ESE 531: Homework 8

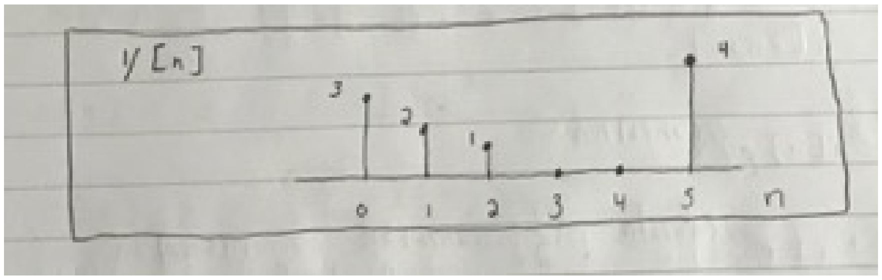
Noah Schwab

April 13, 2022

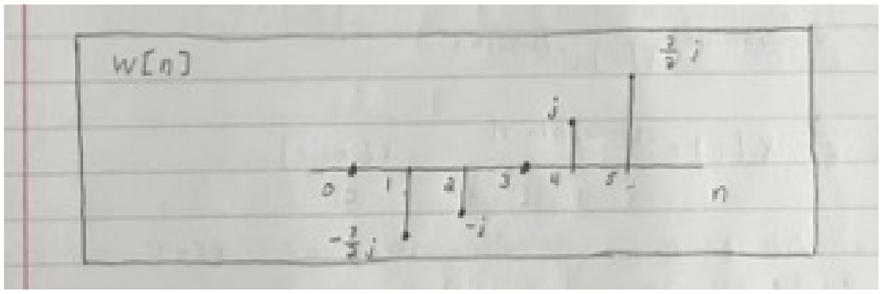
Problem solutions with figures are shown below. Work and code is shown in attachments at end of document.

8.23

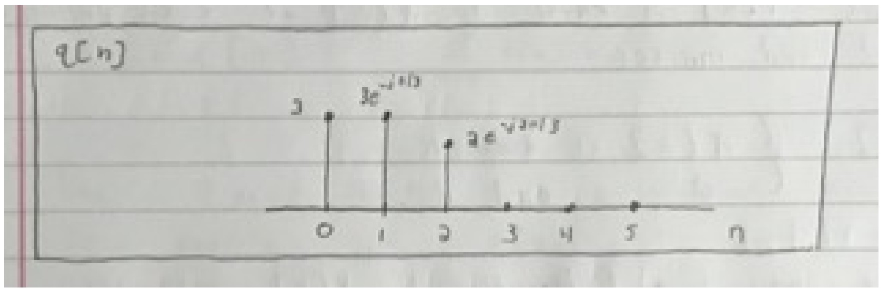
a) $y[n]$



b) $w[n]$



c) $q[n]$



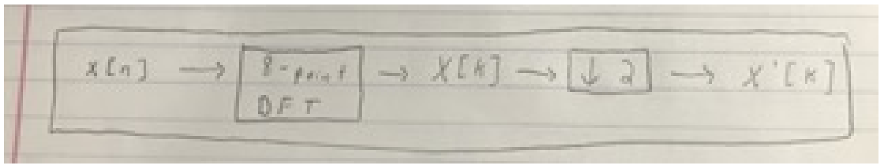
8.14

a) (iii): $r[n] = x[n] + x[n+512], 0 \leq n \leq 511$

b) (v): $y[n] = \frac{1}{2}(x[n] + x[1023 - n]), 0 \leq n \leq 1023$

9.28

We can execute the 4-point DFT by downsampling the 8-point DFT, as shown in the bloc diagram.



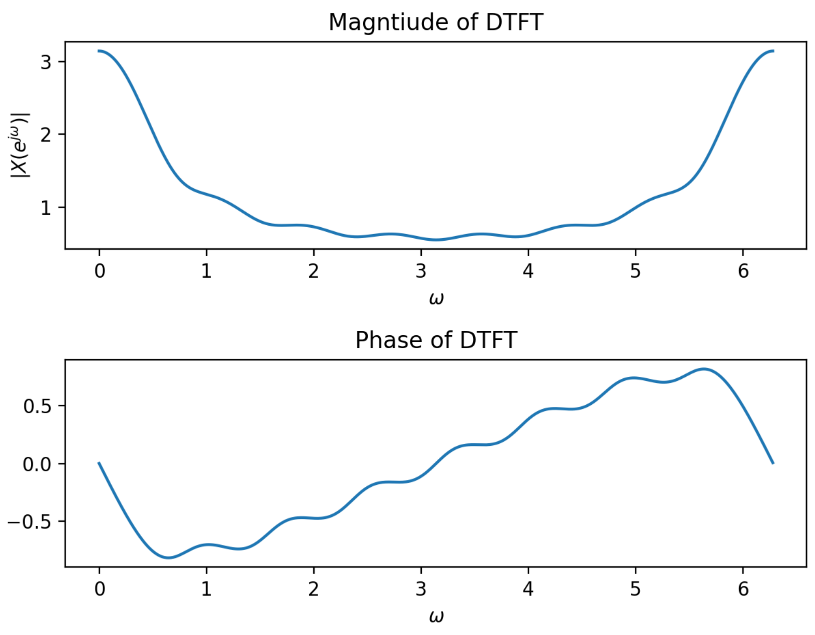
The total cost would be \$1.

