ECE 351: Signals and Systems

Peer Exchange Project

Chuck O'Neill Matthew Wolden

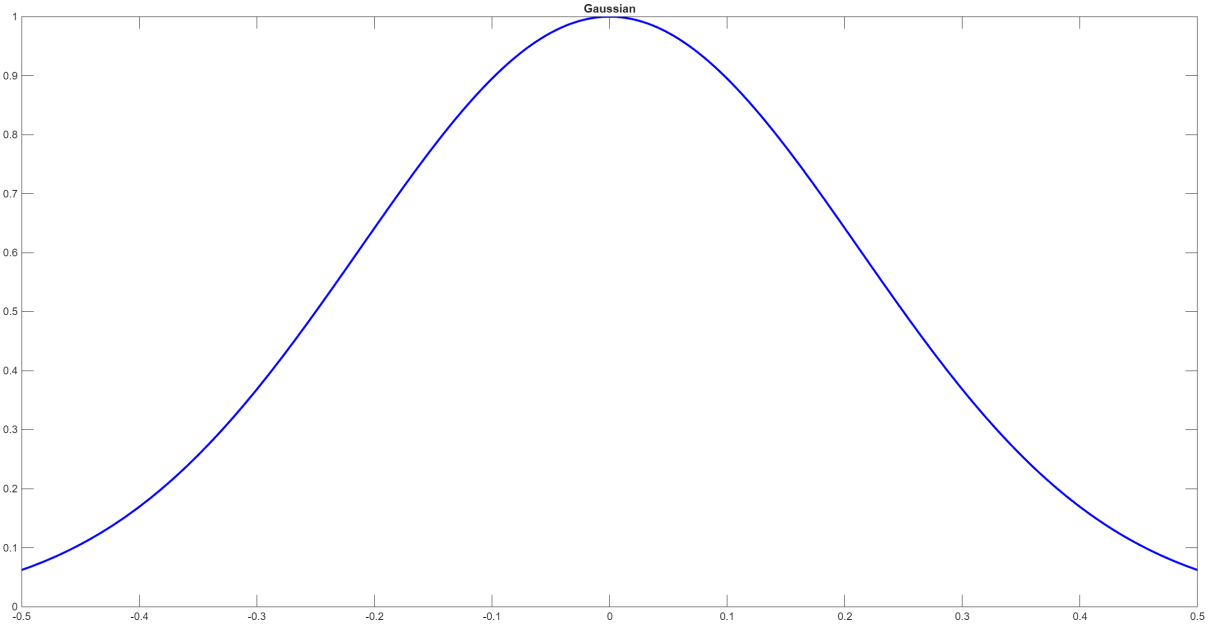
November 13, 2025

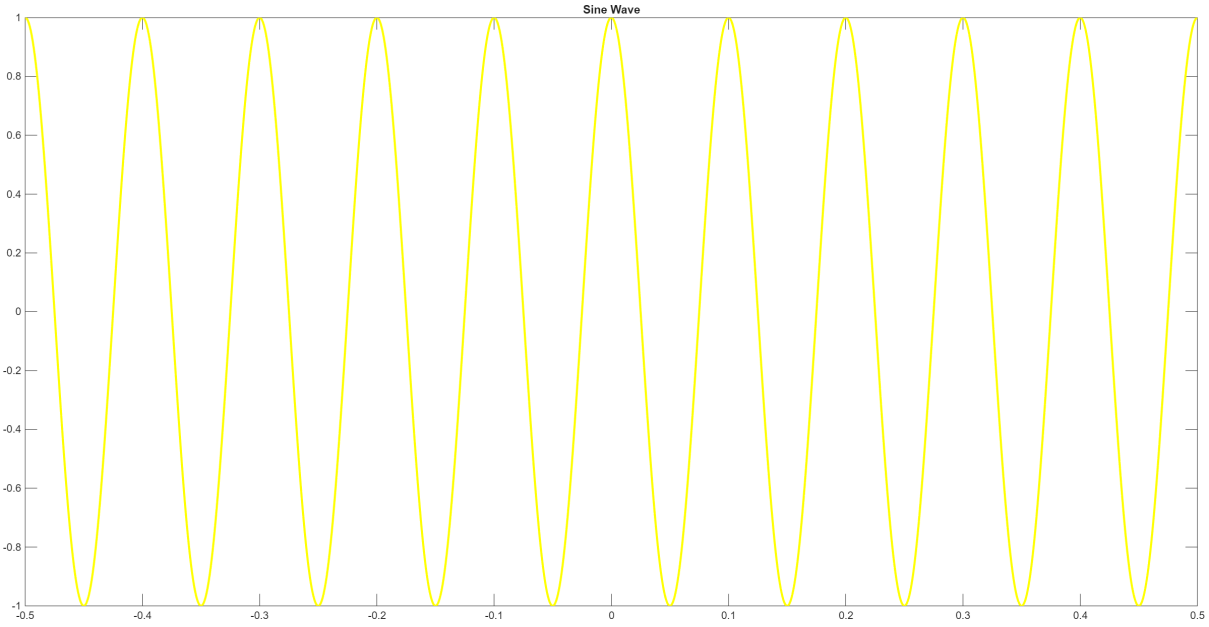
1 Introduction

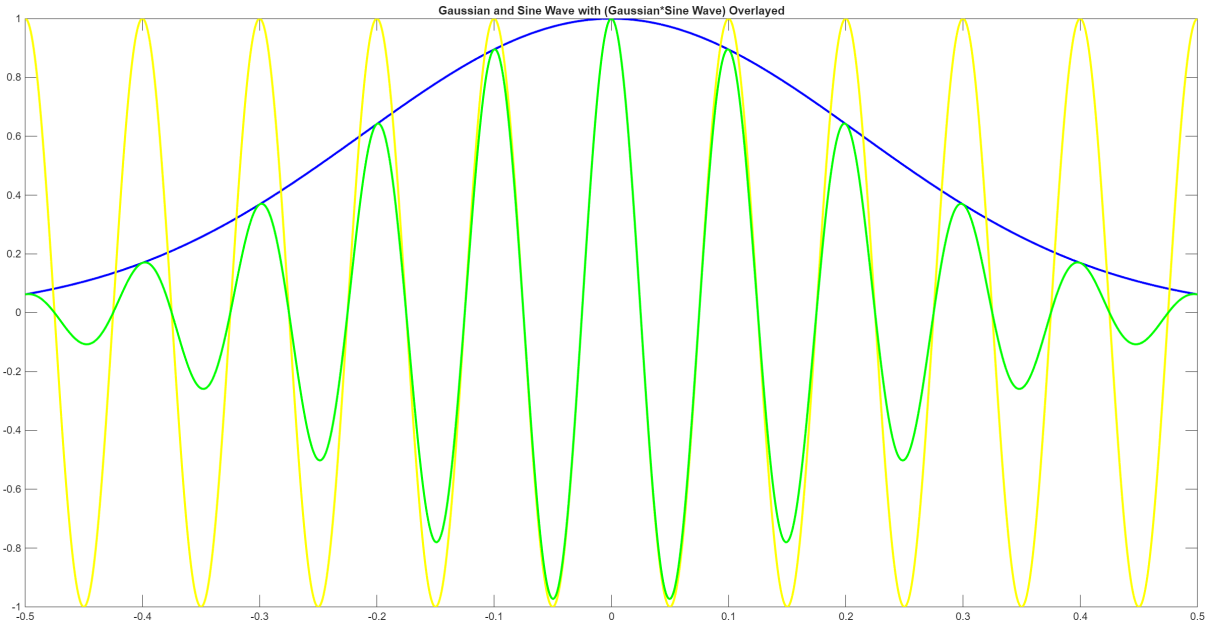
This project is focused on showing that convolution in the time domain is equivalent to multiplication in the frequency domain. It uses a repository available on github. The repository contains a sound file, a file which maps characters to frequencies at the high-end of audible range, and a couple files to embed tones of the aformentioned frequencies into the sound file. There are two files to do a convolution between the altered sound file and wavelets matching the frequencies of the embedded tones then plot the results. This convolution process can be replicated using multiplication in the frequency domain. The task here is to show that through transforming both the wavelet and the audio to the frequency domain, multiplying the two signals, and transforming the signal back into the time domain, one can show where the embedded frequencies exist. These frequencies are embedded in a way that encodes a message when the frequencies are mapped back to their matching characters in the chars.mat file. There are functions provided to do the proceses aside from the Fourier and Inverse Fourier Transforms.

