
(Spring 2007-08)

Lab Assignment #2

Due Tuesday, April 22nd

In this assignment you will implement several methods of joint control and estimate dynamic parameters of the PUMA. This project should be done in a group. See **Chapter 7 in Introduction to Robotics for references.**

Float Control

Modify the procedure `initFloatControl()` and `floatControl()` in `control.cpp` to apply the torques required to compensate for the gravitational forces on the manipulator. This control law has the following format:

$$\tau = G(\theta)$$

Hint: Since $G(\theta)$ is already available in a global variable, you can implement this controller in one line.

Joint Space Non-Dynamic

1. Implement, using direct PD control of the joint torques, a control law to keep the robot at the current configuration. The manipulator should remain at the initial configuration and should return to that configuration if it is disturbed. The control law will be invoked with the user command **“njhold”**. This control law has following format

$$\tau = -k_p(\theta - \theta_d) - k_v(\dot{\theta} - \dot{\theta}_d)$$

where $\dot{\theta}_d = 0$.

2. Using the control law developed in #1, implement a control law to move the robot to a given configuration. The manipulator should move to the given configuration and remain at that configuration even if it is disturbed. The control law will be invoked with the user command **“njmove $q1\ q2\ q3\ q4\ q5\ q6$ ”**, where $q1 - q6$ are the desired joint angles for the robot in degrees. Add gravity compensation to improve performance.

Joint Space Dynamic

1. Using the linearized(partitioned) control method, implement a control law to keep the robot at the current configuration. The manipulator should remain at the initial configuration and should return to that configuration if it is disturbed. The control law will be invoked with the user command **“jhold”**. This control law has following format

$$\begin{aligned}\tau &= \alpha\tau' + \beta \\ \alpha &= M(\Theta) \\ \beta &= B(\Theta)[\dot{\Theta}\dot{\Theta}] + C(\Theta)[\dot{\Theta}^2] + G(\Theta) \\ \tau' &= -k'_p(\theta - \theta_d) - k'_v(\dot{\theta} - \dot{\theta}_d)\end{aligned}$$

where $\dot{\theta}_d = 0$. See Section 7.6 in Introduction to Robotics. Use *gv.kp* and *gv.kv* for k'_p and k'_v .

2. Using the control law developed in #1, implement a control law to move the robot to a given configuration. The manipulator should move to the given configuration and remain at that configuration even if it is disturbed. The control law will be invoked with the user command “**jmove q1 q2 q3 q4 q5 q6**”, where *q1* - *q6* are the desired joint angles for the robot in degrees.

Experiment

1. Gathering data
 - (a) You can gather data by clicking **start** and **stop** in the row of **Gather** in **GUI**.
 - (b) The file name for gathering must have the extension of **.mat**. Otherwise, **Matlab** cannot read the data.
2. PD gains: k_p and k_v .
 - (a) Use $k_p = 400$ and $k_v = 40$ as default values.
 - (b) You can change the gains in the GUI. Each control mode has a separate set of control gains, so make sure you set the control mode before editing k_p or k_v . The value is sent to the server as soon as you hit tab, enter, or click the mouse in another input field.
3. Identify some of dynamic parameters of the manipulator.
 - (a) Use “jmove 0 0 90 0 0 0” to straighten the robot’s elbow.
 - (b) Use “njhold” to change the mode to direct PD (non-dynamic) control.
 - (c) Set k_p and k_v as follows, to stiffen the first two joints and loosen joint 3:

kp:	1600	1600	50	400	400	400
kv:	80	80	10	40	40	40
 - (d) Vary k_{v3} to make joint 3 oscillatory when it is disturbed. (Note: because of the large amount of friction in joint 3, you may need to set $k_{v3} < 0$ for oscillations to occur)
 - (e) Disturb the robot and gather joint positions and time for at least 3 periods.
 - (f) Plot graph of joint 3 position vs time.
 - (g) From the graph find natural damping ratio (ζ_n), natural frequency (ω_n) and effective inertia (m) for joint 3.
 - (h) Compare the effective inertia to the corresponding entry in the kinetic energy matrix given the supplied dynamic parameters. Compute the relative error. You should be able to get the error less than 10%.
 - (i) Use “njmove” to change the mode to direct PD (non-dynamic) with gravity compensation and repeat step (c)-(h).

- (j) Use “jmove” to change the mode to computed torque and repeat step (c)-(h). Note that gains will have to be different in “jmove” mode.
- (k) Compare the three results from “njhold”, “njmove”, and “jmove”.
- (l) Set kp and kv as follows:

kp:	400	400	400	400	400	400
kv:	40	?	40	40	40	40
- (m) Move the robot between “njmove 0 0 -30 0 0 0” and “njmove 0 40 -30 0 0 0”. Change the value of kv for joint 2 to create an example of under damped, critically damped, and over damped. Plot over-lapping graphs of the value of joint 2 in the 3 examples.
- (n) Using the gains you used in (m) for critically damped, move between “njmove 0 0 90 0 0 0” and “njmove 0 40 90 0 0 0”. Plot this movement for joint 2 and compare it to what happened with the same gains in (m).
- (o) Repeat (m) with jmove instead of njmove.
- (p) Write up a report showing the results. Please include calculations of ζ_n , ω_n , and m and discussion about your results and your graphs.
- (q) Please bring your results and calculation from the simulation to your lab period.

Make sure you put enough comments in your code so that we can understand what you’re doing!

Submit a short report describing what you learned in lab. Discuss special features in your code. Incorporate material from lectures.