ACC01-AIAA1043

Trajectory Generation for a Class of Nonlinear Systems with Input and State Constraints*

S. K. Kim† and D. M. Tilbury‡
Department of Mechanical Engineering
University of Michigan, 2250 G.G. Brown, 2350 Hayward St.
Ann Arbor, MI 48109-2125
sungk@engin.umich.edu, tilbury@umich.edu

Submitted to the 2001 American Control Conference

Abstract

Autonomous vehicles with nonlinear dynamics need to have a planned reference trajectory and a feedback controller to accomplish the task of traveling from a launch point to a goal point. Traditionally the planning stage has either been done from a purely geometrical point of view without regarding the dynamic constraints of the system, or by time-consuming numerical optimization including the dynamic constraints and input bounds. This paper presents a new way to generate a trajectory quickly when a nonlinear system is input-output linearizable while satisfying the dynamic constraints. An example of a trajectory based on a simplified nonlinear longitudinal helicopter model with minimum time criteria is presented.

1 Introduction

Although unmanned aerial vehicles (UAV) have been in use for more than 30 years, their potential has proven to be limitless. They have been considered as an excellent cost-effective and safe way to replace human operators/pilots in military, civilian, and commercial areas when there exist significant threats to human lives or when the environment is not suitable for large human-carrying vehicles. For example, UAVs have proven to be highly effective for many areas including military reconnaissance missions at the front line without risking lives, surveying remote and hostile areas for commercial/civilian purposes, sports broadcasting, and more.

Since human pilots are not present on board a UAV, issues such as sensor design for object/obstacle and terrain recognition, radio link and data acquisition, autonomous navigation, and guidance must be carefully considered, designed, and programmed. The design must be reliable and proven to work given the constraints of the environment especially due to lack of immediate and flexible human intervention available on board. Vehicle design and sensor issues have been studied through university competitions [7] to accomplish tasks such as recognizing and moving an object to a designated target. Other successful UAV

*This research was supported in part by the Michigan Space Grant Consortium, the Rackham Graduate School, and the NSF under grant number CMS 98-76039.

†Ph.D. Student

‡Assistant Professor

works with commercial or military use include Sikorsky Aircraft's Cypher [18], Tokyo University's fuzzy-logic helicopter [16], and Bell helicopter's Eagle Eye Tiltrotor [8]. A survey of UAVs can be found in [9].

Although there exist many active research programs in regarding sensor performance, image recognition, and mechanical/electrical system design for UAVs, trajectory generation (path planning) and guidance research has been less popular. Traditionally, path planning methods have used computational geometry to find obstacle-free geometric paths that may not be feasible given the dynamics of the plant. The planned path is only constrained with respect to such factors as the terrain obstacles or hostile areas. Given such a path, the tracking task of the controllers (either an autonomous controller or a human pilot), can be difficult if not impossible. By considering the dynamics of the plant at the planning stage, the planned path or further the planned trajectory becomes less demanding for the controller to track. However, including the dynamic constraints at the planning stage often leads to a time-consuming nonlinear dynamic optimization that may not be practical especially if the dynamic model is complex. To address these issues, this paper presents a fast computational method to find feasible trajectories in the case when the nonlinear dynamic system is input-output linearizable.

The outline of the paper is as follows. First, we briefly review some previous work on path planning. Section 3 defines the trajectory generation method and gives conditions under which the method is guaranteed to find a solution. In section 4, we present the longitudinal mathematical model of a helicopter. We conclude with some discussion of the limitations of the proposed technique, and describe some future work in the area.

2 Previous Work on Path Planning

In general, there have been two different approaches to path planning: one is to consider the planning problem as a geometric constraint problem, and the other is to consider it as a dynamic constraint problem. Each method has pros and cons and we will briefly discuss them in this section.

2.1 Geometric Path Planning

Many researchers have devised ways to find an obstacle free path between two given points in a cluttered environment. The classic book by Latombe [13] describes many of these geometric planners such as the roadmap, cell decomposition, and potential field methods that provide a set of waypoints between the start and goal points which avoid obstacles, although no one method solves all kinds of problems. More recent work on geometric path planning includes Wang et al. [22], who used an algorithm to define and smooth out the boundary of obstacles. The algorithm is able to eliminate local minima ("corner" spots) to prevent the trajectory from getting stuck, a common problem encountered with the potential field method. A numerical optimization method (sequential quadratic programming) is then used to find the shortest path between the start and goal point. Vasudevan et al. [21] designed a case-based path planning system which used past route cases with an annotated map database. New segments are generated if suitable past plans are found. This strategy has a number of advantages such as fast computation, but is somewhat empirical and uses fuzzy sets when choosing a route. Jackson et al. [11] considered following predetermined discrete "waypoints" for aircraft landing maneuvers by optimizing cubic splines to minimize acceleration, curvature, or to obtain constant velocity. Simple linear models with pseudo inputs served as the aircraft model. Although the method suggests that the trajectory is more efficient than conventional straight-line flight trajectories, the advantage of the cubic splines may need more investigation due to its geometry oriented approach. The author points out for certain aircraft, the trajectory may be hard to follow, suggesting that the minimum turning radius of the target aircraft must be small enough to follow the path at all. The main limitation of these geometric planning methods is that they do not take the dynamic model of the vehicle into account; only a geometrically feasible path is given. Before it can be used by a dynamic system, the feasibility of the devised path needs to be investigated. In some cases, the geometric path may be simply impossible to follow for systems with certain dynamic constraints.

It is true that if the system is controllable, any geometric path can be approximated arbitrarily closely by a feasible path which satisfies the dynamic constraints [17, 19]. Nevertheless, since this method relies on a given geometric path as the targeting reference, the resulting feasible path may be awkward to follow for a real system. As shown in [19], the resulting path is extremely oscillatory when small position errors are required between a straight geometric path and the resulting dynamic path.

