**Registration#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

University of Engineering & Technology Lahore

Experiment # 12

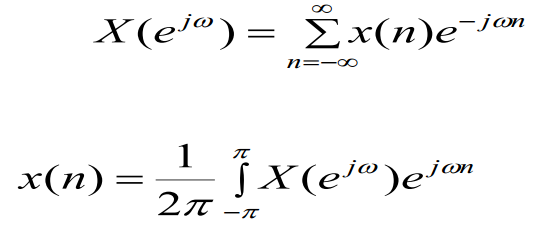
Title: Discrete Fourier Transform (DFT)

Equipment Required: Personal computer (PC) with windows operating system and

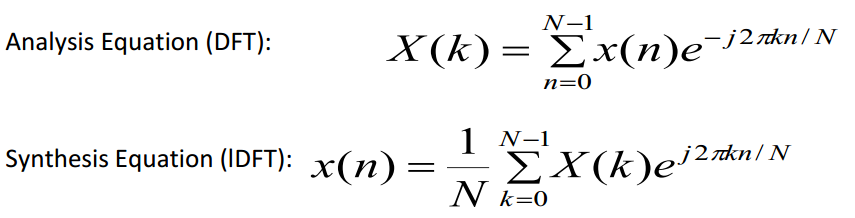
MATLAB software

Theory:

The Discrete Fourier Transform (DFT) is one of the most important tools in Digital Signal Processing. Firstly, the DFT can calculate a signal's *frequency spectrum*. This is a direct examination of information encoded in the frequency, phase, and amplitude of the component sinusoids. For example, human speech and hearing use signals with this type of encoding. Secondly, the DFT can find a system's frequency response from the system's impulse response, and vice versa. This allows systems to be analyzed in the *frequency domain*, just as convolution allows systems to be analyzed in the *time domain*. Third, the DFT can be used as an intermediate step in more elaborate signal processing techniques. The classic example of this is *FFT convolution*, an algorithm for convolving signals that is hundreds of times faster than conventional methods. DTFT is very useful analytically, it usually cannot be exactly evaluated on a computer because equation requires an infinite sum and evaluation of an integral.

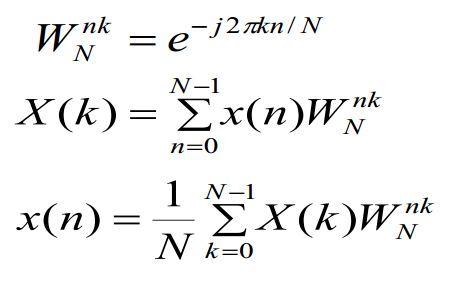


The discrete Fourier transform (DFT) is a sampled version of the DTFT, hence it is better suited to numerical evaluation on computers.



Where X(k) is an N-point DFT of x[n]. Note that X(k) is a function of a discrete integer k, where k ranges from 0 to N-1.

Using the matrix vector multiplication technique used to compute the DTFT, we can calculate the DFT:



**DFT Function**

function [Xk] =dft (xn,N)

%Compute Discrete Fourier Transform n=[0:1:N-1]; k= [0:1:N-1];

WN = exp (-j \* 2 \* pi / N);

nk = n' \* k;

WNnk = WN.^ nk;

Xk = xn \* WNnk;

**Inverse DFT Function**

function [xn] =idft (Xk, N)

%Compute Inverse Discrete Transform

n= [0:1:N-1];

k = [0:1:N-1];

WN = exp (-j \* 2 \* pi / N);

nk = n' \* k;

WNnk = WN .^ (-nk) ;

xn = (Xk \* WNnk)/N;

**Example 1:**

Let x(n)= [1 2 3 4 5]. Evaluate and plot magnitude of DTFT, DFT and IDFT.

