CONVEX MODEL PREDICTIVE CONTROL FOR VEHICULAR SYSTEMS

Thesis by

Tiffany Huang

In Partial Fulfillment of the Requirements

for the Degree of

Bachelor of Science in Mechanical Engineering

California Institute of Technology

Pasadena, CA

2014

(Submitted June 2014)

This thesis is dedicated to the author's parents, Steve and Chennie Huang, who have been unwavering supporters every step of the way.

Acknowledgements

The author would like to thank Professor Joel W. Burdick and Dr. Matanya Horowitz for their patience, mentorship, and help in her academic career and research. Thanks also go to the California Institute of Technology for supporting this research.

Abstract

In this work, the author presents a method called Convex Model Predictive Control (CMPC) to control systems whose states are elements of the rotation matrices $SO(\mathrm{n})$ for n = 2, 3. This is done without charts or any local linearization, and instead is performed by operating over the orbitope of rotation matrices. This results in a novel model predictive control (MPC) scheme without the drawbacks associated with conventional linearization techniques such as slow computation time and local minima. Of particular emphasis is the application to aeronautical and vehicular systems, wherein the method removes many of the trigonometric terms associated with these systems' state space equations. Furthermore, the method is shown to be compatible with many existing variants of MPC, including obstacle avoidance via Mixed Integer Linear Programming (MILP).

Table of Contents

%

List of Figures

1 Introduction

1.1 Objective

In this thesis, we will use the special Euclidean groups $SE(2)$ and $SE(3)$, which are the sets of rigid body transformations in two and three dimensions respectively. A new method will be demonstrated that involves convex relaxation of $SE(2)$ and $SE(3)$ to frame the motion planning of rotational systems as easily solvable convex optimization problems. This method, convex model predictive control (CMPC), will be implemented on simple spacecraft models ranging in complexity from a simple sphere with simple thrusters to a spacecraft with a momentum wheel. This technique will also be applied to the landing and recapturing of unmanned aerial vehicles (UAVs). This thesis will show that exact planning solutions for complex robotic systems that can be represented as rotational matrices can be found reliably and accurately through this method. The circumstances under which CMPC will guarantee a global minimum will also be explored. Lastly, the time, energy, and computing power required for producing a planning solution through our method will be compared with current methods.

1.2 Motivation & Background

Motion planning is required whenever a robotic system, such as a rover or robotic arm, needs to make decisions about moving in an unknown environment [10]. The focus of this thesis is on the motion planning of two specific systems: spacecraft and unmanned aerial vehicles (UAVs). Free flying spacecraft need to be energy efficient when planning their navigation routes because they can only carry a limited supply of fuel. With more efficient motion planning algorithms, spacecraft will be able to travel farther to reaches of space never explored before. Furthermore, spacecraft, which for the most part do not currently have a high level of autonomy, will not have to rely as much on navigation routes sent from Earth, saving valuable time and resources. Similarly, if UAVs could compute their own navigational routes more efficiently, they could stay in the air and be active for longer periods of time. Specifically, if a plane can be easily collected with minimal damage, it can be reused for a different mission with little difficulty. The recapture technique we will be investigating involves having a UAV use a hook to latch onto a rope. The UAV will generate a trajectory to fly into the rope, latch on, and slide down for collection. This recapture method has recently gathered the interest of the U.S. Navy as well as several defense and aerospace companies.

Motion planning for spacecraft has been solved using successful techniques including proportional-derivative (PD)- or linear-quadratic (LQ)-control [14], [17], Lyapunov design procedures [7], and explicit model predictive control [8]. However, cur-

rent methods often require linearizing large amounts of data and as a result introduce not only approximation error in the dynamics, but also significant computing costs. Inspired by recent developments in convex optimization, a new technique has been developed [9] that does not require a large amount of input to produce a solution.

1.3 Explicit Model Predictive Control

A widely used method to control spacecraft is explicit model predictive control (MPC) [8]. This method has historically been popular in industry for process control. In this method, the system decides which path to take by minimizing a user-chosen cost while the current state of the system provides continual feedback. If the system is linear, it takes the form

$$\begin{aligned} s(k+1) &= As(k) + Bu(k) \\ y(k) &= Cs(k), \end{aligned} \tag{1.1}$$

where at time k, $s(k)$ is the state of the system, $u(k)$ is the input, and $y(k)$ is the regulated output.

Linear problems such as these are easily solvable. However, many engineering problems, including that of complex spacecraft are not linear. In these cases, explicit model predictive control would involve taking samples of the system at different times on a predefined horizon. Then control inputs are applied and optimization is repeated with the new state of the system. Essentially, MPC linearizes the non-linear system at

each time sample to change the problem into the linear form shown in 1.1. Oftentimes, MPC uses a receding time horizon scheme where it plans for a short period of time. It then executes one step of the optimization and repeats the planning calculation until time runs out. A particular advantage of this method is its ability to constantly readjust to changes in the environment that might affect the generated trajectory. However, the need to start a new optimization at each time step slows down the planning process considerably. Where our work differs is in our use of the convex hull of $SE(2)$ and $SE(3)$ to constrain the motion of the system, which had previously required an infinite number of Linear Program constraints and thus has only been approximated.

Although MPC works well on many different and complex systems, linearization for complex systems may require significant computation and approximation. With the addition of many joints and robotic arms on spacecraft or with more complicated aerial vehicles like a quadrotor, the motion planning problem can be sufficiently complex to prevent the use of MPC for highly dynamic systems. Furthermore, due to the nature of the method, MPC does not guarantee a global minimum solution, only achieving a local minimum due to its local linearization.

1.4 Convex Optimization and Programming

Convex model predictive control lies fundamentally in convex programming [2]. Commercial convex optimization solvers exist that can solve complex problems involving up to tens of thousands of variables very quickly. Therefore, if it can be shown that

motion planning of vehicular systems can be represented by a convex optimization problem, the computing power currently needed for vehicular control can be drastically reduced. Convex functions can be defined as functions that satisfy the inequality

$$f(\theta u + (1-\theta)v) \leq \theta f(u) + (1-\theta)f(v), \tag{1.2}$$

where f is a convex set, $u, v \in f$, and $0 \leq \theta \leq 1$. This definition often simplifies to:

$$\nabla^2 f \succeq 0. \tag{1.3}$$

Examples of convex functions include linear, or affine, functions, quadratic functions, halfspaces, ellipsoids, and cones. We will be particularly interested in convex sets in the form of affine functions and second-order cones. Using convex functions, we can formulate convex optimization problems that can be solved easily with convex programming. A convex optimization approach is significant because convex optimization problems have a logarithmic convergence curve, which produces very fast results in practice. Also, by definition, convex optimization guarantees that all solutions are global minima, eliminating the local minima problem encountered through model predictive control. In standard form, convex optimization problems are framed

as follows

$$\begin{aligned} \min. f(x) & \qquad \text{(objective function)} \\ \text{such that } g_i(x) \leq 0, i = 1, \ldots, m & \qquad \text{(inequality constraint)} \\ a_i x = b_i, i = 1, \ldots, n & \qquad \text{(equality constraint).} \end{aligned} \tag{1.4}$$

Thus, the solution x^* would satisfy:

$$f(x^*) = \inf\{f(x) | g_i(x) \leq 0, i = 1, \ldots, m, a_i x = b_i, i = 1, ..., n\}. \tag{1.5}$$

1.4.1 Convex Relaxation

The algorithm presented in this work relies on the use of *orbitopes*, the convex set consisting of convex hulls of the orbits given by a compact, algebraic group G acting linearly on a real vector space. The convex hull $\text{conv}(S)$ is the set of all convex combinations of points in S. More specifically, this method deals with a group acting on its identity element, or the *tautological orbitope*. It relaxes the constraint on $SO(2)$, a rotation in 2 dimensions, from a unit sphere, $p^2 + q^2 = 1$, to a unit disk, $p^2 + q^2 \leq 1$ by taking its convex hull, where $p = \cos\,\theta$ and $q = \sin\,\theta$. Through this relaxation, the tautological orbitopes for $SO(\text{n})$, $n = 2, 3$ can be written as a linear matrix inequality

(LMI) [9] as follows

$$\begin{aligned}\mathrm{conv}(SO(2)) &= \{\begin{bmatrix} a & b \\ -b & a \end{bmatrix} : a^2 + b^2 \leq 1\} \\ &= \{\begin{bmatrix} a & b \\ -b & a \end{bmatrix} : \begin{bmatrix} I & \begin{pmatrix} a \\ b \end{pmatrix} \\ \begin{pmatrix} a & b \end{pmatrix} & 1 \end{bmatrix} \succeq 0\}. \end{aligned} \tag{1.6}$$

LMIs are a class of convex constraints that have had a profound effect on the field of optimization and control in recent decades [1], [2]. If the traditional motion planning problem can be formulated into a convex objective or cost function, the tautological orbitope LMIs can be applied as convex restraints to transform the problem into a semidefinite optimization problem.

1.5 Further Applications

This new method looks specifically at rotational groups, making it ideal for spacecraft modeling because spacecraft are essentially rotating systems in space. As scientists are looking to explore new regions of space, more efficient control algorithms are highly desirable. Additionally, new space missions are beginning to explore robotic self-assembly in space. For instance, setting up a large telescope on a remote planet would benefit from a self-assembling system that would need little human intervention. Self-assembly requires precision, complex dynamics, and a large amount of

movement and energy. Developing a fast, reliable motion planning algorithm that could be applied to extremely complex systems would make self-assembly in space a feasible new venture. Systems applicable for this method also include airplanes and cars, which widens applications of convex relaxation to Earth-based vehicles in urban environments. Since convex optimization would take less time than current motion planners for aerial vehicles, our new method would allow for greater maneuverability of autonomous aerial vehicles. Reliable, autonomous aircraft and land vehicles are in high demand for search and rescue operations as well as for surveillance and mapping missions.

2 Methodology

2.1 Analytical Methods

2.1.1 Rigid Body Transformations in $SE(2)$ and $SE(3)$

In this work, the systems analyzed were assumed to act as rigid bodies, meaning that the distance between any two points on the body remains the same regardless of external forces. Therefore, when modeling the dynamics of a vehicle, rigid body transformations must be applied. These transformations are elements in $SE(2)$ and $SE(3)$ [11] containing the rotation (R) and translation (T) of the system at time t. In $SE(2)$, the state of the system is as follows, with x, y representing the translation of the system and θ representing the orientation of the system

$$D = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \tag{2.1}$$

$$D = \begin{bmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{bmatrix}.$$

In $SE(3)$, the state of the system is identical, except T has an additional z component, and R becomes a 3×3 matrix. Rotation through θ about the z-axis is as

follows

$$R^3 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.2}$$

2.1.2 MATLAB CVX

The MATLAB package CVX was used to solve the convex optimization problems formulated for motion planning simulations. The package uses a disciplined convex programming approach, and can also be used for geometric programming. With this approach, a basic library of convex functions and a small set of rules from convex analysis are used to solve the convex optimization problem. The problems involving mixed integer linear programming were numerically solved using MOSEK through CVX because the default CVX solver cannot support Mixed-Integer variables [6].

