COMPUTATIONAL SIMULATION OF DYNAMICAL STRING BASED CAPTURING METHODS

A Thesis

by

DAVID EDWARD CAPPS

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee, Manoranjan Majji
Co-Chair of Committee, Robert E Skelton
Committee Member, Dileep Kalathil
Head of Department, Srinivas R Vadali

August 2021

Major Subject: Aerospace Engineering

ABSTRACT

A novel solution to conditionally tensile member modeling with conditional, comprehensive, body interactions is used to simulate the dynamic interactions of a body with a net made of connected strings. The modeling solution is used to study the capture dynamics of a rigid body using a network of masses. High performance computing implementation of the modeling solutions are discussed to simulate the capture dynamics. Practical implementation considerations including control of the network of masses to ensure optimal coverage of the tumbling rigid body are discussed. To do so, several methods are to be used to give a more developed control theory.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professors Robert Skelton and Manoranjan Majji (Department of Aerospace Engineering), and Professor Dileep Kalathil (Department of Electrical Engineering), and Srinivas R Vadali (Head of the Department of Aerospace Engineering).

All work conducted for the thesis was completed by the student independently.

Funding Sources

Graduate study was supported by Professors Robert Skelton and Manoranjan Majji.

TABLE OF CONTENTS

Page

1. Introduction

1.1 Motivation

Since the advent of spacecraft, the number of objects in Low Earth Orbit (LEO) have increased substantially [1]. While this has greatly improved life here on Earth, there is a danger of too many LEO bodies. The danger is, as Kessler and Cour-Palais described in 1978, the "Kessler Syndrome" [2], where there reaches a time where no more spacecraft can be sent into space safely. In order to prevent this threshold, either fewer objects need to be added and/or more objects must be removed. According to a study conducted in 2012, a net of at least 5 large LEO debris objects must be removed each year in order to prevent the Kessler Syndrome [3]. With this project, one method can be used to solve this problem: capture and deorbit. This research aims to develop an accurate model for a net, made of connected strings, capturing a body. The proposed method will have the advantage of reducing a orbiting body's velocity without destroying it, preventing the creation of dangerous debris in orbit.

1.2 History of the Problem

There are many research projects focused on space debris removal. Many include single missions to remove multiple objects [4, 5]. However, for large bodies in tumbling motion, the difficulty arises in how to rendezvous with them. For example, a harpoon solution, where a projectile is launched at the tumbling body, [6] will run the risk of breaking up an object or failing to get a proper hold on it. Especially in the case where the body is brittle or in fast rotation [7]. To address these complexities, this research proposes to *catch* the body by fully encompassing it with a network of masses (net). While the net's momentum can help reduce the object's rotational angular momentum, the primary reason for the net will be to enable effective deorbiting of the rigid body. There has been interest in doing exactly this by the European Space Agency [8] along with other organizations around the world that are interested in the same topic.

1.3 Research Objective

The goal of this research is to derive equations of motion for an arbitrary number and configuration of masses (net) capturing an arbitrary convex body. In addition to the utility in clearing space debris, there are many possible uses for this dynamical model. One use may be to incorporate this method into an existing method of designing structures, Tensegrity [9] paradigm is a method of using only tensile (strings) and compressive (bars) members to design a structure. This proposed model could be used to more accurately model these structures by considering when the strings are not in tension.

1.4 Implementation Details

One final feature planned for this research is the specification of design of imposed force distribution on the net to drive it to a desired configuration. This is accomplished using principles from control theory. Doing so would allow for greater accuracy and efficiency in capturing an object in addition to providing a design problem of the best approach to developing the net. Also once captured, the captured object could then be further manipulated with these additional control points. One notable advantage to this proposed method compared to other net models is that no assumptions are made to what additional forces act on the masses; as such, it takes no additional work to incorporate control thrusters onto the masses.

1.5 Outline

The first technical chapter, Chapter 2, in this thesis describes the mathematical model of a multibody system, i.e. the net/body system. In it, the exact derivation of the model is gone through, along with definition of terms and variables. Chapter 2 is used as a foundation for the implementation found in the next chapter. The computational implementation of the mathematics is described in Chapter 3. It discussed issues caused by ordinary differential equation solvers, along with a solutions to correct them. In Chapter 4, two primary control methods, optimal con-

trol and error stabilization, are derived and implemented. Some simulations are shown along with some discussion of the utility of both.

2. Modeling

2.1 Multibody System Dynamics

An important aspect to many robotic, manufacturing, deployment, mobility, and automation systems is the use of multibody systems. Multibody dynamic systems, as the name implies, are systems that model the interaction of multiple, usually rigid, bodies. They have a multitude of uses ranging from design to mechanical actuation. As such, there are many methods to analyse multibody dynamics [10].

These methodologies include, but are not limited to, recursive methods [11, 12, 13, 14] and order N formulations [15, 16, 17, 18]. New considerations are emerging, such as flexible bodies, to address the pervasive and useful nature of these systems. In order to deal with the complex nature of theses types of problems, automatic model generation based on a Newton-Lagrange formulations solution have been developed. These models have focused primarily efficient data structures in order to deal with the model generation. Order N dynamics and others are current ways of solving multibody problems. None of these consider inequality constraints associated with the generalized coordinates. Inequality constraints are studied in this work because of their great utility in fields such as tensegrity. Tensegrity system models currently have no ability to consider conditionally taut strings. Giving these models the ability to have strings only when they are needing to be taught is a great advantage in structures that need a greater range of motion or to have minimal control actuators. This would be particularly helpful in deployable structures. Mathematical modeling of systems governing the configuration of multiple rigid and flexible bodies plays an important role in the design, actuation and effective operation of mechanical and aerospace engineering systems [10]. In engineering literature this subject is called multibody dynamics. Various applications in science and engineering find great use of multibody system dynamics. As a result, it is the basis for a thriving research field with a multitude of methods and theories.

This research will consider the equations of motion for a net of an arbitrary number of points

connected by strings that can become slack when the distance between a given two points are below the specified value for the string of interest. In addition, this research will include a set of inequality constraints to determine dynamics of the net's collision with a convex body. A convex body is used for two reasons; 1) most structures in space are convex, and 2) strings in a network used for coverage applications will stretch across a concavity and the dynamics remain mostly the same. Convex polytopes will be used to approximate any body needing to be analyzed.

2.2 Method Approach

To simulate the dynamics of both the net and its interaction with the body, an unconstrained dynamic model is calculated first and a Lagrange multiplier is added to the acceleration of the points to constrain the dynamics within the inequality constraints. Then, to find the Lagrange multiplier, one need only substitute the unconstrained dynamics in the second derivative of the constraint equation. In order to consider the effects the net has on the body, the same reaction force will need to be added to the body's acceleration.

To accurately model the complex interactions in this thesis, a set of definitions for the individual dynamics of a set of points will be needed first. A set of unconstrained points can be written as:

$$\dot{\boldsymbol{r}}_i = \boldsymbol{v}_i \qquad i = 1, 2, \ldots, N_p \tag{2.1}$$

$$m_i \dot{\boldsymbol{v}}_i = \boldsymbol{f}(\boldsymbol{r}, \boldsymbol{v}, t) \tag{2.2}$$

where $\boldsymbol{r}_i$ is a vector describing the position of the i^{th} point and $\boldsymbol{v}_i$ is a vector describing the velocity of the i^{th} point; $\boldsymbol{f}$ is an arbitrary force vector that can be a function of time and any point's position and velocity; and m_i is the mass of the i^{th} point.

Stacking the vectors vertically, we obtain a set of differential equations describing unconstrained point mass dynamics.

$$\dot{\boldsymbol{R}} = \boldsymbol{V} \tag{2.3}$$

$$M\dot{\boldsymbol{V}} = \boldsymbol{F}(\boldsymbol{R}, \boldsymbol{V}, t) \tag{2.4}$$

where $\boldsymbol{R}$ is a stacked set of point mass positions and $\boldsymbol{V}$ is the same for velocity. Similarly, $\boldsymbol{F}$ is a stacked force function and M is a diagonal mass matrix:

$$M_i = \begin{bmatrix} m_i & 0 & 0 \\ 0 & m_i & 0 \\ 0 & 0 & m_i \end{bmatrix} \quad M = \begin{bmatrix} M_1 & 0 & \dots & 0 \\ 0 & M_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & M_{N_p} \end{bmatrix}$$

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \vdots \\ \boldsymbol{r}_{N_p} \end{bmatrix} \qquad \boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \vdots \\ \boldsymbol{v}_{N_p} \end{bmatrix}$$

2.3 Dynamics Modeling Formulation

In order to hold a set constraints, a set of Lagrange multipliers will be added as reaction forces:

$$M\dot{\boldsymbol{V}} = \boldsymbol{F}(\boldsymbol{R}, \boldsymbol{V}, t) + \Theta_R \Lambda \tag{2.5}$$

where Θ_R is the partial of a vector Θ, which is a vector of all active constraints, with respect to the stacked point position vector R.

2.3.1 String Constraint

For only a set of string constraints, Φ will be used for the constraint vector instead of Θ; and λ will be used as the string Lagrance multipliers instead of Λ:

$$M\dot{\boldsymbol{V}} = \boldsymbol{F}(\boldsymbol{R}, \boldsymbol{V}, t) + \Phi_R \lambda \tag{2.6}$$

where Φ_R is the partial of a vector Φ with respect to the stacked point position vector R.
Next, a definition is needed for a slack string assumption. The string condition can be stated as follows:

$$\phi_i = S_i^T S_i - L_i^2 \qquad i = 1, 2, \ldots, N_s \tag{2.7}$$

$$\begin{cases} \phi_i < 0 & \text{slack} \\ \phi_i = 0 & \text{taut} \end{cases} \tag{2.8}$$

where L_i is the length of the unloaded string. N_s is the total number of strings. It is assumed that the string has no effect on the dynamics when it is slack.
S_i is the vector connecting two points where a string is attached,

$$S_i = C_i \boldsymbol{R} \tag{2.9}$$

where C_i is matrix filled with only $1, -1, 0$ such that the resultant row is the difference between the points that are connected by a string.
For convenience a new variable is created for a derivative later:

$$Q_i = 2C_i^T C_i \tag{2.10}$$

With this, the string constraint looks like:

$$\phi_i = \frac{1}{2}\boldsymbol{R}^T Q_i \boldsymbol{R} - L_i^2 \tag{2.11}$$

From here, an important thing to realize is that ϕ_i, defined in Equation (2.11), is zero when the constraint is active. This also implies that $\dot{\phi}_i$ is also zero when on the constraint. In fact, any subsequent time derivative, $\ddot{\phi}_i, \dddot{\phi}_i, \ldots$, is also zero when on the constraint. This can be used to find the dynamics of the points while the constraint is active. It can then be used to find the λ's. Hence:

$$\dot{\phi}_i = \frac{\partial \phi}{\partial \boldsymbol{R}}^T \frac{\partial \boldsymbol{R}}{\partial t} = 0 \tag{2.12}$$

$$= (Q_i \boldsymbol{R})^T \boldsymbol{V} = 0 \tag{2.13}$$

Stacked, this can be written for all string constraints:

$$\dot{\Phi} = \Phi_R^T \boldsymbol{V} = 0 \tag{2.14}$$

Taking another derivative:

$$\ddot{\Phi} = \Phi_R^T \dot{\boldsymbol{V}} + \dot{\Phi}_R^T \boldsymbol{V} = 0 \tag{2.15}$$

where:

$$\Phi_R = \begin{bmatrix} Q_1 \boldsymbol{R} & Q_2 \boldsymbol{R} & \ldots & Q_{N_s} \boldsymbol{R} \end{bmatrix} \tag{2.16}$$

$$\dot{\Phi}_R = \begin{bmatrix} Q_1 \boldsymbol{V} & Q_2 \boldsymbol{V} & \ldots & Q_{N_s} \boldsymbol{V} \end{bmatrix} \tag{2.17}$$

Substituting $\dot{\boldsymbol{V}}$ from Equation (2.6) into Equation (2.15) gives:

$$\ddot{\Phi} = \Phi_R^T(M^{-1}\boldsymbol{F}(\boldsymbol{R},\boldsymbol{V},t) + M^{-1}\Phi_R\lambda) + \dot{\Phi}_R^T\boldsymbol{V} = 0 \tag{2.18}$$

$$= \Phi_R^T M^{-1}\boldsymbol{F}(\boldsymbol{R},\boldsymbol{V},t) + \Phi_R^T M^{-1}\Phi_R\lambda + \dot{\Phi}_R^T\boldsymbol{V} = 0 \tag{2.19}$$

$$= (\Phi_R^T M^{-1}\Phi_R)\lambda + \Phi_R^T M^{-1}\boldsymbol{F}(\boldsymbol{R},\boldsymbol{V},t) + \dot{\Phi}_R^T\boldsymbol{V} = 0 \tag{2.20}$$

Solving for λ gives:

$$\lambda = -(\Phi_R^T M^{-1}\Phi_R)^{-1}(\Phi_R^T M^{-1}\boldsymbol{F}(\boldsymbol{R},\boldsymbol{V},t) + \dot{\Phi}_R^T\boldsymbol{V}) \tag{2.21}$$

This, with the constrained dynamics (2.6), gives the full dynamics of a net. An important note is that λ is zero when not at the constraint, meaning that (2.21) gives only the values of λ that are at the constraint. One additional consideration would be that λ becomes undefined as $(\Phi_R^T M^{-1}\Phi_R)$ becomes singular, which only happens when at least two points exist at the same position. Having described the modeling of the network of strings, the rigid body constraint specification are now discussed.