**MATLAB Script for DTFT**

*n = -1:3; x = 1:5;*

*k = 0:500; w =(pi/500)\*k;*

*X = x \* (exp(-j\*pi/500)) .^ (n'\*k);*

*magX = abs(X); angX = angle(X);*

*realX = real(X); imagX = imag(X);*

*plot(k/500,magX);grid*

*xlabel('frequency in pi units');*

*title('Magnitude');*

**Simulation**

**MATLAB Script for DFT**

*x=[1,2,3,4,5];*

*N= 5;*

*k= [0:1:N-1];*

*X=dft(x,N);*

*mag= abs(X);*

*magX = abs(X); angX = angle(X);*

*realX = real(X); imagX = imag(X);*

*plot(k,magX);grid*

*xlabel('k'); title('Magnitude');*

**Simulation**

Zero-Padding:

It is an operation in which more zeros are appended to the original sequence. The resulting longer DFT provides closely spaced samples of the discrete time Fourier transform of the original sequence. In MATLAB zero padding is implemented using the zeros function. **The zero padding gives high-density spectrum and provides a better displayed version for plotting.** But it does not give a high-resolution spectrum because no new information is added to the signal: only additional zeros are added in the data.

**Example 2:**

Determine and plot x(n) and its discrete time Fourier transform for appropriate value of n.

1. **Spectrum based on the first 10 samples of x(n)**

*n1=[0:1:9];y1 =x( 1:1:10); subplot(2,1, 1);stem(n1,y1);*

*title('signal x(n), 0 <= n <= 9');xlabel('n')*

*Y1=dft(y1, 10);*

*magY1=abs(Y1( 1:1:6));*

*k1=0:1:5;w1 =2\*pi/10\*k1;*

*subplot(2,1 ,2);stem(w1/pi,magY1 );*

*title('Samples of DTFT Magnitude');*

*xlabel('frequency in pi units')*

*axis([0,1 ,0, 10])*

**Simulation:**

1. **High density Spectrum (50 samples) based on the first 10 samples of x(n)**

*n2=[0:1:49];y2=[x(1: 1:10) zeros(1,40)];*

*subplot(2,1, 1);stem(n2,y2);*

*title('signal x(n), 0 <= n <= 9 + 40 zeros');xlabel('n')*

*Y2=fft(y2);magY2=abs(Y2(1:1:26));*

*k2=0: 1:25;w2=2\*pi/50\*k2; subplot(2,1 ,2);plot(w2/pi,magY2);*

*title('DTFT Magnitude');*

*xlabel('frequency in pi units')*

*axis([0,1,0,10])*

**Simulation:**

1. **High density Spectrum (100 samples) based on the first 10 samples of x(n)**

*n3=[0: 1:99];y3=[ x(1: 1:10) zeros( 1,90)];*

*subplot(2,1, 1);stem(n3,y3);*

*title('signal x(n), 0 <= n <= 9 + 90 zeros');xlabel('n')*

*Y3=dft(y3, 100);magY3=abs(Y3(1: 1:51));*

*k3=0:1:50;w3=2\*pi/100\*k3; subplot(2,1 ,2);plot(w3/pi,magY3);*

*title('DTFT Magnitude');xlabel('frequency in pi units')*

*axis([0, 1,0, 10])*

**Simulation:**

1. High resolution spectrum based on 100 samples of the signal x(n)

*n=[0:1:99];*

*x=cos(0.48\*pi\*n)+cos(0.52\*pi\*n); subplot(2,1,1);stem(n,x);*

*title('signal x(n), 0 <= n <= 99');*

*xlabel('n')*

*axis([0, 100,-2.5,2.5])*

*X=dft(x, 100); magX=abs(X(1: 1:51)); k=0:1:50;w=2\*pi/100\*k;*

*subplot(2,1 ,2);plot(w/pi,magX);*

*title('DTFT Magnitude');xlabel('frequency in pi units')*

*axis([0,1,0,60])*

**Simulation:**

***Write a note on what you have learnt from this example.***

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

**Fast Fourier Transform:**

Discrete Fourier Transform (DFT) is discrete version of FT which transforms a signal (discrete sequence) from Time Domain representation to it's Frequency Domain representation, while Fast Fourier Transform (FFT) is an efficient algorithm for calculation of DFT. Computing a DFT of N points using just it's definition, takes Θ(N^2) time, while an FFT can compute the same result in only Θ(N log N) steps, which's quite substantial gain for large sequences. In MATLAB “fft” command is used for fast Fourier transform.

