A Strategy for Multiple Cooperative Spacecraft to Control the Translational Velocity of a Shared Payload

David H. Besson

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Aeronautics and Astronautics

University of Washington

2015

Reading Committee:

Mehran Mesbahi, Chair

Kristi Morgansen

Anshu Narang-Siddarth

Program Authorized to Offer Degree:
Department of Aeronautics and Astronautics

University of Washington

Abstract

A Strategy for Multiple Cooperative Spacecraft to
Control the Translational Velocity of a Shared Payload

David H. Besson

Chair of the Supervisory Committee:
Professor Mehran Mesbahi
Aeronautics and Astronautics

The scale of payloads that can be moved in space is effectively dictated by the propellant storage, maximum thrust, and thrust efficiency of the upper-stages on modern launch vehicles. These vehicles must control the payload at a single attachment point which leads to restrictive mass and symmetry constraints for the payload design. The research presented here examines an alternative strategy for performing velocity control on large payloads, such as future orbital construction projects and captured asteroids. Starting from a robotic manipulator perspective, an approach is developed to grasp, orient, and translate the payload with multiple cooperative spacecraft that are already in orbit. The approach makes use of a new combinatorial optimization algorithm, path planning with a three-dimensional visibility graph road-map method, and constrained model predictive control. All elements of the strategy are tested in multiple MATLAB simulations.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I am indebted to the Robotics, Aeronautics, and Information Networks Laboratory. The torrent of support in the RAIN lab made this work possible.

Chapter 1

INTRODUCTION

The general strategy for controlling the translational velocity of payloads in space has hardly changed since the Soviet Union launched Sputnik in 1957. The payload is placed on top of a large multi-stage rocket and the stages of the rocket detach as they run out of propellant so that the payload can reach specific altitude and velocity thresholds. Once the payload reaches its final orbit, it is very rare for it ever again to come in contact with another spacecraft. All of the payload's remaining propulsion maneuvers are accomplished with its own equipment. This strategy has been sufficient for the missions that the world's space organizations have undertaken over the last 60 years. But eventually a new strategy will be needed as the size of payloads, and the desired velocity changes for those payloads, surpass the capabilities of conventional multi-stage rocket systems.

The goal of this thesis is to examine the feasibility, from a decision and control standpoint, of using the propulsion systems onboard multiple cooperative spacecraft to control the translatinal velocity of a shared payload. In other words, we will investigate whether these multiple spacecraft can be wrapped in a common control strategy such that they collectively perform the role of a single upper-stage launch vehicle.

1.1 Previous Research

There are two historical examples for using a smaller spacecraft to perform a velcoity control maneuver on a larger payload. The first is the Apollo 13 mission to the moon. After an explosion in the Service Module rendered the Service Module Propulsion System too dangerous for use, the crew had to rely on the smaller Lunar Module Decent Propulsion System (DPS) to perform the required propulsion maneuvers. The DPS was never designed to push the Service Module but the crew and ground engineers were able to rebalance the spacecraft and modify the trajectory burn profiles in order to safely operate the DPS [18].

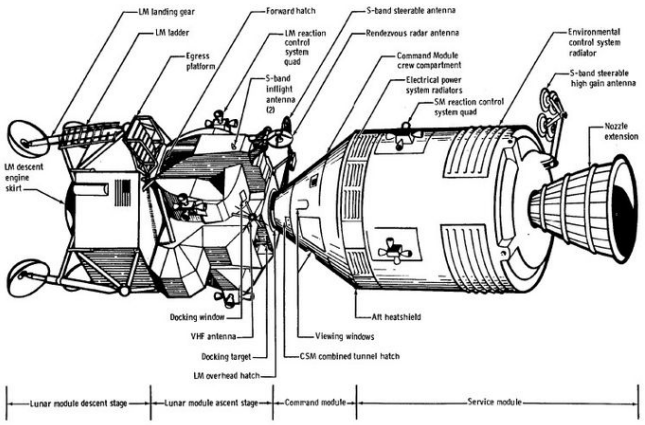

Figure 1.1: An illustration of the Apollo spacecraft [2]. During the Apollo 13 mission, the smaller Lunar Module on the left was used to propel the larger Service Module on the right.

This was a significant accomplishment because it showed that a propulsion system on a spacecraft can be quickly modified to propel a new payload.

The other example is the International Space Station (ISS). The ISS is able to maintain the altitude of its orbit by relying on the thrusters and propellant of the small spacecraft that visit it. The spacecraft, such as the Russian Progress or European Automated Transfer Vehicle, dock at a central attachment point on the ISS. The small spacecraft then uses its own propellant and thrusters to increase the velocity of the station [24]. This is an important achievement because it introduces the notion of interchangeable vehicles that are capable of performing the same velocity control maneuver.

However, both of these examples only utilize a single vehicle at a time. There is a growing body of work looking at the multi-vehicle cooperative payload control problem in two dimensions. For example, a strategy was developed to move a scale model of a United States Navy ship with a swarm of autonomous tugboats [10]. Other work has focused on controlling and coordinating a group of robots for cooperative manipulation tasks through the hierarchical composition of potential fields [31]. And other researchers have built genetic

Figure 1.2: A photograph of the Automated Transfer Vehicle (bottom of image) docked with the International Space Station [9]. Other vehicles such as the Russian Progress are capable of docking at the same location.

algorithms capable of choosing the optimal contact points for constraining a prismatic object by a group of mobile robots [29].

1.2 Research Contributions and Chapter Organization

The primary contribution of this research is to extend the payload velocity control problem to three dimensions with an arbitrary number of cooperative spacecraft. With multiple spacecraft it will be possible to control the velocity of payloads that would be impractical to both move and balance with a single propulsion source. The primary mathematical concerns in this problem are coping with the asymmetry of the spacecraft configuration and non-linearity in the velocity-propellant relationship which arises from the dramatic migration of the combined center of mass throughout the maneuver.

Chapter 2 discusses an algorithm for choosing spacecraft attachment points on the surface of a large payload. In Chapter 3, we develop a path planning algorithm that can be used to take a spacecraft from its initial position relative to the payload and transfer it to its final position at the desired attachment point. In Chapter 4, we propose a strategy

to coordinate the propulsion of the multiple spacecraft based on a model predictive control framework. And finally in Chapter 5, we discuss conclusions for this work and opportunities for future extensions and applications.

Chapter 2

ATTACHMENT POINT SELECTION

The process of manipulating the translational velocity of a large payload in space can be framed in terms of the piano mover's problem and robot motion planning. The piano mover's problem is defined in [28] as "finding a continuous motion that will take a given body from a given initial position to a desired final position, but which is subject to certain geometric constraints during the motion." We can rephrase their definition into the payload velocity control problem as "finding a continuous motion that will take a given payload from its current translational velocity relative to an inertial reference frame to a final velocity, but which is subject to constraints imposed by the geometric and physical characteristics of the cooperative spacecraft." In this context, we can consider the agents to be distributed components of a single robotic manipulator. Latombe in [14] breaks this type of robot manipulator problem into the following three sub-problems:

1. Grasping: The robot must select the configuration with which it will grasp the target object. The robot must then generate a path from its current configuration to the chosen configuration.

2. Transferring: The robot must generate a path that will take target object from its initial configuration to a final configuration.

3. Positioning: The robot must also manipulate the configuration of the target object during the transfer in order to satisfy geometric and physical constraints.

These sub-problems are directly applicable to the payload velocity control problem. The spacecraft must determine where they will attach to the payload, generate a path to get to those positions, generate a collaborative strategy for controlling the payload's velocity, and

determine how to continuously re-orient the payload during the transfer to accommodate any relevant constraints.

The remainder of this chapter will address the problem of choosing where to attach the spacecraft. We assume the payload is a rigid body that will have some number of allowable attachment points. This number may potentially be much greater than the number of available spacecraft. These allowable attachment points are all assumed to be structurally capable of withstanding the force of a spacecraft's thrust. We also assume the orientation of the payload during the propulsion maneuver is dictated by mission and structural constraints. This means the combined force of the spacecraft must act in a predetermined direction without changing the orientation of the payload. With these considerations in mind, we can now propose a solution to the attachment point selection problem.

2.1 Combinatorial Optimization Framework

Selecting attachment points can be framed in terms of a more general combinatorial optimization problem. Our goal is to choose a subset of the total attachment point set that will allow the available spacecraft to maximize their combined force in the desired direction of acceleration while minimizing net force in all other directions and net torque applied to the payload. This formulation is quite similar to the optimal actuator placement problem for dynamical systems. This problem has been studied in the context of structural vibrations [26], fluid flow [4], and controllability and observability Grammians [33]. Another similar formulation is the fixturing problem which aims to allow a robot to automatically select the optimal fixture points needed to grasp an object [7]. These are all intriguing problems but they do not directly address the payload attachment point selection problem. We begin our solution by developing a geometric description for the relevant characteristics of the spacecraft.

Each spacecraft has an identical thruster capable of producing a force of magnitude $F \in [0, \overline{F}]$. The thruster can be gimbaled by angle $\phi_g \in [0, \overline{\phi}_g]$ in any direction away from the its default $\phi_g = 0$ orientation. In the body frame of the payload, the force vector is represented as $\mathbf{F}_B$.

Each attachment point i on the surface of the payload has a position in the body frame

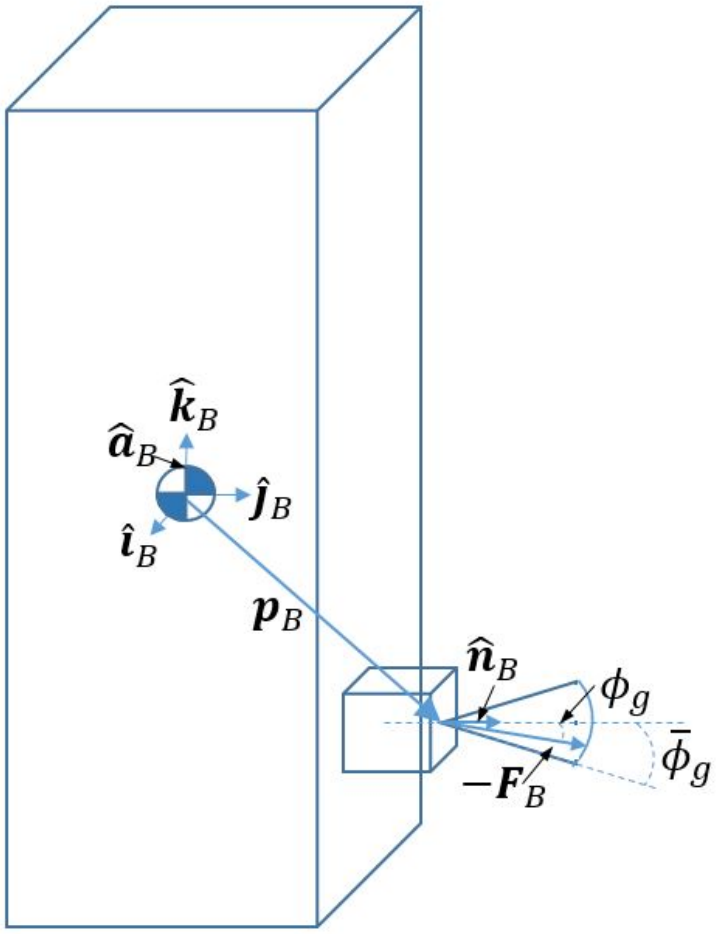

Figure 2.1: An example of the relevant vectors in the body reference frame of the payload. The large prism on the left represents the payload and the small cube on the right represents the spacecraft.

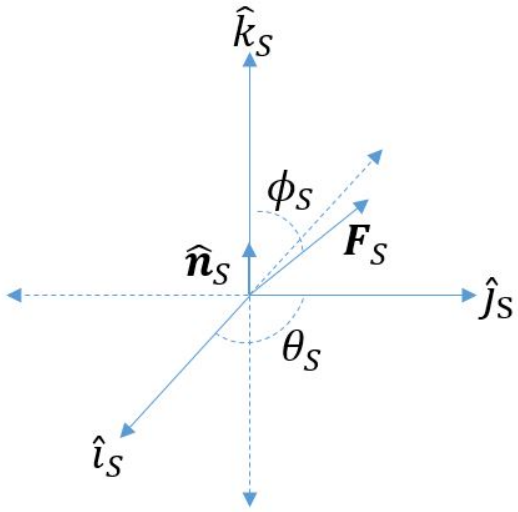

Figure 2.2: The force vector in the frame of a spacecraft.

of the payload denoted by $\mathbf{p}_{i_B}$. This vector points from the center of mass of the payload to the location of the gimbal joint of a thruster at that attachment point. We also define a unit vector $\hat{\mathbf{n}}_{i_B}$ which represents the direction normal to the payload's surface at the attachment point. And finally there is a unit vector $\hat{\mathbf{a}}_B$ in the body frame of the payload indicating the desired direction of acceleration. The problem is set up in such a way that this vector will simply be $\hat{\mathbf{a}}_B = \hat{\mathbf{k}}_B$. All of the vectors we have discussed so far are shown in Figure 2.1.

We now turn to the reference frame of the individual spacecraft. The direction of the gimbal orientation in this frame is given by

$$\hat{\mathbf{n}}_{i_S} = \begin{bmatrix} \cos\theta_i \sin\phi_i \\ \sin\theta_i \sin\phi_i \\ \cos\phi_i \end{bmatrix} \tag{2.1}$$

where θ_i is the angle of rotation about the $\hat{\mathbf{k}}_{i_S}$ axis and ϕ_i is the angle between $\hat{\mathbf{n}}_{i_S}$ and the $\hat{\mathbf{k}}_{i_S}$ axis. These relationships are shown in Fig. 2.2. The $\hat{\mathbf{n}}_{i_S}$ vector is related to $\hat{\mathbf{n}}_{i_B}$ by

$$\hat{\mathbf{n}}_{i_B} = \mathbf{C}_i \hat{\mathbf{n}}_{i_S} \tag{2.2}$$

where $\mathbf{C}_i$ is a direction cosine matrix describing the orientation of the individual spacecraft

frame to the payload frame. So we can also say that the force of a spacecraft is given by $\mathbf{F}_{i_B} = F_i \mathbf{C}_i \hat{\mathbf{n}}_{i_S}$.

Now we consider how the force applied by the spacecraft will torque the payload. If we assume the mass of the payload is much greater than the mass of the spacecraft, then we can neglect the displacement of the combined center of mass from the center of mass of the payload. Under that approximation, the torque $\boldsymbol{\tau}_{i_B}$ about the center of mass of the payload is given by

$$\boldsymbol{\tau}_{i_B} = \mathbf{p}_{i_B} \times \mathbf{F}_{i_B}. \tag{2.3}$$

We also use $\mathbf{F}_{i_B}$ to introduce the notion of a force deficit d_i. This represents the difference between $\overline{F}$ and the component of $\mathbf{F}_{i_B}$ acting in the $\hat{\mathbf{k}}$ direction. This component is denoted by $F_{z_{i_B}}$. The deficit is given by

$$d_i = F_{max} - F_{z_{i_B}}. \tag{2.4}$$

In other words, a deficit of zero means a spacecraft at this attachment point will contribute all of its force towards accelerating the payload in the $\hat{\mathbf{a}}_B$ direction.

We now have enough information to define the combinatorial optimization problem. The overall goal is to choose a subset of attachment points so that the available spacecraft can fire their thrusters at those locations to maximize the net force applied in the $\hat{\mathbf{a}}_B$ direction and minimize first, the net force applied in all other directions, and second, all components of $\boldsymbol{\tau}_{i_B}$. The mathematical problem can be stated as follows.

Given m attachment points on the surface of the payload and n identical spacecraft, we must choose n attachment points such that a total cost function J is minimized. The cost J for the chosen combination is determined by

$$J = a \cdot (\|\Sigma_{i=1}^{n} \boldsymbol{\tau}_{i_B}\|_2) + b \cdot \left(|\Sigma_{i=1}^{n} F_{x_{i_B}}| + |\Sigma_{i=1}^{n} F_{y_{i_B}}| \right) + c \cdot (\Sigma_{i=1}^{n} d_i) \tag{2.5}$$

where a, b, and c are coefficients for the weighting of the total torque magnitude, total magnitude of undesired force components, and total force deficit, respectively.

The formula for determining the number of assignment combinations K given m attachment points and n agents is

$$K = \frac{m!}{n!(m-n)!}. \tag{2.6}$$

A simple method for determining the best combination of attachment points would be to solve Eq. (2.5) for each of the K combinations, and choose the combination with the lowest cost. This method is certainly viable for simple problems. For example, solving Eq. (2.6) for 25 attachment points and 5 agents results in $53,130$ possible combinations. Depending on the application, it may be practical for a basic search algorithm to find the smallest entry in a vector of that size. However, the scale of the problem quickly grows as m and n increase. For example, increasing the previous values for m and n by a factor of 10, so that $m = 250$ and $n = 50$, gives approximately 10^{53} possible combinations. It is clear that we will need a more efficient algorithm in order to find feasible solutions for problems of that scale.

2.2 Attachment Point Selection Algorithm

The algorithm we propose for simplifying and solving this combinatorial optimization problem is based on the exponential growth of favorable subsets of size 2^r, where r is an integer of size $1 \leq r \leq \log_2(n)$ and n is the number of available spacecraft. A subset containing $R = 2^r$ attachment points is favorable if

$$\gamma \cdot J(R) < a \cdot \left(\Sigma_{i=1}^{R} \|\boldsymbol{\tau}_{i_B}\|_2\right) + b \cdot \left(\Sigma_{i=1}^{R} |F_{x_{i_B}}| + \Sigma_{i=1}^{R} |F_{y_{i_B}}|\right) + c \cdot \left(\Sigma_{i=1}^{R} d_i\right) \tag{2.7}$$

where $J(R)$ is the cost for the subset determined by (2.5) and γ is a constant used to represent the strictness of the favorability condition (a low γ would allow most combinations to be deemed favorable while a high γ would allow very few). Notice that the magnitude brackets are inside the summation on the right side of the inequality. The purpose of this inequality is to determine if particular combinations of attachment points allow the spacecraft to cancel out the undesired translational forces and torques by comparing the magnitude of a vector sum and the sum of the vector magnitudes. This greatly simplifies the problem because we are only ever considering combinations of two attachment points at a time (or two subsets of attachment points, as we will show later in this section). This means that at worst, when $R = 2$, we are operating on a vector of size $[K_0 \times 1]$ where

$$K_0 = \frac{m!}{2!(m-2)!} = \frac{m(m-1)}{2}. \tag{2.8}$$

Returning to the previous example where $m = 250$ and $n = 50$, we find that $K_O = 31,125$. This is a much more manageable data set than one with 10^{53} elements. Before addressing the process of constructing larger subsets, we must take a step back and address how $\mathbf{F}_{i_B}$ is chosen at each attachment point.