2.2 Experimental Methods

2.2.1 Ode45 Accuracy

An important measure of the success of this new approach is a comparison between convex optimization and other commonly used methods. In preparation for the comparisons, the MATLAB function *ode45* was tested for long-term accuracy. MATLAB only stores a certain number of decimal places in an output, so if this output is repeatedly used as input into *ode45*, eventually the output will fail to be as accurate as

other differential equation solvers. To test *ode*45, the function was implemented for a simple model two separate times. The first time, *ode*45 was implemented to generate a trajectory for a full time horizon. The second time, the function was applied every few seconds, using the result from the previous few seconds as the input for the next differential equation. The point at which the results differed between the two tests indicated the limits of *ode*45.

2.2.2 Pd Controller

A proportional-derivative (Pd) controller was applied to a spacecraft system to verify the dynamics of the vehicle. The control input, u, was set to be dependent on the state of the system, x

$$u = -Kx \tag{2.3}$$

$$K = \begin{bmatrix} K_{px} & 0 & 0 & 0 & K_{vx} & 0 & 0 & 0 \\ 0 & K_{py} & 0 & 0 & 0 & K_{vy} & 0 & 0 \\ 0 & 0 & K_{p\theta} & 0 & 0 & 0 & K_{v\theta} & 0 \\ 0 & 0 & 0 & K_{p\psi} & 0 & 0 & 0 & K_{v\psi} \end{bmatrix}, \tag{2.4}$$

where K_{px} denotes the constant related to the position of the spacecraft in the x-direction and K_{vx} denotes the constant related to the velocity of the spacecraft in the x-direction. The controller was designed to keep the system in one spot, almost like a spring-mass system. If the system began to drift away, the controller would pull it back to the designated starting point. *Ode45* was then used to simulate the

trajectory of the system from an initial position of disturbance to the origin.

2.3 Convex Model Predictive Control Over $SO(n)$

We are concerned with the finite horizon control of systems which have a set of states $R(t)$ that are restricted to lie in the special orthogonal group representing rotational matrices, i.e. $R(t) \in SO(n)$. We will furthermore assume that the dynamics of the system are linear with respect to this variable. For the remainder of this work, we will in particular consider systems where $R(t)$ consists of the body orientation and forward velocity or acceleration in this body frame. Specifically, we use models of the form

$$\dot{R}(t) = AR(t) + Bu(t) \tag{2.5}$$

$$\dot{s}(t) = R(t)M, \tag{2.6}$$

where M is a fixed forward velocity vector in the body frame such as

$$M = \begin{bmatrix} I & \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 1 \end{bmatrix} & \end{bmatrix}. \tag{2.7}$$

This M matrix moves the vehicle forward a unit length in the x-direction without any rotation. As will be seen, this class of problems encompasses a number of robotic systems, from the Dubins car to satellite systems.

We discretize such systems in time, and place them in the MPC framework, yielding optimizations of the following form.

$$\begin{aligned} \min. \quad & \sum_{t=1}^{T} l\left(s\left(t\right), u\left(t\right)\right) + \phi\left(x\left(T\right)\right) \\ s.t. \quad & R(t+1) = R(t) + hu(t) \\ & s(t+1) = R(t) \cdot M \\ & R(t) \in SO(n), \end{aligned} \tag{2.8}$$

where $n = 2, 3$, and M is some forward vector in the body frame.

Via orbitopes, equation (2.8) is replaced with the convex constraint $R(t) \in \mathrm{Co}\left(SE\left(n\right)\right)$. The question then arises: under what scenarios will this relaxation be exact, i.e. when will the solution to the relaxed and original problems coincide? In related work [9], it was demonstrated that such a guarantee may be provided for a wide variety of computer vision problems. Unfortunately, no such guarantee is available in the MPC context. Instead, we propose optional non-convex restrictions on the variable $R(t)$ based on Mixed-Integer constraints.

Systems with second-order dynamics on the rotation component $R(t)$ are handled in a similar fashion, with the simple addition of a state variable, for instance a variable $\dot{R}(t) = S(t)$, where $\dot{S}(t) = u(t)$.

2.3.1 Receding Time Horizon Control

Since motion planning involves movement in the future, all planning and control techniques need to handle some uncertainty. This uncertainty could come from instrumentation, a changing environment, simplifying assumptions made in the model, or limitations in software or hardware accuracy. Planning entire trajectories and missions at once saves computing time. However, the further the actions are planned to happen in the future, the more uncertainty measures will compound and propagate. Additionally, changes in the environment or other variables may require the model to be adjusted midway through the planned trajectory. To address these issues, a receding time horizon control scheme was incorporated into some of the examples. The total time horizon remained at the same, but at each time step, the vehicle would plan a trajectory for a few time steps into the future and execute its planned movement for only one time step. Using this method, the vehicle begins a new optimization at each time step, dramatically increasing the number of optimizations calculated during one mission.

Because the trajectory was only planned a few time steps ahead, the vehicle could not be expected to reach its final destination in each of its planning steps. Instead of constraining the final position of the system to a chosen goal point, a penalty was put on the end condition. The penalty took the form

$$[x(t) - x(t_f)] \cdot Q \cdot [x(t) - x(t_f)]^T, \tag{2.9}$$

where Q is a diagonal matrix, $x(t)$ is the x, y position of the vehicle at time t, and $x(t_f)$ is the position of the vehicle at the end of the total time horizon. This penalty on the systems end condition was placed in the objective function of the convex optimization scheme.

2.3.2 Mixed-Integer Linear Programming (MILP)

MILP is a modification of a linear program (LP) in which some of the constraints take only integer values. For our purposes, the constraints will only take binary values, 0 or 1. MILP allows for the inclusion of discrete decisions, which are non-convex in the optimization problem. For instance, integer constraints enable capabilities such as obstacle avoidance in which a vehicle must determine whether to go "left" or "right" around an obstacle. By definition, MILP optimization problems are linear, so they will always output a globally optimal solution. In general, MILPs are nondeterministic polynomial time (NP)-complete [5], but in many cases they can be solved by relaxation to their LP form.

In the context of MPC, it has been found that Mixed-Integer constraints may be useful in encoding non-convex constraints in the domain [13]. Our incorporation of Mixed Integer Linear Programs (MILPs) into the framework is twofold: first, to demonstrate that existing MPC methodologies apply readily; and second, because it allows us to constrain the solutions within the convex hull of $SO(n)$ in order to incorporate additional control design criteria. For aeronautical applications, this allows the enforcement of dynamic constraints such as minimum speed.

We review the encoding of obstacle avoidance into MILP constraints following [12]. For the two dimensional case with a rectangular obstacle, $(x_{\min}, y_{\min})$ is defined as the lower left-hand corner of the obstacle and $(x_{\max}, y_{\max})$ is defined as the upper right-hand corner of the obstacle. Points on the path of the vehicle (x, y) must lie outside of the obstacle, which can be formulated as a set of rectangular constraints. To convert these conditions to Mixed-Integer form, we introduce an arbitrary positive number M that is larger than any other variable in the problem. We also introduce integer variables a_k that take values 0 or 1. The rectangular constraints now take the form

$$\begin{aligned} x &\leq x_{\min} + Ma_1, \\ -x &\leq -x_{\max} + Ma_2, \\ y &\leq y_{\min} + Ma_3, \\ -y &\leq -y_{\max} + Ma_4, \\ \sum_{k=1}^{4} a_k &\leq 3. \end{aligned} \tag{2.10}$$

Note that if $a_k = 1$, then the k^{th} constraint from (2.10) is relaxed. If $a_k = 0$, then the k^{th} constraint is enforced. The final MILP constraint ensures that at least one condition from (2.10) is enforced and the vehicle will avoid the obstacle.

2.3.3 Convex Hull MILP Constraints

To use MILP to constrain solutions within the convex hull of $SO(2)$, an "obstacle" is introduced into the convex hull of $SO(2)$. The hull of $SO(2)$, as has been shown, can be represented as a circle of radius 1 centered at the origin, and the obstacle corresponds to a region within it that is admissible, preventing the determinant from obtaining a set of values. This is of importance as the forward velocity vector in the world frame is dependent on the forward velocity in the body frame, M in (2.8), rotated by R. Thus, a decrease in the determinant of R corresponds to a lower velocity, and a slowing of the vehicle. While appropriate in some contexts, e.g. land vehicles, it is undesirable in others, such as aircraft.

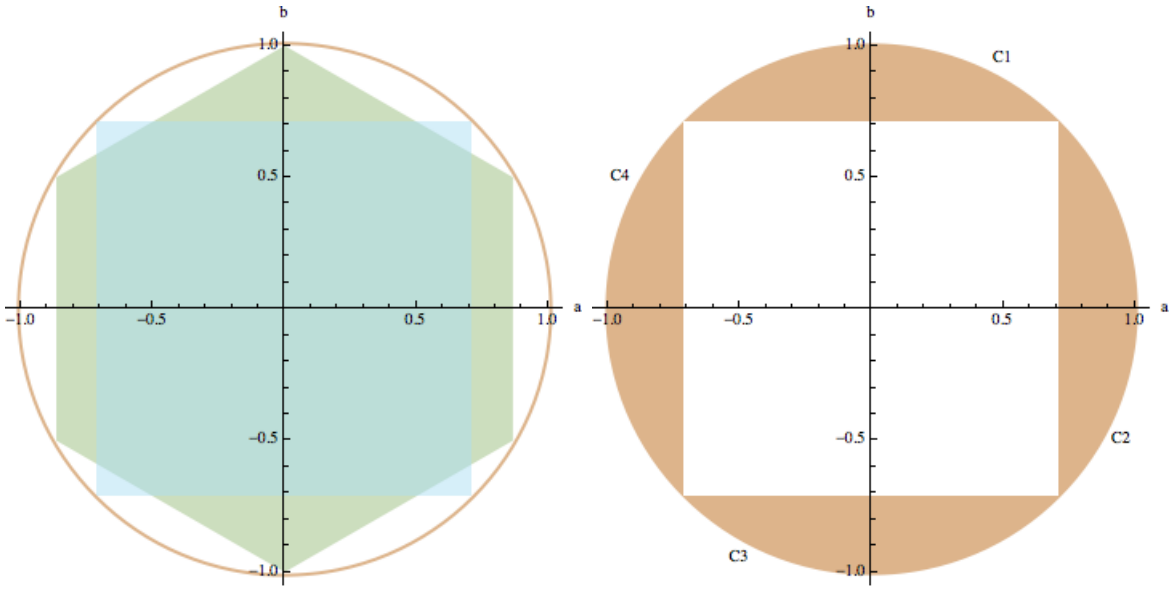


Figure 2.1: Mixed-Integer constraints applied to the determinant of Co($SO(2)$). On the left, we illustrate square or hexagonal regions that are inadmissible for the variables in the rotation $R \in$ Co($SO(2)$). The resulting admissible regions are a union of individually convex sets, illustrated in the right as $C_1, \ldots, C_4$ for the square region.

Ideally, this obstacle would be a circle that restricts the determinant of R from reaching 0. However, MILP constraints cannot represent a circular obstacle, so we start with a square obstacle by using the same form of constraints shown in (2.10). Illustrations of the determinant constraint are shown in Figure 2.1. As the number of faces is increased, the symmetry of the minimum speed constraint may be improved as desired.

