Python 9 assignment
Spectra of non-periodic signals

April 18, 2018

In the previous assignment we looked at functions that were periodic and extracted their spectra. The approach was:

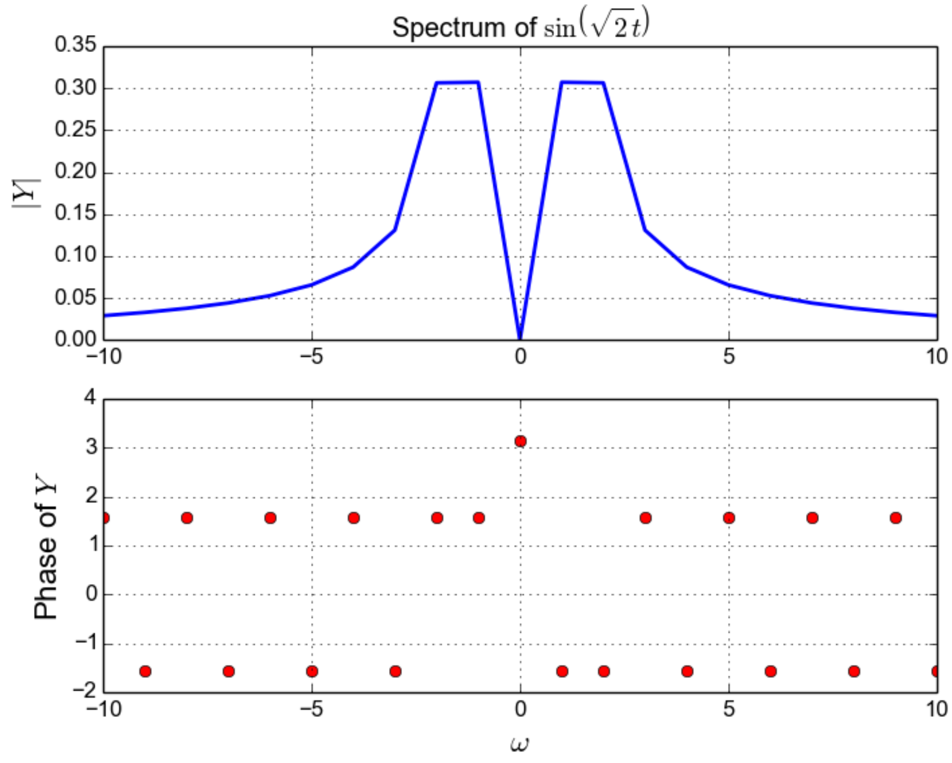
- Sample the signal so that $f_{Nyquist}$ is met, and so that Δf is small enough. Generate the frequency axis from $-f_{max}/2$ to $+f_{max}/2$, taking care to drop the last term.
- Ensure that the signal starts at $t = 0^+$ and ends at $t = 0^-$
- Use 2^k samples
- Obtain the DFT. Rotate the samples so that they go from $f = -f_{max}/2$ to $f = +f_{max}/2 - \Delta f$.
- Plot the magnitude and phase of the spectrum. Usually we would plot the magnitude in dB and the phase in degrees and the frequency axis would be logarithmic. This is to capture polynomial decay of the spectrum.

Now we want to look at non-periodic signals. Our first signal will be $\sin\left(\sqrt{2}t\right)$. Obtained over 0 to 2π with 64 samples, this function looks as follows:

1 $\langle * \ 1\rangle \equiv$
$\quad\langle \textit{eg1}\ 2\rangle$

2 ⟨*eg1* 2⟩≡ (1)

```
from pylab import *
t=linspace(-pi,pi,65);t=t[:-1]
dt=t[1]-t[0];fmax=1/dt
y=sin(sqrt(2)*t)
y[0]=0 # the sample corresponding to -tmax should be set zeroo
y=fftshift(y) # make y start with y(t=0)
Y=fftshift(fft(y))/64.0
w=linspace(-pi*fmax,pi*fmax,65);w=w[:-1]
figure()
subplot(2,1,1)
plot(w,abs(Y),lw=2)
xlim([-10,10])
ylabel(r"$|Y|$",size=16)
title(r"Spectrum of $\sin\left(\sqrt{2}t\right)$")
grid(True)
subplot(2,1,2)
plot(w,angle(Y),'ro',lw=2)
xlim([-10,10])
ylabel(r"Phase of $Y$",size=16)
xlabel(r"$\omega$",size=16)
grid(True)
savefig("fig10-1.png")
show()
```



We expected two spikes, but what we got were two peaks each with two values and a gradually decaying magnitude. The phase is correct though - but take a look at the code. There is one line there:

```
y[0]=0 # the sample corresponding to -tmax should be set zero
```

What is this for? An antisymmetric function has a purely imaginary fourier transform. But what about an antisymmetric set of samples? Suppose

$$\begin{aligned} y[0] &= 0 && \sin(0) \\ y[i] &= -y[N-i] && i = 1,2\ldots\frac{N}{2}-1 \\ y[\frac{N}{2}] &= \sin\left(t_{N/2}\right) && \sin(-t_{max}) \end{aligned}$$

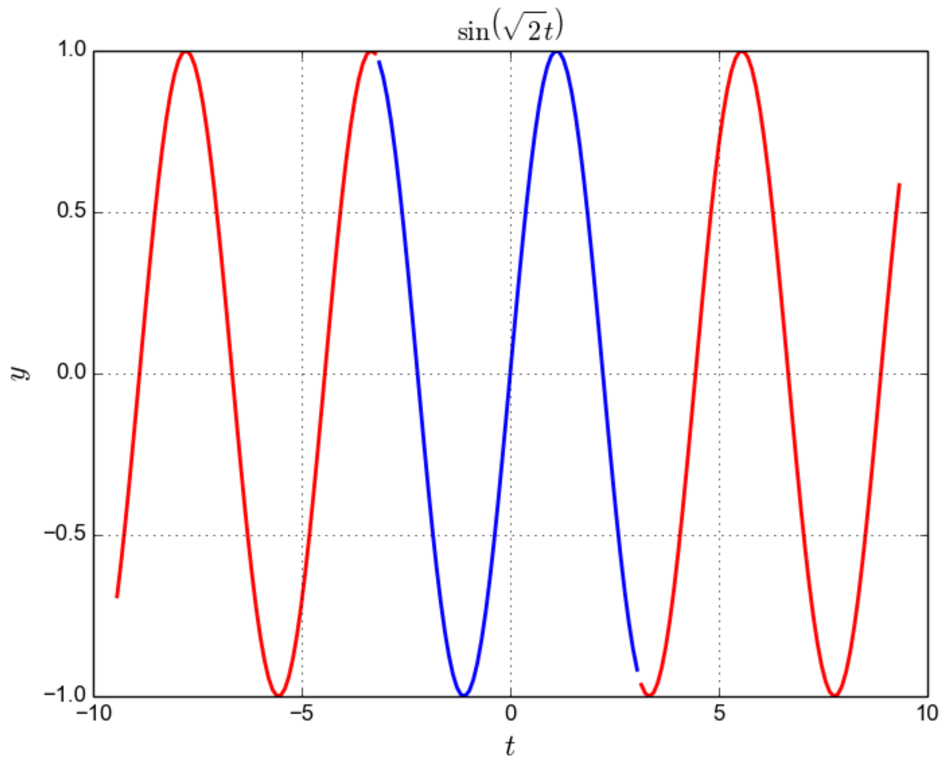
The DFT of this sequence will give us

$$\begin{aligned} Y[k] &= \sum_{n=0}^{N-1} y[n]\exp\left(\frac{2\pi j}{N}kn\right) \\ &= \sum_{n=1}^{N/2-1} y[n]\left(\exp\left(\frac{2\pi j}{N}kn\right) - \exp\left(-\frac{2\pi j}{N}kn\right)\right) + y[\frac{N}{2}]\exp(\pi k j) \\ &= \sum_{n=1}^{N/2-1} -2jy[n]\sin\left(\frac{2\pi}{N}kn\right) + (-1)^k y[\frac{N}{2}] \end{aligned}$$

But this is no longer pure imaginary! Since we need that property, we set $y[N/2]$ to zero. That gives us a purely imaginary phase. But what to do about the magnitude? And what went wrong?

To understand what went wrong, let us plot the time function over several time periods.

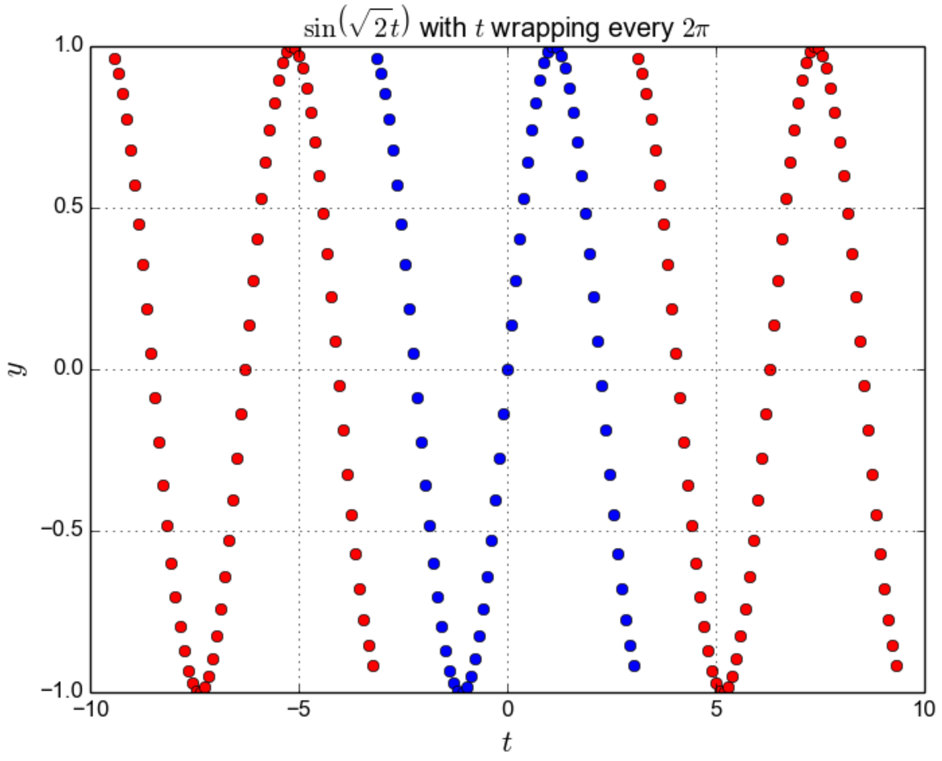
4 ⟨*eg2* 4⟩≡
```
from pylab import *
t1=linspace(-pi,pi,65);t1=t1[:-1]
t2=linspace(-3*pi,-pi,65);t2=t2[:-1]
t3=linspace(pi,3*pi,65);t3=t3[:-1]
# y=sin(sqrt(2)*t)
figure(2)
plot(t1,sin(sqrt(2)*t1),'b',lw=2)
plot(t2,sin(sqrt(2)*t2),'r',lw=2)
plot(t3,sin(sqrt(2)*t3),'r',lw=2)
ylabel(r"$y$",size=16)
xlabel(r"$t$",size=16)
title(r"$\sin\left(\sqrt{2}t\right)$")
grid(True)
savefig("fig10-2.png")
show()
```



