2/24 Recitation

- MPC
- HW 2
- quad rotor MPC

Trajectory optimization

$$\min_{x_{1:N},\, u_{1:N-1}} \quad J(x,u)$$

$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k)$$

$$g(x,u) \le 0$$

This class has been all about setting up and solving control problems as optimization problems, like this canonical trajectory optimization problem. In general, these range from trivial to solve, to impossible.

Riccati via DP

Cost-to-go at last time step $V_N(x) = x^T \overbrace{Q_f}^{P_N} x$

$$Q_{k-1}(x,u) = \ell(x,u) + V_k(Ax+Bu)$$

$$Q_{k-1}(x,u) = x^TQx + u^TRu + (Ax+Bu)^T P_k (Ax+Bu)$$

$Q_{N-1}(x,u)$

① $V_{N-1} = \min_u \ell(x_{N-1}, u) + V_N(Ax_{N-1} + Bu)$

$$\nabla_u Q = 2(Ru + B^TPAx + B^TPBu) = 0$$

$$u = -\underbrace{(R+B^TPB)^{-1}B^TPA}_{K}x$$

plug $u = -Kx$ into ① to get P_{N-1}

FHLQR

$$\min_{x_{1:N},\, u_{1:N-1}} \quad \frac{1}{2}\sum_{i=1}^{N-1}\left(x_i^TQx_i + u_i^TRu_i\right) + \frac{1}{2}x_N^TQ_fx_N$$

$$x_1 = x_{IC}$$

$$x_{k+1} = Ax_k + Bu_k$$

① Solve w/ cvx for x's, u's

② Riccati recursion for $u_i = -k_i x_i$

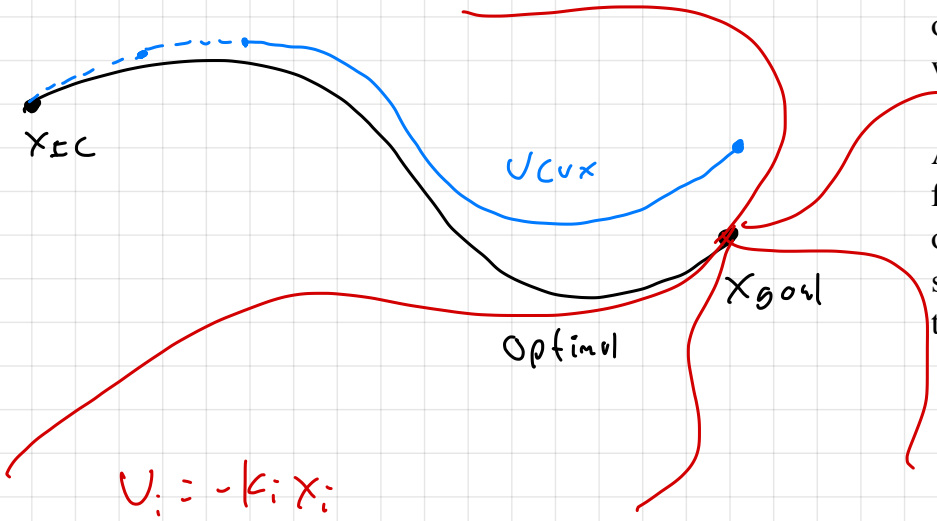
The finite horizon LQR problem is an example of an easy problem to solve. If we have a problem like this, we can solve it with convex optimization (an equality-constrained QP that takes 1 linear solve) to get X and U, or we can use the Ricatti recursion to give us a feedback policy u_i = -K_i x_i. The X and U from cvx is specific to our initial condition, and is not an optimal policy if we start from anywhere else. The LQR feedback policy is globally optimal, no matter where we start in the state space.

Sim to real

- Unmodeled dynamics
- noisy actuators
- state estimation

We have a sim to real gap that causes our modeled dynamics to differ from the real world dynamics. To add to this, we also have some uncertainty in our state estimation. This means that if we try and execute our Ucvx from the convex optimization solution (shown in blue), we will start to differ from the Xcvx. This means we have to resolve the convex optimization problem every time we differ from planned trajectory.

Alternatively, our FHLQR feedback policy is globally optimal, so it doesn't matter if we stray from our planned trajectory, the feedback policy is still optimal.



- if we execute U_{cvx}, we will screw up a little and drift
- if we use $u_i = -K_i x_i$, always optimal (LQR)

FHLQR is great, but there are many (convex) trajectory optimization problems that don't fall into this category. Here is an example of a convex trajectory optimization problem that has non-quadratic costs (it has an L2 and L1 norm), and constraints on the states and controls. This problem is still convex, so it can be solved (quickly) for the optimal solution, but we can no longer just rely on gains K computed offline.

