Homework 2 (<u>Problem 1 due Monday, Oct. 22</u>, at 4pm; The rest due <u>Oct. 26</u> at 5pm.) This homework is lengthy due to its "tutorial" nature. Please do not be intimidated.

1) Lab 2: End Effector Trajectory Planning. To prepare for Lab 2, use MATLAB to plan both end effector (xe, ye) and joint angle (theta1 and theta2) trajectories over time to toss a ball. You should be able to use your functions from Lab 1, ang2xy.m and/or xy2ang.m, to help with this. Use values of L1=0.15 (m) and L2=0.095 (m) for the two arm link lengths. The supplemental document "HW2_pongtoss_example_data.pdf" give an example of a human tossing a ball. Turn in four plots, similar to those below, and take your code with you to Lab 2:

a) Task space trajectories for the ball and for the end effector, similar to Figure 1a, including the ballistic (free falling) trajectory of the ball, after it leaves the arm.
b) Snapshots of arm from t=-.06 seconds to t=+0.10 seconds, to move ball as desired.
c) A plot of joint positions (in deg).
d) Joint velocities (in deg/s).

To get the plots below, begin with the point where the ball would leave the arm, defining this (arbitrarily) as x=0, y=0. Then, go forward in time, assuming the ball would go with zero x acceleration (constant velocity) and -9.8 m/s^2 y acceleration (under gravity). The arm acceleration must be LESS THAN the acceleration of gravity for t>t_launch. Below, I used -20 m/s^2 acceleration in y and a much lower magnitude deceleration in x. To plan backward in time, one can also use a set of constant accelerations in x and y. (However, use different acceleration values for t<t_launch, since you want the end effector in contact here!)

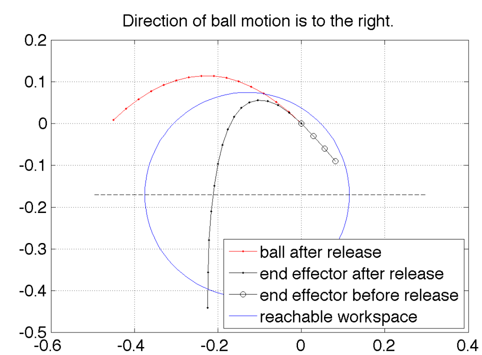

Figure 1a. Ball, end effector trajectories, workspace.

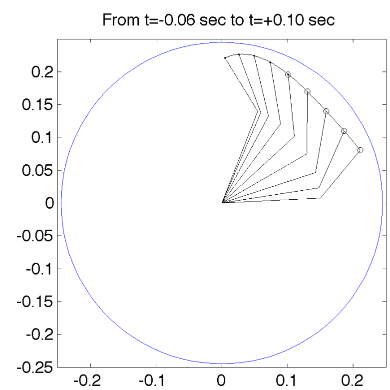

Figure 1b. End effector trajectory and joint positions.

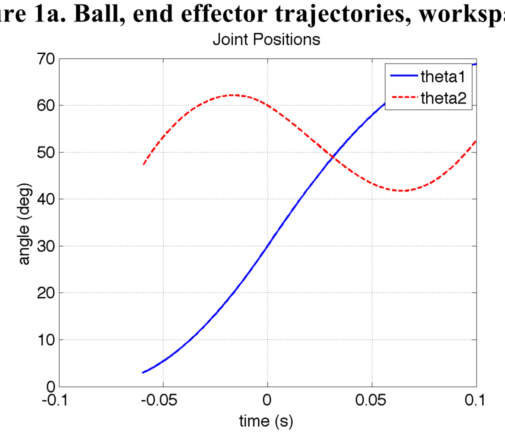

Figure 1c. Joint velocites over time.

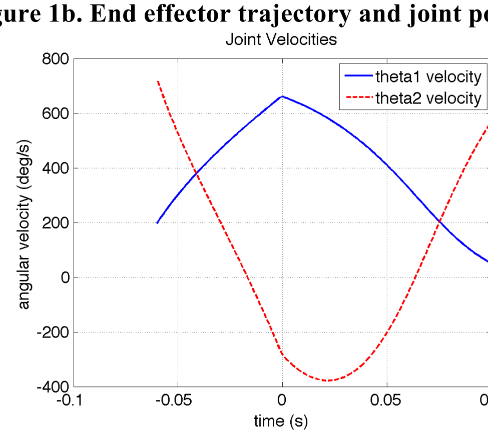

Figure 1c. Joint velocites over time.

**** Bring your MATLAB code for Problem 1 to Lab 2! EXTRA CREDIT: Plan a full trajectory where the velocities of theta1 and theta2 start at zero. ****

2) PID controller tuning via Ziegler-Nichols guidelines. In Lecture 6, we reviewed two methods for obtaining an initial set of gains for a proportional-integral-derivation (PID) controller, based only on a set response for the system. The resulting closed-loop system typically requires further fine-tuning; however, there are often advantages in finding approximate control gains rapidly. In particular, this method can be useful when a model for the plant does not exist, and/or when further system ID (to obtain a frequency response plot) requires the implementation of a "good" controller.

Although this problem looks long, it should be relatively straight-forward to complete rapidly. Parts a and b each use the "first method" of Ziegler and Nichols, which they proposed in a paper back in 1942. (Wow, that's old!) Part c uses a "second method". In all cases, we will assume the following form for the PID controller:

$$C(s) = K\left(1 + \frac{1}{\tau_i s} + \tau_d s\right) = K + \left(\frac{K}{\tau_i}\right)\cdot\frac{1}{s} + (K\tau_d)s$$

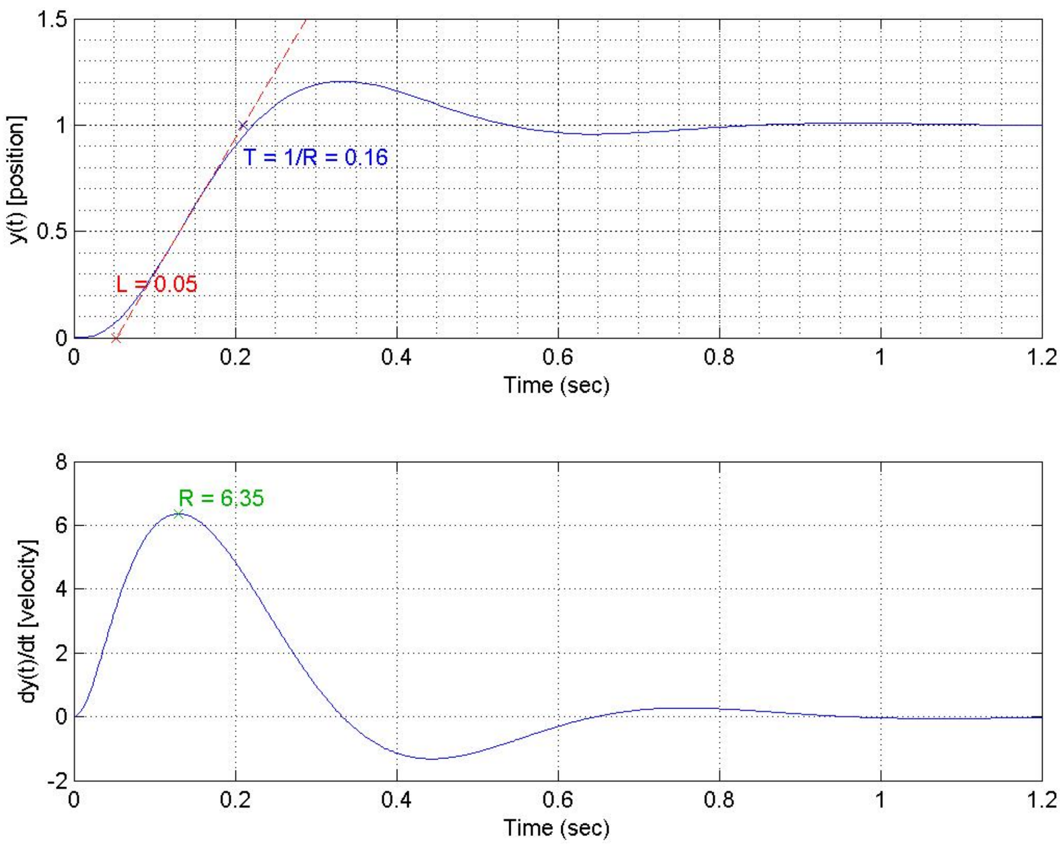

Figure 2a – Open-loop system step response. Position (top) and velocity (bottom) are both shown.

Ziegler-Nichols first method. This method simply requires estimating two values from a unit step response for the open-loop system. To do so, we first draw a tangent line at the inflection point in the S-shaped unit step response for the open-loop system, as shown by the dashed line in Figure 2a. Note that the inflection point is the location with the peak slope (i.e., highest velocity).

Next, estimate the following time parameters: 1) The location, "*L*", in which a tangent line drawn at the inflection point intersects the time axis. 2) The time between the intersection of the tangent line with y=0 and with y=1, which we will call "*T*". Note also that $T=1/R$ (with units adjusted appropriately), where *R* is the peak velocity (i.e., the velocity at the inflection point). Once *L* and *T* have been estimated (this has been done for you in Figure 2a…), the suggested gains for the PID controller are then:

$$K = 1.2\frac{T}{L} = \frac{1.2}{RL} \quad , \qquad \tau_i = 2L \qquad , \qquad \tau_d = \frac{L}{2}$$

a) For part a, *L* and *T* are clearly labeled in Figure 2a. For simulation purposes, assume that the open loop plant for Figure 2a is given by the following transfer function:

$$G_a(s) = \frac{1e4(s^2 + 200s + 5e4)}{(s+500)(s^2 + 10s + 125)(s^2 + 160s + 8000)}$$

i) Simulate and submit a plot of the step response of the closed-loop system, using the nominal PID gains. What is the percent overshoot, approximately?

ii) Now, tweak the PID gain(s) to reduce the overshoot to between 5% and 10%. Plot the resulting response, and clearly state the gains you used. (Hint: Hopefully, you only need to change K_p, leaving the zeros in C(s) unchanged…)

b) Repeat all of part **a)** for the system shown in Figure 2b. Begin by estimating *T* and *L*.

