Module 5: Biomedical Signals

Engineering Disciplines: Biomedical Engineering, Electrical Engineering

For the Module 5 Part 1 Lab Report, each team will submit:

- 1 .m file containing code for Labs 1 & 2
- 1 PDF file (all in one file) containing the Design Worksheet (completed through part D), typed answers to the Labs 1 & 2 discussion questions, and your Project Planning and Management Task List

For the Module 5 Part 2 Lab Report, each team will submit:

- 1 .m file containing all code for this module
- 1 PDF file (all in one file) containing the fully-completed Design Worksheet, typed answers to the Lab 3 discussion questions, and your final Project Planning and Management Task List

Prepare and submit these files in accordance with the ENGR 130 Style Guide and Assignment Submission Guide.

Overview

The overall goal of this module is to develop a device that measures a user's heartrate. In the first three labs, you will generate a signal using a medical sensor and write code to extract the person's heartrate.

Before you use your code to analyze your user-generated signal, you will test it by analyzing known signals that you generate. In the first two labs you will create signals, then develop the code to analyze the signals. Then, in Lab 3, you will use the sensor to collect data and apply your code to the user-generated signal. In Lab 4, you will design a diagnostic device that detects arrhythmias and alerts a clinician.

Labs 1 & 2: Signal Processing

Introduction

What is a signal?

A signal is a measured quantity as a function of at least one independent variable (e.g., time or space). In other words, a signal is the functional representation of a physical quantity or variable containing information about the behavior of the physical quantity. Recall some of the examples from our guest biomedical engineering lecture.

The Time Domain

Signals in MATLAB can be described by a vector of numbers where each entry stands for the value of the signal at a particular point in time. When represented this way, we refer to the signal as being in the **time domain**. The signals can be treated as any other numbers—they can be added together, multiplied by constants or other values, or transformed to extract information.

Many common signals are combinations of sinusoids. A sinusoid has the following mathematical representation (Figure 1):

$$y(t) = A * sin(2\pi ft + \varphi) \tag{1}$$

where:
A = amplitude
f = frequency (Hz)
t = time (s)
φ = phase (shift in the horizontal direction, i.e., time)

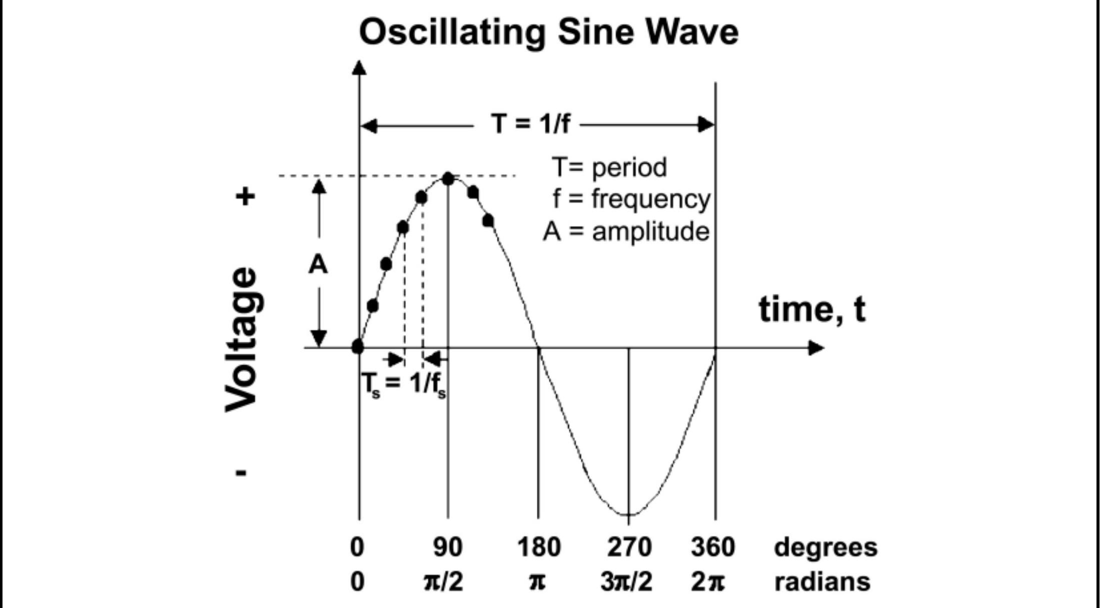

Figure 1: Oscillating sine wave with no phase shift (φ = 0) [source]

Figure 2 is an example of combining two sinusoids together to create a third signal.

The Frequency Domain

The graphs in Figures 1 and 2 show how the signals change in time, and so we say that these are graphs in the time domain. When looking at a signal that is a composite of multiple sinusoids, it can be difficult to tell what the components are with a simple visual inspection. An alternate way of representing this same information is to create a graph in the **frequency domain.** Recall that a signal's frequency is the number of full cycles it goes through each second. Where a time-domain graph displays changes in a signal over time, a frequency-domain graph displays the amplitude or “contribution” of each frequency to the signal (Figure 3). Note that this representation shows both the frequencies of the component sinusoids, as well as the amplitude of each.

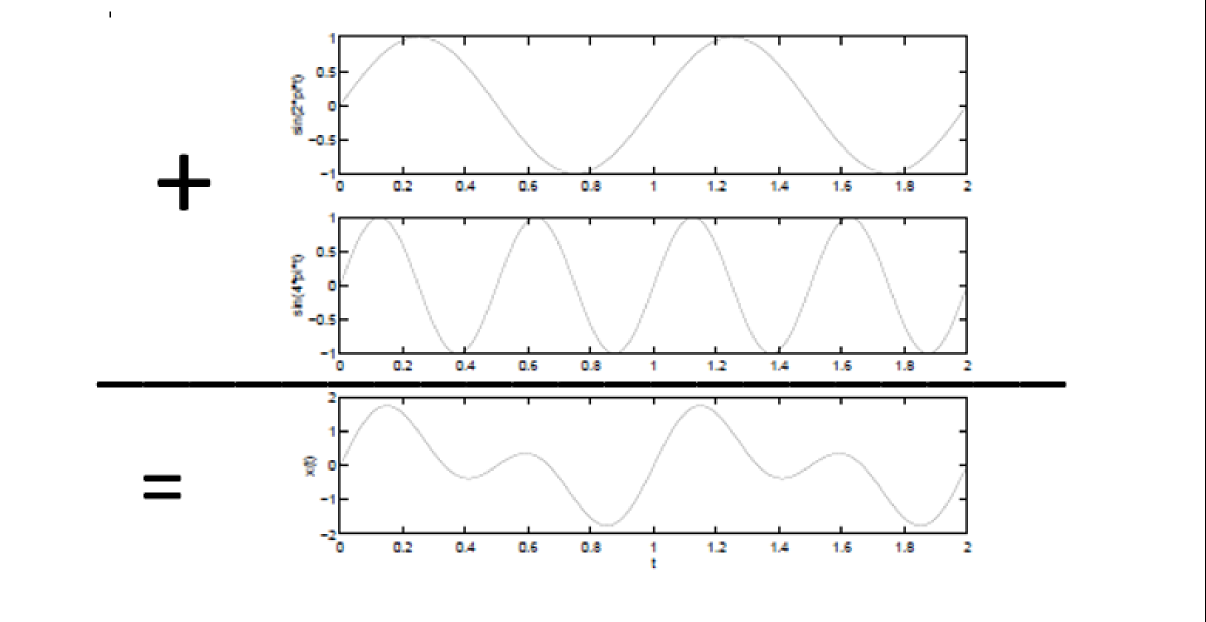

Figure 2: Addition of sinusoidal signals [source]

