ROB 498/599 Fall 2024: Assignment #3

Due on Oct. 18th, at 11:59 pm, on Gradescope.

Submission Instructions: Submit a PDF file with written answers to all questions, images of plots (included on the PDF). Paste your code to the end of the document. Label the document **[uniquename]_ROB498_Assignment3.pdf** and upload to Gradescope.

In this assignment, consider the controller design of a 2-link robot manipulator as shown in Fig. 1. m_1, m_2 are the mass of each link, l_1, l_2 are the length of first and second link, and l_{c1}, l_{c2} are the distances between the center of mass of each link and the center of previous joint. q_1, q_2 are the joint angle, with positive direction being counter-clockwise. g is the acceleration of gravity.

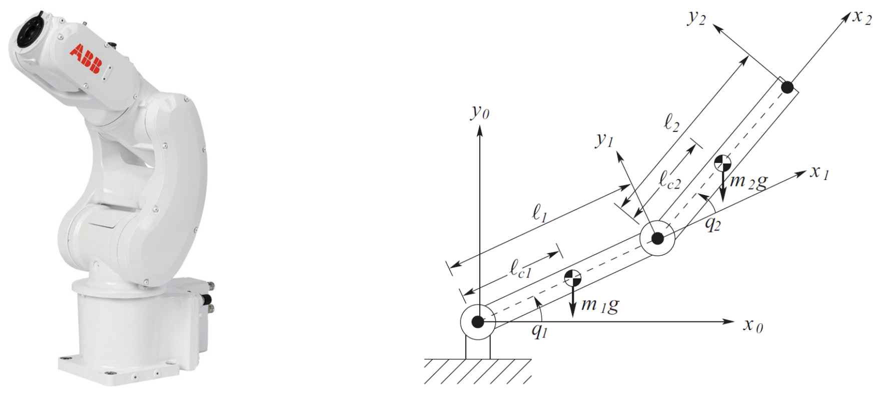

Figure 1: Left: A 2-link RR robot manipulator. Right: Schematic of a 2-link RR robot.

The equation of motion of this robot is

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{N}(\mathbf{q}) = \tau \ ,$$

where

- $\mathbf{q}$ is the generalized coordinate vector. In this case, $\mathbf{q} = \begin{bmatrix} q_1 & q_2 \end{bmatrix}^T$
- $\mathbf{M}(\mathbf{q})$ is the mass/inertia matrix of the robot, defined as

$$\begin{aligned}
\mathbf{M}(1,1) &= m_1 l_{c1}^2 + m_2 \left[l_1^2 + l_{c2}^2 + 2 l_1 l_{c2} \cos(q_2) \right] + I_1 + I_2 \\
\mathbf{M}(1,2) &= m_2 \left[l_{c2}^2 + l_1 l_{c2} \cos(q_2) \right] + I_2 \\
\mathbf{M}(2,1) &= \mathbf{M}(1,2) = m_2 \left[l_{c2}^2 + l_1 l_{c2} \cos(q_2) \right] + I_2 \\
\mathbf{M}(2,2) &= m_2 l_{c2}^2 + I_2,
\end{aligned}$$

  where I_1, I_2 are the moment of inertia of both links.

- $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the Coriolis matrix of the robot, defined as

$$\begin{aligned}
\mathbf{C}(1,1) &= -m_2 l_1 l_{c2} \sin(q_2)\dot{q}_2 \\
\mathbf{C}(1,2) &= -m_2 l_1 l_{c2} \sin(q_2)\left(\dot{q}_1 + \dot{q}_2\right) \\
\mathbf{C}(2,1) &= m_2 l_1 l_{c2} \sin(q_2)\dot{q}_1 \\
\mathbf{C}(2,2) &= 0
\end{aligned}$$

- $\mathbf{N}(\mathbf{q})$ contains the gravity terms and other conservative forces applied to the joints. In this assignment, we only consider gravity, which gives

$$\mathbf{N}(\mathbf{q}) = \begin{bmatrix} m_1 g l_{c1} \cos(\mathbf{q}_1) + m_2 g \left[l_1 \cos(\mathbf{q}_1) + l_{c2} \cos(\mathbf{q}_1 + \mathbf{q}_2)\right] \\ m_2 g l_{c2} \cos(\mathbf{q}_1 + \mathbf{q}_2) \end{bmatrix} \tag{1}$$

Problem 1 [5 pts] Represent the angular acceleration vector $\ddot{\mathbf{q}}$ in terms of

$$\begin{aligned}
\ddot{q}_1 &= f_1(\mathbf{M}, \mathbf{C}, \mathbf{N}, \tau) \\
\ddot{q}_2 &= f_2(\mathbf{M}, \mathbf{C}, \mathbf{N}, \tau)\ .
\end{aligned}$$

Hint: The mass/inertia matrix is always symmetric and invertible.

