Experimental Robotics (CS225A) Homework #03

(Fall 2014-15)

PUMA Control

In this assignment you will implement several methods of joint control and estimate dynamic parameters of the PUMA. For the submission, please include your code and a short report comparing the different controllers.

See Chapter 7 in Introduction to Robotics for reference.

You are expected to work on this assignment and submit it as a group.

Setup

1. Navigate to the *controlDLL* directory within the simulator folder. You'll find a Visual Studio 2010 solution file called **controlDLL.sln**.
   If you have not already installed Visual Studio 2010, you can get it here "https://www.dreamspark.com/Product/Product.aspx?productid=4".

2. Open **controlDLL.sln**. Under the filter **Sources**, you'll find a file called **control.cpp**. Double-click to open it.

3. This file is essentially an ensemble of all the controllers that you've been using so for your previous assignments. We had prepopulated it back then. But the copy provided to you, unfortunately, is empty. You'll be filling it back up over this and the next assignment.

4. The format of every controller in the file is thus:

   (a) There is a **init<control_mode>Control()** method which gets called once when the mode is changed.

   (b) There is a **<control_mode>Control()** method which gets called on every servo loop. You'll plug in your control law here, which converts the desired positions into joint torques.

   (where *<control_mode>* is one of the control methods that you've been using such as **"nj-move"**, **"track"** or **"jmove"**)

5. In addition, there is a single **preProcessControl()** method which is called before a call is made to *<control_mode>Control()* and a **postProcessControl()** method which is called after a call is made to *<control_mode>Control()* on every servo loop. We typically use these methods to do common processing, such as applying torque or joint limits.

6. In **control.cpp**, we have provided you the prototypes for all of these methods. If you change the prototypes, the simulator will refuse to work. For convenience you can define your own methods in the file and work with those. But the simulator will not be aware of them.

7. Finally, all the data you need to compute your joint torques such as current joint positions, desired joint positions, *etc.* is stored in a global variable **gv** which is passed to every method as an argument. Its an instance of the struct *GlobalVariables* which is defined in *GlobalVariables.h* which you can find under *Headers* in your solution tree. Detailed description of all the elements are in the file itself although you'll be introduced to the ones that you need for the assignment along with the problem description below.

Joint Space Control

Float Control

In **initFloatControl()** and **floatControl()** found in **control.cpp**, you will notice that no torque has been applied. Modify the functions 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 Control

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

$$\begin{aligned} \tau &= M(\Theta)\tau' \\ M(\Theta) &= I \\ \tau' &= \ddot{\theta}_d - k'_v(\dot{\theta} - \dot{\theta}_d) - k'_p(\theta - \theta_d) \end{aligned}$$

   where $\dot{\theta}_d = 0$ and $\ddot{\theta}_d = 0$. k'_v and k'_p are velocity and position gains respectively for the unit mass. Since the effect of dynamics is ignored in this problem, the inertia matrix, $M(\Theta)$, is assumed to be the identity matrix, I.

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 Control

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

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

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

Q1 : Simulation

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**. You might need to load the data in **Matlab** as ASCII interpretable.

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 kp and kv 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 kv_3 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 $kv_3 < 0$ 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) Use "njmove" to change the mode to direct PD (non-dynamic) with gravity compensation and repeat step (c)-(h).

(i) 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.

(j) Compare the three results from "njhold", "njmove", and "jmove".

(k) Set kp and kv as follows:

kp:	400	400	400	400	400	400
kv:	40	?	40	40	40	40

(l) 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.

(m) Using the gains you used in (l) 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 (l).

(n) Repeat (l)-(m) with jmove instead of njmove.

(o) Write up a report showing the results. Please include calculations of ζ_n, ω_n, and m and disscussion about your results and your graphs.

(p) Please bring your results and calculation from the simulation to your lab period.

Q2 : Experiment

1. Repeat **Q1** using the actual robot platform.
2. Write up a report showing the results. Please include your results and graphs.