The blue line connects the points whose DFT we took. The red lines show the continuation of the function. Quite clearly, even though $\sin\left(\sqrt{2}t\right)$ is a periodic function, the portion between $-\pi$ and π is not the part that can be replicated to generate the function. So which function is the DFT trying to fourier analyse? For that we have to replicate just the blue points. And that is shown below:

5 ⟨*eg3* 5⟩≡

```
from pylab import *
t1=linspace(-pi,pi,65);t1=t1[:-1]
t2=linspace(-3*pi,-pi,65);t2=t2[:-1]
t3=linspace(pi,3*pi,65);t3=t3[:-1]
y=sin(sqrt(2)*t1)
figure(3)
plot(t1,y,'bo',lw=2)
plot(t2,y,'ro',lw=2)
plot(t3,y,'ro',lw=2)
ylabel(r"$y$",size=16)
xlabel(r"$t$",size=16)
title(r"$\sin\left(\sqrt{2}t\right)$ with $t$ wrapping every $2\pi$ ")
grid(True)
savefig("fig10-3.png")
show()
```



Clearly this function is not $\sin\left(\sqrt{2}t\right)$ and that is why the DFT is not what we expect. But why did that create the problem we saw above?

Gibbs Phenomenon

This is something you know very well. The Fourier transform of the box function

$$f(t)=\begin{cases}1 & |t|\le t_0\\ 0 & |t|>t_0\end{cases}$$

is given by

$$F(\omega)=\frac{2\sin(\omega t_0)}{\omega}$$

The spectrum of the box function decays very slowly, as $2/\omega$.

Now our function is an odd function with a big jump. So let us consider the periodic ramp:

$$f(t)=t,\ -\pi<t<\pi$$

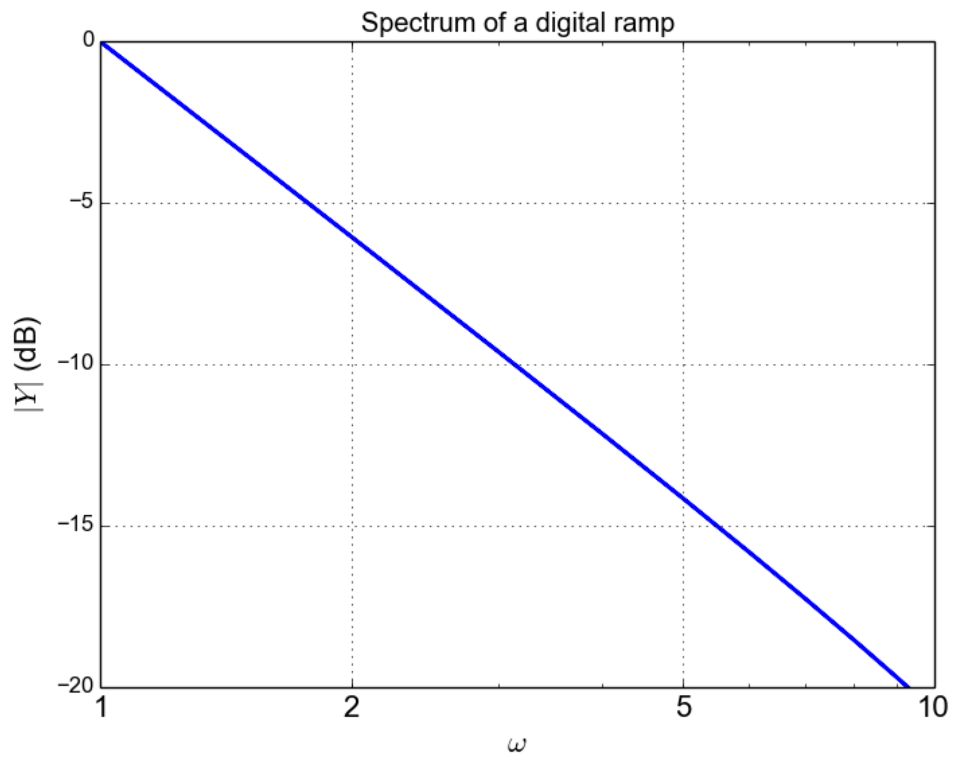
Then the fourier series of this ramp is

$$f(t)=2\left(\frac{\sin t}{1}-\frac{\sin 2t}{2}+\frac{\sin 3t}{3}-\dots\right)$$

Again the coefficients decay very slowly.

