ME 446 Laboratory #3
Inverse Dynamics Joint Control

Report is due at the beginning of your lab time the week of April 10th. One report per group. Lab sessions will be held the weeks of March 13th, March 27th, and the week of April 3rd.

Objectives

- Add Friction Compensation for all three joints and investigate how well it works to cancel friction effects in the arm.
- Design and implement an inverse dynamics control algorithm and compare its performance to the PD plus feedforward control designed in lab 2.
- Investigate how well the parameters of the CRS robot have been identified and see if better values for the parameters of the system can be found.

Part 1: Implement Friction Compensation.

1.1 Identified Friction Curves.

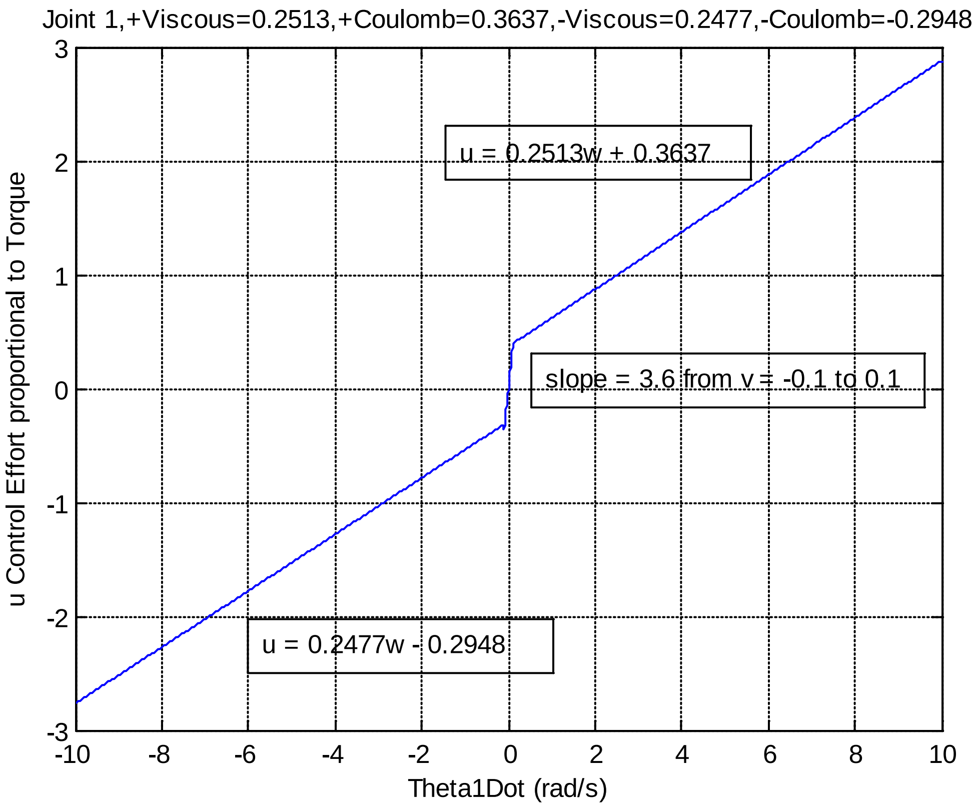
Below are experimental friction plots that were identified on one CRS robot arm. Constant velocity and control effort (proportional to torque) was recorded and these plots show the best fit lines that were fit to the collected data. Use these straight line equations to implement friction compensation for each joint of your robot arm. Make sure to not hard code the friction coefficients because these values will need to be tuned for your robot arm.

Once you have friction compensation implemented, adjust each joint's friction coefficients until you feel you have the correct coefficients for your robot arm.

