

SE420 Laboratory Assignment 1

Welcome to the SE420 Lab (Where: Room 3075 ECEB)

Goals for this Lab Assignment:

1. Provide access to the lab room and its PCs
2. Go over the use of the PCs, where you will be saving your data, etc.
3. Rules for lab usage.
4. Quick introduction to the DSP controller we will be using and its development software called Code Composer Studio.
5. Introduction to GIT and the SE420 GIT repository.
6. System Identification of a cantilever beam system.

Matlab Functions Used: None

Prelab: No Prelab.

Grading Policy

- Each lab is worth 10 points and there are 10 labs so a total possible 100 points.





Points	Grade
90	A
80	B
70	C

- Each lab should be completely signed off by your lab instructors before the beginning date of the next lab assignment.
- If you do not complete the lab by the beginning of the next lab, you will be given a one week extension to complete the lab without losing points. It is highly recommended that you keep up with the given due dates. This way you will not get far behind and have to rush through lab assignments to catch up.

Laboratory Exercise 1, Clone the SE420 Git Repository

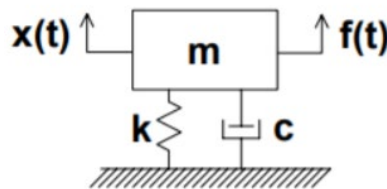
- First follow the other Lab 1 document “Using the SE420 Repository” and its first two sections “Create an account at Github.com” and “Create your Repository” to check out the SE420 repository to your lab PC (and, if you want, your personal laptop). If you are going to be using both the lab PC and your laptop for development you will want to create the same path on both your laptop as on the lab PC. As the “Using the SE420 Repository” states, create a folder using a combination of your NetIDs at the root C:\ drive and keep all your files there. It is also a good idea for you to make backups of your lab files outside of your Git repository as you progress through the semester. Making this extra backup will save you when we can’t figure out what went wrong with Git. Git is awesome and normally works great but every once in a while I have seen issues that caused students to lose versions of their files. This is mainly due to us being beginner Git users. I am getting better each year though.

Laboratory Exercise 2, Load your first Project into Code Composer Studio and Flash the default program to the DSP processor.

- Open Code Composer Studio 12 (CCS 12) and select the “workspace” folder in your repository. For example, if your netIDs were “jonesz3 and smithd4”, you would have checked out your repository in `c:\jonesz3_smithd4\SE420_repo`. The workspace folder then to select would be `c:\jonesz3_smithd4\SE420_repo\workspace`.
- Once CCS 12 is done loading your workspace, you need to load the “LABstarter” project. When you perform this load, the project is copied into your workspace. Therefore, if you rename this project, you will be able to load the “LABstarter” project again if you would like to start another minimal project. I purposely located this “LABstarter” project in the same folder that has many of the example projects you will be studying this semester. These examples are part of the software development stack that TI calls C2000ware. I have copied the needed parts of C2000ware into our repository so that if you accidentally modify something you can easily get it back. Again now if your NetIDs were “jonesz3_smithd4”, perform the following to load your starter project. In CCS select the menu Project->Import CCS Projects. Click “Browse” and explore to `“C:\jonesz3_smithd4\LabRepo\C2000Ware_4_01_00_00_F28377S\device_support\2837xs\examples\cpu1\LABstarter”` and finally press “Finish”. Your project should then be loaded into the CCS environment. Let’s then rename the project “lab1” by right clicking on the project name “LABstarter”. In the dialog box change the name to “lab1”. Also explore into the project and find the file “LABstarter_main.c”. Right-Click on “LABstarter_main.c” and select “Rename”. Change the name to “lab1_main.c”. To save you some headaches in the future, I recommend you perform one more step after you have created a new “LABstarter” project. Right click on your project name in the Project Explorer and select “Properties.” Select “General” on the left hand side. Then in the “Project” tab find the “Connection” drop down and select “Texas Instruments XDS100v2 USB Debug Probe.” “Apply and Close”.
- Now that you have the project loaded you can build the code and download it to the LaunchPad board. Plug in your LaunchPad to the USB of your PC and then in CCS hit the green “Debug” button . This will compile the code and load it to your LaunchPad board.
- Click the Resume button  to start/resume code and the Suspend button  to pause the code. The default code is blinking the red and blue LEDs very fast. Use the Suspend and Resume buttons to prove to yourself that both the blue and red LEDs are blinking on and off.
- Also use the Restart button  to have your code start back at the beginning of main() and start your code again. This will be nice to use in the future when you want to rerun your code
- Read through the main() function of your lab1_main.c file and see if you can find the function that changes the period of the timer functions. Change the period and see if the LEDs blink at a different rate. There are three timers. Which timer interrupt functions blink the LEDs which are connected to General Purpose Input/Output (GPIO) pins 12 and 13?