Matlab Problem 1

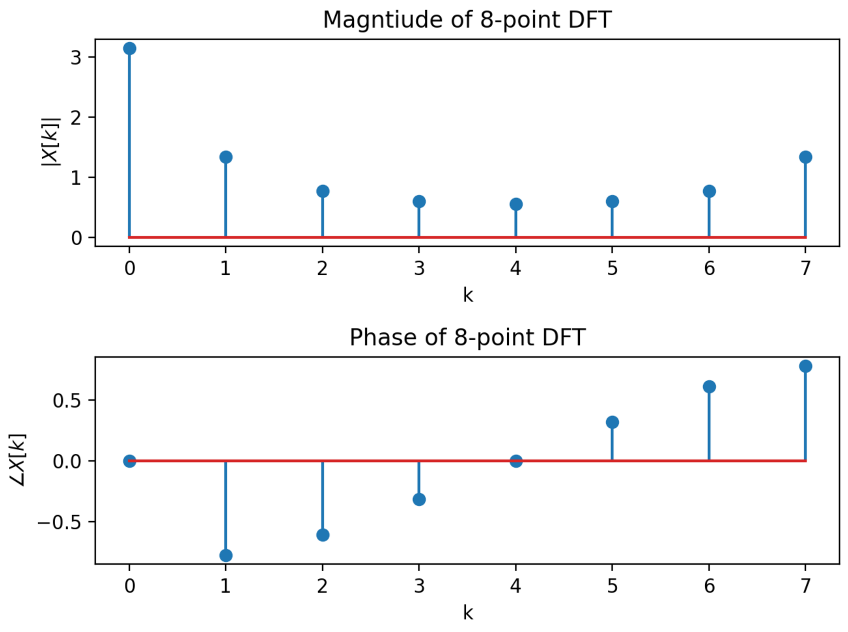
a) Using the definition of the DTFT and the formula for a finite geometric series:

$X(e^{j\omega}) = \sum_{n=0}^{7}(0.7)^n e^{-j\omega n} = \sum_{n=0}^{7}(0.7e^{-j\omega})^n = \frac{1-(0.7e^{-j\omega})^8}{1-0.7e^{-j\omega}}$

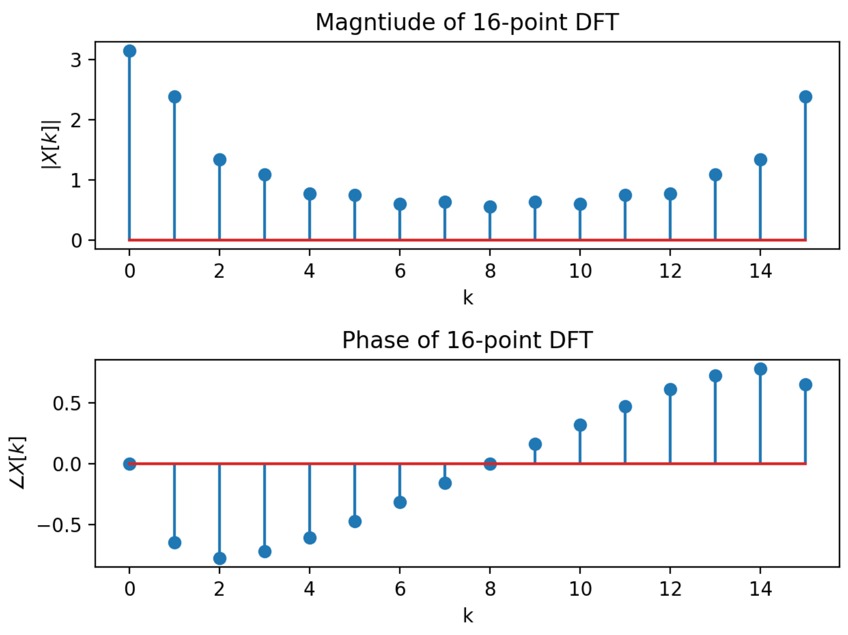
See the plot of the magnitude and phase below.



