Exploring Quadrature Rules in Direct Collocation Methods for Trajectory Optimization

Brian Jackson

***Abstract*—This paper presents an introduction to direct collocation for trajectory optimization using both trapezoidal and Hermite-Simpson quadrature schemes. It sets forth a basic work flow for implementing direct collocation, which is applied to a simple example of controlling an inverted pendulum. The differences between the trapezoidal and Hermite-Simpson quadrature schemes are compared, along with the effect of increasing the number of segments.**

***Index Terms*—trajectory optimization, optimal control, direct collocation, tutorial**

I. Introduction

Nearly all robotic systems are mobile: they need to move around an environment in order to accomplish some task. In order to move efficiently and safely, autonomous robotic systems plan trajectories through their environment. The field of trajectory optimization focuses on finding these trajectories—which may be subject to constraints due to the environment (obstacles), dynamics, actuator limits, or energy reserves—while minimizing some cost function. Given the complex and nonlinear nature of the general form of these optimizations and the exact solution is often nearly impossible to compute, many approximate methods have been developed. In general, methods for solving the trajectory optimization can be categorized into direct and indirect methods. Due to their flexibility in explicitly handling different types of constraints, along with their relative robustness to initial guesses, direct methods are a common and powerful option. Direct collocation (or transcription) methods, in particular, have enjoyed considerable success and attention within the field of trajectory optimization [1], [2].

This purpose of this project is to investigate different quadrature rules used in direct collocation, and present the direct collocation method in an accessible manner for those who may be new to the field of trajectory optimization and the direct collocation method. Section II will briefly define the trajectory optimization problem, and Section III will describe how to apply direct collocation to trajectory optimization problems, using both trapezoidal and Hermite-Simpson quadrature rules. Pseudo-spectral methods will also briefly be discussed. Section IV will give a detailed example of applying direct collocation to control an inverted pendulum, and Section V will describe the results comparing the different quadrature rules. Lastly, some concluding remarks are given in Section VI.

Department of Mechanical Engineering, Stanford University

II. Trajectory Optimization

The trajectory optimization problem can be described as follows: determine the state-control function pair, $t \rightarrow (x \in \mathbb{X}, u \in \mathbb{U})$, and clock times t_0, t_f that solve

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & J(x,u,t_0,t_f) = E(x_0,x_f,t_0,t_f) + \int_{t_0}^{t_f} F(x(t),u(t),t)dt \\ \text{subject to} \quad & \dot{x}(t) = f(x(t),u(t),t) \\ & e^L \leq e(x_0,x_f,t_0,t_f) \leq e^U \\ & h^L \leq h(x(t),u(t),t) \leq h^U \end{aligned}$$

where $E(x_0,x_f,t_0,t_f)$ and $F(x(t),u(t),t)$ are the boundary and integral costs, $f(x(t),u(t),t)$ are the system dynamics, $e(x_0,x_f,t_0,t_f)$ are the boundary constraints, and $h(x(t),u(t),t)$ are the path constraints.

III. Direct Collocation

Direct methods, as opposed to indirect methods, solve the trajectory optimization described above by discretizing the problem and converting it into a non-linear program. A non-linear program is typically formulated as follows:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & J(z) \\ \text{subject to} \quad & f(z) = 0 \\ & g(z) \leq 0 \\ & z^L \leq z \leq z^U \end{aligned}$$

A variety of software packages exist for solving non-linear programs, such as SNOPT, IPOPT, or MATLAB's `fmincon` function.

Direct collocation discretizes the problem in time into N segments with $N+1$ collocation points as follows:

$$\begin{aligned} t &\rightarrow t_0, \ldots, t_k, \ldots, t_N \\ x &\rightarrow x_0, \ldots, x_k, \ldots, t_N \\ u &\rightarrow u_0, \ldots, u_k, \ldots, u_N \end{aligned} \tag{1}$$

The continuous functions of time, states, and controls are approximated as piecewise polynomials, i.e. splines. The points at which the polynomials connect are referred to as knot points. In the traditional methods detailed in this paper, the collocation and knot points coincide; however, this is not necessarily the case. The order of the approximating polynomials is determined by the quadrature rule employed by the collocation.