2.2 Dynamic Path Planning

In contrast to the geometric planning method, traditional numerical optimization planning methods directly include the dynamic system model while finding a path which optimizes a performance criteria. Realistic input/output and state bounds can be included in the optimization routine due to its numerical nature. Assuming the system model is accurate enough, the resulting path is optimized with respect to the given criteria and can be easily applied to a real system because it satisfies the dynamic constraints. Though variations exist, all methods utilize a Newton-based iteration to adjust a finite set of variables [4]. For example, Lee et al. [14] used a sequential gradient restoration technique to guide a helicopter toward a safe autorotation trajectory which minimizes touchdown velocity with bounds on rotor thrust and descent velocity when an engine failure occurs. The author points out that optimal solutions of this type to practical engineering problems can only be found numerically. Since the dynamic model of the system is included in the path generation routine, the scheme is fundamentally complicated and numerically intensive. Furthermore, analytical exact solutions of dynamical optimization problems are only possible when the system equations, the performance index, and constraints of the problem are very simple.

The optimization method described above determines the input necessary to satisfy the optimality criteria given certain constraints. The complexity of the numerical method can be relieved by assuming the input to be a parameterized sinusoidal, polynomial, or piece-wise constant function, then applying it to a system model transformed into a normal form [20]. Although the resulting trajectory is not optimal, an exact analytic result can be quickly obtained. However, bounds on states and inputs are not considered.

Some trial and error planning methods also exist. Arinaga et al. [1] devised an algorithm called a bi-directional motion planning method. First, the waypoints are determined by a global path planning which considers factors such as obstacle avoidance. Once they are determined, the real vehicle moves step by step towards a virtual vehicle which departs from goal point. When they meet, the real vehicle backtracks the path history of the virtual vehicle to reach the goal. This method naturally guarantees the feasibility (dynamic constraint) of the path, but the convergence is not guaranteed due to the hit-or-miss nature of the algorithm.

3 Trajectory Generation Methodology

The approach we will take is to assume that a standard geometric planner [13] gives a set of waypoints which avoid obstacles, and we will find feasible paths which satisfy the input and state constraints on the system and which connect these waypoints. We define the planning problem as follows.

Problem 1 *Given a controllable, nonlinear time-invariant system and initial and final states x_o, x_f*

$$\dot{x} = f(x(t), u(t)) \qquad x \in R^n,\ u \in R^m \tag{1}$$

Find a control input u (or equivalently, find a trajectory η) which takes the system from the initial position $x_o = x(0)$ to final (goal) position $x_f = x(t_f)$

$$\eta = \{(x(t), u(t)) \in R^n \times R^m : 0 \le t \le t_f\} \tag{2}$$

and satisfies the state and input bounds

$$x_{l_i} \leq x_i \leq x_{u_i}, \;\; u_{l_j} \leq u_j \leq u_{u_j} \tag{3}$$

where all of the inputs are constrained but only some of the states have bounds:

$$i \subseteq \{1, \cdots, n\}, \;\; j = \{1, \cdots, m\}$$

Basically, the problem is to find a feasible trajectory of the system (1) from an initial state x_o to a final state x_f where the final time t_f may or may not be defined a priori. If there exists such a path η, then there exist many such paths. If possible, we would like to find one which is optimal is some sense, and assume that a meaningful cost function of the form J can be defined for the system. This leads us to the next problem.

Problem 2 *Given (1), (3), x_o, and x_f, find a control input u (or trajectory η) which connects the start to the goal, satisfies the input and state bounds, and minimizes a given cost function J*

$$J = \gamma(x(0), x(t_f)) + \int_0^{t_f} L[x, u]\, dt$$

Again, the final time t_f may be given or free (as in a minimum time optimization problem). This problem is often called a two point boundary value optimization problem [5]. The solution to this problem is a trajectory η describing the states and inputs over the interval $[0, t_f]$. However, solving this problem analytically is not practical for any except very simple systems. Numerical solution methods can be unstable and/or time-consuming depending on the method used. Most numerical methods require a good initial guess of the solution, and may not guarantee that the optimal solution is found [4]. However, for linear systems, much better optimization techniques exist. Consider the special case when the system model can be expressed as a set of linear ODE's as in the next problem.

Problem 3 *Given a system expressed in the following form with initial and final states $Y(0)$ and $Y(t_f)$*

$$\begin{aligned} y_1^{(k_1)} &= v_1 \\ &\vdots \\ y_m^{(k_m)} &= v_m \end{aligned} \tag{4}$$

with state dimension $w = k_1 + \cdots + k_m$

$$Y = \left[\begin{array}{ccccccccccc} y_1 & \dot{y}_1 & \cdots & y_1^{(k_1-1)} & y_2 & \dot{y}_2 & \cdots & y_2^{(k_2-1)} & \cdots & y_m & \dot{y}_m & \cdots & y_m^{(k_m-1)} \end{array} \right] \in R^w$$

Find a control input v (or equivalently, find a trajectory η_Y) which takes the system from the initial position $Y_o = Y(0)$ to final (goal) position $Y_f = Y(t_f)$

$$\eta_Y = \{(Y(t), v(t)) \in R^w \times R^m : 0 \leq t \leq t_f\} \tag{5}$$

satisfies the state and input bounds,

$$\begin{aligned} Y_{l_i} \leq Y_i \leq Y_{u_i} &\qquad i \subseteq \{1, \cdots, w\} \\ v_{l_j} \leq v_j \leq v_{u_j} &\qquad j = \{1, \cdots, m\} \end{aligned} \tag{6}$$

and minimizes the cost function

$$J_L = \gamma(Y(0), Y(t_f)) + \int_0^{t_f} L[Y, v]\, dt$$

A solution to this problem for the special case where $J_L = t_f$ (minimum-time problem) is given by Baker [3, 2]. Although it seems that finding the time-optimal solution for the linear system (4) should be simple enough, according to Baker, there is no known analytic method to solve this problem when input and state bounds are given together. However, this optimization problem can be solved using the differential inclusion technique [15], which discretizes the equations (4) and (6) and then applies either linear or nonlinear programming (LP/NLP) to minimize the cost function J_L. If the cost function J_L is linear in the states and inputs, then LP can be used to minimize it; otherwise, such as when a quadratic cost is used, NLP is required.