b) 8-point DFT

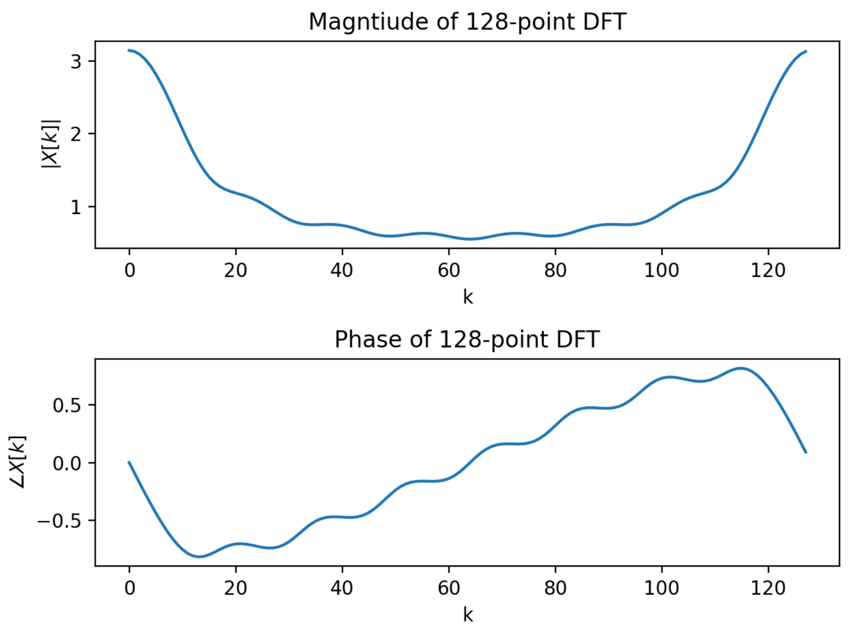


c) 16-point DFT



Zero-padding the signal increases the approximation of the DFT to the DTFT. If we consider the DFT as a sampled version of the DTFT, then zero-padding corresponds enables a higher sampling rate in the frequency domain, which can also be thought of as a better approximation of an infinite-time signal.

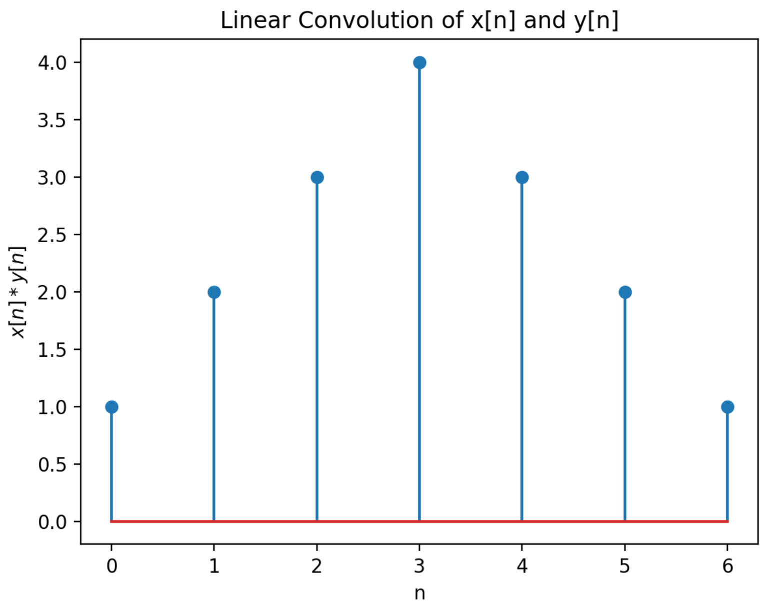
d) 128-point DFT



e) As we can see, the 128-point DFT is a very good approximation of the DTFT from part (a). This relates to the relationship between the digital frequency ω and the discrete frequency k, in that the DFT is a sampled version of the DTFT. Whereas the DTFT transforms an infinite duration, discrete time signal to the continuous frequency domain, the DFT transforms finite duration signals to the discrete frequency domain. Increasing the zero-padding corresponds to better approximating an infinite-duration discrete-time signal, of which the DFT is a better approximation of the DTFT. In other words, longer signals enable higher frequency sampling rates.

Matlab Problem 2

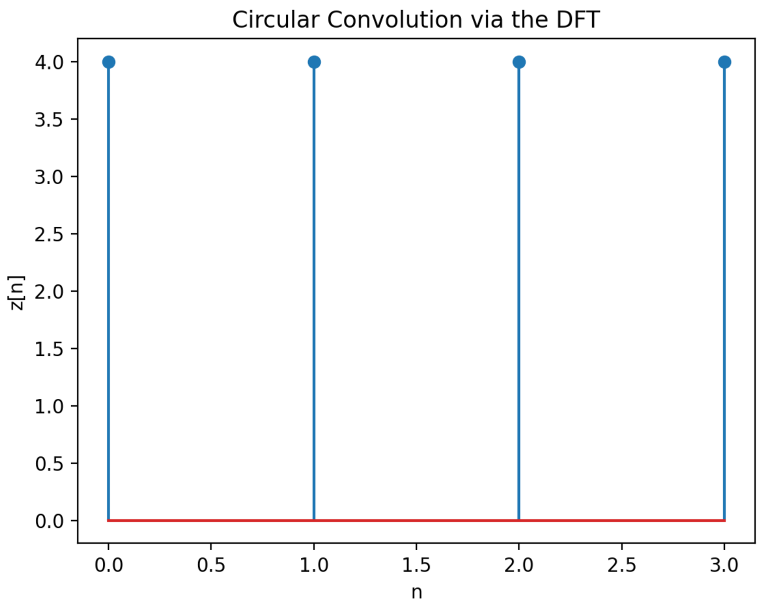
a) Linear Convolution



b) We can see that, since the two signals are identical and are rectangular pulses, the circular convolution is going to result in the inner product of the signals at each index in the output.

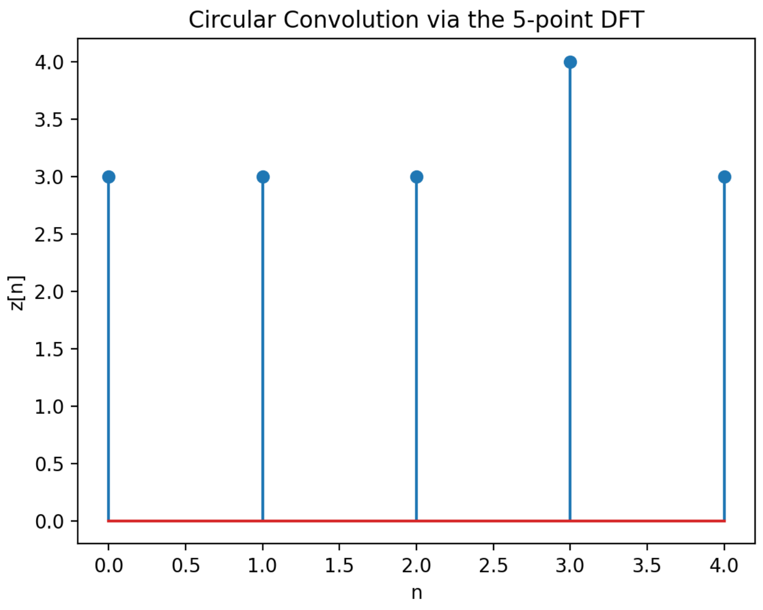
$x[n] \circledast y[n] = \sum_{m=0}^{3} x[m]y[((n-m))_3] = 4,\ 0 \leq n \leq 3$

c) Circular Convolution via the DFT



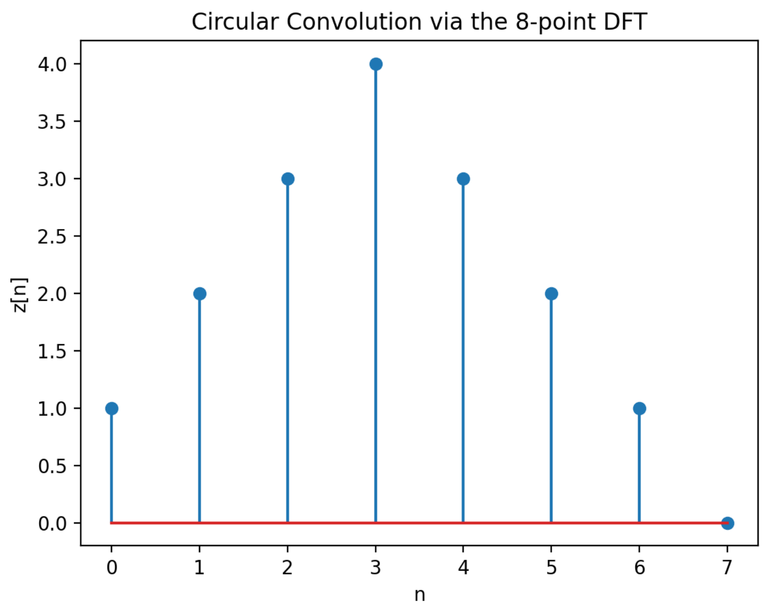
As we can see from the plot, this is consistent with our analytical solution from part (b). The only sample that is consistent with the result from part (a) is z[3] = 4.

d) Circular Convolution via 5-point DFT (one 0 padded)



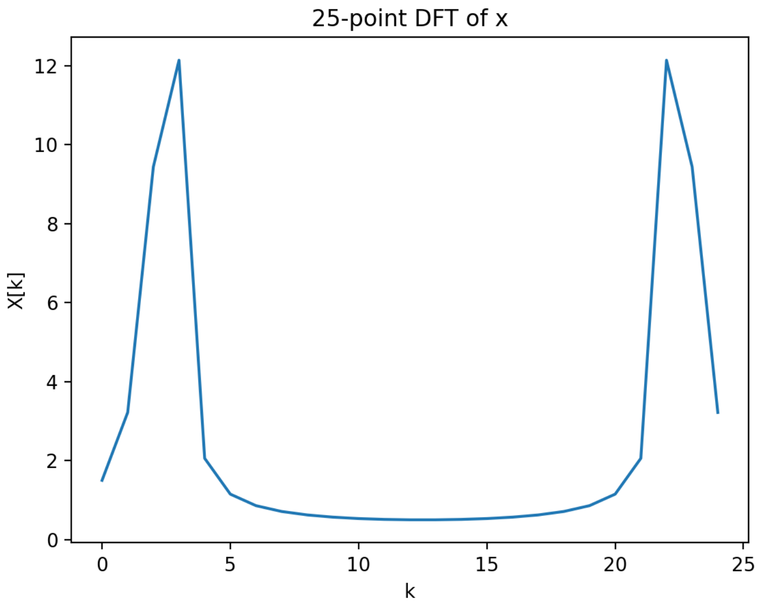
Now z[2], z[3], and z[4] agree with the linear convolution.

