

# ECE 489 Robot Dynamics and Control

Spring 2007

## Lab # 3: PD versus Inverse Dynamics Control

Due Date, April 5, 2007

In this lab you will compare the tracking performance of PD-control, PD-control with feedforward of the reference trajectory, and inverse dynamics control for a two-link revolute joint robot. You will modify and use the Matlab simulation model of a two-link robot that you wrote in Lab 2.

### Procedure:

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.

### I: The Plant Model

The robot to be controlled is a two-link, planar, robot moving in a horizontal plane. Therefore, you first need to modify your robot model by removing the gravity terms. Define inputs  $\tau_1$  and  $\tau_2$  as the joint torques, if your model does not already include them.

Use the following parameters for the robot:  $m_1 = 7.848$ ,  $m_2 = 4.49$ ,  $\ell_1 = 0.3$ ,  $\ell_{c1} = 0.1554$ ,  $\ell_{c2} = 0.0341$ ,  $I_1 = 0.176$ ,  $I_2 = 0.0411$ . All units are in the MKS-system. These parameters correspond to the 2-link direct-drive robots in B-16 CSL. Note that the length of link 2 does not enter into the equations of motion. Do not include any friction terms in your model. Make sure that any errors found in your model in Lab 2 have been corrected.

### II: PD-Control

1. Implement an independent joint PD-controller. The control inputs  $\tau_1$  and  $\tau_2$  are computed as

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

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  $\zeta = 1$  and a natural frequency  $\omega = 10$ . Since all real motors have limited torque capability, the outputs  $\tau_1$  and  $\tau_2$  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  $\pi/2$  radians from  $t = 0$  to  $t = 1$  second, followed by a step back to 0 from  $t = 1$  to  $t = 2$  seconds as shown below

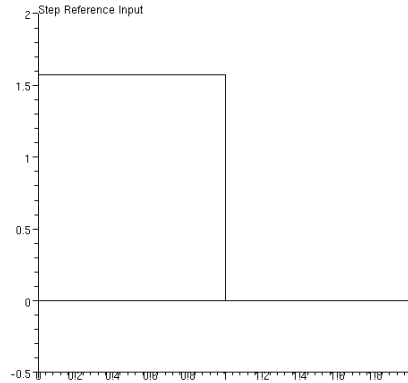


Figure 1: Reference Input for PD-Control

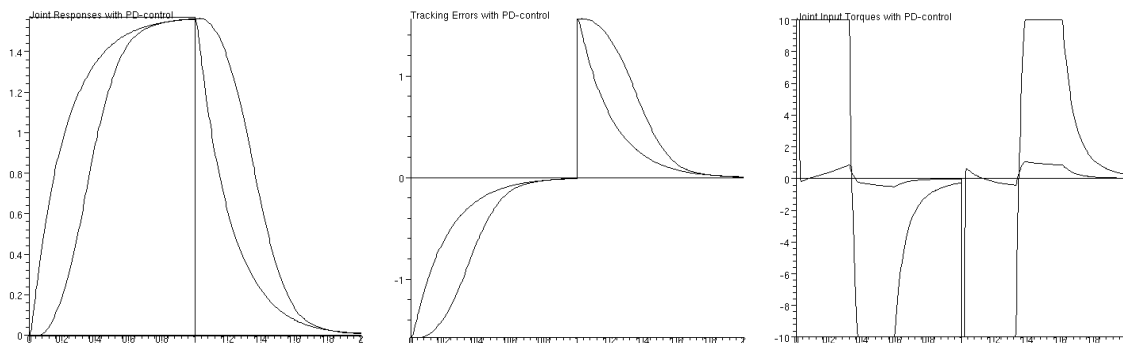


Figure 2: Joint response, tracking error, and joint torques using PD-control

2. 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 below you have an error in your simulation.
3. 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 for inclusion in your lab report.
4. 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.

### III: PD Plus Feedforward Control

1. Write a Matlab program to generate a cubic polynomial trajectory from  $t = 0$  to  $t = 2$  seconds. See Section 7.5 of the text. The trajectory should satisfy

$$q(0) = 0 ; \dot{q}(0) = 0$$

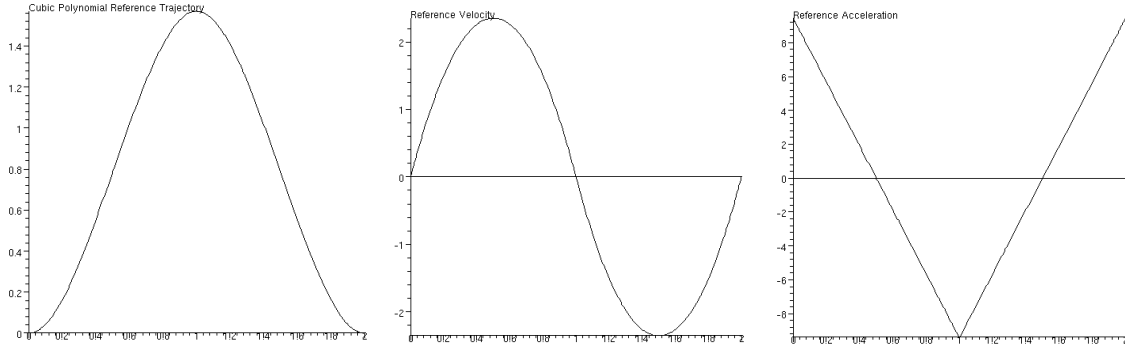


Figure 3: Cubic polynomial reference trajectory, reference velocity, and reference acceleration

$$\begin{aligned} q(1) &= \pi/2 ; \quad \dot{q}(1) = 0 \\ q(2) &= 0 ; \quad \dot{q}(2) = 0 \end{aligned}$$

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 below.

