

LAB TITLE AND CODE: SIGNAL PROCESSING LAB 1A (CCE281)

EXPERIMENT NUMBER : 9

DATE : 14/12/2021

DISCRETE FOURIER TRANSFORM* AIM :

for a given sequence $x[n]$, compute DFT coefficients using the direct and linear transformation methods. Sketch the magnitude and phase spectrum obtained from the same.

* SOFTWARE REQUIRED :

spyder IDE - Anaconda 3 2021.11 (Python 3.9.7 64-bit)
Publisher - Anaconda, Inc.

* THEORY : (DIRECT METHOD)

The discrete Fourier transform (DFT) is a method for converting a sequence of N complex numbers x_0, x_1, \dots, x_{N-1} to a new sequence of N complex numbers,

$$x_k = \sum_{n=0}^{N-1} x_n e^{-j 2\pi k n / N}$$

for $0 \leq k \leq N-1$.

The x_k are thought of as the values of a function, or signal, at equally spaced times $t = 0, 1, \dots, N-1$. The output x_k is a complex number which encodes the amplitude and phase of a sinusoidal wave with frequency $\frac{k}{N}$ cycles per unit time.

[This comes from Euler's formula -

$$e^{-j 2\pi k n / N} = \cos(2\pi k n / N) + j \sin(2\pi k n / N)$$

⑧

The effect of computing the x_k is to find the coefficients of an approximation of the signal by a linear combination of such waves. Since each wave has an integer number of cycles per N time units, the approximation will be periodic with period N . This approximation is given by the inverse Fourier transform -

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{j2\pi k n/N}$$

→ WIDDLER FACTOR -

It is denoted as W_N and defined as $W_N = e^{-j2\pi/N}$. Its magnitude is always maintained at unity. Phase of $W_N = -2\pi/N$. It is a vector on unit circle and is used for computational convenience. Mathematically, it can be shown as -

$$W_N^n = W_N^{n+2N} = W_N^{n+4N} = \dots$$

It is a function of n and period N .

Consider $N=8$, $n=0, 1, 2, 3, \dots, 14, 15, 16, \dots$

$$\Leftrightarrow W_8^0 = W_8^8 = W_8^{16} = \dots = \dots = W_8^{32} = \dots = 1 = 1 \text{ L0}$$

$$W_8^1 = W_8^9 = W_8^{17} = \dots = \dots = W_8^{33} = \dots = \frac{1}{\sqrt{2}} - j \frac{1}{\sqrt{2}} = 1 \text{ L1}$$

→ LINEAR TRANSFORMATION -

We know that -

$$\text{DFT}(k) = \text{DFT}[x(n)] = x\left(\frac{2\pi}{N} k\right) = \sum_{n=0}^{N-1} x(n) \cdot W_N^{-nk};$$

$$k=0, 1, \dots, N-1$$

$$x(n) = \text{IDFT}[X(k)]$$

$$= \frac{1}{N} \sum_{k=0}^{N-1} X(k) \cdot W_N^{-nk}; n=0, 1, \dots, N-1$$

Computation of DFT can be performed with N^2 complex multiplication and $N(N-1)$ complex addition.

(3)

$$\mathbf{x}_N = \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix}; N\text{-point vector of signal } x_N$$

$$\mathbf{x}_N = \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix}; N\text{-point vector of signal } x_N$$

$$W_N = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N & W_N^2 & \cdots & W_N^{N-1} \\ 1 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix}; \text{matrix of linear transformation}$$

N -point DFT in matrix form is given by $X_N = W_N X_N$
 $\Rightarrow \mathbf{x}_N = W_N^{-1} X_N$

IDFT in matrix form is given by -

$$\mathbf{x}_N = \frac{1}{N} W_N^* X_N$$

Comparing these expressions of \mathbf{x}_N ,

$$W_N^{-1} = \frac{1}{N} W_N^* \text{ and } W_N \times W_N^* = N[I]_{N \times N}$$

Therefore, W_N is a linear transformation matrix, an orthogonal unitary matrix. From periodic property of w_N and from its symmetric property, it can be concluded that,

$$W_N^{k+N/2} = -W_N^k$$

(4)

* GRAPH PLOTTING ALGORITHM:

The following steps are followed-

- ① Define the x-axis and corresponding y-axis as lists.
- ② Plot them on canvas using plot() function.
- ③ Give a name to x-axis using xlabel() and ylabel() functions.
- ④ Give a title to your plot using the title() function.
- ⑤ Finally, to view your plot, we use the show() function.

