ME 547 Winter 2024

Lab 3 Instructions: Dynamics and Controls *(20 pts)*

The lab consists of two parts, both on the Franka Emika Panda robot. After the lab, each group needs to submit a group lab report (see the General Lab Report Guidelines).

Objectives:

- Part 1: Comparison of the trajectory-tracking accuracy and the robustness of different controllers.
- Part 2: Designing a force controller on the end-effector.

Pre-Lab (0 pts):

There is no pre-lab for Lab3. However, students are required to have knowledge about the following materials prior to their lab attendance:

- Commonly used blocks in Simulink, including Constant, Gain, Mux, Demux, Scope, To Workspace, Matrix Multiply, Sum, Reshape, Step, Ramp, Sine, Selector, Subsystem, Goto, From, Transpose, etc.
- Simulink blocks which are exclusively designed for the Franka Emika Panda robot (mentioned later in this section).
- Simple controllers including P, PI, PD and the effect of increasing/decreasing their gains on the error signal.

Franka Emika blocks:

1. Apply Control (Figure 1): Represents the robot's block which takes the applied torque of the 7 joints as a 7x1 vector.
2. Get Robot State (Figure 2): Represents the sensor's block which outputs the states of the robot. You can have the parameters you want by double-clicking on this block and writing them in the Parameters section (Figure 3). The list of available signals can be found in this link: https://frankaemika.github.io/libfranka/structfranka_1_1RobotState.html

3. Get Model (Figure 4): Provides the physical parameters of the robot including pose, bodyJacobian, zeroJacobian, mass, Coriolis and gravity. These parameters are described in this link: https://frankaemika.github.io/libfranka/classfranka_1_1Model.html

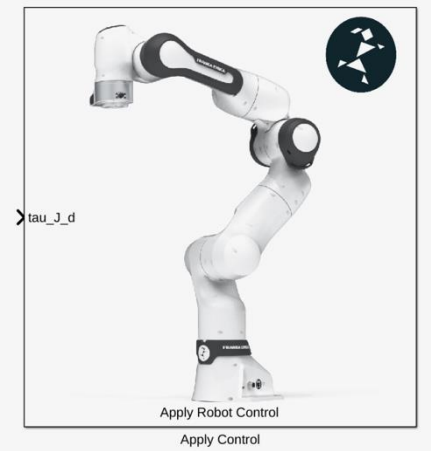

Figure 1- Apply Control block.

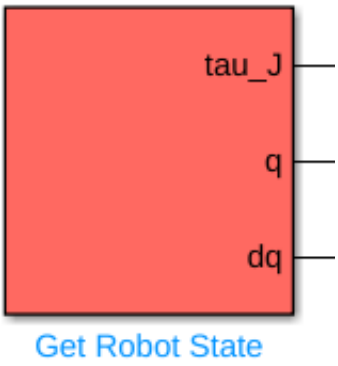

Figure 2 – Get Robot State block.

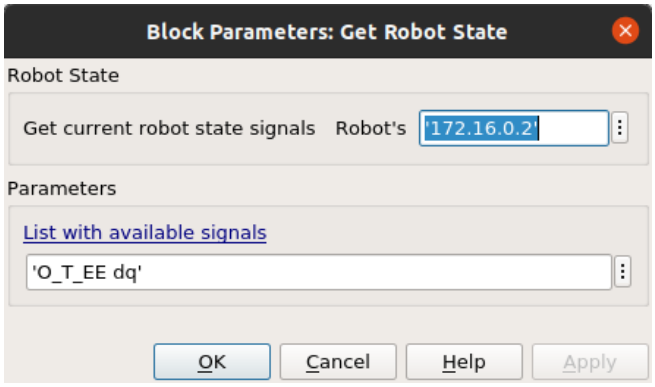

Figure 3 – Robot states specification.

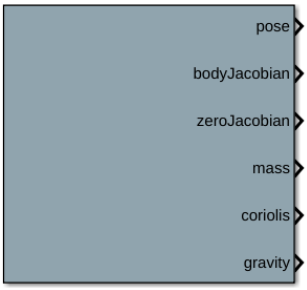

Figure 4- Get Model block.

In-Lab (4 pts):

Part 1: Trajectory tracking control

The objective of this part is to compare the performance of different controllers while the robot follows a circle as the desired trajectory using joints 2 and 4 (see Figure 5). The inverse kinematics is already solved for this robot. Your tasks will include computing the errors and assembling the controller block diagram in the file joint_space_trajectory_control.slx.

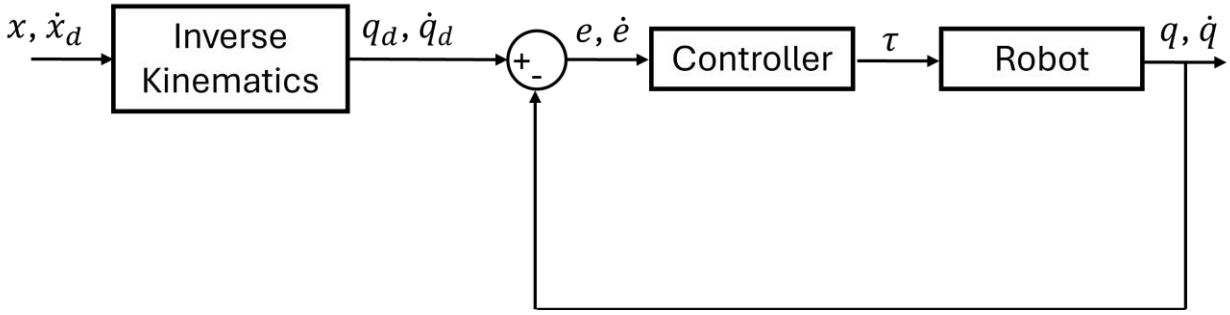

Figure 5: Block diagram

Step 1: Error Generator

The error generator block takes the desired trajectory in joint space, and sensory feedback (position and velocity of joints) and outputs the position and the velocity errors. The desired trajectory is already coded in Simulink (within the "Desired" block and outputs the desired position, velocity and acceleration)

In the Trajectory Generation Block, calculate the position and the velocity error:

$$e = q_d - q$$
$$\dot{e} = \dot{q}_d - \dot{q}$$

where q_d, $\dot{q}_d$, q and $\dot{q} \in R^7$ represent the desired position, desired velocity, current position and current velocity of the robot joints, respectively. To obtain the current position and the velocity of the robot joints, see the "Signaling" block. Add scope to the errors since you need to monitor them while running the robot.

Step 2: PD controller

After completing the trajectory generation block, go to the "PD Controller" block and create PD controllers:

$$\tau_{PD} = K_p e + K_D \dot{e}$$

Once the Trajectory Generation and the PD Controller block are completed. Run your code, open the error scope and tune your PD parameters such that the errors for all joints are within 0.1 rad.

Step 3: Feedforward controller

After tuning the PD controller, add the feedforward controller:

$$\tau_{ff} = M \ddot{q}_d$$

where M is the mass matrix and $\ddot{q}_d$ is the desired acceleration.

Step 4: Coriolis effect

Add the Coriolis effect. See the "Signaling" block to obtain the Coriolis effect.

Once the Structure of the controller is completed, you will need to run the robot and collect data under 6 different conditions:

- Without disturbance
  - PD
  - PD + feedforward (PD-FF)
  - PD + feedforward + Coriolis compensation (PD-FF-CC)
- With disturbance
  - PD
  - PD + feedforward (PD-FF)
  - PD + feedforward + Coriolis compensation (PD-FF-CC)

For each condition change the gain of the torques from each block into 0 or 1 depending on the condition before running. Execute your code for 50 seconds and save the data from MATLAB workspace.

Note: Before running, make sure to record $q, \dot{q}, e, \dot{e}, \tau_{PD}, \tau_{ff}, \tau_{coriolis}, \tau_{dist}, \tau_{total}, \tau_{sensor}$

Note: During running collect a video of your manipulator motion for a few cycles of each controller

Note: After running, make sure to save all data from the Matlab workspace since each time the code is executed, the new data will be overwritten on the previous data.

Part 2: Force control

In this part, you will apply force by the end-effector of the robot on a surface when the robot (1) does not move, (2) moves on a pre-defined path without a bump and (3) move on the same path with a bump.

Based on Figure 6, you will design a PI controller which takes the error (difference between the desired and the current joint torques) and outputs the command torques to the joints.
The desired force is 10N on the desktop.

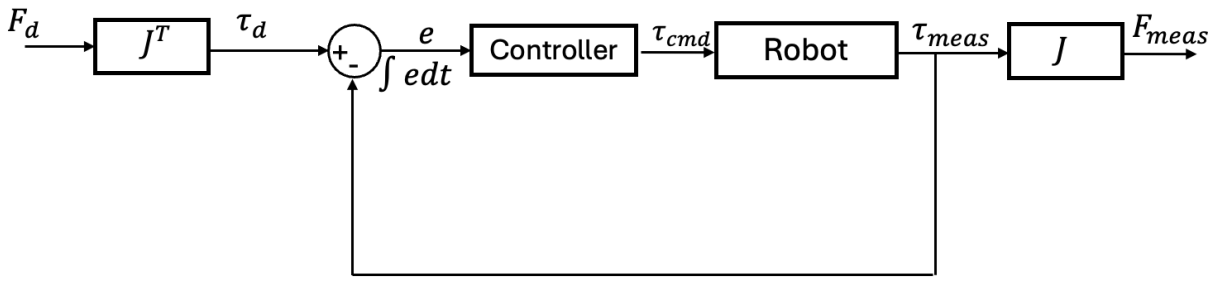

Figure 6 – Block diagram of force control.

Open the file force_control.slx and design a PI controller in the Force Controller > Force Control. Run the robot with the gains Kp = 0.1 and Ki = 0.1, and start tuning the controller (end effector of the robot should be on the desk before the code starts to run. A good tunning will give you an error of less than 0.2 N on the desktop.
Once the PI parameters are tuned, add the desired torque to the output of the controller as the feedforward torques and run the three conditions for 10 seconds:
Condition1: the robot does not move.
Condition2: the robot moves on a pre-defined trajectory.
Condition3: the robot moves on a pre-defined trajectory with a bump.

Note: Before starting to run the code, make sure you are recording the desired force, desired torque, measured force and the measured torque.
Note: during the experiment make sure to video capture your robot behavior.
Note: After running, make sure to save all data from the Matlab workspace since each time the code is executed, the new data will be overwritten on the previous data.

Post-Lab report:

Part 1:

Your report for part 1 should contain the tuned PD controller gains as well as the following results for the **final 10 seconds** of the recorded data (except the end-effector trajectory plot):

Joint trajectories (1 pt)

Create a 2x2 subplot. For each, plot the desired joint trajectory and overlay the joint actual trajectory while PD, PD-FF, and PD-FF-CC are in the loop (each graph should contain 4 lines with the legends “Desired”, “PD”, “ PD-FF”, “ PD-FF-CC”)

- Subplot(2, 2, 1): Joint 2, undisturbed
- Subplot(2, 2, 2): Joint 2, disturbed
- Subplot(2, 2, 3): Joint 4, undisturbed
- Subplot(2, 2, 4): Joint 4, disturbed

Error plot and table (1 pt)

Create a 2x2 subplot. For each, plot the trajectory error when PD, PD-FF, and PD-FF-CC are in the loop (Each graph should contain 3 lines with the legends “PD”, “PD-FF”, “PD-FF-CC”)

- Subplot(2, 2, 1): Joint 2, undisturbed
- Subplot(2, 2, 2): Joint 2, disturbed

- Subplot(2, 2, 3): Joint 4, undisturbed
- Subplot(2, 2, 4): Joint 4, disturbed

Calculate the root mean squared error (RMSE) for each joint and the sum of RMSEs (as Overall) under different conditions and report the values in a table similar to Table 1.
Note: Since there are more than one circle in 10 seconds, report the RMSE in the form of mean and standard deviation (mean(RMSE) $\pm$ std (RMSE)) of the last 10 second cycles.

Table 1: Error.

	Undisturbed			Disturbed		
	PD	PD-FF	PD-FF-CC	PD	PD-FF	PD-FF-CC
Joint 2						
Joint 4						
Overall						

Torque plots (1 pt)

Create a 2x2 subplot. For each, the output torque of the controller and overlay while PD, PD-FF, and PD-FF-CC are in the loop (each graph should contain 3 lines with the legends "PD", "PD-FF", "PD-FF-CC")

- Subplot(2, 2, 1): Joint 2, undisturbed
- Subplot(2, 2, 2): Joint 2, disturbed
- Subplot(2, 2, 3): Joint 4, undisturbed
- Subplot(2, 2, 4): Joint 4, disturbed

Control effort table (1 pt)

Report the control effort in the form of a table similar to Table 2.

Table 2: Control effort

	Undisturbed			Disturbed		
	PD	PD-FF	PD-FF-C	PD	PD-FF	PD-FF-C
Control effort						

The control effort u can be calculated in the form of the sum of the integral absolute torques:

$$IAT = \sum_{j\in\{2,4\}} \int_{t_i}^{t_f} |\tau_j| \, dt$$

End-effector trajectory plot (1 pt)

Using the function ForwardKinematics.m provided on Learn, calculate the position of the end effector, create a 2x1 subplot and plot the desired end-effector trajectory as well as the recorded trajectory under the 3 control conditions (each graph should contain 4 lines with the legends "Desired", "PD", "PD-FF", "PD-FF-CC"):

- Subplot(2, 1, 1): Undisturbed
- Subplot(2, 1, 2): Disturbed

Note: Unlike the previous plots which include the last 10 seconds, this plot should only be for one cycle of motion.
Note: Make sure the plots of the end-effector trajectory have the same lengths in x and y axes since you are reporting circles (If you are using Matlab, you can use `pbaspect([1, 1, 1])`).

Discussion

You should provide the following points in the discussion. For each point, you may need to support your statements with mathematical equations and/or results obtained and reported in your graphs and tables. Bring the numbers into your discussion:

1. **Joint Trajectories: (1 pt)**
    - Compare the performance of the different controllers (PD, PD-FF, and PD-FF-CC) in tracking the desired trajectory for both undisturbed and disturbed conditions. Discuss which controller provided the best tracking performance and why.
    - Analyze the impact of disturbances on the trajectory tracking performance and how each controller compensates for it.
2. **Error Analysis: (1 pt)**
    - Discuss the trends observed in the trajectory error plots for the different controllers and conditions. Highlight any significant differences in error reduction between the controllers.
    - Interpret the root mean squared error (RMSE) values from the table. Discuss how the choice of controller affects the precision of joint movement.
3. **Torque Analysis: (1 pt)**
    - Analyze the torque plots to understand how each controller modulates the torque output in response to disturbances.
    - Discuss the relationship between the torque patterns and the error plots. Explain how changes in torque affect trajectory tracking.
4. **Control Effort: (1 pt)**
    - Compare the control effort required for each controller under both undisturbed and disturbed conditions. Discuss which strategy is more energy-efficient and why.
    - Explore the trade-offs between control effort and tracking performance. Discuss whether a lower control effort compromises the accuracy of trajectory tracking.
5. **End-Effector Trajectory: (1 pt)**
    - Analyze the end-effector trajectories to evaluate the overall performance of the robotic arm under different controllers and conditions.
6. **General Observations: (1 pt)**
    - Summarize the key findings from the experiments and discuss their implications for the design and control of robotic systems.
    - Suggest potential improvements or modifications to the controllers based on the observed results.

Part 2:

Your report for part 2 should contain the tuned controller as well as the following results for the **last cycle of the motion.**

Torque plot (1 pt)

Create seven 2x3 subplots for every joint, the top ones for the desired and the measured joint torques of all 3 conditions, and the bottom ones for the error (difference between the two signals). For example, for joint i:

- Plot i:
  - Subplot(2, 3, 1): Condition1, Joint i, Torque (With the legends "τ_d" , "τ_{meas}")
  - Subplot(2, 3, 4): Condition1, Joint i, Torque error
  - Subplot(2, 3, 2): Condition2, Joint i, Torque (With the legends "τ_d" , "τ_{meas}")
  - Subplot(2, 3, 5): Condition2, Joint i, Torque error
  - Subplot(2, 3, 3): Condition3, Joint i, Torque (With the legends "τ_d" , "τ_{meas}")
  - Subplot(2, 3, 6): Condition3, Joint i, Torque error

Note: You may need to low-pass filter the measured torques before reporting them. For example, you can use a 4th-order Butterworth low-pass filter (butter()) with a cutoff frequency of 10 Hz to minimize the noise, and a zero-phase filter to minimize the lag produced by the Butterworth filter (An example is provided on Learn with the name filter_design_example.m)

Force plot (1 pt)

Create three 2x3 subplots for every Fx, Fy and Fz, the top ones for the desired and the measured end-effector forces of all 3 conditions, and the bottom ones for the error (difference between the two signals). For example, for Fz we have:

- Plot Fz:
  - Subplot(2, 3, 1): Condition1, Fz (With the legends "$F_{z,d}$" , "$F_{z,meas}$")
  - Subplot(2, 3, 4): Condition1, Fz, Force error
  - Subplot(2, 3, 2): Condition2, Fz (With the legends "$F_{z,d}$" , "$F_{z,meas}$")
  - Subplot(2, 3, 5): Condition2, Fz, Force error
  - Subplot(2, 3, 3): Condition3, Fz (With the legends "$F_{z,d}$" , "$F_{z,meas}$")
  - Subplot(2, 3, 6): Condition3, Fz, Force error

Note: You may need to low-pass filter the measured forces before reporting them.

Discussion (3 pts)

You should provide the following points in the discussion. For each point, you may need to support your sentences with mathematical equations and/or results obtained and reported in your graphs and tables. Bring the numbers into your discussion:

Error Analysis: Analyze the error plots for both torque and force. Was the error consistently low, or were there significant deviations? Discuss any trends or patterns you observed in the

error plots. How and to what extent did the presence of a bump in Condition 3 affect the error compared to the other conditions?

System Dynamics: Reflect on how the robot's movement influenced the system's ability to maintain the desired force and torque. How did the pre-defined trajectory and the bump affect the system's dynamics and the controller's performance?

Improvements and Recommendations: Based on your analysis, suggest any improvements that could be made to the controller design or the experimental setup. Are there any additional factors that should be considered in future experiments to enhance the system's performance?