MECH 6v29.002 – Model Predictive Control

L5 – LQ MPC

Outline

- Linear Quadratic MPC
- Three potential approaches
  - Formulations and comparison
- Infinite-horizon case
- HW #1

Linear Quadratic MPC

- Linear refers to linear model

$$x_{k+1} = Ax_k + Bu_k \qquad x_0 = x(0) \qquad x \in \mathbb{R}^n,\ u \in \mathbb{R}^m$$

- Quadratic refers to quadratic cost function

$$J_0(x_0, U_0) = \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k + x_N^T P x_N \qquad U_0 = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} \in \mathbb{R}^{m \cdot N}$$

  - Assume $Q = Q^T \geq 0 \quad P = P^T \geq 0 \quad R = R^T > 0$

Keep an eye out for when this is needed

- Finite-time optimal control problem

$$J_0^*(x_0) = \min_{U_0} \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k + x_N^T P x_N$$

$$s.t.$$

$$x_{k+1} = Ax_k + Bu_k,\ k \in \{0, 1, \ldots, N-1\}$$

$$x_0 = x(0)$$

Linear Quadratic MPC Solutions

- Three potential solution approaches
  - Batch approach
    - Based on the lifted form we derived last lecture
    - Produce an optimal input trajectory based on initial condition
  - Recursive approach (dynamic programming)
    - Produce an optimal feedback control policy
  - Online optimization
    - Not necessary in this most basic formulation
    - But the approach would not change when you add additional elements to the control formulation (e.g. constraints)

Batch Approach

- Last lecture we used $x_{k+1} = Ax_k + Bu_k$ to derive

$$X = S_x x_0 + S_u U_0$$

Previous notation
$X = Px_0 + HU$

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \underbrace{\begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}}_{S_x} x_0 + \underbrace{\begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix}}_{S_u} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

- Using this notation, we can rewrite the cost function

$$J_0(x_0, U_0) = \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k + x_N^T P x_N \quad = X^T \bar{Q} X + U_0^T \bar{R} U_0$$

$$\bar{Q} = blkdiag\{Q, \ldots, Q, P\} \qquad \bar{R} = blkdiag\{\underbrace{R, \ldots, R}_{N \text{ times}}\}$$

$$\bar{Q} \geq 0 \qquad \bar{R} > 0$$

Batch Approach (cont.)

- Substitute lifted state trajectory into cost function

$$J_0(x_0,U_0) = X^T\bar{Q}X + U_0^T\bar{R}U_0$$

$$X = S_x x_0 + S_u U_0$$

$$J_0(x_0,U_0) = (S_x x_0 + S_u U_0)^T\bar{Q}(S_x x_0 + S_u U_0) + U_0^T\bar{R}U_0$$

$\downarrow$ transpose

$$= \left(x_0^T S_x^T + U_0^T S_u^T\right)\bar{Q}(S_x x_0 + S_u U_0) + U_0^T\bar{R}U_0$$

$\downarrow$ multiply and rearrange

$$= U_0^T\left(\underbrace{S_u^T\bar{Q}S_u + \bar{R}}_{H}\right)U_0 + 2x_0^T\left(\underbrace{S_x^T\bar{Q}S_u}_{F}\right)U_0 + x_0^T\left(\underbrace{S_x^T\bar{Q}S_x}_{Y}\right)x_0$$

$$= U_0^T H U_0 + 2x_0^T F U_0 + x_0^T Y x_0$$

- What can we say about H?

$\bar{Q} \geq 0, \quad \bar{R} > 0 \Rightarrow H > 0 \quad \Rightarrow J_0(x_0,U_0)$ is pos. def. function of U_0

Batch Approach (cont.)

$$\Rightarrow J_0\left(x_0, U_0\right) \text{ is pos. def. function of } U_0$$

- This means we can find the minimum of the cost function with respect to the input trajectory by taking the derivative and setting it equal to zero

$$J_0\left(x_0, U_0\right) = U_0^T H U_0 + 2x_0^T F U_0 + x_0^T Y x_0$$

$$\frac{dJ_0}{dU_0} = 2HU_0 + 2F^T x_0 \quad = 0$$

- Optimal input: $U_0^* = -H^{-1} F^T x_0$

$$= -\left(S_u^T \bar{Q} S_u + \bar{R}\right)^{-1} \left(S_x^T \bar{Q} S_u\right)^T x_0$$

$$= -\left(S_u^T \bar{Q} S_u + \bar{R}\right)^{-1} S_u^T \bar{Q} S_x x_0$$

- Optimal cost: $J_0^*\left(x_0\right) = x_0^T \left(Y - F H^{-1} F^T\right) x_0$

Batch Approach (cont.)

- Batch approach produces an optimal input trajectory

$$U_0^* = -H^{-1}F^T x_0$$
$$= -\left(S_u^T \bar{Q} S_u + \bar{R}\right)^{-1} S_u^T \bar{Q} S_x x_0$$

$$U_0^* = \begin{bmatrix} u_0^* \\ u_1^* \\ \vdots \\ u_{N-1}^* \end{bmatrix} \in \mathbb{R}^{m \cdot N}$$

- If we want just the first input, we can always multiply by a matrix

$$u_0^* = \begin{bmatrix} I & 0 & \cdots & 0 \end{bmatrix} U_0^*$$
$$= -\begin{bmatrix} I & 0 & \cdots & 0 \end{bmatrix} H^{-1} F^T x_0$$

Recursive Approach

- Dynamic programming
  - In general, take a complicated problem and break it into simpler problems to be solved in a recursive manner

$$J_0\left(x_0,U_0\right)=\sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k + x_N^T P x_N$$

$$J_0\left(x_0,U_0\right)=x_0^T Q x_0 + u_0^T R u_0 + J_1^*\left(x_1\right)$$

Subject to model constraints

$$J_1^*\left(x_1\right)=\min_{U_1} \sum_{k=1}^{N-1} x_k^T Q x_k + u_k^T R u_k + x_N^T P x_N$$

- Key idea: work your way backwards
  - Find the optimal solution at the last time step
  - Then find the optimal solution at the second-to-last time step based on the solution at the last time step
  - Repeat recursively until you reach time 0

Recursive Approach (cont.)

- Start at the very last time step

$$P_N = P$$

$$J^*_{N-1}\left(x_{N-1}\right) = \min_{u_{N-1}} \quad x^T_{N-1} Q x_{N-1} + u^T_{N-1} R u_{N-1} + x^T_N P_N x_N$$

$$s.t.$$

$$x_N = A x_{N-1} + B u_{N-1}$$

- Plug in state constraint

$$J^*_{N-1}\left(x_{N-1}\right) = \min_{u_{N-1}} \quad x^T_{N-1} Q x_{N-1} + u^T_{N-1} R u_{N-1} + \left(A x_{N-1} + B u_{N-1}\right)^T P_N \left(A x_{N-1} + B u_{N-1}\right)$$

$$= \min_{u_{N-1}} \quad x^T_{N-1}\left(A^T P_N A + Q\right) x_{N-1} + 2 x^T_{N-1}\left(A^T P_N B\right) u_{N-1} + u^T_{N-1}\left(B^T P_N B + R\right) u_{N-1}$$

- Quadratic function of u_{N-1}
- Find optimal input $u^*_{N-1} = \underbrace{-\left(B^T P_N B + R\right)^{-1} B^T P_N A}_{F_{N-1}} x_{N-1}$

$$u^*_{N-1} = F_{N-1} x_{N-1}$$

Recursive Approach (cont.)