* THEORETICAL CALCULATION : (DIRECT METHOD)

Given sequence, $x[n] = [0 \ 1 \ 2 \ 3]$

By keen observation, we find that fundamental period, $N=4$

$$\text{Angular frequency, } \omega_0 = \frac{2\pi}{4}$$

$$x[k] = \sum_{n=0}^{N-1} x[n] e^{-j k \frac{2\pi}{4} n} = \sum_{n=0}^3 x[n] e^{-j k \frac{2\pi}{4} n}$$

$$x[k] = x[0] e^{-j k \frac{2\pi}{4}(0)} + x[1] e^{-j k \frac{2\pi}{4}(1)} \\ + x[2] e^{-j k \frac{2\pi}{4}(2)} + x[3] e^{-j k \frac{2\pi}{4}(3)}$$

$$x[k] = 0(1) + 1(e^{-j k \frac{2\pi}{4}}) + 2(e^{-j k \frac{4\pi}{4}}) + 3(e^{-j k \frac{6\pi}{4}})$$

Substituting the values of k , we get-

$$x[0] = 6 + 0j, \text{ for } k=0$$

$$x[1] = -2 + 2j, \text{ for } k=1$$

$$x[2] = -2 - 2j, \text{ for } k=2$$

$$x[3] = -2 - 2j, \text{ for } k=3$$

Upon computing the magnitude spectrum, we find that -

$$|x[0]| = \sqrt{6^2 + 0^2} = 6.00$$

$$|x[1]| = \sqrt{(-2)^2 + (2)^2} = 2.83$$

$$|x[2]| = \sqrt{(-2)^2 + (0)^2} = 2$$

$$|x[3]| = \sqrt{(-2)^2 + (-2)^2} = 2.83$$

Upon computing the phase spectrum, we find that -

$$(y[n]) = \tan^{-1}\left(\frac{y_1}{y_0}\right) = -^{\circ} = -\times \frac{\pi}{180} \text{ radians}$$

$$(y[0]) = 0^{\circ} = 0 \text{ radians}$$

$$(y[1]) = 135^{\circ} = -0.79 \text{ radians}$$

$$(y[2]) = 180^{\circ} = 3.68 \text{ radians}$$

$$(y[3]) = -135^{\circ} = 0.79 \text{ radians}$$

LINEAR TRANSFORMATION -

The first step is to determine the matrix W_y by exploiting the periodicity property of W_y and the symmetric property -

the matrix W_y may be expressed as -

$$W_y = \begin{bmatrix} W_y^0 & W_y^0 & W_y^0 & W_y^0 \\ W_y^0 & W_y^1 & W_y^1 & W_y^3 \\ W_y^1 & W_y^1 & W_y^2 & W_y^6 \\ W_y^1 & W_y^2 & W_y^4 & W_y^8 \\ W_y^1 & W_y^3 & W_y^6 & W_y^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W_y^1 & W_y^1 & W_y^3 \\ 1 & W_y^2 & W_y^4 & W_y^6 \\ 1 & W_y^3 & W_y^6 & W_y^9 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & 1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

Then,

$$x_y = W_y y_y = \begin{bmatrix} 6 \\ -2+2j \\ -2 \\ -2-2j \end{bmatrix}$$

Upon computing the magnitude spectrum, we find that -

$$|x[0]| = \sqrt{6^2 + 0^2} = 6.00$$

$$|x[1]| = \sqrt{(2)^2 + (2)^2} = 2.83$$

$$|x[2]| = \sqrt{(-2)^2 + (0)^2} = 2$$

$$|x[3]| = \sqrt{(-2)^2 + (-2)^2} = 2.83$$

6

Upon computing the phase spectrum, we find that -
 $(\alpha[n]) = \tan^{-1}\left(\frac{b}{a}\right) = -8^\circ \rightarrow -\frac{\pi}{180} \text{ radians}$

Therefore,

$$\begin{aligned} (\alpha[0]) &= 0^\circ = 0 \text{ radians} \\ (\alpha[1]) &= -135^\circ = -0.79 \text{ radians} \\ (\alpha[2]) &= 180^\circ = 3.14 \text{ radians} \\ (\alpha[3]) &= 135^\circ = 0.79 \text{ radians} \end{aligned}$$

* PROGRAM WITH COMMENTS:

```

1 # Function to calculate x(k) for Direct method
2 def summation
3     sum = 0
4     for n in range(size-n):
5         exponential = np.exp(-2j * np.pi * k + n / size-n)
6         sum = sum + float(a[n]) * exponential
7     return sum
8
9 # import library source files
10 import numpy as np
11 import matplotlib.pyplot as plt
12
13 # Given Input Sequence a[n]
14 size_n = int(input("Enter the size of input a[n]: "))
15 a = [0] * (size_n)
16 print("Enter the elements of the input a[n] one-by-one
as follows: -")
17 for sample in range(0, size_n, 1):
18     a[sample] = input("Element " + str(sample+1) + ": ")
19
20 # sketch Input sequence
21 plt.xlabel('n')
22 plt.ylabel('a[n]')

```

```

①
23 plt.title ("Input sequence  $a[n]$ . format (%))")
24 plt.stem (np.arange (0, size_n), x)
25 plt.grid (True) # Configure the grid lines
26 plt.show ()
27 print ("In The entered input  $n[n]$  is: - " + str(x) + "\n")
28
29 # Compute Discrete Fourier Transform Coefficients Using
30 # Direct Method
31 fourier_transform_direct = []
32 for k in range (size_n):
33     fourier_transform_direct.append (summation (k))
34
35 print ("In The Discrete Fourier Transform (DFT) coefficients of
36 the sequence  $n[n]$  obtained using direct method are as
37 follows: { } ". format (fourier_transform_direct))
38
39 # Compute Magnitude and Phase Spectrum of DFT Coefficients
40 magnitude_spectrum_direct = []
41 phase_spectrum_direct = []
42
43 for sample in range (len (fourier_transform_direct)):
44     magnitude_spectrum_direct.append ((fourier_transform-
45     direct [sample]. real ** 2 + fourier_transform_direct [sample].
46     imag ** 2) ** 0.5)
47
48 phase_direct = np.arctan (fourier_transform_direct
49 [sample]. imag / fourier_transform_direct [sample]. real)
50
51 phase_spectrum_direct.append (phase_direct)
52
53 # Sketch Magnitude Spectrum Using Direct Method
54 plt.xlabel ('k')
55 plt.ylabel ('| $a[k]|$ ')
56 plt.title ("Magnitude spectrum obtained Using Direct
57 method")

```

```

41 plt.stem(np.arange(0, len(magnitude_spectrum_direct)), magnitude_spectrum_direct)
42 plt.grid(True) #Configure the grid lines
43 plt.show()
44 print ("In The magnitude spectrum of DFT coefficients obtained using direct method are as follows : {} ".format(magnitude_spectrum_direct))
45
46 # Sketch Phase spectrum Using Direct method
47 plt.xlabel('k')
48 plt.ylabel('Angle (in radians)')
49 plt.title ('Phase Spectrum obtained using Direct Method')
50 plt.stem(np.arange(0, len(phase_spectrum_direct)), phase_spectrum_direct)
51 plt.show()
52 plt.grid(True) # Configure the grid lines
53 print ("In The phase spectrum of DFT coefficients obtained using direct method are as follows : {} ".format(phase_spectrum_direct))
54
55 # Compute DFT Coefficients Using Linear Transformation Method
56
57 # Compute W(N) 1D Array
58 n1 = c1 = size_n
59 wr = []
60 for i in range(n1):
61     for j in range(c1):
62         wr.append(np.exp(-2j * np.pi * i * j / size_n))
63
64 # numpy.reshape() is used to give a new shape to an array without changing its data.

```

①

```

72 wn_multidim = np.reshape(wn, (n1, c1)) # An N*N w(N) matrix
73 print ("In The matrix w(N) is as follows: \n { } ".format
74 (wn_multidim))
75 x_multidim = np.reshape(x, (n2, c2)) # An N*1 x(N) matrix
76 print ("In The matrix x(N) is as follows: \n { } ".format
77 (x_multidim))

78 # Compute x(N) = w(N) * x(N), an N*1 matrix
79 fourier_transform_multidim = [[0] + c2] * n1
80 # Null Multidimensional Array
81 fourier_transform_l_t = [] # Convert multidimensional
82 array to 1D
83 for i in range (n1):
84     for j in range (c2):
85         fourier_transform_multidim[i][j] = 0
86         for k in range (c1):
87             fourier_transform_multidim[i][j] += 
88             wn_multidim[i][k] * float (x_multidim[k][j])
89             temp = fourier_transform_multidim[i][j]
90             fourier_transform_l_t.append (fourier_transform
91             - multidim[i][j])

92 print ("In The Discrete Fourier Transform (DFT) coefficients
93 of the sequence x[n] obtained using linear transformation
94 are as follows: { } ".format (fourier_transform_l_t))

95 # Compute magnitude and Phase spectrum of DFT Coefficients
96 magnitude_spectrum_l_t = []
97 phase_spectrum_l_t = []
98 for sample in range (len (fourier_transform_l_t)):
99     magnitude_spectrum_l_t.append ((fourier_transform_l_t
100 [sample]).real ** 2 + fourier_transform_l_t [sample].imag ** 2) ** 0.5

