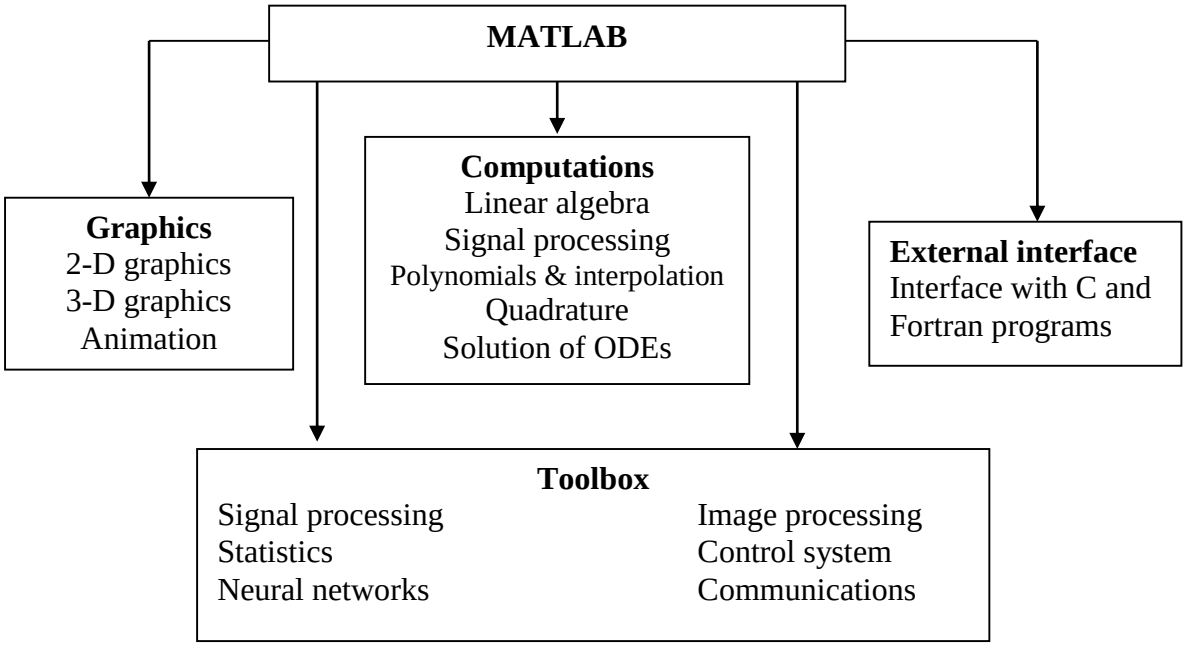
INRODUCTION

MATLAB: MATLAB is a software package for high performance numerical computation and visualization provides an interactive environment with hundreds of built in functions for technical computation, graphics and animation. The MATLAB name stands for MATrix Laboratory

The diagram shows the main features and capabilities of MATLAB.



At its core ,MATLAB is essentially a set (a "toolbox") of routines (called "m files" or "mex files") that sit on your computer and a window that allows you to create new variables with names (e.g. voltage and time) and process those variables with any of those routines (e.g. plot voltage against time, find the largest voltage, etc).

It also allows you to put a list of your processing requests together in a file and save that combined list with a name so that you can run all of those commands in the same order at some later time. Furthermore, it allows you to run such lists of commands such that you pass in data and/or get data back out (i.e. the list of commands is like a function in most programming languages). Once you save a function, it becomes part of your toolbox (i.e. it now looks to you as if it were part of the basic toolbox that you started with).

For those with computer programming backgrounds: Note that MATLAB runs as an interpretive language (like the old BASIC). That is, it does not need to be compiled. It simply reads through each line of the function, executes it, and then goes on to the next line. (In

practice, a form of compilation occurs when you first run a function, so that it can run faster the next time you run it.)

MATLAB Windows:

MATLAB works with through three basic windows

Command Window **:** This is the main window .it is characterized by MATLAB command prompt >> when you launch the application program MATLAB puts you in this window all commands including those for user-written programs ,are typed in this window at the MATLAB prompt

Graphics window**:** the output of all graphics commands typed in the command window are flushed to the graphics or figure window, a separate gray window with white background color the user can create as many windows as the system memory will allow

Edit window**:** This is where you write edit, create and save your own programs in files called M files.

Input-output: MATLAB supports interactive computation taking the input from the screen and flushing, the output to the screen. In addition it can read input files and write output files

Data Type**:** the fundamental data –type in MATLAB is the array. It encompasses several distinct data objects- integers, real numbers, matrices, charcter strings, structures and cells.There is no need to declare variables as real or complex, MATLAB automatically sets the variable to be real.

Dimensioning: Dimensioning is automatic in MATLAB. No dimension statements are required for vectors or arrays .we can find the dimensions of an existing matrix or a vector with the size and length commands.

Basic Instructions in MATLab

- MATLab is case sensitive.
- "%" is used to add comment.
- Help is provided by typing "help" or if the topic is known then type "help Funtion_name"
  or "doc function_name".
- If the exact name of the topic or command that you are looking for is unknown, then type "lookfor keyword" i.e. "lookfor filter".
- If statement is terminated by " ; " , then intermediate results are not displayed in the command window otherwise displays the result.
- Use the Up-arrow or Down-arrow to recall commands without retyping them in command window.

- **T = 0: 1:10**
  This instruction indicates a vector T which as initial value 0 and final value 10 with an increment of 1. Therefore T = [0 1 2 3 4 5 6 7 8 9 10]

- **F= 20: 1: 100** Output F = [20 21 22 23 24 …… 100]

- **T= 0:1/pi: 1** Output T= [0, 0.3183, 0.6366, 0.9549]

- **Row Vector:**
  a= [1 2 3 4] Output a = 1 2 3 4

- **Length of vector**
  length(a) ans = 4

- **Transpose:**
  c=a' Output c =
  1
  2
  3
  4

- **Column vector:**
  b = [1;2;3] Output b =
  1
  2
  3

- **Matrix:**
  M=[1 2 3 4; 5 6 7 8] Output M =
  1 2 3 4
  5 6 7 8

- **Size of array**
  size(M) ans = 2 4

- **Identity Matrix:**
  I= eye(3) Output I =
  1 0 0
  0 1 0

0 0 1

size(I) ans = 3 3

I(1:2,1:2) ans = 1 0
0 1

Accessing a row or column of matrix

Second row: I(2,:) ans = 0 1 0

Second column: I(:,2) ans = 0
1
0

- **zeros (1, 3)** Output = 0 0 0
- **zeros (2,4)** Output = 0 0 0 0
0 0 0 0
- **ones (5,2)** This instruction creates a vector of five rows and two columns

Output = 1 1
1 1
1 1
1 1
1 1

- **Random Matrix or vector**
R=rand(2,3) Output = R = 0.9134 0.0975 0.5469
0.6324 0.2785 0.9575

Access a row or column of matrix: R(4) or R(2,2) ans = 0.2785

- **a = [1 2 3] b = [4 5 6]**

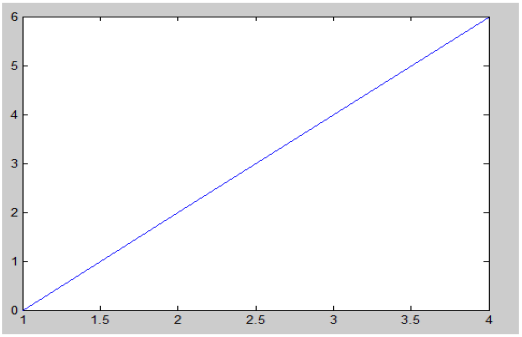
a.*b = 4 10 18 which is multiplication of individual elements. i.e. [4x1 5x2 6x3]

if C= [2 2 2] b.*C results in [8 10 12]

- **p = 2 *(-2) + 3^2 -1/4**

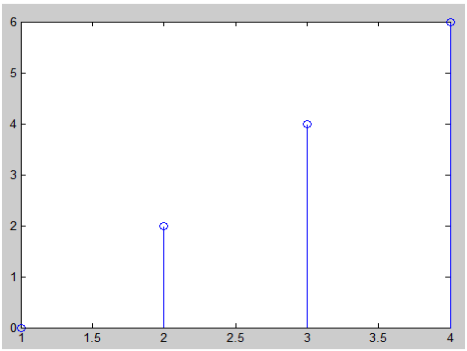
p = 4.7500

➢ **x = [0 2 4 6]; t = [1 2 3 4] ; plot (t, x);**
This instruction will display a figure window which indicates the plot of x versus t



- **stem (t,x)**

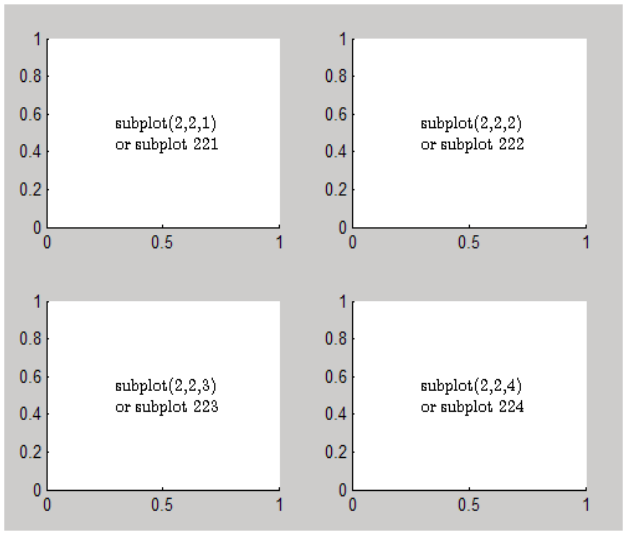
x = [0 2 4 6]; t = [1 2 3 4] ; stem (t, x);



