C106B Discussion 4: Model Predictive Control

1 Introduction

Today, we'll talk about:

1. Optimization Problems & Notation
2. Model Predictive Control
3. Constrained Model Predictive Control

2 Optimization Problems

In general, optimization problems seek to find the minimum of a cost function, $f(x)$, as a function of a decision variable, x, subject to some constraints.

$$\min_{x \in \mathcal{D}} f(x) \tag{1}$$
$$\text{s.t. } g(x) \leq b \text{ Inequality Constraint} \tag{2}$$
$$a(x) = c \text{ Equality Constraint} \tag{3}$$

If we specify *arg min* instead of min, the solution to the optimization problem is the *value* of the decision variable that minimizes f, subject to the constraints.

$$x^* = \arg\min_{x \in \mathcal{D}} f(x) \tag{4}$$
$$\text{s.t. } g(x) \leq b \text{ Inequality Constraint} \tag{5}$$
$$a(x) = c \text{ Equality Constraint} \tag{6}$$

Optimization constraints *must* be a function of the decision variable! Otherwise, they won't constrain the solution to the optimization problem.

3 Model Predictive Control

Can we solve path planning and feedback control with a single optimization problem? Model predictive control offers us a way to approach the two through a single optimization. We optimize a cost function over a horizon, N, which allows us to plan N steps into the future.
Imagine that we want to drive the discrete time nonlinear system:

$$x(k+1) = f(x(k), u(k)) \tag{7}$$

To a desired state x_d. The following is a common formulation of the model predictive control problem for such a system:

$$x^*, u^* = \arg\min_{x,u} \, (x_N - x_d)^T P(x_N - x_d) + \sum_{k=0}^{N-1} [(x_k - x_d)^T Q(x_k - x_d) + u_k^T R u_k] \tag{8}$$

$$\text{s.t. } x_{k+1} = f(x_k, u_k), \; k = 0, 1, ..., N-1 \tag{9}$$

$$x(0) = x_0 \tag{10}$$

Where $Q, P, R \succeq 0$. Solving this problem will give us optimal path and input sequences that will take us towards our goal! It's important to note that we *won't* execute all N inputs at once! In model predictive control, we execute only the first input in the sequence, move to the next state, and then re-solve the model predictive control problem to "close the loop."

Problem 1: What are some positive and negative effects of increasing N? If we had a system with lots of disturbances, why would we not want to execute the entire sequence of optimal inputs?

Solution: By increasing N, we can plan further into the future, but we increase the computation time to solve the optimization problem. Each sequence of inputs is an open loop sequence that doesn't take into account any current state information past the first state. Thus, with disturbances, we won't be able to track the trajectory.

Problem 2: We can dramatically speed up our MPC solution time by "warm-starting" the optimization with an initial guess. One example of an initial guess for x^* is a straight line that goes from x_0 at $k = 0$ to x_d at $k = N$. Find an expression for a warm start guess $x^*(k)$ that achieves this interpolation.

Solution: We can choose:

$$x^*(k) = x_0 + \frac{k}{N}(x_d - x_0) \tag{11}$$

When $k = N$, this will give us the desired state, and when $k = 0$, it will give us the initial state. Otherwise, it will interpolate between the two.

4 Constrained Model Predictive Control

A major advantage of MPC is that since it optimizes over both x and u, we can impose constraints directly on the states we plan over. This means that we can easily encode information about obstacle avoidance by imposing constraints on x.

$$x^*, u^* = \arg\min_{x,u} (x_N - x_d)^T P (x_N - x_d) + \sum_{k=0}^{N-1} [(x_k - x_d)^T Q (x_k - x_d) + u_k^T R u_k] \tag{12}$$

$$\text{s.t. } x_{k+1} = f(x_k, u_k),\ k = 0, 1, ..., N-1 \tag{13}$$

$$x(0) = x_0 \tag{14}$$

$$\text{Position constraints, input constraints, etc.} \tag{15}$$

Problem 3: Suppose that there are p circular obstacles between the current position of our turtlebot, (x, y), and the desired position of our turtlebot, (x_d, y_d). Each obstacle with center position (x_i, y_i) and radius r_i. Write an expression for a constraint on (x, y) that ensures the turtlebot will not collide with the obstacles.

Solution: We can use the constraint:

$$(x - x_i)^2 + (y - y_i)^2 > r_i^2 \tag{16}$$

If we wish to use an inequality constraint with $\geq$ instead of $>$, we could add a small buffer onto r_i.