LP/NLP is an effective technique when the constraints are linear as in the minimum time problem with inequality constraints on states and inputs [4]. Discretizing the dynamic equations (4) at N timesteps between 0 and t_f, we rewrite J_L as

$$\begin{aligned} J_L &\approx \gamma(Y(0), Y(N)) + \sum_{i=0}^{N} L[Y(i), v(i)] \frac{t_f}{N+1} \\ &= \mathcal{F}(\Upsilon) \text{ or } c^T \Upsilon \end{aligned}$$

where

$$\Upsilon = \{ \; y_1(0) \;\; \cdots \;\; y_1(N) \;\; y_2(0) \;\; \cdots \;\; y_2(N) \;\; \cdots \;\; y_m(0) \;\; \cdots \;\; y_m(N) \; \} \in R^{1 \times m(N+1)}$$

Using differential inclusion scheme which discretizes the equation (4):

$$\dot{y}(i) \approx \frac{y(i+1) - y(i)}{\Delta t} \tag{7}$$

the derivatives of y ($\dot{y}, \ddot{y}, \ldots$) and the inputs v can be written as finite differences of the outputs y and hence as functions of Υ. The inequality state and input constraints (6) can thus be expressed as

$$A_1 \cdot \Upsilon^T \leq b_1$$

and the state boundary conditions (initial and final positions) are written in the form

$$A_2 \cdot \Upsilon^T = b_2$$

There are advantages for using LP/NLP over the nonlinear optimization via exact or slack variable methods. LP/NLP often provides a solution more easily, because it deals with bounds (inequality constraints) more efficiently [5]. If the cost function J is a linear function of the states, LP should be used due to its faster computation times.

At this point, remember that the objective of the planning methodology is to obtain a solution for Problem 2. We consider the special case of this problem when the nonlinear system is input-output linearizable. That is, there exists a coordinate transformation from the states, inputs, and an appropriate number of input derivatives into a linear form:

$$\Phi(x, u') = (Y, v)$$

where u' is a vector consisting of of the original inputs u and the derivatives of u that are needed for the input-output linearization [10]. After linearization, the original equations (1) can be expressed as (4). We will use the optimal solution to the linearized system (Problem 3) to get an approximately optimal solution to the nonlinear problem. Because of the coordinate transformation, we can only approximate the bounds, and the solution will not be globally optimal. However, the solution to the linear problem is fast and easy to compute.

The bounds on (Y, v) used in Problem 3 must be carefully chosen so that when the resulting trajectory is transformed back into the original coordinates (x, u), the given bounds are not violated. However, overly conservative bounds for (Y, v) would result in an overly conservative solution which is far from optimal.

The original bounds on the states and inputs given by (3) can be considered as a domain D_x:

$$D_x = \{(x, u) : x_{l_i} \leq x_i \leq x_{u_i}, u_{l_j} \leq u' \leq u_{u_j}\}$$

where $i \subset \{1, \cdots, n\},\ \ j = \{1, \cdots, p\}$. Before mapping this domain through the transformation Φ, bounds on the input derivatives must be determined; these will come from the application specification. The domain D_x can then be mapped into $D_L = \Phi(D_x)$. However, due to the nonlinearity of the transformation Φ, this domain cannot typically be described as a set of inequalities. Thus, we want to find a smaller domain $D_2 \subseteq D_L$ which can be described by a set of inequalities. Using this domain in the description of Problem 3, we can find a trajectory η_Y optimizing J_L in the (Y, v) coordinates. The inverse transformation of this trajectory will be guaranteed to satisfy the original bounds, because

$$\Phi^{-1}(D_2) \subset D_x$$

although it may not satisfy the optimality criteria even if J_L is an exact representation of J in the linear coordinates. See Figure 1 for a pictorial description of these domains.

Through numerical experimentation, we determined that choosing $D_2 \subseteq \Phi(D_x)$ results in overly conservative bounds and often no solution. Due the nonlinearity, direct computation of $\Phi(D_x)$ is time-consuming and numerically expensive. Hence, we adopt instead an approximate technique to choose the domains D_Y to be used in the linear optimization problem. The following algorithm formalizes this method. After the algorithm, sufficient conditions are given for a solution to be returned. An example application of this algorithm can be found in Section 4.

Algorithm 1 (Approximately Optimal Solution to Problem 2) *Suppose that the system model (1) is input-output linearizable with choice of output y. Let Φ denote the transformation from (x, u') to (Y, v).*

Step 1: Determine appropriate bounds for derivatives of u from the system dynamics and available control input (this is not given in the original problem formulation).

Step 2: Find points or ranges of values within D_x at which the transformation Φ is singular [10]. Excluding these singular values, choose a compact set within D_x which includes the initial condition (x_o, u_o) and can be described by a set of inequalities. Call this set D'_x, and the corresponding upper and lower bounds $x'_{l,u}, u'_{l,u}$.

$$D'_x = \{(x, u') \subseteq D_x : \left|\frac{\partial \Phi}{\partial (x, u')}\right| \neq 0, x'_{li} \leq x_i \leq x'_{ui}, u'_{lj} \leq u'_j \leq u'_{uj}\}$$

where $i \subset \{1, \cdots, w\},\ \ j = \{1, \cdots, m\}$

Step 3: Find a set of bounds for (Y, v). For each pair of upper and lower bounds, $x'_{l,u}, u'_{l,u}$, and for each i, compute Y_i and v_i through Φ, and find their maximum and minimum values. If the upper and lower bounds are the same for one or more states or inputs, find stationary points with respect to all other states and inputs by taking partial derivatives; denote these "intermediate points" by x'_m and u'_m. Also take the maximum and minimum of Y_i and v_i over these intermediate points.

