LABORATORY 3: The Signals and the Noise

Contents

Getting Started!

Lab Submission Instructions and Partners:

1) Keep your partners! If a student dropped the course, let us know!
2) Decide if you'd like to do an Alpha Experiment or Omega Exploration.
3) You will go through each lab section to learn some basic steps. ***Then you will prove specified concepts listed at the end of each section.*** This simply means that you will demonstrate how a specific concept works through clear and concise comparisons of mathematical analysis, simulation, and experimental measurements
4) The template for the **Proof of Concepts** document that you will submit can be found here: https://sites.ecse.rpi.edu/courses/F24/ECSE-1010/resources/labs/templates/Proof_of_Concepts_Template_F24.docx

   Please answer any questions related to those concepts and provide mathematical calculation, simulation, and experimental data to support the proof of concept!

Purpose: The objective of Lab03 is to introduce some of the main ideas of signal processing such as signals in the time and frequency domains, Fourier analysis and synthesis, and designing filters to shape/modify a signal's frequency spectrum. Aside from the familiar steps of mathematical analysis by hand, simulation with LTspice, and building and testing a circuit on your protoboard, you will be using Simulink to build and test systems for signal processing.

Student Preparation BEFORE doing this experiment: (Students should be able to)

- Install a Matlab toolbox (DSP System Toolbox and Audio Toolbox required in Simulink)
- Determine the frequency and period of a sinusoidal wave via hand calculations and a waveform in the time domain
- Add two sinusoidal waves together in Matlab & Simulink and plot them in the time domain
- Determine the transfer function of a 2-resistor voltage divider
- Determine the real and imaginary parts of a complex number

Learning Outcomes AFTER doing this experiment: (Students will be able to)

Part A: Properties of Sinusoidal Waves in the Time and Frequency Domains

- Recognize the frequency spectra of sinusoidal waves and identify their frequencies
- Describe how the time domain plot, frequency domain plot, and pitch of sinusoidal waves change with frequency
- Describe the effect that adding a sinusoidal wave of a different frequency to a signal has on its time domain plot, frequency domain plot, and sound

Part B: Analyzing and Synthesizing Signals

- Generate square, sawtooth, and triangle waves of a particular frequency and duty cycle in Simulink
- Explain the link between signal properties in the time domain and the frequency domain and how this determines the qualitative sound of a signal
- Explain the basic concept behind Fourier analysis and synthesis
- Identify high- and low-frequency components of a periodic signal in its time domain plot

Part C: Digital and Analog Filters

- Determine the complex impedance of capacitors and inductors from the element values and angular frequency ω
- Determine the values of capacitors and inductors from the values printed on them
- Determine the transfer function of simple series RC and RL circuits
- Explain the difference between digital and analog filters
- Identify low-pass and high-pass filters by their magnitude Bode plots
- Perform an AC sweep of RC and RL circuits in LTspice and plot both the magnitude and phase of input and output voltages
- Analytically calculate and experimentally determine the corner frequency of a first-order low- or high-pass filter

Software and Equipment Required:

- LTSpice
- MATLAB Simulink
- ADALM2000 (M2K) board with Scopy or Analog Discovery with Digilent Waveforms
- Components from the ADALP2000 parts kit: several different resistors, capacitors, inductors, and wires
- Protoboard

Learning from Proof of Skills applied to this lab:

Professional Accountability: I can **clearly document and compare a calculated, simulated, and experimental result** to answer the question "Is this right?" for myself

Circuit Simulation: I can **perform a transient simulation with a sinusoidal source and plot the resulting waveforms**. I **can perform an AC analysis simulation to plot the frequency response of an RC or RL circuit**.

Experimental Measurements and Personal Instrumentation: I can **use a function generator to create a sinusoidal waveform**. I can **measure voltage across one resistor with a sinusoidal input source.**

MATLAB Basics and Simulink: I have **completed the Simulink Onramp Tutorial**. I can **add two sinusoidal waves and show the display using Simulink**.

PART A [Core] – Properties of Sinusoidal Waves in the Time and Frequency Domains

Material covered: Analysis of Sinusoidal Waves in the Time and Frequency Domains; Time and Frequency Domain Properties of Sums of Sinusoidal Waves

Background: Sinusoidal Waves in the Time and Frequency Domains

Sinusoidal waves appear in many areas of physics and engineering and, as you will learn in this part of the lab, are the basic building blocks of all signals. Sinusoids are most often encountered in the *time domain*, where their amplitude is plotted as a function of time according to the expression

$$y(t) = A\sin(\omega_0 t) = A\sin(2\pi f_0 t),$$

where A is the *amplitude*, $\omega_0 = 2\pi f_0$ is the (*angular*) frequency in rad/s and t is the *time*. Sine waves are periodic functions, which means that they consist of repeating units of length T, which is called the *period* of the function. For a sinusoid of frequency f_0, the period is given by $T = 1/f_0$. As a result, changing the frequency of the sine wave also changes the period, leading to a stretching (smaller f_0) or compression (larger f_0) of the sine wave in time. A sine wave with an amplitude of 0.5V and frequency of 440Hz is shown in Figure 1.

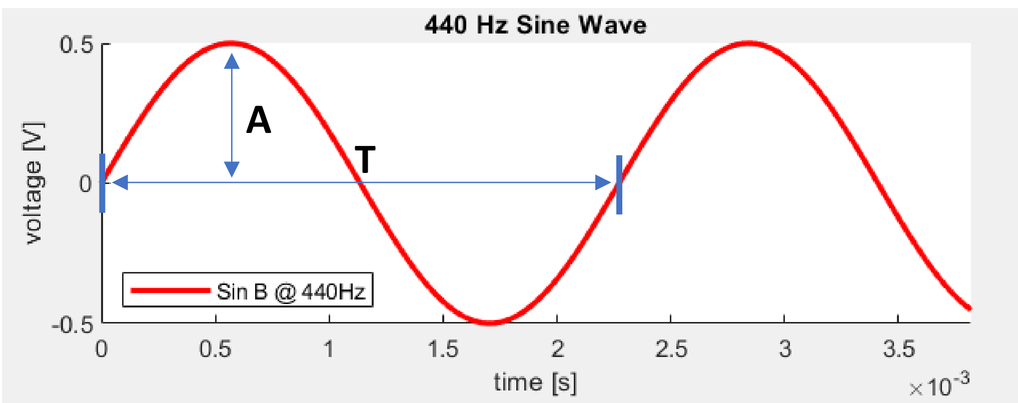


Figure 1 – 0.5V amplitude, 440 Hz sine wave plotted in the time domain

Signals can also be represented in the *frequency domain*, in which a signal's amplitude is represented as a function of frequency instead of as a function of time. This is called a signal's *frequency spectrum*. In a frequency domain plot, the x-axis gives the frequency and the y-axis represents how much power the signal contains at a particular frequency. Since sine waves consist of a single frequency f_0, their representation in the frequency domain is a vertical line at f_0 with an amplitude proportional to the signal's amplitude in the time domain, as shown in Figure 2. Due to the mathematical transform used to convert a signal from the time to frequency domain (Fourier Transform), a peak also appears at $-f_0$.

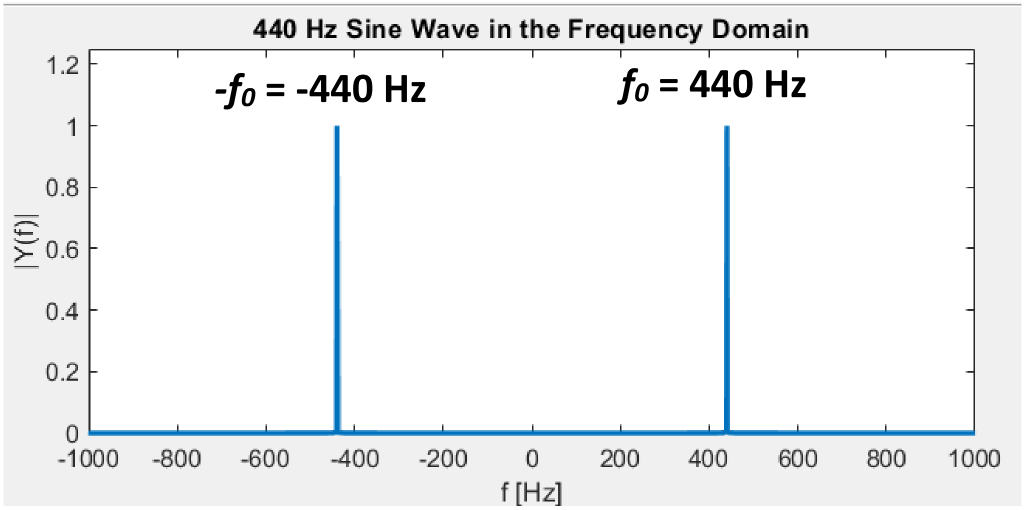


Figure 2 - 440 Hz sine wave in the frequency domain

When two sinusoidal waves are summed in the time domain, the resulting signal is the mathematical sum of the amplitudes of the two waves in time.

$$y(t) = A_0 \sin(2\pi f_0 t) + A_1 \sin(2\pi f_1 t)$$

Likewise, the frequency spectrum of the sum of two sinusoidal waves is also the mathematical sum of their frequency spectra $Y_1(f)$ and $Y_2(f)$, as shown in Figure 3.