2.3.2 Rigid Body Constraint

In order to simulate a net capturing a body, a body constraint must be added to the equations of motion. First some definitions are presented to assist in the derivation. The fact that the network of points cannot penetrate the rigid body can be described by:

$$\psi_j = r(\rho_j)^2 - \rho_j^T\rho_j \qquad j = 1, 2, \ldots, N_p \tag{2.22}$$

$$\rho_j = R_j - P \tag{2.23}$$

$$\begin{cases} \psi_j < 0 & \text{free} \\ \psi_j = 0 & \text{collision} \end{cases} \tag{2.24}$$

where R_j is the position of any given point on the net and P is the position of the center of mass of the body, as seen in Fig. 2.1. N_p is the total number of points in the net.

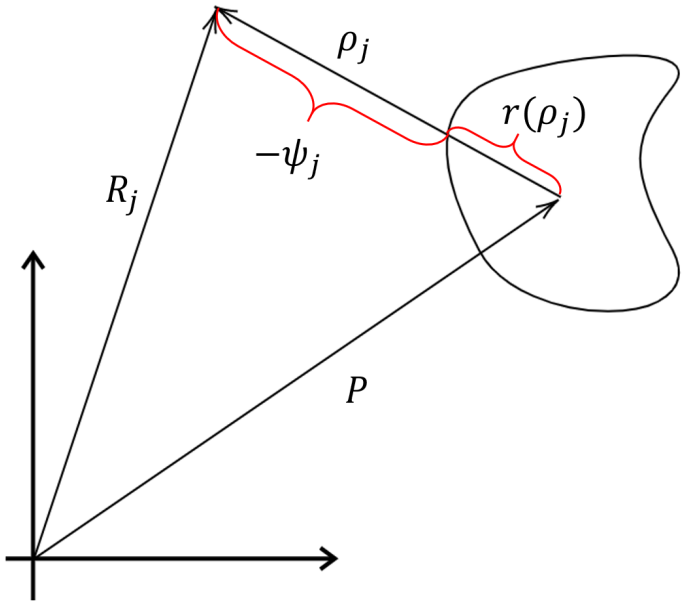


Figure 2.1: Rigid Body Constraint Visualized

It is important to note that $r(\rho_j)$ is a function describing the distance from the center of mass to the surface of the object. Since most objects do not have a simple equation describing its surface, a discretized planar solution will be presented in the next section. Here, though, it will simply be assumed that a $r(\rho_j)$ is known.

$$\dot{\psi}_j = \frac{\partial \psi_j}{\partial \rho_j}^T \frac{\partial \rho_j}{\partial t} \tag{2.25}$$

$$\frac{\partial \psi_j}{\partial \rho_j} = 2r(\rho_j)\frac{\partial r(\rho_j)}{\partial \rho_j} - 2\rho_j \tag{2.26}$$

$$\frac{\partial \rho_j}{\partial t} = \boldsymbol{V}_j - \dot{P} \tag{2.27}$$

$$\tag{2.28}$$

It is important to note that for this derivation the body is assumed to be not rotating. Later when convex polytopes are used, this assumption will be relaxed.

Stacking all the partials it can be redefined as such.

$$\Psi_\rho = \begin{bmatrix} 2r(\rho_1)\frac{\partial r(\rho_1)}{\partial \rho_1} - 2\rho_1 & 0 & \dots & 0 \\ 0 & 2r(\rho_2)\frac{\partial r(\rho_2)}{\partial \rho_2} - 2\rho_2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \dots & 0 & 2r(\rho_{N_p})\frac{\partial r(\rho_{N_p})}{\partial \rho_{N_p}} - 2\rho_{N_p} \end{bmatrix} \tag{2.29}$$

$$\dot{\Psi} = \Psi_\rho^T(\boldsymbol{V} - \dot{\tilde{\boldsymbol{P}}}) \tag{2.30}$$

$$\tilde{\boldsymbol{P}} = \begin{bmatrix} P \\ P \\ \vdots \\ P \end{bmatrix} \implies \dot{\tilde{\boldsymbol{P}}} = \begin{bmatrix} \dot{P} \\ \dot{P} \\ \vdots \\ \dot{P} \end{bmatrix} \implies \ddot{\tilde{\boldsymbol{P}}} = \begin{bmatrix} \ddot{P} \\ \ddot{P} \\ \vdots \\ \ddot{P} \end{bmatrix} \tag{2.31}$$

$$\ddot{\Psi} = \Psi_\rho^T(\dot{\boldsymbol{V}} - \ddot{\tilde{\boldsymbol{P}}}) + \dot{\Psi}_\rho^T(\boldsymbol{V} - \dot{\tilde{\boldsymbol{P}}}) \tag{2.32}$$

$$= \Psi_\rho^T\dot{\boldsymbol{V}} + \dot{\Psi}_\rho^T\boldsymbol{V} + \tilde{\boldsymbol{B}} \tag{2.33}$$

Since both the rigid body assumption and the slack string assumption can be active at the same time, they will be stacked and treated as one list of constraints.

$$\Theta = \begin{bmatrix} \Phi \\ \\ \Psi \end{bmatrix} \tag{2.34}$$

$$\ddot{\Theta} = \Theta_R^T \dot{\boldsymbol{V}} + \dot{\Theta}_R^T \boldsymbol{V} + \boldsymbol{B} \tag{2.35}$$

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{0} \\ \tilde{\boldsymbol{B}} \end{bmatrix} \tag{2.36}$$

$$\Theta_R = \begin{bmatrix} \Phi_R & \Psi_\rho \end{bmatrix} \tag{2.37}$$

Additionally a new set of Lagrange multipliers will need be added.

$$M\dot{\boldsymbol{V}} = \boldsymbol{F}(\boldsymbol{R}, \boldsymbol{V}, t) + \Phi_R \lambda + \Psi_\rho \mu \tag{2.38}$$

$$M\dot{\boldsymbol{V}} = \boldsymbol{F}(\boldsymbol{R}, \boldsymbol{V}, t) + \Theta_R \Lambda \tag{2.39}$$

Substituting (2.39) into (2.35) gives a solution for all Lagrange multipliers.

$$\ddot{\Theta} = \Theta_R^T (M^{-1}\boldsymbol{F}(\boldsymbol{R}, \boldsymbol{V}, t) + M^{-1}\Theta_R \Lambda) + \dot{\Theta}_R^T \boldsymbol{V} + \boldsymbol{B} = 0 \tag{2.40}$$

$$= \Theta_R^T M^{-1}\boldsymbol{F}(\boldsymbol{R}, \boldsymbol{V}, t) + \Theta_R^T M^{-1}\Theta_R \Lambda + \dot{\Theta}_R^T \boldsymbol{V} + \boldsymbol{B} = 0 \tag{2.41}$$

$$\Lambda = -(\Theta_R^T M^{-1}\Theta_R)^{-1}(\Theta_R^T M^{-1}\boldsymbol{F}(\boldsymbol{R}, \boldsymbol{V}, t) + \dot{\Theta}_R^T \boldsymbol{V} + \boldsymbol{B}) \tag{2.42}$$

Λ becomes undefined as $(\Theta_R^T M^{-1}\Theta_R)$ becomes singular, which only happens when at least two points exist at the same position or a point exist at the center of mass of the colliding body.

Since this solution requires one to know an exact function $r(\rho_j)$, there needs to be a way to find it. In the following section, a discretized solution, called a convex polytope, is presented to generate this function.

2.4 Convex Polytopes and Planar Constraint

To simplify any given shape of the captured body, it can be estimated as a convex polytope, a set of intersecting planes. Therefore, if the body constraint were done assuming a planar body, all that need be done is to track which plane a point may intersect.

To begin, a discussion on planar intersection is presented.

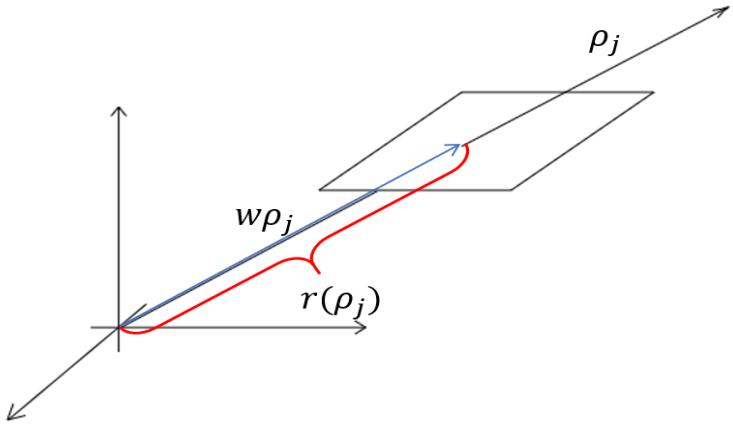


Figure 2.2: Planar Intersection

The scalar distance between the center of mass of the object and the surface of the plane, between the point of interest and the center of mass, can be found as follows.

The definition of a plane can be written as:

$$\mathcal{P} = ax + by + cz = 1 \tag{2.43}$$

$$= A^T X = 1 \tag{2.44}$$

$$A = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad X = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.45}$$

where A can be thought of as the normal vector of the plane from the center of mass of the body to the plane.

For this problem, X is a vector from the center of mass of the object to a point on the plane along the vector ρ_j. It can be written as a scalar multiplied by the ρ_j vector:

$$X = w\rho_j \tag{2.46}$$

where w is a scaling term such that X reaches the plane. This value is is known as a function of $r(\rho_j)$, since X needs to have a magnitude of $r(\rho_j)$:

$$w = \frac{r(\rho_j)}{\|\rho_j\|} \tag{2.47}$$

From here $r(\rho_j)$ can be found:

$$A^T \rho_j w = 1 \tag{2.48}$$

$$r(\rho_j) = \frac{\|\rho_j\|}{A^T \rho_j} \tag{2.49}$$

Substituting into the body constraint and rearranging:

$$\psi_j = \frac{\rho_j^T \rho_j}{\rho_j^T A A^T \rho_j} - \rho_j^T \rho_j \tag{2.50}$$

$$\frac{\psi_j \rho_j^T A A^T \rho_j}{\rho_j^T \rho_j} = 1 - \rho_j^T A A^T \rho_j \tag{2.51}$$

$$\psi_j' = 1 - \rho_j^T A A^T \rho_j \tag{2.52}$$

It can be seen that the scaling factor multiplying ψ_j in (2.51) is always positive, because the magnitude of A is constant and ρ_j is only zero when a point mass exists at the center of mass. Therefore, the constraint can be redefined as (2.52). From here ψ_j' will simply be written as ψ_j for convenience. A new set of Lagrange multipliers can be derived while allowing the body to rotate.

$$b_p = C^T A_p \tag{2.53}$$

$$\dot{\psi}_j = -2(\rho_j^T b_p b_p^T \dot{\rho}_j + b_p^T \rho_j \rho_j^T \dot{b}_p) \tag{2.54}$$

where C is a transformation matrix, allowing for rotational dynamics, and the subscript p denoting the closest plane to a given point, i.

Following past derivations:

$$\ddot{\psi}_j = \dot{\psi}_\rho^T \dot{\rho}_j + \psi_\rho^T \ddot{\rho}_j + \bar{B}_j \tag{2.55}$$

$$\dot{\psi}_\rho = -2\left[b_p b_p^T \dot{\rho}_j + (b_p \dot{b}_p^T + \dot{b}_p b_p^T)\rho_j\right] \tag{2.56}$$

$$\psi_\rho = -2 b_p b_p^T \rho_j \tag{2.57}$$

$$\bar{B}_j = -2\left[\dot{b}_p^T \rho_j \rho_j^T \dot{b}_p + b_p^T(\rho_j \dot{\rho}_j^T + \dot{\rho}_j \rho_j^T)\dot{b}_p + b_p^T \rho_j \rho_j^T \ddot{b}_p\right] \tag{2.58}$$

$$\ddot{\Psi} = \Psi_\rho^T(\dot{\boldsymbol{V}} - \ddot{\tilde{\boldsymbol{P}}}) + \dot{\Psi}_\rho^T(\boldsymbol{V} - \dot{\tilde{\boldsymbol{P}}}) + \bar{\boldsymbol{B}} \tag{2.59}$$

$$\Psi_\rho = \begin{bmatrix} -2b_p b_p^T \rho_1 & 0 & \dots & 0 \\ 0 & -2b_p b_p^T \rho_2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \dots & 0 & -2b_p b_p^T \rho_{N_p} \end{bmatrix} \tag{2.60}$$

$$\bar{\boldsymbol{B}} = -2\begin{bmatrix} \dot{b}_p^T \rho_1 \rho_1^T \dot{b}_p + b_p^T(\rho_1 \dot{\rho}_1^T + \dot{\rho}_1 \rho_1^T)\dot{b}_p + b_p^T \rho_1 \rho_1^T \ddot{b}_p \\ \dot{b}_p^T \rho_2 \rho_2^T \dot{b}_p + b_p^T(\rho_2 \dot{\rho}_2^T + \dot{\rho}_2 \rho_2^T)\dot{b}_p + b_p^T \rho_2 \rho_2^T \ddot{b}_p \\ \vdots \\ \dot{b}_p^T \rho_{N_p} \rho_{N_p}^T \dot{b}_p + b_p^T(\rho_{N_p} \dot{\rho}_{N_p}^T + \dot{\rho}_{N_p} \rho_{N_p}^T)\dot{b}_p + b_p^T \rho_{N_p} \rho_{N_p}^T \ddot{b}_p \end{bmatrix} \tag{2.61}$$

Simplifying to isolate $\dot{\boldsymbol{V}}$:

$$\ddot{\Psi} = \Psi_\rho^T \dot{\boldsymbol{V}} + \dot{\Psi}_\rho^T \boldsymbol{V} + \tilde{\boldsymbol{B}} \tag{2.62}$$

$$\tilde{\boldsymbol{B}} = \bar{\boldsymbol{B}} - \Psi_\rho^T \ddot{\tilde{\boldsymbol{P}}} - \dot{\Psi}_\rho^T \dot{\tilde{\boldsymbol{P}}} \tag{2.63}$$

Combining with Φ and solving with new $\boldsymbol{B}$:

$$\ddot{\Theta} = \Theta_R^T \dot{\boldsymbol{V}} + \dot{\Theta}_R^T \boldsymbol{V} + \boldsymbol{B} \tag{2.64}$$

$$\ddot{\Theta} = \Theta_R^T (M^{-1} \boldsymbol{F}(\boldsymbol{R}, \boldsymbol{V}, t) + M^{-1} \Theta_R \Lambda) + \dot{\Theta}_R^T \boldsymbol{V} + \boldsymbol{B} = 0 \tag{2.65}$$

$$= \Theta_R^T M^{-1} \boldsymbol{F}(\boldsymbol{R}, \boldsymbol{V}, t) + \Theta_R^T M^{-1} \Theta_R \Lambda + \dot{\Theta}_R^T \boldsymbol{V} + \boldsymbol{B} = 0 \tag{2.66}$$

$$\Lambda = -(\Theta_R^T M^{-1} \Theta_R)^{-1} (\Theta_R^T M^{-1} \boldsymbol{F}(\boldsymbol{R}, \boldsymbol{V}, t) + \dot{\Theta}_R^T \boldsymbol{V} + \boldsymbol{B}) \tag{2.67}$$

While (2.42) looks the same as (2.67), it is important to note that Θ_R,$\dot{\Theta}_R$, and $\boldsymbol{B}$ contain many additional terms.