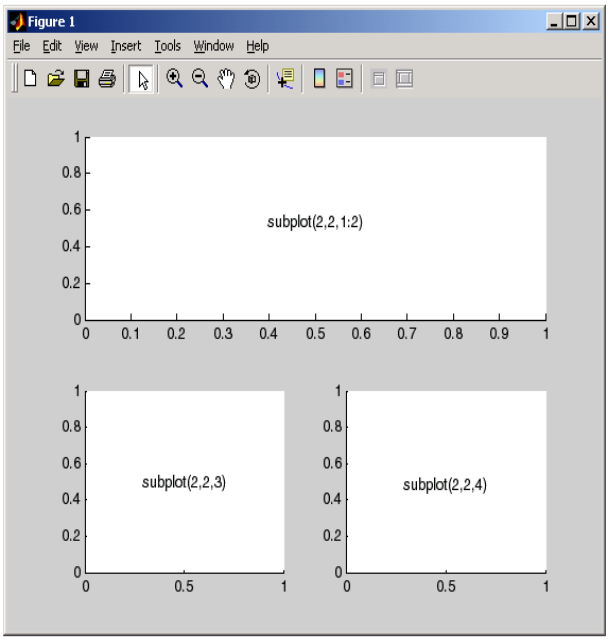
- Subplot: This function divides the figure window into rows and columns

Subplot (2 2 1) divides the figure window into 2 rows and 2 columns 1 represent number of the figure	
1 **(2 2 1)**	**2** **(2 2 2)**
3 **(2 2 3)**	**4** **(2 2 4)**

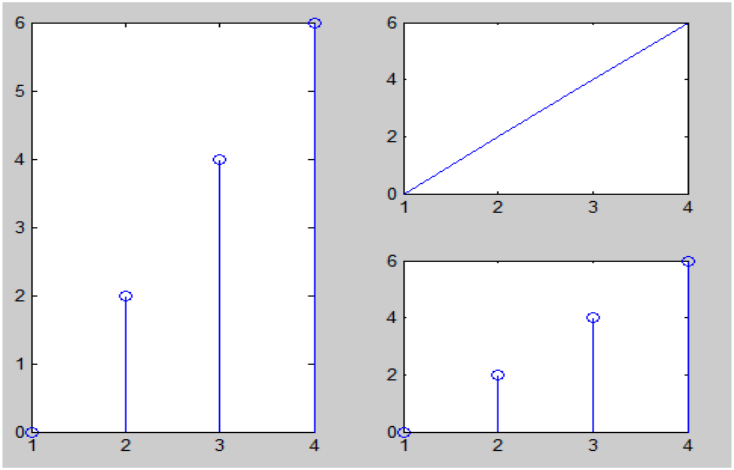
1 **(3, 1, 1)**
2 **(3, 1, 2)**
3 **(3, 1, 3)**




```
subplot(2,2,1:2)
subplot(2,2,3)
subplot(2,2,4)
```




```
x = [0 2 4 6];    t = [1 2 3 4]
;
subplot(2,2,[1 3])
stem (t, x);
subplot(2,2,2)
plot (t, x);
subplot(2,2,4)
stem (t, x);
```



- **Filter** **Syntax:** y = filter(b,a,X)

***Description*:** y = filter(b,a,X) filters the data in vector X with the filter described by numerator coefficient vector b and denominator coefficient vector a.If a(1) is not equal to 1, filter normalizes

the filter coefficients by a(1). If a(1) equals 0, filter returns an error.

- **Impz** **Syntax:** [h,t] = impz(b,a,n)

Description: [h,t] = impz(b,a,n) computes the impulse response of the filter with numerator

coefficients b and denominator coefficients a and computes n samples of the impulse response

when n is an integer (t = [0:n-1]'). If n is a vector of integers, impz computes the impulse response

at those integer locations, starting the response computation from 0 (and t = n or t = [0 n]).

If, instead of n, you include the empty vector [] for the second argument, the number of samples is

computed automatically by default.

- **Fliplr** **Syntax: B =** fliplr(A)

Description: B = fliplr(A) returns A with columns flipped in the left-right direction, that is, about

a vertical axis. If A is a row vector, then fliplr(A) returns a vector of the same length with the

order of its elements reversed. If A is a column vector, then fliplr(A) simply returns A.

- **Flipud** **Syntax: B =** flipud(A)

Flipud: Flip matrix in up/down direction. flipud(A) returns A with columns preserved and rows flipped in the up/down direction.

- **Conv** **Syntax:** w = conv(u,v)

Description: w = conv(u,v) convolves vectors u and v. Algebraically, convolution is the same operation as multiplying the polynomials whose coefficients are the elements of u and v.

Example:

x=[1 2 3 4];
h=[1 1 1 1];
y=conv(x,h)
y = 1 3 6 10 9 7 4
	A=[1 2 3 4];
	B=fliplr(A)
	B = 4 3 2 1
		A= [1;2;3;4]
		A = 1
		2
		3
		4
			B=flipud(A)
			B = 4
			3
			2
			1

- **Disp** **Syntax:** disp(X)

Description: disp(X) displays an array, without printing the array name. If X contains a text string,

the string is displayed. Another way to display an array on the screen is to type its name, but this

prints a leading "X=," which is not always desirable. Note that disp does not display empty arrays.