Velocity Saturation

1. We will begin with velocity saturation for joint space non-dynamic controllers. The joint velocity saturation limits can be accessed from *gv.dqmax*.
2. Implement **njgoto** control using the following control law:

$$\tau = -k_v(\dot{\theta} - \dot{\theta}_{eff}) + G \tag{1}$$

where $\dot{\theta}_{eff}$ is defined as:

$$\begin{aligned} \dot{\theta}_{eff} &= \dot{\theta}_{max},\ \dot{\theta}_{eff} > \dot{\theta}_{max} \\ \dot{\theta}_{eff} &= -\dot{\theta}_{max},\ \dot{\theta}_{eff} < -\dot{\theta}_{max} \\ \dot{\theta}_{eff} &= \frac{k_p}{k_v}(\theta_d - \theta),\ otherwise \end{aligned}$$

3. Add safety checking to prevent robot from hitting joint limits. Do not allow a goal position outside of the joint range.
4. When dynamics are incorporated, we just need to modify our controller slightly. Implement **jgoto** as follows:

$$\tau = A(-k_v(\dot{\theta} - \dot{\theta}_{eff})) + B + G \tag{2}$$

5. **Important Note**: Since we are dividing by k_v in $\dot{\theta}_{eff}$, please make sure that $k_v > 0$. A good way to address this is to add a small value to k_v so that when k_v is zero, we will be dividing with the small value instead of zero.

Task Space Control

Operational Space Non-Dynamic Controllers

1. Before we start, we will first look at some of the important data structures in the simulator.

   (a) **gv.x, gv.xd** : $\mathbb{R}^{7x1}$ vector that represents the pose of the end effector. The vector contains the position in cartesian coordinates and orientation in euler parameters, i.e. $[x\ y\ z\ w\ \nu_x\ \nu_y\ \nu_z]^T$

   (b) **gv.dx, gv.dxd** : $\mathbb{R}^{6x1}$ vector that represents the velocity of the end effector. The vector represents the instantaneous velocities in cartesian coordinates, i.e. $[v_x\ v_y\ v_z\ \omega_x\ \omega_y\ \omega_z]^T$

   (c) **gv.Einverse** : Notated as E^+ in the *PD Control in Operational Space* section. It is a $\mathbb{R}^{6x7}$ transformation matrix that converts the $\mathbb{R}^{7x1}$ pose error to a $\mathbb{R}^{6x1}$ vector that represents errors that are compatible with $[\omega_x\ \omega_y\ \omega_z]^T$.

2. For the non-dynamic task space controllers, we will ignore dynamics and only compensate with static torques, i.e. $\tau = J^T F + G$. F is the generalized force vector.

3. The first controller that we will implement is **nhold**. We will use a PD controller for F, i.e. $F = -k_p E^+(x - x_d) - k_v(\dot{x} - \dot{x}_d)$. When the controller is invoked, the configuration of the robot at that instant is recorded as x_d. The robot will return to x_d when perturbed.

4. Now, lets add velocity saturation and implement **ngoto**. For a spatial workspace, our saturation limits are defined in the vectors *gv.dxmax* and *gv.wmax*. Since we are not dealing with vectors, instead of comparing scalar values (like in **njgoto** and **jgoto**), we will use the length of the vector. The member function *length()* will be very useful here. The generalized force is given as

$$F = -k_v(\dot{x} - \dot{x}_{eff}) \tag{3}$$

5. **Important Note**: Since we are dividing by k_v in $\dot{x}_{eff}$, please make sure that $k_v > 0$.

Operational Space Dynamic Controllers

1. With dynamics, the operational space controller is given as $\tau = J^T(\Lambda F + \mu + p)$

2. Implement both **hold** and **goto**

3. **Important Note**: Since we are dividing by k_v in $\dot{x}_{eff}$, please make sure that $k_v > 0$.

Q3 : Simulation

1. Move the in joint space using **"jgoto"** to $(0, 0, 0, 0, 30, 0)$
2. Collect data from position (.4, .2, .7, .8, 0, .6, 0) to position (.6, -.2, 0.3, .5, .5, .5, .5), using the **ngoto** and **goto** controllers. Examine the trajectory by plotting the end-effector coordinates (x,y,z), linear velocity, linear acceleration, and angular velocity. Compare the results.

Q4 : Experiment

1. Repeat **Q3** using the actual robot platform.
2. Write up a report showing the results. Please include your results and graphs.