Assignment 9

Saurabh Vaishampayan:EP17B028

15 May 2020

1 Introduction

In this assignment, we continue our analysis of signals using Fourier Transforms. This time, we focus on finding transforms of functions which are discontinuous when periodically extended. An example of this is $\sin(\sqrt{2}t)$. The discontiuity causes fourier components in frequencies other than the sinusoids frequency which decay as $\frac{1}{\omega}$, due to Gibbs phenomenon. We resolve this problem using the process of windowing. In this assignment, we focus on one particular type of window - the Hamming window. We use this windowed transform to analyse signals known to contain a sinusoid of unknown frequencies and extract its phase and frequency. We then perform a sliding DFT on a chirped signal and plot a spectrogram or a time-frequency plot.

2 Examples

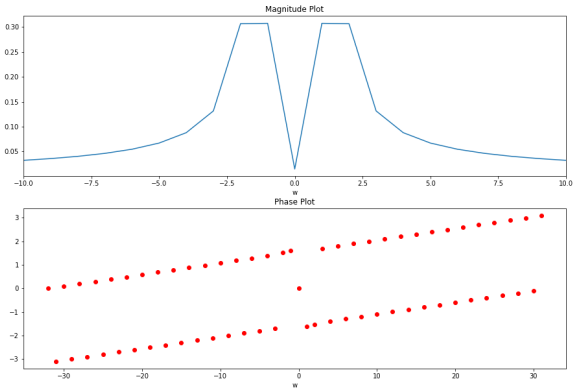
2.1 Spectrum of $\sin(\sqrt{(}2)t)$

2.2 Code

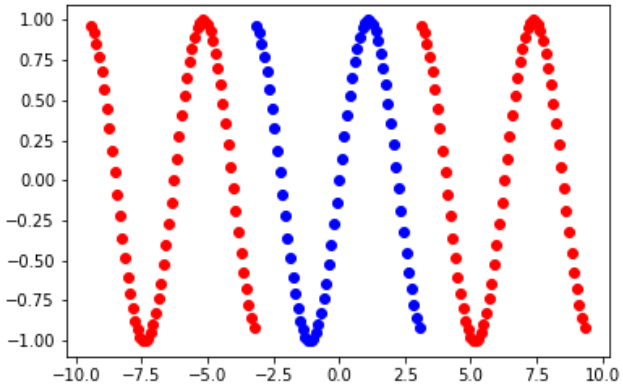
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d

#Question 1: Examples

def fftcompute(x,Tmax):
    dt = Tmax/len(x)
    w = np.linspace(-np.pi/dt,np.pi/dt,len(x)+1)[:-1]
    X = (1/len(x))*np.fft.fftshift(np.fft.fft(np.fft.ifftshift(x)))
    return w, X

def fftplot(w,X,filename,xlim):
    fig, ax = plt.subplots(2,1,figsize=(15,10))

    ax[0].plot(w,np.abs(X))
    if(xlim!=None):
        ax[0].set_xlim(xlim)
    ax[0].set_title('Magnitude Plot')
    ax[0].set_xlabel('w')

    phase = np.angle(X)
```

```
    phase[np.where(np.abs(X)<1e-2)]=0

    ax[1].plot(w,phase,'ro')
    if(xlim!=None):
        ax[0].set_xlim(xlim)
    ax[1].set_xlabel('w')
    ax[1].set_title('Phase Plot')
    fig.savefig(filename+'.png')
    plt.show()


N = 64
Tmax = 2*np.pi
t = np.linspace(-Tmax/2,Tmax/2,N+1)[:-1]
x = np.sin(np.sqrt(2)*t)

w, X = fftcompute(x,Tmax)
fftplot(w,X,'sqrt2sin',[-10,10])