One notable limitation to this solution is that no consideration was given to the effect the net has on the dynamics of $\tilde{\boldsymbol{P}}$ or b_p. However, this consideration is included in the next sections.

2.5 Body Interactions

For a rigid body, there are only two types of movement: translation and rotation. To accurately simulate the network interacting with the body, both types must be considered. While its is relatively simple to just add the forces and torques that the points caused by interacting with the surface of the rigid body, it crucially ignores how this changes the the effect of the Lagrange multipliers. Moreover, the body and the network are a system largely determined by the constraints being held. This means that, in order to accurately capture the dynamics of the system, the Lagrange multipliers must be resolved for both types of rigid body movement.

2.5.1 Translation

Starting from equation (2.59) and a definition of $\ddot{P}$:

$$\ddot{\Psi} = \Psi_\rho^T(\dot{\boldsymbol{V}} - \ddot{\tilde{\boldsymbol{P}}}) + \dot{\Psi}_\rho^T(\boldsymbol{V} - \dot{\tilde{\boldsymbol{P}}}) + \bar{\boldsymbol{B}} \tag{2.68}$$

$$\ddot{P} = \frac{1}{m_P} f_P(P, \dot{P}, t) - \frac{1}{m_P} \Sigma \Psi_\rho H \Lambda \tag{2.69}$$

$$\Sigma = \begin{bmatrix} I_3 & I_3 & \dots & I_3 \end{bmatrix} \tag{2.70}$$

$$H = \begin{bmatrix} \mathbf{0}_{N_p \times N_s} & I_{N_p} \end{bmatrix} \tag{2.71}$$

where m_P is the mass of the object and f_P is the sum of forces applied on the object from other sources. Here Σ acts to sum all of the reaction force vectors from the points onto the body.

Resolving for Λ:

$$\ddot{\Psi} = \Psi_\rho^T[\dot{\boldsymbol{V}} - (\frac{1}{m_P}\tilde{\boldsymbol{f}}_P(P,\dot{P},t) - \bar{\boldsymbol{F}}_R\Lambda)] + \dot{\Psi}_\rho^T(\boldsymbol{V} - \dot{\tilde{\boldsymbol{P}}}) + \bar{\boldsymbol{B}} \tag{2.72}$$

$$\bar{\boldsymbol{F}}_R = \frac{1}{m_P}\begin{bmatrix} \Sigma\Psi_\rho H \\ \Sigma\Psi_\rho H \\ \vdots \\ \Sigma\Psi_\rho H \end{bmatrix} \tag{2.73}$$

$$\ddot{\Psi} = \Psi_\rho^T\dot{\boldsymbol{V}} + \dot{\Psi}_\rho^T\boldsymbol{V} + \tilde{\boldsymbol{F}}_R\Lambda + \tilde{\boldsymbol{B}} \tag{2.74}$$

$$\tilde{\boldsymbol{B}} = \bar{\boldsymbol{B}} - \frac{1}{m_P}\Psi_\rho^T\tilde{\boldsymbol{f}}_P(P,\dot{P},T) - \dot{\Psi}_\rho^T\dot{\tilde{\boldsymbol{P}}} \tag{2.75}$$

$$\tilde{\boldsymbol{F}}_R = \Psi_\rho^T\bar{\boldsymbol{F}}_R \tag{2.76}$$

$$\ddot{\Theta} = \Theta_R^T\dot{\boldsymbol{V}} + \dot{\Theta}_R^T\boldsymbol{V} + \boldsymbol{F}_R\Lambda + \boldsymbol{B} \tag{2.77}$$

$$\boldsymbol{F}_R = \begin{bmatrix} \boldsymbol{0} \\ \tilde{\boldsymbol{F}}_R \end{bmatrix} \tag{2.78}$$

$$\ddot{\Theta} = \Theta_R^T(M^{-1}\boldsymbol{F}(\boldsymbol{R},\boldsymbol{V},t) + M^{-1}\Theta_R\Lambda) + \dot{\Theta}_R^T\boldsymbol{V} + \boldsymbol{F}_R\Lambda + \boldsymbol{B} = 0 \tag{2.79}$$

$$= \Theta_R^T M^{-1}\boldsymbol{F}(\boldsymbol{R},\boldsymbol{V},t) + \Theta_R^T M^{-1}\Theta_R\Lambda + \boldsymbol{F}_R\Lambda + \dot{\Theta}_R^T\boldsymbol{V} + \boldsymbol{B} = 0 \tag{2.80}$$

$$\Lambda = -(\Theta_R^T M^{-1}\Theta_R + \boldsymbol{F}_R)^{-1}(\Theta_R^T M^{-1}\boldsymbol{F}(\boldsymbol{R},\boldsymbol{V},t) + \dot{\Theta}_R^T\boldsymbol{V} + \boldsymbol{B}) \tag{2.81}$$

2.5.2 Rotation

The next derivation will introduce the effects of the net on the body's rotation. It will start with a simple application of the transport theorem on the vectors that define the planes[19].

$$\dot{b}_p = \omega \times b_p = \omega^{\times} b_p \tag{2.82}$$

$$\omega^{\times} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{2.83}$$

$$\ddot{b}_p = (\dot{\omega}^{\times} + 2\omega^{\times}\omega^{\times}) b_p \tag{2.84}$$

where b_p is a vector whose coordinates are expressed in the body frame. It can be seen that the dynamics of all b_p's are only dependent on a single ω since all planes are part of the same rigid body. Hence, the dynamics of ω are governed by that of the rigid body interacting with the net.

$$h = J\omega \tag{2.85}$$

$$\dot{h} = J\dot{\omega} + \omega^{\times}(J\omega) \tag{2.86}$$

$$\dot{h} = l = (-F_P)^{\times}\rho = \rho^{\times}(\Psi_{\rho} H \Lambda) \tag{2.87}$$

$$\rho_j^{\times} = \begin{bmatrix} 0 & -\rho_{jz} & \rho_{jy} \\ \rho_{jz} & 0 & -\rho_x \\ -\rho_{jy} & \rho_{jx} & 0 \end{bmatrix} \tag{2.88}$$

$$\rho^{\times} = \begin{bmatrix} \rho_1^{\times} & \rho_2^{\times} \dots \rho_{N_p}^{\times} \end{bmatrix} \tag{2.89}$$

Note that F_P is the reaction force of the net acting on the body and here is only comprised of the normal component from the constraint force ($\Psi_{\rho} H \Lambda$) and ρ_i is the position vector of the ith particle expressed in the inertial frame. Here $\rho^{\times}$ acts to both take a cross product of each force vector and also to sum all of the torques acting on the body.

Isolating $\dot{\omega}$ and substituting in $\ddot{b}_p$, the interaction dynamics are written as:

$$\dot{\omega} = J^{-1}(\rho^{\times}\Psi_{\rho}H\Lambda - \omega^{\times}J\omega) \tag{2.90}$$

$$\ddot{b}_p = -b_p^{\times}\dot{\omega} + 2\omega^{\times}\omega^{\times}b_p \tag{2.91}$$

$$= -b_p^{\times}J^{-1}(\rho^{\times}\Psi_{\rho}H\Lambda - \omega^{\times}J\omega) + 2\omega^{\times}\omega^{\times}b_p \tag{2.92}$$

$$\ddot{b}_p = -b_p^{\times}J^{-1}\rho^{\times}\Psi_{\rho}H\Lambda + b_p^{\times}J^{-1}\omega^{\times}J\omega + 2\omega^{\times}\omega^{\times}b_p \tag{2.93}$$

From equation (2.59):

$$\ddot{\Psi} = \Psi_{\rho}^{T}(\dot{\boldsymbol{V}} - \ddot{\tilde{\boldsymbol{P}}}) + \dot{\Psi}_{\rho}^{T}(\boldsymbol{V} - \dot{\tilde{\boldsymbol{P}}}) + \Psi_{b}^{T}\ddot{\boldsymbol{b}} + \bar{\boldsymbol{B}} \tag{2.94}$$

$$\Psi_b = \begin{bmatrix} -2\rho_1\rho_1^T b_p & 0 & \dots & 0 \\ 0 & -2\rho_2\rho_2^T b_p & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \dots & 0 & -2\rho_{N_p}\rho_{N_p}^T b_p \end{bmatrix} \tag{2.95}$$

$$\bar{\boldsymbol{B}} = -2\begin{bmatrix} \dot{b}_p^T\rho_1\rho_1^T\dot{b}_p + b_p^T(\rho_1\dot{\rho}_1^T + \dot{\rho}_1\rho_1^T)\dot{b}_p \\ \dot{b}_p^T\rho_2\rho_2^T\dot{b}_p + b_p^T(\rho_2\dot{\rho}_2^T + \dot{\rho}_2\rho_2^T)\dot{b}_p \\ \vdots \\ \dot{b}_p^T\rho_{N_p}\rho_{N_p}^T\dot{b}_p + b_p^T(\rho_{N_p}\dot{\rho}_{N_p}^T + \dot{\rho}_{N_p}\rho_{N_p}^T)\dot{b}_p \end{bmatrix} \tag{2.96}$$

where $\ddot{\boldsymbol{b}}$ is a stacked list of each point's associated plane vector, meaning that it is the same size as R. From equation (2.74)

$$\ddot{\Psi} = \Psi_\rho^T(\dot{\boldsymbol{V}} - \ddot{\tilde{\boldsymbol{P}}}) + \dot{\Psi}_\rho^T(\boldsymbol{V} - \dot{\tilde{\boldsymbol{P}}}) + \Psi_b^T \bar{L}_R \Lambda + \bar{\boldsymbol{B}} \tag{2.97}$$

$$\bar{\boldsymbol{B}} = -2 \begin{bmatrix} \dot{b}_p^T \rho_1 \rho_1^T \dot{b}_p + b_p^T(\rho_1 \dot{\rho}_1^T + \dot{\rho}_1 \rho_1^T)\dot{b}_p + b_p^T \rho_1 \rho_1^T (b_p^\times J^{-1} \omega^\times J\omega + 2\omega^\times \omega^\times b_p) \\ \dot{b}_p^T \rho_2 \rho_2^T \dot{b}_p + b_p^T(\rho_2 \dot{\rho}_2^T + \dot{\rho}_2 \rho_2^T)\dot{b}_p + b_p^T \rho_2 \rho_2^T (b_p^\times J^{-1} \omega^\times J\omega + 2\omega^\times \omega^\times b_p) \\ \vdots \\ \dot{b}_p^T \rho_{N_p} \rho_{N_p}^T \dot{b}_p + b_p^T(\rho_{N_p} \dot{\rho}_{N_p}^T + \dot{\rho}_{N_p} \rho_{N_p}^T)\dot{b}_p + b_p^T \rho_{N_p} \rho_{N_p}^T (b_p^\times J^{-1} \omega^\times J\omega + 2\omega^\times \omega^\times b_p) \end{bmatrix} \tag{2.98}$$

$$\bar{L}_R = \begin{bmatrix} -b_p^\times J^{-1} \rho^\times \Psi_\rho H \\ -b_p^\times J^{-1} \rho^\times \Psi_\rho H \\ \vdots \\ -b_p^\times J^{-1} \rho^\times \Psi_\rho H \end{bmatrix} \tag{2.99}$$

$$\ddot{\Psi} = \Psi_\rho^T \dot{\boldsymbol{V}} + \dot{\Psi}_\rho^T \boldsymbol{V} + \tilde{F}_R \Lambda + \tilde{L}_R \Lambda + \tilde{\boldsymbol{B}} \tag{2.100}$$

$$\tilde{L}_R = \Psi_b^T \bar{L}_R \tag{2.101}$$

Resolving for Λ:

$$\Lambda = -(\Theta_R^T M^{-1} \Theta_R + F_R + L_R)^{-1}(\Theta_R^T M^{-1} \boldsymbol{F}(\boldsymbol{R}, \boldsymbol{V}, t) + \dot{\Theta}_R^T \boldsymbol{V} + \boldsymbol{B}) \tag{2.102}$$

$$L_R = \begin{bmatrix} \mathbf{0} \\ \tilde{L}_R \end{bmatrix} \tag{2.103}$$

These equations need to be interpreted along with the dynamics of the array of particles.

2.6 Summary

While modeling multibody dynamic systems is not a new field of study, increasing computational resources enable new models to emerge. Many already exist, however none in the literature address the complex nature of nonholonomic constraints. This research shows one method for modeling inequality constraints in the form of a set of interconnected strings and a rigid body.

While this research is by no means all encompassing, it is greatly suited for modeling a net deorbiting a rigid body. It can be seen that this method also has a unique property, it is well suited for open loop control laws to be fed into $F(t, R, V)$ in (2.4). These methods also consider the interaction of the point masses on the body, implicitly. While there are many more things to consider, such as computation, control, and observation, this chapter provides a framework for subsequent developments. The following chapter will deal with the implementation of this theory in the form of a set of simulations.

3. High Performance Computing Implementation

3.1 Motivation

The dynamic models developed in Chapter 2 take the form of a set of conditional nonlinear differential equations. While these models are theoretically robust to describe the system, they are mostly meaningless without computational implementation. As will be seen later in this chapter, implementation of the dynamic model has unique problems that the mathematical model does not implicitly have. In order to use this model, it must be solved for computationally.

These simulations made from these models should capture the physical process in order to be considered viable for implementation. They should also hold the modeling principles to be true. In other words, the simulated system must both be a reflection of the real world processes and hold the constraints of the problem to be true.

3.2 Structure

To demonstrate the veracity of this work a Matlab code was written. This code uses an ordinary differential equation solver, "ode45," to solve for the state time history of the net dynamics using this method's differential equation. The solver was chosen for its efficient computation and robustness for solving complex nonlinear differential equations. In order to implement this method, several computation specific problems need to be addressed. These are, in order of implementation, computational overshoot, convex polytopes, Lagrange multiplier omissions, and collision detection and holding.

The code is a fully functional dynamic simulation that can be set to demonstrate a certain aspect of the dynamic model. To start this code, a definition of basic parameters and initial conditions is needed. For most simulations seen here, the net starts with all point masses being at the same point and having velocities that send it towards the desired target. Also defined at the start of the code are the specifics of the net/body configuration. One important parameter is the definition of C_i in equation (2.9). For simplicity, except for the simple two point problem, all dynamics were

simulated with a square grid of points with strings connecting laterally and transversely. Then the initial velocity required to fully deploy the net is calculated for each point.

The rigid body has two types of information needing to be defined or found in order to model the object. The first are the degrees of freedom, and the initial conditions associated, namely three for position, three for velocity, three (or four in the case of quaternions) for attitude, and three for angular velocity. The second is information about the shape of the structure. Since it is assumed that the object is modeled as a convex polytope, all that is needed is to find the parameters that define each plane of the polytope. The details of this is discussed in the convex polytope section, but it primarily consists of defining at least three points on each plane. They can then be rotated to an initial attitude using equation (2.53).

After all of these values are generated or defined, they are all stored in a data structure and fed into the main dynamics function. The dynamics function takes the initial states and parameters and runs an "ode45" solver to generate time histories of the states. The solver requires a differential equation to define the dynamics. This sub-function is implemented using equations (2.39, 2.69, 2.90), along with the detection logic associated with the constraints. All necessary variables are explicitly defined and detailed later on in this chapter. The output of this dynamics function is a time history of position and velocity of the set of discrete points and the rigid body, along with the attitude and angular velocity of the rigid body. Trajectories obtained from representative simulations are shown in plots. Separate plots are used to show constraint subfunction time histories. Final position of the net and body are shown in a geometrical plot shown in three dimensions (3D) separately. Then a video is generated to show the simulation in real time. All of these are helpful in seeing that this model, and simulation, work. Since video cannot be shown here, snapshots will used to show time progression.

3.3 Error Correction

The first thing to address is best seen in a simple problem, two masses joined by a single string, i.e. a bola. Bolas have their own history and uses in fields ranging from hunting to law enforcement. The following example uses the equations derived in Chapter 2 and compares the dynamics to an analytical solution derived in Appendix I. The bola starts with the two points separated by a distance smaller than the length of the string. Both points move in the x-direction, but the bottom point mass has 15% more velocity. This is done to demonstrate the bola's ability to be taut while rotating.

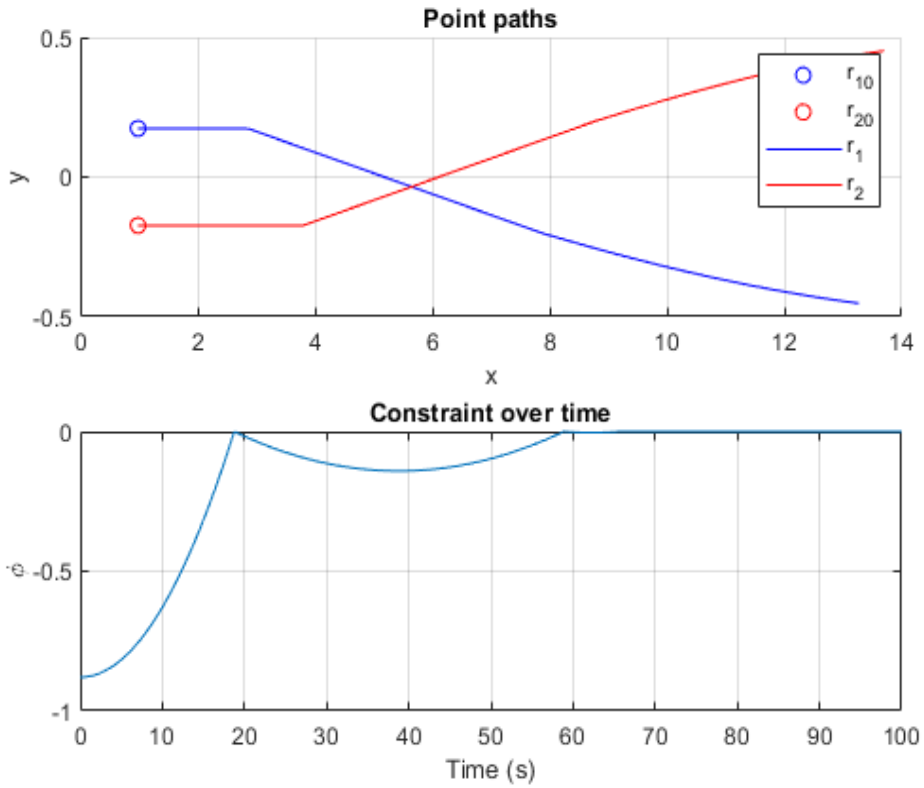


Figure 3.1: Analytical Solution (See Appendix I)

Figure 3.2 show the correct dynamics of a bola with a coefficient of restitution of 0.15 or, in

other words, only 15% of the energy is conserved after the string becomes taut. These dynamics are made by calculating the time of impact and solving for the velocity instantaneously after the impact. There are no dynamics simulating the taut string, it is assumed that the impact was instantaneous. This is then compared to Fig. 3.2.

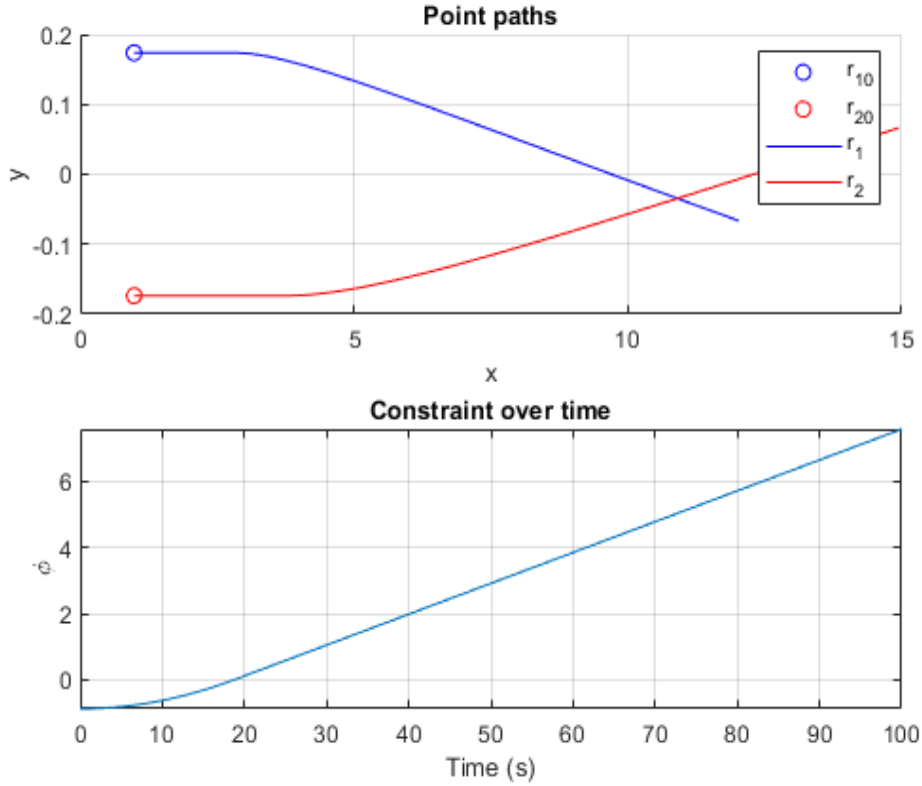


Figure 3.2: Proposed Model without Correction

In Fig. 3.2, it can be seen that while the constraint function is active, it is not enough to force the constraint back down. This failure to hold the constraint is due to the fact that the solution is discritized in the solver. Since the constraint function goes from $\phi < 0$ to $\phi > 0$, passing the constraint in a single step, the constraint is violated during numerical simulation. Nothing in the derivation accounts for this. To solve this overshoot problem, another term will need to be added into the solution to Λ. This solution is best shown in Junkins (2001) [19]. From equation (2.64) a

stabilizing control term ν is added such that:

$$\Lambda = -(\Theta_R^T M^{-1} \Theta_R)^{-1}(\Theta_R^T M^{-1} \boldsymbol{F}(\boldsymbol{R}, \boldsymbol{V}, t) + \dot{\Theta}_R^T \boldsymbol{V} + \boldsymbol{B} - \boldsymbol{\nu}) \tag{3.1}$$

This is Baumgarte's (1983) [20] method for constraint stabilization, where:

$$\boldsymbol{\nu} = -2\xi\omega_n \dot{\Theta} - \omega_n^2 \Theta \tag{3.2}$$

This additional term acts to stabilize the constraint violation. It introduces dynamics for when the constraint is violated. These dynamics take the form of a spring damped system, adjusted by two vectors ξ and ω_n. Because this constraint is modeling a string, a spring damped system is very apt. These two vectors describe the damping ratio, ξ, and the natural frequency, ω_n of each string. When introducing a body, these vectors will be lengthened to include coefficients to model an elastic body. Applying this method to the simple bola dynamics gives the following results:

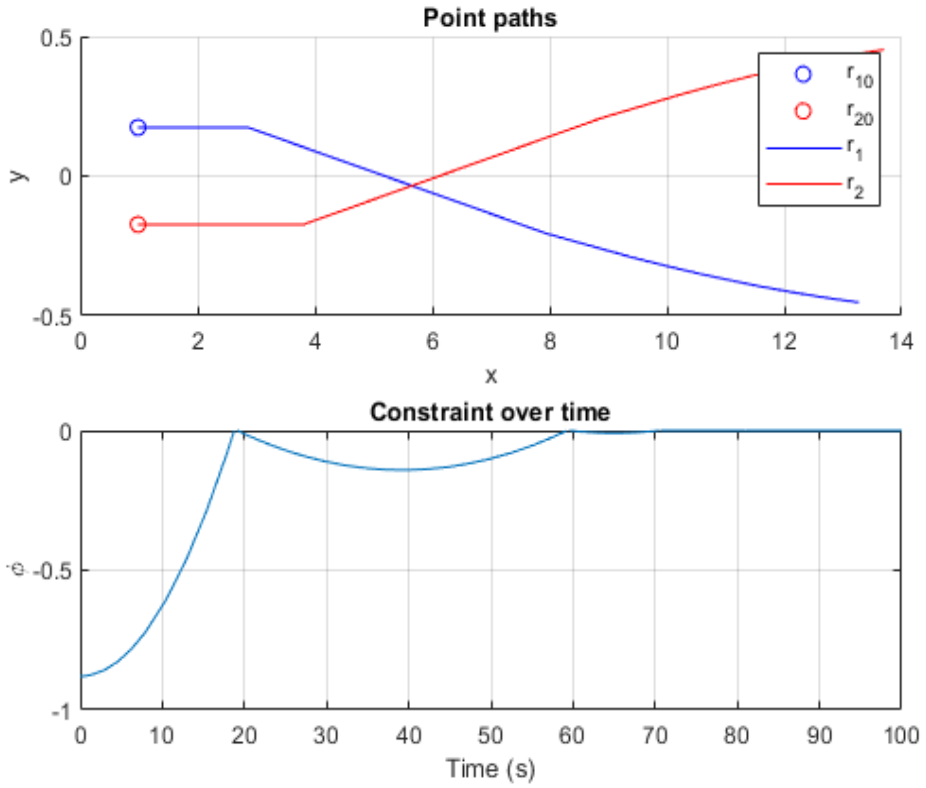


Figure 3.3: Proposed Model with Correction

Here the dynamics closely model the true solution, never deviating more than $\%1$. This was done by carefully choosing $\xi = 0.5$ and $\omega_n = 10$ to match $e = 0.15$. While there are more accurate values for both in order to more closely match the model, this is only to demonstrate the veracity of the model.