2. Generate plots of the reference position, velocity, and acceleration and print them out for later use in your writeup. This trajectory will be used as the reference trajectory for both joints.
3. Implement a PD plus feedforward control. The joint inputs in this case are

$$\begin{aligned} \tau_1 &= a^d + k_{p1}(q^d - q_1) + k_{v1}(v^d - \dot{q}_1) \\ \tau_2 &= a^d + k_{p2}(q^d - q_2) + k_{v2}(v^d - \dot{q}_2) \end{aligned}$$

Note that the outputs of the trajectory generator are required by the controller.

4. Using the same PD gains and bounds on the maximum torque as in Part II, simulate the response of the system for 2 seconds with zero initial conditions. The responses should look like those below. Do not save or print out these responses.
5. Now simulate the system using the same non-zero initial conditions as in Part II, and generate plots of the joint responses, input torques, and tracking errors, as before. 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 Part II?

## Part IV: Inverse Dynamics Control

1. Finally, simulate an inverse dynamics control

$$\tau = D(q)a_q + C(q, \dot{q})\dot{q} + g(q) \quad (1)$$

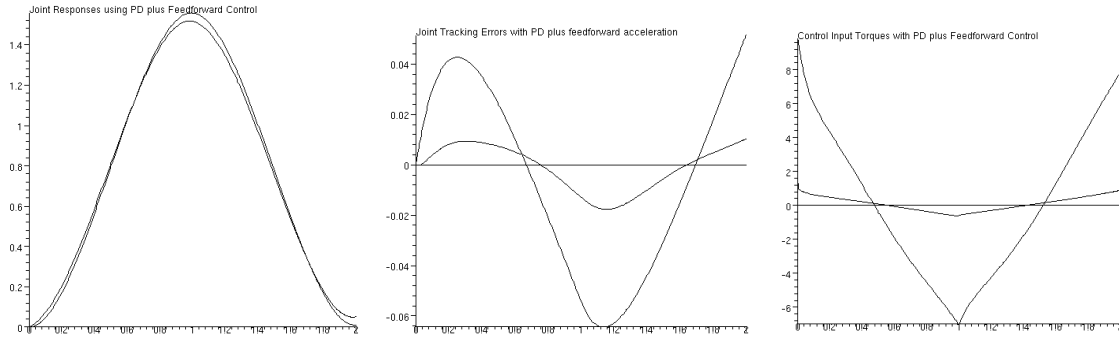


Figure 4: Responses of PD-control with feedforward of the reference trajectory

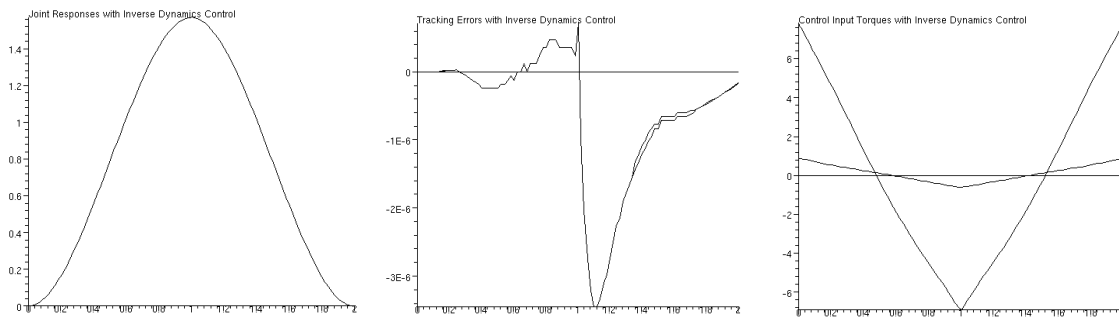


Figure 5: Response using inverse dynamics control

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

$$\begin{aligned}
 a_{q1} &= a^d + k_{p1}(q^d - q_1) + k_{v1}(v^d - \dot{q}_1) \\
 a_{q2} &= a^d + k_{p2}(q^d - q_2) + k_{v2}(v^d - \dot{q}_2)
 \end{aligned}$$

- Using the same gains and torque bounds as in Part III, simulate the closed loop system with zero initial conditions for 2 seconds and plot the joint responses, input torques, and tracking errors. The responses should look like those below. 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 above is mostly due to numerical round-off in the simulation. Therefore, your tracking errors may look somewhat different than the above, but your joint response and torque input curves should look like those above.

- Simulate the response for 2 seconds using the same non-zero initial conditions as in Part III and generate plots of the joint responses, input torques, and tracking errors. 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?

4. Draw conclusions about the performance of the various controllers.

### **What to turn in**

Your lab report should contain plots of the joint responses, joint torques, and joint tracking errors for each of the three cases plus the reference position, velocity, and acceleration for the cubic polynomial trajectory. You should also submit listings of all Matlab programs.