTIBODY SIMULATION

2.1 Overview of Method

This thesis examines a constrained coordinate multibody dynamic simulation technique and develops and analyzes methods for reducing its computation time. Within this method, each individual body is treated as a rigid body allowed to undergo full three dimensional motion. Each body is exposed to coupling forces and moments from other bodies and external loads. A feedback linearization constraint controller calculates those coupling forces and moments and enforces defined joint constraints. These forces and moments are equal and opposite at a joint on the adjacent bodies.

Within this method of multibody dynamic simulation, interbody loads are readily visible, and therefore this method can determine not only how a multibody system moves, but also the internal loads as well.

2.2 Geometry and Reference Frames

Each rigid body has a reference frame attached to it, b_i. At each joint, a few reference frames are defined. Each joint connects two bodies, one of which is called the parent body and the other the child body. The parent and child body reference frames are the body reference frames of the parent and child bodies, respectively. In addition, at the joint two new reference frames are defined: the child joint and parent joint reference frames. It is critical for the constraint controller developed later on in this chapter that these reference frames are aligned when the joint has no rotational displacement. Figure 2.1 shows how these reference frames are defined.

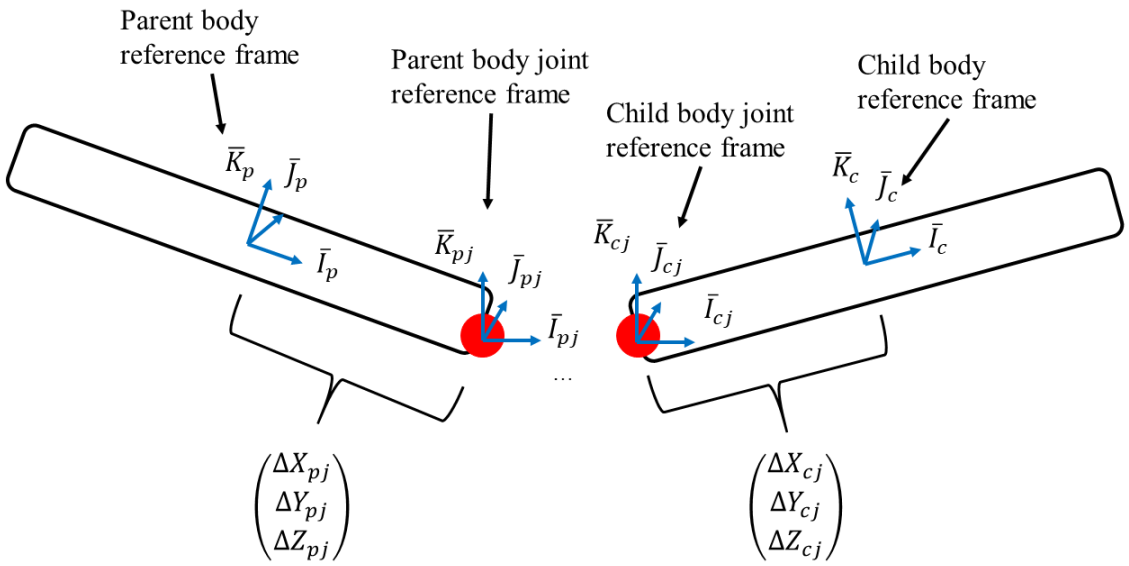

Figure 2.1: The reference frames for a joint connection

The transformation matrices associated with these coordinate systems are denoted differently depending on if the coordinate system is defined on the child body or the parent body for that joint. For transformation from the inertial reference frame into the child and parent body frames, the T_c and T_p transformation matrices are developed such that

$$\begin{pmatrix}\bar{I}_c\\ \bar{J}_c\\ \bar{K}_c\end{pmatrix} = T_c \begin{pmatrix}\bar{I}_I\\ \bar{J}_I\\ \bar{K}_I\end{pmatrix} \tag{2.1}$$

$$\begin{pmatrix}\bar{I}_p\\ \bar{J}_p\\ \bar{K}_p\end{pmatrix} = T_p \begin{pmatrix}\bar{I}_I\\ \bar{J}_I\\ \bar{K}_I\end{pmatrix} \tag{2.2}$$

Similarly, the T_{pj} and T_{cj} transformation matrices are defined to transform from the

respective body coordinate system into the joint coordinate system.

$$\begin{pmatrix} \bar{I}_{cj} \\ \bar{J}_{cj} \\ \bar{K}_{cj} \end{pmatrix} = T_{cj} \begin{pmatrix} \bar{I}_{c} \\ \bar{J}_{c} \\ \bar{K}_{c} \end{pmatrix} \tag{2.3}$$

$$\begin{pmatrix} \bar{I}_{pj} \\ \bar{J}_{pj} \\ \bar{K}_{pj} \end{pmatrix} = T_{pj} \begin{pmatrix} \bar{I}_{p} \\ \bar{J}_{p} \\ \bar{K}_{p} \end{pmatrix} \tag{2.4}$$

A distance between a body center of mass and each joint connection on that body is also defined. This is a vector in the body reference frame. For the child body, this is defined as

$$\mathbb{C}_c(\bar{r}_{\bigotimes_c \to j}) = \begin{pmatrix} \Delta X_{cj} \\ \Delta Y_{cj} \\ \Delta Z_{cj} \end{pmatrix} \tag{2.5}$$

The $\mathbb{C}_c$ operator is used to extract the measure numbers of the vector in the child body frame. For the parent body, the definition is similar, but denoted as $\bar{r}_{\bigotimes_p \to j}$.

2.3 Single Rigid Body Dynamics

Within this method of multibody dynamic simulation, each body is, at the base level, treated as a rigid body which can experience motion in all 6 degrees of freedom.

Each rigid body i has 6 degrees of freedom with states X_i such that

$$X_i = \begin{pmatrix} x & y & z & q_0 & q_1 & q_2 & q_3 & u & v & w & p & q & r \end{pmatrix}^T \tag{2.6}$$

These states include the body position (x, y, z) in the inertial reference frame, its orientation (q_0, q_1, q_2, q_3) using quaternions, its translational velocity (u, v, w) in the body reference frame, and its rotational velocity (p, q, r) in the body reference frame.

For each body, the rotational velocity vector is defined as

$$\mathbb{C}_{bi}(\bar{\omega}_{\frac{bi}{I}}) = \begin{pmatrix} p_i \\ q_i \\ r_i \end{pmatrix} \tag{2.7}$$

The $\mathbb{C}_{bi}$ operator extracts the measure numbers of the rotational velocity in the body i reference frame. The unconstrained dynamics f_i of a rigid body with mass M_i and inertia matrix I_i are defined as follows:

$$f_i = \begin{pmatrix} T_{bi}^T \begin{pmatrix} u_i \\ v_i \\ w_i \end{pmatrix} \\ B_{bi} \begin{pmatrix} p_i \\ q_i \\ r_i \end{pmatrix} \\ -\mathbb{S}_{bi}(\bar{\omega_{\frac{bi}{I}}}) \begin{pmatrix} u_i \\ v_i \\ w_i \end{pmatrix} + \frac{1}{M_i} \begin{pmatrix} F_{xi} \\ F_{yi} \\ F_{zi} \end{pmatrix} \\ -I_i^{-1}\mathbb{S}_{bi}(\bar{\omega_{\frac{bi}{I}}})I_i \begin{pmatrix} p_i \\ q_i \\ r_i \end{pmatrix} + I_i^{-1} \begin{pmatrix} M_{xi} \\ M_{yi} \\ M_{zi} \end{pmatrix} \end{pmatrix} \tag{2.8}$$

Where T_{bi} and B_{bi} are defined as

$$T_{bi} = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix} \tag{2.9}$$

$$B_{bi} = \frac{1}{2}\begin{pmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{pmatrix} \tag{2.10}$$

$\mathbb{S}_{bi}(\bar{\omega_{\overline{I}}^{bi}})$ is the skew-symmetric cross product operator applied to the rotational velocity of body i in the body i reference frame.

$$\mathbb{S}_{bi}(\bar{\omega_{\overline{I}}^{bi}}) = \begin{pmatrix} 0 & -r_i & q_i \\ r_i & 0 & -p_i \\ -q_i & p_i & 0 \end{pmatrix} \tag{2.11}$$

External forces and moments, such as those due to gravity, aerodynamic effects, ground contact, or actuation, are applied to each applicable body within the F_{xi}, F_{yi}, F_{zi} and M_{xi}, M_{yi}, M_{zi} terms, respectively.

2.4 Constraints

Joints within this multibody dynamic simulation method can be considered to be a set of constraint equations which much be satisfied throughout each simulation. For example, a gimbal (or spherical) joint is a joint in which there are three translational and zero rotational constraints. The joint points on the child and parent bodies are not permitted to move translationally relative to each other, but any rotation is allowed. This type of joint is similar to a human hip or shoulder joint.

The constraints within this method then are all derived as errors in position or orientation at joint connections which must be driven to and held at zero.

First, consider translational constraint errors. For a translational constraint, any joint displacement is equivalent to displacement between the parent joint and child joint coordinate system origins. Therefore, the translational constraint error at a joint is given in the

parent joint coordinate system by

$$E_{tj} = T_{pj} Tj \begin{pmatrix} x_{pj} - x_{cj} \\ y_{pj} - y_{cj} \\ z_{pj} - z_{cj} \end{pmatrix} + T_{pj}\mathbb{C}_p(\bar{r}_{\bigotimes_p \to j}) - T_{pj}T_pT_c^T\mathbb{C}_c(\bar{r}_{\bigotimes_c \to j}) \tag{2.12}$$

However, given some joint design, only some of the constraints need be considered. This is defined by the Γ_{tj} matrix, such that

$$E^\star_{tj} = \Gamma^T_{tj} E_{tj} \tag{2.13}$$

A rotational constraint along some axis in the parent joint coordinate system can be enforced by driving the dot product (and thus the angle) between the corresponding axes of the child and parent joint coordinate systems to zero. Rotational error is therefore defined as

$$E_{rj} = T_{pj}T_pT_c^TT_{cj}^T \tag{2.14}$$

Again, just as in the translational constraints, a joint constraint can in general have between 0 and 3 rotational constraints. The constraints of interest are extracted as

$$E^\star_{rj} = \Phi E_{rj} \Psi^T \tag{2.15}$$

In this definition, Φ and Ψ are used to extract the constraints of interest from E_{rj}.

To aid development of the constraint controller, assemble all errors into a single vector that consists of all translational and rotational errors for all M joints.

$$E(X) = \begin{pmatrix} {E^\star_{tj_1}}^T & {E^\star_{rj_1}}^T & \cdots & {E^\star_{tj_M}}^T & {E^\star_{rj_M}}^T \end{pmatrix}^T \tag{2.16}$$

2.5 Combined Equations

To develop the full equations of motion for a system, the unconstrained rigid body dynamics of a system are combined with coupled constraint loads. For N bodies and M joints, the coupled dynamics are

$$\begin{pmatrix} \dot{X}_1 \\ \dot{X}_2 \\ \vdots \\ \dot{X}_N \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{pmatrix} + \begin{pmatrix} g_{11} & g_{12} & \cdots & g_{1M} \\ g_{21} & g_{22} & \cdots & g_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ g_{N1} & g_{N2} & \cdots & g_{NM} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_M \end{pmatrix} \tag{2.17}$$

In this notation, each body state vector $\dot{X}_i$, single body dynamics vector f_i and constraint load vector u_i is a column vector, and each coupling matrix g_{ij} is a sub-matrix of the whole. As will be show later, these sub-matrices are defined according to the structure of the simulated system and describe the constraint load coupling between bodies. The total dynamics can therefore also be represented in complete matrix form as

$$\dot{X} = F + GU \tag{2.18}$$

The sub-matrices of G are only nonzero for the blocks that map some joint j to a parent body and a child body for that joint. The sub-matrices for the parent and child bodies are each defined differently. For the parent body, it is defined as the following.

$$g_p = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{M_p} T_{pj}^T \Gamma_{tj} & 0 \\ I_p^{-1} \mathbb{S}_p(\bar{r}_{\otimes_p \to j}) T_{pj}^T \Gamma_{tj} & I_p^{-1} T_{pj}^T \Gamma_{rj} \end{pmatrix} \tag{2.19}$$

For the child body, it is defined as follows.

$$g_c = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ -\frac{1}{M_c} T_c T_p^T T_{pj}^T \Gamma_{tj} & 0 \\ -I_c^{-1} \mathbb{S}_c(\bar{r}_{\bigotimes_c \to j}) T_c T_p^T T_{pj}^T \Gamma_{tj} & -I_c^{-1} T_c T_p^T T_{pj}^T \Gamma_{rj} \end{pmatrix} \tag{2.20}$$

The Γ_{tj} and Γ_{rj} matrices are defined to ensure correct mapping of joint forces and moments into the correct axes of the corresponding bodies. The width of the Γ_{tj} matrix is equal to the number of translational constraints on that joint, while the width of the Γ_{rj} matrix is equal to the number of rotational constraints on that joint. Each column of Γ_{tj} and Γ_{rj} consists of only one non-zero element, which is a 1, and is located as to map to the constrained axes.

2.6 Constraint Controller

As discussed previously, a constraint load vector U must be calculated, which when applied to the bodies according to the coupling defined in G causes the constraint errors $E(X)$ to be driven to and held at zero. To calculate this constraint load vector, a method from the field of non-linear control is used: the feedback linearization controller. To begin the derivation, take the time derivative of the error vector $E(X)$.

$$\dot{E}(X) = \frac{\partial E}{\partial X}\frac{\partial X}{\partial t} = \frac{\partial E}{\partial X}(F + GU) = \frac{\partial E}{\partial X}F + \frac{\partial E}{\partial X}GU = \frac{\partial E}{\partial X}F \tag{2.21}$$

Since the constraint loads vector U does not appear in $\dot{E}$, a second derivative of E is generated which provides access to U.

$$\ddot{E}(X) = \frac{\partial \dot{E}}{\partial X}\frac{\partial X}{\partial t} = \frac{\partial \dot{E}}{\partial X}(F + GU) = \frac{\partial \dot{E}}{\partial X}F + \frac{\partial \dot{E}}{\partial X}GU \tag{2.22}$$

Now, define $\widetilde{F}$ and $\widetilde{G}$ as

$$\widetilde{F} = \frac{\partial \dot{E}}{\partial X} F \tag{2.23}$$

$$\widetilde{G} = \frac{\partial \dot{E}}{\partial X} G \tag{2.24}$$

This provides the second order dynamics of the error equations as

$$\ddot{E}(X) = \widetilde{F}(X) + \widetilde{G}(X)U(X) \tag{2.25}$$

To develop the feedback linearization constraint controller, let the second order dynamics of the error equations equal a pseudocontrol γ.

$$\gamma = \ddot{E}(X) = \widetilde{F}(X) + \widetilde{G}(X)U(X) \tag{2.26}$$

Now, this pseudocontrol is set such that the error dynamics are exponentially stable, so that any initial errors in the constraint equations will be driven to zero by the controller.

$$\gamma = -2\zeta\omega_n \dot{E} - \omega_n^2 E \tag{2.27}$$

In the above equation, ζ is the dynamics damping ratio and ω_n is the natural frequency. Using this definition for the pseudocontrol, it can be seen algebraically that the constraint force and moment vector U is

$$U = -\widetilde{G}^{-1}(2\zeta\omega_n \dot{E} + \omega_n^2 E + \widetilde{F}) \tag{2.28}$$

With U developed in this way, it can be seen that the error dynamics are therefore an uncoupled set of damped oscillators. If ζ and ω_n are chosen properly, the error equations

will be driven exponentially to zero, and the joint constraints will be satisfied.

$$\ddot{E} + 2\zeta\omega_n\dot{E} + \omega_n^2 E = 0 \tag{2.29}$$

In this setup, the zero dynamics of the system are the dynamics of the actual simulated system including the coupling effects of the joint connections.

So far, the error equations have been taken as known, but they are fully developed in the appendix.

CHAPTER 3

ANALYSIS OF FEEDBACK LINEARIZATION CONSTRAINT CONTROLLER

This method of performing multibody dynamic simulations involves solving a set of differential algebraic equations which define the system dynamics. Let the system states be X. The set of equations which need to be solved are as follows.

$$\dot{X} = F + GU \tag{3.1}$$

$$E = 0 \tag{3.2}$$

The state time derivatives $\dot{X}$ are therefore determined by some unconstrained dynamics F, which include external loads such as gravity, contact, or aerodynamic effects, and some coupled dynamics GU which are determined by the connections between bodies. The error equations E consist of all errors in joint displacements at all inter-body connections. A key step with this method is calculating the inter-body forces and moments U. This is done with a constraint controller, as described in Chapter 2. The constraint controller calculates inter-body forces and moments such that the defined joint constraints are maintained.

In this thesis, a feedback linearization constraint controller was used. This controller was derived in Chapter 2. Here, steps within the constraint controller are analyzed and their computational complexity is determined. Knowledge of which sections are the most computationally intensive allows for a targeted approach at reducing computation time overall.

3.1 Limitations of Analysis

This section focuses on the linear algebra operations needed to calculate the inter-body loads. It ignores operations needed to assemble preliminary vectors and matrices used within the constraint controller. As will be seen, certain operations dominate the computational requirements to such an extent as to make smaller computations insignificant.

3.2 Analysis of Methods

For comparison between methods, let N be the number of states in the simulation. For the derivation in Chapter 2 using quaternions, $N = 13 \times N_B$ where N_B is the number of rigid bodies in the simulation. Likewise, let M be the number of constraints to satisfy. Also, assume the following matrices and vectors (defined in Chapter 2) have been assembled.

$$F, G, E, \dot{E}, \frac{\partial \dot{E}}{\partial X} \tag{3.3}$$

These are used to assemble the feedback linearization constraint controller. The first step is to perform a matrix-vector multiplication to form part of the error dynamics.

$$\widetilde{F} = \frac{\partial \dot{E}}{\partial X} F \tag{3.4}$$

The matrix $\frac{\partial \dot{E}}{\partial X}$ is in $\mathbb{R}^{M \times N}$ while F is in $\mathbb{R}^N$. Naively, this step has complexity $\mathcal{O}(MN)$.

The next step is another computation of part of the error dynamics. The following matrix-matrix multiplication must be performed.

$$\widetilde{G} = \frac{\partial \dot{E}}{\partial X} G \tag{3.5}$$

The matrix G is in $\mathbb{R}^{N \times M}$. Naively, this step has complexity $\mathcal{O}(M^2 N)$.

The next two steps are to first form the right-hand side of the following system of

equations, and then to solve for U.

$$\widetilde{G}U = -(2\zeta\omega_n\dot{E} + \omega_n^2 E + \widetilde{F}) \tag{3.6}$$

Assembling the right hand side consists of vector addition, and is of complexity $\mathcal{O}(M)$. Solving the system of equations, however, is harder. Naively, this has complexity $\mathcal{O}(M^3)$.

Finally, the coupling matrix G must be multiplied with the inter-body forces and moments vector U to form part of the system dynamics. These coupled dynamical effects are added to the unconstrained dynamics.

$$\dot{X} = F + GU \tag{3.7}$$

The matrix-vector multiplication operation dominates here, and this step is of complexity $\mathcal{O}(MN)$.

The complexities of all the steps are listed in Table 3.1.

Computation	Complexity
$\widetilde{F} = \frac{\partial \dot{E}}{\partial X} F$	$\mathcal{O}(MN)$
$\widetilde{G} = \frac{\partial \dot{E}}{\partial X} G$	$\mathcal{O}(M^2 N)$
$V = \widetilde{F} + 2\zeta\omega_n\dot{E} + \omega_n^2 E$	$\mathcal{O}(M)$
$U = -\widetilde{G}^{-1} V$	$\mathcal{O}(M^3)$
$\dot{X} = F + GU$	$\mathcal{O}(MN)$

Table 3.1: Complexity of steps of feedback linearization constraint controller [4]

For systems with many bodies and joint constraints, it is clear that two operations dominate computation: the matrix multiplication needed to calculate $\widetilde{G}$, and the linear system solving step needed to calculate U. These two steps will be the focus of the methods in this thesis that attempt to reduce computation time. It will be shown that some of the proposed methods indeed significantly improve the computation time of this simulation method for

large systems.

CHAPTER 4

COMPUTATIONAL IMPROVEMENT TECHNIQUES

A wide range of techniques were applied in the attempt to reduce computation time. The algorithmic changes mainly focus on two topics. First, methods to improve the matrix multiplication performed to calculate $\widetilde{G}$ are examined. Second, a variety of full-order-direct, banded-direct, and iterative methods for solving linear systems are evaluated for use within the constraint controller.

4.1 Constraint Controller - Matrix Multiplication

The calculation of $\widetilde{G}$ is expensive for large systems. This complexity may seem like a roadblock to using the constrained coordinate technique for large systems. However, the matrix multiplication to form $\widetilde{G}$ often involves sparse matrices, and computation time can be reduced by eliminating trivial multiplications by zero.

4.1.1 Compressed Sparse Column Sparse Methods

A common method for handling sparse matrices is to store them in compressed sparse column (CSC) form. In this form, the entire sparse matrix is not stored. Instead, three vectors are formed to represent the matrix. The first consists of all of the non-zero values in the matrix, stored column-wise first. The second consists of the row indices corresponding with each non-zero value stored in the first vector. The third vector consists of a list of indices of the first vector where each consecutive column begins. Once this form is created for the $\frac{\partial \dot{E}}{\partial X}$ matrix, each non-zero element is multiplied by the corresponding elements in G to form $\widetilde{G}$ efficiently.

As an illustrative example, consider the CSC storage of the following matrix M.

$$M = \begin{pmatrix} 4 & 5 & 0 & 0 & 8 \\ 0 & 0 & 0 & 1 & 0 \\ 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 10 & 6 & 0 & 0 \end{pmatrix} \tag{4.1}$$

The matrix M can be stored using the CSC method as three vectors.

$$A = \begin{pmatrix} 4 & 3 & 5 & 10 & 2 & 6 & 1 & 8 \end{pmatrix} \tag{4.2}$$

$$B = \begin{pmatrix} 0 & 2 & 0 & 4 & 3 & 4 & 1 & 0 \end{pmatrix} \tag{4.3}$$

$$C = \begin{pmatrix} 0 & 2 & 4 & 6 & 7 \end{pmatrix} \tag{4.4}$$

The A vector holds all of the non-zero elements of M in column-major order (top-to-bottom, left-to-right). The vector B holds the original row index for each element in A. Finally, the vector C consists of the indices of the vector A at which a new column begins.

Stored as a dense matrix, M has 25 elements. The three vectors of the CSC version of M have 21 elements total. While this is a minor reduction in storage needs, the overhead of performing the compression outweigh the benefits in practice. However, for much larger and more sparse matrices, the CSC storage method can greatly reduce storage needs. In addition, once a matrix is stored in CSC format, it can be multiplied using an algorithm that eliminates multiplications by zero. This is the most important benefit for reducing computation time.

4.1.2 Blockwise Sparse Methods

Another method for reducing computation time by reducing trivial multiplications by zero is to multiply $\frac{\partial \dot{E}}{\partial X}$ and G blockwise. Each of those matrices has blocks of non-zero elements. The location of the blocks is determined by the structure of connections of the simulated system, and a map of their locations can be extracted at the beginning of the simulation for efficiency. When $\frac{\partial \dot{E}}{\partial X}$ and G are multiplied, the naive dense matrix multiplication is identical to multiplying corresponding non-zero blocks, except that the block-wise method avoids most trivial multiplications by zero.

Both $\frac{\partial \dot{E}}{\partial X}$ and G are formed blockwise according to the connection structure of a simulated system. First, consider the G matrix. Blocks of this matrix are designated $g_{i,j}$. These block matrices have non-zero elements only where connection j is attached to body i.

$$G = \begin{pmatrix} g_{1,1} & g_{1,2} & \cdots & g_{1,m} \\ g_{2,1} & g_{2,2} & \cdots & g_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ g_{n,1} & g_{n,2} & \cdots & g_{n,m} \end{pmatrix} \tag{4.5}$$

As an example, if connection 2 is not connected to body 1, $g_{1,2}$ is a block zero matrix. The structure of G is dependent on the physical structure of the simulated system, as well as the body and connection numbering scheme used.

Where connection j is attached to body i, the sub-matrix $g_{i,j}$ has non-zero elements in a specific structure. Let the number of rotational constraints on a joint be N_{MJ} and the number of translational constraints be N_{FJ}. For this section's analysis, the actual values within non-zero elements will be ignored. Non-zero sections are marked as NZ. The sizes

of each block are indicated by the numbers above and to the left of the matrix.

$$g_{i,j} = \begin{array}{c} \\ 3 \\ 4 \\ 3 \\ 3 \end{array} \begin{array}{c} \begin{array}{cc} N_{FJ} & N_{MJ} \end{array} \\ \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ NZ & 0 \\ NZ & NZ \end{pmatrix} \end{array} \tag{4.6}$$

$\frac{\partial \dot{E}}{\partial X}$ is filled in a similar way. It also consists of block matrices, designated here as $z_{j,i}$. As with the $g_{i,j}$ submatrices, these have non-zero elements only where connection j is attached to body i. Note that the i, j index values are switched.

$$\frac{\partial \dot{E}}{\partial X} = \begin{pmatrix} z_{1,1} & z_{1,2} & \cdots & z_{1,n} \\ z_{2,1} & z_{2,2} & \cdots & z_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ z_{m,1} & z_{m,2} & \cdots & z_{m,n} \end{pmatrix} \tag{4.7}$$

$z_{j,i}$ also has block sections that are always zero and block sections that are non-zero. These sections are in a specific structure.

$$z_{j,i} = \begin{array}{c} \\ N_{FJ} \\ N_{MJ} \end{array} \begin{array}{c} \begin{array}{cccc} 3 & 4 & 3 & 3 \end{array} \\ \begin{pmatrix} 0 & NZ & NZ & NZ \\ 0 & NZ & 0 & NZ \end{pmatrix} \end{array} \tag{4.8}$$

It is worth noting how the blockwise multiplication proceeds in order to form $\widetilde{G}$. Both G and $\frac{\partial \dot{E}}{\partial X}$ are filled with zeros except in the blocks where connection j is attached to i. Since they have this psuedosymmetry, the blockwise multiplication is predictable. Non-

zero blocks of $\frac{\partial \dot{E}}{\partial X}$ multiply into the non-zero blocks of G.

$$\widetilde{G} = \begin{pmatrix} z_{1,1} & z_{1,2} & \cdots & z_{1,n} \\ z_{2,1} & z_{2,2} & \cdots & z_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ z_{m,1} & z_{m,2} & \cdots & z_{m,n} \end{pmatrix} \begin{pmatrix} g_{1,1} & g_{1,2} & \cdots & g_{1,m} \\ g_{2,1} & g_{2,2} & \cdots & g_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ g_{n,1} & g_{n,2} & \cdots & g_{n,m} \end{pmatrix} \tag{4.9}$$

To illustrate how this blockwise multiplication occurs, consider a four-body system, connected in a chain. Connection 1 is between bodies 1 and 2, connection 3 is between bodies 2 and 3, and connection 2 is between bodies 3 and 4. To determine the actual values in the G and $\frac{\partial \dot{E}}{\partial X}$ matrices, more information is needed. However, with this structure and numbering scheme information, the blockwise representation can be assembled.

$$\widetilde{G} = \begin{pmatrix} z_{1,1} & z_{1,2} & 0 & 0 \\ 0 & 0 & z_{2,3} & z_{2,4} \\ 0 & z_{3,2} & z_{3,3} & 0 \end{pmatrix} \begin{pmatrix} g_{1,1} & 0 & 0 \\ g_{2,1} & 0 & g_{2,3} \\ 0 & g_{3,2} & g_{3,3} \\ 0 & g_{4,2} & 0 \end{pmatrix} \tag{4.10}$$

The blocks can then be multiplied through.

$$\widetilde{G} = \begin{pmatrix} z_{1,1}g_{1,1} + z_{1,2}g_{2,1} & 0 & z_{1,2}g_{2,3} \\ 0 & z_{2,3}g_{3,2} + z_{2,4}g_{4,2} & z_{2,3}g_{3,3} \\ z_{3,2}g_{2,1} & z_{3,3}g_{3,2} & z_{3,2}g_{2,3} + z_{3,3}g_{3,3} \end{pmatrix} \tag{4.11}$$

It should be clear that by only considering the non-zero blocks, a significant number of trivial multiplications by zero are avoided. Even so, within the blockwise multiplications, about half of the multiplications are trivial. Fortunately, the trivial multiplications are in

known locations and can be easily removed. Consider the following block multiplication.

$$z_{j,i}g_{i,j} = \begin{pmatrix} 0 & NZ_1 & NZ_2 & NZ_3 \\ 0 & NZ_4 & 0 & NZ_5 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ NZ_6 & 0 \\ NZ_7 & NZ_8 \end{pmatrix} = \begin{pmatrix} NZ_2NZ_6 + NZ_3NZ_7 & NZ_3NZ_8 \\ NZ_5NZ_7 & NZ_5NZ_8 \end{pmatrix} \tag{4.12}$$

Since the first 7 rows of any $g_{i,j}$ are zero, all corresponding multiplications are trivially zero and can be ignored. In this way, trivial computations can be further reduced.

