MPC for Robot Arm Trajectory Control

Lab Session 4

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!

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

- 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:

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

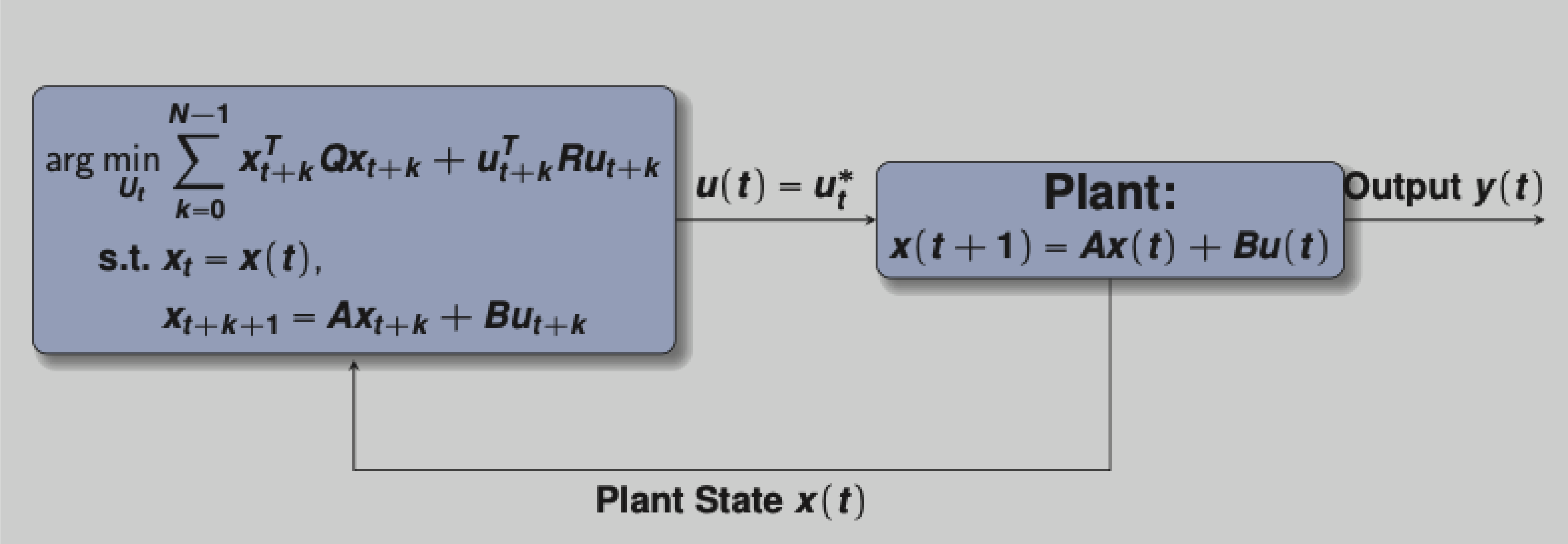
$$\boxed{\ddot{q} = a} \tag{6}$$

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 3: Cost matrices

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

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

- The R matrix is the control input cost

```
R = 0.1 * np.eye(num_controls)
```

- The calculation of the matrices is given in ‘tracker_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 (52)

$$\min_{z} \sum_{k=0}^{N-1} \|W^y(y_{k+1} - r(t))\|_2^2 + \|W^{\Delta u}\Delta u_k\|_2^2$$
$$[\Delta u_k \triangleq u_k - u_{k-1}],\ u_{-1} = u(t-1)$$

$$z = \begin{bmatrix} \Delta u_0 \\ \Delta u_1 \\ \vdots \\ \Delta u_{N-1} \end{bmatrix} \text{ or } z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$\min_{z} \quad J(z, x(t)) = \tfrac{1}{2} z' H z + [x'(t)\, r'(t)\, u'(t-1)] F' z$$

Tasks (damping flag to 0, spring flag 0, no noise, no delay)

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

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

3 change the ref variable to control only the position

4 change the ref variable to track position and velocity

5 (OPTIONAL) design a new reference like linear or polynomial and try to track them

6 (OPTIONAL) try to add constraints on state and action for the MPC tracker (I'll give you the constraints structure)

Code

- The code is available here:

https://github.com/VModugno/lab_sessions_COMP0211_PUBLIC/tree/main/week_5