Model Predictive Control on Drone + Pole System

6.8210 Underactuated Robotics Final Project

Vanessa Hernandez-Cruz
Department of Mechanical Engineering
vanessa7@mit.edu

Ishita Goluguri
Department of Computer Science
ishi@mit.edu

Rashmi Ravishankar
Aeronautical and Astronautical Engineering
rashmir@mit.edu

***Abstract*—This paper analyzes using Linear Quadratic Regulator (LQR) method versus Model Predictive Control (MPC) with Direct Collocation to stabilize a drone with a pole (DP) attached. The main goals of this project is to model, control, and simulate the dynamical system. This is an interesting modeling and control problem, as it combines the challenging problem of controlling an inverted pendulum with the rich dynamic behavior of the drone. There is no physical prototype associated with our project. A successful stabilization will take our system to the origin at an upright position for the pole. To aid with more open-source DRAKE-related code, the code is publicly available at this GitHub link.**

***Index Terms*—LQR, MPC, control, drone, DRAKE, pole, Direct Collocation**

I. Introduction

The drone dynamics were captured by a multibody plant in DRAKE.The pole was described with a custom URDF then attached using the MultiBody class in DRAKE. Stabilization in both cases, MPC and LQR, was done about a fixed point defined as the upright position for the pendulum and x-y-z positions at the origin. This is an interesting project because it allows familiarization w DRAKE and a close analysis of how LQR, MPC, and direct collocation can all drive the DP system to the goal states. Control techniques and optimization tools are in use. There was a major learning curve associated with using DRAKE; however, all members are more comfortable with using it and debugging successfully. Having to implement MPC from scratch was also an interesting task as a deeper understanding of trajectory optimization was needed.

II. Related Work

The problem of balancing an inverted pendulum on a quadrotor has been tackled a number of times, and a large variety of methods have been used. One of the most simplest methods has been implementing the well-known PID controller on the system to keep the pendulum upside down and the y-axis position of the quadrotor at zero steady state error [2]. A PID (Proportional-Integral-Derivative) controller takes inputs of the desired set-point and the current position, and ouputs a weighted error that has been multiplies by the P constant, and has an I and D term added to it. This combination of error weighting is more robust than any one of these coefficients acting alone. Despite its simplicity, one of the issues this method faces is that it works for only short periods of time, because the parameters (P, I, and D) are constant.

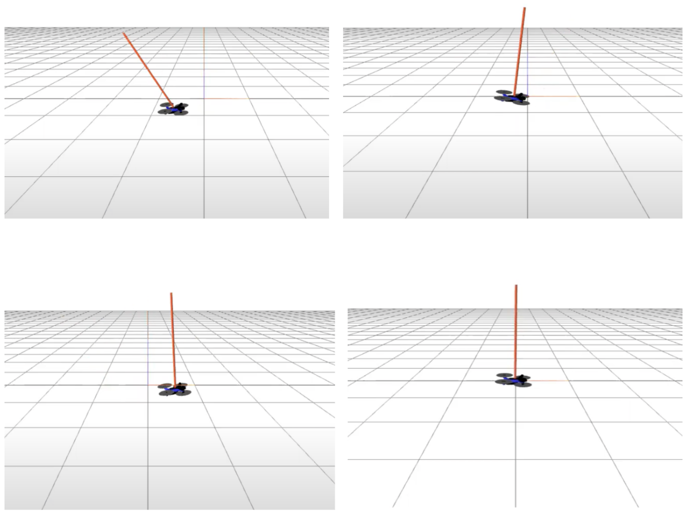
Fig. 1. Control of an inverted pendulum on a flying drone with actuation exclusively from the propellers - MPC in action from the initial state to final equilibrium

A second method implemented a PID controller to control the altititude of the quadrotor in the same way as the first, but differed in their control of the inverted pendulum [3]. For this, they used an LQR (Linear Quadratic Regulator), an optimal control technique that uses a quadratic cost function to design control signals for a linear dynamical system. One of the cons of this method was its sensitivity to parameters used, especially in the PID controller. We used this LQR method as a starting point for our experimentation, and compared with it. This project was implemented using the Drake software package, a collection of tools for analyzing controls and complex robot dynamics [4].

III. Methodology

To the best of our knowledge, there is no previous work using MPC with direct collocation on a DP system.

A. LQR

This control method, as the name suggests, uses a quadratic cost to penalize deviations from the desired points. An LQR controller takes in the system dynamics, Q, and R gains. The Q gains are used to regulate the states, and it is customary to have the positions-related states have a gain of 10 or more. For velocity-related states such as $\dot{\theta}$, it is typical to multiply those by one. The R gains specify the importance, or weight, that the input variables should have in the cost function.

