Applied Programming Lab: Assignment8

Saurabh Vaishampayan: EP17B028

10 May 2020

1 Introduction

The assignment involves implementation of Fast Fourier Transform, which is the fastest way to compute DFT. We implement FFT and analyse the results. Finally we try to approximate the spectrum of a gaussian signal(which is not bandlimited), by iteratively increasing the window size and samples till error falls below required threshold.

2 FFT and IFFT

2.1 Description

We implement the FFT and IFFT on a random vector and compare the reconstructed vector to the original to see how well it can be reconstructed

2.2 Code

```
import numpy as np
import matplotlib.pyplot as plt

#Question 1: Working out examples-

#FFT of a random sequence:
x = np.random.rand(8)
X = np.fft.fft(x)
y = np.fft.ifft(X)
print(np.c_[x,y])
print(np.abs(x-y).max())
```

2.3 Results

Left column represents the original while right is reconstruuucted: [[0.1384572 +0.j 0.1384572 +0.j]
[0.7350184 +0.j 0.7350184 +0.j]
[0.92227381+0.j 0.92227381+0.j]
[0.9475903 +0.j 0.9475903 +0.j]
[0.4541859 +0.j 0.4541859 +0.j]
[0.78131609+0.j 0.78131609+0.j]
[0.63843986+0.j 0.63843986+0.j]

[0.51598968+0.j 0.51598968+0.j]]
Error = 1.6653345369377348e-16
We observe that the reconstructed signal almost exactly matches the original albeit for a small error. There is also some non zero imaginary component added

3 Spectrum of sin(5t)

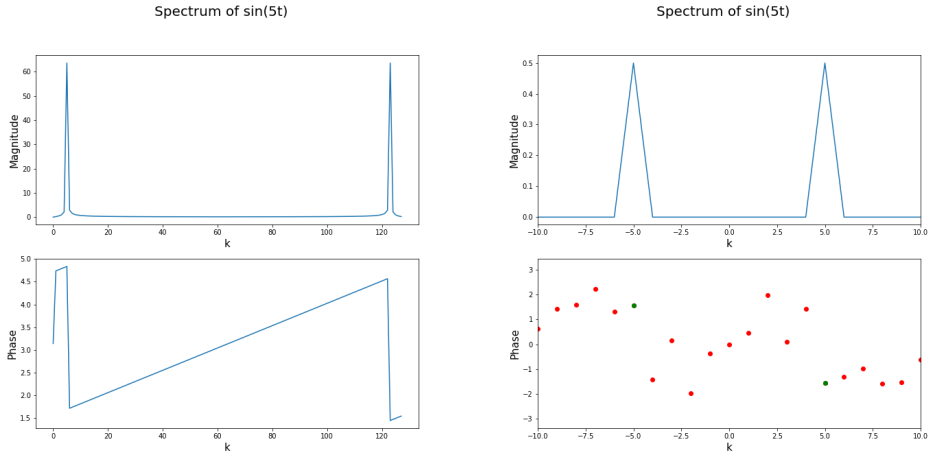
3.1 Description

First the examples given in the assignment are worked out:

3.2 Codes

```
N = 128
t = np.linspace(0, 2*np.pi, N)
x = np.sin(5*t)

X = np.fft.fft(x)
fig, ax = plt.subplots(2,1,figsize=(10,10))
ax[0].plot(np.abs(X))
ax[0].set_xlabel('k', size=15)
ax[0].set_ylabel('Magnitude', size=15)
ax[1].plot(np.unwrap(np.angle(X)))
ax[1].set_xlabel('k',size=15)
ax[1].set_ylabel('Phase',size=15)
fig.suptitle('Spectrum of sin(5t)',size=20)
fig.savefig('sin5t_1.png')
plt.show()


N = 128
t = np.linspace(0,2*np.pi,N+1)
t = t[:-1]
x = np.sin(5*t)
X = (1/N)*np.fft.fftshift(np.fft.fft(x))
w = np.linspace(-N/2,N/2,N+1)[:-1]
fig, ax = plt.subplots(2,1,figsize=(10,10))

ax[0].plot(w, np.abs(X))
ax[0].set_xlabel('k', size=15)
ax[0].set_ylabel('Magnitude', size=15)
ax[0].set_xlim([-10,10])

ax[1].plot(w, np.angle(X),'ro')
ii = np.where(np.abs(X)>1e-3)
ax[1].plot(w[ii], np.angle(X[ii]),'go')
ax[1].set_xlabel('k',size=15)
ax[1].set_ylabel('Phase',size=15)
ax[1].set_xlim([-10,10])
fig.suptitle('Spectrum of sin(5t)',size=20)
fig.savefig('sin5t_2.png')
plt.show()
```

3.3 Plots



4 Amplitude Modulation

4.1 Description

Finding the DFT of the Amplitude modulated wave:

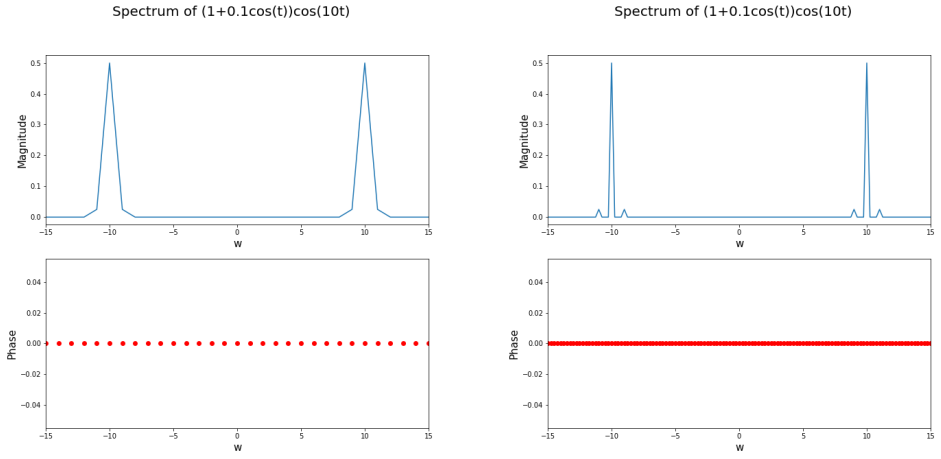
$$x(t) = (1+0.1\cos(t))\cos(10t)$$

4.2 Codes

```
def fftcompute(x,Twindow):
    N = len(x)
    X = (1/N)*np.fft.fftshift(np.fft.fft(x))

    w_int = np.pi/Twindow
    w = w_int*np.linspace(-N,N,N+1)[:-1]
    return w, X

def fftplot(w,X,funcname,figtitle,tol,xlimit):

    fig, ax = plt.subplots(2,1,figsize=(10,10))

    mag = np.abs(X)

    ax[0].plot(w, mag)
    ax[0].set_xlabel('w', size=15)
    ax[0].set_ylabel('Magnitude', size=15)
    if(xlimit!=None):
        ax[0].set_xlim(xlimit)

    phase = np.angle(X)
```

```
    phase[np.where(mag<tol)]=0
    phase[np.where(np.abs(phase)<tol)]=0
    ax[1].plot(w, phase,'ro')
    ax[1].set_xlabel('w',size=15)
    ax[1].set_ylabel('Phase',size=15)
    if(xlimit!=None):
        ax[1].set_xlim(xlimit)
    fig.suptitle('Spectrum of '+funcname,size=20)
    fig.savefig(figtitle+'.png')
    plt.show()

N = 128
t = np.linspace(0,2*np.pi,N+1)[:-1]
x = np.cos(10*t)+0.1*np.cos(10*t)*np.cos(t)
w, X = fftcompute(x,2*np.pi)
fftplot(w,X,'(1+0.1cos(t))cos(10t)','am1',1e-3,[-15,15])

N = 512
beg = -4*np.pi
end = 4*np.pi
Twind = end-beg
t = np.linspace(beg,end,N+1)[:-1]
x = np.cos(10*t)+0.1*np.cos(10*t)*np.cos(t)
w, X = fftcompute(x,Twind)
fftplot(w,X,'(1+0.1cos(t))cos(10t)','am2',1e-3,[-15,15])
```

4.3 Plots



4.4 Inference

The phase is 0 everywhere as expected, as the terms are only cosine waveforms which have real DFTs. The magnitude has two large peaks at frequency of 10 corresponding to the carrier cosine wave. This wave is then multiplied by a low amplitude low frequency cosine in the time domain. This corresponds to convolution in the frequency domain. This results in the two sidebands next to each of the carrier bands. Upon increasing the window, resolution of spectrum increases

5 Spectra of $\sin^3(t)$ and $\cos^3(t)$

5.1 Description

$$\cos^3(t) = \frac{3}{4}\cos(t)+\frac{1}{4}\cos(3t)$$

We expect phase to be zero everywhere and amplitude to be half of 0.75 at 1rad/s and half of 0.25 at 3rad/s

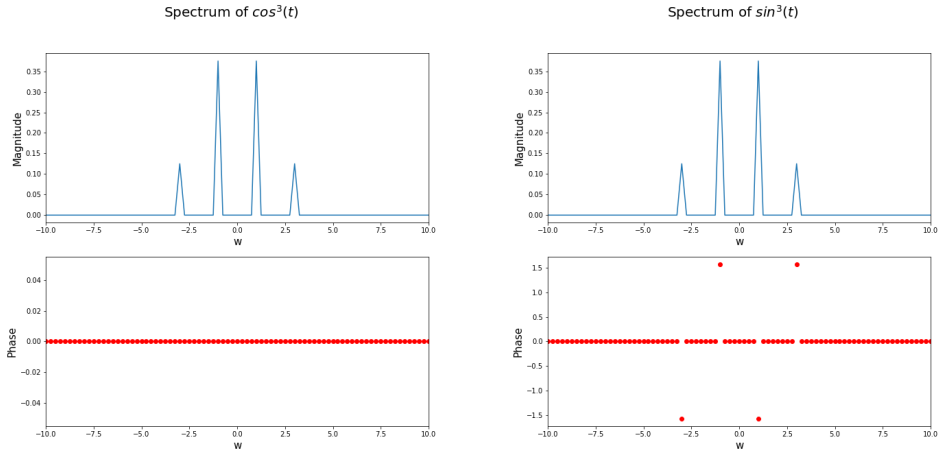
$$\sin^3(t) = \frac{3}{4}\sin(t)-\frac{1}{4}\sin(3t)$$

We expect amplitude to be half of 0.75 at 1rad/s and half of 0.25 at 3rad/s and phase to be $\pm\ \pi/2$ since one is a negative sinusoid

5.2 Codes

```
x1 = (np.cos(t))**3
x2 = (np.sin(t))**3

w, X = fftcompute(x1, Twind)
fftplot(w,X,r'$cos^{3}(t)$','coscubed',1e-3,[-10,10])

w, X = fftcompute(x2, Twind)
fftplot(w,X,r'$sin^{3}(t)$','sincubed',1e-3,[-10,10])
```

5.3 Plots



5.4 Inference

We obtain peaks at 1rad/s and 3rad/s with intensities of half of 0.75 and 0.25, as expected. Phase is zero for $\cos^3(t)$ as expected and $\pm\pi/2$ for $\sin^3(t)$ as expected

6 Frequency Modulation

6.1 Description

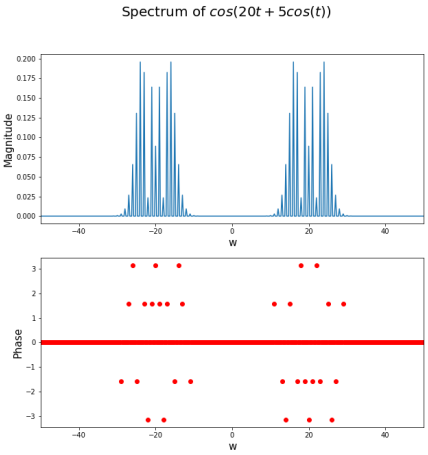
We find the DFT of the following frequency modulated signal:

$$x(t) = \cos(20t+5\cos(t))$$

6.2 Code

x = np.cos(5*np.cos(t)+20*t) w, X = fftcompute(x, Twind) fftplot(w,X,r'$cos(20t+5cos(t))$','fm',1e-3,[-50,50])

6.3 Plots



6.4 Inference

There are many more sidebands compared to AM in the case of FM. Most of the energy of the signal is also present in these sidebands rather than the carrier band in the case of AM.

7 FT of Gaussian

7.1 Description

The fourier transform of a signal x(t) is defined as follows:

$$X(j\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt$$

We can approximate this by the fourier transform of the windowed version of the signal x(t), with a sufficiently large window. Let the window be of size T. We get:

$$X(j\omega) \approx \int_{-T/2}^{T/2} x(t)e^{-j\omega t}dt$$

We can write the integral approximately as a Reimann sum:

$$\mathrm{X(j}\omega) \approx \frac{\triangle t}{2\pi}\sum_{-N/2}^{N/2-1} x(n\triangle t)e^{-j\omega\triangle t}$$

Where we divide the integration domain into N parts, each part being $\triangle t/2\pi$ apart. Also we will be sampling at $\omega = \frac{2\pi k}{n}$, where k runs from -N/2 to N/2 -1.

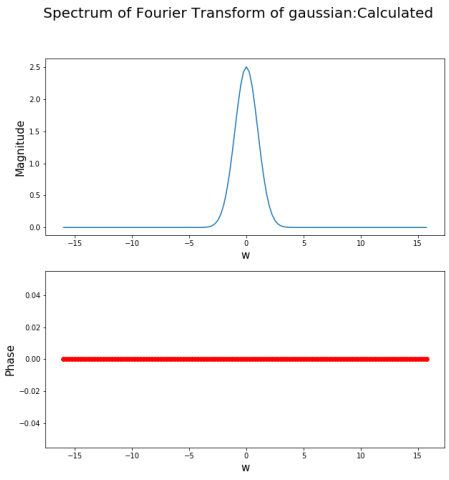
$$\mathrm{X(j}\omega) \approx \frac{\triangle t}{2\pi}\ \mathrm{DFT[x(n}\triangle\mathrm{t)]}$$

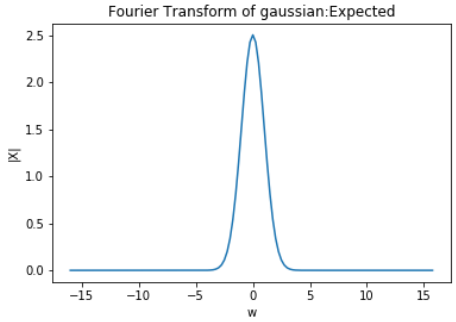
We implement a function to estimate this continuous time fourier transform by windowing the input function, sampling, and then finding the DFT is written below. The function iteratively increases window size and sample number until the consecutive total absolute error between estimates reduces below a given threshold. We also compare with the analytical expression of the Fourier Transform

7.2 Code

```
T = 2*np.pi
N = 32
tol = 1e-6
err = 1+tol

t = np.linspace(-T/2,T/2,N+1)[:-1]
x = np.exp(-t**2/2)
Xold = (T/N)*np.fft.fft(np.fft.ifftshift(x))

while(err>tol):

    T = 2*T
    N = 2*N

    t = np.linspace(-T/2,T/2,N+1)[:-1]
    x = np.exp(-t**2/2)

    Xnew = (T/N)*(np.fft.fft(np.fft.ifftshift(x)))

    err = np.sum(np.abs(Xnew[::2]-Xold))

    Xold = Xnew

print(err)
w = np.linspace(-np.pi*N/T,np.pi*N/T,N+1)[:-1]
Xo = (np.sqrt(2*np.pi))*np.exp(-w**2/2)
Xnew = np.fft.fftshift(Xnew)
fftplot(w, Xnew, 'Fourier Transform of gaussian:Calculated','gaussian',1e
    -6,None)

plt.plot(w,np.sqrt(2*np.pi)*np.exp(-w**2/2))
plt.title('Fourier Transform of gaussian:Expected')
plt.xlabel('w')
plt.ylabel('|X|')
plt.savefig('gaussexp.png')
plt.show()
```

7.3 Plots





7.4 Inference

Error(Output) : 3.36186609963244e-08

1. From the above pairs of plots, it is clear that with a sufficiently large window size and sampling rate, the DFT approximates the CTFT of the gaussian.

2. This is because the magnitude of the gaussian quickly approaches 0 for large values of time. This means that there is lesser frequency domain aliasing due to windowing.