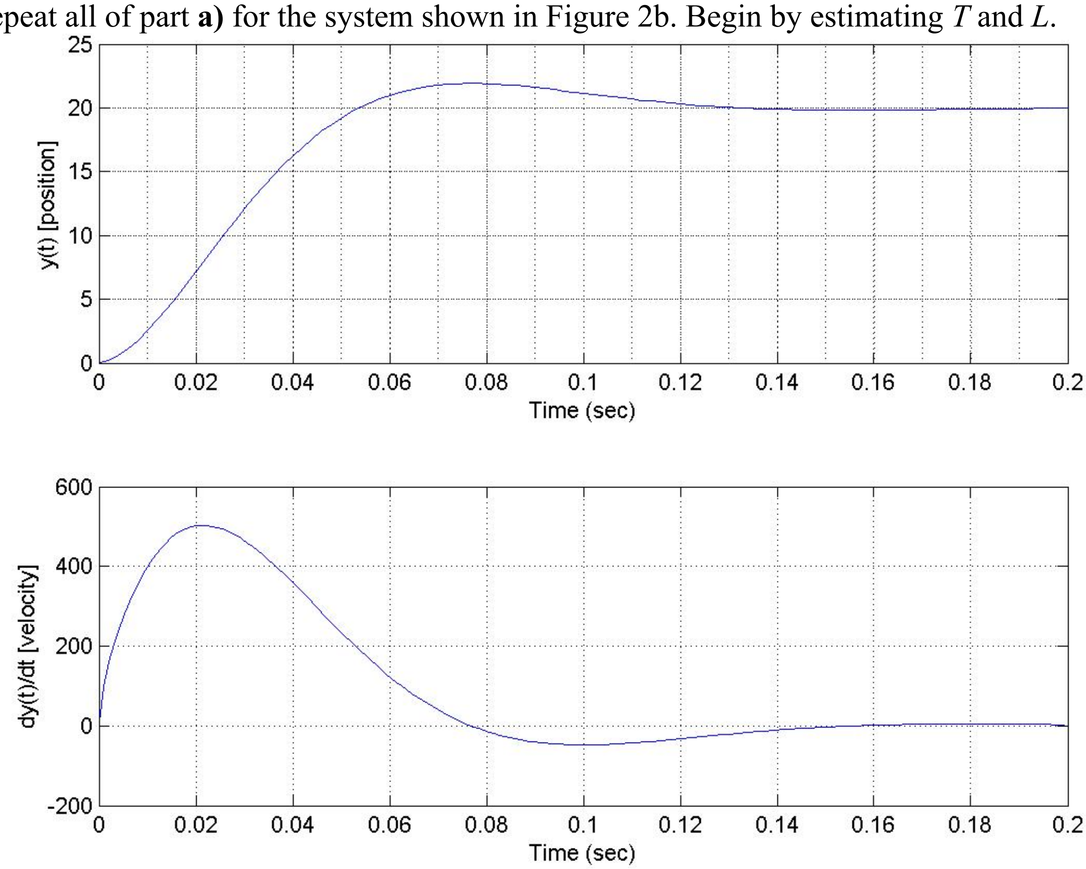

Figure 2b - Open-loop system step response. Position (top) and velocity (bottom) are both shown.

To tune the controller for part b, assume the open-loop plant is:

$$G_b(s) = \frac{124{,}000(s+400)}{(s+1000)(s^2+60s+2500)}$$

c) Ziegler-Nichols second method, or "ultimate cycle" method. Sometimes, the open-loop system will not have a steady-state response to a unit step input. (For example, the motors in the Lego systems will spin continuously, so the position output approaches a ramp over time.) In such situations, the system can first be controlled with a simple, proportional (P) gain that has been increased to the point where the system is almost "marginally stable". In other words, the gain is increased to a value K_{crit} for which continuous oscillations in the output are observed.

Once the critical gain, K_{crit}, for continuous (marginally stable) oscillations is found, we then define the period of time for a single oscillation as P_{crit}. The PID gains are then set initially to the values given below:

$$K = 0.6K_{crit} \quad , \qquad \tau_i = \frac{P_{crit}}{2} \quad , \qquad \tau_d = \frac{P_{crit}}{8}$$

i) Use the ultimate cycle method to find initial PID gains for the system shown in Figure 2c. This figures shows the response of the system when a proportional controller is used with a gain of 97; that is, we may assume here that $K_{crit} \approx 97$.