For the rest of this assignment, use the following parameters for the robot: $m_1 = 7.848, m_2 = 4.49, l_1 = 0.3, l_{c1} = 0.1554, l_{c2} = 0.0341, I_1 = 0.176, I_2 = 0.0411$. All units are in the MKS-system.

To assist you with the simulation and visualization, three MATLAB m-files: `manipulator.m`, `cubicpoly.m`, and `robotAnimation.m` have been provided.

- `manipulator.m` is the function for the manipulator's dynamics. Note that this function takes the state vector $\mathbf{x}$ as an input argument, defined as $\mathbf{x} = \begin{bmatrix} q_1 & \dot{q}_1 & q_2 & \dot{q}_2 \end{bmatrix}^T$. Please don't confuse this with $\mathbf{q}$ in the equation of motion.
- `cubicpoly.m` is the function that generates the cubic polynomial trajectory needed for Problem 4, 5, and 6.
- `robotAnimation.m` is a function that visualizes the response of the manipulator. To use this function, you'll need to install the MATLAB Robotics System Toolbox. While the visualization component is not required for this assignment, it will make this assignment more interesting. This is also a good chance to try the Robotics System Toolbox, which is useful for robotics applications.

Problem 2 [10 pts] Implement an independent joint PD-controller. The control inputs τ_1 and τ_2 are computed as

$$\begin{aligned}
\tau_1 &= k_{P1}(q_1^d - q_1) - k_{D1}\dot{q}_1 \\
\tau_2 &= k_{P2}(q_2^d - q_2) - k_{D2}\dot{q}_2
\end{aligned}$$

For this problem, use gains $k_{Pi} = 50, k_{Di} = 10, i = 1, 2$. Since all real motors have limited torque capability, the outputs τ_1 and τ_2 of the controller should be limited to the range $-50 \leq \tau_i \leq 50, i = 1, 2$. The reference input q_1^d and q_2^d should be a step from 0 to $\frac{\pi}{2}$ radians

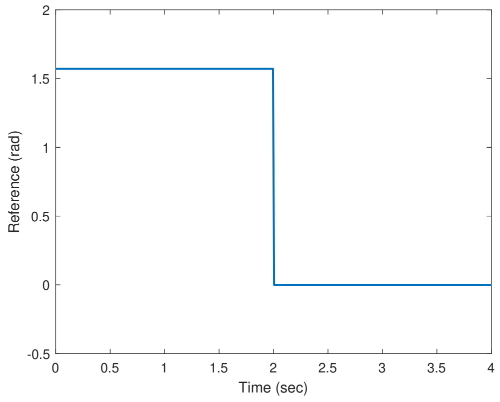

Figure 2: Reference step input for Problem 2

at $t = 0$ until $t = 2$, followed by a step back to 0 at $t = 2$ until $t = 4$ seconds, as shown in Fig. 2.

Simulate the response of the closed loop system for 4 seconds with zero initial conditions (i.e., $\mathbf{q}_0 = \dot{\mathbf{q}}_0 = \begin{bmatrix}0 & 0\end{bmatrix}^T$). Plot the joint responses, joint torques, and tracking errors for each joint. What do you observed with the error? Is it possible to eliminate the error by tuning k_P and k_D?

Problem 3 [10 pts] A technique commonly used in robot manipulator control is called gravity compensation. Combine independent joint PD-control with gravity compensation via control inputs

$$\tau_1 = k_{P1}(q_1^d - q_1) - k_{D1}\dot{q}_1 + N_1$$
$$\tau_2 = k_{P2}(q_2^d - q_2) - k_{D2}\dot{q}_2 + N_2 \ ,$$

where $N_i, i = 1, 2$ are the first and second row of $\mathbf{N}(\mathbf{q})$ in (1), corresponding to the torque produced by the force from gravity. Let $k_{Pi} = 50, k_{Di} = 10, i = 1, 2$. The torque limits and reference input are the same as those in Problem 2.

