PHY321F CP: Worksheet 3 **10 May 2005**

The fast Fourier transform (FFT) is an important and useful algorithm. In this worksheet we will explore some properties. Suppose we have some data, $g(t)$. This might, for example, be the amplitude of a pendulum as a function of time. The FFT enables us to determine the frequency spectrum $\hat{g}(\omega)$ of this data. (A pendulum is only harmonic in the small angle approximation.)

Start simply. Create some data $g(t) = sin(\omega t)$ in an array. We will plot out the data so be sure to import from `pylab` (or `visual.graph` if you prefer).

```
from pylab *
t=arange(1024.)
g=sin(0.1*t)
plot(t,g)
xlim(0.0,1023.0)  # set limits to x axis
show()            # don't forget this!
```

Explore various frequencies. Find a value that gives you many cycles in the plot.

The frequency content of the data can be explored by taking the FFT. Import the library: `from FFT import *` (this may need to be `Fft` on some versions of Windows). Take the FFT of the data: `ghat=fft(g)` (easy enough?). Plot it against frequency (for the moment: `freq=arange(1024.)`). Hit a problem? The FFT is *complex* so we must choose what to plot: `ghat.real` or `ghat.imag`; these real and imaginary parts are attributes of the `ghat` object. For a proper normalization, the FFT should probably be divided by the number of points in the data (useful function: `len(g)`).

You should now have a plot with *two* peaks. The FFT returns both positive and negative frequencies $-\omega_{\max}/2 \leq \omega \leq \omega_{\max}/2$, the latter packed by convention into the *top half* of the array. (This convention is not restricted to Python; in fact the FFT used by Python is from a standard Fortran package, FFTPACK).

We can restrict the plot to positive frequencies by using array *slices*:
`plot(freq[0:512],ghat[0:512].real)`

Why have we used 1024 points? The FFT works best with the number of points a power of two (the algorithm works by dividing the data into halves, quarters, etc). You can try 512, 4096, etc.

What about a proper scale for the time or frequency axis? Suppose the range of the time axis is T and the time step is Δt. Then the range of the frequency (not angular frequency!) axis is $1/\Delta t$ and the frequency step is $1/T$. [The best way of producing the time or frequency arrays, since we need a power of two, is probably something like `t=array( range(1024)*dt )` (this avoids potential floating point problems with `arange`). Alternatively, `pylab` has a function `linspace(tmin,tmax,N)` which will produce an array of the right size.] Use this to generate a proper time and frequency axis. Does the peak position correspond to the frequency you used?

Note that the frequency step is determined only by the *range* of the data, not by the time step! To make the frequency step smaller, i.e to observe lower frequencies, we need to