ECE 3085 Lego Project

Dr. Bonnie Heck Ferri
School of ECE
Georgia Tech

Logistics:
Form 3 person groups. You may check out one of the motor assemblies from me. Please reserve the units on the sheet on my door. The required material for turn-in is listed at the end of the project description.

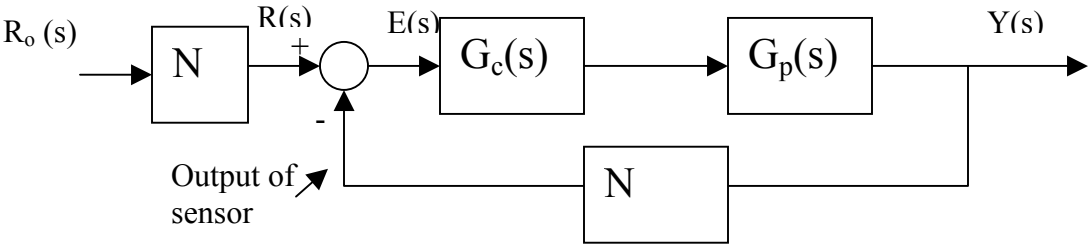
Project Description:
The goal of this project is to design and implement a lead compensator for the Lego motor assembly. (My goal is for you to see how the theory of control is put to practice.)

The transfer function for the motor assembly is

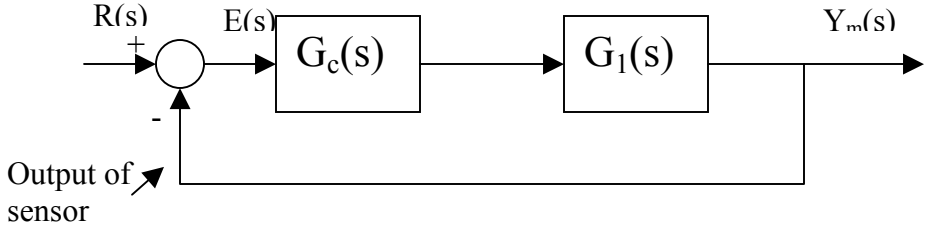
$$G_p(s) = \frac{Y(s)}{V(s)} = \frac{A}{Ts^2 + s}\frac{1}{N}$$

Where y is the angle of the output shaft, v is the voltage to the motor, T is the time constant of the unit, A is a constant, and N is the gear ratio. The objective is to have the angle of the output shaft be regulated around a setpoint (or reference) value, r_o .

So, we decide to implement the following block diagram:

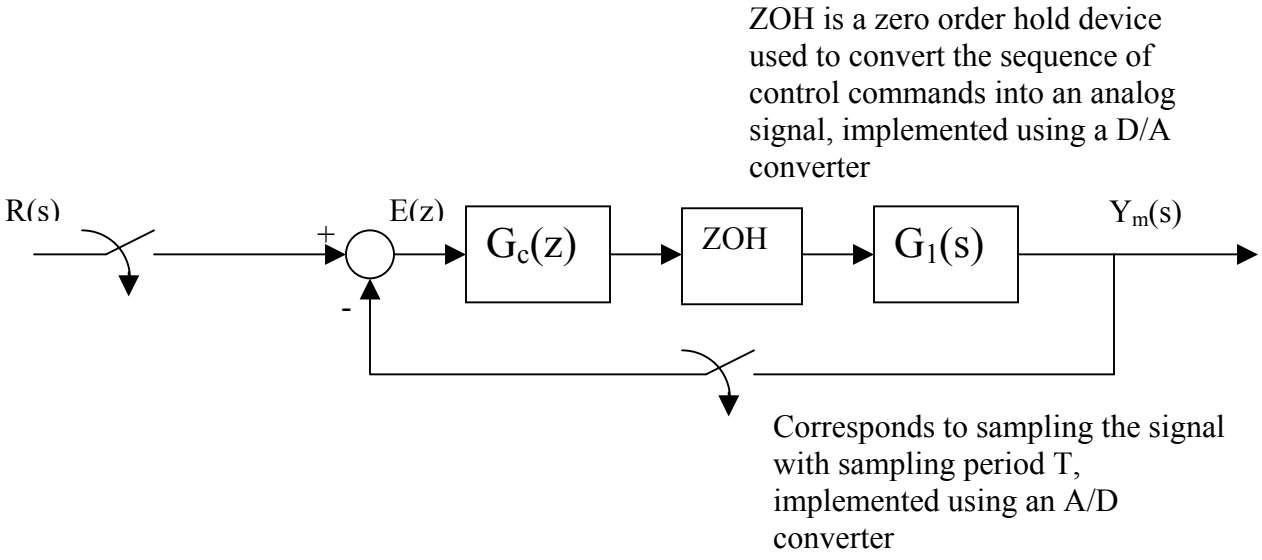


Note that the sensor is on the motor shaft, not the output shaft (hence the factor of N in the feedback loop). Also, the reference value, $r_o(t)$, for the output shaft can be scaled by N to get an equivalent reference, r(t), for the motor shaft. Thus, we can reformulate this block diagram into a unity feedback as follows:

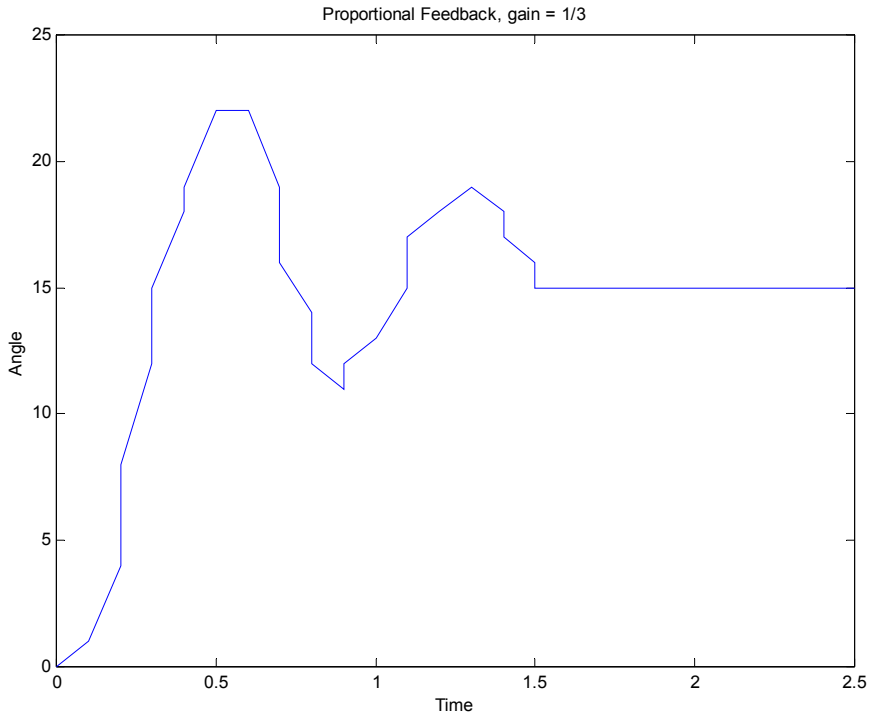


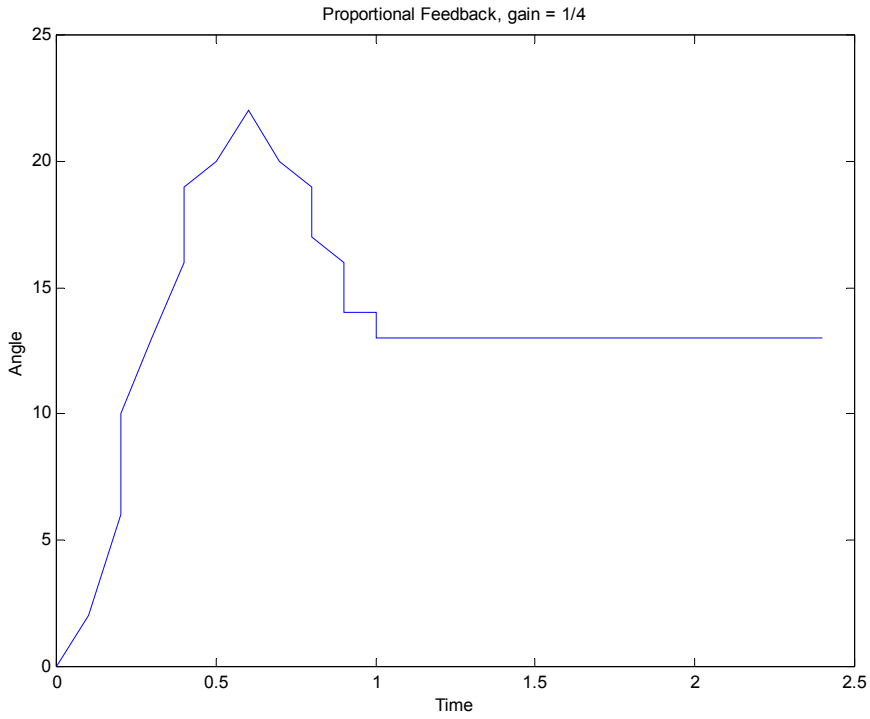
Where $y_m(t)$ is the angle output of the rotation sensor on the motor shaft, and $G_1(s)=G_p(s)N$.