$$Y(f) = Y_1(f) + Y_2(f)$$

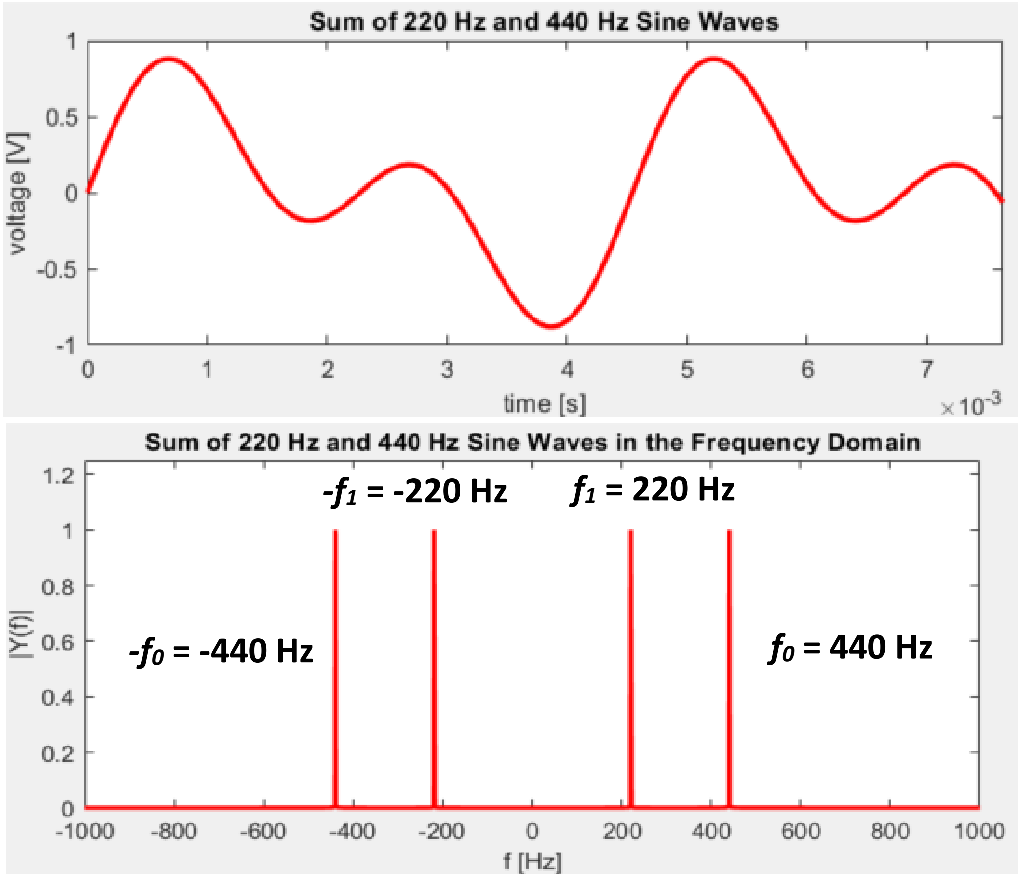


Figure 3 - Sum of 220 Hz and 440 Hz sine waves in the time and frequency domain

Experiment: Sinusoidal Waves in the Time and Frequency Domains

In Part A of this lab's experiment, you will be learning more about the time and frequency domain properties of sinusoidal waves and sums of sinusoidal waves, as well as how they change with frequency. This will lay a foundation for Part B, in which we will analyze different, non-sinusoidal waveforms in terms of their frequency spectra.

Analysis: Time and Frequency Domain Properties of Sinusoids

1. Write the expressions for sinusoidal waves with amplitude 0.5 and frequency f = 500 Hz and f = 750 Hz in the time domain. Calculate the period of each of the sinusoids. Sketch both waves on the same axes.
2. Sketch the frequency spectrum of each of the sinusoidal waves on the same axes.
3. Sketch the sum of the two sinusoids in the time domain (you may use a calculator or Matlab to assist you).
4. Sketch the frequency spectrum of the sum of the two sinusoidal waves.

Simulation: Simulink

Simulink is a convenient software package for basic signal processing, as it allows for quick generation of a multitude of common waveforms and has many tools for modifying and analyzing them. As a note, the steps below could also be carried out in a Matlab script. *Note*: Before beginning this part of the lab, be sure that you have installed the Digital Signal Processing (DSP) System Toolbox and Audio Toolbox for Matlab. If it has been installed successfully, you should see the "DSP System Toolbox" and "Audio Toolbox" categories in Simulink when you open the Simulink Library Browser.

1. Open a blank model in Simulink, then add the following blocks from the Simulink Library Browser and connect them as shown in Figure 4:
    a. "DSP Sine Wave", found under DSP System Toolbox > Sources. Once placed in the model, open the block and set the amplitude to 0.5, frequency to 500 Hz, and sample time to 1/(44100 Hz). 44.1kHz is the standard sampling rate for digital audio.
    b. "Buffer", found under DSP System Toolbox > Signal Management. Set the output buffer size to 1024. This allows you to play back the generated signals smoothly.
    c. "Audio Device Writer", found under DSP System Toolbox > Sinks. This outputs the signal to your laptop speakers.
    d. "Time Scope", found under DSP System Toolbox > Sinks. This allows you to view the signal in the time domain.

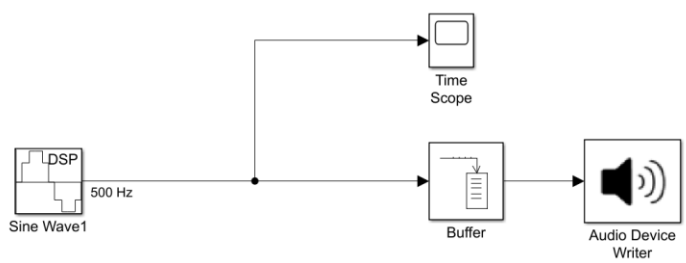


Figure 4 – Simulink model for playing a 500 Hz sine wave and viewing it in the time domain

2. Set the Stop Time at the top of the Simulink window to 2 (seconds).

3. Click "Run" to run the simulation. A 500 Hz tone will play through the computer's speakers and the time scope will plot the sine wave over a time span of 2 seconds.

4. On the scope, zoom in on the sine wave so that about two periods are visible. Using the "Cursor Measurements" tool ( ) from the toolbar near the top of the window, measure the length of one period of the sinusoid. If you need more points in your time-domain plot, you can change your sample time so that there are more points per second. Does this match what you calculated above?

5. Add a "Spectrum Analyzer" block (from the DSP System Toolbox) to the model and connect the Sine Wave block to it. This allows you to view the signal in the frequency domain. Open the settings for the spectrum analyzer and navigate to Estimation > Frequency Options. Set "Frequency Span" to "Start and Stop Frequencies" and enter -2000 into the "Start Frequency (Hz)" box and +2000 into the "Stop Frequency (Hz)" box: this will limit the spectrum analyzer to processing frequencies

between -2000 Hz and +2000 Hz, which increases its frequency resolution. You will also need to navigate to the Spectrum tab and check the “Two-sided spectrum option” in order to see negative frequencies on the spectrum analyzer. It is always a good idea to adjust this setting to be just large enough to include all frequencies of interest to you. Run the simulation again to update your plot. If you are not using a two-sided spectrum, you can increase the resolution of your plot at low frequencies by setting “Frequency Scale” to “Linear” in the Spectrum tab.

6. The Spectrum Analyzer plot should open automatically and show a symmetrical plot with two peaks. Zoom in on the peaks, then use the Peak Finder tool () from the toolbar at the top of the window to find the location of the two peaks in frequency. At which frequencies are the peaks located?

7. Copy the sine wave block and place it below the first sine wave block. Change the frequency of the original Sine Wave block to 750Hz. Connect both Sine Wave blocks to the time scope and spectrum analyzer and label the wires, as shown in Figure 5.

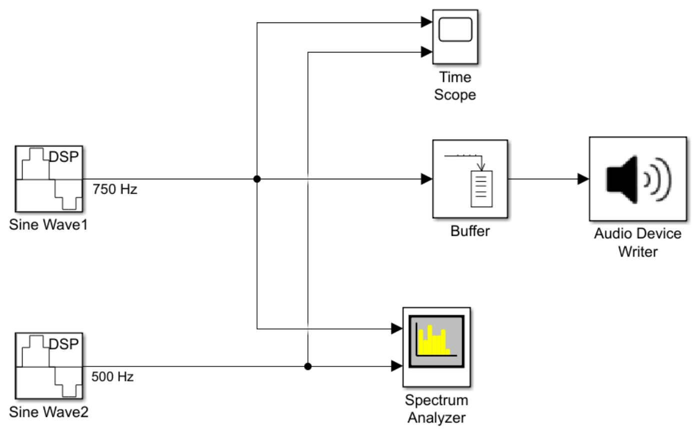


Figure 5 – Simulink model with two sinusoids, spectrum analyzer, and time scope

8. Run the simulation to listen to the 750Hz sine wave and view the results in the time scope and spectrum analyzer. On the time scope, verify that the period of the 750Hz sine wave matches your previous calculations and that the frequency peaks on the spectrum analyzer match the frequency of the sine wave. What happens to the time-domain plot (time scope), the frequency-domain plot (spectrum analyzer), and pitch of a sine wave, as its frequency increases?