- Optimal one-step cost function

$$J_{N-1}^{*}\left(x_{N-1}\right)=x_{N-1}^{T}Qx_{N-1}+u_{N-1}^{*T}Ru_{N-1}^{*}+x_{N}^{T}P_{N}x_{N}$$

$$x_{N}=Ax_{N-1}+Bu_{N-1}^{*} \qquad u_{N-1}^{*}=F_{N-1}x_{N-1}$$

$$\Downarrow$$

$$J_{N-1}^{*}\left(x_{N-1}\right)=x_{N-1}^{T}P_{N-1}x_{N-1}$$

$$P_{N-1}=A^{T}P_{N}A+Q-A^{T}P_{N}B\left(B^{T}P_{N}B+R\right)^{-1}B^{T}P_{N}A$$

Recursive Approach (cont.)

- Now we can do this for any time step k

$$u_{N-1}^* = -\left(B^T P_N B + R\right)^{-1} B^T P_N A x_{N-1} \quad \Rightarrow \quad \begin{aligned} u_k^* &= -\left(B^T P_{k+1} B + R\right)^{-1} B^T P_{k+1} A x_k \\ &= F_k x_k, \quad k \in \{0, \dots, N-1\} \end{aligned}$$

$$J_k^*(x_k) = x_k^T P_k x_k$$

$$P_k = A^T P_{k+1} A + Q - A^T P_{k+1} B \left(B^T P_{k+1} B + R\right)^{-1} B^T P_{k+1} A$$

- This is the Discrete-time Riccati Equation
  - Initialized at $P_N = P$
  - Solved backwards $P_{k+1} \rightarrow P_k$

- Optimal feedback control policy $u_k^* = F_k x_k$

Online optimization

- Or, if you don't care about the specific relationships between the current state and the optimal inputs, then you can just formulate and solve the optimization problem

$$J_0^*(x_0) = \min_{U_0} \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k + x_N^T P x_N$$
$$\begin{aligned} &s.t. \\ &x_{k+1} = Ax_k + Bu_k,\ k \in \{0,1,\dots,N-1\} \\ &x_0 = x(0) \end{aligned}$$

- This is a quadratic program – relatively easy to solve
- Main advantage:
  - Approach is unchanged if now we have

$$J_0^*(x_0) = \min_{U_0} \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k + x_N^T P x_N$$
$$\begin{aligned} &s.t. \\ &x_{k+1} = Ax_k + Bu_k,\ k \in \{0,1,\dots,N-1\} \\ &x_{k+1} \in \mathcal{X},\ u_k \in \mathcal{U},\ k \in \{0,1,\dots,N-1\} \\ &x_0 = x(0) \end{aligned}$$

  - Other approaches become significantly more difficult

Comparison of Approaches

- Remember that we are currently solving a finite-time optimization problem (not thinking about receding horizon MPC yet)

$$
\begin{aligned}
J_0^*(x_0) = \min_{U_0} \; & \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k + x_N^T P x_N \\
s.t. \; & \\
& x_{k+1} = A x_k + B u_k, \; k \in \{0, 1, \ldots, N-1\} \\
& x_0 = x(0)
\end{aligned}
$$

- If model is perfect (no model error, no disturbances), then the three approaches will result in the same state and input trajectories
- But the batch approach is an open-loop optimization, while the recursive, dynamic programming approach is an optimal control policy based on the current state
- If there is model uncertainty (or disturbances), we would expect the recursive approach to be more robust

Infinite Horizon Problem

- What if our system operates forever? $N \to \infty$

$$J_0^*(x_0) = \min_{U_0} \sum_{k=0}^{\infty} x_k^T Q x_k + u_k^T R u_k$$

$$\begin{aligned} &s.t. \\ &x_{k+1} = Ax_k + Bu_k,\ k \in \{0,1,\ldots,\infty\} \\ &x_0 = x(0) \end{aligned}$$

