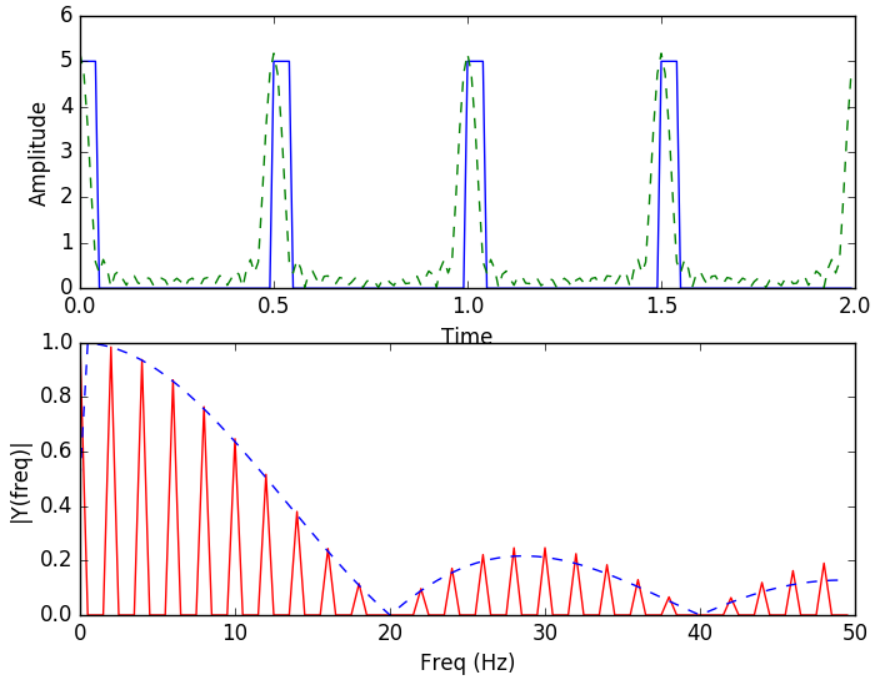
Pulse Analysis

Demo: https://youtu.be/jMIYdTF40HM

Plotting pulses

With repetitive digital signals we have a discrete frequency response, which has a fundamental frequency, and a number of harmonics. In the pulse train below we have a pulse period of 2Hz, thus the fundamental frequency is 2Hz, followed by 4Hz, 6Hz, and so on:



We can plot pulses and the frequency spectrum with:

http://www.asecuritysite.com/comms/pulse

Use the code from the link to create this Python program:

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

file ='1111'
```

```
freq=2.0
duty=0.2

if (len(sys.argv)>1):
        file=str(sys.argv[1])
if (len(sys.argv)>2):
        duty=float(sys.argv[2])
if (len(sys.argv)>3):
        freq=float(sys.argv[3])

Fs = 100.0;  # sampling rate
Ts = 1.0/Fs; # sampling interval

t = np.arange(0,2,Ts)

y = 2.5*signal.square(2 * np.pi * freq * t, duty=duty)+2.5

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 = 2*Y[range(n/2)]

fig,myplot = plot.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

frq =np.delete(frq,0,0)

y1=2*5*duty*np.sin(np.pi*duty*frq/freq)/(np.pi*duty*frq/freq)

y1=np.insert(y1,0,5*duty)
frq=np.insert(frq,0,0)

myplot[1].plot(frq,abs(y1),'--b')
myplot[1].set_xlabel('Freq (Hz)')
myplot[1].set_ylabel('|Y(freq)|')

#print frq
print 'Freq = 0 Hz\t',str((y1)[0])
print 'Freq =',frq[2*freq],'Hz\t',str(y1[2*freq])
print 'Freq =',frq[4*freq],'Hz\t',str(y1[4*freq])
print 'Freq =',frq[6*freq],'Hz\t',str(y1[6*freq])
```

```
print 'Freq =',frq[8*freq],'Hz\t',str(y1[8*freq])
print 'Freq =',frq[10*freq],'Hz\t',str(y1[10*freq])
print 'Freq =',frq[12*freq],'Hz\t',str(abs(Y[12*freq]))
print 'Freq =',frq[14*freq],'Hz\t',str(abs(Y[14*freq]))
print 'Freq =',frq[16*freq],'Hz\t',str(abs(Y[16*freq]))

h1=y1[0]
h2=y1[2*freq]
h3=y1[4*freq]
h4=y1[6*freq]
h5=y1[8*freq]
h6=y1[10*freq]
h7=y1[12*freq]

yupdate = h1+h2*np.cos(2 * np.pi * freq * t)+h3*np.cos(2*2 * np.pi * freq * t) \
+h4*np.cos(3*2 * np.pi * freq * t)+h5*np.cos(4*2 * np.pi * freq * t) \
+h6*np.cos(5*2 * np.pi * freq * t)+h7*np.cos(6*2 * np.pi * freq * t)

myplot[0].plot(t,abs(yupdate),'--g')

f1 = file+".svg"
plot.savefig(f1)

f2= file+".png"
plot.savefig(f2,format='PNG')

plot.show()
```

Now, in the table below, calculate the expected values for the frequency components, and the ones that the program shows:

Frequency	Calculated (2 sig fig)	Program
2Hz		
4Hz		
6Hz		
8Hz		

Duty cycle = 0.5
Freq=2Hz
Amplitude= 5

The data set for this is created with a sin(x)/x response:

```
y1=2*5*duty*np.sin(np.pi*duty*frq/freq)/(np.pi*duty*frq/freq)
```

Duty cycle

The duty cycle relates to the ratio of the pulse width to the reputation time. The frequency response and the average value of the pulse varies with the duty cycle.

For a Duty Cycle of the following, what would you estimate as the zero frequency component – the average value of the pulse train:

(i) Duty Cycle = 0.1, Pulse amplitude = 5V
(ii) Duty Cycle = 0.3, Pulse amplitude = 5V
(iii) Duty Cycle = 0.5, Pulse amplitude = 5V

Now run your code, and determine the "Freq = 0 Hz" value, and confirm your values.

Analyse the frequency response for a duty cycle of 0.1 and 0.9. What is the main difference between the two, and what similarities do they have?

Which of the responses is similar to a duty cycle of 0.2?

In the plot of the recovered signal – which is plotted in green – we show the first seven frequencies. Update the program so that it prints up to 15 frequencies.

What effect does it have on the reconstructed waveform?

If our channel restricts higher frequencies, which pulse train will be more affected. One with a duty cycle of 0.5 or one with a duty cycle of 0.1?

Reconstructing the signal

In this part of the tutorial we will construct the signal – shown in green in the charts. Modify the program so that it charts the constructed signal as a separate chart.

A clock of 2 kHz is used on system. If we block frequencies above a certain limit, and using your program, sketch the output pulses (note, just use the 2Hz signal and a 0.5 duty cycle, as it will be represented of the response):

If all the frequencies of 4 kHz and above are blocked:

If all the frequencies of 8 kHz and above are blocked:

How might we use this method in order to recover the clock rate of a signal?