This leads us to MPC, where we are going to setup and solve convex trajectory optimization problems at every time step (to deal with our imperfect model/ disturbances).

$$\min_{x_{1:N},\, u_{1:N-1}} \quad \sum_{i=1}^{N-1} \left(\|x_i - x_g\|_2 + \|u_i\|_1 \right) + x_N^T Q_N x_N$$

$$\text{s.t.} \quad x_1 = x_{IC} \qquad \text{MPC Input}$$

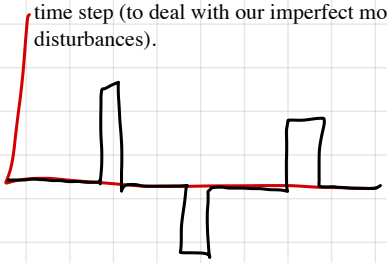
$$x_{k+1} = A x_k + B u_k$$

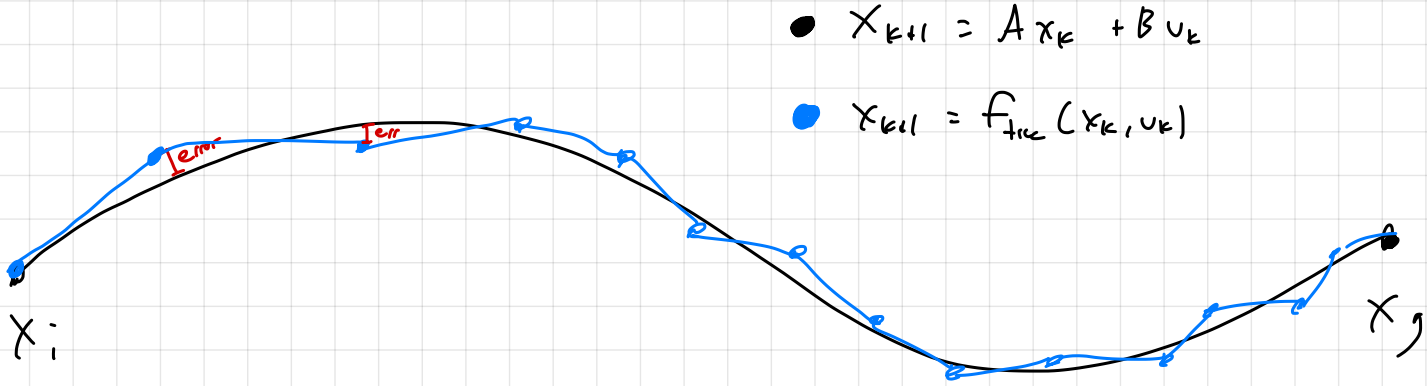
$$\|u_i\|_2 \le u_{max}$$

$$\underline{x} \le x_i \le \bar{x}$$

$$x_N = x_{goal}$$

u^*

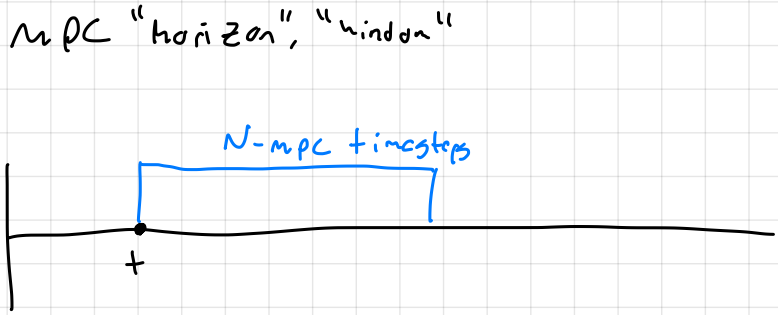




At the start of our trajectory, we solved for an optimal trajectory (shown in black). As we go to execute this, we see that even after just the first control input, we strayed a little bit from our anticipated trajectory. This means we have to resolve our problem with a new initial condition, and then execute the first control.

MPC works by setting up and solving a convex trajectory optimization problem at every time step that starts at the current initial condition, and then from the solution, you just execute the first control that you solved for. You will do this at every time step.

In general, we like having a constant MPC "horizon" of maybe 10-50 time steps, where we only consider that far into the future in our planning. You will see this called the MPC "horizon" or "window".



In black is our optimal trajectory from the convex trajectory optimization problem. After executing the first command (shown in blue) we have deviated from this planned (black) trajectory. This is because the black trajectory is using the linear dynamics model (no modeled disturbances or anything), and the real (blue) trajectory, is the real life dynamics.