Assignment 9: Spectra of non-periodic signals

Siddharth D P
EE18B072

May 4, 2020

1 Abstract

The goal of this assignment is the following.

- To obtain the DFT of various non-periodic signals.
- To see how the use of windowing functions, primarily the Hamming Window, can help in smoothening out the DFT.
- To plot graphs for a better understanding.

2 Assignment

2.1 Importing all the standard libraries

```
from pylab import *
```

2.2 Calculating FFT of a non-periodic sinusoid. Part 1, Example 1

We try to take the FFT of a sinusoid which does not cover n full cycles in the time period.

```
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 zero
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)
```

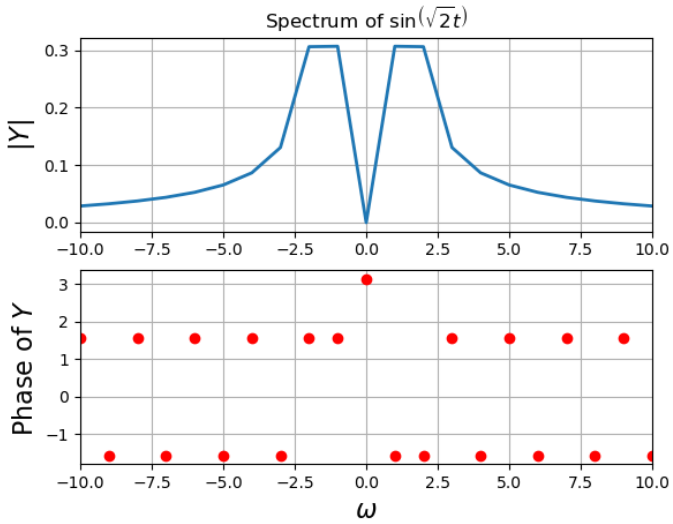

Figure 1

We wanted two spikes at about $\sqrt{2}$, but what we got were two peaks each with two values and a gradually decaying magnitude. The phase is correct. We set y[N/2]=0 so that the DFT is purely imaginary.

2.3 Going through the examples, Examples 2 and 3

To understand the above result, we try and plot the initial function from -3π to 3π.

```
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"$\sin\left(\sqrt{2}t\right)$")
```

The blue line connects the points whose DFT we took. Even though $\sin(\sqrt{2}t)$ is a periodic function, the portion between $-\pi$ and π is not the part that can be replicated to generate the function.

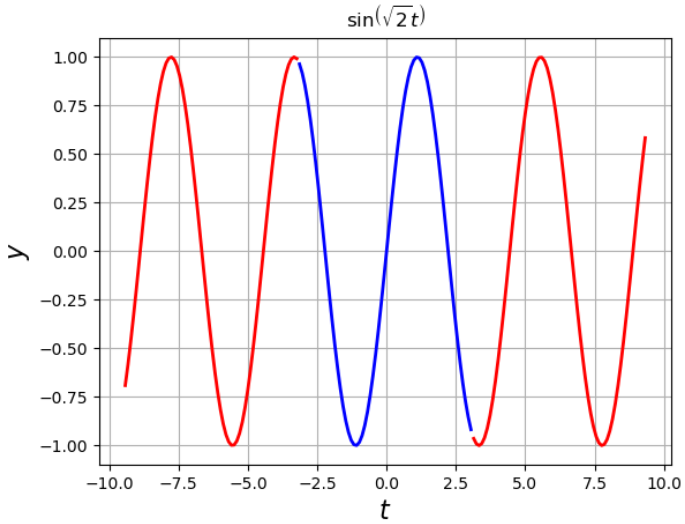

Figure 2

Replicating just the blue points, we get the following graph. Clearly, this function is not $\sin(\sqrt{2}t)$ and that is why the DFT is not what we expected.

```
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 $2\pi$ ")
```

