2. Analysing Signals

Demo: https://youtu.be/HfMZOQbxoYo

We can either look at signals in the time domain or in the frequency domain. Often the frequency domain will give us more details on the information contained in the signal, along with how it will be processed by a system.

1 Plotting signals

Enter the following code:

```
import matplotlib.pyplot as plot
import math
xs=[]
ys=[]
for x in range(1000):
   xs.append(0.01*x)
   ys.append(math.sin(xs[x]))
plot.xlabel('time')
plot.ylabel('amplitude')
plot.plot(xs,ys)
plot.show()
```

Run the program, and plot the output:

We can improve the code using the numpy library, and add the arrange method which will fill a list for x with a start time and an end time, and then for a time step:

```
import matplotlib.pyplot as plot
import numpy as np
xs = np.arange(0.0, 2, 0.01)
ys = np.sin(2*np.pi*xs)

plot.xlabel('time')
plot.ylabel('amplitude')
plot.plot(xs,ys)
plot.show()
```

Now add a cosine function with

```
import matplotlib.pyplot as plot
import numpy as np

xs = np.arange(0.0, 2, 0.01)
ys = np.sin(2*np.pi*xs)
zs = np.cos(2*np.pi*xs)

plot.xlabel('time')
plot.ylabel('amplitude')
plot.plot(xs,ys)
plot.plot(xs,zs)
plot.show()
```

2 Other waveforms (Sawtooth, Square and Gaussian modulated sinusoid)

Next we will plot a sawtooth waveform. For this we will use the scipy library:

```
import matplotlib.pyplot as plot
import numpy as np
from scipy import signal

xs = np.arange(0.0, 2, 0.01)
ys= signal.sawtooth(2 * np.pi * 5 * xs)

plot.xlabel('time')
plot.ylabel('amplitude')
plot.plot(xs,ys)

plot.show()
```

Plot the waveform produced:

A square wave can be integrated with (with a frequency of 5Hz):

```
signal.square(2 * np.pi * 5 * t))
```

Integrate a square waveform, and plot the response:

We can plot a Gaussian modulated sinusoid:

```
ys= signal.gausspulse(xs,fc=5,bw=0.5,bwr=-6,retquad=0)
```

Integrate a Gaussian modulated sinusoid waveform, and plot the response:

Can you determine what the parameters are used for:

We can produce a frequency-swept cosine generator with chirp():

```
ys = signal.chirp(xs, f0=0, t1=1, f1=5, method='linear', phi=0)
```

Integrate a chirp waveform, and plot the response:

Can you determine what the parameters are used for:

There are two parameters that we can pass to the sawtooth() method. One is *t*, and

which defines the time array. The other is *width* which defines the portion of the rising ramp that is used for the total cycle. We can produce a triangular waveform by setting the *width* value in the sawtooth() method to 0.5 (where the default is set to 1):

```
ys= signal.sawtooth(2 * np.pi * 5 * xs, width=0.5)
```

Integrate a triangular waveform, and plot the response:

How does the form of the waveform vary as the value of *t* changes:

3 Frequency plot

We can now run a Fast Fourier Transform (FFT) on the time-based signal, in order to analysis the frequency content of the signal. For this create a Python program with:

```
import matplotlib.pyplot as plt
import numpy as np

Fs = 150.0;  # sampling rate
Ts = 1.0/Fs; # sampling interval

freq = 5;   # frequency of the signal

t = np.arange(0,1,Ts)
y = np.sin(2*np.pi*freq*t)

n = len(y) # length of the signal
k = np.arange(n)
T = n/Fs
frq = k/T # two sides frequency range
frq = frq[range(n/2)] # one side frequency range

Y = np.fft.fft(y)/n # fft computing and normalization
Y = Y[range(n/2)]

fig,myplot = plt.subplots(2, 1)
```

```
myplot[0].plot(t,y)
myplot[0].set_xlabel('Time')
myplot[0].set_ylabel('Amplitude')
myplot[1].plot(frq,abs(Y),'r') # plotting the spectrum
myplot[1].set_xlabel('Freq (Hz)')
myplot[1].set_ylabel('|Y(freq)|')

plt.show()
```

Run the program, and determine the peak frequency for the signal:

Now modify the program so that it plots a 10 Hz signal, and now determine the peak frequency:

Now integrate a sawtooth waveform.

Plot the frequency response:

Now integrate a triangular waveform.

Plot the frequency response:

Now integrate a Gaussian modulated sinusoid waveform.

Plot the frequency response:

Now integrate a chirp waveform.

Plot the frequency response:

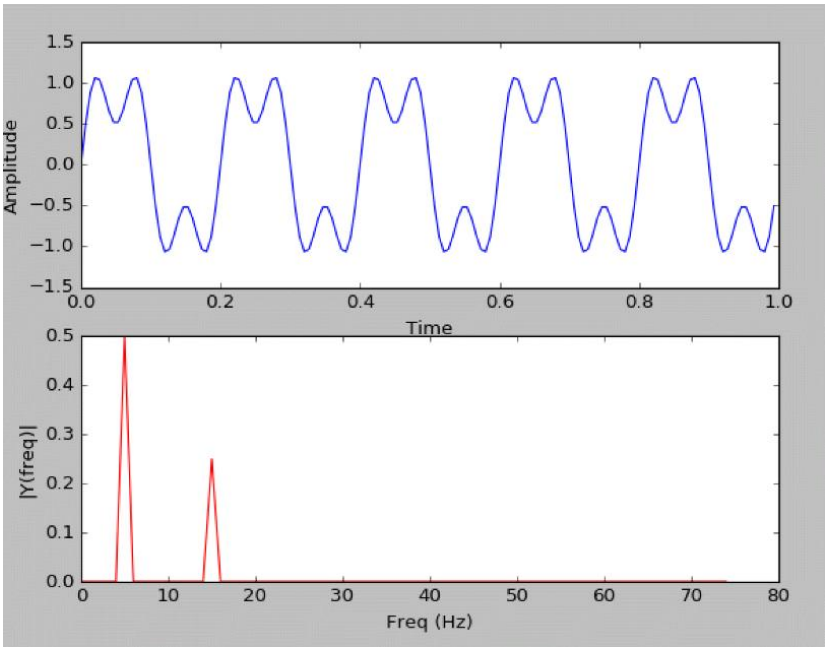
We can produce a triangular waveform by

4 Harmonics

Now update the program from the previous section, so that the waveform has a third harmonic which is 0.5 in amplitude:

```
y = np.sin(2*np.pi*freq*t) + 0.5* np.sin(2*np.pi*3*freq*t)
```

Show that the output is:



From the FFT plot, determine the frequencies in the signal:

5 Square waves

A square wave is produced using a harmonic frequency, and then 1/3 of the amplitude of the fundamental of the third frequency, then 1/5 of the amplitude for the fifth harmonic, and so on. Using the fundamental, 3rd, 5th and 7th harmonic, plot the waveform.

Outline the plot:

If we have a digital waveform with a time interval of 1ms, and filter so that all but the fundamental and third harmonic can pass. Which are the frequencies which are passed?

Can you draw what the waveforms will look like?

6 Other waveforms

A sawtooth waveform can be created from:

$f(t)=1/2+1/pi[\sin(\omega t)+\sin(2\omega t)/2+\sin(3\omega t)/3+\sin(4\omega t)/4+\sin(5\omega t)/5+.....]$

where $\omega = 2 * pi * f$

Write a Python program to create the waveform using the first nine frequencies. Then plot the output for a fundamental frequency of 5Hz: