Assignment No 9

Subhankar Chakraborty

April 10, 2019

1 Spectrum of non-periodic signals

1.1 Spectrum of $sin\sqrt{2}t$

In the earlier assignment we found out the DFT of periodic signals. Here, we are going to find the spectrum of non periodic signals. Let's start with $sin\sqrt{2}t$. Obtained over 0 to 2π with 64 samples, the functions looks like. The code for the above is

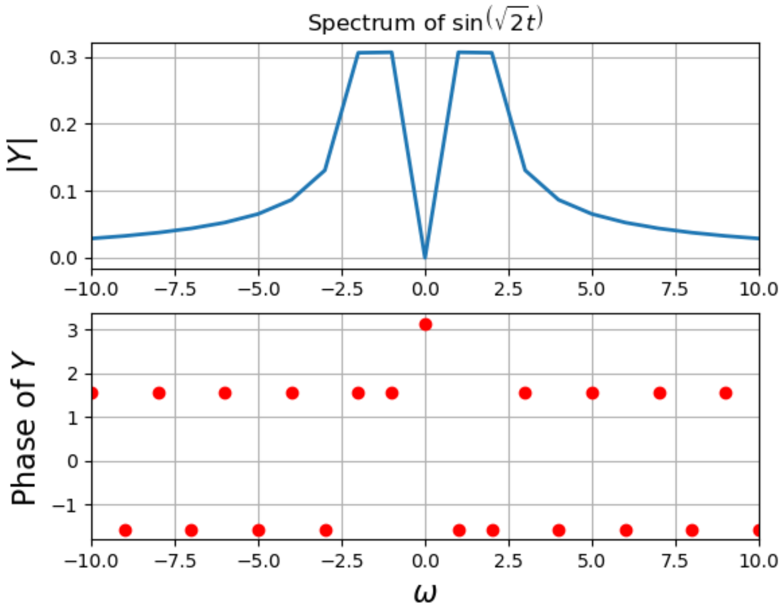

Figure 1: DFT of $sin\sqrt{2}t$

```
y=sin(sqrt(2)*t)
y[0]=0
y=fftshift(y)
Y=fftshift(fft(y))/64.0
```

```
w=linspace(-pi*fmax,pi*fmax,65);w=w[:-1]
```

The *y[0]=0* ensures that the specturm of an odd symmetric signal is purely imaginary. We notice a few issues with the magnitude plot. Instead of having two sharp peaks, we have 4 peaks here. How can that be explained? Let's look at the function over several time periods.