```

10

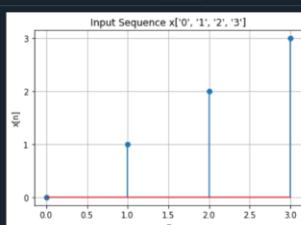
```

96 phase - l - t = np. arctan (fourier - transform - l - t [sample]
97 imag / fourier - transform - l - t [sample] . real)
98 phase - spectrum - l - t . append (phase - l - t)
99 # sketch Magnitude spectrum using Linear Transformation
100 method
101 plt . xlabel ('k')
102 plt . ylabel ('|x[k]|')
103 plt . title ("Magnitude spectrum Obtained Using Linear
104 Transformation Method")
105 plt . grid (True)
106 plt . stem (np . arange (0, len (magnitude - spectrum - l - t)), 
107 magnitude - spectrum - l - t)
108 plt . show ()
109 print ("The magnitude spectrum of DFT coefficients obtained
110 using linear transformation are as follows : {3} . format
111 (magnitude - spectrum - l - t))
112 # sketch Phase spectrum using Linear Transformation Method
113 plt . xlabel ('k')
114 plt . ylabel ('Angle in radians')
115 plt . title ("Phase spectrum Obtained Using Linear
116 Transformation method")
117 plt . stem (np . arange (0, len (phase - spectrum - l - t)), 
118 phase - spectrum - l - t)
119 plt . grid (True) # Configure the grid lines
120 plt . show ()
121 print ("The phase spectrum of DFT coefficients obtained
122 using linear transformation method are as follows : {3} .
123 format (phase - spectrum - l - t))
124
125
126

```

RESULTS

VERIFIED

Given Input Sequence x[n]


Spyder (Python 3.9)