2.3.4 Sources of Error

Some cases of spacecraft systems or UAV recapture problems might generate conditions for which convex model predictive control will not guarantee an exact solution. When the determinant of the R (rotation) matrix is not equal to 1, the solution is not exact. A possible solution would be to project these conditions onto the set of proper rotations to create an estimate of the translation, which could then be used with existing methods to guarantee the optimal solution. We have preliminary results that guarantee the convex relaxation is exact, and to mitigate other inapplicable answers, we will continue to broaden the space of exact solutions through continued theoretical analysis. Another source of error is the use of the MATLAB package CVX to solve our convex optimization problems. CVX is a general, readily available, commercial solver that was not coded to solve our problems in the most efficient manner. Consequently, computing times recorded for our method in the various cases may be significantly slower than predicted times. One way to speed up the convex programming is by coding the solver directly in a language such as C++ without parsing.

Mixed-Integer constraints are not currently supported with MOSEK in the 3D case, which precluded us from reproducing some of our examples in $SE(3)$.

3 Results

3.1 Simple Satellite

For this experiment, a simple satellite experiencing constant thruster forces in the $x, y,$ and θ directions were modeled in state space form. The MATLAB function *ode45* was applied to the model from time $t = 0$ to $t = t_f$ with the satellite starting at the origin with 0 velocity. Then, *ode45* was applied to the same model except at time $t = 10$, the position and velocity of the spacecraft were taken and used as the initial condition inputs for *ode45* from time $t = 10$ to time $t = 20$. The process was repeated every 10 seconds until time $t = t_f$. With this example, *ode45* was found to be accurate for up to 670 seconds. At times after 670 seconds, the finite arithmetic errors of Matlab began to affect the results non-trivially. Fortunately, all the experiments performed in this thesis deal with time horizons less than 100 seconds, so the uncertainty propagation through the usage of *ode45* for these purposes is negligible.

3.2 Spacecraft with Momentum Wheel

The next step up in spacecraft complexity was the addition of a momentum wheel. A momentum wheel is attached to a satellite with an electric motor (3.1). When the

electric motor changes the rotational speed of the momentum wheel, the spacecraft begins to rotate in the opposite direction due to conservation of angular momentum. The spacecraft was once again subjected to constant thruster forces and a constant torque was applied to the momentum wheel. The mass of the satellite was arbitrarily set to five times the mass of the wheel.

Figure 3.1: Momentum wheel [3]

The initial conditions of the spacecraft were set to $[x, y, \theta, \psi] = [10, 10, 10, 10]$ and the Pd controller was designed to bring the spacecraft back toward the origin. Here, θ represents the orientation of the spacecraft and ψ represents the angle of the momentum wheel with respect to the spacecraft orientation. The controller produced trajectories as follows in Figures 3.2, 3.3, and 3.4, showing that the simple Pd controller was effective in generating a suitable path from an initial position to a goal point.

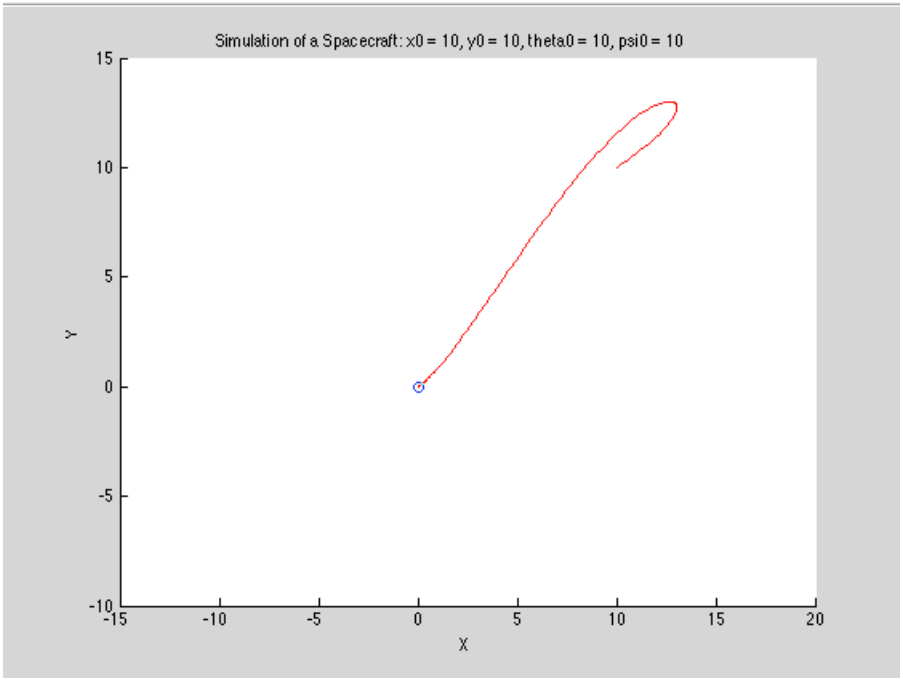


Figure 3.2: x, y trajectory of spacecraft with momentum wheel using a Pd controller. The trajectory begins at $(x, y) = (0, 0)$ and ends at $(10, 10)$.

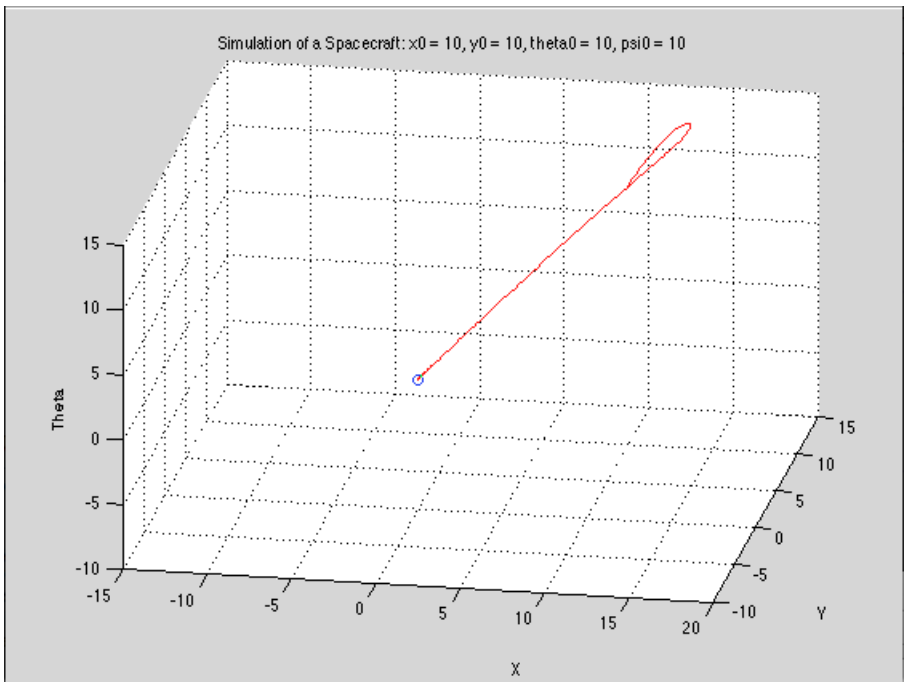


Figure 3.3: x, y, θ trajectory of spacecraft with momentum wheel using a Pd controller.

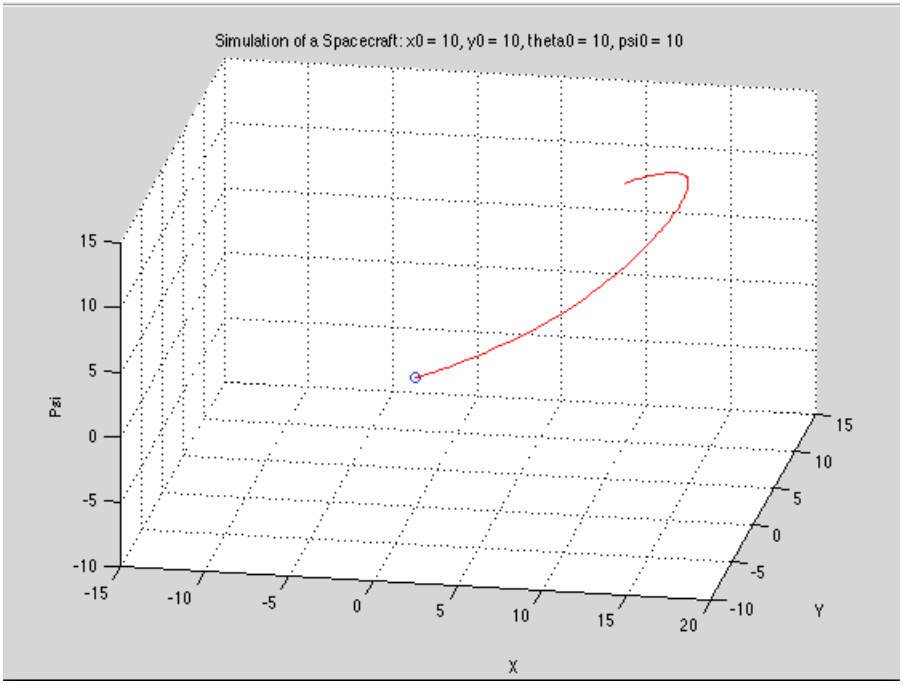


Figure 3.4: x, y, ψ trajectory of spacecraft with momentum wheel using a Pd controller.

3.3 Transfer Orbits

The simple motion planning problem of transfer orbits was used as a starting point for the convex optimization experiments and simulations. In this problem, a spacecraft moves from one orbit to another (3.5).

Using the theory detailed in [9] and [2], the spacecraft was modeled as a second-order system both with and without a momentum wheel in $SE(2)$ and $SE(3)$ using CMPC. This problem required second-order dynamics because the velocity of the spacecraft cannot be controlled. The thrusters in our model only have an on mode and an off mode, so only the acceleration of the spacecraft can be directly controlled by user input. As explained in the analytical methods section, the state of the system

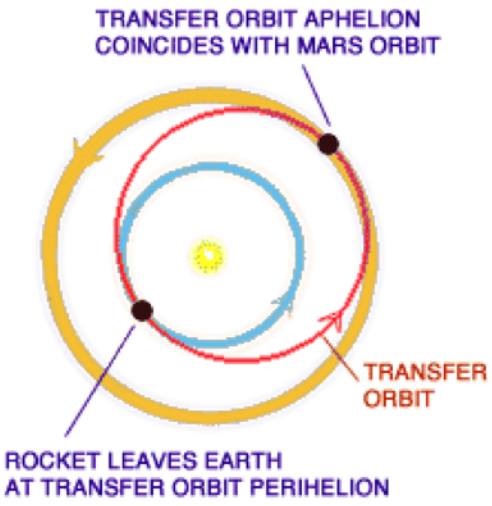


Figure 3.5: Transfer orbit of a rocket from Earth to Mars [15]

was modeled by a rigid body transformation.

For this simulation, the initial orbit was set to x, y coordinates of (10, 10) and the final orbit was set to (0, 0). An initial velocity was set to the x and y velocity (0.5, 0) and the spacecraft was constrained to have a final velocity of (0, 0). A trajectory was successfully generated as shown in Figure 3.6 with a total CPU time of about 2.10 seconds and an optimal value of 1.82192e-05. The optimal value shows that the trajectory generated through convex relaxation requires very little energy. In addition, the curving at the beginning of the trajectory suggests that in this model, maintaining high speeds with a wider curvature is more energy efficient than minimizing path curvature.

