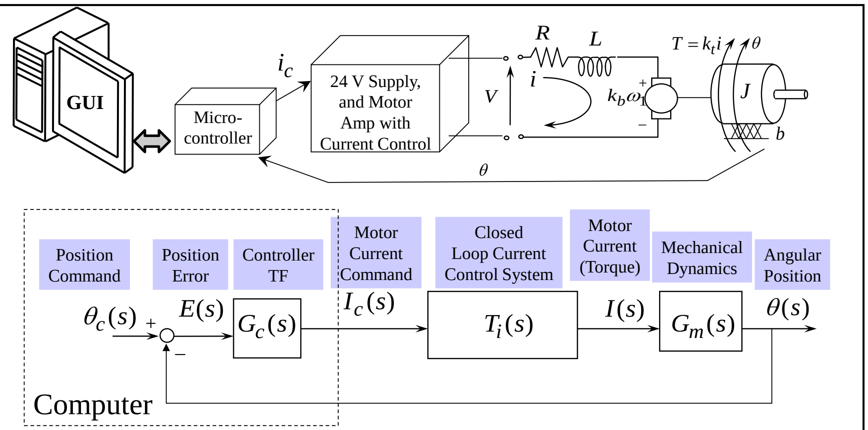
Laboratory #10

In previous labs you experimented with a proportional (P) controller for position control in the Motorlab. We found that as we raised the gain of the P controller that we could obtain somewhat better control of the system, but that the improvement was limited. We could only raise the gain so much before the response became very oscillatory. Furthermore, the settling time could not be improved. Now you are to compare a proportional controller to a proportional-derivative (PD) controller. The PD controller adds a zero to the open-loop TF, changing the shape of the root locus. To obtain experimental data you should use the following three controllers.



$G_m(s) = \dfrac{\theta(s)}{I(s)} = \dfrac{k_t k_{dr}}{Js^2 + bs}$

$G_c(s) = K_p$, proportional controller

and

$G_c(s) = K_p + K_d s$, PD controller

$= K_d(s + K_p / K_d) = K_d(s + z)$

J = motor inertia and collar (from motor lab handout)

b = viscous friction coeficient (estimated to be 3×10^{-5} N·m·s/rad)

k_t = motor torque constant (from motor lab handout)

$k_{dr} = 180^\circ/\pi\,(rad)$ = angular postion unit conversion

$T_i(s) = 1$ (amplifier assumed "fast")

K_p = proportional gain of controller

K_d = derivative gain of controller

$z = K_p/K_d$ = magnitude of controller zero

$\theta(t)$ = angular position (measured in degrees for this problem)

Obtaining Data from Motorlab

One problem with real systems (rather than mathematical models) is saturation. When we use a derivative term in the controller, a step input will saturate the output of the controller in a real system. Therefore, we often do not get a good match between experimental results and theoretical models for a step response when derivative control is used. We will discuss this in the lab preparation. **DO NOT LET THIS DISCOURAGE YOU FROM USING DERIVATIVE CONTROL**. It can be very important in obtaining an optimal closed-loop system. Remember that step inputs are usually used as test signals – not as the actual commands in the operation of real systems such as machine tools, aircraft, etc. Because of this problem we will be using a triangle wave for the test command in this lab. **You should obtain the triangle wave response for the three different position control systems. Use a triangle wave with amplitude of 2000 degrees and a wave frequency of 0.5 Hz. You should obtain three different plots.** Do ***NOT*** click Calculate Step Response Timing since we are not running a square wave.

Three controllers for experimental data

1. P controller, $K_p = 0.001$ Amp/deg (save as data1)
2. PD controller, $K_d = 0.00007$ Amp·s/deg, $z = 10$ rad/s (save as data2)
3. PD controller, $K_d = 0.001$ Amp·s/deg, $z = 10$ rad/s (save as data3)

Using MATLAB –*Declare the Transfer Function Gm first*

Using the "Sisotool" you should play with the systems to get a feel for the responses as the gains change. You should also use the Sisotool to get a feel for root locus. We will do an introduction to the Sisotool in lab. Another related MATLAB function is "rlocus" – try it.

By hand, using the basic rules, you should draw two root locus plots.

- One plot should be for the P controller.
- The other should be for the PD controller with the zero at -10.
- On the plots clearly indicate the closed loop poles for the three different systems. (i.e. Where on the root locus are you for each set of gains?)

Things to Turn In

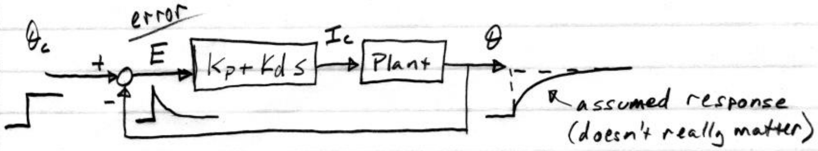
- A plot with all three step responses from the models.
- A plot with all three triangle wave responses.

- The root locus plots (by hand).
- Closed-loop poles and zeros for all three systems marked on page with hand drawn root loci.
- Include a documented (including units) copy of your MATLAB code.
- Completed Fill-in-the-Blanks (with units).

Difficulty with Step responses and controller derivatives

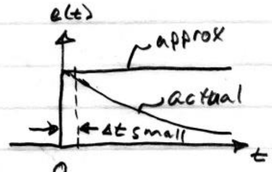
- The difficulty is that we often don't get a good match between the step response predicted by the model and the step response of the actual system when we use a derivative term in the controllor.
- DON'T let this stop you from using derivative control.
- Example: Suppose we use a PD controller

$$G_c = K_p + K_d s = K_d(s+z), \quad z = K_p/K_d$$

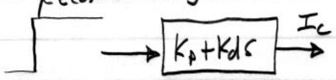


suppose $\begin{cases} -\theta_c = 2000\,deg \text{ step} \\ -K_d = 0.0001\ A\cdot sec/deg \end{cases}$

Now approximate $e(t)$ (error) by a step to capture the initial part of $e(t)$

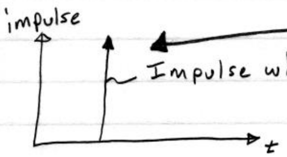


$e(t) = 2000\,deg$ Step $\Rightarrow E(s) = \frac{2000}{s}$



$$I_c(s) = E(s)\cdot G_c(s) = \frac{K_p\cdot 2000}{s} + K_d\cdot 2000$$

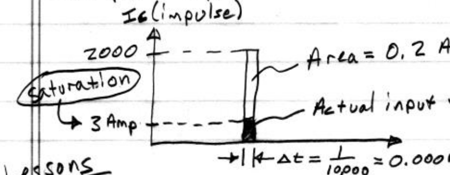
$$\mathcal{L}^{-1}\{I_c(s)\} = \underbrace{K_p\cdot 2000\, u(t)}_{step} + \underbrace{K_d\cdot 2000\,\delta(t)}_{Impulse}$$



Impulse with Area $= K_d\cdot 2000 = 0.2\,A\cdot sec$

Computer's Approximation of the Impulse

- computer opperates on a clock with 10 KHz frequency,



Lessons

- Unless we use very small steps the actual energy that makes it to the system is much smaller than that predicted by the linear model, because of saturation.
- Saturation can also be a factor with large steps and proportional control. It is much easier to calculate when this will happen. $K_p \cdot$ stepsize = saturation value
- Sometimes we want to use signals other than steps as the test signals.

Example C function for PID control

```
float simple_PID_controller(float error, float delta_time)
{
  float output, Kp=1, Ki=2, Kd=3, max_output=100;
  static float integral, last_error;                    // static vars for memory

  integral = integral + Ki*error*delta_time;            // numerical integration
  if (integral < -max_output) integral = -max_ouput;    // anti-integral windup
  if (integral >  max_output) integral =  max_ouput;    // anti-integral windup

  output = Kp*error + integral + Kd*(error-error_last)/delta_time;      //PID

  error_last = error;                                   // remember for next call

  return(output);
}
```