Formally, let the set $h = \{x'_l,\ x'_u,\ x'_m,\ u'_l,\ u'_u\ , u'_m\}$ where $(x'_m, u'_m) = \{(x, u') : \frac{\partial(Y,v)}{\partial(x,u')} = 0\}$. Define the upper and lower bounds on (Y, v) as:

$$\begin{aligned} Y_{u_i} &= \max_h Y_i \\ Y_{l_i} &= \min_h Y_i \\ v_{u_j} &= \max_h v_j \\ v_{l_j} &= \min_h v_j \end{aligned}$$

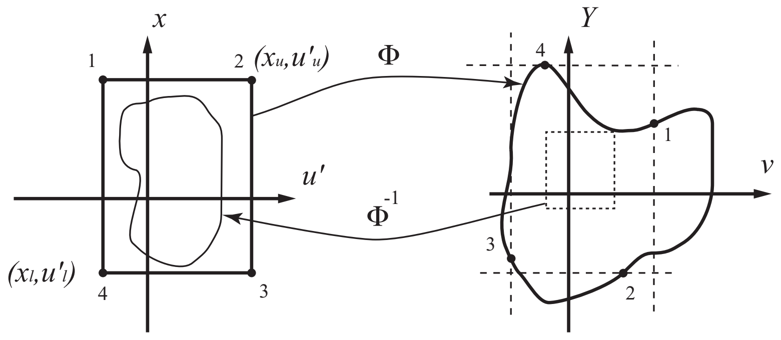

Figure 1: A conceptual diagram showing the transformation between (x, u') and (Y, v). The large dashed box in the right picture is the initial domain D_L which is eventually shrunk to the smaller box inside by continually reducing α. The smaller box is transformed back to (x, u') while satisfying the bounds given to the original states and inputs.

where $i \subset \{1, \cdots, w\}$, $j = \{1, \cdots, m\}$.

Step 4: Let $D_Y = \{(Y, v) : Y_{l_i} \le Y_i \le Y_{u_i},\ v_{l_j} \le v_j \le v_{u_j}\}$*, where* $i \subset \{1, \cdots, w\}$, $j = \{1, \cdots, m\}$ *on* D_Y*. Solve problem 3 using (non)linear programming and the differential inclusion technique.*

Step 5: Transform the solution obtained in Step 4 back into the (x, u) *coordinates via* Φ^{-1}*. Check whether the bounds on* (x, u) *are satisfied. If so, stop.*

If not, reduce the size of D_Y *as follows. This will reduce the size of the domain from all sides the equal amount.*

$$\begin{aligned} (Y, v)_u &\leftarrow (Y, v)_u - [(Y, v)_u - (Y, v)_l]\,\alpha \\ (Y, v)_l &\leftarrow (Y, v)_l + [(Y, v)_u - (Y, v)_l]\,\alpha \end{aligned}$$

where $0 < \alpha < 1/2$*. Choose an appropriate scale factor* α*, starting with 0 then gradually increasing toward 1/2 until a feasible solution is obtained or* $\alpha = 1/2$*.*

Go back to Step 4.

The following proposition gives sufficient conditions for the algorithm to terminate successfully.

Proposition 1 *If there exists a hypercube* Ψ *completely enclosed within the set* $\Phi(D_x)$*, then Algorithm 1 produces a solution to problem 2, although it may not be optimal.*

Proof: Since $\Psi \subset \Phi(D_x)$, and Φ is a diffeomorphism, we must have $\Phi^{-1}(\Psi) \subset D_x$. If such a hybercube exists within $\Phi(D_x)$, the reduction in Step 5 will eventually converge to a subset of this hypercube. A solution to Problem 3 which satisfies the constraints given by the hybercube Ψ will be transformed by Φ^{-1} to a solution to the nonlinear system which satisfies the constraints given by D_x. Because only a subset of the input and state bounds are used in the linear optimization, optimality in the nonlinear system cannot be guaranteed.

This proof basically guarantees the inverse transformation of the hypercube is indeed a valid trajectory in the original (x, u) coordinates. Therefore, an optimal trajectory obtained from Problem 3 with constraints based on Ψ, transformed back to (x, u) coordinates, will satisfy the constraint requirement for Problem 2. Unfortunately, the method presented here does not guarantee that such a hypercube exists. The method fails when $\alpha \to 1/2$ and $\Phi^{-1}(D_Y) \not\subseteq D_x$. Figure 2 shows various possibilities for D_Y.

For many practical problems, however, getting a reasonable solution quickly is often sufficient. The method presented in this section should help find such a solution. In addition, if the sufficient conditions of Proposition 1 are satisfied, then the same set of linear constraints Ψ and the same linear programming setup

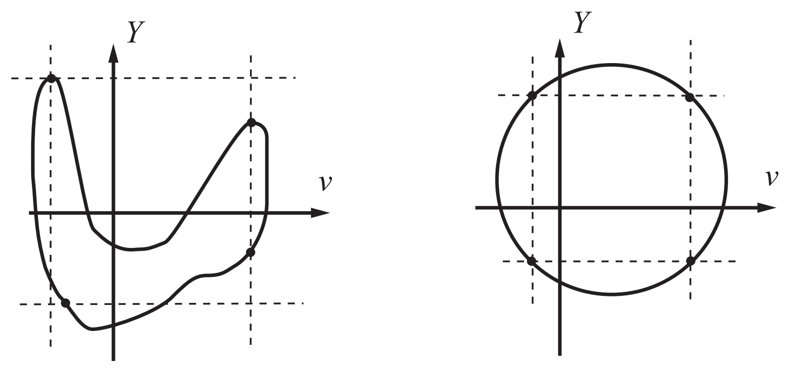

Figure 2: Two extreme cases for initial D_L. The left figure is the case where one cannot find a square within the initial domain D_L even when $\alpha = 1/2$. The right figure is the case when a solution will be found on the first trial with $\alpha = 0$.

can be used to solve a family of path planning problems. The linear program will need to be solved for each new start and goal position of the system, but the input-output linearization and the algorithm to find the linear bounds needs only to be executed once.