This method is summarized as Algorithm 1.

Algorithm 1 Sparse-block Matrix Multiplication

procedure SPARSE-BLOCK MATRIX MULTIPLICATION(G, $\frac{\partial \dot{E}}{\partial X}$)
    Let P_i be the parent body index of connection i
    Let C_i be the child body index of connection i
    for $i \leftarrow 1, NC$ **do**
        for $j \leftarrow 1, NC$ **do**
            $S_i \leftarrow \{P_i, C_i\}$
            $S_j \leftarrow \{P_j, C_j\}$
            if $S_i \cup S_j \neq 0$ **then**
                Perform corresponding sub-multiplication into $\widetilde{G}$
            end if
        end for
    end for
end procedure

This block-wise method was tested within the the full simulation program and shown to offer significant speed improvements. The results will be shown later.

4.1.2.1 Sparse Block Matrix Multiplication Computational Complexity

With knowledge of how the sparse-block matrix multiplication is formed, it is possible to derive the asymptotic computational complexity of the method. Here we assume N is the number of states in the simulation, and M is the number of constraints to satisfy. Using a naive implementation of the matrix multiplication algorithm, the calculation of $\widetilde{G}$ is of complexity $\mathcal{O}(M^2N)$.

The sparse-block matrix multiplication method here consists of a number of small matrix multiplications, between sub-matrices of G and $\frac{\partial \dot{E}}{\partial X}$. The sizes of the sub-matrices can vary by the number of constraints on each joint, and so the computational complexity of those sub-multiplications can also vary. However, the sizes of the sub-matrices are capped since the number of constraints cannot exceed the available DOF, which is 6. The complexity of each of these sub-multiplications is therefore limited, and does not change with the system size. The overall complexity of the sparse-block matrix multiplication method is therefore driven by the number of sub-multiplications needed to fully generate $\widetilde{G}$.

A straight-forward way to determine when sub-multiplications are necessary is to examine Algorithm 1. The connections are all compared to each other, and a set of body indices for each joint is generated. A sub-multiplication is performed whenever there is a body that is in the body indices set for each joint being compared.

A clear consequence of this is that there are 2 sub-multiplications for each connection in the system. Clearly, when a connection is compared to itself, the body indices set will be identical for both the child and parent body.

Next, it helps to consider the system of bodies and connections as a graph. Each connection is a vertex in the graph, and each body is a set of one or more edges on the graph. An example of this representation is provided in Figure 4.2. Within Algorithm 1, a sub-multiplication is performed if the connections are each connected to some body. This is equivalent to there being an edge between the vertices that correspond to the connections. Note that two sub-multiplications would be performed twice for each edge on the graph,

since if some node a is connected to some node b, then node b is connected to node a, and a sub-multiplication will be performed for each of those two comparisons.

Let the number of connections in the system be n_c, and the number of edges in the graph representation of the system be n_e. The number of sub-multiplications is $2n_c + 2n_e$ and the computational complexity of this method is therefore $\mathcal{O}(n_c + n_e)$.

4.2 Constraint Controller - Linear System Solve

The linear system solving step within the constraint controller is computationally expensive for large systems. This thesis evaluates a variety of methods to reduce computation time within this step.

4.2.1 Direct Methods

4.2.1.1 LU-Decomposition With Pivoting

A well studied and numerically robust algorithm for solving systems of linear equations is LU-decomposition with pivoting. This algorithm was implemented and used as a baseline for the testing of various other methods. This is a well known method that can be found in many linear algebra textbooks [5].

4.2.1.2 Gauss-Jordan Method

LU-decomposition is efficient for solving a system of linear equations $Ax = b$ when the matrix A is constant and only the vector b is changing. However, within the constraint controller, both the matrix A and vector b change throughout the course of the simulation. Therefore, it is in fact more efficient to use Gaussian Elimination without forming the LU-decomposition to solve the system. This is a faster full-order direct solving algorithm implemented in this thesis as another baseline comparison. This is a well known method that can be found in many linear algebra textbooks [5].

4.2.2 Iterative Methods

The system of linear equations $\widetilde{G}U = -V$ must be solved for U every time the constraint controller is called. However, each time the constraint controller is called, the system changes only very slightly, such that the solution U from the previous iteration very nearly solves the system for the current timestep. Therefore it is proposed that iterative methods may be able to quickly converge on a solution for the current timestep based on the quality "guess" of the previous solution.

4.2.2.1 Conditioning Requirements

Many iterative methods to solve the system of linear equations $\widetilde{G}U = -V$ for U require conditioning on the matrix $\widetilde{G}$ which are not necessarily met in the system of equations being solved in the constraint controller. $\widetilde{G}$ is generally indefinite, not diagonally dominant, and not symmetric, which disqualifies it from the direct use of iterative methods such as the Jacobi method or the Gauss-Seidel method. Therefore, many methods would require preconditioning on $\widetilde{G}$, but that would generally require a costly matrix multiplication, eliminating any benefit of iterative methods in the first place.

However, there are a few methods that do not require special conditioning on $\widetilde{G}$ and can therefore be used directly. These include residual norm steepest descent (RNSD) and the full orthogonalization method (FOM).

4.2.2.2 Residual Norm Steepest Descent

Residual norm steepest descent is a well known projection based iterative method to solve the system of linear equations $Ax = b$ that does not require specific conditioning on the matrix A [6]. It can therefore, in theory, work for solving $\widetilde{G}U = -V$ within the constraint controller. However, no convergence rate is guaranteed, and it can be poor. In practice, the algorithm is given a maximum number of iterations, and if it does not converge in fewer iterations than that, the algorithm is presumed to fail. A direct method would then be used

to solve the system of equations. This method is described in Algorithm 2.

Algorithm 2 Residual Norm Steepest Descent

procedure RNSD($A, b, x_{guess}, maxiter, threshold$)
    $r \leftarrow b - Ax_{guess}$
    $x \leftarrow x_{guess}$
    $converged \leftarrow false$
    while $converged \neq true$ and $iteration < maxiter$ **do**
        $v \leftarrow A^T r$
        $\alpha \leftarrow \frac{\|v\|^2}{\|Av\|^2}$
        $x \leftarrow x + \alpha v$
        $r \leftarrow r - \alpha Av$
        if $change < threshold$ **then**
            $converged \leftarrow true$
        end if
    end while
    if $converged = true$ **then**
        return x
    else
        return $Error$
    end if
end procedure

4.2.2.3 Full Orthogonalization Method

The full orthogonalization method (FOM) is an iterative method based on projection onto Krylov subspaces [6]. This method is described in Algorithm 3.

Algorithm 3 Full Orthogonalization Method

procedure FOM(A, b, x_{guess}, m)
  $r_0 \leftarrow b - Ax_{guess}$
  $\beta \leftarrow \|r_0\|_2$
  $v_1 \leftarrow \frac{r_0}{\beta}$
  $H_m \leftarrow 0$
  $p \leftarrow m$
  for $j = 1 : m$ **do**
    $\omega_j \leftarrow Av_j$
    for $i = 1 : j$ **do**
      $H_m[i, j] \leftarrow \omega_j \cdot v_i$
      $\omega_j \leftarrow \omega_j - H_m[i, j]v_i$
    end for
    $H_m[j + 1, j] \leftarrow \|\omega_j\|_2$
    if $H_m[j + 1, j] = 0$ **then**
      $p \leftarrow j$
      break
    end if
    $v_{j+1} \leftarrow \frac{\omega_j}{H_m[j+1,j]}$
  end for
  $y_p \leftarrow H_p^{-1} V_p^T r_0$
  $x \leftarrow x_{guess} + V_p y_p$
  return x
end procedure

4.2.3 Banded Methods

While iterative methods may be able to provide a quick estimate of the solution to the equations $\widetilde{G}U = -V$, it is also possible in certain cases to speed up computation of direct solvers. In particular, banded solvers can take advantage of the bandwidth of $\widetilde{G}$, and can solve the system in $\mathcal{O}(b^2 n)$ time for matrix bandwidth b and $\widetilde{G} \in \mathbb{R}^{n \times n}$, compared to $\mathcal{O}(n^3)$ time for a dense $\widetilde{G}$. If the bandwidth of $\widetilde{G}$ is small compared to its size, a banded solver could provide significant speed improvements for this step of the constraint controller.

Therefore, it is important to investigate what determines the bandwidth of $\widetilde{G}$. If its bandwidth can be reduced, banded direct linear system solving methods can reduce computation time without a loss of accuracy.

$\widetilde{G}$ is a block matrix, and the non-zero blocks map loading effects between directly interacting connections. In other words, blocks are non-zero when the corresponding connections are attached to the same body. An implication of this is that the diagonal blocks of $\widetilde{G}$ are always non-zero, since any connection is clearly connected to the same body as itself.

To provide some insight into how this block-wise mapping occurs, consider the example system below. This is a "lander" system with a large main body and 4 legs with 2 bodies in each leg. A representation of the physical layout of this system is shown in Figure 4.1.

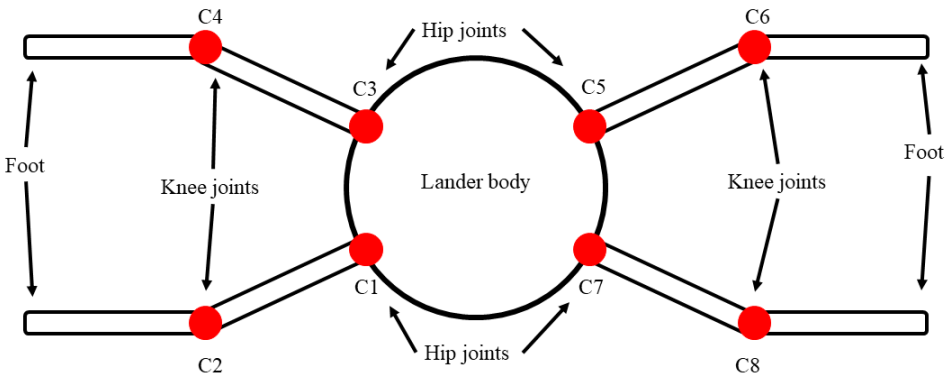

Figure 4.1: 4-Legged Lander System Joint Numbering

From this representation, an undirected graph can be made where each joint is a node and each body is a set of one or more edges. An edge connects two nodes in the graph if the corresponding joints connect to the same body. Figure 4.2 shows how this can be formed for the given example system here.

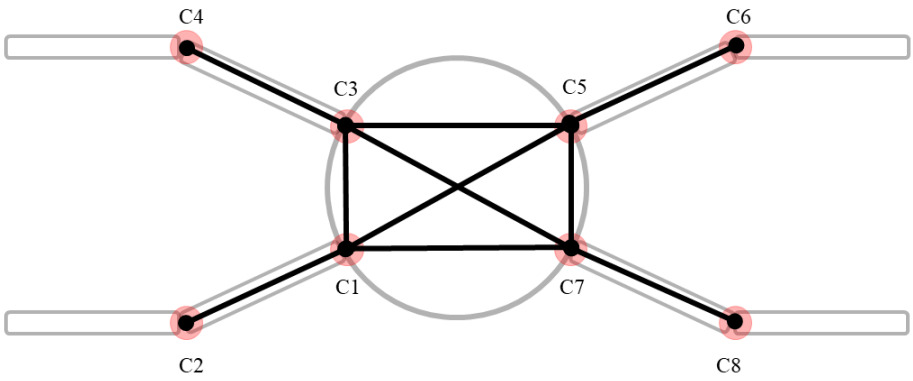

Figure 4.2: 4-Legged Lander System Generated Graph Representation

From this undirected graph, the adjacency matrix can be formed [7]. Note that the adjacency matrix of an undirected graph is always symmetric. This adjacency matrix is a nearly direct map representation of the non-zero blocks of $\widetilde{G}$. The only difference is that an adjacency matrix typically has zeros along the diagonal, whereas the diagonal blocks are always non-zero for $\widetilde{G}$. For the example system presented here, this matrix has the

following form.

$$\widetilde{G}_{map} = \begin{pmatrix} X & X & X & 0 & X & 0 & X & 0 \\ X & X & 0 & 0 & 0 & 0 & 0 & 0 \\ X & 0 & X & X & X & 0 & X & 0 \\ 0 & 0 & X & X & 0 & 0 & 0 & 0 \\ X & 0 & X & 0 & X & X & X & 0 \\ 0 & 0 & 0 & 0 & X & X & 0 & 0 \\ X & 0 & X & 0 & X & 0 & X & X \\ 0 & 0 & 0 & 0 & 0 & 0 & X & X \end{pmatrix} \tag{4.13}$$

The fact that the adjacency matrix of the undirected graph representation of a system has the same form as the block representation of $\widetilde{G}$ is a critical development in this thesis. To effectively use banded methods to solve the system of linear equations $\widetilde{G}U = -V$ for the constraint loads U within the constraint controller, the bandwidth of $\widetilde{G}$ should be minimal. This bandwidth can be reduced by performing a blockwise permutation of $\widetilde{G}$, which is equivalent to renumbering the connections for the system.

The problem of renumbering the joints to reduce the bandwidth of $\widetilde{G}_{map}$ is therefore equivalent to renumbering the nodes of an undirected graph to minimize the bandwidth of its adjacency matrix, which is well known to be NP-complete [8].

Minimizing the bandwidth of $\widetilde{G}_{map}$ is not identical to minimizing the bandwidth of $\widetilde{G}$. Rather, it is equivalent to a blockwise bandwidth minimization of $\widetilde{G}$. Therefore, bandwidth minimization of $\widetilde{G}_{map}$ implies only an approximate bandwidth minimization of $\widetilde{G}$, but has some other benefits that make it desirable. First, $\widetilde{G}$ is significantly larger than $\widetilde{G}_{map}$. For some systems, a brute force approach may be reasonable to perform on $\widetilde{G}_{map}$ but would be impossible on $\widetilde{G}$. For the above example, $\widetilde{G}_{map}$ is in $\mathbb{R}^{8\times 8}$ and $\widetilde{G}$ could be in $\mathbb{R}^{48\times 48}$. A brute force bandwidth minimization approach on $\widetilde{G}_{map}$ may take minutes, but may never be possible on $\widetilde{G}$. Second, any permutation of $\widetilde{G}_{map}$ is simply a joint number reordering.

Within the algorithm, the joint numbers are for bookkeeping, and no equations or methods need change. However, a general permutation on $\widetilde{G}$ could involve mixing up forces and moments between different connections.

An important result here is that the number of non-zero elements in the adjacency matrix of the graph representation of a system is equal to twice the number of edges. The $\widetilde{G}_{map}$ matrix is the adjacency matrix with non-zero elements on the diagonal. The number of non-zero elements in this matrix can be tied back to the computational complexity of the block-sparse matrix multiplication method discussed earlier. Let n_{nz} be the number of non-zero elements in $\widetilde{G}_{map}$, and let n_c be the number of connections in the system. The computational complexity of the block-sparse matrix multiplication is therefore $\mathcal{O}(n_c + n_{nz})$.

Because the computation needed to solve the system of equations in the constraint controller can be a significant portion of the total computation for a simulation, and since banded solvers offer significant computation reductions in certain cases, this implies some interesting results that are important to discuss.

First, it would seem possible that computation time could in fact be reduced for certain systems by adding connections and bodies. Naively, this seems like an odd result. After all, this thesis discusses many times how computation time increases with system size. However, there is no inconsistency. Using naive techniques (full order matrix multiplication and full order direct linear system solvers), adding connections and bodies would surely increase computation time. But the methods presented in this thesis operate faster under certain conditions. Surely, if the system layout is developed to take advantage of this, computation time should be reduced. Later in this thesis, a system is tested which shows that this can in fact occur.

Second, it should be noted that while finding a lower bandwidth ordering for a system may take a significant amount of time, it only needs to be done once for any system. If a trade study of hundreds or thousands of simulations is to be done, the better ordering can be used for all of the cases, and a significant computational improvement may be achieved

for the entire project.

Finally, since the base problem with reducing system bandwidth is NP-complete it might seem impossible for a simulation developer to improve a simulation in this way. However, this is generally not the case. While there are no known polynomial time algorithms to find the optimal joint numbering, there are polynomial time algorithms to find approximations of the optimal solution. Further, the system developer may be able to find better layouts by inspection. Finally, there are exhaustive search algorithms that reject obviously non-optimal orderings early and can, in practice, often provide the minimal bandwidth ordering of a system in a reasonable amount of time. Two of these algorithms are described here. The algorithms examined here are not meant to be an exhaustive list of bandwidth minimization algorithms. Since the bandwidth minimization problem is NP-complete, it is an incredibly important problem, and huge amounts of effort have been put into developing algorithms to solve it. The two algorithms shown here are merely examples of approximate and exhaustive search methods, respectively.

4.2.3.1 Reverse Cuthill-McKee

The Cuthill-McKee algorithm is an approximate method for reducing the bandwidth of symmetric sparse matrices [9]. The reverse Cuthill-McKee (RCM) method is a slight modification in which the resulting permutation is reversed, and often results in an ordering more suitable for use with banded linear system solvers. While this method does not guarantee the optimal bandwidth will be found, in practice it often produces a bandwidth reduction and it runs in polynomial time.

Within this method, a matrix is considered as an adjacency matrix of a corresponding graph. The method returns an ordered set Q of the original vertices that corresponds to the new ordering. The full method is described in Algorithm 4.

Algorithm 4 Reverse Cuthill McKee

procedure RCM($A \in \mathbb{R}^{n \times n}$)
    $v \leftarrow$ minimum degree vertex in A
    $Q \leftarrow v$
    for $i = 1, 2, ...$ while $|Q| < n$ **do**
        $A_i = Adj(Q_i) \setminus Q$
        Sort A_i by vertex order
        Append A_i to Q
    end for
    return Q
end procedure

4.2.3.2 Minimum Bandwidth by Iterative Deepening

Del Corso and Manzini developed an algorithm called, "Minimum Bandwidth by Iterative Deepening" (MB-ID) and showed that it could find the minimum bandwidth for some matrices in a small amount of time [10]. This algorithm uses a depth-first search technique, and is designed to reject any permutation that cannot possibly satisfy a bandwidth requirement as early as possible, in order to shrink the search space. Del Corso and Manzini tested this algorithm on matrices of sizes between 40 and 100, and were able to find the minimal bandwidth ordering for many matrices in time ranging from a few seconds to two hours. Further, they showed that for many of the matrices that the algorithm did solve within two hours, the minimal bandwidth it found was around 10% smaller than the bandwidth found by approximation techniques [10].

4.2.3.3 Banded Gauss-Jordan Method

The Gauss-Jordan method is readily applicable to banded systems, and can offer significant computational advantages without loss in accuracy if the bandwidth is small relative to the

size of the matrix. This banded method simply avoids performing trivial computations.

CHAPTER 5

COMPUTATIONAL IMPROVEMENTS METHODS TESTING

In order to test the effects of the methods developed in the previous chapter, a variety of example systems were simulated. Each system was simulated using various combinations of system size and computational methods in order to determine how computation time was affected. In addition, the differences in their final states were recorded to ensure simulation accuracy was not being lost. Simulations were also developed to demonstrate how it is not only the computational methods used that affect computation time, but also the design of the system model developed by the engineer. It was shown that the internal mathematical representation of a system can significantly affect computation time.

5.1 Testing Setups

5.1.1 Chain Simulation

The first simulation setup is a chain topology. In this case, *N* bodies are connected by *N-1* joints. Each joint is a hinge joint consisting of 5 DOF that are held rigid by the simulation constraint controller, and 1 DOF that is not. That remaining 1 DOF is set up with a rotational spring-damper setup whose stiffness and damping are defined as a piecewise linear function. An example representation of such a system is shown in Figure 5.1.

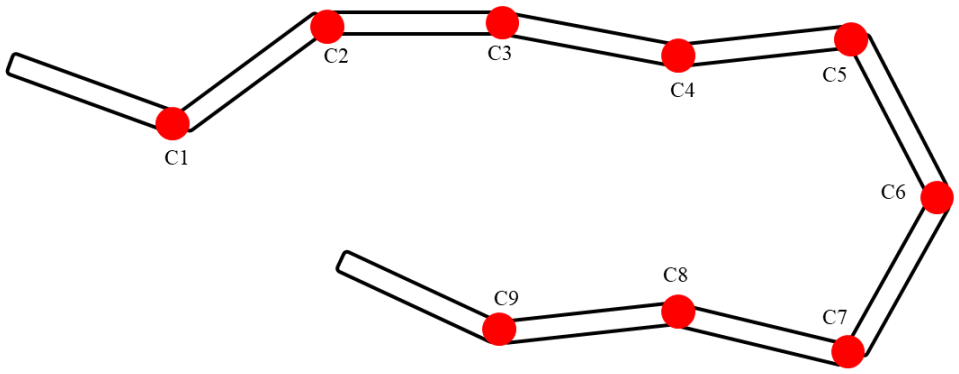

Figure 5.1: Token representation of a 10 body chain system

Such a setup could be useful in the simulation of snake-like robots, or for the simulation of ropes and chains.

5.1.2 Lander Simulation

The second simulation setup is used to mimic a legged lander with N legs. Such a system is made up of one main lander body, and N legs equally spaced around it. Each leg is made up of two bodies, and has two joints - a hip joint and a knee joint. Each joint is a hinge joint. There are N contact points on the tips of the lower leg bodies to simulate feet, and the main lander body has five contact points on its belly. An example representation of such a system is shown in Figure 5.2.

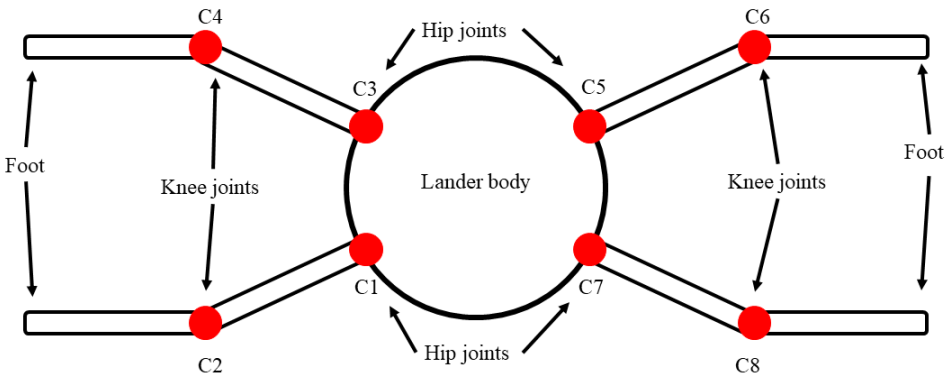

Figure 5.2: Token representation of a 4-legged lander system

Such a system is discussed further in Chapter 6 in the simulation of a flexible legged lander system proposed for Europa exploration.

5.1.3 Stubby Lander Simulation

The next simulation setup that was tested is called the "stubby" legged lander. In this case, a number of legs are connected to a central body. Each leg consists of a single rigid body. This simulation is important in that it shows how computation time can be reduced by adding bodies and connections to a simulation. In fact, two systems that are nominally the same are tested here. The first, shown in Figure 5.3, is the naive setup. One central body acts as a hub to which a number of bodies attach. The graph representation of this system whose adjacency matrix can be used to form $\widetilde{G}_{map}$ is complete, therefore the adjacency matrix is fully dense. Because of this, banded linear system solvers offer no computation reduction.

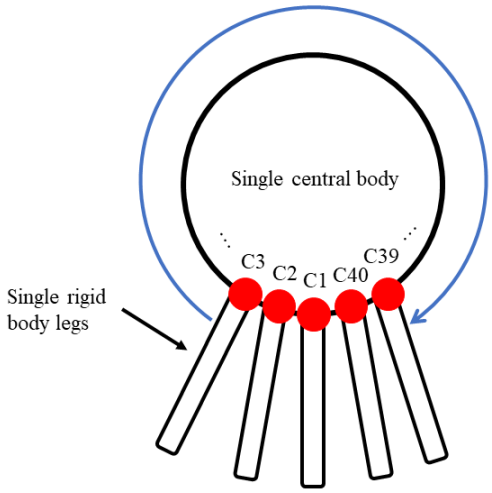

Figure 5.3: Naive stubby leg system layout with 40 legs

In contrast, the second version has a central section comprised of two bodies. These two bodies are connected with a joint with six DOF of constraints, effectively rigidly connecting them. Each of those two bodies has half the original number of legs attached to it. This setup adds one body and one connection, but since the $\widetilde{G}_{map}$ is not fully dense, banded linear system solvers can help reduce computation time. This setup is shown in Figure 5.4.

