Assignment 9: The Spectra of Non-Periodic Signals

EE18B120: Sarvani Cheruvu

1 Introduction

Continuing our analysis of signals using Fourier transforms, we find frequency spectrums of non-periodic functions in this assignment.

2 Theory

Non-periodic functions have a discontinuity when they are extended periodically. The discontinuity manifests as a $1/w$ decay due to Gibbs phenomenon. To avoid this, we remove the higher frequency components that are causing the discontinuity, by multiplying with a parameter called the 'Hamming window'.

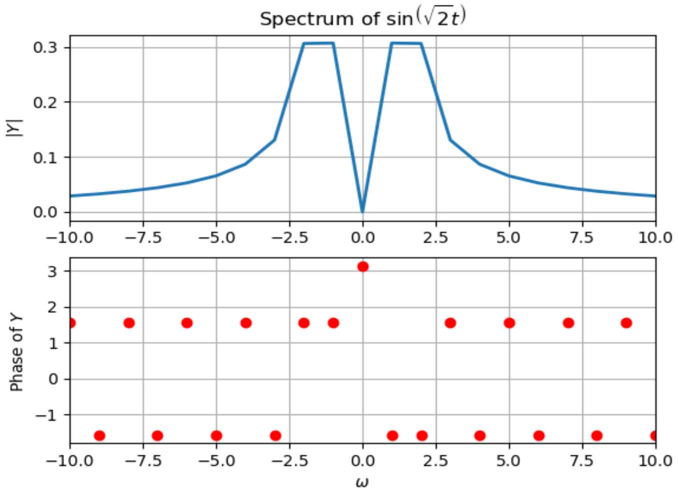
$$x[n] = 0.54 + 0.46$$

3 Assignment

3.1 Part 1

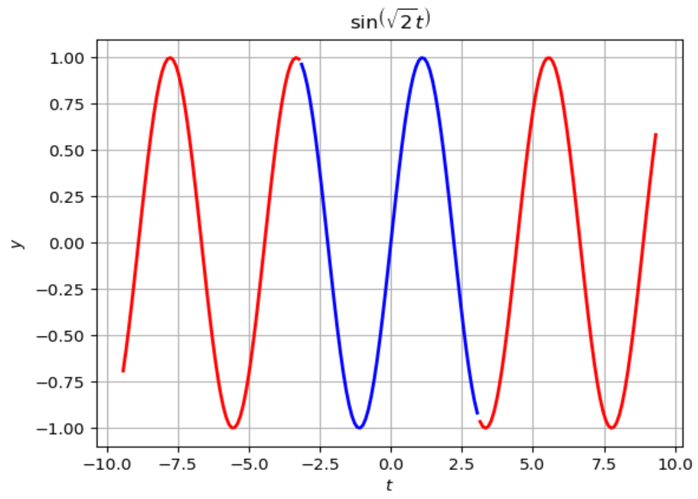
Spectrum of $\sin\sqrt{2}t$ is calculated using the code given below.

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



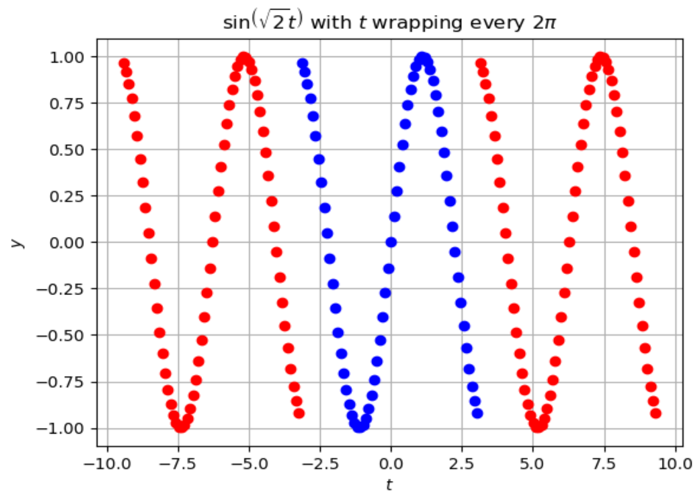
This is the original function for which we want DFT (i.e the portion of the graph is extended on both sides to check if the portion between $-\pi$ and π is the one that needs to be replicated.
The code is listed below.

```
t1 = np.linspace(-1*np.pi,np.pi,65)
t1 = t1[:-1]
t2 = np.linspace(-3*np.pi,-1*np.pi,65)
t2 = t2[:-1]
t3 = np.linspace(np.pi,3*np.pi,65)
t3 = t3[:-1]
plt.plot(t1, np.sin(np.sqrt(2)*t1),'b',lw=2)
plt.plot(t2,np.sin(np.sqrt(2)*t2),'r',lw=2)
plt.plot(t3,np.sin(np.sqrt(2)*t3),'r',lw=2)
plt.ylabel(r"$y$")
plt.xlabel(r"$t$")
plt.title(r"$\sin\left(\sqrt{2}t\right)$")
plt.grid(True)
plt.show()
```