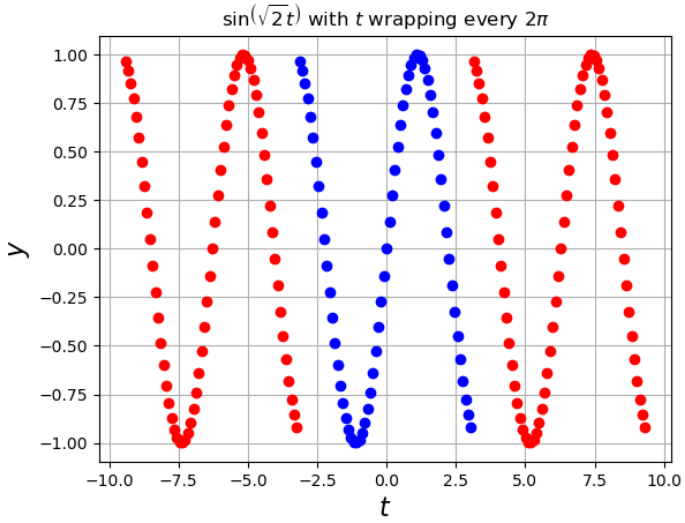

Figure 3

2.4 Gibbs Phenomenon and Windowing

The Gibbs phenomenon is the reason for slow decay in the Fourier coefficients. We can verify the same for a digital ramp. Since the spikes occur at the end of the periodic interval, we damp the function near these values by multiplying the function by the raised cosine window, also called the Hamming Window, which boosts values near the centre and suppresses those at the edges.

```
#<eg 4 7>
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)
```

```
#<eg 5 8>
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 $2\pi$ ")
```

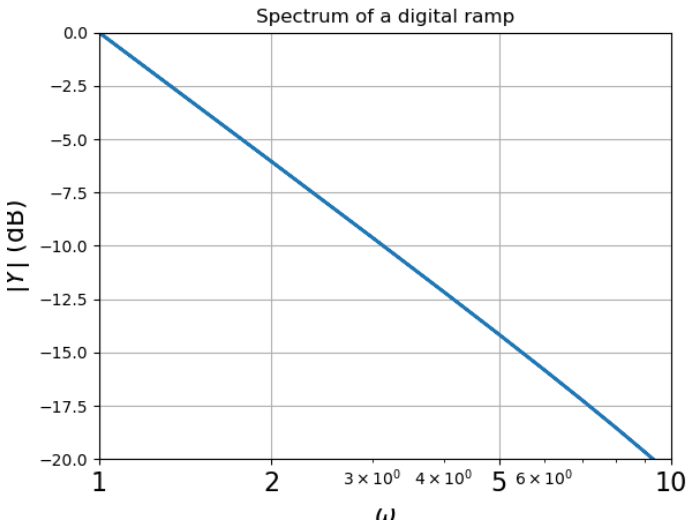

Figure 4

Clearly, the spectrum of the digital ramp decays very slowly due to the presence of discontinuities in the time domain expression. After multiplying with the Hamming Window, the jump is very much reduced. A slight jump is present,which gives us an extra 10dB of suppression.

2.5 Calculating FFT of sinusoid, Part 2

We now take the windowed cosine and find the FFT with the initial number and four times the number of points for getting a more accurate FFT.

```
t=linspace(-pi,pi,65);t=t[:-1]
dt=t[1]-t[0];fmax=1/dt
```

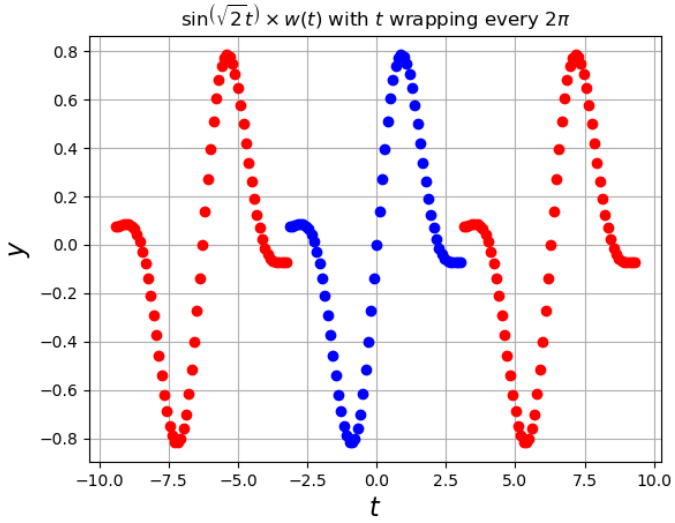

Figure 5

```
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)

#<eg7 12>
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 zero
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)
```

The magnitude plot is much better. The peak is still 2 samples wide because $\sqrt{2}$ lies between 1 and 2, which are the two Fourier components available. At four times the above frequency, the detail is much better. We still see two peaks, but this is because of multiplying the FFT with the Hamming window causing the broadening of the peak.

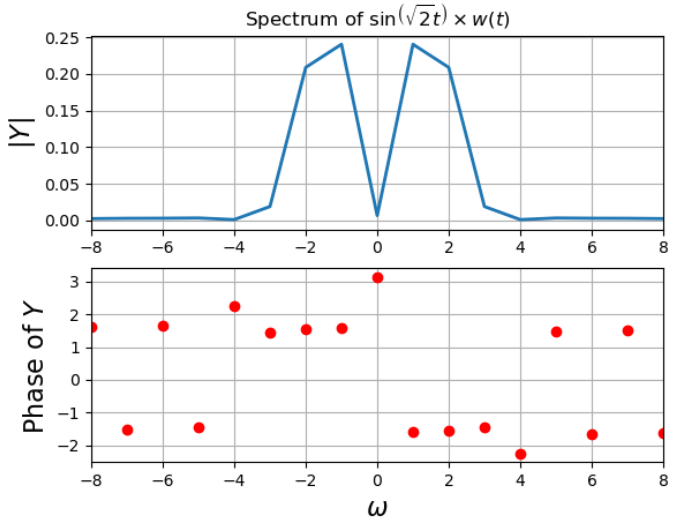

Figure 6

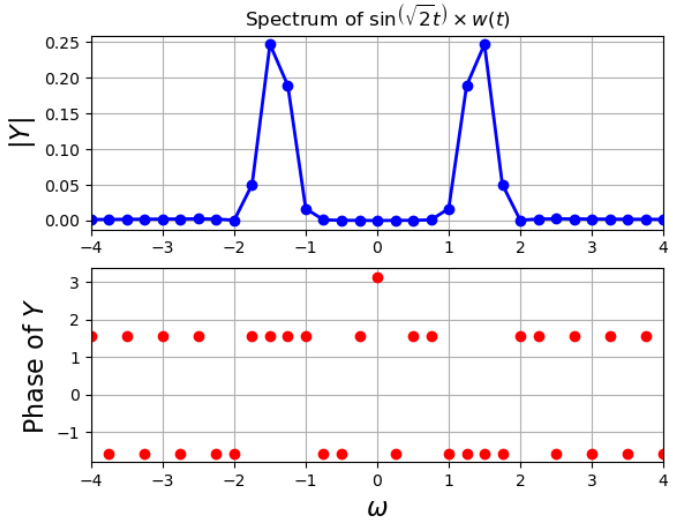

Figure 7

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

We find the spectrum of $cos^3(\omega_0 t)$ for $\omega_0 = 0.86$ with and without a Hamming window.

```
t=linspace(-4*pi,4*pi,257);t=t[:-1]
dt=t[1]-t[0];fmax=1/dt
wo=0.86
y=(cos(wo*t))*(cos(wo*t))*(cos(wo*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))/256.0
w=linspace(-pi*fmax,pi*fmax,257);w=w[:-1]
figure(8)
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 $\cos^{3}(\omega_{o} t)$ without window")
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)
```

```
# co^3(wot). With window
n=arange(256)
wnd=fftshift(0.54+0.46*cos(2*pi*n/256))
y=(cos(wo*t))*(cos(wo*t))*(cos(wo*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(9)
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 $\cos^{3}(\omega_{o} t)$ with window")
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)
```