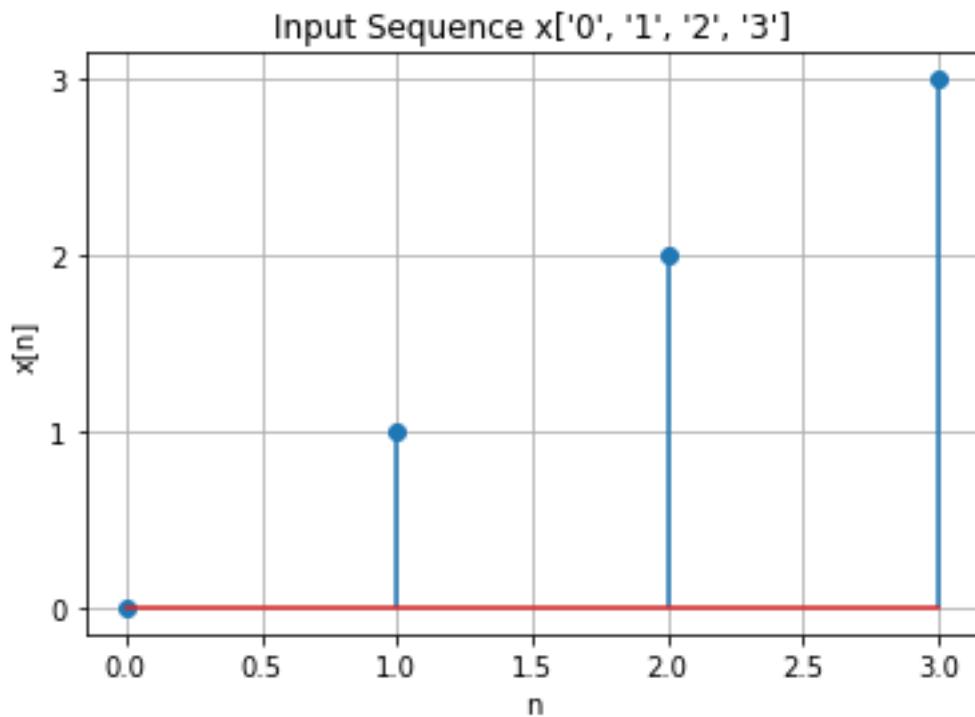
```

1 # Function to calculate X(k) for Direct Method
2 def summation(k):
3     sum = 0
4     for n in range(size_x):
5         exponential = np.exp(-2j * np.pi * k * n / size_x)
6         sum = sum + float(x[n]) * exponential
7     return(sum)
8
9 # Import library source files
10 import numpy as np
11 import matplotlib.pyplot as plt
12
13 # Given Input Sequence x[n]
14 size_x = int(input("Enter the size of input x[n]: "))
15 x = [0] * (size_x)
16 print("Enter the elements of the input x[n] one-by-one as follows: -")
17 for sample in range(0, size_x, 1):
18     x[sample] = input("Element " + str(sample + 1) + ": ")
19
20 # Sketch Input Sequence
21 plt.xlabel('n')
22 plt.ylabel('x[n]')
23 plt.title('Input Sequence x[' + str(x) + ']')
24 plt.stem(np.arange(0, size_x), x)
25 plt.grid(True) # Configure the grid lines
26 plt.show()
27 print("The entered input x[n] is: -" + str(x) + "\n")
28
29 # Compute Discrete Fourier Transform Coefficients Using Direct Method
30 fourier_transform_direct = []
31 for k in range(size_x):
32     fourier_transform_direct.append(summation(k))
33 print("The Discrete Fourier Transform (DFT) coefficients of the sequence x[n] obtained using direct method are as follows: { }".format(fourier_transform_direct))
34
35 # Compute Magnitude and Phase Spectrum of DFT Coefficients
36 magnitude_spectrum_direct = []
37 phase_spectrum_direct = []
38 for sample in range(len(fourier_transform_direct)):
39     magnitude_spectrum_direct.append((fourier_transform_direct[sample].real**2 +
40         fourier_transform_direct[sample].imag**2)**0.5)
41     phase_direct = np.arctan(fourier_transform_direct[sample].imag /
42         fourier_transform_direct[sample].real)
43     phase_spectrum_direct.append(phase_direct)
44
45 print(magnitude_spectrum_direct)
46 print(phase_spectrum_direct)

```

In [1]: runfile('E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 9.py', wdir='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 9')

Enter the size of input x[n]: 4
Enter the elements of the input x[n] one-by-one as follows: -
Element 1: 0
Element 2: 1
Element 3: 2
Element 4: 3
The entered input x[n] is: -[0, 1, 2, 3]



Step 1: Import library source files, NumPy and matplotlib.pyplot.

Step 2: Enter the size of input x[n] and declare an array with the number of elements equal to the value of size.

Step 3: Enter the elements of the input x[n] and show the labelled plot.

Step 4: Compute Discrete Fourier Transform (DFT) coefficients using the direct method and find its magnitude and phase spectrum.

Step 5: Compute W(N) 1D Array followed by X(N) = W(N) * x(N), an N*1 matrix.

Step 6: Repeat step 4 for the linear transformation method.

Compute DFT Coefficients Using Direct Method

The screenshot shows the Spyder Python IDE interface. On the left, the code for Experiment 9.py is displayed. The code performs the following steps:

- Computes the Discrete Fourier Transform (DFT) coefficients using the direct method.
- Sketches the Magnitude Spectrum (|X[k]|) and Phase Spectrum (Angle in radians).
- Computes the W(N) matrix.
- Computes the X(N) matrix.
- Computes the Multidimensional Array.
- Prints the results of each step.