First we let $F_i = \overline{F}$ for all thrusters. Next we make three rules for choosing the gimbal orientation.

1. If $\cos^{-1}\left(\frac{\mathbf{C}_i\hat{\mathbf{k}}_{i_S}\cdot\hat{\mathbf{a}}_B}{|\mathbf{C}_i\hat{\mathbf{k}}_{i_S}||\hat{\mathbf{a}}_B|}\right) \leq \overline{\phi}_g$, then $\hat{\mathbf{n}}_{i_B} = \hat{\mathbf{a}}_B$.

2. If $\overline{\phi}_g < \cos^{-1}\left(\frac{\mathbf{C}_i\hat{\mathbf{k}}_{i_S}\cdot\hat{\mathbf{a}}_B}{|\mathbf{C}_i\hat{\mathbf{k}}_{i_S}||\hat{\mathbf{a}}_B|}\right) < 90^{\text{o}}$, then $\phi_i = \overline{\phi}_g$ and θ_i is chosen to maximize the $\hat{\mathbf{a}}_B$ component of $\hat{\mathbf{n}}_{i_B}$.

3. If $\cos^{-1}\left(\frac{\mathbf{C}_i\hat{\mathbf{k}}_{i_S}\cdot\hat{\mathbf{a}}_B}{|\mathbf{C}_i\hat{\mathbf{k}}_{i_S}||\hat{\mathbf{a}}_B|}\right) \geq 90^{\text{o}}$, then $\hat{\mathbf{n}}_{i_B} = \mathbf{C}_i\hat{\mathbf{k}}_{i_S}$.

These restrictions greatly simply the attachment point selection problem. By fixing the orientation and magnitude of the thrust at each attachment point, we are removing several degrees of freedom from the optimization process. The effect of more complex rules could be explored in future work related to this project. But it is important to realize that these rules are only used to increase the efficiency of the selection algorithm. We will eventually be taking full advantage of the operating range of F and ϕ_g in Chapter IV.

We now create an $[m \times 8]$ matrix whose columns store an identification number, $\tau_{x_{i_B}}$, $\tau_{y_{i_B}}$, $\tau_{z_{i_B}}$, $F_{x_{i_B}}$, $F_{y_{i_B}}$, d_i, and J for each attachment point. These values are all generated based on the relationships and rules developed so far in this chapter. The remainder of the attachment point selection algorithm follows a simple iterative process.

As alluded to earlier, the algorithm will build increasingly larger subsets containing $R = 2^r$ attachment points, where r is an integer of size $1 \leq r \leq \log_2(n)$. By only building subsets that meet the favorable subset condition from Eq. (2.8) and contain independent assortments of attachment points, the selection pool of subset choices containing R attachment points will quickly decrease as R increases. This segment of the algorithm is expressed as a flow chart in Fig. 2.3. In the figure we use the word "tier" to represent R, meaning the

"tiers" will contain subsets of $(1, 2, 4, 8, ...)$ attachment points. The final output of this algorithm will be a single matrix containing the identification numbers, net torque components, net undesired force components, net force deficit, and net cost for all favorable subsets in each tier. Each tier in the matrix is sorted by cost. The subsets in each tier will also be filtered so that attachment points will not appear in multiple subsets.

The next step is to pick a combination of the available favorable subsets so that n attachment points, corresponding to the number of available spacecraft, are present in the total subset. We accomplish this by picking the largest subset that contains at most n attachment points. If the subset contains fewer than n attachment points, we fill the remaining vacancies with increasingly smaller subsets. Before adding a subset, we verify that none of its attachment points are present in the current total subset. The final output of this algorithm will be an $[n \times 8]$ matrix whose columns contain the identification number, $\tau_{x_{i_B}}$, $\tau_{y_{i_B}}$, $\tau_{z_{i_B}}$, $F_{x_{i_B}}$, $F_{y_{i_B}}$, d_i, and J for each attachment point in the final subset. This stage of the algorithm is shown in Fig. 2.4.

A reduction in the total cost could be obtained by choosing the subset from the current tier that causes the total subset to be most favorable. But this would decrease the efficiency for the algorithm and since the tiers are already sorted by minimum cost, choosing the subsets in this way would not lead to a large reduction in J.

2.3 Selection Algorithm Performance

The easiest way to test the accuracy and efficiency of the attachment point selection algorithm is to place all the attachment points on the surface of a sphere. However, it is tempting to look at the results of these tests and conclude the selection algorithm somehow takes advantage of the geometric symmetry of the sphere. This is not the case. The algorithm only considers the data in the $[m \times 8]$ matrix mentioned above and would display the same performance on a test generated from an arbitrary payload geometry. The sphere is just a tool for quickly modifying the number and position of available attachment points. Our first goal is to compare how the algorithm's choices compare to the best choices as determined by a brute force approach.

We initiate the simulation be creating a 100 m sphere to represent the payload. Each

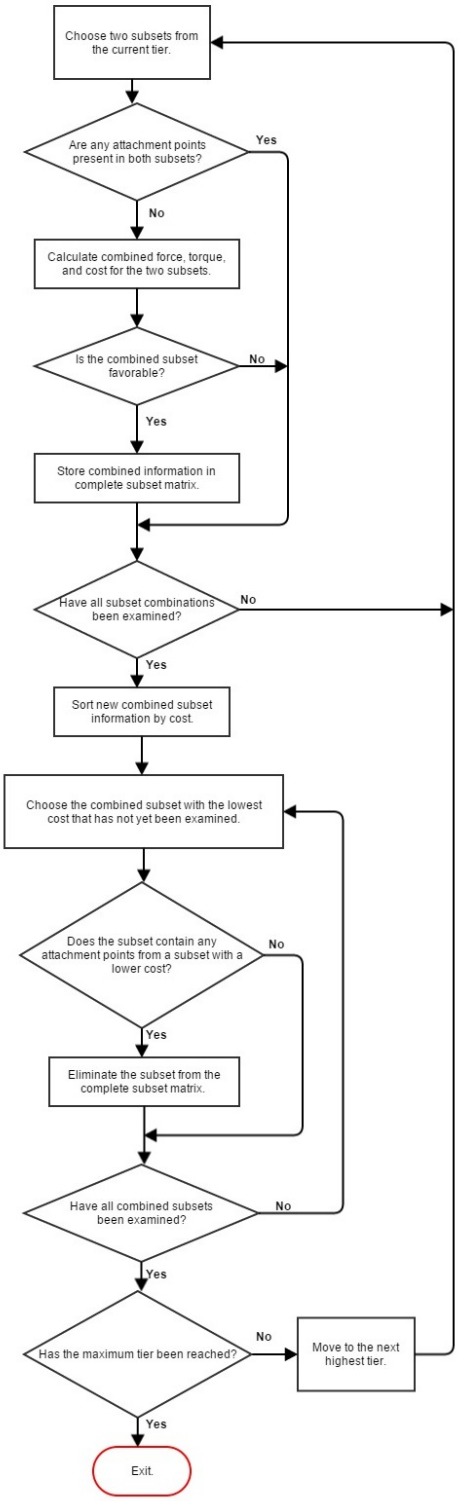

Figure 2.3: Flow chart describing the algorithm for building the complete matrix of favorable subsets.

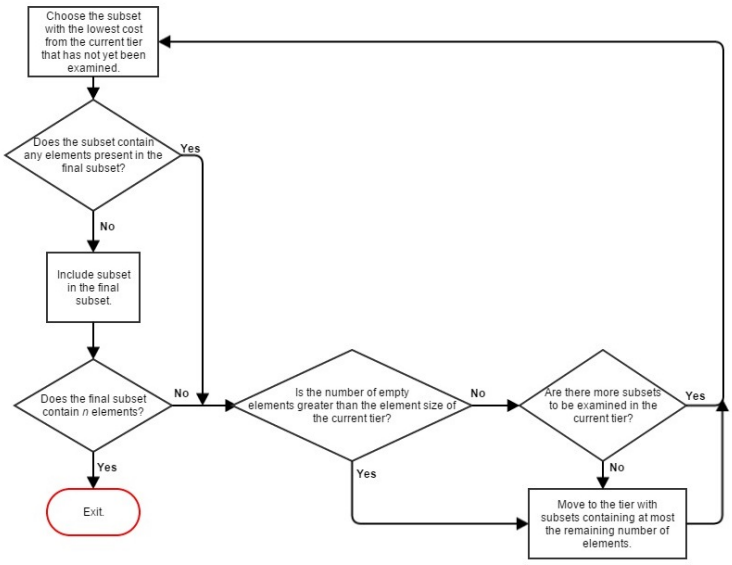

Figure 2.4: Flow chart describing the algorithm for building the final list of attachment points from the complete matrix of favorable subsets.

spacecraft is capable of producing 1 N of thrust and has a maximum gimbal angle of 15°. We create 22 attachment points on the surface of the sphere. The direction of desired acceleration is along the vertical axis through the center of the sphere. For reference, the distribution of combinations for 22 attachment points based on Eq. (2.6) is shown in Fig. 2.5.

We now search through all possible combinations for the combination with the smallest C using a brute force approach and then compare this result to the combination chosen by the selection algorithm. In Fig. 2.6 we compare the standardized costs of the selection algorithm results to the minimum, maximum, and mean standardized costs of the brute force results. The standardized cost for a given selection of attachment points is calculated by dividing each of the three terms of the cost by the corresponding maximum value of that as determined by the brute force approach. This makes 1.0 the maximum possible value of a term and 3.0 the maximum standardized cost for a selection of attachment points.

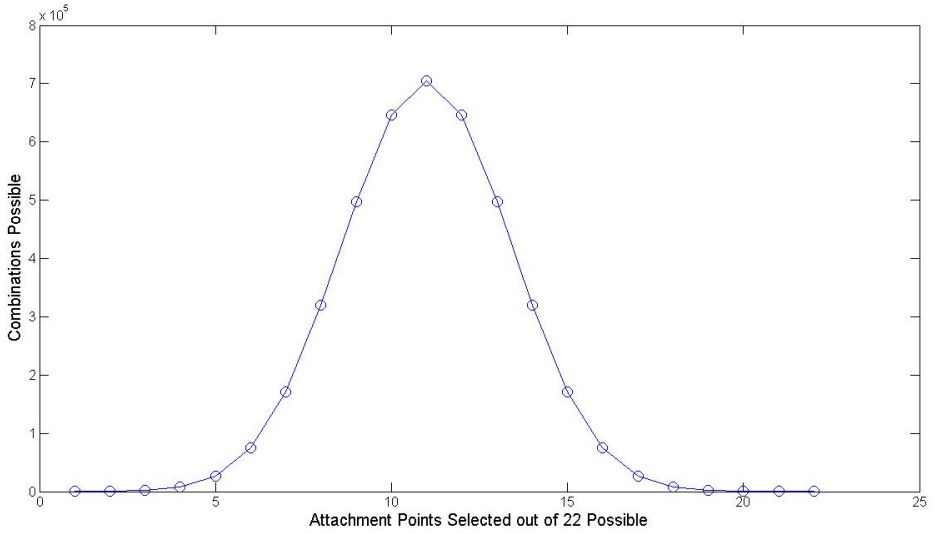

Figure 2.5: The total number of combinations when choosing n attachment points out of a set of 22 points.

In some cases the algorithm chooses the same selection of attachment points as the brute force method. And in the worst case for this example, when choosing five attachment points, the selection algorithm result has a standardized cost that is 0.4585 greater than the minimum brute force cost. This drop in performance comes from the fact that the selection algorithm is attempting to build favorable subsets from a very coarse data set. So although a small set of available attachment points is an advantage for the brute force method, it is actually a severe disadvantage for the selection algorithm. The results in the next figure are more promising for the utility of the selection algorithm. In Figure 2.7 we compare the computation time required for each method to find the combination with minimum cost. The selection algorithm takes 0.225 seconds to find the best selection of nine out of 22 attachment points while the brute force method takes $4,628$ seconds. Note that the graph for the brute force method's computation time roughly follows the behavior shown in Fig. 2.5.

We also compare the methods using 26 available attachment points that are located on

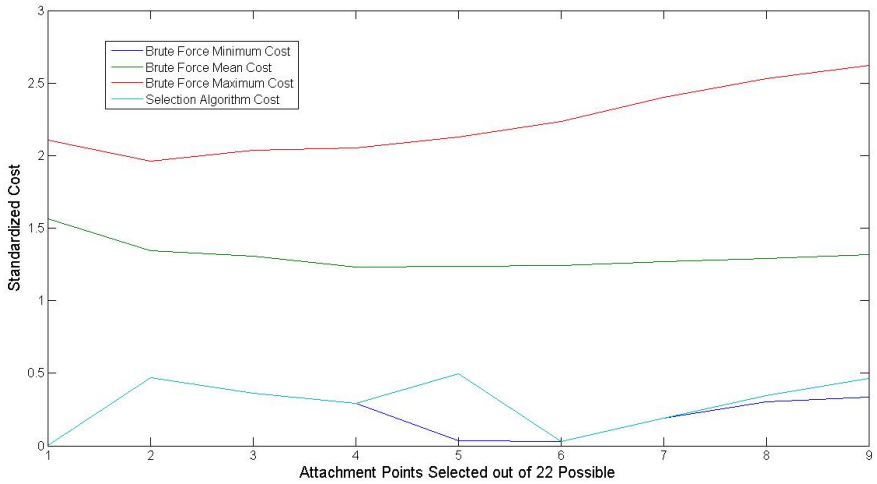

Figure 2.6: A comparison of the standardized costs as determined by the brute force method and selection algorithm on points taken from the surface of a sphere.

the bottom half of a sphere. This is intended to model a data set that has been chosen to not include any attachment points that would contribute force opposing the desired direction of acceleration. The results are shown in Figure 2.8. In this case, the improvement of the costs of the selection algorithm's results over the mean costs are at most 21% less than the improvement of the optimal costs over the mean costs.

So it appears that the more densely packed data set improved the selection algorithm's results. And again its time performance was clearly superior to the brute force method's. The brute force method took $6,768$ seconds to find the best selection of seven attachment points while the selection algorithm took only 0.346 seconds.

We can now test the algorithm on problems that would be computationally unfeasible for the brute force method. We again use attachment points located on the bottom half of a sphere, but for this test we include 240 possible attachment points. We determine the computation time using the tic-toc command in MATLAB. The results in Fig. 2.9 show an approximately constant computation time for selections including at least two attachment points. This indicates that the computation time is largely dictated by the time required

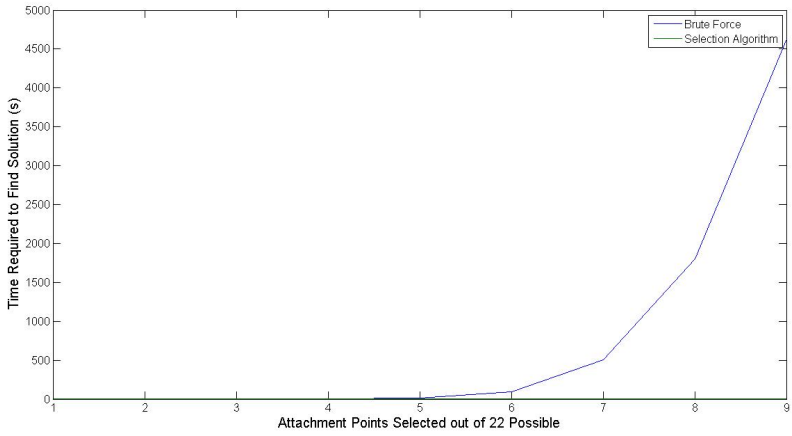

Figure 2.7: Time comparison of brute force method and selection algorithm on points taken from the surface of a sphere.

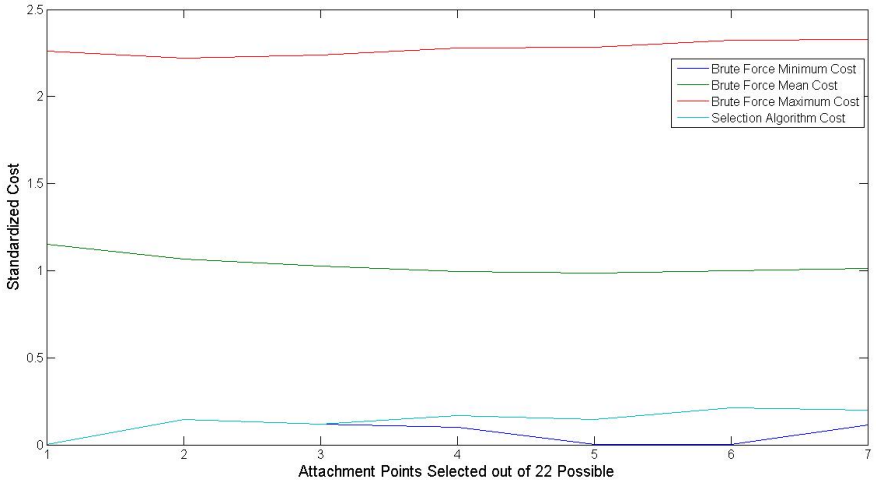

Figure 2.8: A comparison of the standardized costs as determined by the brute force method and selection algorithm on points taken from the surface of the bottom half of a sphere.