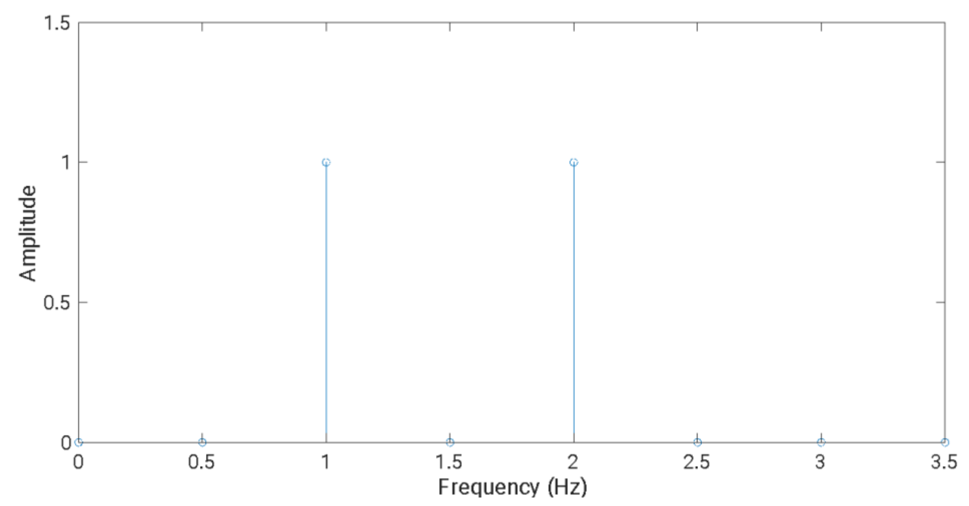

Figure 3: Frequency-Domain Graph of Summed Sinusoidal Signals in Figure 2

Fourier Transform

You saw in Figure 2 that simple sinusoidal signals can be added to form composite signals. In fact, any periodic function, no matter its shape, can be represented by a sum of sines and cosines. This powerful principle is called Fourier analysis and is the foundation of signal processing. The **Fourier transform** is the mathematical method that decomposes a signal into its component sines and cosines; its output is presented in the frequency domain, like in Figure 3. Again, this representation shows the frequency and amplitude of each component sinusoid.

The ENGR130_Fourier.m Function

The mathematics underlying the Fourier transform are beyond the scope of this course. Therefore, we are providing you with the following function: `ENGR130_Fourier.m`. You need to download this function and read its documentation (lines 2-20) to understand what it does. Once you have done this, include it as a user-defined function at the end of your script file for this module. **DO NOT CHANGE THIS FUNCTION.**

Materials

- Computer running MATLAB
- `ENGR130_Fourier.m`
- `mystery_signal.mat`

Procedures

1. Project planning and management

Create your own Project Planning and Management Task List for all tasks in this module, following the instructions in the Project Planning and Management document. You will continue to edit your task list this week and submit your task list, current as of the submission date, as part of your Part I Report.

2. Sinusoidal signal generation

a. Create a variable `dt` that will be the time step and make it equal to 1/1024 seconds. Use `dt` to generate an equally spaced time vector, `time`, such that `time(1) = 0` and `time(1024) = 1 - dt`.
   i. It is critical that `numel(time) = 1024`.
   ii. Use `time` anytime you need a time vector in Labs 1 & 2.

b. Write a function named `getSinusoid` to generate a sinusoidal signal, $y(t) = A * sin(2\pi f t)$. The function should receive `f` (scalar), `A` (scalar), and `t` (vector) as inputs and return sinusoidal signal `y` (vector).
   i. Document this function similarly to `ENGR130_Fourier.m`.
   ii. Remember that all user-defined functions must be included in a single section at the end of your script file.

c. Write a function named `getComposite` for adding multiple sinusoids together to make a composite signal. This function should receive a *vector* of frequencies (`[f1, f2,`

f3,...]), a *vector* of amplitudes ([A1, A2, A3,...]), and a time vector. It should return a *composite* signal generated by summing the sine waves defined by the amplitudes and frequencies, i.e.,
composite = A1*sin(2*pi*f1*t)+A2*sin(2*pi*f2*t)+...

i. Use getSinusoid inside getComposite to get each sine wave.
ii. getComposite should be able to accept amplitude and frequency vectors of any length as long as these two vectors have the same length. Throw an error using the error() function if the length of these vectors differ.
iii. Document getComposite similarly to ENGR130_Fourier.m.

Recommended End Point for Lab 1

d. Use getComposite to make a new composite signal (y_comp) using the amplitude and frequencies listed in Table 1. Plot this signal with appropriate axis labels and a title.