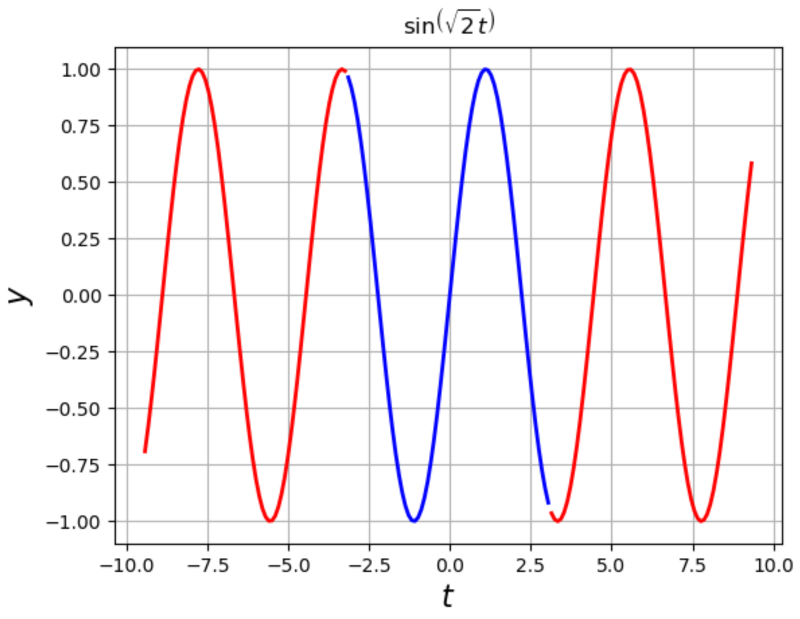

Figure 2: Time domain plot of $sin\sqrt{2}t$

The points we are sampling correspond to the blue region of the plot. It should be obvious by now that we cannot recreate the original signal by repeating the blue part periodically. So what exactly is the DFT doing? It is pretty clear from figure-3 that whatever the DFT is trying to analyse is not $sin\sqrt{2}t$. So what exactly is the issue?

- There is a big jump near 0 and *pi*.
- When we try to place such signals adjacent to each other the jump ensures that we have a big error. To get a correct spectrum, we must take care of the jumps.

1.2 Windowing

We multiply our signal with a window to take care of the jumps. The window we use here will be the hamming window(w[n]).The hamming window is

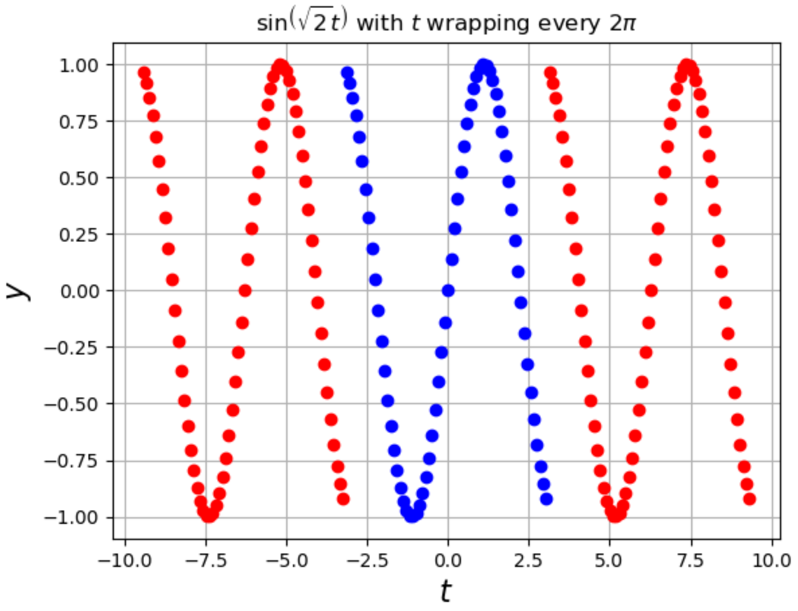

Figure 3: Periodically repeated values of the samples we are taking

given by

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

Let's look at $sin\sqrt{2}t$ multiplied by w(t) now.

The situation is much better now as the discontinuity is of a much smaller magnitude. Let's take the DFT of this signal and see what do we get. 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.

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.

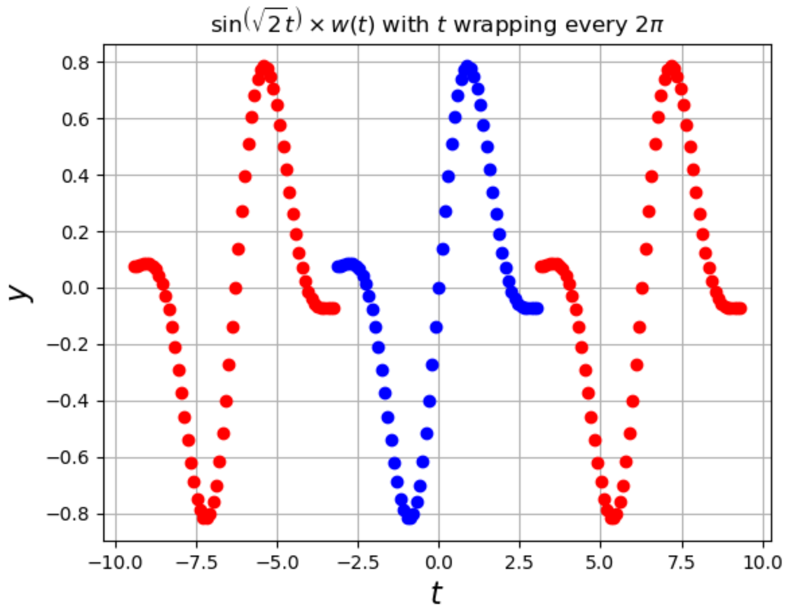

Figure 4: Periodically repeated values of $sin\sqrt{2}t * w(t)$

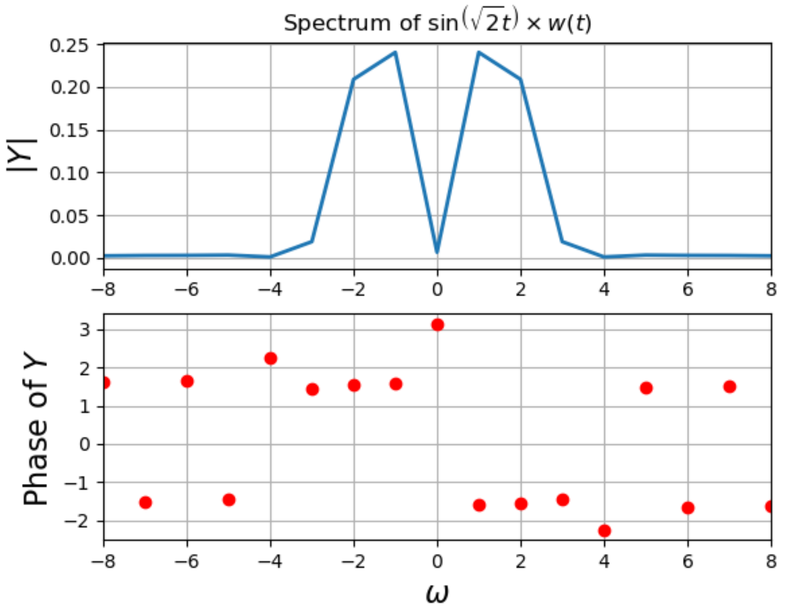

Figure 5: Spectrum of $sin\sqrt{2}t * w(t)$

2 Spectrum of $cos^3(\omega_0 t)$

Note that the value of ω_0 here is 0.86.

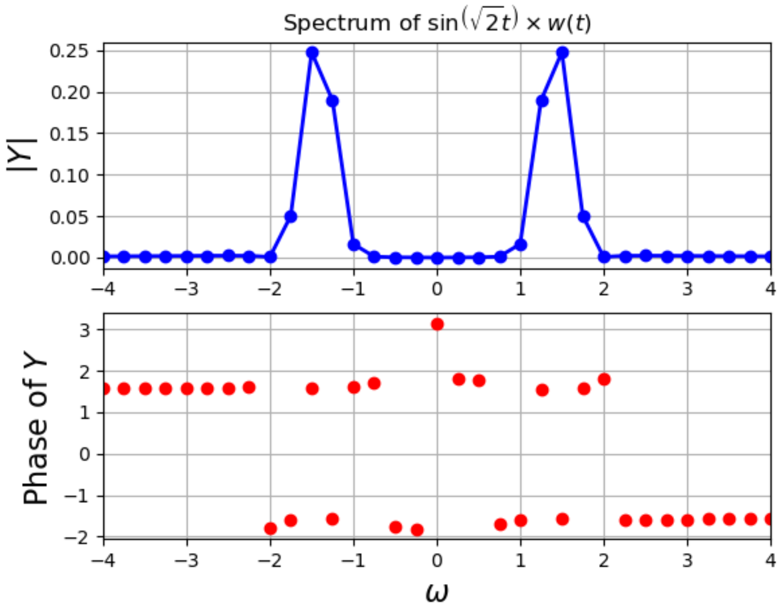

Figure 6: Spectrum of $sin\sqrt{2t} * w(t)$

2.1 Without windowing

Figure-7 shows the spectrum without windowing. As expected, we get peaks near ω_0 and $3\omega_0$. However, we get other peaks as well. Also the spectrum for values of ω=0 is not exactly zero. All this happens because $cos\omega_0(t)$ is not an exact periodic function in the discrete time domain.