A similar example was replicated in 3D in which a spacecraft was given an orientation aligning its thrust with the negative x-axis, and a desired end location at $(x, y, z) = (5, 10, 25)$ that must be achieved with zero velocity. The resulting trajectory is shown in Figures 3.7 and 3.8. As is seen, the determinant drops in the middle

of the trajectory, largely allowing the spacecraft to coast and conserve fuel.

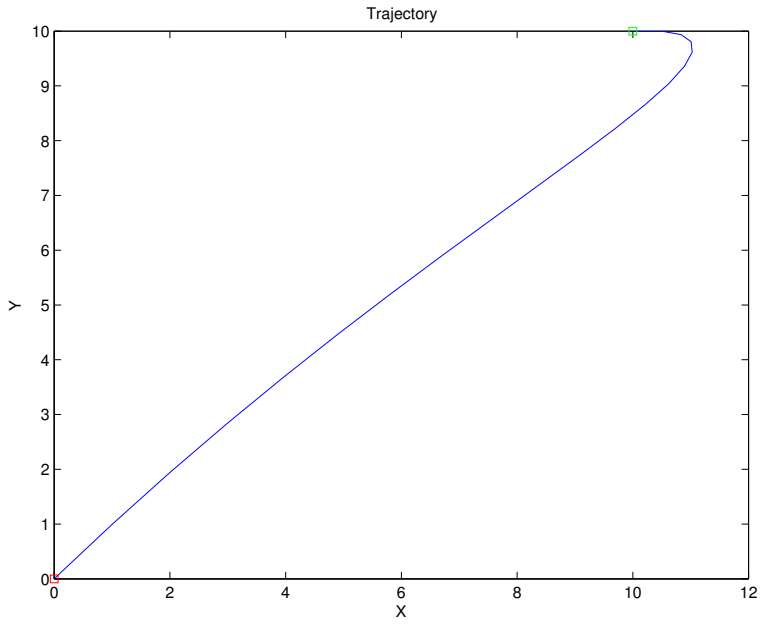


Figure 3.6: Trajectory of spacecraft between orbits using convex relaxation. The start point highlighted in green is at $(10, 10)$ and the goal point highlighted in red is $(0, 0)$.

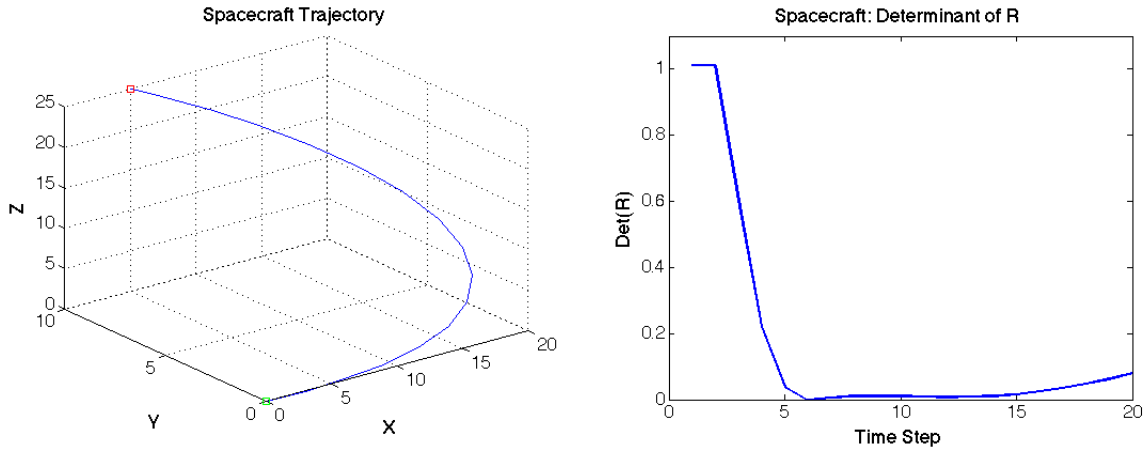


Figure 3.7: Trajectory for a 3D spacecraft maneuver is shown on the left. On the right is the determinant of the rotational state over the course of the maneuver.

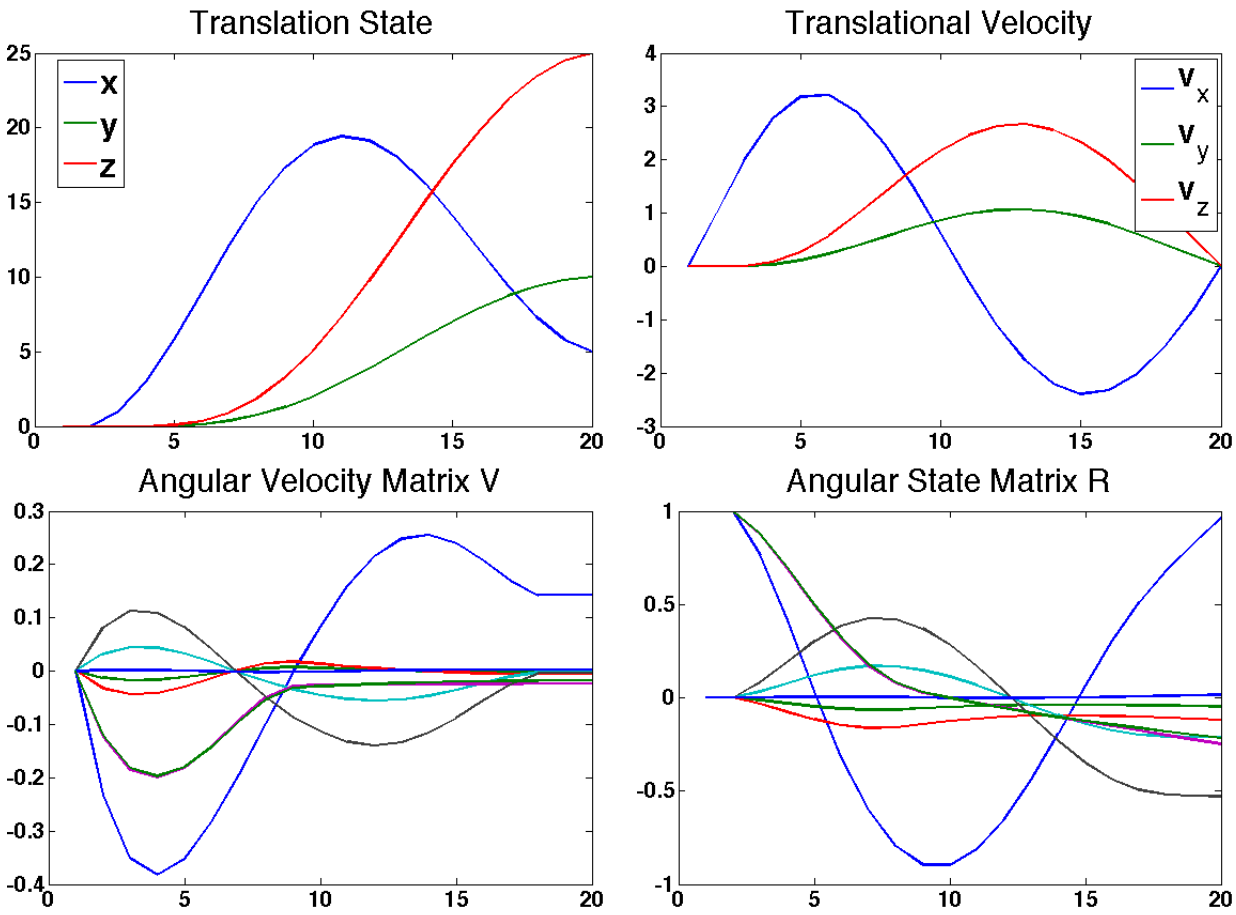


Figure 3.8: Trajectory data for the 3D spacecraft maneuver.

3.4 Dubins Car Model

For the UAV recapture problem, a Dubins car type model was first implemented and solved. In this model, the objective is to find the Dubins path, or the shortest path that connects two points in the two-dimensional space. For a vehicle following the car type model, the maximum turn rate results in a minimum turning radius, and backward motion is not allowed. The length of the path is optimized using the function

$$L(\vec{q}, \vec{u}) = \int_0^{t_f} \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2}\, dt, \tag{3.1}$$

where t_f is the time when the goal is reached, $\dot{x}$ is the velocity in the x-direction, and $\dot{y}$ is the velocity in the y-direction. The configuration of the vehicle is designated by $q = (x, y, \theta)$. The solution of this Dubins path problem is given by

$$\begin{aligned} \dot{x} &= V\cos\,\theta \\ \dot{y} &= V\sin\,\theta \\ \dot{\theta} &= u, \end{aligned} \tag{3.2}$$

where V is the constant speed of the vehicle and u is the turn rate [4]. The example on which we applied CMPC consisted of a UAV with no inertia attempting to latch onto a vertical rope for recapture as seen in Fig 3.9. Using a time horizon of 60 seconds, the UAV had its initial position set at the origin facing due north. The rope was set at a particular location, for instance at (5,10) where the distance was measured in feet. CMPC was implemented with the final position of the UAV constrained to equal the rope's position at time $t = t_f$.

Once a trajectory was generated, the translational velocity of the UAV was investigated by plotting the determinant of the R matrix over time. The R matrix is the rotation matrix given in (2.1) for the rigid body transformation of the plane. Since $R * M$ determines the displacement of the plane at each time step (2.8), the determinant of R is indicative of the movement of the plane. If the determinant of R

Figure 3.9: UAV skyhook recapture system [16]

shrinks, the plane's velocity decreases. When the determinant of R goes to zero, the plane will come to a complete stop, which is not physically possible for an airplane in mid-flight. Consequently, to prevent the determinant of R from shrinking too much, the time horizon was shortened to about 20 seconds to give the plane less time to slow down. For the Dubins example, the aerial vehicles initial and final position were constrained to be the origin and (5, 10) respectively. The initial velocity of the vehicle was set to an x, y velocity of (1, 0) and the final velocity was constrained to (0, 0). As seen in this trajectory (Figure 3.10) much more than in the transfer orbit example, maintaining a high velocity and large turn radius produces the most efficient path. The convex relaxation method output a low optimal value of 6.05796e-05, suggesting that our method is both fast and efficient. When the solution is exact, the

determinant of R should equal unity. In examining the determinant of the rotation matrix R we see that the convex relaxation is not always exact. Lower values of the determinant correspond to rotation matrices that lie inside the convex hull of SO(2), since the system is traveling at a velocity lower than maximum speed, $v < V$. As shown in Figure 3.11, the vehicle reaches top speed, then alternates slowing down and speeding up. The solution took approximately $0.1s$ to compute on a 2.3 GHz Intel i7 processor.