To better understand the constraint stabilization dynamics, below is a visual demonstration of Baumgarte's method seen in the constraint at the first collision.

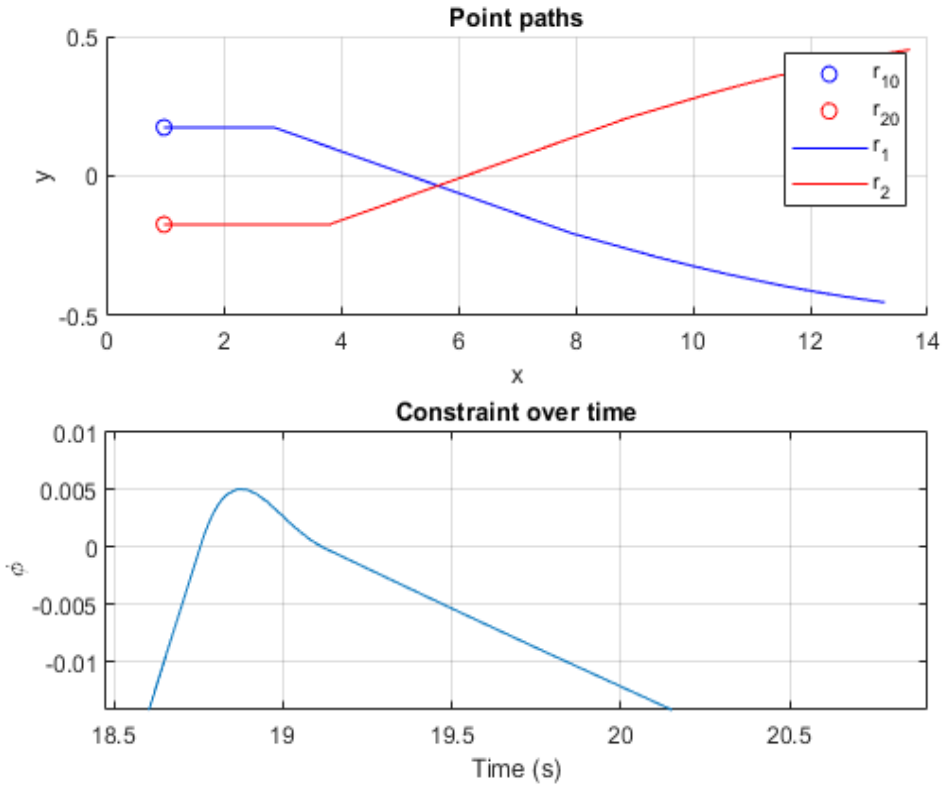


Figure 3.4: Constraint Stabilization Dynamics

Based on the values chosen for the stabilization, a smaller magnitude constraint violation can be attained, but the vales ξ and ω_n also determine the energy lost in the string becoming taut, making them very important to carefully consider based on the strings one would want to model. Comparing this small dynamics to the impact analytical solution shows:

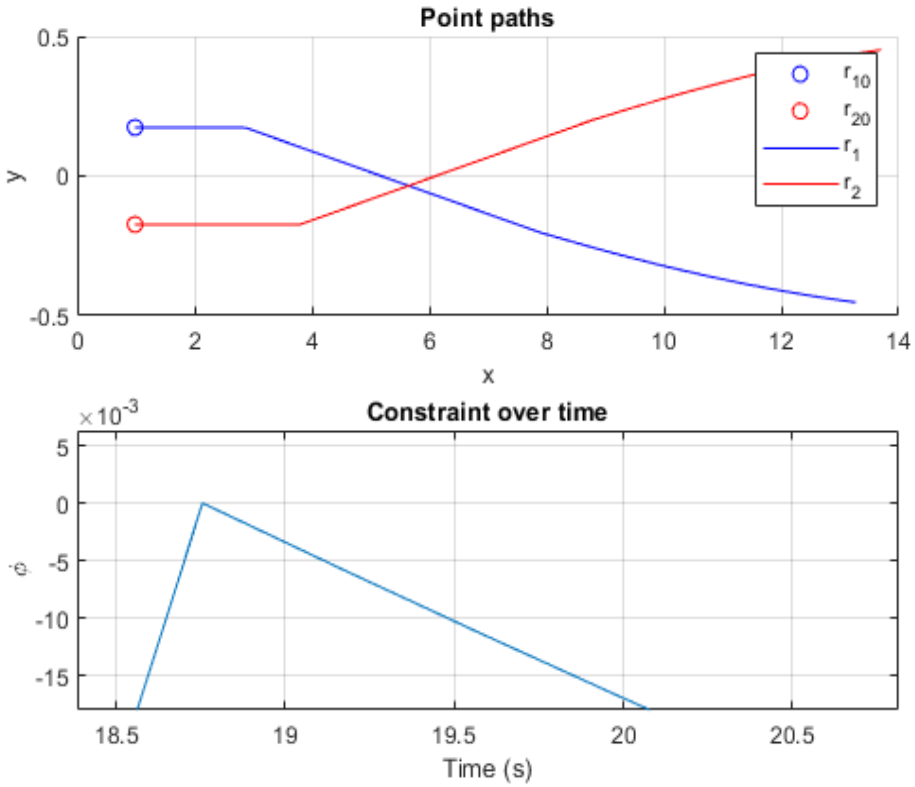


Figure 3.5: Analytical Solution at Constraint

This demonstrates the difference in the two models. Theoretically, the constraint should never be violated, however in attempting to mathematically model the constraint holding dynamics, a more physical system is revealed. One where the string can stretch and have taut dynamics. In the analytical solution, the constraint is never violated; it meets the constraint exactly and comes off it immediately.

This same method is also applied to the body constraint, with ξ and ω_n chosen differently to more accurately model elastic bodies. In the following section a better demonstration of this constraint holding is shown.

3.4 Convex Polytope in Simulation

An important aspect of this research is an approximation of the rigid bodies by using convex polytopes. Thus, generation and usage of them are addressed in the following section.

The generation method needs to take some information about the body. While there are numerous ways to generate a set of planes from a defined body, this research uses an approach for visualization purposes. This method is not unique, however, and there may be more efficient ways to generate polytopic approximations. It is important to note that once these planes are generated for a body, they need only be saved and fed into the dynamics of the problem. Therefore, the efficiency of this step is only worth considering if the body has a large number of planes required to represent it.

Once a set of planes is represented, an automated means to determine which plane a point may collide with is necessary. Since each plane is theoretically infinite, the point should only collide where a plane exists on the body. This section gives one solution to this problem. It is important to consider the computational efficiency in this process since it will need to be done at each instant Ψ is being calculated.

3.4.1 Polytope Generation

No matter what shape one would need to simulate, an approach to generate the A_p vectors from equation (2.44) is needed. While there are many ways to do this, one consideration is made to determine the method. In order to visualize the dynamics, a video is created in Matlab. To represent the body's planes the function "fill3" is used. This function takes a set of points and fills in a plane section. This allows only the desired section of each plane to be shown, making a fully connected set of planes making up the body. Therefore it is useful to use vertices of a convex polytope to define the A_p vectors. The "fill3" function *fills* inside a set of points on the corners of each plane. In this way, vertices are defined first and will be used in both the generation of the A_p vectors. Since each set of vertices lay on its associated plane, a simple system of linear equations can be used to solve for the A_p vector:

$$a_p x_i + b_p y_i + c_p z_i = 1 \tag{3.3}$$

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} A_p = \mathbf{1} \tag{3.4}$$

From any three vertices an A_p vector can be found.

To show this method of vector generation, consider an the example where it is desired to approximate cylinder with a convex polytope. An N sided regular polygon is used to represent the circular cross section, making it an N sided prism. For $N = 8$, an octagonal prism, can be generated.

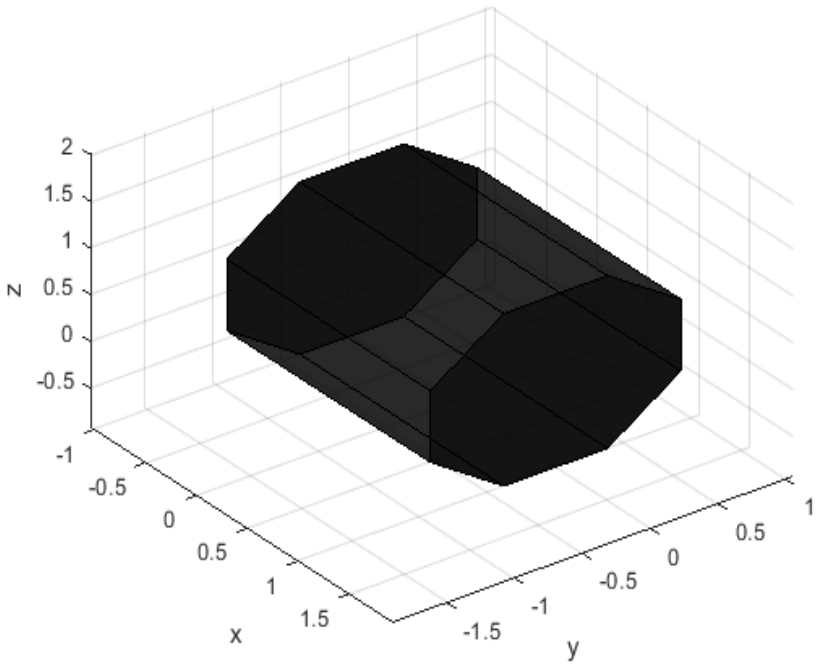


Figure 3.6: Octagonal Prism

Fig. 3.6 is generated using the intersection of a octagon with a circle of radius $r = 1$ and finding the vertices. These define the coordinates of the vertices in the x-y plane. To get the vertices in the z direction, add and subtract half the length of the cylinder, $L = 2$, to get each side.

Since these vertices are also used to create a visualization, when the body translates or rotates, they will need to be adjusted accordingly. To address translation, simply add P to each vertex vector. For rotation, multiply each vertex by the rotation transformation matrix; similar to equation (2.53). Any convex shape can be modeled in this way. For other shapes, vertices will need to be defined or solved for first, then each A_p will be solved for and fed into the solver. Further research may include a automatic generation of these vertices from an arbitrary body input.

3.4.2 Finding Appropriate Plane

After the planes themselves are defined, a consideration is needed in the differential equations that has, up to this point, not been addressed. A solution is needed to determine which b_p is associated with each point. This needs to be determined at each instant of time, since it can change, even after a point has collided with the body. The solution presented in this research addresses the issue without needing to calculate plane intersections or any new calculations. Since each ρ_i has a $r_p(\rho_i)$ for each plane, the smallest positive $r_p(\rho_i)$ is the plane corresponding to ρ_i. This method only works for convex polytopes, the body can have no concave sections, else this logic does not give the correct plane. The logic here has a computation cost for each plane and for each point at every instant of time. Below is a visual representation of this logic.

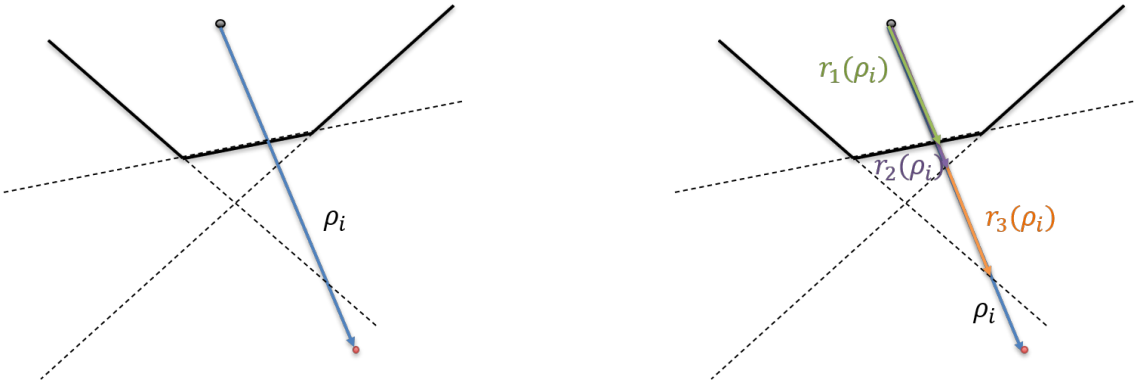


Figure 3.7: Convex Plane Finding

In the Fig. 3.7, the dark black lines represent the planes the defined the body. The dotted black lines are the parts of the planes that are not part of the defined body. Since each plane is infinite, only one plane can to be active at a time for each point. It can be seen that $r_1(\rho_i)$ is the corresponding plane to ρ_i because it is the smallest positive $r(\rho_i)$ between the three planes.

3.5 Summary

Mathematical dynamic models are somewhat meaningless without a form of implementation. As such, this chapter explored a way to implement the dynamics models formulated in Chapter 2. The implementation takes the form of a differential equation solver in Matlab. The ODE solver used was chosen for its robustness and utility. This code addresses complications not seen in the mathematical models. After resolving these issues, successful and accurate simulations can then be used in the following chapter.

4. Control

4.1 Motivation

One aspect that separates this method to some other slack string methods [21, 22], is the ability to introduce control laws into the model. This control is introduced in the term $\boldsymbol{F}(\boldsymbol{R}, \boldsymbol{V}, t)$. By including thrust vectors to some or all of the points, the system as a whole can be controlled. The following sections are two methods to control this system with demonstrations of the utility of the control law. The two methods are a simple optimal control law for the free points and an error stabilization for the system.