The right side of the interface shows the execution results. It includes a plot titled "Magnitude Spectrum Obtained Using Direct Method" with data points at k=0, 1, 2, 3. It also shows a plot titled "Phase Spectrum Obtained Using Direct Method" with data points at k=0, 1, 2, 3. A console window displays the input sequence x[n] and the calculated DFT coefficients.

```

fouriier_transform_direct(sample).real)
    phase_spectrum_direct.append(phase_direct)

# Sketch Magnitude Spectrum Using Direct Method
plt.xlabel('k')
plt.ylabel('|x[k]|')
plt.title("Magnitude Spectrum Obtained Using Direct Method")
plt.stem(np.arange(0, len(magnitude_spectrum_direct)), magnitude_spectrum_direct)
plt.grid(True) # Configure the grid lines
print("\nThe magnitude spectrum of DFT coefficients obtained using direct method are as follows: {}".
format(magnitude_spectrum_direct))

# Sketch Phase Spectrum Using Direct Method
plt.xlabel('k')
plt.ylabel('Angle (in radians)')
plt.title("Phase Spectrum Obtained Using Direct Method")
plt.stem(np.arange(0, len(phase_spectrum_direct)), phase_spectrum_direct)
plt.grid(True) # Configure the grid lines
print("\nThe phase spectrum of DFT coefficients obtained using direct method are as follows: {}".
format(phase_spectrum_direct))

# Compute DFT Coefficients Using Linear Transformation Method

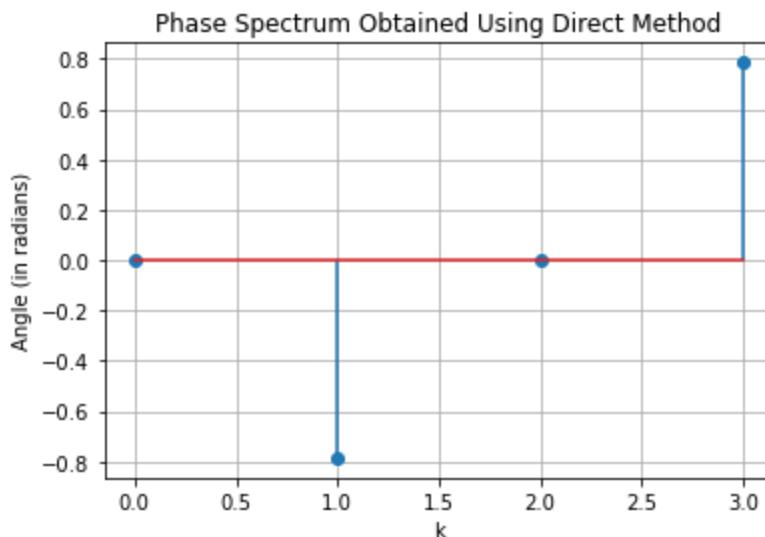
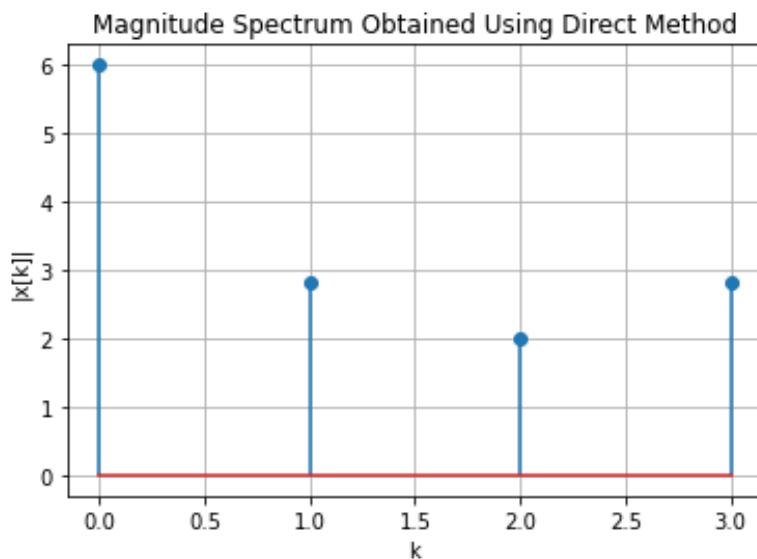
# Compute W(N) 1D Array
r1 = c1 = size_x
wn = []
for i in range(r1):
    for j in range(c1):
        wn.append((np.exp(-2j * np.pi * i * j / size_x)))

# numpy.reshape() is used to give a new shape to an array without changing its data.

wn_multidim = np.reshape(wn, (r1, c1)) # An N*M W(N) matrix
print("The matrix W(N) is as follows:{n}.".format(wn_multidim))
r2 = size_x; c2 = 1
x_multidim = np.reshape(x, (r2, c2)) # An M^1 x (N) matrix
print("The matrix x(N) is as follows:{n}.".format(x_multidim))

# Compute X(N) = W(N) * x(N), an N^1 x (N), an M^1 matrix
fourier_transform_multidim = [[0]*c2]*r1 # Null Multidimensional Array
fourier_transform_1d = [] # Convert Multidimensional Array to 1D
for i in range(r1):
    for j in range(c2):
        .....

