Mini Project 2: Beam Vibration Suppression

Introduction

For mini project 2, your task is to design a vibration suppression control system for the flexible beam/DC motor system you performed system ID on in lab 6. A recommended block diagram for the vibration suppression controller is shown in Figure 1. The goal of the project is to design $A(s)$ and $D(s)$ to minimize the settling time for both the encoder and the accelerometer when the system is given a step-change in desired encoder position $U(s)$. The block diagram of Figure 1 is just a recommendation; you are free to experiment with other approaches. For the final verification test, your encoder must complete a step response of 100 counts and the accelerometer must settle close to zero as described in the next section.

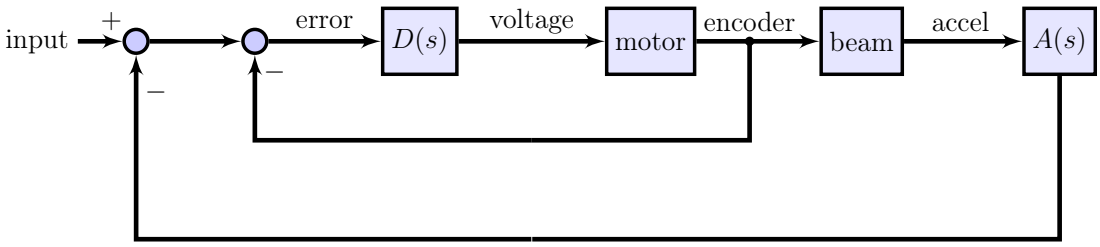

Figure 1: Block diagram for vibration suppression control based on accelerometer feedback.

Settling Time

Settling time is defined in the FPE controls textbook in Section 3.4.3. The settling time for a step response is the time that it takes for the output to get very close to the desired stopping point. Generally, a window of ± 1% of the final value is established as the definition of being close to the desired stopping point. The settling time is the amount of time that it takes for the response to get to where it stays within this window for the remainder of the response.

Figure 2 shows the step response for the encoder. Figure 3 zooms in to clarify the definition of settling time. Since the final value is 100 counts, ± 1% leads to a window of 99-101 counts.

Settling time for the accelerometer signal is harder to define because the desired final value is 0 and the accelerometer signal can be noisy.

The acceleration settling time is being defined as the time when the acceleration signal gets to and stays within ±50 counts. This number may be adjusted if needed at the discretion of the instructor. Lowpass filtering of the accelerometer signal before calculating settling time may be considered for the final competition.

Figure 4 shows the accelerometer step response for Dr. Krauss' first design. Figure 5 zooms in to show the effect of noise on the settling time measurement.

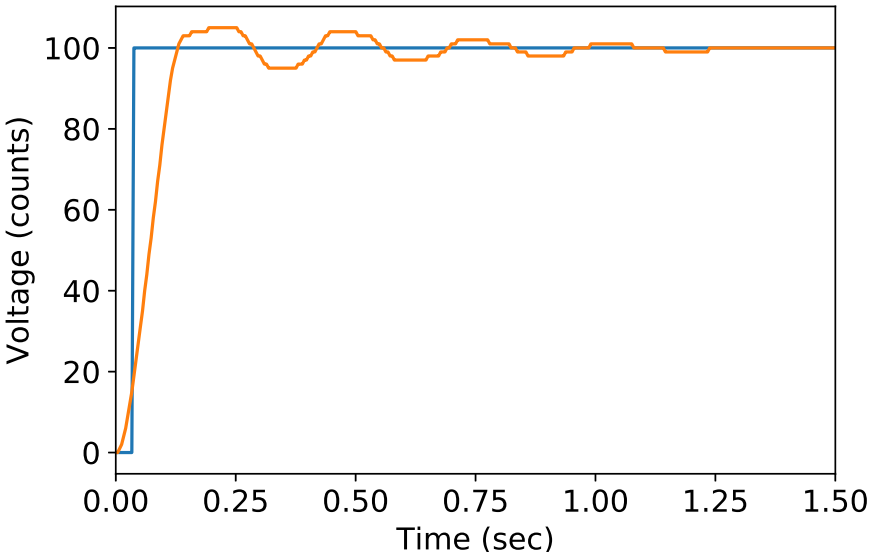

Figure 2: Step response for the encoder given a step input in $u(t)$.

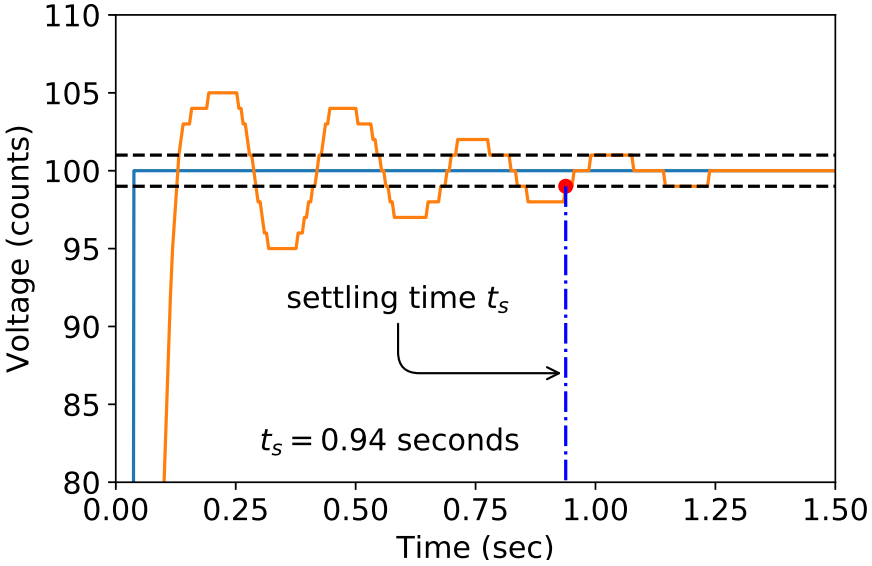

Figure 3: Zooming in on the step response for the encoder to define settling time.