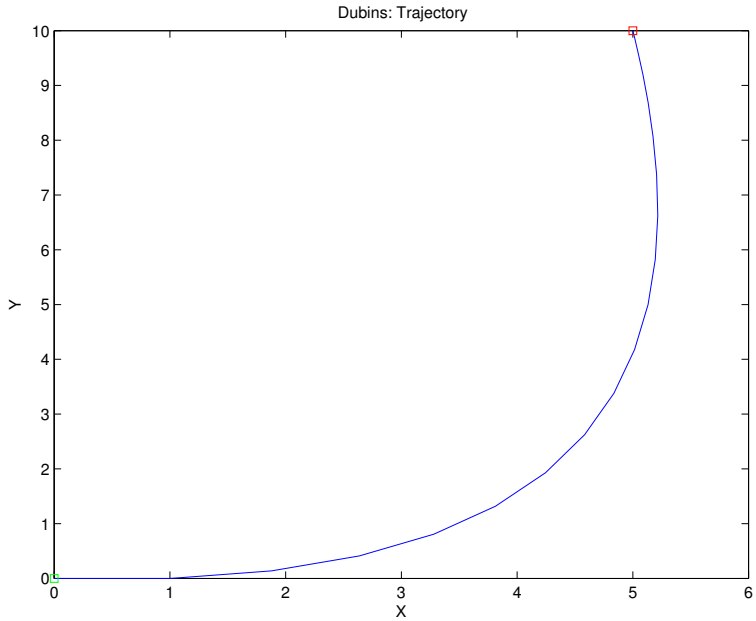


Figure 3.10: Trajectory of an aerial vehicle using the Dubins car model. The trajectory begins at $(x, y) = (0, 0)$ marked in green and ends at (5,10) marked in red.

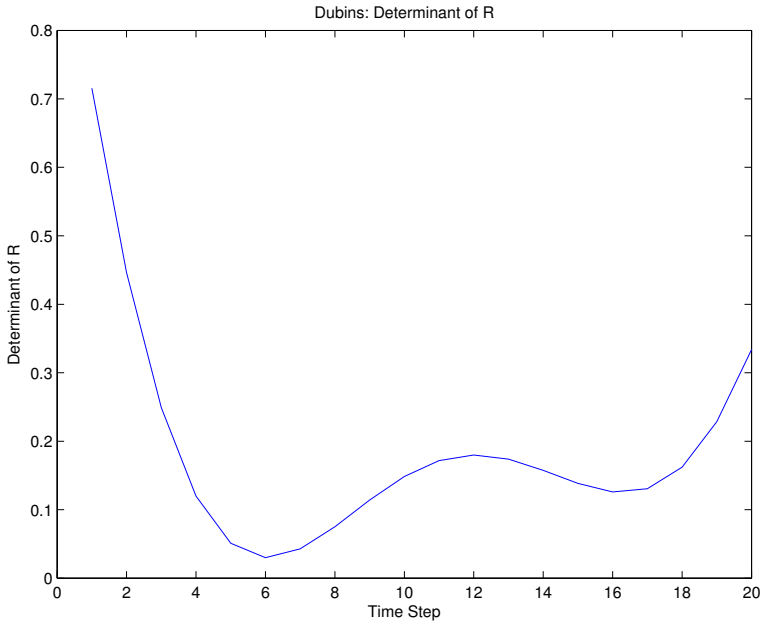


Figure 3.11: Determinant of the rotation matrix R of an aerial vehicle using the Dubins car model.

3.5 Receding Time Horizon

The next example presented is the CMPC scheme for a UAV with receding time horizon control. The initial position and velocity were constrained to be $(x, y, \theta) = (0, 0, 0)$ and $(1,\ 0,\ 0)$ respectively. The total time horizon remained at 30 seconds, but at each time step, the plane planned a trajectory for five time steps into the future and executed its planned movement for one time step. Since the trajectory was only planned five time steps ahead, the UAV could not be expected to reach its final destination in each of its planning steps. Instead of constraining the final position of the UAV to the position of the rope, a quadratic penalty was put on the

end condition (2.9).

As the condition $det(R) = 0$ corresponds to the UAV stopping altogether, physically impossible for this application, we restrict $det(R) > 0.3$ using a MILP square constraint as detailed in the methods section. Each receding horizon iteration took approximately $0.16s$ to compute and the optimal value was 1.43268e-09. This example takes a larger amount of CPU time to compute because of the increased number of iterations. However, the increased amount of iterations allows for a more reliable, accurate, and efficient solution as seen by the optimal value being significantly lower in this example than in others. The trajectory at time $t = 3$ is shown (Figure 3.12) to illustrate the planning performed at each time step. The green trajectory is the planned path and the blue line is the trajectory the vehicle has already travelled. The final trajectory generated is shown in Figure 3.13.

3.6 Receding Time Horizon Rope Interception

In the next stage of the UAV recapture problem, a moving rope scenario was investigated. For the open sea or any other outdoor environment, a reasonable assumption for the recapture of a UAV is that the rope to which the UAV needs to hook on is not stationary. The rope's movement could be affected by a number of factors including wind, waves, or movement of a connected structure. Consequently, the convex optimization method was applied to a moving rope model, which was implemented and

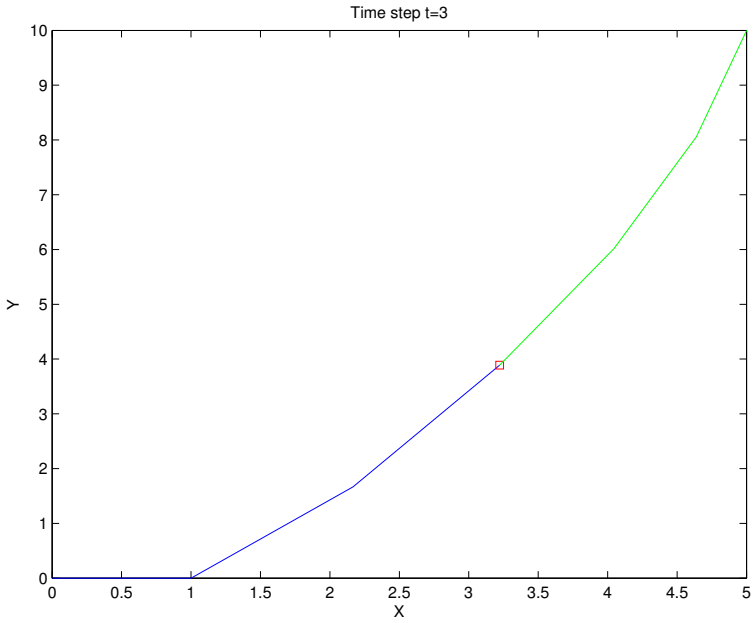


Figure 3.12: Trajectory of an aerial vehicle using a receding time horizon scheme with CMPC at time $t = 3$. The green line marks the planned path and the blue line marks the path already traveled. The red dot signifies the vehicle's current position.

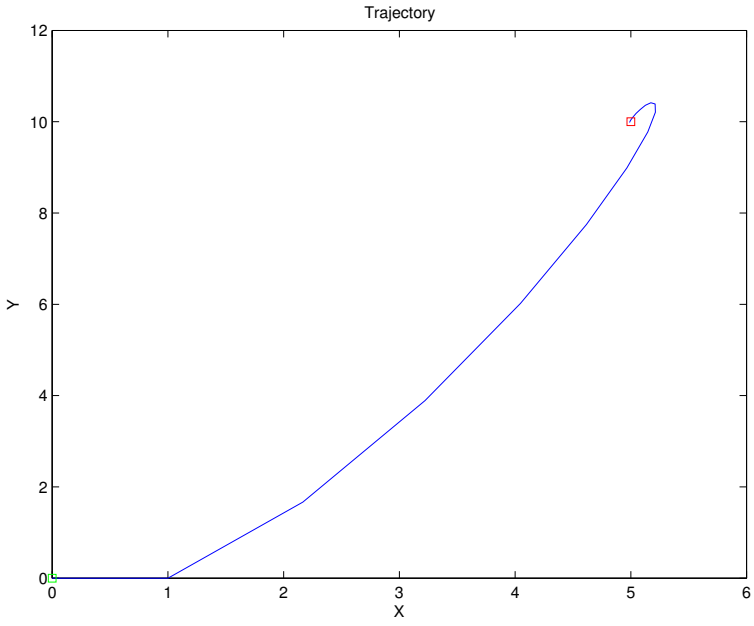


Figure 3.13: Final trajectory of an aerial vehicle using a receding time horizon model with CMPC. The vehicle starts at $(x, y) = (0, 0)$ and ends its trajectory at (5,10).

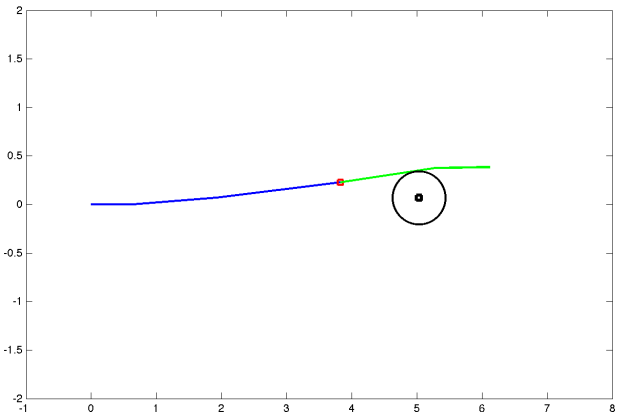
solved in MATLAB using CVX. The rope's motion was modeled as follows

$$\dot{x}_r = A_x \cos(\omega t) \tag{3.3}$$

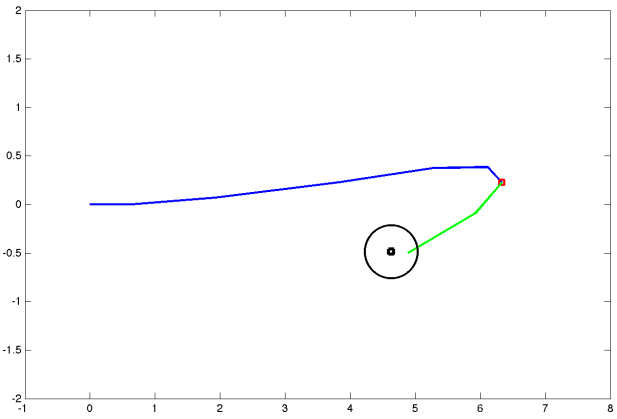
$$\dot{y}_r = A_y \sin(\omega t + \psi)$$

where $\dot{x}_r$ is the rope's velocity in the x-direction, $\dot{y}_r$ is the rope's velocity in the y-direction, and A_x, ω, and ψ are constants. The current position of the rope as determined by the model in (3.3) was entered as the destination point and end constraint for the UAV in the receding horizon control method. Therefore, at each time step, the UAV had a different end condition based on the rope's predicted position at the end of the planning time horizon. The rope oscillated around the point $(x, y) = (5, 0)$ with amplitude along each coordinate of $A_x = 0.5, A_y = 0.5$. The resulting trajectory is shown in Figure 3.14. As can be seen, the receding horizon framework allows for automated planning in the event the rope was missed. The capture terminated when the rope was achieved at $t = 24$. The goal location for the problem was updated to be the current state of the rope. Of course, if the future location of the rope is known a-priori, this problem reduces to those already examined. Resulting trajectories are shown.

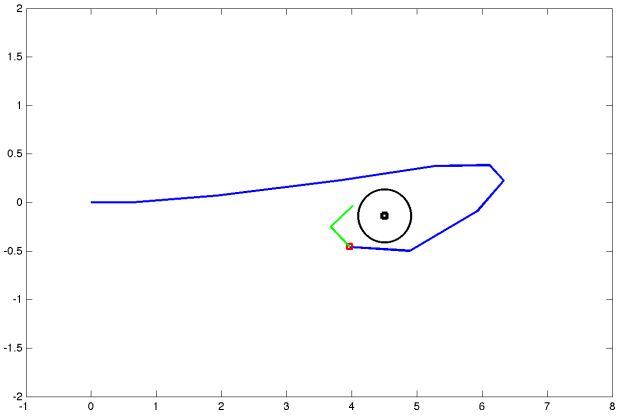
t=4



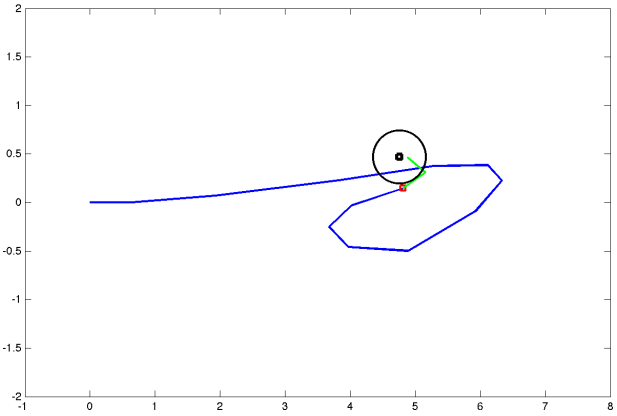
t=7



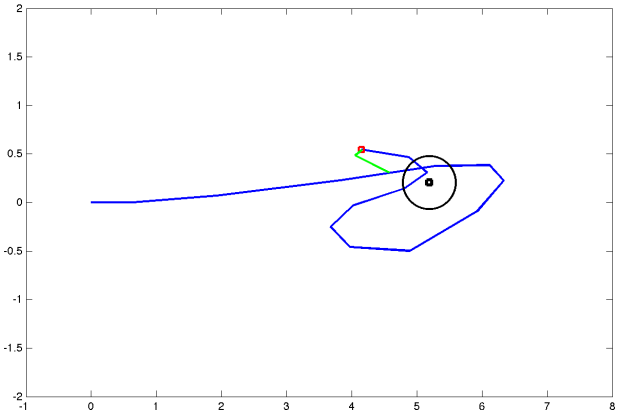
t=10



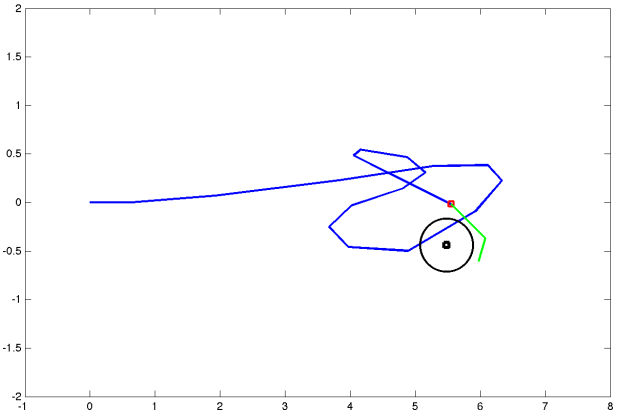
t=13



t=16



t=19



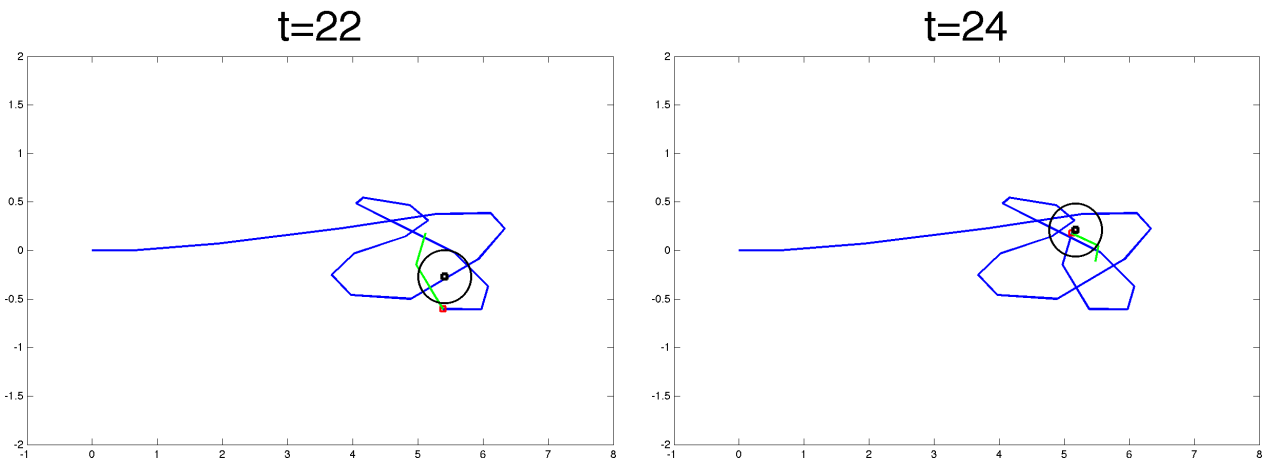


Figure 3.14: Example trajectory for the UAV when performing retrieval via interception with rope location.

3.7 MPC

To compare our new convex relaxation method with traditional path planning mechanisms, a MPC scheme was created to generate a trajectory with the initial conditions $(x, y, \theta) = (0, 0, 0)$ and final conditions (5, 10, 0). This nonlinear system was linearized at each time step to produce a path. The initial trajectory was set as a horizontal line in the x-direction and the control scheme was allowed to re-linearize and shape the trajectory for 30 time steps. The total CPU time required for this method was 2.05 seconds and the optimal value found was 0.134073.

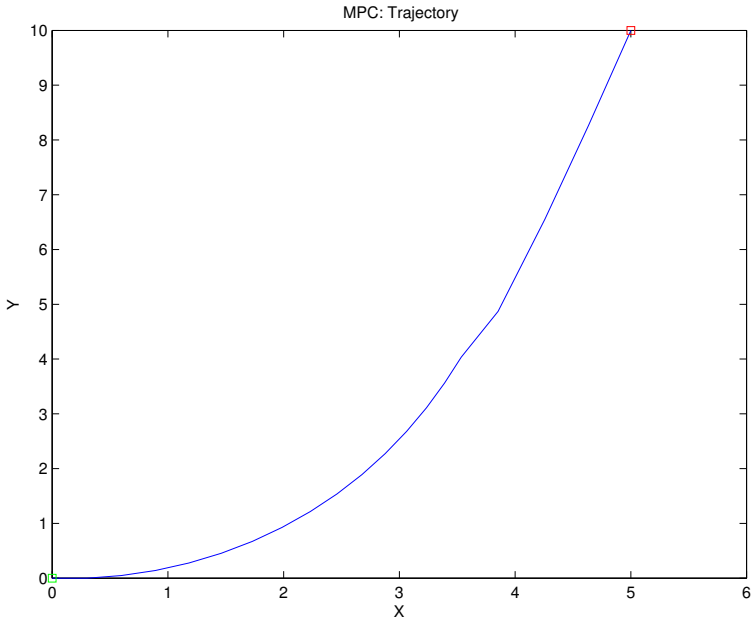


Figure 3.15: Trajectory of an aerial vehicle using model predictive control. The trajectory begins at $(x, y) = (0, 0)$ and ends at (5, 10).

3.8 Comparison Between MPC and Convex Relaxation

A script was run to compare computational time and optimal values for MPC and CMPC using the Dubins car model. Each method was run 1000 times with the same random initial and final conditions. Histograms were generated to compare the distribution of computational time (Figures 3.16 and 3.17) and energy required to generate trajectories (Figures 3.18 and 3.19). Overall, MPC took between about 1.4 - 2.3 seconds to compute and the optimal values seemed to range mostly between 0 - 20 with heavy outliers (optimal values greater than 100) thrown out. For the Dubins

convex relaxation method however, it took about 1.4 - 2.3 seconds to generate a trajectory on average and the optimal values ranged from 0 - 1.4. Therefore, both methods took about the same amount of computational time, but CMPC generated solutions about up to 10 times more efficient than the currently used MPC method.

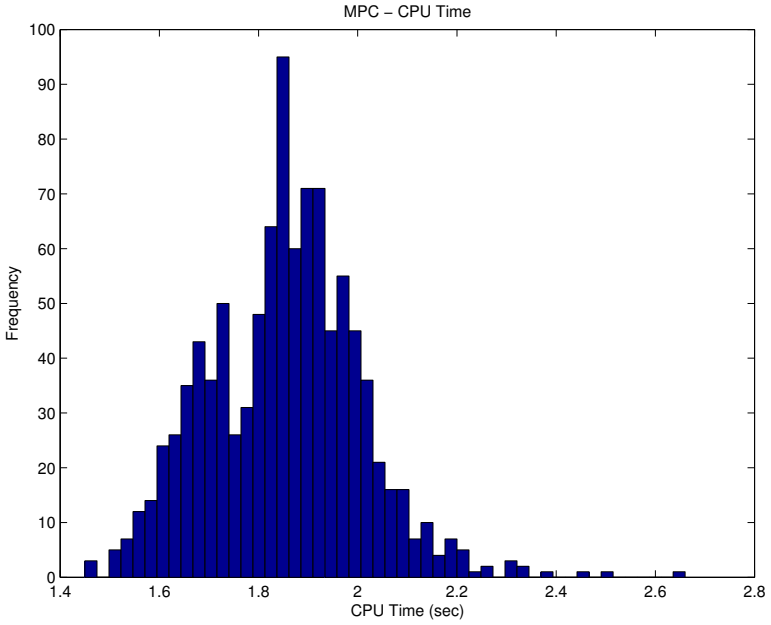


Figure 3.16: Total CPU time of the MPC method over 1000 runs.

Figure 3.20 shows the ratio of MPC optimal values to Dubins optimal values. For about 155 runs, or 15.5% of the runs, MPC produced a smaller optimal value than Dubins, which translates to MPC producing a more efficient (lower control cost) solution. However, CMPC was able to generate a more efficient path 84.5% of the time, with very similar computational cost. Further analysis shows that when CMPC with the Dubins car model outperforms MPC in terms of control cost, our method produces paths that are mostly greater than 70% more efficient (Figure 3.21).