Following this pattern, we can see that a circular convolution between the signals zero-padded to double the length (N=8) will reproduce the linear convolution since the time-aliasing will be prevented, as shown in the plot below. This corresponds to padding the signals with 4 zeros.



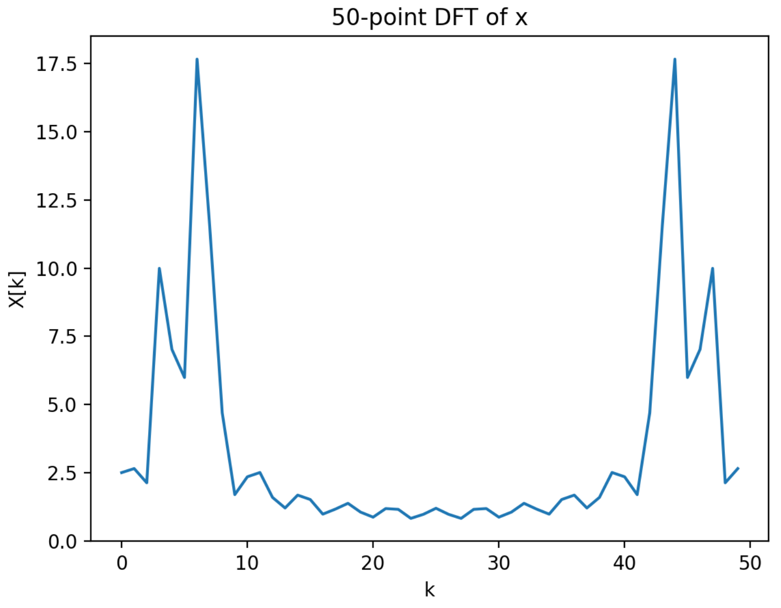
Matlab Problem 3

a) 25-point DFT

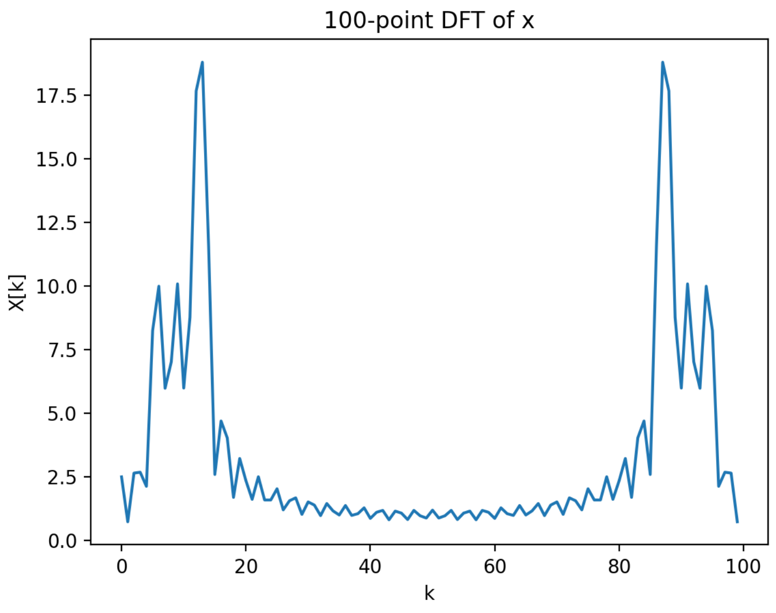


In the plot above, we can see two distinct frequencies at k=3 and k=23.

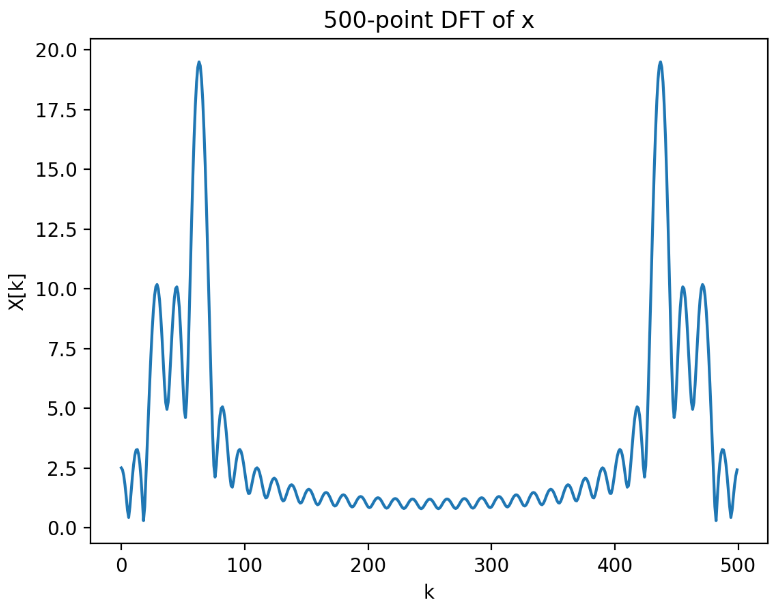
b) 50-point DFT



100-point DFT



500-point DFT



We do not gain any additional frequencies after the 100-point DFT. There are 6 distinguishable tones with values at $k = \pm 62.5, \pm 45.5, \pm 31.25$, or $\omega = \pm\pi/4, \pm 2\pi/11$, and $\pm \pi/8$