t2 = np.linspace(-3*np.pi,-np.pi,N+1)[:-1]
t3 = np.linspace(np.pi,3*np.pi,N+1)[:-1]
plt.plot(t,x,'bo')
plt.plot(t2,x,'ro')
plt.plot(t3,x,'ro')
plt.savefig('1.png')
plt.show()
```

2.3 Plots



While calculating DFT, by sampling over a finite time window, we end up calculating the DFT of the following periodic signal:



The sharp discontinuity near the boundary results in a slow decay which distorts the spectrum we're looking for.
Hence we attenuate the signal near the boundaries by multiplying by a window function.

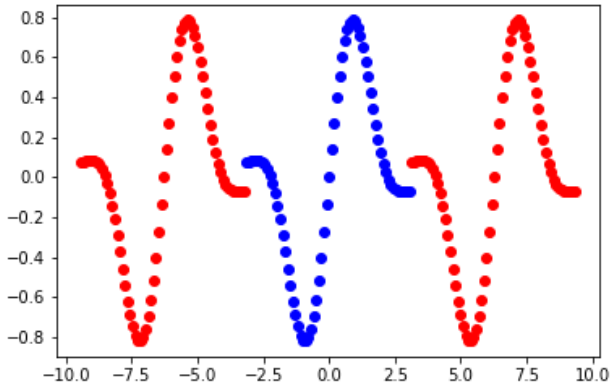
2.4 Hamming window

In this assignment we use the Hamming window of size N:

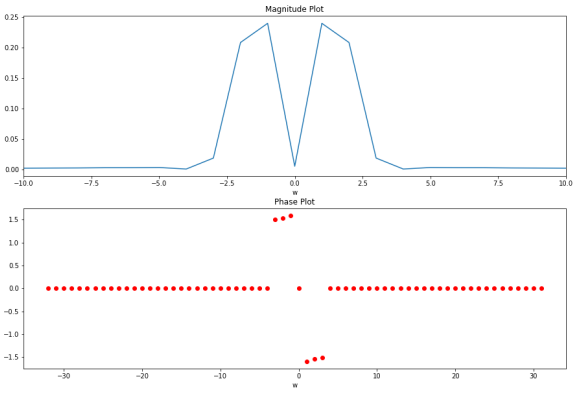
$$\mathrm{x[n]} = 0.54 + 0.46\cos(\tfrac{2\pi n}{N-1})$$

```
def hamming(n):
    wnd = np.fft.fftshift(0.54+0.46*np.cos(2*np.pi*np.arange(n)/(n-1)))
    return wnd
```

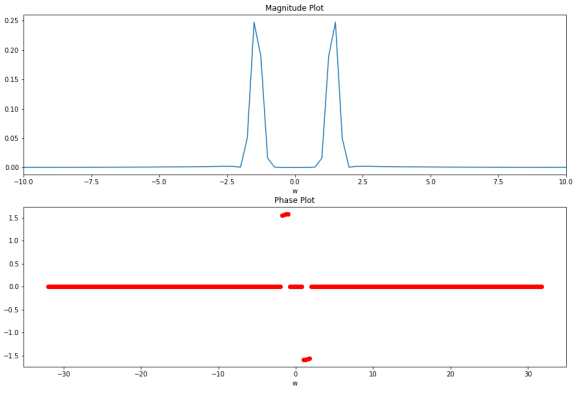
We now multiply our signal with the Hamming window and then periodically extend it:



The spectrum is found below using a window size of 2π:



And with a window size of 8π:



2.5 Code

```
y = x*hamming(N)
w, Y = fftcompute(y, Tmax)
fftplot(w,Y,'hammingsqrt2_1',[-10,10])
plt.plot(t,y,'bo')
plt.plot(t2,y,'ro')
plt.plot(t3,y,'ro')
plt.savefig('2.png')
plt.show()

N = 256
Tmax = 8*np.pi
t = np.linspace(-Tmax/2,Tmax/2,N+1)[:-1]
x = np.sin(np.sqrt(2)*t)
y = x*hamming(N)

```

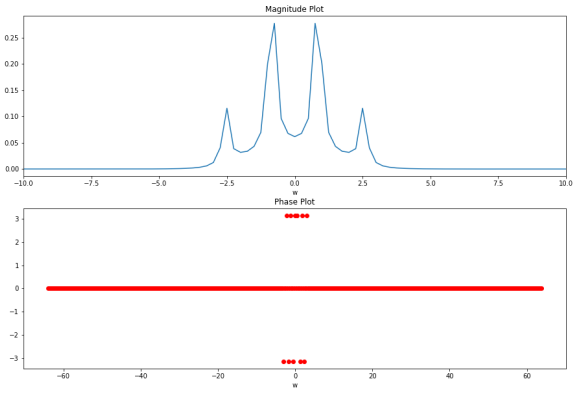
```
w, Y = fftcompute(y,Tmax)
fftplot(w,Y,'hammingsqrt2_2',[-10,10])
```

3 FFT of $\cos^3(0.86t)$

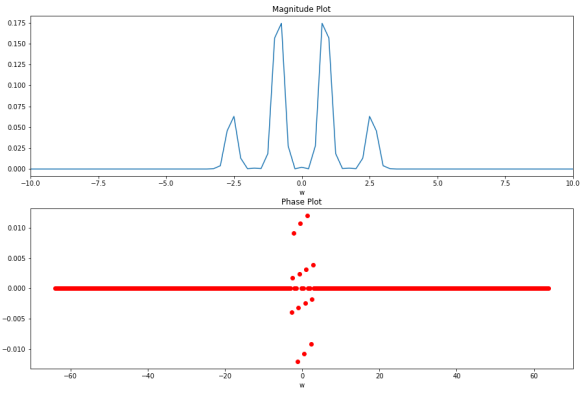
```
N = 512
Tmax = 8*np.pi
t = np.linspace(-Tmax/2,Tmax/2,N+1)[:-1]
x = (np.cos(0.86*t))**3
w, X = fftcompute(x,Tmax)
fftplot(w,X,'q2withouthamming',[-10,10])

