Course: MECH 567: Robot Kinematics and Dynamics **Assigned Date:** 04/11/2024
Instructor: Daniel Bruder, PhD **Due:** 11:59 PM on Thursday, 04/25/2024

Homework 6

Procedure

You will compare the tracking performance of PD-control, PD-control with feedforward of the reference trajectory, and inverse dynamics control for the two-link revolute joint robot.

The task is to control the robot so that each link moves from 0 to 90 degrees and back to 0 degrees smoothly, with no overshoot, in a total time of 2 seconds. The robot to be controlled is a two-link, planar, robot moving in a horizontal plane (Check problem 6 of HW4). Therefore, you first need to modify your robot model by removing the gravity terms. Do not include any friction terms in your model. Define inputs τ_1 and τ_2 as the joint torques.

Write a MATLAB program to simulate the dynamic equations of the planar elbow manipulator from the notes. You should create a file $< filename.m >$ containing the equations of motion in suitable form. See the next page for hints on how to do this. The function $< filename.m >$ should return the derivatives of the state variables, i.e.,

$$xdot = [\ q1dot;\ q1doubledot;\ q2dot;\ q2doubledot\ ]$$

The MATLAB command

$$[t, x] = ode45('< filename.m >',\ tspan,\ x0)$$

can then be invoked to simulate the system, where tspan specifies the duration of the simulation, e.g.,

$$tspan = [0 \quad 20]$$

and x0 is the vector of initial conditions.

You should specify the dynamic parameters as l1, m1, etc., so that they can be easily changed later. 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. Note that the length of link 2 does not enter into the equations of motion.

Refer to a MATLAB reference for additional details about how to simulate differential equations and plot the results.

How to Simulate a Second Order System

In order to simulate the nonlinear system

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau$$

in MATLAB, or most other packages, such as Mathematica, etc., you must first solve for the acceleration vector $\ddot{q}$ as

$$\ddot{q} = D(q)^{-1}\{\tau - C(q,\dot{q})\dot{q} - g(q)\}$$

For the 2–link planar RR–manipulator example, we have

$$\begin{aligned} d_{11}\ddot{q}_1 + d_{12}\ddot{q}_2 + c_{121}\dot{q}_1\dot{q}_2 + c_{211}\dot{q}_2\dot{q}_1 + c_{221}\dot{q}_2^2 + \phi_1 &= \tau_1 \\ d_{21}\ddot{q}_1 + d_{22}\ddot{q}_2 + c_{112}\dot{q}_1^2 + \phi_2 &= \tau_2 \end{aligned}$$

For simplicity, define

$$\begin{aligned} a_1 &= \tau_1 - c_{121}\dot{q}_1\dot{q}_2 - c_{211}\dot{q}_2\dot{q}_1 - c_{221}\dot{q}_2^2 - \phi_1 \\ a_2 &= \tau_2 - c_{112}\dot{q}_1^2 - \phi_2 \end{aligned}$$

Then the equations of motion may be succintly written as

$$\begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$

This is just a system of two equations in two unknowns. Recall that the inverse of a 2×2 matrix

$$\begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix}$$

is just

$$\frac{1}{\Delta}\begin{bmatrix} d_{22} & -d_{12} \\ -d_{21} & d_{11} \end{bmatrix}$$

where $\Delta = d_{11}d_{22} - d_{21}d_{12}$ is the determinant of the matrix. Note that Δ is always non–zero because the robot inertia matrix is always positive definite.