Noah Schwab
ESE 531

04/12/22

HW 8: DFT, FFT

8.23

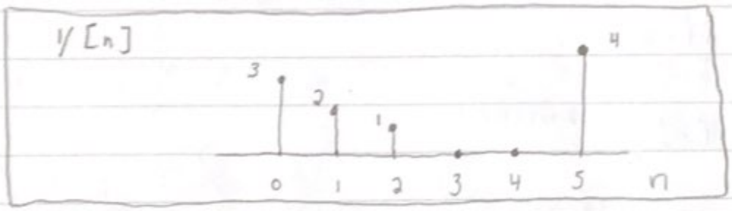
$$X[k] = \sum_{n=0}^{5} x[n] W_6^{kn} = \sum_{n=0}^{5} x[n] e^{-j(2\pi/6)kn}$$

a) $y[n] = \frac{1}{6} \sum_{k=0}^{5} Y[k] e^{j(2\pi/6)kn}$

$$= \frac{1}{6} \sum_{k=0}^{5} e^{-j(2\pi/6)5k} X[k] e^{j(2\pi/6)kn}$$

$$y[n] = \frac{1}{6} \sum_{k=0}^{5} X[k] e^{j(2\pi/6)k(n-5)} = x[n-5]$$

$y[n]$ is a five-sample circular shift of $x[n]$



b) $W[k] = \mathrm{Im}\{X[k]\}$

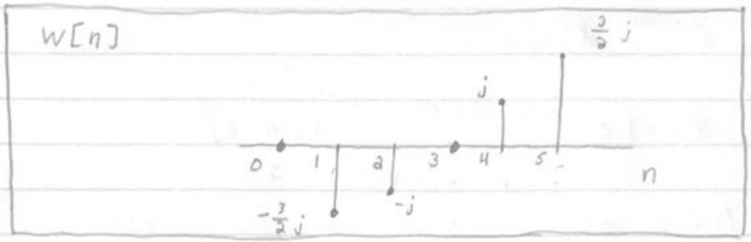
Eq 8.111: $x_{op}[n] \overset{DFT}{\longleftrightarrow} j\,\mathrm{Im}\{X[k]\}$

$$\Rightarrow -j x_{op}[n] \overset{DFT}{\longleftrightarrow} \mathrm{Im}\{X[k]\}$$

Therefore, $w[n] = -j x_{op}[n]$
$= -j \cdot \frac{1}{2} \{ x[((n))_N] - x^*[((-n))_N] \}$

Since $x[n]$ is real-valued:
$$w[n] = -\frac{1}{2}j \{ x[((n))_6] - x[((-n))_6] \}$$

$w[0] = -\frac{1}{2}j\,(x[0] - x[0]) = 0$
$w[1] = -\frac{1}{2}j\,(x[1] - x[5]) = -\frac{3}{2}j$
$w[2] = -\frac{1}{2}j\,(x[2] - x[4]) = -j$
$w[3] = -\frac{1}{2}j\,(x[3] - x[3]) = 0$
$w[4] = -\frac{1}{2}j\,(x[4] - x[2]) = j$
$w[5] = -\frac{1}{2}j\,(x[5] - x[1] = \frac{3}{2}j$



c) $Q[k] = X[2k+1]$

$X[k] = \sum_{n=0}^{5} x[n] e^{-j(2\pi/6)kn}$

$= 4 + 3e^{-j(2\pi/6)k} + 2e^{-j(2\pi/6)2k} + e^{-j(2\pi/6)3k}$

$Q[k] = 4 + 3W_6^{(2k+1)} + 2W_6^{2(2k+1)} + W_6^{3(2k+1)}$

$= 4 + 3W_6^1 W_6^{2k} + 2W_6^2 W_6^{4k} + W_6^3 W_6^{6k}$

Due to power property, $W_6^{2k} = W_3^k$, $W_6^{4k} = W_3^{2k}$, $W_6^{6k} = W_3^{3k}$

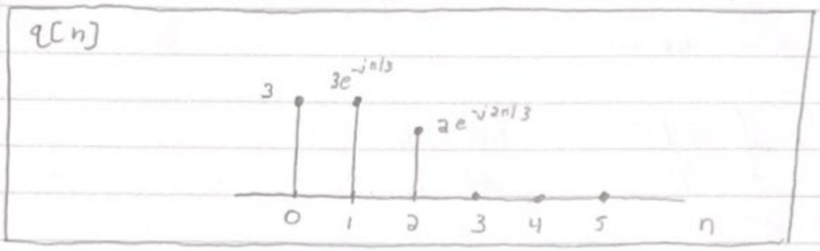
$= 4 + 3W_6^1 W_3^k + 2W_6^2 W_3^{2k} + W_6^3 W_3^{3k}$

