CALIFORNIA INSTITUTE OF TECHNOLOGY
Computing and Mathematical Sciences

CDS 110

Eric Mazumdar
Fall 2024

Problem Set #7

Issued: 12 Nov 2024
Due: 19 Nov 2024

Problem 1. Introduction to Optimal Trajectory Generation (10 pts)

Over the course of this homework, you will learn the fundamentals of generating trajectories using optimization. Let's start with an introduction to one of the most popular nonlinear optimization solvers: CasADi. Note that if you have not used Casadi before, that's okay! There is a ton of documentation, and problem 1 will hopefully help with grasping the fundamentals. All of the skeleton code for every problem of the homework is in the following colab document here. We recommend you make a copy of the notebook and work there!

For the first problem, consider a 1-dimensional double integrator system. This could be a spacecraft with one engine, a cart on a frictionless rail, or really any system that has a physical instantiation. The Euler-discretized discrete-time dynamics are therefore

$$\begin{bmatrix} x_{k+1} \\ \dot{x}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix} + \begin{bmatrix} 0 \\ \Delta t \end{bmatrix} u_k \tag{1}$$

Let's also set an input limit of $|u| \leq 1$, to make the system more realistic (can't burn the engine infinitely fast, etc). The starting point for the system is at $x_0 = \dot{x}_0 = 0$, and we want to get to $x_f = 10, \dot{x}_f = 0, t_f = 7$. In other words, we want to come to a stop at the point $x = 10$ in 7 seconds. Additionally, we use the timestep $\Delta t = 0.01$. Finally, we want to minimize

$$J = \sum_{k=0}^{n-1} u_k^2 \tag{2}$$

The trajectory optimization solution is presented in the first section of the colab notebook. Read through it and make sure you understand what it's doing. Try a few different values of q and r and see how they affect the solution. Once you've done that, you're done, no deliverable required! Who doesn't love a few free points?

Problem 2. Transcriptional Regulation (30 pts)

Let's now try the same optimization based trajectory generation using a different system to see how else it could be applicable outside of physical systems. Consider a genetic circuit consisting of a single gene.

$$\begin{bmatrix} \dot{m} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} \alpha_0 - \delta m \\ \kappa m - \gamma p \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} v \tag{3}$$

We see from this model that each of the mRNA (m) and the protein (p) have decoupled dynamics that experience exponentially decay. However, using our input v, we can increase the transcription of mRNA, and therefore increase the production of the protein. Finally, α_0 is the constant base transcription rate, δ and γ are the decay rates of the mRNA and protein respectively, and κ is the speed at which the mRNA produces the protein.

(a) Discretize the continuous time dynamics using Euler discretization.

(b) Now, we want to produce a certain amount of protein quickly in response to rapidly changing external conditions for our organism. Let $\alpha_0 = 1, \delta = 1, \kappa = 1, \gamma = 1$ for simplicity. Over the course of 5 seconds, starting at $m_0 = p_0 = 0$, we need to create 5 units of protein, i.e. $p_f = 5$, but we don't want to keep making the protein afterward so $m_f = 0$. Additionally, $|v| \leq 10$ as we only can command so much mRNA to be created. Finally, we want to minimize the square of the input similar to problem 1. Use CasADi to find an optimal trajectory for the mRNA input and plot both the input and the states.

Problem 3. Trajectory Generation for the Bicycle Model (60 pts)

In the last couple problem sets, you experienced tracking trajectories for a reasonably detailed reduced order model of a vehicle. Here, we're going to find out how to generate those trajectories around obstacles!

Consider the following kinematic model of a vehicle's dynamics: the (hopefully familiar) bicycle model!

$$\begin{aligned} \dot{x} &= \cos(\theta) v_x \\ \dot{y} &= \sin(\theta) v_x \\ \dot{\theta} &= \frac{v_x}{L} \tan(u_\delta) \\ \dot{v_x} &= u_v \end{aligned}$$

Note that this differs from the one you saw in previous problem sets as this is the *kinematic* model instead of the *dynamic* model that you used in prior homeworks. Because of the computational complexity of nonlinear optimization, we often use simpler models for the trajectory/planning part of the control hierarchy, then use the more complex models to track these trajectories. The states are the x and y position, heading, and forward velocity, L is wheelbase, and inputs are u_v, the accelerator, and u_δ, the steering angle.

(a) As per usual, we will be using the discrete time form of the dynamics for our simulation. Discretize the bicycle model dynamics using Euler discretization.

(b) Now that we have the discrete time dynamics, we can go ahead and try to find trajectories. In our case, we will have n circular obstacles centered at (x_k, y_k) with radius r_k. We need to encode in our trajectory optimizer that we cannot pass through these circles. Write out what the constraints will need to be in the optimization in order to avoid obstacle collisions at any time step.

(c) Using the trajectory optimization framework presented in class, write out the complete optimization problem with the extra constraints to account for the obstacles.

(d) Time to put it in practice! Use CasADi to find the optimal trajectory from the start to the goal state in the presented colab notebook. Note that we only require that the car's position is at the goal, we do not care about the heading or the wheel direction, so this should be reflected in your cost function. Additionally, there is only one obstacle, as nonlinear optimization can be quite slow, and running the optimization with multiple obstacles on google colab can take quite a while (but it should be clear to you how to extend your solution to multiple obstacles). Plot the final trajectory.