Simulate the response of the closed loop system for 4 seconds with zero initial conditions (i.e., $\mathbf{q}_0 = \dot{\mathbf{q}}_0 = \begin{bmatrix}0 & 0\end{bmatrix}^T$). Plot the joint responses, joint torques, and tracking errors for each joint. Compare with the results from Problem 2, what are the differences?

Problem 4 [10 pts] In practice, the references of a robot manipulator won't always be a set point. Cubic polynomials are a common type of trajectory. In this problem, you need to first generate a cubic polynomial trajectory from $t = 0$ to $t = 4$ seconds, which satisfies:

$$\begin{aligned} q(0) &= 0 & \dot{q}(0) &= 0 \\ q(2) &= \frac{\pi}{2} & \dot{q}(2) &= 0 \\ q(4) &= 0 & \dot{q}(4) &= 0 \end{aligned}$$

You can use `cubicpolytraj` function provided by MATLAB Robotics System Toolbox, or use the `cubicpoly` function provided on Canvas. Both functions should return the position, velocity and acceleration of the trajectory, denoted as q^d, v^d and a^d. The cubic polynomial trajectory you generated should look like the curves shown in Fig. 3

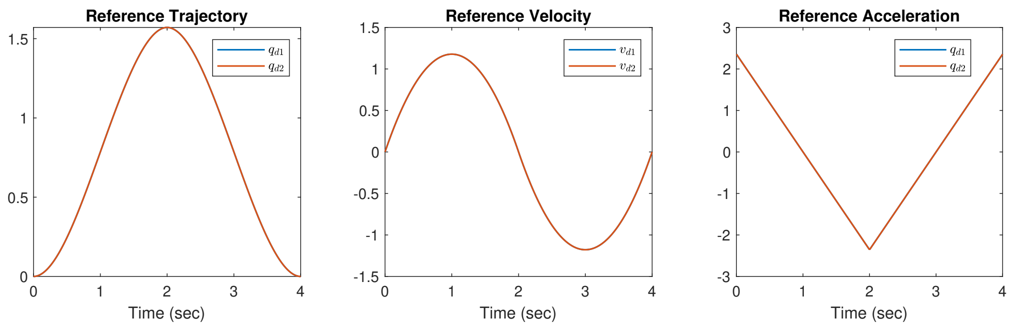

Figure 3: Reference step input for Problem 4

Using the same PD controller with the same gains and torque bounds as in Problem 2, simulate the response of the system for 4 seconds with zero initial conditions. Plot the joint responses, joint torques, and tracking errors for each joint. You will see that PD control cannot track this continuously-varying reference well. What could be the reason?

Problem 5 Now implement a PD + feed forward controller to track the same cubic polynomial trajectory in Problem 4. The feed forward torque should be

$$\tau_{FF} = \tilde{\mathbf{M}}(\mathbf{q})\mathbf{a}^d + \tilde{\mathbf{C}}(\mathbf{q}, \dot{\mathbf{q}})\mathbf{v}^d + \tilde{\mathbf{N}}(\mathbf{q})$$

where $\tilde{\mathbf{M}}(\mathbf{q}), \tilde{\mathbf{C}}(\mathbf{q}, \dot{\mathbf{q}})$, and $\tilde{\mathbf{N}}(\mathbf{q})$ are the estimated mass/inertia matrix, Coriolis matrix, and gravity terms, and $\mathbf{a}^d = [a_1^d \quad a_2^d]^T, \mathbf{v}^d = [v_1^d \quad v_2^d]^T$.

The control inputs are then given as

$$\begin{aligned}\tau_1 &= \tau_{FF}(1) + k_{P1}(q_1^d - q_1) + k_{D1}(v_1^d - \dot{q}_1)\\ \tau_2 &= \tau_{FF}(2) + k_{P2}(q_2^d - q_2) + k_{D2}(v_2^d - \dot{q}_2)\ .\end{aligned}$$

(a) [**5 pts**] Assume we have a perfect estimation of system dynamics, i.e., $\tilde{\mathbf{M}}(\mathbf{q}) = \mathbf{M}(\mathbf{q})$, $\tilde{\mathbf{C}}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$, and $\tilde{\mathbf{N}}(\mathbf{q}) = \mathbf{N}(\mathbf{q})$. Using the same gains and torque bounds as in Problem 2, simulate the closed-loop system with zero initial condition for 4 seconds. Plot the joint responses, joint torques, and tracking errors for each joint. You should only see very small tracking error caused by numerical round-off in the simulation.