A. The Collocation Process

The steps of the collocation (or transcription) process can be summarized as follows:

1) Discretize time
2) Choose initial guess
3) Create the augmented state
4) Calculate the constraints
5) Solve the NLP
6) Interpolate

1) Discretize time: This consists of simply selecting $N+1$ points in time. For simplicity, these points are generally uniformly spaced. This discretizes the problem as shown in Eq. 1.

2) Choose initial guess: Both indirect and direct methods require an initial guess for the solution. It is usually sufficient to guess a linear interpolation from beginning to final conditions. For more ideas on initialization, see [2].

3) Create the augmented state: The state variable z passed into the NLP solver is the augmented state of all of the decision variables, including the initial and final time, state, and control values at each of the collocation points:

$$z = [t_0, t_f, x_0, u_0, x_1, u_1, \ldots, x_k, u_k, \ldots, x_N, u_N]$$

For ease of implementation, it is good practice to define functions for the mappings $z \rightarrow \{(t_i, t_f), (x_0, \ldots, x_N), (u_0, \ldots, u_N)\}$ and $\{(t_i, t_f), (x_0, \ldots, x_N), (u_0, \ldots, u_N)\} \rightarrow z$.

4) Calculate the constraints: The path and boundary constraints are straight-forward to formulate at each of the collocation points (i.e. for each of the augmented states). The differential constraints imposed by the system dynamics, however, are determined by the quadrature rule selected. Details on formulating these constraints for trapezoidal and Hermite-Simpson quadratures are given in Section III-B.

5) Solve the NLP: With the augmented state and constraints formulated, the only missing piece to formulate the NLP is the objective function. The objective function is formulated as follows:

$$J = E(x_0, x_N) + \sum_{k=0}^{N} w_k F(x_k, u_k, t_k) \tag{2}$$

where w_k are the weights at each collocation point, determined by the quadrature scheme.

6) Interpolate: The NLP solver will solve for the state and control values at the collocation points. However, the collocation grid often too coarse for fine-tuned control. The values between collocation points can be approximated by interpolating over the piece-wise polynomials defined by the quadrature scheme. It is important to note that the quadrature defines a n^{th} order polynomial over $\dot{x}$ and u. This results in an $(n+1)^{\text{th}}$ order polynomial for the state values.

B. Quadrature Schemes

Quadrature rules determine 3 aspects of the collocation process: 1) formulation for the dynamic constraints, 2) weights applied on the objective function, and 3) interpolation functions. Each of these will be shown for both trapezoidal and Hermite-Simpson quadrature schemes. Lastly, pseudospectral methods will be briefly described. In the following formulas, let $\Delta t_k := t_{k+1} - t_k$, and $\tau := t - t_k$.

1) Trapezoidal: For a generic integral, the trapezoid approximation is:

$$\int_a^b f(x) \approx \frac{\Delta t}{2}(f(a) + f(b))$$

Dynamic constraints: We can express the dynamics constraint as follows:

$$\begin{aligned} \dot{x} &= f \\ \int_{t_k}^{t_{k+1}} \dot{x} dx &= \int_{t_k}^{t_{k+1}} f dt \\ x_k - x_{k+1} &\approx \frac{\Delta t_k}{2}(f_k + f_{k+1}) \end{aligned} \tag{3}$$

which is applied to all collocation points for $k \in 0, 1, \ldots, (N-1)$.

Weights: The objective function is approximated as:

$$\begin{aligned} \int_{t_0}^{t_f} F(x(t), u(t), t) dx \quad &\approx \sum_{k=0}^{N-1} \frac{\Delta t_k}{2}(F_k + F_{k+1}) \\ &\approx w_N F_N \sum_{k=0}^{N-1} w_k F_k \end{aligned} \tag{4}$$

where $w_0 = \frac{1}{2}, w_1, \ldots, w_{N-1} = 1, w_N = \frac{1}{2}$.