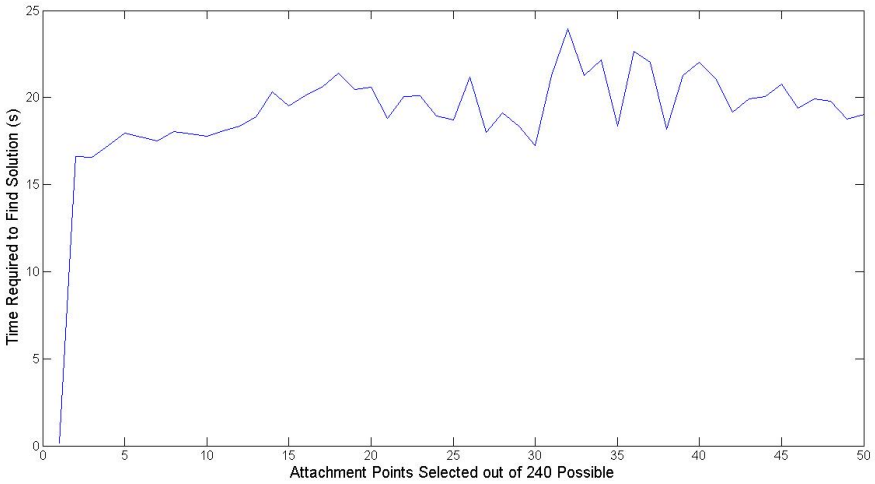

Figure 2.9: Computation time for selection algorithm operating on 240 possible attachment points

to compute the data for K_0 combinations. This is an enormous improvement over the computation time that would have been required by a brute force method.

And lastly we can examine the physical results of the selection algorithm's choices. The first graph in Fig. 2.10 shows, at least in this test, that the force in the desired direction of acceleration increases linearly as more agents are included. The other two figures show that the undesirable parameters are kept very close to zero.

2.4 Chapter Summary

In this chapter we developed an algorithm that can be used to select a suitable combination of attachment points on the surface of a payload for a group of external propulsion agents. The algorithm produces solutions that are close to optimal and its efficiency is many orders of magnitude better than a brute force method.

The application of this algorithm is not limited to the payload velocity control problem. The strategy of solving combinatorial optimization problems through the construction of favorable subsets could be useful in agriculture and finance. In the case of agriculture, a

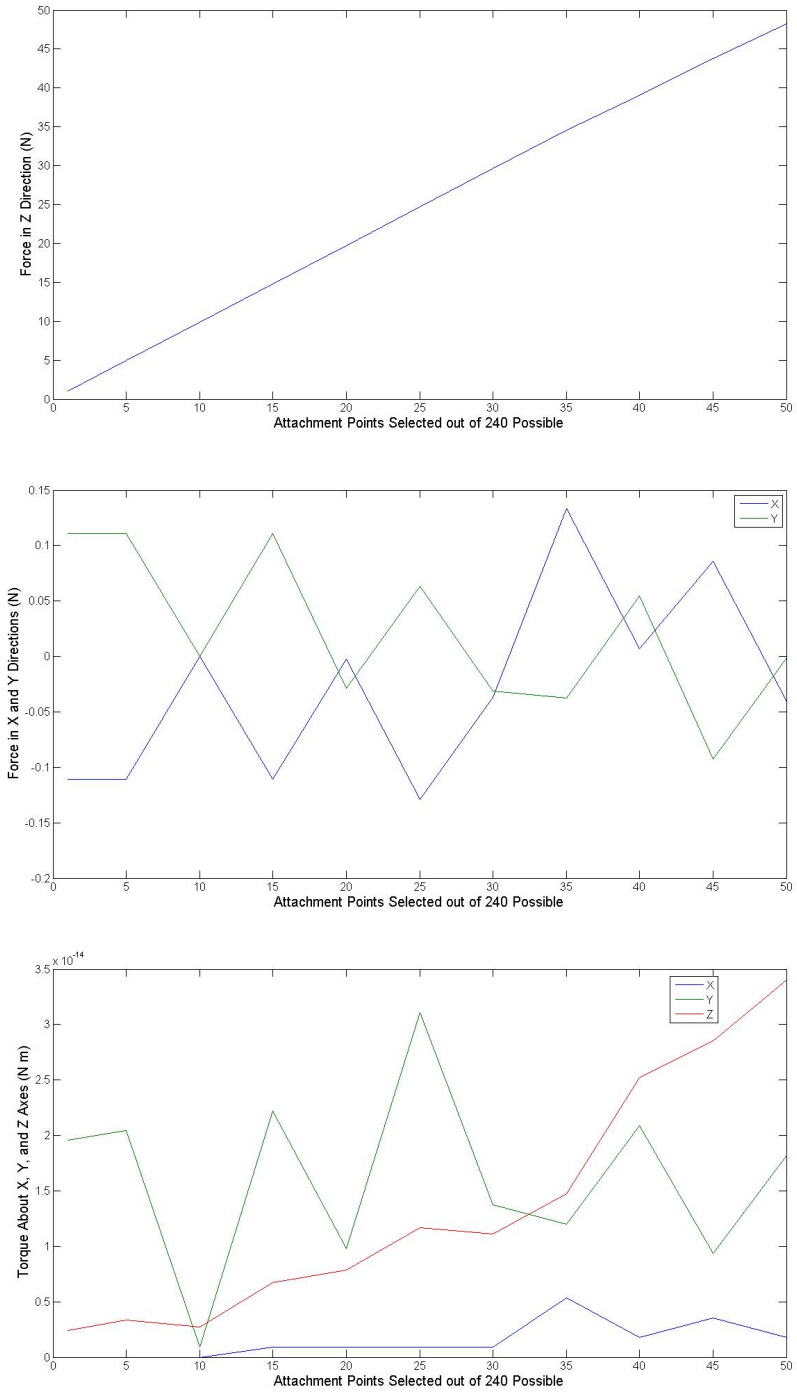

Figure 2.10: The development in force and torque as attachment points are selected out of 240 possible points. The data corresponds to subsets containing multiples of five points (1,5,10,...,50).

farmer could be looking to invest in new properties. Some of the properties are dairy farms and the other properties are apple orchards. If the farmer wants to maximize his return on investment but maintain a neutral carbon footprint, he should pick a balance of properties such that profit is maximized but methane produced by the cows is balanced by the carbon dioxide consumed by the apple trees.

And in the finance example, a portfolio manager could be looking to maximize an investment portfolio's expected rate of return but minimize the deviation of its average volatility from some benchmark. So he should pick a balance of investments that maximize expected earnings but whose volatility collectively balances around the desired average.

The attachment point selection algorithm could be useful in both of these examples.

Chapter 3

PATH PLANNING

The previous chapter addressed the problem of selecting suitable attachment points for the spacecraft on the surface of the payload. This chapter will examine the three-dimensional path planning problem of transferring the spacecraft from their initial positions near the payload to their attachment points. Three-dimensional path planning is being studied in several different fields including unmanned aerial vehicles [21], climbing robots [37], and medical imaging [11]. Our proposed solution is based on the visibility graph roadmap method in [14]. The visibility graph roadmap method is used here because of its compatibility with constrained model predictive control.

Additional path planning approaches include exact cell decomposition, approximate cell decomposition, and potential field methods [14]. Exact cell decomposition breaks the environment into a collection of non-overlapping regions. A connectivity graph representing the adjacency relation among the cells is constructed and searched for a desirable path. Approximate cell decomposition is similar but it requires the cells to have a simple prespecified shape. Potential field methods treat the vehicle as a particle under the influence of an artificial potential field. Artificial forces in this potential field either attract the vehicle to the target or repel the vehicle away from obstacles.

3.1 Modified Visibility Graph

Latombe in [14] explains that the visibility graph is "constructed by connecting every pair of vertices in [the configuration space] by a straight segment, if this segment does not traverse the interior of [any obstacles]." In a different section he states that "once constructed, [this] roadmap R is used as a set of standardized paths. Path planning is reduced to connecting the initial and goal configurations to R, and searching R for a path." A simple two dimensional example of a visibility graph roadmap is shown in Fig. 3.1.

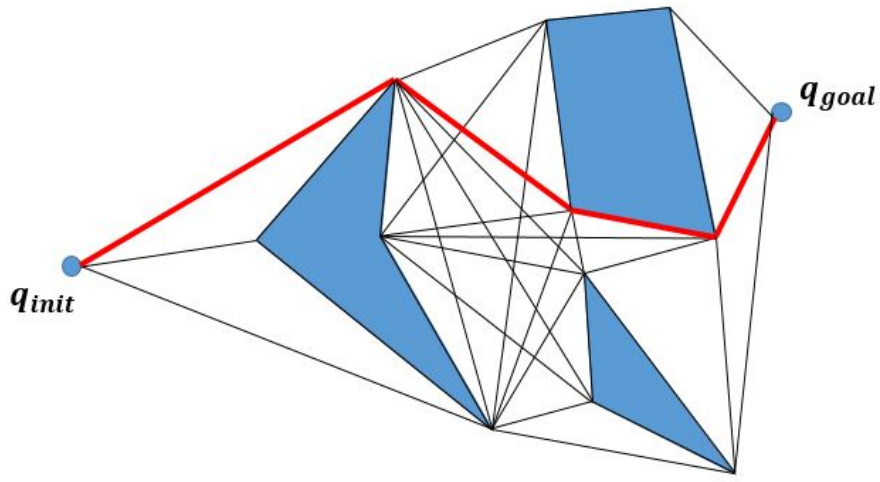

Figure 3.1: An example of a two dimensional visibility graph.

In our case, the three dimensional obstacle will be the payload, the initial configuration will be the spacecraft's initial position relative to the payload, and the final configuration will be the position of the spacecraft at the attachment point. In order to simplify the problem and add an additional layer of safety, we will enclose the payload in an imaginary polyhedron hull. This will reduce the number of vertices in the roadmap and allow for us to develop safety constraints later in this chapter.

In order to simplify the trajectory planning process, we will assume the spacecraft only use their reaction thrusters to maneuver around the payload. If a spacecraft were to use its main thruster then it would also be required to constantly orient the thruster away from the payload as discussed in [6]. We also assume the spacecraft are already in the orientation required for attachment to the payload. This allows us to focus on the problems of building the visibility graph and developing a control strategy for the spacecraft to follow the chosen trajectory.

An example of a simplified three dimensional visibility graph and a suitable path is shown in Figure 3.2. This visibility graph is simplified because it is missing many possible path segments.

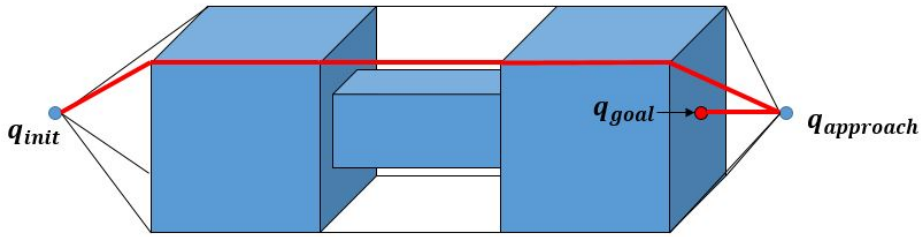

Figure 3.2: An example of a simplified three dimensional visibility graph.

We can now begin to develop the visibility graph. The proposed method is designed so that a computer can automatically generate the visibility graph based on the vertices and faces of the obstacle. The problem that the program must solve is to determine if a path segment between two vertices is admissible. The path segment is admissible if it does not intersect the interior region of a face of the obstacle. So for every path segment and every face of the obstacle, the program must determine if the path segment intersect the plane containing the polygon face and then if the intersection point is inside the boundary of the face. The following three subsections develop the tools needed to solve those problems.

3.1.1 Intersecting Line Segments

We start with the process of determining if two line segments intersect. An illustration of intersecting line segments is shown in Figure 3.3. Consider the following two lines

$$\mathbf{p}_1(t) = \mathbf{u}_1 + t\mathbf{v}_1 \tag{3.1}$$

$$\mathbf{p}_2(s) = \mathbf{u}_2 + s\mathbf{v}_2 \tag{3.2}$$

where $s, t \in \Re$ and $\mathbf{u}_1, \mathbf{v}_1, \mathbf{x}_1, \mathbf{u}_2, \mathbf{v}_2, \mathbf{x}_2 \in \Re^3$. The line segments we will be considering include the added restriction that $s, t \in [0, 1]$. If the two line segments intersect, then there will be a pair s and t satisfying the following equation

$$\mathbf{u}_1 - \mathbf{u}_2 = s\mathbf{v}_2 - t\mathbf{v}_1. \tag{3.3}$$

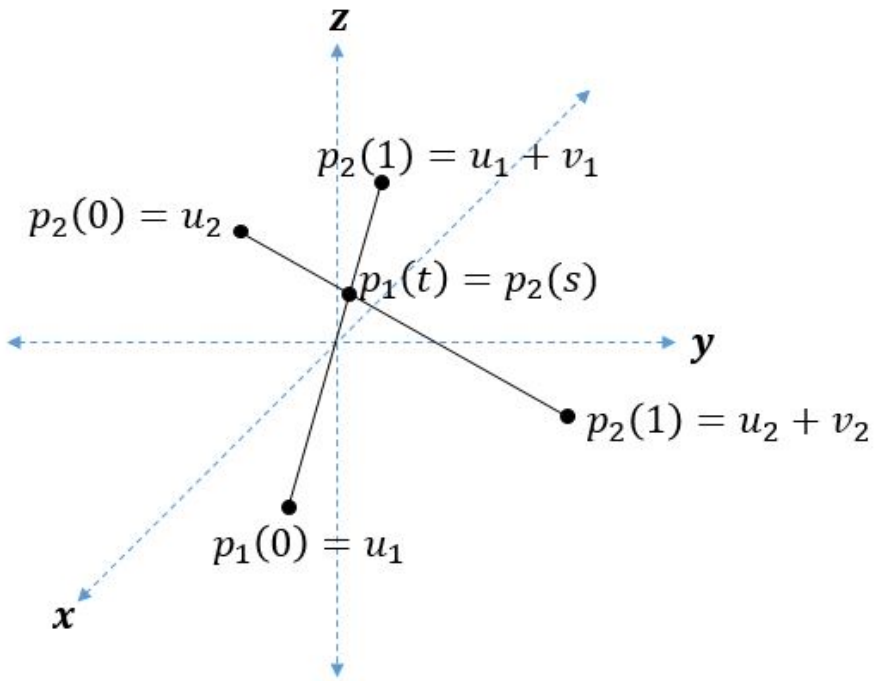

Figure 3.3: Equations describing the intersection of two line segments.

The coordinates of $\mathbf{u}_1, \mathbf{v}_1, \mathbf{u}_2$, and $\mathbf{v}_2$ can be written as follows

$$\mathbf{u}_0 = [x_{1_i}, y_{1_i}, z_{1_i}]' \tag{3.4}$$

$$\mathbf{v}_0 = [x_{1_f}, y_{1_f}, z_{1_f}]' \tag{3.5}$$

$$\mathbf{u}_1 = [x_{2_i}, y_{2_i}, z_{2_i}]' \tag{3.6}$$

$$\mathbf{v}_1 = [x_{2_f}, y_{2_f}, z_{2_f}]' \tag{3.7}$$

so that (3.3) can be written as

$$\begin{bmatrix} (x_{1_i} - x_{2_i}) \\ (y_{1_i} - y_{2_i}) \\ (z_{1_i} - z_{2_i}) \end{bmatrix} = \begin{bmatrix} x_{2_f} & x_{1_f} \\ y_{2_f} & y_{1_f} \\ z_{2_f} & z_{1_f} \end{bmatrix} \begin{bmatrix} s \\ -t \end{bmatrix}. \tag{3.8}$$

This equation takes the form $Ax = b$ and can be solved easily. If the resulting $s, t \in [0, 1]$ then we can say the line segments intersect. And if the resulting $s, t \in (0, 1)$ then we can say the line segments intersect at a point that is not a boundary point for either line segment.

3.1.2 Line Segment Intersecting a Plane

Next we must determine if a line segment intersects a closed polygon. First consider the plane in which the polygon exists. Let the vertices of the polygon be $\mathbf{V}_1, \mathbf{V}_2, ..., \mathbf{V}_n \in \Re^3$. Then we can say the vector normal to the plane containing the polygon is

$$\mathbf{n} = (\mathbf{V}_2 - \mathbf{V}_1) \times (\mathbf{V}_n - \mathbf{V}_1). \tag{3.9}$$

If we use the parametric equation for a line given by Eq. (3.2), we can say that the line segment is parallel to the plane if $\mathbf{n} \cdot \mathbf{v} = 0$. If the line and plane are not parallel, then the line must intersect the plane at a unique point [34]. We can find this point as

$$s_I = \frac{-\mathbf{n} \cdot \mathbf{w}}{\mathbf{n} \cdot \mathbf{u}} \tag{3.10}$$

where $\mathbf{w}$ connects $\mathbf{P}_i$ and $\mathbf{V}_0$ as shown in Fig. 3.4. If $s_I \in (0, 1)$ then the line segment intersects the plane at a point that is not one of the line segment's boundary points.

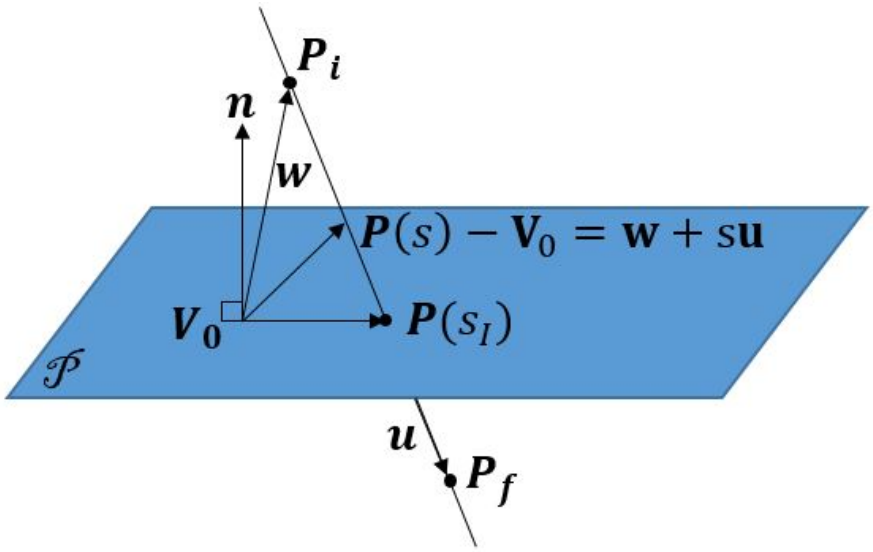

Figure 3.4: A line segment intersecting a plane.

