Trajectory Optimizaiton: Cart-pole Swing Up

Maxwell Urbani
School of Engineering and Applied Science
Harvard College
urbani@college.harvard.edu

Kahlil Wassel
School of Engineering and Applied Science
Harvard College
kahlilwassell@college.harvard.edu

Abstract— This document provides a brief overview of the steps taken to create a Trajectory Optimization simulation for the cart-pole up swing maneuver. It contains a motivation and description of the task to be undertaken, the approach to research and current examples of this kind of program, what was discovered throughout the process of implementing the program, and some reflections and analysis after having written and run the simulation.

I. MOTIVATION

Trajectory Optimization is a widely applicable tool and active area of research for many different branches of science. Trajectory Optimization is useful in as varying spaces as planning the trajectory for a rocket travelling from Earth to Mars and calculating the optimal gait for a walking robot. The motivation to study the cart-pole example was to under-stand how to implement a nonlinear program for a relatively complex system. This example could be used as a baseline with which one could understand trajectory optimization in a broader context. In particular we wanted to gain a better understanding of Trajectory Optimization because of its applicability to robotic manipulation. These techniques can be found in calculating the optimal gait for a walking robot, and even in planning the path of many industrial manipulators. This exercise would leave us better equipped with the tools to understand the programming tools necessary for more com-plex robotic systems such as manipulators with much higher degrees of freedom. Additionally Trajectory Optimization as a technique offers many benefits that other path planning techniques do not.

II. TERMINOLOGY

- Decision Variable - A variable being adjusted to satisfy the problem constraints and minimize the objective function. [1]
- Objective Function - The function being minimized in the optimization. (Mathematically describes the optimal trajectory) [1]
- Feasible trajectory - A Trajectory that satisfies the problem constraints [1]
- Nonlinear Program - Class of constrained optimization problems where the constraints or the objective function are nonlinear. [1]
- Optimal trajectory - The Trajectory that satisfies the problem constraints while also minimizing the Objective Function. [1]

III. DESCRIPTION

This project's scope and ultimate goal was to devise and simulate a trajectory for the Cart-pole swing up maneuver using the least control force on the cart. Throughout the creation of this simulation we hoped to be able to better understand the process of Trajectory Optimization, along with the nonlinear program techniques used to solve and simulate these problems. This particular system has two degrees of freedom, the cart position along the track, q1, and the angle of the pendulum relative to the track, q2. The simulation created describes the series of states that takes the Cart Pole system from the initial state, completely at rest with the pole free hanging vertically below the cart, q1 = 0, q2 = 0 , q1dot = 0 and q2dot = 0, to the final state with the pole at rest balanced vertically above the cart, q1 = 0.8, q2 = p, q1dot = 0 and q2dot = 0. The dot representing the time derivative of each degree of freedom. The simulation animates a graphic displaying the motion of a Cart Pole system as it performs this up swing with the least force on the cart, and plots the path followed by the pendulum during this maneuver. After setting up the nonlinear problem, we used Matlab's Nonlinear Program solver fmincon to solve for a discrete set of states to get us from our initial to final state, which will be discussed in the implementation section of this paper. Much of our work involved setting the problem up in such a way that fmincon could solve the system and then evaluating the results. Additionally this example allows us to evaluate the results by using a different set of options within the fmincon solver, which will be discussed in more detail in the Analysis and Results sections of this paper.

IV. APPROACH

a. Choosing the topic

We initially chose this topic for a final project because it engages with and practice methods discussed in the course such as system dynamics, coordinate transformations, and trajectory planning, while at the same time expanding our class discussions into new territories. One idea mentioned in our course discussions several times, though not in great detail, was using numerical approaches rather than analytical, for solving complex problems such as inverse kinematics. Thus, in our approach to solving an optimized trajectory problem, we were happy to learn about nonlinear programming and Matlab's numerical solver fmincon. We were also drawn to this problem as interesting combination of it being unstable but controllable. This project would then serve as a foundation for further development of a controller.

b. Research

We found that there are generally two methods for solving optimization problems, indirect and direct methods. Essentially, indirect methods optimize and then discretize, while direct methods discretize and then optimize. Indirect methods usually provide more accurate results, but they also are more difficult to describe, and have a smaller region of convergence, requiring a better initialization. Due to these challenges, we chose a direct method for our first project in optimization. Due to us not having a good initial guess, we found direct collocation more appropriate than direct shooting methods. Our approach to this trajectory optimization problem was based on a paper outlining the method of direct collocation by Matthew Kelly [1].

