BHARATI VIDYAPEETH DEEMED UNIVERSITY
COLLEGE OF ENGINEERING, PUNE - 43

DEPARTMENT OF COMPUTER ENGINEERING

Lab Manual

Digital signal Processing

B.Tech Computer Sem VI

Name of Faculty: Gauri Rao

VISION OF THE DEPARTMENT

To pursue and excel in the endeavor for creating globally recognized computer engineers through quality education.

MISSION OF THE DEPARTMENT

a) To impart engineering knowledge and skills confirming to a dynamic curriculum.
b) To develop professional, entrepreneurial & research competencies encompassing continuous intellectual growth.
c) To produce qualified graduates exhibiting societal and ethical responsibilities in working environment.

Program Educational objectives

The B. Tech Computer Engineering Programme graduates upon completion of this Programme will able to:

a) Demonstrate technical and professional competencies by applying engineering fundamentals, computing principles and technologies.
b) Learn, Practice, and grow as skilled professionals/ entrepreneur/researchers adapting to the evolving computing landscape.
c) Demonstrate professional attitude, ethics, understanding of social context and interpersonal skills leading to a successful career.

Program Outcomes

a) To apply knowledge of computing and mathematics appropriate to the domain.
b) To logically define, analyze and solve real world problems.
c) To apply design principles in developing hardware/software systems of varying complexity that meet the specified needs.
d) To interpret and analyze data for providing solutions to complex engineering problems.
e) To use and practice engineering and IT tools for professional growth.
f) To understand and respond to legal and ethical issues involving the use of technology for societal benefits.
g) To develop societal relevant projects using available resources.
h) To exhibit professional and ethical responsibilities.

i) To work effectively as an individual and a team member within the professional environment.

j) To prepare and present technical documents using effective communication skills.

k) To demonstrate effective leadership skills throughout the project management life cycle.

l) To understand the significance of lifelong learning for professional development.

Course Name: - Digital signal processing
Hours Per Week: 2 Hrs.

Course Outcomes

1. Classify discrete time signals and systems
2. Analyze LTI system in frequency domain using FT and DFT
3. Analyze discrete time signals and LTI system using ZT.
4. Design structures for discrete time systems.
5. Design and Implement FIR and IIR filters.
6. Describe Architecture of DSP processor and applications of DSP

CO-PO mapping

PO →	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	**PSO1**	**PSO2**
CO1	3	3	2	3	2			3	3	3		2	3	2
CO2	3	3	2	3	2			3	3	3		3	3	3
CO3	3	3	2	3	2			3	3	3		3	3	3
CO4	2	2	3	3	2			3	3	3		2	3	3
CO5	3	3	3	3	3			3	3	3		2	3	2
CO6	2	2	3	2				3	3	3		3		

LIST OF PRACTICAL Assignments:

1. WAP to generate samples of Sine, Cosine, Square and Random signal.
2. WAP to compute linear convolution.
3. WAP to compute Circular convolution using Matlab
4. WAP to find N point DFT and IDFT of a given sequence
5. WAP to implement Radix-2 DIT FFT Algorithm.
6. WAP to implement Goertzel Algorithm.
7. WAP to compute Z transform and draw pole zero plot using Matlab.
8. WAP to design Butterworth filter using bilinear transformation method in Matlab.
9. Write a c program for FIR filter design using windows.
10. To implement the difference equation of the LSI system
11. Study of architecture of DSP processor ADSP21XX.
12. Study of applications of DSP in speech and image processing.

Procedure for writing Programs in MATLAB

1. Click on the MATLAB Icon on the desktop.
2. MATLAB window open.
3. Click on the 'FILE' Menu on menu bar.
4. Click on NEW M-File from the file Menu.
5. An editor window open, start typing commands.
6. Now SAVE the file in directory.
7. Then Click on DEBUG from Menu bar and Click Run.

Assignment 1:- WAP to generate samples of Sine, Cosine, Square and Random signal.

Theory:-

Signals are classified into the following categories:

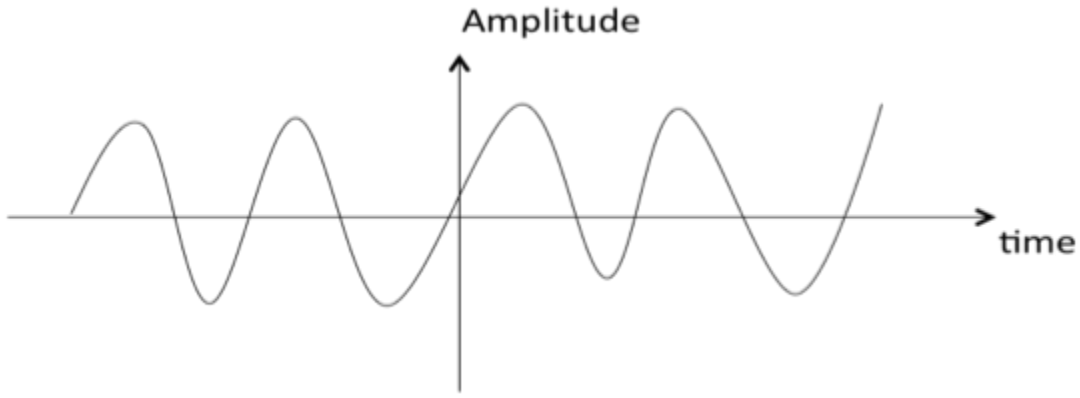
- Continuous Time and Discrete Time Signals
- Deterministic and Non-deterministic Signals
- Even and Odd Signals
- Periodic and Aperiodic Signals
- Energy and Power Signals
- Real and Imaginary Signals

Continuous Time and Discrete Time Signals

A signal is said to be continuous when it is defined for all instants of time.

Continuous signal

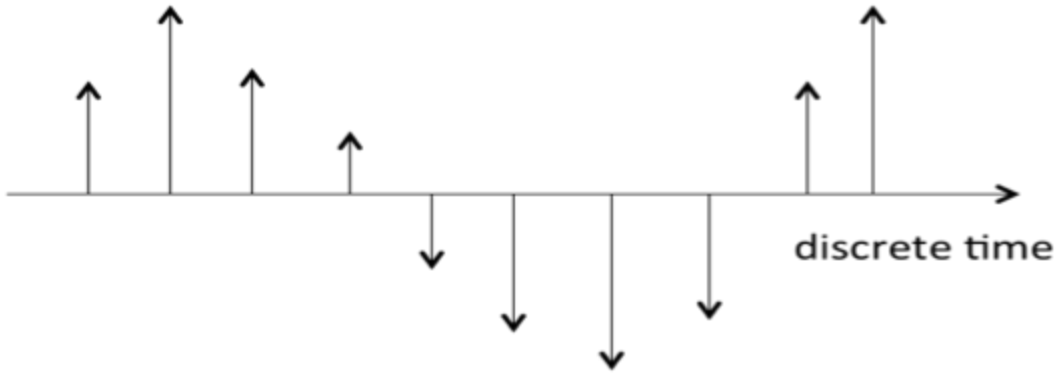
A signal is said to be discrete when it is defined at only discrete instants of time/



Discrete signal

Deterministic and Non-deterministic Signals

A signal is said to be deterministic if there is no uncertainty with respect to its value at any instant of time. Or, signals which can be defined exactly by a mathematical formula are known as deterministic signals.

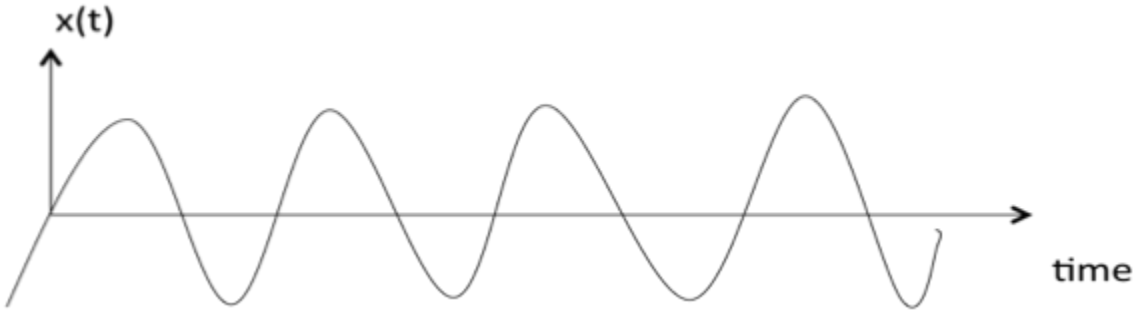


Deterministic and Non-deterministic Signals

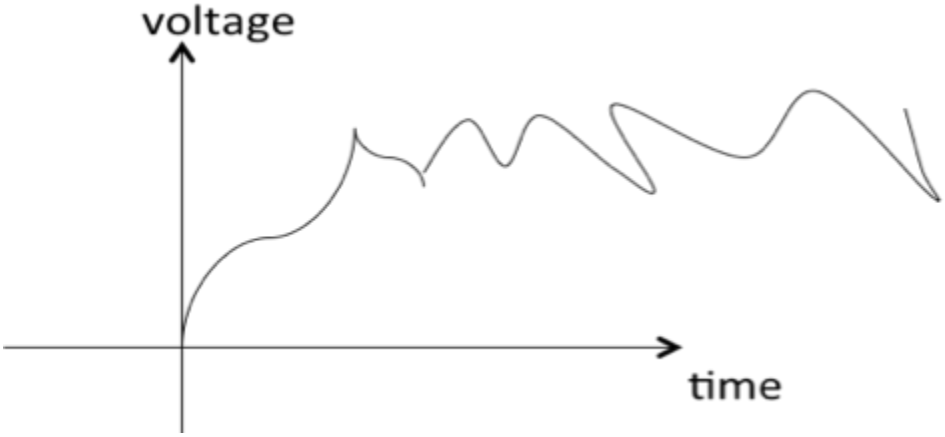
A signal is said to be deterministic if there is no uncertainty with respect to its value at any instant of time. Or, signals which can be defined exactly by a mathematical formula are known as deterministic signals.