3.1.3 Point Inside a Closed Polygon

Next we must determine if the intersection point is inside a closed polygon. The Jordan Curve Theorem for Polygons will be a convenient tool for this test [19]. Roughly speaking, the theorem states that a point is inside a polygon if a ray starting from the point and cast in any direction, within the plane containing the polygon, intersects the boundary of the polygon an odd number of times. And conversely the point is outside of the polygon if the ray intersects the polygon an even number of times. A visualization for the theorem is shown in Figure 3.5. Let the edge list E for the polygon contain all pairs of vertices $[(V_i, V_j), (V_j, V_k), \cdots, (V_n, V_i)]$ that form the edges of the closed polygon. Now let $\mathbf{k} = \mathbf{P}(s_I) + c \cdot t \cdot (\mathbf{V}_0 - \mathbf{P}(s_I))$ be a line segment in the plane of the polygon that originates at the point of intersection between the original line segment and the plane of the polygon, where c is a scaling constant. In order for $\mathbf{k}$ to sufficiently simulate a ray, we should choose a c that is much larger than the scale of the problem. So if the dimensions of the spacecraft are on the order of hundreds of meters, we could choose a c on the order of millions of meters.

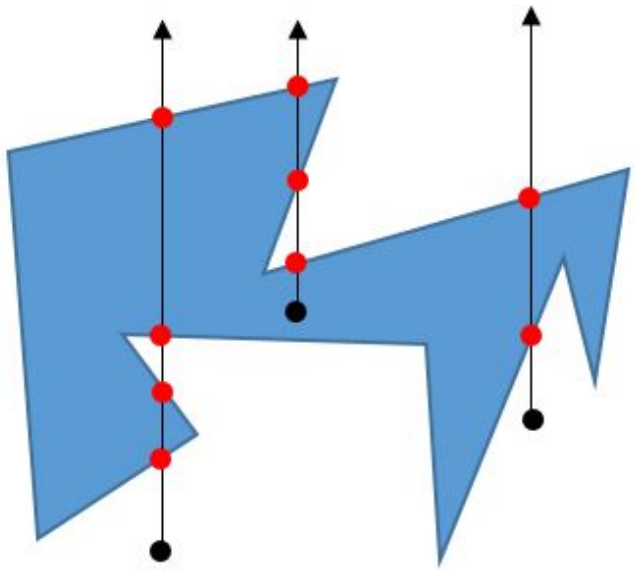

Figure 3.5: An illustration of the Jordan curve theorem for polygons. Only rays with an even number of intersections are inside the polygon.

Now it is only a matter of determining how many of the polygon's edges are intersected by **k**. This can easily be done with the process described in Subsection 3.1.1. By calculating s_I for the each edge and t_I for **k**, we can determine which of the edges **k** intersects. If we find that $s, t \in (0, 1)$, then we can say that **k** intersects the edge at a point that is not a boundary point of either line segment. If we find that $s = [0, 1]$ and $t \in (0, 1)$ then we can say **k** intersects a vertex of the polygon. If $s \in [0, 1]$ and $t = 0$, then the point of intersection is on an edge of the polygon. Now by repeating this process for every edge of the polygon, we can determine how many times **k** intersects the edge of the polygon. If this number is odd, then we can finally say that the original line segment passes through the polygon in three dimensional space. And then by repeating the entire process for every face of the polyhedron hull, we can determine if the original line segment intersects the obstacle.

3.2 Path Selection

Once the visibility graph roadmap is generated, we must find the shortest path that connects $\mathbf{q}_{init}$ to $\mathbf{q}_{approach}$. Fortunately there are several algorithms capable of solving this problem. The most intuitive of these is the A^* algorithm. This algorithm was developed by Hart, Nilsson, and Raphael and is summarized in [14]. Latombe explains that the A^* algorithm "explores [the graph] G iteratively by following paths originating at [the initial node] N_{init}. At the beginning of every iteration, there are some nodes that the algorithm has already visited, and there may be others that are still unvisited. For each visited node N, the previous iterations have produced one or several paths connecting N_{init} to N, but the algorithm only memorizes a representation of of a path of minimum cost (among those so far constructed). At any instant, the set of all such paths forms a spanning tree T of the subset of G so far explored. T is represented by associating to each visited node N (except N_{init}) a pointer to its parent node in the current T."

More specifically, the algorithm "assigns a cost function $f(N)$ to every node in N in the current T. This function is an estimate of the cost of the minimum-cost path in G connecting N_{init} to N_{goal} and constrained to go through N." It is computed with

$$f(N) = g(N) + h(N) \tag{3.11}$$

where $g(N)$ is the cost of the path between N_{init} and N in the current T, $h(N)$ is a heuristic estimate of the cost $h^*(N)$ of the minimum-cost path between N and N_{goal} in G.

Further the heuristic function h is said to be admissible if and only if it satisfies:

$$\forall N \in G : 0 \leq h(N) \leq h^*(N). \tag{3.12}$$

Under the condition that h is admissible, A^* is guaranteed to generate a minimum-cost path between the initial and goal nodes whenever these two nodes are connected by a path in G, and to return failure otherwise. Additionally the heuristic function h is said to be locally consistent if and only if for every pair of nodes N and N', such that N' is adjacent to N, it verifies

$$0 \leq h(N) \leq h(N') + k(N, N'). \tag{3.13}$$

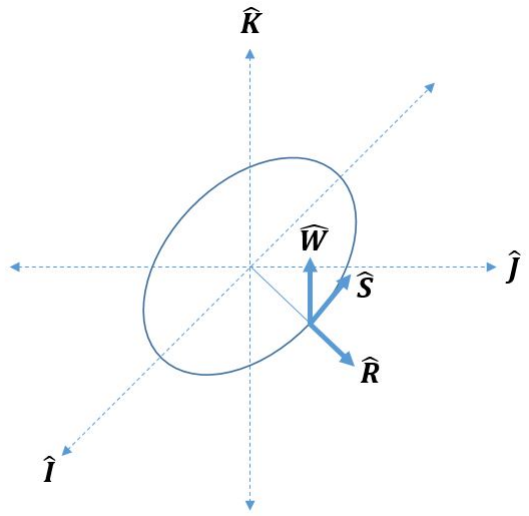

Figure 3.6: The RSW coordinate frame of an object that is orbiting around the Earth.

The heuristic function we will use is is the Euclidean distance from N to N_{goal}, meaning $k(N_1, N_2)$ is the length of the path in three-dimensional space joining N_1 and N_2. This function is locally consistent and admissible. See Appendix C in [14] for a full explanation of the A^* algorithm.

3.3 Dynamics in the RSW Coordinate Frame

The relative motion of two spacecraft in close proximity is best expressed in the *RSW* coordinate frame. As described by David Vallado in [35], in this frame "the R axis always points from the Earth's center along the radius vector toward the satellite as it moves through the orbit. The S axis points in the direction of (but not necessarily parallel to) the velocity vector and is perpendicular to the radius vector. The W axis is normal to the orbital plane. The S axis is usually not aligned with the velocity vector except for circular orbits or for elliptical orbits at apogee and perigee. The coordinate system applies to all orbit types." The RSW coordinate frame is shown in Figure 3.6.

If we assume a circular orbit, the dynamics $\ddot{\mathbf{r}}_{rel_R}(t)$ of an interceptor satellite relative to

a target satellite is given by

$$\ddot{\mathbf{r}}_{rel_R} = -\omega^2(x\hat{R} + y\hat{S} + z\hat{W} - 3x\hat{R}) + \mathbf{F}_{thrust} + 2\omega\dot{y}\hat{R} - 2\omega\dot{x}\hat{S} + \omega^2 x\hat{R} + \omega^2 y\hat{S} \quad (3.14)$$

where x is the displacement in the $\hat{R}$ direction, y is the displacement in the $\hat{S}$ direction, z is the displacement in the $\hat{W}$ direction, $\mathbf{F}_{thrust}$ is the applied force vector, and ω is the angular rate of the target's circular orbit. The angular rate ω can be found with

$$\omega = \sqrt{\frac{\mu}{r_{tgt}^3}} \quad (3.15)$$

where $\mu = 398,600.441\frac{km^3}{s^2}$ is the gravitational parameter of the Earth and r_{tgt} is the radius of the target's orbit from the center of the Earth. Writing each vector component of (3.14) separately yields

$$\ddot{x} - 2\omega\dot{y} - 3\omega^2 x = f_x \quad (3.16)$$

$$\ddot{y} + 2\omega\dot{x} = f_y \quad (3.17)$$

$$\ddot{z} + \omega^2 z = f_z \quad (3.18)$$

which are collectively referred to as Hill's equations or the Clohessy-Wiltshire equations. They are the equations that will govern the dynamics of the spacecraft as they move toward their attachment points on the payload.

3.4 Review of Constrained Model Predictive Control

The goal of this section is to design a control strategy that will allow the spacecraft to follow the path created by the visibility graph roadmap while maintaining a safe distance from the payload. Constrained model predictive control will be our primary tool in building this strategy. This type of control originated in chemical processing plants but is becoming increasingly popular in a wide variety of applications [32],[22],[36]. The following summary of model predictive control is taken from the publicly available lecture notes in [3] from the University of Oxford.

We start with the discrete-time linear system given by

$$x(k+1) = Ax(k) + Bu(k) \quad (3.19)$$

where $x(k)$ and $u(k)$ are the model state and input vectors at the kth sampling instant. We use the following notation to represent a predicted input sequence over N sampling intervals and the corresponding state predictions

$$\mathbf{u}(k) = \begin{bmatrix} u(k|k) \\ u(k+1|k) \\ \vdots \\ u(k+N-1|k) \end{bmatrix}, \quad \mathbf{x}(k) = \begin{bmatrix} x(k+1|k) \\ x(k+2|k) \\ \vdots \\ u(k+N|k) \end{bmatrix} \tag{3.20}$$

where $u(k+1|k)$ and $x(k+1|k)$ denote input and state vectors at time $k+i$ that are predicted at time k.

Then the evolution of the discrete state-space model over time can be described with

$$x(k|k) = x(k) \tag{3.21}$$

$$x(k+1|k) = Ax(k) + Bu(k|k) \tag{3.22}$$

$$x(k+2|k) = A^2x(k) + ABu(k|k) + Bu(k+1|k) \tag{3.23}$$

$$\vdots \tag{3.24}$$

$$x(k+i|k) = A^ix(k) + C_i\mathbf{u}(k),\ i = 0, \ldots, N \tag{3.25}$$

We can write this in a simpler form as

$$\mathbf{x}(k) = \mathbf{M}x(k) + \mathrm{C}\mathbf{u}(k) \tag{3.26}$$

where

$$\mathrm{M} = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix} \tag{3.27}$$

and

$$\mathrm{C} = \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1} & A^{N-2}B & \cdots & B \end{bmatrix}. \tag{3.28}$$

The cost function for the optimization is given by

$$J(k) = \sum_{i=0}^{N-1} [x^T(k+i|k)Qx(k+i|k)+u^T(k+i|k)Ru(k-i|k)]+x^T(k+N|k)\bar{Q}x(k+N|k). \quad (3.29)$$

Substituting for $x(k+i|k)$ in (3.29) and collecting terms gives

$$J(k) = \mathbf{u}^T(k)H\mathbf{u}(k) + 2x^T(k)F^T\mathbf{u}(k) + x^T(k)Gx(k) \quad (3.30)$$

where

$$H = \mathrm{C}^T\tilde{Q}C + \tilde{R} \quad (3.31)$$

$$F = C^T\tilde{Q}\mathrm{M} \quad (3.32)$$

$$G = \mathrm{M}^T\tilde{Q}\mathrm{M} + Q \quad (3.33)$$

with

$$\tilde{Q} = \begin{bmatrix} Q & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & Q & 0 \\ 0 & \cdots & 0 & \overline{Q} \end{bmatrix} \quad (3.34)$$

$$\tilde{R} = \begin{bmatrix} R & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & R & 0 \\ 0 & \cdots & 0 & R \end{bmatrix}. \quad (3.35)$$

We can write linear input and state constraints:

$$\underline{u} \leq u(k) \leq \overline{u} \quad (3.36)$$

$$\underline{x} \leq x(k) \leq \overline{x} \quad (3.37)$$

in the form $A_c\mathbf{u} \leq b_c$, which can be interpreted by a quadratic programming solver. The input constraints can be expressed in terms of $\mathbf{u}(k)$ as

$$\begin{bmatrix} I \\ -I \end{bmatrix} \mathbf{u}(k) \leq \begin{bmatrix} \mathbf{1} \otimes \overline{u} \\ -\mathbf{1} \otimes \underline{u} \end{bmatrix} \quad (3.38)$$

where $\otimes$ denotes the Kronecker product, I is an identity matrix of size $[(N \cdot n_u) \times (N \cdot n_u)]$, $\mathbf{1}$ is a vector of ones of size $[N \times 1]$, and n_u is the dimension of u.

The state constraints can be written as

$$\begin{bmatrix} C_i \\ -C_i \end{bmatrix} \mathbf{u}(k) \leq \begin{bmatrix} \overline{x} \\ -\underline{x} \end{bmatrix} + \begin{bmatrix} -A^i \\ A^i \end{bmatrix} x(k),\ i = 1, \ldots, N. \tag{3.39}$$

Taken together, the input and state constraints can then be expressed as

$$A_c \mathbf{u}(k) \leq b_0 + B_x x(k) \tag{3.40}$$

where A_c, b_0, B_x are constant matrices that can be computed offline. The complete optimization problem can finally be written as:

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & \mathbf{u}^T H \mathbf{u} + 2x^T(k) F^T \mathbf{u} \\ \text{subject to} \quad & A_c \mathbf{u}(k) \leq b_0 + B_x x(k) \end{aligned}$$

which can be efficiently solved by a quadratic programming solver using an active set or interior point algorithm. We will be making use of MATLAB's "quadprog" solver which can be set to use either type of algorithm.

3.5 Model and Constraints

3.5.1 Discrete Time Representation

The goal now is to apply the Clohessy-Wiltshire equations and the discrete linear quadratic tracker to the nominal path generated from the visibility graph roadmap and path selected by the A^* algorithm. We can start by writing the Clohessy-Witshire equations (3.18) in the standard continuous state-space system form:

$$\dot{x} = A_c x + B_c u. \tag{3.41}$$

Assuming we can perfectly observe the spacecraft's position and velocity, this system will be

$$\dot{\mathbf{X}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3\omega^2 & 0 & 0 & 0 & 2\omega & 0 \\ 0 & 0 & 0 & -2\omega & 0 & 0 \\ 0 & 0 & -\omega^2 & 0 & 0 & 0 \end{bmatrix} \mathbf{X} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 \\ 0 & \frac{1}{m} & 0 \\ 0 & 0 & \frac{1}{m} \end{bmatrix} \mathbf{U} \tag{3.42}$$

where m is the mass of the spacecraft. The state vector $\mathbf{X}$ is taken to be

$$\mathbf{X} = \begin{bmatrix} x : (\text{displacement in } R \text{ direction}) \\ y : (\text{displacement in } S \text{ direction}) \\ z : (\text{displacement in } W \text{ direction}) \\ \dot{x} : (\text{displacement velocity in } R \text{ direction}) \\ \dot{y} : (\text{displacement velocity in } S \text{ direction}) \\ \dot{z} : (\text{displacement velocity in } W \text{ direction}) \end{bmatrix} \tag{3.43}$$

and the input vector $\mathbf{U}$ is taken to be

$$\mathbf{U} = \begin{bmatrix} f_x : (\text{force applied in } R \text{ direction}) \\ f_y : (\text{force applied in } S \text{ direction}) \\ f_z : (\text{force applied in } W \text{ direction}) \end{bmatrix}. \tag{3.44}$$

However, in order to use model predictive control techniques, we must transform (3.42) into a discrete-time representation

$$\mathbf{X}(k+1) = A\mathbf{X}(k) + B\mathbf{U}(k). \tag{3.45}$$

We can find the discrete-time state space matrices with the following equation [5]

$$\begin{bmatrix} A & B \\ 0^{[3\times 6]} & I^{[3\times 3]} \end{bmatrix} = e^{\begin{bmatrix} A_C & B_C \\ 0^{[3\times 6]} & 0^{[3\times 3]} \end{bmatrix}}. \tag{3.46}$$

3.5.2 Input Constraints

The input constraints are taken to be

$$\underline{f_x} \leq f_x(k) \leq \overline{f_x} \tag{3.47}$$

$$\underline{f_y} \leq f_y(k) \leq \overline{f_y} \tag{3.48}$$

$$\underline{f_z} \leq f_z(k) \leq \overline{f_z} \tag{3.49}$$

$$\tag{3.50}$$

which can easily be expressed in the MPC constraint form as

$$\begin{bmatrix} I \\ -I \end{bmatrix} \mathbf{u}(k) \leq \begin{bmatrix} \mathbf{1} \otimes \begin{bmatrix} \overline{f_x} \\ \overline{f_y} \\ \overline{f_z} \end{bmatrix} \\ -\mathbf{1} \otimes \begin{bmatrix} \underline{f_x} \\ \underline{f_y} \\ \underline{f_z} \end{bmatrix} \end{bmatrix}. \tag{3.51}$$

3.5.3 State Constraints

Since the planes of the forbidden zone around the payload are not necessarily perpendicular to the planes of the coordinate system, we cannot simply create constraints based on the individual states. If we assume we are only concerned with the spacecraft's position relative to one plane of the forbidden zone, we must instead frame the problem in terms establishing a point's existence within a half-space. So as the spacecraft moves from waypoint to waypoint, we simply need to consider the spacecraft's current position relative to different half-spaces defined by these planes.