4.2 Optimal Control

The most ideal control law for any system would be one that is mathematically proven to be optimal. This method uses criteria for optimization and solves for the optimal control. These criteria are by no mean unique, but this section will demonstrate some simple solutions and some greater issues with this approach.

First, a cost function will need to be defined in order to determine what is optimal. Since one of the best uses for this method is a net capturing a piece of space debris, a cost function that penalized the total thrust over time is best used. In orbit, thrust is expensive due to the mass of the fuel, which translates directly into monetary expense. Therefore, the following cost function will be used for this problem:

$$J = \frac{1}{2} \int_{t_0}^{t_f} \boldsymbol{U}^T \boldsymbol{U} dt \tag{4.1}$$

where $\boldsymbol{U}$ is a stacked set of thrust vectors applied to the set of points. Ideally, four separate control laws can be written: one for no active constraints, one for at least one active string constraint, and one for at least one body constraint is active, and one where at least one of each string on body constraints are active. The following subsection deals with the case where neither the string nor

the body constraints are active.

4.2.1 No Constraints Optimal Control

$$\dot{\boldsymbol{R}} = \boldsymbol{V} \tag{4.2}$$

$$M\dot{\boldsymbol{V}} = \boldsymbol{F}(\boldsymbol{R}, \boldsymbol{V}, t) = \boldsymbol{W} + \boldsymbol{U} \tag{4.3}$$

$$\implies \dot{\boldsymbol{X}} = \begin{bmatrix} \dot{\boldsymbol{R}} \\ \dot{\boldsymbol{V}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{V} \\ M^{-1}(\boldsymbol{W} + \boldsymbol{U}) \end{bmatrix} \tag{4.4}$$

Here, $\boldsymbol{W}$ is the sum of all external forces on the set of points. Using another Lagrange multiplier, a Hamiltonian can be defined:

$$H = \frac{1}{2}\boldsymbol{U}^T\boldsymbol{U} + \nu^T\dot{\boldsymbol{X}} \tag{4.5}$$

$$\nu = \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix} \tag{4.6}$$

The ν vector acts to enforce the dynamics when minimizing the cost function. Finding the set of conditions will give a solution for $\boldsymbol{U}$:

$$\dot{\boldsymbol{X}} = \frac{\partial H}{\partial \nu} = \begin{bmatrix} \boldsymbol{V} \\ M^{-1}(\boldsymbol{W} + \boldsymbol{U}) \end{bmatrix} \tag{4.7}$$

$$\dot{\nu} = \begin{bmatrix} \dot{\nu}_1 \\ \dot{\nu}_2 \end{bmatrix} = -\frac{\partial H}{\partial \boldsymbol{X}} = -\begin{bmatrix} (\frac{\partial \boldsymbol{W}}{\partial \boldsymbol{R}})^T M^{-1}\nu_2 \\ \nu_1 + (\frac{\partial \boldsymbol{W}}{\partial \boldsymbol{V}})^T M^{-1}\nu_2 \end{bmatrix} \tag{4.8}$$

$$\boldsymbol{0} = \frac{\partial H}{\partial \boldsymbol{U}} = M^{-1}\boldsymbol{U} + \nu_2 \tag{4.9}$$

If this system is supposed to model a net capturing a body in orbit or in deep space, it can be assumed that $\boldsymbol{W}$ is simply zero. Alternatively, if air resistance is ignored while using a constant gravity, the result is the same. Taking either of these assumptions, the solution for $\boldsymbol{U}$ is greatly simplified:

$$\dot{\nu} = \begin{bmatrix} \dot{\nu}_1 \\ \\ \dot{\nu}_2 \end{bmatrix} = -\frac{\partial H}{\partial \boldsymbol{X}} = -\begin{bmatrix} \mathbf{0} \\ \\ \nu_1 \end{bmatrix} \tag{4.10}$$

$$\mathbf{0} = \frac{\partial H}{\partial \boldsymbol{U}} = M^{-1}\boldsymbol{U} + \nu_2 \tag{4.11}$$

$$\implies \boldsymbol{U} = -M(-\nu_1 t + \nu_{2_0}) \tag{4.12}$$

The initial conditions and final conditions are as follows:

$$\boldsymbol{R}(0) = \boldsymbol{R}_0 \tag{4.13}$$

$$\boldsymbol{V}(0) = \boldsymbol{V}_0 \tag{4.14}$$

$$\boldsymbol{R}(t_f) = \boldsymbol{R}_f \tag{4.15}$$

$$\boldsymbol{V}(t_f) = \boldsymbol{V}_f \tag{4.16}$$

Substituting this solution into equations (2.4) and using the initial and final conditions, as well as the set final time gives:

$$M\dot{\boldsymbol{V}} = -M(-\nu_1 t + \nu_{2_0}) \tag{4.17}$$

$$\boldsymbol{V} = -\nu_1 \frac{t^2}{2} + \nu_{2_0} t + \boldsymbol{V}_0 \tag{4.18}$$

$$\boldsymbol{R} = -\nu_1 \frac{t^3}{6} + \nu_{2_0} \frac{t^2}{2} + \boldsymbol{V}_0 t + \boldsymbol{R}_0 \tag{4.19}$$

Solving for ν_1 and ν_{2_0}:

$$\boldsymbol{V}(t_f) = \boldsymbol{V}_f = -\nu_1 \frac{t_f^2}{2} + \nu_{2_0} t_f + \boldsymbol{V}_0 \tag{4.20}$$

$$\boldsymbol{R}(t_f) = \boldsymbol{R}_f = -\nu_1 \frac{t_f^3}{6} + \nu_{2_0} \frac{t_f^2}{2} + \boldsymbol{V}_0 t_f + \boldsymbol{R}_0 \tag{4.21}$$

$$\begin{bmatrix} -\frac{t_f^2}{2} I_{N_p} & t_f I_{N_p} \\ -\frac{t_f^3}{6} I_{N_p} & \frac{t_f^2}{2} I_{N_p} \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_{2_0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{V}_f - \boldsymbol{V}_0 \\ \boldsymbol{R}_f - \boldsymbol{V}_0 t_f - \boldsymbol{R}_0 \end{bmatrix} \tag{4.22}$$

$$\implies \begin{bmatrix} \nu_1 \\ \nu_{2_0} \end{bmatrix} = \begin{bmatrix} -\frac{6}{t_f^2} I_{N_p} & \frac{12}{t_f^3} I_{N_p} \\ -\frac{2}{t_f} I_{N_p} & \frac{6}{t_f^2} I_{N_p} \end{bmatrix} \begin{bmatrix} \boldsymbol{V}_f - \boldsymbol{V}_0 \\ \boldsymbol{R}_f - \boldsymbol{V}_0 t_f - \boldsymbol{R}_0 \end{bmatrix} \tag{4.23}$$

This gives a closed form solution to the optimal control problem. Choosing values that drive the net to be fully expanded state just before it collides with the body gives the following figures:

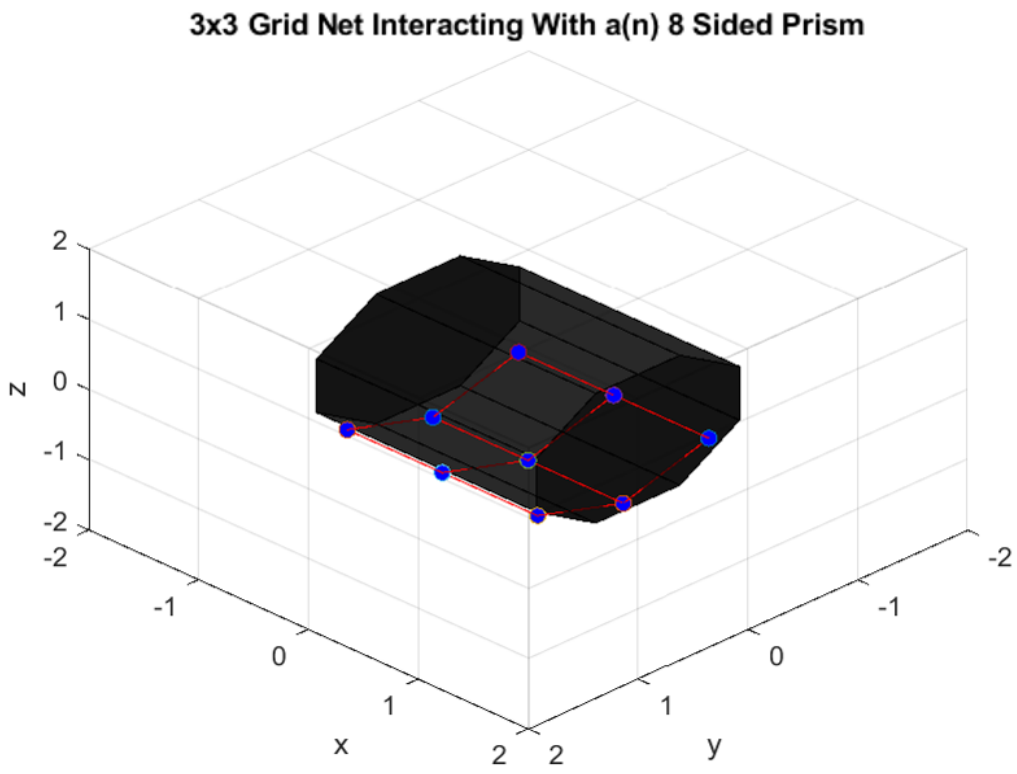


Figure 4.1: Final State of Optimal Control Solution

Here the final state is exactly at the surface of the body and the strings are fully extended. This is meant to show the ability to be controlled when the strings are not active. In the next subsection it will be shown the difficulty in writing a control law for when either the string or body constraints are active.

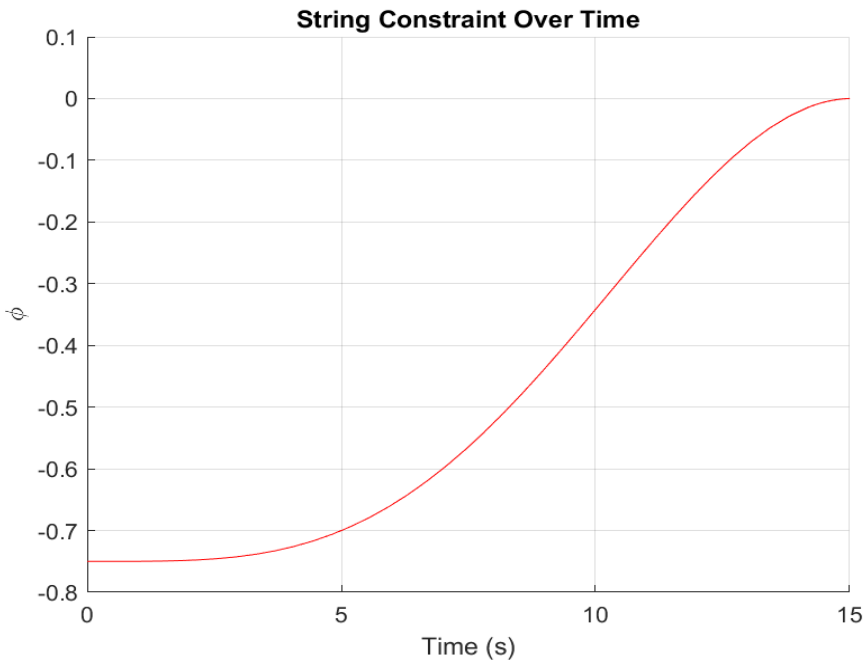


Figure 4.2: Final State of Optimal Control Solution

It can be seen that all string constraints follow the exact same path. This is due to the linear symmetric nature of the control law. The string constraints also all end at exactly zero, meaning that the control law sends all string to be taut by the end of the dynamics.

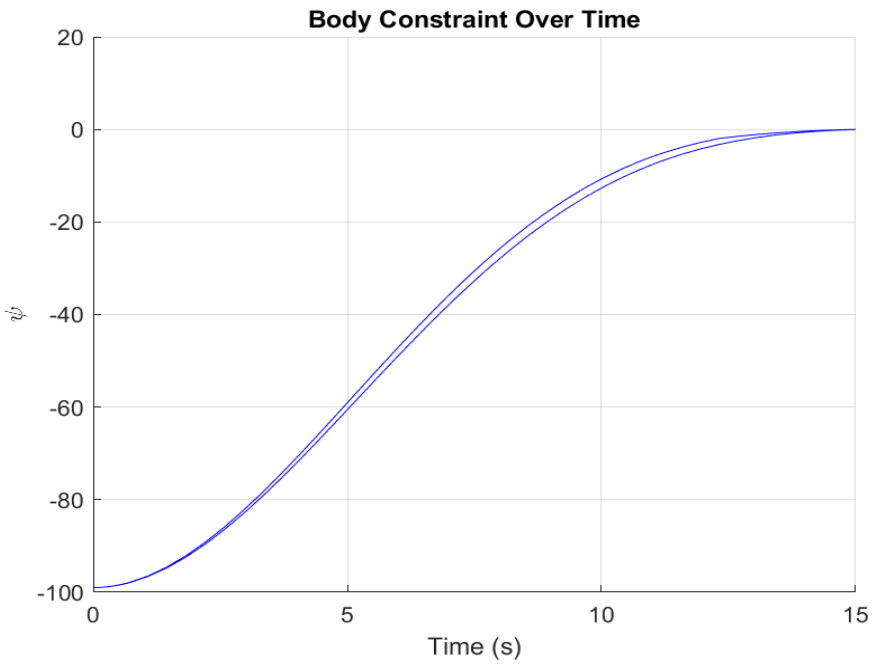


Figure 4.3: Final State of Optimal Control Solution

In the body constraint, the path splits and rejoins itself by the end of the controlled time. This is due to the fact that three points are headed directly to the center plane and the others are headed to other planes. Again, the constraint ends at exactly zero because this was the desired condition. Between these figures, it can be demonstrated that the optimal control law can deal with the system all the way up to the constraints. Theoretically, a separate control law would take over to control the active system. The following subsection will show the difficulty in implementing that.

