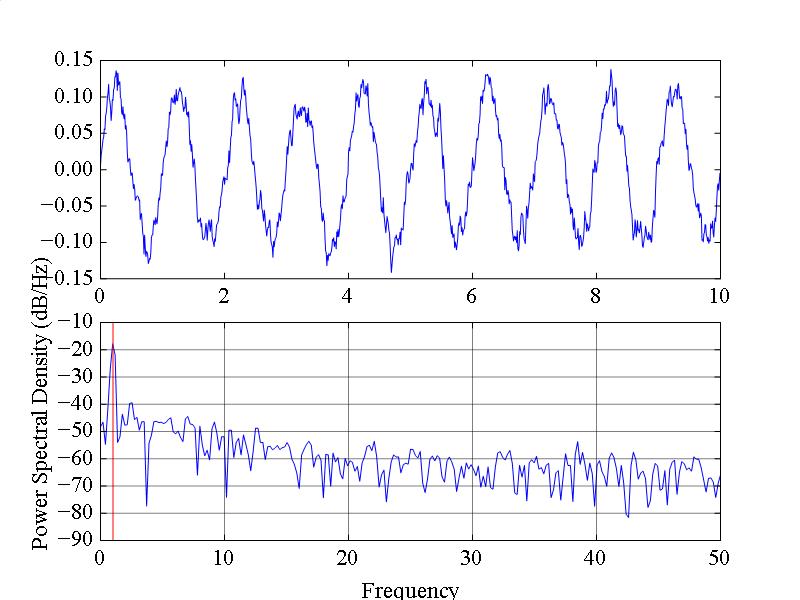
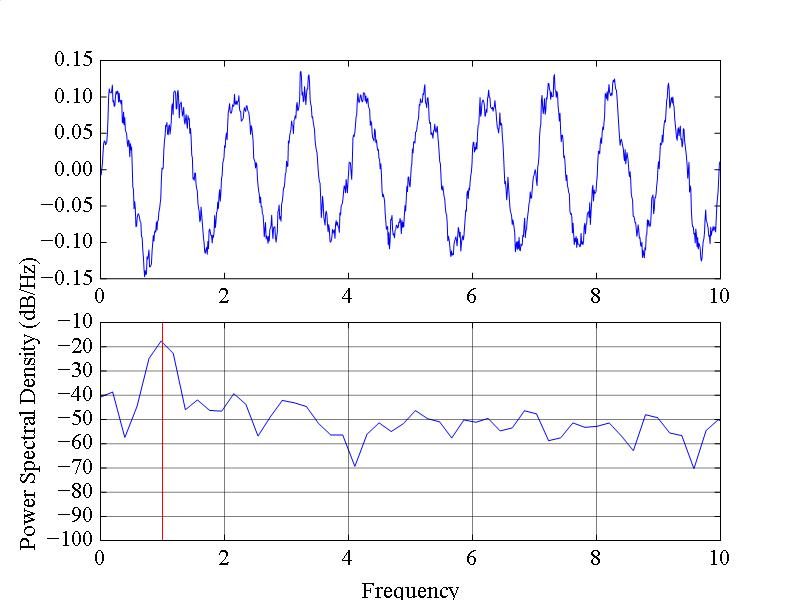
The Fourier Transform works by taking a function in time-space and converts it to frequency-space. It does this by approximating a function with a series of sine waves of all different frequencies such that when they are added together, they generate the original function. This implies that at each frequency, there is a corresponding amplitude for the sine wave. To convert to frequency-space, the amplitudes are plotted against the frequencies.

The graph below is the sample FFT performed on matplotlib’s website: <http://matplotlib.org/api/pyplot_api.html>

It graphs a sine wave with equation 0.1\*sin(2pi\*t). Therefore, we can expect that after performing the Fourier transform, there should be a large spike at a frequency of 1 hz, as this is the main frequency of the actual function. The random other amplitudes of a considerably lower value (remember that dB is a logarithmic scale) come as a result of the randomness artificially generated by the program. This randomness is normal of actual data. A red line is drawn at x=1.



This graph zoomed in shows that there is a peak at 1 hz.



from pylab import \*

dt = 0.01

t = arange(0,10,dt)

nse = randn(len(t))

r = exp(-t/0.05)

cnse = convolve(nse, r)\*dt

cnse = cnse[:len(t)]

s = 0.1\*sin(2\*pi\*t) + cnse

subplot(211)

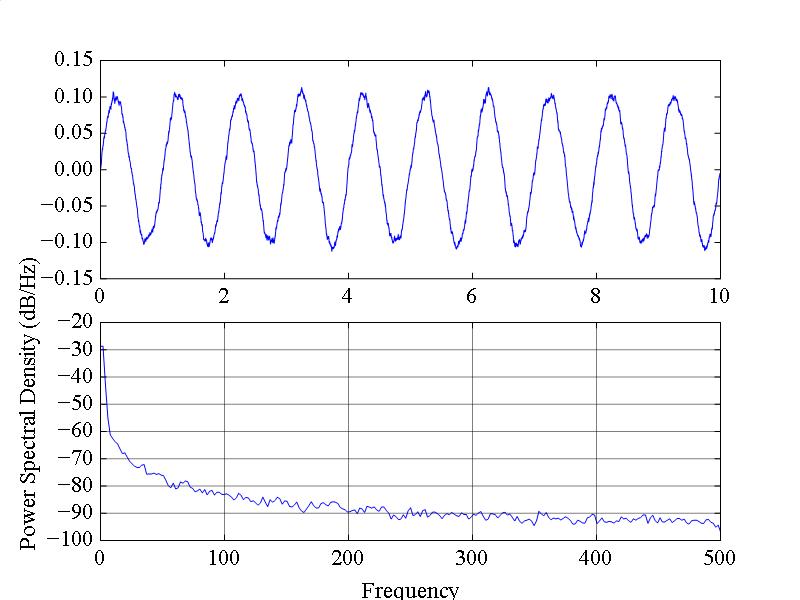
plot(t,s)

subplot(212)

psd(s, 512, 1/dt)

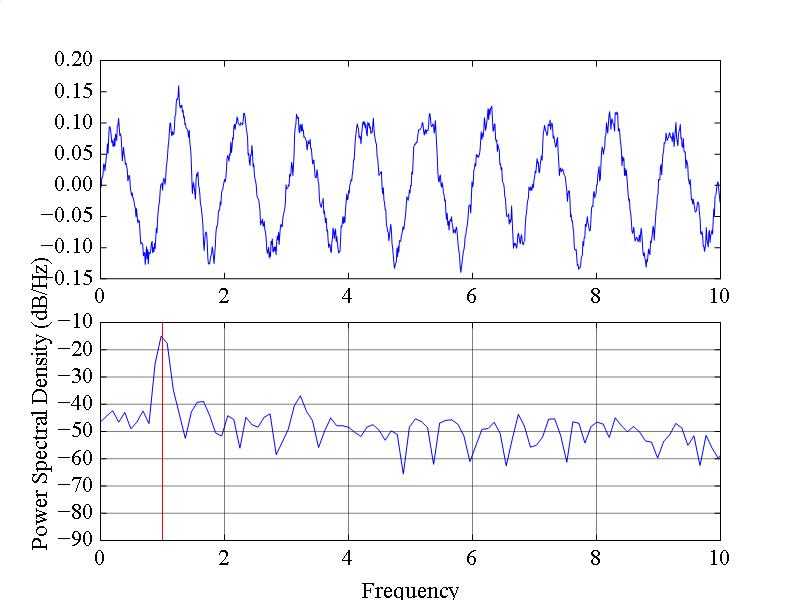
savefig('example1.jpg')

show()

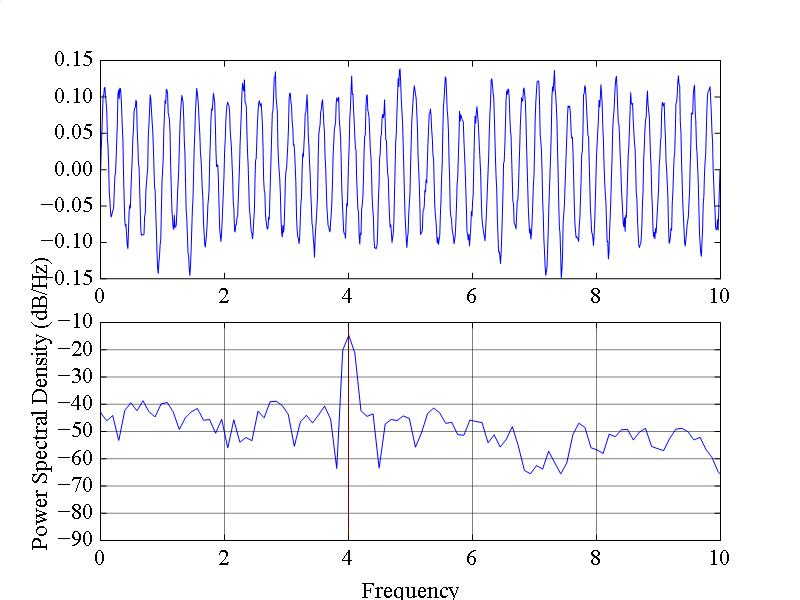
Above is the python code that generated the plot. Dt represents 1/Fs where Fs represents the sampling frequency. Notice that the sampling frequency, 100, is twice the total number of frequencies plotted after the Fourier transform. This is because the data starts to bounce back upon itself if frequencies greater than half the sampling frequency are tested. Below is the graph if dt = 0.001 or Fs = 1000. Notice that frequencies up to 500 are graphed via the psd function.

The actual ppd function according to matplotlib is matplotlib.pyplot.**psd**(x, NFFT=256, Fs=2, Fc=0, detrend=<function detrend\_none at 0x26a6ed8>, window=<function window\_hanning at 0x26aa140>, noverlap=0, pad\_to=None, sides='default', scale\_by\_freq=None, hold=None, \*\*kwargs)

Right now, in the program, the psd is used as psd(x, 512, 1/dt) We already discussed that 1/dt represents the frequency, but the meaning of NFFT is unclear. NFFT determines how many different frequency bins the Fourier Transform will test. Right now, NFFT = 512, but the default is 256. NFFT is most optimal if it is a power of 2. The number of frequencies that the Fourier transform will actually plot is NFFT/2. This is most clear by an example. dt = 0.01 and NFFT = 1024.



Just to show that the Fourier transform is in fact matching the actual frequency of the curve, if the function becomes 0.1\*sin(8\*pi\*t), there is a peak at 4 rather than 1.



So the sampling frequency determines the highest frequency at which the FFT will test, and the NFFT determines how many bins are being generated. So, the bandwidth, or the distance between each frequency point on the x-axis is dependent on both of those. Since the max frequency is Fs/2, and the number of bins is NFFT/2, the bandwidth is given by Fs/NFFT. Below is a graph which demonstrates this.