Interpolation: For the control values, we can interpolate using a simple linear interpolation for $t \in [t_k, t_{k+1}]$:

$$u(t) \approx u_k + \frac{\tau}{\Delta t_k}(u_{k+1} - u_k)$$

For the state values, we have the following from the collocation equations: $f(t) = \dot{x} \approx f_k + \frac{\tau}{\Delta t_k}(f_{k+1} - f_k)$. To interpolate the state we integrate and apply initial conditions $x(0) = x_k$:

$$x(t) \approx x_k + f_k \tau + \frac{\tau^2}{2\Delta t_k}(f_{k+1} - f_k)$$

2) Hermite-Simpson: For a generic integral, the Simpson approximation is:

$$\int_a^b f(x) \approx \frac{\Delta t}{6}(f(a) + 4f(c) + f(b))$$

where $c := \frac{b-a}{2}$ is the midpoint of the interval.

Dynamic constraints: Following an identical procedure to the trapezoidal method, we find:

$$x_k - x_{k+1} \approx \frac{\Delta t}{6}(f_k + 4f_{k+\frac{1}{2}} + f_{k+1})$$

With Hermite-Simpson we now constrain the midpoints of the intervals to get second-order approximation:

$$x_{k+\frac{1}{2}} = \frac{1}{2}(x_k + x_{k+1}) + \frac{1}{8}(f_k - f_{k-1})$$

This relation can either be substituted into Eq III-B2 to get *compressed form*, or used as additional collocation points *separated form*. This paper will assumed separated form.

Weights: The integral term in the cost function can be approximated as:

$$\int_{t_0}^{t_f} F(x(t), u(t), t)dx \approx w_N F_N \sum_{k=0}^{N-1} w_k F_k$$

with $w_0 = \frac{1}{3}, w_{2,4,\ldots,N-3,N-1} = \frac{4}{3}, w_{3,5,\ldots,N-2} = \frac{2}{3}, w_N = \frac{1}{3}$

Interpolation: The derivations for the interpolation equations for quadratic and cubic are not given here, but are given in [2], and stated here for reference:

$$u(t) = \frac{2}{\Delta t_K^2}(\tau - \frac{\Delta t_k}{2})(\tau - \Delta t_k)u_k - \frac{4}{\Delta t_k^2}(\tau)(\tau - \Delta t_k)u_{k+\frac{1}{2}} + \frac{2}{\Delta t_K^2}(\tau)(\tau - \frac{\Delta t_k}{2})u_{k+1} \quad (5)$$

$$x(t) = x_k + f_k\frac{\tau}{\Delta t_k} + \frac{1}{2}(-3f_k + 4f_{k+\frac{1}{2}} - f_{k-1})\Big(\frac{\tau}{\Delta t_k}\Big)^2 + \frac{1}{3}(2f_k - 4f_{k+\frac{1}{2}} + 2f_{k+1})\Big(\frac{\tau}{\Delta t_k}\Big)^3 \quad (6)$$

3) Pseudospectral Methods: One can readily assume that better accuracy in collocation methods can be achieved by increasing the order of the polynomial basis functions. This is accomplished using orthogonal polynomials, typical of the Chebyshev or Legendre type. These arbitrarily high-order polynomials can either be applied between collocation points, or can be applied globally to fit the entire trajectory, when it is then referred to as pseudospectral methods [3].

Grid selection in pseudospectral methods is non-trivial, since the specific points and their assigned weights can significantly affect the numerical performance of the algorithm. Uniform grids, for example, exhibit extremely poor performance. Typical grids include Gauss, Radau, and Lobatto points. Essentially these variants change whether or not points are placed at the endpoints of the spline. For Lobatto points, collocation points are placed at both of the endpoints; therefore, trapezoidal and Hermite-Simpson methods can both be categorized as Lobatto.

One of the most important reasons to use pseudospectral methods is for their convergence characteristics. It can be shown that the convergence rate is exponential in order of the polynomial [2], under certain conditions. These methods have been employed on a variety of systems with significant success. In 2006, pseudospectral control methods were used on the International Space Station for a certain attitude adjustment maneuver, and are particularly well-suite for aerospace applications [3].

IV. Example Application

A. Setting up the problem

For a concrete example, direct collocation is implemented on an inverted pendulum with both trapezoidal and Hermite-Simpson quadrature rules. The goal is to move the system from the origin to $x = 0.5$ and $\theta = \pi$ in 2 seconds, minimizing energy. The horizontal movement is constrained within $-0.6 \leq x \leq 0.6$, with forces under 30 N.