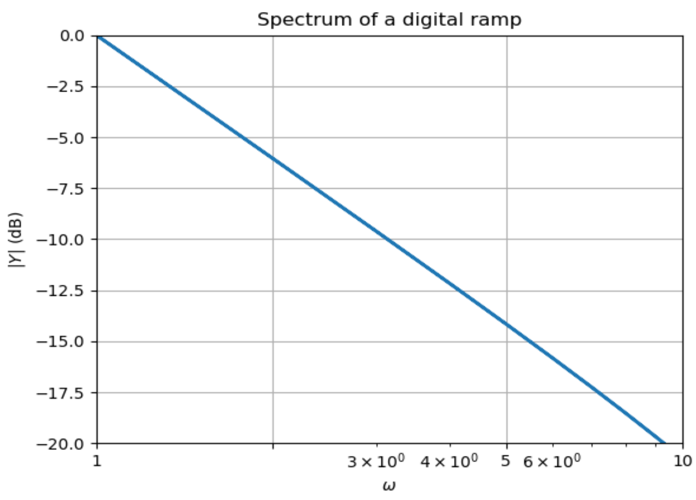
Since DFT is computed over a finite time interval, we actually plot the DFT for this function.

```
y=np.sin(np.sqrt(2)*t1)
plt.plot(t1,y,'bo',lw=2)
plt.plot(t2,y,'ro',lw=2)
plt.plot(t3,y,'ro',lw=2)
plt.ylabel(r"$y$")
plt.xlabel(r"$t$")
plt.title(r"$\sin\left(\sqrt{2}t\right)$ with $t$ wrapping every $2\pi$")
plt.grid(True)
plt.show()
```



There are some discontinuities which lead the FFT's non-harmonic components to decay as $1/omega$. To verify this, we plot the spectrum of the ramp:

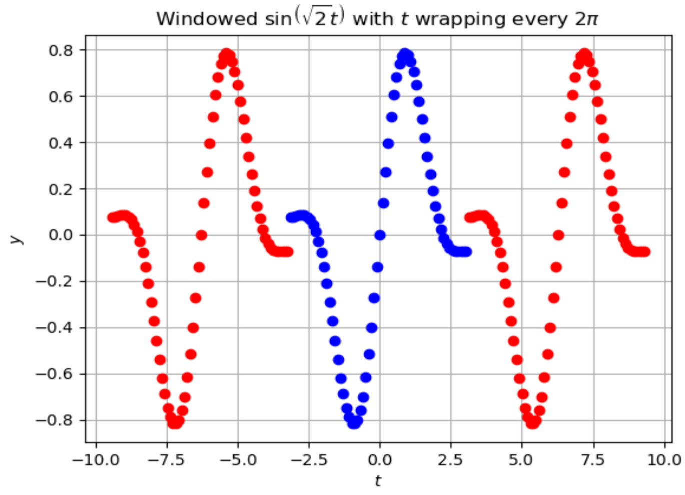
```
y=np.sin(np.sqrt(2)*t1)
plt.plot(t1,y,'bo',lw=2)
plt.plot(t2,y,'ro',lw=2)
plt.plot(t3,y,'ro',lw=2)
plt.ylabel(r"$y$")
plt.xlabel(r"$t$")
plt.title(r"$\sin\left(\sqrt{2}t\right)$ with $t$ wrapping every $2\pi$")
plt.grid(True)
plt.show()

t=np.linspace(-1*np.pi,np.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 zero
y=np.fft.fftshift(y)
Y=np.fft.fftshift(np.fft.fft(y))/64.0
w=np.linspace(-np.pi*fmax,np.pi*fmax,65)
w=w[:-1]
plt.figure()
plt.semilogx(np.abs(w), 20*np.log10(np.abs(Y)),lw=2)
plt.xlim([1,10])
plt.ylim([-20,0])
plt.xticks([1,2,5,10],["1"," ","5","10"])
plt.ylabel(r"$|Y|$ (dB)")
plt.xlabel(r"$\omega$")
plt.title(r"Spectrum of a digital ramp")
plt.grid(True)
plt.show()
```



To effectively remove the discontinuities, we multiply the original signal with the hamming window. In this way, the jump at the end of the window is attenuated.
The below code give us the time sequence for sin $\sqrt{2}t$.

