MPC for Robot Arm Trajectory Control

Lab Session 3

Yunqi Huang

The Control Structure for this Lab Session (1)

- In this lab session we want to use a linear MPC for regulation to control the robot

- As we already know the robot dynamic model is highly non linear!

$$\boxed{M(q)\ddot{q} + \underbrace{n(q,\dot{q})}_{c(q,\dot{q}) + g(q) + \text{friction model}} = u}$$

- How can we solve this problem?

The Control Structure for this Lab Session (2)

- How can we solve this problem?

- We can cancel out the robot dynamics!

The Control Structure for this Lab Session (3)

Given the dynamic equation:

$$M(q)\ddot{q} + n(q, \dot{q}) = u \tag{2}$$

and given the control equation;

$$u = M(q)a + n(q, \dot{q}) \tag{3}$$

We want to solve for $\ddot{q}$, the acceleration. Start by isolating $\ddot{q}$ on one side of the equation:

$$M(q)\ddot{q} = u - n(q, \dot{q}) = M(q)a + \cancel{n(q, \dot{q})} - \cancel{n(q, \dot{q})} \tag{4}$$

Assuming $M(q)$ is invertible, multiply both sides by $M(q)^{-1}$, the inverse of the mass matrix:

$$\ddot{q} = \cancel{M(q)^{-1}}\,\cancel{M(q)}u \tag{5}$$

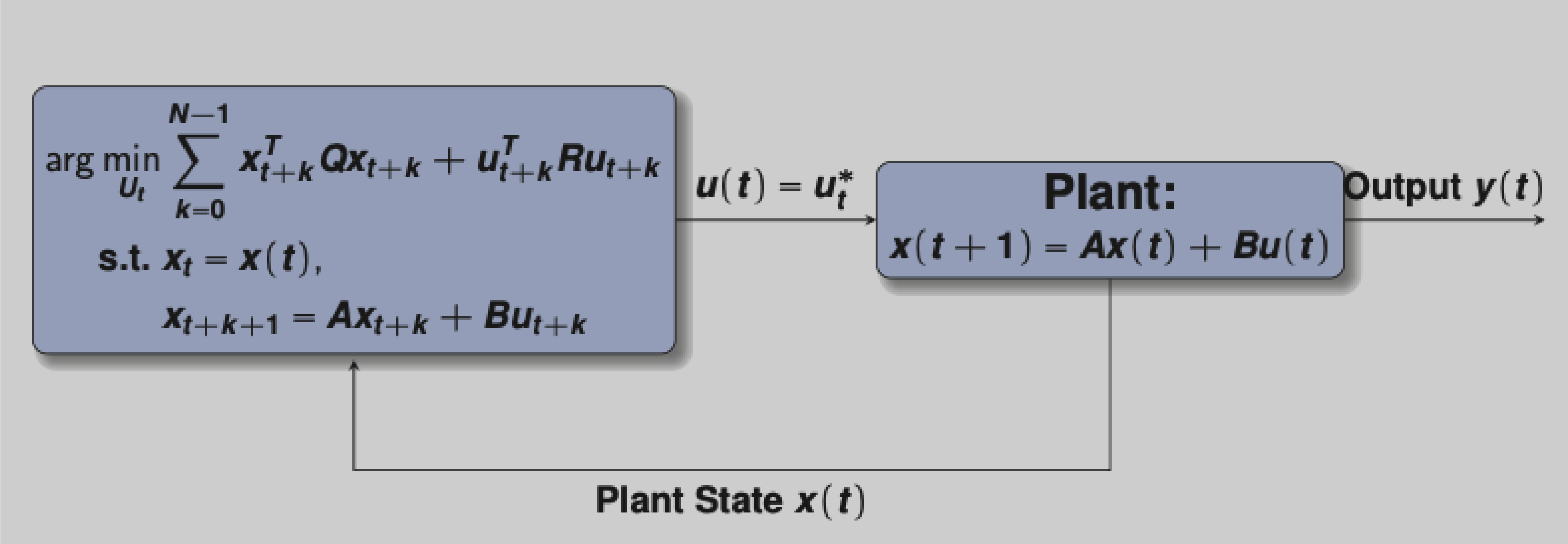
$$\boxed{\ddot{q} = u} \tag{6}$$

The Control Structure for this Lab Session (4)

- In the code the function that perform the system linearization is

```
cmd.tau_cmd = dyn_cancel(dyn_model, q_mes, qd_mes, u_mpc)
```

Simplified MPC



Goal: Find these matrices, A,B,Q,R, in the robot arm control

Step 1: Find State and Control Input

- The state vector contains joint positions in the first half and joint velocities in the second half.

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} q \\ \dot{q} \end{pmatrix}$$

- The control input vector is the acceleration of each joints.

$$\boldsymbol{u} = \ddot{\boldsymbol{q}}$$

Step 2: Prediction Model

- The A matrix describes how the system state evolves from one time step to the next, assuming no control input.

$$q_{k+1} = q_k + \dot{q}_k \cdot \Delta t$$

- The Control input vector is the accelerations of each joints.

$$\dot{q}_{k+1} = \dot{q}_k + \ddot{q}_k \cdot \Delta t$$

Step 3: Cost matrices

- The Q matrix is the state cost matrix. Its dimension should be aligned with the state vector

Q = 1000000 * np.eye(num_states)
Q[num_joints:, num_joints:] = 0.0

Solve the OCP

- Assume that $\Phi^T \bar{Q} \Phi + \bar{R}$ is positive definite. It can be true if $Q > 0$ and $R > 0$.
- Take the first derivative of J_t[1]

$$\frac{\partial J_t}{\partial U_t} = 2(\Phi^T \bar{Q} \Phi + \bar{R}) U_t + 2\Phi^T \bar{Q} F x(t) \tag{8}$$

- The necessary condition of the minimum J_t is obtained as

$$\begin{aligned} &\frac{\partial J_t}{\partial U_t} = 0 \\ \Longleftrightarrow \quad & U_t^* = -(\Phi^T \bar{Q} \Phi + \bar{R})^{-1} \Phi^T \bar{Q} F x(t) \end{aligned} \tag{9}$$

- The R matrix is the control input cost

R = 0.1 * np.eye(num_controls)

- The calculation of the matrices is given in ‘regulator_model.py’

Step 3: Cost matrices (Cont)

The calculation of the matrices is given in ‘regulator_model.py’, which follows the notation in

https://cse.lab.imtlucca.it/~bemporad/teaching/mpc/imt/1-linear_mpc.pdf

Page (39 - 41)

$$J(z, x_0) = x'_N P x_N + \sum_{k=0}^{N-1} x'_k Q x_k + u'_k R u_k$$

The optimum is obtained by zeroing the gradient

$$\nabla_z J(z, x_0) = Hz + Fx_0 = 0$$

and hence $z^* = \begin{bmatrix} u_0^* \\ u_1^* \\ \vdots \\ u_{N-1}^* \end{bmatrix} = -H^{-1} F x_0$ (“batch” solution)

Tasks (damping flag to 0)

1. Finish the code of ‘getSystemMatrices()’, and get the prediction model matrices A and B (without damping).

2. Play with the parameters in cost matrices ‘getCostMatrices()’

2.1. Change the parameter of Q to ‘1000’, ‘10000’, ‘100000’… and compare the results

2.2. What is happening if comment the line ‘Q[num_joints:, num_joints:] = 0.0’ and what’s the reason

2.3. Design your cost matrices and analyze how different parameters influence the result (optional)

3. Is it possible to consider the damping of each joint into the system matrices and what is the result