2 Morlet Wavelet

This section exists in hopes that it makes this process make more intuitive sense. It is mostly paraphrased from this video. A handful of relevant notes are mentioned, but the aim is to visualize the most important ones. The wavelet itself is a multiplication of a Gaussian and a Sine wave. The full-width half maximum value of the wavelet must be wider than the frequency burst we are looking for. The duration of the wavelet will ideally not be too much longer than the full-width half maximum, as long tails of the wavelet with minimal oscillation are not useful. A visualization of a wavelet and the Gaussian and Sine Wave from which it was created are shown below.

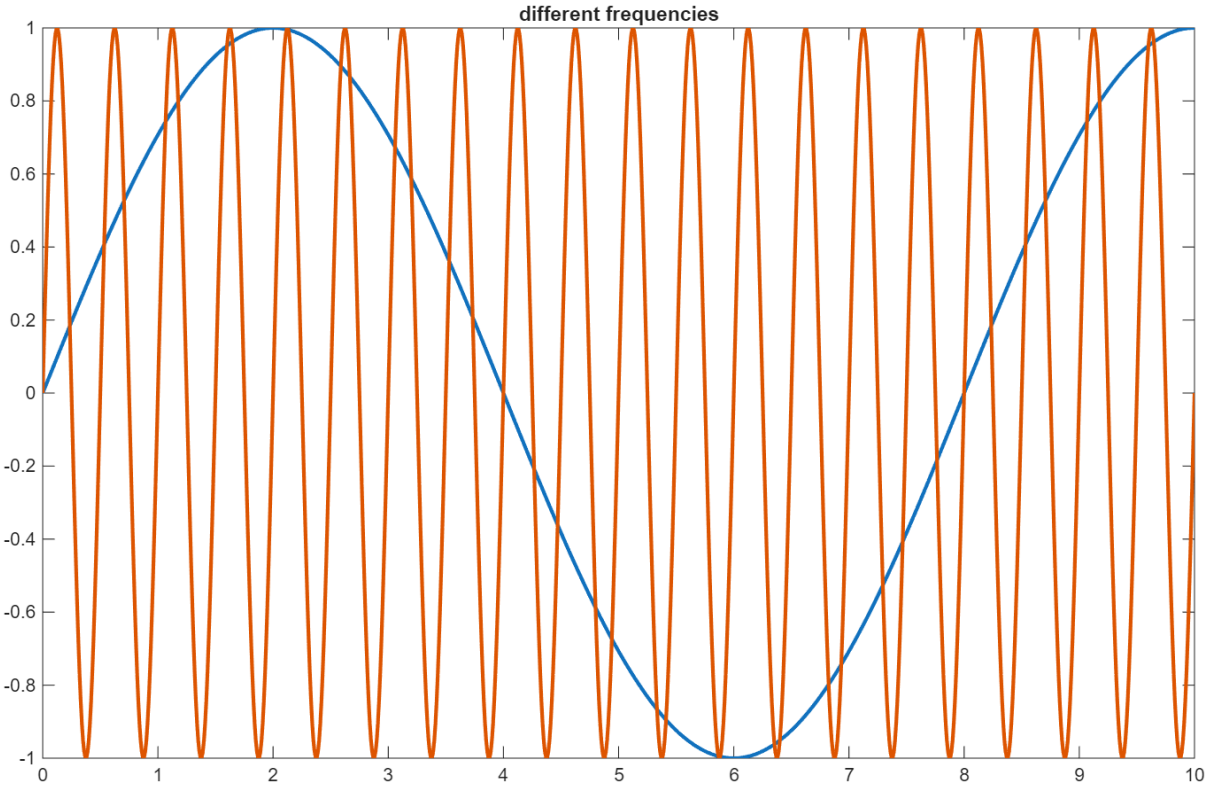

Gaussian
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5

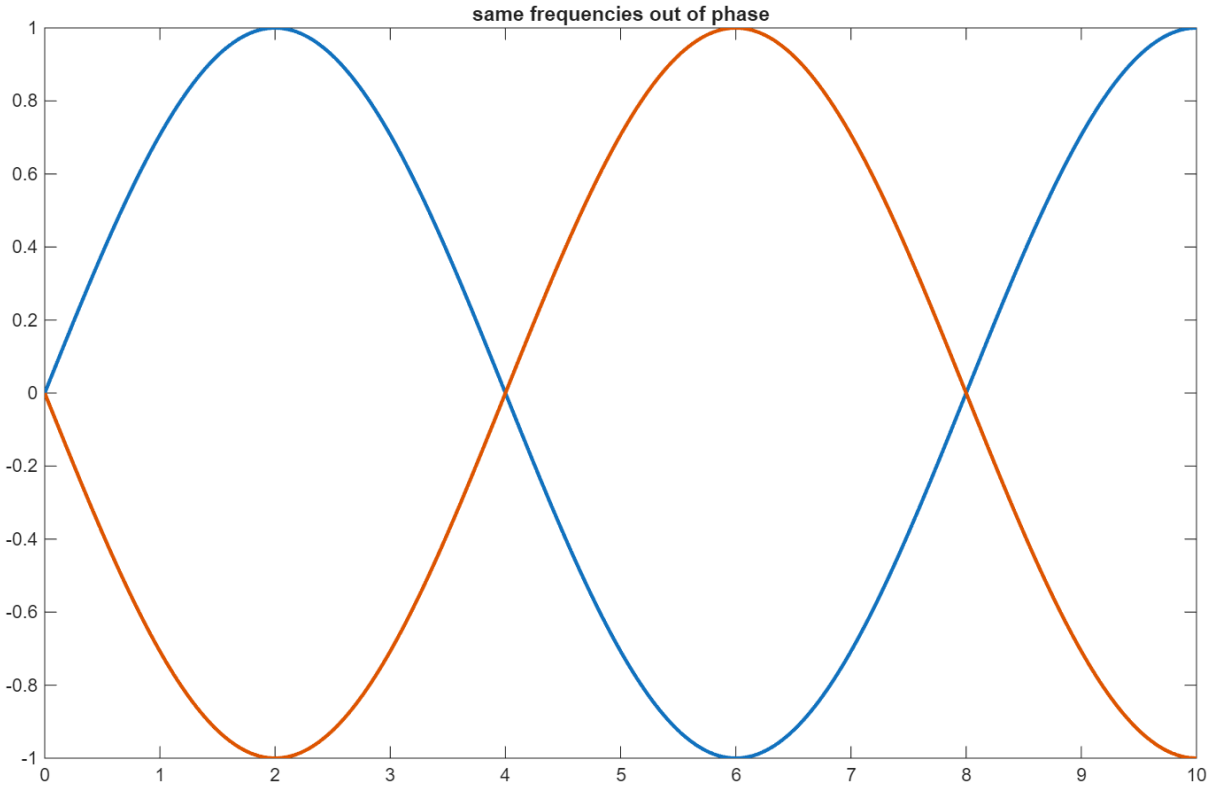

Sine Wave
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5


Gaussian and Sine Wave with (Gaussian*Sine Wave) Overlayed
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5

3 Convolution

Another useful concept is considering why a wavelet of a specific frequency picks out similar frequencies from our signal. Because, the wave oscillates above and below zero equally, it has an average of zero. Convolution on a discrete system is just a number of equally spaced points at which the two signals are multiplied together. The resulting products are then summed. When one of these signals oscillates much faster than another, the higher-frequency wave can oscillate above and below zero many times while the slower wave's y-value may change very little. If the higher-frequency wave is much greater in frequency, as we zoom in to make its oscillations visible, the change in the y-value of the slower wave may not be discernable at all. The inverse of this situation is if we have two waves of the same frequency. The magnitudes of their product will be much larger. A couple of sets of waves which fit the aforementioned descriptions are shown below. Keeping these two ideas in mind, should help with intuition about why convolution with a wavelet produces spikes at similar frequencies and valleys at dissimilar ones.

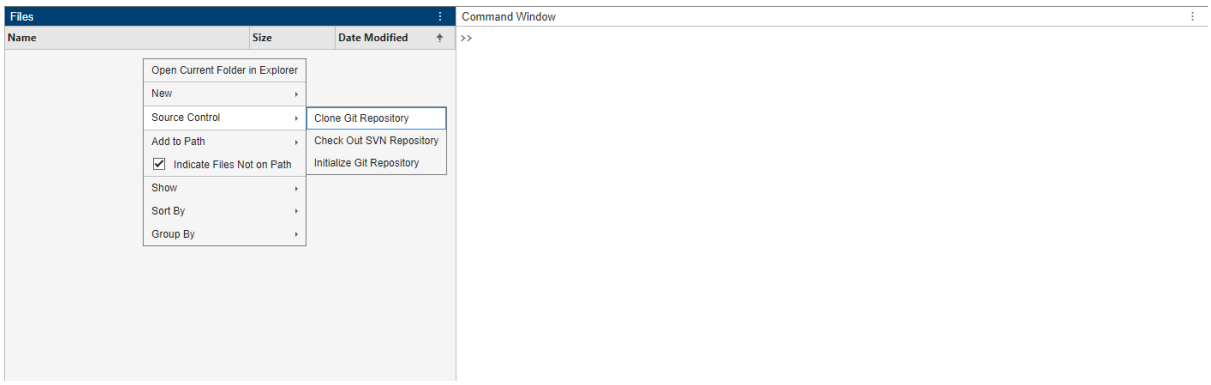


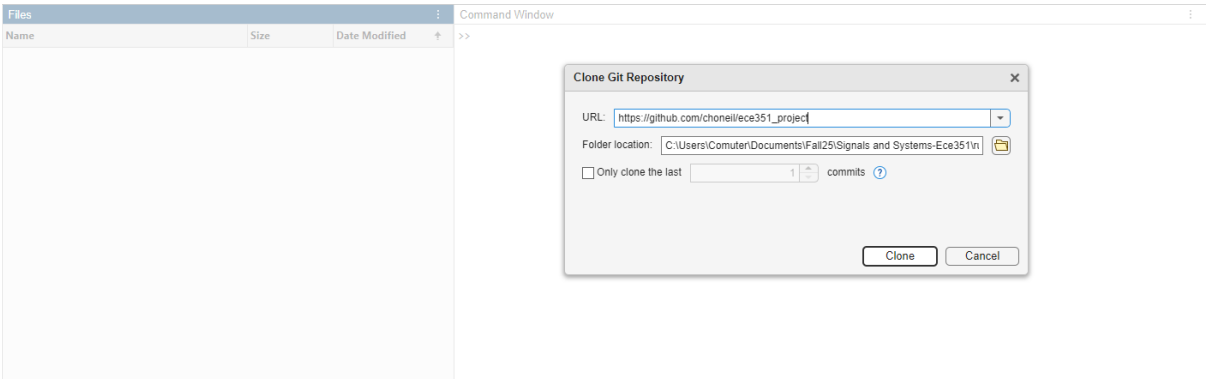


4 Setup

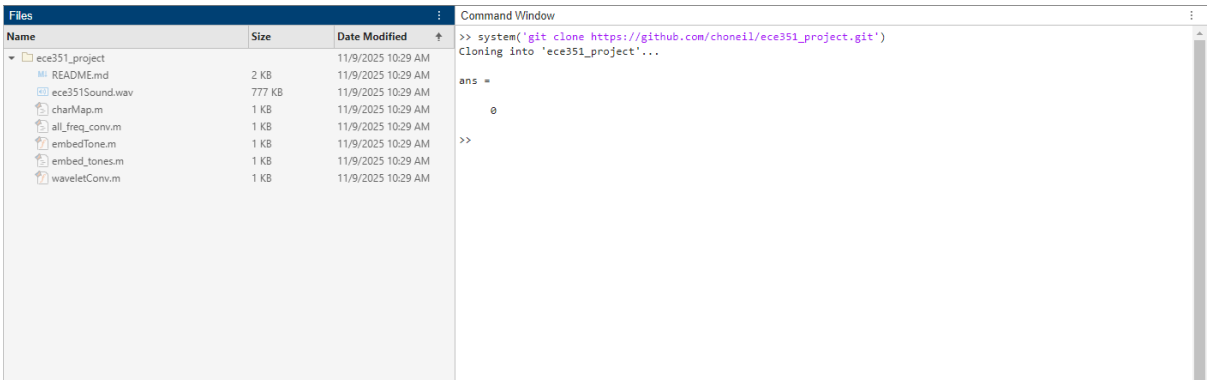
1. In your MATLAB environment, create a new folder which you will be working in. In this environment, clone the GitHub repository by either right-clicking in the file explorer or typing `git clone https://github.com/choneil/ece351_project` into your command window.

   1a. Right-click and highlight Source Control, then select “Clone GitHub Repository.” Enter the URL into the field and click "Clone".





1b. Cloning using the command window instead.



2. Run the `charMap.m` file.

3. Run The `embed_tones.m` file.

4. Run the `all_freq_conv.m` file.

The output of the final file should show a graph of the convolution results. Only the convolution results which had a high variance, which was used to indicate a significant spike, were plotted. The legend of the graph shows which color plots match with which characters. The characters are encoded in the same order they appear in the message, so the message can be decoded using this graph.

5 Matlab Functions

This section is quick overview of Matlab functions which will be useful for this task.

1. `audioread` takes an audio file as input and returns the datapoints of the audio as well as the sampling rate.

2. `fft` returns Y the fast Fourier transform of X. Matlab documentation shows it as Y = fft(X,n,dim) where all inputs aside from X are optional. n is the length of the transform and dim is the dimension which it is operating on.

3. `ifft` returns Y the inverse fast Fourier transform of X. Matlab documentation shows it as Y = ifft(X, n, dim) where all inputs aside from X are optional. n is the length of the inverse transform and dim is the dimension which it is operating on.

4. `fieldnames` can take an input argument that is a struct and return the fieldnames. "fnames = fieldnames(chars)". The return of this function is given as a cell array, so its elements can be accessed as "char = fnamesi". Accessing the frequency which matches the character in the struct can then be done using something like "freq = chars.(char)".

5. `numel` can be used to get the number of elements in a struct. n = numel(fnames) would give you the number of elements in the struct.

6. `conj(x)` will give you the complex conjugate of x.

7. `real(x)` will give you the real portion of a complex number. This will be useful for graphing.

6 Matlab Operators

This section is quick overview of Matlab functions which will be useful for this task.

1. `Element-wise operator(./, .*)` can be used to do element-wise operations. For example two arrays using element-wise multiplication: [1 2 3] .* [4 5 6] = [4 10 18]. This operator will be useful for multiplying the transforms.

2. `Column Vectors` will be useful when doing the transform's multiplication. A row vector "w" which has dimensions [1xlength] can be turned into a column vector [lengthx1] using w = w(:).

7 Process

1. Graph the sound file.

2. Create the transform of the sound file and graph it.

3. Use the getWavelet.m function to create a wavelet of 20.2kHz and graph its return.

4. Create the transform of this wavelet and graph the wavelet in the frequency domain. Pad out the wavelet to the length of the sound file using the n input of fft. Graph this as well.

5. Use element-wise multiplication on these two signals in the frequency domain.

6. Transform the result of this multiplication back to the time domain using ifft().

7. Graph the result in time domain.

8. Loop through the frequencies which are mapped to characters repeating the same process for each. In this loop you can check the variance of waveform after transforming it back into the time domain using var(waveform). Plot the results which have a high enough variance. Some of the structure for this loop can be based off the `all_freq_conv.m` file structure.

9. In a short document, include the graphs created in this processes as well as the code written to do these processes. Also, include a short answer to the following questions: From any of the processes in either the provided Github repo or the code you created throughout this project, where could you see this applied within a topic you find interesting?

Category & Description	**5**	**3**	**1**	**Points**
`Wavelet Graphs` Provide two graphs: one of the wavelet in the frequency domain and one in the time domain.	Graph correct and well-labeled.	Graph has minor errors.	Graph has significant errors	/5
`Audio Graphs` Provide two graphs: one of the audio file in the frequency domain and one in the time domain	Graph correct and well-labeled.	Graph has minor errors.	Graph has significant errors	/5
`Graph of the results(x2)` Provide a graph of the results from the multiplication in the frequency domain after it has been transformed back to time domain.	Graph correct and well-labeled.	Graph has minor errors.	Graph has significant errors	/10
`Code and formatting(x2)` Documented code is included as well as the short response.	All required items are included and well-documented.	Small errors within the included items.	Significant errors within the included items.	/10

Table 1: Project rubric