Lab 1. Data Logging, Visualization, and Motor Identification.

Due date: **Each lab group must submit a *short but well-written* "Lab 2 Report" which combines results from both Labs 1 and 2 by 5pm Mon., October 29 (in dropbox).** Only one lab report per lab group is required.

Overview

All 6 labs throughout the term use Lego NXT robotics and MATLAB. The goals of this lab are (1) to learn how to build and run models designed in MATLAB's simulink on the NXT block, (2) to learn how to log data, so it can be processed in MATLAB, (3) to process logged data from the encoders to visualize the end effector position and estimate motor velocities, and (4) to estimate model parameters for a Lego motor.

Tasks

Follow instructions from your TA to complete the numbered tasks below. Tasks 1-3 MUST be done in lab, as they involve data collection and require that you have access to the equipment. Use any remaining time to start completing the remaining tasks, which can also be done outside of the lab period. Your lab report will primarily contain several required plots; plots 1-4 are described in this hand-out. A report for labs 1 and 2, combined, is due one week after Lab 2.

1. Open lab1.mdl and follow TA instructions to build the model and then download code onto the Lego NXT block. This initial model will do nothing but collect motor encoder positions. Also measure lengths L_1 and L_2 lengths of the arm and record for later use.

2. Set the two-link arm in a fully extended position with, both θ_1 and θ_2 near zero. Enable Bluetooth data logging to MATLAB and run the program on the NXT brick. Move the two links of the robot arm to "draw" some path with the end effector, to collect the encoder values of the two joints over time. Although you are not running the motors here, the motor encoders can still be used as passive sensors of joint positions. However, the encoders always initialize at a value of zero, regardless of the arm configuration, so they each output angle with respect to the initial angle when you began to run the program.

   SAVE YOUR DATA FROM PART 2. Also, plot the raw data of encoder positions while in lab, just to be sure it was actually recorded (and not all blank for example).

3. Now, modify your simulink model to send a step response command to the third motor, which is NOT a part of the arm. Set the power level, P, for the step to 100 (its maximum value). Build code, download to the brick, and log data for the resulting motor step response. (Be sure the motor can spin freely when you do this, of course!)

   Repeat to collect step response data for P=80, P=60, and P=40. Also test for P=-100 and P=-60, to check if these responses are symmetric to the positive input, as expected.

   SAVE EACH DATA SET FROM PART 3. Also, plot all data, to be sure it was recorded.

You are now done with the required tasks that must be completed during lab. Make sure each lab group member gets the raw data (e.g., through email or USB key) and use the rest of your time in lab to process the data to produce plots and motor parameter values needed for your Lab 2 report.

4. In a single figure, use two subplots to plot the data described below. (Type "help subplot" in MATLAB to learn how to use subplots. Your use of subplots will help the TAs grade reports and will avoid wasting paper, too.)

   Plot 1: In one subplot of figure 1, plot each of the two encoder angles versus time. In the other one, plot the end effector path, (x_e, y_e), using your code from the prelab to do so, along with values of L_1 and L_2 that you recorded earlier. Use 1:1 scaling for subplot 2.

5. Next, estimate the velocities of each joint, using the simple difference equation below, where *n* is an index into a vector of values in MATLAB (for a discrete step in time).

   $$\dot{\theta}_1(n) = (\theta_1(n) - \theta_1(n-1))/(t(n) - t(n-1))$$

   This just says the velocity estimate is the difference in angle between two time steps divided by the difference in time. Put each plot of velocity ($\dot{\theta}_1$ and $\dot{\theta}_2$) versus time in a separate subplot. You should notice that this approach to estimate velocity gives a very "jagged looking" output that tends to jump to discrete values.

   Joint velocity estimates will be needed in order to implement PD (proportional plus derivative) control in future labs. A "noisy" estimate, such as one you have done here, can result in a less stable controller, with a lot of high-frequency noise. One way to produce a smoother estimate is to use a low-pass filter. There is a trade-off, however, because a low-pass filter will also add some time delay to the estimate. The slower the low-pass time constant, the smoother the estimate but the larger the time delay, and a large enough time delay will eventually cause a controller to go unstable.

6. Now, estimate velocity by using a discrete-time (DT) filter approximation of the continuous-time (CT) transfer function $s/(\tau s+1)$. This simply combines the "d/dt" operator, s, with a first-order transfer function with a time constant tau and a steady-state gain of 1. First, create a CT transfer function (TF) using the tf command in MATLAB:

```
tau=0.05  % test BOTH 0.05 AND 0.10 seconds
cttf=tf([1 0],[tau 1]) % numer: "s"; denom: "tau*s+1".
```

   Next, create the DT transfer fn by using the command c2d ("continuous to discrete"), specifying the discrete time step and method of approximating the CT TF as a DT TF:

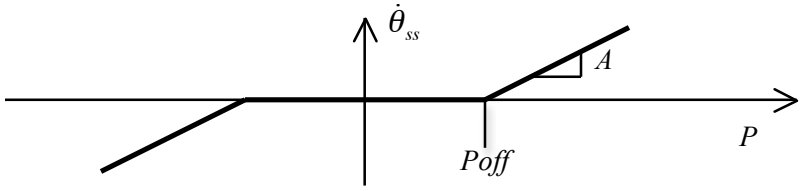
```
dt=.01  % time step for the data, in seconds.
dttf=c2d(cttf,dt,'tustin')  % create the DT low-pass filter
dtheta=filter(dttf.num{:},dttf.den{:},theta); % filter data
```

   REPEAT this for tau=0.10 (in addition to tau=0.05). In later labs, we will estimate velocity with a DTTF block within the simulink model, where the value of "dt" will be smaller. Here, dt matches the rate at which data were sampled during data logging.

   Plot 2: Put your answers to parts 5 and 6 in a single MATLAB figure. One subplot should include all velocity estimates for θ_1 and the other should have estimates for only θ_2. Be sure to label each line in the plot (for example, using the "legend" command).

7. **Plot 3:** For figure 3, use one subplot to plot all step responses from part 3 on a single set of axes. Use a second subplot to again plot only the data for P=100. Zoom in on the initial part of the step in the second subplot, and estimate graphically what the time constant for the motor response is. (In other words, assume the motor model is a first-order system, from power level input, P, to output motor velocity, $\dot{\theta}_m$. Methods for estimation of tau were discussed in Lecture 4.)

8. **Plot 4:** For figure 4, plot the final, steady state velocity of the motor as a function of input P. There should be a "dead zone" where the velocity is zero across some range of P. Then there should be a nearly constant slope for values greater in magnitude (either negative or positive). You only have a few data points, so your plot will be approximate. Provide your best estimate of the true shape of the plot, based on the plots you have. You can do this plot either in MATLAB or (preferably) as a small sketch (clearly labeled) by hand. Your plot should look somewhat like what is below, but with values for A and P_{off}:



9. **Table 1:** From parts 7 and 8, create a table summarizing your estimates of the following three parameters for a simple, nonlinear motor model: A, P_{off}, and τ. This corresponds to a model that can be represented (via somewhat sloppy notation) as:

$$\dot{\theta}(s) = \frac{A}{\tau s + 1}(P(s) - P_{off}), \text{ so long as both } P > P_{off} \text{ and } \dot{\theta} > 0$$

Intuitively, this is as if the commanded power level, P, is decreased in magnitude by P_{off} if the motor has a non-zero value. (And if it is currently at rest, it stays put until P exceeds P_{off} in magnitude.)

To cover all situations of power level and motor velocity with more care, the nonlinear dynamics can be written out as the following set of differential equations, depending on the current state:

$$\ddot{\theta} = \begin{cases} \frac{1}{\tau}(A(P - P_{off}) - \dot{\theta}) & ,if\ (\dot{\theta} > 0)\ or\ ((\dot{\theta} = 0)\ and\ (P > P_{off})) \\ 0 & ,if\ (\dot{\theta} = 0)\ and\ (|P| < P_{off}) \\ \frac{1}{\tau}(A(P + P_{off}) - \dot{\theta}), & if\ (\dot{\theta} < 0)\ or\ ((\dot{\theta} = 0)\ and\ (P < -P_{off})) \end{cases}$$

Summary: Your Lab 2 report will include 5 items from this lab: Plots 1-4 and Table 1.