MAE C163C / C263C Homework #4

(Due via Gradescope by 11:59pm PDT on Friday, 5/2)

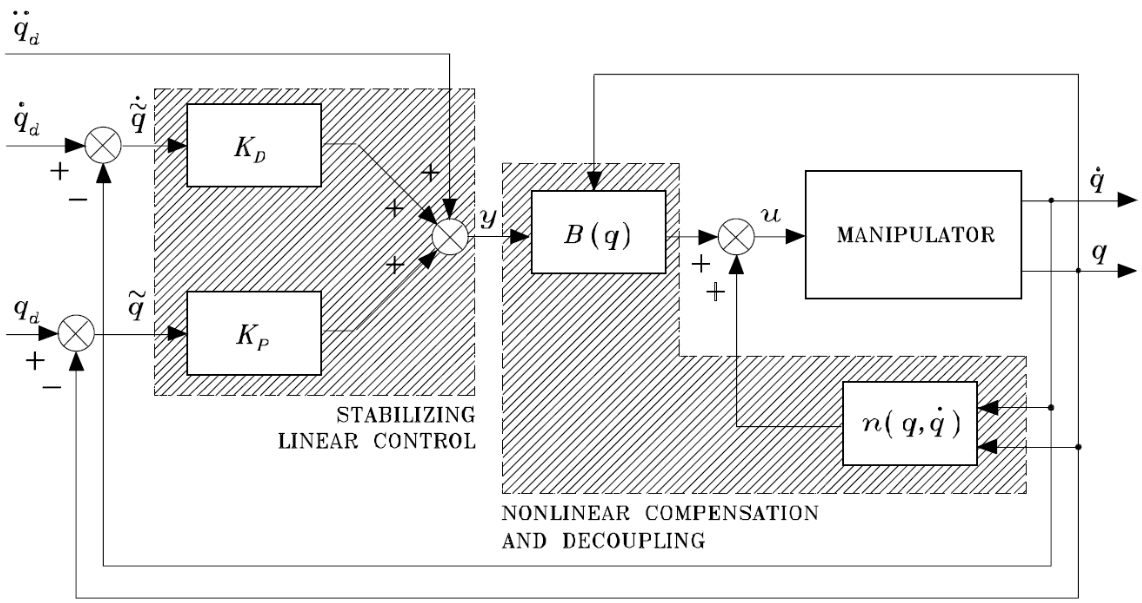
1. Analytical Jacobian

a) Report the 3x3 transformation matrix $T(\phi_e)$ that relates an angular velocity vector $\underline{\omega}_e$ and the time derivative of Euler angles $\dot{\underline{\phi}}_e$ through the equation $\underline{\omega}_e = T(\phi_e)\dot{\underline{\phi}}_e$. Consider XYX Euler angles that describe the (i) rotation of the reference frame by an angle φ about axis x, followed by (ii) the rotation of the current frame by an angle ϑ about axis y', followed by (iii) the rotation of the current frame by an angle ψ about axis x''.

b) Report the 6x6 transformation matrix $T_A(\phi_e)$ that relates the geometric Jacobian J and the analytical Jacobian J_A through the equation $J = T_A(\phi_e)J_A$.

Hint: Reference Siciliano et al., Sec. 2.4 for a review of Euler angles and Sec. 3.6 for a discussion of the analytical Jacobian.

2. Joint Space Inverse Dynamics Control



The block diagram above describes joint space inverse dynamics control.

Consider a two-link planar arm with the following parameters:

$$a_1 = a_2 = 1 \text{ m}, \quad l_1 = l_2 = 0.5 \text{ m}, \quad m_{l_1} = m_{l_2} = 9 \text{ kg}, \quad I_{l_1} = I_{l_2} = 3 \text{ kg} \cdot \text{m}^2$$

The arm is assumed to be driven by two identical actuators with the following parameters:

$$k_{r_1} = k_{r_2} = 50, \quad m_{m1} = m_{m2} = 1 \text{ kg}, \quad I_{m1} = I_{m2} = 0.007 \text{ kg} \cdot \text{m}^2,$$

$$F_{m1} = F_{m2} = 0.01 \text{ N} \cdot \text{m} \cdot \text{s/rad}$$

The generalized inertia matrix for an augmented link model of this planar 2R manipulator can be written as:

$$\mathrm{B}(\boldsymbol{q}) = \begin{bmatrix} B_{11} & m_{l_2} l_2 (a_1 \cos(q_2) + l_2) + k_{r_2} I_{m_2} + I_{l_2} \\ m_{l_2} l_2 (a_1 \cos(q_2) + l_2) + k_{r_2} I_{m_2} + I_{l_2} & B_{22} \end{bmatrix}$$

where

$$B_{11} = I_{l_1} + m_{l_1} l_1^2 + k_{r_1}^2 I_{m_1} + I_{l_2} + m_{l_2}(a_1^2 + l_2^2 + 2a_1 l_2 \cos(q_2))$$

and

$$B_{22} = I_{l_2} + m_{l_2} l_2^2 + k_{r_2}^2 I_{m_2}$$

The motion of this manipulator is subject to the following constraints:

1) The 4th quadrant represents a fixed obstacle such that no part of the manipulator can cross over the positive x-axis or negative y-axis into the 4th quadrant.
2) There exists a “ceiling” at y = 1.5 m such that no part of the manipulator can invade the space y > 1.5 m.
3) The “elbow” joint represented by joint coordinate q_2 is restricted such that link 2 cannot swing through link 1. That is, $-\pi \leq q_2 \leq \pi$.

__Implement the Drake Simulation Python function__:

Complete the function `run_simulation` in *HW4.py* using the inline instructions within the comments.

__Simulate the Passive Dynamics of the Manipulator__

1. Complete the section labeled “Section 1” in *HW4.py* to simulate the __complete nonlinear equations of motion__ of the manipulator with gravity, but without any control torques applied. Assume an initial rest configuration of $\underline{q}_i = [q_{1,i} \;\; q_{2,i}]^T = [30° \; -60°]^T$ and a simulation duration of 2.5 sec.
2. Using the provided Python function `animate_2R_planar_arm_traj`, animate the motion of your planar 2R manipulator to verify that it looks like a double-pendulum falling under its own weight. Note that the manipulator will pass through the 4th quadrant without applied control torques.
3. Using the provided Python function `plot_snapshots`, plot snapshots of the manipulator configurations starting from $t_0 = 0$ and increasing in $\Delta t = 0.1$ sec increments to the final time of $t_f = 2.5$ sec.

__Design an Inverse Dynamics Controller and Simulate the Manipulator with Control Torques__:

1. Simulate the manipulator under the action of an inverse dynamics controller by completing section labeled “Section 2” in *HW4.py* using tips in this problem statement. Assume an initial rest configuration of $\underline{q}_i = [q_{1,i} \;\; q_{2,i}]^T = [30° \; -60°]^T$, a final desired rest configuration of $\underline{q}_d = [q_{1,d} \;\; q_{2,d}]^T = [240° \; 60°]^T$, and a simulation duration of 2.5 sec.

Report your K_P and K_D matrices. Your controller must satisfy the conditions $|q_{1,d} - q_1| < 2 \text{ deg}$ and $|q_{2,d} - q_2| < 2 \text{ deg}$ over the entire simulated trajectory (the subscript d specifies the desired joint angles).

In this greatly simplified version of inverse dynamics control (also known as computed torque control), you will use an "average" $\hat{B}_{avg}$ generalized mass matrix and neglect centripetal, Coriolis, friction, and gravitational terms. You will further simplify the mass matrix by diagonalizing it in order to treat each joint independently for controller design purposes.

Thus, you will attempt to control the arm by inverting a model of the arm that you know to be inaccurate, but which you hope will be good enough. <u>You will design your controller using an estimated model of the arm dynamics, but simulate the effects of your controller using the full model of the arm dynamics.</u>

$$\boldsymbol{u} = \hat{B}_{\text{avg}}\ddot{\boldsymbol{q}}_d + K_D(\dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}}) + K_P(\boldsymbol{q}_d - \boldsymbol{q})$$

where:

$\hat{B}_{avg} =$ "average" generalized mass matrix (you will create this in *HW4.py*)

$K_D =$ diagonal damping matrix (i.e., diagonal matrix of derivative control gains)

$K_P =$ diagonal stiffness matrix (i.e., diagonal matrix of proportional control gains)

$\underline{q}_d, \dot{\underline{q}}_d, \ddot{\underline{q}}_d =$ desired joint angles, angular vel., and angular accel. from trajectory planner

There is no single correct answer for the gain matrices. There are many viable combinations that will satisfy the design requirements.

2. Using the provided Python function `animate_2R_planar_arm_traj`, animate the motion of your planar 2R manipulator to verify that it follows the desired trajectory. Note that you are applying a controller designed for a simplified system to a manipulator whose motion should abide by its complete nonlinear equations of motion.

3. Using the provided Python function `plot_snapshots`, plot snapshots of the actual and desired manipulator configurations starting from $t_0 = 0$ and increasing in $\Delta t = 0.1$ sec increments to the final time of $t_f = 2.5$ sec.

4. Plot the time history of both joint position errors in <u>deg</u> in the same figure. Use a solid red line for joint 1 position errors and a solid blue line for joint 2 position errors. Add black horizontal dashed lines at y = -2 deg and y = 2 deg.

5. Plot the joint angles in <u>deg</u> as a function of time. Plot the actual and desired values for both joint coordinates on the same plot. Use a red solid line for q_1, a red dashed line for $q_{1,d}$, a blue solid line for q_2, and a blue dashed line for $q_{2,d}$. Add a vertical dashed line at 1.25 sec (the time at which the manipulator must pass through a via point that has been designed into the pre-planned desired trajectory) and a legend.

6. Plot the joint angle velocities in deg/s as a function of time. Plot the actual and desired values for both joint angle velocities on the same plot. Use a red solid line for $\dot{q}_1$, a red dashed line for $\dot{q}_{1,d}$, a blue solid line for $\dot{q}_2$, and a blue dashed line for $\dot{q}_{2,d}$. Add a vertical dashed line at 1.25 seconds and a legend.

7. Plot the control torques in N·m as a function of time. Use a red solid line for τ_1 and a blue solid line for τ_2. Add a vertical dashed line at 1.25 sec and a legend.

Summary of deliverables:

Your submission should include:

- Plots of animation snapshots
- Your K_P and K_D gain matrices
- Labeled time history plots
- A plot of your final Drake block diagram
- Your **completed** *HW4.py* file converted to a PDF (To facilitate grading, see the relevant PyCharm help page for how to print a *.py* file to a PDF).

NOTE: Each student must submit their own independent work. **For full credit, you must submit to Gradescope all custom Python code** (e.g. *HW4.py*) **and requested plots with labels.** You may save this content to PDF or take screenshots for electronic submission via Gradescope. Files of the *.py* and *.toml* format cannot be directly uploaded to Gradescope and should not be e-mailed to instructors for grading. The more intermediate results and comments you provide, the greater the opportunity for partial credit.