2.2 With windowing

The spectrum after windowing is given in figure-8.

The following improvements are obvious.

- The low frequency contents in the samples are almost totally killed.
- The peaks around ω_0 and $3\omega_0$ are much sharper.

3 Finding the frequency from a given vector

We are given a 128 length vector which corresponds to values of a function $cos(\omega_0 t + \delta)$ where t goes from pi to π. We have to estimate ω_0 and δ. How do we go about generating this signal in the first place? The following code generates this random signal.First we generate this random signal and find its spectrum.

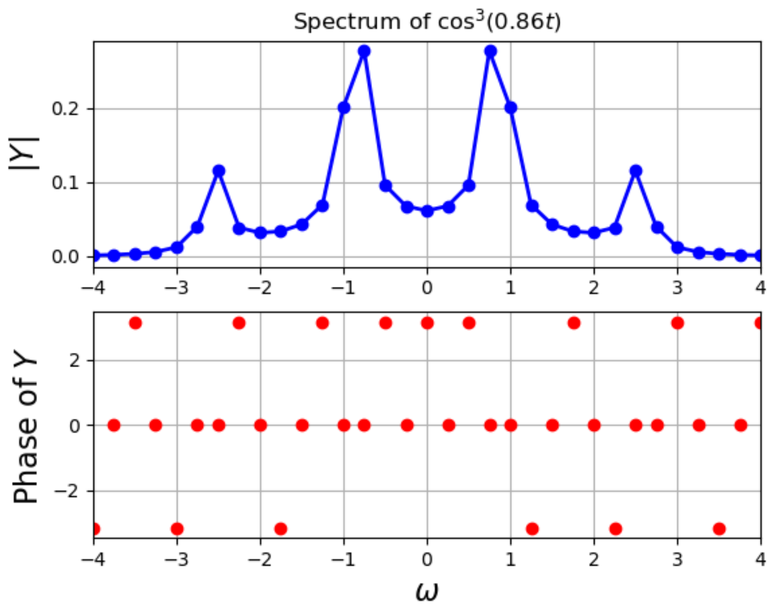

Figure 7: Spectrum of $cos^3(\omega_0 t)$

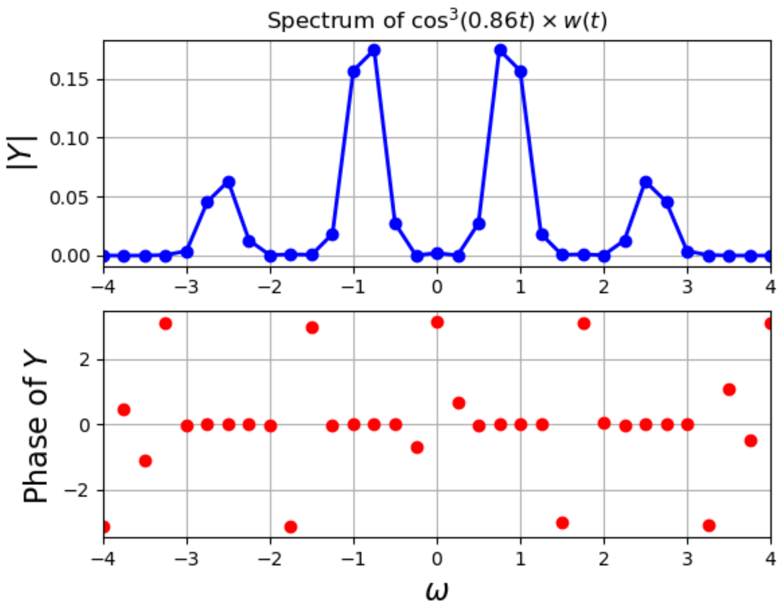

Figure 8: Spectrum of $cos^3(\omega_0 t) * w(t)$

```
Omega = np.random.normal()
delta = np.random.uniform(low=-pi,high=pi)
```

```
y=cos(Omega*t+delta)
wnd=fftshift(0.54+0.46*cos(2*pi*n/127))
y=y*wnd
y=fftshift(y)
Y=fftshift(fft(y))/128
w=linspace(-pi*fmax,pi*fmax,129);w=w[:-1]
```

Now, the manner in which ω is defined, if we just take the value at which the peak occurs, we will get either 1 or zero only as the resolution is limited to 1 integer only. So we try something like a power spectral density analysis.

- We take weighted values of ω, the weights being some power of the energy carried by that frequency.
- We divide it by the total weights to get a normalized frequency.
- What exponent of the energy is to be takes is chosen by taking multiple random signals and choosing the value which gives closest real and predicted values of

The following code does it.

```
predicted_omega = sum(abs(Y)**2.1*abs(w))/sum(abs(Y)**2.1)
```

For finding the value of δ, we can roughly assume that the phase at the peak(the rest is a cosine function) is due to delta only. That will give us the value of delta. The following code does it.

```
N_max = np.argmax(abs(Y))
predicted_delta = -angle(Y[N_max])
```

We get a maximum deviation of about 5% from this method. One example case is
real ω=1.078
real δ =2.35
predicted ω=1.079
predicted δ=2.34
The plots for one of the cases is given in figure-9. Note that the phase has been plotted only for points for which the magnitude is non-negligible.

4 Frequency of a noise corrupted signal

It is given that the signal is corrupted by a noise here. The noise is known to be normally distributes with an amplitude of 0.1. We proceed with a mathod similar to the previous section.

- The method to determine δ remains practically same.

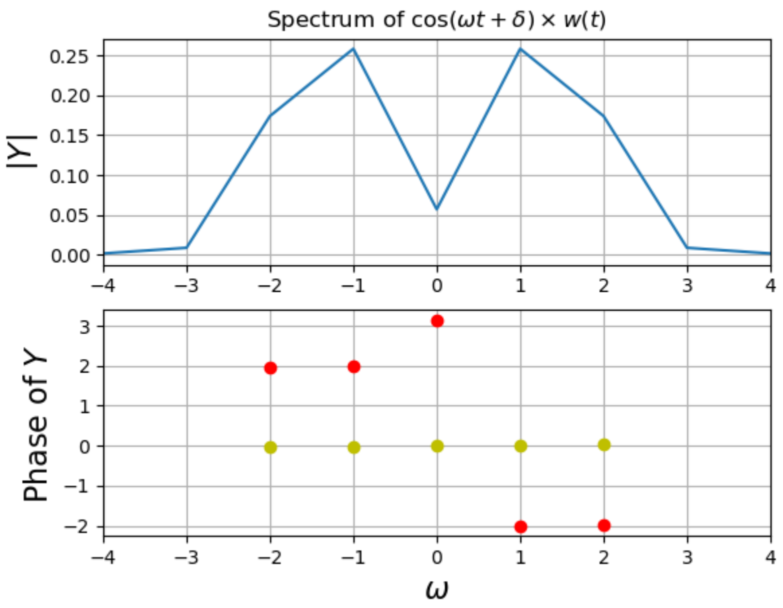

Figure 9: Spectrum of $cos(\omega_0 t + \delta) * w(t)$

- The method to determine ω_0 is similar with higher weightings to the energy values to compensate for the energy carried by the noise.

The code sections shown below are used.

```
N_max = np.argmax(abs(Y))
predicted_delta = -angle(Y[N_max])
a = sum(abs(Y)**(2.8)*abs(w))/sum(abs(Y)**2.8)
```

One particular set of values is real ω=1.33
real δ=-0.62
predicted noisy ω=1.36
predicted noisy δ=-0.63
A plot for one such case is

Note that the yellow dots here correspond to the phase of cos(0.5t), to be used as a reference that the phase plots for cos(ωt) is zero for real ω at points where the DFT has significant magnitude.

5 Chirped signal

The signal represented by

$$f(t) = cos(16(1.5 + \frac{t}{2\pi})t)$$

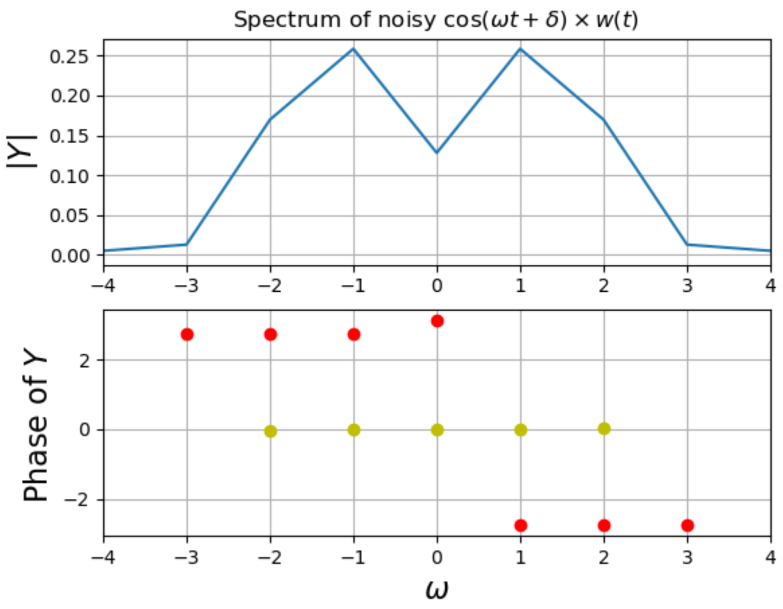

Figure 10: Spectrum of noisy $cos(\omega_0 t + \delta) * w(t)$

is known as a chirped signal. It's frequency increases linearly from 16 radians per second to 32 radians per second. This also means the period is 64 samples near $-\pi$ and 32 samples near π.We take t going from $[-\pi, \pi)$ in 1024 steps. We plot the signal first. This is given in figure-11

From the figure it's clear that the frequency is continuously increasing from $-\pi$ to π. The spectrum of the chirped signal(after windowing) is given in figure-12.

We see most of the frequencies are concentrated around 2 bands.Now we break the signal into vectors of length of 64 and find out their spectrum. We plot the spectrum as a function of time and the n^{th} sample. The following code does it.

```
y_full = np.zeros((16,64), dtype = np.complex)
for k in range(16):
    y_temp = y[64*k:64*(k+1)]
    # y_temp_fft = np.fft.fft
    y_temp=fftshift(y_temp)
    Y_temp=fftshift(fft(y_temp))/64
    y_full[k] = Y_temp
```

The surface plot of the magnitude of the spectrum as a function of the bin index(N) and time step(from 1 to 16) is given in figure-13.