Therefore the acceleration vector is given by

$$\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} = \frac{1}{\Delta}\begin{bmatrix} d_{22} & -d_{12} \\ -d_{21} & d_{11} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$

Multyplying this out gives,

$$\begin{aligned} \ddot{q}_1 &= \frac{1}{\Delta}(d_{22}a_1 - d_{12}a_2) \\ \ddot{q}_2 &= \frac{1}{\Delta}(-d_{21}a_1 + d_{11}a_2) \end{aligned}$$

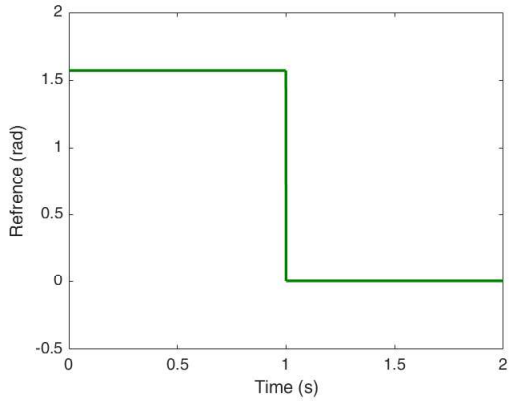
which can easily be put into MATLAB syntax.

Problem 1 - PD control

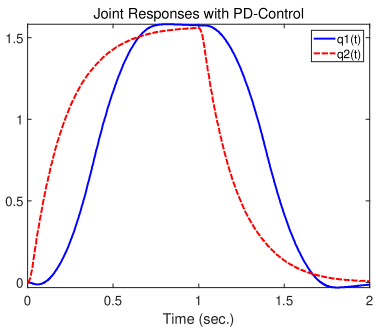
Implement an independent joint PD-controller. The control inputs τ_1 and τ_2 are computed as

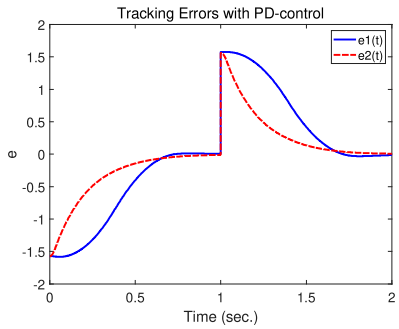
$$\tau_1 = k_{p1}\,(q_1^d - q_1) - k_{d1}\,\dot{q}_1$$
$$\tau_2 = k_{p2}\,(q_2^d - q_2) - k_{d2}\,\dot{q}_2$$

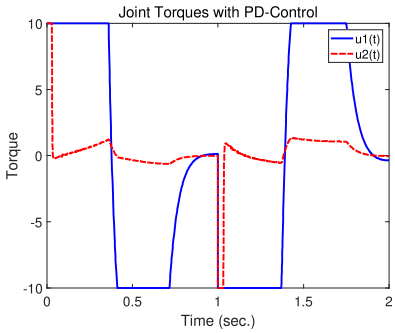
Your MATLAB file should contain the PD control law and should compute the tracking errors, $e_1 = q_1 - q_1^d$ and $e_2 = q_2 - q_2^d$, for later saving and plotting. Use gains, $k_{pi} = 100$, $k_{di} = 20$, $i = 1, 2$, which, for a linear second-order system, correspond to a damping ratio $\xi = 1$ and a natural frequency $\omega = 10$. Since all real motors have limited torque capability, the outputs τ_1 and τ_2 of the controller should be limited to the range, $-10 \leq \tau_i \leq 10$, $i = 1, 2$. The reference input q_1^d and q_2^d should be a step from 0 to $\frac{\pi}{2}$ radians from $t = 0$ to $t = 1$ second, followed by a step back to 0 from $t = 1$ to $t = 2$ seconds as shown in following figure.



Simulate the response of the closed loop system for 2 seconds with zero initial conditions. Display the joint responses, joint torques, and tracking errors for each link on the screen, but do not save or print the plots. If the responses do not look like the ones in the following figure you have an error in your simulation.







(a) Since one can never guarantee that the initial conditions will be exactly at zero, simulate the response again, this time using the initial conditions, $q_1(0) = 0.05 = q_2(0)$ which corresponds to about 3-degrees. Generate, label, and print out plots of the joint responses, input torques, and tracking errors for each link.

