



CAL POLY

EE 367 Lab 1

Sampled Signals and Spectra

Report by:

Ethan Vosburg (evosburg@calpoly.edu)

Isaac Lake (ilake@calpoly.edu)

Ryan Ceballos (rceballo@calpoly.edu)

January 30, 2024

Table of Contents

1 Interpreting Spectra with Signals Generated in Python.....	3
1.1 Part 1 Design Specifications:	3
2 Impact of Delay on Frequency Spectra	6
3 Impact of Amplitude Modulation on Frequency Spectra	6
4 Impact of Absolute Value on Frequency Spectra.....	9

1 Interpreting Spectra with Signals Generated in Python

1.1 Part 1 Design Specifications:

For part 1 we will be making a wave file with Python. The wave file will be constructed according to these specifications.

Attribute	Specification
Duration	2 Seconds
Frequency	2000Hz
Sample Rate	16kHz
Channel Count	Mono
Amplitude	75%
Phase	45°

Table 1: Wave Specifications

1. How many samples are generated in total?

There would be 32,000 Samples

2. How many bytes are expected for the WAV file?

With the addition of the 44 bytes for in the header, the resulting calculation is $32,000 + 44$ totaling 32,044 bytes.

3. What is your chosen amplitude?

The chosen amplitude is calculated by multiplying available bits for numeric values 2^{15} by the desired amplitude 75% resulting in 24,576 samples.

4. What is the digital frequency in rads/sample?

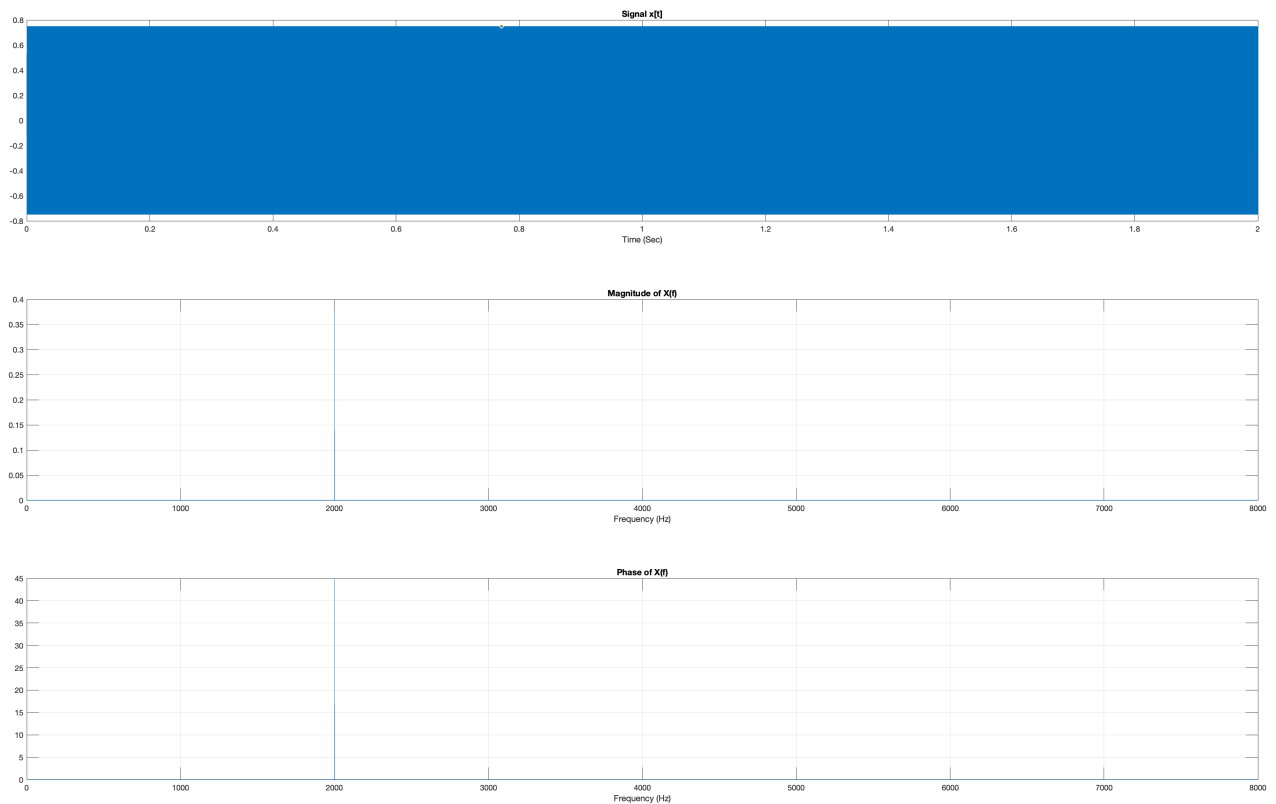
The digital frequency can be calculated by the following:

$$F = \frac{f_0}{f_s}$$

$$F = \frac{2kHz}{16kHz}$$

$$F = \frac{1}{8} = 0.125$$

5. MatLab Plot

**Figure 1: Caption**

6. Expected magnitude at $f = 2000$ Hz? Any error seen in the plot?
The expected magnitude was .75 which matches the plot
7. Expected phase at $f = 2000$ Hz? Any error seen in the plot?
 45° is the expected phase. No error seen in the plot
8. Python routine `gen_wav`

Listing 1: Python Routine for Generating 2000Hz

```

1  """
2  File: EE367-Lab1-1.py
3  Author: Ethan Vosburg
4  Date: 01-25-24
5  Description: This file contains the code for part 1 of Lab 1 for EE367. This
6  file creates a 2 second long 2000 Hz cosine wave with a 45 degree phase shift.
7  """
8
9  # Import Statements
10 import sys
11 import time
12 import base64
13 import random as random
14 import datetime

```

```

15 import time
16 import math
17
18 from cpe367exp1.cpe367_wav import cpe367_wav
19
20
21 def wavgen(dur, freq, rate, channels, amp, phase, delay, fpath_wave_out):
22     """This function generates a WAV file
23
24     Arguments:
25         dur :           Duration of the wave in seconds
26         freq :          Frequency of the wave in Hz
27         rate :          Sample rate of the wave in Hz
28         channels :      Number of channels in the wave
29         amp :           Amplitude of the wave
30         phase :         Phase of the wave
31         delay :         Delay of the wave in seconds
32         fpath_wave_out : File path of the wave
33
34     Returns:
35         True or False
36
37     """
38     # Create a new wav file object
39     wav_out = cpe367_wav('wavOut', fpath_wave_out)
40
41     # Configure the wav file object
42     wav_out.set_wav_out_configuration(
43         channels,
44         16,
45         rate,
46     )
47
48     # Check if the wav file object was configured correctly
49     if wav_out == False:
50         print("Error setting wav configuration")
51         return False
52
53     # Open the wav file
54     working_file = wav_out.open_wav_out()
55     if working_file == False:
56         print("Error opening file for writing")
57         return False
58
59     # Calculate number of samples needed and the angular frequency
60     sample_count = int(dur * rate)
61     w1 = 2 * math.pi * freq / rate
62
63     # Generate the samples and write them to the wav file
64     for i in range(sample_count):
65         working_sample = amp * math.cos(w1 * (i - delay * rate) + (phase/360 * 2 * math.pi))
66         working_sample = int(round(working_sample))
67
68         working_file = wav_out.write_wav(working_sample)
69         if working_file == False: break
70         if working_file == False: break
71
72     # Close the wav file
73     wav_out.close_wav()
74     return True
75
76 # Run the main function
77 def main():
78     try:
79         fpath_wav_out = 'Lab1-1.wav'
80         wavgen(2, 2000, 16000, 1, 24576, 45, 0, fpath_wav_out)
81     except:
82         print("Error: ", sys.exc_info()[0])
83         return False
84

```

```
85     return True
86
87 # Checking if program is run as main
88 if __name__ == '__main__':
89     if (main() == True):
90         print("Waveform Generated Successfully")
91
92     # Exit the program
93     quit()
```

2 Impact of Delay on Frequency Spectra

1. What is the equivalent delay?

A 45° delay is equivalent to $\frac{1}{8}$ the period or $62.5\mu s$

2. What is the equivalent phase shift, theta, (in radians)?

A 45° phase shift is equivalent to $\frac{\pi}{4}$ radians.

3. Describe similarities and differences between the delayed and original version

The similarities are that they have the same amplitude and frequency. The differences are that they are out of phase and are offset at any given time from their non-shifted versions.

4. Describe similarities and differences between the delayed version and the one with -A. Explain why these similarities or differences arise.

A negative A is the equivalent of shifting by π in other words it will be even more offset from our mere $\frac{\pi}{4}$ offset. It should similarly still share the same frequency and magnitude.