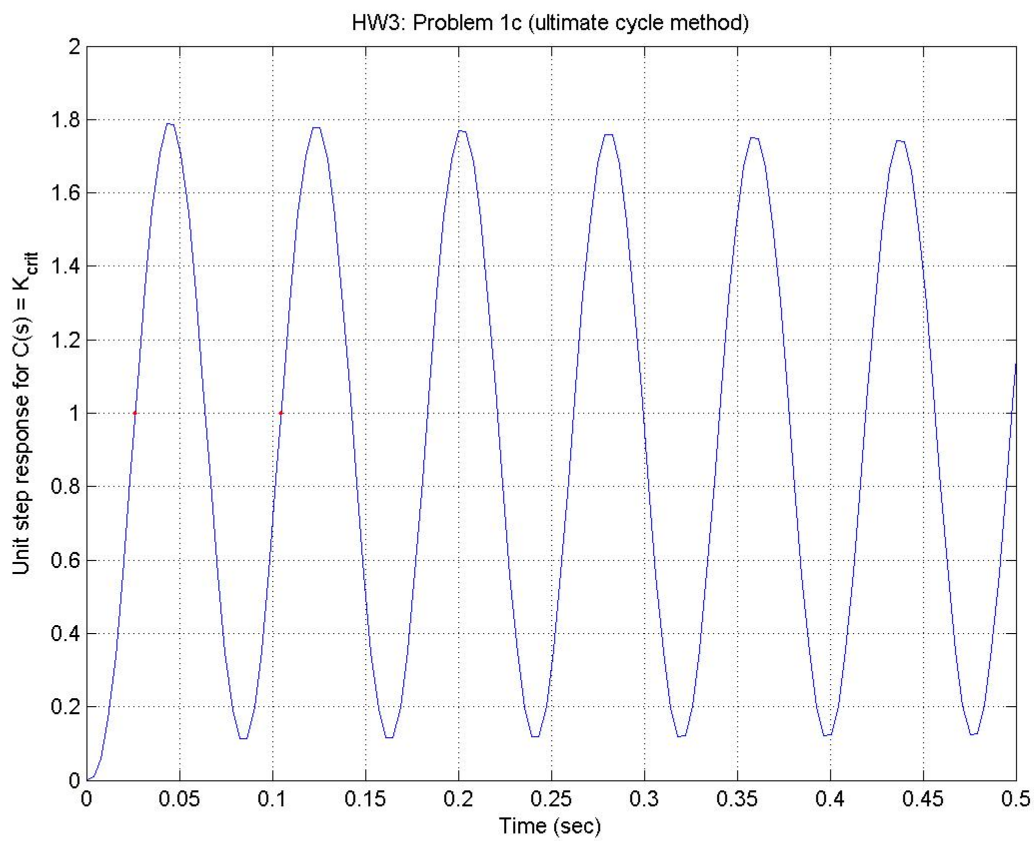

Figure 2c – Data for ultimate cycle method. Shown above is the closed-loop response when K=97.

ii) As before, tune the initial PID controller and include a step response and set of gains as your solution. However, in this case, aim for 25% overshot. When simulating the closed-loop response, the actual plant is:

$$G_3(s) = \frac{10{,}000(s+0.01)(s+0.5)}{s(s+2)(s+8)(s+50)(s+100)}$$

3) Feedforward. Recall (from Lecture 5) that feedforward control can sometimes be used to (theoretically) result in zero steady state error (if there are no external disturbances to the system). Also recall that the required transfer function for the feedforward portion, F(s), is the inverse of the plant, G(s), and that this only works if the resulting F(s) is stable.

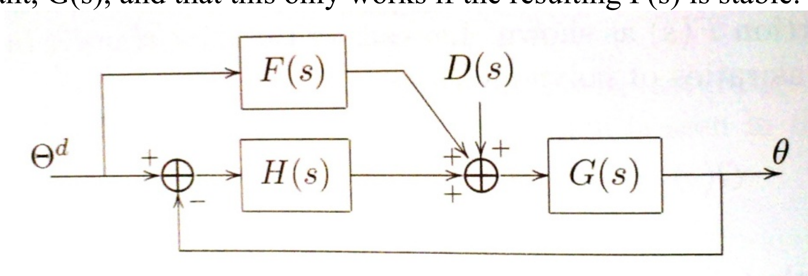

Figure 3a – Feedforward configuration, from Spong (Figure 6.17, page 218).

For all parts below, assume the open-loop plant is: $G(s) = \frac{1}{Js^2 + Bs} = \frac{1}{2s^2 + 10s}$

a) First, just assume F(s)=0. Create a PD feedback controller, *H(s)*, such that the closed-loop response has a natural frequency, ω_n, of 10 rad/s and a damping ratio, ζ, of 0.6. Generate a step response of the closed-loop response, using MATLAB.

b) Now, also include a feedforward controller, $F(s) = 1/G(s)$, in addition to *H(s)* from part a. Generate a step response of the closed-loop response, using MATLAB.

c) Using the same transfer functions for H(s) and F(s) from parts a and b, now assume that the actual plant changes. Specifically, on a single set of axes, generate step responses for each of the following 4 scenarios, each of which involves being off by a factor of 2 in either J or B:

i) J=1, B=10
ii) J=4, B=10
iii) J=2, B=5
iv) J=2, B=20

(Be sure to label which plot is which!)

4) Decay envelopes for Coulomb friction vs viscous damping. Viscous damping is a linear loss term which can be modeled with ease in a transfer function and which results in exponential decay envelopes. Coulomb friction is a nonlinear loss effect which cannot be incorporated precisely into a transfer function and which results in a decay envelope with a constant slope (i.e., the decay envelope has straight sides). There are a lot of hints throughout this problem, which may be helpful. **The questions you must actually answer are written in bold.**