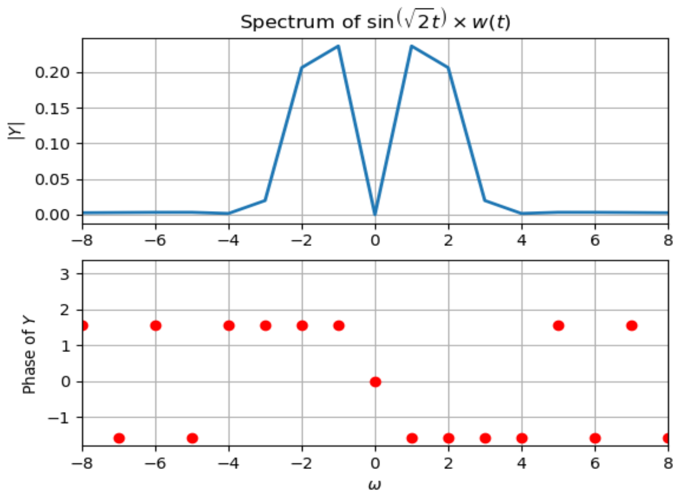
```
t1 = np.linspace(-1*np.pi,np.pi,65)
t1 = t1[:-1]
```

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



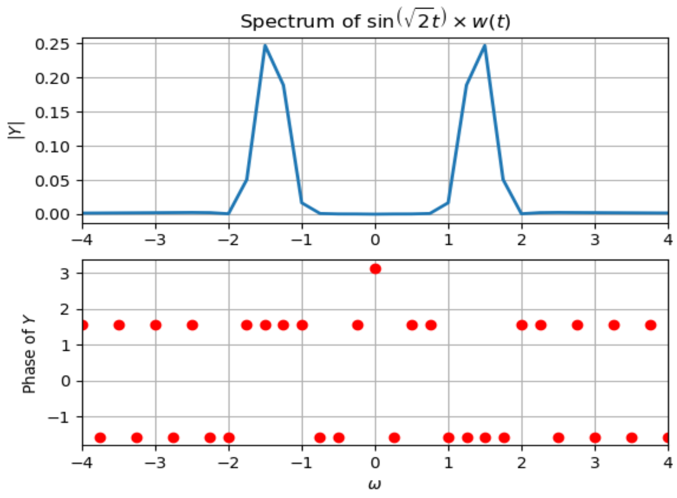
The below code gives us the DFT for this function.

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



The results are improved when we use 4 times the number of points in our computation.

```
t=np.linspace(-4*np.pi,4*np.pi,257)
t=t[:-1]
dt=t[1]-t[0] ; fmax=1/dt
n = np.arange(256)
wnd = np.fft.fftshift(0.54+0.46*np.cos(2*np.pi*n/256))
y = np.sin(np.sqrt(2)*t) * wnd
y[0] = 0 #the sample corresponding to -tmax should be set zero
y=np.fft.fftshift(y)
Y=np.fft.fftshift(np.fft.fft(y))/256.0
w=np.linspace(-np.pi*fmax,np.pi*fmax,257)
w=w[:-1]
plt.figure()
plt.subplot(2,1,1)
plt.plot(w,np.abs(Y),lw=2)
plt.xlim([-4,4])
plt.ylabel(r"$|Y|$")
plt.title(r"Spectrum of $\sin\left(\sqrt{2}t\right)\times w(t)$")
plt.grid(True)
plt.subplot(2,1,2)
plt.plot(w, (np.angle(Y)),'ro',lw=2)
plt.xlim([-4,4])
plt.ylabel(r"Phase of $Y$")
plt.xlabel(r"$\omega$")
plt.grid(True)
plt.show()
```

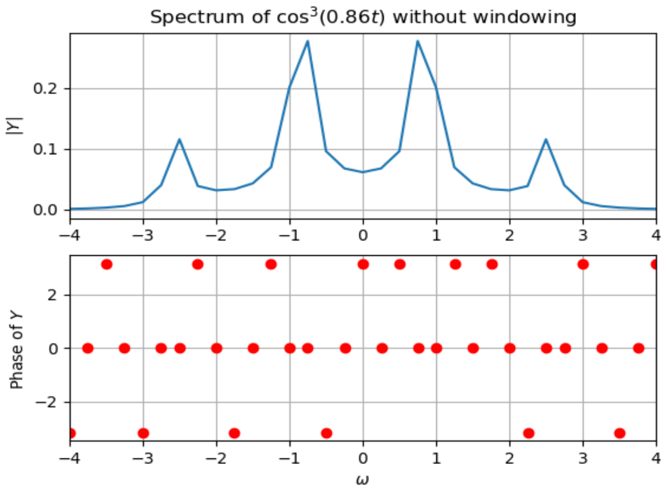


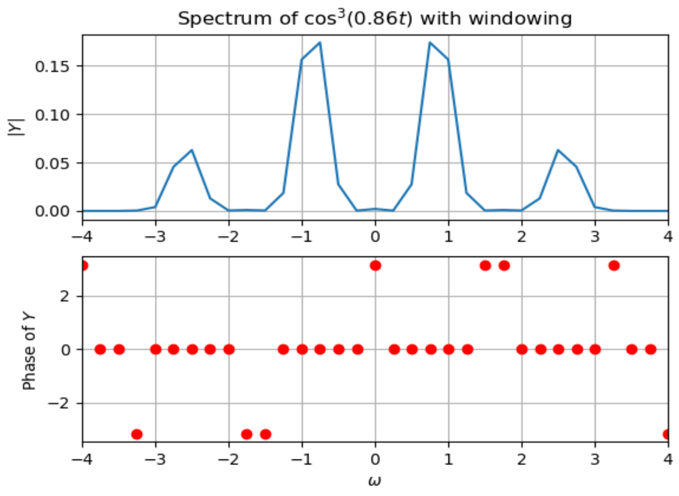
3.2 Part 2

In this portion, we obtain the spectrum for $(\cos(w_0 t))^3$ at $w_0 = 0.86$.
The code looks like this.

```
def fn_2(windowing):
  t=np.linspace(-4*np.pi,4*np.pi,257)
  t=t[:-1]
  dt=t[1]-t[0] ; fmax=1/dt
  n = np.arange(256)
  w0 = 0.86
  y = np.power(np.cos(w0*t),3)
  str='without windowing'
  if windowing:
    window = np.fft.fftshift(0.54+0.46*np.cos(2*np.pi*n/256))
    str='with windowing'
    y=y*window
  y[0] = 0 #the sample corresponding to -tmax should be set zero
  y=np.fft.fftshift(y)
  Y=np.fft.fftshift(np.fft.fft(y))/256.0 #normalizing
  w=np.linspace(-np.pi*fmax,np.pi*fmax,257)
  w=w[:-1]
  plt.figure()
  plt.subplot(2,1,1)
  plt.plot(w,np.abs(Y))
  plt.xlim([-4,4])
  plt.ylabel(r"$|Y|$")
  plt.title(r"Spectrum of $\cos^3\left(0.86t\right)$ {windowing}".format(windowing=str))
  plt.grid(True)
  plt.subplot(2,1,2)
  plt.plot(w, (np.angle(Y)),'ro')
  plt.xlim([-4,4])
  plt.ylabel(r"Phase of $Y$")
  plt.xlabel(r"$\omega$")
  plt.grid(True)
  plt.show()

fn_2(False)
fn_2(True)
```

The spectrums looks like this.