While LQR is originally designed for linear systems, it can still be useful for nonlinear systems. By linearizing the nonlinear system around an operating point, we can approximate it as a linear system. LQR can then be applied to this linearized system to obtain a local approximation of the optimal control solution.To linearize the nonlinear system, we choose an operating point or equilibrium point around which the linearization is performed. In our case the equilibrium point happens to be the vertical position of the inverted pendulum.

The inputs of our system are the four propellers defined in the multibody plant of the drone that are treated as external forces. For the purposes of LQR, the values of the propellers are set to the DP's weight over four to create enough balance for the drone to hover. In the simulation, the context of the DP system is set to be offset from the origin and away from the upright position. With position-related gains of 10 in Q and the rest set to 1, the LQR controller was able to smoothly bring the DP system to the origin and upright position. It is important to note that this controller is not robust to any initial conditions.

An avenue we would like to explore in the future is a more robust LQR controller. Some possibilities to be explore are expanding the region of attraction for the LQR controller. This involves investigating techniques to increase the stability margins, enhance disturbance rejection capabilities, and optimize controller parameters. By broadening the region of attraction, the LQR controller can handle more diverse and challenging scenarios, making it a more versatile control solution for various applications. The focus of this project was to learn how to implement MPC and still drive the DP system to the origin and upright position, so making a robust LQR controller is a focus of future work.

B. MPC with Direct Collocation

The advantage of MPC is that it is solving an optimization problem, in this case in the form of direct collocation, at every time step, so the best next step is always taken. This process ensures that the optimal action and inputs are enforced. The basic recipe for MPC as described by Professor Russ Tedrake is: first, measure the current state, second, optimize a trajectory from the current state, third, execute the first action from the optimized trajectory, and fourth, let the dynamics evolve for one step and repeat [1]. Before implementing the MPC recipe described above, the optimization problem was solved in one shot with direct collocation: this helped us ensure that a solution was feasible.

Without MPC, the number of time samples was set to 21 which is directly correlated to the number of decisions variables. This was able to find trajectory that drove the DP system

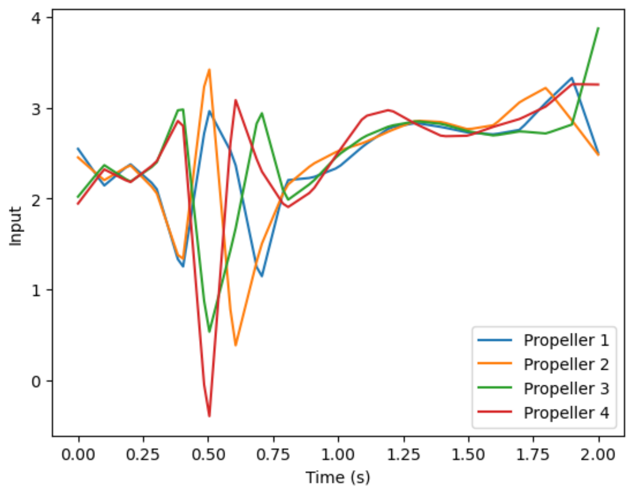

Fig. 2. Rapidly evolving propeller inputs/loads of the quadrotor plant under model predictive control(MPC) with direct collocation

from the initial state, [0.5,0.6,0.9,0,0,0,3*pi/4,0,0,0,0,0,0,0], to the final state where all states are zero in roughly 2 seconds. The optimization needed a minimum of 8 time samples to find a solution in the one shoot case. When implementing MPC, the final cost was not enforced given that we were not concerned with finding the fastest solution, yet. The only cost was the running cost which was a quadratic cost of the four inputs - the four propellers. Since an optimization problem is being solved at every time step, the number of time samples was reduced from 21 to 7 in the MPC formulation. This allowed for less decision variables overall for each optimization problem. However, it was really strange how the time for the DP system to go to the final state was achieved over the course of 0.2s. This, of course, does not include computation time, as MPC definitely took longer to compute a viable trajectory.

The only modification to go from direct collocation to MPC with direct collocation was reducing the time samples, changing the AddBoundingBoxConstraint from the initial state values to the current one in the for loop, and storing each optimal action to later plot. This would work best as an offline planner assuming no additional variables, such as obstacles or wind, are added to the system. For future work, we would investigate how to add dynamic obstacles into the simulation forcing a more robust and real-time analysis.

