RBE 500 — FOUNDATIONS OF ROBOTICS

Instructor: Siavash Farzan

Spring 2022

Problem Set 3

Due: March 16, 2022 at 11:59 pm

Please show your work. Correct answers not accompanied by sufficient explanations will receive little or no credit.

Problem 1 (4 points)

Consider the 2-DoF planar prismatic-revolute robot manipulator shown in Figure 1. The position of the end-effector with respect to Frame 0 is given by:

$$o_2^0 = \begin{bmatrix} x_{ef} \\ z_{ef} \end{bmatrix} = \begin{bmatrix} l_1 + l_2 \cos\theta_2 \\ d_1 + l_2 \sin\theta_2 \end{bmatrix}$$

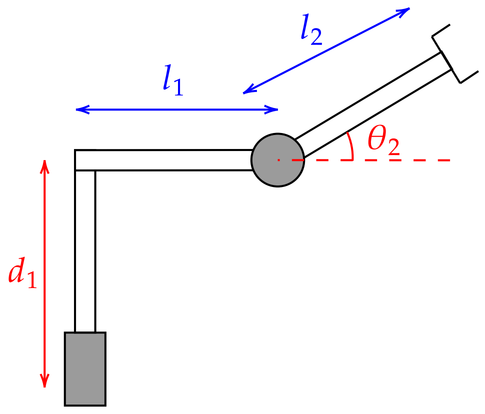


Figure 1: Planar robot manipulator for Problem 1.

a) Find the 2×2 Jacobian matrix that relates the two joint velocities to the linear velocity of the end-effector.

b) Using the Jacobian matrix calculated in part (a), determine the singular configurations of the robot manipulator (if any), and explain what DoF is lost at each singularity (resulting in a restricted motion of the end-effector).

Problem 2 (4 points)

a) Consider the 4R planar robot in Figure 2. The manipulator Jacobian in the configuration $\theta_1 = \theta_2 = 0$, $\theta_3 = \pi/2$, $\theta_4 = -\pi/2$ is given by:

$$J(\theta) = \begin{bmatrix} 0 & 0 & 0 & l_3 \\ 0 & -l_1 & -l_1 - l_2 & -l_1 - l_2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

We apply a force $f = [10, 10, 0]^T$ and a moment $n = [0, 0, 10]^T$ to the end-effector. What torques are required at the joints to keep the robot in place?

b) Determine the singularities of a 4R robot manipulator whose Jacobian is:

$$J(\theta) = \begin{bmatrix} l_3 s_4 + l_2 s_{34} + l_1 s_{234} & l_3 s_4 + l_2 s_{34} & l_3 s_4 & 0 \\ l_4 + l_3 c_4 + l_2 c_{34} + l_1 c_{234} & l_4 + l_3 c_4 + l_2 c_{34} & l_4 + l_3 c_4 & l_4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

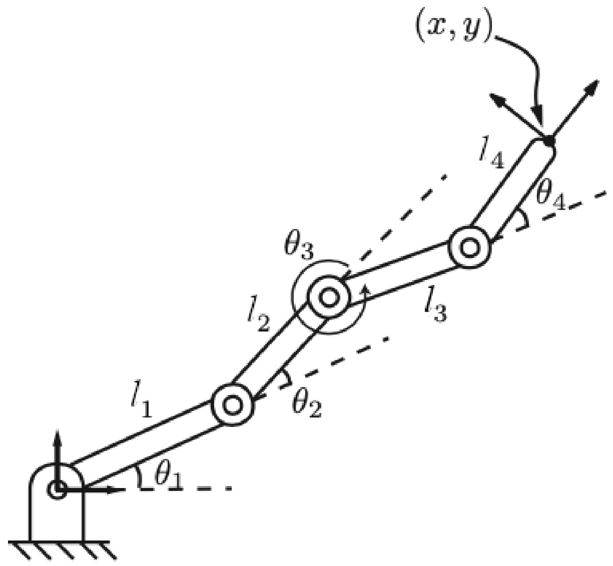


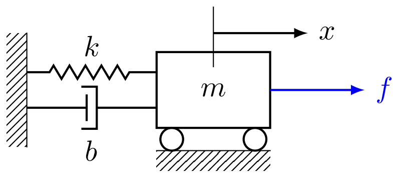
Figure 2: The 4-DoF planar robot for Problem 2.

Problem 3 (6 points)

Consider a one-DoF mass-spring damper system, for which the system dynamics are given in a differential equation of the form:

$$m\ddot{x} + b\dot{x} + kx = f$$

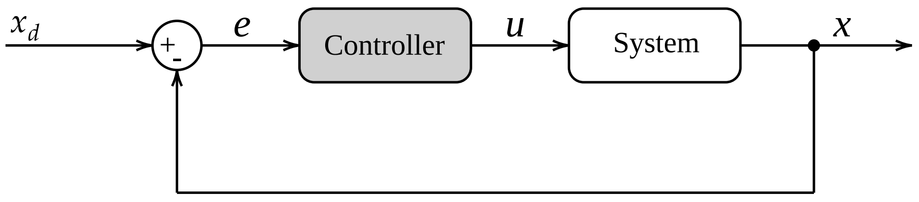
where x is the horizontal position of the mass, and f is the control force.



A controller is designed as:

$$f = k_p e + k_d \dot{e}$$

Consider the following block diagram for the overall closed-loop system:



a) Convert the system model and the controller to the Laplace domain.

b) Find the transfer function for $\frac{F(s)}{E(s)}$, $\frac{X(s)}{F(s)}$ and $\frac{X(s)}{E(s)}$.

c) Find the closed-loop transfer function.