**Example 3:**

Solve example 2 using Fast Fourier transform.

Properties:

Linearity: The DFT is a linear transform



If x1(n) and x2(n) have N1-points and N2 -points respectively, then N3= max(N1, N2)

**Circular Folding:** If an N point sequence is folded, then the result x(-n) would not be an N point sequence, and it would not be possible to compute its DFT. Therefore, we use modulo-N operation on the argument(-n) and define the folding by



This a called a circular folding. To visualize it, imagine that the sequence x(n) is wrapped around a circle in the counterclockwise direction that n= 0 and n=N overlap. Then x((-n))N can be viewed as a clockwise wrapping of x(n) around the circle. That’s why it is called circular folding.

In MATLAB the circular folding can be achieved by

**Example 4:** Let

1. Determine and plot x((-n))11
2. Verify the circular folding property

**MATLAB Script**

**a.**

n=0:100; x= 10\*(0.8).^n;

y=x(mod(-n,11)+1);

subplot(2,1,1);

stem(n,x); title('original sequence')

xlabel('n'); ylabel('x(n)');

subplot(2,1,2); stem(n,y); title('circularly folded sequence')

xlabel('n'); ylabel('x(-n) mod 10')

**b.**

n=0:10;

x= 10\*(0.8).^n; X= fft(x,11);

Y=fft(x(mod(-n,11)+1), 11);

subplot(2,2,1); stem(n, real(X));

title('Real {DFT of x(n)}'); xlabel('k')

subplot(2,2,2); stem(n, imag(X));

title('Imag {DFT of x(n)}'); xlabel('k')

subplot(2,2,3); stem(n, real(Y));

title('Real {DFT of x(-n)11}'); xlabel('k')

subplot(2,2,4); stem(n, imag(Y));

title('Imag {DFT of x(-n)11}'); xlabel('k')

**Simulation**

**Circular Shift property:** If an N-point sequence is shifted in either direction, then the result is no longer between 0 n N-1. Therefore, we first convert *x(n)* into its periodic extension*,* and then *shift* it by *m* samples to obtain



****This is called periodic shift. then this periodic shift is converted into N point sequence. Resulting sequence is

**Example 5:** Let . Determine and plot x((n-6))15

**MATLAB Script**

function y = cirshftt(x,m,N)

% Circular shift of m samples u r t size N in sequence X: (time domain)

%...................................................................

%[y] = cirshft(x, m, N)

% y = output sequence containing the circular shift

%X = input sequence of length <= N

% m = sample shift

% N = size of circular buffer

% Method: y(n) = x((n-m) mod N

% Check for length of x

if length(x) > N

error('N must be >= the length of X ' )

end

X = [X zeros(1, N-length(x))];

n = [0:1:N-1];

n = mod(n-m,N) ;

y = x(n+l);

n=0:10;

x=10\*(0.8).^n;

y= cirshftt(x, 6,15);

n= 0:14;

x= [x, zeros(1,4)];

subplot(2,1,1); stem(n,x);title('original sequence')

xlabel('n'); ylabel('x(n)');

subplot(2,1,2); stem(n,y); title('circularly shift sequence, N=15')

xlabel('n'); ylabel('x(n-6) mod 15');

**Simulation:**

**Tasks:**

1. Determine the DFT and IDFT of following signal.

1. Let be a 4 point sequence

1. The first five values of the 8-point **DFT** of a real-valued sequence x***(n)*** are given by

Determine the DFT of