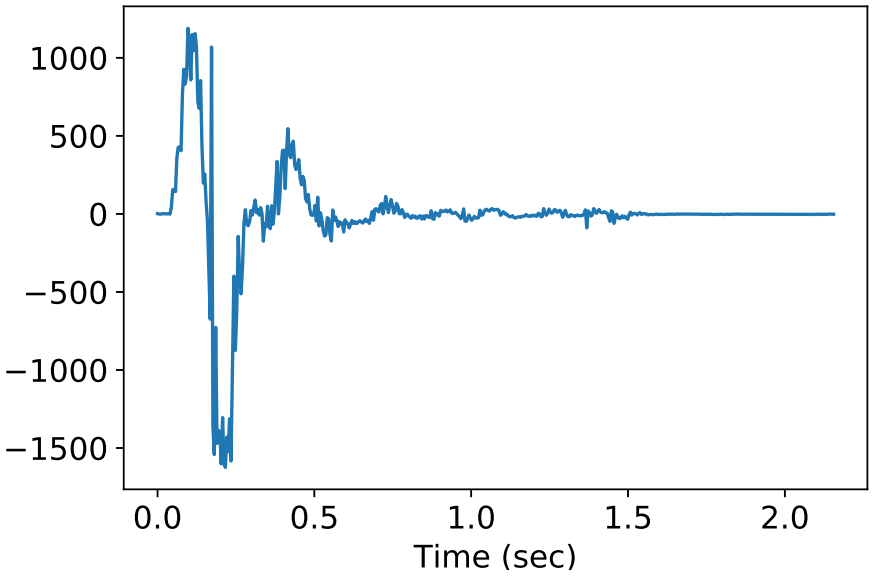

Figure 4: Accelerometer response to a step input in u.

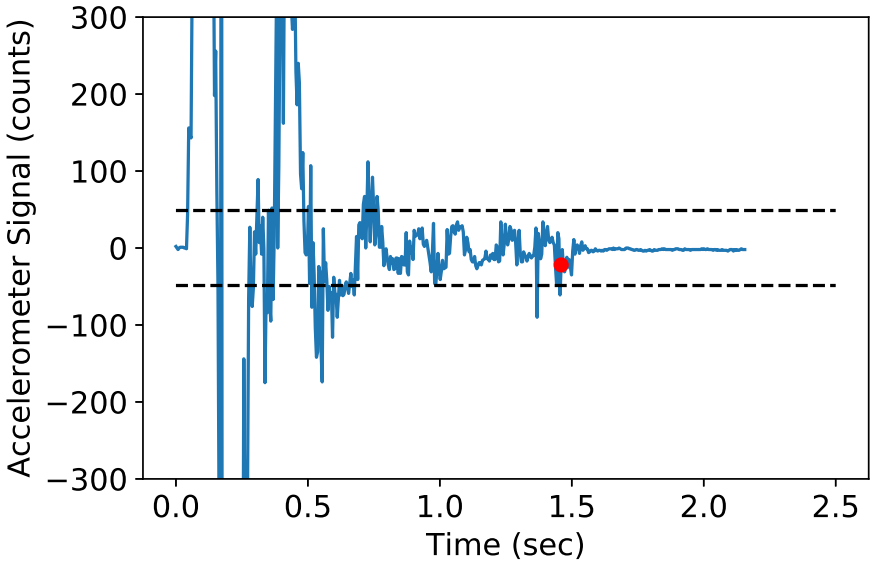

Figure 5: Zooming in on the accelerometer step response, showing how noise affects the step response value.

Recommended Steps

Note: It is fairly likely that your vibration suppression design will at some point drive the system unstable. One team member should always be ready to disconnect the 6AA battery pack to stop power from going to the motor.

- create a block diagram for encoder feedback of the DC motor with the accelerometer output, but without $A(s)$
  - you will use the block diagram to tune $D(s)$ and perform system ID
- tune the motor controller $D(s)$ so that it causes vibration when you do a step response
  - if it is not tuned agressively enough to cause vibrations, it will not be able to suppress them later
  - https://youtu.be/zsC8R_xJmCw
- run one or more swept sine tests and plot the Bode for Accel/U
  - this was done in Lab 6, but may need to be revised
- design your first prototype for $A(s)$ based on root locus and the transfer function you found for Accel/U in Lab 6
  - note that the overall gain for $A(s)$ can be a menu parameter
    - this is essentially the same as varying K on the root locus
- try a step response based on your first design for $A(s)$
  - start with a small gain and work your way up and see how it affects the settling time
- try various forms for $A(s)$
- consider optimizing $D(s)$

`digcomp` block

- Note that PID tuning is highly unlikely to work for this project
  - Your root locus should show you that P control won't work
  - PD and PID are also not likely to work
- In order to use an arbitrary TF in the time domain, we need to convert it from the `s` domain to the digital `z` domain using a function from the Python control module called `c2d`. An example notebook will be provided that will help you convert your $A(s)$ into digital numerator and denominator arrays that can be pasted into the wxPython GUI. The Arduino code can accept the gain of a digcomp block as a menu parameter. Even though the wxPython GUI will allow you to choose the numerator and denominator arrays as menu parameters, this is not yet supported on the Arduino side.