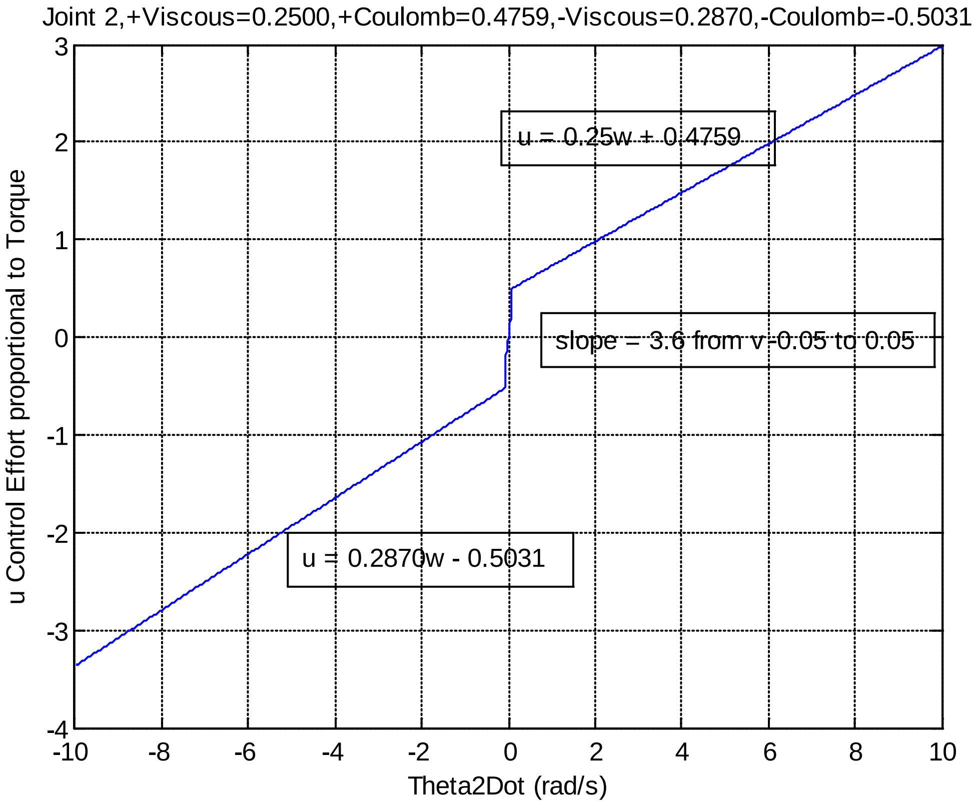
Pseudocode:

```
if (joint_velocity > minimum_velocity) {
        u_fric = Viscous_positive*joint_velocity + Coulomb_positive ;
} else if (joint_velocity < -minimum_velocity) {
        u_fric = Viscous_negative*joint_velocity + Coulomb_negative;
} else {
        u_fric = slope_between_minimums*joint_velocity;
}
```