The DFT is just like the fourier series, except that both time and frequency are samples. So, if the time samples are like a ramp, the frequency samples will decay as $1/\omega$. Let us verify this for the ramp itself

7 ⟨*eg4* 7⟩≡
```
from pylab import *
t=linspace(-pi,pi,65);t=t[:-1]
dt=t[1]-t[0];fmax=1/dt
y=t
y[0]=0 # the sample corresponding to -tmax should be set zeroo
y=fftshift(y) # make y start with y(t=0)
Y=fftshift(fft(y))/64.0
w=linspace(-pi*fmax,pi*fmax,65);w=w[:-1]
figure()
semilogx(abs(w),20*log10(abs(Y)),lw=2)
xlim([1,10])
ylim([-20,0])
xticks([1,2,5,10],["1","2","5","10"],size=16)
ylabel(r"$|Y|$ (dB)",size=16)
title(r"Spectrum of a digital ramp")
xlabel(r"$\omega$",size=16)
grid(True)
savefig("fig10-4.png")
show()
```



Clearly the spectrum decays as 20 dB per decade, which corresponds to $1/\omega$. The big jumps at $n\pi$ force this slowly decaying spectrum, which is why we don't see the expected spikes for the spectrum of $\sin\left(\sqrt{2}t\right)$.

Windowing

So what do we do? Well the spikes happen at the end of the periodic interval. So we damp the function near there, i.e., we multiply our function sequence $f[n]$ by a "window" sequence $w[n]$:

$$g(n) = f(n)w(n)$$

The new spectrum is got by convolving the two fourier transforms:

$$G_k = \sum_{n=0}^{N-1} F_n W_{k-n}$$

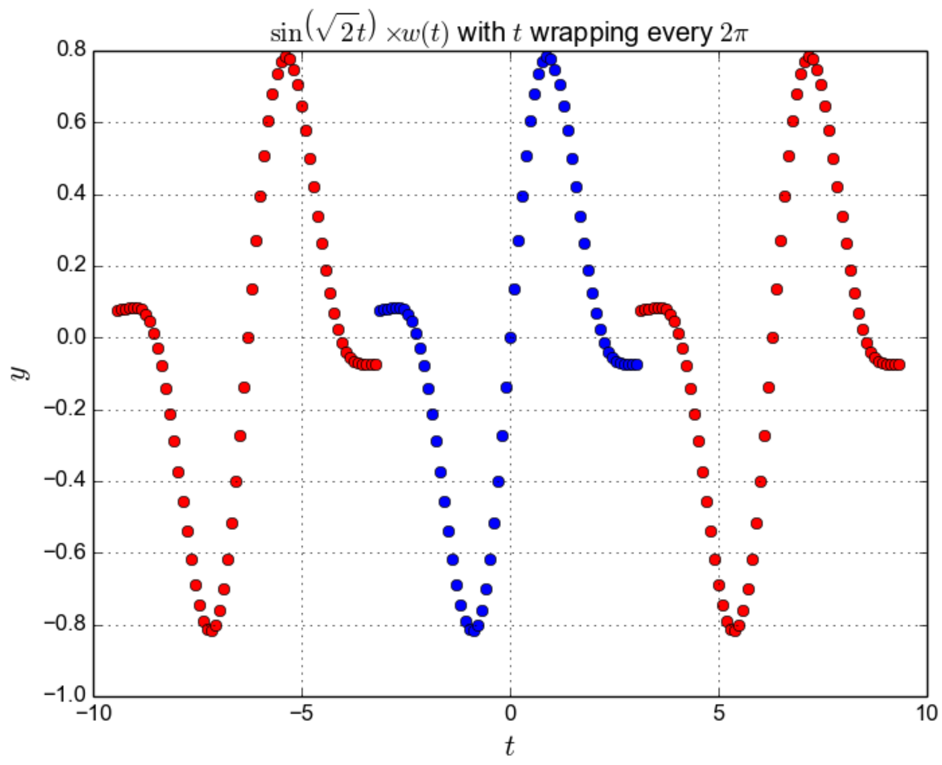
Suppose f_n is a sinusoid. Then F_k has two spikes. But the two spikes are now smeared out by W_k. So we expect to get broader peaks. But what this also does is to suppress the jump at the edge of the window. The window we will use is called the Hamming window:

$$w[n] = \begin{cases} 0.54 + 0.46\cos\left(\frac{2\pi n}{N-1}\right) & |n| \le \frac{N-1}{2} \\ 0 & \text{else} \end{cases}$$

Let us look at our time sequence for $\sin\left(\sqrt{2}t\right)$ now . . .

8 ⟨*eg5* 8⟩≡
```
from pylab import *
t1=linspace(-pi,pi,65);t1=t1[:-1]
```

```
t2=linspace(-3*pi,-pi,65);t2=t2[:-1]
t3=linspace(pi,3*pi,65);t3=t3[:-1]
n=arange(64)
wnd=fftshift(0.54+0.46*cos(2*pi*n/63))
y=sin(sqrt(2)*t1)*wnd
figure(3)
plot(t1,y,'bo',lw=2)
plot(t2,y,'ro',lw=2)
plot(t3,y,'ro',lw=2)
ylabel(r"$y$",size=16)
xlabel(r"$t$",size=16)
title(r"$\sin\left(\sqrt{2}t\right)\times w(t)$ with $t$ wrapping every $2\pi$ ")
grid(True)
savefig("fig10-5.png")
show()
```



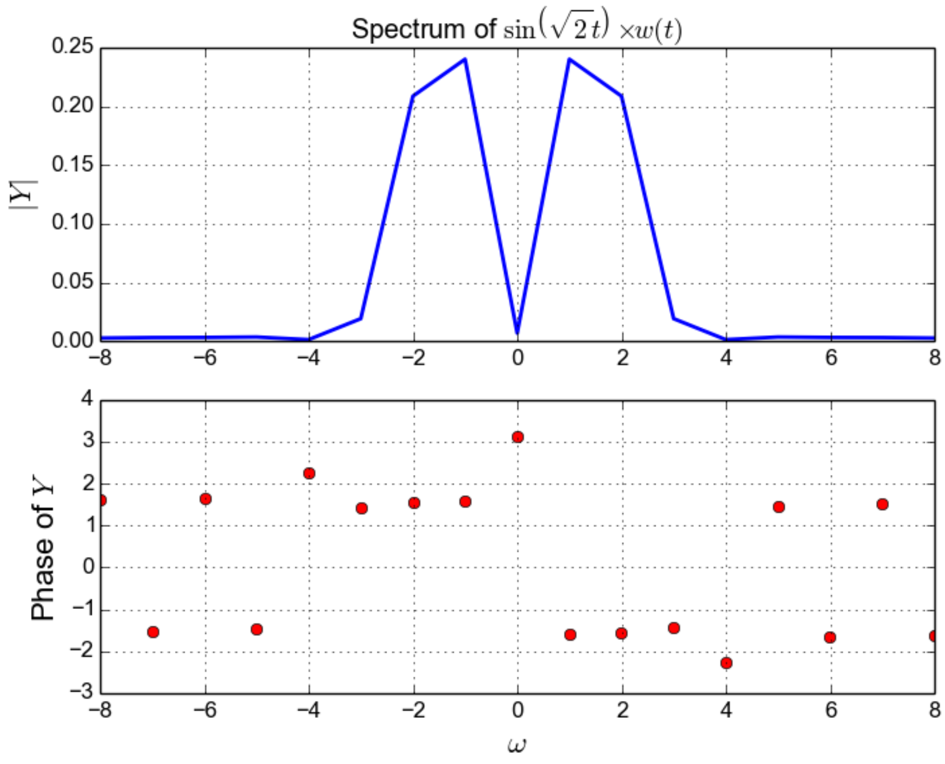
The jump is still there, but it is much reduced. There is a little bit of magic in keeping some of the jump - it gives us an extra 10 db of suppression.

Now let us take the DFT of this sequence and see what we get:

10 ⟨*eg6* 10⟩≡

```
from pylab import *
t=linspace(-pi,pi,65);t=t[:-1]
dt=t[1]-t[0];fmax=1/dt
n=arange(64)
wnd=fftshift(0.54+0.46*cos(2*pi*n/63))
y=sin(sqrt(2)*t)*wnd
y[0]=0 # the sample corresponding to -tmax should be set zeroo
y=fftshift(y) # make y start with y(t=0)
Y=fftshift(fft(y))/64.0
w=linspace(-pi*fmax,pi*fmax,65);w=w[:-1]
figure()
subplot(2,1,1)
plot(w,abs(Y),lw=2)
xlim([-8,8])
ylabel(r"$|Y|$",size=16)
title(r"Spectrum of $\sin\left(\sqrt{2}t\right)\times w(t)$")
grid(True)
subplot(2,1,2)
plot(w,angle(Y),'ro',lw=2)
xlim([-8,8])
ylabel(r"Phase of $Y$",size=16)
xlabel(r"$\omega$",size=16)
grid(True)
savefig("fig10-6.png")
```

```
show()
```

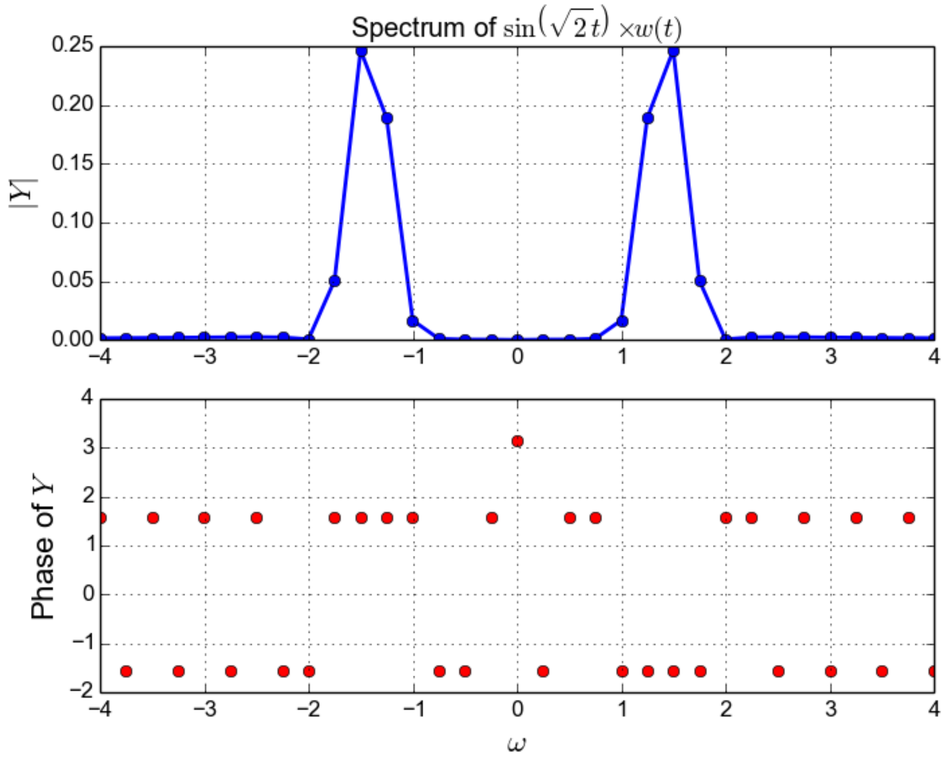


Compare to our first plot and you can see that the magnitude is greatly improved. We still have a peak that is two samples wide. But that is because $\sqrt{2}$ lies between 1 and 2, which are the two fourier components available. If we use four times the number of points we should get better results.

