EE2703 : Applied Programming Lab Assignment 8

Andapally Snehitha
EE20B008

April 13, 2022

Introduction

In this assignment, we shall look at DFTs, using the numpy.fft toolbox.

The Accuracy of the numpy.fft Package

To check how powerful and accurate the package is, we shall take the DFT of a sequence of random numbers, then take it's IDFT and compare the two, as to how close they are. This is done using the commands np.fft.fft and np.fft.ifft. We also have to use np.fft.fftshift to shift the $[\pi, 2\pi]$ portion of the DFT to the left as it represents negative frequency, i.e., $[-\pi, 0]$.
The following piece of code accomplishes this task:

```python
xOriginal = np.random.rand(128)
X = fft.fft(xOriginal)
xComputed = fft.ifft(X)
plt.figure(0)
t = np.linspace(-64, 63, 128)
plt.plot(t, xOriginal, 'b', label='Original $x(t)$')
plt.plot(t, np.abs(xComputed), 'g', label='Computed $x(t)$')
plt.xlabel(r'$t\ \to$')
plt.grid()
plt.legend()
plt.title('Comparison of actual and computed $x(t)$')
maxError = max(np.abs(xComputed-xOriginal))
print(r'Magnitude of maximum error between actual and
    computed values of the random sequence: ', maxError) #
    order of 1e-15
```

The error came out to be of the order of 10^{-15}. When the two sequences xOriginal and xComputed are plotted together, this is the result:

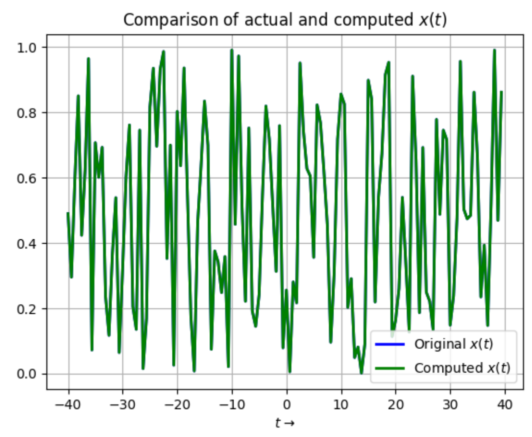

Figure 1: Comparison of the true and computed values of the random sequence

The two sequences overlap and cannot be differentiated !! This shows that the numpy.fft package is very accurate.

Spectrum of sin(5t)

We calculate the DFT of f(t) by the method mentioned in the above section. Then, we plot the phase and magnitude of the DFT and the following is obtained for sin(5t):

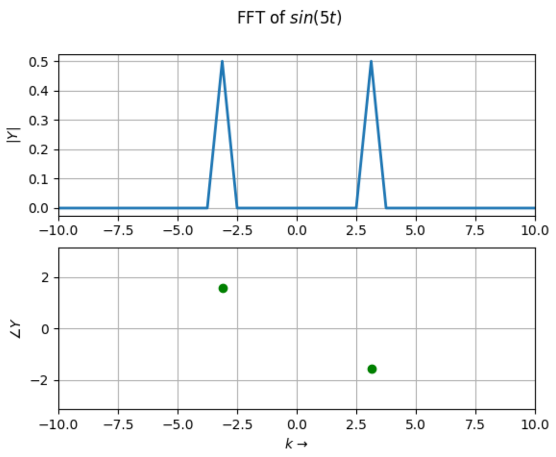

Figure 2: Spectrum of sin(5t)

This is expected, because:

$$sin(5t) = (e^{5jt} - e^{-5jt})/2 - - > (1)$$

So, the frequencies present in the DFT of sin(5t) and the phase associated with them are $\omega = \pm 5 rad/secand \phi = \pi/2$ rad/sec respectively. This is exactly what is shown in the above plot.

Amplitude Modulation with (1 + 0.1cos(t))cos(10t)

We have,

$$(1+0.1cos(t))cos(10t) = (e^{10jt}+e^{-10jt})/2+0.1*(e^{11jt}+e^{-11jt}+e^{9jt}-e^{-9jt})/4-- > (2)$$

Plotting the DFT using the numpy.fft package, we get:

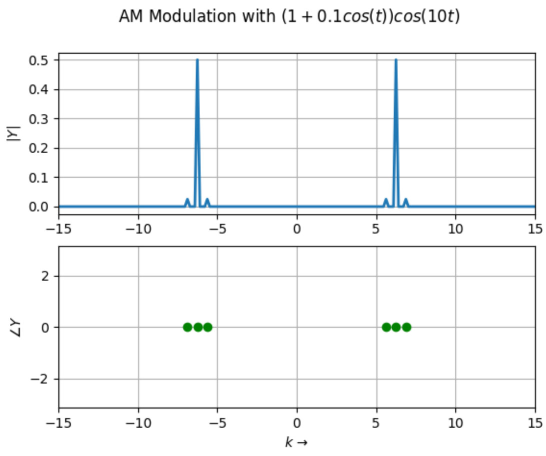

Figure 3: DFT of (1 + 0.1cos(t))cos(10t)

Figure (3) is in line with the theory we have discussed above.

Spectrum of $sin^3(t)$ and $cos^3(t)$

DFT Spectrum of $sin^3(t)$:

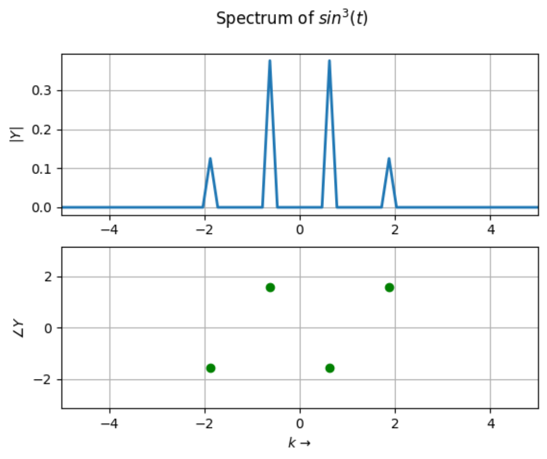

Figure 4: Spectrum of $sin^3(t)$

DFT Spectrum of $cos^3(t)$:

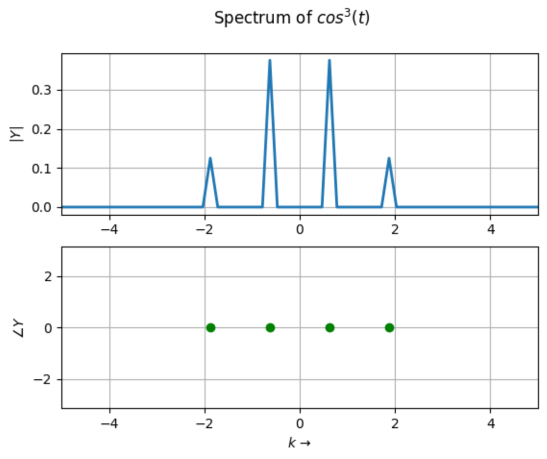

Figure 5: Spectrum of $cos^3(t)$

The above 2 figures are expected because:

sin$^3(t)$=(3sin(t)-sin(3t))/4 −¿(3)
$cos^3(t)$=(3cos(t)+cos(3t))/4 −¿(4)

So, we expect peaks $\omega = \pm rad/sec and \omega = \pm 3 rad/sec$.

Frequency Modulation with cos(20t + 5cos(t))

When we compare this result with that of the Amplitude Modulation as seen in Fig (3), we see that there are more side bands, and some of them have even higher energy than $= \pm 20$ rad/sec.
The DFT of cos(20t + 5cos(t)) can be seen below:

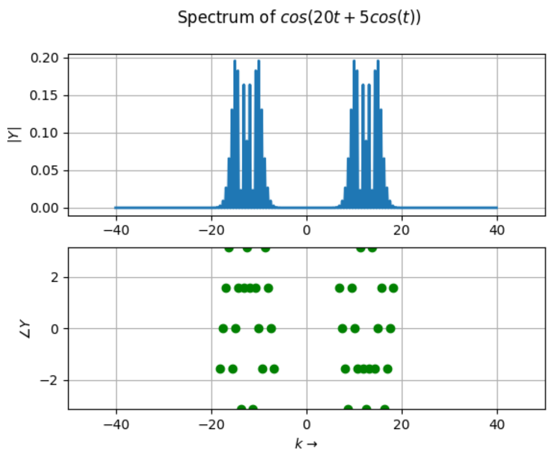

Figure 6: DFT of cos(20t + 5cos(t))

DFT of a Gaussian

The DFT of a gaussian is also a gaussian, as shown below:

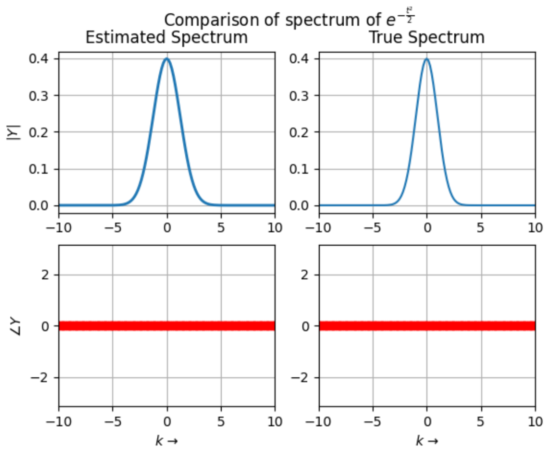

Figure 7: Gaussian Spectrum

Conclusion

1. We have thus found out the DFT's of various signals using the numpy.fft package. We need to use the numpy.fft.fftshift() and numpy.fft.ifftshift() methods to fix distortions in the phase response.
2. FFT works well for signals with samples in 2^k, as it divides the samples into even and odd and goes dividing further to compute the DFT.
3. That's why we use no of samples in the problems in the problems above taken as powers of 2.