3 Impact of Amplitude Modulation on Frequency Spectra

1. What mathematical identity allows you to predict the spectrum of $x(t)$?

$$A \cos \alpha * B \cos \beta = A * B(1/2)[\cos(\alpha + \beta) + \cos(\alpha - \beta)]$$

2. What are your values for A and B?

$$A = 128, B = 128$$

3. What are your values for θ_1 and θ_2 ?

$$\theta_1 = +45^\circ \text{ and } \theta_2 = -45^\circ$$

4. If the order of the cosine functions is swapped then would $x(t)$ change?

If the order of the cosine functions were swapped, $x(t)$ would not change.

5. MatLab Plots

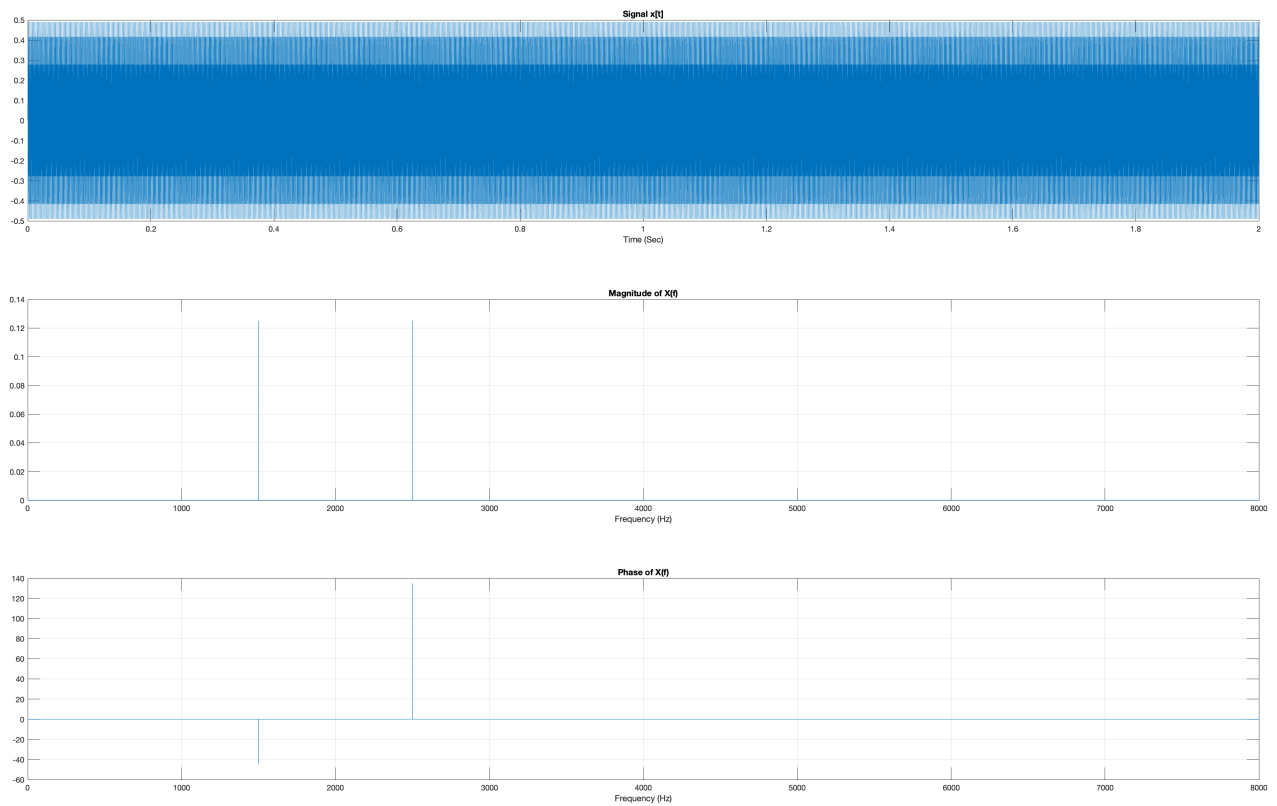


Figure 2: Caption

6. Python routine 'gen_wav', with comments

Listing 2: Python Routine for Generating Modulating Signal

```

1  """
2  File: EE367-Lab1-3.py
3  Author: Ethan Vosburg
4  Date: 01-25-24
5  Description: This file contains the code for part 1 of Lab 1 for EE367. This
6  file creates a 2 second long modulated wave with a 2000 Hz cosine wave and a
7  5000 Hz cosine wave.
8  """
9
10 # Import Statements
11 import sys
12 import time
13 import base64
14 import random as random
15 import datetime
16 import time
17 import math
18
19 from cpe367exp1.cpe367_wav import cpe367_wav
20
21

```

```

22 def wavgenmod(dur,
23               freq1,
24               freq2,
25               rate,
26               channels,
27               amp1,
28               amp2,
29               phase1,
30               phase2,
31               delay,
32               fpath_wave_out):
33     """This function generates a WAV file
34
35     Arguments:
36         dur :                Duration of the wave in seconds
37         freq1 :              Frequency of the wave in Hz
38         freq2 :              Frequency of the wave in Hz
39         rate :               Sample rate of the wave in Hz
40         channels :           Number of channels in the wave
41         amp :                Amplitude of the wave
42         phase :              Phase of the wave
43         delay :              Delay of the wave in seconds
44         fpath_wave_out :     File path of the wave
45
46     Returns:
47         True or False
48
49     """
50     # Create a new wav file object
51     wav_out = cpe367_wav('wavOut', fpath_wave_out)
52
53     # Configure the wav file object
54     wav_out.set_wav_out_configuration(
55         channels,
56         16,
57         rate,
58     )
59
60     # Check if the wav file object was configured correctly
61     if wav_out == False:
62         print("Error setting wav configuration")
63         return False
64
65     # Open the wav file
66     working_file = wav_out.open_wav_out()
67     if working_file == False:
68         print("Error opening file for writing")
69         return False
70
71     # Calculate number of samples needed and the angular frequency
72     sample_count = int(dur * rate)
73     w1 = 2 * math.pi * freq1 / rate
74     w2 = 2 * math.pi * freq2 / rate
75
76     # Generate the samples and write them to the wav file
77     for i in range(sample_count):
78         wave1 = amp1 * math.cos(w1 * (i - delay * rate) + (phase1/360 * 2 * math.pi))
79         wave2 = amp2 * math.cos(w2 * (i - delay * rate) + (phase2/360 * 2 * math.pi))
80         working_sample = wave1 * wave2
81         working_sample = int(round(working_sample))
82
83         working_file = wav_out.write_wav(working_sample)
84         if working_file == False: break
85         if working_file == False: break
86
87     # Close the wav file
88     wav_out.close_wav()
89     return True
90
91 # Run the main function

```



```
92 def main():
93     try:
94         fpath_wav_out = 'Lab1-3.wav'
95         wavgenmod(2, 2000, 500, 16000, 1, 128, 128, 45, -45, 0.00025, fpath_wav_out)
96     except:
97         print("Error: ", sys.exc_info()[0])
98         return False
99
100     return True
101
102 # Checking if program is run as main
103 if __name__ == '__main__':
104     if (main() == True):
105         print("Waveform Generated Successfully")
106
107     # Exit the program
108     quit()
```

4 Impact of Absolute Value on Frequency Spectra

1. Describe the frequency components that are present in this signal. Is there a pattern? What are the lowest two frequencies present? Is DC present?

The lowest two frequencies present in this signal are 320 Hz and 640Hz. Yes, DC is present and has a magnitude of 0.477305. Since the frequency of the cosine function is 160Hz, we see double this frequency on the positive side and our graphs show this perfectly. There is a pattern present in the data, the frequency found in the magnitude response increases by 320Hz every time.