A signal is said to be non-deterministic if there is uncertainty with respect to its value at some instant of time.

Non-deterministic signals are random in nature hence they are called random signals. Random signals cannot be described by a mathematical equation. They are modelled in probabilistic terms.

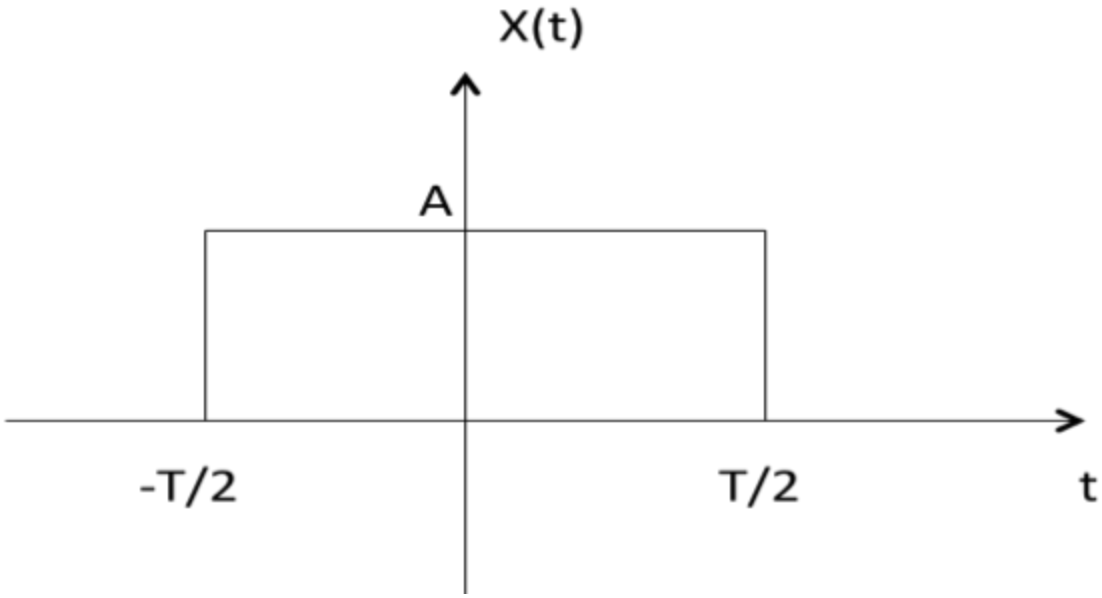


Non-deterministic signal



Even and Odd Signals

A signal is said to be even when it satisfies the condition x(t) = x(-t)

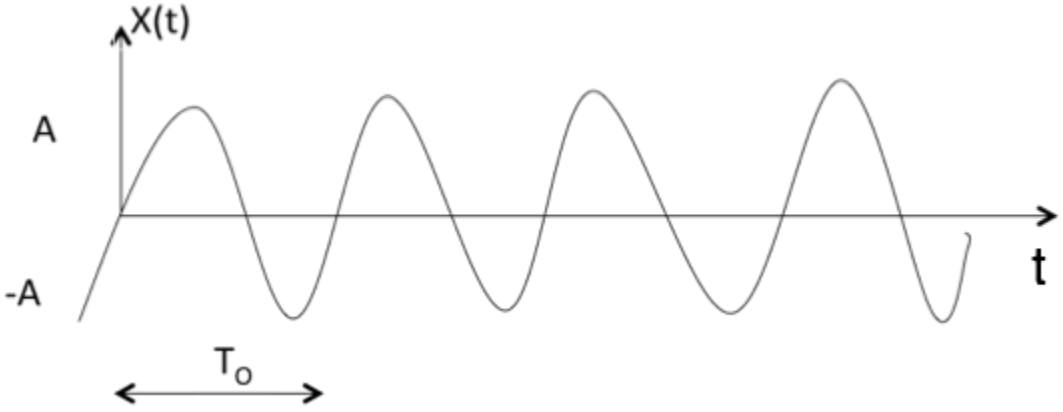


Periodic and Aperiodic Signals
A signal is said to be periodic if it satisfies the condition x(t) = x(t + T) or x(n) = x(n + N).

Where

T = fundamental time period,

1/T = f = fundamental frequency.



Periodic and aperiodic signals
The above signal will repeat for every time interval T0 hence it is periodic with period T0.

Energy and Power Signals

A signal is said to be energy signal when it has finite energy.

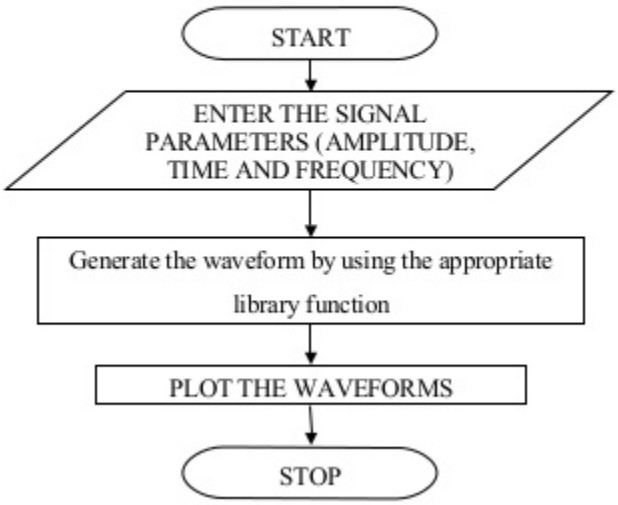
Power of energy signal = 0

Energy of power signal = ∞

Real and Imaginary Signals
A signal is said to be real when it satisfies the condition x(t) = x*(t)

A signal is said to be odd when it satisfies the condition x(t) = -x*(t)

FLOWCHART:



Program to generate samples of some standard of some standard Signals at specified sampling frequencies.

A) Sine wave
B) Square wave
C) Exponential and
D) Random signal

INPUTS:

1) Frequencies of sine and square waves
2) Sampling frequency
3) Choice for the signal to be generated.

OUTPUTS:

1) The samples of the discrete time signal are stored in the array.

2) The discrete time signal is displayed.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
#include<math.h>
void main()
{
 float x[700],A,F,Fs,n,Y;
 int i,gd=DETECT,gm,X,ch,k;
 char ans;
 clrscr();
 initgraph(&gd,&gm,"E:\\tc\\bgi");
 i=640;
 A=0.5;
 printf("\n\t GENERATION OF DISCRETE TIME SIGNAL ");
 do
 {
  printf("\n\t ENTER THE FREQUENCY OF ANALOG SIGNAL:");
  scanf("%f",&F);
  printf("\n\t ENTER THE SAMPLING FREQUENCY: ");
  scanf("%f",&Fs);

  printf("\n\t YOU HAVE FOLLOWING OPTIONS:");
  printf("\n\t 1)SINE WAVE ");
  printf("\n\t 2)SQUARE WAVE ");
  printf("\n\t 3)EXPONENTIAL SIGNAL ");
  printf("\n\t 4)RANDOM NOISE  ");
  printf("\n\t 5)EXIT");
  printf("\n\n\t ENTER YOUR CHOICE: ");
  scanf("%d",&ch);
  switch(ch)
  {
   case 1:
   cleardevice();
```