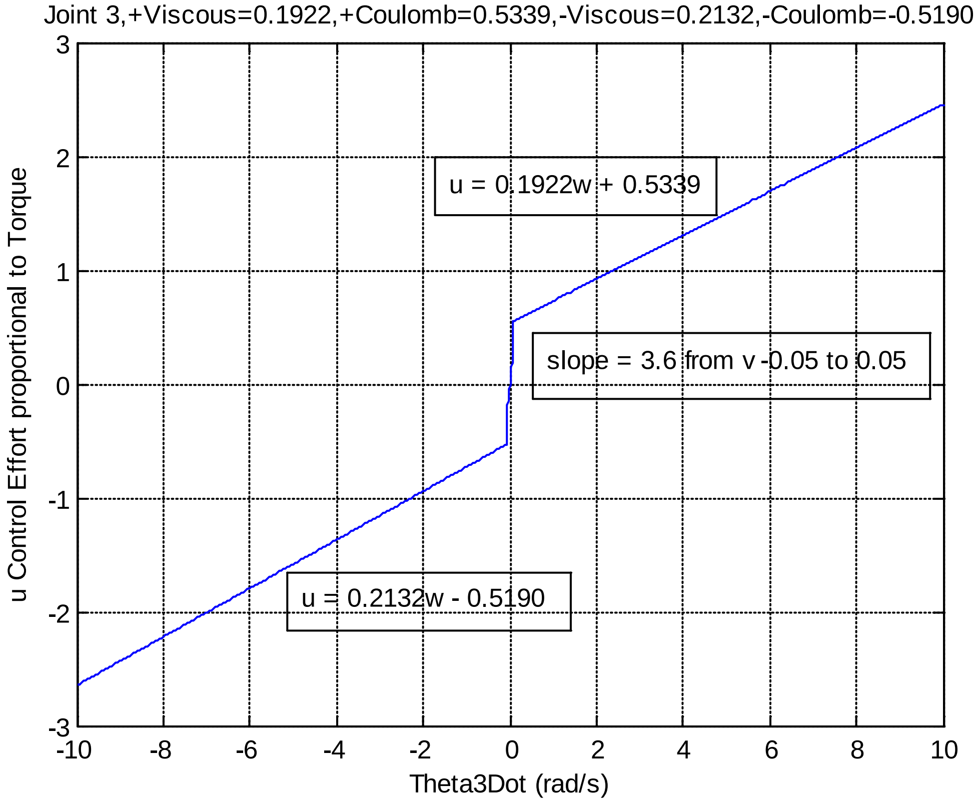
Then after full control effort has been calculated, add u_fric to your calculated control effort. When testing only friction compensation simply set control effort only to u_fric.



Joint 1,+Viscous=0.2513,+Coulomb=0.3637,-Viscous=0.2477,-Coulomb=-0.2948



Joint 2,+Viscous=0.2500,+Coulomb=0.4759,-Viscous=0.2870,-Coulomb=-0.5031



Part 2: Implement the Inverse Dynamics Control Law on Joints Two and Three.

2.1 Inverse Dynamics

This document does not go into detail on Inverse Dynamics control. Please see Section 8.2 of Robot Modeling and Control or section 11.3 of "Robot Dynamics and Control" second edition.

Modify your feed forward control (but make sure to keep a backup) from lab 2 to implement Inverse Dynamics control for joint 2 and joint 3. Start out by using the given parameters from lab 2, **p** = [0.0300 0.0128 0.0076 0.0753 0.0298]. Recall from lab 2 that **p** is equal to:

$$\begin{aligned} p_1 &= m_1 l_{c1}^2 + m_2 l_1^2 + I_1 \\ p_2 &= m_2 l_{c2}^2 + I_2 \\ p_3 &= m_2 l_1 l_{c2} \\ p_4 &= m_1 l_{c1} + m_2 l_1 \\ p_5 &= m_2 l_{c2} \end{aligned}$$

For joint one keep the same feed forward control from lab 2 tracking the cubic polynomial.

So for joints two and three implement the inverse dynamics control

$$\tau = D(\theta)a_\theta + C(\theta,\dot{\theta})\dot{\theta} + g(\theta)$$

to track the cubic polynomial reference trajectory from part 4 of lab 2. This control equation is called the inner loop because it cancels the nonlinearities leaving the equation $a_\theta = \ddot{\theta}$. (This of course assumes we know the systems perfectly). Then an outer loop control is needed to control this linear set of equations. For the outer loop control use PD plus feed forward control and the same cubic trajectory from lab 2

$$a_{\theta_2} = \ddot{\theta}_2^d + K_{P2} * \left(\theta_2^d - \theta_2\right) + K_{D2} * \left(\dot{\theta}_2^d - \dot{\theta}_2\right)$$

$$a_{\theta_3} = \ddot{\theta}_3^d + K_{P3} * \left(\theta_3^d - \theta_3\right) + K_{D3} * (\dot{\theta}_3^d - \dot{\theta}_3).$$

Note that your Kp and Kd gains will be different from the PD plus feed forward values from lab 2.

So given these equations, the flow of your code each sample period should be:

1) Calculate the desired trajectory
2) Given measured thetas, calculate actual states, error, error_dot, theta_dot.
3) Calculate the outer loop control to come up values for a_{θ_2} and a_{θ_3}.
4) Calculate the inner loop control to find control effort to apply to joint 2 and 3.
5) Calculate Lab 2 feed forward control for joint 1 to find control effort to apply.
6) Calculate friction compensation control effort given the velocities of joint 1, 2 and 3.
7) Add the friction compensation to the control efforts calculated in 3 and 4 above.
8) Write control efforts to PWM outputs to drive each joint.

NOTE: When performing the inner loop calculations there are a number of sin() and cos() calls. Try to minimize the number of calls to sin() and cos() by creating a float variable say "float sintheta2". Then each time into your lab() function set sintheta2 = sin(theta2). Then use this variable every place in your calculations where sin(theta2) is needed. This way the sin(theta2) is only calculated once per sample period.

Tune your Kp and Kd gains to minimize error as the joints following the trajectory.

After you have tuned your Kp and Kd gains for the two second trajectory you designed in Lab 2, create a faster moving trajectory to allow you to better tune your Kp and Kd gains and also to see the advantages of the Inverse Dynamics control law. Have the trajectory start, t=0.0, at 0.25 radians. Have it follow a cubic path for 0.33 seconds to 0.75 radians. Then have the joints stay at 0.75 radians until 4 seconds have elapsed. From 4 seconds to 4.33 seconds follow a cubic trajectory back to 0.25. Have time repeat every 8 seconds so that there is a pause between each cubic path to the new joint angle.

In addition to the cubic trajectory, your instructor will lecture on another way to produce a trajectory. It uses a discrete approximation of a transfer function to filter a step input to produce a smooth command similar to the cubic trajectory. The M-file to produce this C code is found here, http://coecsl.ece.illinois.edu/me446/filtersteptoC.m , and it should also be in your repository's Matlab folder.

With this faster trajectory (which ever one you choose) tune your Kp and Kd gains to minimize error. Produce trajectory response plots along with error plots for your report.

Part 3: Compare Inverse Dynamics Controller to PD controller.

Part 3.1 Compare PD plus feedforward to Inverse Dynamics

Below you will run a few comparisons between your PD control from Lab 2 and the inverse dynamics control you just designed. Hopefully these comparisons will demonstrate that the inverse dynamics control does a slightly better job controlling the linkage. Each step is listed.

1. Implement your PD control from lab 2, but make sure to add the friction compensation you implemented with your inverse dynamics controller. For this comparison remove the integral control. Have the PD control follow them same quicker trajectory you just designed. With the PD implemented, tune its Kp and Kd gains to achieve similar error responses as your inverse dynamics control.
2. Now that you have both your inverse dynamics controller with friction compensation and the PD controller with friction compensation working quite similar, we are going to add a known mass at the end effector of the robot. Then to compare, you will leave your PD controller gains unchanged and check the error responses with this mass added. For the inverse dynamics control, you will modify the parameters of the system that take into account this extra mass. With the new parameters applied, run your inverse dynamics controller and see if there is a noticeable improvement over the PD controller. Compare both the error peaks and the steady state error. The new parameters of the system with mass added can be found in the M-file http://coecsl.ece.illinois.edu/me446/ID_CRS_withDisk.m. Notice how the parallel access theorem is used to add in the additional mass.
3. By comparing these two controllers can you say anything about the parameters we are using? Does Inverse Dynamics perform better? For example has gravity effects been improved? If not can any of the parameters be adjusted to improve the Inverse Dynamics control. I do not want you spending a huge amount of time on this but enough time so that you can give some observations in your report.

Report: (Minimal Requirements)

1. Include the final version of your C code.
2. Include any Matlab M-files you created (if any)
3. In your own words, explain the inverse dynamics control algorithm.
4. Answer the questions found in the lab.
5. Trajectory response plots and error plots.
6. You observations about the parameters used.