- **xlabel** **Syntax:** xlabel('string')

  Description: xlabel('string') labels the x-axis of the current axes.

- **ylabel** **Syntax :** ylabel('string')

  Description: ylabel('string') labels the y-axis of the current axes.

- **Title** **Syntax :** title('string')

  Description: title('string') outputs the string at the top and in the center of the current axes.

- **grid on** **Syntax :** grid on

  Description: grid on adds major grid lines to the current axes.

- ***Program to generate sine ware***

```
clc;  clear all;  close all;
t = [0:0.001:0.1];
f = input('Enter the input frequency: ');
x1=sin(2*pi*f*t);
figure(1);
plot(t,x1,'b');   xlabel('time');  ylabel('amplitude');
title(' First signal ');
grid on;
t = [0:0.0001:.2];
x2=sin(2*pi*f*t);
figure(2);
plot(t,x2,'r*');   xlabel('time'); ylabel('amplitude');
title('Second signal');
```

- ***Basic Signals***

```
clc; clear all; close all;

N = 10;
n = -N:1:N;
% unit Impulse
x1=[zeros(1,N)  1   zeros(1,N)];
subplot(3,2,1); stem(n,x1); title(' Impulse  Signal');

%unit Step signal
x2=[zeros(1,N)   1   ones(1,N)];
subplot(3,2,2); stem(n,x2); title('Unit step signal');
```

```
%Unit Ramp signal
a1= 2;
x3=a1*n;
subplot(3,2,3); stem(n,x3); title('Unit ramp signal');

%Exponential growing/decaying signal
n2=0:0.1:N;
a2=2;
x4=a2.^n2;
subplot(3,2,4); stem(n2,x4); title( 'Exponential growing signal');
a3=0.5;
x5=a3.^n2;
subplot(3,2,5); stem(n2,x5); title( 'Exponential decaying signal');

%cosine siganl
x6=cos(n2);
subplot(3,2,6); stem(n2,x6);  title('Cosine signal');
```

- **Zplane**

```
%For data sampled at 1000 Hz, plot the poles and zeros
%of a 4th-order elliptic lowpass digital filter
%with cutoff frequency of 200 Hz, 3 dB of ripple in the passband,
%and 30 dB of attenuation in the stopband:
clc;
[z,p,k] = ellip(4,3,30,200/500);
zplane(z,p);
title('4th-Order Elliptic Lowpass Digital Filter');
```

Programs demonstrated in workshop

Program1-Simple program

```
clc
%This is an example program
%% is used to write comment
A=10;%assigns 10 to variable A
t=0:.2:1 %Creates a row vector t with contents 0 to 1 in steps of 0.2
xt=A*exp(2*t)
stem(t,xt);
title('Plot of x(t)=10exp(2t)');
xlabel('Time');
ylabel('Magnitude');
%grid on;
%Program to generate different signals
%1. To Generate a CT unit step signal
t=0:.01:1;
ut=ones(1,length(t));
plot (t,ut,'r');
hold on % to display next graphs in same graphical window
```

Program2-To generate different signals

```
%2. To Generate a CT unit ramp signal
rt=2*t;
plot (t,rt,'b');


%3. To Generate a CT exponential signal
a=input('Enter the value of a  ');
xt=exp(a*t);
plot(t,xt,'c','LineWidth',3);


%4. To Generate a CT Sinusoidal signal signal
f=input('Enter the frequency of the signal    ');
Sxt=a*sin(2*pi*f*t);
plot(t,Sxt,'r');


%5. To Generate a CT damped Sinusoidal signal signal

dSxt=xt.*sin(2*pi*f*t);
plot(t,dSxt,'r');

hold off;
```

Program3-Creation and calling of Function program

```
%Example program to understand function program
%This is a Main program which calls functions sum-diff and product

a=10;
b=5;
[P,Z]=sum_diff(a,b);
disp([P,Z]);
%Different ways of calling function
fprintf('result of function program product is %d \n',product(a,b));
result=product(a,b)*10
result1=product(a,b)*[P,Z]

%This is a function program
function [P,Z]=sum_diff(x,y)
P=x+y;
Z=x-y;

%Save this program as sum_diff.m

%This is another function program
function prod=product(x,y)
prod=x*y;

%Save this program as product.m
```