V. IMPLEMENTATION

a. Process Outline

The process we devised to implement the method of direct collocation is as follows,

i. Derive the system dynamics.

ii. Discretize the trajectory (generates collocation points). Convert the system dynamics into a set of constraints to be applied at the collocation points.

iii. Determine an appropriate objective function.

iv. Pass the dynamics constraints, initial and final state constraints, initial guess, and objective function into the nonlinear solver and receive an optimized evolution of state.

v. Experiment with these factors and the options of the solver to hone in on the most accurate solution.

1. System Dynamics.

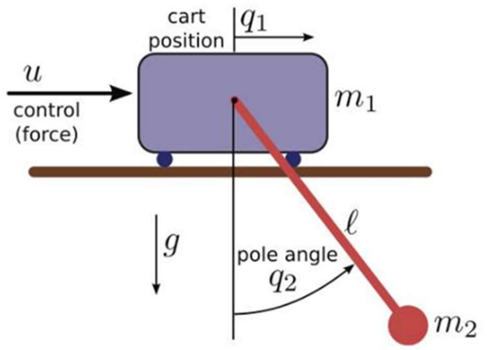

Fig. 1. Physical model of the cart-pole problem. The pendulum is free to rotate about its attachment point on the cart. [1]

The system dynamics for the cart-pole system are most easily determined using the Lagrange method. From this method we derived the dynamics to be,

$$\ddot{q}_1 = \frac{\ell\, m_2 \sin(q_2)\, \dot{q}_2^2 + u + m_2\, g\, \cos(q_2)\, \sin(q_2)}{m_1 + m_2\left(1 - \cos^2(q_2)\right)}$$

$$\ddot{q}_2 = -\frac{\ell\, m_2 \cos(q_2)\, \sin(q_2)\, \dot{q}_2^2 + u\, \cos(q_2) + (m_1 + m_2)\, g\, \sin(q_2)}{\ell\, m_1 + \ell m_2\left(1 - \cos^2(q_2)\right)}$$

This is one of the most essential parts of the process.

2. Discretize the trajectory and dynamics constraints.

We used the variable K to represent the resolutions of our discretization. The core idea of direct collocation is to discretize the trajectory into a set of points and tell the solver that the system dynamics must be satisfied at every collocation point. Mathematically this says that the change in state between two collocation points is equal to the integral of the state derivative over that region. If this constraint is met, than the overall trajectory will approximately follow these dynamics. To evaluate the integral of the state derivative, we used the Euler integration approximation. Other common approximations used in direct collocation are trapezoidal quadrature, and the Hermite-Simpson method. Of the three, the method we used was the easiest to implement, but provides the least accuracy. Using quadratic approximations, the Hermite-Simpson method is the highest order accuracy of these three methods. While we like to think of this constraint as being the change in state must equal the change in system dynamics, the solver wants all terms on one side, equating to zero. The solver works by then taking the initial guess of the states evolution, and calculates this difference. Through iterations it drives this difference to zero, at which point the dynamics at each discretization of the state are met. These constraints are passed in as nonlinear equality constraints. We had no nonlinear inequality constraints.

3. Objective Function

The objective function is what is minimized in the whole process. Thus, the objective function is the description of optimality and will drive the evolution of state. Often it is possible that the objective function, given the nonlinear constraints, cannot be perfectly minimized. Thus the program uses a region of convergence to decide when the objective function is at a minimum. This is an option that can be altered within solvers. We used 1e-3 as our optimality tolerance. In developing a program to simply run, we initially used an objective function that summed the force applied to the cart. This however, has the possibility of being minimized through high magnitude positive and negative forces that cancel out. After getting our program running, we then switched to a sum of squares, which should minimize the magnitudes of forces applied to the cart. This minimization could be useful for decreasing motor specs, or adapted to drive a trajectory which minimizes the energy consumed. One useful advantage for this objective function is that it has one global minimum. These programs can run into trouble when there are numerous solutions. One way to get around this is often to add in a term like the sum of forces squared, to force the objective function into having a unique global solution. This objective function also has the benefits of being smooth on all intervals, resulting in a relatively fast running program [1].