The digital implementation is given as:



Implementing the code using a proportional controller, $G_c(s)=K$, gives the following results where the angle plotted is the angle of the motor shaft.





Note that the results do not look as nice as the simulations that we have run on similar systems. That is because there are a lot of nonideal attributes to this:

- Discrete levels of power to the actuator (0 is the lowest up to 7, which is the highest, plus the float output, which is for 0 power)
- The sensor resolution is 1/16 of a revolution. I added the gearing to effectively increase the resolution on the output shaft that we are regulating (so, if the gear ratio is 1:5, then the resolution would be improved by a factor of 5).
- There is slop in the gears, causing dead-zone behavior.

The above plots were obtained by saving the data in a datalog and then uploading them from the processor to the PC. The command for uploading them and saving them to a file is: **nqc –datalog >filename.txt** where filename.txt is the name of your data file. This can be plotted in Matlab by first editing the file to remove the text on the first line. If your file name is data.txt, then type **load data.txt** from the Matlab prompt. For my plots, I had to separate the data into the time and the angle points. I did that using commands, **time = data(1:2:100);** and **angle = data(2:2:100);** The data for time is in tenths of seconds, so multiply the time values by 0.1 to get time in seconds. The angle measurement relates to the number of $1/16^{th}$ of a revolution for the motor shaft. Multiply by 360/16 to get degrees of the motor shaft; to convert this value to degrees of the output shaft, divide by the gear ratio (N=5).

Assignment:

1) Find the approximate model for the motor assembly, $G_1(s)$, in the form of $G_1(s) = \frac{a}{s(s+b)}$. All the information you need can be obtained from the closed loop step response and from a root locus analysis.
2) Design a lead compensator and implement it digitally. It should have a smaller time constant than the proportional controller. To do this, you will need to know the sampling time. This can be found by writing your code with arbitrary values for your compensator parameters and running the code for a set number of loops. Initialize the timer at the beginning of the first loop and then read the time at the end of the last loop. Divide this time by the number of times through the loop to get the average sampling time.

Your write-up should include a description of the problem, your model with a description of how you obtained it, the analog compensator design, the discretization to digital, the copy of your code, angle plots from the different compensators that you have designed, and an electronic version of your code. You must also meet with me to demonstrate your design.

Resources:

NQC website: www.bricxcc.sourceforge.net/nqc

You will find the downloadable code for executing NQC, the programmer's guide and the user's guide (for compiling and downloading it).

Sample code for the proportional feedback controller.

Appendix:

Just as a comment (not required): The power level to the motor ranges from 0 (lowest) to 7 (highest). Below is a plot of the normalized speed versus the power level command to the motor. (It is normalized by the fastest speed achievable with this motor assembly.) Ideally, it should be a straight line going through the origin. Thus, we really should create a table look-up that linearizes this curve. To the 0-7 power level motor commands, we add the command "Float", which is equivalent to sending 0 voltage.

We should devise a function that inputs the compensator command (the command is the output of the compensator block) and outputs a power level command in the range 0 –7 and float (a total of 9 possible values). The goal is to linearize the above plot.