Due to the periodicity of the DFT, $W_3^{3k} = W_3^{0k} = 1$

$Q[k] = 4 + 3e^{-j\pi/3} W_3^k + 2e^{-j2\pi/3} W_3^{2k} + e^{-j\pi}$
$= 3 + 3e^{-j\pi/3} W_3^k + 2e^{-j2\pi/3} W_3^{2k}$

Applying the inverse 3-point DFT :

$$q[n] = 3\delta[n] + 3e^{-j\pi/3}\delta[n-1] + 2e^{-j2\pi/3}\delta[n-2]$$



8.43

a) $R[K] = X[2K]$

$$r[n] = \frac{1}{N}\sum_{K=0}^{N-1} R[K]\, W_N^{-Kn} = \frac{1}{N}\sum_{K=0}^{N-1} X[2K]\, W_N^{-Kn}$$

$$r[n] = \frac{1}{512}\sum_{K\,\text{even}}^{1023} X[K]\, W_{512}^{-Kn}$$

$r[n]$ is the 512-point IDFT of the even samples of $X[K]$, which is essentially the 512-point IDFT of the 512-point DFT of $x[n]$. Since $x[n]$ is length 1024, and we take a 512-point DFT, the decimation results in aliasing in the time-domain. The time-domain representation is expanded by a factor of 2, so we get overlap between half modulo indices $\underline{r[n] = x[n] + x[n+512]}$.

$\boxed{\text{Statement (iii)}}$

b) $y[n]$ is the 1024-point IDFT of the upsampled version of $R[k]$. The upsampled version of $R[k]$ is equivalent to zero-inserting at odd indices after expansion. $Y[k]$ is equal to $X[k]$ at even indices and zero at odd indices

$$Y[k] = \begin{cases} X[k], & \text{even } k \\ 0, & \text{odd } k \end{cases}$$

In other words, we have transformed $X[k]$ into its periodic even components

$$y[n] = x_{ep}[n] = \frac{1}{2}\left(x[n] + x^*[((-n))_N]\right), \quad 0 \le n \le N-1$$

Assuming $x[n]$ is real:

$$\boxed{y[n] = \frac{1}{2}\left(x[n] + x[1023-n]\right), \quad 0 \le n \le 1023}$$

$\boxed{\text{Statement (v).}}$

9.28

The 4-point DFT of a 4-point seavence is given by

$$X[k] = \sum_{n=0}^{3} x[n] W_4^{kn}$$

The 8-point DFT on a 4-point seavence entails zero-padding, s.t $x[n] = 0$ for $4 \leq n \leq 7$, and computing

$$X'[k] = \sum_{n=0}^{7} x[n] W_8^{kn}$$

We can see that $X[k]$ is simply an expanded version of $X'[k]$, where $X[k] = X'[2k]$

$$X'[2k] = \sum_{n=0}^{7} x[n] W_8^{2kn} = \sum_{n=0}^{7} x[n]^{-j(2\pi/8)2kn}$$

$$= \sum_{n=0}^{3} x[n] e^{-j(2\pi/4)kn} = \sum_{n=0}^{7} x[n] W_4^{kn}$$

Since $x[n] = 0$ for $4 \leq n \leq 7$,

$$X'[2k] = \sum_{n=0}^{3} x[n] W_4^{kn} = X[k]$$

Therefore, we can execute the 4-point DFT by downsampling the 8-point DFT

$$\boxed{\begin{tikzcd} x[n] \arrow[r] & \boxed{\text{8-point DFT}} \arrow[r] & X[k] \arrow[r] & \boxed{\downarrow 2} \arrow[r] & X'[k] \end{tikzcd}}$$