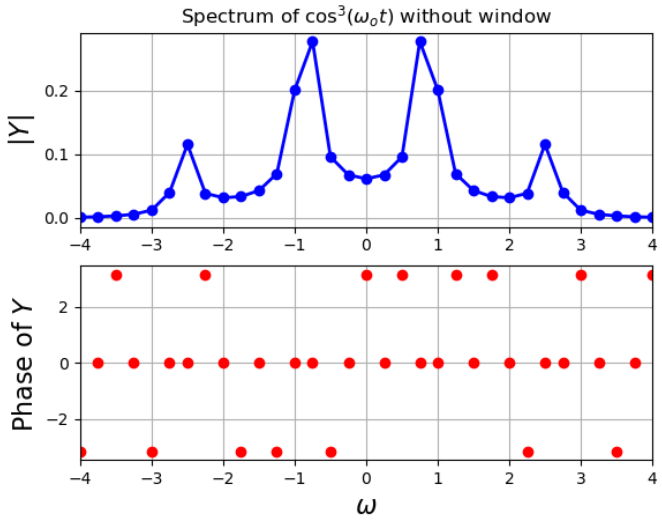

Figure 8

We can see that applying a Hamming Window definitely helps determining

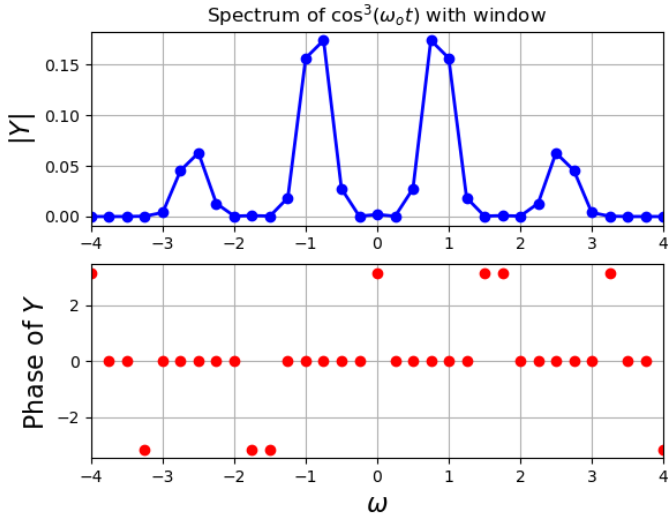

Figure 9

the peaks more accurately and cuts off the spectrum far enough away from the peaks.

2.7 Estimating phase and frequency using FFT

We need to estimate ω and δ for a signal $cos(\omega t+\delta)$ for 128 samples between $[\pi, \pi)$. We estimate omega using a weighted average. We have to extract the digital spectrum of the signal and find the two peaks at $\pm\omega_0$, and estimate ω and δ.

```
t=linspace(-pi,pi,129);t=t[:-1]
dt=t[1]-t[0];fmax=1/dt
n=arange(128)
wnd=fftshift(0.54+0.46*cos(2*pi*n/128))
wo=1.5
d=0.5
y=cos(wo*t+d)
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))/128.0
w=linspace(-pi*fmax,pi*fmax,129);w=w[:-1]
figure(10)
subplot(2,1,1)
plot(w,abs(Y),'b',w,abs(Y),'bo',lw=2)
xlim([-10,10])
```

```
ylabel(r"$|Y|$",size=16)
title(r"Spectrum of $cos(\omega_o t+\delta)$ with $\omega_o$={}, $\delta$={}"
.format(wo,d))
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)

def est_omega(w,Y):
    ii = where(w>0)
    omega = (sum(abs(Y[ii])**2*w[ii])/sum(abs(Y[ii])**2))#weighted average
    print ("Omega = ",omega)

def est_delta(w,Y,sup = 1e-4,window = 1):
    ii_1=np.where(np.logical_and(np.abs(Y)>sup, w>0))[0]
    np.sort(ii_1)
    points=ii_1[1:window+1]
    print ("Delta = ", np.sum(np.angle(Y[points]))/len(points))#weighted average
    for first 2 points

est_omega(w,Y)
est_delta(w,Y)

#Question 4. With added white Gaussian noise
y=y*wnd+0.1*randn(128)
est_omega(w,Y)
est_delta(w,Y)
show()
```

We estimate omega by performing a mean average of over the magnitude of $(j\omega)$ |. For δ, we consider a widow on each half of ω (split into positive and negative values) and extract their mean slope. The intuition behind this is that a circular shift in the time domain of a sequence results in the linear phase of the spectra. This is what I got for the two parts-
Omega = 1.516317
Delta = 0.506776 (without noise)
Omega = 2.055074
Delta = 0.489037 (with Gaussian noise added)