C. MPC with Direct Shooting

In our implementation of direct shooting, we began by linearizing the dynamics system. We get the coefficients for this linearization using the Drake FirstOrderTaylorApproximation() function. Then, the states $x[n]$ are expressed as functions of $x[0]$ and inputs $u[0]...u[n]$ and eliminated from the optimization problem. We can represent

$$x[n] = A^n x_0 + \sum_{k=0}^{n-1} A^{n-k-1} B u[k].$$

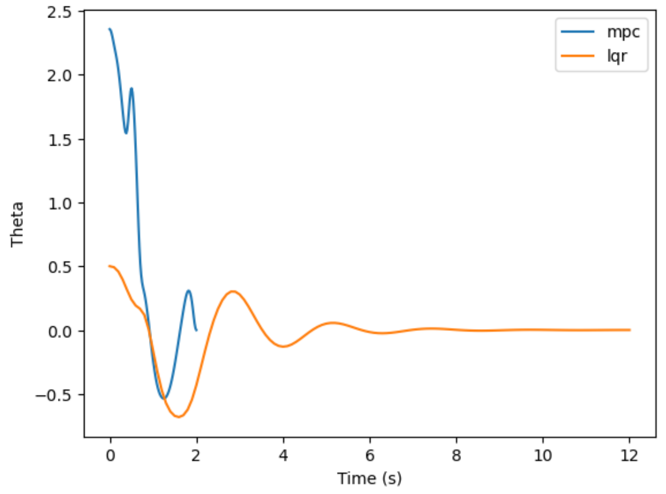

Fig. 3. Comparison of LQR and MPC stabilization trajectories. MPC control is significantly faster and more stochastic while LQR is slower and smoother. MPC is able to work with larger initial displacement

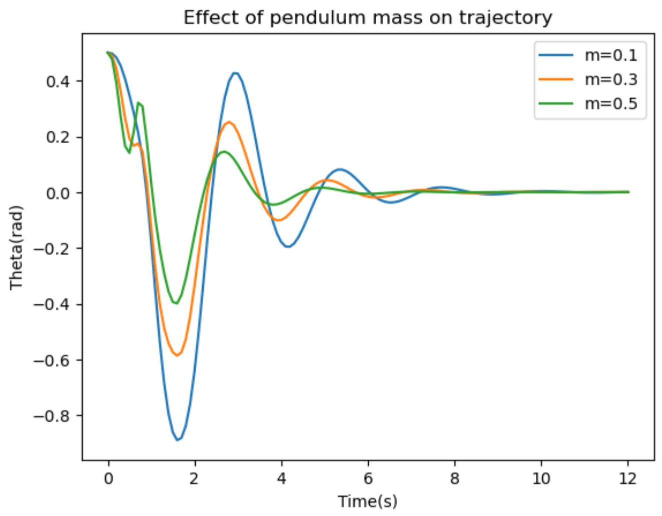

Fig. 4. The effect of pendulum mass on stabilization of the inverted pendulum. Interestingly we see that higher the mass, the quicker to equilibrium. Curiously enough, length of pendulum had no similar effect.

We can write this in matrix form as well as $x = \mathbb{A}x_0 + \mathbb{B}u$. We use the LQR objective in our computations. We can manipulate this objective to get our optimal cost. Once we get the optimal inputs, we can plug them back into the recursive linearized expression for $x[n]$ to get the next state.

We use the same steps for MPC outlined in the MPC with Direct Collocation section.The initial state is once again set to [0.5,0.6,0.9,0,0,0,3*pi/4,0,0,0,0,0,0,0]. At every step, direct shooting for 25 steps is computed. With a nonlinear system, direct shooting can be computationally intensive, but on a linear system, it runs efficiently. The direct shooting function outputs the optimal input signals and cost. The first input signal is used to compute the next step, and the state is updated. Then, direct shooting is run again for the entire horizon with the new state.

One challenge encountered during this implementation was an appropriate horizon for both the direct collocation and the appropriate amount of time for MPC. With horizons that were too short for the direct shooting, we saw that the cost shot up to infinity. Similarly, with time too long for MPC (for instance, 40 time steps instead of the 25 implemented right now), the same issue occurs. For direct shooting, we believe this is because the system doesn't have enough horizon to figure out the true optimal path. We are still unsure why the error occurs with MPC, and require further research. We also hypothesize that the choice of QN terminal cost matrix affects both these factors. Currently, the cost matrix is set to the cost matrix output by the Discrete Time LQR function. However, more experimentation and research is needed to find the optimal cost matrix here.