9. Add an “Sum” block to the model before the buffer block and connect the model such that each of the sine waves are being fed into the sum block and the output of the add block is being fed to the time scope, spectrum analyzer, and audio device writer, as shown in Figure 6.

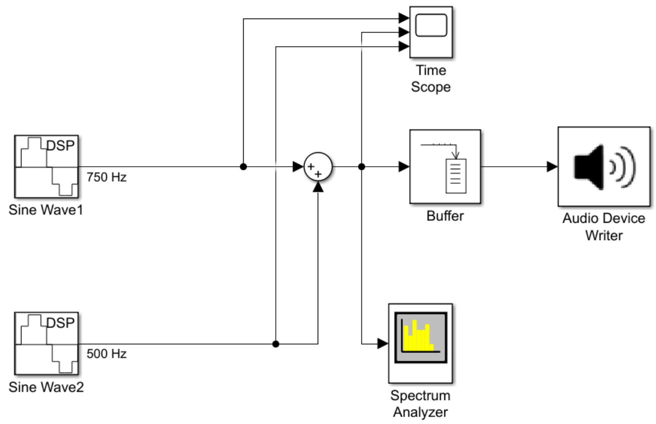


Figure 6 - Simulink model of a sum of two sinusoidal waves

10. Run the simulation to listen to the sum of the sinusoidal waves and view the output of the time scope and spectrum analyzer. How does the signal sound? What happens when two sinusoidal waves are summed in the time domain? What about in the frequency domain? Is this what you expected?

Part A: Proof of Concepts List

1. **Prove the result of summing *three* sinusoidal waves in the time and frequency domains.** ***For the Mathematical Analysis step, you may use Matlab to plot the time domain signal, but you must hand sketch the frequency spectrum. There is no circuit experiment for this concept; proving it via a simulation in Simulink is enough.***

PART B [Core] – Analyzing and Synthesizing Signals

Material covered: Generation of Basic Non-Sinusoidal Waveforms in Simulink; Fourier Analysis of Non-Sinusoidal Signals; Fourier Synthesis for Approximating a Signal

Background: Fourier Analysis & Synthesis

Fourier analysis is the decomposition of a signal into a sum of sinusoidal waves of different frequencies. This is a tool (math!) that helps us better understand the frequency content of a signal. As we saw in Part A, the frequency spectrum of a sine wave consists of two narrow peaks located at the sine wave's frequency f_0 (and $-f_0$). Looking at the frequency spectrum of non-sinusoidal signals, such as the spectrum of a square wave in Figure 7, we see that the spectrum is made up of many narrow peaks at different frequencies.

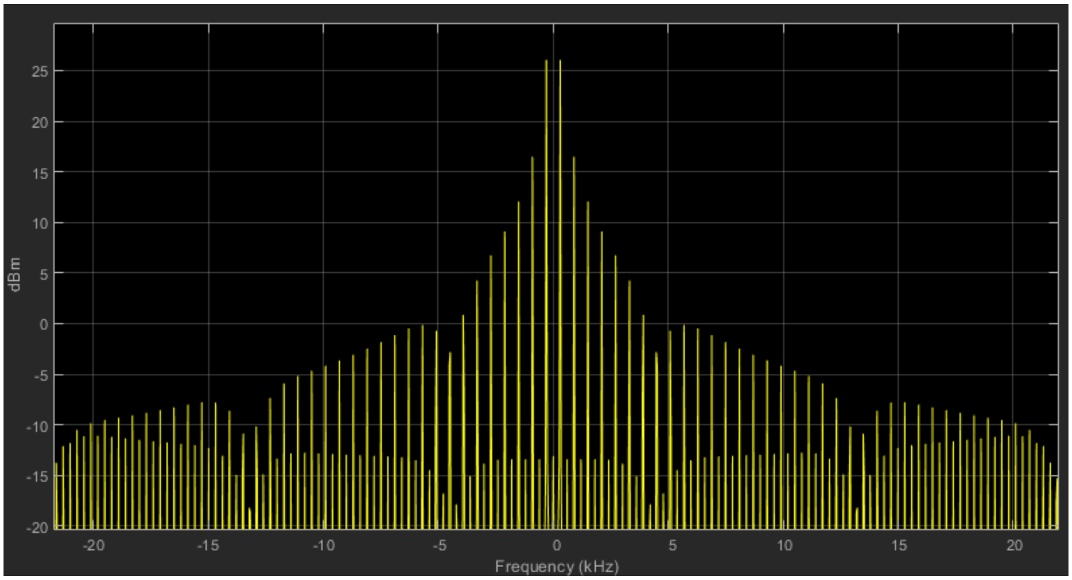


Figure 7 - Frequency spectrum of a 300 Hz square wave

From what we learned in Part A, we can consider each of these peaks to represent a sinusoidal wave with particular frequency f_i and amplitude a_i (sine) or b_i (cosine). Following this logic, we can **reconstruct or approximate any signal *f(t)* by a sum of sine waves of different frequencies and amplitudes**; this is known as *Fourier synthesis*. As a result, any signal *f(t)* can be represented in time *t* by a sum of an infinite number of sinusoidal waves at different frequencies:

$$f(t) = \sum_{i=0}^{\infty} a_i \sin(2\pi f_i t) + b_i \cos(2\pi f_i t)$$

An approximation of a square wave in the time domain for different numbers of summed sine waves of different frequencies is shown in Figure 8.

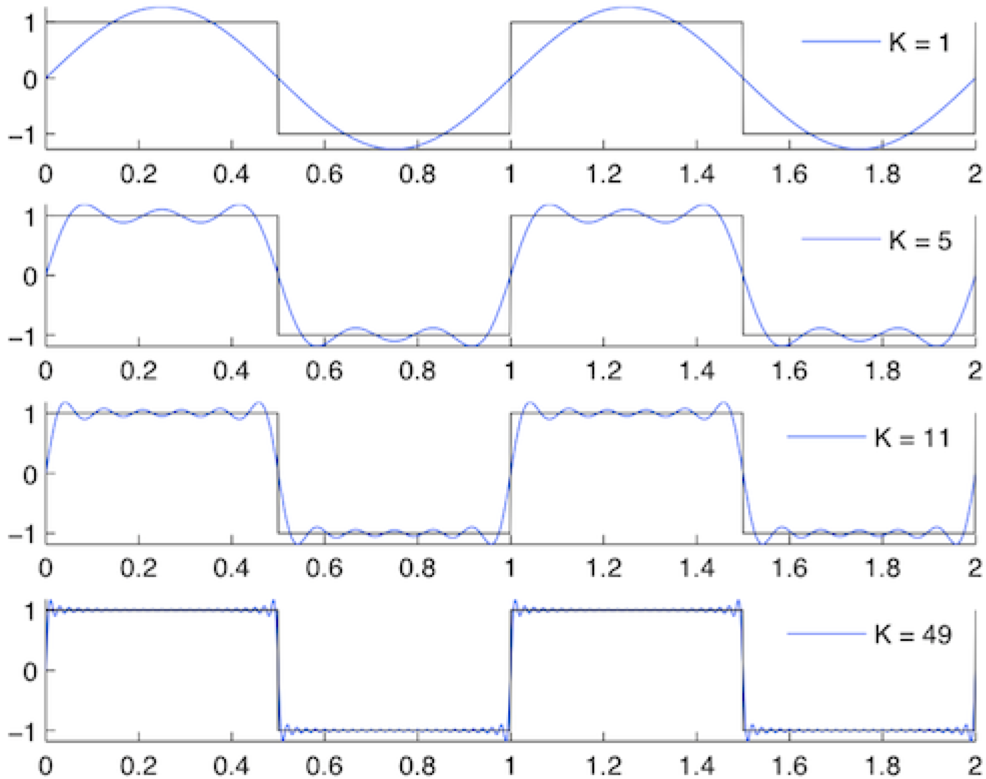


Figure 8 - Fourier synthesis of a square wave using different number of sine waves (K)
https://cnx.org/contents/d2CEAGW5@15.4:VjQZd9A-@9/Common-Discrete-Fourier-Series

Although a single sine wave (K=1) captures the general qualitative characteristics of a square wave like its frequency and where it crosses zero, it is a poor approximation of the signal overall because it cannot reproduce the sharp corners of the square wave. However, as additional sine waves are added with higher frequencies, the sum of sinusoids begins to better approximate the vertical transitions and corners of the square wave. From this trend we can conclude that signals which contain rapid changes in the time domain contain high frequency components and therefore have peaks at higher frequencies in their frequency spectra. In order for a signal in the time domain to have changes as rapid as the vertical transitions and corners of a square wave, the frequency spectrum must contain very high frequency components; the square wave frequency spectrum in Figure 7 confirms this line of reasoning.

Experiment: Fourier Analysis

In Part B, you will be listening to a few different waveforms that are common in electrical engineering (and audio engineering), comparing their time domain and frequency domain properties, and approximating one of them via a sum of sinusoidal waves.