2. MatLab plots

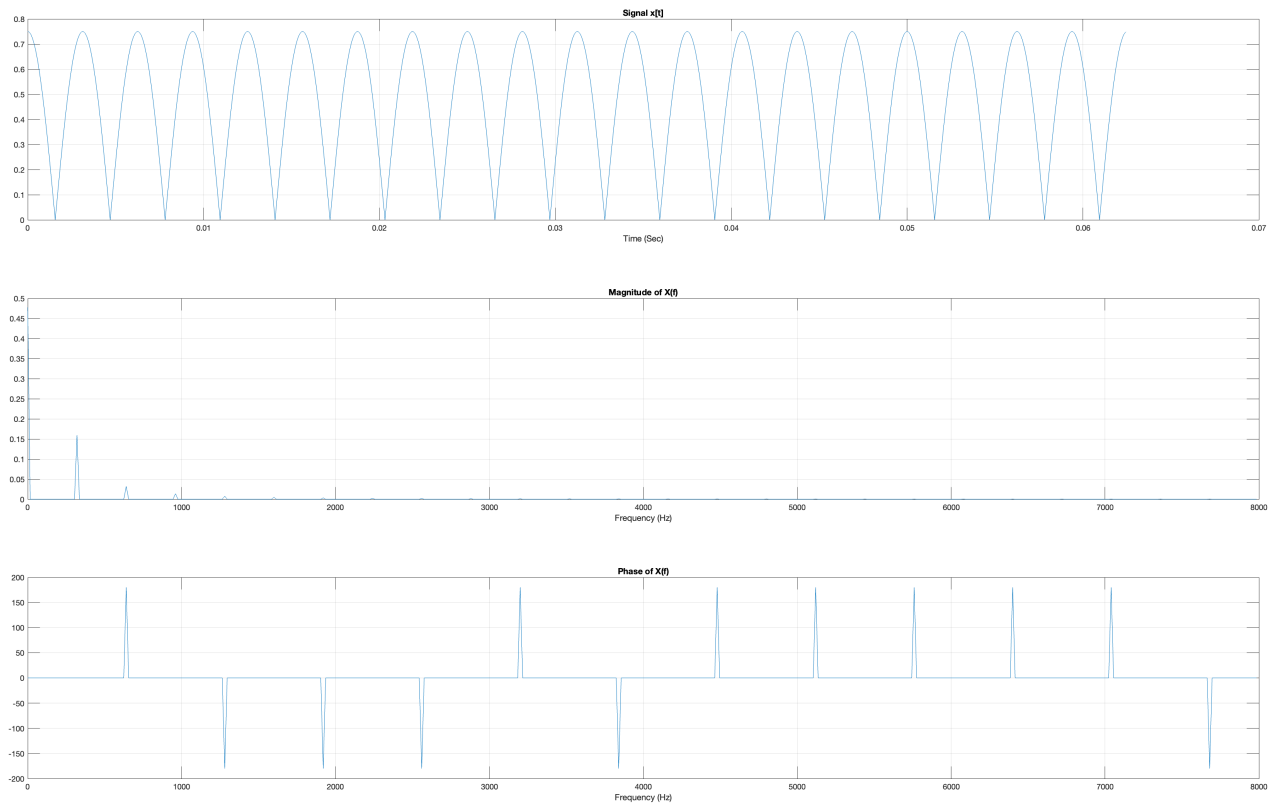


Figure 3: Caption

3. Python routine 'gen_wav', with comments

Listing 3: Python Routine for Generating Absolute 160Hz

```

1  """
2  File: EE367-Lab1-4.py
3  Author: Ethan Vosburg
4  Date: 01-25-24
5  Description: This file contains the code for part 1 of Lab 1 for EE367. This
6  file creates a 1/16 second long 160 Hz cosine wave.
7  """
8
9  # Import Statements
10 import sys
11 import time
12 import base64
13 import random as random
14 import datetime
15 import time
16 import math
17
18 from cpe367exp1.cpe367_wav import cpe367_wav
19
20
21 def wavgen(dur, freq, rate, channels, amp, phase, delay, fpath_wave_out):

```

```

22     """This function generates a WAV file
23
24     Arguments:
25         dur :           Duration of the wave in seconds
26         freq :          Frequency of the wave in Hz
27         rate :          Sample rate of the wave in Hz
28         channels :      Number of channels in the wave
29         amp :           Amplitude of the wave
30         phase :         Phase of the wave
31         delay :         Delay of the wave in seconds
32         fpath_wave_out : File path of the wave
33
34     Returns:
35         True or False
36
37     """
38     # Create a new wav file object
39     wav_out = cpe367_wav('wavOut', fpath_wave_out)
40
41     # Configure the wav file object
42     wav_out.set_wav_out_configuration(
43         channels,
44         16,
45         rate,
46     )
47
48     # Check if the wav file object was configured correctly
49     if wav_out == False:
50         print("Error setting wav configuration")
51         return False
52
53     # Open the wav file
54     working_file = wav_out.open_wav_out()
55     if working_file == False:
56         print("Error opening file for writing")
57         return False
58
59     # Calculate number of samples needed and the angular frequency
60     sample_count = int(dur * rate)
61     w1 = 2 * math.pi * freq / rate
62
63     # Generate the samples and write them to the wav file
64     for i in range(sample_count):
65         working_sample = abs(amp * math.cos(w1 * (i - delay * rate) + (phase/360 * 2 * math.pi)
66         ))
67         working_sample = int(round(working_sample))
68
69         working_file = wav_out.write_wav(working_sample)
70         if working_file == False: break
71         if working_file == False: break
72
73     # Close the wav file
74     wav_out.close_wav()
75     return True
76
77 # Run the main function
78 def main():
79     try:
80         fpath_wav_out = 'Lab1-4.wav'
81         wavgen(0.0625, 160, 16000, 1, 24576, 0, 0, fpath_wav_out)
82     except:
83         print("Error: ", sys.exc_info()[0])
84         return False
85
86     return True
87
88 # Checking if program is run as main
89 if __name__ == '__main__':
90     if (main() == True):
91         print("Waveform Generated Successfully")

```

```
91 |  
92 |   # Exit the program  
93 |   quit()
```