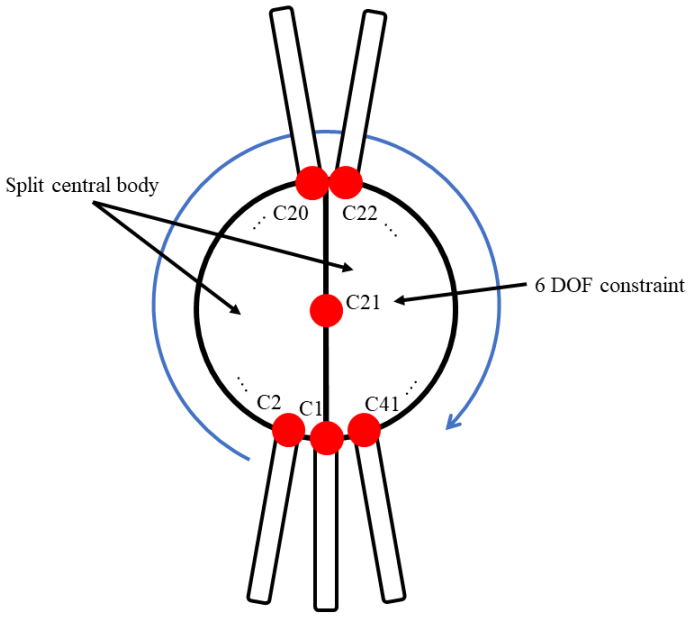

Figure 5.4: Split stubby leg system layout with 40 legs

5.1.4 Chain Simulation Bandwidth Modification

As previously shown, a reduction of the bandwidth of $\widetilde{G}$ combined with banded linear system solvers should result in a reduction in computation time. To test this, the chain simulation model was run with identical initial conditions and two different joint numbering schemes. These systems were physically identical, but the numbering of joints was set in one case to be optimal, and in the other case to be the worst possible. Systems set up in this manner were simulated over a range of number of bodies, from 15 to 50 bodies. To understand how this numbering scheme changed the computation time, consider the following chain system layout with 10 bodies. The model with optimal numbering is represented in Figure 5.5. Any two consecutive connections have a maximum numbering difference of one. This results in the optimally reduced bandwidth for $\widetilde{G}_{map}$.

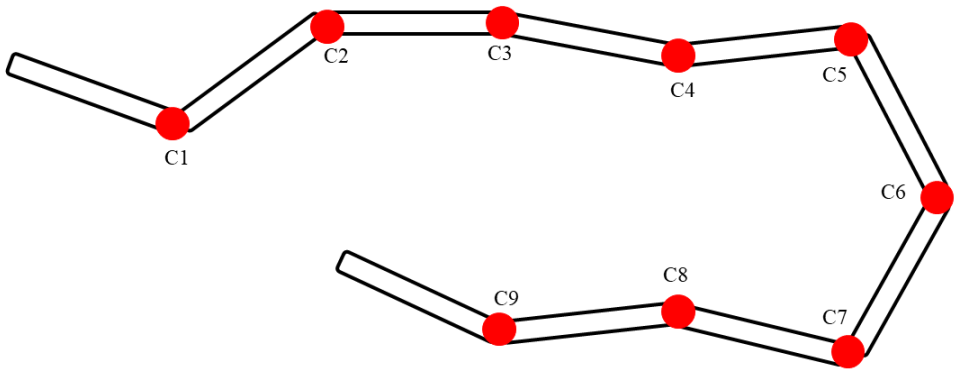

Figure 5.5: 10 body chain simulation with optimal joint numbering

For this optimal ordering, $\widetilde{G}_{map}$ has the following form. Clearly, this system has a very tight bandwidth. In fact, for such a system, the bandwidth does not increase with the number of bodies.

$$\widetilde{G}_{map} = \begin{pmatrix} X & X & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ X & X & X & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & X & X & X & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & X & X & X & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & X & X & X & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & X & X & X & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & X & X & X & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & X & X & X \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & X & X \end{pmatrix} \tag{5.1}$$

On the contrary, Figure 5.6 shows a numbering scheme that results in a maximal band-

width, which prevents the use of banded linear solvers. In this case, the lowest and highest numbered connections (joint 1 and joint 9) are connected through a body. This results in maximal bandwidth.

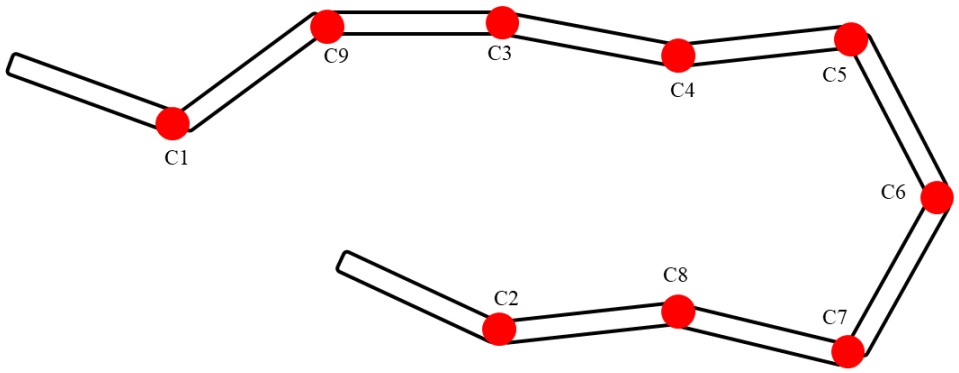

Figure 5.6: 10 body chain simulation with worst joint numbering

This worst numbering scheme results in $\widetilde{G}_{map}$ with the following form. The blocks in the top right and bottom left corners are generally non-zero, resulting in maximal bandwidth. Of course, the rest of this map could change while keeping the bandwidth maximal. Therefore, while this is a poor numbering scheme, there are others equivalently bad, at least from a bandwidth perspective.

$$\widetilde{G}_{map} = \begin{pmatrix} X & 0 & 0 & 0 & 0 & 0 & 0 & 0 & X \\ 0 & X & 0 & 0 & 0 & 0 & 0 & X & 0 \\ 0 & 0 & X & X & 0 & 0 & 0 & 0 & X \\ 0 & 0 & X & X & X & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & X & X & X & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & X & X & X & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & X & X & X & 0 \\ 0 & X & 0 & 0 & 0 & 0 & X & X & 0 \\ X & 0 & X & 0 & 0 & 0 & 0 & 0 & X \end{pmatrix}$$

5.2 Testing Results

5.2.1 Chain Simulation

For a chain simulation in which consecutive joints are numbered consecutively, the $\widetilde{G}$ matrix has a small bandwidth and is sparse. For this example simulation, each joint is a hinge, so that the bandwidth is always 9, regardless of the number of bodies or joints. Sparse multiplication and banded solvers therefore can greatly speed up the simulation of such a system.

To examine how the computational improvement methods affect simulation time as the system size changes, simulations were run for a variety of system sizes with every permutation of the matrix multiplication techniques and linear system solver techniques discussed in this thesis. Both the Runge-Kutta 4th order (RK4) and Runge-Kutta-Fehlberg (RKF45) numerical integration methods were used.

Figure 5.7 shows testing results from a sweep of system size from 5 bodies to 45 bodies. Two methods were compared. The naive method used naive matrix multiplication and LU-decomposition within the constraint controller. The improved method used sparse-block matrix multiplication and banded Gaussian elimination within the constraint controller.

All times were normalized by the fastest case, which occurred using the improved method for 5 bodies.

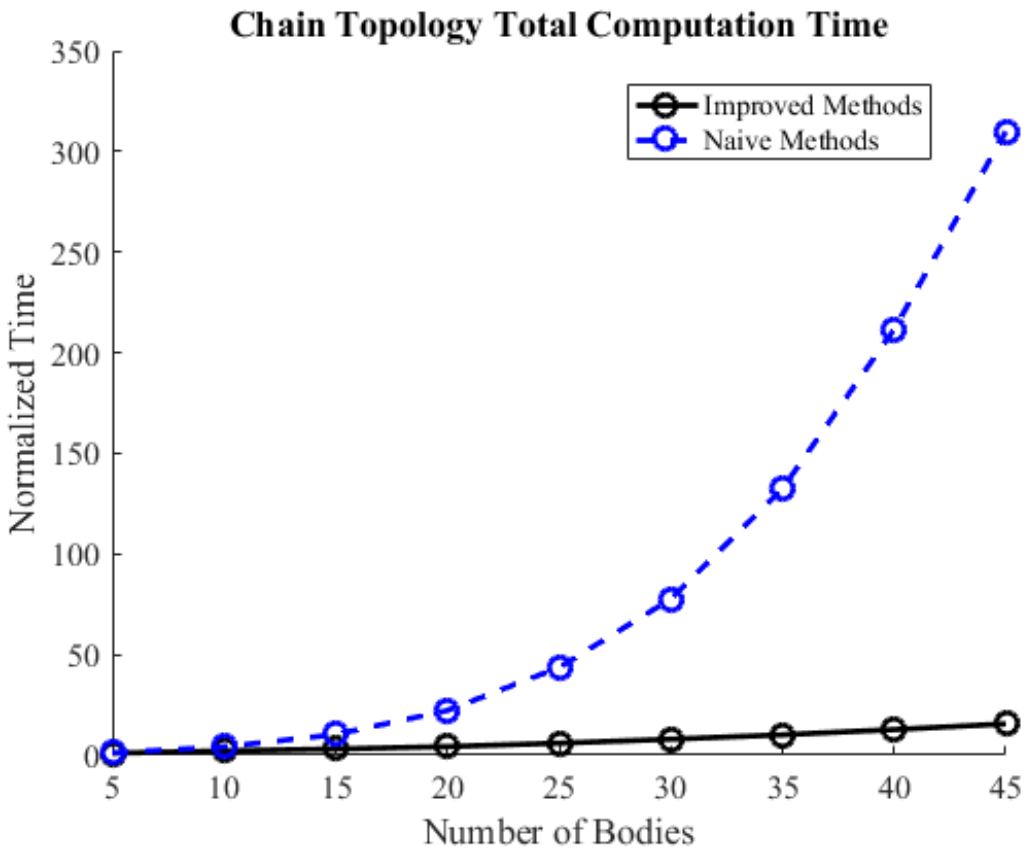

Figure 5.7: Normalized total computation time for chain topology systems of size from 5 to 45 bodies

Clearly, the improved methods offer significant computation time reduction. Figure 5.8 shows the computation time reduction achieved by the improved computational methods for systems of various size.

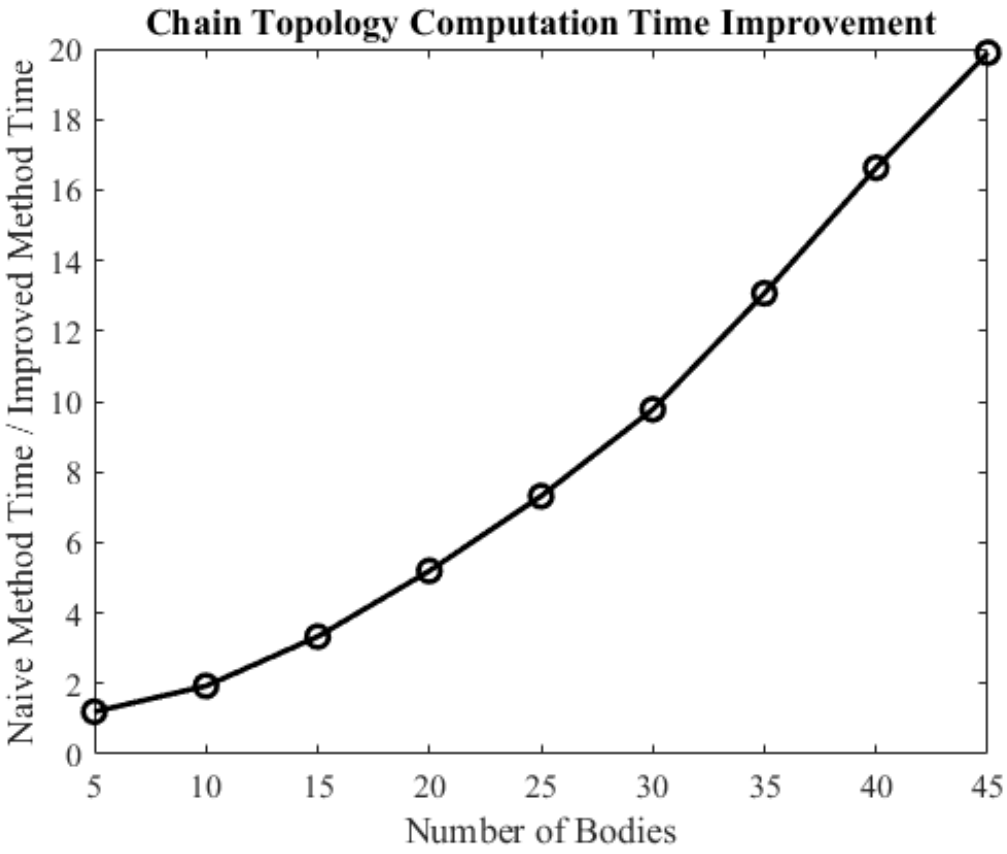

Figure 5.8: Computation time reduction for improved methods for chain topology systems of size from 5 to 45 bodies

As the system size increases, the computational cost of performing naive matrix multiplication and linear system solve steps within the constraint controller increases quickly. Figure 5.9 shows the percentage of total computation time taken solely within the constraint controller matrix multiplication and linear system solving steps. As the system size increases, those two sections dominate the total computation time.

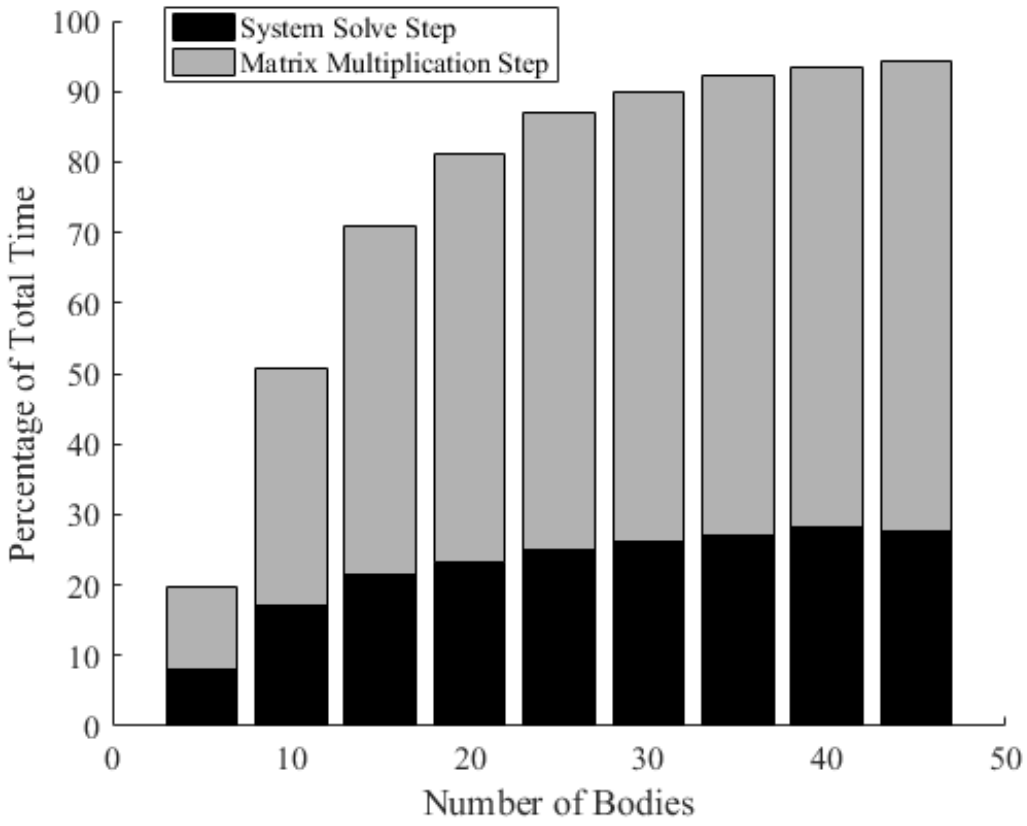

Figure 5.9: Percentage of total computation time used within constraint controller matrix multiplication and linear system solve steps using naive methods

When the improved methods are used, however, the percentage of total computation time used within those two steps stays relatively constant as the system size changes, as shown in Figure 5.10.

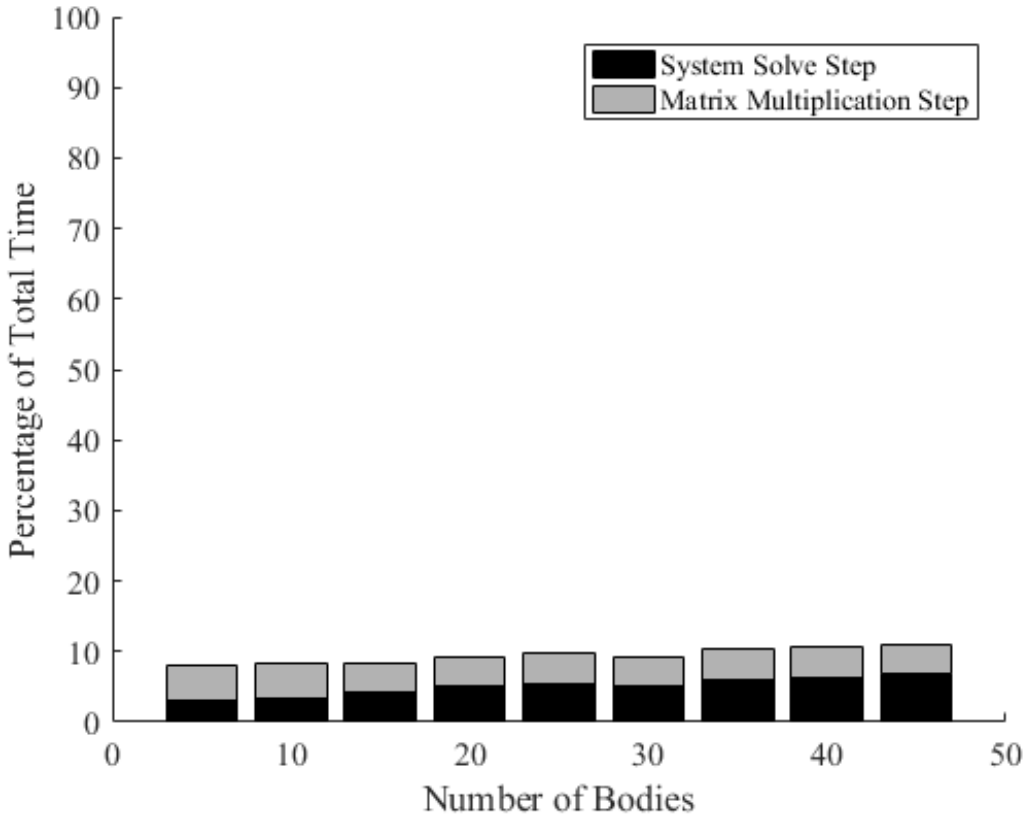

Figure 5.10: Percentage of total computation time used within constraint controller matrix multiplication and linear system solve steps using improved methods

The cases presented in Figure 5.7 were all run using the same integration method, RKF45, so that the methods used to improve the constraint controller are solely responsible for computational time reductions. However, this is a case in which the system is in free fall for the start of the simulation, and then bounces off the ground. The dynamics therefore change significantly throughout the simulation, and the RKF45 integration method should offer significantly faster simulations than a fixed timestep method such as RK4. To test this, simulations were run of a 45-body chain topology system, using RK4 and RKF45, with a mix of computational improvement methods used within the constraint controller. This chain topology system had its joints ordered for optimal bandwidth reduc-

tion. The system states at the end of the simulation were recorded and compared, to ensure that the use of any of these methods was not producing a different simulation result.

All results are normalized by the fastest time, which came during Case 2. The setup for the cases tested here is described in Table 5.1. The results from the tests are shown in Figure 5.11.

Case #	Integrator	C.C. Matrix Multiplication Method	C.C. Linear System Solve Method
1	RKF45	Fortran matmul	PLU
2	RKF45	Sparse-block	Banded Gauss-Jordan
3	RKF45	Sparse-block	FOM
4	RKF45	Sparse-block	RNSD
5	RK4	Fortran matmul	PLU
6	RK4	Sparse-block	Banded Gauss-Jordan
7	RK4	Sparse-block	FOM
8	RK4	Sparse-block	RNSD

Table 5.1: Setup of each case presented in Figure 5.11

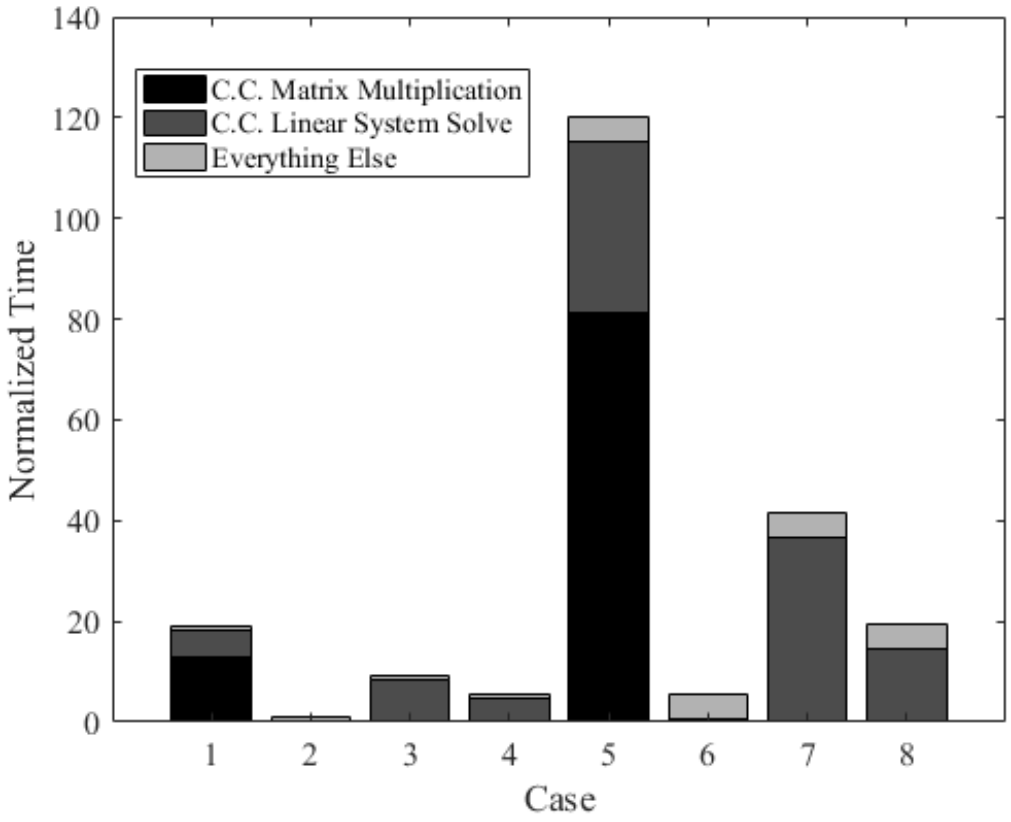

Figure 5.11: Normalized computation time required for a variety of cases for a chain topology system

With identical numerical integration techniques, the computational improvement methods within the constraint controller result in about a 19-fold reduction in computation time for a 45-body chain topology system. When RK4 and RKF45 integration techniques are also compared, the computation time reduction increases to 120-fold.

Clearly, the methods developed here are highly effective in practice in a chain topology system. These methods focus on large matrix multiplication and linear system solving steps. Naively, the computational complexity of each of these steps increases with the cube of the number of bodies simulated. However, if the connections are optimally ordered to minimize the bandwidth of $\widetilde{G}$, the bandwidth does not change with the number of bodies

in the simulation. Since the banded linear solver has computational complexity $\mathcal{O}(b^2 n)$, where n is the number of constraint equations, the computational complexity of the linear system solve step for a chain topology system with optimal joint ordering grows linearly with the number of bodies simulated.

Furthermore, the computational complexity of the sparse-block matrix multiplication is $\mathcal{O}(n_c + n_e)$, where n_c is the number of connections and n_e is the number of edges in a graph representation of a system. For a chain topology, the number of connections is the number of bodies minus 1, and the number of edges in the graph is the number of bodies minus 2. Therefore, the computational complexity of the matrix multiplication step also grows linearly with the number of bodies simulated.

For the chain topology case then, the two steps with cubically growing computational complexity with system size are reduced to having linear computational complexity with the methods presented in this thesis. This is the reason for the vast reduction in computation time seen during testing.

5.2.2 Lander Simulation

Just as for the chain topology systems, a variety of simulations were performed for the lander topology systems. Simulations were run for a number of system sizes using various numerical integration, matrix multiplication, and linear system solving methods.

Figure 5.12 shows testing results from a sweep of system size from 5 bodies to 41 bodies. Two methods were compared. The naive method uses naive matrix multiplication and LU-decomposition within the constraint controller. The improved method uses sparse-block matrix multiplication and banded Gaussian elimination within the constraint controller. All times were normalized by the fastest case, which occurred using the improved method for 5 bodies.

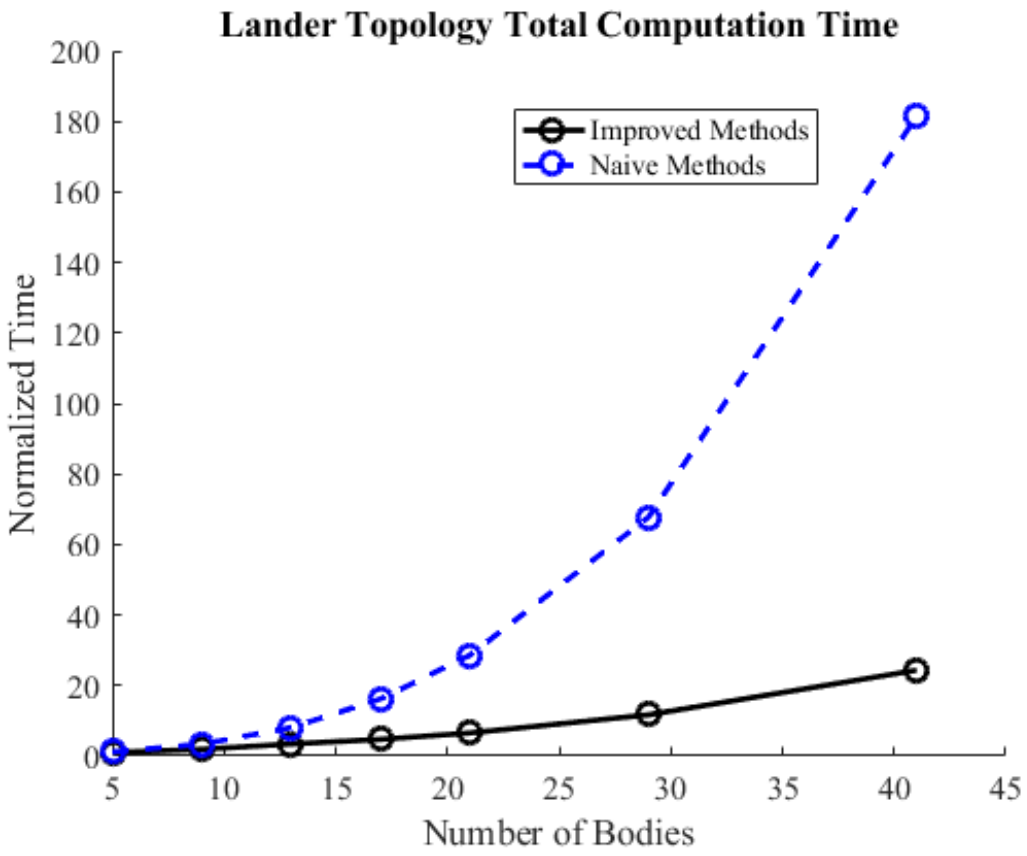

Figure 5.12: Normalized total computation time for lander type systems of size from 5 to 41 bodies

Just as for the chain topology simulations, the improvement methods grow more effective as the system size increases. This is due to the improved methods reducing computation time within steps that computationally scale poorly with system size. Figure 5.13 shows the computation time reduction using improved methods for various system sizes. While the gains are not as significant as for a chain topology system, they are still substantial.

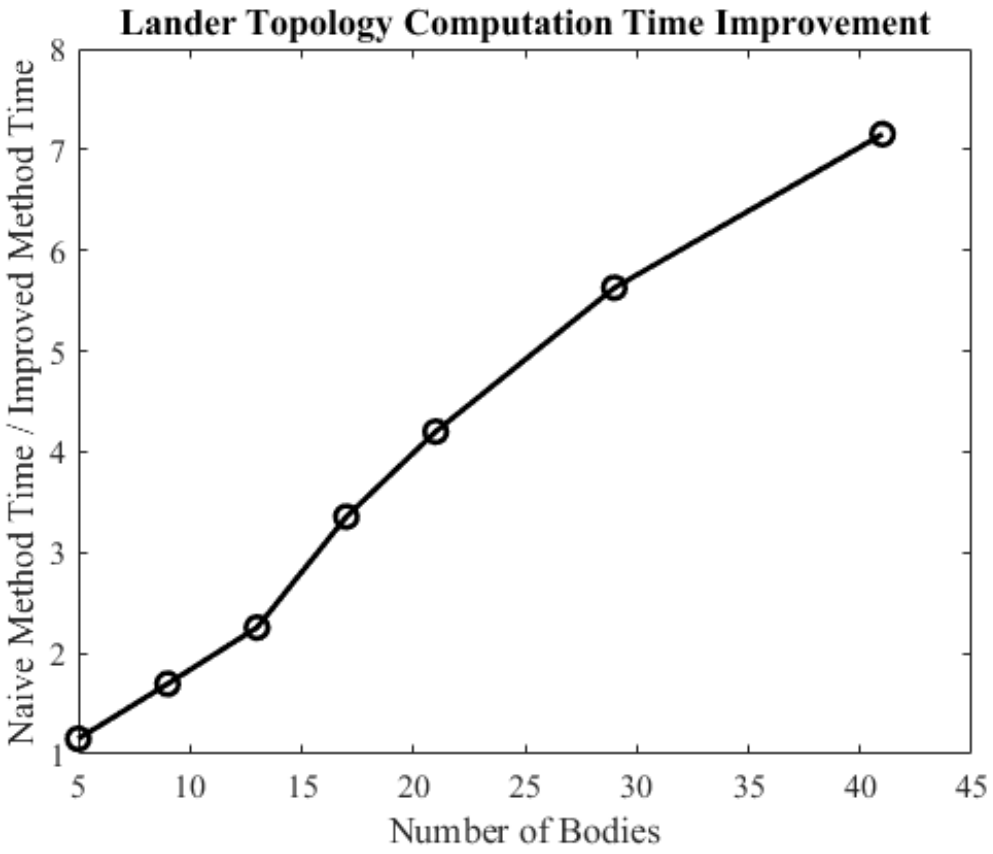

Figure 5.13: Computation time reduction for improved methods for lander type systems of size from 5 to 41 bodies

With naive methods, sections within the constraint controller quickly dominate total computation time as the system size increases. Figure 5.14 displays the percentage of total computation time used within the matrix multiplication and linear system solve steps in the constraint controller.

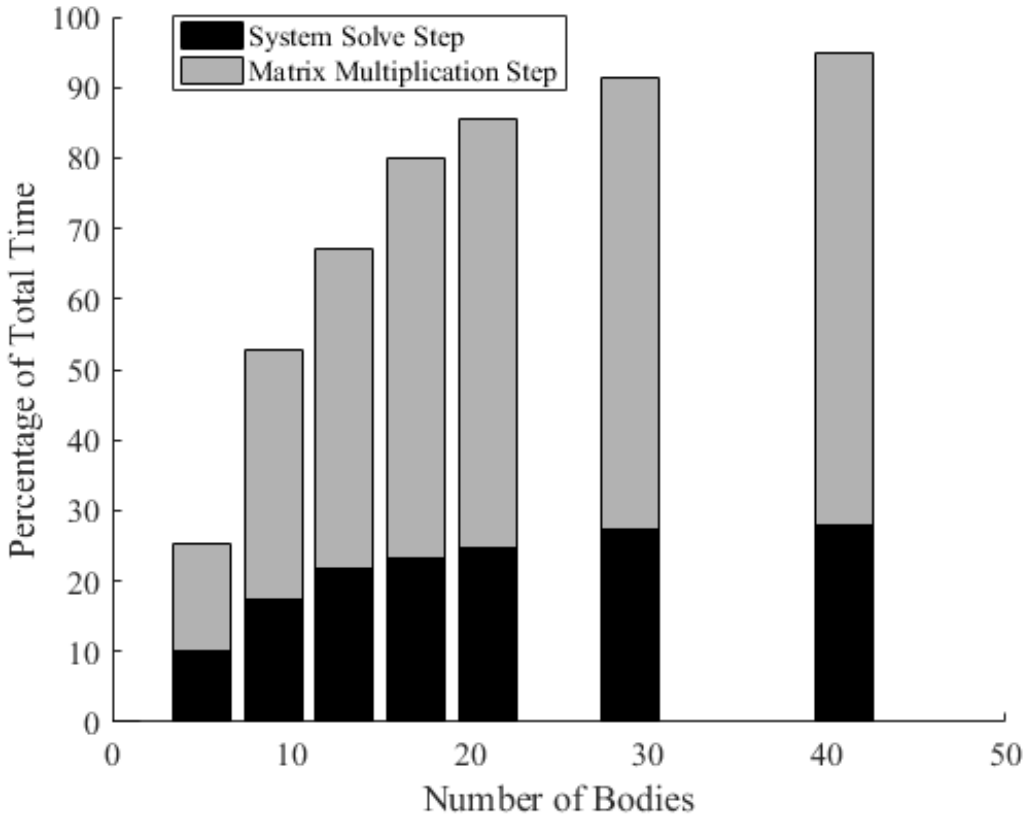

Figure 5.14: Percentage of total computation time used within constraint controller matrix multiplication and linear system solve steps using naive methods

With improved methods, the percentage of computation time spent within the constraint controller is reduced, as seen in Figure 5.15.

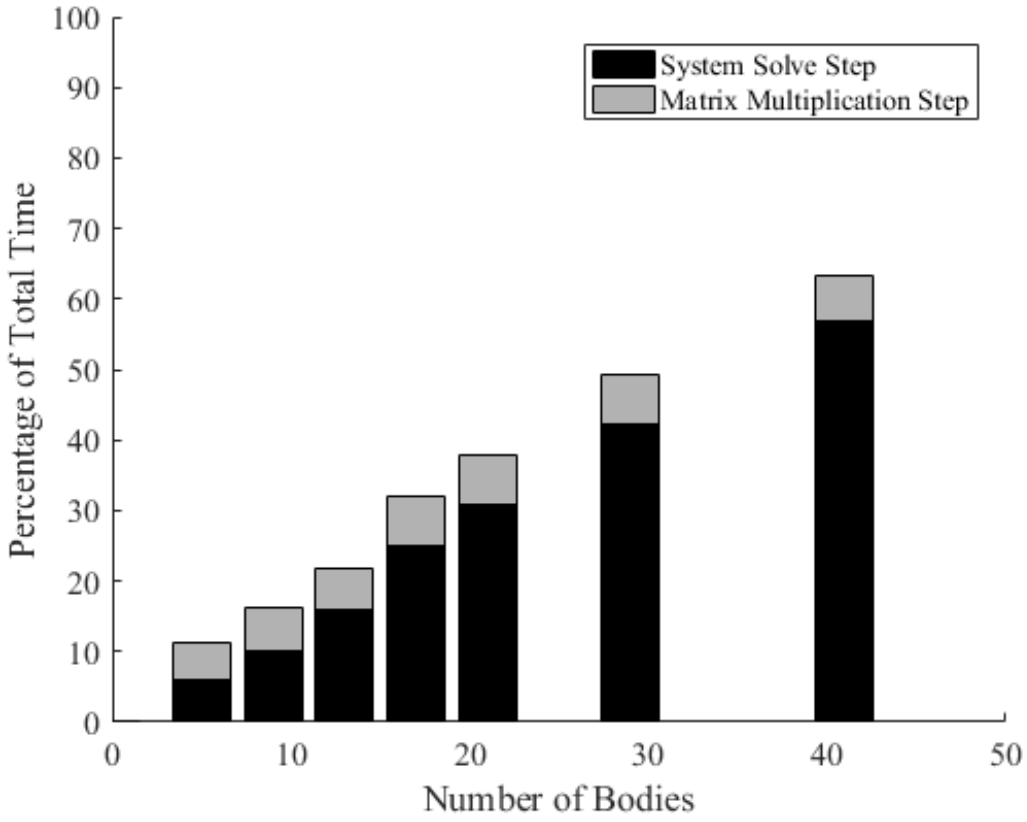

Figure 5.15: Percentage of total computation time used within constraint controller matrix multiplication and linear system solve steps using improved methods

While the improved methods used here clearly result in reduced computation time, the system solve step still takes a significant portion of the total computation time, much larger than for the equivalently sized chain topology system. For a lander system, the minimum linear system bandwidth that can be achieved is much larger than for a chain system, and so banded linear system solving methods are less effective.

Just as in the chain topology tests, the RKF45 integration method was used for all results in Figure 5.12. To compare how the RK4 method would perform for these simulations, other cases were run. These cases are described in Table 5.2.

Case #	Integrator	C.C. Matrix Multiplication Method	C.C. Linear System Solve Method
1	RKF45	Fortran matmul	PLU
2	RKF45	Sparse-block	Banded Gauss-Jordan
3	RKF45	Sparse-block	FOM
4	RKF45	Sparse-block	RNSD
5	RK4	Fortran matmul	PLU
6	RK4	Sparse-block	Banded Gauss-Jordan
7	RK4	Sparse-block	FOM
8	RK4	Sparse-block	RNSD

Table 5.2: Setup of each case presented in Figure 5.16

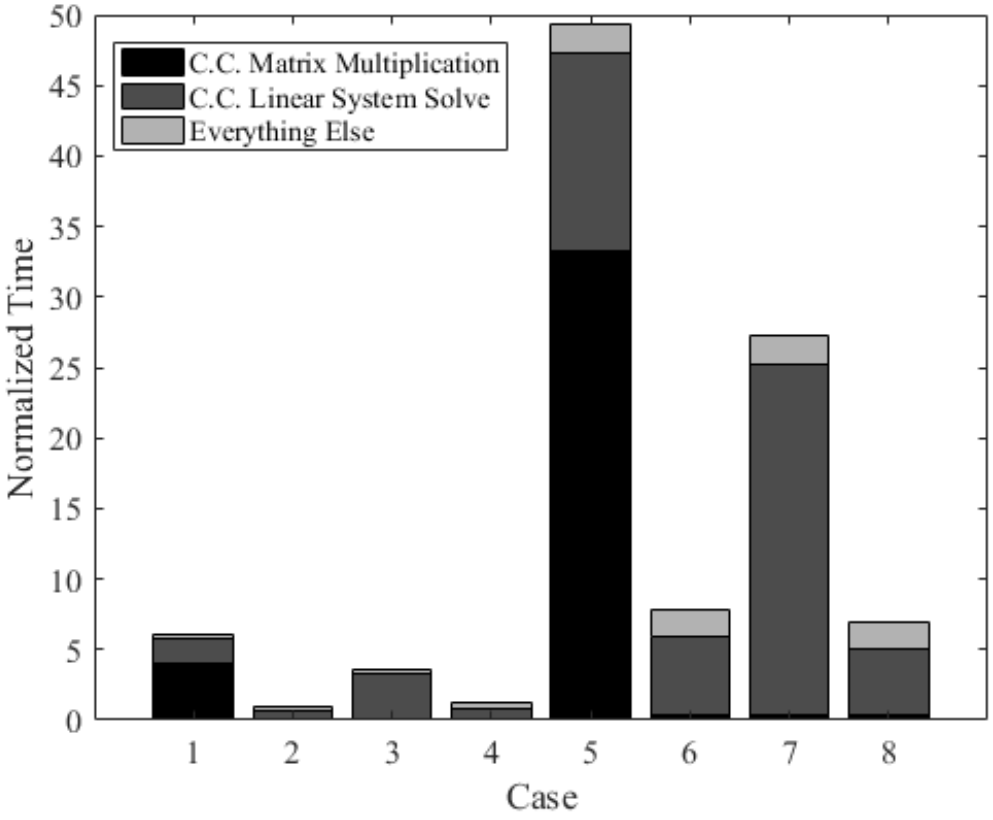

Figure 5.16: Normalized computation time required for a variety of cases for a lander topology system

5.2.3 Stubby Lander Simulation

Previously it was explained how splitting a body into two bodies and constraining them together could result in computation time improvements when combined with some of the methods developing in this thesis for improving the constraint controller. To examine this concept of splitting the bodies, an example with 40 legs was run. The naive case is one which is meant to directly represent the system, and is how an engineer would likely naively design the simulation. The system is modeled as a single central body with all 40 legs connected directly to it. In the split case, the central body is split into two identical bodies, connected to each other by a rigid 6 DOF joint. Each of the two central bodies then

has 20 legs connected to it, so that its $\widetilde{G}$ bandwidth is lowered by about a factor of 2. The cases run here are described in Table 5.3 and testing results are shown in Figure 5.17. All cases were simulated using RKF45.

Case #	C.C. Matrix Multiplication Method	C.C. Linear System Solve Method	Split
1	Fortran matmul	PLU	No
2	Fortran matmul	PLU	Yes
3	Sparse-block	Banded Gauss-Jordan	No
4	Sparse-block	Banded Gauss-Jordan	Yes

Table 5.3: Setup of each case presented in Figure 5.17

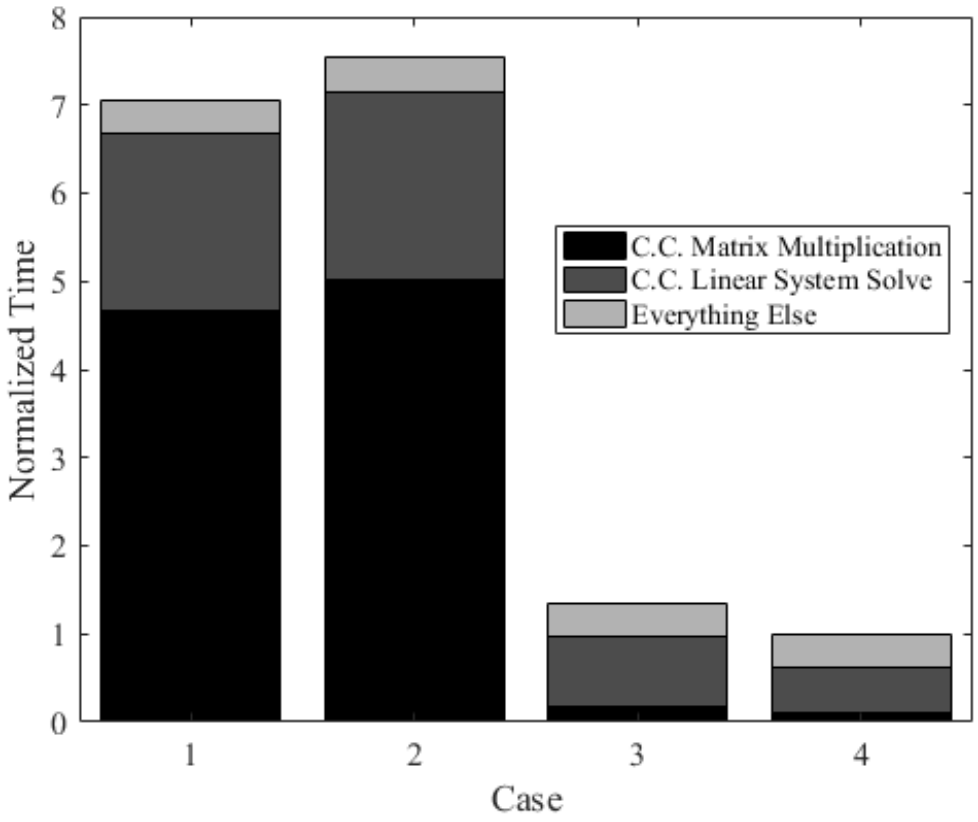

Figure 5.17: Normalized computation time required for a variety of cases for the stubby lander topology

When using naive methods for matrix multiplication and linear system solving, the split version is slower overall, which follows since this version has one extra body and one extra connection. The matrices and vectors in the constraint controller have larger dimensions, and computations involving them are slower. However, when sparse block matrix multiplication and a banded linear system solver are used, the split solution is overall faster than the original. Compared with the original using naive solvers, it is about seven times faster. When both the naive and split body cases are simulated with the improved computational methods, the split body case is about 25% faster than the naive setup.

The computation time reduction comes from both the matrix multiplication step and

the linear system solve step. Previously in this thesis, it was shown how the computational complexity of the sparse-block matrix multiplication method is $\mathcal{O}(n_c + n_e)$, where n_c is the number of connections and n_e is the number of edges in a graph representation of a system. The naive setup has 40 connections and, since its graph representation is complete, 780 edges. The split model has 41 connections, and each side of the system has a sub-graph that is complete. Since these sub-graphs are about half the size of the complete naive graph, the number of total edges is reduced to 421. This implies that we should expect the sparse-block matrix multiplication method to perform better on the split body setup than the naive version. In addition, since the matrix bandwidth of $\widetilde{G}$ is reduced by almost a factor of 2, banded linear system solving methods improve computation time for that step of the constraint controller as well. This theoretical result is confirmed in practice in Figure 5.17, where both of those sections within the constraint controller are faster for the split body case than for the naive case.

This result is significant since it shows that the computational speed of a simulation is not only dependent on the computational methods used, but also on how the engineer models the system to be simulated. Further, a seemingly identical system can have different computational challenges purely based on its mathematical representation. This concept is expanded in the next section, in which it is shown that the joint numbering scheme for a system can have a significant impact on computation time.

5.2.4 Chain Simulation Bandwidth Modification

Previous analysis in this thesis showed that the joint numbering scheme of a system can affect computation time. This section provides experimental results to confirm that theoretical development.

Simulations examined in this section cover a wide parameter space. All simulations consist of a chain topology system. Half of the simulations are for a system in which the joints are numbered sequentially, which minimizes the matrix bandwidth of $\widetilde{G}$. The other

half of the simulations are of a system which is numbered such that the matrix bandwidth of $\widetilde{G}$ is equal to its size, so banded linear system solving methods do not offer any benefit. For each of those ordering schemes, systems of varying size were simulated, between 15 and 50 bodies. Finally, for each system size and for each joint ordering scheme, simulations were run with naive and improved constraint controller computational methods. The naive methods are traditional "textbook" matrix multiplication, and LU-decomposition with pivoting for solving the linear system. The improved methods used were sparse-block matrix multiplication and banded Gaussian elimination for solving the linear system.

These simulations were all run using the RKF45 integration method. They have the same initial conditions (they are, of course, physically identical). They also end the simulations with the same final states, showing that the results are not affected by the methods used to improve computation time.

First, examine the computation time for the cases that used naive computational methods. The two systems are physically identical, and only differ by their joint numbering, which is purely mathematical bookkeeping. The computational methods used are not affected by the joint ordering, and this results in no change in computation time under joint reordering. Of course, computation time increases quickly with the increasing number of bodies simulated. The normalized computation times for these simulations are shown in Figure 5.18. All times were normalized by the overall best computation time, which occurred when the simulation was run with 15 bodies and all computational improvement methods enabled.

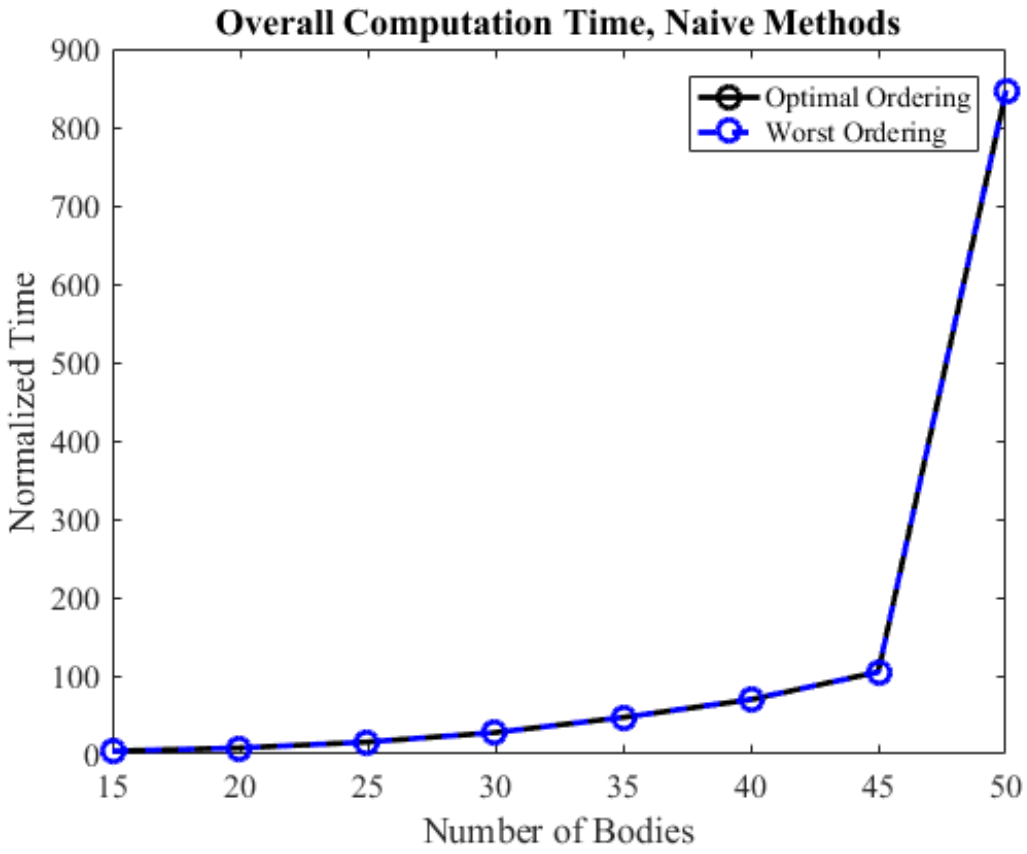

Figure 5.18: Normalized total computation time for joint reordering test cases using naive computational methods

It is not entirely known why the computation time increases so much between 45 and 50 bodies. However, it is thought that this is due to limitations of the cache size of the CPU used to perform this simulation. When examining only the matrix multiplication portions of these cases, as in Figure 5.19, the same sharp increase in computation time between 45 and 50 bodies is seen. It is beyond the scope of this thesis to discuss optimization of a simulation based on CPU cache size, but it is important to note that cache size limitations can have a significant impact on simulation times. It is also worth noting that the sparse-block matrix multiplication method could be performed using significantly less memory than full multiplication, which could offer further computation time improvements. There

is evidence of this in these simulations. When sparse-block matrix multiplication was used, the same sharp uptick in computation time between 45 and 50 bodies was not seen.

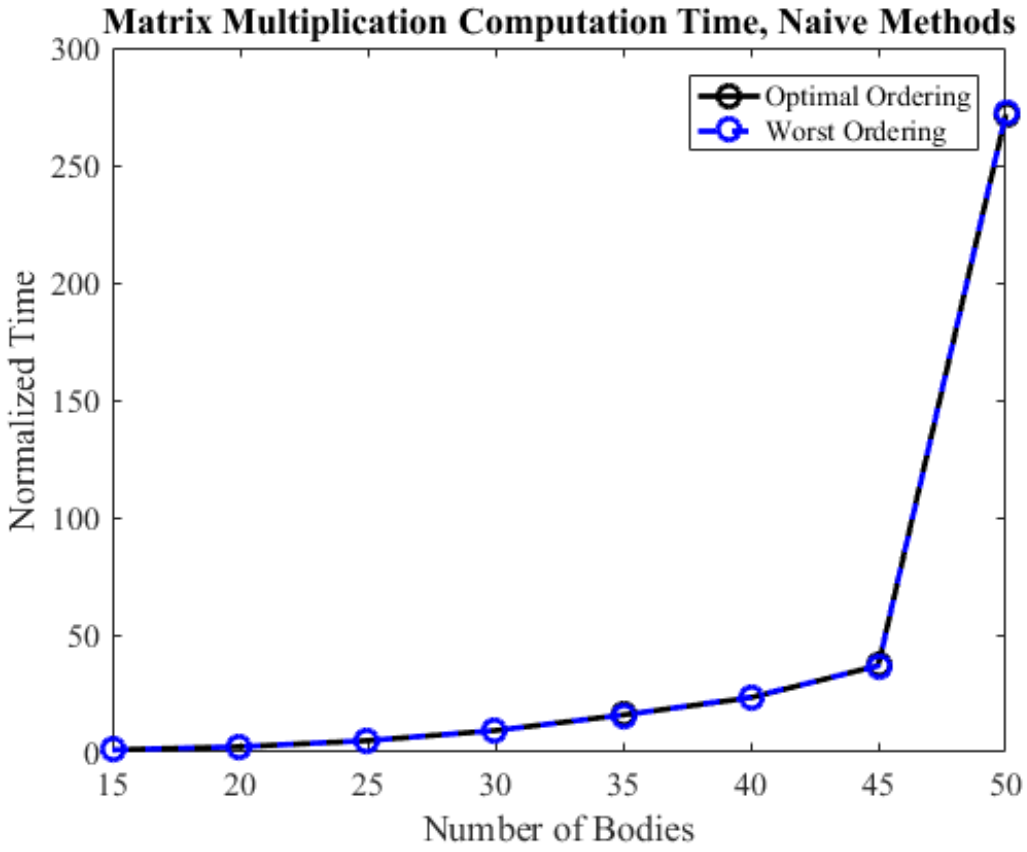

Figure 5.19: Normalized $\widetilde{G}$ matrix multiplication computation time for joint reordering test cases using naive computational methods

These experimental results confirm what was expected. Namely, that joint reordering has no affect on computation time if naive computational methods are used. Only when improved methods are used is there a need to carefully number joints. This fact is shown in Figure 5.20. The computation times shown there were normalized by the same case as for the naive methods shown in Figure 5.18, and can be directly compared.