Furthermore, the result of this path planning methodology can be used as an initial guess towards a more accurate numerically optimal solution. The trajectory we obtain here can also be directly applied to real-time planning applications due to its fast computation.

In summary, the key ingredient in Algorithm 1 is transforming the original nonlinear system expression into a simple linear expression such that a optimal trajectory in the linear coordinates can be easily obtained. That is, if the given nonlinear system (1) is input-output linearizable, we can obtain an approximate solution for Problem 2 by first transforming it into a linear form (4), then solving Problem 3 using Algorithm 1, and finally transforming the linear solution back to the original (x, u) coordinates. One shortcoming of Algorithm 1 is that the exact optimization result from the Problem 3 does not imply that the optimal solution will be obtained for Problem 2. We therefore call this an approximate optimization. Another limitation is that even a linear cost function J may be nonlinear when transformed through the diffeomorphism Φ. However, a minimum time problem $J = t_f$ will remain linear in both coordinate systems.

Since we are attempting to satisfy the cost function J_L after the transformation, we need the states of interest to be optimized as a function of Y. Therefore, optimization problems such as minimum system input may be difficult to pose correctly in this framework.

4 Helicopter Trajectory Generation

Currently, unmanned autonomous vehicles such as model-scale helicopters are often considered suitable for reconnaissance missions in hostile areas due to their versatile maneuverability. The mission criteria often require the vehicle to follow a given trajectory in minimum time while avoiding obstacles such as enemy radar or terrain. As an example, given start/goal positions and suggested way points that are free of obstacles between any two points, we could use the method presented in Section 3 to design a trajectory to guide the helicopter to satisfy the mission requirement.

The model helicopter is capable of moving in 3-dimensional space between any two points. It is an under-actuated system with 6DOF and 4 inputs, therefore it has a non-holomonic constraint due to the coupling between the rotational and the translational motions [12]. For example, while at hover, the helicopter cannot move forward unless the body is first tilted forward.

In this section, we consider a simplified longitudinal model of the helicopter with a hingeless rotor. The model is written with respect to the earth-fixed inertial coordinates. The horizontal and vertical positions are x and z; the pitch angle is θ. The two inputs, rotor thrust and pitch moment, are T and τ. The dynamics of the pitch damper, called a flybar, is modeled as $\dot{\theta} = \pi\tau$. The constant π represents that the helicopter is capable of 180° rotation/sec when the maximum moment input is applied. The mass of the helicopter m is 1.4 kg and the gravity acceleration constant g is 9.8 m/s^2.

$$\begin{aligned} \dot{x} &= v_x \\ \dot{v}_x &= \frac{T}{m}\sin\theta \\ \dot{z} &= v_z \\ \dot{v}_z &= \frac{T}{m}\cos\theta + g \\ \dot{\theta} &= \pi\tau \end{aligned} \tag{8}$$

with input and state bounds

$$\begin{aligned} -2mg \leq {}& T \leq 0 \\ -1 \leq {}& \tau \leq +1 \\ -\pi/2 < {}& \theta < +\pi/2 \end{aligned}$$

This nonlinear model is input-output linearizable with outputs (x, z). The choice of these outputs is not unique and unfortunately there is no known method to choose outputs which will result in an input-output linearized model.

Taking the third derivatives of each output, we get

$$\frac{d^3}{dt^3}\begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} \frac{\sin\theta}{m} & \frac{T\cos\theta}{m} \\ \frac{\cos\theta}{m} & \frac{T\sin\theta}{m} \end{bmatrix}\begin{bmatrix} \dot{T} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{\sin\theta}{m} & \frac{\pi T\cos\theta}{m} \\ \frac{\cos\theta}{m} & \frac{\pi T\sin\theta}{m} \end{bmatrix}\begin{bmatrix} \dot{T} \\ \tau \end{bmatrix}$$

We then have an input transformation:

$$\begin{bmatrix} \dot{T} \\ \tau \end{bmatrix} = \begin{bmatrix} \frac{\sin\theta}{m} & \frac{\pi T\cos\theta}{m} \\ \frac{\cos\theta}{m} & \frac{\pi T\sin\theta}{m} \end{bmatrix}^{-1} V \tag{9}$$

which results in the input-output linearized form for the longitudinal helicopter model:

$$\dot{Y} = \left[\begin{array}{ccc|ccc} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] Y + \left[\begin{array}{c|c} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ \hline 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{array}\right] V$$

where

$$\begin{aligned} Y &= \begin{bmatrix} x & \dot{x} & \ddot{x} & z & \dot{z} & \ddot{z} \end{bmatrix}^T \\ V &= \begin{bmatrix} x^{(3)} & z^{(3)} \end{bmatrix}^T \end{aligned}$$

The actual inputs to the system, T and τ, can be derived from equation (9). The thrust T can be found by integrating $\dot{T}$.

Based on the helicopter model presented, we will show how the method presented in Section 3 can be used to generate a trajectory. Consider the situation when the helicopter starts from a hover, moves $100m$ in both the x and z directions, and then hovers again:

$$\begin{aligned} Y(0) &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \\ Y(t_f) &= \begin{bmatrix} 100 & 0 & 0 & -100 & 0 & 0 \end{bmatrix}^T \end{aligned}$$

The optimization criteria is to minimize the time required for this maneuver, given the input and state constraints. We follow through the steps presented in the last section.

Step 1: The linearization yielded $V_1 = \dot{T}$, which is the derivative of the input T. We choose the bounds for $\dot{T}$ as

$$-2mg \leq \dot{T} \leq +2mg$$

The choice assumes the rotor can go from zero to full thrust in 1 second. This quantity should indeed be finite due to physical limitations of the helicopter.

Step 2: We calculate the Jacobian of Φ, and its determinant should be non-zero to be nonsingular. That is,

$$\frac{\pi T^4}{m^4} \neq 0$$

This suggests that when $T = 0$, the transformation Φ^{-1} is not defined. The initial inputs with respect to the choice of initial condition given previously are

$$T(0) = -mg, \ \ \tau(0) = 0, \ \ \dot{T}(0) = 0$$

We should choose input bounds which exclude $T = 0$ but include the above initial input values. They are

$$\begin{aligned} -2mg \leq \ & T & < 0 \\ -2mg \leq \ & \dot{T} & \leq +2mg \\ -1 \leq \ & \tau & \leq +1 \end{aligned}$$

Step 3: We find the upper and lower bounds of Y and V with respect to the input bounds from step 2 and the state bound $|\theta| < \pi/2$.

$$Y_{l_3} = -2g, \ \ Y_{u_3} = 2g$$

$$V_{l_1} = -2g, \ \ V_{u_1} = 2g$$

$$V_{l_2} = -2g\pi, \ \ V_{u_2} = 2g\pi$$

We realize $Y_{l_6} = Y_{u_6} = 0$. We take partial derivatives of $Y_6 = \ddot{z}$ with respect to θ and T, to find the stationary point $\theta = 0$ within the bounds of θ and T, and substitute into Y_6 to find distinct upper and lower bounds:

$$V_{l_6} = -g, \ \ V_{u_6} = g$$

Step 4: We now know the initial domain D_Y with the bounds found from step 3, and can solve problem 3 using the input and state constraints found above. Although finding the time-optimal solution analytically for a linear system should be simple enough, according to Baker [3, 2], there is no known pure analytic method for this problem when arbitrary input and state bounds are given together. As suggested by Baker, we approach this minimum time problem from a maximum range problem view point by guessing a sufficiently large final time t_f, then obtaining a bang-bang input sequence using the linear programming method.

Meanwhile, we note that Y_1, Y_4 and their corresponding inputs V_1, V_2 are decoupled from each other and the bounds found in step 4 are not the same for matching states and controls. For the given final condition of $(Y_1, Y_4) = (100, -100)$, this will obviously result in different minimum time results for x and z since Y_1 and Y_4 have the same magnitude. One easy way to solve this problem is to simply assume the same state and control bounds for each direction. Similar to the strategy in Algorithm 1, we choose the maximum possible bounds for both. That is,

$$\begin{aligned} V_{l1,l2} &= -2g\pi, \qquad V_{u1,u2} = 2g\pi \\ Y_{l3,l6} &= -2g, \qquad Y_{u3,u6} = 2g \end{aligned}$$

Using these bounds, we solve a linear programming problem for the x-coordinate. The result for the z-coordinate can be found by simply multiplying by -1. The input sequence is found to be

$$V_1 = \begin{bmatrix} 2g & 0 & -2g & 0 & 2g \end{bmatrix}$$

Using this sequence data and the boundary condition, we use the well-known solution formula for a linear system to obtain an accurate time sequence.

$$Y_1(t_f) = 100 = \int_0^{t_f} e^{A(t_f - s)} B V_1(s) ds \tag{10}$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

There are 4 different points where input-stepping occurs, and they are

$$\begin{aligned} t_{step} &= \begin{bmatrix} 0 & t_1 & t_2 & t_3 & t_4 & t_f \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0.32 & 2.11 & 2.74 & 4.53 & 4.85 \end{bmatrix} \end{aligned}$$

<u>Step 5</u>: Starting from $\alpha = 0$, we gradually increase α until all of the states and inputs (x, u) stay within the given bounds. We stop at $\alpha = 0.375$, at which point the bounds are

$$\begin{aligned} |V_{1,2}| &\leq 2g\pi * 0.25 \\ |Y_{3,6}| &\leq 2g * 0.25 \end{aligned}$$

As shown in Figure 3, the $V_{1,2}$ input is bang-bang with switching times and final time

$$t_{step} = \begin{bmatrix} 0 & 0.32 & 4.36 & 5.00 & 9.04 & 9.36 \end{bmatrix}$$

The final time t_f is found as 9.36 sec. See Figure 4 for the final result after transforming the linear (Y, V) trajectory back to the original (x, u) coordinates. The top figure shows the trajectory of the helicopter with respect to the x-z axis. Small line segments represents the pitch angle θ. The minimum principle [6] suggests when the input appears linearly in the state equation, the time-optimal input must be bang-bang. Since the resulting system inputs T and τ do not show this shape and the maximum values are less than the bounds,

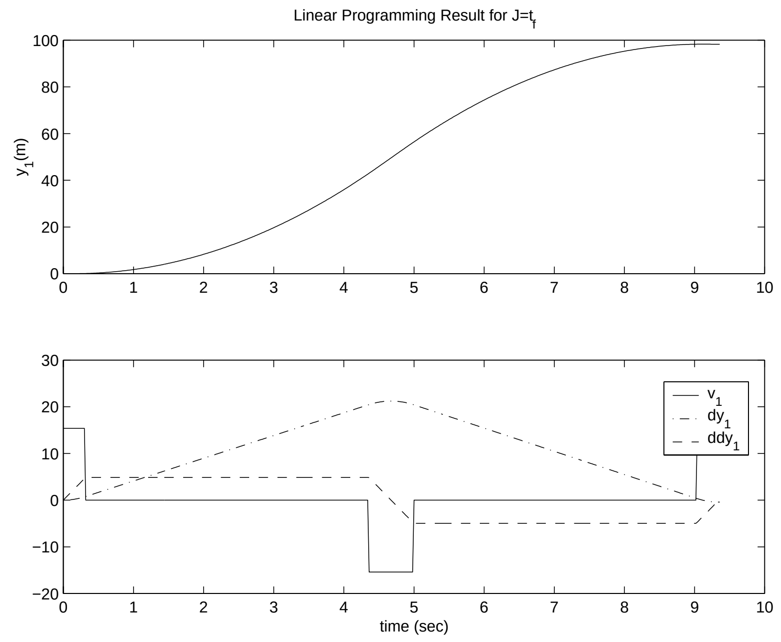

Figure 3: After the linear programming and solving for $t_1 \cdots t_f$ using the equation (10), we can obtain an accurate time-optimal trajectory for each state of interest.

the solution is not truly time-optimal. Nevertheless, since one can obtain this solution without referring to the nonlinear system dynamics during optimization, the solution was obtained much faster than using a pure numerical nonlinear optimization method within the nonlinear coordinate.