1) Discretize time: The time was discretized into N segments of equal length from 0 to 2 seconds.

2) Choose initial guess: The initial guess used a simple linear interpolation from the initial to final state at each of the collocation points.

3) Create augmented state: Since there are 4 states in the inverted pendulum problem, $\mathbf{x} = [x, v, \theta, \omega]^T$, and 1 control, the augmented state was created by stacking each of the N states and controls, along with the initial and final time, into a vector of length $2 + 5(N + 1)$ for the trapezoidal method. For the Hermite-Simpson method, the separated form was used so there were $2 + 5(2N + 1)$ states.

4) Calculate the constraints: The constraints were calculated according to the forms given in Section III-B.

5) Solve the NLP: MATLAB's fmincon solver was used to solve the NLP. The objective function summing the weighted square of the control values was passed in with the initial state, along with the `lb` and `ub` parameters, both of which were of the same size as the state vector. The dynamics constraints were implemented using the `nonlcon`, argument, a function handle which returned a $4(2N+1)$ length vector in `Ceq`. The problem was solved using the default interior-point algorithm.

6) Interpolate: After getting the results at the collocation points from fmincon, interpolation function for both controls and states were used to find values between collocation points.

B. Error Analysis

There are several different ways of quantifying error in collocation methods, but a common one is to see how well the problem satisfies the dynamics between collocation points. Let $\hat{x}(t)$ and $\hat{u}(t)$ be the interpolated state and control values at an arbitrary time t. We can evaluate the dynamics using these estimated value as $\hat{f}(\hat{x}, \hat{u})$. We can then compare this to the dynamics subject to the spline interpolation directly on the dynamics values by calculating $f_k(x_k, u_k)$ at each collocation point. The intermediate values can then be interpolated using the same interpolation used for the controls (linear order for trapezoidal, quadratic for Hermite-Simpson). We will denote these as $\hat{\dot{x}}(t)$. The error can then be expressed as:

$$\epsilon(t) = \hat{\dot{x}} - \hat{f}(\hat{x}, \hat{u})$$

which can then be used to calculate a total error between each collocation point:

$$\eta_k = \int_{t_k}^{t_{k+1}} |\epsilon(\tau)| d\tau \quad (7)$$

which is typically evaluated using numerical integration with Rhomberg quadrature.

V. Results

Figures 1 and 2 give some results from solving the problem with Hermite-Simpson quadrature and 20 segments. Figure 1 shows frames from the motion of the inverted pendulum. It can be easily seen that the behavior matches the expected behavior when the goal is placed close to the right-hand boundary in x. Figure 2 shows the state values along with collocation errors for each segment, as calculated by Eq. 7. It's easy to see that

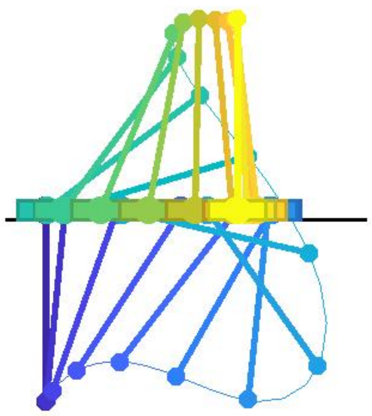

Fig. 1. Trajectory for the inverted pendulum

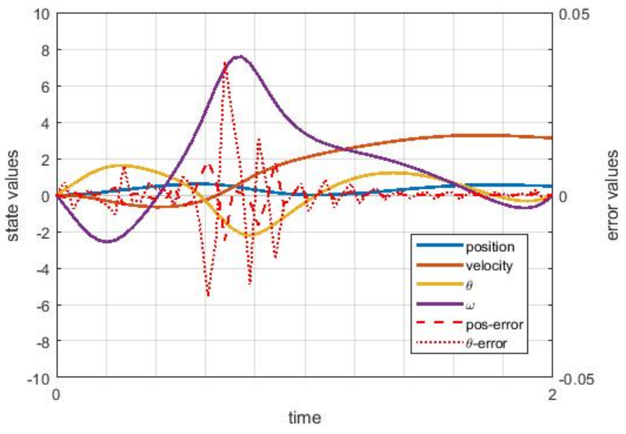

Fig. 2. State and state error values. Note that the maximum error corresponds to states with high velocities