We begin the development of this strategy by again defining the equation of a plane as

$$\mathbf{n} \cdot (\mathbf{x} - \mathbf{x}_0) = 0 \tag{3.52}$$

where $\mathbf{n}$ is a vector pointing perpendicular to the plane and $\mathbf{x}$ and $\mathbf{x}_0$ are points inside the plane. We can define an open half-space as

$$\mathbf{n} \cdot (\mathbf{x} - \mathbf{x}_0) < 0. \tag{3.53}$$

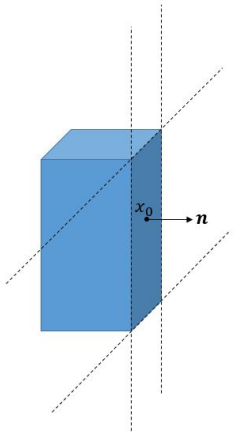

Figure 3.7: An example of a half-space defined by a plane of a forbidden zone.

If we substitute in a new point $\mathbf{x}_n$ for $\mathbf{x}$ and find that

$$\mathbf{n} \cdot (\mathbf{x}_n - \mathbf{x}_0) \geq 0 \tag{3.54}$$

then we can say that the point does not exist in the open half-space.

We can rewrite this constraint in the form required for MPC as

$$\mathbf{n} \cdot ((A_i x(k) + C_i u(k)) - \mathbf{x}_0) \geq 0, \ i = 1, \ldots, N \tag{3.55}$$

which can be rearranged into the standard form for a quadratic programming constraint as

$$-(\mathbf{n} \cdot C_i \mathbf{u}(k)) \leq -\mathbf{n} \cdot \mathbf{x}_0 + \mathbf{n} \cdot A_i x(k), \ i = 1, \ldots, N. \tag{3.56}$$

3.6 Path Planning Example

We can illustrate the tools developed in this chapter with a simple example. A single spacecraft, that we will treat as a point-mass, is trailing a large non-rotating cube in a circular orbit 600 km above the Earth. The goal is to take the spacecraft from its initial position relative to the cube and transport it safely to its attachment point on the far side of the cube. The cube is surrounded by a forbidden zone that the spacecraft is required not to cross until it makes its final straight-line approach to the attachment point.

Table 3.1: Vertices used in path planning example

Vertex	RSW Position (km)
Initial	(0,-1.5,0)
Approach	(-0.25,1,0.25)
Final	(-0.25,0.25,0.25)
Obstacle Vertex 1	(-0.5,-0.5,0.5)
Obstacle Vertex 2	(-0.5,0.5,0.5)
Obstacle Vertex 3	(0.5,0.5,0.5)
Obstacle Vertex 4	(0.5,-0.5,0.5)
Obstacle Vertex 5	(-0.5,-0.5,-0.5)
Obstacle Vertex 6	(-0.5,0.5,-0.5)
Obstacle Vertex 7	(0.5,0.5,-0.5)
Obstacle Vertex 8	(0.5,-0.5,-0.5)

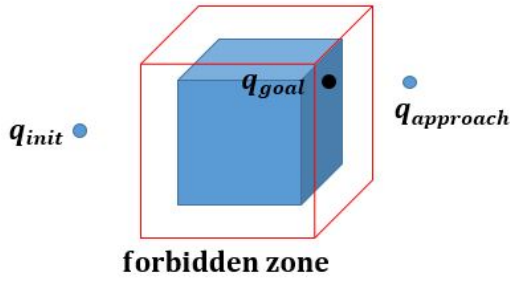

Figure 3.8: The initial conditions for the path planning example.

Table 3.2: Simulation parameters used in path planning example

Parameter	Value
Spacecraft mass	1,000 kg
Orbital altitude	600 km
Maximum thrust	1,000 N
Δt	3 s
Horizon	5 steps
Q weight for position states	1
Q weight for velocity states	0
R weight for inputs	1×10^{-6}
Tolerance for waypoint 1	0.5 km
Tolerance for waypoint 2	0.25 km
Tolerance for waypoint 3	0.1 km
Tolerance for waypoint 4	0.4 km

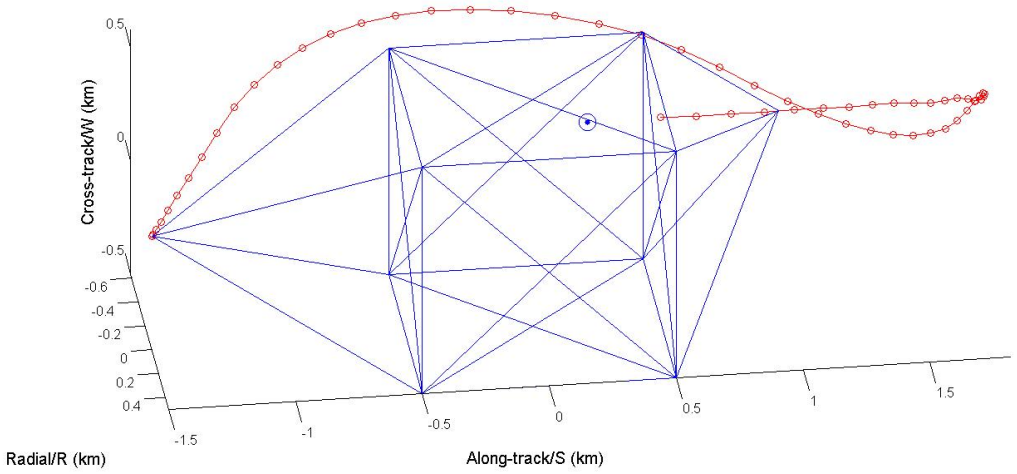

Figure 3.9: The visibility graph, chosen path, and trajectory for the path planning example.

An illustration of this problem is shown in Fig. 3.8. The simulation parameters are shown in Tables 3.1-3.2. We start by building the visibility graph roadmap. The forbidden zone surrounding the payload is a 0.5 km $\times$ 0.5 km $\times$ 0.5 km cube centered at the origin of the RSW coordinate frame. The spacecraft is initially trailing the payload by 1.5 km, or $[0, -1.5, 0]$ km in RSW coordinates. The spacecraft is to move to the approach coordinates at $[-0.25, 1, 0.25]$ km before making its final approach to its attachment point at $[-0.25, 0.25, 0.25]$ km. The generated visibility graph should then connect the vertices of the forbidden zone, initial coordinates, and approach coordinates with all possible paths that do not violate the forbidden zone. This expected result matches the actual graph, shown in blue path segments, in Fig. 3.9.

The A^* algorithm then correctly chooses the shortest path from the visibility graph that connects the initial and approach coordinates. The trajectory generated by MPC that links these waypoints is shown as the red path in Fig. 3.9. The individual states are plotted in Fig. 3.10. The control inputs are plotted in Fig. 3.11. The inputs are always bounded by a thrust limit of $1,000$N. This trajectory created by MPC was heavily influenced by the chosen values for the mass of the spacecraft, maximum allowable thrust, tolerance levels

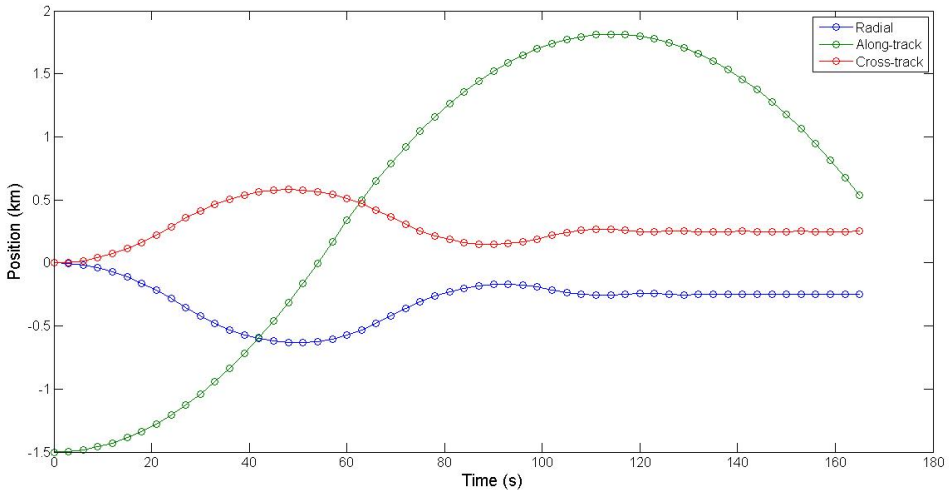

Figure 3.10: The spacecraft position in the RSW coordinates for the initial path planning example.

for when the spacecraft was deemed to be close enough to the waypoints, horizon length, and the value of the input weighting matrix R. These values are so important because the effect of the orbital mechanics on the spacecraft's motion relative to the payload becomes more pronounced as time passes. In other words, the spacecraft will be forced to follow increasingly erratic trajectories as the time required to reach a waypoint increases.

An example of an erratic trajectory is shown in Fig. 3.12. The maximum thrust is lowered to 100 N and the simulation ends when the spacecraft is within 100 m of the first waypoint. This erratic behavior can be overcome by increasing the horizon used in the MPC predictions. Unfortunately this greatly increases the computation time required to choose the inputs which makes including the state constraints impractical. An example of a trajectory created using a horizon of 50 steps, a maximum thrust of 100 N, and a position tolerance of 100 m for all waypoints is shown in Fig. 3.13.

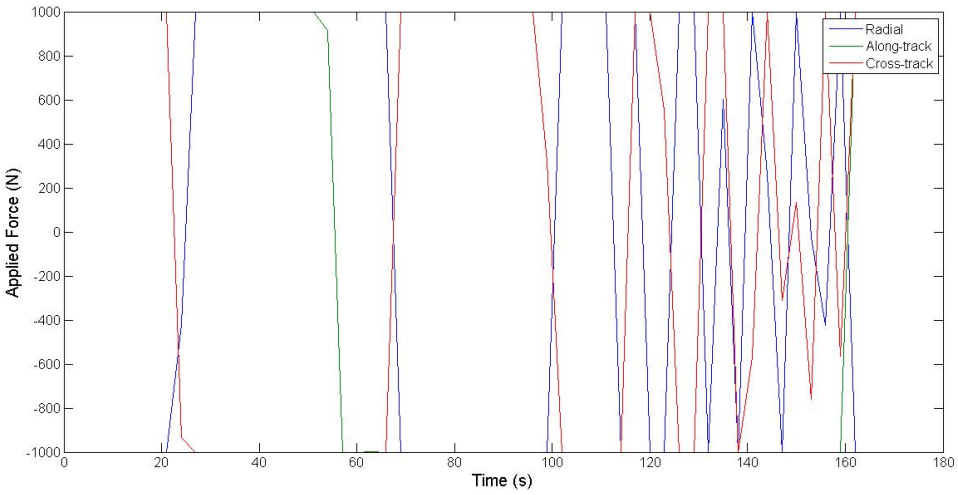

Figure 3.11: The applied force for the initial path planning example.

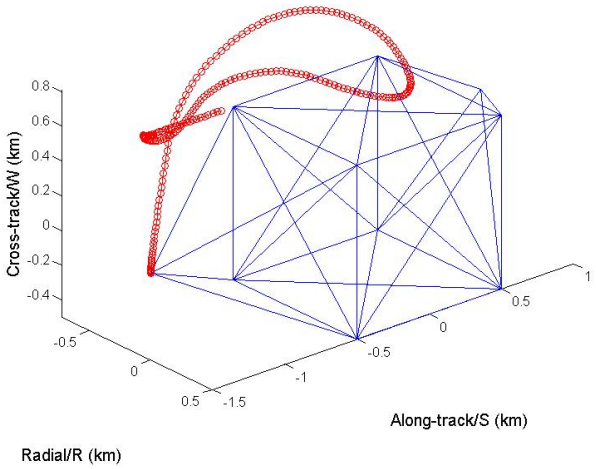

Figure 3.12: An example of an eratic trajectory caused by ortital mechanics.

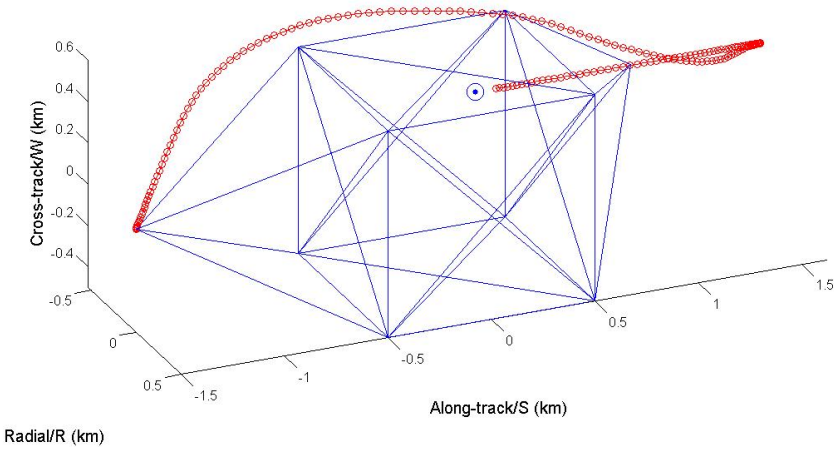

Figure 3.13: An example of an improved trajectory created by increasing the MPC horizon.

3.7 Chapter Summary

In this chapter we attempted to develop a strategy to safely and efficiently transfer a spacecraft from its initial position relative to the payload to its attachment point. The proposed method incorporated several elements from robotic path planning, geometric positioning, and model predictive control. Unfortunately, the strategy appears to be best suited for small spacecraft with powerful reaction control thrusters. It does not seem feasible to apply the proposed method to large spacecraft attempting to navigate with a point-to-point roadmap over long distances. But we will likely need to develop a new strategy if our goal is to have large spacecraft autonomously navigate around obstacles in low-Earth orbit. However, this method could still be practical if the spacecraft are in much higher orbits since the effect of the orbital mechanics on the spacecraft's relative motion will be much more gradual at these altitudes.

Chapter 4

COOPERATIVE VELOCITY CONTROL

The goal now is to investigate how the multiple spacecraft that are attached to the payload can cooperatively change its along-track velocity without inducing any undesired translational or rotational motion. The overall strategy will be to frame the relevant optimization problems in terms of tractable linear and quadratic programs. These programs can then be solved by existing solution algorithms.

Rather than solving a specific spacecraft trajectory optimization problem, we will focus on the general problem of increasing a payload's velocity in a single direction. We will not consider any gravitational or disturbing forces. We will also treat the relevant objects in the problem as point masses.

4.1 Gimbal Orientation

The complexity of the control optimization problem greatly increases if we allow the spacecraft to change the direction and magnitude of their thrust vectors simultaneously. We can instead optimize the direction and magnitude of the thrusters as separate problems. We begin by considering the direction of the thrust vector on each of the spacecraft.

4.1.1 Spherical Surface Approximation

Let us start with the reference frame of an individual spacecraft. In this frame, we allow the thrust vector to exist within a three-dimensional cone. If the spacecraft is operating at maximum thrust, then the thrust exists on a section of the surface of a sphere. In order to simplify future calculations, we will approximate this region as a constrained plane. Let the radius of the sphere be given by $\overline{r}$ and let the cone be oriented so that it is aligned with the

positive z axis. Then we can define the cone using the following equations and constraints

$$x = r\cos\theta\sin\phi \tag{4.1}$$

$$y = r\sin\theta\sin\phi \tag{4.2}$$

$$z = r\cos\phi \tag{4.3}$$

$$0 \leq r \leq \overline{r} \tag{4.4}$$

$$0 \leq \theta \leq 2\pi \tag{4.5}$$

$$0 \leq \phi \leq \overline{\phi} \tag{4.6}$$

where θ is the angle of rotation about the z axis and ϕ is the angle away from the positive z axis. Now if we only consider the square region contained within the top of the cone when $r = \overline{r}$, we have effectively limited the force vector to a set of two-dimensional coordinates. We can approximate this square region with the following equations and constraints

$$x = \overline{r}c_x \tag{4.7}$$

$$y = \overline{r}c_y \tag{4.8}$$

$$z = \overline{r} \tag{4.9}$$

$$-\overline{c}_x \leq c_x \leq \overline{c}_x \tag{4.10}$$

$$-\overline{c}_y \leq c_y \leq \overline{c}_y \tag{4.11}$$

Once the variables c_x and c_y are chosen by the optimization solver, the original spherical coordinates can be recovered with the following equations

$$\theta = \tan^{-1}\frac{c_y}{c_x} \tag{4.12}$$

$$\phi = \sqrt{c_x^2 + c_y^2} \tag{4.13}$$

We must now represent the thrust vector in the reference frame of the payload. We perform this transformation with the following equation

$$\mathbf{F}_p = \mathbf{C}\mathbf{F}_s \tag{4.14}$$

where $\mathbf{C}$ is a direction cosine matrix relating the spacecraft frame to the payload frame.

4.1.2 Minimization Problem Construction

In the next section we will consider the rotational transformation from the payload frame to the inertial frame, but for now we will assume the two frames are in the same orientation. Let a spacecraft's thrust vector in the payload frame be given by the following equations and constraints

$$\mathbf{F}_s = \mathbf{C} \begin{bmatrix} \overline{f} c_x \\ \overline{f} c_y \\ \overline{f} \end{bmatrix} \tag{4.15}$$

$$-\overline{c}_x \leq c_x \leq \overline{c}_x \tag{4.16}$$

$$-\overline{c}_y \leq c_y \leq \overline{c}_y \tag{4.17}$$

where $\overline{f}$ is the maximum thrust magnitude. Also let the vector $\mathbf{r}_s$ be the position of the thruster relative to the combined center of mass of the payload and all attached spacecraft. Then the torque $\boldsymbol{\tau}$ generated by $\mathbf{F}_s$ about the combined center of mass will be $\boldsymbol{\tau} = \mathbf{r}_s \times \mathbf{F}_s$. Now for n spacecraft, let the combined force and torque vectors be given by

$$\mathbf{F}_{net} = \sum_{i=1}^{n} \mathbf{F}_i \tag{4.18}$$

$$\boldsymbol{\tau}_{net} = \sum_{i=1}^{n} \boldsymbol{\tau}_i \tag{4.19}$$

Let us say that we want to minimize the net force in the x and y directions and all components of net torque. Later in this chapter it will be clear that consequences of payload rotation will be much worse than any drift in the x and y directions. So let us form the minimization problem in the following way.

$$\begin{aligned} \text{minimize} \quad & |\sum_{i=1}^{n} F_{x_i}| + |\sum_{i=1}^{n} F_{y_i}| \\ \text{subject to} \quad & |\boldsymbol{\tau}_{net}| \leq \begin{bmatrix} \overline{\tau} \\ \overline{\tau} \\ \overline{\tau} \end{bmatrix} \end{aligned}$$

If we let $\overline{\tau} = 0$ then we will be searching for the c_x and c_y for each spacecraft that result in zero net torque. If a solution does not exist, then $\overline{\tau}$ can gradually be increased until

a solution is found. We must now recast several portions of this minimization problem in order to solve it as a linear program.

4.1.3 Linear Program Formulation

The first issue we will consider is the absolute value in the objective function. It is possible to rewrite the absolute value operator as a linear function with linear constraints [16]. Let us start with the simple problem

$$\text{minimize} \quad |X| + Y$$

where X and Y are both linear combinations of the form $X = a_1x_1 + a_2x_2 + \cdots + a_nx_n$ for which the a terms are constant coefficients and the x, y terms are the variables that are to be optimized. Let us create a new variable $X' = |X|$ that is constrained by

$$X \leq X' \tag{4.20}$$

$$-X \leq X' \tag{4.21}$$

Then we can rewrite the original problem as

$$\begin{aligned} \text{minimize} \quad & X' + Y \\ \text{subject to} \quad & X \leq X' \\ & -X \leq X' \end{aligned}$$

In our case we are trying to minimize the sum of two absolute values, so we can repeat this process with the Y term as follows

$$\begin{aligned} \text{minimize} \quad & X' + Y' \\ \text{subject to} \quad & X \leq X' \\ & -X \leq X' \\ & Y \leq Y' \\ & -Y \leq Y' \end{aligned}$$

The constraints based on $\boldsymbol{\tau}_{net}$ can be handled in a similar way. We can rewrite the $\boldsymbol{\tau}_{net}$ constraint as

$$-\begin{bmatrix} \overline{\tau} \\ \overline{\tau} \\ \overline{\tau} \end{bmatrix} \leq \boldsymbol{\tau}_{net} \leq \begin{bmatrix} \overline{\tau} \\ \overline{\tau} \\ \overline{\tau} \end{bmatrix} \tag{4.22}$$

which is equivalent to

$$\begin{bmatrix} \boldsymbol{\tau}_{net} \\ -\boldsymbol{\tau}_{net} \end{bmatrix} \leq \begin{bmatrix} \overline{\tau} \\ \overline{\tau} \\ \overline{\tau} \\ \overline{\tau} \\ \overline{\tau} \\ \overline{\tau} \end{bmatrix} \tag{4.23}$$

Now we can combine the objective and constraints to form the complete linear program

$$\begin{aligned} \text{minimize} \quad & X' + Y' \\ \text{subject to} \quad & \begin{bmatrix} F_{\text{net}_x} \\ -F_{\text{net}_x} \\ F_{\text{net}_y} \\ -F_{\text{net}_y} \\ \boldsymbol{\tau}_{net} \\ -\boldsymbol{\tau}_{net} \\ c_{x_i},\ i = 1, \ldots, n \\ -c_{x_i},\ i = 1, \ldots, n \\ c_{y_i},\ i = 1, \ldots, n \\ -c_{y_i},\ i = 1, \ldots, n \end{bmatrix} \leq \begin{bmatrix} X' \\ X' \\ Y' \\ Y' \\ \overline{\tau} \\ \overline{\tau} \\ \overline{\tau} \\ \overline{\tau} \\ \overline{\tau} \\ \overline{\tau} \\ \overline{c}_x \\ \overline{c}_x \\ \overline{c}_y \\ \overline{c}_y \end{bmatrix} \end{aligned}$$

4.2 Thrust Magnitude Control

In this section we will consider how to control the thrust magnitude of the individual rockets during a given gimbal orientation. Each thrust vector $\mathbf{F}_i$ will have a fixed orientation so that its components can be defined in each spacecraft frame as

$$F_x = f \cos\theta \sin\phi \tag{4.24}$$

$$F_y = f \sin\theta \sin\phi \tag{4.25}$$

$$F_z = f \cos\phi \tag{4.26}$$

$$0 \leq f \leq \overline{f} \tag{4.27}$$

where θ and ψ are fixed and the thrust magnitude f is bounded by the maximum attainable thrust $\overline{f}$. We must now incorporate a basic model of chemical rocket propulsion. Let the equation of motion for a simple rocket be defined as

$$m\frac{d\mathbf{v}}{dt} = \mathbf{c}\frac{dm}{dt} \tag{4.28}$$

where m is the mass of the rocket, $\mathbf{v}$ is its velocity vector, $\mathbf{c}$ is the exit velocity vector of the thrust, and $\frac{dm}{dt}$ can be interpreted as the mass flow rate of the rocket [35]. If we temporarily assume the decrease in m due to the mass flow rate is negligible and $\mathbf{c}$ is held fixed, then we can write the effect of the rocket's thrust on the change in velocity as

$$\mathbf{F} = m\frac{d\mathbf{v}}{dt} = \frac{dm}{dt}\mathbf{c} \tag{4.29}$$

In this formulation we can see that $\frac{dm}{dt}$, or $\dot{m}$, will be the control variable and $\mathbf{c}$ will be in the direction of the chosen gimbal orientation. If we define the magnitude of the exit velocity as c then we can rewrite the thrust in the spacecraft frame as

$$F_x = c\dot{m} \cos\theta \sin\phi \tag{4.30}$$

$$F_y = c\dot{m} \sin\theta \sin\phi \tag{4.31}$$

$$F_z = c\dot{m} \cos\phi \tag{4.32}$$

$$0 \leq \dot{m} \leq \overline{\dot{m}} \tag{4.33}$$

$$m_{\text{prop}} \geq 0 \tag{4.34}$$

where m_{prop} is the mass of propellant remaining in the spacecraft. Next we again use the direction cosine matrix $\mathbf{C}$ to transform $\mathbf{F}$ into the frame of the payload so that $\mathbf{F}_p = \mathbf{C}\mathbf{F}_s$. Now consider the location $\mathbf{r}$ of the thruster relative to the center of mass of the payload in the frame of the payload. We can calculate the location of the combined center of mass $\mathbf{x}_0$ of the payload and all n spacecraft, relative to the center of mass of the payload, with the equation

$$\mathbf{x}_0 = \frac{\sum_{i=1}^{n} m_i \mathbf{r}_i}{\sum_{i=1}^{n} m_i} \tag{4.35}$$

We now create a new reference frame that is centered at $\mathbf{x}_0$ and in the same orientation as the payload frame. This frame will translate relative to the origin of the payload frame as propellant is burned by the individual rockets. Let the position of the thrusters in this new frame be defined as $\mathbf{r}_{CM} = \mathbf{r} - \mathbf{x}_0$. The torque generated in the CM frame will then be $\boldsymbol{\tau}_{CM} = \mathbf{r}_{CM} \times \mathbf{F}$. If we treat the spacecraft and payload as point masses, can calculate the moments and products of inertia in this frame as [27]

$$I_{xx} = ((-x_{0_y})^2 + (-x_{0_z})^2)m_p + \sum_{i=1}^{n}(r_{CM_{i_y}}^2 + r_{CM_{i_z}}^2)m_i \tag{4.36}$$

$$I_{yy} = ((-x_{0_x})^2 + (-x_{0_z})^2)m_p + \sum_{i=1}^{n}(r_{CM_{i_x}}^2 + r_{CM_{i_z}}^2)m_i \tag{4.37}$$

$$I_{xx} = ((-x_{0_x})^2 + (-x_{0_y})^2)m_p + \sum_{i=1}^{n}(r_{CM_{i_x}}^2 + r_{CM_{i_y}}^2)m_i \tag{4.38}$$

$$I_{xy} = I_{yx} = -x_{0_x}x_{0_y}m_p - \sum_{i=1}^{n} r_{CM_{i_x}} r_{CM_{i_y}} m_i \tag{4.39}$$

$$I_{yz} = I_{yx} = -x_{0_y}x_{0_z}m_p - \sum_{i=1}^{n} r_{CM_{i_y}} r_{CM_{i_z}} m_i \tag{4.40}$$

$$I_{xz} = I_{yx} = -x_{0_x}x_{0_z}m_p - \sum_{i=1}^{n} r_{CM_{i_x}} r_{CM_{i_z}} m_i \tag{4.41}$$

which are arranged into the inertia tensor $\mathbf{J}$ as

$$\mathbf{J} = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}. \tag{4.42}$$

We can now write the rotational equations of motion in the CM frame as

$$\mathbf{J}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\mathbf{J}\boldsymbol{\omega}) = \boldsymbol{\tau}_{\text{net}} \tag{4.43}$$

Now if switch to PQR notation so that $P = \omega_x, Q = \omega_y$ and $R = \omega_z$, we can expand the previous equation and find that

$$I_{xx}\dot{P} - (I_{yy} - I_{zz})QR - I_{xy}(\dot{Q} - PR) - I_{yz}(Q^2 - R^2) - I_{xz}(\dot{R} + PQ) = \tau_{\text{net}_x} \tag{4.44}$$

$$I_{yy}\dot{Q} - (I_{xx} - I_{zz})PR - I_{xy}(\dot{P} + QR) - I_{yz}(\dot{R} - PQ) + I_{xz}(P^2 - R^2) = \tau_{\text{net}_y} \tag{4.45}$$

$$I_{zz}\dot{R} - (I_{yy} - I_{xx})PQ + I_{xy}(Q^2 - P^2) - I_{yz}(\dot{Q} + PR) - I_{xz}(\dot{P} - QR) = \tau_{\text{net}_z} \tag{4.46}$$

Since we will be aiming to keep P, Q and R as close to zero as possible, we can neglect terms that are products of the angular rates so that the system is approximately linear. This results in the modified system of equations

$$I_{xx}\dot{P} - I_{xy}\dot{Q} - I_{xz}\dot{R} \approx \tau_{\text{net}_x} \tag{4.47}$$

$$I_{yy}\dot{Q} - I_{xy}\dot{P} - I_{yz}\dot{R} \approx \tau_{\text{net}_y} \tag{4.48}$$

$$I_{zz}\dot{R} - I_{yz}\dot{Q} - I_{xz}\dot{P} \approx \tau_{\text{net}_z} \tag{4.49}$$

which we can rewrite in terms of the angular rates as

$$\begin{bmatrix} \dot{P} \\ \dot{Q} \\ \dot{R} \end{bmatrix} \approx \mathbf{J}^{-1} \begin{bmatrix} \tau_{\text{net}_x} \\ \tau_{\text{net}_y} \\ \tau_{\text{net}_z} \end{bmatrix} \tag{4.50}$$

We now need a second rotation matrix $\mathbf{D}$ to express the dynamics in the inertial frame. Let this matrix be the 321 Euler rotation matrix defined by

$$\mathbf{D} = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix} \tag{4.51}$$

The Euler-angle rates are governed by the equations [27]

$$\dot{\phi} = P + Q\sin\phi\tan\theta + R\cos\phi\tan\theta \tag{4.52}$$

$$\dot{\theta} = Q\cos\phi - R\sin\phi \tag{4.53}$$

$$\dot{\psi} = (Q\sin\phi + R\cos\phi)\sec\theta \tag{4.54}$$

If we again assume the angles and angular rates are kept very close to zero, we can approximate the Euler-angle rates as

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \approx \begin{bmatrix} P \\ Q \\ R \end{bmatrix} \tag{4.55}$$

And similarly we can write the translational equations of motion as

$$\begin{bmatrix} \dot{X}_I \\ \dot{Y}_I \\ \dot{Z}_I \end{bmatrix} = \mathbf{D} \begin{bmatrix} U \\ V \\ W \end{bmatrix} \tag{4.56}$$

$$= \begin{bmatrix} U\cos\theta\cos\psi + V(\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi) + W(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi) \\ U\cos\theta\sin\psi + V(\sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi) + W(\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi) \\ U\sin\theta - V\sin\phi\cos\theta - W\cos\phi\cos\theta \end{bmatrix} \tag{4.57}$$

where if we assume small angles we can approximate the translational motion as

$$\begin{bmatrix} \dot{X}_I \\ \dot{Y}_I \\ \dot{Z}_I \end{bmatrix} \approx \begin{bmatrix} U \\ V \\ W \end{bmatrix} \tag{4.58}$$

Now we can begin to assemble the linear model of the entire cooperative propulsion system. We start by considering an approximation of the dynamics for the continuous-time system. In the next section we will address the effect of the shifting center of mass, but for now we will assume total mass and location of the combined center of mass does are constant as the propellant is burned. Under this assumption, the dynamics and physical constraints of

the system will be

$$\dot{\mathbf{X}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} \dot{X}_I \\ \dot{Y}_I \\ \dot{Z}_I \\ \phi \\ \theta \\ \psi \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ m_1 \\ \vdots \\ m_n \end{bmatrix} + \begin{bmatrix} \frac{1}{m_{\text{net}}}\mathbf{c}_1 & \cdots & \frac{1}{m_{\text{net}}}\mathbf{c}_n \\ 0 & \cdots & 0 \\ 0 & \cdots & 0 \\ 0 & \cdots & 0 \\ \mathbf{J}^{-1}[\mathbf{r}_1]_\times \mathbf{c}_1 & \cdots & \mathbf{J}^{-1}[\mathbf{r}_n]_\times \mathbf{c}_n \\ -1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & -1 \end{bmatrix} \begin{bmatrix} \dot{m}_1 \\ \vdots \\ \dot{m}_n \end{bmatrix} \tag{4.59}$$

$$m_1, \cdots, m_n \geq 0$$
$$0 \leq \dot{m}_1, \cdots, \dot{m}_n \leq \overline{\dot{m}}$$

The constraints describing the desired behavior of the system are given by

$$|\dot{X}_I(t)| \leq \overline{\dot{X}}_I$$
$$|\dot{Y}_I(t)| \leq \overline{\dot{Y}}_I$$
$$|\phi(t)| \leq \overline{\phi}$$
$$|\theta(t)| \leq \overline{\theta}$$
$$|\psi(t)| \leq \overline{\psi}$$
$$|\dot{\phi}(t)| \leq \overline{\dot{\phi}}$$
$$|\dot{\theta}(t)| \leq \overline{\dot{\theta}}$$
$$|\dot{\psi}(t)| \leq \overline{\dot{\psi}}$$

And the desired final velocity in the $\hat{\mathbf{z}}$ direction is given by

$$\dot{Z}_I(t_f) = \dot{Z}_I(t_i) + \Delta V_z \tag{4.60}$$

The approximated linear dynamics developed in this section will now be incorporated into a model predictive control strategy. The chosen control inputs will be fed into the nonlinear dynamics which will be simulated in a differential equation solver such as MATLAB's ODE45.

4.3 Cooperative Velocity Control Strategy

We start with the solution to the simple rocket equation given in one dimension as

$$m\frac{dv}{dt} = -c\frac{dm}{dt}. \tag{4.61}$$

If we assume $\frac{dm}{dt}$ is held constant, then the solution to this equation will be

$$\Delta v = c\log\frac{m_0}{m_f}. \tag{4.62}$$

We can rewrite this equation as a function of the mass of propellant as

$$\Delta v(m_p) = c\log\frac{m_0}{m_0 - m_p} \tag{4.63}$$

If we plot this equation with $c = 1$ m/s and $m_0 = 1$ kg we see that the evolution of the velocity resembles the graph shown in Fig. 4.1. The shape of this graph represents the change in one-dimensional velocity for a chemical rocket burning 90% of its initial mass at a constant rate. Basically it is showing that the effect of burning the same amount of propellant on the increase in velocity is more pronounced when the total mass of the system is less. Our aim is to model this behavior as a linear system, so we will have to consider one segment of the burn at a time. For example, we could model the burn in 10% increments as shown in Fig. 4.2. So the overall control strategy will be to gimbal the thrusters to maximize the acceleration in the $\hat{\mathbf{z}}$ direction at the beginning of each of these linear intervals and then use thrust magnitude control on the individual rockets during the interval to achieve the desired acceleration. We have already developed a method to orient the gimbals so we only need to construct the thrust magnitude control as a model predictive control problem.

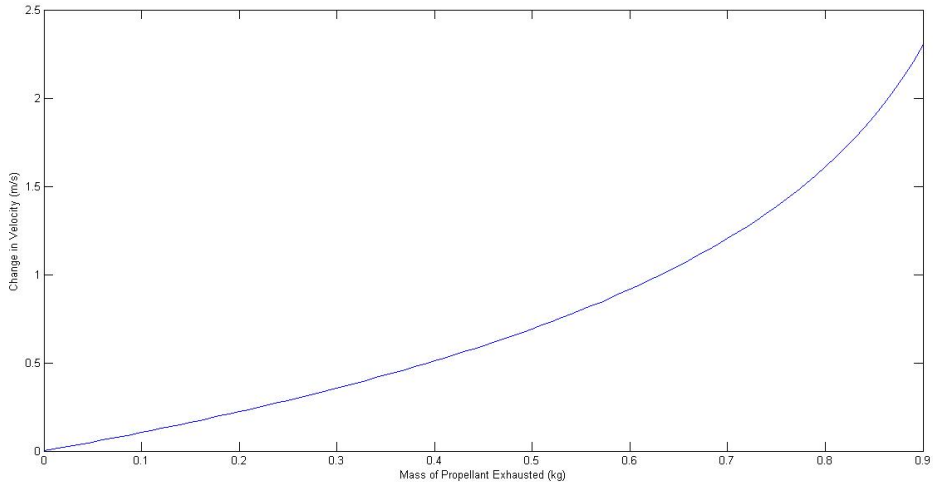

Figure 4.1: Change in velocity as a function of propellant expended.

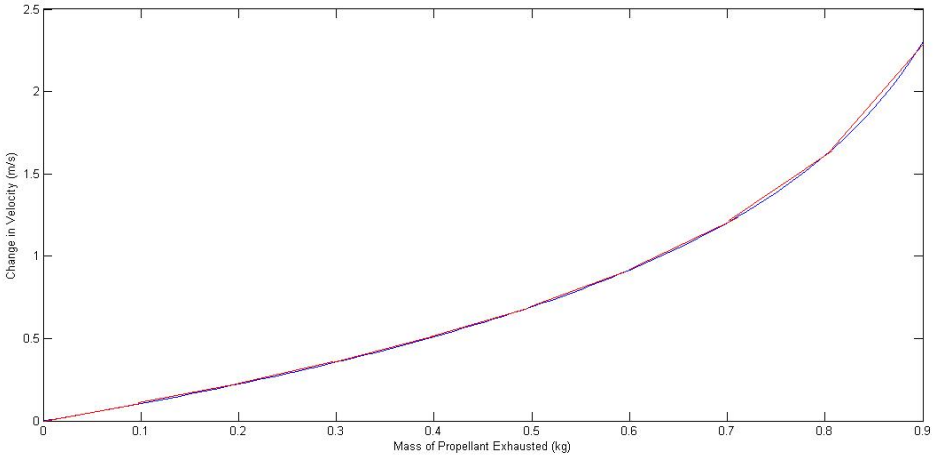

Figure 4.2: Change in velocity broken into ten linear segments.

In the last chapter we showed how to find the discrete-time representation of a continuous linear state-space system. We can use the **A** and **B** matrices from (4.59) to solve for the discrete dynamics. The linear constraints can be found using the following relationships based on the constraint equations developed in the previous section. Starting with the input constraints we have

$$\begin{bmatrix} I \\ -I \end{bmatrix} \begin{bmatrix} \dot{m}_1 \\ \vdots \\ \dot{m}_n \end{bmatrix} \leq \begin{bmatrix} \mathbf{1} \otimes \begin{bmatrix} \overline{\dot{m}} \\ \vdots \\ \overline{\dot{m}} \end{bmatrix} \\ -\mathbf{1} \otimes \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \end{bmatrix}. \tag{4.64}$$

The inequality constraints on the states can be written as

$$C_i \mathbf{u}(k) \leq \overline{x} - A^i x(k),\ i = 1, \ldots, N. \tag{4.65}$$

$$-C_i \mathbf{u}(k) \leq -\underline{x} + A^i x(k),\ i = 1, \ldots, N. \tag{4.66}$$

4.4 Example

We will illustrate the cooperative velocity control strategy with the following example. A new space station has been built in low-Earth orbit and the customer wishes to place the station into a higher altitude orbit. This requires a velocity increase of 1,000 m/s. The propulsion maneuver will be accomplished with eight spacecraft. In this example, the station and spacecraft will all be represted by point masses. The station will have a mass similar to the International Space Station and the spacecraft are modeled after the SIV-B rocket that was used as the upper stage of the Saturn V. The spacecraft are positioned at various distances and orientations relative to the payload as shown in Fig. 4.3. The position of the center of mass relative to the other point masses will change as the propellant is burned but the orientation of the coordinate system fixed at the center of mass will always be aligned in its initial orientation. All external forces will be ignored. The goal is to investigate whether it is possible for the spacecraft to perform the maneuver under ideal conditions.

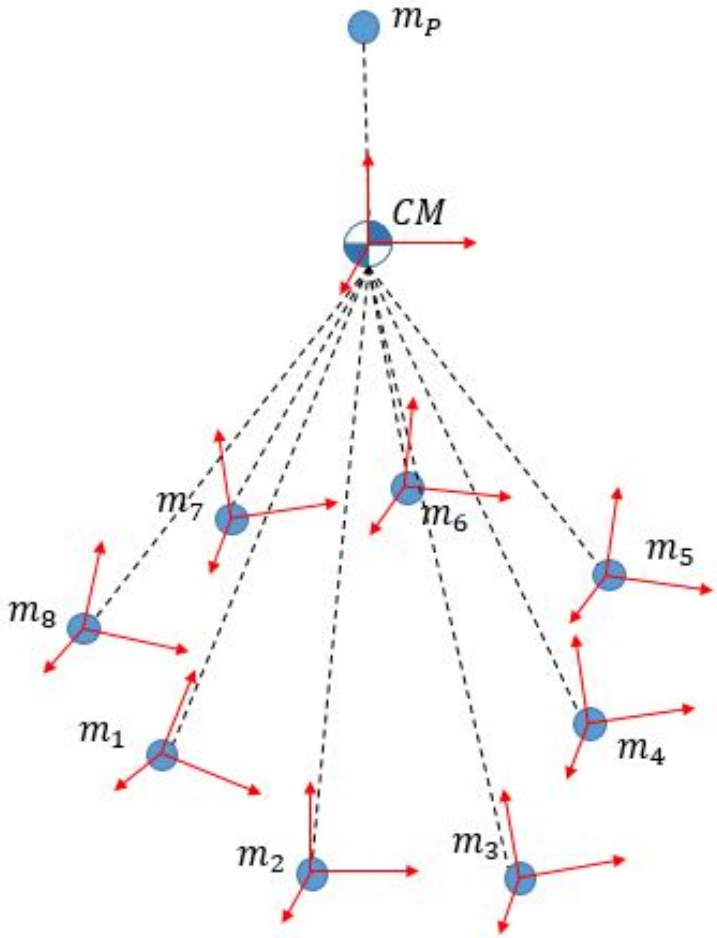

Figure 4.3: An illustration of the cooperative velocity control example. The positions are not to scale.

Table 4.1: Simulation physical parameters

Object	Initial Mass (kg)	X,Y,Z Position (m)	123 Euler Angle Orientation (degree)
Payload	500,000	(0,0,50)	(0,0,0)
Spacecraft 1	100,000	(0,-20,2)	(15,0,0)
Spacecraft 2	100,000	(16,-14,-4)	(8,3,0)
Spacecraft 3	100,000	(20,3,0)	(0,36,0)
Spacecraft 4	100,000	(14,14,3)	(10,20,0)
Spacecraft 5	100,000	(0,20,-1)	(-9,0,0)
Spacecraft 6	100,000	(-15,16,-6)	(-5,-15,0)
Spacecraft 7	100,000	(-19,0,5)	(0,-25,0)
Spacecraft 8	100,000	(-17,-16,2)	(-20,-10,0)

The physical parameters for this example are shown in Table 4.1 and the constraints are given in Table 4.2.

4.4.1 Thrust Magnitude Control

In the first example the eight spacecraft are required to execute the cooperative propulsion maneuver without optimizing their gimbal orientations. So the configuration described in Table 4.1 is used for the duration of the maneuver. This configuration includes the location of the combined center of mass that is used in the MPC algorithm. The simulation lasted 1,250 seconds. The translational velocities of the combined center of mass are shown in Figs. 4.4-4.5.

Clearly the translational velocities are uncontrollable under the current configuration. The system accelerates gradually for about the first 100 seconds but then falls into a tumble that cannot be recovered with the throttled thrusters in their current orientations. This instability could be overcome by increasing the horizon for the MPC optimization. However this greatly increases the computation time required to run the simulations.

Table 4.2: Desired behavior and constraints

Parameter	Value
Desired $\dot{Z}_I$	1,000 m/s
Maximum $\lvert\dot{X}_I\rvert$	1 m/s
Maximum $\lvert\dot{Y}_I\rvert$	1 m/s
Maximum $\lvert\phi\rvert$	5°
Maximum $\lvert\theta\rvert$	5°
Maximum $\lvert\psi\rvert$	5°
Maximum $\lvert\dot{\phi}\rvert$	1°/s
Maximum $\lvert\dot{\theta}\rvert$	1°/s
Maximum $\lvert\dot{\psi}\rvert$	1°/s
Minimum $\lvert\dot{m}\rvert$	0 kg/s
Maximum $\lvert\dot{m}\rvert$	255 kg/s
Maximum $\lvert\phi_g\rvert$ (gimbal)	7°
Minimum m (spacecraft mass)	10,000 kg
v_e (exit velocity)	4,130 m/s
Δt	0.1 s
Horizon	10 steps
Q weight for $\dot{Z}_I$	1
Q weight for all other states	0
R weight for inputs	1

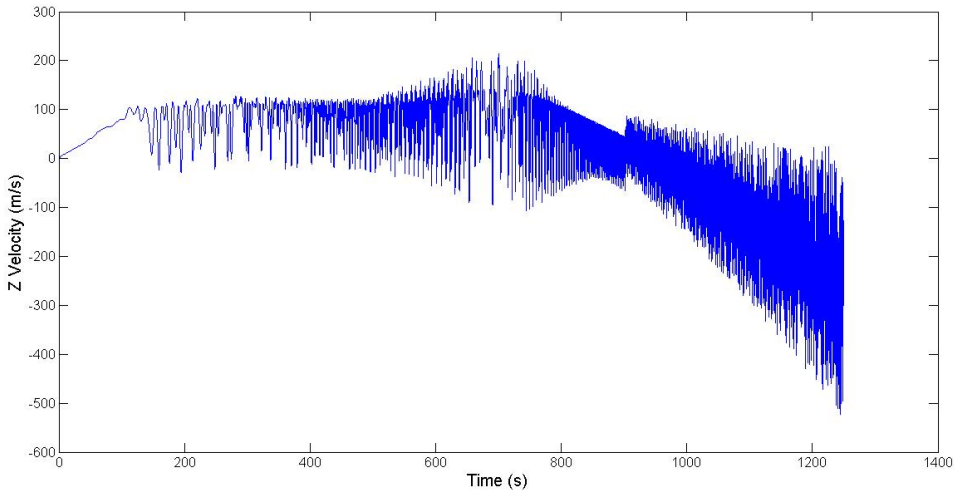

Figure 4.4: The z-axis velocity during thrust magnitude control.

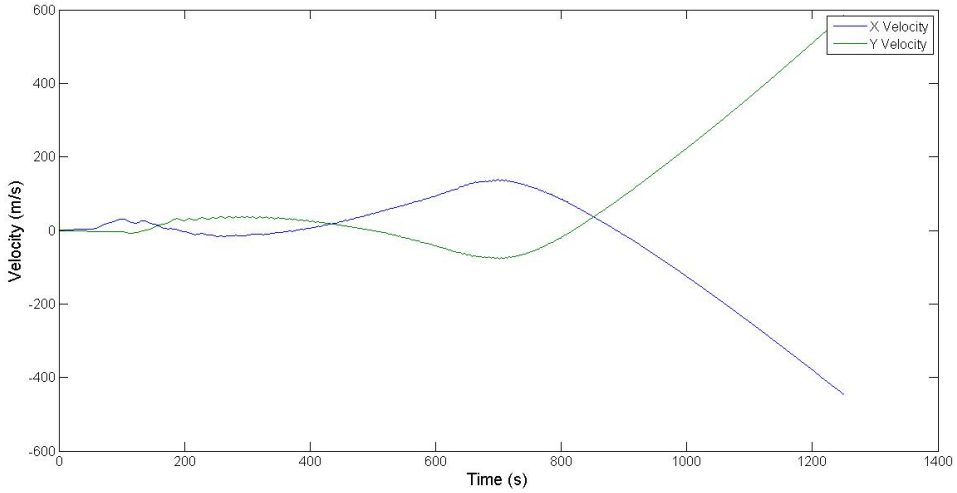

Figure 4.5: The x-axis and y-axis velocities during thrust magnitude control.

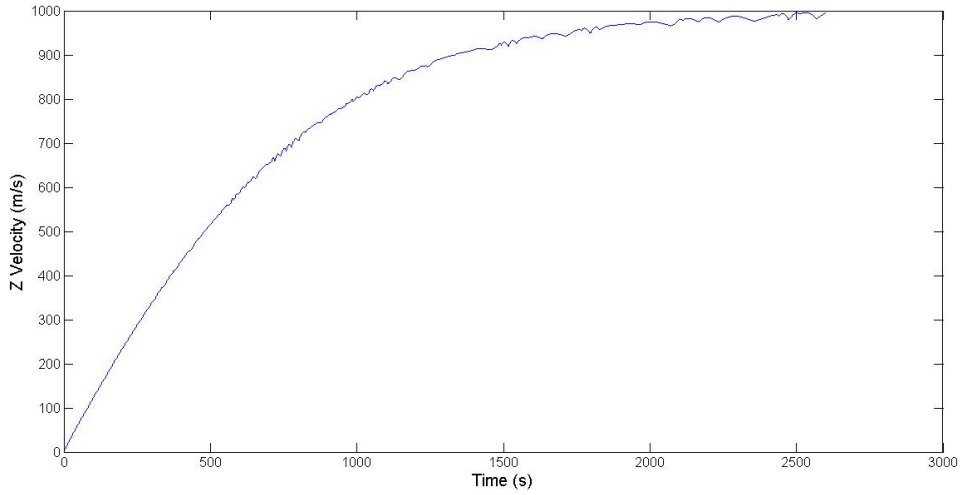

Figure 4.6: The z-axis velocity with an initial gimbal optimization and thrust magnitude control.

4.4.2 Thrust Magnitude Control with Initial Gimbal Optimization

We see much more desirable behavior in the next example. This time the spacecraft are allowed to optimize their gimbal orientations before the thrust magnitude control begins. These new orientations are used for the duration of the simulation.

We see that the eight spacecraft successfully increased the z-axis velocity of the payload by 994 m/s in about 2,600 seconds. Additionally the x-axis and y-axis velocities were kept within their ± 1 m/s limits for almost the entirety of the simulation. The true strength of MPC can be seen in the control of the Euler angles. In Figs. 4.8-4.9 we see that the deviation of the Euler angles and Euler angle rates are kept very close to zero throughout the simulation.

The decreases in propellant mass for all eight spacecraft are shown in Fig. 4.10. The mass flow rate for the first spacecraft is shown in Fig. 4.11. We can see that mass flow rate follows an overall trend of decreasing towards zero but experiences frequent spikes toward the maximum and minimum flow rates as the multiple spacecraft compensate for translational and rotational drift. The other seven spacecraft follow a similar pattern.

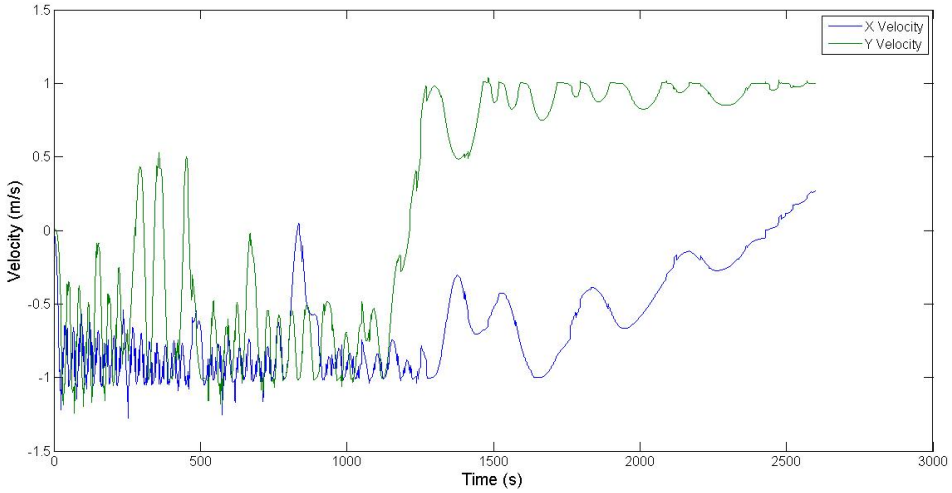

Figure 4.7: The x-axis and y-axis velocities with an initial gimbal optimization and thrust magnitude control.

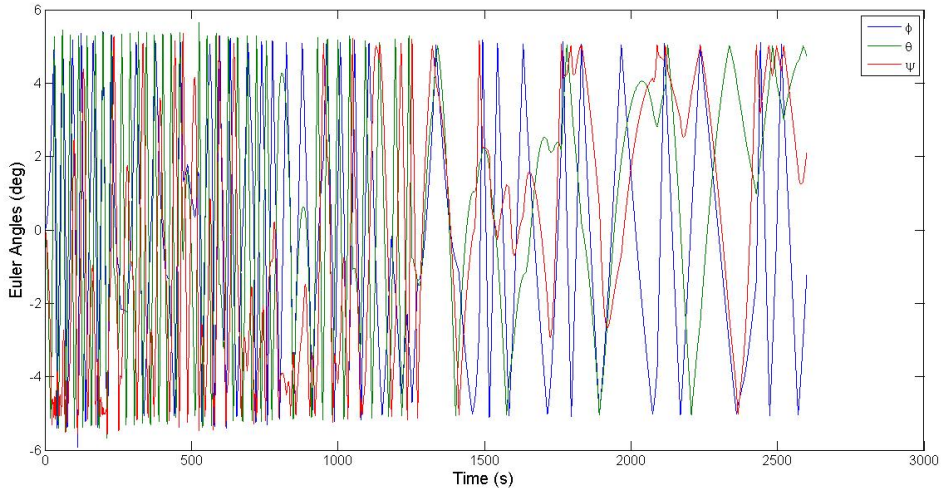

Figure 4.8: The Euler angles with an initial gimbal optimization and thrust magnitude control.

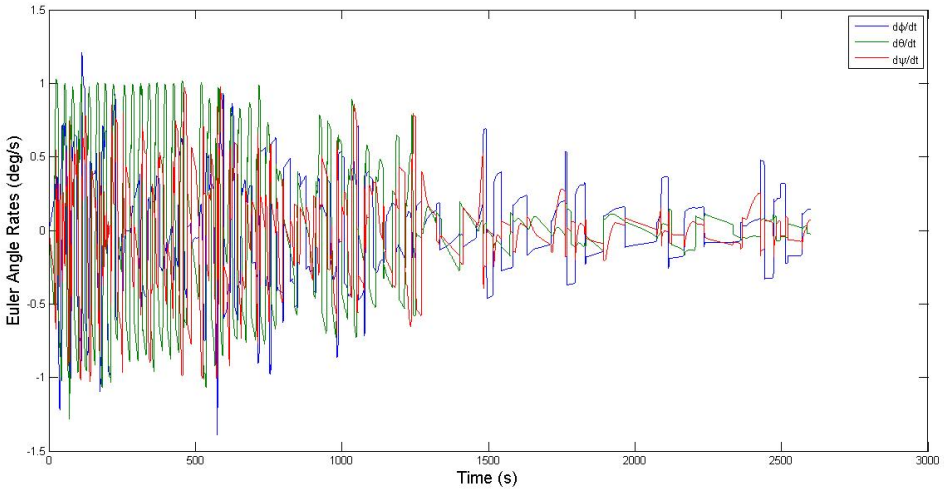

Figure 4.9: The Euler angle rates with an initial gimbal optimization and thrust magnitude control.

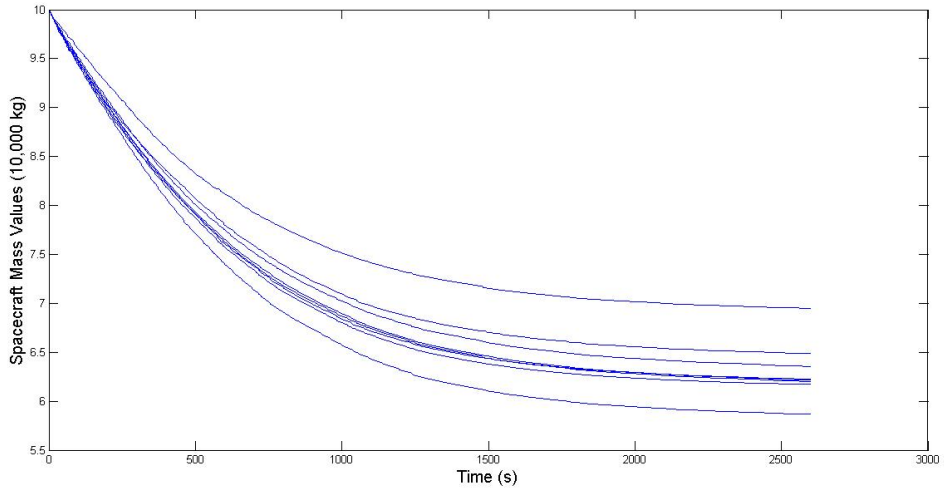

Figure 4.10: The spacecraft mass values with an initial gimbal optimization and thrust magnitude control.

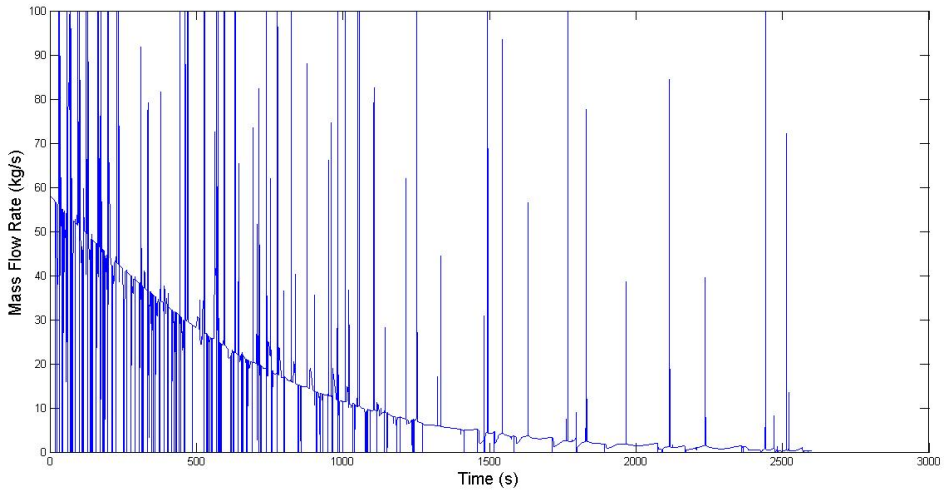

Figure 4.11: The mass flow rate for a spacecraft with an initial gimbal optimization and thrust magnitude control.

4.4.3 Thrust Magnitude Control with Ten Gimbal Optimizations

The performance displayed in the previous example can be improved by reorienting the gimbal angles of the spacecraft and updating the MPC state-space model throughout the maneuver. In the following example we perform these updates whenever the payload's z-axis velocity passes 100 m/s benchmarks. So the algorithm performs ten updates, including the gimbal optimization at the beginning of the simulation.

We can see that the additional gimbal optimizations and model updates allow the payload to reach the z-axis velocity goal slightly faster. In this example the payload reaches 996 m/s in about 2,320 seconds, compared to 2,600 seconds for the previous example. Additionally the y-axis velocity is kept closer to zero.

4.5 Chapter Summary

The work in this chapter has shown that constrained model predictive control is an effective tool for the cooperative velocity control problem. The combination of gimbal reorientation and thruster throttling allowed a group of poorly aligned spacecraft to control the trans-

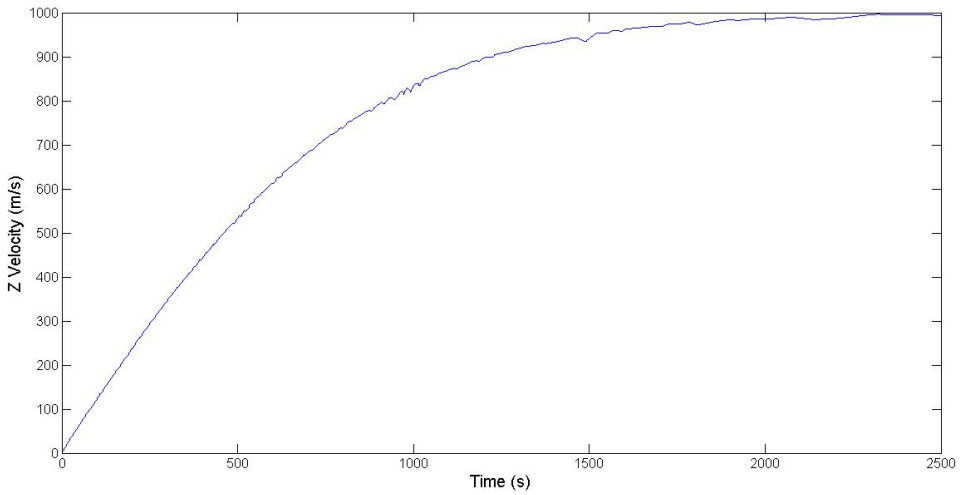

Figure 4.12: The z-axis velocity with ten gimbal optimizations and thrust magnitude control.

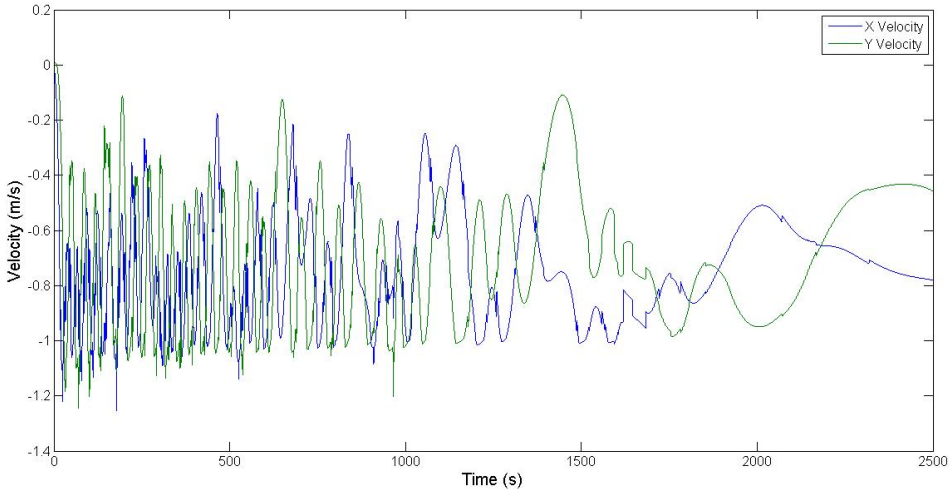

Figure 4.13: The x-axis and y-axis velocities with ten gimbal optimizations and thrust magnitude control.

lational velocity of a shared payload. It would be worth exploring whether more frequent gimbal optimization, perhaps on the order of once every second, would significantly improve the performance of the system.

Chapter 5

CONCLUSIONS AND FUTURE WORK

There is still a great deal of work to be done in order to incorporate more realism into the simulations we presented. In terms of modeling considerations, we will need to examine the loading conditions experienced within the payload so that the spacecraft do not tear it apart during the propulsion maneuver. Potentially we could do this with a finite element model of the payload so that the loads within the finite elements can be worked into constraints for the model predictive control algorithm. The physical limitations of the gimbaled rocket thrusters will also have to be addressed. The instantaneous transitions from zero mass flow rate to maximum thrust are obviously not realistic. But again, it should be possible to form these limitations as constraints for MPC. Along the same lines, it will be important to keep track of the temperature of the rocket nozzles so that they do not overheat. We should also attempt to model the power consumed by the spacecraft since re-orienting the large rocket nozzles will require a significant amount of electricity.

In terms of path planning, a cell decomposition or potential field method may offer better performance [8],[15],[25],[1],[12],[30],[17]. Since the straight line path segments do not appear to be feasible for low altitude orbits, it may also be prudent to include the orbital dynamics in the path segment construction process. And finally we suggest future work be dedicated to extending some of the existing two dimensional cooperative robotics algorithms to three dimensions and incorporating them into the cooperative velocity control solution [23],[20],[13].

The original goal of this thesis was to examine the feasibility of the core elements needed to control the translational velocity of a shared payload with multiple cooperative spacecraft. We showed how these core elements, while not perfect, could fit into a solution to that problem. With more research, the ultimate consequences of this proposal could be far-reaching. The cooperative velocity control strategy has the potential to radically increase

the scale of objects that can be built and moved in space.

BIBLIOGRAPHY

[1] K. Al-Sultan and M. Aliyu, "A new potential field-based algorithm for path planning," *Journal of Intelligent and Robotic Systems*, vol. 17, no. 3, pp. 265–282, 1996.

[2] BusinessInsider, "Apollo Spacecraft Diagram." [Online]. Available: http://static.businessinsider.com/image/516ed8a36bb3f73a54000011/image.jpg

[3] M. Cannon, "C21 Model Predictive Control," University of Oxford, Oxford, Tech. Rep., 2015.

[4] K. Chen and C. Rowley, "Fluid flow control applications of H2 optimal actuator and sensor placement," in *2014 American Control Conference*, Portland, 2014, pp. 4044–4049.

[5] R. DeCarlo, *Linear Systems: A State Variable Approach with Numerical Implementation.* Upper Saddle River: Prentice-Hall, 1989.

[6] L. DiGirolamo, A. Hoskins, K. Hacker, and D. Spencer, "A Hybrid Motion Planning Algorithm for Safe and Efficient Close Proximity, Autonomous Spacecraft Missions," in *AIAA/AAS Astrodynamics Specialist Conference*, 2014, pp. 1–19.

[7] D. Ding, Y.-H. Liu, and M. Wang, "Automatic selection of fixturing surfaces and fixturing points for polyhedral workpieces," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 2, Maui, 2001, pp. 1147–1152.

[8] J. C. Elizondo-Leal, E. F. Parra-Gonzalez, and J. G. Ramirez-Torres, "The Exact Euclidean Distance Transform: A New Algorithm for Universal Path Planning," *International Journal of Advanced Robotic Systems*, vol. 10, no. 266, pp. 1–10, 2013.

[9] ESA, "ATV Docked with ISS." [Online]. Available: http://www.esa.int/Our_Activities/Operations/

[10] J. Esposito, M. Feemster, and E. Smith, "Cooperative Manipulation on the Water Using a Swarm of Autonomous Tugboats," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on.* Pasadena, CA: IEEE, 2008, pp. 1501–1506.

[11] J. Helferty, E. Hoffman, G. McLennan, and W. Higgins, "Three-dimensional path planning for virtual bronchoscopy," *IEEE Transactions on Medical Imaging*, no. 11, pp. 1365–79, 2004.

[12] Y. L. Hu and Q. S. Zhang, "Multi-Robots Path Planning Based on Improved Artificial Potential Field Method," *Advanced Materials Research*, vol. 562, pp. 937–940, 2012.

[13] C. R. Kube and E. Bonabeau, "Cooperative transport by ants and robots," *Robotics and Autonomous Systems*, vol. 30, no. 1-2, pp. 85–101, 2000.

[14] J.-C. Latombe, *Robot Motion Planning*, 5th ed. Norwell: Kluwer Academic Publishers, 1991.

[15] F. Lingelbach, "Path planning using probabilistic cell decomposition," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 1, 2004, pp. 467–472.

[16] LPSolve, "Absolute Values." [Online]. Available: http://lpsolve.sourceforge.net/5.1/absolute.htm

[17] E. Masehian and M. Amin-Naseri, "A voronoi diagram-visibility graph-potential field compound algorithm for robot path planning," *Journal of Robotic Systems*, vol. 21, no. 6, pp. 275–300, 2004.

[18] NASA, "Apollo 13," 2009. [Online]. Available: http://www.nasa.gov/mission_pages/apollo/missions/apollo13.html#.VO-xvvnF_Rs

[19] U. of Edinburgh, "The Jordan Curve Theorem for Polygons." [Online]. Available: http://www.maths.ed.ac.uk/ aar/jordan/cr.pdf

[20] T. Otani and M. Koshino, "Applying a path planner based on RRT to cooperative multirobot box-pushing," *Artificial Life and Robotics*, vol. 13, no. 2, pp. 418–422, 2009.

[21] I. Oz, H. R. Topcuoglu, and M. Ermis, "A meta-heuristic based three-dimensional path planning environment for unmanned aerial vehicles," *Simulation*, no. 89, pp. 903–920, 2013.

[22] J. Park, D. Kim, Y. Yoon, H. Kim, and K. Yi, "Obstacle avoidance of autonomous vehicles based on model predictive control," in *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 223, 2009, pp. 1499–1516.

[23] E. F. Parra-Gonzalez, G. Ramirez-Torres, and G. Toscano-Pulido, "Motion Planning for Cooperative Multi-robot Box-Pushing Problem," in *Advances in Artificial Intelligence IBERAMIA 2008.* Springer Berlin Heidelberg, 2008, pp. 382–391.

[24] P. Plait, "Space station gives physics a boost," 2011. [Online]. Available: http://blogs.discovermagazine.com/badastronomy/2011/11/10/space-station-gives-physics-a-boost/#.VO-yWfnF_Rs

[25] J. Rossel and P. Iniguez, "Path planning using Harmonic Functions and Probabilistic Cell Decomposition," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 2005, pp. 1303–1308.

[26] C. Schafer and S. Ulbrich, "Optimal Actuator Placement for Dynamical Systems," in *84th Annual Meeting of the International Association of Applied Mathematics and Mechanics*, vol. 13. Novi Sad: PAMM, 2013, pp. 469–470.

[27] D. K. Schmidt, *Modern Flight Dynamics*, 1st ed. New York: McGraw-Hill Higher Education, 2011. [Online]. Available: http://books.google.com/books?id=TofEXwAACAAJ

[28] J. Schwartz and M. Sharir, "On the Piano Movers' Problem I. The Case of a Two-dimensional Rigid Polygonal Body Moving Amidst Polygonal Barriers," *Communications on Pure and Applied Mathematics*, vol. 36, no. 3, pp. 345–398, 1983.

[29] P. Sharma, A. Dutta, and A. Saxena, "Determination of Optimal Contact Points for Constraining a Prismatic Object by a Group of Mobile Robots," in *Robotics, Automation and Mechatronics, 2006 IEEE Conference on.* Bangkok: IEEE, 2006, pp. 1–5.

[30] P. Shi and J. N. Hua, "Mobile Robot Dynamic Path Planning Based on Artificial Potential Field Approach," *Advanced Materials Research*, vol. 490, pp. 994–998, 2012.

[31] P. Song and V. Kumar, "A Potential Field Based Approach to Multi-robot Manipulation," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, vol. 2. Washington, DC: IEEE, 2002, pp. 1217–1222.

[32] B. Stewart, A. Venkat, J. Rawlings, S. Wright, and G. Pannocchia, "Cooperative distributed model predictive control," *Systems & Control Letters*, vol. 59, no. 8, pp. 460–469, 2010.

[33] T. Summers, F. Cortesi, and J. Lygeros, "On Submodularity and Controllability in Complex Dynamical Networks," in *IEEE Transactions on Control of Network Systems*, 2014, pp. 1–10.

[34] D. Sunday, "Line and Segment Intersections." [Online]. Available: http://geomalgorithms.com/a05-_intersect-1.html

[35] D. Vallado, *Fundamentals of Astrodynamics and Applications.* Hawthorne: Microchosm Press, 2007.

[36] Y. Xia, G. L. Liu, P. Shi, J. Chen, and D. Rees, "Robust Constrained Model Predictive Control Based on Parameter-Dependent Lyapunov Functions," *Circuits, Systems & Signal Processing*, vol. 27, no. 4, pp. 429–446, 2008.

[37] R. Yue, J. Xiao, S. Wang, and S. L. Joseph, "Three-Dimensional Path Planning of a Climbing Robot Using Mixed Integer Linear Programming," *Advanced Robotics*, vol. 24, no. 15, pp. 2089–2118, 2010.