EE2703 Applied Programming Lab - Assignment No 9

Name: Atishay Ganesh
Roll Number: EE17B155

April 13, 2019

1 Abstract

The goal of this assignment is the following.

- Obtaining the DFT of non-periodic functions.
- To see how the use of windowing functions (e.g Hamming Window) can help in making the DFT better.
- To plot graphs to understand this.

2 Assignment

2.1 Setting up the variables

Importing the standard libraries

```
from pylab import *
import mpl_toolkits.mplot3d.axes3d as p3
```

We define a DFT Function that will plot the DFT using the FFT algorithm.

```
def dft(func,tim= None,n_p=512,fig_no=0,name= None):
    '''Function to plot the dft using fft algorithm
func: lambda/function whose fourier transform is required
time: time interval in the form (start_time,end_time)
default is (-4*pi,4*pi)
n_p: number of samples in time domain
fig_no: Figure number
name: Name of function for the graph
'''
    if tim is None:
        t=linspace(-4*pi,4*pi,n_p,endpoint= False)
```

```
    else:
        start,end = tim
        t = linspace(start,end,n_p,endpoint=False)
    y = func(t)
    y[0]=0# the sample corresponding to -tmax should be set zeroo
    Y=fftshift(fft(fftshift(y)))/n_p#fftshift so the plot is in terms we know
    w=linspace(-pi,pi,n_p,endpoint= False);
    w = w*n_p/(end-start)#the range of frequencies
    fig, (ax1, ax2) = plt.subplots(2, 1)

    Ysig = where(abs(Y)>10**-5)#only plot significant points phase
    ax1.plot(w,abs(Y),lw=1)
    #to ensure that we dont go too far on each side

    ax1.set_ylabel(r"$|Y|$",size=16)
    ax1.grid(True)
    ax2.plot(w[Ysig],angle(Y[Ysig]),'ro')

    ax2.set_ylabel(r"Phase of $Y$",size=16)
    ax2.set_xlabel(r"$\omega$",size=16)
    grid(True)
    title("Spectrum of {}".format(name))

    return ax1,ax2,Y
```

2.2 Calculating FFT of sinusoid, part 1

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

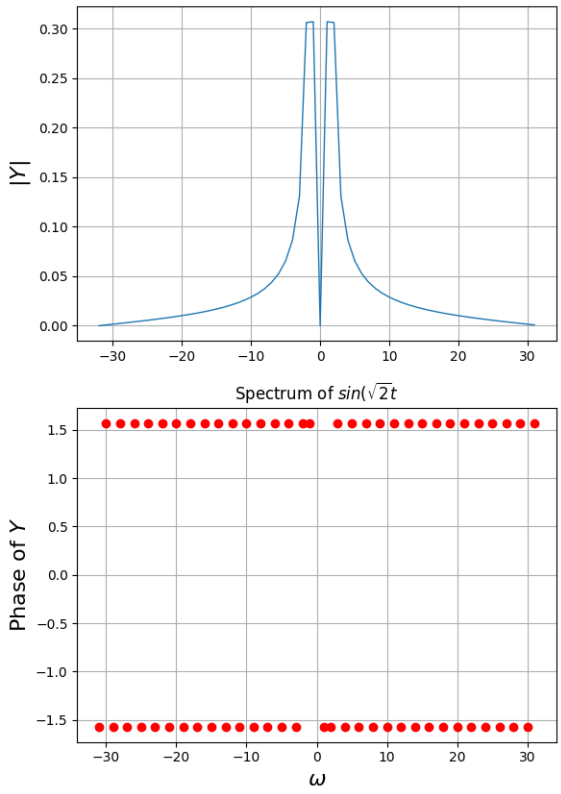
```
y1 = lambda t: sin(sqrt(2)*t)
dft(y1,(-pi,pi),64,1,r'$sin(\sqrt{2}t$')
show()
```

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[$\frac{N}{2}$]=0 so that the DFT is purely imaginary

2.3 Understanding the above results

To understand the above result we try and plot the initial function from -3π to 3π. We then plot the windowed function time repeated to the left and right.