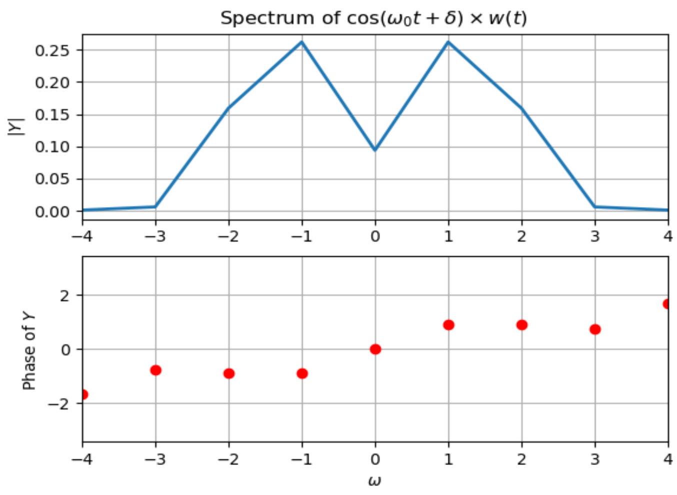
3.2.1 Part 3 and 4

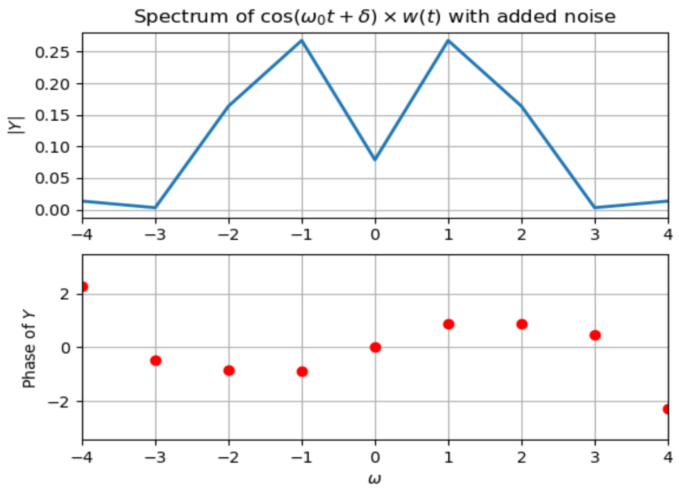
In part 3, we are given a 128- element vector, which contains $\cos(w_0 t + \delta)$ for arbitrary δ, where $-0.5 < w_0 < 0.5$ and $-\pi < t < \pi$. We also have to estimate w_0 and δ using the two peaks at $\pm w_0$. This question is extended in part 4 by addition of Gaussian white noise to the original signal.
The code looks like this:

```
def fn_3_4(number):
  t=np.linspace(-1*np.pi,np.pi,129)
  t=t[:-1]
  dt=t[1]-t[0] ; fmax=1/dt
  n = np.arange(128)
  window = np.fft.fftshift(0.54+0.46*np.cos(2*np.pi*n/128))
  w0=1.2
  delta=0.9
  y = np.cos(w0*t + delta) * window
  str1=''
  if (number==4):
    y = y + 0.1*np.random.randn(128)
    str1='with added noise'
  y[0] = 0 #the sample corresponding to -tmax should be set zero
  y=np.fft.fftshift(y)
  Y=np.fft.fftshift(np.fft.fft(y))/128.0
  w=np.linspace(-np.pi*fmax,np.pi*fmax,129)
  w=w[:-1]
  plt.figure()
  plt.subplot(2,1,1)
  plt.plot(w,np.abs(Y),lw=2)
  l=int(len(Y)/2)
  y1=np.abs(Y[l:])
```

```
    freq = (1*y1[1]+2*y1[2])/(y1[1]+y1[2])
    print("Frequency = ", freq)
    print("Phase difference = ", np.abs(np.angle(Y[np.argmax(np.abs(Y))])))
    plt.xlim([-4,4])
    plt.ylabel(r"$|Y|$")
    plt.title(r"Spectrum of $\cos\left(\omega_0t + \delta\right)\times w(t)$ {add}".format(add=str1))
    plt.grid(True)
    plt.subplot(2,1,2)
    plt.plot(w, (np.angle(Y)),'ro',lw=2)
    plt.xlim([-4,4])
    plt.ylabel(r"Phase of $Y$")
    plt.xlabel(r"$\omega$")
    plt.grid(True)
    plt.show()

print('Solving question 3 now.')
fn_3_4(3)
print('Solving question 4 now.')
fn_3_4(4)
```

The plots look like these.