4. Other Constraints and Passing to Solver

We then developed our other constraints. Inherent to the physically of the system are that the track the cart rides on has finite length, and the motors can supply finite strength. More importantly are the state constraints. These determine the final result of the trajectory. We want our system to go from a stable state of rest to the unstable state of inverted balance. Our initial state was expressed as all zeros, (home position and no velocity, and our final state was described as .8 meters for the cart, an angle of pi (upright) for the pole, and velocities of 0, so that the pole is at rest in its inverted balance. The choice of .8 meters was somewhat arbitrary. This choice provided a good final solution for us. In another case, if the cart was desired to end at some other final state, than this parameter would be specified for that end, or in further development one might want to find way to leave this final position of the cart free for the program to manipulate in its optimization. We passed these to fmincon as linear equalities. The final parameter needed to pass into the solver to obtain a solution is an initial guess. Per recommendation of our reference paper ([1]), we used a linear interpolation between the start state and end stand. It is notable that this initial guess violates the collocation (system dynamics) constraints. This is okay though, as the nonlinear solver will drive this violation to zero. We also tried the initial guess of all zeros, which can sometimes return a better solution. The benefit of this guess is that, while it violates the linear constraint of initial and final states, it does not violate the nonlinear constraint of system dynamics. In the end we found that the linear interpolation provided a better result, though this quick trick, which we learned from Irina, is an easy enough thing to test with any trajectory. As mentioned earlier, the flexibility of this initial guess is the primary reason we chose direct collocation as our method for trajectory optimization. With direct collocation, as shown in our linear interpolation, can be highly inaccurate and still return good results. This makes direct collocation particularly useful when and initial guess at the trajectory is unknown or difficult to devise. When more accurate trajectories are desired, but an initial guess is unknown, one method that can be used is to first run direct collocation, and then use the trajectory returned from this process as an initial guess passed into other trajectory optimization methods which return more accurate solutions but require a better initial guess.

5. Tweaking

The final step of our implementation was to then tweak and experiment with the parameters passed into the solver to achieve the best solution possible. This is discussed in greater detail in the next section, discussing final results and analysis.

VI. RESULTS & ANALYSIS

1. Results

We were very happy to see that our program was able to converge to a unique minimum solution, providing a smooth and realistic trajectory, which can be seen in the animation. The specific output of our program is a set of vectors describing a discrete progression of states throughout this maneuver. Attached below is a figure that shows an example of the graphic that is generated along with a look at the numerical output of our simulation which comes in the form of a set of 1x5 vectors for each knot point (collocation point). It is important to note that the first entry in the state vector is the cart x position the second is the pendulum angle and the third and fourth entries are the time derivatives of the aforementioned quantities respectively. The last entry in the vector is the decision variable, in this case the force that is put on the cart.

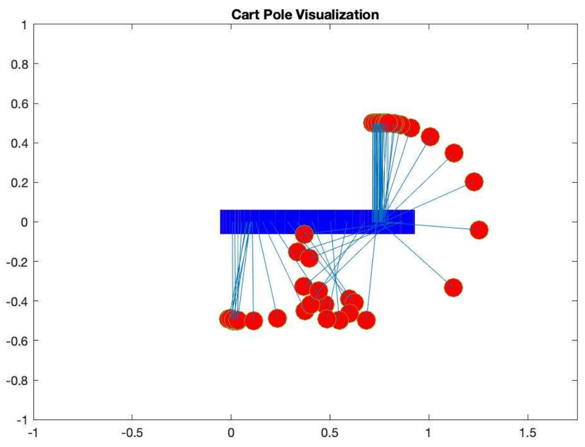

Fig. 2. Progression of Cart Pole Trajectory (40 Knot Points)

```
-0.1487    0.7975    3.7826    2.2086    0.9487
-0.1567   -0.0160   -6.0890    4.3043    2.7457
 0.5162    1.0949    3.1338   -1.4253   -1.1111
 0.0541    0.2973   -5.9733    8.1540   -0.7391
 0.0000    0.2040    0.1240    0.4615    0.3482
```