(b) [**10 pts**] In practice, it is always hard, if not impossible, to obtain a perfect estimation of system dynamics. Repeat the simulation in **(a)** with $\tilde{\mathbf{M}}(\mathbf{q}) = 0.9 \cdot \mathbf{M}(\mathbf{q})$, $\tilde{\mathbf{C}}(\mathbf{q}, \dot{\mathbf{q}}) = 0.9 \cdot \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$, and $\tilde{\mathbf{N}}(\mathbf{q}) = 0.9 \cdot \mathbf{N}(\mathbf{q})$. Plot the joint responses, joint torques, and tracking errors for each joint. Compare your results with what you get in part (a) and Problem 4. What are the differences?

Problem 6 Simulate the inverse dynamics (computed torque) control

$$\tau = \tilde{\mathbf{M}}(\mathbf{q})\mathbf{a}_q + \tilde{\mathbf{C}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \tilde{\mathbf{N}}(\mathbf{q})\ ,$$

to track the cubic polynomial reference trajectory from Problem 4, where $\mathbf{a}_q = \begin{bmatrix} a_{q1} & a_{q2} \end{bmatrix}^T$ and

$$\begin{aligned}a_{q1} &= a_1^d + k_{P1}(q_1^d - q_1) + k_{D1}(v_1^d - \dot{q}_1)\\ a_{q2} &= a_2^d + k_{P2}(q_2^d - q_2) + k_{D2}(v_2^d - \dot{q}_2)\ .\end{aligned}$$

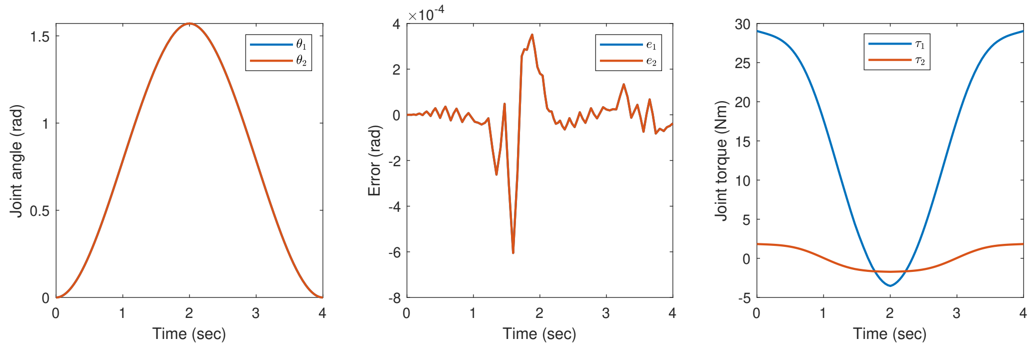

Figure 4: Inverse dynamics control system response with zero initial condition.

Using the same gains and torque bounds as in Problem 2, simulate the closed-loop system with perfect system dynamics estimation and zero initial condition for 4 seconds. Plot the joint responses, joint torques, and tracking errors for each joint. **You don't need to submit these plots**, but please make sure that they look like those in Fig. 4. Note that, with the exact inverse dynamics, there will be essentially zero tracking in the ideal case. For this reason, the tracking error shown below is mostly due to numerical round-off in the simulation. Therefore, your tracking errors may look somewhat different than the below, but your joint response and torque input curves should look like those below.

(a) **[5 pts]** Now, simulate the closed-loop system with **nonzero** initial condition $\mathbf{q}_0 = \begin{bmatrix} 0.05 & 0.05 \end{bmatrix}^T$ and $\dot{\mathbf{q}}_0 = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$ for 4 seconds. Assume your system dynamics estimation is perfect, i.e., $\tilde{\mathbf{M}}(\mathbf{q}) = \mathbf{M}(\mathbf{q})$, $\tilde{\mathbf{C}}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$, and $\tilde{\mathbf{N}}(\mathbf{q}) = \mathbf{N}(\mathbf{q})$. Plot the joint responses, joint torques, and tracking errors for each joint.

(b) **[5 pts]** Repeat the simulation in **(a)** but this time let $\tilde{\mathbf{M}}(\mathbf{q}) = 0.9 \cdot \mathbf{M}(\mathbf{q})$, $\tilde{\mathbf{C}}(\mathbf{q}, \dot{\mathbf{q}}) = 0.9 \cdot \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$, and $\tilde{\mathbf{N}}(\mathbf{q}) = 0.9 \cdot \mathbf{N}(\mathbf{q})$. Plot the joint responses, joint torques, and tracking errors for each joint. Compare your results to what you get in part (a) and Problem 5.