Application Programs

Program4-To find partial fraction of function and plot poles and zeros

```
% Program to find partial fraction expansion of system
% residuez--> Z-transform partial-fraction expansion.
% [R,P,K] = residuez(B,A)
% where B and A are the numerator and denominator polynomial coefficients,
% respectively, in ascending powers of z^(-1).
%It finds the residues, poles and direct terms of the
%partial-fraction expansion of B(z)/A(z),
%
%        B(z)       r(1)                r(n)
%        ---- = ------------ +...  ------------ + k(1) + k(2)z^(-1) ...
%        A(z)   1-p(1)z^(-1)       1-p(n)z^(-1)
%
% R and P are column  vectors containing the residues and poles,
respectively.
%K contains the direct terms in a row vector.
%The number of poles is  n = length(A)-1 = length(R) = length(P)
% The direct term coefficient vector is empty if length(B) < length(A)
(proper fraction);
% otherwise,length(K) = length(B)-length(A)+1
% For Example
% Compute the partial fraction expansion of the following transfer
%H(z)=1/(1-Z^-1),num=[1], den=[1 -1]
%  Similarly try function H(z) = (1 + 2z^-1) / (1 - z^-1 + 2z^-2).

    num = [1];                % Numerator coefficients
    den = [1 -1];             % Denominator coefficients
    [r,p k] = residuez(num,den)    % H(z) = r(1)/(1-p(1)z^-1) +r(2)/(1-
p(2)z^-1)+...
    zplane(num,den)%plots pole zero plot of TF
```

Program5- System Analysis

```
%Program to analyse a system defined by nth order difference equation
%i.e. to find impulse response, response of system for step,sine, any input
%let us take a 2nd order difference equation
%y(n)-4y(n-1)+4y(n-2)=x(n)  -->here a0=1,a1=-4,a2=4,b0=1
%Try here a0=1,a1=2,a2=2,b0=1
%1.To find impulse response of the system

b=1;
% a=[1 -4 4];
a=[1 1];%y(n)-y(n-1)=x(n) i.e.h(n)=iZT[1/(1+Z^-1)] i.e.h(n)=(-1)^n*u(n)
%Try impz(b,a)
%Try impz(b,a,5)
[h n]=impz(b,a);%computes h(n)
subplot(311);stem(h);title('impulse response')

%To find response of a system for given input
%Let us find response of above system for step input

x=ones(1,length(h));%Define a step signal of lentgth equal to length of
h(n) or any length
y=filter(b,a,x);
```

```
subplot(312);stem(x);title('Step input');
subplot(313);stem(y);title('Step response');

figure%Create new figure window

%To find response of system for sine wave
xsin=2*sin(2*pi*n/10);%Create sine wave input
ysin=filter(b,a,xsin);

subplot(211);stem(xsin);title('Sine input');
subplot(212);stem(ysin);title('Sine response');

figure%Create new figure window

%To find response of system for other input
xin=2*n;%Create any input signal
yin=filter(b,a,xin);

subplot(211);stem(xin);title('Input');
subplot(212);stem(yin);title('S/m Response');
```

Program6 –To find N point DFT of a signal

```
%Program to fing N point DFT of a signal
%Let x(n)=[1 3 4 5 2] later try for x(n)=sin(2*pi*n/N1) where N1 is period
% xn=input('Enter the sequence');
%for x(n) =x(n)=sin(2*pi*n/N) -->comment above statement and uncomment
following three statements
N1=10;
n=0:N1-1;
xn=sin(2*pi*n/N1)
N=input('Enter the length of the DFT   ')
Xk=[];
for k=0:N-1
    X=0;
    for n=0:length(xn)-1
        X=X+xn(n+1)*exp(-i*2*pi*k*n/N);
    end
    Xk(k+1)=X;
end
disp(Xk)

%To plot mag. and phase of DFT
k=0:N-1;
n=0:length(xn)-1;
subplot(311);stem(n,xn,'r');
title('Original signal')
subplot(312);stem(k,abs(Xk),'g');
title('Magnitude plot');
subplot(313);stem(k,angle(Xk),'b');
title('Phase plot')


%Verify using built in command fft(x,N);
Xkc=fft(xn,N);
d=Xk-Xkc;

figure
```

```
subplot(211);stem(k,abs(Xk),'r','filled');title('Without using built in
command');
subplot(212);stem(k,abs(Xkc),'g');title('Using built in command');
```

Program 7-To find DTFT of a signal

```
%Let x(n) be a DTNP signal x(n)=a^n
%Later try with unit step signal
N=input('length of x(n)    ');
n=0:N-1;
xn=2.^n;
% xn=ones(1,N);% For step signal

m=1;
DTFTxn=[];


for w=-4*pi:.2:4*pi
    xw=0;
    for n1=0:N-1
    xw=xw+xn(n1+1)*exp(-i*w*n1);
    end
    DTFTxn(m)=xw;
    m=m+1;
end

%To plot DTFT

w=-4*pi:0.2:4*pi

subplot(211);stem(xn);title('DT Signal')
subplot(212);plot(w/pi,abs(DTFTxn));

xlabel('frequency in \pi radians');
title('DTFT of DT signal  and its nature9Continuous in Freq. and
Periodic)')
```

Program 8-To find DTFS of a signal

```
%To find FS of a signal
N=10;
n=0:19
% xt=sin(2*pi*n/N);
N1=5;
xt=sin(2*pi*n/N)+sin(2*pi*n/N1);;
for k=0:19
    Xk=0;
    for j=0:19
        Xk=Xk+xt(j+1)*exp(-i*2*pi*j*k/N);
    end
    X(k+1)=Xk/10
```

```
end
subplot(211);stem(n,xt);
subplot(212);stem(n,abs(X));
```

Program9- Audio recording

```
% Program to record audio from system microphone and ply back
recObj=audiorecorder% Prepares audio device for recording
%when program is run the properties are displayed
disp('Start speaking.');%Start speaking once this message is displayed
recordblocking(recObj, 5);
disp('End of Recording.');% It records for 5 seconds and displays this
message.you can increase time
play(recObj,recObj.samplerate);%plays the recorded sound with same
frequency as recording.
```

Program10- Audio signal visualization

```
%Program to see
clc
close all;
clear all;
f=0.8;
n=6
a=fir1(n,f,'high')
b=fir1(n,f,'low')
[y,fs]=audioread('file_example_WAV_1MG.wav')
sound (y,fs);%plays the audio file
o=filter(a,1,y);
p=filter(b,1,o);
%Filter Visualization Tool-> use help fvtool to know more
fvtool(a);%displays HPF response
fvtool(b);%displays LPF response
subplot(3,1,1);
plot(y);%displays sound signal
subplot(3,1,2);
plot(o);%displays sound signal when passed through HPF
subplot(3,1,3);
plot(p);%displays sound signal when passed through LPF
```

Program11- Create a signal flow diagram of a system

```
% Create a signal flow diagram given a set of b and a coefficients of a
discrete system
% example usage:
   b = [ 2 -0.5];
   a = [1 -.025]
%    function create_signal_flow(b, a)
% By David Dorran, david.dorran@dit.ie, September 2012.

% function create_signal_flow(b,a) % by uncommenting this line and...
% commenting lines 3 and 4 you can make it in to function

% The diagram consists of lines, arrows, boxes and circles. The following
% variables set up the dimiensions of these components.
x_arrow = 0.2;
y_arrow_original = 0.9;
y_arrow = y_arrow_original;
```

```
arrow_len = 0.1;
delay_line_len = 0.1;
mult_line_len = 0.02;
circle_dia = 0.05;
box_width = 0.05;
y_increment = -delay_line_len-box_width;

% Check that the parameters passed to the function are oc
if(length(b) < 1)
    error('There must be at least one b coefficient');
end
if(length(b) > 6 | length(a) > 6)
    error('Unfortunately the max number of coeeficients is 6');
end

%make the output arrow longer if there is feedback in the system
if(length(a) > 1)
    op_len_mult =3.5;
else
    op_len_mult =1;
end



figure
annotation('textbox', [x_arrow-0.1 y_arrow-delay_line_len/2-
box_width/2+0.01 box_width box_width],'fontname','Times New
Roman','fontsize',16,'String', '\it{x}', 'LineStyle','none' ,
'HorizontalAlignment','Center' ,'VerticalAlignment','middle');



if(length(b) > 1 | length(a) > 1)
    annotation('textbox',
[x_arrow+arrow_len+circle_dia+arrow_len+circle_dia+arrow_len*op_len_mult+0.
01 y_arrow-delay_line_len/2-box_width/2+0.01 box_width
box_width],'fontname','Times New Roman','fontsize',16,'String', '\ity',
'LineStyle','none' , 'HorizontalAlignment','Center'
,'VerticalAlignment','middle');
else
    annotation('textbox', [x_arrow+arrow_len+circle_dia+arrow_len+0.01
y_arrow-delay_line_len/2-box_width/2+0.01 box_width
box_width],'fontname','Times New Roman','fontsize',16,'String', '\ity',
'LineStyle','none' , 'HorizontalAlignment','Center'
,'VerticalAlignment','middle');
end
annotation('line', [x_arrow-0.05 x_arrow+arrow_len],[y_arrow-
delay_line_len/2 y_arrow-delay_line_len/2] );

%multiplier
annotation('ellipse', [x_arrow+arrow_len y_arrow-circle_dia/2-
delay_line_len/2 circle_dia circle_dia] );
annotation('line', [x_arrow+arrow_len+circle_dia/4
x_arrow+arrow_len+3*circle_dia/4],[y_arrow+circle_dia/4-delay_line_len/2
y_arrow-circle_dia/4-delay_line_len/2] );
annotation('line', [x_arrow+arrow_len+circle_dia/4
x_arrow+arrow_len+3*circle_dia/4],[y_arrow-circle_dia/4-delay_line_len/2
y_arrow+circle_dia/4-delay_line_len/2] );
annotation('arrow', [x_arrow+arrow_len+circle_dia
x_arrow+arrow_len+circle_dia+arrow_len],[y_arrow-delay_line_len/2 y_arrow-
delay_line_len/2] );
```

```
%multiplier value
annotation('line', [x_arrow+arrow_len+circle_dia/2
x_arrow+arrow_len+circle_dia/2],[y_arrow-circle_dia/2-mult_line_len-
delay_line_len/2 y_arrow-circle_dia/2-delay_line_len/2] );
annotation('Textbox', [x_arrow+arrow_len y_arrow-circle_dia/2-
mult_line_len-box_width-delay_line_len/2 box_width box_width ],  'String',
num2str(b(1)), 'LineStyle','none' , 'HorizontalAlignment','Center'
,'VerticalAlignment','middle');

%An adder is only required if there are delays in the system
if(length(b) > 1 | length(a) > 1)
    %adder
    annotation('ellipse', [x_arrow+arrow_len+circle_dia+arrow_len y_arrow-
circle_dia/2-delay_line_len/2 circle_dia circle_dia] );
    annotation('line', [x_arrow+arrow_len+circle_dia+arrow_len+circle_dia/4
x_arrow+arrow_len+circle_dia+arrow_len+3*circle_dia/4],[y_arrow-
delay_line_len/2 y_arrow-delay_line_len/2] );
    annotation('line', [x_arrow+arrow_len+circle_dia+arrow_len+circle_dia/2
x_arrow+arrow_len+circle_dia+arrow_len+circle_dia/2],[y_arrow+circle_dia/4-
delay_line_len/2  y_arrow-circle_dia/4-delay_line_len/2 ] );
    annotation('arrow', [x_arrow+arrow_len+circle_dia+arrow_len+circle_dia
x_arrow+arrow_len+circle_dia+arrow_len+circle_dia+arrow_len*op_len_mult],[y
_arrow-delay_line_len/2 y_arrow-delay_line_len/2] );
end

%Add a feedforward delay line for each b coefficient
for k = 2: length(b)
    if(k == max([length(a) length(b)]) & length(a)~= length(b))
        line_extra = circle_dia/2;
        line_type = 'line' ;
    else
        line_extra = 0;
        line_type = 'arrow' ;
    end

    % delay box feedforward
    annotation('line', [x_arrow x_arrow],[y_arrow-delay_line_len/2 y_arrow-
delay_line_len] );
    annotation('rectangle', [x_arrow-box_width/2 y_arrow-delay_line_len-
box_width box_width box_width  ] );
    annotation('textbox', [x_arrow-box_width/2 y_arrow-delay_line_len-
box_width box_width box_width  ], 'String', 'D' ,
'HorizontalAlignment','Center' ,'VerticalAlignment','middle');
    annotation('line', [x_arrow x_arrow],[y_arrow-delay_line_len-box_width
y_arrow-3*delay_line_len/2-box_width] );

    %line to multiplier
    annotation('line', [x_arrow x_arrow+arrow_len],[y_arrow-
3*delay_line_len/2-box_width y_arrow-3*delay_line_len/2-box_width] );

    %multiplier
    annotation('ellipse', [x_arrow+arrow_len y_arrow-3*delay_line_len/2-
box_width-circle_dia/2 circle_dia circle_dia] );
    annotation('line', [x_arrow+arrow_len+circle_dia/4
x_arrow+arrow_len+3*circle_dia/4],[y_arrow-3*delay_line_len/2-box_width-
circle_dia/4 y_arrow-3*delay_line_len/2-box_width+circle_dia/4] );
    annotation('line', [x_arrow+arrow_len+circle_dia/4
x_arrow+arrow_len+3*circle_dia/4],[y_arrow-3*delay_line_len/2-
box_width+circle_dia/4 y_arrow-3*delay_line_len/2-box_width-circle_dia/4]
);
```

```
    %line out of multiplier
    annotation(line_type, [x_arrow+arrow_len+circle_dia
x_arrow+arrow_len*2+circle_dia+line_extra],[y_arrow-3*delay_line_len/2-
box_width y_arrow-3*delay_line_len/2-box_width] );

    % multiplier value
    annotation('line', [x_arrow+arrow_len+circle_dia/2
x_arrow+arrow_len+circle_dia/2],[y_arrow-3*delay_line_len/2-box_width-
circle_dia/2 y_arrow-3*delay_line_len/2-box_width-circle_dia/2-
mult_line_len] );
    annotation('textbox', [x_arrow+arrow_len+circle_dia/2-box_width/2
y_arrow-3*delay_line_len/2-box_width-circle_dia/2-mult_line_len-box_width
box_width box_width],  'String', num2str(b(k)), 'LineStyle','none' ,
'HorizontalAlignment','Center' ,'VerticalAlignment','middle' );

    %Adder
    if(k ~= max([length(a) length(b)])  | length(a) == length(b))
        annotation('ellipse', [x_arrow+arrow_len*2+circle_dia y_arrow-
3*delay_line_len/2-box_width-circle_dia/2 circle_dia circle_dia] );
        annotation('line', [x_arrow+arrow_len*2+circle_dia+circle_dia/2
x_arrow+arrow_len*2+circle_dia+circle_dia/2],[y_arrow-3*delay_line_len/2-
box_width-circle_dia/4 y_arrow-3*delay_line_len/2-box_width+circle_dia/4]
);
        annotation('line', [x_arrow+arrow_len*2+circle_dia+circle_dia/4
x_arrow+arrow_len*2+circle_dia+3*circle_dia/4],[y_arrow-3*delay_line_len/2-
box_width y_arrow-3*delay_line_len/2-box_width] );
    end
    %line out of adder
    annotation('arrow', [x_arrow+arrow_len*2+circle_dia+circle_dia/2
x_arrow+arrow_len*2+circle_dia+circle_dia/2],[y_arrow-3*delay_line_len/2-
box_width+circle_dia/2-line_extra y_arrow-delay_line_len/2-circle_dia/2]);

    y_arrow = y_arrow+y_increment;
end
y_arrow = y_arrow_original;
x_arrow = x_arrow+(arrow_len+circle_dia+arrow_len+circle_dia/2)*2;
for k = 2: length(a)
    %
    if(k == max([length(a) length(b)]) & length(a)~= length(b))
        line_extra = circle_dia/2;
        line_type = 'line' ;
    else
        line_extra = 0;
        line_type = 'arrow' ;
    end



    % delay box feedforward
    annotation('line', [x_arrow x_arrow],[y_arrow-delay_line_len/2 y_arrow-
delay_line_len] );
    annotation('rectangle', [x_arrow-box_width/2 y_arrow-delay_line_len-
box_width box_width box_width  ] );
    annotation('textbox', [x_arrow-box_width/2 y_arrow-delay_line_len-
box_width box_width box_width  ], 'String', 'D' ,
'HorizontalAlignment','Center' ,'VerticalAlignment','middle');
    annotation('line', [x_arrow x_arrow],[y_arrow-delay_line_len-box_width
y_arrow-3*delay_line_len/2-box_width] );

    %line to multiplier
```

```
    annotation('line', [x_arrow x_arrow-arrow_len],[y_arrow-
3*delay_line_len/2-box_width y_arrow-3*delay_line_len/2-box_width] );

    %multiplier
    annotation('ellipse', [x_arrow-arrow_len-circle_dia y_arrow-
3*delay_line_len/2-box_width-circle_dia/2 circle_dia circle_dia] );
    annotation('line', [x_arrow-arrow_len-circle_dia/4 x_arrow-arrow_len-
3*circle_dia/4],[y_arrow-3*delay_line_len/2-box_width-circle_dia/4 y_arrow-
3*delay_line_len/2-box_width+circle_dia/4] );
    annotation('line', [x_arrow-arrow_len-circle_dia/4 x_arrow-arrow_len-
3*circle_dia/4],[y_arrow-3*delay_line_len/2-box_width+circle_dia/4 y_arrow-
3*delay_line_len/2-box_width-circle_dia/4] );


    %line out of multiplier
    annotation(line_type, [x_arrow-arrow_len-circle_dia x_arrow-
arrow_len*2-circle_dia-line_extra],[y_arrow-3*delay_line_len/2-box_width
y_arrow-3*delay_line_len/2-box_width] );

    % multiplier value
    annotation('line', [x_arrow-arrow_len-circle_dia/2 x_arrow-arrow_len-
circle_dia/2],[y_arrow-3*delay_line_len/2-box_width-circle_dia/2 y_arrow-
3*delay_line_len/2-box_width-circle_dia/2-mult_line_len] );
    annotation('textbox', [x_arrow-arrow_len-circle_dia/2-box_width/2
y_arrow-3*delay_line_len/2-box_width-circle_dia/2-mult_line_len-box_width
box_width box_width],  'String', num2str(-1*a(k)/a(1)), 'LineStyle','none'
, 'HorizontalAlignment','Center' ,'VerticalAlignment','middle' );

    %Adder

    if(k ~= max([length(a) length(b)]) | length(a)== length(b))
        annotation('ellipse', [x_arrow-arrow_len*2-circle_dia*2 y_arrow-
3*delay_line_len/2-box_width-circle_dia/2 circle_dia circle_dia] );
        annotation('line', [x_arrow-arrow_len*2-circle_dia-circle_dia/2
x_arrow-arrow_len*2-circle_dia-circle_dia/2],[y_arrow-3*delay_line_len/2-
box_width-circle_dia/4 y_arrow-3*delay_line_len/2-box_width+circle_dia/4]
);
        annotation('line', [x_arrow-arrow_len*2-circle_dia-circle_dia/4
x_arrow-arrow_len*2-circle_dia-3*circle_dia/4],[y_arrow-3*delay_line_len/2-
box_width y_arrow-3*delay_line_len/2-box_width] );
    end
    %line out of adder
    annotation('arrow', [x_arrow-arrow_len*2-circle_dia-circle_dia/2
x_arrow-arrow_len*2-circle_dia-circle_dia/2],[y_arrow-3*delay_line_len/2-
box_width+circle_dia/2-line_extra y_arrow-delay_line_len/2-circle_dia/2]);

    y_arrow = y_arrow+y_increment;
end
%end
```

All programs RUN properly in MATLAB14a