4.2.2 Active Constraints Optimal Control Law

Starting with the same cost function (4.1), initial conditions (4.13-4.16), and assumptions, a new control law can be developed. Starting from equations (2.6) and (2.21) to address the concept:

$$M\dot{\boldsymbol{V}} = \boldsymbol{F}(\boldsymbol{R}, \boldsymbol{V}, t) + \Phi_R \lambda \tag{4.24}$$

$$= \boldsymbol{U} - \Phi_R(\Phi_R^T M^{-1} \Phi_R)^{-1}(\Phi_R^T M^{-1}\boldsymbol{U} + \dot{\Phi}_R^T \boldsymbol{V}) \tag{4.25}$$

$$= -\Phi_R(\Phi_R^T M^{-1} \Phi_R)^{-1}\dot{\Phi}_R^T \boldsymbol{V} + [I_{N_p} - \Phi_R(\Phi_R^T M^{-1} \Phi_R)^{-1}\Phi_R^T M^{-1}]\boldsymbol{U} \tag{4.26}$$

When moving to finding the costates, ν_1 and ν_2, a difficulty arises:

$$\dot{\nu} = \begin{bmatrix} \dot{\nu}_1 \\ \dot{\nu}_2 \end{bmatrix} = -\frac{\partial H}{\partial \boldsymbol{X}} \tag{4.27}$$

$$\dot{\nu}_1 = \frac{\partial}{\partial \boldsymbol{R}} \left[\nu_2^T \left(\Phi_R(\Phi_R^T M^{-1} \Phi_R)^{-1}\dot{\Phi}_R^T \boldsymbol{V} + [I_{N_p} - \Phi_R(\Phi_R^T M^{-1} \Phi_R)^{-1}\Phi_R^T M^{-1}]\boldsymbol{U} \right) \right] \tag{4.28}$$

$$\dot{\nu}_2 = \nu_1 + \frac{\partial}{\partial \boldsymbol{V}} \left[\nu_2^T \left(\Phi_R(\Phi_R^T M^{-1} \Phi_R)^{-1}\dot{\Phi}_R^T \boldsymbol{V} \right) \right] \tag{4.29}$$

By just including one active string constraint, finding the costates becomes unpractical. Therefore a more reliable and simple solution is more desirable. The next section shows a much more manageable method of controlling the system when the constrains are active.

4.3 Error Stabilization

A common method for dealing with particularly difficult nonlinear differential equations is an error stabilization procedure. The idea is to define an error that can be sent to zero and define convenient dynamics to send it there.

$$\boldsymbol{E} = \boldsymbol{R} - \boldsymbol{R}_f \tag{4.30}$$

$$\dot{\boldsymbol{E}} = \boldsymbol{V} - \boldsymbol{V}_f \tag{4.31}$$

$$\ddot{\boldsymbol{E}} = \dot{\boldsymbol{V}} \tag{4.32}$$

The desired dynamics can be defined and solved for $\boldsymbol{U}$:

$$\ddot{\boldsymbol{E}} = -2\zeta\omega_n\dot{\boldsymbol{E}} - \omega_n^2\boldsymbol{E} \tag{4.33}$$

$$\dot{\boldsymbol{V}} = M^{-1}(\boldsymbol{W} + \boldsymbol{U}) + M^{-1}\Theta_R\Lambda = -2\zeta\omega_n\dot{\boldsymbol{E}} - \omega_n^2\boldsymbol{E} \tag{4.34}$$

Substituting Λ from the most general case to get a function of $\boldsymbol{U}$:

$$\dot{\boldsymbol{V}} = M^{-1}(\boldsymbol{W} + \boldsymbol{U}) - M^{-1}\Theta_R G_R^{-1}(\Theta_R^T M^{-1}(\boldsymbol{W} + \boldsymbol{U}) + \boldsymbol{B}_V) \tag{4.35}$$

$$G_R = \Theta_R^T M^{-1}\Theta_R + F_R + L_R \tag{4.36}$$

$$\boldsymbol{B}_V = \dot{\Theta}_R^T\boldsymbol{V} + \boldsymbol{B} \tag{4.37}$$

$$= \boldsymbol{G}\boldsymbol{U} + \boldsymbol{G}\boldsymbol{W} + \boldsymbol{B}_E \tag{4.38}$$

$$\boldsymbol{B}_E = -M^{-1}\Theta_R G_R^{-1}\boldsymbol{B}_V \tag{4.39}$$

$$\boldsymbol{G} = M^{-1} - M^{-1}\Theta_R G_R^{-1}\Theta_R^T M^{-1} \tag{4.40}$$

Solving for $\boldsymbol{U}$:

$$\boldsymbol{G}\boldsymbol{U} + \boldsymbol{G}\boldsymbol{W} + \boldsymbol{B}_E = -2\zeta\omega_n\dot{\boldsymbol{E}} - \omega_n^2\boldsymbol{E} \tag{4.41}$$

$$\boldsymbol{U} = -G^{-1}(2\zeta\omega_n\dot{\boldsymbol{E}} + \omega_n^2\boldsymbol{E} + \boldsymbol{G}\boldsymbol{W} + \boldsymbol{B}_E) \tag{4.42}$$

Using the same conditions as the previous section, this control law can be used to envelope an octagonal prism:

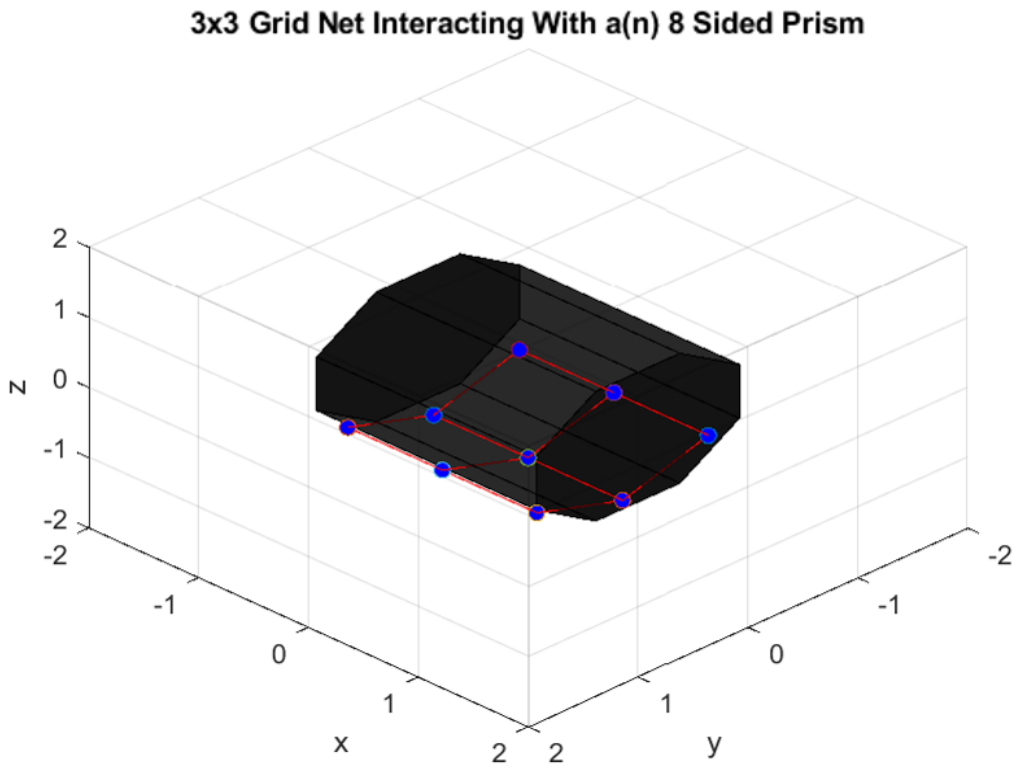


Figure 4.4: Final State of Error Stabilization Solution

Above is the final state of the implemented control law. The net stayed with the body for around of third of the total simulation time. It also moves with the body for a short distance, while the body constraint is active. This control law sends the net to the body significantly faster than the optimal control law. However this was due to the values of $\zeta = 1$ and $\omega_n = 1$. Choosing these values differently will cause the control law to behave in a variety of desirable ways. Below is the string constraint:

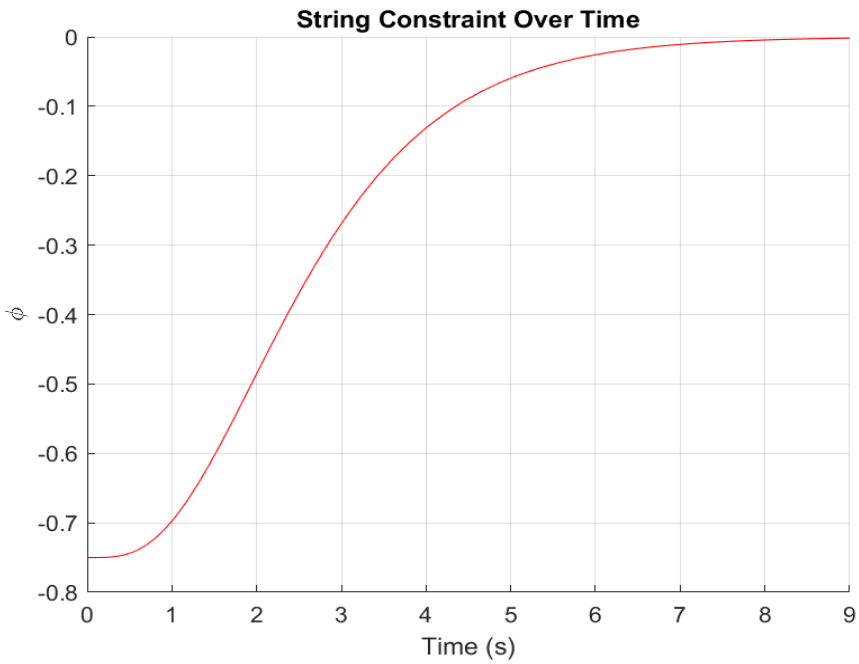


Figure 4.5: Final State of Error Stabilization Solution

Since the final positions for the points is a fully extended net encompassing the body, the simulation ends when all constraints reach zero. Based on the values chosen for the error stabilization, it happens that the string constraints are the last to reach zero. In the following figure it can be seen that the body constraint reach ed zero much earlier:

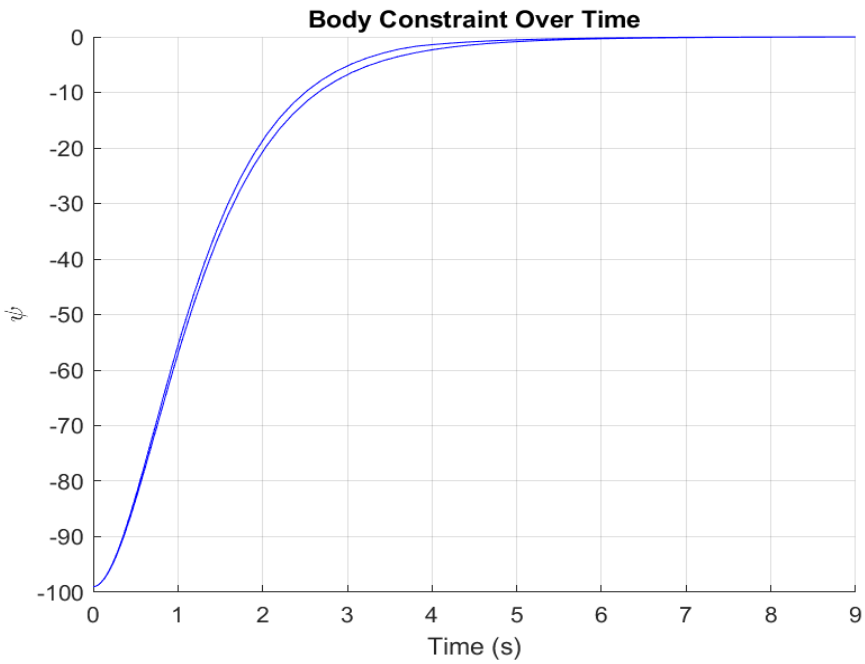


Figure 4.6: Final State of Error Stabilization Solution

Just like the optimal control solution, the body constraints split into two because the center three points are closer to their final positions on the body from the start than the other six. Once the net fully reaches the body, it travels along the edge of the body to get to its desire location. Given different chosen values of ζ and ω_n this control law would do the same with the string constraint; spending time fully extended before hitting the body.

4.4 Summary

In order to make full use of the mathematical models and the computational implementation, a set of control laws were written and implemented in this chapter. First an optimal control law that was derived with reduced complexity. Then a more impractical, but more comprehensive, optimal control law was described, but due to its unwieldiness, it was not simulated in the code. Instead, a more versatile nonlinear control law was shown, known as error stabilization. Although being less optimal, it incorporated the inherently nonlinear dynamics. While these methods are by no means unique, they demonstrate the utility of the method and the computational implementation.

5. Conclusion

In an effort to address the increasing need to reduce hazardous objects in space, this thesis provides one multibody dynamic model, along with a system for simulation and control.