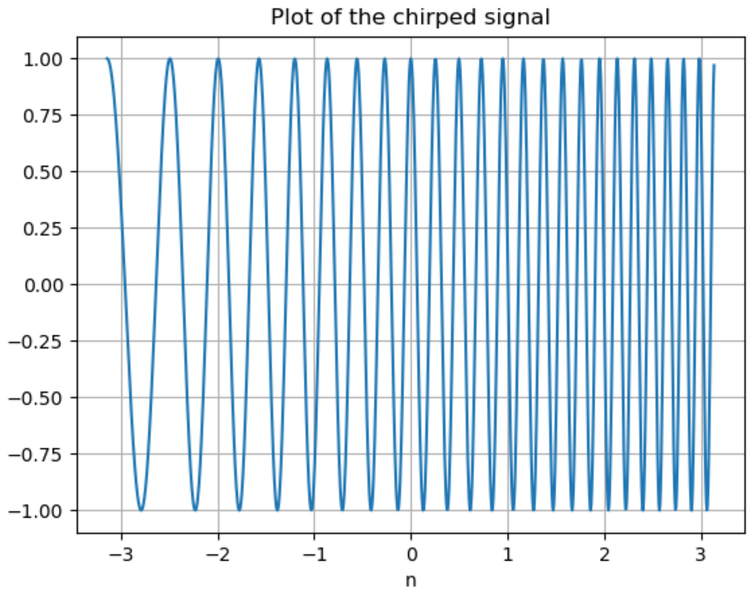

Figure 11: $cos(16(1.5 + \frac{t}{2\pi})t)$

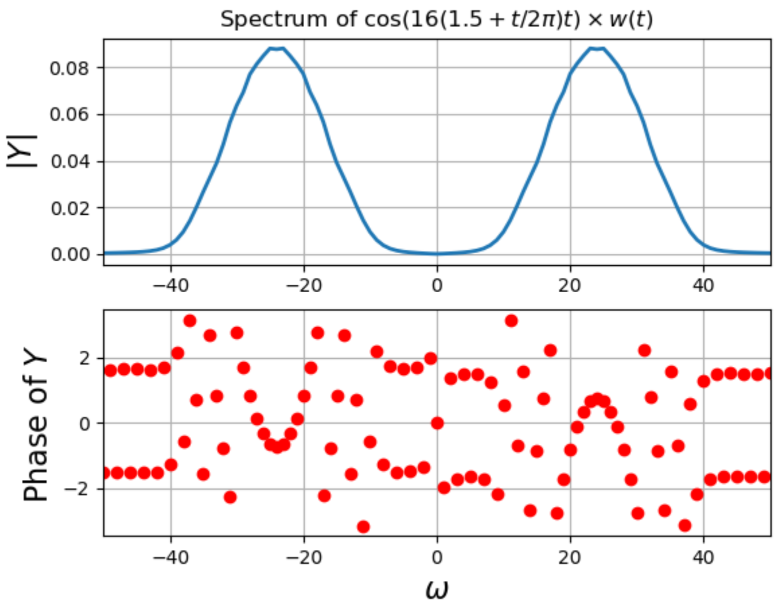

Figure 12: Spectrum of $cos(16(1.5 + \frac{t}{2\pi})t)$

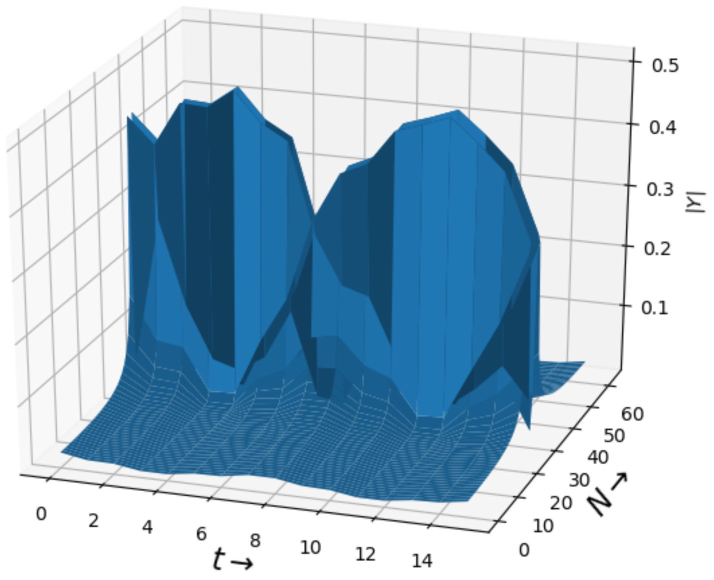

Figure 13: Surface plot of DFT of $cos(16(1.5 + \frac{t}{2\pi})t)$