No more terminal cost

- Batch approach is no longer applicable
- Recursive approach can still work
- Using the Discrete-time Riccati Equation initialized as $P_0 = Q$

$$P_k = A^T P_{k+1} A + Q - A^T P_{k+1} B \left(B^T P_{k+1} B + R \right)^{-1} B^T P_{k+1} A \qquad k \to -\infty$$

- If we assume that these iterations converge then $P_k \to P_\infty$

$$P_\infty = A^T P_\infty A + Q - A^T P_\infty B \left(B^T P_\infty B + R \right)^{-1} B^T P_\infty A$$

- Discrete-time Algebraic Riccati Equation (DARE)

Infinite Horizon Problem (cont.)

$$P_\infty = A^T P_\infty A + Q - A^T P_\infty B\left(B^T P_\infty B + R\right)^{-1} B^T P_\infty A$$

- Optimal control input $u_k^* = -\left(B^T P_\infty B + R\right)^{-1} BP_\infty A x_k$
  $= F_\infty x_k, \quad k \in \{0, \ldots, \infty\}$
- Optimal infinite horizon cost $J_0^*(x_0) = x_0^T P_\infty x_0$

- This solution corresponds exactly to the discrete-time, infinite-horizon, LQR solution
  - In Matlab: [K,S,e] = dlqr(A,B,Q,R,N)
  - https://www.mathworks.com/help/control/ref/dlqr.html

$$u[n] = -Kx[n]$$

minimizes the quadratic cost function

$$J(u) = \sum_{n=1}^{\infty} \left(x[n]^T Q x[n] + u[n]^T R u[n] + 2x[n]^T N u[n]\right)$$

for the discrete-time state-space mode

$$x[n+1] = Ax[n] + Bu[n]$$

Infinite Horizon Problem (cont.)

- We assumed that the discrete-time Riccati equation integrations converged
- But, we can prove this
- **Theorem**: If (A, B) is a stabilizable pair and $\left(Q^{1/2}, A\right)$ is an observable pair, then the discrete-time Riccati equation with $P_0 \geq 0$ converges to the unique pos. def. solution P_∞ of the DARE and all the eigenvalues of $(A + BF_\infty)$ lie strictly inside the unit circle.

  - Stabilizable
    - So that cost is finite – system can be driven to the origin
  - Observable
    - Think about rewriting the cost function as $x_k^T Q x_k = \left(x_k^T Q^{1/2}\right)\left(Q^{1/2} x_k\right)$
    - Now the “output” is penalized $y_k = Q^{1/2} x_k$
    - Driving outputs to zero means that states converge to zero
  - Closed-loop system $x_{k+1} = Ax_k + Bu_k \qquad u_k = F_\infty x_k$

$$x_{k+1} = (A + BF_\infty) x_k$$

One Potential Approach

- What if we want to be able to prove that MPC is closed-loop stable?
  - In a previous lecture, we saw that adding the constraint $x_N = 0$ can help (and we can prove this)
  - But we can also add a terminal costs based on the DARE (discrete-time, infinite-horizon, LQR solution)

$$
\begin{aligned}
J_0^*(x_0) = \min_{U_0} \ & \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k + x_N^T P_\infty x_N \\
s.t.\ & \\
& x_{k+1} = A x_k + B u_k,\ k \in \{0,1,\ldots,N-1\} \\
& x_0 = x(0)
\end{aligned}
$$

  - We will be able to use this to prove stability as well and avoids some of the issues associated with the terminal constraint $x_N = 0$

Homework #1 (Due: Sept. 15)

- Two problems
- Problem 1:
  - Implement and compare the batch and recursive approaches in Matlab.
  - Determine the stabilizing effects of a long prediction horizon.
- Problem 2:
  - Implement the online optimization approach in Matlab using the YALMIP toolbox
  - Become familiar with YALMIP example before attempting (let me know if you have any questions)

- In a single PDF, type your responses to the various questions, provide well formatted Matlab plots, and your Matlab code
  - All of this helps me provide you with more feedback