For the next steps, assume that the physical parameters of the system are given as:

$$m = 4\ kg, \quad b = 2\ Ns/m, \quad k = 0.1\ N/m$$

d) What is the natural frequency and damping ratio of the *uncontrolled system*? Is the uncontrolled system overdamped, underdamped, or critically damped? What is the 2% settling time of the response?

e) Design a P controller $f = K_p\ e$, where $e = x_d - x$ is the position error. What value of K_p yields critical damping?

f) Design a D controller $f = K_d\ \dot{e}$. What value of K_d yields critical damping?

g) Design a PD controller that yields critical damping and a 2% settling time of 0.01 *sec*.

Problem 4 **(6 points)**

Consider the following robot joint model:

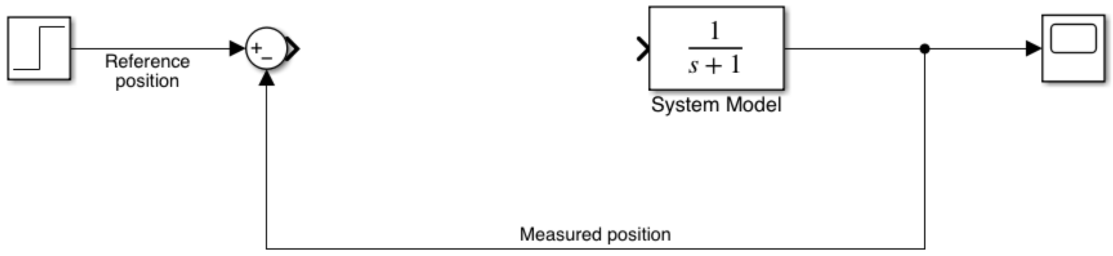
$$M\ddot{\theta}(t) + b\dot{\theta} = u(t) + d(t)$$

where M is the inertia of the link, b is the effective damping on the link, θ is the joint angle, u is the actuator torque (input), and d is the disturbance acting on the system.

a) First, assume that disturbance is zero and take $M = 10,\ b = 1$. Design a PD controller such that the closed-loop system is *critically damped*, and *settling time is 2 seconds*. Do not do this by directly tuning the gains, instead, calculate the K_p and K_d gains using natural frequency and damping ratio.

b) Simulate the system above in Simulink by following the steps below:

- Open MATLAB and open a new script file (`m` file).
- Write the system parameters to this script file, for example: `M=10; b=1; ...` (and other parameters you might need).
- Run this script so that the parameters are loaded to the workspace. Please beware that if you add new parameters or change the values of the parameters, you will need to run the script again to load them to the workspace.
- Now, run Simulink by typing `simulink` in the command window. The Simulink add-on will be lunched.
- Implement a closed-loop system in Simulink. First, put down the general structure as below.



For that you will need to first create a `Blank Model`, and then open the `Library Browser` with the icon at the ribbon. The block library includes a vast number of blocks for various purposes. The most used blocks are "`Commonly Used Blocks`", "`Continuous`", "`Discrete`", "`Math operations`", "`Sinks`" and "`Sources`".
Please examine these categories first. Then implement your closed-loop system using the "`transfer fcn`", "`scope`", "`sum`" and "`step`" blocks. You can drag the block into the Simulink workspace for that purpose. Modify the system model transfer function according to your system model. For that, double click on the transfer function block, and specify the coefficients of the numerator and denominator so that the system model represents the robot joint model of this problem. Enter the inertia and damping values parameterically (as M, b, etc. not as numbers), so that when you change and load them in the script file they will be changed in the Simulink model too.

- Now add your PD controller. You will be using the "`gain`", "`sum`" and "`derivative`" blocks. (Simulink has a PID block itself. Please do not use this block. Implement it with basic blocks as explained). While assigning values to the gains, again, use parametric names (K_p, K_d) and specify them in the script file (don't forget to run the script file again so that the values are loaded).

– The "`source`" or "`reference position`" in this simulation is a step signal that changes from zero to one. The changes in the position of the system is displayed with a "`scope`" block. Modify the step block and set 'Step time" to 0. Now run your simulation by pressing the play button. After the simulation is finished, double-click on the scope block to see the response of the system. Explain the process and be sure to include the scope plot and a screenshot of the Simulink model in your report.

c) Tuning: You may have noticed that the system performance (overshoot and settling time) is quite close to your goal, but it is not exactly there. Report the current settling time in your report (you may need to zoom in to the plot to see the exact settling time). This is because the equations defining the relations between the damping ratio, natural frequency, settling time and overshoot, are approximations, not exact relations. Specifically, the settling time equation is only accurate when damping ratio is $<< 1$. Now, tune the values of K_p and K_d so that you get closer to your goal performance (no overshoot, settling time of 2 seconds). While doing that, check the effect of K_p and K_d on the system performance. Explain your process and include a plot showing the results after tuning.

d) Now add a constant disturbance to the system, by using "`constant`" block from "`Sources`". Assign the value D to the block. Define this same parameter in the script and set its value to $D = 0.5$. Provide the plot for θ with the same gain values to the previous question. You will see some steady-state error. Report the amount of steady-state error.

e) Keeping the disturbance the same, add the integral term to your controller (using the "`integrator`" and "`gain`" Simulink blocks) and tune the P, I and D gains to obtain the desired performance and remove the steady state error even under the disturbance effect. Report the current K_p, K_d and K_i values together with your new plot and a screenshot of the Simulink model.