12 ⟨*eg7* 12⟩≡

```
from pylab import *
t=linspace(-4*pi,4*pi,257);t=t[:-1]
dt=t[1]-t[0];fmax=1/dt
n=arange(256)
wnd=fftshift(0.54+0.46*cos(2*pi*n/256))
y=sin(sqrt(2)*t)
# y=sin(1.25*t)
y=y*wnd
y[0]=0 # the sample corresponding to -tmax should be set zeroo
y=fftshift(y) # make y start with y(t=0)
Y=fftshift(fft(y))/256.0
w=linspace(-pi*fmax,pi*fmax,257);w=w[:-1]
figure()
subplot(2,1,1)
plot(w,abs(Y),'b',w,abs(Y),'bo',lw=2)
xlim([-4,4])
ylabel(r"$|Y|$",size=16)
title(r"Spectrum of $\sin\left(\sqrt{2}t\right)\times w(t)$")
grid(True)
subplot(2,1,2)
plot(w,angle(Y),'ro',lw=2)
xlim([-4,4])
ylabel(r"Phase of $Y$",size=16)
```

```
xlabel(r"$\omega$",size=16)
grid(True)
savefig("fig10-7.png")
show()
```



Is it better? Well it is quite a bit better since we are now zoomed in and see a lot more detail. But why is it not just a single peak? The reason for that is $w(t)$. Multiplication in time is convolution in frequency and vice versa. So by multiplying with $w(t)$, we got rid of the $1/f$ decay. But the delta function is now replaced by the shape of the DFT of $w[n]$. That gives us a factor of two broadening over the peak when there is no window, which is why we still see a peak whose width is two samples.

Note that it is *not* because $\sqrt{2}$ is between 1.25 and 1.5. To verify, there is an alternate function in the above code, namely $\sin(1.25t)$. But this gives a broad peak as well. That is because of $w[n]$.

The Assignment

1. Work through the example codes and understand them.

2. Consider the function $\cos^3(\omega_0 t)$. Obtain its spectrum for $\omega_0 = 0.86$ with and without a Hamming window.

3. Write a program that will take a 128 element vector known to contain $\cos(\omega_0 t + \delta)$ for arbitrary δ and $0.5 < \omega_0 < 1.5$. The values of t go from $-\pi$ to π. You have to extract the digital spectrum of the signal, find the two peaks at $\pm\omega_0$, and estimate ω_0 and δ.

4. Suppose the data in Q3 has added "white gaussian noise". This can be generated by randn() in python. The extent of this noise is 0.1 in amplitude (i.e., 0.1*randn(N), where N is the number of samples). Repeat the problem and find the ω_0 and δ

5. Plot the DFT of the function

$$\cos\left(16\left(1.5+\frac{t}{2\pi}\right)t\right)$$

for t going from $-\pi$ to π in 1024 steps. This is known as a "chirped" signal, and its frequency continuously changes from 16 to 32 radians per second. This also means that the period is 64 samples near $-\pi$ and is 32 samples near $+\pi$.

6. For the same chirped signal, break the 1024 vector into pieces that are 64 samples wide. Extract the DFT of each and store as a column in a 2D array. Then plot the array as a surface plot to show how the frequency of the signal varies with *time.*

   This is new. So far we worked either in time or in frequency. But this is a "time-frequency" plot, where we get localized DFTs and show how the spectrum evolves in time.