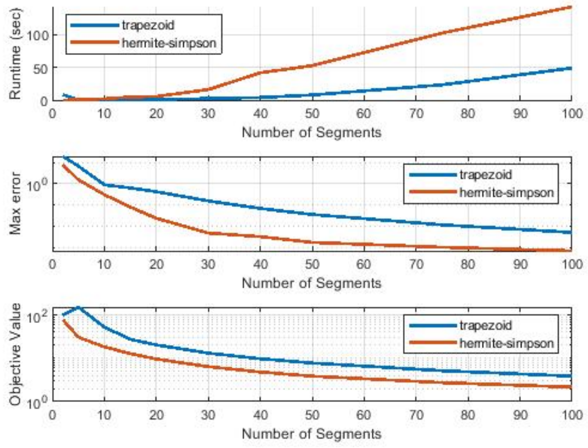

Fig. 3. Comparison between quadrature rules for a varying number of segments

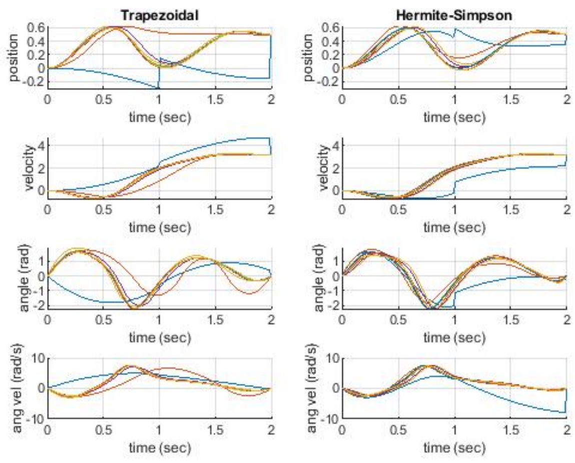

Fig. 4. Overlay of trajectories when solved with the different quadrature schemes and different number of segments. As the number of segments increases, the two quadrature schemes approach the same solution. However, for solutions with few segments (the blue and light red lines) the solution is significantly different.

the maximum errors coincide with high velocities, particular angular velocities. It would therefore be reasonable to refine the collocation mesh at those points to reduce errors.

Figures 3 and 4 show results from comparing the two quadrature schemes with an increasing number of segments (note that since the separated form of Hermite-Simpson collocation was used, the number of collocation points is effectively doubled over that of the trapezoidal method for the same number of segments). The results match the expected behavior, since the trapezoidal methods run significantly faster but at the cost of higher error and objective values. Figure 4 shows that having too few segments will result in poor trajectories that don't match the expected behavior. This is also shown by the relatively sharp decreases in error and objective value when increasing the number of collocation points when there are few points.

Lastly, I tried initializing the solver with the results of previous runs. In most cases, the time savings was not significant, which was surprising. This could be due to some of the tolerance values in the solver, and the fact that very few of the collocation points will match when increasing the number of points and using a uniform grid.

VI. Conclusion

This project offers an overview of implementing direct collocation from scratch, and is written with the intent of helping others will relatively little introduction to collocation methods gain a better understanding of the methods and how to implement them. I definitely have gained a much deeper understanding of how these algorithms work and clarified many points that were very unclear to me after the lecture on collocation. I look forward to taking this knowledge and applying it to more difficult and meaningful optimal control problems in the future.

Acknowledgment

I would like to acknowledge the author of [2], Matthew Kelly, whose tutorial and code served as the basis for my own implementation. I would recommend his tutorial and code to any student seeking to gain a better understanding of direct collocation.

The code for this project is located at:
https://github.com/bjack205/DirectCollocation.

References

[1] C. R. Hargraves and S. W. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *Journal of Guidance, Control, and Dynamics*, vol. 10, no. 4, pp. 338–342, 1987.

[2] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[3] I. M. Ross and M. Karpenko, "A review of pseudospectral optimal control: From theory to flight," *Annual Reviews in Control*, vol. 36, no. 2, pp. 182–197, 2012.