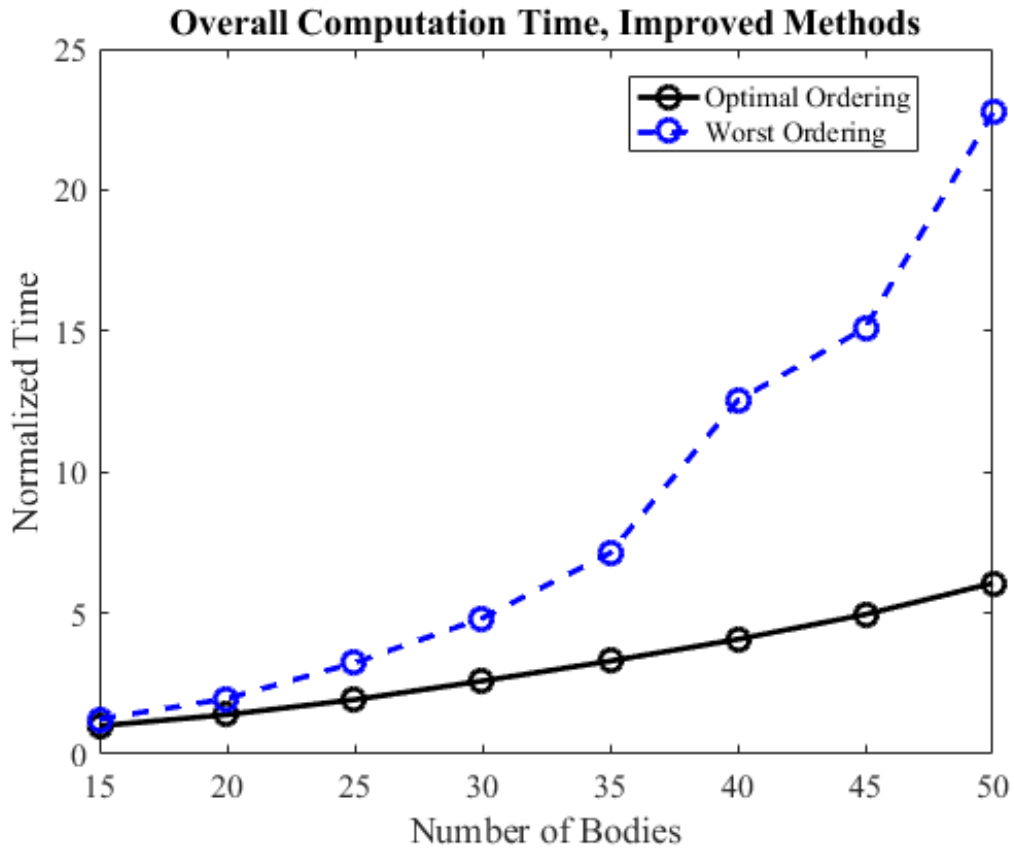

Figure 5.20: Normalized total computation time for joint reordering test cases using improved computational methods

Clearly, for this system topology, joint ordering is important. The optimal ordering offers significant computational time advantages over the worst ordering, due to the reduced computational time in the linear system solve step. Overall, with 50 bodies, the optimal ordering offers computational time reduction of about 4-fold compared with the worst ordering.

All together, a 50-body chain-topology system with optimal joint ordering can be simulated using the improved computational methods over 140-times faster than the same system using naive computational methods, without any loss of accuracy.

Figure 5.21 shows the total time spent in the linear system solving section of the con-

straint controller for each case. The times shown are normalized by the overall best time, which occurred for the optimal ordering with 15 bodies.

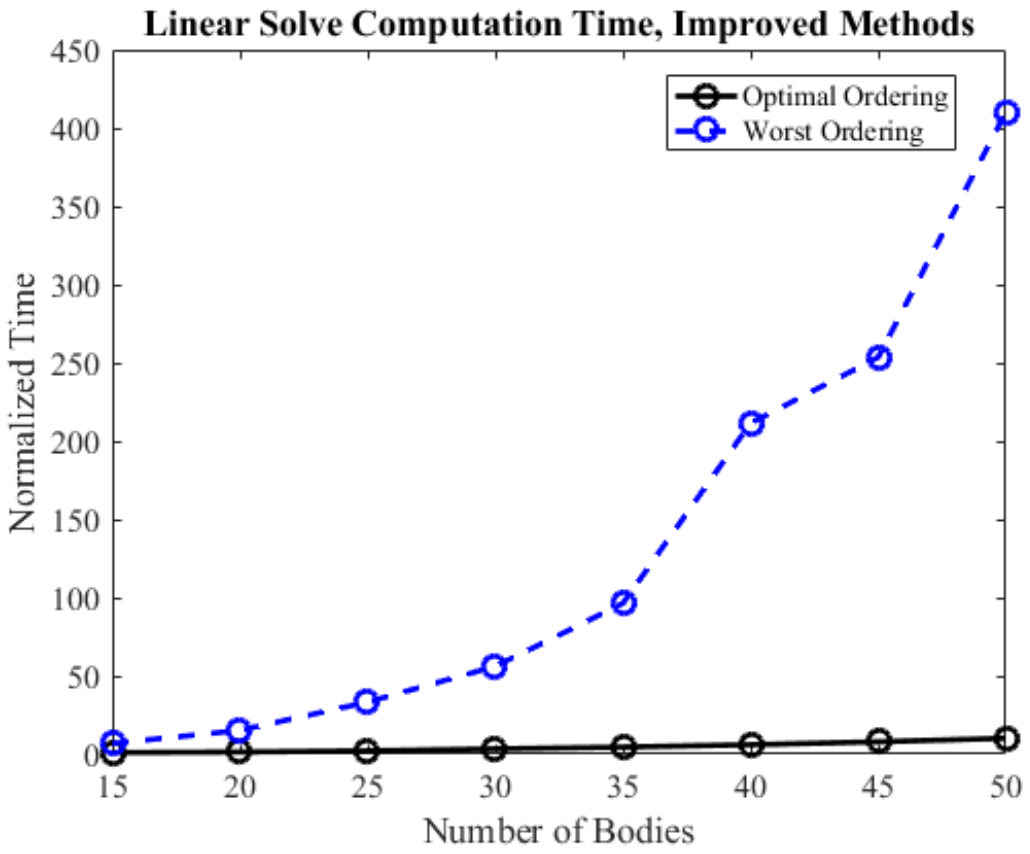

Figure 5.21: Normalized total linear solve computation time for joint reordering test cases using banded Gaussian elimination

As the number of bodies increases from 15 to 50, the overall time spent in the linear system solving section of the constraint controller increases by a factor of about 10 in the optimal numbering case, and by a factor of about 410 in the worst numbering case. Clearly, the joint numbering of a system can be critically important for reducing computation time.

Computation time improvements can be lumped here into two categories: those due to algorithmic improvements, and those due to tuning the layout of a system to perform well with those algorithmic improvements.

Therefore, the engineer developing a simulation using the NLCT multibody dynamic simulation algorithm cannot rely solely on using intelligent computation techniques, but must also take care to set up the simulated model so that the computational techniques can perform at their best. Failure to do so can result in significant lost time.

To achieve the best performance, algorithms should be implemented to perform joint reordering for bandwidth reduction at the beginning of a set of simulations. As shown earlier, this problem is NP-complete. However, some intelligent brute force methods such as MB-ID can sometimes find the optimal joint ordering reasonably quickly. For larger systems, polynomial time approximate methods such as RCM can be used. Since this is purely a joint reordering step, it only needs to be performed once for any system topology, regardless of actual system initial conditions or physical properties. In other words, if a system is part of a Monte-Carlo simulation with thousands of cases, the joint reordering algorithm need only be run once, and the resulting joint ordering can then be used throughout.

CHAPTER 6
EUROPA LANDER SIMULATION

6.1 Motivation

Europa, a moon of Jupiter that is slightly smaller than Earth's moon, is of interest to scientists due to its water ice surface and suspected sub-surface salty ocean [11]. These features, along with possible hydrothermal activity on the seafloor, may make it a suitable candidate for harboring life. Because of this, NASA is in the evaluation process for a Europa lander mission.

However, not much is known about the surface of Europa on the scale of such a lander. The best images we have of the icy moon's surface come from NASA's Galileo mission. Even the best images from that mission show a relatively low amount of detail. In fact, the highest resolution image of Europa from Galileo was taken with a scale of 6 meters per pixel, and is shown in Figure 6.1.

While even that image does not provide sufficient resolution at the lander scale, the local terrain of the rest of the surface of Europa is even more unknown. NASA estimates that only about 10% of the surface of Europa has been imaged at a scale better than 300 meters per pixel [13].

Because of the relatively poor resolution of images of Europa's surface, any obstacles or terrain on the scale of a lander are completely hidden. To help combat this issue, a flyby mission has been proposed by NASA that would be able to collect data at a higher resolution. This mission, called the "Europa Multiple Flyby Mission" (EMFM), would provide images of certain locations of interest at a scale of about 50 cm per pixel [13]. Of course, even at that scale, objects that would interfere with landing could still be hidden. Furthermore, parts of the surface of Europa are thought to change on the timescale of these

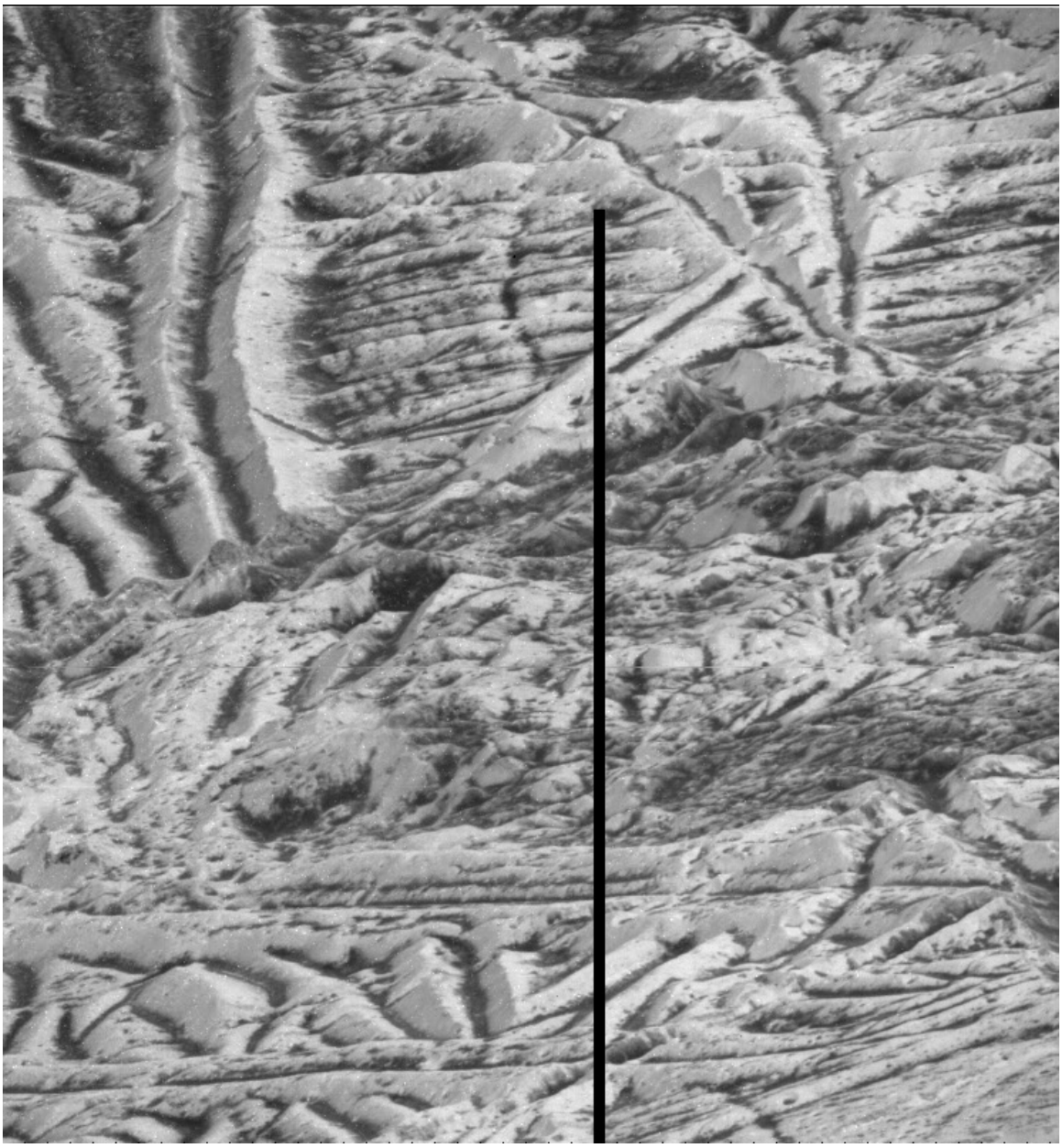

Figure 6.1: Highest resolution image of the surface of Europa taken by Galileo. Scale is 6 meters per pixel. The black bar in the image is due to missing data that was not sent by Galileo [12]

missions. An area that is suitable as a landing site during the flyby mission may not be the same when the lander arrives years later. It is therefore important that the lander can adapt to a variety of terrains.

There are many factors to consider when choosing the optimal landing location for a mission to Europa. Not only must the landing location have suitable terrain to allow the lander to touch down undamaged, but the location must also be scientifically compelling and have sufficiently low radiation levels, among other requirements. Therefore, by expanding the envelope of types of terrain that are suitable for the lander, it may be possible to choose a landing location that is better for performing scientific measurements. Performing a successful landing on uncertain terrain requires a system that is insensitive to local terrain features. A lander with flexible, passive legs could provide some of this insensitivity. Such a lander might be able to compensate for unknown variations and ensure a safe landing. To evaluate if flexible legs would be beneficial for a Europa lander, a simulation was developed.

6.2 Simulation Design

To evaluate the possible performance benefits of a flexible legged lander design versus a more traditional fixed leg design, a multibody dynamic simulation was developed. The NLCT constrained coordinate method described previously was used, along with the best of the computational improvement techniques developed earlier in this thesis. The physical design of the lander was based on the proposals from NASA reports on the topic.

Of particular importance to the simulation is the ground contact model. Very little is known about the tribological properties of the surface of Europa, which makes accurate simulation challenging. The uncertainty in ground contact interactions is handled by using the LuGre friction model, which can model a wide variety of tribological effects, and by performing a large number of simulations with a wide range of parameters.

6.2.1 Lander Design

The design of the simulated lander was inspired by the preliminary design proposed in the Europa Study 2012 Report, created by NASA/JPL-Caltech. This report proposed a lander with 6 stabilizer legs, with a foot-to-foot distance of 4 meters. The lander was hexagonal with a width of 1.5 meters and a height of 0.9 meters. This design is seen in figures 6.2 and 6.3. It was predicted to have a wet mass of 900 kg [11]. These parameters were used as the baseline design.

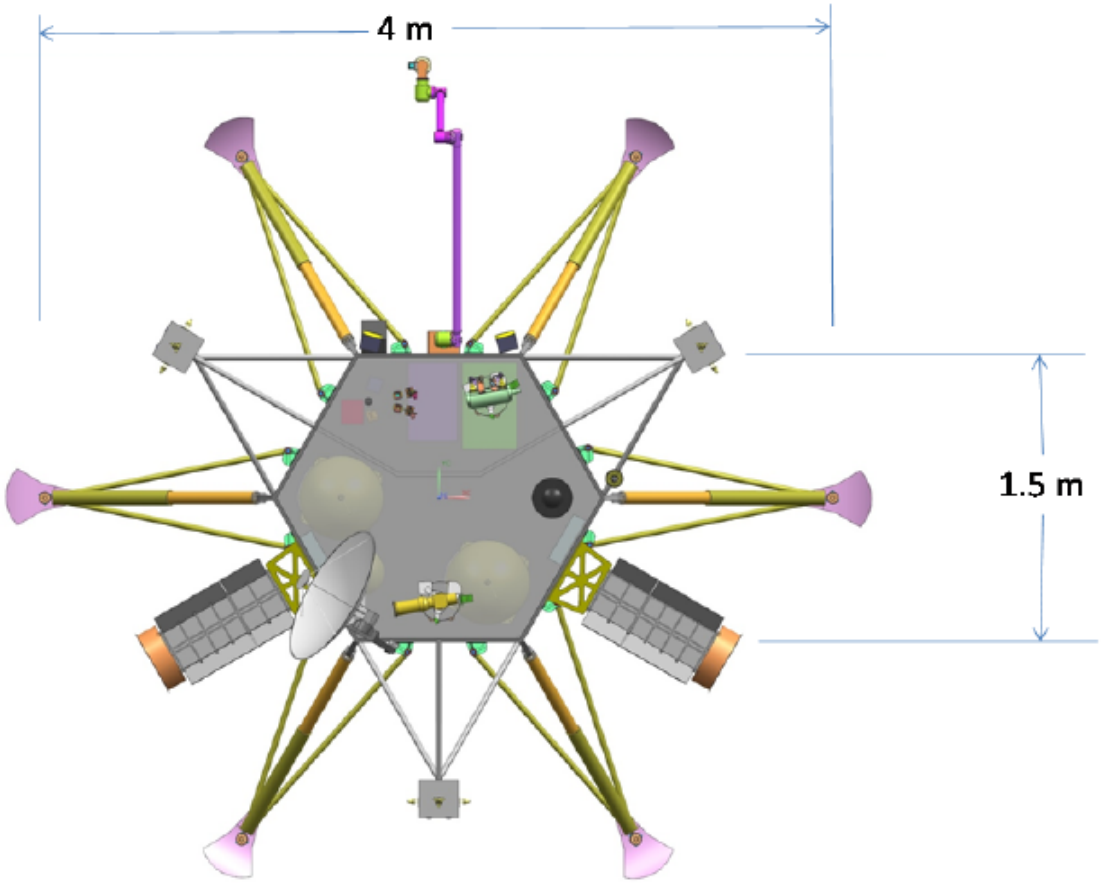

Figure 6.2: Top view of proposed lander from 2012 NASA/JPL-Caltech report [11]

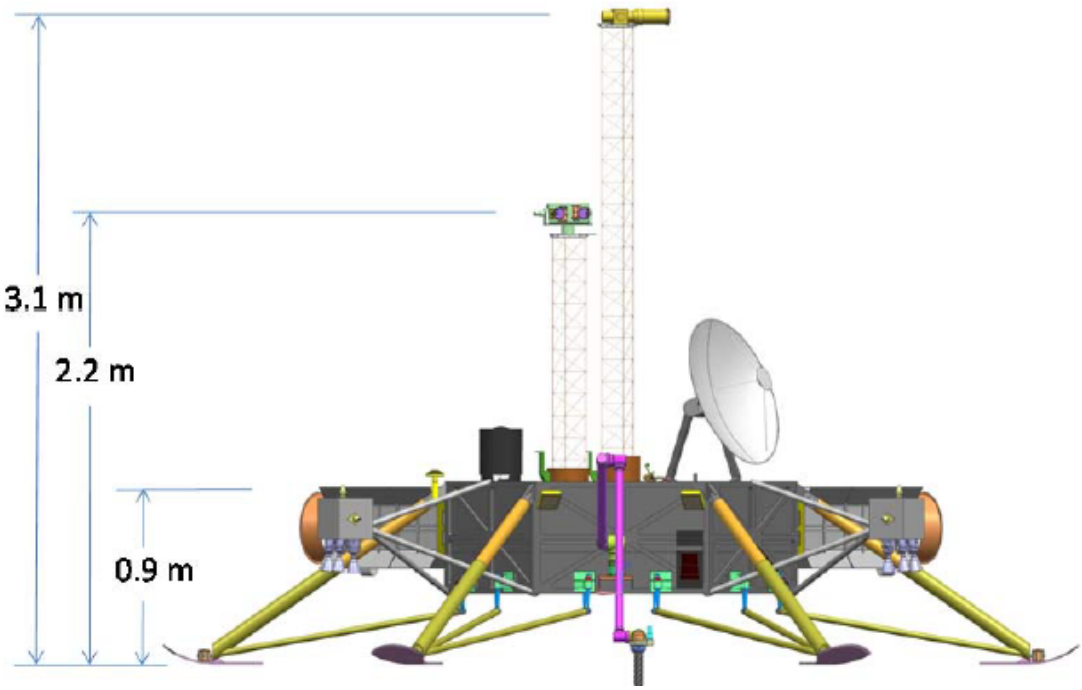

Figure 6.3: Side view of proposed lander from 2012 NASA/JPL-Caltech report [11]

Four different systems were simulated. They were nominally of the same size and mass of the lander proposed by NASA and each had 6 stabilizer legs. The legs on each system were simulated as two rigid bodies per leg. Each leg had a hip and a knee joint - both modeled as hinge joints. Namely, this means that the joints were rigid in all but one degree-of-freedom (DOF). In that one DOF, a rotation, the joint was modeled as a rotational linear spring-damper system. In the first system, these parameters were set such that the legs were effectively rigid. In the second system, the legs were flexible, but still able to support the lander body from touching the surface when stationary. This was called the shock-absorbing model. The legs of the third system were even more flexible, and could not support the weight of the lander, so the lander comes to rest with the bottom of the lander body on the surface. This was called the conforming model. The conforming model is shown in Figure 6.4. The fourth system was simulated with a basic active impedance control method. Nominally, the stiffness of its legs was the same as for the shock-absorbing

model. However, upon landing, if not all of its legs were contacting the ground, the stiffness parameters of the contacting legs were modified to attempt a softer landing. The logic for the impedance control is shown in Figure 6.5.

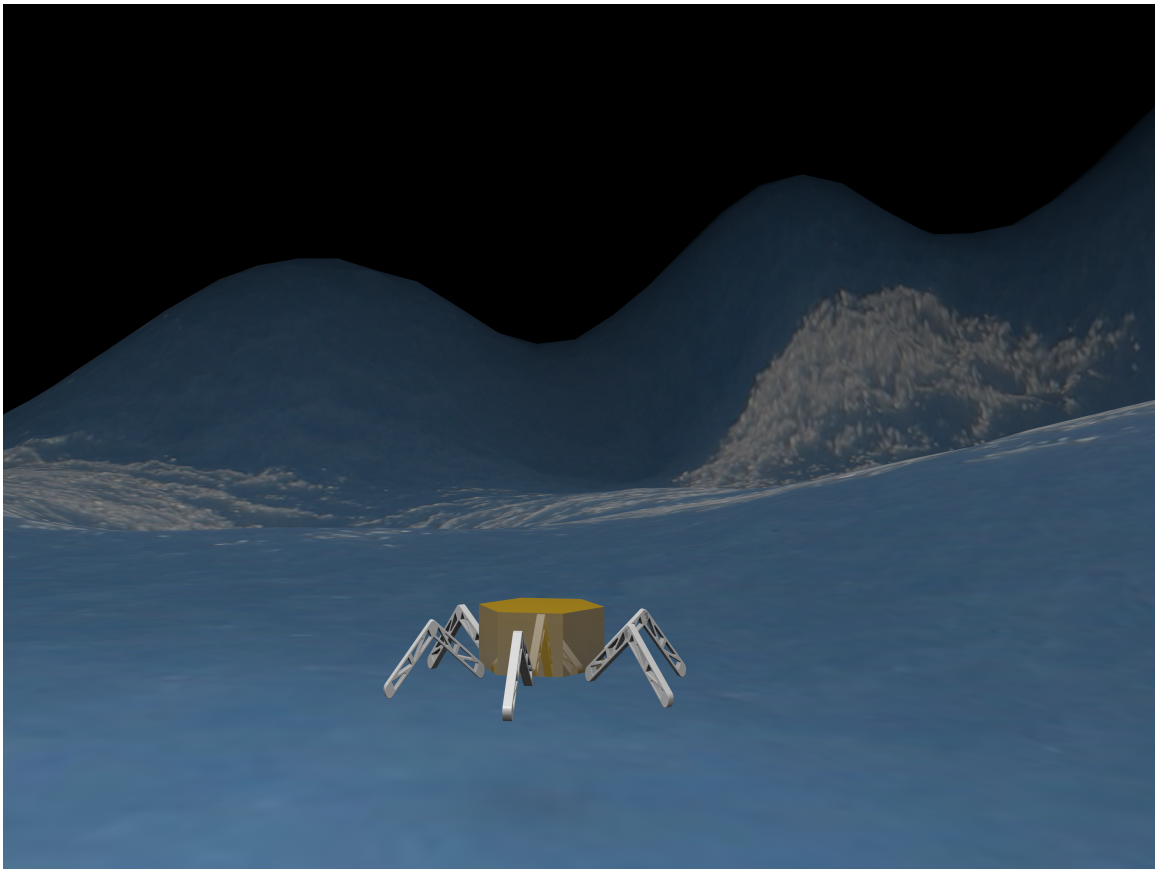

Figure 6.4: Basic design for simulated flexible legged lander

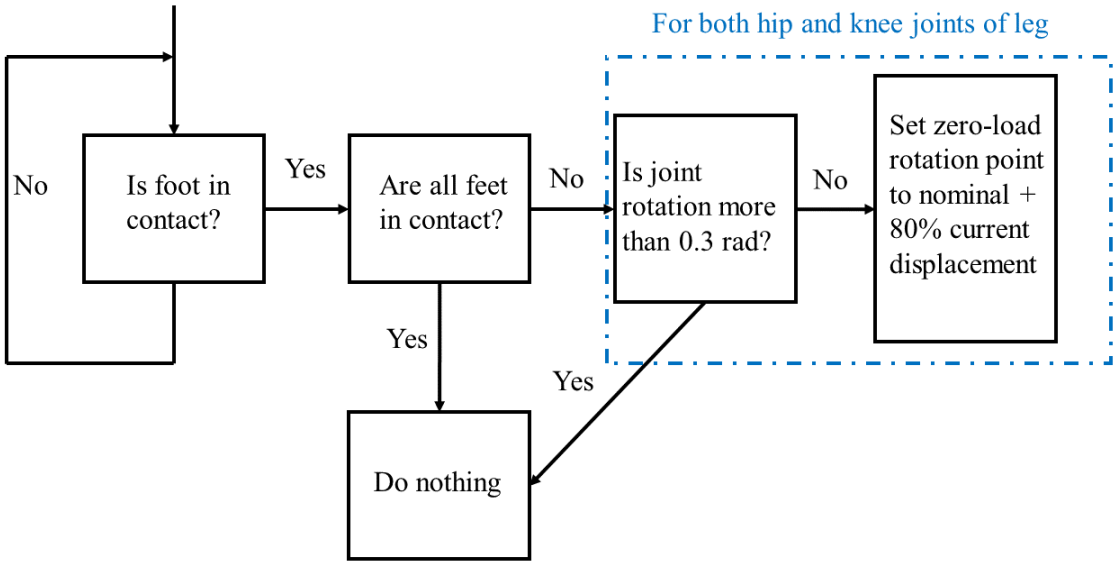

Figure 6.5: Impedance controller logic

It should be noted that in the NASA/JPL-Caltech Europa Lander Study 2016 Report the proposed design for the lander is somewhat different. It uses 4 stabilizer legs which can conform to the terrain. The artist's representation of this design is shown in Figure 6.6. While this concept is fairly similar to the use of 6 flexible legs on the simulated lander, there are some differences. Specifically, the flexible and controlled legs in the simulated lander are not intended to only stabilize the lander on a variety of terrain, but to also ensure successful and undamaged landing over a wider variety of terminal velocities and orientations.