```
for(n=0;n<i;n++)
{
 x[n]=A*sin(2*M_PI*F*(n/Fs));
}
break;

case 2:
cleardevice();

k=0;
do
{
 for(n=(k*Fs)/(2*F);n<((k+1)*Fs)/(2*F);n++)
 {
  x[n]=A;
 }
 for(n=((k+1)*Fs)/(2*F);n<((k+2)*Fs)/(2*F);n++)
 {
  x[n]=-A;
 }
 k+=2;
}while(n<i);
break;

case 3:
cleardevice();

for(n=0;n<i;n++)
x[n]=A*exp(-(n/Fs));
break;

case 4:
cleardevice();

for(n=0;n<i;n++)
x[n]=(float)(rand()%1000)/1000.00;

break;
case 5:
exit(0);
```

```
   break;
  }
  Y=X=0;

  line(1,50,1,350);
  line(1,200,550,200);

  for(n=0;n<i;n++)
  {
   Y=200-x[n]*100;
   putpixel(X,Y,WHITE);
   delay(40);
   X++;
  }
  getch();
  clrscr();
  cleardevice();
  printf("\n\n\t DO YOU WANT TO CONTINUE ? ");
  scanf("%s",&ans);
 }while(ans=='y' || ans=='Y');
 closegraph();
}
```

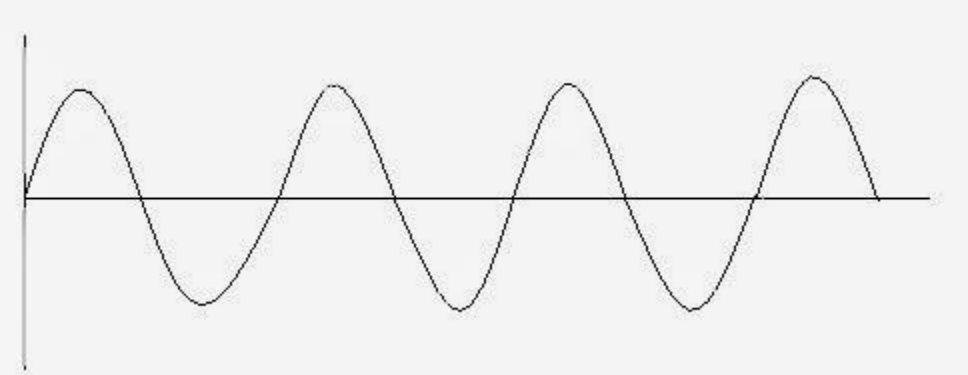
Output

GENERATION OF DISCRETE TIME SIGNAL:

ENTER THE FREQUENCY OF ANLOG SIGNAL=10
ENTER THE SAMPLING FREQUENCY=1000

YOU HAVE FOLLWING OPTIONS:
1) SINE WAVE
2) SQUARE WAVE
3) EXPONENTIAL SIGNAL
4) RANDOM NOISE
5) EXIT

ENTER YOUR CHOICE: 1



DO YOU WANT TO CONTINUE? : Y

ENTER THE FREQUENCY OF ANLOG SIGNAL=10
ENTER THE SAMPLING FREQUENCY=1000

YOU HAVE FOLLWING OPTIONS:
1) SINE WAVE
2) SQUARE WAVE
3) EXPONENTIAL SIGNAL
4) RANDOM NOISE

5) EXIT

ENTER YOUR CHOICE: 2

DO YOU WANT TO CONTINUE? : Y

ENTER THE FREQUENCY OF ANLOG SIGNAL=10
ENTER THE SAMPLING FREQUENCY=1000

YOU HAVE FOLLWING OPTIONS:
1) SINE WAVE
2) SQUARE WAVE
3) EXPONENTIAL SIGNAL
4) RANDOM NOISE
5) EXIT

ENTER YOUR CHOICE: 3

DO YOU WANT TO CONTINUE? : Y

ENTER THE FREQUENCY OF ANLOG SIGNAL=10
ENTER THE SAMPLING FREQUENCY=1000

YOU HAVE FOLLWING OPTIONS:
1) SINE WAVE
2) SQUARE WAVE
3) EXPONENTIAL SIGNAL
4) RANDOM NOISE
5) EXIT

ENTER YOUR CHOICE: 4

DO YOU WANT TO CONTINUE? : N

Questions:-

1. Define a signal and classify the types of signals.
2. State the characteristics of discrete time signal.
3. Define a system and classify different types of systems.
4. List and represent the standard signals
5. Differentiate between
    i. Analog and digital signals
    ii. Energy and power signals
6. State sampling theorem.
7. Describe the process of analog to digital conversion.
8. Define aliasing effect. How it can be eliminated.
9. Define Sampling frequency and Nyquist rate
10. Describe the process of sampling and quantization.

Assignment 2:- WAP to compute linear convolution of given sequences.

THEORY:

Convolution is a formal mathematical operation, just as multiplication, addition, and integration. Addition takes two numbers and produces a third number, while convolution takes two signals and produces a third signal. Convolution is used in the mathematics of many fields, such as probability and statistics. In linear systems, convolution is used to describe the relationship between three signals of interest: the input signal, the impulse response, and the output signal.

In this equation, x1(k), x2(n-k) and y(n) represent the input to and output from the system at time n. Here we could see that one of the input is shifted in time by a value every time it is multiplied with the other input signal. Linear Convolution is quite often used as a method of implementing filters of various types.

Algorithm:

1. Enter the value for the sequence x and h.
2. Compute the linear convolution using the formula

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

3. Plot the sequence

* Linear Convolution of two sequences

This program computes the linear Convolution of two sequences x(n) and h(n)

INPUTS:

1) Number of samples in x(n) (casual sequence)

2) Samples of x(n) in the form x[0],x[1],x[2],...x[n-1]

3) Number of samples in h(n) (casual sequence)

4) Samples of h(n) in the form h[0],h[1],h[2],...h[n-1]

OUTPUT: Convolution sequence of x (n) and h (n) */

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
 int n,k,N,M;
 float h[20],x[20],y[20],Total;
 char ch;
 clrscr();
 do
 {
  printf("\n\t\t Linear Convolution ");
  printf("\n\n Enter the number of samples in h(n)=");
  scanf("%d",&N);
  printf("\n Enter the sequence h(n)=");
  for(n=0;n<N;n++)
  {
   printf("\n\n h[%d]=",n);
   scanf("%f",&h[n]);
  }
  printf("\n\n Enter the number of samples in x(n)=");
  scanf("%d",&M);
  printf("\n Enter the sequence x(n)=");
  for(n=0;n<M;n++)
```

```
 {
  printf("\n\n x[%d]=",n);
  scanf("%f",&x[n]);
 }
 printf("\n The result of Convolution (y(n)=x(n)*h(n))is:");
 for(n=0;n<(N+M-1);n++)
 {
  Total=0.0;
  for(k=0;k<M;k++)
  {
   if(n<k || (n-k)>=N)
   continue;
   Total+=x[k]*h[n-k];
  }
  y[n]=Total;
  printf("\n y[%d]=%f",n,y[n]);
 }
 printf("\n DO YOU WANT TO CONTINUE?:(Y/N) ");
 ch=getche();
}while(ch=='y'||ch=='Y') ;
getch();
}
*****************************
```

Output

Linear Convolution

Enter the number of samples in h(n)=3

Enter the sequence h(n)=

h[0]=1

h[1]=2

h[2]=3

Enter the number of samples in x(n)=3

Enter the sequence x(n)=

x[0]=3

x[1]=2

x[2]=1

The result of Convolution (y(n)=x(n)*h(n))is:

y[0]=3.000000

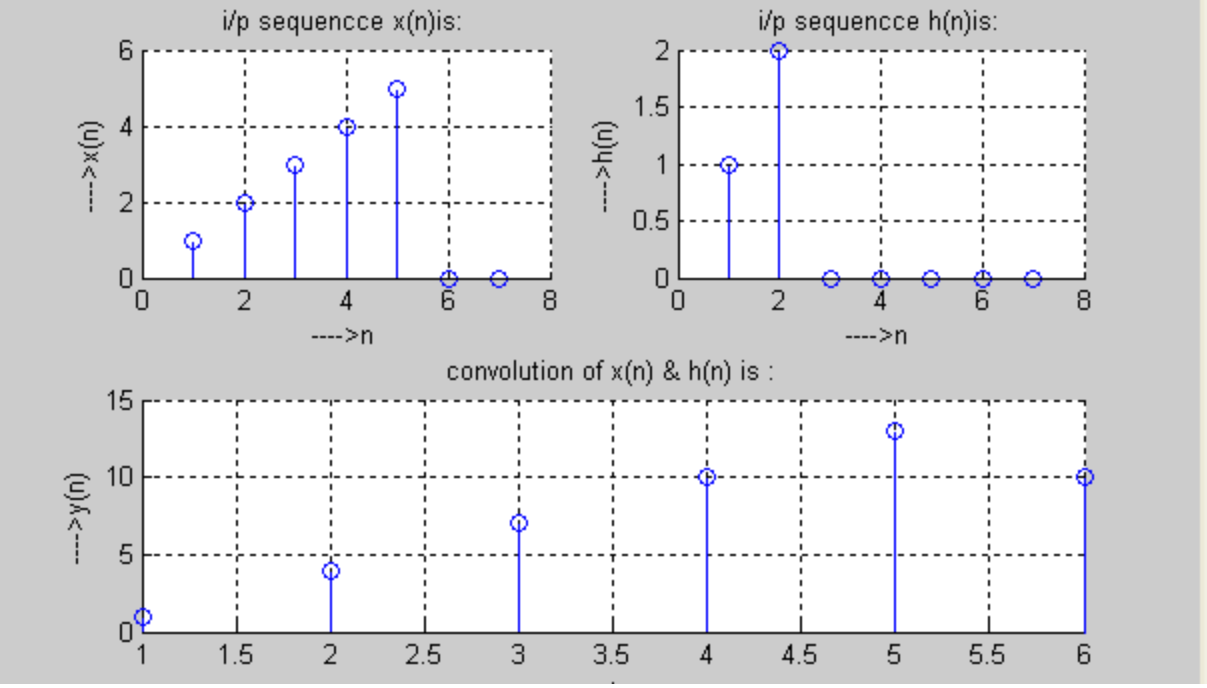
y[1]=8.000000

y[2]=14.000000

y[3]=8.000000

y[4]=3.000000

DO YOU WANT TO CONTINUE?:(Y/N)N



Questions:-

1. Define linear convolution.
2. State the properties of linear convolution
3. What is the difference between folding and shifting operations.
4. Write formula for linear convolution and list the basic operations carried out during linear convolution
5. How convolution is used in image processing?
6. Determine linear convolution between x(n) = {1,2,3,4}and h(n)= {1,1,1}using graphical method.

Assignment 3 WAP to compute Circular Convolution.

THEORY

Circular convolution is another way of finding the convolution sum of two input signals. It resembles the linear convolution, except that the sample values of one of the input signals is folded and right shifted before the convolution sum is found. Also note that circular convolution could also be found by taking the DFT of the two input signals and finding the product of the two frequency domain signals. The Inverse DFT of the product would give the output of the signal in the time domain which is the circular convolution output. The two input signals could have been of varying sample lengths.

Let x1(n)x1(n) and x2(n)x2(n) be two given sequences. The steps followed for circular convolution of x1(n)x1(n) and x2(n)x2(n) are

- Take two concentric circles. Plot N samples of x1(n)x1(n) on the circumference of the outer circle (maintaining equal distance successive points) in anti-clockwise direction.
- For plotting x2(n)x2(n), plot N samples of x2(n)x2(n) in clockwise direction on the inner circle, starting sample placed at the same point as 0^{th}sample of x1(n)x1(n)
- Multiply corresponding samples on the two circles and add them to get output.
- Rotate the inner circle anti-clockwise with one sample at a time.

Comparison points	Linear Convolution	Circular Convolution
Shifting	Linear shifting	Circular shifting
Samples in the convolution result	$N_1 + N_2 - 1$	$Max(N_1, N_2)$
Finding response of a filter	Possible	Possible with zero padding

Algorithm:

- Enter the value for the sequence x and h.

Compute the circular convolution using the formula

$$y[n] = \sum_{k=0}^{N-1} x[k]h[(n-k)_N], where\ n = 0,1,\ldots N-1$$

. Plot the sequence

/* Program for CIRCULAR CONVOLUTION of two sequences h(n) and x(n).

Inputs:
1) Length of two sequences N.
2) Samples of two sequences.

Output: Circular Convolution sequence of h(n) and x(n).

Assumptions: The given sequence are real and causal.

```
/*=====================================================================*/

#include<stdio.h>
#include<conio.h>
#include<math.h>

void main()
{
 int N,M,n,k,m;
 float total,h[20],x[20],y[20];

 clrscr();
 printf("\n\t\t\t Circular Convolution\n\n\n");

 printf("\n Enter the value of N=");
 scanf("%d",&N);

 printf("\n\n Enter the sequence h(n):");
 for(n=0;n<N;n++)
 {
  printf("\n\n h[%d]=",n);
  scanf("%f",&h[n]);
 }

 printf("\n\n Enter the sequence x(n):");
 for(n=0;n<N;n++)
 {
  printf("\n\n x[%d]=",n);
  scanf("%f",&x[n]);
 }

 printf("\n\n\n Circular convolution is:");
 for(m=0;m<N;m++)
 {
  total=0.0;
  for(k=0;k<N;k++)
```

```
 {
  if((m-k)>=0)
   n=m-k;
  else
   n=m-k+N;
  total=total+x[k]*h[n];
 }
 y[m]=total;
 printf("\n\n y[%d]=%f",m,y[m]);
}
getch();
}
```

Circular Convolution

```
Enter the value of N=4
Enter the sequence h(n):
h[0]=2
h[1]=1
h[2]=2
h[3]=1


Enter the sequence x(n):
x[0]=1

x[1]=2

x[2]=3

x[3]=4

Circular convolution is:

y[0]=14.000000

y[1]=16.000000

y[2]=14.000000

y[3]=16.000000
```

CIRCULAR CONVOLUTION USING DFT AND IDFT

```
/*--------------------CIERCULAR CONVOLUTION USING DFT AND IDFT----------------
 This program computes the circular convolution of two causal sequences x(n)
 and h(n) using DFT and IDFT.

 Inputs: 1.Length of the sequence i.e.N
  2. Samples of the two sequences to be convolved

 Outputs:  Circular convolution sequence of x(n) and h(n)

 Assumptions: The given sequences are real and causal.
================================================================================*/
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
 float h[20],x[20],y[20];
 float h_real[20],h_imag[20],x_real[20],x_imag[20];
 float N,M,n,k,m,y_real[20],y_imag[20];

 clrscr();
 printf("\t\tCircular Convolution using DFT and IDFT\n\n");

 printf("\n Enter the value of N=");
 scanf("%f",&N);

 printf("\n Enter the sequence h(n):\n");
 for(n=0;n<N;n++)
 {
  printf("\n h[%1.0f]= ",n);
  scanf("%f",&h[n]);
 }
 printf("\n\n Enter the Sequence x(n):\n");
 for(n=0;n<N;n++)
 {
  printf("\n x[%1.0f]= ",n);
  scanf("%f",&x[n]);
 }
 for(k=0;k<N;k++)
 {
  h_real[k]=h_imag[k]=0.0;
  for(n=0;n<N;n++)
  {
   h_real[k]=h_real[k]+h[n]*cos((2*M_PI*k*n)/N);
```

```
  h_imag[k]=h_imag[k]+h[n]*sin((2*M_PI*k*n)/N);
 }
 h_imag[k]=h_imag[k]*(-1.0);
}
for(k=0;k<N;k++)
{
 x_real[k]=x_imag[k]=0.0;
 for(n=0;n<N;n++)
 {
  x_real[k]=x_real[k]+x[n]*cos((2*M_PI*k*n)/N);
  x_imag[k]=x_imag[k]+x[n]*sin((2*M_PI*k*n)/N);
 }
 x_imag[k]=x_imag[k]*(-1.0);
}

for(k=0;k<N;k++)
{
 y_real[k]=h_real[k]*x_real[k]-h_imag[k]*x_imag[k];
 y_imag[k]=h_real[k]*x_imag[k]+h_imag[k]*x_real[k];
}
for(n=0;n<N;n++)
{
 y[n]=0;
 for(k=0;k<N;k++)
 {
  y[n]=y[n]+y_real[k]*cos((2*M_PI*k*n)/N)
  -y_imag[k]*sin((2*M_PI*k*n)/N);
 }
 y[n]=y[n]/N;
}
printf("\n The Circular of convolution is.....");
for(n=0;n<N;n++)
 printf("\n\n y[%1.0f]= %f",n,y[n]);
getch();
}
***op***
        Circular Convolution using DFT and IDFT
Enter the value of N=4
Enter the sequence h(n):
h[0]= 0
h[1]= 1
h[2]= 2
h[3]= 3
Enter the Sequence x(n):
```

x[0]= 2
x[1]= 1
x[2]= 1
x[3]= 2

The Circular of convolution is.....