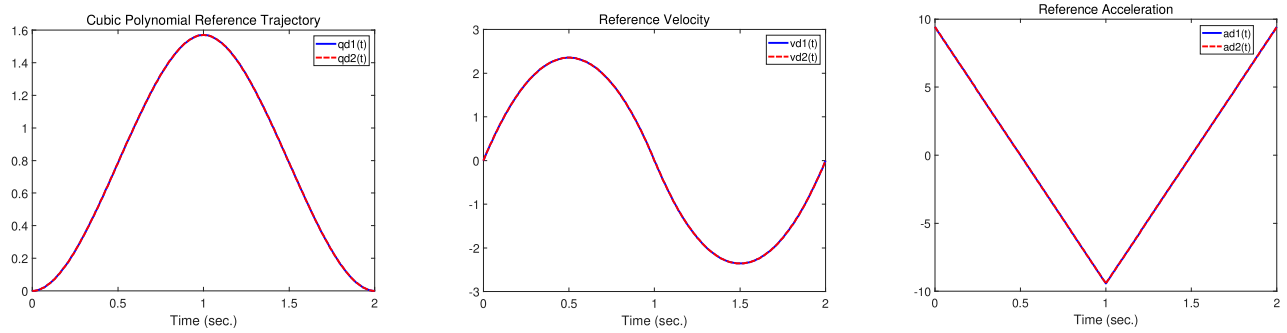
(b) What is the error in position at t = 1 second? at t = 2 seconds? Did the torque inputs saturate? for how long? Explain the large initial tracking errors and the large initial torques.

Problem 2 - PD Plus Feedforward Control

Write a MATLAB program to generate a cubic polynomial trajectory from t = 0 to t = 2 seconds. See Section 7.5 of the textbook (SHV). The trajectory should satisfy

$$q(0) = 0 \qquad \dot{q}(0) = 0$$
$$q(1) = \frac{\pi}{2} \qquad \dot{q}(1) = 0$$
$$q(2) = 0 \qquad \dot{q}(2) = 0$$

The outputs of the program should be the reference position q^d, the reference velocity v^d, and the reference acceleration a^d, and should look like the curves in the following figure.

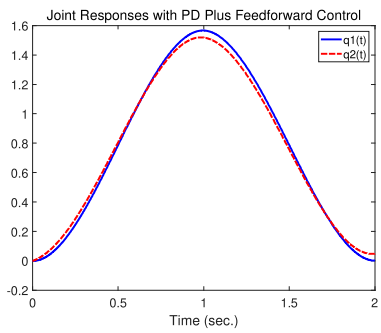


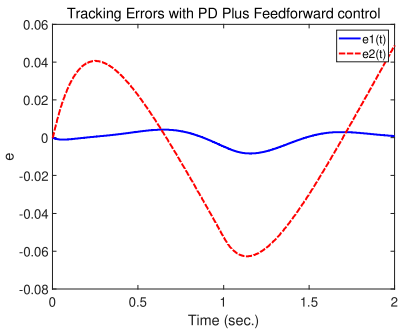
Generate plots of the reference position, velocity, and acceleration and print them out for later use in your write up. This trajectory will be used as the reference trajectory for both joints.

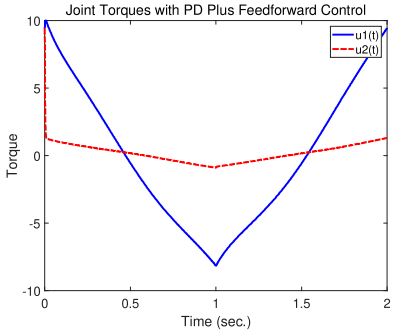
Implement a PD plus feedforward control. The joint inputs in this case are

$$\tau_1 = a_1^d + k_{p1}\left(q_1^d - q_1\right) + k_{d1}\left(v_1^d - \dot{q}_1\right)$$
$$\tau_2 = a_2^d + k_{p2}\left(q_2^d - q_2\right) + k_{d2}\left(v_2^d - \dot{q}_2\right)$$

Using the same PD gains and bounds on the maximum torque as in Problem 1, simulate the response of the system for 2 seconds with zero initial conditions. The responses should look like those in the following figure. Do not save or print out these responses. Note that the outputs of the trajectory generator are required by the controller.







(a) Now simulate the system using the same nonzero initial conditions as in Problem 1 and generate plots of the joint responses, input torques, and tracking errors, as before.

(b) What is the error in each joint angle at t = 1 second? at t = 2 second? How do the joint tracking errors and input torques compare to the pure PD control of Problem 1?

Problem 3 - Inverse Dynamics Control

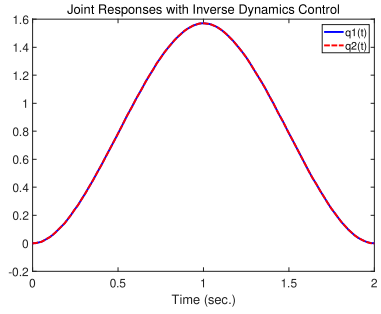
Simulate an inverse dynamics control

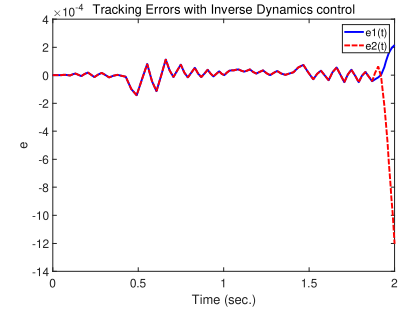
$$\tau = D(q)\, a_q + C(q, \dot{q})\, \dot{q} + g(q)$$

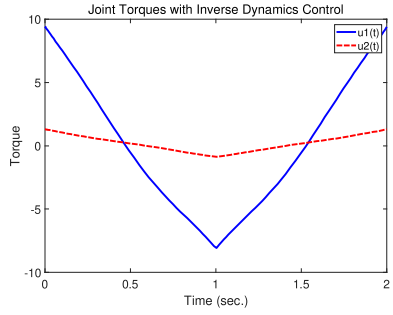
to track the cubic polynomial reference trajectory computed in problem 2. The outer loop control a_q should be the PD plus feedforward control you used as the inner loop control in problem 2, i.e.,

$$a_{q_1} = a_1^d + k_{p1}\,(q_1^d - q_1) + k_{d1}\,(v_1^d - \dot{q}_1)$$
$$a_{q_2} = a_2^d + k_{p2}\,(q_2^d - q_2) + k_{d2}\,(v_2^d - \dot{q}_2)$$

Using the same gains and torque bounds as in problem 2, simulate the closed loop system with zero initial conditions for 2 seconds and and plot the joint responses, input torques, and tracking errors. The responses should look like those in the following figure. Do not save or print out these responses. 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) Simulate the response for 2 seconds using the same nonzero initial conditions as in Problem 2 and generate plots of the joint responses, input torques, and tracking errors.

(b) What is the error after t = 1 second? after t = 2 second? How do the joint torques and tracking errors compare with those of the previous two cases?

Problem 4

Compare the performance of PD Control, PD + Feed Forward Control, and Inverse Dynamics Control based on the results obtained in the preceding questions. Analyze their respective strengths and weaknesses. Please limit your response to one paragraph.

What to turn in

Your homework should contain all requested plots, e.g., joint responses plus the reference position, joint torques, and joint tracking errors for every problem. All plots should be well labeled and titled, and **include a short explanation** of what they represent. Paste the codes of 5 m-files $< filename.m >$ in the end: (1) manipulator (2) the cubic polynomial reference trajectory (3) PD control (4) PD Plus Feedforward Control (5) Inverse Dynamics Control.

You should also upload all MATLAB programs to the Gradescope website (a .tar.gz or .zip file). Make sure to name the corresponding files that generate the plots as
< uniqname_problem1.m >, < uniqname_problem2.m >, < uniqname_problem3.m >.