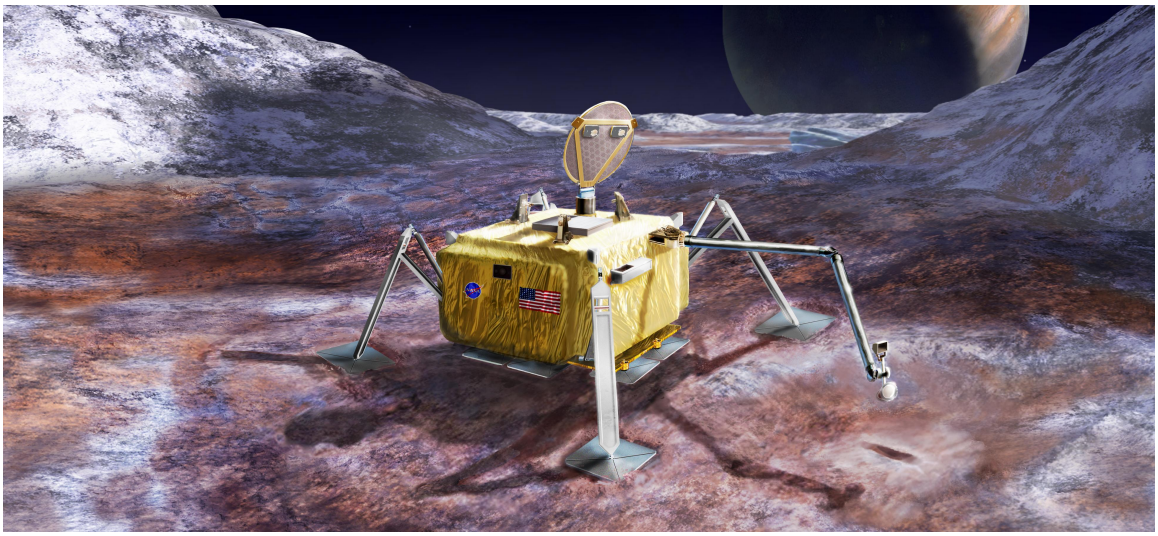

Figure 6.6: Artist's rendering of proposed lander from 2016 NASA/JPL-Caltech report [14]

6.2.2 Ground Contact Modeling

6.2.2.1 Normal Interaction - Standard Linear Solid

The contact interactions normal to the ground are simulated as a standard linear solid model. This model consists of two stages. The first stage consists of a linear spring, while the second stage consists of a linear spring in parallel with a linear damping element [15].

$$F_n = k_e \epsilon_e = k_v \epsilon_v + c_v \dot{\epsilon_v} \tag{6.1}$$

$$\epsilon = \epsilon_e + \epsilon_v \tag{6.2}$$

$$\dot{F_n} = -\frac{k_e + k_v}{c_v} F_n + \frac{k_e k_v}{c_v} \epsilon + k_e \dot{\epsilon} \tag{6.3}$$

6.2.2.2 *Tangential Interaction - LuGre Model*

A representative friction model is necessary for accurate simulation of ground impact scenarios. For the simulation of the Europa lander, the LuGre friction model was used. This is a dynamic friction model that accounts for a wide variety of phenomena, such as stick-slip, velocity dependence, and the Stribeck effect [16].

The LuGre friction model has an internal state z, which is dependent on the relative velocity v of the two contact surfaces. It is described by

$$\dot{z} = v - \sigma_0 \frac{|v|}{g(v)} z \tag{6.4}$$

The velocity dependent function $g(v)$ is often given as

$$g(v) = \mu_k + (\mu_s - \mu_k) e^{-|\frac{v}{v_s}|^\alpha} \tag{6.5}$$

The parameter α varies in the literature, usually between 0.5 and 2 [16]. In this simulation, a value of 1 was used.

From the internal friction state z, its time derivative $\dot{z}$, and the relative velocity v between the contact surfaces, and the normal force between the two surfaces, the tangential frictional force is calculated as

$$F_t = |F_n|(\sigma_0 z + \sigma_1 \dot{z} + \sigma_2 v) \tag{6.6}$$

For this model to provide accurate results of ground contact interactions, the parameters σ_0, σ_1, σ_2, μ_s, μ_k, and v_s must be chosen carefully. Often, these values are chosen using experimental data. For a theoretical lander on Europa, clearly this is not an option. These values must then be chosen intelligently in another way.

The first step in choosing LuGre friction model parameters for this simulation is reducing the size of the parameter space. The parameter σ_1 mainly represents damping under

micro-displacement. It is therefore most important for systems where accurate prediction of movement on the micro and nano scale is important [16]. For a lander, this is not necessary. This parameter can therefore be made dependent on other parameters to reduce the size of the parameter space [16]. σ_1 is made dependent on σ_0, σ_2, some effective mass m_e and a damping parameter ζ.

$$\sigma_1 = 2\zeta\sqrt{\sigma_0 m_e} - \sigma_2 \tag{6.7}$$

For this lander simulation, m_e is taken as one-sixth of the nominal lander mass, since there are six legs on the lander. ζ was set to 1 to critically damp the micro-movement behavior.

Now consider steady-state velocity situation with velocity v_{ss}. In this case, the internal state reaches an equilibrium as its time-derivative is zero.

$$0 = v_{ss} - \sigma_0 \frac{|v_{ss}|}{g(v_{ss})} z_{ss} \tag{6.8}$$

Rearranging, the steady-state internal state z_{ss} is

$$z_{ss} = \frac{v_{ss} g(v_{ss})}{|v_{ss}|\sigma_0} = sgn(v_{ss}) \frac{g(v_{ss})}{\sigma_0} \tag{6.9}$$

The total frictional force is then determined by

$$F_{t,ss} = |F_{n,ss}|(sgn(v_{ss})g(v_{ss}) + \sigma_2 v_{ss}) \tag{6.10}$$

In this case, with the normal force held constant, the function $g(v)$ provides an approximation of the Stribeck effect, with the friction coefficients μ_s and μ_k corresponding to friction coefficients in the Coulomb friction model for a system at rest and in motion, respectively. The parameter v_s controls how fast the transition between those parameters occurs. In addition, there is an extra purely velocity dependent frictional force determined

by σ_2.

By this analysis, μ_k and μ_s can be chosen by considering the simpler and widely used Coulomb friction model, and choosing approximate experimentally determined friction coefficients for a variety of surface type pairs. The parameter v_s is then chosen to determine where the transition between those parameters should occur. Similarly, σ_2 is chosen to determine the purely velocity dependent frictional force desired.

The final parameter to determine is σ_0. This parameter mainly determines micro-displacement interactions. Consider the LuGre model under a zero-slip condition. An external force F_{ext} is applied to the system, and the velocity is low, so the viscous friction determined by σ_2 is ignored. The system is stationary on a surface, and so the normal force is assumed equal to the gravity force.

$$m_e\dot{v} = F_{ext} - F_t \tag{6.11}$$

$$\dot{z} = v - \sigma_0\frac{|v|}{g(v)}z \tag{6.12}$$

$$F_t = m_e g(\sigma_0 z + \sigma_1\dot{z}) \tag{6.13}$$

When these equations are linearized at $z = 0$ and $v = 0$, (6.12) produces

$$\dot{z} = v \tag{6.14}$$

In this situation, it is assumed that the system is at rest before the external force is applied, and so

$$z = x \tag{6.15}$$

Here, x is the micro-displacement of the system. Therefore, since $v \approx 0$, σ_0 therefore

corresponds to a spring force for this micro-displacement.

$$F_t \approx m_e g \sigma_0 x \tag{6.16}$$

6.3 Simulation Results

Since very little is known about the surface of Europa, a wide variety of simulations were completed in order to evaluate a lander's performance under significant uncertainty. Overall, there were three types of ground evaluated. First, the ground contact parameters were tuned to represent landing on an icy surface. The lander system was expected to slide significantly, and the surface is relatively hard. The second case represents a snowy surface. This surface is softer than ice, but still fairly slippery. Finally, the last case was intended to represent landing in a sand-like substance. This surface is somewhat softer than ice or snow, but very little sliding is expected. Within these three cases, nominal system parameters were chosen, and many of them were varied according to a normal distribution, to cover a wide variety of landing scenarios.

All simulations had the same nominal lander physical configuration, shown in Table 6.1. Parameters that were not varied are indicated by "N/A" for their standard deviation. All six legs on a given system had the same initial upper and lower leg angles for any simulation, but those angles were varied between simulations.

Parameter	Nominal Value	Standard Deviation	Unit
Total Mass, m	900	9	kg
Inertia, I_{xx}	151	7.5	$kg \cdot m^2$
Inertia, I_{yy}	151	7.5	$kg \cdot m^2$
Inertia, I_{zz}	281	14	$kg \cdot m^2$
Inertia, I_{xy}	0	N/A	$kg \cdot m^2$
Inertia, I_{xz}	0	N/A	$kg \cdot m^2$
Inertia, I_{yz}	0	N/A	$kg \cdot m^2$
Upper Leg Section Angle	65	3	deg
Upper Leg Section Length	0.63	N/A	m
Lower Leg Section Angle	94	5	deg
Lower Leg Section Length	0.63	N/A	m
Horizontal Impact Velocity, $\dot{x}$,	1.5	1.5	m/s
Horizontal Impact Velocity, $\dot{y}$,	1.5	1.5	m/s
Vertical Impact Velocity, $\dot{z}$,	3	1.5	m/s
Initial Orientation, ϕ	0	11.5	deg
Initial Orientation, θ	0	11.5	deg
Initial Orientation, ψ	0	5.7	deg

Table 6.1: Nominal lander physical parameters used in all simulations

For every ground contact setup, cases were run with legs of all three passive stiffness levels as well as with the impedance controller. Rotational stiffness and damping parameters for the unconstrained rotational DOF on each joint were set to switch between these cases. For the legs of the conforming model, the joint parameters are shown in Figure 6.2.

Parameter	Nominal Value	Standard Deviation	Unit
Hip Stiffness	210	10.5	$N \cdot m/rad$
Hip Damping	50	2.5	$N \cdot m \cdot s/rad$
Knee Stiffness	370	18.5	$N \cdot m/rad$
Knee Damping	50	2.5	$N \cdot m \cdot s/rad$

Table 6.2: Nominal conforming leg joint parameters

The joint parameters for the shock-absorbing model are shown in Figure 6.3. These are also the nominal values used for the model with impedance control.

Parameter	Nominal Value	Standard Deviation	Unit
Hip Stiffness	2100	105	$N \cdot m/rad$
Hip Damping	390	19.5	$N \cdot m \cdot s/rad$
Knee Stiffness	3700	185	$N \cdot m/rad$
Knee Damping	390	19.5	$N \cdot m \cdot s/rad$

Table 6.3: Nominal shock-absorbing and impedance-controlled leg joint parameters

Finally, the joint parameters for the stiff legged model are shown in Figure 6.4.

Parameter	Nominal Value	Standard Deviation	Unit
Hip Stiffness	8.4e4	4.2e3	$N \cdot m/rad$
Hip Damping	390	19.5	$N \cdot m \cdot s/rad$
Knee Stiffness	8.4e4	4.2e3	$N \cdot m/rad$
Knee Damping	520	26	$N \cdot m \cdot s/rad$

Table 6.4: Nominal stiff leg joint parameters

6.3.1 Icy Surface

To simulate an icy surface, ground contact parameters were chosen to allow the system to slide significantly after contact. These parameters are shown in Table 6.5.

Parameter	Value	Unit
SLS Spring, k_e	22000	N/m
SLS Spring, k_v	920	N/m
SLS Damping, c_v	5550	$N \cdot s/m$
LuGre Parameter, μ_s	0.09	Unitless
LuGre Parameter, μ_k	0.05	Unitless
LuGre Parameter, σ_0	0.0015	Unitless
LuGre Parameter, σ_1	0.1	Unitless
LuGre Parameter, σ_2	0.05	Unitless
LuGre Parameter, v_s	0.2	m/s
LuGre Parameter, α	1	Unitless

Table 6.5: Icy surface ground contact parameters

400 cases were run for each leg stiffness setup for the landing on an icy surface. These simulations determined that the shock-absorbing legs resulted in the lowest peak lander body acceleration, followed by the impedance controlled legs, then the conforming legs, and then the stiff legs. The shock-absorbing legs resulted in a mean peak acceleration reduction of about 42% compared to the stiff legs. These results are shown in Table 6.6.

Leg Type	Peak Lander Acceleration, Mean, $[m/s]$	Peak Lander Acceleration, Standard Deviation, $[m/s]$
Stiff	23.8	12.0
Shock-absorbing	13.7	8.0
Conforming	17.7	7.1
Impedance Control	17.3	6.8

Table 6.6: Icy surface lander body acceleration statistics

Since the initial conditions of the lander were varied significantly to generate a range of impact velocities and angles, the distribution of peak lander acceleration values is fairly wide. Figure 6.7 shows that the systems with stiff legs nonetheless tended to experience higher peak acceleration upon impact.

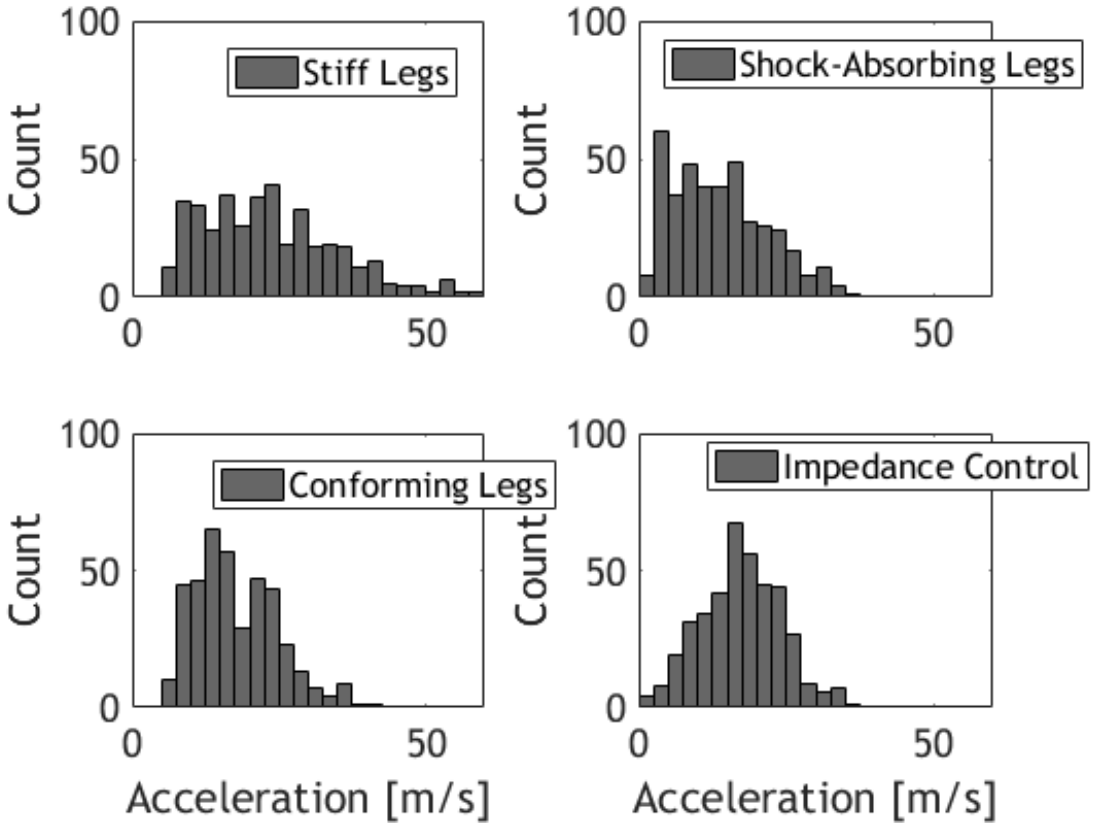

Figure 6.7: Peak lander body acceleration distribution for the various systems landing on an icy surface

In order for the legs to be effective, the system must survive impact. Meanwhile, unnecessary weight on a spacecraft is unacceptably expensive to launch, so the legs must be as light as possible while maintaining sufficient strength to survive landing. It is therefore beneficial to lower the peak loads that the legs must endure. Figure 6.8 shows the magnitude of the peak joint forces endured by each of the 12 joints on each system, normalized by the total lander weight on Europa.

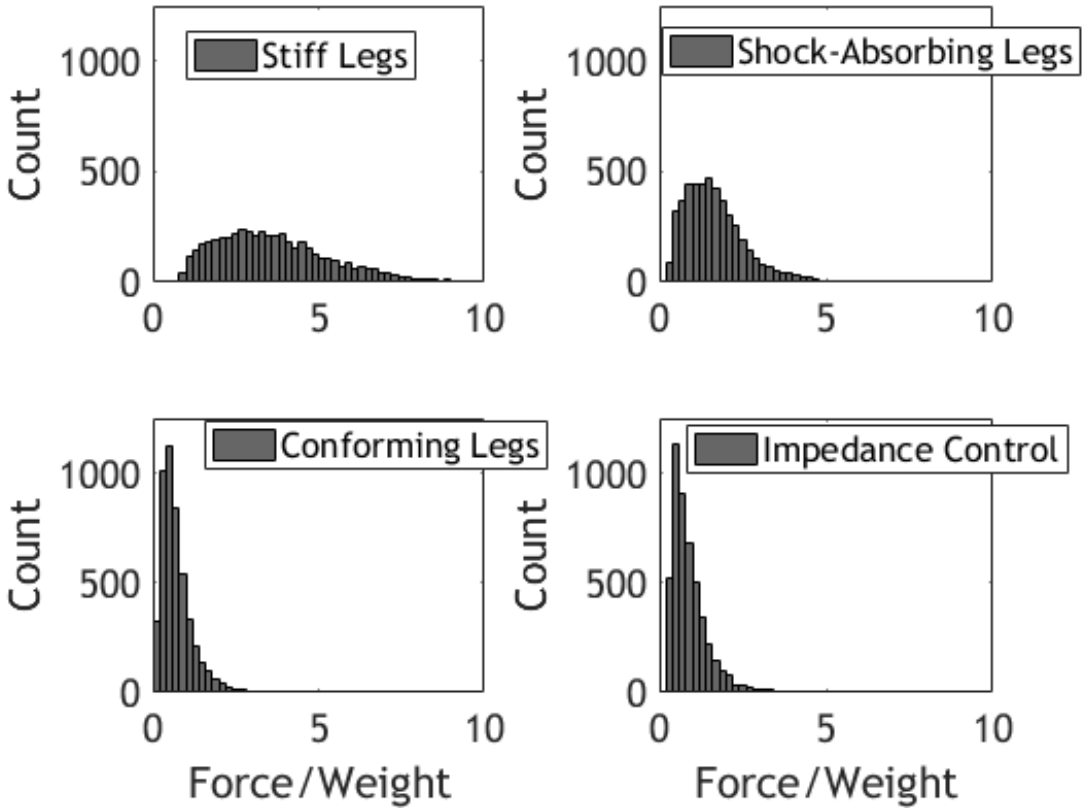

Figure 6.8: Peak lander joint forces distribution for the various systems landing on an icy surface. Peak joint forces from all 12 joints in each body are shown, normalized by the system's weight on Europa

From this view, both flexible leg cases performed better than the stiff leg case, with the conforming legs experiencing lower forces than the shock-absorbing legs. Since the conforming legs move out of the way more easily, it is clear why they would experience lower peak force. Meanwhile, the shock-absorbing legs endure higher force since they require more force to deflect. The impedance controlled legs experience similar peak loads to the conforming legs.

While a lander certainly must be able to survive the loads and acceleration of landing, it must also land right-side-up. A lander rollover could damage instrumentation and cause

the lander to be useless. To check for rollover, the maximum absolute angle from level throughout the simulation was recorded. A lander was considered to have rolled over if this value was greater than 90 degrees at any point in the simulation. Only 0.75% of the conforming leg cases (three cases out of 400) resulted in rollover, and none of the shock-absorbing cases did. 2.5% of the impedance controlled cases rolled over. The stiff legged system performed much worse, with 31% of cases resulting in rollover. Flexible and active impedance controlled legs can substantially reduce rollover risk on an icy surface.

6.3.2 Snowy Surface

The ground contact parameters for a snowy surface were chosen so the surface would be softer than ice, while still allowing for significant sliding. The parameters used are enumerated in Table 6.7.

Parameter	Value	Unit
SLS Spring, k_e	12000	N/m
SLS Spring, k_v	420	N/m
SLS Damping, c_v	3550	$N \cdot s/m$
LuGre Parameter, μ_s	0.2	Unitless
LuGre Parameter, μ_k	0.1	Unitless
LuGre Parameter, σ_0	0.0015	Unitless
LuGre Parameter, σ_1	0.2	Unitless
LuGre Parameter, σ_2	0.05	Unitless
LuGre Parameter, v_s	0.2	m/s
LuGre Parameter, α	1	Unitless

Table 6.7: Snowy surface ground contact parameters

Just as for the icy surface, 400 simulations were run for each leg stiffness setup for landing on a snowy surface. These simulations determined that the shock-absorbing legs

resulted in the lowest peak lander body acceleration, followed by the impedance controlled legs, then the conforming legs, and then the stiff legs. The shock-absorbing legs resulted in a mean peak acceleration reduction of about 40% compared to the stiff legs. These results are shown in Table 6.8.

Leg Type	Peak Lander Acceleration, Mean, $[m/s]$	Peak Lander Acceleration, Standard Deviation, $[m/s]$
Stiff	19.8	10.2
Shock-absorbing	11.8	6.6
Conforming	13.9	5.7
Impedance Control	12.3	5.7

Table 6.8: Snowy surface lander body acceleration statistics

Figure 6.9 shows the distribution of peak lander acceleration for the various systems. Just as for the icy surface simulations, there is a fairly broad spread due to the variety of initial conditions provided.

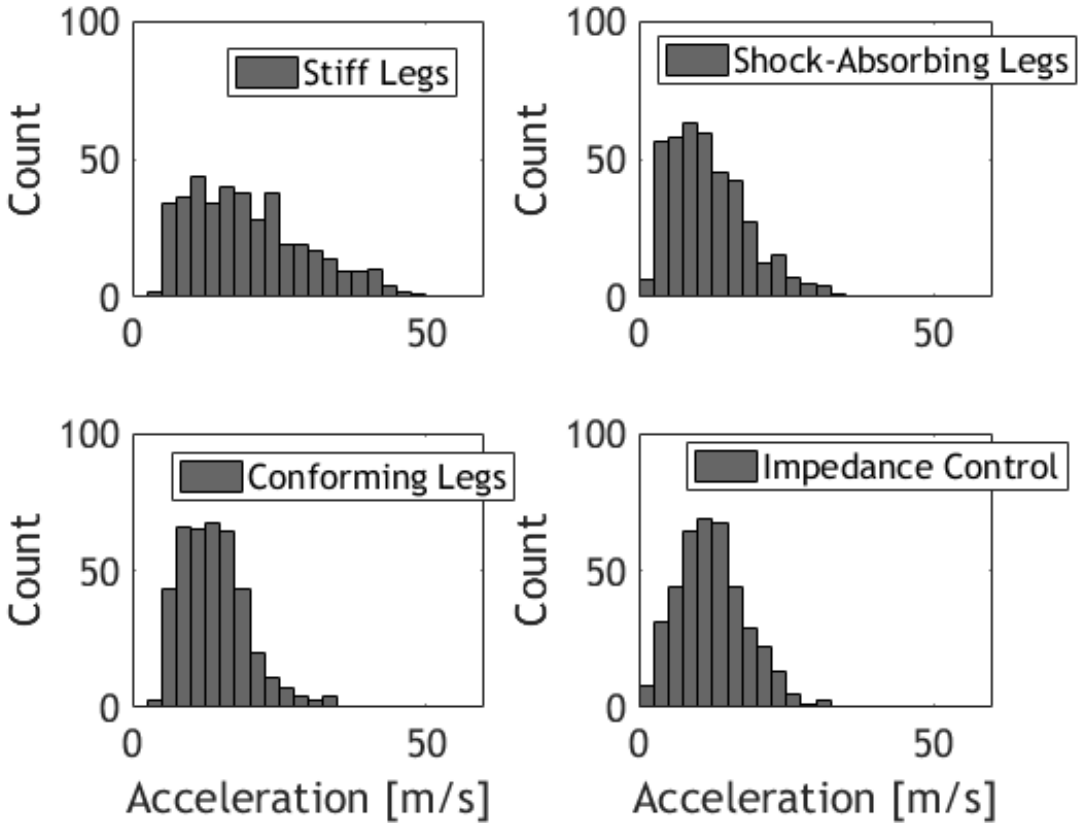

Figure 6.9: Peak lander body acceleration distribution for the various systems landing on a snowy surface

The normalized joint force distribution shown in Figure 6.10 mimics that for the icy surface, in that the stiff legs result in the highest forces and the conforming legs result in the lowest forces. The impedance controlled legs had peak joint forces similar to those of the conforming legs.

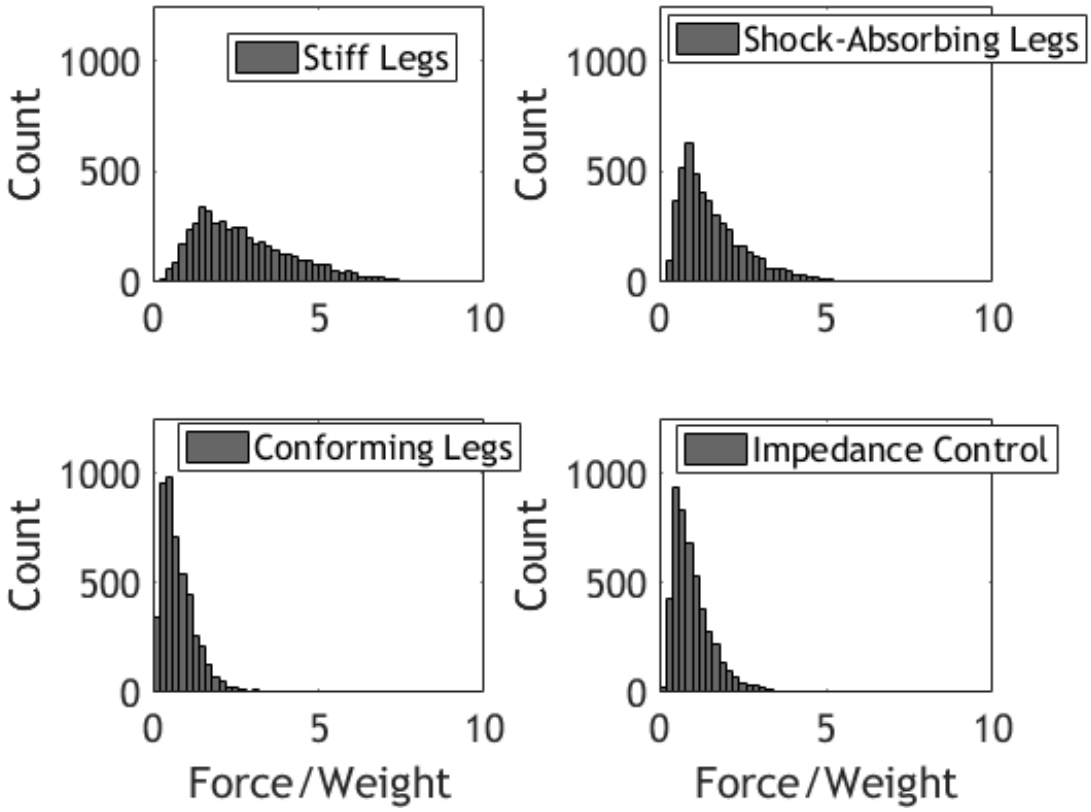

Figure 6.10: Peak lander joint forces distribution for the various systems landing on a snowy surface. Peak joint forces from all 12 joints in each body are shown, normalized by the system's weight on Europa

The rollover rate for landing on a snowy surface was also examined. Only one percent of the conforming leg cases (four cases out of 400) resulted in rollover, and none of the shock-absorbing cases did. The impedance controlled leg model also performed well, with only 0.75% of cases resulting in rollover. However, 13.5% of the stiff leg cases rolled over.

6.3.3 Sandy Surface

A sandy surface was simulated as being softer than snow, but highly damped in sliding. Table 6.9 displays the parameters used for the sandy surface simulations.