1. Create a blank Simulink model and add the following blocks:
    a. Audio Oscillator (located in Audio Toolbox Sources) – this will be your signal source. Use the following settings to generate a 250 Hz sine wave with an amplitude of 0.1V:

i. Signal type: sine
ii. Frequency: 250 (Hz)
iii. Amplitude: 0.1 (V)
iv. Samples per frame: 512
v. Sample rate: 44100 Hz

b. Buffer block. Set output buffer size to 1024.
c. Time scope block.
d. Spectrum analyzer block.
e. Audio device writer block.

2. Connect the blocks as shown in Figure 9. Label the wire coming from the audio oscillator "sine". Set the stop time to 2, then run the simulation to hear the sine wave and see the time domain plot.

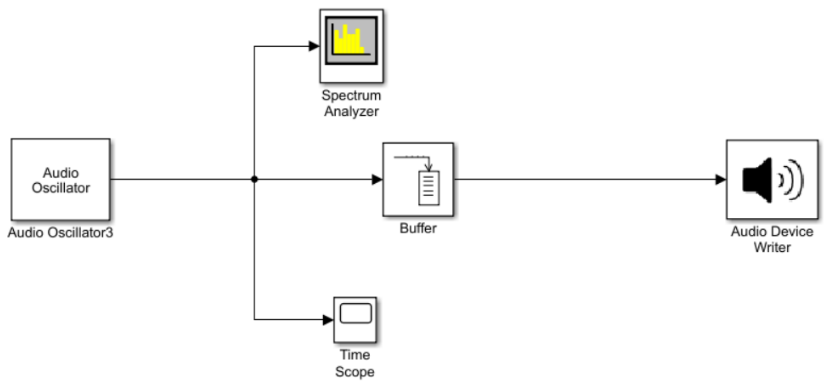


Figure 9 – Simulink model schematic for listening to and viewing a signal in the time and frequency domains

3. Copy the audio oscillator block, place it below the existing one, and set the signal type to "square". Label the wire coming from the oscillator "square". Connect the square wave audio oscillator block to the spectrum analyzer and time scope. Delete the wire between the sine wave audio oscillator and the buffer and feed the square wave audio oscillator to the buffer so that it will play through the audio device writer when you run the simulation. *Note*: you may want to turn down the volume on the computer before playing the square wave – it's much louder than the sine wave. Other than the volume, how does it sound compared to the sine wave? How is the time domain plot different? What are the differences in the frequency domain?

4. Repeat Step 3, but this time add an audio oscillator with a "sawtooth" signal type. How does the sound of the sawtooth wave compare with the sine and square waves? How are these signals different in the time and frequency domain? If all three of these signals have the same frequency, why do they qualitatively sound different from each other?

Experiment: Fourier Synthesis

Now that we've analyzed a signal in terms of its qualitative sound and frequency spectrum, we'll approximate a sawtooth wave with a sum of sinusoids of different frequencies and amplitudes in Simulink.

5. Make a copy of the Simulink model from Step 4. Once opened, delete the "sine" and "square" wave audio oscillator sources. Add a gain block between the audio oscillator and the buffer, then set the gain to "-1". This inverts the signal, which is necessary for comparing the approximate signal with the actual sawtooth wave. Label the wire exiting the gain block "sawtooth", so that the curve is labeled in the time scope and spectrum analyzer. Set the stop time to 2. The model diagram for this step is shown in Figure 10.

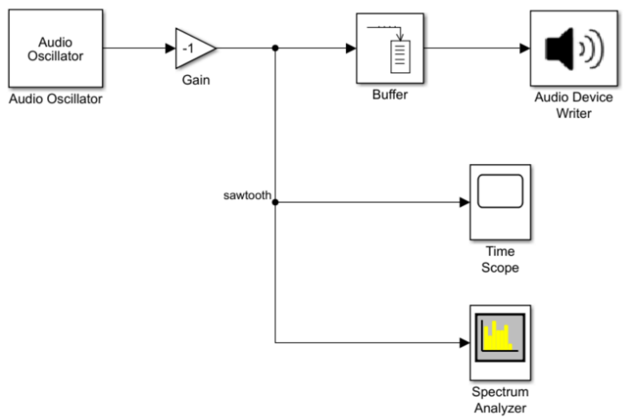


Figure 10 – Simulink model for analyzing a sawtooth wave in the time and frequency domains

6. Run the simulation and view the signal in the time domain. Verify that the frequency of the signal in the time scope matches the frequency entered in the audio oscillator block.

7. On the spectrum analyzer plot, determine the frequencies and magnitudes of the largest 10 peaks with frequencies greater than 0 Hz and fill out Table 1 below. Round your frequencies to the nearest 10 Hz. If your peak frequencies are not close to a multiple of 50 Hz, adjust the span in the spectrum settings to increase the frequency resolution of the spectrum analyzer.

Peak Number	Frequency (Hz)	Magnitude (dBm)
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

Table 1 – Table of frequencies and magnitudes of the 10 largest, positive frequency peaks

8. How are the frequencies of the peaks related to the frequency of the sawtooth wave you entered in the audio oscillator block? Which peak is the largest? Why do you think that is?
9. In your Simulink model, add a DSP sine wave block and connect its output to the time scope and spectrum analyzer so that it is plotted along with the original sawtooth wave. Disconnect the original sawtooth wave from the buffer and connect the new sine wave source to the buffer instead so that you can hear it when you run the simulation. Label the wire “1 sine wave”. Input the following settings for the sine wave source:
    i. Amplitude: 0.1V
    ii. Frequency: the frequency of Peak #1 from Table 1
    iii. Sample time: 1/44100
    iv. Samples per frame: 512
10. Run the simulation. Open the time scope and zoom in until you can see 3-5 periods of the sawtooth wave and sine wave. To see the labels for the curves, open the “View” menu in the time scope and click on “Legend”. Does the frequency of the sine wave match the frequency of the sawtooth wave? Does the amplitude match?
11. Open the spectrum analyzer and zoom in on the lowest frequency peak. Do the magnitudes of the peaks of the sawtooth and sine wave match? If not, adjust the amplitude of the sine wave in the block settings until the magnitudes of the two peaks match (within about 0.25 dBm is close enough). How well does a single sinusoid approximate the sawtooth wave in the time domain?
12. Add a sum block to the model and connect its output to the buffer (disconnect the single sine wave), time scope, and spectrum analyzer. Label the wire “5 sine waves”. Open the sum block and add more inputs: this is done by adding additional “+” signs in the “List of signs” field until there are 5 “+” signs in total. Connect the output of the sine wave source to one of the inputs of the sum block. Copy the sine wave block and paste a new one below the existing one. Set the frequency of the new

sine wave to the frequency you identified as having the 2nd lowest frequency. Your model should be connected as in Figure 11.

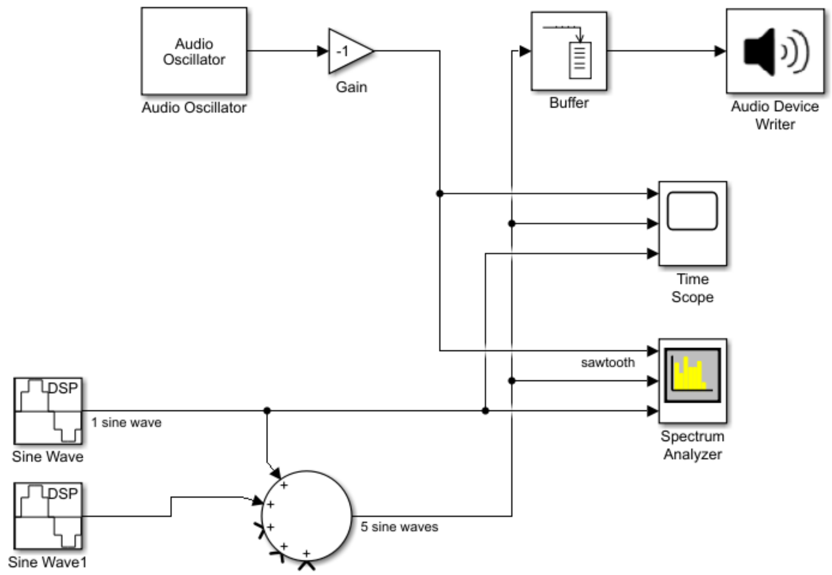


Figure 11 - Simulink model comparing a sum of 1 and 2 sinusoidal waves to a sawtooth wave

13. Run the simulation. How has the sound of the signal changed? As with the first sine wave, adjust the amplitude of the 2nd sine wave so that its peak magnitude in the frequency analyzer matches that of the original sawtooth wave. Is this sum of two sinusoids a better approximation of the sawtooth wave than a single sinusoid?
14. Repeat this procedure for a 3rd, 4th, and 5th sine wave with frequencies equal to the locations of the 3rd, 4th, and 5th lowest frequency peaks you identified. For these peaks, it's fine if the magnitude differs by up to 1 dBm compared with the sawtooth wave.
15. After adjusting the peak magnitudes of all the sine waves, look at the time scope. How does the time domain signal evolve as you add more sinusoids? How does the sound change as you add higher-frequency sinusoids? Is this a better approximation of the sawtooth wave in terms of the time-domain plot and sound?
16. Add another sum block to the model with 6 inputs, connect its output to the time scope, spectrum analyzer, and buffer (disconnect the existing buffer input) and feed the output of the first sum block into one of its inputs. The rest of the inputs to the sum block will be sine waves with the frequencies of the 6th through 10th peaks from Table 1, with their amplitudes adjusted to match the frequency response peaks of the sawtooth wave. Your model should look like the diagram in Figure 12. Again, how does the sound evolve as more sine waves are added? How does the approximated sawtooth wave sound compared to the original sawtooth wave? Do 10 sinusoidal waves provide a better approximation to the sawtooth wave in the time domain than 5? How many sinusoids would it take to accurately reconstruct of the sawtooth wave in sound and in the time domain?