Laboratory Exercise 3, Identify the Continuous Transfer Function of Cantilever Beam System

- As a nice little exercise to get you working with lab equipment, you are going to find an approximation of the continuous transfer function of the cantilever beam system at your bench. In a future lab, you will convert this to a discrete transfer function and design a discrete controller to change the response of the system.
- The MagLev (cantilever beam) experiment at your bench is a thin brass beam with two magnets at the end. Forty coils of printed circuit board traces are located below the magnets that produce a proportional force on the magnets when driven with current. Analog Hall Effect sensors are placed to the side of the magnets that sense the position of the magnets and output a proportional voltage. The below figure shows a free body diagram of the approximate model of the system. Your input (EPWM8A) to the MagLev beam is the force (f) in the figure. Your feedback signal that we will connect to the Oscilloscope (later labs connected to the DSP through an analog to digital converter, ADCB2) is the position (x) you want to monitor, and in a later lab, control.



If you flick the cantilever beam, you will see that its response is very oscillatory. This makes our job of system identification easier because this oscillation is close to the system's natural frequency ω_n . The approximate equation of motion for this mass spring system is

$$m\ddot{x} + c\dot{x} + kx = f(t).$$

Which as the transfer function

$$\frac{X(s)}{F(s)} = \frac{1}{k} * \frac{k/m}{s^2 + c/m s + k/m}$$

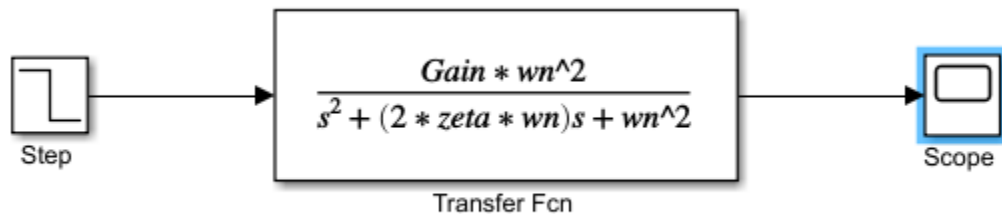
We are not as interested in the values of k , m , c but instead the full transfer function. Thus, you can think of this transfer function as the standard second order transfer function

$$\frac{X(s)}{F(s)} = Gain * \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}, \quad poles = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2}, \quad with \omega_d = \omega_n\sqrt{1-\zeta^2}$$

With this transfer we only have to identify the Gain, ω_n and ζ .

- Have your instructor show you how to setup the MagLev experiment. You will be hooking up the 9V power supply input for the coil and the oscilloscope to measure hall effect position measurement. As you instructor will show you how to enable and disable the MagLev's amplifier, please do not leave the MagLev's amplifier enabled for a long period of time (over 3 minutes or so) because it gets pretty hot. Best practice is to collect the needed responses and then disable the amplifier.
- Flick the beam and notice the highly oscillatory response of the system. Your instructor will demonstrate for you how to use the cursors of the oscilloscope to measure time and voltage differences on its display.

- The next step is for you to write a bit of C code that drives an open loop step input in to the beam system. The open loop step that I would like you to create starts at 0 input and then steps to -1.0 after 15 seconds. Then after 15 seconds it steps back to 0. Then the input repeats at the 15 second interval. To output this step value use the function `setEPWM8A(0)` to output 0 and `setEPWM8A(-1.0)` to output -1.0. To get the timing correct use Timer 2's interrupt function and set it up so that it is called every 1 second (remember what you did in the previous section to change the blink rate of the LEDs). Then notice that inside `cpu_timer2_isr()` there is a count variable `CpuTimer2.InterruptCount` that is incremented by 1 each time in the function. Use this count to know when 15 seconds have gone by, (notice the use of the % (modulus) to determine when 50 counts have happened). Finally create an integer variable to keep track of whether the output is set to 0 or set to -1.0. Every 15 seconds check if the output is at 0 then set the output to -1.0, else if it is at -1.0 set the output to 0. Again use the `setEPWM8A()` function to change the output.
- Once your program is running correctly stepping back and forth, collect data on the oscilloscope. Always remember to disable the beam's amplifier when you have collected data and are performing the measurements on the scope. Adjust the time division of the scope to produce plots that allow you to measure ω_d and *Gain*. Since our system is so oscillatory we are going to equate $\omega_n = \omega_d$. To find *Gain* remember that it is the steady state output over the input (Output/Input). If your response plot is still oscillating at 15 seconds, approximate the steady state value as the center of the oscillations.
- The final step is to find ζ . A simple way to find ζ is guessing and checking ζ using a Simulink simulation of just the open loop transfer function given a step input of -1.0. Start ζ at 0.05 and keep on lowering it until the simulated response approximately matches the oscillation amplitudes of the actual system response. Start Matlab and create the following Simulink model



Check your values of *Gain*, ω_n and ζ with your instructor.

- Once you have found *Gain*, ω_n and ζ , make sure you keep the values because you will need them later in the semester. To force you to save these values, I would like you to create some lines of comments in your lab 1 C-file and commit your lab 1 files to your repository. Perform the steps in “Back up your Local Repositories Workspace folder to GitHub” to commit and push your code to Github.

Lab Check Off:

1. Demonstrate to your TA that you have successfully created, built and ran your first DSP project.
2. Demonstrate changing the period of the blink rate of the blue and red LEDs.
3. Show your identified *Gain*, ω_n and ζ values to your instructor and show that you have committed your Lab 1 C code to your Git repository.