y[0]= 7.000000 y[1]= 9.000000 y[2]= 11.000000 y[3]= 9.000000

Assignment 4 WAP to find N-point DFT and IDFT of given sequence

---

$$\text{DFT: } X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \qquad k = 0, 1, \ldots, N-1$$

$$\text{IDFT: } x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-kn} \qquad n = 0, 1, \ldots, N-1$$

where W_N is defined as

$$W_N = e^{-j2\pi/N}$$

```
/* C Program to compute Discrete Fourier Transform (DFT)
   of a sequence X(n) of Length N

   Inputs: 1) Number of samples of x (n) i.e. N
     2) Values of samples of x (n)
   Outputs: N point DFT X (k) with its real and imaginary parts.

   Assumptions: The sequence x (n) is considered real.
==============================================================*/
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<graphics.h>

void main()
{
 float static X[100],X_Real[100],X_Imag[100];
 float k,n,N;
 clrscr();
```

```
printf("\t\t\t Discrete Fourier Transform(DFT)");
printf("\n\n Enter the number samples in the sequence X(n),N=");
scanf("%f",&N);
printf("\n\n Enter the samples of sequence X(n):");

for(n=0;n<N;n++)
{
 printf("X(%d)=",n);
 scanf("%f",&X[n]);
}

for(k=0;k<N;k++)
{
 X_Real[k]=X_Imag[k]=0.0;

 for(n=0;n<N;k++)
 {
  X_Real[k]=X_Real[k]+X[n]*cos((2*M_PI*k*n)/N);
  X_Imag[k]=X_Imag[k]+X[n]*sin((2*M_PI*k*n)/N);
 }
 X_Imag[k]=X_Imag[k]*(-1.0);
}
printf("\n The %d  point DFT of given sequence is..",N);
printf("\n\t Real X(k)\t\t Imaginary X(k)");
for(k=0;k<N;k++)
 printf("\n X(%d)=%f\t\t %f",k,X_Real[k],X_Imag[k]);
getch();
}
```

```
                    Discrete Fourier Transform(DFT)

Enter the number samples in the sequense X(n)=4

Enter the samples of sequence X(n):

X(0)=1

X(1)=2

X(2)=0

X(3)=1
```

The 4 point DFT of given sequence is:

Real X[k]	Imaginary X[k]
X(0)=4.000000	-0.000000
X(1)=1.000000	-1.000000
X(2)=-2.000000	-0.000000
X(3)=1.000000	1.000000

To compute the IDFT of N point X(k).

```
 /*-------------INVERSE DISCRETE FOURIER TRANSFORM(IDFT)----------------
  This program computes the IDFT of N point X(k).

  Inputs:
   1) Length of DFT i.e. N.
   2) Real and imaginary parts of DFT X(k).
  Outputs: N point sequence i.e. x(n).

  Assumptions: The sequence x(n) is considered real.
 ========================================================================*/

#include<stdio.h>
#include<conio.h>
#include<math.h>

void main()
{
 float static X[100],X_Real[100],X_Imag[100];
 float k,n,N;
 clrscr();

 printf("\t\t\t Inverse Discrete Fourier Transform(IDFT)");
 printf("\n\n Enter the length of DFT N=");
 scanf("%f",&N);

 printf("\n Enter the real and imaginary parts of X(k) as follows:\n\n"
 "X(k) =Real{X(k)} Img{X(k)} \n" );

 for(k=0;k<N;k++)
 {
  printf("X(%1.0f)=",k);
  scanf("%f%f",&X_Real[k],&X_Imag[k]);
 }

 for(n=0;n<N;n++)
 {
  X[n]=0;
  for(k=0;k<N;k++)
  {
   X[n]=X[n]+X_Real[k]*cos((2*M_PI*k*n)/N)-X_Imag[k]*sin((2*M_PI*k*n)/N);
  }
  X[n]=X[n]/N;
 }
```

```
 printf("\n\n The sequence x(n) is as follows...");
 for(n=0;n<N;n++)
 {
  printf("\n\n X(%1.0f)=%3.6f",n,X[n]);
 }

 getch();
}
```

Output

Inverse Discrete Fourier Transform(IDFT)

Enter the length of DFT N=4

Enter the real and imaginary parts of X(k) as follows:

X(k) =Real{X(k)} Img{X(k)}

X(0) = 4 0

X(1) = 1 -1

X(2) = -2 0

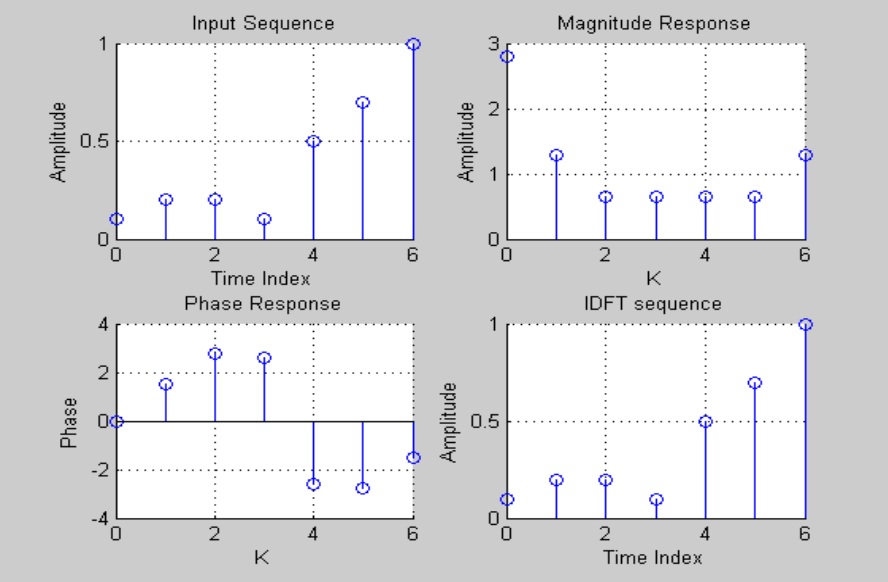
X(3) = 1 1

The sequence x(n) is as follows...

X(0)=1.000000

X(1)=2.000000

X(2)=-0.000000



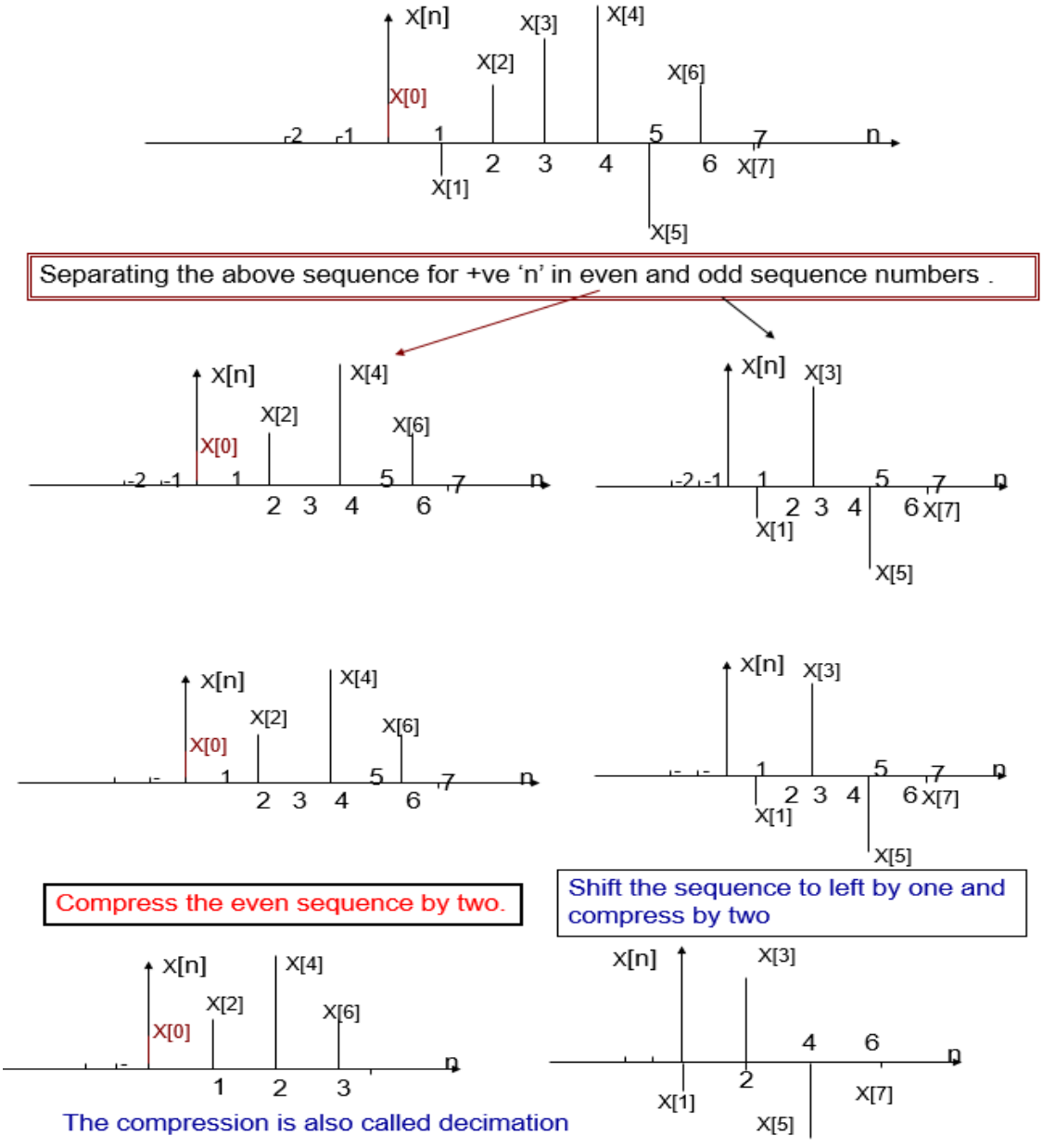
Questions:-

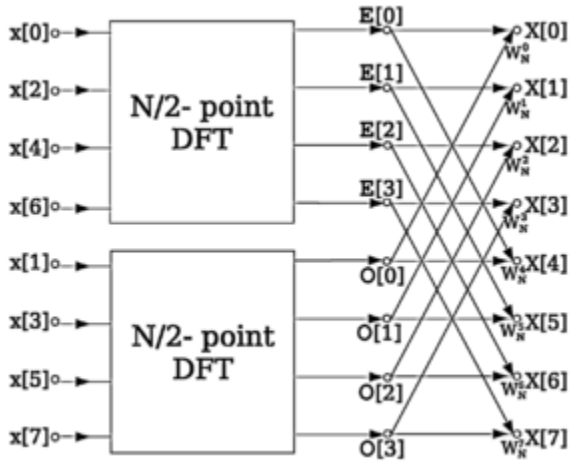
1. What is the difference between linear and circular convolution?
2. State convolution property of DFT.
3. Can we obtain same result from circular convolution as that of linear convolution
4. Define correlation.
5. Determine circular convolution between x(n) = {1,2,1,2}and h(n)= {1,2,1}using tabular method.

Assignment 5 : To implement Radix-2 DIT FFT algorithm

Radix-2 is the first FFT algorithm. It was proposed by Cooley and Tukey in 1965.This FFT algorithm converts a time domain signal to DFT efficiently.First step of process of decimation is splitting a sequence in smaller sequences.A sequence of 16 numbers can be splitted in 2 sequences of 8. Further, each sequence of 8 can be be splitted in two sequences of 4;Subsequently each sequence of 4 can be splitted in two sequences of two.There can be various combinations and varied complexities.

The process of Decimation is as shown in this example:






```
/*  C program to compute N-point Radix-2 DIT FFT algorithm.

 Inputs:
  1.Number of samples of x(n)
  2.Values of samples of x(n)

 Output: Real and imaginary part of DFT X(k).
==================================================================*/

#include<stdio.h>
#include<conio.h>
#include<math.h>
#define SWAP(a,b)var=(a);(a)=(b);(b)=var;

void main()
{
 int N,n,m,j,k,i,p;
 float data[200],real1,imag1,real2,imag2,var;
 float costheta,sintheta,t,Theta;

 clrscr();
 printf("\n\t\t Readix-2 DIT FFT algorithm\n\n");

 printf("\n\n Enter the number of samples in the sequence x(n),N=");
 scanf("%d",&N);

 printf("\n\n Enter the Samples of the Sequence x(n):\n");
 printf("\n Real part  Imaginary part");
```

```
for(n=1;n<=N;n++)
{
  printf("\n x(%d)=",n-1);
  scanf("%f%f",&data[2*n-1],&data[2*n]);
}
n=N<<1;
j=1;

for(i=1;i<n;i=i+2)
{
 if(j>i)
 {
  SWAP(data[j],data[i]);
  SWAP(data[j+1],data[i+1]);
 }
 m=n>>1;
 while(m>=2 && j>m)
 {
  j-=m;
  m>>=1;
 }
 j+=m;
}

k=1;m=1;t=0.0;
while((N/(2*k))>=1)
{
 p=pow(2,m);
 n=1;
 Theta=((2*M_PI)/(float)p)*t;
 costheta=cos(Theta);
 sintheta=sin(Theta);

 for(i=1;i<=2*N;)
 {
  real1=data[i]+costheta*data[i+p]+sintheta*data[i+1+p];
  imag1=data[i+1]+costheta*data[i+1+p]-sintheta*data[i+p];
  real2=data[i]-costheta*data[i+p]-sintheta*data[i+1+p];
  imag2=data[i+1]-costheta*data[i+1+p]+sintheta*data[i+p];

  data[i]=real1;
  data[i+1]=imag1;
  data[i+p]=real2;
```

```
  data[i+p+1]=imag2;

  if(n<k)
  {
   t=t+1;
   Theta=((2*M_PI)/(float)p)*t;
   costheta=cos(Theta);
   sintheta=sin(Theta);
  }
  else
  {
   i=i+p+2;
   n=1;
   t=0;
   Theta=((2*M_PI)/(float)p)*t;
   costheta=cos(Theta);
   sintheta=sin(Theta);
  }
 }
 k=k<<1;
 m++;
}

 printf("\n\n Output of DIT FFt is as follows:\n");
 printf("\n\n Real part of X[k]       Imaginary part of X[k]");
 for(n=1;n<=N;n++)
 {
  printf("\n%f\t\t %f ",data[2*n-1],data[2*n]);
 }
 getch();
}
```

op*********

Readix-2 DIT FFT algorithm

Enter the number of samples in the sequence x(n),N=8

Enter the samples of the sequence x(n):

Real Part Imaginary Part

x(0)= 0.5 0

x(1)= 0.5 0

x(2)= 0.5 0

x(3)= 0.5 0

x(4)= 0 0

x(5)= 0 0

x(6)= 0 0

x(7)= 0 0

Output of DIT FFT is as follows

Real part of X(k)	Imaginary part of X(k)
2.000000	0.000000
0.500000	-1.207107
0.000000	0.000000
0.500000	-0.207107
0.000000	0.000000
0.500000	0.207107
0.000000	0.000000
0.500000	1.207107

Questions:-

1. Define FT,DFT,IDFT
2. Determine relation between FT and DFT
3. Define twiddle factor and state its properties.
4. Compare DIT FFT algorithm with DIF FFT
5. Compute 4 point DFT of x(n)= {1,2,0,4}

Assignment 6: Write a program to find DFT of a given sequences using Goertzel algorithm.

This algorithm exploits **periodicity** property of twiddle factor.

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \qquad (1)$$

Since W_N^{-kN} is equal to 1, multiplying both sides of the equation by this results in;

$$X(k) = W_N^{-kN} \sum_{m=0}^{N-1} x(m) W_N^{mk} = \sum_{m=0}^{N-1} x(m) W_N^{-k(N-m)} \qquad (2)$$

This is in the form of a convolution $y_k(n) = x(n) * h_k(n)$

$$y_k(n) = \sum_{m=0}^{N-1} x(m) W_N^{-k(n-m)} \qquad (3)$$

$$h_k(n) = W_N^{-kn} u(n) \qquad (4)$$

Where $y_k(n)$ is the out put of a filter which has impulse response of $h_k(n)$ and input x(n).

The output of the filter at n = N yields the value of the DFT at the freq $\omega_k = 2\pi k/N$

The filter has frequency response given by

$$H_k(z) = \frac{1}{1 - W_N^{-k} z^{-1}} \qquad (6)$$

The above form of filter response shows it has a pole on the unit circle at the frequency $\omega_k = 2\pi k/N$.

Entire DFT can be computed by passing the block of input data into a parallel bank of N single-pole filters (resonators)

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

void main()
{
int k,n,N;
float static X[100],X_Real[100],X_Imag[100];

clrscr();
printf("\tDiscrete Fourier Transform(DFT)\n");

printf("\n Enter the number samples in the sequence X(n)=");
scanf("%d",&N);
printf("Enter the number samples of sequence X(n)\n");

for(n=0;n<N;n++)
{
printf("X(%d)=",n);
scanf("%f",&X[n]);
}

for(k=0;k<N;k++)
 {
  X_Real[k] = X_Imag[k]=0.0;
    for(n=0;n<N;n++)
     {
       X_Real[k]=X_Real[k]+X[n]*cos((2*M_PI*k*(n-N))/N);
       X_Imag[k]=X_Imag[k]+X[n]*sin((2*M_PI*k*(n-N))/N);
     }
     X_Imag[k]=X_Imag[k]*(-1.0);
 }

printf("\nThe %d point DFT of given sequence is:\n",N);
printf("\n\n\tReal X(k)\t\tImaginary X(k)\n");
   for(k=0;k<N;k++)
   printf("\nX(%d)= %f\t\t\t%f\t\t",k,X_Real[k],X_Imag[k]);
```

```
getch();
}
```

***op**

Discrete Fourier Transform(DFT)

```
 Enter the number samples in the sequence X(n)=8
Enter the number samples of sequence X(n)
X(0)=-1
X(1)=0
X(2)=2
X(3)=0
X(4)=-4
X(5)=0
X(6)=2
X(7)=0
```

The 8 point DFT of given sequence is:

	Real X(k)	Imaginary X(k)
X(0)=	-1.000000	-0.000000
X(1)=	3.000000	-0.000000
X(2)=	-9.000000	0.000000
X(3)=	3.000000	-0.000000
X(4)=	-1.000000	0.000000
X(5)=	3.000000	-0.000000
X(6)=	-9.000000	0.000000
X(7)=	3.000000	-0.000000

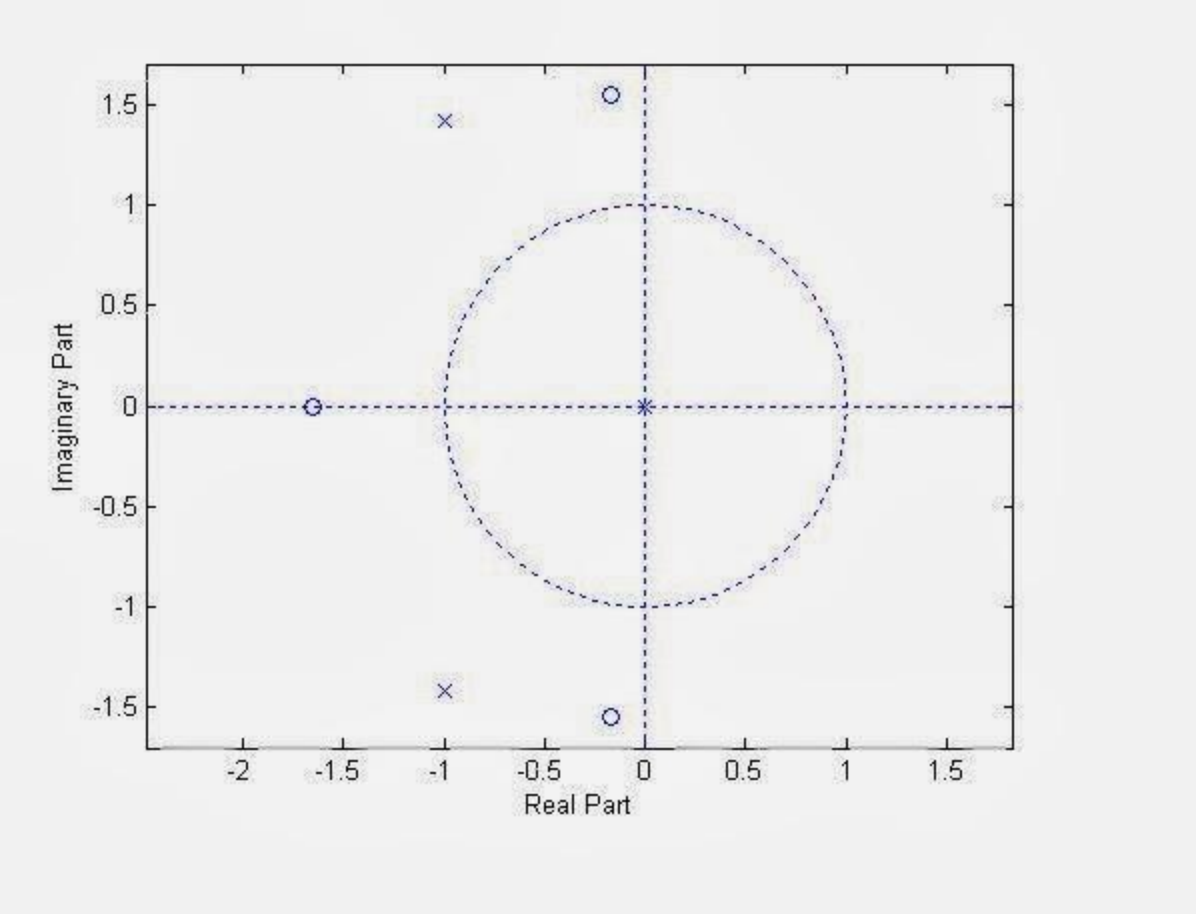
Questions:-

1. State and prove periodicity property of twiddle factor
2. How will you obtain inverse DFT?
3. Describe computation of linear filtering using FFT
4. Derive and explain Goertzel algorithm for computation of DFT
5. Describe FFT butterfly structure.

Assignment 7: WAP to compute Z transform and draw pole zero plot using Matlab

Theory:
Pole Zero Plot The pole-zero plot of a rational z-transform G(z) can be readily obtained using the function zplane. There are two versions of this function. If the z-transform is given in the form of a rational function , the command to use is zplane(num, den) where num and den are row vectors containing the coefficients of the numerator and denominator polynomials of G(z) in ascending powers of z−1. On the other hand, if the zeros and poles of G(z) are given, the command to use is zplane(zeros, poles) where zeros and poles are column vectors. In the pole-zero plot generated by MATLAB, the location of a pole is indicated by the symbol × and the location of a zero is indicated by the symbol º.

```
Close  all;
clear all;
disp('For plotting poles and zeros');
b=input('Input the numerator polynomial coefficients');
a=input('Input the denominator polynomial coefficients');
[b,a]=eqtflength(b,a);
[z,p,k]=tf2zp(b,a);
zplane(z,p);
disp('zeros');
disp(z);
disp('poles');
disp(p);
disp('k');
disp(k);
```



Questions:-

1. Determine relationship between FT,DFT and ZT
2. Define Z transform and Z domain
3. Define ROC and explain its significance.
4. State the properties of Z transform.
5. Explain the condition for stability and causality in Z domain

Assignment 8:-

Write a program to design Butterworth filter design using Bilinear Transformation.

1. Determine the CT filter class:
   1.1 Butterworth
   1.2 Chebychev Type I or Type II
   1.3 Elliptic
   1.4 ...
2. Transform the DT filter specifications to CT (sampling period T_d is arbitrary) **including prewarping the band edge frequencies**
3. Design CT filter based on the magnitude squared response $|H_c(j\Omega)|^2$
   - Determine filter order
   - Determine cutoff frequency
4. Determine $H_c(s)$ corresponding to a stable causal filter
5. Convert to DT filter $H(z)$ via bilinear transform such that

$$H(z) = H_c\left(\frac{2}{T_d}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)\right)$$

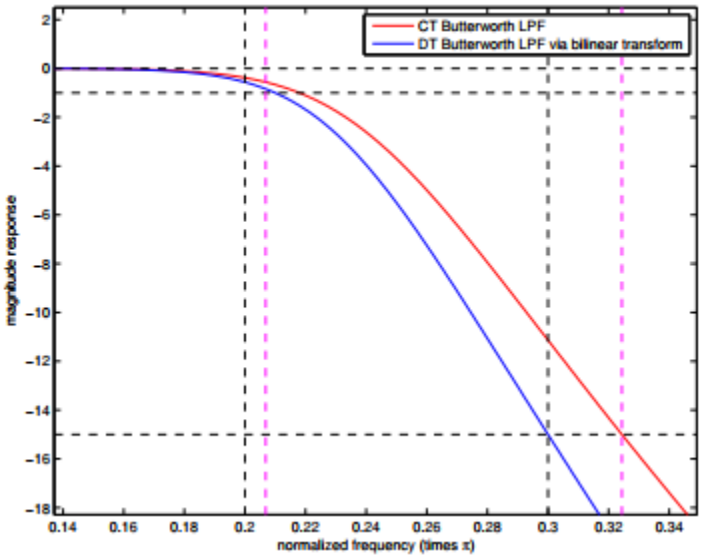
Given a causal stable LTI CT filter $H_c(s)$, we simply substitute

$$\boxed{s = \frac{2}{T_d}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)}$$

to get $H(z)$.

This substitution is based on converting $H_c(s)$ to a differential equation, performing trapezoidal numerical integration with step size T_d to get a difference equation, and then converting the difference equation to a transfer function $H(z)$.

Characteristics of Butterworth filter



Inputs:
1. Order of the Butterworth filter (N)
2. Cutoff frequency of digital filter (Wc).

Outputs: Coefficients of digital filter(cascaded second order).

==========*/

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
 int i;
 float bi[20],b[20][3],a[20][3],p_real,p_imag;
 float wc,N,theta,OmegaC,Den;
 clrscr();
 printf("\t\t Butterworth filter design using Billinear Transformation\n");
 printf("\n\n Enter the order of the filter N= ");
 scanf("%f",&N);
 printf("\n Enter the cutoff frequency of digital filter Wc:");
 scanf("%f",&wc);
 OmegaC=tan(wc/2);
 for(i=0;i<N/2;i++)
 {
  theta=((N+2*i+1)*M_PI)/(2*N);
  p_real=-1*OmegaC*cos(theta);
  p_imag=OmegaC*sin(theta);
  Den=1+2*p_real+p_real*p_real+p_imag*p_imag;
  bi[i]=(OmegaC*OmegaC)/Den;
  b[i][0]=1;
  b[i][1]=2;
  b[i][2]=1;
  a[i][0]=1;
  a[i][1]=(2*(p_real*p_real+p_imag*p_imag)-2)/Den;
  a[i][2]=(1-2*p_real+p_real*p_real+p_imag*p_imag)/Den;
 }
 if((N/2)!=i)
 {
  i--;
  Den=OmegaC+1;
  bi[i]=OmegaC/Den;
```

```
 b[i][0]=1;
 b[i][1]=1;
 b[i][2]=0;
 a[i][0]=1;
 a[i][1]=(OmegaC-1)/Den;
 a[i][2]=0;
 }
 printf("\n The Coefficients of cascaded second order");
 printf(" \n\n sections of digital filter as follows...\n");
 for(i=0;i<N/2;i++)
 {
  printf("\n B[%d]= %f\tb[%d][0]= %f\ta[%d][0]= %f",
     i,bi[i],i,b[i][0],i,a[i][0]);
  printf("\n\t\tb[%d][1]= %f\ta[%d][1]= %f",i,b[i][1],i,a[i][1]);
  printf("\n\t\tb[%d][2]= %f\ta[%d][2]= %f",i,b[i][2],i,a[i][2]);
 }
 getch();
}
```

****o/p***

```
        Butterworth filter design using Billinear Transformation
 Enter the order of the filter N= 2
 Enter the cutoff frequency of digital filter Wc: 90
 The Coefficients of cascaded second order
 Sections of digital filter as follows...
 b[0] = 0.443609     b[0][0]= 1.000000       a[0][0]= 1.000000
             b[0][1]= 2.000000       a[0][1]= 0.549059
             b[0][2]= 1.000000       a[0][2]= 0.225377
```

Questions:-

1. Define bilinear transformation and state its properties
2. Define frequency warping and prewarping
3. List the steps to design digital filter from analog filter.
4. Compare impulse invariance method with bilinear transformation method
5. Describe the characteristics of Butterworth filter approximation.

Assignment 9:- Write a program for FIR filter design using windows

Different Windows

The table below gives the equations for different window types.

Window Type	Weight Equation
Rectangular	$w(n) = 1$
Bartlett	$w(n) = 1 - \frac{2\left\|n - \frac{M}{2}\right\|}{M}$
Hanning	$w(n) = 0.5 - 0.5\ cos\left(\frac{2\pi n}{M}\right)$
Hamming	$w(n) = 0.54 - 0.46\ cos\left(\frac{2\pi n}{M}\right)$
Blackman	$w(n) = 0.42 - 0.5\ cos\left(\frac{2\pi n}{M}\right) + 0.08\ cos\left(\frac{4\pi n}{M}\right)$

Input:

1. Number of coefficients of the filter (M),
2. Cutoff frequency of digital filter (wc),
3. Choice: high pass or low pass.
4. Choice of the window.

Output: Coefficients of the corresponding filter

```
#include<conio.h>
#include<stdio.h>
#include<math.h>
```

```
void main()
 {
  float wc,tou,M,hd[50],h[50],wn,pi,n;
  int ch,p;
  char HPF;
  clrscr();
  printf("\t\tFIR filter design using windows\n\n");
  printf("n\nEnter the length(M) of the filter(coefficient):");
  scanf("%f",&M);
  p=(int)M;
  printf("\n\nEnter the cutoff Frequency(Discrete Frequency) Wc:");
  scanf("%f"&wc);
  printf("\n\nChoice for the filter:\n");
  printf("\n\nEnter 'Y' for High Pass Filter=");
  HPF=toupper(getch());
  printf("%c"HPF);
  tou=(M-1)/2;
  pi=22.0/7.0;
  for(n=0;n<=M-1;n++)
    {
     hd[n]=(sin(wc*(n-tou)))/(pi*(n-tou));
     if(n==tou)&&((p/2)*2!=p))hd[n]=wc/pi;
       {
        if(HPF=='Y')
         {
           for(n=0;n<=M;n++)
            {
              hd[n]=(sin(pi*(n-tou))-sin(wc*(n-tou)))/(pi*(n-tou));
              if(n==tou)&&((p/2)*2!=p))hd[n]=1(wc/pi);
            }
         }
        printf("\n\nEnter the windows you want to use");
        printf("\n1.Rectangular window \n2.Triangular(bartlett)
window\n3.Hamming Window\n4.Hanning window");
        printf("\nEnter your choice");
        scanf("%d",&ch);
        switch(ch)
         {
           case 1:for(n=0;n<=M;n++)
                  {
```

```
                wn=1;
                h[n]=hd[n]*wn;
               }
               break;
        case 2:for(n=0;n<=M-1;n++)
               {
                wn=1-(2*abs(n-tou)/(M-1));
                h[n]=hd[n]*wn;
               }
               break;
        case 3:for(n=0;n<=M-1;n++)
              {
               wn=0.54=0.46*cos((2*pi*n)/(M-1));
               h[n]=hd[n]*wn;
              }
               break;
       case 4:for(n=0;n<=M-1;n++)
              {
               wn=(1-cos((2*pi*n)/(M-1)));
               h[n]=hd[n]*wn;
              }
              break;
     }
    if(HPF=='Y')
     printf("\n\nCoefficient of High Pass FIR Filter are as follows");
    else
     printf("\n\nCoefficient of Low Pass FIR Filter are as follows");
     for(n=0;n<=M-1;n++)
      {
       printf("\n\nh[%1.0f]=%f",n,h[n]);
      }
     getch();
}
```

****

FIR filter design using windows

Enter the length(M) of the filter (coefficient):4

Enter the cutoff frequency(descrete frequency) wc =50

Choice for the filter
Enter 'y' for High Pass Filter
Press key 'l' for Low Pass Filter = Y

Enter the Window you want to use
rectangle window (Enter 1)
triangular(bartlett) window (Enter 2)
hamming window (Enter 3)
hanning window (Enter 4)
choice =1

coefficient of lowpass FIR filter are as follows...
h0=-0.082257
h1=-0.084224
h2=-0.084224
h3=-0.082257

Questions:-

1. Compare IIR filters with FIR filters
2. Illustrate why FIR filter is always stable.
3. State the desirable characteristics of Window
4. Describe Gibb's phenomenon.
5. List the steps for designing FIR filter using window method.
6. State the advantages and disadvantages of window method.

Assignment 10 : To implement the difference equation of the LSI system

To implement the difference equation of the LSI system which is given as
y(n)=-[a1*y(n-1)+a2*y(n-2)+a3*y(n-3)+.....]+b0*x(n)+b1*x(n-1)+b2*x(n-2)+....

INPUTS: 1.Number of coefficients ak, denoted as N.
2. Values of a1, a2, a3...etc.
3. Number of coefficients bk, denoted as M.
4. Values of b0, b1, b2...etc.
5. Number of samples of x(n),denoted as L.
6. Values of x(0),x(1),x(2),...etc.

OUTPUTS: The computed output sequence y(n) is according to the specified format of difference equation as above.

ASSUMPTIONS:
1. The number of samples computed for y(n) are same as number of input samples.
2. All initial conditions are assumed zero.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

void main()
{
 int N,M,k,L,n;
 float a[10],b[10],x[100],y[100],sum_x,sum_y;

 clrscr();

 printf("\n\t Implementation of General Difference Eqn based array mapping");

 printf("\n\n Enter the number of coefficients a[k]=");
 scanf("%d",&N);
 printf("\n\n Enter the values coefficients a[k]:\n");

 for(k=1;k<=N;k++)
 {
  printf("a%d=",k);
```

```
 scanf("%f",&a[k]);
}

printf("\n\n Enter the number of coefficients b[k]=");
scanf("%d",&M);
printf("\n\n Enter the values coefficients b[k]:\n");

for(k=0;k<M;k++)
{                   // values of b0,b1,b2,....
 printf("b%d=",k);
 scanf("%f",&b[k]);
}

printf("\n\n Enter the number of samples of x(n):");
scanf("%d",&L);

for(k=0;k<L;k++)
{                   // values of x(0),x(1),x(2),....
 printf("x(%d)=",k);
 scanf("%f",&x[k]);
}

printf("\n The computed values of y(n) are:");
for(n=0;n<L;n++)
{
 sum_y=0;
 sum_x=0;

 for(k=1;(k<=n)&&(k<=N);k++)   //computation of a1*y(n-1)+a2*y(n-2)+a3*y(n-
3)+..
 {
  sum_y+=a[k]*y[n-k];
 }

 for(k=0;(k<=n)&&(k<=n)&&(k<M);k++)
 {                          //computation of b0*x(n)+b1*x(n-1)+b2*x(n-2)+..

  sum_x+=b[k]*x[n-k];
 }
 y[n]=-sum_y+sum_x;
```

```
 printf("\n y(%d)=%f",n,y[n]);
 }
 getch();
}
```

Output

Implementation of General Difference Equation based array mapping

```
Enter the number of coefficients a[k]=3
Enter the values coefficients a[k]:
a1=1
a2=2
a3=3

Enter the number of coefficients b[k]=3
Enter the values coefficients b[k]:

b0=3
b1=2
b2=1

Enter the number of samples of x(n):2
x(0)=1
x(1)=2

The computed values of y (n) are:
y (0)=3.000000
y (1)=5.000000
```

Questions

1. Define difference equation and how will you obtain total solution
2. Specify steps for homogeneous solution
3. Specify steps for particular solution
4. What is recursive and non-recursive system?
5. What are the basic operations required to implement the specified difference equation of a system.
6. Define canonic and non-canonic structures.