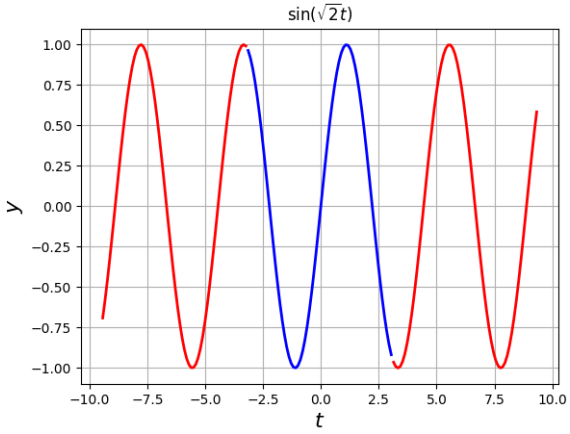
```
t1=linspace(-pi,pi,65);t1=t1[:-1]
t2=linspace(-3*pi,-pi,65);t2=t2[:-1]
```

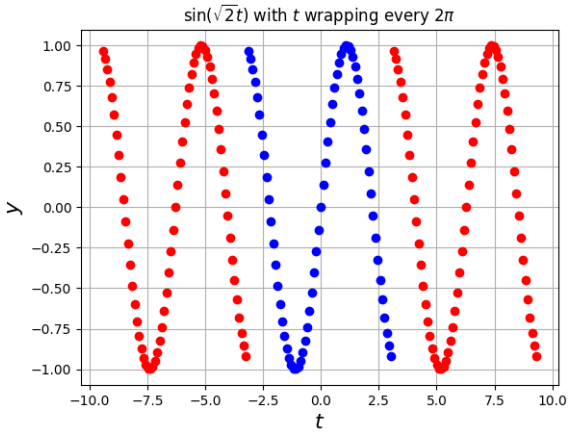



```
t3=linspace(pi,3*pi,65);t3=t3[:-1]
# y=sin(sqrt(2)*t)
figure(2)
plot(t1,sin(sqrt(2)*t1),'b',lw=2)
plot(t2,sin(sqrt(2)*t2),'r',lw=2)
plot(t3,sin(sqrt(2)*t3),'r',lw=2)
ylabel(r"$y$",size=16)
xlabel(r"$t$",size=16)
title(r"$\sin\left(\sqrt{2}t\right)$")
grid(True)
show()

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)
show()
```

Clearly we observe that the two are not the same. Furthermore there is in the discontinuity leading to Gibbs Phenomena, which is why there is a gradually decaying sinusoid.





2.4 Gibbs Phenomenon and Windowing

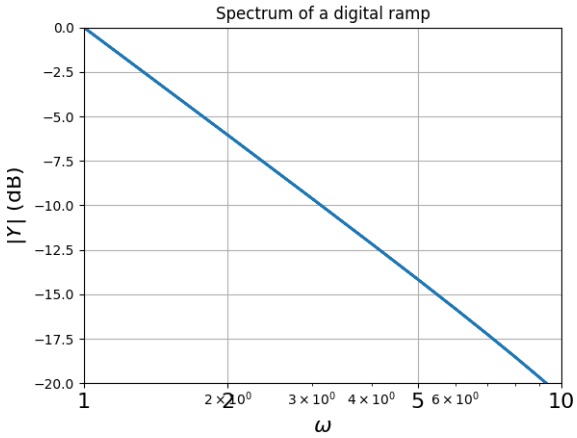
The Gibbs phenomenon is the reason for slow decay in the FFT. We can verify this for a digital ramp. Since the spikeks happen at the end of the

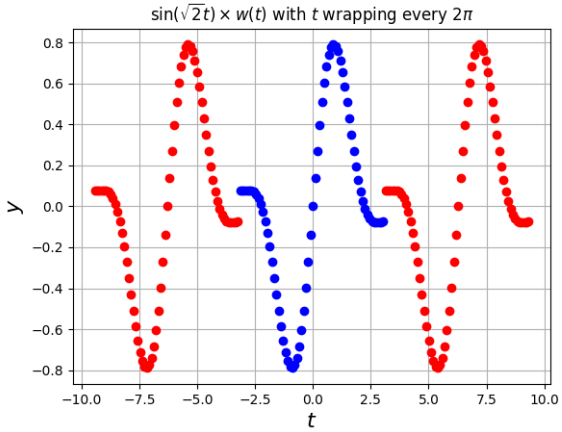
periodic interval, we damp the function near there by multiplying the function by the raised cosine (Hamming Window) window, which boosts values near the centre and suppresses those at the edges.

```
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)
show()


# In[6]:


wnd= lambda t: fftshift(0.54+0.46*cos(2*pi*t/len(t)))#hamming window
y1 = lambda t: sin(sqrt(2)*t)*wnd(t)
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)
y=sin(sqrt(2)*t1)*wnd(n)
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)
show()
```

Clearly the spectrum of the digital ramp decays very slowly due to there being discontinuities in the time domain expression. After multiplying with

the Hamming Window, the jump is very much reduced. A little bit of jump is there, which gives us slightly more suppression.





2.5 Calculating FFT of sinusoid, part 2

We take the windowed cosine and find the FFT with the initial number and four times the number of points

```
y2 = lambda t,n: y1(t)*wnd(arange(n))
y264 = lambda t: y2(t,64)
ax1,ax2,w = dft(y264,(-pi,pi),64,1,r'$sin(\sqrt{2}t*w(n)$')
ax1.set_xlim(-10,10)
ax2.set_xlim(-10,10)

show()
```

```
# In[8]:


y2256 = lambda t: y2(t,256)
ax1,ax2,w = dft(y2256,(-4*pi,4*pi),256,1,
r'$sin(\sqrt{2}t*w(n) with better sampling$')
ax1.set_xlim(-10,10)
ax2.set_xlim(-10,10)

show()
```

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 not because of an approximation, but actually it is because of multiplying with the Hamming window causing the broadening of the peak.

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

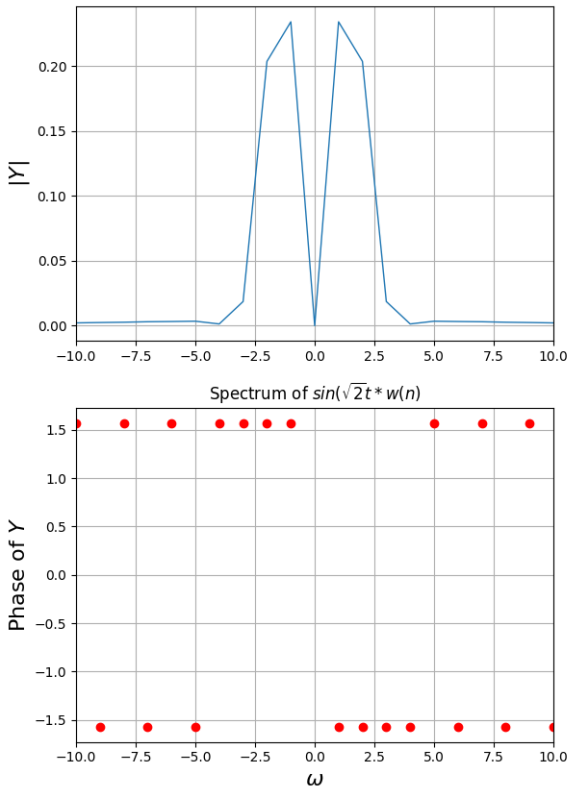
We find the spectrum of $\cos^3(\omega_0 t)$ for $\omega_0 = 0.86$ with and without a Hamming window

```
y3 = lambda t:(cos(0.86*t))**3
ax1,ax2,w = dft(y3,(-pi,pi),256,1,r'$cos^{3}(\omega_{0}t)$')
ax1.set_xlim(-10,10)
ax2.set_xlim(-10,10)
show()
y4 = lambda t,n: y3(t)*wnd(arange(n))
y464 = lambda t: y4(t,256)
ax1,ax2,w = dft(y464,(-4*pi,4*pi),256,1,r'$cos^{3}(\omega_{0}t)$')
ax1.set_xlim(-10,10)
ax2.set_xlim(-10,10)
show()
```

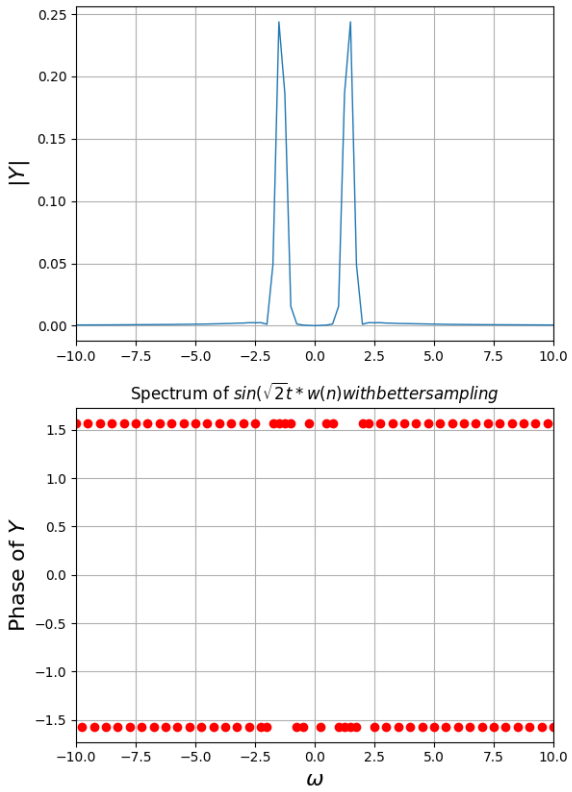
We can see that applying a Hamming Window definitely helps determining where the peaks are and cuts off the spectrum far enough away from the peaks.

2.7 Estimating phase and frequency using FFT

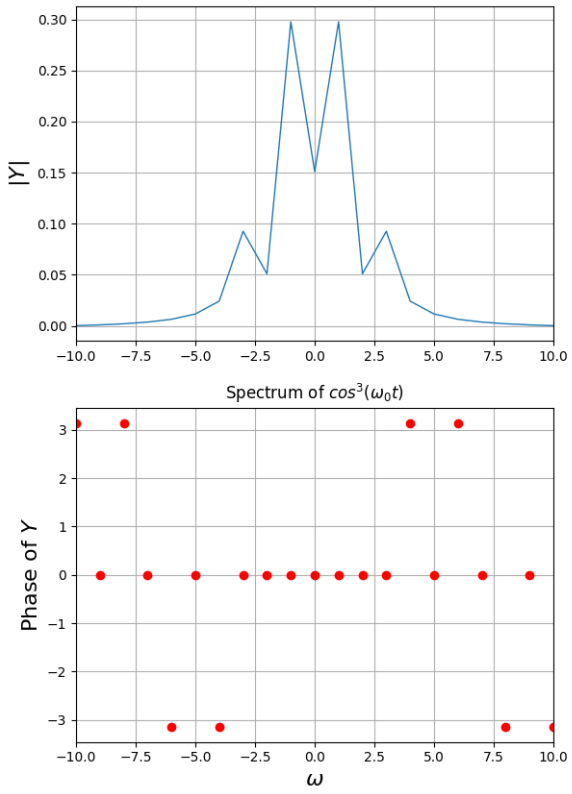
We make a program that will take a 128 element vector which is the cosine of a function with arbitrary phase and frequency in certain limits. We extract the spectrum and estimate the quantities. We do the same with and without Gaussian noise




```
def cosest(w,d):
    fn1 = lambda t:  cos(w*t+d)*wnd(arange(len(t)))
    ax1,ax2,Y = dft(fn1,(-pi,pi),128,1,r'$cos(\omega_{0}t+\delta)$')
    ax1.set_xlim(-10,10)
    ax2.set_xlim(-10,10)

    print(sum(
    abs((Y*abs(arange(len(Y))-len(Y)//2)))[64:64+10])/sum(abs(Y)[64:64+5]))
    print(angle(Y[len(Y)//2+1]))
    return(Y)

Yf = cosest(1.21,0.75)
show()

# In[14]:
```




```
def cosestrand(w,d):
    fn1 = lambda t:  cos(w*t+d)*wnd(arange(len(t))) +0.1*randn(len(t))
    ax1,ax2,Y = dft(fn1,(-pi,pi),128,1,r'$cos(\omega_{0}t+\delta) with noise$')
    ax1.set_xlim(-10,10)
    ax2.set_xlim(-10,10)

    print(sum(
    abs((Y*abs(arange(len(Y))-len(Y)//2)))[64:64+5])/sum(abs(Y)[64:64+5]))
    print((angle(Y[len(Y)//2+1])))
    return(Y)

Yf = cosestrand(1.21,0.75)
show()
plt.show()
```

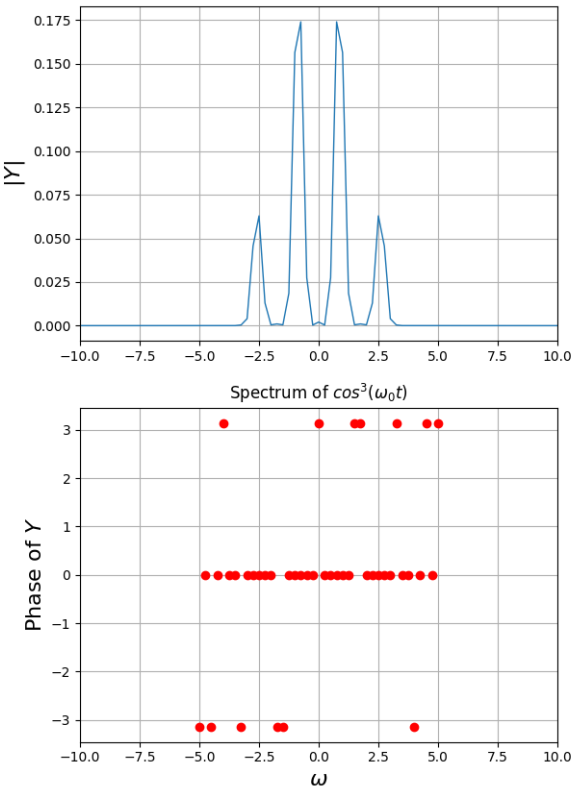


No noise
ω_0 1.2214619719196165
δ 0.7574596213758671
Noisy
ω_0 1.1292269848867516
δ 0.7592720815427652

Clearly the estimation strategy of using the weighted average of the fft points for the magnitude, and the first point for the phase, seem to give accurate enough results. Noise does not seem to affect it too much as the noise is mostly high frequency components.

2.8 FFT for Chirped signal

We plot the DFT of the chirped signal. and the windowed signal in time domain $\cos\big(16(1.5+\frac{t}{2\pi})t\big)$

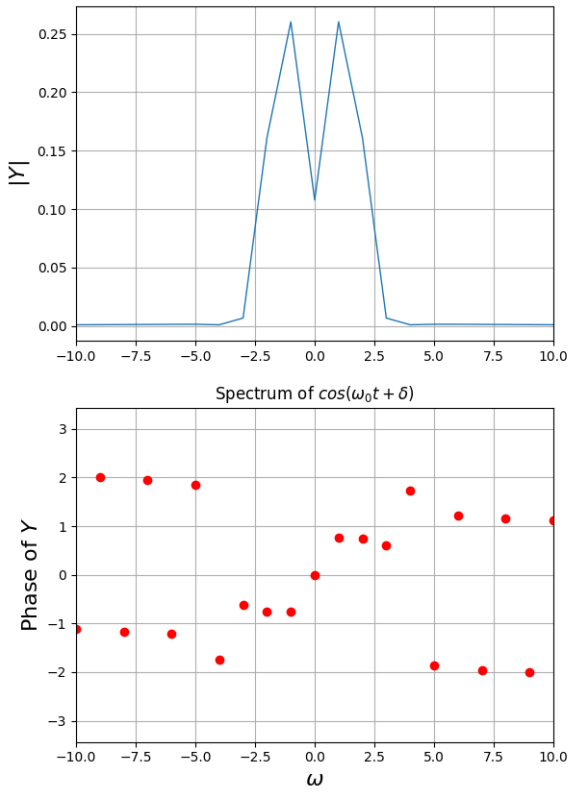



```
ychirp = lambda t: cos(16*t*(1.5+t/(2*pi)))*wnd(arange(len(t)))
ax1,ax2,Ychirp = dft(ychirp,(-pi,pi),1024,1,r'chirped signal')
ax1.set_xlim(-60,60)
ax2.set_xlim(-60,60)
figure(2)
t = linspace(-pi,pi,1024)
plot(t,ychirp(t))
show()
```

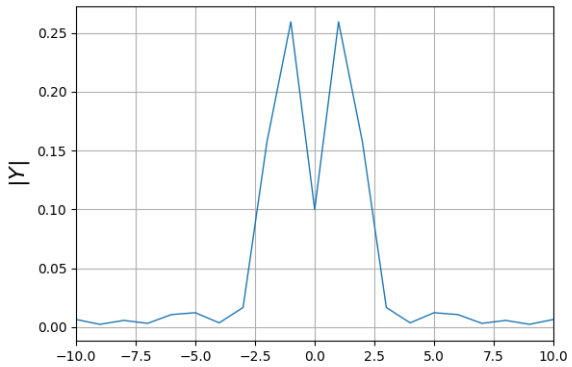
Clearly we see that the chirped signal has frequencies from 16-32

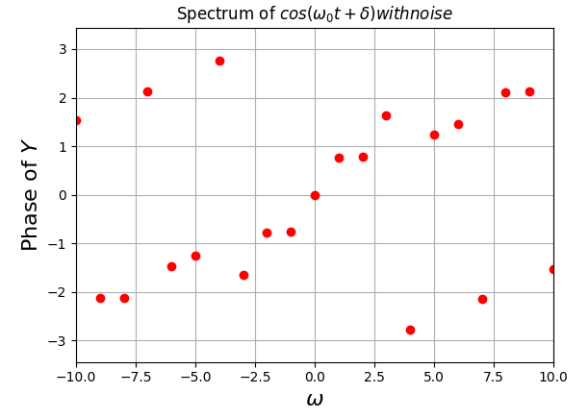
2.9 Time Frequency Plot of FFT for chirped Signal

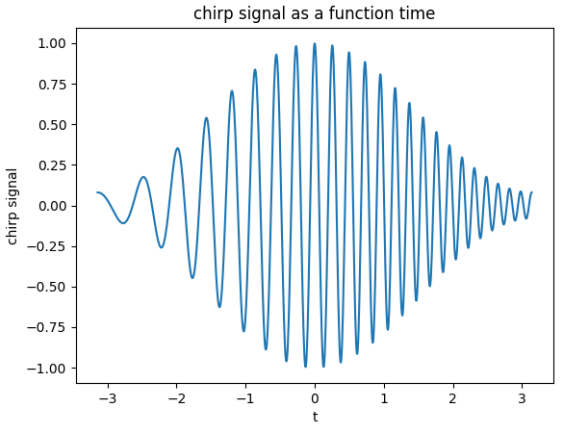
Since the chirped signal's frequency varies with time, we break it up into various time intervals, window each of those, and take the FFT for each and then plot them in a 3-d spectrogram. We also do a contour plot which is more understandable.



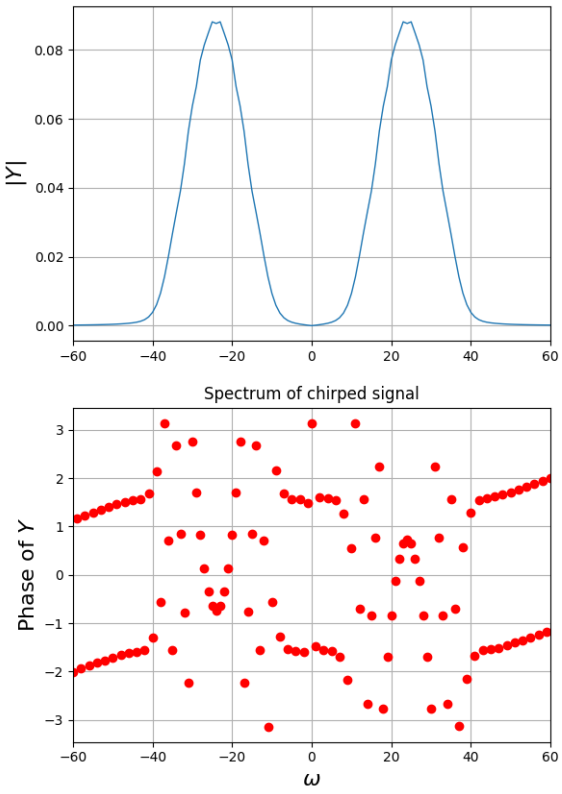

```
def dftchirp(func,n_p=512,fig_no=0,name= None):
    '''Function to plot the dft for the chirped signal using fft algorithm
func: chirped signal we are calculating the fft for
n_p: number of samples in time domain
fig_no: Figure number
name: Name of function for the graph
'''
    Y=fftshift(fft(ifftshift(func),n=64) )/n_p
    #fftshift so the plot is in looks like what we expect
    w=linspace(-pi,pi,64,endpoint= False);
    w = w*n_p*16/(2*pi)#the range of frequencies

    fig2=figure(2)
    ax=p3.Axes3D(fig2)
    title('3-D spectrogram')
    X = linspace(-pi,pi,16,endpoint=False)
```



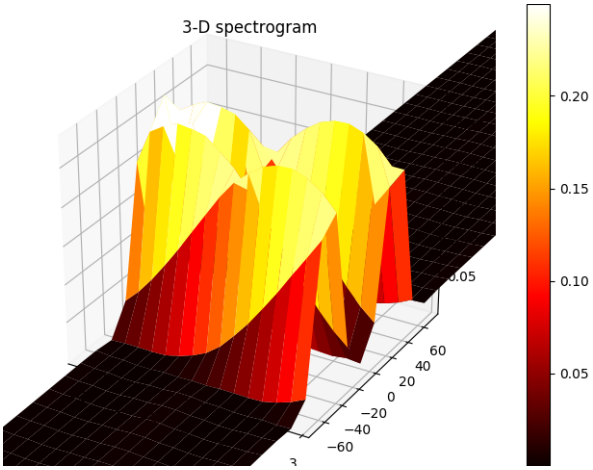


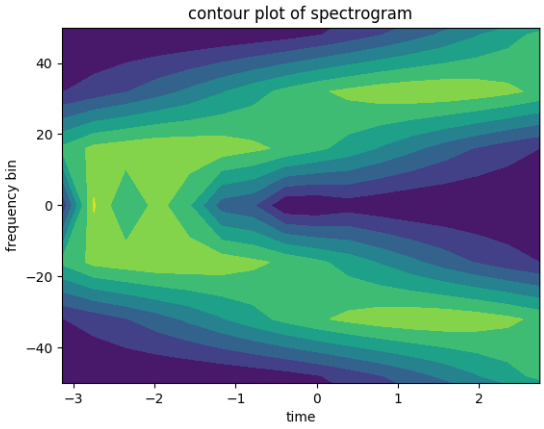



```
w,X =  meshgrid(w,X)
surf = ax.plot_surface(X, w, abs(Y), rstride=1, cstride=1, cmap=matplotlib.cm.ho
```




```
    fig2.colorbar(surf)
    ax.set_ylim(-75,75)
    show()
    figure(1)
    c = contourf(X,w,abs(Y))
    ylim(-50,50)
    title("contour plot of spectrogram")
    show()
    return Y

t = linspace(-pi,pi,1024,endpoint=False)
yorg = cos(16*t*(1.5+t/(2*pi)))*tile(wnd(arange(64)),16)
yorg[0]=0 #the sample corresponding to -tmax should be set zero
y = asarray(split(yorg,16))
Y = dftchirp(y,64,1,r'chirped signal')
show()
```





The First plot is just the piece-wise windowed Chirped Signal. We take the piece-wise FFT and plot, and we get the 3-D spectrogram of the same. The frequency increases from 16 to 32 as time increases, but it is not too clear in this. In the contour plot it is very clear how the frequency is changing from 16 to 32 as time varies from -3 to 3.

3 Conclusions

- We Obtained the DFT of non-periodic functions.
- We saw how the use of windowing functions (e.g Hamming Window) improved the results.
- We made the 3-D Spectrogram of the chirped signal.
- To plotted graphs to understand the same.