As mentioned in the previous section, when solving a traditional nonlinear trajectory optimization problem (Problem 2), one often needs to provide an initial guess to the optimization program. However, many guesses can lead to an unstable result. In fact, for this system, a default initial guess of zero input does not return any feasible solution to the optimization problem. Figure 5 is the nonlinear optimization result from the nonlinear model (9) using the sequential quadratic programming method, where the trajectory from the linear optimization (Figure 4) is used as an initial guess. The solution converged, and the final time t_f is found as 9.10 sec, a small improvement from the linear (approximate) optimization result. This shows that the use of the linear optimization result can be a good step toward finding a globally optimal result.

5 Conclusion

This paper presented a new way to generate a feasible trajectory for an input-output linearizable while satisfying input and state constraints. The system model is transformed into a linear form, then suitable bounds are obtained for the linear coordinates. Within the linear coordinates, an optimal trajectory was found using linear programming techniques; this trajectory was then transformed back to the original nonlinear coordinates without violating the bounds given on the original nonlinear system.

Although the resulting trajectory is not optimal due to the transformation and the limiting choice of the bounds, one can easily obtain a reasonable solution quickly as was shown in the helicopter trajectory generation example. Systems such as spacecrafts, helicopters, airplanes, and cars are often input-output linearizable, and this method may be used to find feasible trajectories for these and other autonomous systems. When the method works, it can be used to quickly find trajectories which connect any pairs of points. If a geometric planner is used to find a set of waypoints which connect the start and goal points

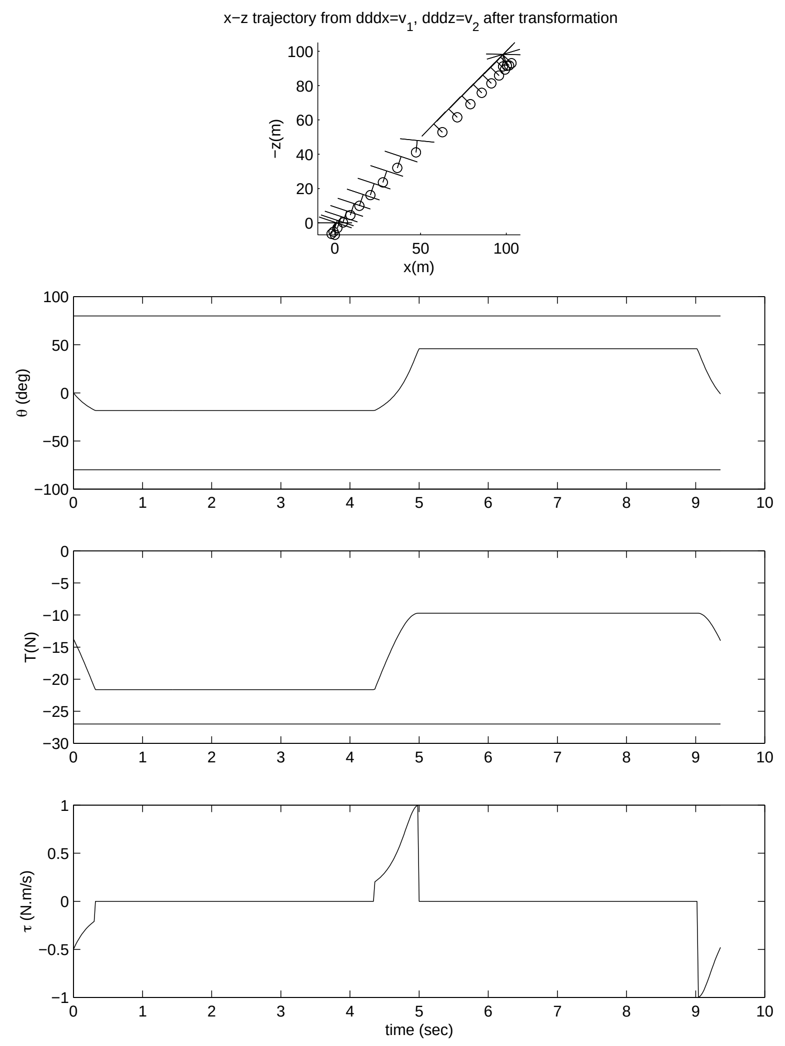

Figure 4: The time-optimal trajectory obtained from the linear (Y, V) coordinates is applied to the actual helicopter model. All the states and inputs are within bounds.