IV. Results

Post the successful implementation of MPC and LQR, we compared the different control approaches implemented. The first clear observation is that MPC gets to equilibrium significantly quicker, as can be seen in figure 3. We observe that LQR trajectory resembles a slow, smooth, damped oscillation. MPC trajectory (see Fig 3) and control (see Fig 2) look rapid and non-differentiable/stochastic in nature. Figure 2 visualizes propeller thrusts (inputs for MPC) which further validates the trend.

Though MPC outperforms LQR on nearly all metrics, a significant caveat is that MPC is constrained by computation time. So while MPC control is "fast" in the sense that it is an extremely effective controller and brings teh system to equilibrium quickly, the runtime is not compatible with its use in real-time. MPC should be used preferably as an offline controller because it is held back by computational expensiveness.

We were also curious to see how system variables and initial conditions affect control and trajectory. We varied the length of the pendulum and the mass of the pendulum to see what happens. Both gave interesting insights. Seen in Figure 4 is the plot of pendulum theta with time (trajectory of the inverted pendulum before it reaches equilibrium at $\theta = 0$). The mass fraction of the pendulum was varied between 0.1 and 0.5 keeping the total weight of the DP system constant. Mass clearly affects the trajectory with greater mass being easier to control and quicker to equilibrium which is interesting to note. The length of the inverted pendulum was varied around 1.0, going between 0.5 and 2. Length had no effect whatsoever since varying lengths produced the exact same theta values on each run with different lengths.

Conclusion and Future Work

In conclusion, we were able to solve the drone pole problem with both a classical control approach as well as a reinforcement learning approach, successfully implementing:

1) LQR

2) MPC with direct shooting
3) MPC with direct collocation

in Python, using DRAKE for modeling and simulating the multi-body drone-pole system, URDFs for designing and linking the physical system, and Meshcat for visualizing the performance of our controllers.

An avenue we would like to explore in the future is a more robust LQR controller. Some possibilities to be explore are expanding the region of attraction for the LQR controller. This involves investigating techniques to increase the stability margins, enhance disturbance rejection capabilities, and optimize controller parameters. By broadening the region of attraction, the LQR controller can handle more diverse and challenging scenarios, making it a more versatile control solution for various applications. The focus of this project was to learn how to implement MPC and still drive the DP system to the origin and upright position, so making a robust LQR controller is a focus of future work.

There is also significant scope to optimize MPC control and upgrade it to perform faster, cheaper, and under more uncertainty. A challenge encountered during the implementation of MPC was finding an appropriate horizon for both the direct collocation and the appropriate amount of time for MPC. With horizons that were too short for the direct shooting, we saw that the cost shot up to infinity. Similarly, with time too long for MPC, the same issue occurs. We are still unsure why the error occurs with MPC, and require further research. We also hypothesize that the choice of QN terminal cost matrix affects both these factors. However, more experimentation and research is needed to find the optimal cost matrix here.

Contributions

All team members worked together to render the DP system in simulation. This involved building the appropriate connections in MeshCat and having an accurate URDF for the pole.

Vanessa figured out how to drive the DP system from a random state to the origin and upright position with LQR, Direct Collocation, and MPC with Direct Collocation. In addition, she also successfully got the simulations of the paths taken associated with each method described above. In addition, she worked on the abstract, introduction, and methodology of the report.

Rashmi figured out how to adapt the quadrotor code to the DP problem, automated and functionalized code for the dynamic creation of URDFs, and implementation of LQR and MPC for analytics. She also generated the plots and figures in the report.

Ishita figured out how to implemented Direct Shooting and MPC with Direct Shooting. In addition, she also worked on the related works section of the paper.

Acknowledgment

Special thanks to Meenal Parakh for her extra time dedicated to helping with the simulations, to Richard for insightful debugging tips, and to Sunbochen Tang for his time dedicated to figuring out the direct shooting horizon error.

References

[1] Russ Tedrake. Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation (Course Notes for MIT 6.832). Downloaded from https://underactuated.csail.mit.edu/

[2] Gunawan, W.W. Likafia, A. Joelianto, Endra Widyotriatmo, A.. (2018). Inverted Pendulum stabilization with flying quadrotor. Internetworking Indonesia Journal. 10. 29-35.

[3] Wang, Hongliang Dong, Haobin He, Lianghua Shi, Yongle Zhang, Yuan. (2010). Design and Simulation of LQR Controller with the Linear Inverted Pendulum. Proceedings - International Conference on Electrical and Control Engineering, ICECE 2010. 10.1109/iCECE.2010.178.

[4] Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics. 2019