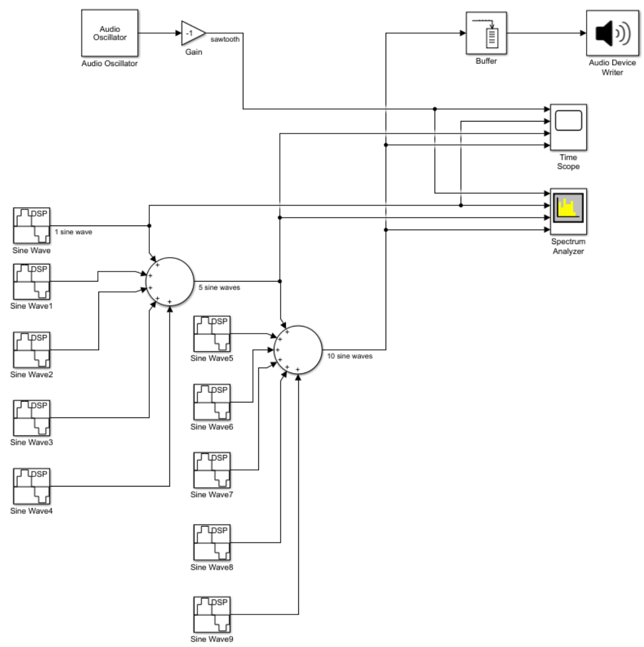


Figure 12 – Simulink model comparing a sum of 1, 5, and 10 sinusoidal waves to a sawtooth wave

Part B: Proof of Concepts List

2. **Prove the concept of Fourier Analysis by determining which two basic types of signals (and their frequencies) were summed to create this mystery signal: Mystery Signal Audio File. Use the block called "From Multimedia File" in the Audio Toolbox to use the audio file as a source in your model.** *Since we haven't taught you the Mathematical Analysis required for Fourier Analysis, simply label which parts of the frequency spectrum correspond to the basic signals you have identified, state their frequencies, and explain your reasoning. Again there is no circuit experiment for this proof of concept. Make sure to limit your simulation time to the length of the mystery signal (1 second).*

3. **Prove the concept of Fourier Synthesis by reconstructing the mystery signal in the time domain.** *For the Mathematical Analysis step, write down the mathematical expression for the mystery signal. Again, there is no circuit experiment for this proof of concept.*

PART C [Core] – Digital and Analog Filters

Material covered: Impedance of Capacitors and Inductors, Transfer Functions of Filters, Low-Pass Filters, High-Pass Filters

In Part B, you learned that non-sinusoidal signals are comprised of many frequencies, they can be approximated by a sum of sinusoidal signals of different frequencies, and that the frequency content of the signal determines how it sounds. In Part C you will learn that the properties of signals can be modified by filters, which attenuate, or remove power, from a signal at certain frequencies. Filters come in two main types: digital and analog, depending on the domain in which the signal exists. In Simulink, you have been generating and manipulating digital signals, whose amplitudes can only take on certain discrete values; these signals can be filtered by digital filters, which are programmed in a computer. On the other hand, analog signals must be manipulated by analog filters, which are comprised of familiar circuit elements like resistors, capacitors, and inductors.

Background: Basic Digital Filters

In the digital domain, filters are simply a transfer function in the frequency domain, that when multiplied with the frequency response of a signal, attenuate the signal at certain frequencies. If X($j\omega$) is the frequency spectrum of the input signal and H($j\omega$) is the transfer function of the filter in terms of frequency, the resulting signal in the frequency domain $Y(j\omega)$ is given by:

$$Y(j\omega) = X(j\omega)H(j\omega).$$

For example, an ideal low-pass filter leaves the signal unaffected below a particular frequency f_c, the corner frequency, but removes power in the signal above that frequency, giving the transfer function:

$$H(j\omega) = \begin{cases} 1 \text{ for } f < f_c \\ 0 \text{ for } f \geq f_c \end{cases}$$

thus "passing" low frequencies and rejecting high frequencies, as shown in Figure 13.

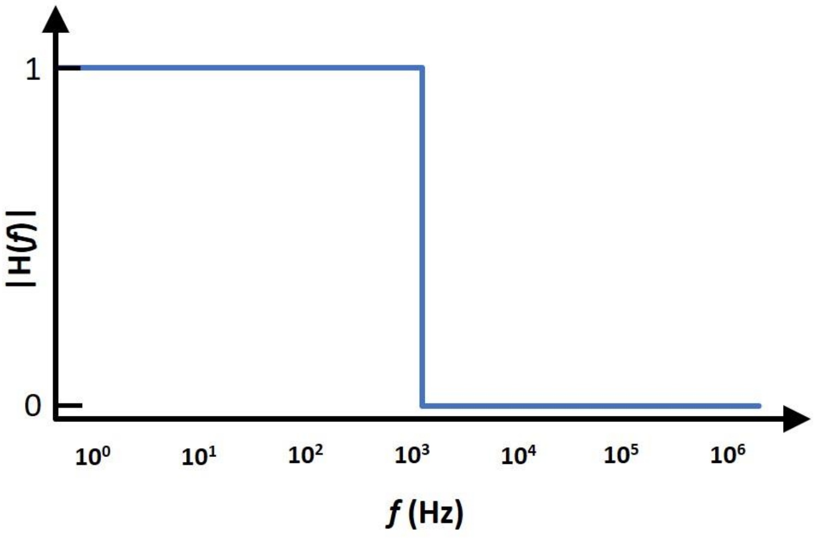


Figure 13 – Magnitude of the transfer function of an ideal, digital low-pass filter

Since creating filters in the digital domain only requires the definition of a function in the frequency domain, almost anything is possible! Accordingly, digital filters are a flexible and powerful tool for processing digital signals, such as digital audio. For example, a digital equalizer is a fundamental tool in music production that enables the manipulation or shaping of the frequency response of an audio signal by applying multiple filters to it. Figure 14 shows an equalizer in the digital audio workstation (DAW) program Reaper, configured to use 6 different filters to shape the sound of a guitar track.

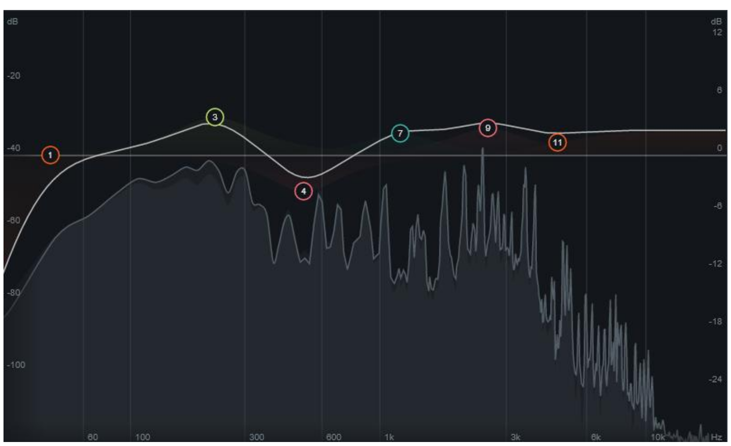


Figure 14 – Equalizer tool in the DAW Reaper using six filters (denoted by circles) to modify the audio signal from a guitar

Although the possibilities for digital filters are endless and there are certainly many varieties of filters, they are all based on four fundamental filter types: what are they? In Figure 14, what type of filters do you see (each encircled number represents a filter)?

Experiment: Digital Filters in Simulink

1. Make a copy of the model file from end of Part B, in which the sawtooth wave was compared with a sum of 10 sine waves. You will be using it in this part of the lab to experiment with digital filters.

2. Open the model and add a “Highpass Filter” block after the output of the 2nd sum block and connect the output of the filter block to the spectrum analyzer, time scope, and buffer; label the wire so that you can identify it in the plot legends. Delete the “1 sine wave” and “5 sine waves” wires that feed into the spectrum analyzer and time scope and delete the empty ports on each of the scopes. Open the high pass filter block and set the “Stopband edge frequency” to 1100Hz and the “Passband edge frequency” to 1300Hz. The model should look like the diagram in Figure 15. In the settings for the filter block, view the frequency response of the filter. Does the transfer function look like you expected? What frequencies of the sum of 10 sine waves should it filter out?

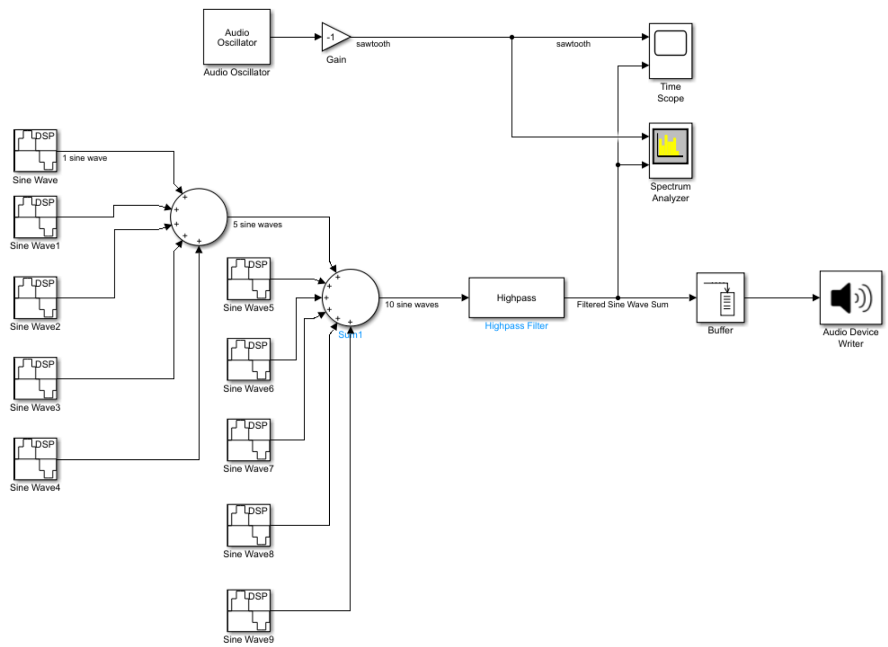


Figure 15 – Simulink model for Part C. A high pass filter applied to the sum of 10 sine waves from Part B.

3. Run the simulation. How does the filtered signal sound compared to the sum of 10 sine waves? Look at spectrum analyzer – did the filter remove the frequencies you expected? How did the filter change the signal in the time domain? Why?

4. Replace the “Highpass Filter” block with a “Lowpass Filter” block and set “Passband edge frequency” to 1100 Hz and the “Stopband edge frequency” to 1300 Hz. Repeat step 3 and answer the same questions.

5. Copy the “Lowpass Filter” block and use it to low-pass filter the sawtooth wave as well (label the wire appropriately), as in Figure 16. Compare the low-pass filtered sum of sine waves to the low-pass filtered sawtooth wave in the time scope. How good of an approximation is the sum of 5 sine waves to the low-pass filtered sawtooth wave?

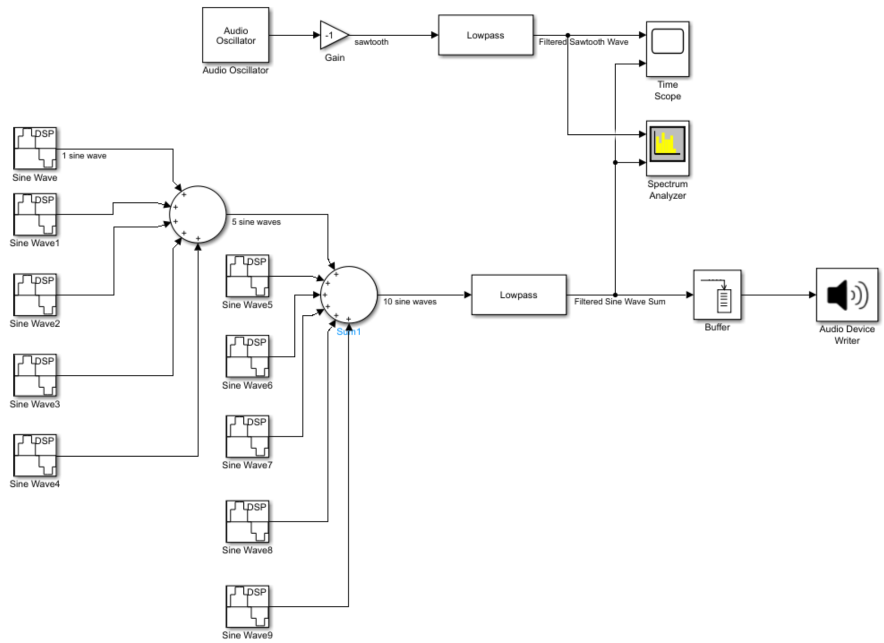
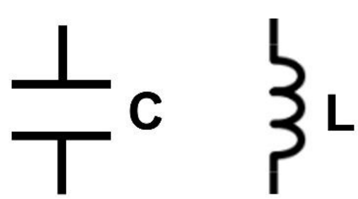


Figure 16 – Simulink model for low-pass filtering the sum of sine waves and sawtooth wave

6. Open the Simulink Library and navigate to DSP System Toolbox -> Filtering -> Filter Designs. Choose a filter in the list that isn't one of the four basic filter types. What does this filter do? What is it used for?

Background: Analog Filters - Impedance of Capacitors and Inductors

So far, you have learned that resistors are circuit elements which are characterized by a linear relationship between voltage and current, called resistance. However, there are two other fundamental circuit elements which are characterized by a linear relationship between voltage and current (at a given frequency): capacitors (C) and inductors (L). Unlike resistors though, this relationship between voltage and current for capacitors and inductors is expressed in terms of complex numbers; this generalized resistance is called *impedance* and is represented by the letter *Z*. The equivalent of Ohm's law for capacitors and inductors is then V = IZ. The circuit symbols and impedances of the capacitor and inductor are shown in Figure 17:



$$Z_C = \frac{1}{j\omega C} \qquad Z_L = j\omega L$$

Figure 17 – Circuit symbols and impedances of the capacitor and inductor

Although the impedances are not a function of voltage or current, they are a function of the signal frequency, $\omega=2\pi f$. Circuit components that have a different resistance to current flow at different frequencies allow us to make circuits that permit or reject certain frequencies in a signal: these circuits are called *analog filters*. Although analog filters must be realized with physical components whose tolerances and non-idealities can introduce error into the filter design, analog filters are often cheaper and simpler to implement, as no external computer circuitry is required to implement the filter, and they do not cause large delays in the signal output, since the signal does not need to be processed by a computer.

Background: Analog Low-Pass Filter

To demonstrate how this frequency-dependent impedance plays a role in modifying signals, let's look at a simple circuit consisting of a voltage source, resistor, and capacitor:

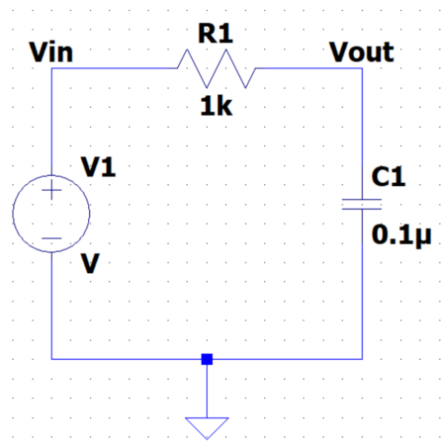


Figure 18 – Circuit diagram of an RC low-pass filter

To understand what the circuit in Figure 18 does, we first need to know how V_{out} is related to V_{in}, also known as the transfer function $H(j\omega)$. V_{out} and H can be derived with the help of a voltage divider:

$$V_{out} = V_{in}\frac{Z_C}{Z_R+Z_C} \rightarrow H = \frac{V_{out}}{V_{in}} = \frac{Z_C}{Z_R+Z_C} = \frac{\frac{1}{j\omega C_1}}{R_1+\frac{1}{j\omega C_1}} = \frac{1}{j\omega C_1 R_1+1}$$

Now that we have the transfer function, we can analyze the behavior of this circuit as a function of frequency. What happens to H when ω is small (aka close to zero)? H is very close to 1. Why? If we look at the circuit, when ω is very small, Z_c, the resistance of the capacitor to current flow is very large, which means that the capacitor looks like an open circuit, as shown in Figure 19a.

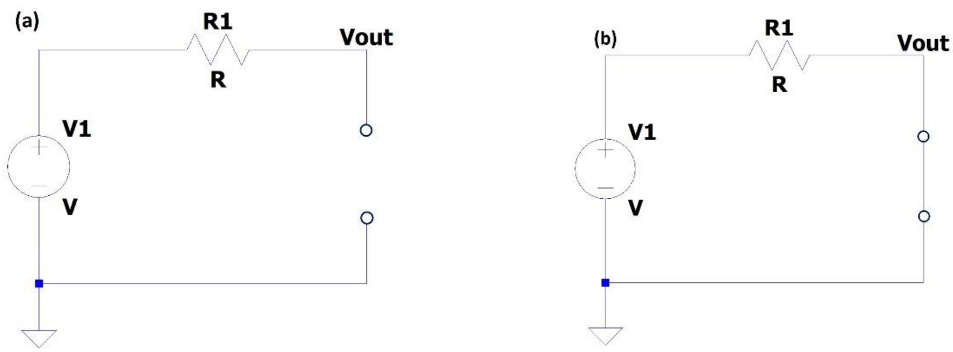


Figure 19 – Representation of a (a) low-pass RC filter at low frequencies and a (b) low-pass RC filter at high frequencies

If Z_C is very large compared with R_1, we know from working with voltage dividers that almost all of V_{in} will be dropped across Z_C, leading to $V_{out}/V_{in} \approx 1$. At very high frequencies, the denominator of the transfer function becomes large and forces the magnitude of the transfer function to become very small. Looking at the filter circuit, the impedance of the capacitor at high frequencies is very small, meaning that (again via voltage dividers) very little voltage will be dropped across the capacitor and V_{out} is very small, as in Figure 19b. If we assume that V_{out} transitions gradually from V_{in} to 0, we get the plot of the magnitude of the transfer function for this circuit, as shown in Figure 20: a low-pass filter.

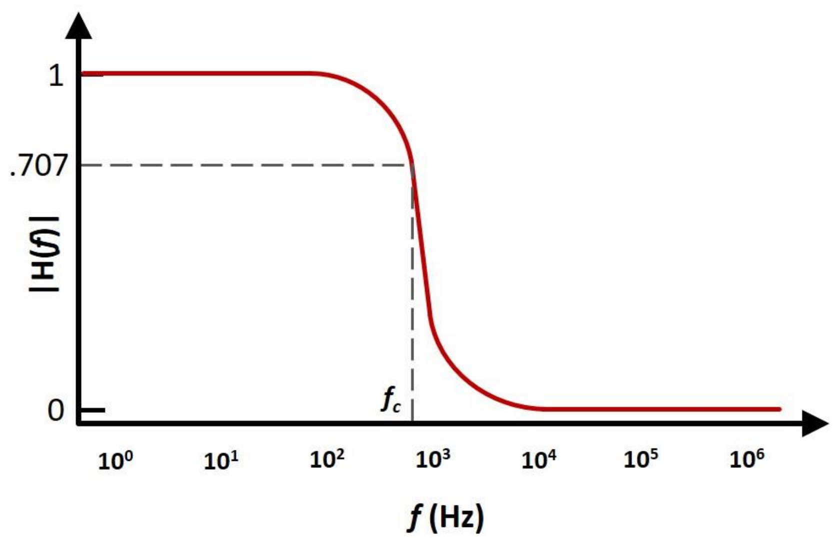


Figure 20 – Magnitude of the transfer function of an analog low-pass filter

Even though we plotted the transfer function on a semilogarithmic scale here, transfer functions for filters are often plotted on a log-log scale, where both axes are logarithmic and the y-axis is given in units called "decibels"; these types of plots are called Bode plots.

Although a digital filter can be designed to have the sharp frequency response shown in Figure 13, analog filters have a more gradual transition from the passband to the stopband. The point at which the transfer function reaches $1/\sqrt{2} = 0.707$ is an important feature of a filter called the *corner frequency* $\omega_c = 2\pi f_c$. Mathematically, it is the frequency at which the power transfer function reaches ½. In a more practical sense, it is the frequency at which the filter begins to significantly attenuate the signal. For a first-order RC circuit, $\omega_c = \frac{1}{RC}$, while for a first-order LR circuit, $\omega_c = \frac{R}{L}$.

Experiment: Analog Low-Pass Filter

You will be building a low-pass filter on your protoboard using a resistor and a capacitor for this part of the experiment and then verifying its function using the M2K or Analog Discovery 2.

Analysis: RC Low-Pass Filter

1. Draw the circuit diagram for the RC low-pass filter in Figure 18, without specific component values.
2. Choose values of R and C from your kit that will yield a filter with a corner frequency f_c as close to 10kHz as possible.
3. Derive the transfer function for the filter and find the magnitude of the transfer function.
4. Sketch the transfer function as a function of frequency by choosing at least 5 points in frequency and calculating the magnitude of the transfer function at those points. One of these points must be f_c, two must be frequencies near f_c (one above and one below), and two must be far from f_c (one above and one below).

Simulation: RC Low-Pass Filter

5. Build the circuit of the RC low-pass filter in LTspice using the component values you determined above. Set the voltage source to have an AC amplitude of 500mV under Small Signal AC analysis.
6. Run an AC sweep of type “decade” from 1 Hz to 1 MHz (remember: megahertz is “Meg” in LTspice) with 100 points per decade. Plot both the input and output voltage.
7. Right click on the y-axis on the left side of the plot and change the scale to linear.
8. Add a trace to the plot and plot V_{out}/V_{in} to calculate the transfer function. From the plot, find f_c for the filter – does this match what you calculated above?
9. Do your hand-calculated values of the magnitude of the transfer function from the analysis section agree with the transfer function in LTspice?

Experiment: RC Low-Pass Filter

10. Build the filter circuit on your protoboard.
11. Use your instrumentation board to both supply a 1 V_{pp} sine wave to the circuit as the input and measure the output voltage across the capacitor.
12. Find f_c for your filter by varying the frequency of the sinusoidal waveform until you find the frequency at which $V_{out}/V_{in} = 0.707$. Note: be sure to use the measured value of V_{in} for this calculation as it may change with frequency (i.e., even though your instrumentation board is set to supply 1 V_{pp}, it may *actually* supply less voltage at higher frequencies).

13. Measure V_{out}/V_{in} at the same 5 frequencies as you used to sketch the transfer function. What happens to the amplitude of V_{out} as frequency increases? How do the measured values of V_{out}/V_{in} compare with what you calculated and simulated? What could be the source of any differences you observe?
14. Instead of measuring V_{out} across the capacitor, measure across the resistor instead. Repeat the measurements of V_{out}/V_{in} at the same 5 frequencies as above. What kind of filter is this? Use KVL to explain why measuring the output voltage across the resistor gives this particular type of filter, as compared with when you measure the output voltage across the capacitor.

Part C: Proof of Concepts List

4. **Prove the concept of an analog high-pass filter using an inductor and a resistor. How could you make a low-pass filter using the same components?**

PART D [Alpha/Omega] – Applications: Signal Processing

1. Choose an application below in either Alpha Experiment or Omega Exploration! (You will find opportunities to filter a noisy signal in Simulink, isolate or eliminate a particular instrument in a song in Simulink or Audacity, create your own n-band equalizer in LTspice [analog] or Simulink [digital], simulate a communications system in LTspice [analog] or Simulink/Matlab [digital], create your own audio synthesizer in LTspice [analog] or Simulink [digital], create a song using Matlab or Simulink, etc.)
2. Draw a block diagram of your system with inputs and outputs to show how the signal will be affected by each system element.
3. Define the functional details of each block, such as signal source waveform types and frequencies and filter frequency responses. Additionally, sketch the frequency spectrum of the input and expected output signal(s).
4. Implement your application in Matlab, Simulink, Audacity (or other audio processing software) [digital applications] or LTspice [analog applications]. For analog applications, you may also implement your application with circuit components on a protoboard, as long as you are able to test your circuit's functionality.

Choose from the Alpha Experiment or Omega Exploration options on the next page!

Choose Your Adventure! Alpha Experiments or Omega Exploration

Alpha Experiments

Everyday Signals, Music, and Filters

Test out the filter response Signal Viewer and Audio Player (created by Prof. Rich Radke!) to see how an IDEAL filter works and how common, everyday signals are made. Video of Prof. Radke explanation/playing with it

Link to MATLAB player and instructions

1. Find and isolate frequencies in a common everyday sounds like dial tone or number for your phone.

2. Recreate this sound using Simulink by combining frequencies and play through your computer.

3. Can you make a song or specific note? Identify the note and show the overtones.

4. Use your Favorite Song and isolate frequency ranges to modify it in some way.

5. Review ANY portion of the mathematical description in Prof. Radke's video. Find a term you don't fully understand that would help you create signals and music. Identify ECSE courses that will help you understand it!

6. OPTIONAL: Build your application (hardware personal instrumentation and components in your parts kit) and demonstrate its function. Compare.

Omega Exploration

Explore applications of signals and filters in communications and music in **one** of the following:

- **Decode the Hidden Message** – design a system in Simulink that isolates and amplifies the message hidden in the noisy signal.
- **Modify your Favorite Song using Filters** – choose and design filters that will isolate or eliminate certain instruments in your favorite song in Simulink or a DAW.
- **Software Synthesizer** – design audio waveforms in Simulink by applying filters and other signal modification methods.
- **Analog Communication System** – design a communication system in Simulink, including filtering, modulation and demodulation.

Technical and functional requirements for your application are coming soon!

Part D: Proof of Concepts List

5. ***Alpha and Omega*: Prove the concept of using Fourier Analysis to associate audible features of an audio signal with specific frequency ranges in its frequency spectrum**

   OR

   ***Alpha and Omega*: Prove the concept of using Fourier Synthesis to design audio signals**

6. ***Alpha and Omega*: Prove the concept of filtering out or isolating a particular range of frequencies of a signal.**

Omega Exploration Application Requirements

A short description and the technical specifications for each Omega Exploration option are listed below.

Decode the Hidden Message

Filtering can be used to separate wanted information (the "message") from unwanted information (the noise) in a signal. In this Omega Exploration, you will try to decode the message hidden in a very noisy audio clip by designing and applying filters in Simulink to eliminate the noise. No physical circuit is required for this project – your simulation counts as your experiment in this case.

A .wav file of the noisy signal can be downloaded here. *Note*: turn your volume down before opening it!

Requirements:

- ☐ *System architecture*: in Simulink
  - You must filter the signal to isolate the message and will likely need to use multiple filters. In each case where you've chosen a filter design, discuss at least one other filter type that could have performed the same function and justify why you chose one filter type over the other.
  - You must also include one amplifier to boost the signal volume.
- ☐ *Output signal*:
  - You must display both your original signal and filtered signal in the frequency domain and time domain.
  - You must play your filtered signal through speakers.
- ☐ *Experimental Verification / Test Cases*:
  - Show your original signal and filtered signal in both the frequency domain and time domain.
  - Play your filtered signal through speakers (include in your presentation video).
  - Decode the message hidden in the signal. *Hint*: what are different ways of encoding information via taps or dots and dashes?

