# Rajshahi University of Engineering & Technology

## Department : Electrical & Computer Engineering

**Course No: ECE 4124**

**Course Name: Digital Signal Processing Sessional**

**Submitted By:**

Sumaiya Tabassum

Roll: 1810023

**Submitted To:**

Md. Nahiduzzaman

Lecturer,Dept of ECE

**Experiment No:8**

**Experiment Date:**16.7.23

**Experiment Name:** Computation of DFT using matrix method.

**Theory:**

The Discrete Fourier Transform (DFT) is of paramount importance in all areas of digital signal processing. It is used to derive a frequency-domain (spectral) representation of the signal. It is a fundamental mathematical tool used in various fields, including signal processing, image processing, and communications. The matrix method involves representing the DFT as a matrix multiplication, making it an intuitive approach to understanding the transformation from time-domain to frequency-domain representation. The matrix method provides a straightforward way to compute the DFT by utilizing the DFT matrix, which consists of roots of unity. The DFT matrix is constructed using complex exponential values known as "twiddle factors." The twiddle factor, denoted by $W_N^k$ ,is a complex number used in the construction of the DFT matrix. It is defined as:

$W_N^k = e^{-j*2*\pi*k/N}$ ,where:

N is the size of the input sequence (the number of samples in the sequence).

k is the index for each element of the DFT matrix, ranging from 0 to N-1.

j is the imaginary unit (√(-1)).

π is the mathematical constant pi.

$$W = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix}$$

Fig:DFT Matrix

**Steps for calculating DFT by matrix method:**

Let's assume a sequence of N complex numbers: x[0], x[1], ..., x[N-1]. To compute the DFT using the matrix method, you can follow these steps:

**Step 1:** Defining the DFT matrix; The DFT matrix is an NxN matrix whose elements are the roots of unity. The matrix is given by:

$W_N^k = e^{-j*2*\pi*k/N}$

where N is the length of the input sequence, and k is the row and column index of the matrix (ranging from 0 to N-1).

**Step 2:** Creating the input vector as a column vector X:

X = [x[0] x[1] ... x[N-1]]