```



Compute DFT Coefficients Using Linear Transformation Method

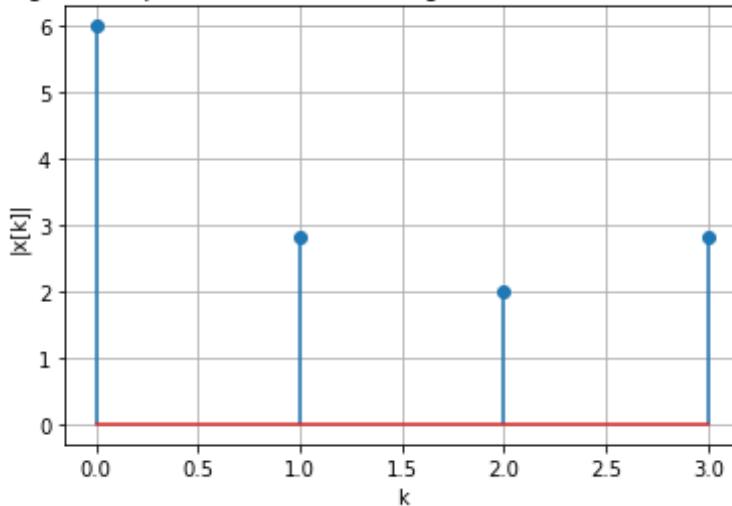
The screenshot shows the Spyder Python IDE interface. On the left, the code editor displays a script named Experiment 9.py containing Python code for calculating DFT coefficients using linear transformation. The code includes imports for numpy and matplotlib, defines a sequence x[n], performs linear transformation, and then calculates the magnitude and phase spectra. On the right, there are two plots: one titled "Phase Spectrum Obtained Using Linear Transformation Method" showing a plot of Angle (in radians) vs k with values at 0, 10, 20, and 30, and another titled "Magnitude Spectrum Obtained Using Linear Transformation Method" showing a plot of |x[k]| vs k with values at 0, 10, 20, and 30. Below the plots is a console window showing the output of the code.

```

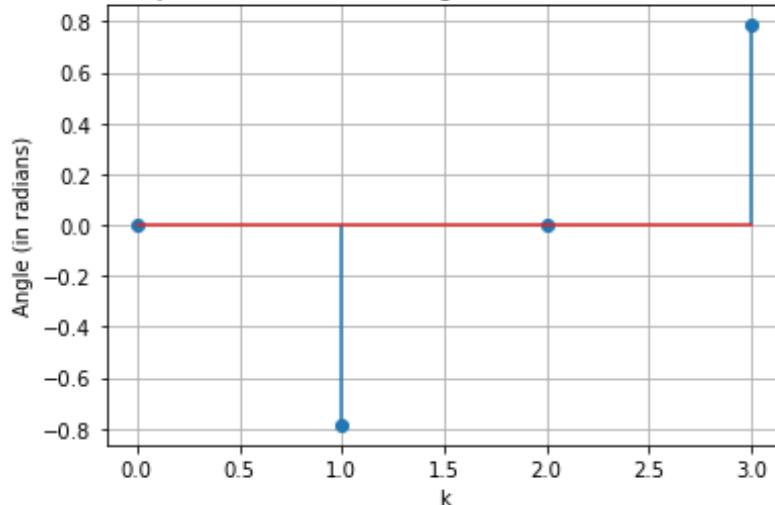
77
78 # Compute X(N) = W(N) * x(N), an N! matrix
79 fourier_transform_multidim = [[0]*c2]*r1 # Null Multidimensional Array
80 fourier_transform_1_t = [] # Convert Multidimensional Array to 1D
81 for i in range(r1):
82     for j in range(c2):
83         fourier_transform_multidim[i][j] = 0
84         for k in range(c1):
85             fourier_transform_multidim[i][j] += w_m_multidim[i][k] * float(x_multidim[k][j])
86             temp = fourier_transform_multidim[i][j]
87             fourier_transform_1_t.append(fourier_transform_multidim[i][j])
88
89 print("\nThe Discrete Fourier Transform (DFT) coefficients of the sequence x[n] obtained using linear transformation method are as follows: {}".format(fourier_transform_1_t))
90
91 # Compute Magnitude and Phase Spectrum of DFT Coefficients
92 magnitude_spectrum_1_t = []
93 phase_spectrum_1_t = []
94 for sample in range(len(fourier_transform_1_t)):
95     magnitude_spectrum_1_t.append((fourier_transform_1_t[sample].real**2 + fourier_transform_1_t[sample].imag**2)**0.5)
96     phase_1_t = np.arctan(fourier_transform_1_t[sample].imag / fourier_transform_1_t[sample].real)
97     phase_spectrum_1_t.append(phase_1_t)
98
99 # Sketch Magnitude Spectrum Using Linear Transformation Method
100 plt.xlabel('k')
101 plt.ylabel('|x[k]|')
102 plt.title("Magnitude Spectrum Obtained Using Linear Transformation Method")
103 plt.stem(np.arange(0, len(magnitude_spectrum_1_t)), magnitude_spectrum_1_t)
104 plt.grid() # Configure the grid lines
105 plt.show()
106 print("\nThe magnitude spectrum of DFT coefficients obtained using linear trasformation method are as follows: {}".format(magnitude_spectrum_1_t))
107
108 # Sketch Phase Spectrum Using Linear Transformation Method
109 plt.xlabel('k')
110 plt.ylabel('Angle (in radians)')
111 plt.title("Phase Spectrum Obtained Using Linear Transformation Method")
112 plt.stem(np.arange(0, len(phase_spectrum_1_t)), phase_spectrum_1_t)
113 plt.grid() # Configure the grid lines
114 plt.show()
115 print("\nThe phase spectrum of DFT coefficients obtained using linear trasformation method are as follows: {}".format(phase_spectrum_1_t))
116

