1 Introduction

There has been growing interest in cooperative group robotics [8], with potential applications in construction and assembly. Most of this research focuses on grounded or mobile manipulator robots, which have a long history of use in industry. However, more recent studies have introduced UAVs to this setting. For instance, [4] is one of the first studies to develop an autonomous aerial construction system: a team of quadcopters is used to assemble "cubic structures" composed of beams and connectors, and the team is evaluated by the time taken to assemble the structures. However, the authors describe diminishing returns with an increasing number of quadcopters, as they avoid collisions between them by enforcing a "serial" assembly line in which the quadcopters wait for an area to clear before entering. They note that the system needs path-planning algorithms in which quads could navigate simultaneously through the same area. Another study [1] also used UAVs for construction, this time planning mutually avoidant trajectories based on bounding boxes, and replanning if a collision is detected in the near future.

There is a varied number of approaches to path planning. Some are closer to graph-based methods such as A* or D*, or Rapidly-exploring Random Trees [1]. Others are methods for global optimization, including potential fields, Particle Swarm Optimization (PSO) [3], and evolutionary or genetic algorithms [6].

At its root, path planning is an optimization problem. Most problems in a noisy environment with complex constraints are highly nonconvex, and it can be difficult to develop theoretical guarantees of convergence or optimality. However, in some cases, the formulation of the constraints is mostly convex, and may be modified such that it can be approached with general convex solvers. This is done by [9], which formulate trajectory planning as a non-convex optimization problem, and develop an algorithm for solving it optimally.

This works will start with a detailed summary of the Successive Convexification algorithm developed in [9]. Then, it will outline the problem formulation, derivation, and results for a free-final-time trajectory planning problem and a problem with object avoidance for connected quadcopters.

2 Successive Convexification Algorithm

In general, non-convexity can arise from the objective function, from state or control constraints, or from nonlinear dynamics. The first is usually easy to manage by transferring the nonconvexity from the objective to the constraints through a change of variables. For the second case, some methods have been developed to convert them to convex constraints while guaranteeing an optimal solution [2]. Finally, successive convexification (SCVX), developed by Mao, Szmuk, and Acikmese in [9], is an algorithm for solving optimal control problems with nonconvex constraints or dynamics by iteratively creating and solving a sequence of convex problems. This algorithm is described below.

SCVX assumes a nonconvex equality or inequality constraint $f(x(t), u(t), t)$ for $0 \leq t \leq T$. The state trajectory $x : [0, T] \to R^n$ is assumed to be initialized to x_0 at time $t = 0$; the control input $u : [0, T] \to R^m$ is assumed to be Lebesque integrable over the time domain; and $f : R^n \times R^m \times R \to R^k$ is assumed to be Frechet-differentiable with respect to all arguments. Additionally, x and u are subject to constraints $x \in X$ and $u \in U$, where the sets $X \in R^n$ and

$U \in R^m$ are assumed to be convex and time-invariant. Finally, the cost $C(x, u)$ is assumed to be convex and Frechet-differentiable. Under these assumptions, the algorithm solves the following problem:

Non-Convex Optimal Control Problem (NCOCP): *Determine a control function u^* and a state trajectory x^* which minimize the functional $C(x, u)$ subject to the constraints:*

$$f(x(t), u(t), t) \leq 0 \qquad \forall t \in [0, T] \tag{1}$$

$$\begin{aligned} x \in X \qquad & \forall t \in [0, T] \\ u \in U \qquad & \forall t \in [0, T] \end{aligned} \tag{2}$$

Now, the authors use a first-order Taylor approximation of this nonconvex problem to create a sequence of convex problem, with some additional requirements to guarantee convergence and optimality. Let the solution of the previous iteration be x^{i-1}, u^{i-1}. Define

$$\begin{aligned} A(t) &= \frac{\partial}{\partial x} f(x(t), u(t), t) \mid_{x^{i-1}(t), u^{i-1}(t), t} \\ B(t) &= \frac{\partial}{\partial u} f(x(t), u(t), t) \mid_{x^{i-1}(t), u^{i-1}(t), t} \\ D(t) &= \frac{\partial}{\partial t} f(x(t), u(t), t) \mid_{x^{i-1}(t), u^{i-1}(t), t} \end{aligned} \tag{3}$$

Then, the new constraint defined by the first-order Taylor linearization, is

$$\begin{aligned} f(x(t), u(t), t) = f(x^{i-1}(t), u^{i-1}(t), t) &+ A(t) * (x(t) - x^{i-1}(t)) \\ &+ B(t) * (u(t) - u^{i-1}(t)) \\ &+ D(t) + \text{(higher order terms)} \end{aligned} \tag{4}$$

In conjunction with the other constraints of the problem, this becomes the ith iteration, and therefore defines a sequence of convex problems. However, there are two possible issues introduced by this linearization.

One such issue is *artificial infeasibility.* Namely, this sequence of problems can generate an infeasible problem, even if the original nonconvex problem was feasible. The authors mitigate this by introducing **virtual control** – a term that acts as an unrestricted, but heavily penalized, control input, turning constraint (3) into:

$$\begin{aligned} f(x(t), u(t), t) = f(x^{i-1}(t), u^{i-1}(t), t) &+ A(t) * (x(t) - x^{i-1}(t)) \\ &+ B(t) * (u(t) - u^{i-1}(t)) \\ &+ D(t) + \tau \end{aligned} \tag{5}$$

In effect, virtual control allows any state in the state space to be reached on a subsequent iteration, which removes infeasibility while still encouraging a solution which only "uses" the real control input.

The second issue is *approximation error*: the convex problem might be starkly different further from the point of approximation, potentially even being unbounded. To avoid divergence from the true problem due to this error, the authors enforce a **trust region** on the control input, which simply limits the change in the control vector between iterations:

$$||u(t) - u^{i-1}(t)||_2 \leq \mu \quad \text{for } 0 \leq t \leq T \tag{6}$$

Note that the trust region is not a variable; it is fixed for every iteration.

With these constraints, the authors define the sequence of convex problems, where the ith iteration is given by:

Convex Optimal Control Problem (COCP): *Determine a control function u^* and a state trajectory x^* which minimizes the functional $C(x,u) + \lambda(\sum \tau)$ subject to the constraints (2), (4), (5).*

After the solution to the ith problem is found, there is an evaluation step in which the original, non-convex cost for the solution is computed. A comparison of the convex and non-convex costs determines whether a certain iteration is accepted or rejected, the convergence of the algorithm, and the trust region size for the following iteration.

The trust region is updated depending on the accuracy of the approximation. If the approximation is close to the true (nonconvex) cost, the trust region is expanded; if it is sufficiently different, it may be reduced in size. Specific parameters are chosen by the user.

An iteration is accepted if the nonconvex cost decreases – in other words, if this trajectory is found to be a better solution for the original problem than the previous trajectory. Otherwise, the iteration is rejected.

Finally, the algorithm converges when the change predicted by the linearization is (close to) zero.

SCVX is guaranteed to converge, and is the solution after convergence is feasible, it is optimal. A proof of convergence in continuous time is given in [9].

3 Minimum-time Problem

The (discretized) physical dynamics of a trajectory are given by

$$p[i] = p[i-1] + v[i-1] * t + (a[i] + g) * \frac{t^2}{2}, \quad i \in [1, N]$$

where p is the position, v is the velocity, a is the acceleration (or control), g is the gravity vector, N is the number of points in the trajectory, and t is the time step (equal to the final time T divided by N).

For a fixed final time, and therefore fixed t, the dynamics are linear in the variables p, v, and a, and therefore are a convex constraint. So, the optimization problem defining the minimum-energy trajectory between defined start (p_1, v_1) and goal (p_N, v_N) states would be:

Convex Minimum-Energy Trajectory Problem (CMETP)

$$\begin{aligned}
\underset{p,v,a,\tau}{\text{minimize}} \quad & \sum s \\
\text{subject to} \quad & p[1] = p_1,\ v[1] = v_1 \\
& p[N] = p_N,\ v[N] = v_N \\
& ||v[i]||_2 \leq v_{max},\ \ \forall\, i = 1, \dots N \\
& ||a[i]||_2 \leq a_{max},\ \ \forall\, i = 1, \dots N \\
& p[i] = p[i-1] + v[i-1] * t + (a[i] + g) * \frac{t^2}{2},\ \ \forall\, i = 2, \dots N \\
& v[i] = v[i-1] + (a(i) + g) * t,\ \ \forall\, i = 2, \dots N \\
& ||a[i]||_2 \leq s[i],\ \ \forall\, i = 1, \dots N
\end{aligned}$$

This is a second-order cone program (SOCP), can can very quickly be solved with interior point methods. However, the applicability of this formulation is limited, as it depends on a fixed final time. Furthermore, as shown in Figure 1, the resulting trajectories vary largely with the final time.

To expand the applicability of this formulation, we will consider t as a variable, and will use SCVX to solve the non-convex, free-final-time problem.

First, compute the partials with respect to each variable.

$$\begin{aligned}
\frac{dp[i]}{dp[i-1]} &= 1 \\
\frac{dp[i]}{dv[i-1]} &= t \\
\frac{dp[i]}{da[i]} &= \frac{t^2}{2} \\
\frac{dp[i]}{dt} &= v[i-1] + (a[i] + g) * t
\end{aligned} \tag{7}$$

Now, we can construct the problem for the kth iteration of the SCVX algorithm. As before, there are constraints on the initial and final position, and upper bounds on the norms of the velocity and acceleration:

$$\begin{aligned}
& p[1] = p_1,\ v[1] = v_1 \\
& p[N] = p_N,\ v[N] = v_N \\
& ||v[i]||_2 \leq v_{max},\ \ \forall\, i = 1, \dots N \\
& ||a[i]||_2 \leq a_{max},\ \ \forall\, i = 1, \dots N
\end{aligned} \tag{8}$$

The linearized physical dynamics, with a virtual control term τ, are now:

$$\begin{aligned}
p[i] = p^{k-1}[i] &+ 1 * (p[i-1] - p^{k-1}[i-1]) + t^{k-1} * (v[i-1] - v^{k-1}[i-1]) \\
&+ \frac{(t^{k-1})^2}{2} * (a[i-1] - a^{k-1}[i-1]) \\
&+ (v^{k-1}[i-1] + (a^{k-1}[i] + g) * t^{k-1}) * (t - t^{k-1}) \\
&+ \tau[i]
\end{aligned} \tag{9}$$

for all $i = 2, \dots N$.

A similar derivation and constraint was done for velocity.

In this formulation, we can now solve the sequence of convex problems to convergence. The sequence of solutions is shown in Figure 2, and the converged solution is shown in Figure 3. Velocity and control information is shown in Figure 4.

4 Linked Trajectory Problem

An interesting extension of this algorithm is to solve for multiple, interrelated trajectories. More specificaly, a possible application to the study of UAVs for construction is to find trajectories of quadcopters that are carrying a rod. Suppose the previous constraints hold: initial and final positions are fixed, the velocity and acceleration are bounded, and the trajectories are constrained to satisfy physical dynamics. Then, if the positions of the two trajectories are given by p_1 and p_2, the nonconvexity is in the constraint

$$||p_1 - p_2||_2 = d$$

for some fixed rod length d.

We can linearize this constraint and solve the problem with SCVX.
The partials are:

$$\begin{aligned} \frac{\partial}{\partial p_1}(||p_1 - p_2||_2 - d) &= \frac{(p_1 - p_2)}{||p_1 - p_2||_2} \\ \frac{\partial}{\partial p_2}(||p_1 - p_2||_2 - d) &= \frac{(p_2 - p_1)}{||p_1 - p_2||_2} \end{aligned} \tag{10}$$

The constraint, with an included virtual control term τ, will be:

$$||p_1^{i-1} - p_2^{i-1}||_2 - d + \frac{(p_1^{i-1} - p_2^{i-1})^T}{||p_1^{i-1} - p_2^{i-1}||_2}(p_1 - p_1^{i-1}) + \frac{(p_2^{i-1} - p_1^{i-1})^T}{||p_1^{i-1} - p_2^{i-1}||_2}(p_2 - p_2^{i-1})] = \tau \tag{11}$$

The overall control problem includes the constraints described in the CMETP applied to two trajectories p_1 and p_2, a convex slope constraint, and the constraint above, with a minimum-energy objective. A solution to this problem is shown in Figure 5.

We can extend this formulation to consider obstacles. For example, suppose the environment contains a cylindrical obstacle centered at a point C with radius r. We would like to avoid collisions not only with the two quadcopters, but also with the line segment between them, at each point of the trajectory. The convex and nonconvex constraint derivations for this problem are described in Appendix A.

Since the constraint should only be applied for a line segment passing through p_1 and p_2, and not for the full line, we scaled the constraint by a windowing function. While a square window would have most accurately represented the constraint, it caused convergence difficulties. Instead, we used the smoother function, equal to 1 between p_1 and p_2, and a Gaussian on either side with a standard deviation of 0.2.

The solution to this problem is shown from two angles in Figures 6 and 7.

5 Discussion

Using successive convexification, both of the two problems that were described above led to intuitive results. The free-time problem converged to a solution in which the norm of the acceleration is touching its the upper bound, and travels along a straight line in the x-y plane. The linked-quadcopter problem successfully avoids the obstacle while keeping the distance between trajectories constant, and the slope relatively even.

On of the advantages of this method is that it is general, in that new (nonconvex) constraints in the original problem can be linearized and added to the formulation of the convex problem. Additionally, unlike many other methods, it has theoretical guarantees on convergence and optimality.

There are some potential difficulties that can arise with this algorithm. While it is guaranteed to converge, and the solution is proven to be optimal if it is feasible, it may also converge to an infeasible solution. Experimentally, this occurred in the case of the linked quadcopter problem for some parameter values. Additionally, the number of iterations to convergence also varies with parameters such as the trust region size, the tolerance, the factors in front of the trust region term, thresholds for trust region updates, and so on.

6 Conclusion

SCVX provides an approach for efficiently solving path planning problems, and can be used in real-time applications. It can address fixed- and free- final time problems, as well as problems with multiple interrelated trajectories. Future work may extend SCVX to a larger set of interrelated trajectories, or perhaps try applying the algorithm to path planning for manipulators.

A Derivative calculations

Finding the shortest distance from a point to a line:

Let p_1, p_2 be points in R^3 that define a line segment, and let $C \in R^3$ be the center of an obstacle of radius r. The line segment can be parameterized as $p_1 + (p_2 - p_1)t$ for $t \in [0, 1]$.

For a fixed t, the distance between the point $p_t = p_1 + (p_2 - p_1)t$ and C is

$$s(t) = ||p_1 + (p_2 - p_1)t - C||_2$$

To find $\min_t s(t)$, set the derivative to zero:

$$\frac{d}{dt}s(t) = \frac{(p_1 + (p_2 - p_1)\hat{t} - C)^T(p_2 - p_1)}{||p_1 + (p_2 - p_1)t - C||_2} = 0$$

Multiply by the denominator and simplify:

$$(p_1 + (p_2 - p_1)\hat{t} - C)^T(p_2 - p_1) = (p_1 - C)^T(p_2 - p_1) + ||p_2 - p_1||_2^2\hat{t} = 0$$

Therefore,

$$\hat{t} = \frac{(C - p_1)^T(p_2 - p_1)}{||p_2 - p_1||_2^2}$$

This (scalar) value of t corresponds to the point

$$p_1 + \frac{(C-p_1)^T(p_2-p_1)}{||p_2-p_1||_2^2}(p_2-p_1)$$

The distance between this point and C is

$$||p_1 + \frac{(C-p_1)^T(p_2-p_1)}{||p_2-p_1||_2^2}(p_2-p_1) - C||_2$$

Therefore, the final (nonconvex) constraint is

$$||p_1 + \frac{(C-p_1)^T(p_2-p_1)}{||p_2-p_1||_2^2}(p_2-p_1) - C||_2 - r \geq 0$$

Note that this is a constraint on the shortest distance from a point to a line, not to a line segment. Ideally, we would only like to impose this constraint if $\hat{t} \in [0,1]$.

Finding the linearized constraint:

To simplify notation, introduce the following expressions:

$$\begin{aligned}
x(p_1,p_2) &= \frac{(C-p_1)^T(p_2-p_1)}{||p_2-p_1||_2^2}\\
Y(p_1,p_2) &= p_1 + (p_2-p_1)\cdot x(p_1,p_2) - C\\
Z(p_1,p_2) &= ||Y||_2^2 - r^2
\end{aligned}$$

With this notation, the original, nonconvex constraint is

$$Z(p_1,p_2) \geq 0$$

Find the partial derivatives:

$$\begin{aligned}
\frac{dZ}{dp_1} &= 2Y \cdot \frac{dY}{dp_1}\\
\frac{dZ}{dp_2} &= 2Y \cdot \frac{dY}{dp_2}
\end{aligned}$$

$$\begin{aligned}
\frac{dY}{dp_1} &= (1 - x(p_1,p_2))I + p_2(\nabla_{p_1} x(p_1,p_2))^T\\
\frac{dY}{dp_2} &= x(p_1,p_2)I + (p_1-p_2)(\nabla_{p_1} x(p_1,p_2))^T
\end{aligned}$$

$$\begin{aligned}
\nabla_{p_1} x(p_1,p_2) &= \frac{(2p_1 - C - p_2)}{||p_2-p_1||_2^2} + \frac{2(p_2-p_1)^T(C-p_1)\cdot(p_2-p_1)}{||p_2-p_1||_2^4}\\
\nabla_{p_2} x(p_1,p_2) &= \frac{(C-p_1)}{||p_2-p_1||_2^2} - \frac{2(p_2-p_1)^T(C-p_1)\cdot(p_2-p_1)}{||p_2-p_1||_2^4}
\end{aligned}$$

The linearized constraint, including the virtual control term, will be

$$Z(p_1^{i-1}, p_2^{i-1}) + (p_1 - p_1^{i-1})\frac{dZ}{dp_1}|_{(p_1^{i-1}, p_2^{i-1})} + (p_2 - p_2^{i-1})\frac{dZ}{dp_2}|_{(p_1^{i-1}, p_2^{i-1})} + \tau \geq 0$$

where p_1, p_2, p_1^{i-1}, and p_2^{i-1} are short for $p_1[k], p_2[k], p_1[k]^{i-1}$, and $p_2[k]^{i-1}$ at each index $k \in [1, N]$ of points along the trajectory.

References

[1] D. Alejo et al. “Collision-Free 4D Trajectory Planning in Unmanned Aerial Vehicles for Assembly and Structure Construction”. In: *Journal of Intelligent Robotic Systems* 73.1 (2014), pp. 783–795.

[2] J. Carson B. Acikmese and L. Blackmore. “Lossless convexification of non-convex control bound and pointing constraints of the soft landing optimal control problem”. In: *IEEE Transactions on Control Systems Technology* (2013).

[3] Yong Bao, Xiaowei Fu, and Xiaoguang Gao. “Path planning for reconnaissance UAV based on Particle Swarm Optimization”. In: *2010 Second International Conference on Computational Intelligence and Natural Computing*. Vol. 2. 2010, pp. 28–32. DOI: `10.1109/CINC.2010.5643794`.

[4] Hugh Durrant-Whyte, Nicholas Roy, and Pieter Abbeel. “Construction of Cubic Structures with Quadrotor Teams”. In: *Robotics: Science and Systems VII*. MIT Press, 2012, pp. 177–184.

[5] M. McEvoy, E. Komendera, and N. Correll. “Assembly path planning for stable robotic construction”. In: *2014 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*. 2014, pp. 1–6. DOI: `10.1109/TePRA.2014.6869152`.

[6] Y. Volkan Pehlivanoglu. “A new vibrational genetic algorithm enhanced with a Voronoi diagram for path planning of autonomous UAV”. In: *Aerospace Science and Technology* 16.1 (2012), pp. 47 –55.

[7] Vincent Roberge, Mohammed Tarbouchi, and Gilles Labonté. “Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning”. In: *IEEE Transactions on Industrial Informatics* 9.1 (2013), pp. 132–141.

[8] Justin K Werfel, Kirsten Petersen, and Radhika Nagpal. “Distributed multi-robot algorithms for the TERMES 3D collective construction system”. In: Institute of Electrical and Electronics Engineers. 2011.

[9] Behcet Acikmese Yuanqi Mao Michael Szmuk. “Successive Convexification of Non-Convex Optimal Control Problems and Its Convergence Properties”. In: *IEEE Conference on Decision and Control* (2016).

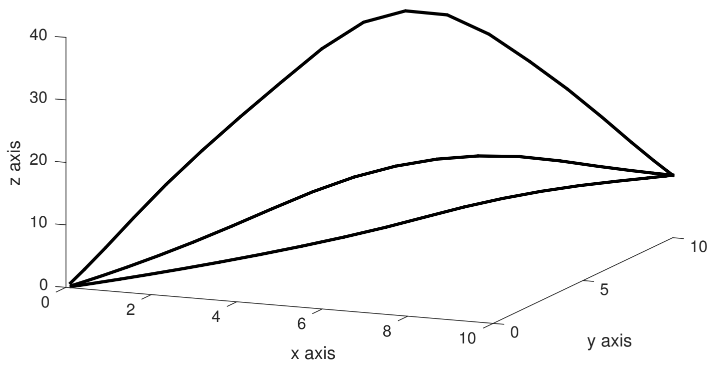

Figure 1: Trajectories for three possible values of fixed final times.

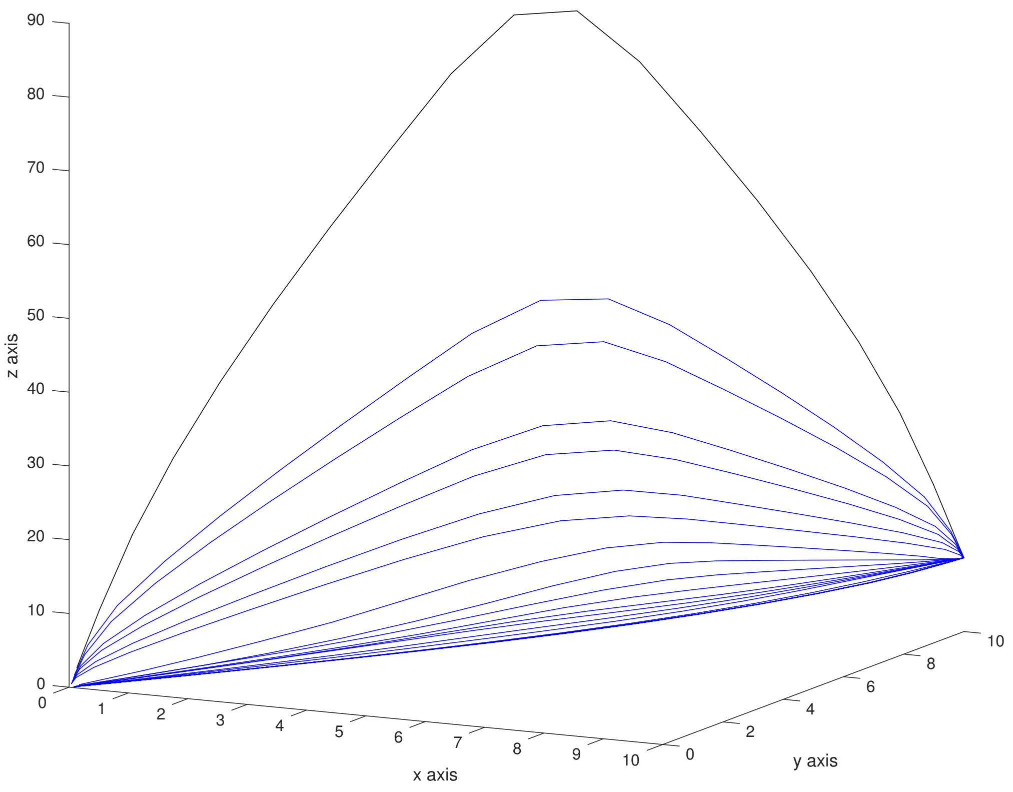

Figure 2: Trajectories across the iterations for the free-final-time problem. The top curve is the initial guess.

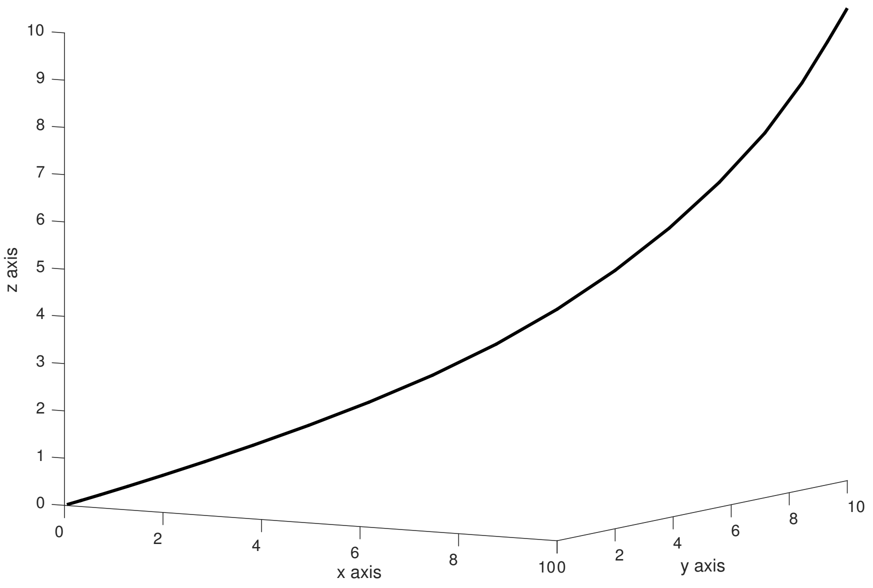

Figure 3: The solution for the free final time problem.

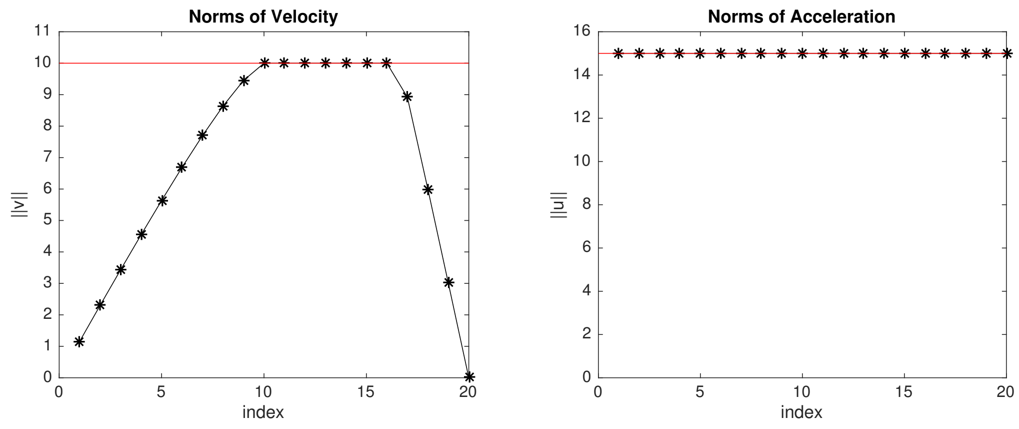

Figure 4: Plots of the norms of the velocity and acceleration for the free final time problem. The red lines indicate the upper bounds.

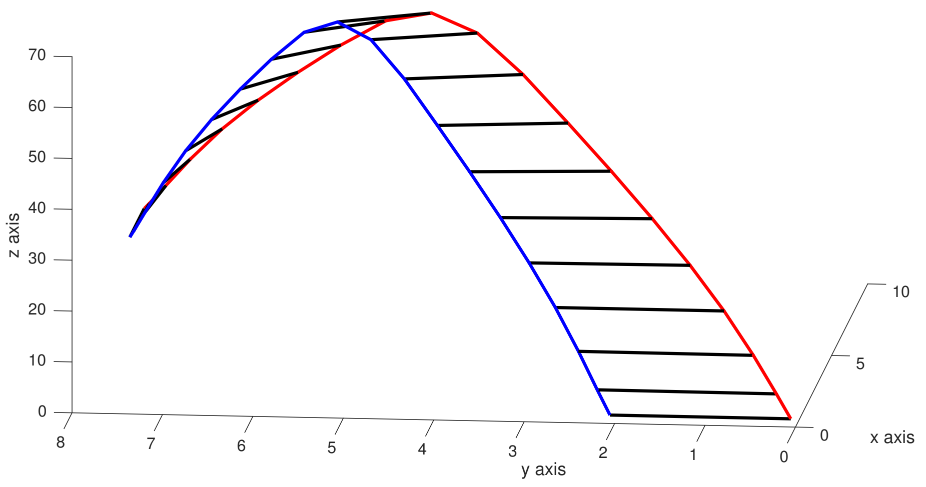

Figure 5: Plot of solution to the connected quadcopter problem. The red and blue lines indicate the individual trajectories, and the black lines indicate the rod at each time point.

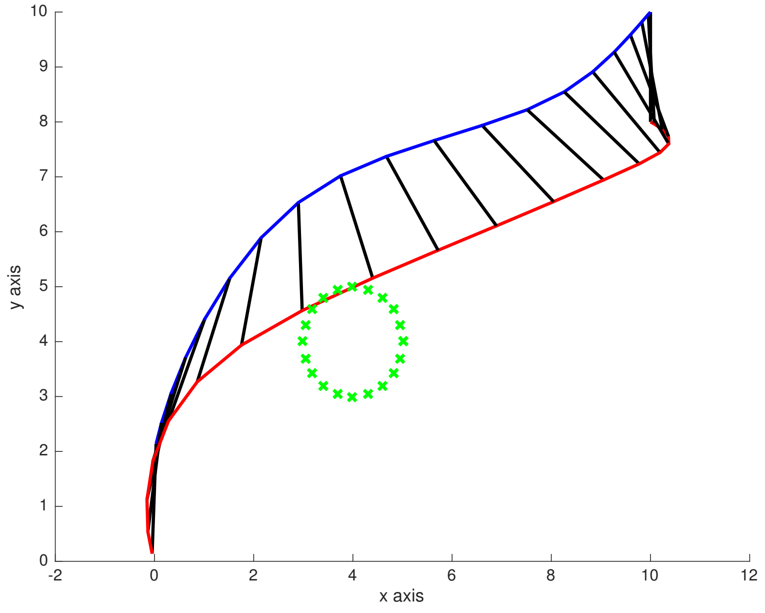

Figure 6: Plot of object avoidance for two connected quadcopters in the x-y dimension. The red and blue lines indicate the individual trajectories, and green circle represents the x-y projection of the cylindrical obstacle, and the black lines indicate the rod at each time point.

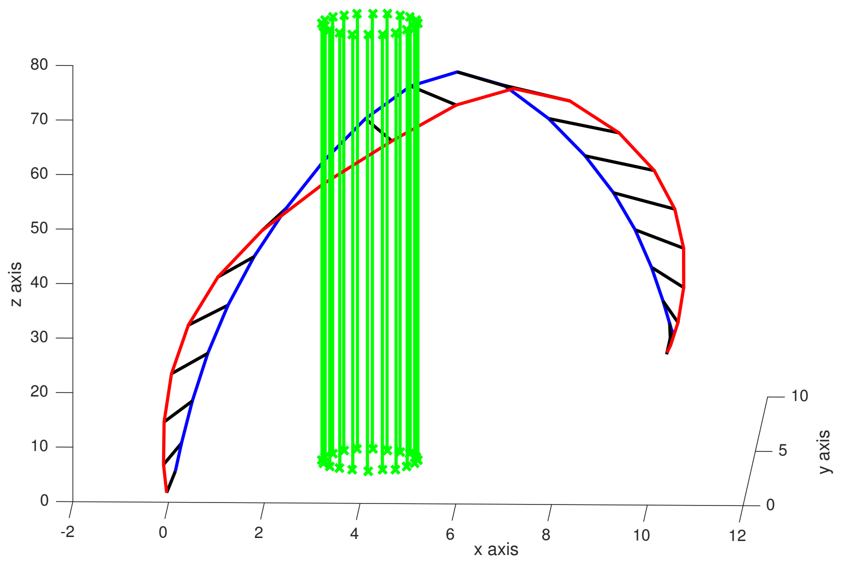

Figure 7: Plot of object avoidance for two connected quadcopters. The red and blue lines indicate the individual trajectories, the cylindrical obstacle is in green, and the black lines indicate the rod at each time point.