**Step 3:** Compute the DFT Multiply the DFT matrix (W) with the input vector (X) to get the DFT result (Y):

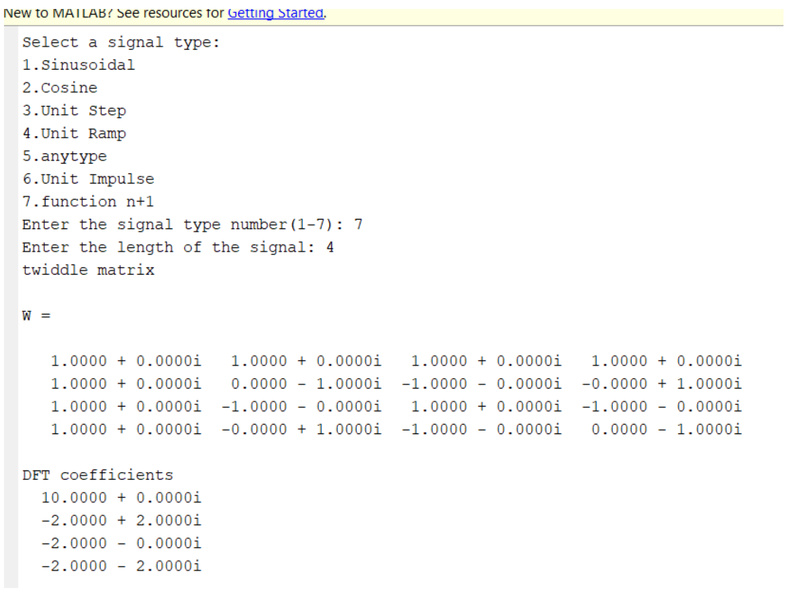
Y = W * X

**Code:**

```
clc;

clear all;
%Signal types
disp('Select a signal type:');
disp('');
disp('1.Sinusoidal');
disp('2.Cosine');
disp('3.Unit Step');
disp('4.Unit Ramp');
disp('5.anytype');
disp('6.Unit Impulse');
disp('7.function n+1');
signalType=input('Enter the signal type number(1-7): ');
N=input('Enter the length of the signal: ');

%Generate the input signal based on the selected type
switch signalType
    case 1 % sin
        t=0:N-1;
        x=sin(pi*t);
    case 2 %cosine
        t=0:N-1;
        x=cos(pi*t)
    case 3 %unit step
        x=ones(1,N);
    case 4 % unit ramp
        x=0:N-1;
    case 5 %  user
        x=input('Enter the input: ');
    case 6 %  impulse
        x= zeros(1,N);
        impulseIndex=input('Enter the impulse of the index:');
        x(impulseIndex)=1;
    case 7 %  function
          t=0:N-1;
          x=t+1;
end
%N=length(x);
%initialization of twiddle matrix
W=zeros(N,N);
```

```
%Generate the twiddle matrix
for k=0:N-1
   for n=0:N-1
       W(k+1,n+1)=exp(-1j*2*pi*k*n/N);
   end
end
disp('twiddle matrix');
W
disp('DFT coefficients');
X=W*x';
disp(X);

%plot result
mag_X=zeros(1,length(X));
phi_X=zeros(1,length(X));
for k=1:length(X)
    mag_X(k)=sqrt((X(k))^2+imag(X(k))^2);
    phi_X(k)=atan2(imag(X(k)),real(X(k)));
end
%plot result
subplot(2,1,1);
stem(mag_X);
xlabel('Frequency');
ylabel('Magnitude');
title('Magnitude spectrum');

subplot(2,1,2);
stem(phi_X);
xlabel('Frequency');
ylabel('phase');
title('Phase spectrum');
```

**Output:**



New to MATLAB? See resources for Getting Started.

```
Select a signal type:
1.Sinusoidal
2.Cosine
3.Unit Step
4.Unit Ramp
5.anytype
6.Unit Impulse
7.function n+1
Enter the signal type number(1-7): 7
Enter the length of the signal: 4
twiddle matrix

W =

   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i
   1.0000 + 0.0000i   0.0000 - 1.0000i  -1.0000 - 0.0000i  -0.0000 + 1.0000i
   1.0000 + 0.0000i  -1.0000 - 0.0000i   1.0000 + 0.0000i  -1.0000 - 0.0000i
   1.0000 + 0.0000i  -0.0000 + 1.0000i  -1.0000 - 0.0000i   0.0000 - 1.0000i

DFT coefficients
  10.0000 + 0.0000i
  -2.0000 + 2.0000i
  -2.0000 - 0.0000i
  -2.0000 - 2.0000i
```

## Plotting :

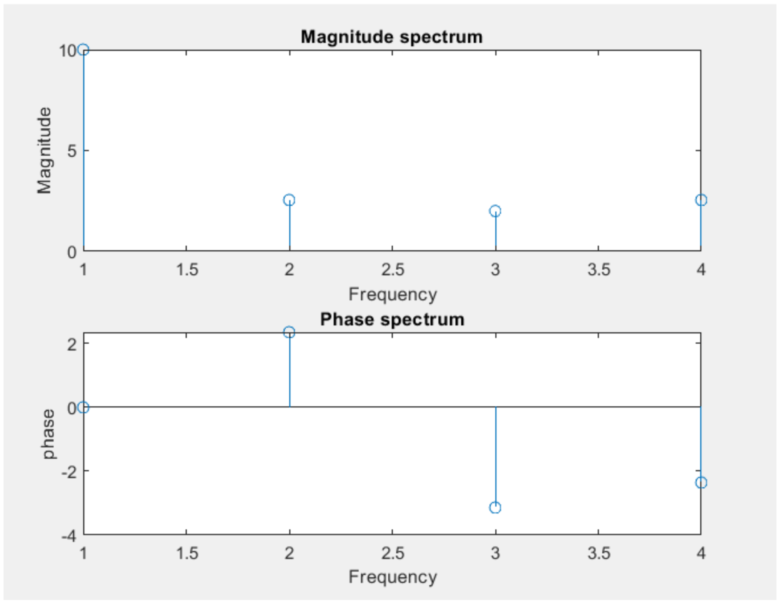


Fig:Magnitude & Phase Spectrum

## Conclusion:

The matrix method which is implemented by MATLAB here provides an intuitive understanding of the DFT, it might not be the most efficient way to compute the DFT in practice. In real-world applications, algorithms like the Fast Fourier Transform (FFT) are commonly used, which take advantage of specific properties of the DFT to significantly reduce the number of computations required. The FFT is much faster than the direct matrix multiplication method for larger values of N.

## Reference:

1.https://www.sciencedirect.com/topics/engineering/discrete-fourier-transform#:~:text=The%20Discrete%20Fourier%20Transform%20(DFT)%20is%20of%20paramount%20importance%20in,spectral)%20representation%20of%20the%20signal.

2.https://www.tutorialspoint.com/digital_signal_processing/dsp_discrete_fourier_transform_introduction.htm