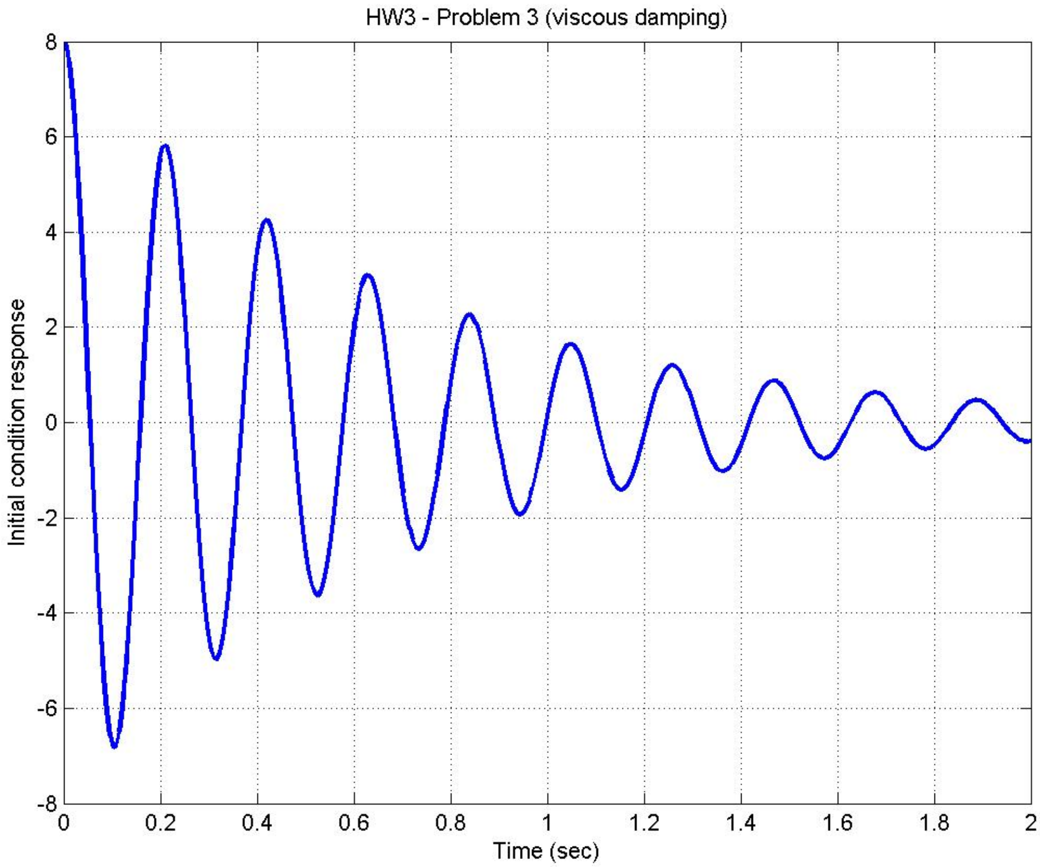

Figure 4a – Step response for a system with viscous damping.

a) Figure 4a shows the step response of a spring-mass system where m=1kg and k=900 N/m. The EOM is: $m\ddot{x} + b\dot{x} + kx = 0$, where losses are due to linear, viscous damping.

i) From the step response, estimate the linear damping, b.

Hint, simulate with MATLAB to check your answer! Once you define variables `m`, `b`, and `k`, you can see a step response by typing (for example):

```
step(tf([1],[m b k]))
```

Note that the decay envelope of the response is defined by the real part of the poles of the system. The characteristic equation for the system (compare with the EOM…) can be written as:

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0$$

If the system is underdamped, this equation has complex roots at $s = -\zeta\omega_n \pm \omega_d j$, where $\omega_d = \omega_n\sqrt{1-\zeta^2}$.

ii) Using MATLAB, produce a plot of the approximate exponential decay envelope bounding the response in Figure 4a. For this system, what are: the time constant of the decay envelope, ω_n, and ζ?