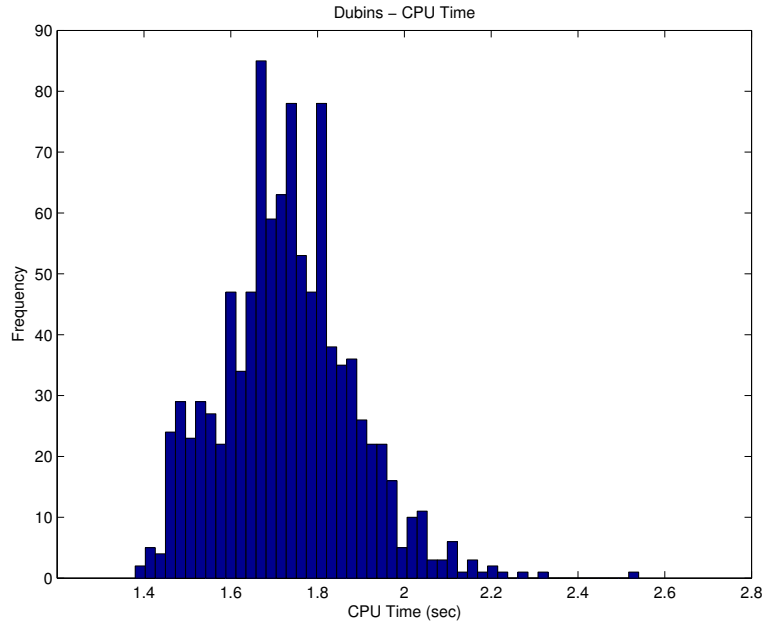


Figure 3.17: Total CPU time of CMPC using the Dubins car model over 1000 runs.

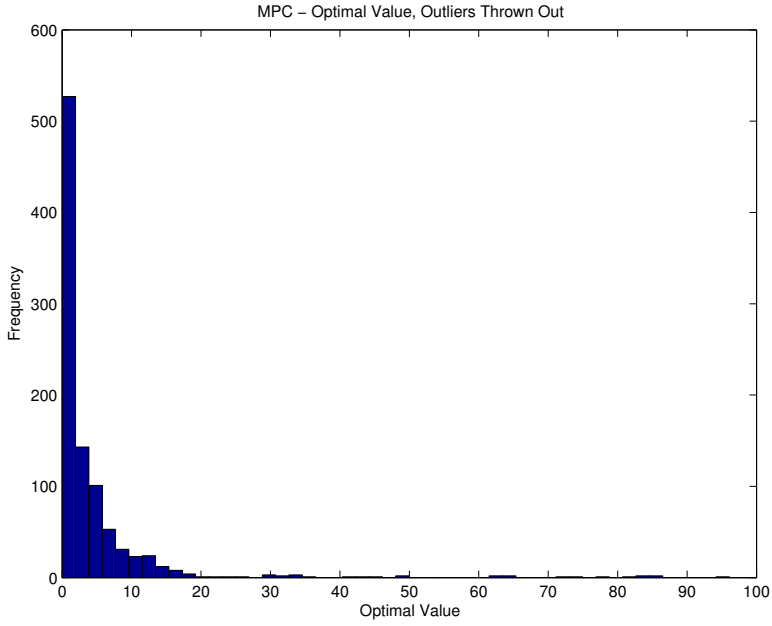


Figure 3.18: Optimal values of the MPC method over 1000 runs with heavy outliers thrown out.

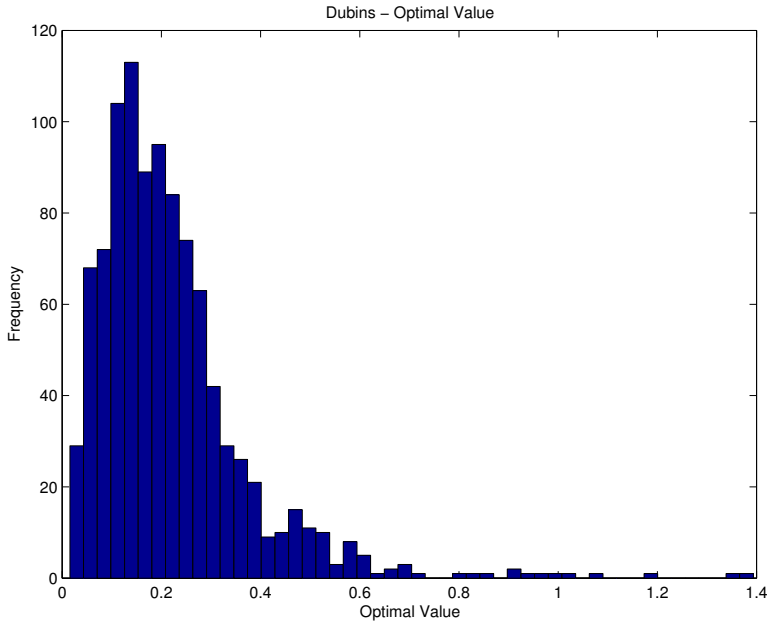


Figure 3.19: Optimal values of the convex relaxation method using the Dubins car model over 1000 runs.

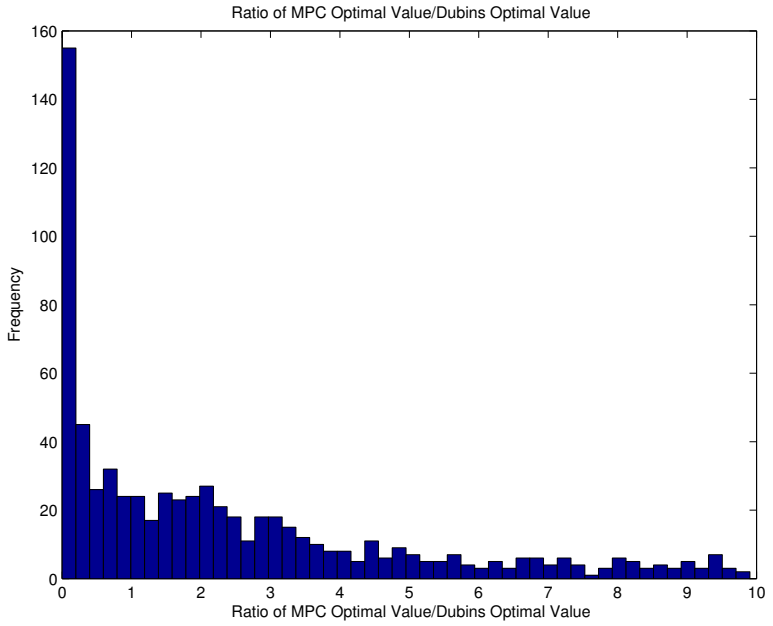


Figure 3.20: Ratio of MPC optimal value to Dubins optimal value over the subset of 1000 runs with ratios ≤ 10.

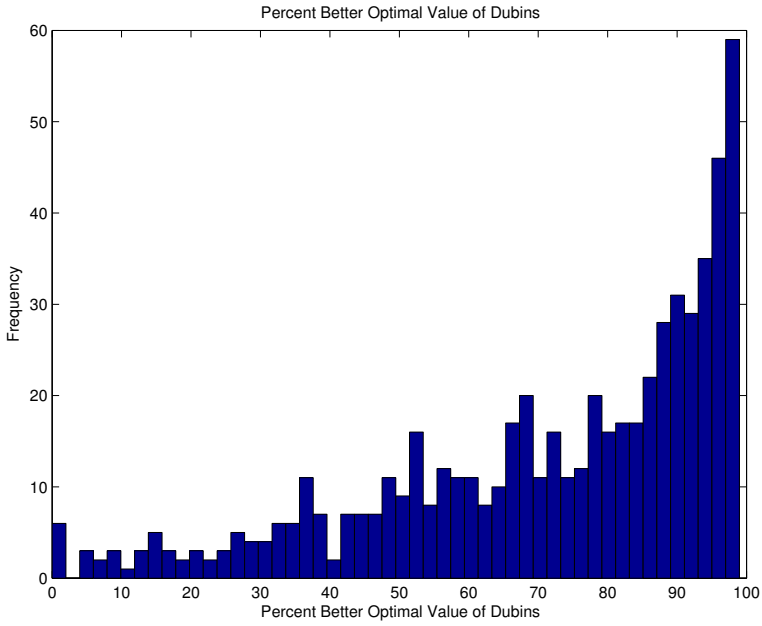


Figure 3.21: Increased performance of Dubins over MPC for the subset of 1000 runs with Dubins $\leq 100\%$ better.

3.9 Mixed-Integer Linear Programming (MILP)

Mixed-Integer constraints were added to the receding time horizon example in order to incorporate obstacle avoidance capabilities into both the MPC and convex relaxation examples. For simplicity and testing purposes, the initial simulations were performed using simple, rectangular obstacles. In both the receding time horizon (Figure 3.23) and MPC (Figure 3.22) cases, the physical obstacle was successfully avoided by introducing Mixed-Integer constraints as detailed in (2.10).

For the CMPC examples, an "obstacle" was also introduced within the convex hull of $SO(2)$ (Figure 2.1) to force the vehicle to maintain a certain minimum speed. By doing so, the possibility of the optimization suggesting that the vehicle reach 0 velocity mid-flight was eliminated. We demonstrate the effect of imposing a mini-

mum size on the determinant of R, also shown in the Figures 3.24 and 3.25. With the minimum speed constraint the solution took approximately one minute to compute on a 2.3GHz Intel i7 processor. The differences between enforcing a minimum speed and not including that restriction are shown in Figures 3.24 and 3.25. Introducing an obstacle in the convex hull of $SO(2)$ works as expected. The red dashed line in Figure 3.25 shows that the determinant of R in the Dubins car model example was forced to stay above the designated speed of $det(R) = \frac{1}{2}$.

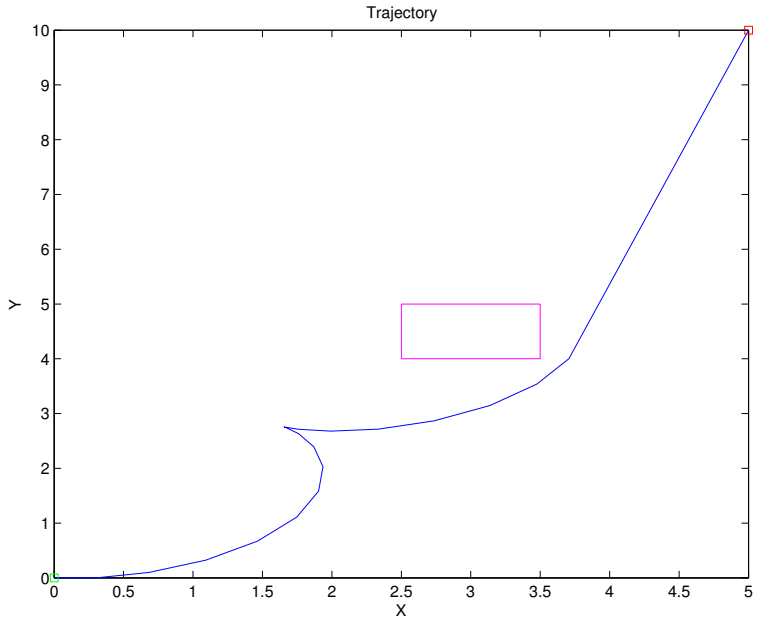


Figure 3.22: Trajectory generated by MPC method incorporating obstacle avoiding mixed integer constraints. The vehicle begins at $(x, y) = (0, 0)$ and stops at the goal (5, 10).

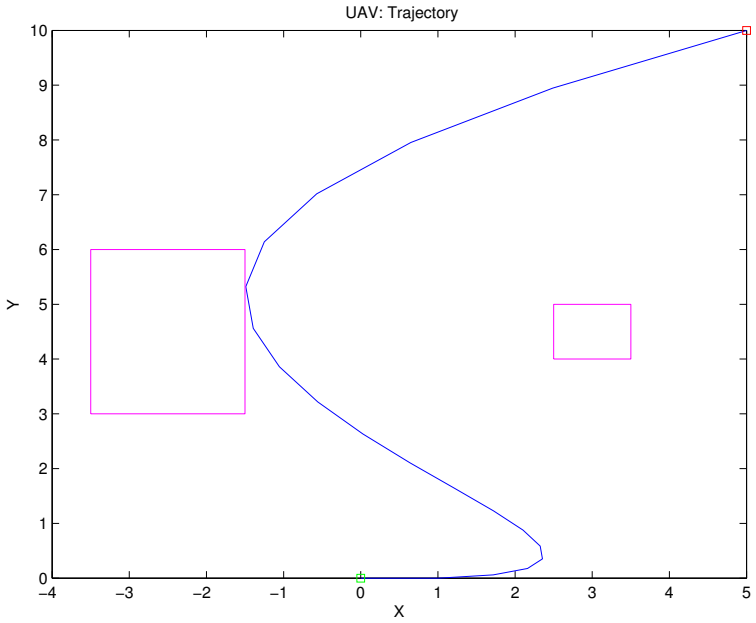


Figure 3.23: Trajectory generated by CMPC incorporating obstacle-avoiding Mixed-Integer constraints for both a physical obstacle and a convex hull obstacle. The vehicle begins at $(x, y) = (0, 0)$ and stops at the goal (5, 10).

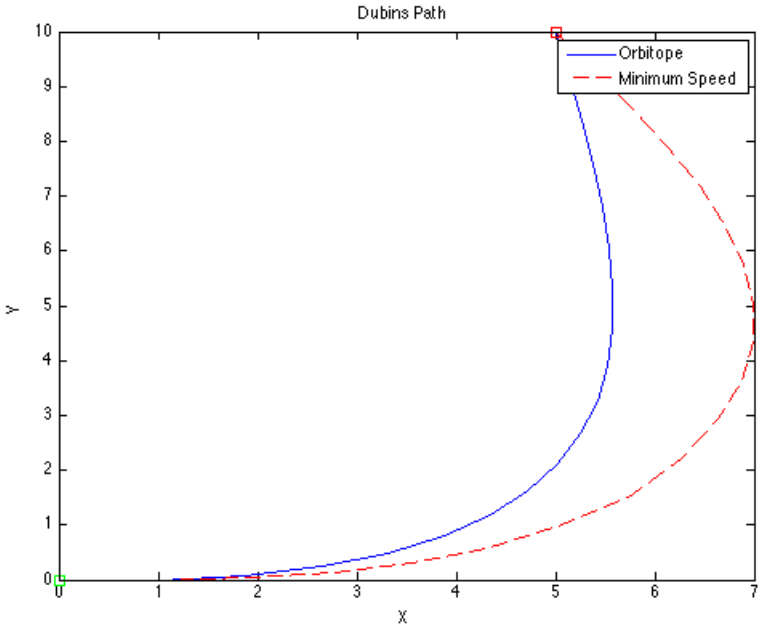


Figure 3.24: Example trajectories for the Dubins car with and without a minimum speed of $det(R) = \frac{1}{2}$, corresponding to the right of Figure 2.1. The trajectory begins at the origin and ends at $(x, y) = (5, 10)$.

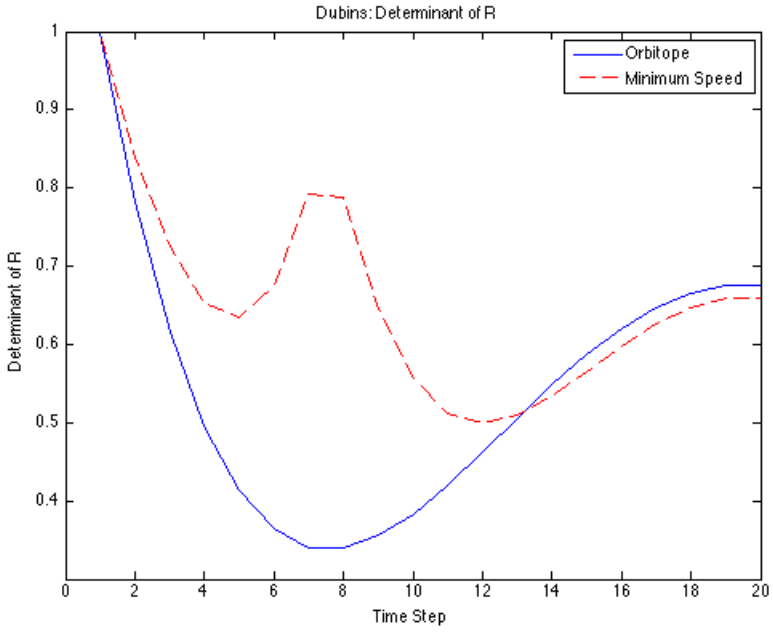


Figure 3.25: Determinant of the $SO(2)$ state R for the Dubins car trajectories shown in Figure 3.24 with and without the determinant constraint.

To simulate and show the applications of obstacle avoidance using MILP, a two-dimensional satellite docking example was introduced. In this example, a satellite needs to avoid collision with the spacecraft as it connects to a specific docking port on the spaceship. The satellite was given an initial position of $(x, y) = (-1, 5)$ and a docking goal point of (3.5, 6). Mixed-Integer constraints were used to represent the spacecraft as several obstacles that the satellite needed to avoid. The satellite was given 20 time steps to complete the mission. Given that integer constraints are needed for each obstacle at each time step, the number of integer constraints in this problem is very large. The computational time of MILP scales poorly with the number of constraints, which resulted in a program which took on the order of hours to run. The result shown in Figure 3.26 is the result of CMPC with Mixed-Integer

constraints after about 5 hours. The program was terminated at this time, so the trajectory shown is a sub-optimal solution. However, the algorithm is shown to allow the satellite to avoid the spacecraft successfully.

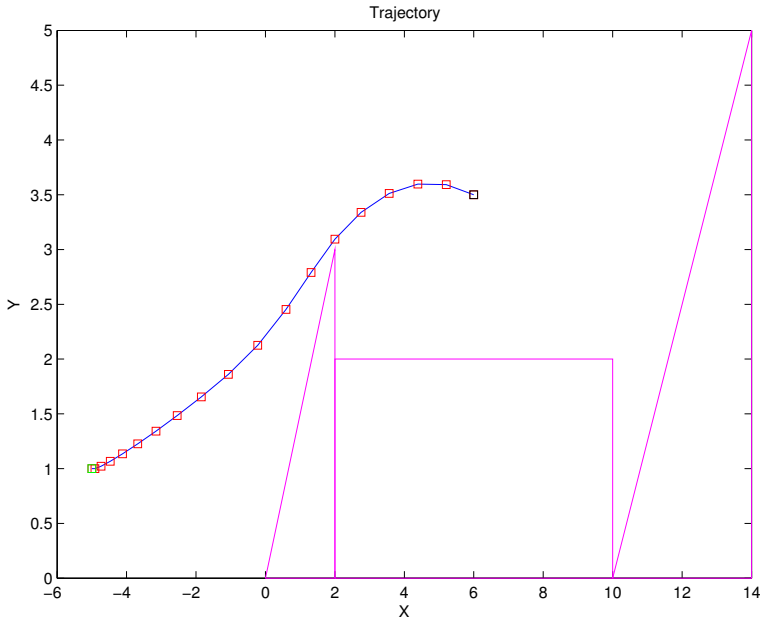


Figure 3.26: Trajectory for a satellite docking on a spacecraft using Mixed-Integer constraints to avoid collision with the spaceship.

Note: Current commercial solvers do not support Mixed Integer Semidefinite Programs, precluding the ability to include incorporate obstacles or minimum speed constraints for states that evolve on $SO(3)$. However, due to the growth in interest in semidefinite programs, such functionality is anticipated in the near future.

4 Conclusion

In this work a novel method to plan over systems with states in $SO(2)$ and $SO(3)$ has been proposed. The convex model predictive control method relies on the relaxation to the convex hull of the orbitopes of these groups. The results are convex programs that may be solved quickly to obtain a globally optimal solution. In direct comparison between CMPC and the traditional MPC, our method proves to produce more efficient, lower control cost solutions a large majority of the time without incurring additional computation cost. Furthermore, the implementation of CMPC is shown not to affect the Mixed-Integer and receding horizon variants of MPC, allowing for obstacle avoidance, a minimum speed, and a degree of robustness to be added to the formulation. Examples ranging from the Dubins car to a spacecraft with integrated dynamics have shown the success and wide applicability of the method.

Bibliography

[1] S.P. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. Society for Industrial and Applied Mathematics, 1997.

[2] S.P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[3] Nona Cheeks. Photo: A small, high-torque reaction/momentum wheel. `http://ipp.gsfc.nasa.gov/ft_tech_reaction_moment_whl.shtm`, November 2012.

[4] Lester E Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, pages 497–516, 1957.

[5] Christodoulos A Floudas. *Nonlinear and mixed-integer optimization: fundamentals and applications*. Marcombo, 1995.

[6] Michael Grant, Stephen Boyd, and Yinyu Ye. Cvx: Matlab software for disciplined convex programming, 2008.

[7] Christopher D Hall, Panagiotis Tsiotras, and Haijun Shen. Tracking rigid body motion using thrusters and momentum wheels. *Journal of the Astronautical Sciences*, 2003.

[8] Øyvind Hegrenæs, Jan T Gravdahl, and Petter Tøndel. Spacecraft attitude control using explicit model predictive control. *Automatica*, 2005.

[9] Matanya B Horowitz, Nikolai Matni, and Joel W Burdick. Convex relaxations of se (2) and se (3) for visual pose estimation. *arXiv preprint arXiv:1401.3700*, 2014.

[10] Jean-Claude Latombe. *Robot Motion Planning*. Springer, 1990.

[11] Richard M Murray, Zexiang Li, and Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. Boca Raton: CRC, 1994.

[12] Arthur Richards and Jonathan How. Mixed-integer programming for control. In *American Control Conference, 2005. Proceedings of the 2005*, pages 2676–2683. IEEE, 2005.

[13] Arthur Richards and Jonathan P How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *American Control Conference, 2002. Proceedings of the 2002*, volume 3, pages 1936–1941. IEEE, 2002.

[14] L.-L. Show, Jyh Ching Juang, Chen-Tsung Lin, and Y.-W. Jan. Spacecraft Robust Attitude Tracking Design: Pid Control Approach. In *Proceedings of the 2002 American Control Conference*, pages 1360–1365, 2002.

[15] Photo: Interplanetary trajectories. `http://www2.jpl.nasa.gov/basics/bsf4-1.php`, November 2013.

[16] Photo: An unmanned aerial vehicle (uav) scan eagle lands in the skyhook for recovery on the flight deck aboard the amphibious assault ship uss saipan, 2006.

[17] John T-Y Wen and Kenneth Kreutz-Delgado. The attitude control problem. *IEEE Transactions on Automatic Control*, 36(10):1148–1162, October 1991.