University of Washington

Computer Science and Engineering

Winter 2007

CSE 490 I: Design in Neurobotics

Lab 2 Date: 1/16/2007 or 1/18/2007

Write-up Due: 1/23/2007 or 1/25/2007

Submit electronically

Goals:

1. To learn feedback control with a one degree-of-freedom robot

2. To understand the robot behavioral difference with different controllers and gains

Assignment:

In this lab, you will use one of the joints of the Lynxmotion robot --- the shoulder joint of the robot arm.

Last week, we used Lynx::getPos to read the positions of arm joints during the movement. This is the target (desired) position designed by the control chip of the robot at that time stamp. Some of you may noticed that if you set merely the start position and the end position for the robot to pick up the screw, the track points you recorded are almost located at two locations (the reading also depends on your sampling frequency), which doesn't reflect the actual course of the movement.

In this lab, we add a sensor to the joint so that we can capture the "real" positions. The new method of Class Lynx, getInput, is used to read out analog signals from the sensor. The plot (fig.1) shows the difference of two readings. The robot is sent a command to set the shoulder at pi/2 degree (straight up). The blue line is the returns of getPos() calls, which remains constant through the process; the red curve is the returns of getInput() calls, the real trace of the movement.

The controller’s goal is to have the actual position (red) to match up with the desired position (blue).

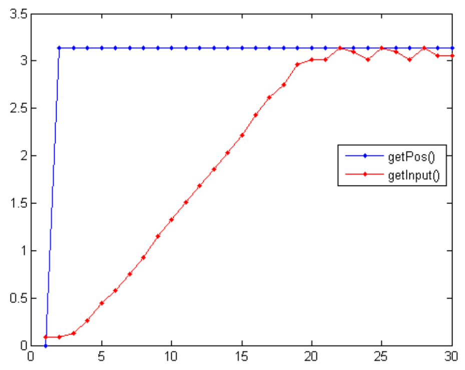

Figure 1. Blue line is the desired position (getpos()) and the redline is the actual position (getInput()).

Important Information (read this before programming, AND read this after programming but before executing the program):

1. Download the updated arm interface code. We have one sensor on the robot, so only one value is meaningful in the return array. The syntax is as:

Lynx::LynxInput inp = l.getInput(); // l is an instance of Lynx

printf("The curreint position is : %7.5f\n", inp.analog_v[0]); // we only need port A

2. The position read by getInput() is the analog signal, which is NOT equal to the angle of the joint. Rather, it is a linear mapping of the angle. So, before you record any reading, you need to find out the linear mapping parameters between joint angles and analog signals. We suggest you to set the joint to 0, get a reading; then set it to pi, get another reading. Thus, you can get the analog signal values corresponding to the maximum and minimum angles, and learn the linear mapping equation.

3. Set the speed a negative value, which means full speed of the movement.

4. Record at 100 Hz. This means that in the following equations, the time duration between two samples, T, is 0.01 seconds. If you decide to change the recording frequency, flag the TA first before doing it (doing so could produce erratic behavior that could harm the robot).

5. Your task is to command the robot joint to move instantaneously to 90 degrees (straight up). The initial position of the joint should be at 0 degree, and the command to move to 90 degrees should be sent at **1** second.

6. IMPORTANT!!! Change the controller gains with a small incremental step, because it is possible to make the robot to go out of control and break it. DO NOT USE GAINS OUTSIDE OF THE RANGE SUGGESTED. Always start with the smallest suggested value.

7. You can produce the same output files for both partners in this lab.

8. You will be asked to answer questions after the lab about what happened with the change of different gain values within one type of controller and among different controllers. Make notes during the lab about what you observed.

Instructions:

Write a C/C++ program for each of the following controllers. Record the actual position over time for 3 seconds, starting at second 1 (when the command is sent out). Save the file into a text file where the first column is the actual position and the second column is the time stamp.

1. Write an **open loop controller** (no correction of the movement based on the actual position) with no gain (Kopen = 1). Save the output position and time to LastnameOpen.txt.

2. Write a **P controller** which closes the feedback loop. This means that your control signal at time step *m* is:

   $f(m) = K_p error(m)$, where $error(m) = desired(m) - actual(m)$.

   Try *Kp* of 0.2,0.6, and 0.8. ***Kp* should never exceed 1.** Command the same desired output as before and record the actual position and time to LastnameP05.txt, LastnameP10.txt, and LastnameP20.txt.

3. Write a **PI Controller.** In addition to Kp, we will add an integral term, Ki. To do so, your control signal at time step *m* is:

   $f(m) = K_p error(m) + KiT\sum_{i=0}^{m} e(i)$, where T is the sampling period.

   For each Kp you used in 2, try different values of Ki. Try Ki of 10, 20, and 30. *Ki* should never exceed 45, Pick one scenario that demonstrate the effect of the integral term the best and record them into a file named as LastnamePI.txt.

4. Write a **PD Controller.** Now in addition to *Kp*, we will add an derivative term, *Kd* rather than *Ki*. To do so, your control signal at time step *m* is:

   $$f(m) = K_p error(m) + K_d \frac{e(m) - e(m-1)}{T}.$$

   Try *Kd* of 0.001, 0.003, and 0.005. DO NOT GO HIGHER THAN 0.008. Do this for different *Kp*'s you used. Pick one scenario that demonstrate the effect of the derivative term the best and record them into a file named as LastnamePD.txt.

5. Write a **PID Controller.** By now you must know the drill. We will design a PID controller with its control signal at time step *m* is:

   $$f(m) = K_p error(m) + KiT\sum_{i=0}^{m} e(i) + K_d \frac{e(m) - e(m-1)}{T}.$$

   Play around with different gains (within the range of values you were given above). Identify the best combination of gains, and record this trial (actual position and time) into a file LastnamePID.txt.

6. **Steady Noise:** Add a small noise to the command every time you send it to the robot. Then rerun on all of your best P, PI, PD and PID controllers. Record output in LastnameNoiseP.txt (and so forth).

Questions:

Please submit individually electronically. However, you CAN work together with your partner and submit the same solutions as your partner's. Write your partner's name on top of your assignment.

1. Report the best gains selected for P, PI, PD and PID controllers.
2. Using MATLAB, plot LastnameP05, LastnameP10, LastnameP20 on the same graph (as usual, different color lines, label lines/axes, etc), and comment about the difference between them.
3. Using MATLAB, plot Open, P, PI, PD, and PID output plots on the same graph.
4. Comment on the effect of (Using 2-3 sentences for each of these categories):
    1. Adding closed loop with Kp (difference between Open and P)
    2. Adding Ki (difference between P and PI)
    3. Adding Kd (difference between P and PD)
    4. Adding all of them together (difference between P/PI/PD vs. PID)
    5. Increased Kp
    6. Increased Ki
    7. Increased Kd
5. Using MATLAB, plot P, PI, PD and PID output with noise on the same graph.
6. Comment on the controllers when the system has noise added.
    1. Explain what happened to the output (2-3 sentences) when the noise was added (if different things happened to different controllers, separately explain them.
    2. Which controller(s) performed the best?
    3. What is the measurement of "best" for you?
7. Rank in the order of controllers that performed the best (overall, with and without noise) and explain your order (in 2-3 sentences).