Parameter	Value	Unit
SLS Spring, k_e	8000	N/m
SLS Spring, k_v	420	N/m
SLS Damping, c_v	2550	$N \cdot s/m$
LuGre Parameter, μ_s	1.3	Unitless
LuGre Parameter, μ_k	1.1	Unitless
LuGre Parameter, σ_0	0.0015	Unitless
LuGre Parameter, σ_1	2.5	Unitless
LuGre Parameter, σ_2	0.15	Unitless
LuGre Parameter, v_s	0.2	m/s
LuGre Parameter, α	1	Unitless

Table 6.9: Sandy surface ground contact parameters

Just as for the previous simulation sets, 400 simulations were run for each leg stiffness setup for landing on a sandy surface. These simulations determined that the conforming legs resulted in the lowest peak lander body acceleration, followed by the impedance controlled legs, then the shock-absorbing legs, and then the stiff legs. This deviates from the previous results, since the shock-absorbing legs resulted in the lowest peak lander acceleration for landing on icy or snowy surfaces. For a sandy surface, the conforming legs resulted in a mean peak acceleration reduction of about 31% compared to the stiff legs. These results are shown in Table 6.10.

Leg Type	Peak Lander Acceleration, Mean, $[m/s]$	Peak Lander Acceleration, Standard Deviation, $[m/s]$
Stiff	19.0	8.2
Shock-absorbing	16.9	7.2
Conforming	13.1	6.8
Impedance Control	14.0	6.2

Table 6.10: Sandy surface lander body acceleration statistics

Figure 6.11 shows the distribution of peak lander acceleration for the various systems. Just as for the previous simulations, there is a fairly broad spread due to the variety of initial conditions provided.

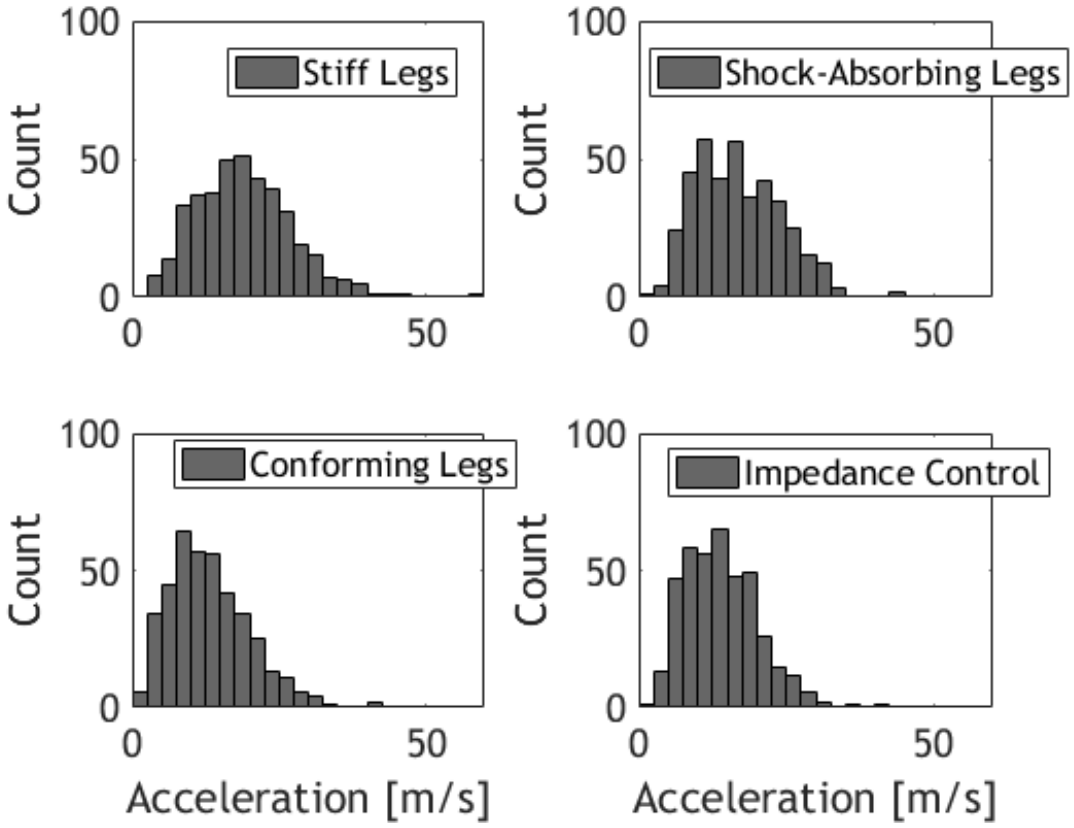

Figure 6.11: Peak lander body acceleration distribution for the various systems landing on a sandy surface

The normalized joint force distribution shown in Figure 6.12 shows much more overlap in loads compared to the landing cases on icy or snowy surfaces. It is believed that since the legs cannot slide as easily as in those cases, that they grip the surface better and experience higher peak forces while slowing the system down.

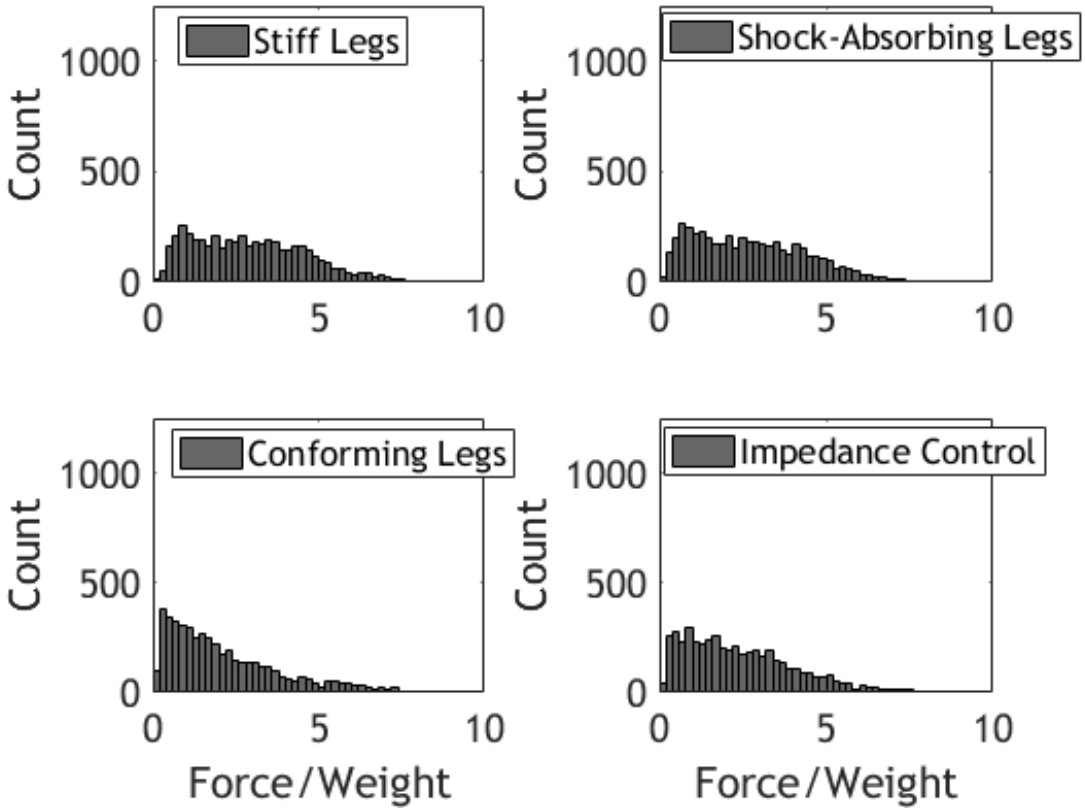

Figure 6.12: Peak lander joint forces distribution for the various systems landing on a sandy surface. Peak joint forces from all 12 joints in each body are shown, normalized by the system's weight on Europa

The rollover rate for landing on a sandy surface was examined, just as it was for the icy and snowy surface cases. The sandy surface resulted in many more rollover events than for the icy or snowy surfaces. The legs grip onto the surface more substantially, which can result in rollover with sufficient horizontal impact velocity. Nonetheless, the passive flexible leg cases and impedance controlled leg cases performed better than the stiff leg cases. Systems with conforming legs rolled over 15.5% of the time, systems with impedance controlled legs rolled over 15.75% of the time, systems with shock-absorbing legs rolled over 24% of the time, and systems with stiff legs rolled over 35% of the time.

An example landing is shown in Figure 6.13. Here, shock-absorbing legs were used. The lander had a horizontal velocity upon impact of about 2.9 meters per second, and a vertical velocity of about 1.1 meters per second. The front legs contacted first, caught in the sandy surface, and triggered a lander rollover.

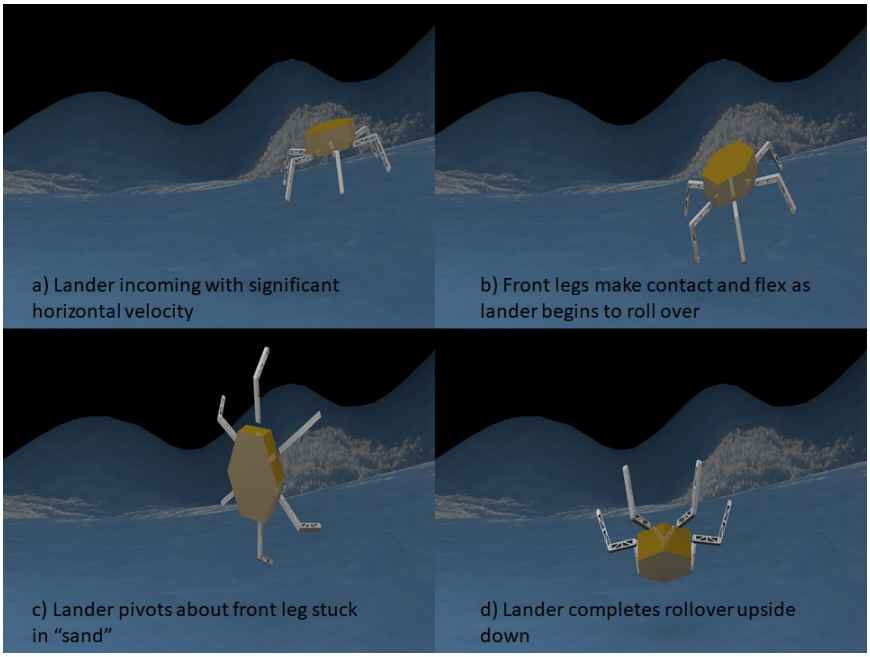

Figure 6.13: This landing event occurred with shock-absorbing legs on a sandy surface. The lander has horizontal velocity of about 2.9 meters per second. The front legs touched the surface first, and triggered a lander rollover

6.4 Analysis of Results

Clearly, flexible legs can offer benefits to a Europa lander. Lower peak lander acceleration can prevent instrument damage, and lower peak loads on legs may allow for lighter legs and a higher portion of total mass for instrumentation use. A lower rollover risk also allows for a higher probability of mission success. However, there are still some concerns. Specifically, all cases for the sandy surface scenario resulted in relatively high rollover risk. While it may be expected that such a surface grabs the lander feet harder and can more easily cause

rollover, it is worth examining this case further to analyze what cases resulted in rollover and how the rollover risk may be lowered further.

Critically, all the landing event scenarios were performed with a variety of horizontal velocities centered about a nonzero value. This was done intentionally to put the lander into challenging lander scenarios more often, and to test the envelope of landing ability. Since the simulated lander is axially symmetric, there is no concern in ignoring landing in the opposite direction. Since these scenarios have some nominal nonzero horizontal velocity, the true rollover risk is likely lower than seen here, assuming a vertical landing is more likely in practice.

Obviously the landing dynamics of this lander are highly complex. Rollover risk is affected by impact velocity, orientation at impact, ground contact dynamics, leg flexibility, and more. To determine how these factors affect the rollover results, the stiff legged landing scenarios on icy and sandy surfaces were examined further.

First, consider the stiff legged lander on an icy surface. Figure 6.14 examines the initial lander roll and pitch for all 400 cases, and classifies them based on whether the lander rolled over or not. The lander cases were not given any initial rotational velocity, so the initial orientation is also the orientation at impact. Surprisingly, there is no obvious bias for rollover cases when the lander hits at more extreme angles.

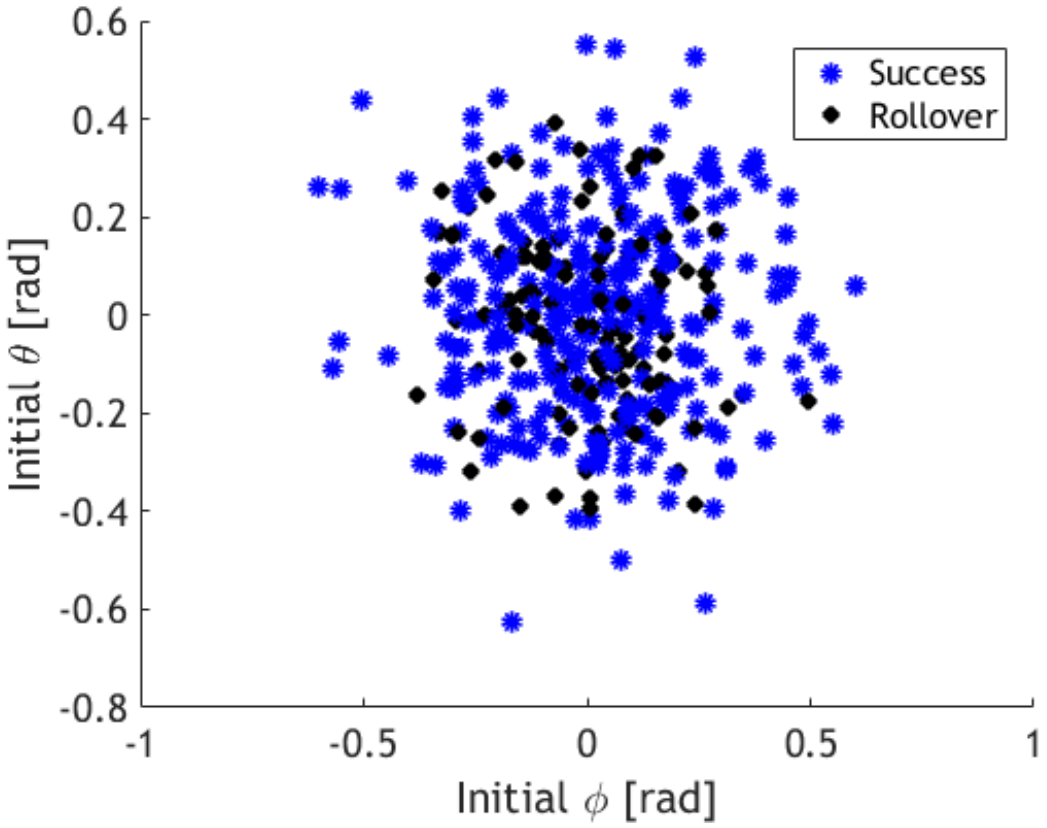

Figure 6.14: Rollover classification for a stiff legged lander on an icy surface based on initial lander orientation

Figure 6.15 then classifies rollover for all 400 cases based on horizontal velocity at impact. Here also, there is no obvious bias for rollover cases when the lander hits with a larger horizontal velocity.

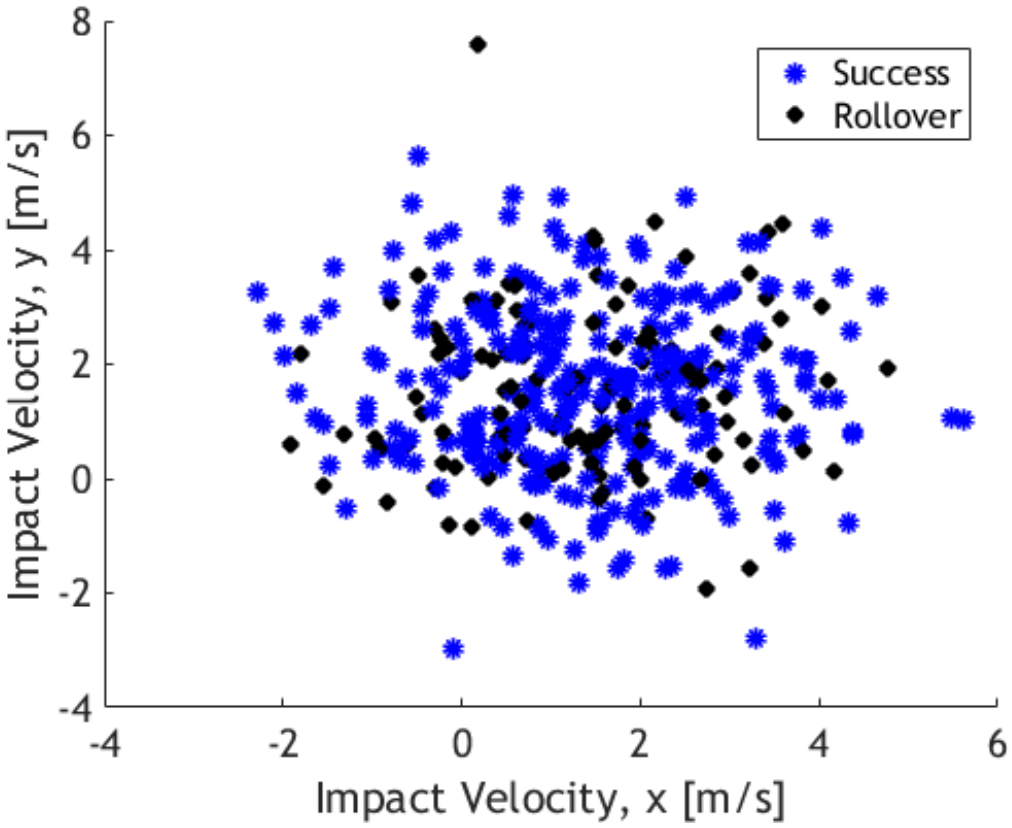

Figure 6.15: Rollover classification for a stiff legged lander on an icy surface based on horizontal velocity at impact

Finally, some clear separation of rollover cases in seen in Figure 6.16. As the vertical impact velocity increases above about 3.5 meters per second, the lander is much more likely to roll over.

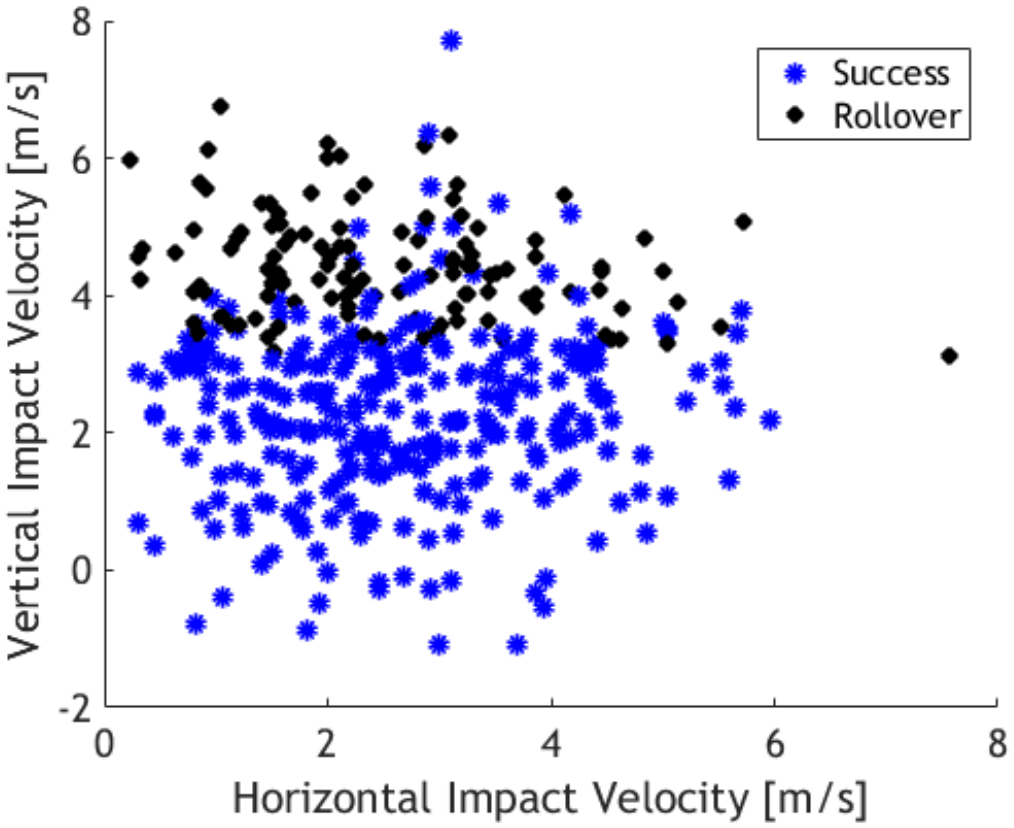

Figure 6.16: Rollover classification for a stiff legged lander on an icy surface based on horizontal and vertical velocities at impact

For an icy surface, the feet do not experience significant frictional force on impact, but with the hard surface they do bounce vertically. This explains these results. Horizontal velocity and orientation have little effect on an icy surface, since very little horizontal frictional force is applied to the feet to cause rollover. However, upon a hard landing, the system can bounce and roll over.

A proposed composite factor to consider is the projection of the impact velocity vector onto the Z-axis of the lander body frame. This factor combines the effects of impact velocity, orientation, and the coupling between their directions. If the two vectors are roughly in the same direction, the projection will be positive. This corresponds to the lander tilted

back upon landing. If the projection is negative, the lander is tilted forward upon landing. Figure 6.17 provides a simple representation of this projection.

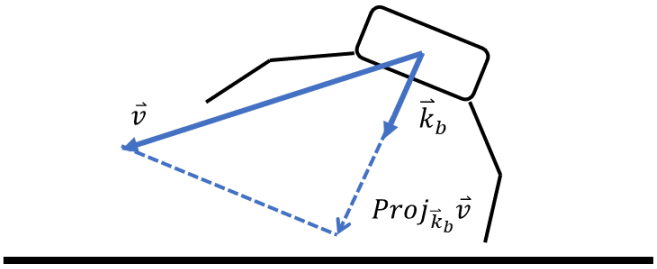

Figure 6.17: Simple representation of the projection of the impact velocity vector onto the z-axis of the lander body

Figure 6.18 shows the use of this composite factor for classification of rollover events of a stiff legged lander on an icy surface. There is a fairly clear separation between successful and rolled landing events.

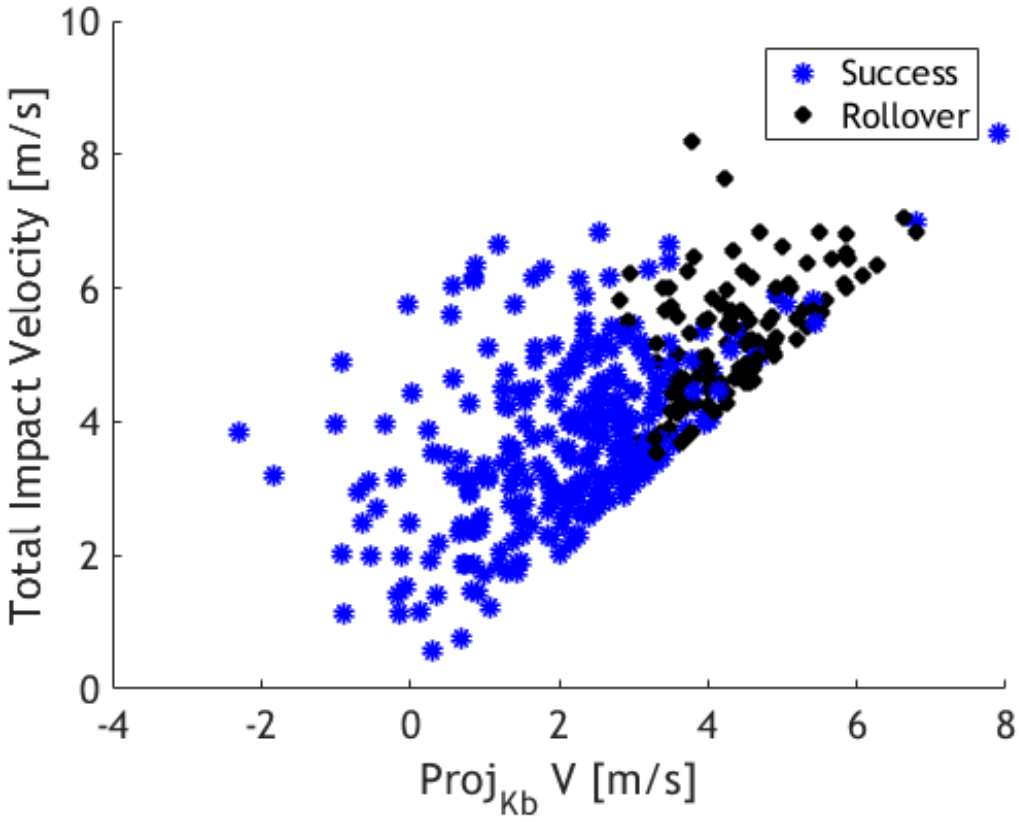

Figure 6.18: Rollover classification for a stiff legged lander on an icy surface based on the projection of the velocity vector onto the lander body Z-axis at impact

Second, consider the stiff legged lander on a sandy surface. On this surface, there were substantially more rollover cases than for an icy surface. It was proposed that perhaps this was because the feet gripped the surface better, and so horizontal velocity and tilted initial orientation would result in rollover. Figure 6.19 examines rollover classification by initial orientation. Surprisingly, there is no clear separation visible, even though the feet grip the sandy surface much better.

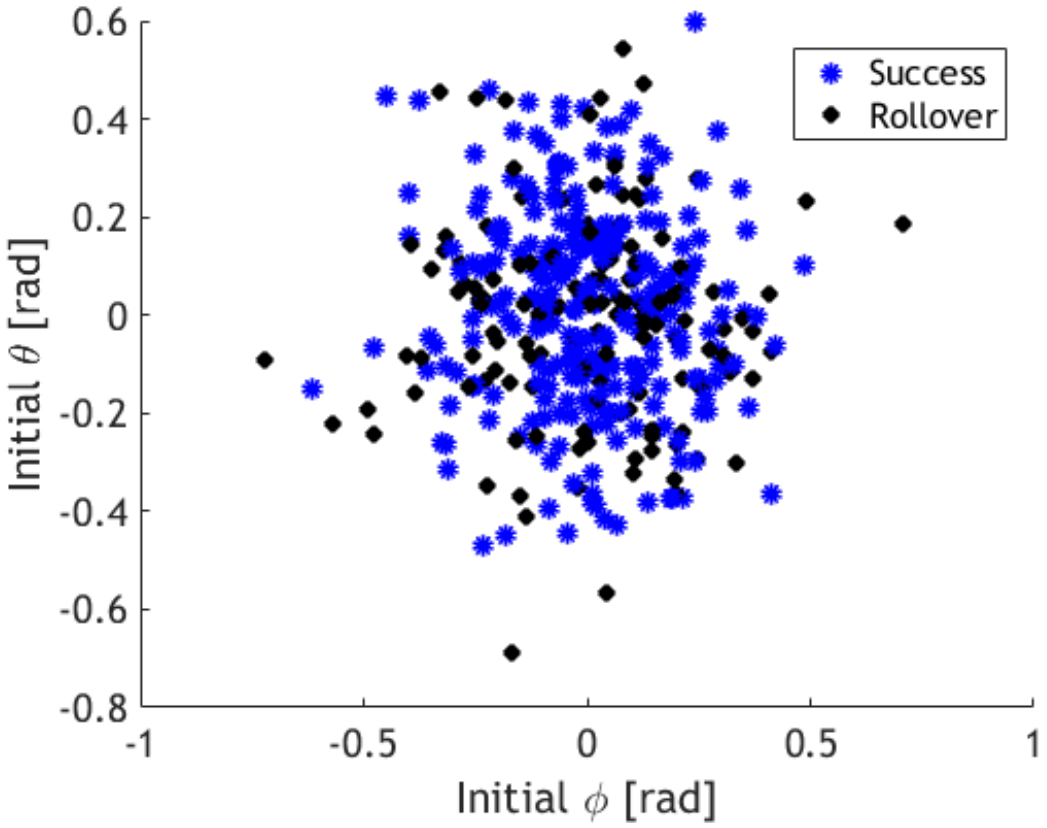

Figure 6.19: Rollover classification for a stiff legged lander on an sandy surface based on initial lander orientation

Likewise, Figure 6.20 classifies rollover by horizontal impact velocity. Again, there is no obvious separation of rollover based on horizontal velocity alone. Clearly, the situation is more complex.

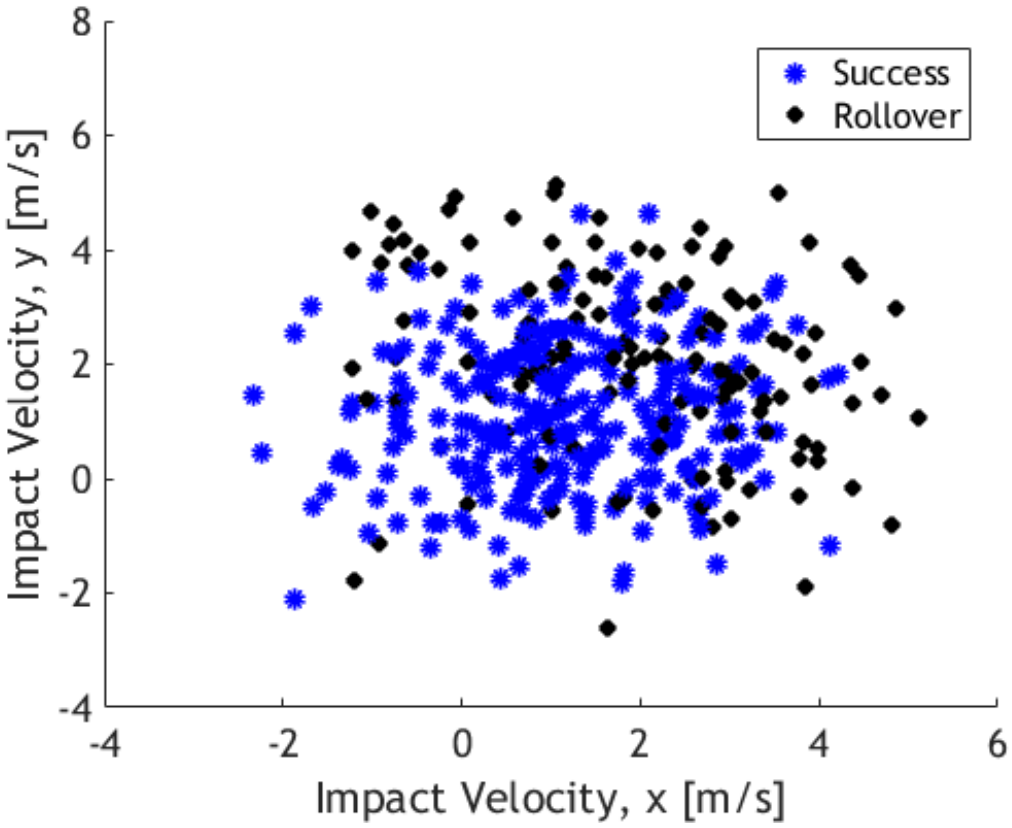

Figure 6.20: Rollover classification for a stiff legged lander on an sandy surface based on horizontal velocity at impact

Just as for the icy surface, separation is seen when examining cases by their horizontal and vertical velocities, as seen in Figure 6.21. However, the classification is not as simple as it was for the icy surface. Whereas for the icy surface the classification was mainly driven by vertical velocity, for the sandy surface it is driven by some combination of vertical and horizontal velocities.

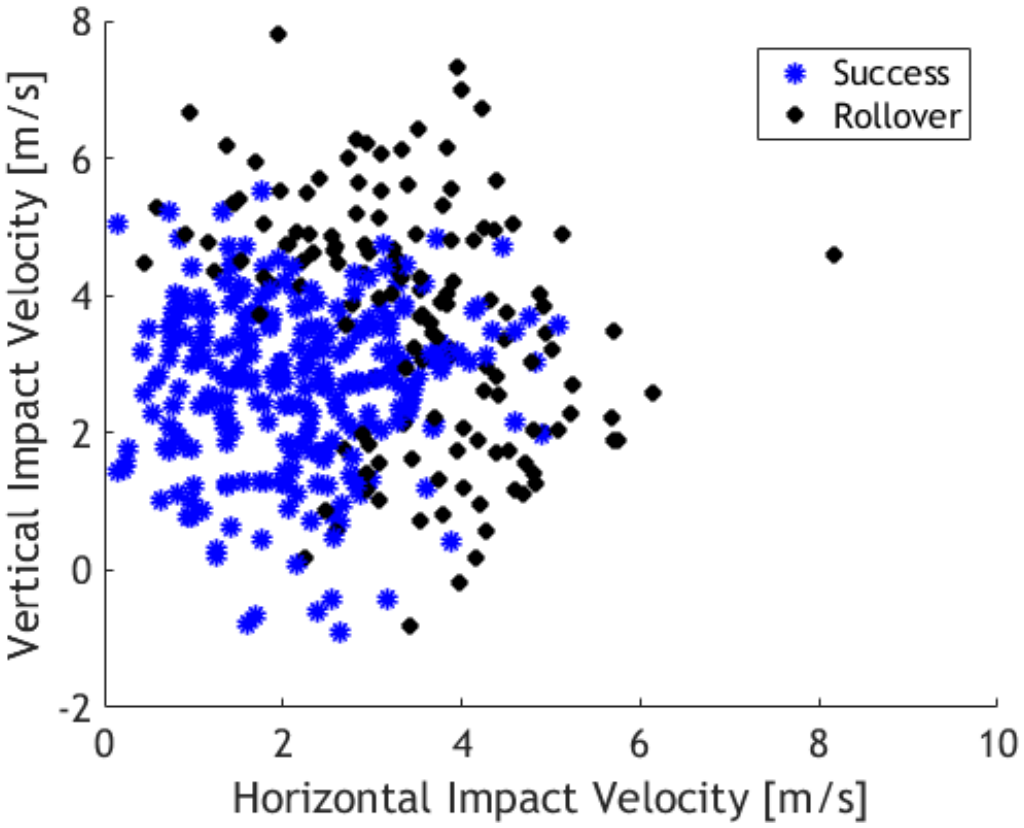

Figure 6.21: Rollover classification for a stiff legged lander on an sandy surface based on horizontal and vertical velocities at impact

The composite factor presented earlier consisting of the projection of the impact velocity vector onto the lander Z-axis at impact also allows for classification of rollover events on a sandy surface. Figure 6.22 shows this separation. This result implies that some combination of impact velocity and orientation is responsible for rollover cases here, and that neither impact velocity nor orientation alone is responsible.

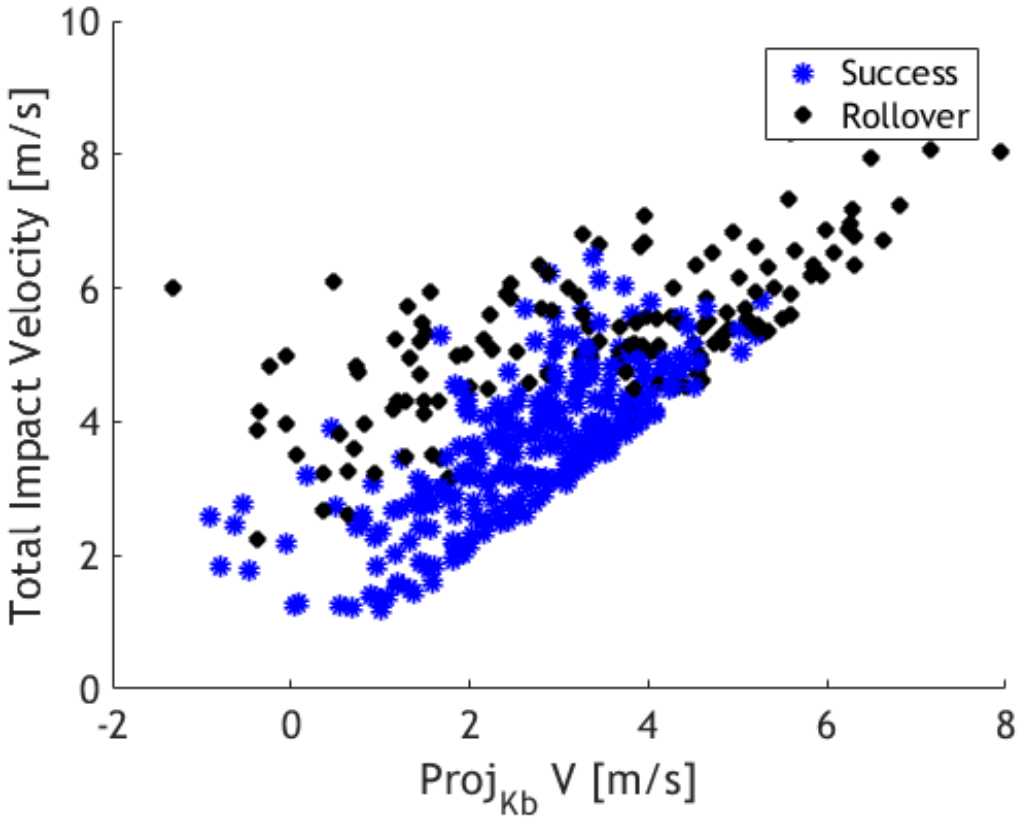

Figure 6.22: Rollover classification for a stiff legged lander on an sandy surface based on the projection of the velocity vector onto the lander body Z-axis at impact

CHAPTER 7
CONCLUSION

The analysis and experimentation in this thesis show that a naive implementation of the nonlinear control theory constrained coordinate multibody dynamic simulation technique is computationally expensive for systems with large numbers of constraints and bodies. In particular, sections of the feedback linearization constraint controller are determined to be computationally expensive if performed naively, and to computationally scale poorly with the number of bodies and joints simulated. Namely, these sections are a matrix multiplication and a linear system solve step. It is shown that these sections together can comprise the vast majority of computation time for a simulation with many bodies. Methods were developed and shown to greatly reduce the computation time needed for those two important sections of the constraint controller in a few token systems. Notably, the methods which were most effective do not reduce simulation accuracy at all and are not approximation techniques. For many systems, the computational time reductions are significant. One particular case which works especially well with these methods was shown experimentally to have an overall computational time reduction of 140-fold.

The computational improvements developed in this thesis enable the practical use of this nonlinear control theory constrained coordinate multibody dynamic simulation technique for larger systems than were previously possible. It is the hope of the author that this tool will be useful to other researchers in their studies of multibody systems.

The computational improvements developed here were applied to a simulation of a passive flexible legged lander system design for use on Europa. In tests, the computational methods developed here reduce computation time for Europa lander simulations by about 24-fold.

Trade studies were run to examine the performance of flexible leg designs versus a

rigid leg lander design. Simulations were performed under a variety of surface conditions, impact angles, and impact velocities. It was determined that a lander with flexible legs experiences lower peak acceleration upon impact, lower joint loads, and a lower risk of rollover. Flexible legs lowered peak lander acceleration by about 42% and 40% on simulated icy and snowy surfaces, respectively. Flexible legs were also able to virtually eliminate rollover risk when landing on those surfaces. On a simulated sandy surface with significantly higher damping, flexible legs reduced peak lander acceleration by about 31%. In addition, while landers with stiff legs rolled over in this sandy surface scenario about 35% of the time, landers with very flexible legs rolled over only 15.5% of the time.

Appendices

APPENDIX A

CONSTRAINT ERROR DERIVATIVES

$$\dot{E}^{\star}_{tj} = \Gamma^{T}_{tj} T_{pj} \Bigg(\begin{pmatrix} u_p \\ v_p \\ w_p \end{pmatrix} - T_p T_c^T \begin{pmatrix} u_c \\ v_c \\ w_c \end{pmatrix} - \mathbb{S}_p(\bar{\omega^{}_{\frac{bp}{I}}}) T_p \begin{pmatrix} x_p - x_c \\ y_p - y_c \\ z_p - z_c \end{pmatrix} + \mathbb{S}_p(\bar{\omega^{}_{\frac{bp}{I}}}) T_p T_c^T \mathbb{C}_c(\bar{r}_{\bigotimes_c \to j}) - T_p T_c^T \mathbb{S}_c(\bar{\omega^{}_{\frac{bc}{I}}}) \mathbb{C}_c(\bar{r}_{\bigotimes_c \to j}) \Bigg) \tag{A.1}$$

$$\chi_p = \begin{pmatrix} x_p & y_p & z_p \end{pmatrix}^T \tag{A.2}$$

$$Q_p = \begin{pmatrix} q_{0p} & q_{1p} & q_{2p} & q_{3p} \end{pmatrix}^T \tag{A.3}$$

$$\nu_p = \begin{pmatrix} u_p & v_p & w_p \end{pmatrix}^T \tag{A.4}$$

$$\Omega_p = \begin{pmatrix} p_p & q_p & r_p \end{pmatrix}^T \tag{A.5}$$

$$X_p = \begin{pmatrix} \chi_p^T & Q_p^T & \nu_p^T & \Omega_p^T \end{pmatrix}^T \tag{A.6}$$

$$\chi_c = \begin{pmatrix} x_c & y_c & z_c \end{pmatrix}^T \tag{A.7}$$

$$Q_c = \begin{pmatrix} q_{0c} & q_{1c} & q_{2c} & q_{3c} \end{pmatrix}^T \tag{A.8}$$

$$\nu_c = \begin{pmatrix} u_c & v_c & w_c \end{pmatrix}^T \tag{A.9}$$

$$\Omega_c = \begin{pmatrix} p_c & q_c & r_c \end{pmatrix}^T \tag{A.10}$$

$$X_c = \begin{pmatrix} \chi_c^T & Q_c^T & \nu_c^T & \Omega_c^T \end{pmatrix}^T \tag{A.11}$$

$$\frac{\partial \dot{E}^{\star}_{tj}}{\partial X_p} = \begin{pmatrix} \frac{\partial \dot{E}^{\star}_{tj}}{\partial \chi_p} & \frac{\partial \dot{E}^{\star}_{tj}}{\partial Q_p} & \frac{\partial \dot{E}^{\star}_{tj}}{\partial \nu_p} & \frac{\partial \dot{E}^{\star}_{tj}}{\partial \Omega_p} \end{pmatrix}^T \tag{A.12}$$

$$\frac{\partial \dot{E}^{\star}_{tj}}{\partial \chi_p} = -\Gamma_{tj}^T T_{pj} \mathbb{S}_p(\bar{\omega_{\frac{bp}{I}}}) T_p \tag{A.13}$$

$$\frac{\partial \dot{E}^{\star}_{tj}}{\partial q_{ip}} = \Gamma_{tj}^T T_{pj} \mathbb{S}_p(\bar{\omega_{\frac{bi}{I}}}) \frac{\partial T_p}{\partial q_{ip}} \left(T_c^T \mathbb{C}_c(\bar{r}_{\bigotimes_c \to j}) - \begin{pmatrix} x_p - x_c \\ y_p - y_c \\ z_p - z_c \end{pmatrix} \right) -$$
$$\Gamma_{tj}^T T_{pj} \frac{\partial T_p}{\partial q_{ip}} T_c^T \left(\begin{pmatrix} u_c \\ v_c \\ w_c \end{pmatrix} - \mathbb{S}_c(\bar{r}_{\bigotimes_c \to j}) \begin{pmatrix} p_c \\ q_c \\ r_c \end{pmatrix} \right) for \quad i = 0, ..., 3 \tag{A.14}$$

$$\frac{\partial T_p}{\partial q_{0p}} = 2 \begin{pmatrix} q_{0p} & q_{3p} & -q_{2p} \\ -q_{3p} & q_{0p} & q_{1p} \\ q_{2p} & -q_{1p} & q_{0p} \end{pmatrix} \tag{A.15}$$

$$\frac{\partial T_p}{\partial q_{1p}} = 2 \begin{pmatrix} q_{1p} & q_{2p} & q_{3p} \\ q_{2p} & -q_{1p} & q_{0p} \\ q_{3p} & -q_{0p} & -q_{1p} \end{pmatrix} \tag{A.16}$$

$$\frac{\partial T_p}{\partial q_{2p}} = 2\begin{pmatrix} -q_{2p} & q_{1p} & -q_{0p} \\ q_{1p} & q_{2p} & q_{3p} \\ q_{0p} & q_{3p} & -q_{2p} \end{pmatrix} \tag{A.17}$$

$$\frac{\partial T_p}{\partial q_{3p}} = 2\begin{pmatrix} -q_{3p} & q_{0p} & q_{1p} \\ -q_{0p} & -q_{3p} & q_{2p} \\ q_{1p} & q_{2p} & q_{3p} \end{pmatrix} \tag{A.18}$$

$$\frac{\partial \dot{E}^{\star}_{tj}}{\partial \nu_p} = \Gamma^T_{tj} T_{pj} \tag{A.19}$$

$$\frac{\partial \dot{E}^{\star}_{tj}}{\partial \Omega_{p_i}} = \Gamma^T_{tj} T_{pj} \frac{\partial \mathbb{S}_p(\bar{\omega_{\frac{bp}{I}}})}{\partial \Omega_{p_i}} T_p \Bigg(- \begin{pmatrix} x_p - x_c \\ y_p - y_c \\ z_p - z_c \end{pmatrix} + T_c^T \mathbb{C}_c(\bar{r}_{\bigotimes_c \to j}) \Bigg) \quad for \quad i = p_p, q_p, r_p \tag{A.20}$$

$$\frac{\partial \mathbb{S}_p(\bar{\omega_{\frac{bp}{I}}})}{\partial p_p} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \tag{A.21}$$

$$\frac{\partial \mathbb{S}_p(\bar{\omega_{\frac{bp}{I}}})}{\partial q_p} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \tag{A.22}$$

$$\frac{\partial \mathbb{S}_p(\bar{\omega_{\frac{bp}{I}}})}{\partial r_p} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{A.23}$$

$$\frac{\partial \dot{E}^{\star}_{tj}}{\partial X_c} = \begin{pmatrix} \frac{\partial \dot{E}^{\star}_{tj}}{\partial \chi_c} & \frac{\partial \dot{E}^{\star}_{tj}}{\partial Q_c} & \frac{\partial \dot{E}^{\star}_{tj}}{\partial \nu_c} & \frac{\partial \dot{E}^{\star}_{tj}}{\partial \Omega_c} \end{pmatrix}^T \tag{A.24}$$

$$\frac{\partial \dot{E}^{\star}_{tj}}{\partial \chi_c} = \Gamma^T_{tj} T_{pj} \mathbb{S}_p(\omega^{-}_{\frac{bp}{I}}) T_p \tag{A.25}$$

$$\frac{\partial \dot{E}^{\star}_{tj}}{\partial q_{ic}} = -\Gamma^T_{tj} T_{pj} T_p \frac{\partial T^T_c}{\partial q_{ic}} \left(\begin{pmatrix} u_c \\ v_c \\ w_c \end{pmatrix} - \mathbb{S}_c(\bar{r}_{\bigotimes_c \to j}) \begin{pmatrix} p_c \\ q_c \\ r_c \end{pmatrix} \right) +$$
$$\Gamma^T_{tj} T_{pj} \mathbb{S}_p(\omega^{-}_{\frac{bp}{I}}) T_p \frac{\partial T^T_c}{\partial q_{ic}} \mathbb{C}_c(\bar{r}_{\bigotimes_c \to j}) \quad for \quad i = 0, ..., 3 \tag{A.26}$$

$$\frac{\partial T_c}{\partial q_{0c}} = 2 \begin{pmatrix} q_{0c} & q_{3c} & -q_{2c} \\ -q_{3c} & q_{0c} & q_{1c} \\ q_{2c} & -q_{1c} & q_{0c} \end{pmatrix} \tag{A.27}$$

$$\frac{\partial T_c}{\partial q_{1c}} = 2 \begin{pmatrix} q_{1c} & q_{2c} & q_{3c} \\ q_{2c} & -q_{1c} & q_{0c} \\ q_{3c} & -q_{0c} & -q_{1c} \end{pmatrix} \tag{A.28}$$

$$\frac{\partial T_c}{\partial q_{2c}} = 2 \begin{pmatrix} -q_{2c} & q_{1c} & -q_{0c} \\ q_{1c} & q_{2c} & q_{3c} \\ q_{0c} & q_{3c} & -q_{2c} \end{pmatrix} \tag{A.29}$$

$$\frac{\partial T_c}{\partial q_{3c}} = 2 \begin{pmatrix} -q_{3c} & q_{0c} & q_{1c} \\ -q_{0c} & -q_{3c} & q_{2c} \\ q_{1c} & q_{2c} & q_{3c} \end{pmatrix} \tag{A.30}$$

$$\frac{\partial \dot{E}^{\star}_{tj}}{\partial \nu_c} = -\Gamma^T_{tj} T_{pj} T_p T^T_c \tag{A.31}$$

$$\frac{\partial \dot{E}^{\star}_{tj}}{\partial \Omega_c} = \Gamma^T_{tj} T_{pj} T_p T^T_c \mathbb{S}_c(\bar{r}_{\bigotimes_c \to j}) \tag{A.32}$$

$$\dot{E}^{\star}_{rj} = -\Phi T_{pj} \left(\mathbb{S}_p(\bar{\omega^{}_{\frac{bp}{I}}}) T_p T^T_c - T_p T^T_c \mathbb{S}_c(\bar{\omega^{}_{\frac{bc}{I}}}) \right) T^T_{cj} \Psi^T \tag{A.33}$$

$$\frac{\partial \dot{E}^{\star}_{rj}}{\partial \chi_p} = 0 \tag{A.34}$$

$$\frac{\partial \dot{E}^{\star}_{rj}}{\partial q_{ip}} = -\Phi T_{pj} \left(\mathbb{S}_p(\bar{\omega^{}_{\frac{bp}{I}}}) \frac{\partial T_p}{\partial q_{ip}} T^T_c - \frac{\partial T_p}{\partial q_{ip}} T^T_c \mathbb{S}_c(\bar{\omega^{}_{\frac{bc}{I}}}) \right) T^T_{cj} \Psi^T \quad for \quad i = 0, ..., 3 \tag{A.35}$$

$$\frac{\partial \dot{E}^{\star}_{rj}}{\partial \nu_p} = 0 \tag{A.36}$$

$$\frac{\partial \dot{E}^{\star}_{rj}}{\partial \Omega_p} = -\Phi T_{pj} \frac{\partial \mathbb{S}_p(\bar{\omega^{}_{\frac{bp}{I}}})}{\partial \Omega_{pi}} T_p T^T_c T^T_{cj} \Psi^T \quad for \quad i = p_p, q_p, r_p \tag{A.37}$$

$$\frac{\partial \dot{E}^{\star}_{rj}}{\partial \chi_c} = 0 \tag{A.38}$$

$$\frac{\partial \dot{E}^{\star}_{rj}}{\partial q_{ic}} = -\Phi T_{pj} \left(\mathbb{S}_p(\bar{\omega^{}_{\frac{bp}{I}}}) T_p \frac{\partial T_c}{\partial q_{ic}} - T_p \frac{\partial T_c}{\partial q_{ic}} \mathbb{S}_c(\bar{\omega^{}_{\frac{bc}{I}}}) \right) T^T_{cj} \Psi^T \quad for \quad i = 0, ..., 3 \tag{A.39}$$

$$\frac{\partial \dot{E}^{\star}_{rj}}{\partial \nu_c} = 0 \tag{A.40}$$

$$\frac{\partial \dot{E}^{\star}_{rj}}{\partial \Omega_c} = \Phi T_{pj} T_p T^T_c \frac{\partial \mathbb{S}_p(\bar{\omega^{}_{\frac{bc}{I}}})}{\partial \Omega_{ci}} T^T_{cj} \Psi^T \quad for \quad i = p_p, q_p, r_p \tag{A.41}$$

REFERENCES

[1] E. Leylek, W. M., and M. Costello, "Flight dynamic simulation for multibody aircraft configurations," *Journal of Guidance, Control, and Dynamics*, vol. 35, 6 Nov. 2012.

[2] J. Kiefer, M. Ward, and M. Costello, "Rotorcraft hard landing mitigation using robotic landing gear," *Journal of Dynamic Systems, Measurement, and Control*, vol. 138, Mar. 2016.

[3] *Computational Improvements to Multibody Projectile Dynamics Simulation*, AIAA Aviation, Jun. 2014.

[4] G. Golub and C. Van Loan, *Matrix Computations*. The John Hopkins University Press, 1996.

[5] G. Allaire and K. S. M., *Numerical Linear Algebra*. Springer-Verlag New York, 2008.

[6] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2003.

[7] J. Harris, J. Hirst, and M. Mossinghoff, *Combinatorics and Graph Theory*. Springer Science+Business Media, LLC, 2008.

[8] C. H. Papadimitriou, "The np-completeness of the bandwidth minimization problem," *Computing*, vol. 16, pp. 263–270, Jun. 1975.

[9] *Reducing the bandwidth of sparse symmetric matrices*, Aug. 1969.

[10] G. M. Del Corso and G. Manzini, "Finding exact solutions to the bandwidth minimization problem," *Computing*, vol. 62, pp. 189–203, Mar. 1999.

[11] Europa Study Team, "Europa study 2012 report," Tech. Rep., 2012.

[12] NASA/JPL-Caltech, *Pia21431: Highest-resolution europa image & mosaic from galileo*, Feb. 2017.

[13] K. P. Hand, A. E. Murray, J. B. Garvin, W. B. Brinckerhoff, B. C. Christner, K. S. Edgett, B. L. Ehlmann, C. R. German, A. G. Hayes, T. M. Hoehler, S. M. Horst, J. I. Lunine, K. H. Nealson, C. Paranicas, B. E. Schmidt, D. E. Smith, A. R. Rhoden, M. J. Russell, A. S. Templeton, P. A. Willis, R. A. Yingst, C. B. Phillips, M. L. Cable, K. L. Craft, A. E. Hofmann, T. A. Nordheim, R. P. Pappalardo, and

The Project Engineering Team, "Report of the europa lander science definition team," Tech. Rep., 2017.

[14] NASA/JPL-Caltech, *Pia21048*, Feb. 2017.

[15] S. Marques and G. Creus, *Computational Viscoelasticity*. Springer, 2012.

[16] K. J. Åström and C. Canudas de Wit, "Revisiting the lugre friction model," *IEEE Control Systems*, vol. 28, pp. 101–114, 6 Dec. 2008.