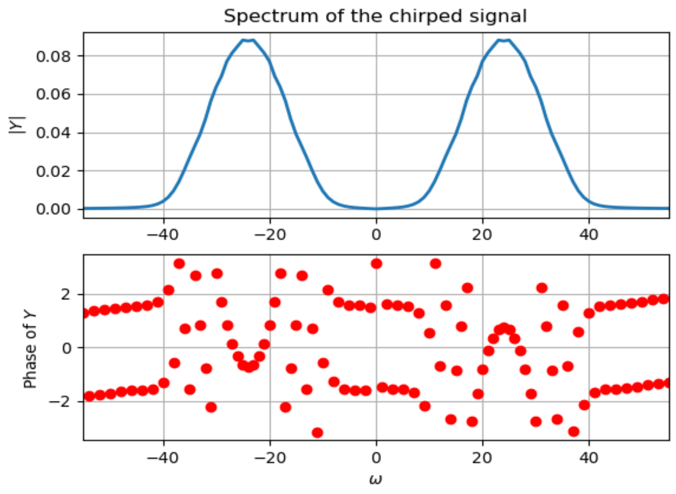


3.3 Part 5

We are given the following 'chirped' signal i.e. $\cos\big(16(1.5 + \frac{t}{2\pi})t\big)$. t goes from $-\pi$ to π in 1024 steps. We are to plot the DFT of this function.
This is the code.

```
t=np.linspace(-1*np.pi,np.pi,1025)
t=t[:-1]
dt=t[1]-t[0]
fmax=1/dt
n = np.arange(1024)
wnd = np.fft.fftshift(0.54+0.46*np.cos(2*np.pi*n/1024))
y = np.cos(16*np.multiply((1.5 + (t/(2*np.pi))),t)) * wnd
y[0] = 0 #the sample corresponding to -tmax should be set zero
y=np.fft.fftshift(y)
Y=np.fft.fftshift(np.fft.fft(y))/1024.0
w=np.linspace(-np.pi*fmax,np.pi*fmax,1025)
w=w[:-1]
plt.figure()
plt.subplot(2,1,1)
plt.plot(w,np.abs(Y),lw=2)
plt.xlim([-55,55])
plt.ylabel(r"$|Y|$")
plt.title(r"Spectrum of the chirped signal")
plt.grid(True)
plt.subplot(2,1,2)
plt.plot(w, (np.angle(Y)),'ro',lw=2)
plt.xlim([-55,55])
plt.ylabel(r"Phase of $Y$")
plt.xlabel(r"$\omega$")
plt.grid(True)
plt.show()
```

This is what the plot looks like.



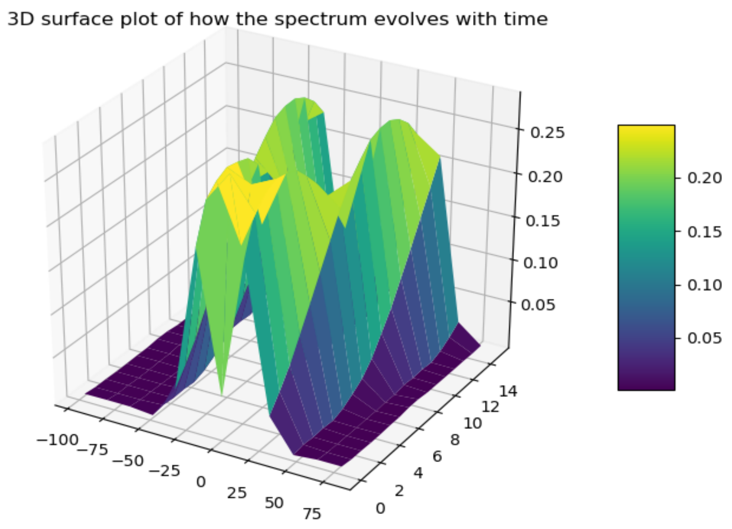
3.4 Part 6

This is an extension for part 5. For the 'chirped' signal, we break the entire vector into samples of length 64. For each of these, we extract the DFT. This is plotted as a surface plot to show the frequency versus time variation.

```
x = np.split(t,16)
dt=t[1]-t[0]
fmax=1/dt
w=np.linspace(-np.pi*fmax,np.pi*fmax,65)
w=w[:-1]
n = np.arange(64)
wnd = np.fft.fftshift(0.54+0.46*np.cos(2*np.pi*n/64))
A=np.full((64,0),0.0)
for i in range(16):
        t=x[i]
        y = np.cos(16*np.multiply((1.5 + (t/(2*np.pi))),t)) * wnd
        y[0] = 0
        y=np.fft.fftshift(y)
        Y=np.fft.fftshift(np.fft.fft(y))/64.0
        A = np.concatenate((A,Y[:,None]),axis=1)
A=np.abs(A)
time = np.arange(16)
# x-axis is w, y-axis is time.
w=w[26:-26]
F,T = np.meshgrid(w,time)
A=A[26:-26]
fig1 = plt.figure()
```

```
ax=p3.Axes3D(fig1)
plt.title('3D surface plot of how the spectrum evolves with time')
surf = ax.plot_surface(F, T, A.T, rstride=1, cstride=1, cmap='viridis')
fig1.colorbar(surf, shrink=0.5, aspect=5)
plt.show()
```

This is what the plot looks like.



4 Conclusion

We successfully analysed the Fourier transform of non-periodic functions, understood Gibbs phenomenon and a viable solution for it through the Hamming window, and performed sliding DFT on a chirped signal. The results of all these were analysed.