Fig. 3. Example Numerical Output (5 Knot Points)

2. Analysis

These results are very exciting as we have achieved the goal that we initially set for ourselves. One particularly interesting result that we find from our simulation is the importance of particular parameters. Our best results came from 40 knot points and with a time step(dt) of 0.125 seconds, for a total swing up time of 5 seconds. Changing these parameters changes the results of the simulation drastically. Initially we were not sure that our results were making sense, but we realized that we had far too large of a time step between Knot Points and also far too few Knot Points. After adjusting with these two parameters we got the result presented above.

Another parameter we tested was options argument in fmincon. Initially we began using the Interior Point method as suggested by Irina, but we also tried one other solver SQP, or Sequential Quadratic Programming. What we found after testing these two solvers was that the interior point method worked far better and created a series of states that appeared far more physical than SQP. SQP resulted in the Pendulum jumping suddenly in an unphysical manner. This illustrated how big of a difference that the solver type you use within your nonlinear program can make. The Interior Point method provided very smooth solutions that result in the maneuver we want whereas SQP provides far more jagged and even unphysical sets of trajectories. This makes sense as interior point methods by definition keep iterations within the feasible region, whereas SQP does not. In summary our solver seems to provide us with a physically accurate solution to the cart pole problem, but the simulation is highly dependent on particular parameters and lends itself to particular solving methods.

VII. REFLECTIONS

Throughout our exploration of Trajectory Optimization, Max and I were very lucky to have the opportunity to self-educate about a topic that went almost completely unstudied during the lessons of this course. It was quite a task to learn about the topic essentially from scratch, but also extremely rewarding.

In particular, for this project, we learned a lot about solving problems where you have a complex set of nonlinear constraints, a topic that neither of us had really explored before. Additionally, this project required us to learn a lot about the process with which you can discretize a smooth function to approximate it within a simulated environment. We learned a lot about the different types of collocation methods, particularly methods of direct collocation.

This project also had a rather steep learning curve with regard to coding. We spent a lot of time in the middle phase of our project trying to understand the current methods used in nonlinear programming. We focused our efforts on the function fmincon in Matlab and understanding how to set up this complex dynamic problem in such a way that fmincon could solve it.

Even with all the new knowledge that we acquired to tackle this project we did use many of the skills that we learned throughout the course. In particular we initially used a lot of our understanding of how path planning worked to inform our initial exploration of the problem. Additionally, our extensive use of Matlab throughout this course, both in lab and on problem sets enabled us to be able to take on such a complex and interesting topic and create a physically accurate simulation.

If there was one particular topic that would have been most helpful to have studied throughout the semester it would have been helpful to have been taught a bit more about the different types of path planning that are present and the fact that trajectory optimization is a very different numerical exercise from gradient descent. It also would have been helpful to have had some knowledge of what went into Trajectory Optimization. It is a truly interesting and applicable topic that we didn't get to chance to cover at all, yet is becoming increasingly important as robots become more Complex.

In the end, our project, simulating the Cart-Pole up swing maneuver as a nonlinear programming problem was an extremely rewarding experience. This project required Max and I to learn a lot about a topic we knew little about, yet also use many of the numerical and coding skills that we had gained throughout the semester in order to complete it.

Contributions

We essentially did the entire project together. We both compiled code and we also very collaboratively wrote this paper. In specific we both worked on the main function of the code,more specifics listed below.

Kahlil - CartPoleVisual function, KWandMUCartPole function, GetCPDynamics function, CPNonLinCon function, Results and Analysis section of paper, Description and Motivation sections of paper.

Maxwell - KWandMUCartPole function, GetXGuess, CPNonLinCon function, Implementation and Approach sections of paper.

Acknowledgment

We want to express our gratitude to Professor Rob Wood and Irina Tolkova for their continued support and guidance throughout the creation of this simulation, and helping provide us with resources throughout our exploration of this topic.

References

[1]. Matthew Kelly, “An Introduction to Trajectory Optimization : How to Do Your Own Direct Collocation”, Society for Industrial and Applied Mathematics, 2017.