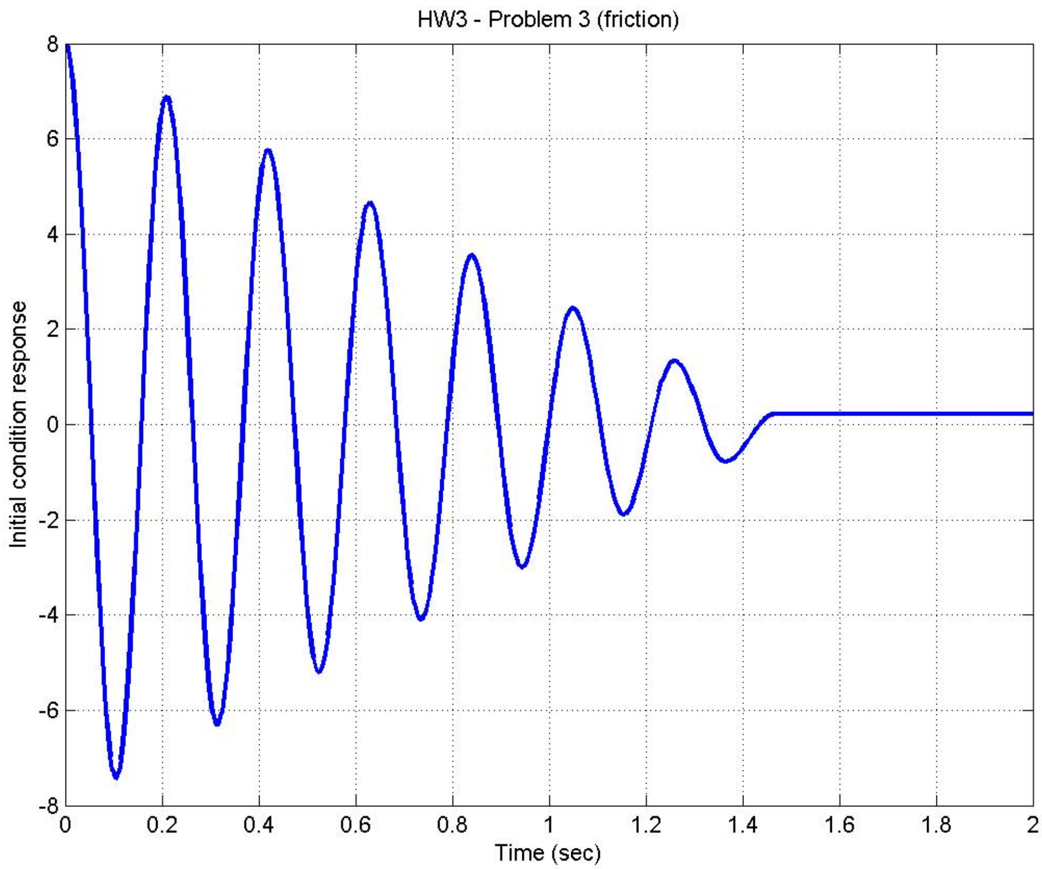

Figure 4b – Step response for a system with Coulomb friction.

b) Figure 4b also shows the step response of a spring-mass system where m=1kg and k=900 N/m, except here losses are due to nonlinear friction. The equation of motion here is:

$$m\ddot{x} + f \cdot sign(\dot{x}) + kx = 0$$

i) The system starts at x=8 meters at t=0. **When the system comes to rest, what is (approximately) the total path distance, x_{path}, the mass has moved?** (For example, if the mass goes from +8 to -8 and back to +8, the path distance would be 2x16=32 meters.) Reason about the geometry of the response curve to solve this…

ii) Note that the magnitude of the friction force is a constant, *f*, any time the mass is moving. Also note that the total energy dissipated will be equal to: $E_{loss} = \int f \cdot sign(\dot{x})dx$. **Write an expression for E_{loss} in terms of the variable f and x_{path}.** (Yes, this is a simple equation!)

iii) The initial energy in the system is equal to the potential energy in the spring: $\frac{1}{2}kx^2$. The response slows down because this initial energy is dissipated, due to friction. **Solve for *f*.**

c) Note in Figure 4b that the response does NOT come to rest at x=0. (The final value on the y axis is offset from 0.) **Explain how this is possible. Will the final value be different, given different initial conditions? Why or why not?**

5) Shaft compliance. In this problem we will look at the effects of shaft compliance ("springiness") on control. Specifically, recall that it is important to know whether your sensor is measuring position on the same side vs on the opposite side to the actuator.

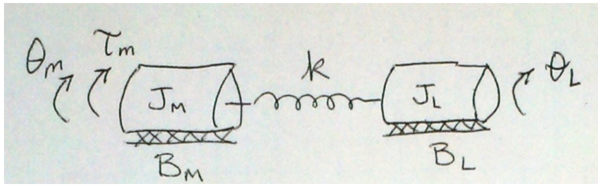

Figure 5a – Model of a system with shaft compliance.

For the system shown in Figure 5a, there are two inertias, Jm and JL, each associated with a degree of freedom. Correspondingly, there are two equations of motion, representing the relationship between total torque and angular acceleration (analogous to "F=ma", except balancing torque, instead of force). These two equations can be found by inspection, by identifying the torque-generating elements attached to each mass, respectively. The "motor" side (m) is driven by τ_m, with both damping, $-B\dot{\theta}_m$, and a spring, $k(\theta_L - \theta_m)$, attached to the mass:

$$\tau_m - B_m\dot{\theta}_m - k\theta_m + k\theta_L = J_m\ddot{\theta}_m$$

At the "load" side (L), terms are similar, except there is a different damper attached (only to the load side), and there is now no driving torque:

$$-B_L\dot{\theta}_L - k\theta_L + k\theta_m = J_L\ddot{\theta}_L$$

Figure 5b shows a block diagram of the system dynamics. **Values for parameters are:**
$J_m = 1$ (kg*m^2) $J_L = 2$ (kg*m^2) $k = 500$ (N*m/rad) $B_m = B_L = 20$ (N*m/(rad/s))

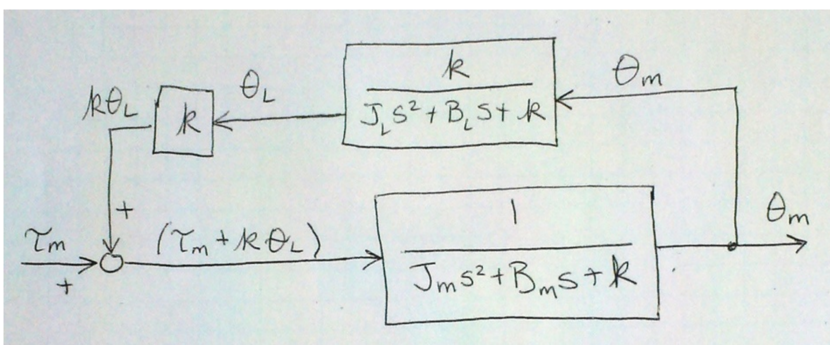

Figure 5b – Block diagram for system with shaft compliance.

a) Derive the transfer functions from torque to motor angle, $\theta_m(s)/\tau_m(s)$, and from torque to load angle, $\theta_L(s)/\tau_m(s)$. Use the bode command in MATLAB to produce Bode plots for each of the two transfer functions on the SAME SET OF AXES. *You should see that the frequency responses look similar at "low" frequencies but differ in important ways at "high" frequency.*

b) Derive the closed-loop transfer function for each system using a proportional controller with $K_p = 500$. Plot step responses for each of the resulting systems on the SAME SET OF AXES. *You should see that the overshoot is larger when θ_L is used for feedback than when θ_m is used.*

Hint: Note Figure 5b shows a positive feedback. Be careful to close the loop appropriately, as shown in Figure 5c.

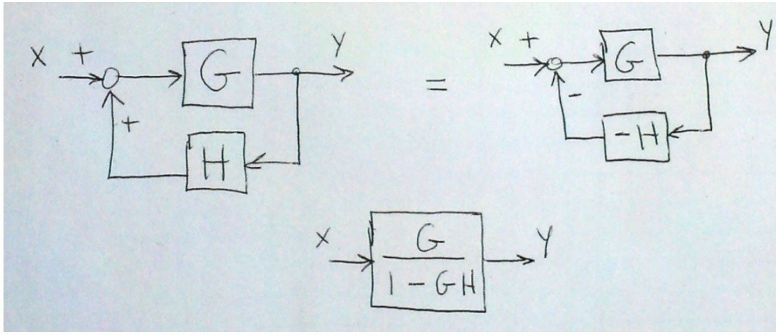

Figure 5c – Closing a loop with "positive feedback".

c) Design a PD controller, $C(s) = K(s+a)$, to get a faster closed-loop step response for $\theta_m(s)/\tau_m(s)$. For example, aim for the step response to settle to within 5% of its final value within about 0.1 seconds. (*Note, "a" sets the location of the zero. Set this first, to add phase at crossover; then set K to adjust the crossover frequency*.) Plot the resulting step response. Try to use your new PD controller for $\theta_L(s)/\tau_m(s)$. Explain why controlling $\theta_L(s)/\tau_m(s)$ is more challenging. (I suggest using your Bode plots from part **a** to answer.)