Modify your Favorite Song using Filters

Active filters (filters that can also amplify certain parts of a signal) are used extensively during the music production process to shape the sound to meet an artist's aesthetic goals. This stage of music production in which different audio tracks are adjusted so that they can later be combined together into a single, final signal (or track) is called mixing.

On the other hand, once a song is finished, you can also use filters to try to single out or eliminate individual instruments. In this Omega Exploration, you will design filters to change the sound of a chosen song, and isolate (keep) and eliminate (remove) individual instruments from it. While you can accomplish this in Simulink, many programs are much more convenient for audio editing.

Some free options for audio editing software include digital audio workstations (DAWs) such as Reaper and Waveform Free. Other programs, such as Audacity, provide many tools for audio editing, but offer fewer conveniences than DAWs for manipulating music tracks.

No physical circuit is required for this project – your simulation counts as your experiment in this case.

Requirements:

- ☐ *System architecture*: in Simulink or digital audio workstation (DAW)
  - You must apply filters to modify your chosen song. In each case where you've chosen a filter design, discuss at least one other filter type that could have performed the same function and justify why you chose one filter type over the other. *Note*: in audio engineering, there are multiple types of high pass, low pass, band pass, and band stop filters.
  - You must use at least three different types of filters.
- ☐ *Output signal*:
  - You must display both your original and filtered signals in the frequency domain.
  - You must play your filtered signal through speakers to hear the audible difference between the original and filtered version of your chosen song.
- ☐ *Experimental Verification / Test Cases*:
  - You must list the frequency ranges corresponding to each of the instruments you are trying to isolate or eliminate.
  - You must try to isolate a single instrument (and filter out all others).
  - You must try to eliminate a single instrument.
  - You must try to isolate **OR** eliminate **two** instruments.
  - In each case, you must show your original signal and the filtered signal in the frequency domain for one point in time when the difference is clear.
  - In each case, you must play your original signal and the filtered signal through speakers to verify that your filter designs have achieved their audio goals (include this in your presentation video).

Software Synthesizer

A synthesizer is a musical instrument that can produce a wide variety of sounds by modifying and combining simple waveforms, such as sine, square, sawtooth, and triangle waves. Although the most basic function of a synthesizer is to produce a waveform at with specific fundamental frequency (musical note), synthesizers also use a variety of tools, such as filters, envelopes, and modulators, to modify the character of the waveform's sound. In this Omega Exploration, you will design your own signals by experimenting with combining waveforms and applying filters, envelopes, and modulators. No physical circuit is required for this project – your simulation counts as your experiment in this case.

Requirements:

- ☐ *System architecture*: in Simulink
  - Input stage: blocks that generate a sine, square or triangle wave
  - Signal modification stages must include
    - Filter(s). In each case where you use a filter, justify why you chose that filter to achieve your goals instead of another filter.
    - At least one of the following special effects:
      - Amplitude modulator (you may not use Communication Toolbox blocks specifically for AM – you must do some math to implement AM yourself!)
      - Envelope generator (you don't have to follow the ADSR model, but you should show that you can control the shape of the envelope to some extent)
      - Frequency modulator (you may not use Communication Toolbox blocks specifically for FM – you must do some math to implement FM yourself!)
  - Output stage:
    - "To Multimedia File" block – to save the signal as an audio file
    - Spectrum Analyzer
- ☐ *Output signal*:
  - Must play resulting waveforms through speakers
  - Must display the waveform after the modification stage and after the filter stage
- ☐ *Experimental Verification / Test Cases*:
  - You must design "instruments" with the following characteristics using your synthesizer. In each case, you should describe the initial idea for the sound you wanted and the design choices you made to try to produce that sound.
    - Bass instrument: primarily low/bass frequencies
    - Rhythm instrument: primarily middle frequencies
    - Lead instrument: primarily high/treble frequencies
  - Each of your designed instruments must include a filter
  - One of your designed instruments must be sum of at least **two** signals
  - One of your designed instruments must use the chosen special effect in your synthesizer
  - You must play samples of each of your designed instruments through speakers
  - You must show the frequency spectrum of each of your designed instruments and explain how it corresponds to the sound of that instrument

Analog Communication System

Communication systems transfer information from once place to another by sending and receiving electromagnetic signals. Each of these systems consists of multiple stages meant to prepare the information, or signal, for transmission from one place to another. For example, in order to send a signal wirelessly, a process called modulation is required, which "encodes" the data onto a wave which can travel long distances through the atmosphere. The signal is then sent through the air via an antenna and can be picked up by a receiver. Once a receiver picks up the signal, it must be demodulated to recover the original signal. In this Omega Exploration, you will design a basic communication system in Simulink and send an audio signal of your choice through it. No physical circuit is required for this project – your simulation counts as your experiment in this case.

Requirements:

- ☐ *System architecture*: in Simulink
    - Input stage: audio file
    - You must include the following stages in your system
        - Modulation stage – amplitude modulation (AM) **OR** frequency modulation (FM).
        - Added white Gaussian noise (AWGN) channel block – simulates noise added to the signal between the transmitter and receiver. *Note*: you will need the Communications Toolbox for this block.
        - Demodulation stage
        - Filtering stage
        - *Note*: you may not use Communications Toolbox blocks specifically for AM or FM in your modulation or demodulation stages – you must implement them yourself!
    - Output stage:
        - "To Multimedia File" block – to save the signal as an audio file
        - Spectrum Analyzer and Time Scope
- ☐ *Operating conditions*
    - AWGN channel block settings → mode: SNR; SNR (dB): 0; Input Signal Power: ≥10 mW
- ☐ *Output signal*:
    - Must play resulting signal through speakers
    - Must display the signal in the frequency and time domains
- ☐ *Experimental Verification / Test Cases*:
    - Your input audio signal must pass through a modulation stage, an AWGN channel block, a demodulation stage, and a filter stage.
    - You must display the signal in the frequency and time domains after each stage.
    - You must compare the audio quality of the signal at three stages: at the input, immediately after demodulation and after the final filtering stage.

Note: Due to the Nyquist-Shannon sampling theorem the sampling frequency for all of your signals and spectrum analyzer must be at least twice the maximum frequency of any of your signals. For this reason, you may want to low-pass filter your input signal before modulating it.

SUMMARY of Concepts

Concept List that must be accounted for in your Proof of Concepts

PART A:

1. **Prove the result of summing 3 sinusoidal waves in the time and frequency domains. For the Mathematical Analysis step, you may use Matlab to plot the time domain signal, but you must hand sketch the frequency spectrum. There is no circuit experiment for this concept; proving it via a simulation in Simulink is enough.**

PART B:

2. **Prove the concept of Fourier Analysis by determining which two basic types of signals (and their frequencies) were summed to create this mystery signal. Again, there is no circuit experiment for this proof of concept.**
3. **Prove the concept of Fourier Synthesis by reconstructing the mystery signal in the time domain. For the Mathematical Analysis step, write down the mathematical expression for the mystery signal. Again, there is no circuit experiment for this proof of concept.**

PART C:

4. **Prove the concept of an analog high-pass filter using an inductor and a resistor. How could you make a low-pass filter using the same components?**

PART D:

5. ***Alpha and Omega*: Prove the concept of using Fourier Analysis to associate audible features of an audio signal with specific frequency ranges in its frequency spectrum**

   OR

   ***Alpha and Omega*: Prove the concept of using Fourier Synthesis to design audio signals**
6. ***Alpha and Omega*: Prove the concept of filtering out or isolating a particular range of frequencies of a signal.**

How You'll Be Graded (with Standards Based Assessment):

You will be graded on the following Standards. Please ensure to achieve each standard. If you do not, you can resubmit to the missing standard to the end of the semester. CLEARLY mark the changes you make in you Proof of Concept submission by either Tracking Changes in Word or highlighting changes by writing comments in a different color and/or changing the color of the updated work.

Lab 03 Standards

1. **I can sum three sinusoidal waves in the time and frequency domain.**
2. **I can use the concept of Fourier Analysis to determine the components of a mystery signal.**
3. **I can use the concept of Fourier Synthesis to reconstruct a mystery signal.**
4. **I can create a high pass and low pass filter and find its transfer function.**
5. ***Alpha and Omega*: I can associate audible features of an audio signal to ranges in the frequency spectrum.**

   OR

   ***Alpha and Omega*: I can design audio signals with an understanding of the signal in the frequency spectrum.**
6. ***Alpha and Omega*: I can filter out or isolate particular ranges of a signal with a digital filter in Simulink or other software program.**

Additionally, the following standards apply to proving each of the above concepts:

- **I can demonstrate "good failure" whenever applicable by providing accurate results in my experience and speculating what went wrong.**
- **I can identify non-idealities or unexpected results and attempt to explain why they may exist.**
- **I can answer for myself "Is this right?" by comparing mathematical calculations to simulation and experimental results.**
- **I can show plots and diagrams that are easy to read, scaled correctly and clearly labeled.**
- **I can use consistent variable labels and component values in mathematical calculation, simulation and experimental results for easy comparison.**