5.1 Results

This thesis explores a method of multibody dynamics that is used to simulate a network of points (net) colliding with a rigid body. The method was derived with the use of Lagrange multipliers to hold constraints. It was then simulated using a Matlab code that used a method created by Baumgarte in 1983 [20]. An optimal control law was shown for the system, along with its limitations. Error stabilization was also used as a more applicable control law for such a nonlinear system. Altogether, it was shown that this method, along with its implementation, was a viable and efficient method of simulated the capture of a tumbling rigid body by a net.

5.2 Future Work

The most promising use for future work with this model would be its use in other multibody dynamics such as tensegrity. Tensegrity is a structural design method that uses only axial loaded members. For the tensile members (strings) this method lends itself to being incorporated into the analysis of tensegrity.

Additionally, this work would be greatly improved by some real world tests of its physical simulation. These experiments should serve to both test the methods' veracity as well as its computational efficiency when used as part of a control law. Further development on these methods could include a more computationally efficient code to be used in real time control laws.

REFERENCES

[1] A. Fanfani, S. Jayousi, S. Morosi, L. S. Ronga, E. Del Re, and L. Rossettini, "Feasibility study of an alert messaging system by means of cubesat, sdr and web service technologies," pp. 1–7, 2017.

[2] D. J. Kessler and B. G. Cour-Palais, "Collision frequency of artificial satellites: The creation of a debris belt," *Journal of Geophysical Research: Space Physics*, vol. 83, no. A6, pp. 2637–2646, 1978.

[3] N. R. Council, *Continuing Kepler's Quest: Assessing Air Force Space Command's Astrodynamics Standards*. Washington, DC: The National Academies Press, 2012.

[4] J. Missel and D. Mortari, "Path optimization for space sweeper with sling-sat: A method of active space debris removal," *Advances in Space Research*, vol. 52, pp. 1339–1348, 10 2013.

[5] V. Braun, A. Lüpken, S. Flegel, J. Gelhaus, M. Moeckel, C. Kebschull, C. Wiedemann, and P. Vörsmann, "Active debris removal of multiple priority targets," *Advances in Space Research*, vol. 51, p. 1638–1648, 05 2013.

[6] C. Peck, D. Adams, J. McElreath, A. Verras, J. Hiemerl, M. Majji, M. Benedict, and J. Junkins, "Autonomous deployment of payload packages to spinning rocket bodies: Approach, apparatus, and emulation using ground robotics," 03 2020.

[7] S. Edward M., "Space environmental effects on spacecraft: Leo materials selection guide," tech. rep., 1995.

[8] U. Battista, A. Landini, W. Gołębiowski, R. Michalczyk, A. Czerwiński, K. Duda, and A. Sochaczewska, "Design of net ejector for space debris capturing," 04 2017.

[9] R. Motro, "Tensegrity systems: The state of the art," *International Journal of Space Structures*, vol. 7, no. 2, pp. 75–83, 1992.

[10] A. K. Banerjee, "Contributions of multibody dynamics to space flight: A brief review," *Journal of Guidance, Control, and Dynamics*, vol. 26, no. 3, pp. 385–394, 2003.

[11] A. A. Shabana, Y. L. Hwang, and R. A. Wehage, "Projection methods in flexible multibody dynamics part i: Kinematics," *International Journal for Numerical Methods in Engineering*, vol. 25, no. 10, pp. 1927–1939, 1992.

[12] R. A. Wehage, A. A. Shabana, and Y. L. Hwang, "Projection methods in flexible multibody dynamics part ii: Dynamics and recursive projection methods," *International Journal for Numerical Methods in Engineering*, vol. 25, pp. 1941–1966, December 1992.

[13] K. Changizi and A. A. Shabana, "A recursive formulation for the dynamic analysis of open loop deformable multibody systems," *Journal of Applied Mechanics*, vol. 55, no. 3, pp. 687–693, 1988.

[14] A. Jain and G. Rodriguez, "Recursive flexible multibody system dynamics using spatial operators," *Journal of Guidance Control and Dynamics*, vol. 15, no. 6, pp. 1453–1466, 1992.

[15] K. Anderson, "An order-n formulation for motion simulation of general constrained multi-rigid-body systems," *Computers and Structures*, vol. 43, no. 3, pp. 565–572, 1992.

[16] K. S. Anderson, "An order-n formulation for the motion simulation of general multi-rigid body tree systems," *Computers and Structures*, vol. 46, no. 3, pp. 547– 559, 1993.

[17] R. Featherstone, "A divide and conquer articulated body algorithm for parallel o(log(n)) calculation of rigid body dynamics," *International Journal of Robotics Research*, vol. 18, no. 9, pp. 867–875, 1999.

[18] S. Pradhan, V. J. Modi, and A. K. Misra, "Order-n formulation for flexible multibody systems tree topology: Lagrangian approach," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 4, pp. 665–672, 1997.

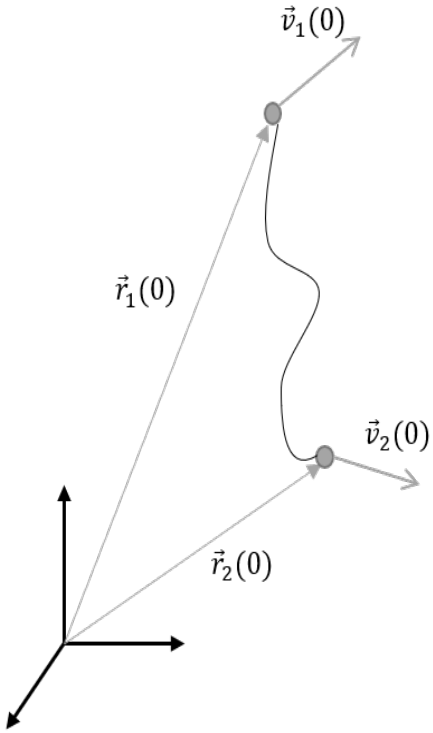
[19] J. L. Junkins, M. R. Akella, and A. J. Kurdila, "Adaptive realization of desired constraint stabilization dynamics in the control of multibody systems," *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, vol. 359, no. 1788, pp. 2231–2249, 2001.

[20] J. W. Baumgarte, “A New Method of Stabilization for Holonomic Constraints,” *Journal of Applied Mechanics*, vol. 50, pp. 869–870, 12 1983.

[21] Y. Zhao, F. Zhang, and P. Huang, “Capture dynamics and control of tethered space net robot for space debris capturing in unideal capture case,” *Journal of the Franklin Institute*, vol. 357, no. 17, pp. 12019–12036, 2020.

[22] J. Si, Z. Pang, Z. Du, and C. Cheng, “Dynamics modeling and simulation of self-collision of tether-net for space debris removal,” *Advances in Space Research*, vol. 64, no. 9, pp. 1675–1687, 2019.

6. APPENDIX I: ANALYTICAL BOLA DERIVATION



Assumptions:

1. Point masses
2. Inertia of connecting string can be ignored, i.e. massless string
3. Impact collisions
4. Constant gravity for simplicity

The point masses will continue in a analytically determined trajectory, given the initial conditions, until the string constraint is met:

$$\boldsymbol{r}_1(0) = \boldsymbol{r}_{1_0}; \boldsymbol{r}_2(0) = \boldsymbol{r}_{2_0}; \boldsymbol{v}_1(0) = \boldsymbol{v}_{1_0}; \boldsymbol{v}_2(0) = \boldsymbol{v}_{2_0} \tag{6.1}$$

$$(\boldsymbol{r}_2(t_i) - \boldsymbol{r}_1(t_i))^T(\boldsymbol{r}_2(t_i) - \boldsymbol{r}_1(t_i)) = L^2 \tag{6.2}$$

Where L is the length of the string connecting the two masses.

The trajectories will be as follows:

$$\boldsymbol{v}_1(t) = \boldsymbol{g}t + \boldsymbol{v}_{1_0} \tag{6.3}$$

$$\boldsymbol{v}_2(t) = \boldsymbol{g}t + \boldsymbol{v}_{2_0} \tag{6.4}$$

$$\boldsymbol{r}_1(t) = \boldsymbol{g}\frac{t^2}{2} + \boldsymbol{v}_{1_0}t + \boldsymbol{r}_{1_0} \tag{6.5}$$

$$\boldsymbol{r}_2(t) = \boldsymbol{g}\frac{t^2}{2} + \boldsymbol{v}_{2_0}t + \boldsymbol{r}_{2_0} \tag{6.6}$$

From here the position, velocity and gravity vectors will be collected into single vectors for simplicity:

$$\boldsymbol{v}(t) = \boldsymbol{g}t + \boldsymbol{v}_0 \tag{6.7}$$

$$\boldsymbol{r}(t) = \boldsymbol{g}\frac{t^2}{2} + \boldsymbol{v}_0t + \boldsymbol{r}_0 \tag{6.8}$$

$$\boldsymbol{r} = \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \end{bmatrix} \boldsymbol{v} = \begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \end{bmatrix} \boldsymbol{g} = \begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{g} \end{bmatrix} \tag{6.9}$$

The position of both particles are known for all time before the first impact, where the string will become taught and cause the masses to *bounce* back at each other. Therefore, the first impact time, position, and velocity can be found.

$$C = \begin{bmatrix} I_3 & -I_3 \end{bmatrix} \tag{6.10}$$

$$(C\boldsymbol{r}(t_i))^T(C\boldsymbol{r}(t_i)) - L^2 = 0 \tag{6.11}$$

For a constant gravity term, this gives a quadratic equation in t_i. In a no gravity or large velocity case, it is a simple quadratic.

This equation has a solution if the constraint is not already met or violated and as long as the initial velocities aren't the same vector.

Infinitesimally before the impact, a velocity can be found.

$$\boldsymbol{v}_1(t_i^-) = \boldsymbol{g}t_i^+ + \boldsymbol{v}_{1_0} \tag{6.12}$$

$$\boldsymbol{v}_2(t_i^-) = \boldsymbol{g}t_i^+ + \boldsymbol{v}_{2_0} \tag{6.13}$$

The impact will only effect the components of the velocities along the direction of the string:

$$\boldsymbol{l} = C\boldsymbol{r} = \boldsymbol{r}_1(t_i) - \boldsymbol{r}_2(t_i) \tag{6.14}$$

$$\boldsymbol{v}_{1_n}(t_i^+) = \frac{(\boldsymbol{v}_1^T\boldsymbol{l})\boldsymbol{l}}{||\boldsymbol{l}||^2} \tag{6.15}$$

$$\boldsymbol{v}_{2_n}(t_i^+) = \frac{(\boldsymbol{v}_2^T(-\boldsymbol{l}))(-\boldsymbol{l})}{||\boldsymbol{l}||^2} \tag{6.16}$$

$$A_1 = \begin{bmatrix} m_1 I_3 & m_2 I_3 \\ -I_3 & I_3 \end{bmatrix} \tag{6.17}$$

$$A_2 = \begin{bmatrix} m_1 I_3 & m_2 I_3 \\ eI_3 & -eI_3 \end{bmatrix} \tag{6.18}$$

$$\boldsymbol{v}_n(t_i^+) = A_1^{-1} A_2 \begin{bmatrix} \boldsymbol{v}_{1_n} \\ \boldsymbol{v}_{2_n} \end{bmatrix} \tag{6.19}$$

$$\boldsymbol{v}(t_i^+) = \boldsymbol{v}(t_i^-) - \boldsymbol{v}_n(t_i^-) + \boldsymbol{v}_n(t_i^+) \tag{6.20}$$

Where e is the coefficient of restitution and it determines the energy lost in the impact. When $e = 1$, the impact looses no energy and rebounds at equal and opposite normal velocities. In this case, the cycle repeats and new impact times are repeatedly found until a final time or position is set. In the case where $0 \leq e < 1$, energy is lost and the time in between impacts and decreases until the impact times are so small it can be considered that the two masses are constantly meeting

the constraint. When this happens, a new set of dynamic equations is needed:

$$\boldsymbol{l} = C\boldsymbol{r} \tag{6.21}$$

$$\boldsymbol{l}^T\boldsymbol{l} = L^2 \tag{6.22}$$

$$\frac{d}{dt}(\boldsymbol{l}^T\boldsymbol{l}) = 0 \tag{6.23}$$

$$\dot{\boldsymbol{l}}^T\boldsymbol{l} + \boldsymbol{l}^T\dot{\boldsymbol{l}} = 0 = 2\boldsymbol{l}^T\dot{\boldsymbol{l}} \tag{6.24}$$

$$\dot{\boldsymbol{l}}^T\dot{\boldsymbol{l}} + \boldsymbol{l}^T\ddot{\boldsymbol{l}} = 0 \tag{6.25}$$

$$-\dot{\boldsymbol{l}}^T\dot{\boldsymbol{l}} = \boldsymbol{l}^T\ddot{\boldsymbol{l}} \tag{6.26}$$

$$\ddot{\boldsymbol{l}} = -(\boldsymbol{l}^T)^+\dot{\boldsymbol{l}}^T\dot{\boldsymbol{l}} \tag{6.27}$$

$$\ddot{\boldsymbol{l}} = -\frac{1}{L^2}\boldsymbol{l}\dot{\boldsymbol{l}}^T\dot{\boldsymbol{l}} \tag{6.28}$$

$$\boldsymbol{R} = \frac{1}{m_1 + m_2}\begin{bmatrix} m_1 I_3 & m_2 I_3 \end{bmatrix}\boldsymbol{r} \tag{6.29}$$

$$\ddot{\boldsymbol{R}} = \boldsymbol{g} \tag{6.30}$$