```

Magnitude Spectrum Obtained Using Linear Transformation Method



Phase Spectrum Obtained Using Linear Transformation Method



```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]  
Type "copyright", "credits" or "license" for more information.
```

```
IPython 7.29.0 -- An enhanced Interactive Python.
```

```
Restarting kernel...
```

```
In [1]:      'E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/  
Signal Processing Lab (19CCE281)/Assignments/Experiment 9/Experiment 9.py'      = 'E:/Plan  
B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab  
(19CCE281)/Assignments/Experiment 9'
```

```
Enter the size of input x[n]: 4
```

```
Enter the elements of the input x[n] one-by-one as follows: -
```

```
Element 1: 0
```

```
Element 2: 1
```

```
Element 3: 2
```

```
Element 4: 3
```

```
The entered input x[n] is: -['0', '1', '2', '3']
```

```
The Discrete Fourier Transform (DFT) coefficients of the sequence x[n] obtained using  
direct method are as follows: [(6+0j), (-2.000000000000004+1.999999999999998j),  
(-2-7.34788079488412e-16j), (-1.999999999999984-2.00000000000001j)]
```

```
The magnitude spectrum of DFT coefficients obtained using direct method are as follows:  
[6.0, 2.8284271247461903, 2.0, 2.82842712474619]
```

```
The phase spectrum of DFT coefficients obtained using direct method are as follows: [0.0,  
-0.7853981633974482, 3.67394039744206e-16, 0.785398163397449]
```

```
The matrix W(N) is as follows:
```

```
[[ 1.0000000e+00+0.000000e+00j  1.0000000e+00+0.000000e+00j  
  1.0000000e+00+0.000000e+00j  1.0000000e+00+0.000000e+00j]  
 [ 1.0000000e+00+0.000000e+00j  6.1232340e-17-1.000000e+00j  
  -1.0000000e+00-1.2246468e-16j -1.8369702e-16+1.000000e+00j]  
 [ 1.0000000e+00+0.000000e+00j -1.0000000e+00-1.2246468e-16j  
  1.0000000e+00+2.4492936e-16j -1.0000000e+00-3.6739404e-16j]  
 [ 1.0000000e+00+0.000000e+00j -1.8369702e-16+1.000000e+00j  
  -1.0000000e+00-3.6739404e-16j  5.5109106e-16-1.000000e+00j]]
```

```
The matrix x(N) is as follows:
```

```
[['0']  
 ['1']  
 ['2']  
 ['3']]
```

```
The Discrete Fourier Transform (DFT) coefficients of the sequence x[n] obtained using
```

linear transformation method are as follows: [(6+0j),
(-2.000000000000004+1.99999999999998j), (-2-7.34788079488412e-16j),
(-1.999999999999984-2.00000000000001j)]

The magnitude spectrum of DFT coefficients obtained using linear trasnformation method are as follows: [6.0, 2.8284271247461903, 2.0, 2.82842712474619]

The phase spectrum of DFT coefficients obtained using linear trasnformation method are as follows: [0.0, -0.7853981633974482, 3.67394039744206e-16, 0.785398163397449]

In [2]: