A New Computational Approach to Real-Time Trajectory Generation for Constrained Mechanical Systems [1]

Mark B. Milam, Kudah Mushambi, and Richard M. Murray[2]
{milam,kudah,murray}@cds.caltech.edu

Abstract

Preliminary results of a new computational approach to generate aggressive trajectories in real-time for constrained mechanical systems are presented. The algorithm is based on a combination of nonlinear control theory, spline theory, and sequential quadratic programming. It is demonstrated that real-time trajectory generation for constrained mechanical systems is possible by mapping the problem to one of finding trajectory curves in a lower dimensional space. Performance of the algorithm is compared with existing optimal trajectory generation techniques. Numerical results are reported using the NTG software package.

Keywords: Real-time optimization, nonlinear control design, optimal control, constrained trajectory generation, guidance.

1 Introduction

A simple approach to tracking a target using a feedback control system is to subtract the current state of the tracking system from the target and feed this error into a controller. This approach is frequently referred to as a one degree of freedom design. It is well known, for a large class of nonlinear mechanical systems with constraints, that this approach generally does not work well since we are likely tracking a drifting equilibrium configuration which is an infeasible trajectory of the system. Furthermore, achieving high performance while guaranteeing stability in the presence of constraints is difficult.

Another approach to tracking a target is the two degree of freedom design depicted in Figure 1. The two-degree of freedom design consists of a trajectory generator and a linear feedback controller. The trajectory generator provides a feasible feed-forward control and reference trajectory in the presence of system and actuation constraints. Given inherent modeling uncertainty, a feedback controller provides stability around the reference trajectory. This approach has the advantage that the system is tracking a feasible trajectory along which the system can be stabilized.

[1]Research supported in part by AFOSR and DARPA.

[2]Control and Dynamical Systems, Mail Code 107-81, California Institute of Technology, Pasadena, CA 91125.

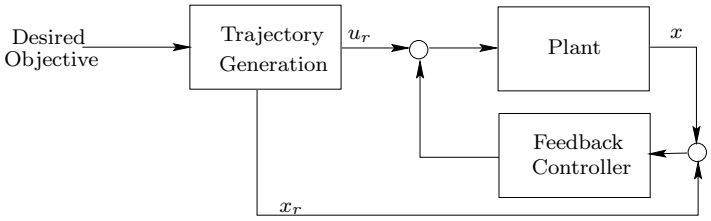

Figure 1: Two Degree of Freedom Design

A prime example of a case where a two degree of freedom design would be implemented is an Uninhabited Combat Air Vehicle (UCAV). The desired objective of the UCAV would be commanded by the operator or pre-programmed without operator intervention. The desired objective may be to go to a single point, pass through several way-points, or track a target. For some missions, UCAV's will autonomously fly highly aggressive trajectories frequently on the fringe of the flight envelope. The idea of directly commanding the UCAV to track a target, as in the one degree of freedom design, would be impractical considering the fast dynamics and constraints of a typical UCAV. Therefore, the UCAV control problem is to construct, in real-time, a trajectory that satisfies the desired objective as well as the system dynamics while ensuring that the aircraft is stabilized and operating within flight and actuation constraints.

A wealth of work has been done on feedback controllers for stabilization of nonlinear systems around a trajectory, for example, Li [15], and Pesch [17, 18], and will not be covered in this paper. Much less work has been done in the area of real-time trajectory generation. Generating real-time trajectories will be the main topic of this paper.

There are two common techniques for generating real-time trajectories. The first is based on searching and interpolating over a large trajectory database in real-time, for example, Atkeson [2]. Searching a trajectory database and piecing together trajectories can be very time consuming and sub-optimal for large dimensional problems. The second approach, which we will advocate in this paper, is to solve trajectory generation problem online in real-time.

A classical approach to solving trajectory generation problem online would be to solve an optimal control problem in real-time. The standard indirect, direct, and homotopy methods for the numerical solution of optimal control problems can be found in von Stryk [22], Hargraves and Paris [12], and Chen and Huang [4], respectively. These methods generally run too slow to be adapted to real-time implementation.

A modern approach to solving the trajectory generation problem would be to use nonlinear geometric control techniques. These techniques depend on finding outputs such that the complete differential behavior of the system can be found in terms outputs and their derivatives. The theory of the existence and finding such outputs are subjects of Isidori [13], Rathinam [19], Charlet *et al*. [3], and Fliess [9]. Of course, there are many classes of systems such that these outputs cannot be found even if they do exist. It is usually not too difficult to find an output that will characterize part of the system behavior. In this case, careful attention must be paid to the stability of the resulting uncharacterized zero dynamics which could lead to unbounded controls and states. Techniques were developed in Devasia and Chen [7], Devasia [6], and Verma and Junkins [21] to circumvent this problem. Some methods concerning the determination of real-time reference trajectories without constraints for diffentially flat systems are illustrated in van Nieuwstadt [20]. An approach to finding feasible trajectories for constrained differentially flat systems by approximating constraints with linear functions is given in Faiz and Agrawal [8]. Using nonlinear control techniques for trajectory generation have the advantage that they are likely to be implementable in real-time. However, most of these techniques do not take into consideration actuation and state constraints and do not give optimal trajectories. Furthermore, the user must estimate the time horizon of the trajectory.

In light of the UCAV example, we consider the combination of optimal reference trajectories, time horizon determination, real-time computation restrictions, convergence, and trajectory determination in the presence of control and state constraints all to be crucial requirements that must be considered in real-time trajectory generation. Therefore, the purpose of this paper will be to present an introduction to a methodology that appears to satisfy all these requirements.

The methodology we will present is based on finding trajectory curves in a lower dimensional space as in nonlinear control theory. Once the curves are parameterized by B-splines, sequential quadratic programming is used to find the coefficients of the B-splines to satisfy the optimization objectives as well as the constraints. Section 2 describes the elements of the methodology. A numerical comparison is given in Section 3 between the nonlinear trajectory generation (NTG) software package resulting from the proposed methodology, direct collocation, and the direct shooting software package RIOTS [1]. Illustrative examples of how we would use NTG in a real-time environment are provided in Section 4. A summary and future directions are provided in the final section.

2 Problem Setup and Proposed Method of Solution

Consider the system

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

where all vector fields and functions are real-analytic. It is desired to find a trajectory of (1) that minimizes the cost

$$\begin{array}{rcl} J & = & \phi_f(x(t_f),u(t_f)) + \phi_0(x(t_0),u(t_0)) + \\ & & \int_{t_0}^{t_f} L(x(t),u(t))dt \end{array} \tag{2}$$

subject to a vector of N_0 initial time, N_f final time, and N_t trajectory constraints,

$$\begin{array}{rcl} lb_0 & \leq & \psi_0(x(t_0),u(t_0)) \leq ub_0, \\ lb_f & \leq & \psi_f(x(t_f),u(t_f)) \leq ub_f, \\ lb_t & \leq & S(x,u) \leq ub_t, \end{array} \tag{3}$$

respectively.

There are three primary components to the real-time trajectory generation methodology we propose. The first is to determine outputs such that Equation (1) can be mapped to a lower dimensional output space. Once this is done the cost in Equation (2) and the constraints in Equation (3) can also be mapped to the output space. The second is to parameterize the outputs in terms of B-spline curves. Finally, sequential quadratic programming is used to solve for the coefficients of the B-splines to minimize the cost subject to constraints in output space.

The key to the approach is to map Equation (1) to a lower dimensional output space. The idea being that it will be easier as well as computationally more efficient to solve a lower dimensional problem. In most cases it is desirable to find an output $z = z_1, ..., z_q$ of the form

$$z = A(x,u,u^{(1)},\ldots,u^{(r)}) \tag{4}$$

such that $(x(t),u(t))$ can be determined completely from

$$(x,u) = B(z,z^{(1)},\ldots,z^{(s)}) \tag{5}$$

where $x^{(i)}$ denotes the ith time derivative of x. By doing this the differential constraints in Equation (1) are automatically satisfied and Equation (1) is said to be differentially flat. A necessary condition for the existence of such an output can be found in Fliess [9].

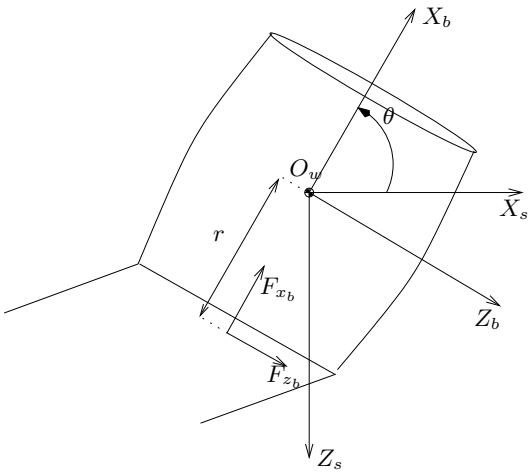

Figure 2: Planar Ducted Fan Coordinate System Definition

Examples of differentially flat systems can be found in Fliess *et al*. [10]. In general, finding a flat output may be very difficult even if one could prove it does exist. Therefore, in the case that we cannot determine a flat output or no flat output exists we will map the system dynamics in Equation (1) to the lowest dimensional space possible. Therefore, $(x(t), u(t))$ will be completely determined from

$$\begin{array}{rll} (x,u) = & C_1(z, z^{(1)}, \ldots, z^{(p_1)}) & \text{and} \\ & C_2(z, z^{(1)}, \ldots, z^{(p_2)}) & = 0. \end{array} \tag{6}$$

We will illustrate different mappings of the system dynamics by an example. The example for illustration will be a simplified model of a UCAV. This system is commonly referred to as the planar ducted fan and is depicted in Figure 2. The ODE's for this system may be written

$$\begin{array}{rcl} m\ddot{x}\cos\theta - (m\ddot{z} - mg)\sin\theta & = & F_{X_b} \\ m\ddot{x}\sin\theta + (m\ddot{z} - mg)\cos\theta & = & F_{Z_b} \\ (J/r)\ddot{\theta} & = & F_{Z_b}. \end{array} \tag{7}$$

Following the work of Rathinam in [19], the planar ducted fan has five dependent variables x, z, θ, F_{X_b}, and F_{Z_b}. Since there are three equations of motion, the system is under determined by two equations. Choosing any two differentially independent variables yields a fully determined set of ODE's depending on a number of unknown constants. An example of two differentially dependent variables would be F_{Z_b} and θ since they are related in Equation (7) in which no other variables are present. Suppose F_{X_b} and F_{Z_b} are chosen for the time interval of interest, each equation in Equation (7) would have to be integrated twice yielding x, z, and θ and six constants. However, it is not desirable to integrate any equations in our approach. It is obvious that F_{X_b} and F_{Z_b} are not flat outputs of Equation (7) since it was necessary to integrate three equations. In order to fit the framework of equation (6), the following outputs must be chosen

$$z_1 = x \quad z_2 = z \quad z_3 = \theta \quad z_4 = F_{X_b} \quad z_5 = F_{Z_b}. \tag{8}$$

There is no advantage to selecting this set of variables since they must also satisfy the differential constraints in Equation (7).

Eliminating F_{Z_b} in Equation (7) the following relation is obtained

$$m\ddot{x}\sin\theta + (m\ddot{z} - mg)\cos\theta = (J/r)\ddot{\theta}. \tag{9}$$

If x and z is selected as a function of time in Equation (9), it is now necessary to integrate Equation (9) twice to yield θ and two constants. In order to fit the framework of equation (6), the following outputs are chosen

$$z_1 = x \qquad z_2 = z \qquad z_3 = \theta. \tag{10}$$

However, the outputs must be chosen in such a way that Equation (9) is satisfied, namely,

$$m\ddot{z_1}\sin z_3 + (m\ddot{z_2} - mg)\cos z_3 = (J/r)\ddot{z_3}. \tag{11}$$

For the planar ducted fan it turns out that there exists two variables such that all of the dependent variables can be recovered without any constants. These outputs are flat outputs and are given by

$$\begin{array}{rcl} z_1 & = & x + (J/rm)\cos\theta \\ z_2 & = & z - (J/rm)\sin\theta \end{array} \tag{12}$$

The key to recovering the dependent variables can be seen by taking two derivatives of the outputs. In this set of equations it is possible to solve for θ in terms of the outputs with

$$\tan\theta = \frac{(mg - m\ddot{z_2})}{m\ddot{z_1}}.$$

The other dependent variables can easily be found once θ is found.

The point of this example is to illustrate that choice of outputs in the proposed trajectory generation methodology is highly dependent on the problem under consideration. For example, suppose in the above example we choose the flat outputs and we impose an input constraint on F_{X_b} and F_{Z_b}. The expressions for these variables are a very complicated and possibly ill-conditioned function of the flat outputs and their derivatives. Choosing the flat outputs could have deleterious effects on the numerical optimization. Numerical investigations show that in this case it is advantageous to choose Equation (10) as the outputs and include the trajectory constraint Equation (11). The slight increase in trajectory generation times by choosing the flat outputs does not offset the the lower robustness to random initial guesses.

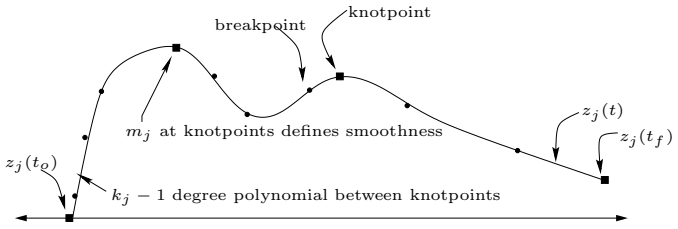

Figure 3: Spline Representation of Outputs

Furthermore, suppose that it was not possible to find the flat outputs in Equation (12), the outputs in Equation (10) may be the lowest dimensional space with the constraint Equation (11) that the solution curves can be found.

Once chosen, the outputs are parameterized in terms of B-spline basis functions. B-splines are chosen as basis functions for their flexibility and ease of enforcing continuity across knot points. A complete treatment of B-splines can be found in DeBoor [5]. A pictorial representation of an output is given in Figure 3. The outputs are written in terms of finite dimensional B-spline curves as

$$\begin{aligned} z_1 &= \textstyle\sum_{i=1}^{p_1} B_{i,k_1}(t) C_i^1 \text{ for the knot sequence } \mathbf{t_1} \\ z_2 &= \textstyle\sum_{i=1}^{p_2} B_{i,k_2}(t) C_i^2 \text{ for the knot sequence } \mathbf{t_2} \\ &\vdots \\ z_q &= \textstyle\sum_{i=1}^{p_q} B_{i,k_q}(t) C_i^q \text{ for the knot sequence } \mathbf{t_q} \\ &\quad \text{and} \quad p_j = l_j(k_j - m_j) + m_j \end{aligned}$$

where $B_{i,k_j}(t)$ is the B-spline basis function defined in Deboor [5] for the output z_j with order k_j, C_i^j are the coefficients of the B-spline, l_j is the number of knot intervals, and m_j is number of smoothness conditions at the knots. After the outputs have been parameterized in terms of B-spline curves, the coefficients of the B-spline basis functions will be found using sequential quadratic programming.

To cast the problem into the framework necessary for sequential quadratic programming, it is necessary to discretize the time interval into w intervals with $w+1$ breakpoints. It is emphasized that the constraints in Equation (3) will only be satisfied at a finite number of points. In addition, the discrete approximation to the continuous cost in Equation (2) is dependent on the choice of integration technique.

The problem now can be stated in the form

$$\min_{y \in \mathbb{R}^M} F(y) \qquad \text{subject to} \quad lb \leq c(y) \leq ub$$

where

$$\begin{aligned} y &= (C_1^1, \ldots, C_{p_1}^1, C_1^2, \ldots, C_{p_2}^2, \ldots, C_1^q, \ldots, C_{p_q}^q), \\ &\quad \text{and } M = \textstyle\sum_{i=1}^{q} p_i \end{aligned}$$

and $F(y)$ is the discrete approximation in output space to the objective in Equation (2). The vector $c(y)$ contains the constraints in output space from Equation (3) and any other constraints represented in output space as a result of not choosing the flat output. Note that the trajectory constraints will be satisfied at $w+1$ breakpoints chosen by the user. The lower and upper bounds for the constraints denoted by the vectors lb and ub.

The two sequential quadratic programming packages CFSQP and NPSOL are being considered for the sequential quadratic programming routines at this time. CFSQP [14] has the advantage that it generates feasible iterates throughout the optimization process. Although NPSOL [11] only guarantees satisfaction of the nonlinear constraints when the optimal point is reached, it is much faster and appears to handle larger dimensional problems than CFSQP. Ultimately, an investigation on a real system will determine which code is the best for the proposed trajectory generation methodology. For this paper, NPSOL will be used in all examples.

3 Performance Comparison

In this section, a comparison will be made between NTG, the software package developed using the algorithm described in Section 2, RIOTS, and direct collocation. These techniques were chosen since all could solve general nonlinear constrained optimization problems. Only initial random guesses are considered in this comparison. Two problems were chosen for comparison: The forced VanDerPol Oscillator [1] and the constrained planar ducted fan. A comparison will be based on computation times and convergence from random initial conditions. All tests were conducted on a Sun Ultra 10 333 MHz computer.

The first problem under consideration is the forced VanDerPol oscillator. The comparison will be between RIOTS and NTG. Direct Collocation was not considered competitive for this problem. The cost, dynamics, and constraints are the following:

$$\min_u J(u) \doteq \frac{1}{2} \int_0^5 x_1^2 + x_2^2 + u^2 dt$$

subject to

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= -x_1(t) + (1 - x_1^2(t)) x_2(t) + u(t) \\ x_1(0) &= 1,\ x_2(0) = 0,\ x_2(5) - x_1(5) - 1 = 0. \end{aligned}$$

The fact that the forced VanDerPol oscillator is differentially flat with the output $z_1(t) = x_1(t)$ was used when implementing this problem in the NTG code. The smoothness and order of the B-spline parameterization for each interval was taken to be three and five, respectively. The number of breakpoints was chosen to be four times the number of coefficients.

For RIOTS, the input was parameterized by a second order B-spline for each interval. Trapezoidal integration was used in both software packages.

Method	CPU Time (s)	Intervals	Cost
NTG	.002	1	1.9127
RIOTS	.0178	11	1.7081
NTG	.1191	30	1.6859
RIOTS	.2261	200	1.6857

Table 1: RIOTS and NTG VanDerPol Comparison

First, a comparison was made between CPU usage and the cost. Each point on the first plot of Figure 4 is the average cost and CPU time of 100 random initial guesses for the free variable coefficients in both RIOTS and NTG. The plot shows that as the number of coefficients representing the input in RIOTS and the output in NTG was increased the lower the cost. RIOTS needed a minimum of 11 intervals for convergence from a random initial guess while NTG needed only one interval. The second plot in Figure 4 shows the trajectories at the lowest number of intervals that converged for both RIOTS and NTG. Table 1 shows that NTG's

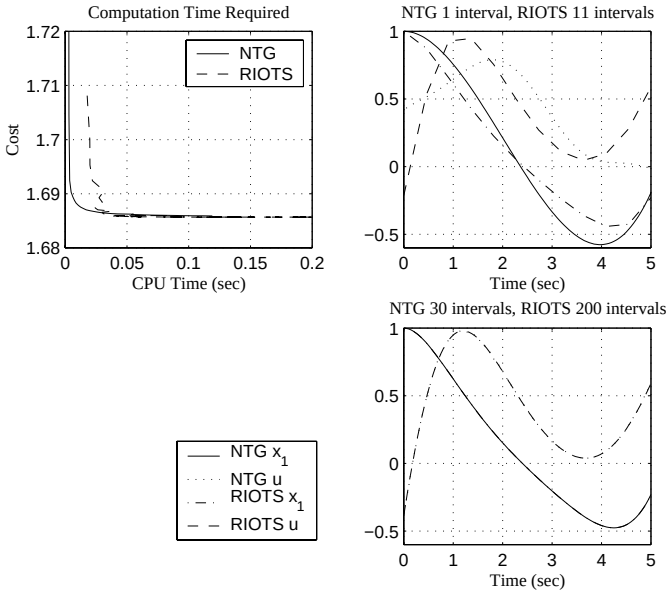

Figure 4: RIOTS and NTG VanDerPol Comparison

computation time is one eighth that of RIOTS with a 12 percent increase in cost for the minimum interval case. The third plot in Figure 4 shows that both RIOTS and NTG converge to same cost for increasing numbers of coefficients. The results of this comparison show that for low intervals NTG can compute trajectories at significantly lower CPU times than RIOTS at competitive cost. For some real-time applications computing a feasible, albeit sub-optimal, trajectory may be necessary as a result of processing limitations.

The planar ducted fan given outlined in Section 2 will be used in the next comparison. The objective will be to move from equilibrium point to equilibrium point in minimum time subject to a thrust vectoring input constraint of the form

$$0 \leq F_{X_b} \leq 17 \text{ and } -F_{X_b}/3 \leq F_{Z_b} \leq F_{X_b}/3.$$

The boundary conditions are the following:

$$x(t_0) = (*, *, *, *, \pi/2, 0) \text{ and } x(t_f) = (*, *, *, *, \pi/2, 0)$$

where $x(t) = (x, \dot{x}, z, \dot{z}, \theta, \dot{\theta})$ and $*$ can be either 1, 0, or -1. There are 6561 possible combinations of boundary conditions.

In order to account for the free final time variable, the planar ducted fan equations are scaled to yield

$$\begin{aligned} mx'' \cos\theta - (mz'' - \xi^2 mg)\sin\theta &= \xi^2 F_{X_b} \\ mx'' \sin\theta + (mz'' - \xi^2 mg)\cos\theta &= \xi^2 F_{Z_b} \\ (J/r)\theta'' &= \xi^2 F_{Z_b} \\ \xi' &= 0 \end{aligned} \tag{13}$$

where x' denotes $\frac{dx}{d\tau}$ and $\tau = t/\xi$.

The outputs $z_1 = x, z_2 = z, z_3 = \theta$, and $z_4 = \xi$ are chosen for NTG. Note that the flat outputs were not chosen in order to illustrate NTG for non-flat systems. In total, NTG has four trajectory constraints (three due to the constraint on the inputs and one due to the output selection). Sixth order B-splines with C^3 continuity across knot points and four intervals will be chosen for the first three outputs. A first order B-spline with one interval is chosen to parameterize the final output. The constraints are satisfied for twenty equally spaced breakpoints. Twenty break-points were chosen to guarantee that the error in satisfying the constraints between breakpoints was kept small.

In order to use direct collocation, the states x, z, and θ were approximated with fourth order B-splines. Approximating the inputs F_{X_b} and F_{Z_b} with third order B-splines produced the best results. The resulting constraints were required to be satisfied at 20 equally spaced constraints. RIOTS was not included in this comparison since the problem is highly constrained and nonlinear. Single shooting based techniques, such as RIOTS, often do not work well for highly nonlinear constrained systems.

The point of this example is to compare the convergence of NTG with other techniques. Since there are no guarantees of convergence for non-convex sequential quadratic programming based optimization techniques, it would be expected that any technique used in real-time application would need to be robust to the initial guess.

The test conducted to test convergence was the following: Choose 500 random initial guesses for NTG and

100 for direct collocation for the unknown free variables in each of the 6561 test cases and test for convergence. Figure 5 gives the results of the optimization. The first plot shows that for any given 6561 test case most of 500 initial guess converged to a solution using NTG. In fact, all of the 6561 test cases converged for more than 20 of the 500 initial guesses. On the other hand, the second plot in Figure 5 shows that most of the 6561 test cases did not converge for any initial guess using direct collocation. This test illustrates that it is

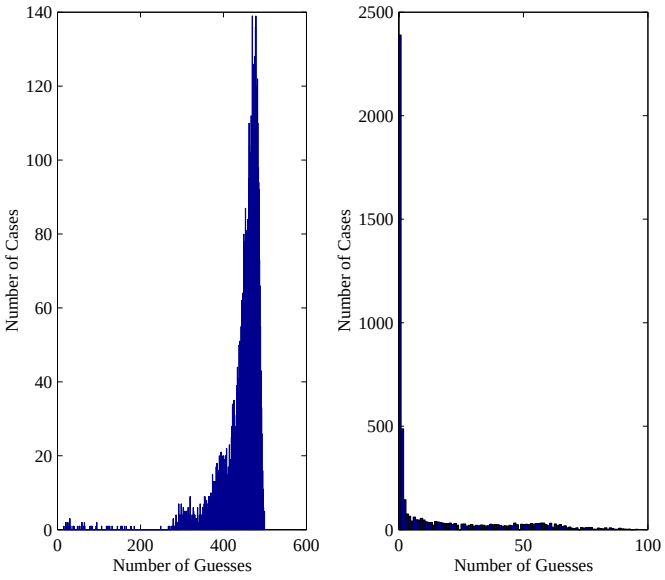

Figure 5: NTG and Direct Collocation Convergence Analysis

advantageous to parameterize an output in a lower dimensional space instead parameterizing the inputs and the states when solving trajectory generation problems. However, it was not expected that the results would be so skewed. Further testing will have to done to remove any elements of subjectivity. For example, direct collocation may work better in the case that the desired objective was to minimize the energy of the inputs over a fixed time interval.

4 Terrain Avoidance

The purpose of this section is to illustrate how NTG would be used in determining trajectories in a real-time environment.

The planar ducted fan given in Equation 13 will be used in this example. The mass properties and constraints are chosen to be similar to that of Caltech Ducted Fan [16] without aerodynamics.

The objective is to move in minimum time from equilibrium to equilibrium point subject to the terrain and inputs constraints. The terrain constraint was modeled as a B-spline with sufficient smoothness across knot points.

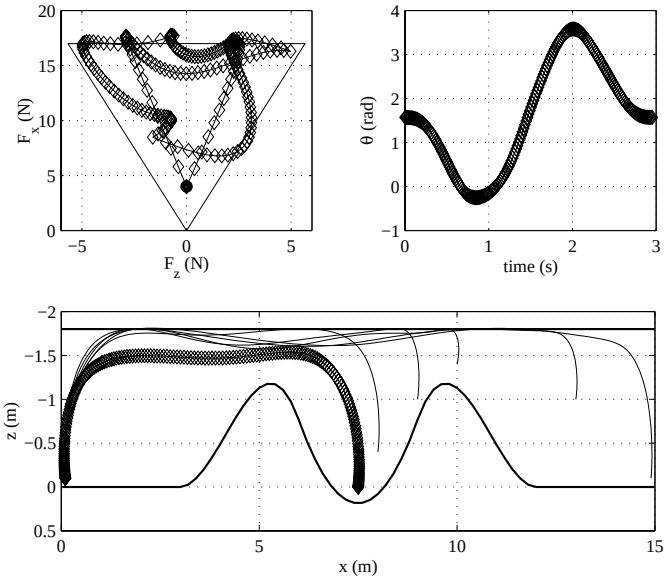

Figure 6: NTG Terrain Avoidance with Warm Start Example

Start Method	CPU Time (s)	Cost (s)
Cold	1.26	2.95
Warm	.09	3.04
Warm	.18	3.27
Warm	.13	3.48
Warm	.31	3.64
Warm	.15	3.86

Table 2: NTG Terrain Avoidance with Warm Start

An random initial guess of the trajectory was given to NTG to find the marked trajectory shown in Figure 6. For NPSOL, this is considered a cold start since the Lagrange Multipliers are not known. The other five trajectories were found by using a warm start. A warm start uses the previous trajectories Lagrange Multipliers as an initial starting point in finding the active set of constraints. Table 2 provides summary of the cost as well as the CPU computation times for the trajectories. Note the significant improvement in CPU times for the warm start trajectories.

5 Summary and Future Directions

We have presented a promising new methodology for real-time trajectory generation. A new software package NTG has been written to implement this methodology. Favorable comparisons have been made for NTG over RIOTS and direct collocation. Examples show that real-time implementation is possible. However, much more testing needs to to be done to identify the strengths and weakness of the methodology. Currently, we are working on further performance comparisons.

We are currently implementing an automatic mesh and breakpoint determination algorithm as well as an option to include the CFSQP sequential quadratic programming code. We are also establishing a measure the acceptable error of constraints between breakpoints.

Our goals for the immediate future are to get the methodology working on the real Caltech Ducted Fan. This problem has the added difficulties of tabular data and redundant actuators and should prove to be an excellent problem for NTG. Currenly NTG is showing promising results for real-time trajectory generation for the model the the Caltech Ducted Fan with aerodynamics. The ultimate test for NTG will be real-time implementation in a two degree of freedom design on the Caltech Ducted Fan.

References

[1] A.Schwartz. *Theory and Implementation of Numerical Methods Based on Runge-Kutta Integration for Solving Optimal Control Problems*. PhD thesis, U.C. Berkeley, 1996.

[2] C. Atkeson. *Using Trajectory Optimizers to Speed Up Global Optimization in Dynamic Programming*, chapter 6. Morgan Kaufmann, 1994.

[3] B. Charlet, Levine J., and Marino R. On dynamic feedback linearization. *Systems and Control Letters*, 13:143–151, 1989.

[4] Y. Chen and J. Huang. A new computational approach to solving a class of optimal control problems. *International Journal of Control*, 58(6):1361–1383, 1993.

[5] C. de Boor. *A Practical Guide to Splines*. Springer-Verlag, 1978.

[6] S. Devasia. Approximated stable inversion for nonlinear systems with nonhyperbolic internal dynamics. *IEEE Trans. Automat. Contr.*, 44(7):1419–1425, Feb 1999.

[7] S. Devasia, D. Chen, and B. Paden. Nonlinear inversion-based output tracking. *IEEE Trans. Automat. Contr.*, 42:930–943, July 1996.

[8] N. Faiz and S. Agrawal. Trajectory planning of differentially flat systems with dynamics and inequalities. Preprint.

[9] M. Fliess, J. Levine, P. Martin, and P. Rouchon. Flatness and defect of non-linear systems: introductory theory and examples. *International Journal of Control*, 61(6):1327–1360, 1995.

[10] M. Fliess, J. Levine, P. Martin, and P. Rouchon. A lie-backlund approach to equivalence and flatness of nonlinear systems. *IEEE Trans. Auto. Cont.*, 44(5):928–937, 1999.

[11] P. Gill, W. Murray, M. Saunders, and M. Wright. *User's Guide for NPSOL 5.0: A Fortran Package for Nonlinear Programming*. Systems Optimization Laboratory, Stanford University, Stanford, CA 94305.

[12] C. Hargraves and S. Paris. Direct trajectory optimization using nonlinear programming and collocation. *AIAA J. Guidance and Control*, 10:338–342, 1987.

[13] A. Isidori. *Nonlinear Control Systems*. Springer-Verlag, 1989.

[14] C. Lawrance, J. Zhou, and A. Tits. *User's guide for CFSQP Version 2.5*. Institute for Systems Research, University of Maryland, College Park, MD 20742.

[15] P. Li. Constrained tracking control of nonlinear systems. *Systems and Control Letters*, 27:305–314, 1996.

[16] M. Milam and R.M. Murray. A testbed for nonlinear control techniques: The Caltech Ducted Fan. In *1999 IEEE Conference on Control Applications*, 1999.

[17] H. Pesch. Real-time computation of feedback controls for constrained optimal control problems. part1: Neighboring extremals. *Optimal Control Applications and Methods*, 10:129–145, 1989.

[18] H. Pesch. Real-time computation of feedback controls for constrained optimal control problems. part2: A correction method based on neighboring extremals. *Optimal Control Applications and Methods*, 10:147–171, 1989.

[19] M. Rathinam. *Differentially flat nonlinear control systems*. PhD thesis, Cal. Inst. of Tech., 1997.

[20] M. van Nieuwstadt. *Trajectory generation for nonlinear control systems*. PhD thesis, Cal. Inst. of Tech., 1997.

[21] A. Verma and J. Junkins. Inverse dynamics approach for real-time determination of feasible aircraft reference trajectories. In *Proc. AIAA Guidance, Control, and Navigation Conference*, pages 545–554, 1999.

[22] O. von Stryk and R. Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of Operations Research*, 37:357–373, 1992.