Table 1. Components of Composite Signal

Component	Amplitude	Frequency (Hz)
1	1	12
2	20	17
3	5	20
4	3	5
5	17	15
6	1.5	5

3. Analyze your signal via Fourier transform

a. Use ENGR130_Fourier to perform the Fourier transform of y_comp. Fourier transform amplitudes are represented by the capital letter of the signal's variable. Therefore, the amplitudes of y_comp's Fourier transform should be stored in Y_comp and the frequencies should be in f_comp.

b. With the above step, we have switched from the time domain to the frequency domain. To create a standard Fourier plot, use the stem function to create a plot of Y_comp with frequency on the x-axis and amplitude on the y-axis. Adjust your x-axis limits to see the results clearly and label the graph appropriately. Save this plot to be included in your lab report. Check that your results match the expected frequencies and amplitudes from your constructed signal.

4. Deconstruct a mystery signal

a. To have MATLAB find the major components of a composite signal for you, write a function named getComponentParams to find the major frequency components of the Fourier transform. This function will receive the f and Y outputs from

`ENGR130_Fourier` along with an amplitude threshold, `thresh`, for a component to be considered "major". It should return a 2-column matrix where the first column contains the major frequencies and the second column contains the corresponding amplitudes.

b. Download this mystery signal. The name of the variable in the .mat file is "mystery."

c. Load the mystery signal and, using `ENGR130_Fourier,` take the Fourier transform of the mystery signal using the vector `time`.

d. Create Figure 1, a 2 row, 1 column subplot. In the upper subplot show the mystery signal vs. time. In the bottom subplot show the Fourier transform amplitude vs. frequency using the `stem` function. Adjust your x-axis limits to better show the Fourier transform. Title each plot and include axis labels. Save this plot to be included in your module report.

e. Use `getComponentParams` on the mystery signal to find the frequencies whose amplitudes are greater than 4.

f. Remake the mystery signal using the results of the step above and `getComposite`. Subtract the restored signal from the mystery signal and plot the difference with axis labels and a title. Save this plot to be included in your module report.

Questions

1. Include your plot from Part 3b.
2. Include your plot from Part 4d.
3. Include your plot from Part 4f.
4. Do the values in Table 1 and the values you found from the Fourier transform for `y_comp` agree? Why or why not?
5. Did you find all the mystery signal components? How can you tell?
6. White, brown, and pink noise are oftentimes found in signals, especially audio signals. Research and describe each type. How would each kind of noise appear on `y_comp`'s Fourier transform stem plot?

END OF LABS 1 & 2

Lab 3: Analyzing Heartbeat Signals

Note: this lab is not intended to confer medical advice or diagnose any diseases or conditions. See a medical professional for healthcare needs.

Introduction

Did you know that *you* are composed of signals and systems? Biomedical engineering has a large scope, but much of the discipline is centered around the signals your cells make and what they do when they receive signals from other cells. These signals are chemically created by ions, but as engineers, we transduce (i.e., convert) them using different methods. This week we will attempt to measure your heartbeat using a relatively new method of transduction called photoplethysmography (PPG).

Supplies

- Triple channel DC power supply
- 2 banana cables (1 black and 1 red) with clips on the end (alligator or witch hats)
- Oscilloscope and probe
- Photoplethysmography (heartbeat) sensor
- USB thumb drive

Procedures

1. *Use the sensor to visualize a signal on the oscilloscope*
   a. The first thing to do is familiarize yourself with the photoplethysmography sensor (the heartbeat sensor) and its outputs. Looking at the sensor you will notice that it has three pins (Figure 1):
      i. Power 5V pin (indicated by a + on the back)
      ii. Data out pin (middle)
      iii. Ground pin (indicated by a - on the back)

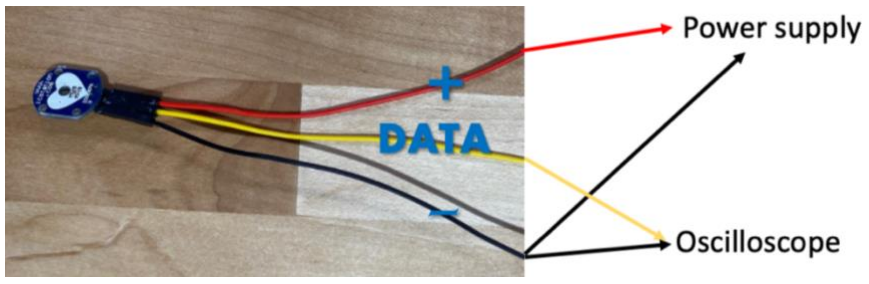

Figure 1: Heartbeat sensor with attached leads. Power 5 V (red/+), data out (yellow/middle), ground (black/-).

b. Turn on your power supply (Figure 2).
    i. You will be using CH 3 for this lab. Select CH 3 and use the V-set and I-Set buttons along with the keypad to change the voltage and current limits of that channel. Set your power supply limits to 5 V and 0.01 A. Plug your banana cables into the CH 3 outputs, being careful to keep the clips of the cables away from metal objects, particularly your laptops!!
    ii. Push the ON/OFF button under Output.

c. Attach the power supply's ground cable (black) to the sensor's ground (-) pin and the power supply's red cable to the sensor's positive (+) pin.

d. Turn on your oscilloscope and make sure your probe is calibrated. Attach the oscilloscope probe to the sensor's data pin (middle) and the oscilloscope grounding clip to the sensor's ground (-) pin. Set the scope's acquisition Time Mode to "Roll" via the "Acquire" button's menu.
    i. It will be useful to set the oscilloscope to 500 ms/div.
    ii. Ensure your attenuation switch on the probe is 10x.

e. Gently hold the sensor between your thumb and index finger. (If this doesn't work, set the sensor on the table and *gently* place your finger over the green light.) Check that there is a signal on the oscilloscope.
    i. For reference, an average heart rate ranges from 60 to 100 beats per minute.

f. Freeze the signal with the "Stop" button once you have several consistent cycles spanning the entire screen horizontally. When you have a signal that you like enough to save for analysis, do not make any further adjustments to the image on the screen before saving it.

2. Collect and process data from the oscilloscope

a. Use a USB Flash drive to save a capture of the data as a CSV file.

b. Import the CSV directly into MATLAB (importing into Excel first will round the values and cause errors during processing). Remove any rows containing unusable data and create two vector variables: `time` and `voltage`.

c. Use the `ENGR130_Fourier` function to perform a Fourier transform of the heartbeat data. Make a stem plot of the frequency-amplitude results with appropriate titles and labels. (If there is a large maximum at 0 Hz, do not include it; this will help the y-axis scale be more appropriate.) Save this plot to be included in your module report.

3. Analyze the signal in the frequency domain

a. Use `getComponentParams` to find your heartbeat frequency. You may perform post-processing on the matrix returned by `getComponentParams` to get the frequency you are most interested in, but your threshold passed to the function should still be informed through analyzing the heartbeat data CSV. Report the heart rate to the command window in beats per minute.

   i. Your heart rate is a local maximum on the amplitude vs. frequency plot, but which one?!

   ii. Note the maximum at 0 Hz; it is not your heartbeat (at least we hope it isn't!) but think about what it could be. Hint: it corresponds to a sine wave with a frequency of 0 Hz.

4. Prepare for Lab 4's Design Challenge

Read through the details for Lab 4. Arrive to class for Lab 4 with at least a basic idea of your team's approach to the design challenge.

Questions

1. Include your plot from Part 2c.

2. What do you notice about the frequency domain for the heartbeat signal compared to the `y_comp` and `mystery` from last week? Why do you think these are different? Please be specific.

3. What do you think the non-heartbeat frequencies represent? Could there be a physiological or electronic explanation for the different frequency bands of the signal?

4. Does a single frequency represent the heartbeat accurately? Why or why not?

END OF LAB 3

Lab 4: Design Challenge

Overview

One of the major ways doctors and clinicians diagnose heart conditions is by using the electrocardiogram (ECG or EKG). This test utilizes electrodes placed on this skin that measure the electrical activity of the heart to construct a heartbeat waveform. A typical ECG waveform and its components are shown in Figure 1.

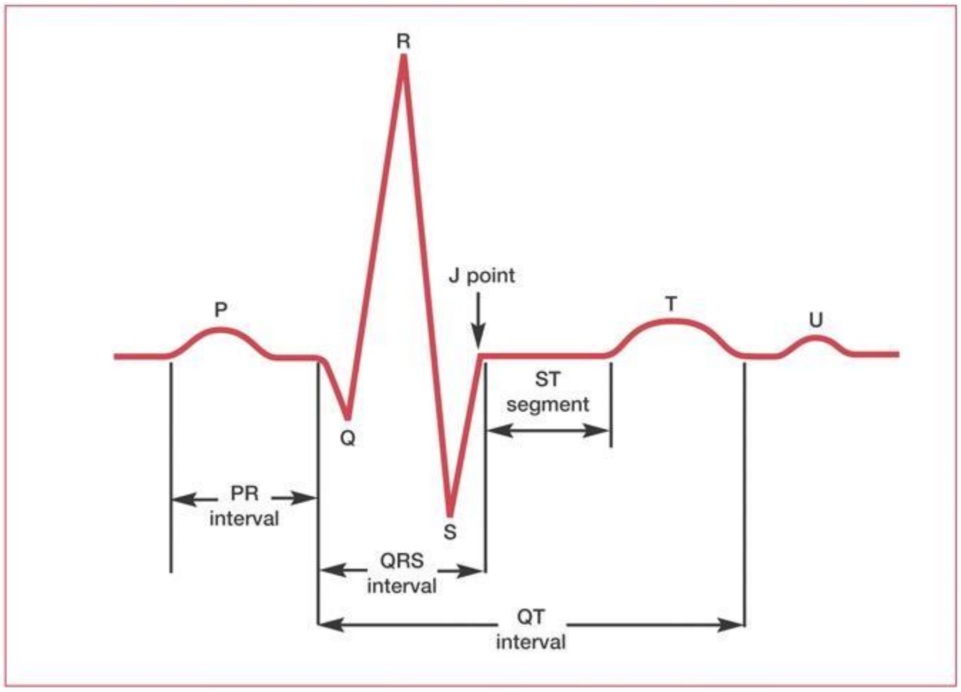

Figure 1: Typical points of an ECG. The QRS complex represents the main contraction of the heart - where your ventricle contracts and pushes blood to the rest of your body.[1]

The Challenge

As a biomedical engineer, it is your job to create devices, tools, and processes that help clinicians give better care to patients and improve patient outcomes. In this challenge, you will design a prototype device that detects abnormal ECG patterns (arrythmias) and alerts a clinician. Be sure to think about the clinical need of your device, as well as who the users are and what will be easiest for them - this is essential to biomedical engineering.

Details

Your code must diagnose at least one arrythmia in an ECG and alert the clinician. You will complete and submit the Module 5 Design worksheet with your lab report. You have 40 minutes to build a prototype of your device. During this time, you will also prepare a PowerPoint slide to project on the screen at your team's table when it is time to share your device. The slide should indicate your team number, the name of your device, and a one-sentence description of the device's purpose. A template for the slide is available. You are not required to use this template.

Constraints: You may use one Arduino and extra equipment that was available for the Module 3 design challenge. You will have access to CSV files containing undiagnosed ECG arrythmias for your practice and demonstration. There are five CSV files with arrythmias and one normal ECG for comparison. In each file, the first column is time (in seconds) and the second column is the ECG signal.

Understanding ECGs: In previous labs of this module, we have used the Fourier Transform to determine the major frequencies and calculate the heart rate of the signal. While this is a great technique, an ECG can be analyzed more simply. The beats per minute (bpm) is calculated as the 60 divided by the number of seconds between two consecutive R peaks (RR):

$$bpm = \frac{60}{RR}$$

An arrhythmia is any abnormal heart rhythm, but it can take many forms. Some types are included below, which you may want to use in your design challenge. You can also read more about them here.

- Bradyarrhythmia: characterized by a heart rate of fewer than 60 beats per minute
- Tachyarrhythmia: characterized by a heart rate higher than 100 beats per minute
- Atrial fibrillation: a type of tachycardia that is characterized by irregular QRS complexes, and a fast heart rate
- First degree AV block: characterized by a PR interval of the ECG being greater than 200 ms

Project Planning and Management

Complete your final Project Planning and Management Task List. You will turn in a pdf of your final task list with your Part II Lab Report.

END OF LAB 4