<u>The total cost would be $1.</u>

```python
# ESE 531: HW8 Problem 2
# Author: Noah Schwab

# import libraries
import matplotlib.pyplot as plt
import numpy as np

# part a

# analytical expressision for DTFT of x[n]
w = np.arange(0, 2 * np.pi, 0.01)
X = np.array([(1 - (0.7 * np.exp(-1j * i)) ** 8) / (1 - 0.7 * np.exp(-1j * i)) for i in w])


# plot the magntiude and phase of DTFT
fig, axs = plt.subplots(2)

axs[0].plot(w, np.abs(X))
axs[0].set_title('Magntiude of DTFT')
axs[0].set_xlabel(r'$\omega$')
axs[0].set_ylabel(r'$|X(e^{j \omega})|$')

axs[1].plot(w, np.angle(X))
axs[1].set_title('Phase of DTFT')
axs[1].set_xlabel(r'$\omega$')
axs[1].set_ylabel(r'$\angle X(e^{j \omega})$')

plt.tight_layout()
plt.show()

# part b

# define x
x = np.array([0.7 ** n for n in range(8)])

# compute 8-point DFT and plot its magnitude and phase
X8 = np.fft.fft(x, n=8)

fig, axs = plt.subplots(2)

axs[0].stem(np.abs(X8))
axs[0].set_title('Magntiude of 8-point DFT')
axs[0].set_xlabel(r'k')
axs[0].set_ylabel(r'$|X[k]|$')

axs[1].stem(np.angle(X8))
axs[1].set_title('Phase of 8-point DFT')
axs[1].set_xlabel(r'k')
axs[1].set_ylabel(r'$\angle X[k]$')

plt.tight_layout()
plt.show()

# part c

# compute 16-point DFT and plot its magnitude and phase
# note: np.fft.fft automatically zero pads the input signal if the DFT length is larger than the signal length
X16 = np.fft.fft(x, n=16)

fig, axs = plt.subplots(2)

axs[0].stem(np.abs(X16))
axs[0].set_title('Magntiude of 16-point DFT')
axs[0].set_xlabel(r'k')
axs[0].set_ylabel(r'$|X[k]|$')

axs[1].stem(np.angle(X16))
```

```
axs[1].set_title('Phase of 16-point DFT')
axs[1].set_xlabel(r'k')
axs[1].set_ylabel(r'$\angle X[k]$')

plt.tight_layout()
plt.show()

# part d

# compute 128-point DFT and plot its magnitude and phase
# note: np.fft.fft automatically zero pads the input signal if the DFT length is larger than the signal length
X128 = np.fft.fft(x, n=128)

fig, axs = plt.subplots(2)

axs[0].plot(np.abs(X128))
axs[0].set_title('Magntiude of 128-point DFT')
axs[0].set_xlabel(r'k')
axs[0].set_ylabel(r'$|X[k]|$')

axs[1].plot(np.angle(X128))
axs[1].set_title('Phase of 128-point DFT')
axs[1].set_xlabel(r'k')
axs[1].set_ylabel(r'$\angle X[k]$')

plt.tight_layout()
plt.show()
```

```python
# ESE 531: HW8 Problem 3
# Author: Noah Schwab

# import libraries
import matplotlib.pyplot as plt
import numpy as np

# define sequences x[n] and y[n]
x = np.array([1, 1, 1, 1])
y = np.array([1, 1, 1, 1])

# part a

# perform linear convolution and plot the result
z = np.convolve(x, y)

plt.stem(z)
plt.title('Linear Convolution of x[n] and y[n]')
plt.xlabel('n')
plt.ylabel(r'$x[n] * y[n]$')
plt.show()

# part c

# compute the circular convolution via the DFT
X = np.fft.fft(x)
Y = np.fft.fft(y)

# product of DFT is equivalent to circular convolution of signal
Z = X * Y
z = np.fft.ifft(Z).real

# plot the result
plt.stem(z)
plt.title('Circular Convolution via the DFT')
plt.xlabel('n')
plt.ylabel('z[n]')
plt.show()

# part d

# we can simply repeat part c, but specify the length of DFT, as the fft function will automatically
# zero pad the signal if N is greater than the length of the signal
X = np.fft.fft(x, n=5)
Y = np.fft.fft(y, n=5)
Z = X * Y
z = np.fft.ifft(Z).real

plt.stem(z)
plt.title('Circular Convolution via the 5-point DFT')
plt.xlabel('n')
plt.ylabel('z[n]')
plt.show()

# we know that the signals must be zero-padded to double their length to produce the linear convolution result
X = np.fft.fft(x, n=8)
Y = np.fft.fft(y, n=8)
Z = X * Y
z = np.fft.ifft(Z).real

plt.stem(z)
plt.title('Circular Convolution via the 8-point DFT')
plt.xlabel('n')
plt.ylabel('z[n]')
plt.show()
```

```python
# ESE 531: HW8 Problem 4
# Author: Noah Schwab

# import libraries
import matplotlib.pyplot as plt
import numpy as np
from scipy.io import loadmat as loadmat

# load in tones.mat file
s =loadmat('/Users/noahhschwab/Desktop/ESE-531/HW8/tones.mat')
x = np.ravel(s['y1'])

# compute the 25-point DFT of x and plot its magnitude
X = np.fft.fft(x, n=25)

plt.plot(np.abs(X))
plt.title('25-point DFT of x')
plt.xlabel('k')
plt.ylabel('X[k]')
plt.show()

# compute the 50-point DFT of x and plot its magnitude
X = np.fft.fft(x, n=50)

plt.plot(np.abs(X))
plt.title('50-point DFT of x')
plt.xlabel('k')
plt.ylabel('X[k]')
plt.show()

# compute the 100-point DFT of x and plot its magnitude
X = np.fft.fft(x, n=100)

plt.plot(np.abs(X))
plt.title('100-point DFT of x')
plt.xlabel('k')
plt.ylabel('X[k]')
plt.show()

# compute the 500-point DFT of x and plot its magnitude
X = np.fft.fft(x, n=500)

plt.plot(np.abs(X))
plt.title('500-point DFT of x')
plt.xlabel('k')
plt.ylabel('X[k]')
plt.show()
```