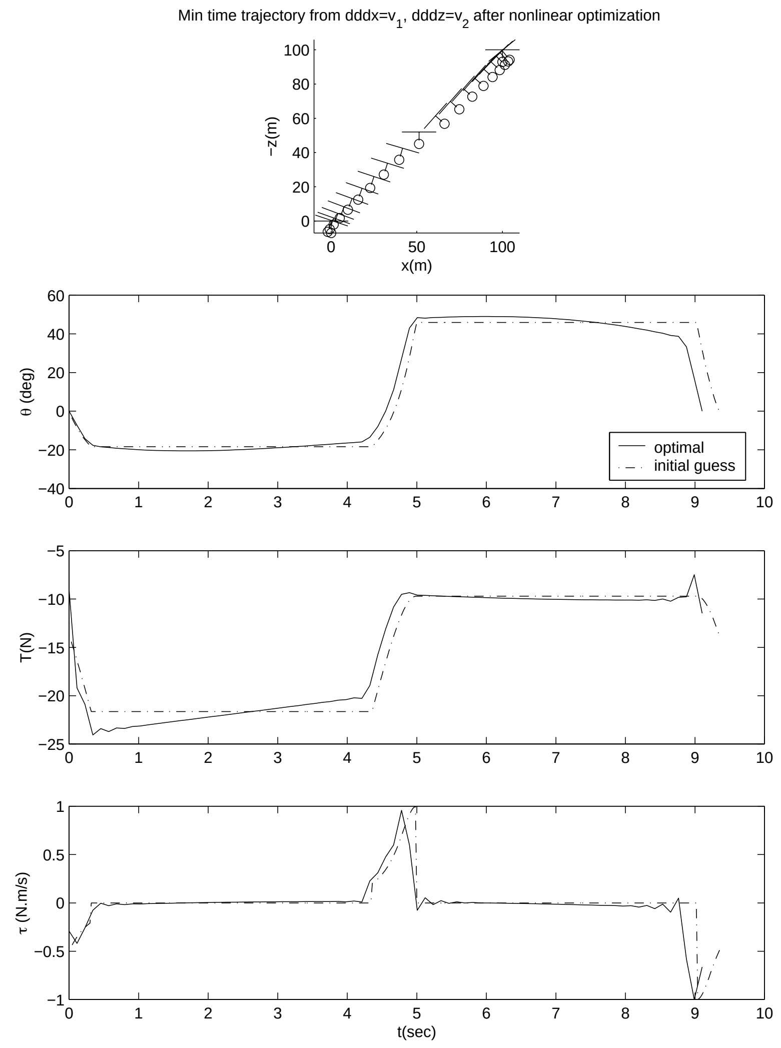

Figure 5: Nonlinear optimization result (solid line) using sequential quadratic programming method with the linear optimization result (dashed line) as an initial guess.

and are obstacle-free, the planning method presented in this paper could be used to find feasible trajectories connecting these waypoints.

Future work includes examining more example trajectories and other nonlinear systems. In addition, we are continuing to work on finding a better way to express the bounds in the linear coordinates to make the trajectory approach closer to the true optimal result. Another area of interest is the integration of trajectory tracking (feedback control) with trajectory planning.

References

[1] S. Arinaga, S. Nakajima, H. Okabe, A. Ono, and Y. Kanayama. A Motion Planning Method for an AUV. In *Proceedings of the IEEE Symposium on Autonomous Underwater Vehicle Technology*, pages 477–484, 1996.

[2] D. Baker. Exact solutions to some minimum time problems and their behavior near inequality state constraints. In *Proceedings of the IEEE Conference on Decision and Control*, volume 2, pages 1622–1626, 1987.

[3] D. Baker. Exact solutions to some minimum time problems with inequality state constraints. *Mathematics of Control, Signals, and Systems*, 2(2):137–169, 1989.

[4] J. Betts. Survey of numerical methods for trajectory optimization. *AIAA Journal of Guidance, Control, and Dynamics*, 21(2):193–207, 1998.

[5] A. Bryson. *Dynamic Optimization*. Addison Wesley Longman, Inc., 1999.

[6] A. Bryson and Y. Ho. *Applied Optimal Control*. Hemisphere, New York, 1975.

[7] Aerial Robotics Competition. As part of the Association for Unmanned Vehicle Systems International. The home page is at http://avdil.gtri.gatech.edu/AUVS/IARCLaunchPoint.htm.

[8] R. Fortenbaugh, K. Builta, and K. Schulte. Development and testing of flying qualities for manual operation of a tiltrotor UAV. In *Proceedings of the American Helicopter Society 51st Annual Forum*, volume 1, pages 299–320, 1995.

[9] R. Howard and I. Kaminer. Survey of unmanned air vehicles. In *Proceedings of the American Control Conference*, pages 2950–2953, 1995.

[10] A. Isidori. *Nonlinear Control Systems*. Springer-Verlag Berlin, Heidelberg, 1989.

[11] J. Jackson and P. Crouch. Curved path approaches and dynamic interpolation. *IEEE Aerospace and Electronic Systems Magazine*, 6(2):8–13, 1991.

[12] S. K. Kim and D. M. Tilbury. Mathematical modeling and experiemental identification of a model helicopter. Submitted to the *AIAA Journal of Guidance, Control, and Dynamics*, 2000.

[13] J. Latombe. *Robot Motion Planning*. Boston : Kluwer Academic Publishers, 1991.

[14] A. Lee, A. Bryson, and W. Hindson. Optimal landing of a helicopter in autorotation. *AIAA Journal of Guidance, Control, and Dynamics*, 11(1):7–12, 1988.

[15] H. Seywald. Trajectory optimization based on differential inclusion. *AIAA Journal of Guidance, Control, and Dynamics*, 17(3):480–487, 1994.

[16] M. Sugeno. Fuzzy hierarchical control of an unmanned helicopter, 1994. Available at ftp://ftp.cs.arizona.edu/japan/kahaner.reports/sugeno.94.

[17] H. Sussmann and W. Liu. Limits of highly oscillatory controls and the approximation of general paths by admissible trajectories. In *Proceedings of the IEEE Conference on Decision and Control*, pages 437–442, 1991.

[18] C. Thornberg and J. Cycon. Sikorsky aircraft's UAV, Cypher: System description and program accomplishments. In *Proceedings of the American Helicopter Society 51st Annual Forum*, May 1995.

[19] D. Tilbury, J. Laumond, R. Murray, S. Sastry, and G. Walsh. Steering car-like systems with trailers using sinusoids. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1993–1998, 1992.

[20] D. Tilbury, R. Murray, and S. Sastry. Trajectory generation for the N-trailer problem using Goursat normal form. In *IEEE Transactions on Automatic Control*, pages 802–819, 1995.

[21] C. Vasudevan and K. Ganesan. Case-Based Path Planning for Autonomous Underwater Vehicles. *Autonomous Robots*, 3:79–89, 1996.

[22] Y. Wang and D. Lane. Subsea vehicle path planning using nonlinear programming and constructive solid geometry. In *IEE Proceedings of Control Theory and Applications*, volume 144, pages 143–152, 1997.