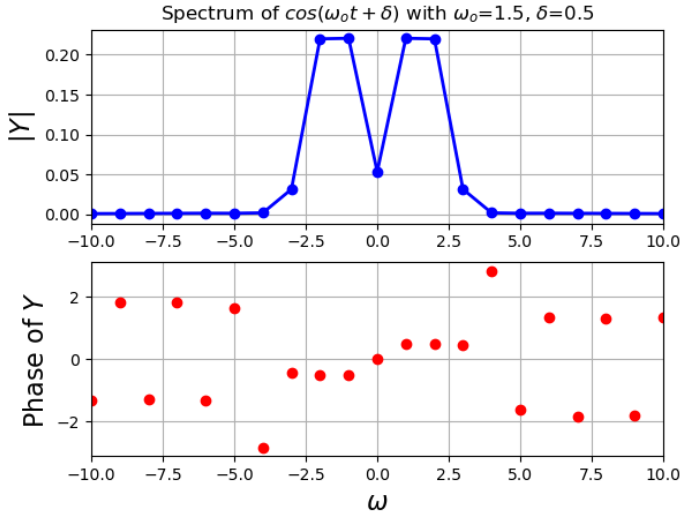

Figure 10

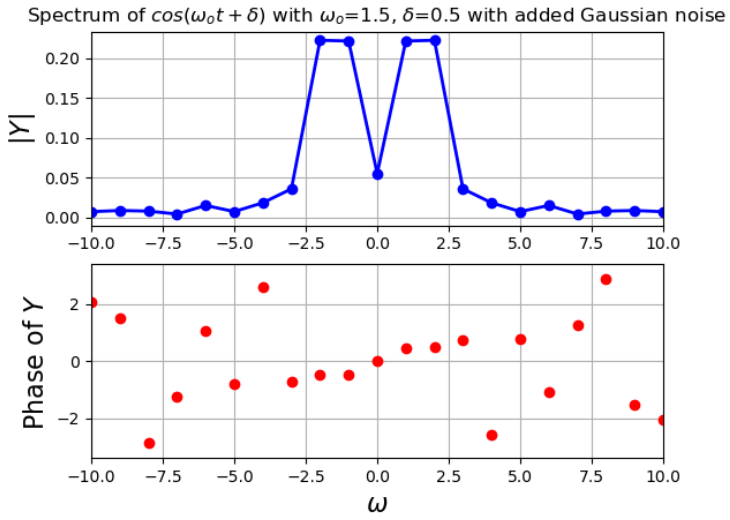

Figure 11

2.8 FFT for Chirped signal

We analyze the chirp signal $cos(16(1.5 + t/2)t)$ which is an FM signal where frequency is directly proportional to time. The FFT of the chirp is given by: We note that the frequency response is spread between 5-50 rad/s. A large section of this range appears due to the Gibbs phenomenon. On windowing, only frequencies between 16 and 32 rad/s remain.

```
t=linspace(-pi,pi,1025);t=t[:-1]
dt=t[1]-t[0];fmax=1/dt
y1=cos(16*(1.5+t/(2*pi))*t)
n=arange(1024)
wnd=fftshift(0.54+0.46*cos(2*pi*n/1024))
y=y1*wnd
y1[0]=0; y[0]=0
y1=fftshift(y1); y=fftshift(y)
Y1=fftshift(fft(y1))/1024.0; Y=fftshift(fft(y))/1024.0
w=linspace(-pi*fmax,pi*fmax,1025);w=w[:-1]
ph=angle(Y); ph1=angle(Y1)
mag = abs(Y); mag1 = abs(Y1)
ph[where(mag<3e-3)] = 0; ph1[where(mag1<3e-3)] = 0
figure(12)
subplot(2,1,1)
plot(w,abs(Y1),'b',lw=2)
xlim([-75,75])
ylabel(r"$|Y|$",size=16)
title(r"Spectrum of chirp function without window")
grid(True)
subplot(2,1,2)
plot(w,angle(Y1),'ro',lw=2)
xlim([-75,75])
ylabel(r"Phase of $Y$",size=16)
xlabel(r"$\omega$",size=16)
grid(True)
figure(13)
subplot(2,1,1)
plot(w,abs(Y),'b',lw=2)
xlim([-75,75])
ylabel(r"$|Y|$",size=16)
title(r"Spectrum of chirp function with window")
grid(True)
subplot(2,1,2)
plot(w,ph,'ro',lw=2)
xlim([-75,75])
ylabel(r"Phase of $Y$",size=16)
xlabel(r"$\omega$",size=16)
grid(True)
```