y = x*hamming(N)
w, Y = fftcompute(y,Tmax)
fftplot(w,Y,'q2withhamming',[-10,10])
```

3.1 Plots

We first find FFT without Hamming window:



Now with the hamming window:



3.2 Inferences

- We observe that in the case without a Hamming window, a lot of energy in the spectrum was in frequencies other than those of the signal. This is because of the Gibbs phenomenon.
- We observe that the windowed transform is much better in terms of the magnitude spectrum. Only components near the frequencies of the input signal are present, while others are mostly 0. The reason that some frequencies near the actual peak are present is because multiplying by a window in the time domain corresponds to a convolution in the frequency domain with its fourier transform. This means that the delta functions in the frequency domain are smeared out by the spectrum of the Hamming window.

4 Peak frequency and phase estimator

We implement an estimator to calculate peak frequency and phase of non periodic sinusoids with added noise.
The estimator calculates error for a given vector, and increases windowing time and samples until the error falls below the threshold.
The estimator is based on the following: Calculate estimate of frequency by a weighted average of frequencies with $|X(jw)|^2$ serving as weights. Then calculate the phase at the estimated frequency.
Errors are also calculated in a similar way(weighted sum of deviations)
Actual $\omega = 0.6$
Actual Phase $= 0.3$

4.1 Code

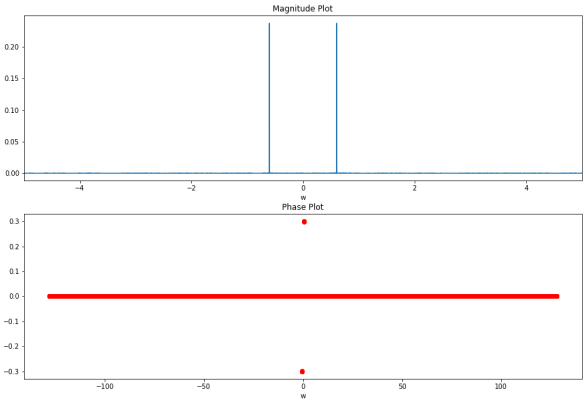
```

#Estimator for frequency and phase:
```

```
def estimator(w,X):
    mag = np.abs(X)
    maxi = np.max(mag)
    mag[np.where(np.abs(X )<1e-3*maxi)]=0
    w_est = np.sum(np.abs(w)*mag**2)/np.sum(mag**2)
    w_err = np.sqrt((np.sum((np.abs(np.abs(w)-w_est)**2)*mag**2))/np.sum(
    mag**2))
    w_est_arrindex = int(w_est/(w[1]-w[0])+len(w)/2)
    phase_est = np.angle(X[w_est_arrindex])
    w_err_arrindex = int(w_err/(w[1]-w[0]))
    X_temp = X[w_est_arrindex-1-w_err_arrindex:w_est_arrindex+2+
    w_err_arrindex]
    phase_err = np.sqrt(np.sum(np.abs((np.angle(X_temp)-phase_est)*(X_temp)
    **2))/np.sum(np.abs(X_temp)**2))
    return w_est, w_err, phase_est, phase_err

N = 256
Tmax = 2*np.pi

w_err = 1
w_est = 1
while(w_err>0.25*w_est):
    t = np.linspace(-Tmax/2,Tmax/2,N+1)[:-1]
    x = np.cos(0.6*t+0.3)+0.1*np.random.randn(N)
    w, X = fftcompute(x*hamming(N),Tmax)
    w_est, w_err, phase_est, phase_err = estimator(w,X)
    N = 2*N
    Tmax = 2*Tmax

print(w_est)
print(w_err)
print(phase_est)
print(phase_err)
fftplot(w,X,'noisy',[-5,5])
```

4.2 Outputs

Actual $\omega = 0.6$
Estimated $\omega = 0.6000003159957366$
Error in $\omega = 0.00016395895153615018$
Actual phase $= 0.3$
Estimated phase $= 0.2999798744761099$
Error in phase $= 0.007749512060093491$

4.3 Plots



5 Chirp signal

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

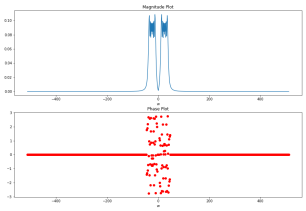
We find FFT of chirp with and without windowing,
Then we make slices of the vector and find FFT of the slices, with and without windowing to plot instantaneous frequency

5.1 Codes

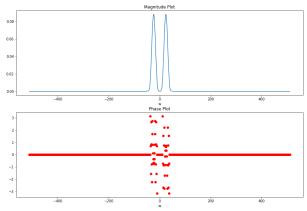
```
N = 1024
wind = 64
Tmax = 2*np.pi
t = np.linspace(-Tmax/2,Tmax/2,N+1)[:-1]
dt = t[1]-t[0]
x = np.cos(16*t*(1.5+0.5*t/np.pi))
#Without Hamming
w, X = fftcompute(x,Tmax)
fftplot(w, X,'noslicenohamm',None)
#With Hamming
y = x*hamming(len(x))
w, X = fftcompute(y,Tmax)
fftplot(w, X,'nosliceyeshamm',None)

#Question 6:
x_slice = np.zeros((N-wind,wind))
y_slice = np.zeros((N-wind,wind))
X = np.zeros((N-wind,wind))
Y = np.zeros((N-wind,wind))

for i in range(N-wind):
    x_slice[i] = x[i:i+wind]
    y_slice[i] = x_slice[i]*hamming(wind)
```

```
    X[i] = (1/wind)*np.fft.fftshift(np.fft.fft(np.fft.ifftshift(x_slice[i]))).real
    Y[i] = (1/wind)*np.fft.fftshift(np.fft.fft(np.fft.ifftshift(y_slice[i]))).real

w = np.linspace(-np.pi/dt,np.pi/dt,wind+1)[:-1]
time = np.linspace(-Tmax,Tmax-wind*Tmax/N,N-wind+1)[:-1]


fig = plt.figure(figsize=(15,10))
ax = plt.axes(projection='3d')

W, Time = np.meshgrid(w, time)

ax.plot_surface(W, Time, np.abs(X), cmap='viridis', edgecolor='none')
ax.set_title('Surface plot of |X(t)|')
ax.set_xlabel('w')
ax.set_ylabel('Time')
plt.savefig('surfnohamm.png')
plt.show()

fig2 = plt.figure(figsize=(15,10))
ax2 = plt.axes(projection='3d')

ax2.plot_surface(W, Time, np.abs(Y), cmap='viridis', edgecolor='none')
ax2.set_title('Surface plot of |Y(t)|')
ax2.set_xlabel('w')
ax2.set_ylabel('Time')
plt.savefig('surfnohamm.png')
plt.show()
```

5.2 Plots



(a) FFT of full signal: Without windowing



(b) FFT of full signal:With Hamming windowing

After windowing, we now observe that the frequencies are more confined to the range between 16 and 32, as expected. The extra components due to the discontinuity have been suppressed.

Now we we take the DFT of a small window of samples around each time instant, and plot a 2D surface of the resulting spectra vs time. We do this first for without a hamming window, analyse the results and apply windowing

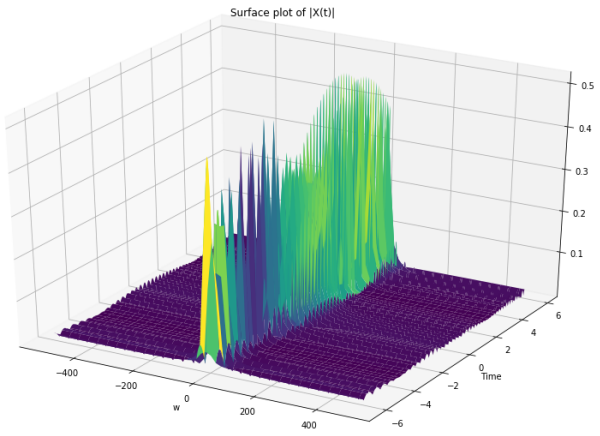


Figure 1: Surface plot of $|X_s(jw,t)|$: Without windowing

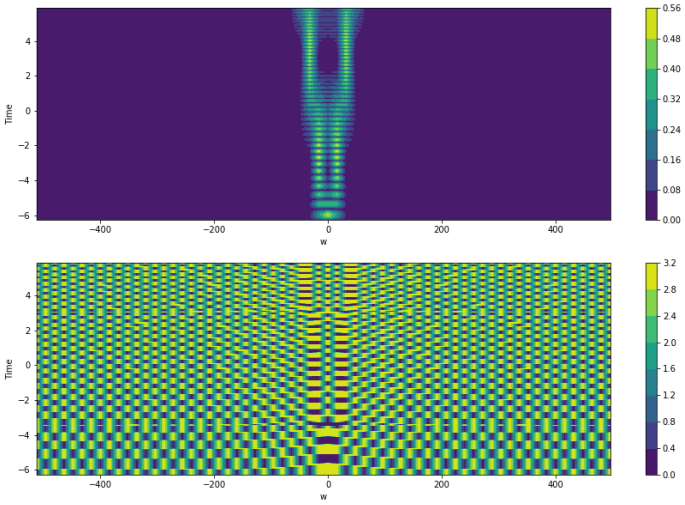


Figure 2: Magnitude and phase contour plots of $|X_s(jw,t)|$: Without windowing

We observe that the frequency components with high magnitude are concentrated around a frequency of 16 in the first half of the chirp, and then suddenly move to a frequency of 32 in the second half. We expect only the inital and final frequencies to be 16 and 32 respectively, and not over the whole of the first and second halves. This is

because we have not windowed the signal
We now apply windowing:

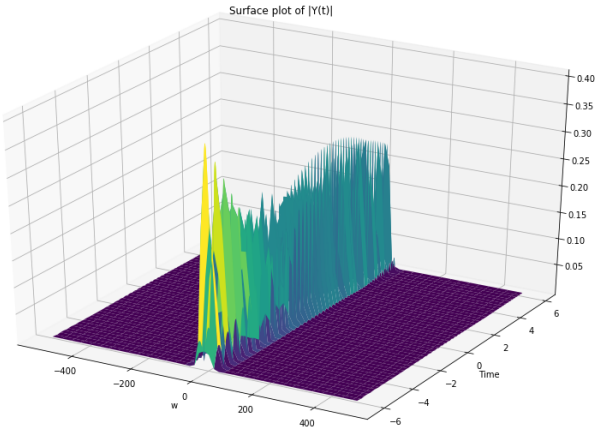


Figure 3: Surface plot of $|Y_s(jw,t)|$: With windowing

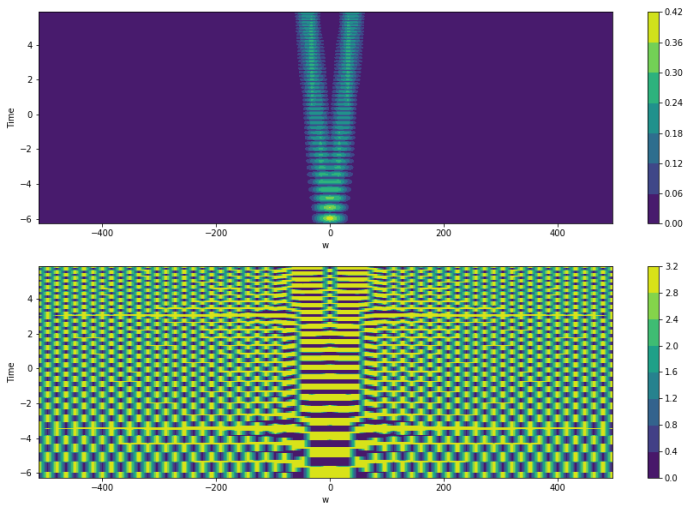


Figure 4: Magnitude and phase contour plots of $|Y_s(jw,t)|$: With windowing

We observe a much more gradual change in the frequency in the signal. The change is linear from 16 to 32, matching with our expectations.

6 Conculsions

- From the above examples, it is clear that using a Hamming window before taking a DFT helps in reducing the effect of Gibbs phenomenon arising due to discontinuities in periodic extensions.
- However, this comes at the cost of spectral leakage. This is basically the blurring of the sharp peaks in the DFT. It occurs because of convolution with the spectrum of the windowing function. Deltas in the original spectrum are smoothed out and replaced by the spectrum of the windowing function.
- We used this windowed DFT to estimate the frequency and phase of an unknown sinusoid from its samples.
- By performing localized DFTs at different time instants, we obtained a time-frequency plot which allowed us to better analyse signals with varying frequencies in time.