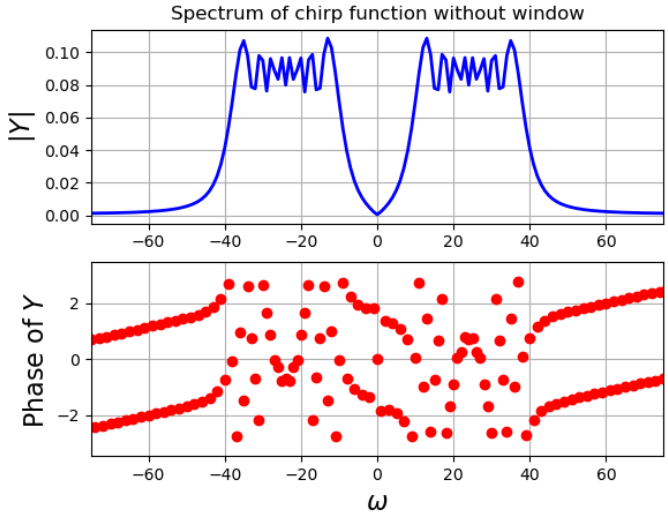

Figure 12

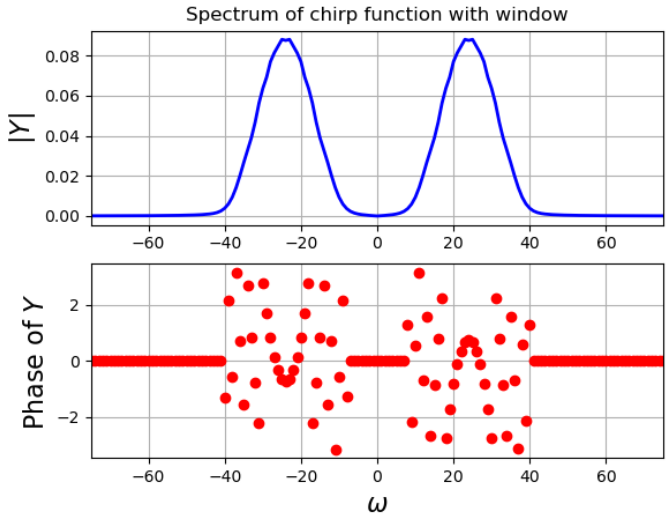

Figure 13

2.9 Time Frequency Plot of FFT for chirped Signal

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

```
t=linspace(-pi,pi,1025);t=t[:-1]
t_arrays=split(t,16)
```

```
Y_mags=zeros((16,64))
Y_angles=zeros((16,64))
#splitting array and doing fft
for i in range(len(t_arrays)):
    t = t_arrays[i]
    dt=t[1]-t[0];fmax=1/dt
    y=cos(16*(1.5+t/(2*pi))*t)
    y[0]=0
    y=fftshift(y)
    Y=fftshift(fft(y))/64.0
    w=linspace(-pi*fmax,pi*fmax,65);w=w[:-1]
    Y_mags[i] =  abs(Y)
    Y_angles[i] =  angle(Y)
#plotting
fig = figure(14)
ax = fig.add_subplot(111, projection='3d')

t=np.linspace(-pi,pi,1025);t=t[:-1]
fmax = 1/(t[1]-t[0])
t=t[::64]
w=linspace(-fmax*pi,fmax*pi,65);w=w[:-1]
t,w=np.meshgrid(t,w)

surf=ax.plot_surface(w,t,Y_mags.T,cmap=cm.coolwarm,linewidth=0, antialiased=False)
fig.colorbar(surf, shrink=0.5, aspect=5)
ylabel("Time")
xlabel("Frequency")
title("Surface time-frequency plot of the Chirp function")
```

This is a piece-wise windowed Chirped Signal. We noted the case of sparse number of slices and hence took more closely spaced slices. The general properties of the Fourier spectra for a chirped signal are observable in this time-frequency plot - existence of peaks (slow growth) and vanishing of chirp effects in case of a windowed transform.

3 Conclusion

- We obtained the DFT of non-periodic functions.
- We saw how the use of Hamming window improved the results of the DFT.
- The last question addresses the time varying spectra for a chirped signal, where we plot Fourier spectra for different time slices of a signal.

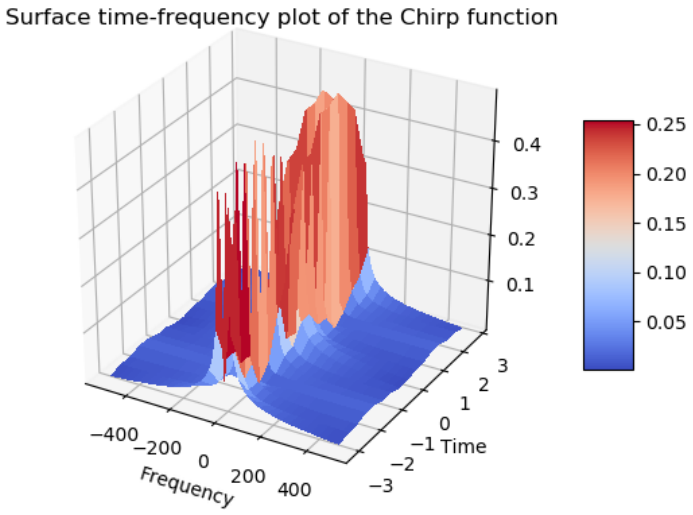
Surface time-frequency plot of the Chirp function
0.4
0.3
0.2
0.1
−400
−200
0
200
400
Frequency
3
2
1
0
−1
−2
−3
Time
0.25
0.20
0.15
0.10
0.05

Figure 14