`

DATE: 30/07/2024

NAME: ARUNKARTHICK R
REG NO: 412522106013

Expt. No. 1 GENERATION OF SEQUENCES

AIM:

To write a program to generate different waveforms using MATLAB

SOFTWARE REQUIRED:

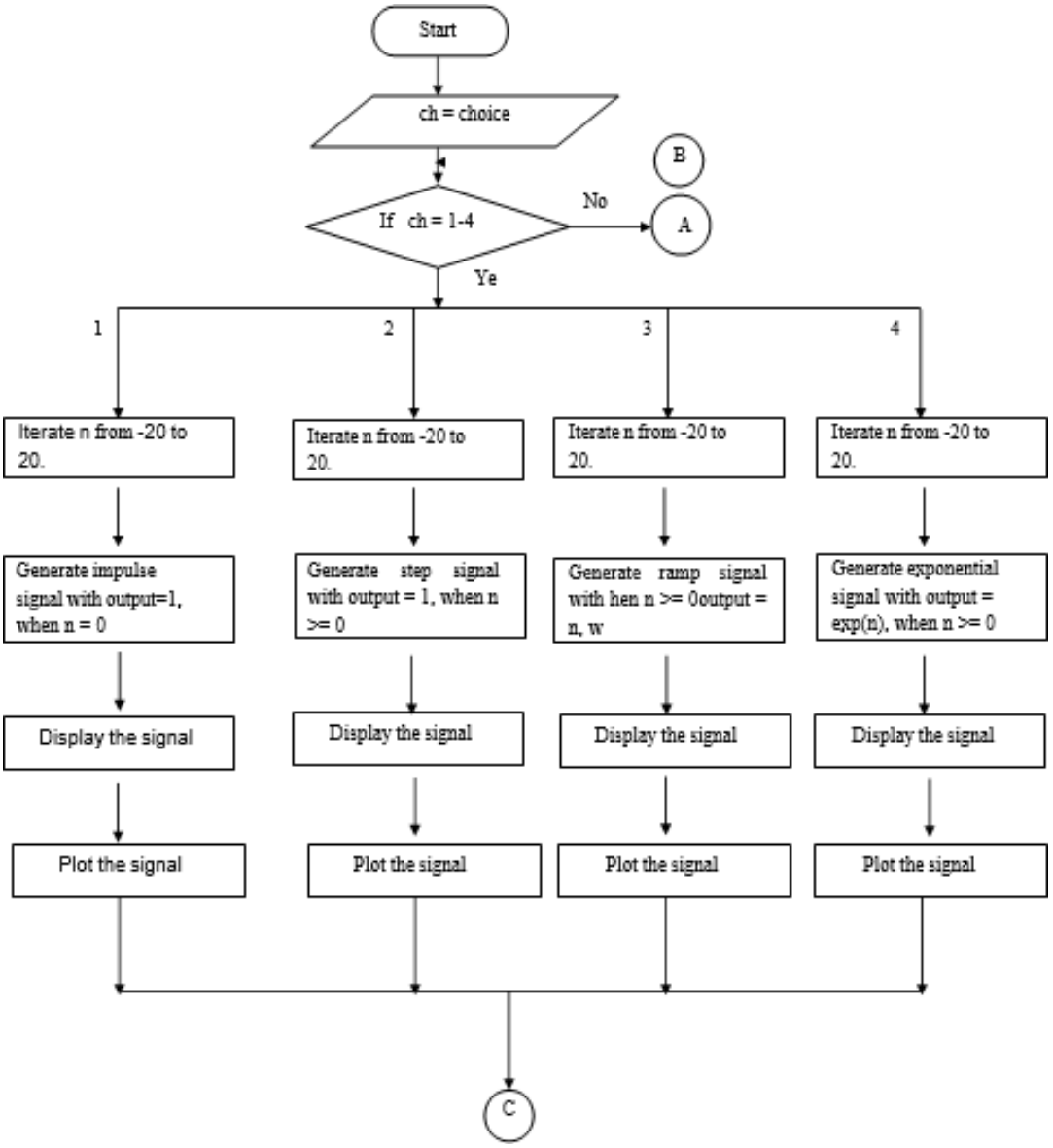
MATLAB Software

ALGORITHM:

1. Clear command window.
2. Get the choice from user to select the waveform to be generated.
3. Use switch case to execute the code for different waveforms.
4. Generate
   - i) Unit impulse: Iterate n from -20 to 20. Generate output=1 when n = 0.
   - ii) Unit step: Iterate n from -20 to 20. Generate output=1 when n >= 0.
   - iii) Ramp: Iterate n from -20 to 20. Generate output=n when n >= 0.
   - iv) Exponential: Iterate n from -20 to 20. Generate output=exp(n) when n >= 0.
   - v) Sine: Iterate n from 0 to 4*pi. Generate output = sin(n)
   - vi) Cosine: Iterate n from 0 to 4*pi. Generate output = cos(n)
   - vii) Triangular: Iterate n from 0 to 20 in steps of 0.2. Generate output=sawtooth(n,0.5)
   - viii) Sawtooth: Iterate n from 0 to 20 in steps of 0.2. Generate output=sawtooth(n,1)
5. Plot the signal.
6. Get the input from user if another waveform needs to be generated.
7. If yes, jump to Step 4, else terminate the program.

`

FLOWCHART:

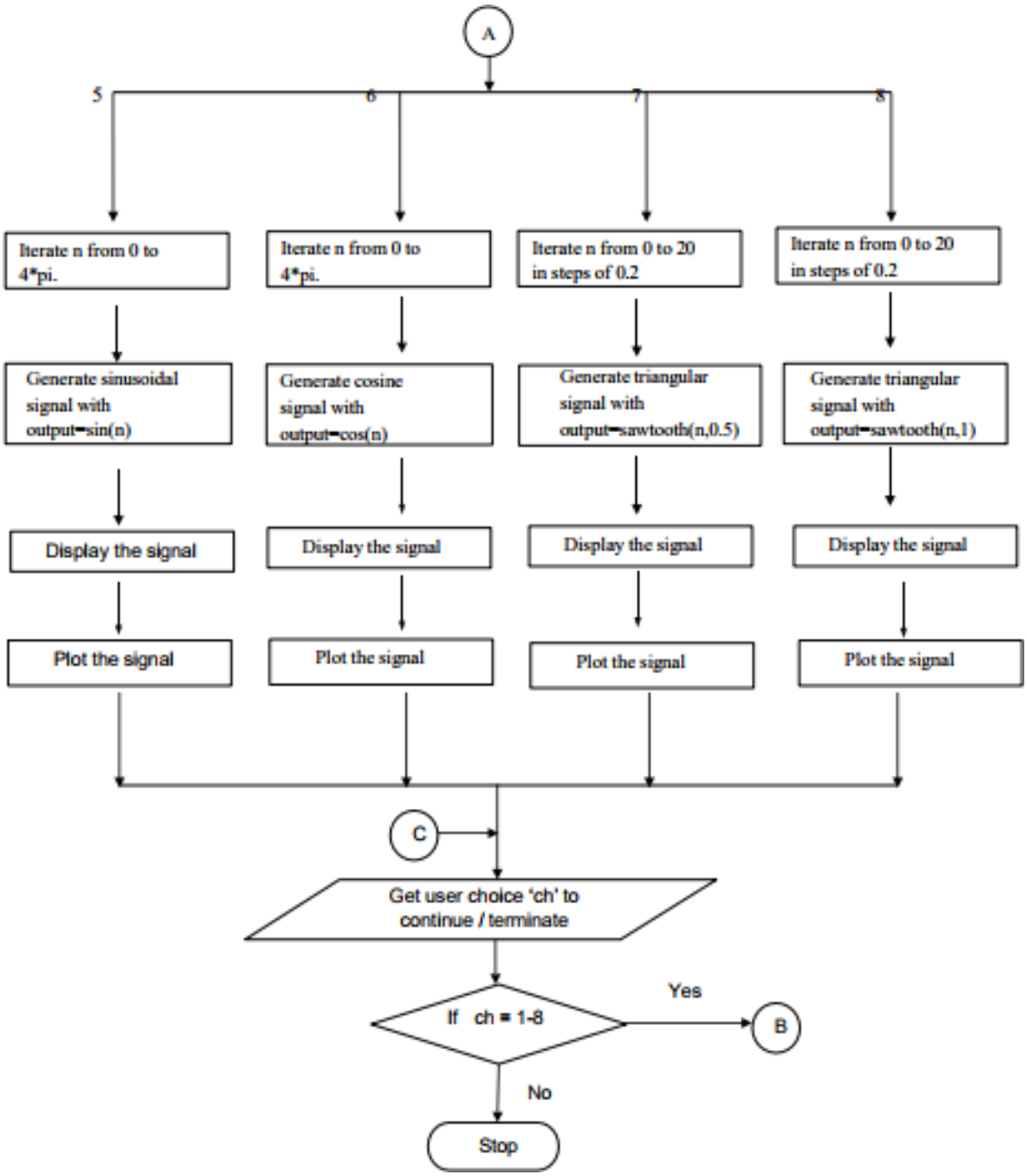

Start
ch = choice
B
If ch = 1-4
No
A
Ye
1
2
3
4
Iterate n from -20 to 20.
Iterate n from -20 to 20.
Iterate n from -20 to 20.
Iterate n from -20 to 20.
Generate impulse signal with output=1, when n = 0
Generate step signal with output = 1, when n >= 0
Generate ramp signal with hen n >= 0output = n, w
Generate exponential signal with output = exp(n), when n >= 0
Display the signal
Display the signal
Display the signal
Display the signal
Plot the signal
Plot the signal
Plot the signal
Plot the signal
C

`

PROGRAM:

```
clc
clear all
close all
disp('Program for Waveform generation');
opt=1;
while(opt==1)
        disp('Which waveform you want to generate?');
disp('1.Impulse,2.Step,3.Ramp,4.Exponential,5.Sine,6.Cosine,7.Triangle,8.Sawtooth,9.Random Signal');
k=input('ENTER YOUR CHOICE:');
switch k
%IMPULSE WAVEFORM
case 1
n = -20:1:20
for k=1:1:length(n)
if(n(k)==0)
x(k)=1;
else
x(k)=0;
end
end
% disp(x);
subplot(5,2,1)
stem(n,x);
xlabel('n -->');
ylabel('amplitude');
title('UNIT IMPULSE SIGNAL');
%STEP WAVEFORM
case 2
n =-20:1:20;
for k=1:1:length(n)
if(n(k)>=0)
x(k)=1;
else
x(k)=0;
end
end
% disp(x);
subplot(5,2,2)
stem(n,x);
xlabel('n -->');
ylabel('amplitude');
title('UNIT STEP SIGNAL');
%RAMP WAVEFORM
```

`


A
5
6
7
8
Iterate n from 0 to 4*pi.
Iterate n from 0 to 4*pi.
Iterate n from 0 to 20 in steps of 0.2
Iterate n from 0 to 20 in steps of 0.2
Generate sinusoidal signal with output=sin(n)
Generate cosine signal with output=cos(n)
Generate triangular signal with output=sawtooth(n,0.5)
Generate triangular signal with output=sawtooth(n,1)
Display the signal
Display the signal
Display the signal
Display the signal
Plot the signal
Plot the signal
Plot the signal
Plot the signal
C
Get user choice 'ch' to continue / terminate
If ch = 1-8
Yes
B
No
Stop

`

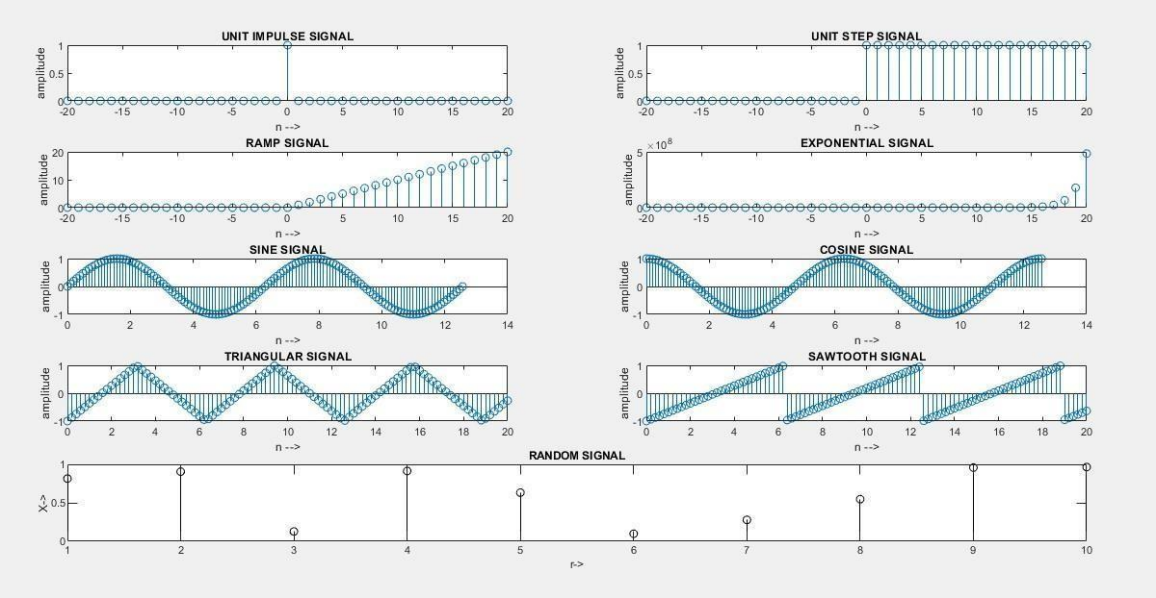
```
case 3
n = -20:1:20;
for k=1:1:length(n)
if(n(k)>=0)
x(k)=n(k);
else
x(k)=0;
end
end
% disp(x)
subplot(5,2,3)
stem(n,x);
xlabel('n -->');
ylabel('amplitude');
title('RAMP SIGNAL');
%EXPONENTIAL WAVEFORM
case 4
n = -20:1:20;
for k=1:1:length(n)
if(n(k)>=0)
x(k)=exp(n(k));
else
x(k)=0;
end
end
% disp(x);
subplot(5,2,4)
stem(n,x);
xlabel('n -->');
ylabel('amplitude');
title('EXPONENTIAL SIGNAL');
%SINE WAVEFORM
case 5
n = 0:(pi/32):(4*pi);
x = sin(n);
% disp(x);
subplot(5,2,5)
stem(n,x);
xlabel('n -->');
ylabel('amplitude');
title('SINE SIGNAL');
%COSINE WAVEFORM
case 6
n = 0:(pi/32):(4*pi);
x = cos(n);
```

`

Command prompt:

```
Program for Waveform generation
Which waveform you want to generate?
1.Impulse,2.Step,3.Ramp,4.Exponential,5.Sine,6.Cosine,7.Triangle,8.Sawtooth,9.Random Signal
ENTER YOUR CHOICE:1
```

OUTPUT GRAPH:



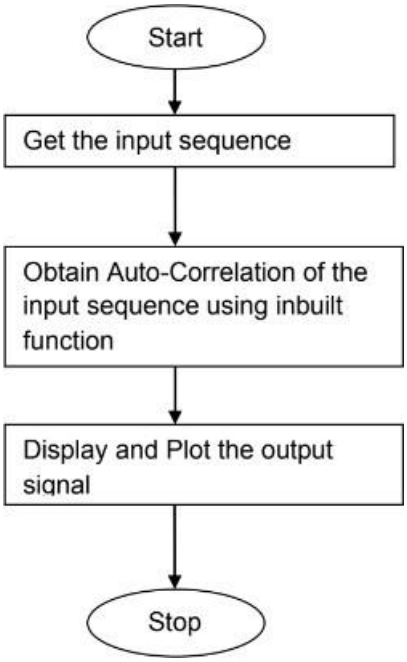
```
% disp(x);
subplot(5,2,6)
stem(n,x);
xlabel('n -->');
ylabel('amplitude');
title('COSINE SIGNAL');
%TRIANGULAR WAVEFORM
case 7
n = 0:0.2:20;
x =sawtooth(n,0.5);
% disp(x);
subplot(5,2,7)
stem(n,x);
xlabel('n -->');
ylabel('amplitude');
title('TRIANGULAR SIGNAL');
%SAWTOOTH WAVEFORM
case 8
n = 0:0.2:20;
x=sawtooth(n,1);
% disp(x);
subplot(5,2,8)
stem(n,x);
xlabel('n -->');
ylabel('amplitude');
title('SAWTOOTH SIGNAL');
%RANDOM SIGNAL
case 9
r=10;
x=rand(r,1);
% disp(x);
subplot(5,2,[9 10])
stem(x,'k')
xlabel('r->')
ylabel('X->')
title('RANDOM SIGNAL');
otherwise
disp('INVALID CHOICE');
end
disp('Do you want to continue?');
opt=input('If YES, press 1:');
end
```

RESULT:

Thus, the program to generate different waveforms using MATLAB is executed and the outputs are verified.

`

FLOWCHART:


Start
Get the input sequence
Obtain Auto-Correlation of the input sequence using inbuilt function
Display and Plot the output signal
Stop

`

DATE: 30/07/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

Expt. No. 2a. AUTO CORRELATION

AIM:

To write a program to obtain auto correlation of the given sequence using MATLAB.

SOFTWARE REQUIRED:

MATLAB Software

ALGORITHM:

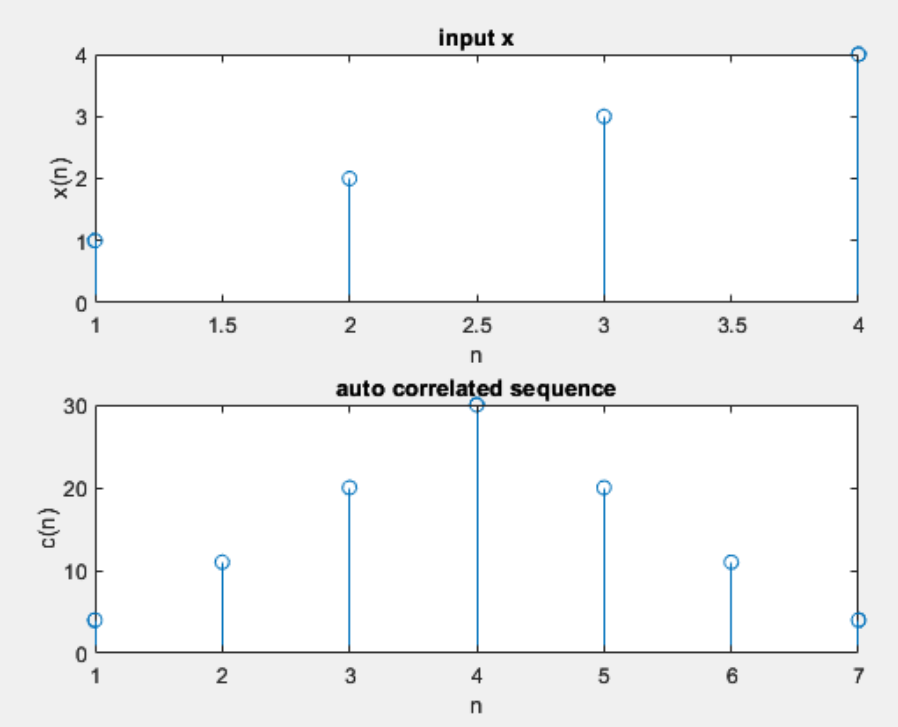
1. Start the program.
2. Give the input sequence.
3. Obtain the autocorrelation of the input sequence using the built in function, xcorr(x,x).
4. Display and plot the output.
5. Terminate the program.

`

COMMAND PROMPT:

```
enter the input sequence x[1 2 3 4]

x =

     1     2     3     4


c =

    4.0000   11.0000   20.0000   30.0000   20.0000   11.0000    4.0000

auto correlated sequence
    4.0000   11.0000   20.0000   30.0000   20.0000   11.0000    4.0000
```

OUTPUT GRAPH:



`

PROGRAM:

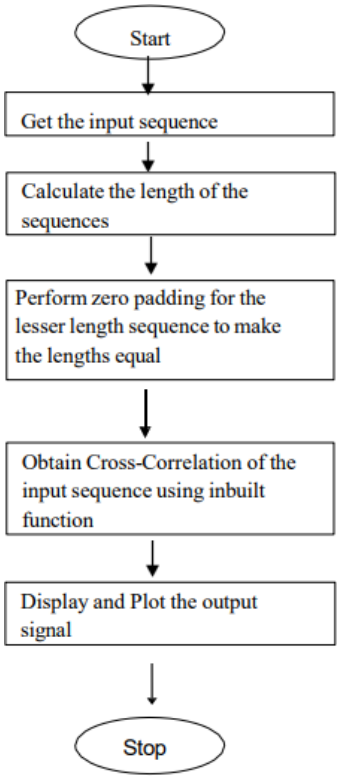
```
clc
clear all
close all
x=input('enter the input sequence x')
c=xcorr(x,x) %correlation using the function 'xcorr'
subplot(2,1,1)
stem(x)
xlabel('n')
ylabel('x(n)')
title('input x')
disp('auto correlated sequence')
disp(c)
subplot(2,1,2)
stem(c)
xlabel('n')
ylabel('c(n)')
title('auto correlated sequence')
```

RESULT:

Thus, the program to find the auto correlation of the given sequences using MATLAB is executed and the output is verified.

`

FLOWCHART:


Start
Get the input sequence
Calculate the length of the sequences
Perform zero padding for the lesser length sequence to make the lengths equal
Obtain Cross-Correlation of the input sequence using inbuilt function
Display and Plot the output signal
Stop

`

DATE:30/07/2024

NAME: ARUNKARTHICK R
REG.NO: 412522106013

Expt. No. 2b. CROSS CORRELATION

AIM:

To write a program to obtain cross correlation of the given sequences using MATLAB.

SOFTWARE REQUIRED:

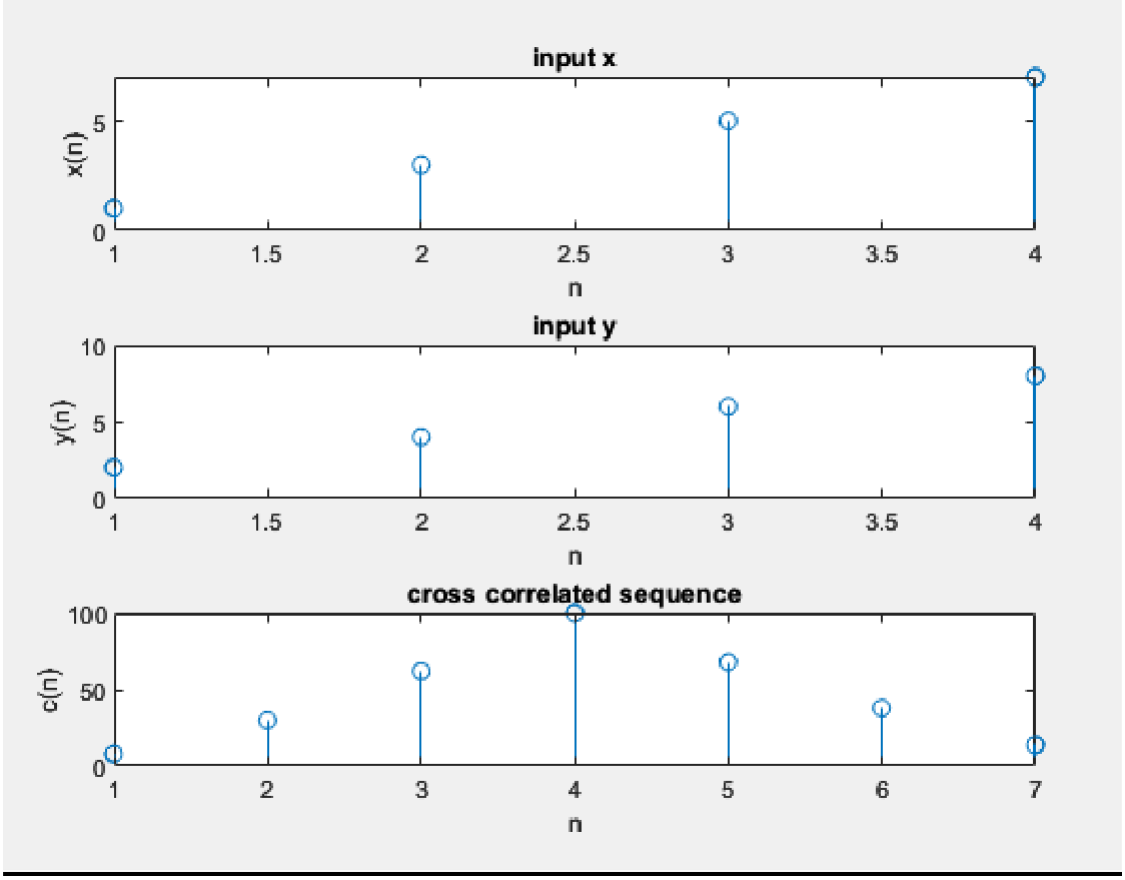
MATLAB Software

ALGORITHM:

1. Start the program.
2. Give the two input sequences.
3. Obtain the autocorrelation of the input sequence using the built in function, xcorr(x,x).
4. Display and plot the output.
5. Terminate the program.

`

OUTPUT GRAPH:



COMMAND PROMPT:

```
enter the input sequence x[1 3 5 7]
enter the input sequence y[2 4 6 8]
cross correlated sequence
    8.0000   30.0000   62.0000  100.0000   68.0000   38.0000   14.0000
```

`

PROGRAM:

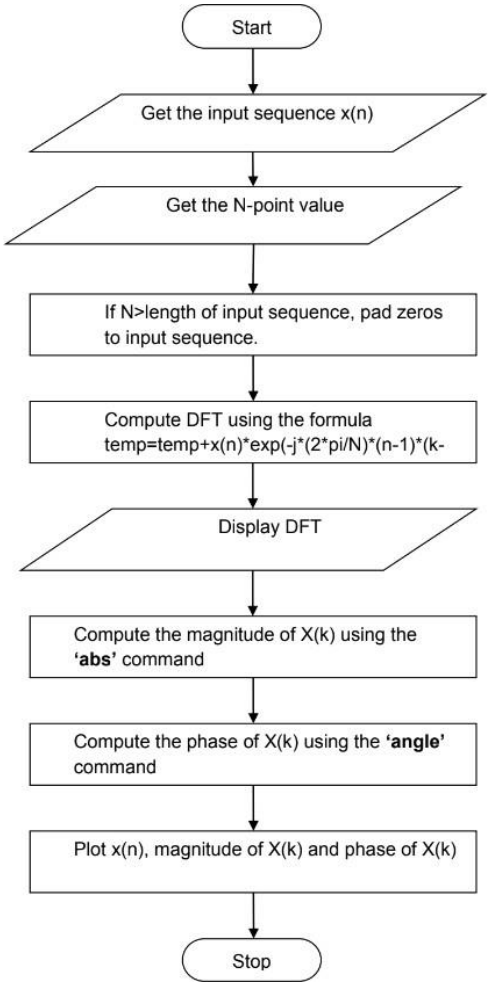
```
clc
clear all
close all
x=input('enter the input sequence x');
y=input('enter the input sequence y');
m=length(x); %length of x
n=length(y); %length of x
if (m-n) ~= 0
if m>n
y=[y zeros(1,(m-n))]; %append m-n number of zeros to the sequence ‘y’
n=m;
else
x=[x zeros(1,(n-m))]; %append n-m number of zeros to the sequence ‘x’
m=n;
end
end
c=xcorr(x,y); %correlation using the function ‘xcorr’
subplot(3,1,1)
stem(x)
xlabel('n')
ylabel('x(n)')
title('input x')
subplot(3,1,2)
stem(y)
xlabel('n')
ylabel('y(n)')
title('input y')
disp('cross correlated sequence')
disp(c)
subplot(3,1,3)
stem(c)
xlabel('n')
ylabel('c(n)')
title('cross correlated sequence
```

RESULT:

Thus, the program to find the cross correlation of the given sequences using MATLAB is executed and the output is verified.

`

FLOWCHART:


Start
Get the input sequence x(n)
Get the N-point value
If N>length of input sequence, pad zeros to input sequence.
Compute DFT using the formula temp=temp+x(n)*exp(-j*(2*pi/N)*(n-1)*(k-
Display DFT
Compute the magnitude of X(k) using the 'abs' command
Compute the phase of X(k) using the 'angle' command
Plot x(n), magnitude of X(k) and phase of X(k)
Stop

`

DATE:06/08/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

Expt. No. 3a. DFT & IDFT

AIM:

To write a program to find the Discrete Fourier Transform of the given sequence using MATLAB and plot the magnitude and phase response.

SOFTWARE REQUIRED:

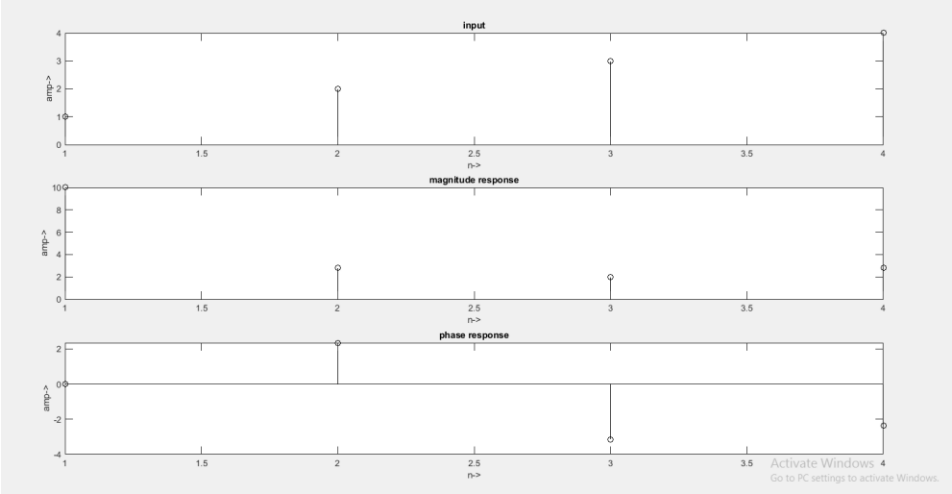
MATLAB Software

ALGORITHM:

1. Clear the command window.
2. Get the input sequence x(n).
3. Get the N-point value.
4. If N>length of input sequence, pad zeros to input sequence.
5. For each value of X(k), compute temp=temp+x(n)*exp(-j*(2*pi/N)*(n-1)*(k-1))
6. Display DFT of the input sequence
7. Compute the magnitude of X(k) using the command mag_dft=abs(x_dft)
8. Compute the phase of X(k) using the command phase_dft=angle(x_dft)
9. Plot the input sequence, magnitude of X(k), and phase of X(k) in a single window.

`

OUTPUT GRAPH:



`

PROGRAM:

```
clc;
clear all;
close all;
x=input('enter the sequence');
N=input('enter the length')
if(N>length(x))
x=[x zeros(1,(N-length(x)))]
end
for k=1:1:N
X(k)=0;
for n=1:1:length(x)
X(k)=X(k)+x(n)*exp(-j*(2*pi/N)*(n-1)*(k-1));
end
disp(X(k))
end
subplot(3,1,1)
stem(x,'k')
xlabel('n->')
ylabel('amp->')
title('input')
mag_X=abs(X)
subplot(3,1,2)
stem(mag_X,'k')
xlabel('n->')
ylabel('amp->')
title('magnitude response')
phase_X=angle(X)
subplot(3,1,3)
stem(phase_X,'k')
xlabel('n->')
ylabel('amp->')
title('phase response')
for n=1:1:N
y(n)=0;
for k=1:1:length(X)
y(n)=y(n)+(1/N)*X(k)*exp(j*(2*pi/N)*(n-1)*(k-1));
end
disp(y(n))
end
subplot(3,1,1)
stem(x,'k')
xlabel('n->')
ylabel('amp->')
title('input')
```

`

COMMAND PROMPT:

```
enter the sequence [ 1 2 3 4]
enter the length4
    10

  -2.0000 + 2.0000i

  -2.0000 - 0.0000i

  -2.0000 - 2.0000i

   1.0000 - 0.0000i

   2.0000 - 0.0000i

   3.0000 - 0.0000i

   4.0000 + 0.0000i
```

`

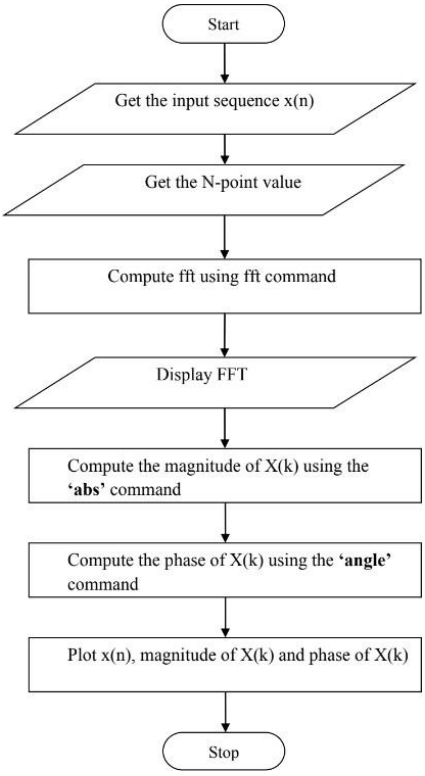
```
mag_X=abs(X)
subplot(3,1,2)
stem(mag_X,'k')
xlabel('n->')
ylabel('amp->')
title('magnitude response')
phase_X=angle(X)
subplot(3,1,3)
stem(phase_X,'k')
xlabel('n->')
ylabel('amp->')
title('phase response')
for n=1:1:N
y(n)=0;
for k=1:1:length(X)
y(n)=y(n)+(1/N)*X(k)*exp(j*(2*pi/N)*(n-1)*(k-1));
end
disp(y(n))
end
```

RESULT:

Thus, the program to find the Discrete Fourier Transform and IDFT of the given sequence using MATLAB is executed and the output is verified.

`

FLOWCHART:


Start
Get the input sequence x(n)
Get the N-point value
Compute fft using fft command
Display FFT
Compute the magnitude of X(k) using the 'abs' command
Compute the phase of X(k) using the 'angle' command
Plot x(n), magnitude of X(k) and phase of X(k)
Stop

`

DATE:06/08/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

Expt. No. 3b. FFT and IFFT

AIM:

To write a program to find the FFT of the given sequence using MATLAB and plot the magnitude and phase response.

SOFTWARE REQUIRED:

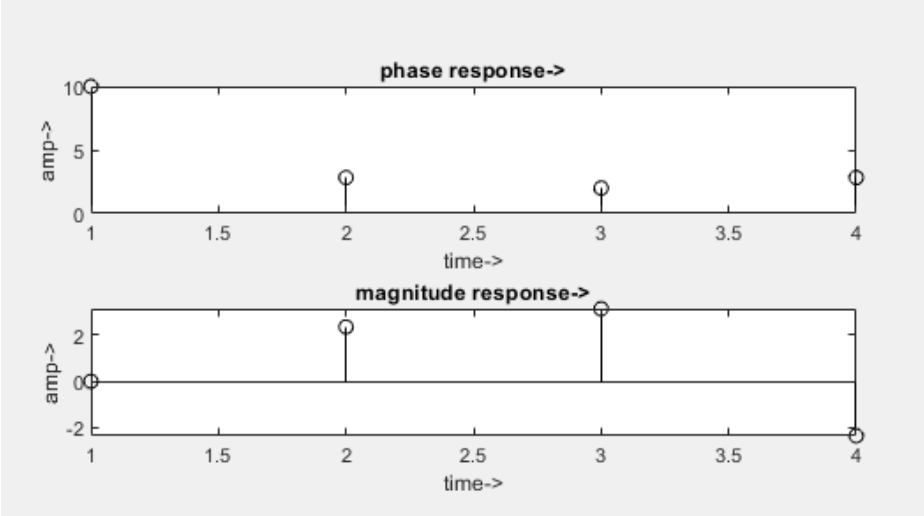
MATLAB Software

ALGORITHM:

1. Clear command window.
2. Get the input sequence x(n).
3. Get the N-point value.
4. Compute FFT using x_fft=fft(x,n).
5. Display FFT of the input sequence
6. Compute the magnitude of X(k) using the command mag_fft=abs(x_fft)
7. Compute the phase of X(k) using the command phase_fft=angle(x_fft)
8. Plot the input sequence, magnitude of X(k), and phase of X(k) in a single window.

`

OUTPUT GRAPH:



COMMAND PROMPT:

```
enter the sequence[1 2 3 4]
enter the length4

N =

     4


X =

  10.0000 + 0.0000i  -2.0000 + 2.0000i  -2.0000 + 0.0000i  -2.0000 - 2.0000i


mag_X =

   10.0000    2.8284    2.0000    2.8284


phase_X =

         0    2.3562    3.1416   -2.3562


y =

     1     2     3     4
```

`

PROGRAM:

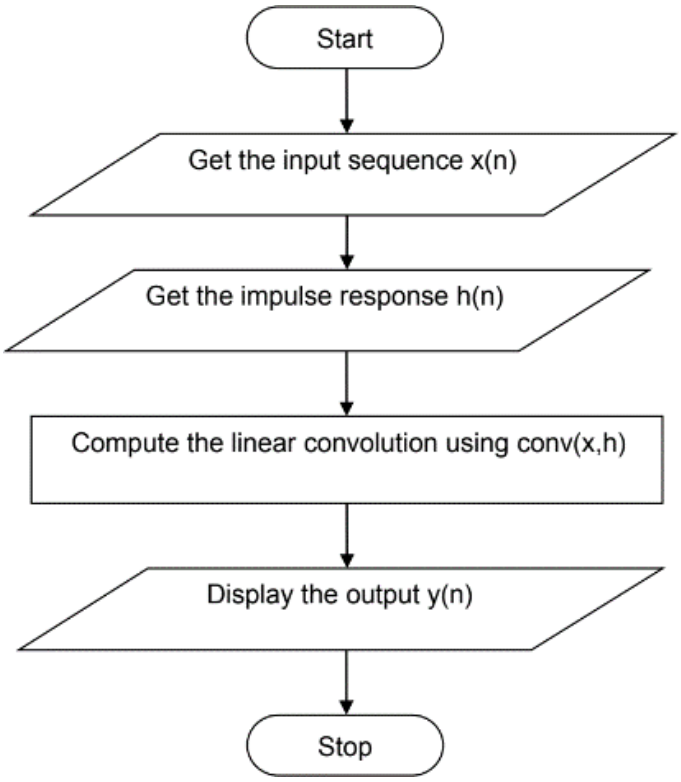
```
clc
clear all
close all
x=input('enter the sequence')
N=input('enter the length
X=fft(x)
functionsubplot(3,1,1)
stem(x,'k')
xlabel('time->')
ylabel('amp->')
title('input->')
mag_X=abs(X)
subplot(3,1,2)
stem(mag_X,'k')
xlabel('time->')
ylabel('amp->')
title('phase response->')
phase_X=angle(X)
subplot(3,1,3)
stem(phase_X,'k')
xlabel('time->')
ylabel('amp->')
title('magnitude response->')
y=ifft(X)
```

RESULT:

Thus, the program to find the FFT and IFFT of the given sequence using MATLAB is executed and the output is verified.

`

FLOWCHART:


Start
Get the input sequence x(n)
Get the impulse response h(n)
Compute the linear convolution using conv(x,h)
Display the output y(n)
Stop

`

DATE:06/08/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

Expt. No. 4a Linear Convolution

AIM:

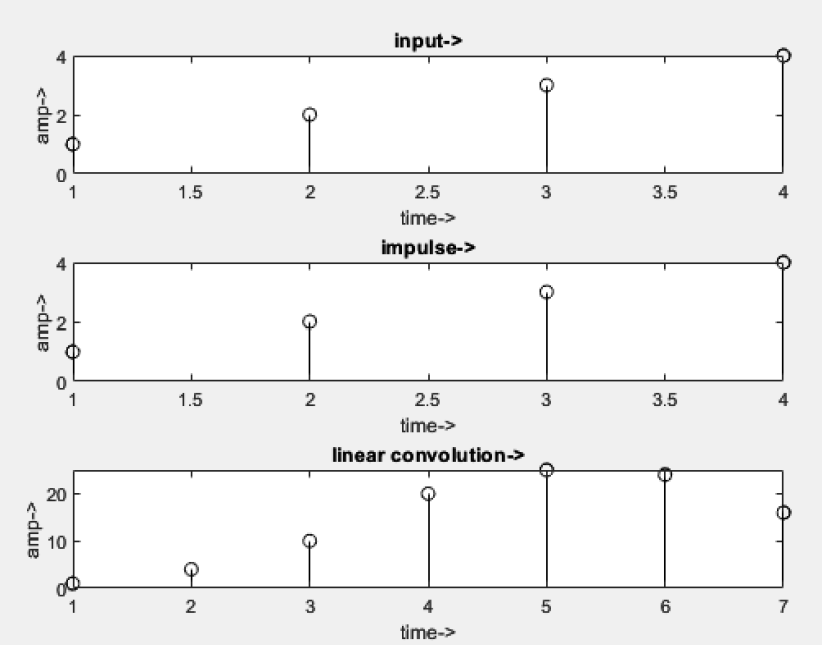
To compute linear convolution of two sequences using in-built function in MATLAB.

ALGORITHM:

1. Clear command window.
2. Get the input sequence x(n).
3. Get the impulse response h(n).
4. Compute the linear convolution using conv(x,h) command.
5. Display the output.
6. Plot the input sequence, impulse response and output sequence in a single window.

`

OUTPUT GRAPH:



COMMAND PROMPT:

```
enter the input sequence [ 1 2 3 4]

x =

     1     2     3     4

eneter the impulse response [ 1 2 3 4]

h =

     1     2     3     4


l =

     7


y =

     1     4    10    20    25    24    16
```

`

PROGRAM

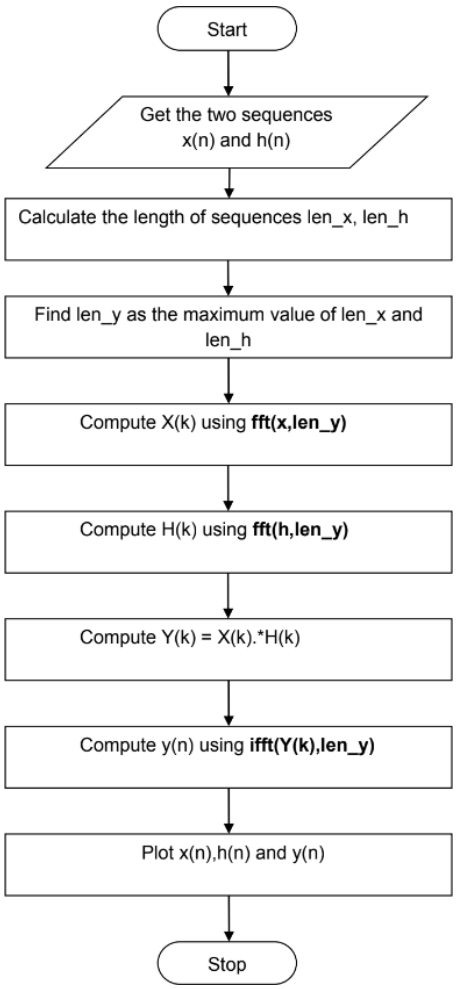
```
x=input('enter the input sequence')
h=input('eneter the impulse response')
l=length(x)+length(h)-1
y=convx,h) subplot(3,1,1) stem(x,'k')
xlabel('time->')
ylabel('amp->')
title('input->') subplot(3,1,2) stem(h,'k')
xlabel('time->')
ylabel('amp->') title('impulse->') subplot(3,1,3) stem(y,'k')
xlabel('time->')
ylabel('amp->')
title('linear convolution->')
```

RESULT:

Thus, the program to find the linear convolution of two sequences using in-built function in MATLAB is executed and the output is verified

`

FLOWCHART:


Start
Get the two sequences x(n) and h(n)
Calculate the length of sequences len_x, len_h
Find len_y as the maximum value of len_x and len_h
Compute X(k) using fft(x,len_y)
Compute H(k) using fft(h,len_y)
Compute Y(k) = X(k).*H(k)
Compute y(n) using ifft(Y(k),len_y)
Plot x(n),h(n) and y(n)
Stop

`

DATE:06/08/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

Expt. No. 4b Circular convolution using FFT

AIM:

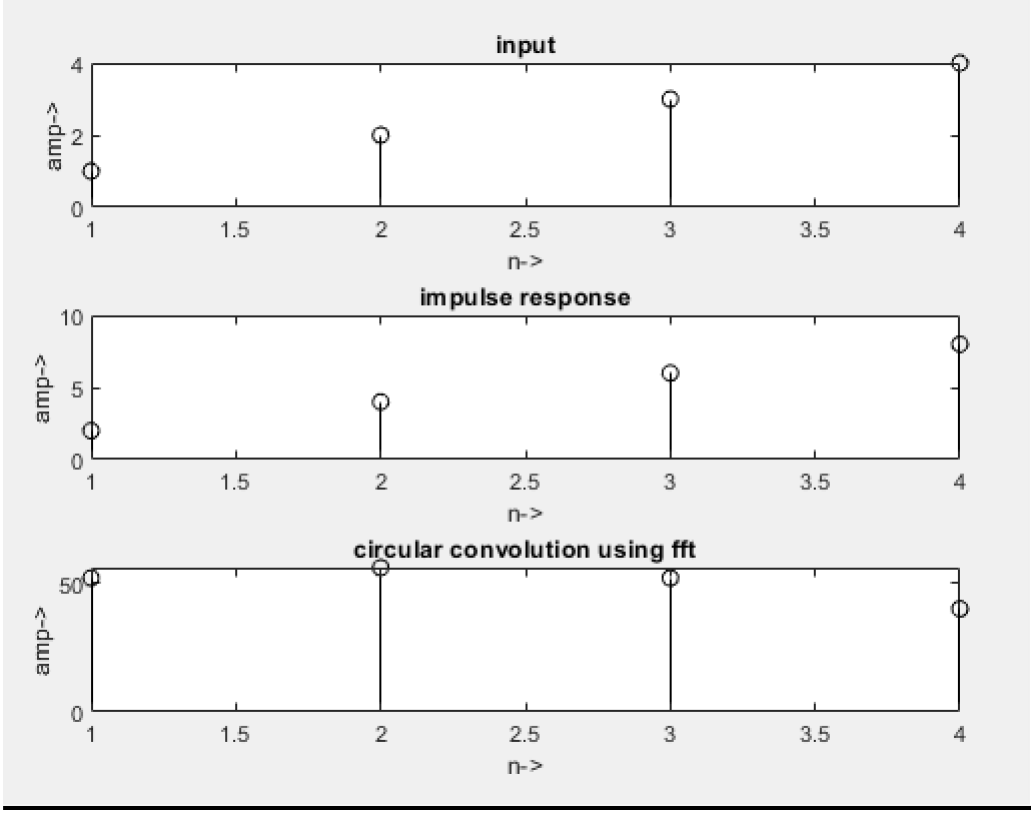
To compute circular convolution of two sequences using FFT in MATLAB.

ALGORITHM:

1. Clear command window.
2. Get the two sequences x(n) and h(n) from user.
3. Calculate the length of sequences len_x, len_h and find the maximum value.
4. Compute X(k) using **fft** command by specifying fft length as maximum length of x and h.
5. Compute H(k) using **fft** command by specifying fft length as maximum length of x and h.
6. Multiply the two fft sequences element by element and store in y_fft .
7. Calculate inverse FFT of y_fft using **ifft** command and store it in y.
8. Display the output.
9. Plot the two input sequences and the output sequence.

`

OUTPUT GRAPH:



COMMAND PROMPT:

```
enter then input sequence[1 2 3 4]
enter the impulse response[2 4 6 8]
    52    56    52    40
```

`

PROGRAM:

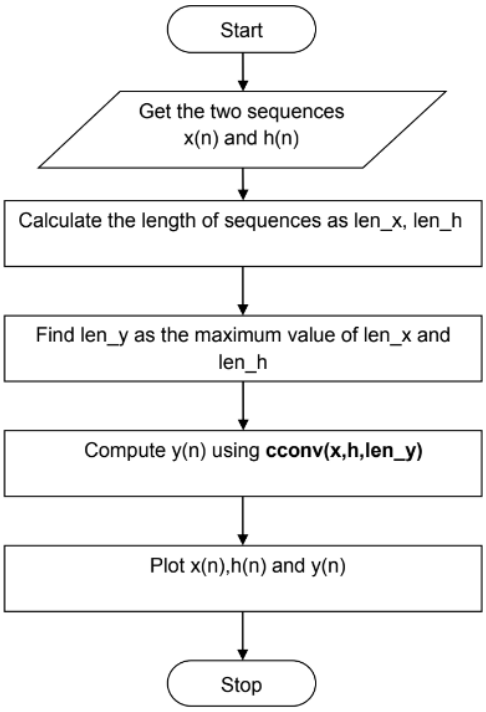
```
clc
clear all
close all
x=input('enter then input sequence');
h=input('enter the impulse response');
l1=length(x);
l2=length(h);
l3=max(l1,l2);
X=fft(x);
H=fft(h);
for i=1:1:l3
Y(i)=X(i)*H(i);
end
y=ifft(Y);
disp(y)
subplot(3,1,1)
stem(x,'k')
xlabel('n->')
ylabel('amp->')
title('input')
subplot(3,1,2)
stem(h,'k')
xlabel('n->')
ylabel('amp->')
title('impulse response')
subplot(3,1,3) stem(y,'k')
xlabel('n->')
ylabel('amp->')
title('circular convolution using fft')
```

RESULT:

Thus, the program to find the circular convolution of two sequences using FFT in MATLAB isexecuted and the output is verified

`

FLOWCHART:


Start
Get the two sequences x(n) and h(n)
Calculate the length of sequences as len_x, len_h
Find len_y as the maximum value of len_x and len_h
Compute y(n) using **cconv(x,h,len_y)**
Plot x(n),h(n) and y(n)
Stop

`

DATE:06/08/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

Expt. No. 4c Circular Convolution using in-built function

AIM:

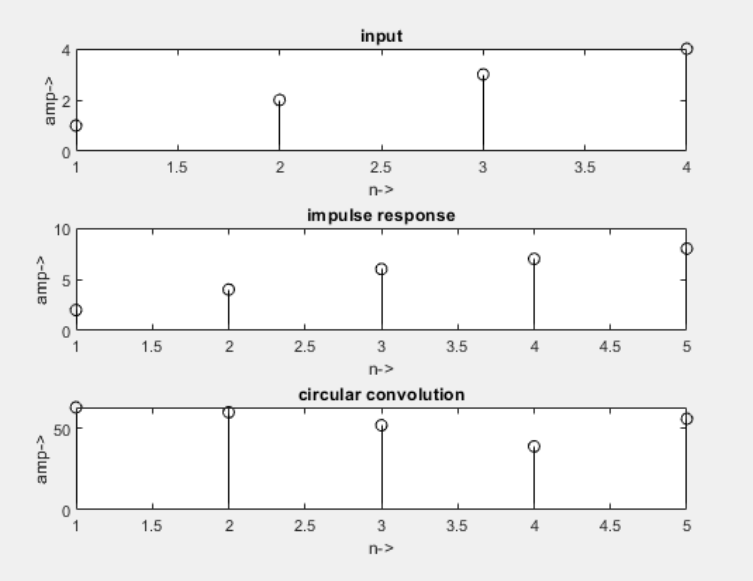
To compute the circular convolution of two sequences using in-built function in MATLAB.

ALGORITHM:

1. Clear command window.
2. Get the two sequences x(n) and h(n) from user.
3. Calculate the length of sequences len_x, len_h and find the maximum value.
4. Perform the circular convolution using 'cconv' function by specifying output length asmaximum length of x and h.
5. Display the output.
6. Plot the two input sequences and the output sequence.

`

OUTPUT:



COMMAND PROPMT:

```
enter the input sequence[1 2 3 4]

x =

     1     2     3     4

enter the impulse response[2 4 6 7 8]

i =

     2     4     6     7     8

l1 =

     4

l2 =

     5

l3 =

     5

y =

    63    60    52    39    56
```

`

PROGRAM:

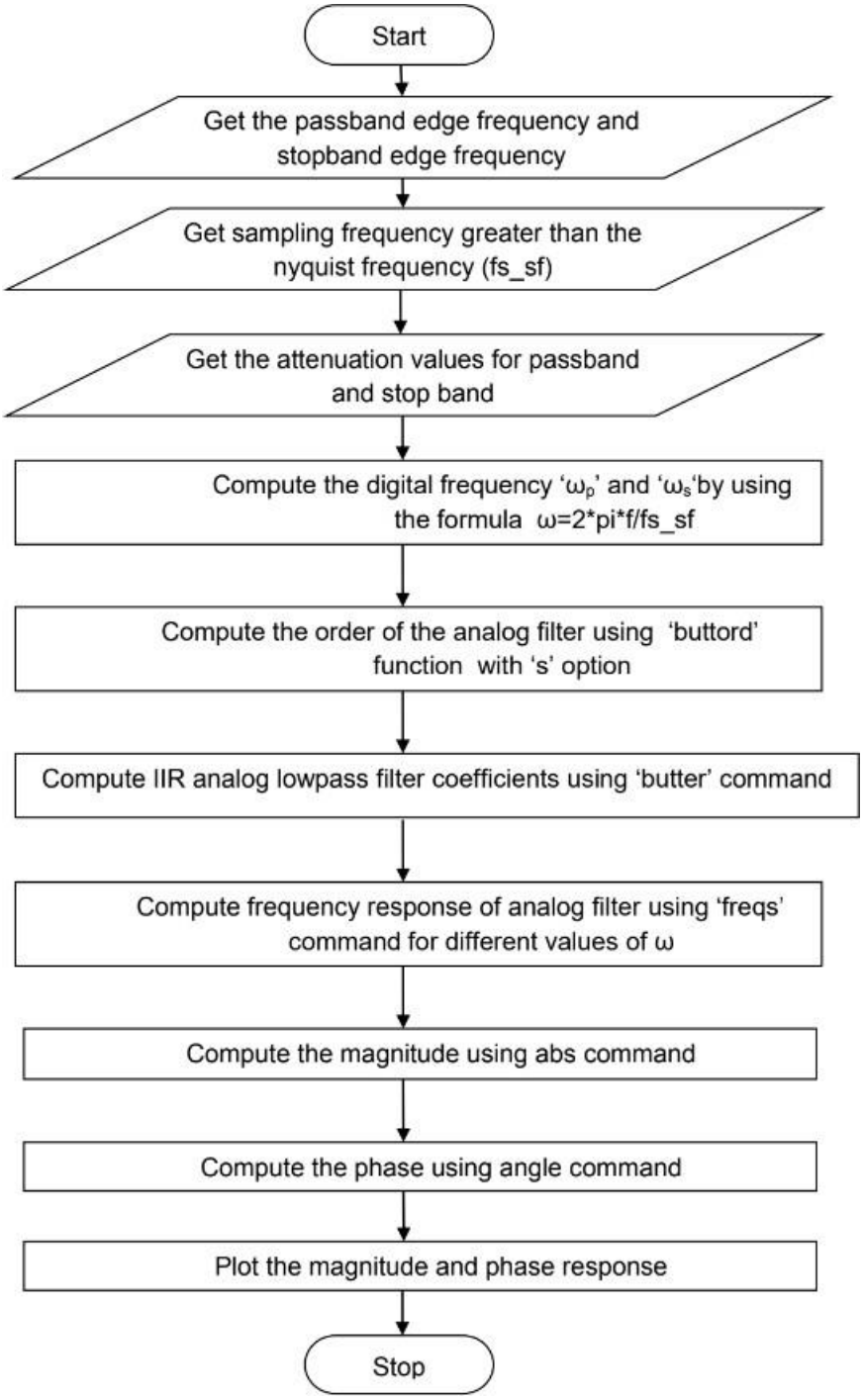
```
clc
clear all
close all
x=input('enter the input sequence')
i=input('enter the impulse response')
l1=length(x)
l2=length(i)
l3=max(l1,l2)
y=cconv(x,i,l3)
subplot(3,1,1)
stem(x,'k')
xlabel('n->')
ylabel('amp->')
title('input')
subplot(3,1,2)
stem(i,'k')
xlabel('n->')
ylabel('amp->')
title('impulse response')
subplot(3,1,3)
stem(y,'k')
xlabel('n->')
ylabel('amp->')
title('circular convolution')
```

RESULT:

Thus, the program to find the circular convolution of two sequences using using in-built function in MATLAB is executed and the output is verified

`

FLOWCHART:


Start
Get the passband edge frequency and stopband edge frequency
Get sampling frequency greater than the nyquist frequency (fs_sf)
Get the attenuation values for passband and stop band
Compute the digital frequency 'ωp' and 'ωs'by using the formula ω=2*pi*f/fs_sf
Compute the order of the analog filter using 'buttord' function with 's' option
Compute IIR analog lowpass filter coefficients using 'butter' command
Compute frequency response of analog filter using 'freqs' command for different values of ω
Compute the magnitude using abs command
Compute the phase using angle command
Plot the magnitude and phase response
Stop

`

DATE:13/08/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

Expt. No. 5a. DESIGN OF ANALOG IIR BUTTERWORTH LOW PASS FILTER

AIM:

To write a program in MATLAB to plot the magnitude and phase response of analog IIR Butterworth low pass filter.

SOFTWARE REQUIRED:

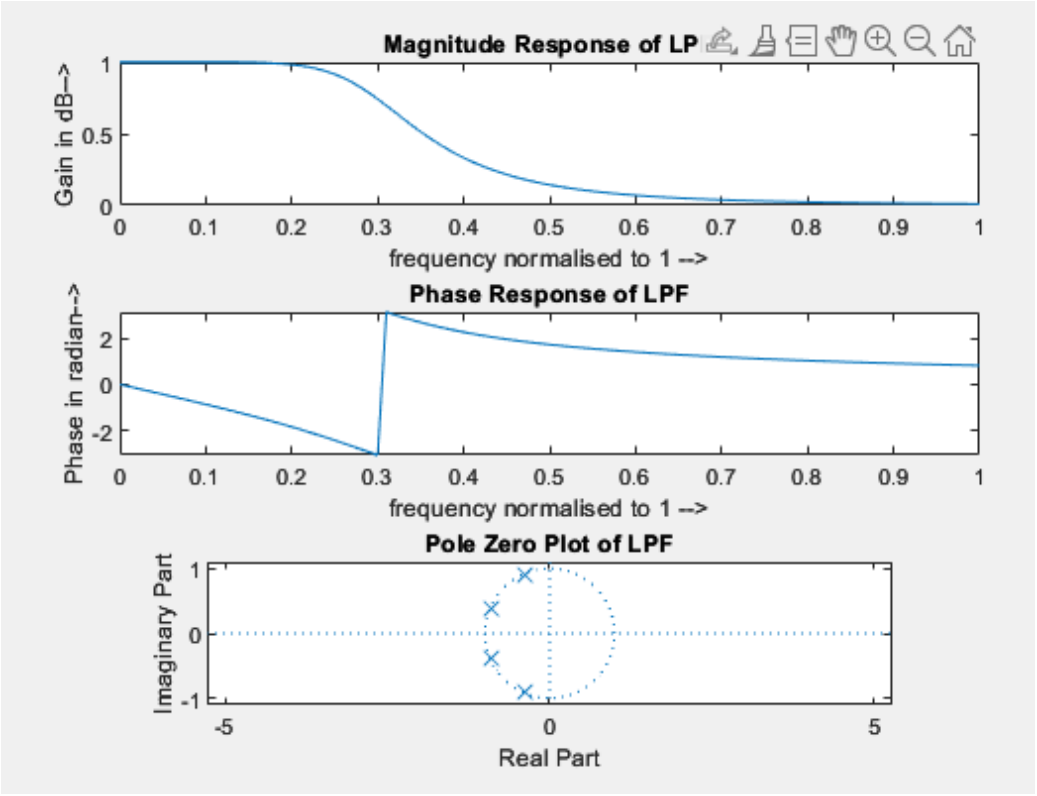
MATLAB Software

ALGORITHM:

1. Clear the command window.
2. Get the passband edge frequency and stopband edge frequency
3. Get sampling frequency greater than the nyquist frequency(fs_sf)
4. Get the attenuation values for passband and stop band.
5. Compute the digital frequency 'ωp' and 'ωs'by using the formula ω=2*pi*f/fs_sf
6. Compute the order of the analog filter using 'buttord' function with 's' option
7. Compute IIR analog lowpass filter coefficients using 'butter' command.
8. Compute frequency response of analog filter using 'freqs' command for different values of ω
9. Compute the magnitude using abs command.
10. Compute the phase using angle command.
11. Plot the magnitude and phase response.

`

OUTPUT:


Magnitude Response of LP
Gain in dB-->
frequency normalised to 1 -->
Phase Response of LPF
Phase in radian-->
frequency normalised to 1 -->
Pole Zero Plot of LPF
Imaginary Part
Real Part

`

PROGRAM:

```
%Program for Butterworth IIR Lowpass analog filter
clc;
close all;
clear all;
fprintf('Program for Butterworth IIR Lowpass analog filter\n\n');
%We get the passedge, stopedge and sampling frequencies in Hz
fp=input('Enter the pass edge frequency: ');
fs=input('Enter the stop edge frequency: ');
%fs_min should be twice the maximum frequency. Here, fs_min =
2*fs.
fs_min = 2*fs;
fprintf('\nEnter the sampling frequency greater than %d\n',fs_min');
fs_sf=input('Enter the sampling frequency: ');
%We get the attenuation in dB. rp will be around 0 to 3 dB
%rs will be around 30 to 50 dB
rp = input('\nEnter the passband ripple in dB: ');
rs = input('Enter the stopband attenuation in dB: ');
rp1=20*log10(rp)
rs1=20*log10(rs)

%We need to normalise wp,ws to pi. It is similar to finding the digital
%frequency; digital omega = analog omega* Ts = analog
omega/fs_sf
wp=2*pi*fp/fs_sf;
ws=2*pi*fs/fs_sf;
%The normalised frequencies are wp and ws
fprintf('\nwp is %d\n',wp);
fprintf('ws is %d\n',ws);
%Computing the order(N) and cutoff frequency(wc) using
wp,ws,rp,rs using
%the buttord command with 's' option for analog filter.
%Finding the coefficients of filter [b a] using butter command with 's'
%option
[N wc]=buttord(wp,ws,rp1,rs1,'s');
[b a]=butter(N,wc,'s');
%Computing the frequency response using freqs command
%Computing h for specific values of w i.e. 0, pi/100, 2*pi/100... till
pi
%and storing the corresponding the w values
w=0:(pi/100):pi;
[h omega] = freqs(b,a,w);
hf=tf(b,a)
```

`

COMMAND PROMPT:

```
Program for Butterworth IIR Lowpass analog filter

Enter the pass edge frequency: 1500
Enter the stop edge frequency: 3000

Enter the sampling frequency greater than 6000
Enter the sampling frequency: 7000

Enter the passband ripple in dB: 0.15
Enter the stopband attenuation in dB: 60

            wp is 1.346397e+00
            ws is 2.692794e+00
              tf with properties:

                  Numerator: {[0 0 0 0 0.8764]}
                Denominator: {[1 2.5284 3.1963 2.3670 0.8764]}
                   Variable: 's'
                    IODelay: 0
                 InputDelay: 0
                OutputDelay: 0
                  InputName: {''}
                  InputUnit: {''}
                 InputGroup: [1×1 struct]
                 OutputName: {''}
                 OutputUnit: {''}
                OutputGroup: [1×1 struct]
                      Notes: [0×1 string]
                   UserData: []
                       Name: ''
                         Ts: 0
                   TimeUnit: 'seconds'
                SamplingGrid: [1×1 struct]
```

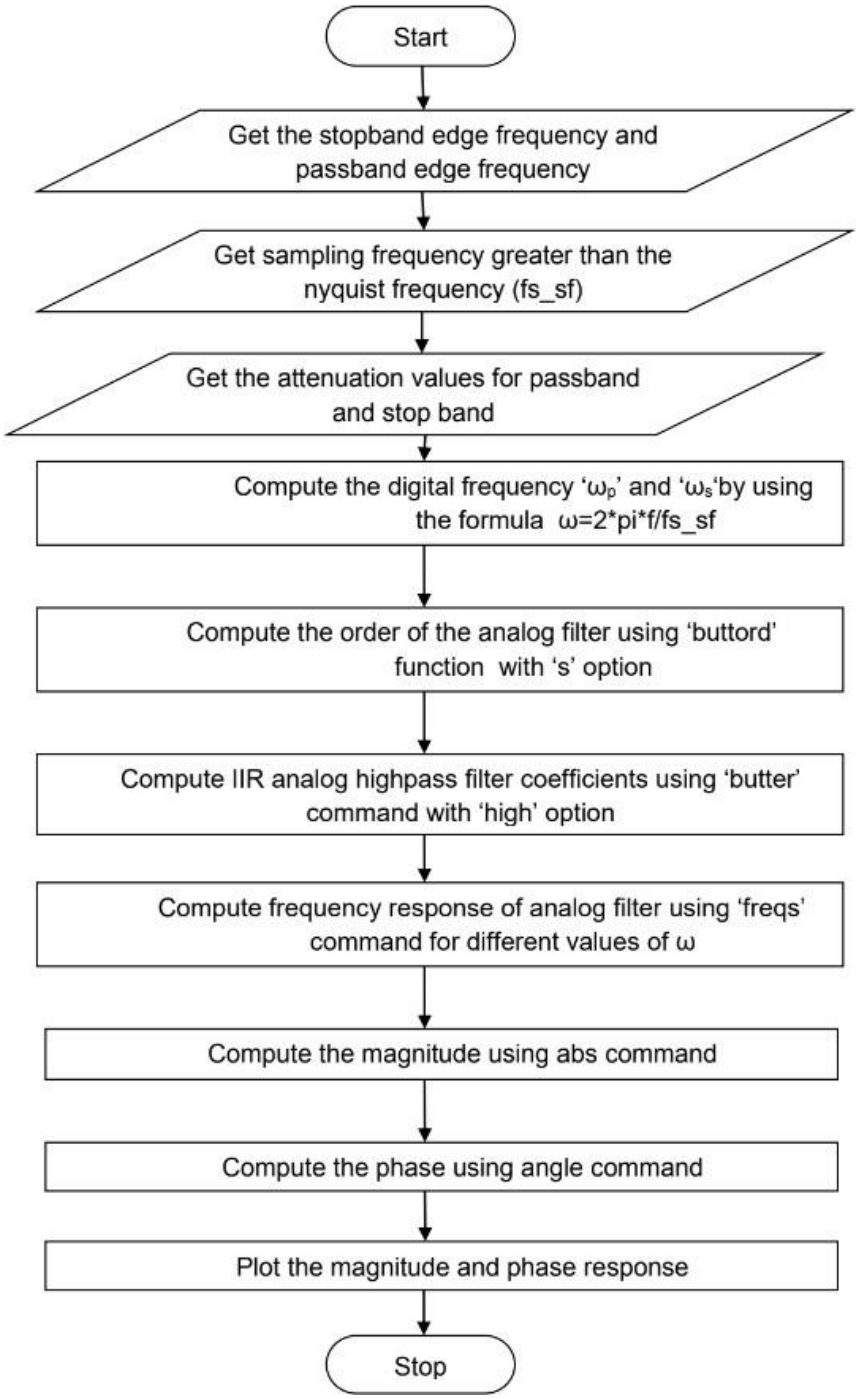
```
disp(hf)
[z p]=tf2zp(b,a)

%Finding the magnitude response. Note: log10 should be used.
%Plotting magnitude versus omega.
mag_h=abs(h);
figure(1);
subplot(3,1,1);
plot(omega/pi,mag_h);
xlabel('frequency normalised to 1 -->');

ylabel('Gain in dB-->');
title('Magnitude Response of LPF');
%Finding the phase response.
%Plotting phase versus omega.
angle_h = angle(h);
subplot(3,1,2);
plot(omega/pi,angle_h)
xlabel('frequency normalised to 1 -->');
ylabel('Phase in radian-->');
title('Phase Response of LPF');
subplot(3,1,3)
zplane(z,p,'k')
title('Pole Zero Plot of LPF')
```

RESULT:

Thus, the magnitude and phase response of the analog IIR Butterworth low pass filter is plotted using MATLA

`

FLOWCHART:


Start
Get the stopband edge frequency and passband edge frequency
Get sampling frequency greater than the nyquist frequency (fs_sf)
Get the attenuation values for passband and stop band
Compute the digital frequency 'ωp' and 'ωs'by using the formula ω=2*pi*f/fs_sf
Compute the order of the analog filter using 'buttord' function with 's' option
Compute IIR analog highpass filter coefficients using 'butter' command with 'high' option
Compute frequency response of analog filter using 'freqs' command for different values of ω
Compute the magnitude using abs command
Compute the phase using angle command
Plot the magnitude and phase response
Stop

`

DATE:13/08/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

Expt. No. 5b. DESIGN OF ANALOG IIR BUTTERWORTH HIGH PASS FILTER

AIM:

To write a program in MATLAB to plot the magnitude and phase response of analog IIR Butterworth high pass filter.

SOFTWARE REQUIRED:

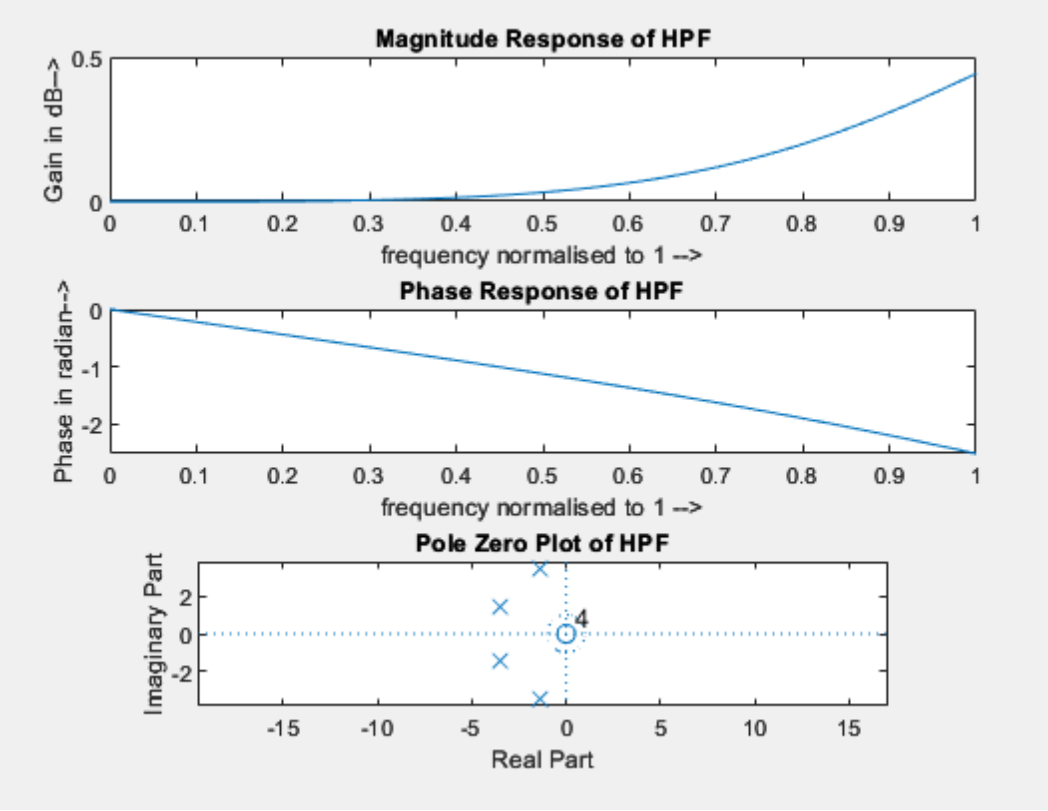
MATLAB Software

ALGORITHM:

1. Clear the command window.
2. Get the passband edge frequency and stopband edge frequency
3. Get sampling frequency greater than the nyquist frequency(fs_sf)
4. Get the attenuation values for passband and stop band.
5. Compute the digital frequency 'ω_p' and 'ω_s'by using the formula ω=2*pi*f/fs_sf
6. Compute the order of the analog filter using 'buttord' function with 's' option
7. Compute IIR analog highpass filter coefficients using 'butter' command with 'high' option.
8. Compute frequency response of analog filter using 'freqs' command for different values of ω
9. Compute the magnitude using abs command.
10. Compute the phase using angle command.
11. Plot the magnitude and phase response.

`

OUTPUT:


Magnitude Response of HPF
Gain in dB-->
0.5
0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
frequency normalised to 1 -->
Phase Response of HPF
Phase in radian-->
0
-1
-2
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
frequency normalised to 1 -->
Pole Zero Plot of HPF
Imaginary Part
2
0
-2
4
-15 -10 -5 0 5 10 15
Real Part

`

PROGRAM:

```
%Program for Butterworth IIR Highpass analog filter
clc;
close all;
clear all;
fprintf('Program for Butterworth IIR Highpass analog filter\n\n');
%We get the passedge, stopedge and sampling frequencies in Hz
fs=input('Enter the stop edge frequency: ');
fp=input('Enter the pass edge frequency: ');
%fs_min should be twice the maximum frequency. Here, fs_min = 2*fp.
fs_min = 2*fp;
fprintf('\nEnter the sampling frequency greater than %d\n',fs_min');
fs_sf=input('Enter the sampling frequency: ');
%We get the attenuation in dB. rp will be around 0 to 3 dB
%rs will be around 30 to 50 dB
rp = input('\nEnter the passband ripple in dB: ');
rs = input('Enter the stopband attenuation in dB: ');
rp1=20*log10(rp)
rs1=20*log10(rs)
%We need to normalise wp,ws to pi. It is similar to finding the digital
%frequency digital omega = analog omega* Ts = analog omega/fs_sf
wp=2*pi*fp/fs_sf;
ws=2*pi*fs/fs_sf;
%The normalised frequencies are wp and ws
fprintf('\nws is %d\n',ws);
fprintf('wp is %d\n',wp);
%Computing the order(N) and cutoff frequency(wc) using wp,ws,rp,rs using
%the buttord command with 's' option for analog filter.
%Finding the coefficients of filter [b a] using butter command with 'high' and 's' option.
[N wc]=buttord(wp,ws,rp1,rs1,'s');
[b a]=butter(N,wc,'high','s');
%Computing the frequency response using freqs command
%Computing h for specific values of w i.e. 0, pi/100, 2*pi/100... till pi
%and storing the corresponding the w values
w=0:(pi/100):pi;
[h omega] = freqs(b,a,w);
hf=tf(b,a)
disp(hf)
[z p]=tf2zp(b,a)
%Finding the magnitude response. Note: log10 should be used.
%Plotting magnitude versus omega.
mag_h=abs(h);
```

`

COMMAND PROMPT:

```
Program for Butterworth IIR Highpass analog filter

Enter the stop edge frequency: 1500
Enter the pass edge frequency: 3000

Enter the sampling frequency greater than 6000
Enter the sampling frequency: 7000

Enter the passband ripple in dB: 0.15
Enter the stopband attenuation in dB: 60

      ws is 1.346397e+00
      wp is 2.692794e+00
        tf with properties:

            Numerator: {[1 0 0 0 0]}
          Denominator: {[1 9.7917 47.9383 137.4826 197.1439]}
             Variable: 's'
              IODelay: 0
           InputDelay: 0
          OutputDelay: 0
            InputName: {''}
            InputUnit: {''}
           InputGroup: [1×1 struct]
           OutputName: {''}
           OutputUnit: {''}
          OutputGroup: [1×1 struct]
                Notes: [0×1 string]
             UserData: []
                 Name: ''
                   Ts: 0
             TimeUnit: 'seconds'
         SamplingGrid: [1×1 struct]
```

`

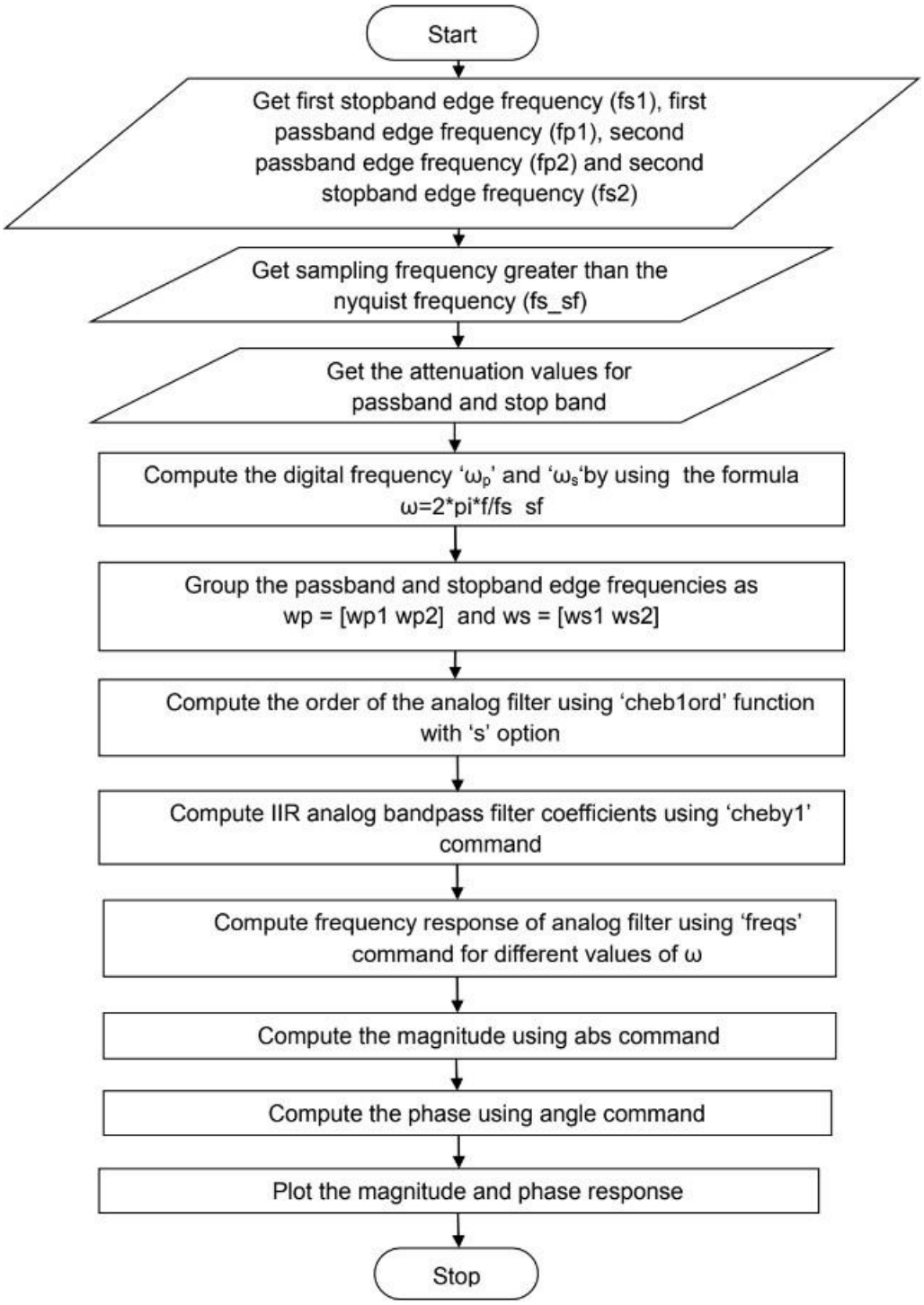
```
figure(1);
subplot(3,1,1);
plot(omega/pi,mag_h);
xlabel('frequency normalised to 1 -->');
ylabel('Gain in dB-->');

title('Magnitude Response of HPF');
%Finding the phase response.
%Plotting phase versus omega.
angle_h = angle(h);
subplot(3,1,2);
plot(omega/pi,angle_h)
xlabel('frequency normalised to 1 -->');
ylabel('Phase in radian-->');
title('Phase Response of HPF');
subplot(3,1,3)
zplane(z,p,'k')
title('Pole Zero Plot of HPF')
```

RESULT:

Thus, the magnitude and phase response of the analog IIR Butterworth high pass filter is plotted using MATLAB.

`

FLOWCHART:


Start
Get first stopband edge frequency (fs1), first passband edge frequency (fp1), second passband edge frequency (fp2) and second stopband edge frequency (fs2)
Get sampling frequency greater than the nyquist frequency (fs_sf)
Get the attenuation values for passband and stop band
Compute the digital frequency 'ωp' and 'ωs'by using the formula ω=2*pi*f/fs sf
Group the passband and stopband edge frequencies as wp = [wp1 wp2] and ws = [ws1 ws2]
Compute the order of the analog filter using 'cheb1ord' function with 's' option
Compute IIR analog bandpass filter coefficients using 'cheby1' command
Compute frequency response of analog filter using 'freqs' command for different values of ω
Compute the magnitude using abs command
Compute the phase using angle command
Plot the magnitude and phase response
Stop

`

DATE:13/08/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

Expt. No. 5c. DESIGN OF ANALOG IIR CHEBYSHEV BAND PASS FILTER

AIM:

To write a program in MATLAB to plot the magnitude and phase response of analog IIR Chebyshev band pass filter.

SOFTWARE REQUIRED:

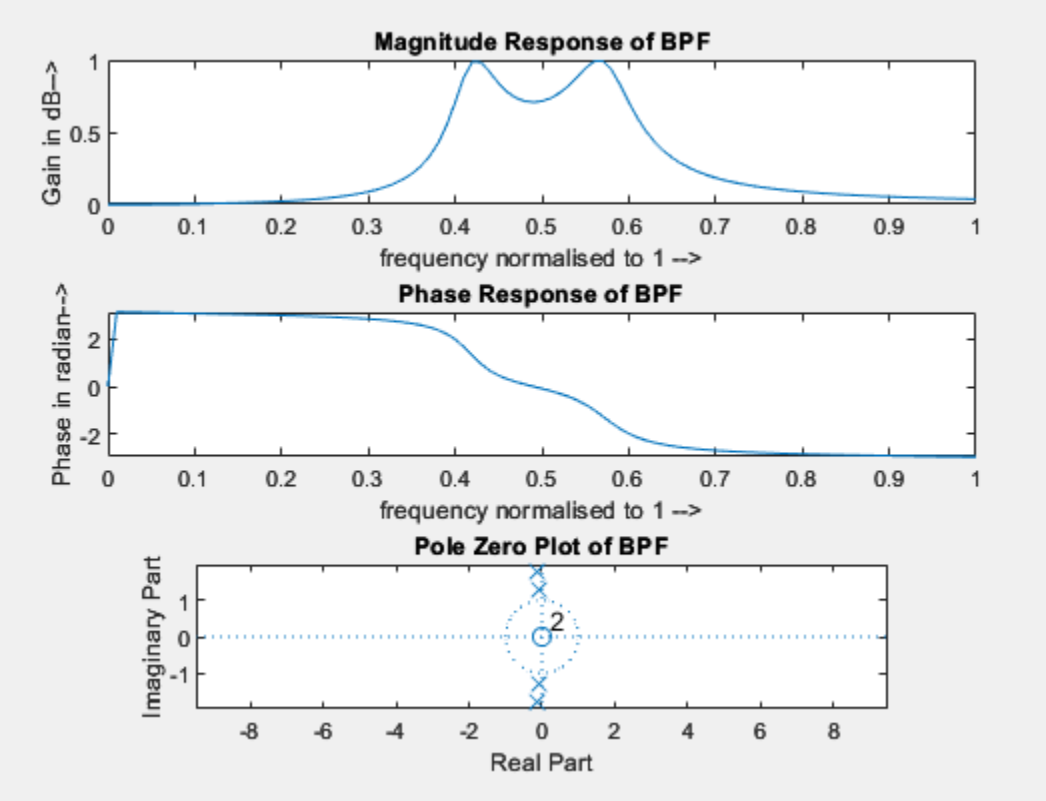
MATLAB Software

ALGORITHM:

1. Clear the command window.
2. Get the first stopband edge frequency (fs1), first passband edge frequency (fp1), second passband edge frequency (fp2) and second stopband edge frequency(fs2).
3. Get sampling frequency greater than the nyquist frequency
4. Get the attenuation values for passband and stop band.
5. Compute the digital frequency 'ω_p' and 'ω_s'by using the formula ω=2*pi*f/fs_sf
6. Group the passband and stopband edge frequencies as wp = [wp1 wp2] and ws = [ws1 ws2]
7. Compute the order of the analog filter using 'cheb1ord' function with 's' option
8. Compute IIR analog bandpass filter coefficients using 'cheby1' command.
9. Compute frequency response of analog filter using 'freqs' command for different values of ω
10. Compute the magnitude using abs command.
11. Compute the phase using angle command.
12. Plot the magnitude and phase response.

`

OUTPUT:


Magnitude Response of BPF
Gain in dB-->
frequency normalised to 1 -->
Phase Response of BPF
Phase in radian-->
frequency normalised to 1 -->
Pole Zero Plot of BPF
Imaginary Part
Real Part
2

`

PROGRAM:

```
%Program for Chebtshev IIR Bandpass analog filter
clc;
close all;
clear all;
fprintf('Program for Butterworth IIR Bandpass analog filter\n\n');
%We get the passedge, stopedge and sampling frequencies in Hz
fs1=input('Enter the stop edge frequency1: ');
fp1=input('Enter the pass edge frequency1: ');
fp2=input('Enter the pass edge frequency2: ');
fs2=input('Enter the stop edge frequency2: ');
%fs_min should be twice the maximum frequency. Here, fs_min = 2*fs2.
fs_min = 2*fs2;
fprintf('\nEnter the sampling frequency greater than %d\n',fs_min');
fs_sf=input('Enter the sampling frequency: ');
%We get the attenuation in dB. rp will be around 0 to 3 dB
%rs will be around 30 to 50 dB
rp = input('\nEnter the passband ripple in dB: ');
rs = input('Enter the stopband attenuation in dB: ');
rp1=20*log10(rp)
rs1=20*log10(rs)
%We need to normalise wp,ws to pi. It is similar to finding the digital
%frequency digital omega = analog omega* Ts = analog omega/fs_sf
ws1=2*pi*fs1/fs_sf;
wp1=2*pi*fp1/fs_sf;
wp2=2*pi*fp2/fs_sf;
ws2=2*pi*fs2/fs_sf;
%The normalised frequencies are wp and ws
fprintf('\nws1 is %d\n',ws1);
fprintf('wp1 is %d\n',wp1);
fprintf('wp2 is %d\n',wp2);
fprintf('ws2 is %d\n',ws2);
%Computing the order(N) and cutoff frequency(wc) using wp,ws,rp,rs using
%the cheb1ord command with 's' option for analog filter.
%Finding the coefficients of filter [b a] using cheby command with 's' option.
wp = [wp1 wp2];
ws = [ws1 ws2];
[N wc]=cheb1ord(wp,ws,rp1,rs1,'s');
[b a]=cheby1(N,rp,wc,'s');
%Computing the frequency response using freqs command
%Computing h for specific values of w i.e. 0, pi/100, 2*pi/100... till pi
%and storing the corresponding the w values
w=0:(pi/100):pi;
[h omega] = freqs(b,a,w);
hf=tf(b,a)
disp(hf)
[z p]=tf2zp(b,a)
%Finding the magnitude response. Note: log10 should be used.
```

`

COMAND PROMPT:

```
Program for Butterworth IIR Band pass analog filter

Enter the stop edge frequency1: 1000
Enter the pass edge frequency1: 4000
Enter the pass edge frequency2: 6000
Enter the stop edge frequency2: 9000

Enter the sampling frequency greater than 18000
Enter the sampling frequency: 20000

Enter the passband ripple in dB: 3
Enter the stopband attenuation in dB: 50


      wsl is 3.141593e-01
      wpl is 1.256637e+00
      wp2 is 1.884956e+00
      ws2 is 2.827433e+00
        tf with properties:

            Numerator: {[0 0 0.1979 0 0]}
          Denominator: {[1 0.4052 5.0169 0.9598 5.6108]}
             Variable: 's'
              IODelay: 0
           InputDelay: 0
          OutputDelay: 0
            InputName: {''}
            InputUnit: {''}
           InputGroup: [1×1 struct]
           OutputName: {''}
           OutputUnit: {''}
          OutputGroup: [1×1 struct]
                Notes: [0×1 string]
             UserData: []
                 Name: ''
                   Ts: 0
             TimeUnit: 'seconds'
         SamplingGrid: [1×1 struct]
```

`

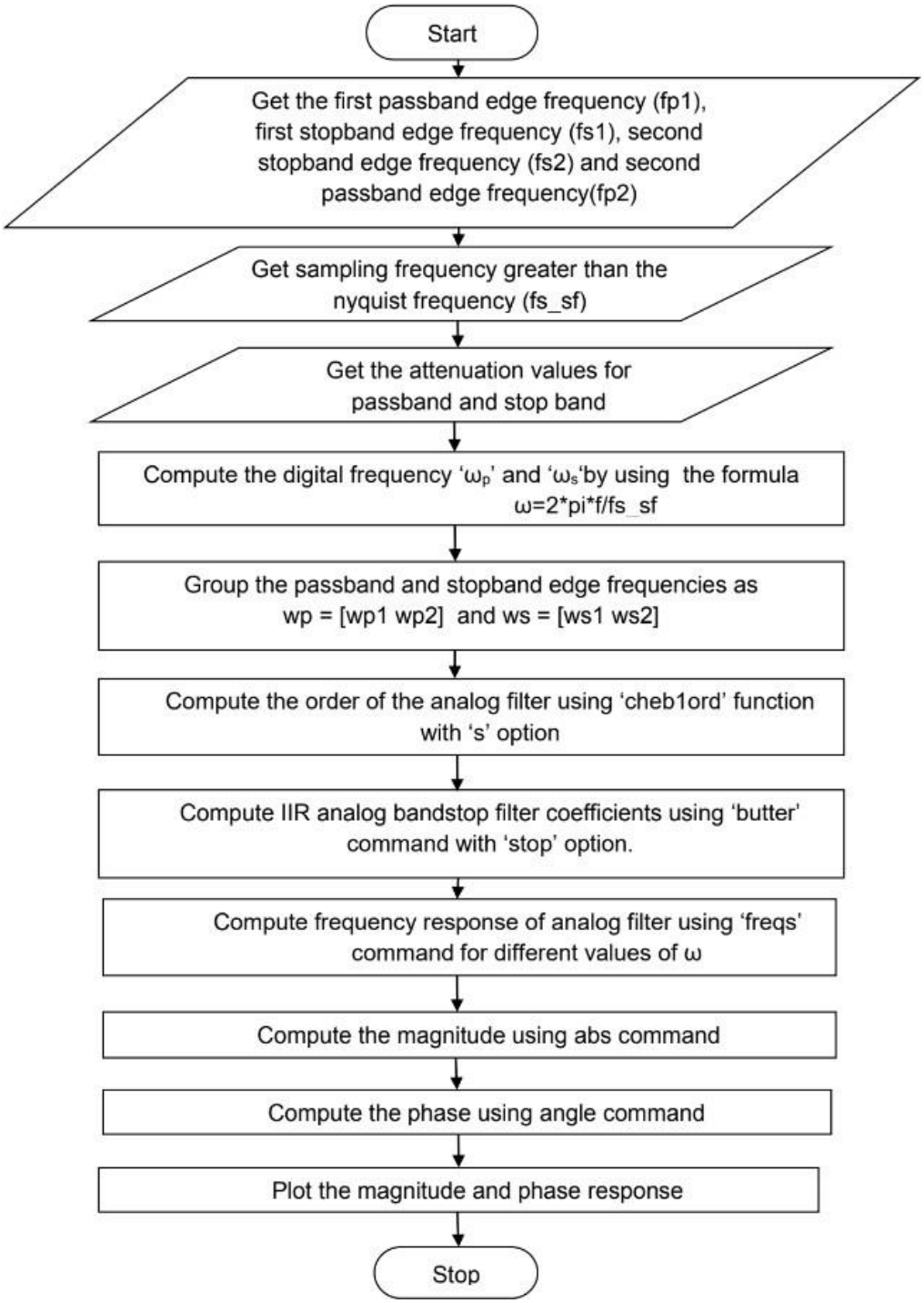
```
%Plotting magnitude versus omega.
mag_h=abs(h);
figure(1);
subplot(3,1,1);
plot(omega/pi,mag_h);
xlabel('frequency normalised to 1 -->');
ylabel('Gain in dB-->');
title('Magnitude Response of BPF');
%Finding the phase response.
%Plotting phase versus omega.
angle_h = angle(h);
subplot(3,1,2);
plot(omega/pi,angle_h)
xlabel('frequency normalised to 1 -->');
ylabel('Phase in  radian-->');
title('Phase Response of BPF');
subplot(3,1,3)
zplane(z,p,'k')
title('Pole Zero Plot of BPF')
```

RESULT:

Thus, the magnitude and phase response of the analog IIR Chebyshev band pass filter is plotted using MATLAB

`

FLOWCHART:


Start
Get the first passband edge frequency (fp1), first stopband edge frequency (fs1), second stopband edge frequency (fs2) and second passband edge frequency(fp2)
Get sampling frequency greater than the nyquist frequency (fs_sf)
Get the attenuation values for passband and stop band
Compute the digital frequency 'ωp' and 'ωs'by using the formula ω=2*pi*f/fs_sf
Group the passband and stopband edge frequencies as wp = [wp1 wp2] and ws = [ws1 ws2]
Compute the order of the analog filter using 'cheb1ord' function with 's' option
Compute IIR analog bandstop filter coefficients using 'butter' command with 'stop' option.
Compute frequency response of analog filter using 'freqs' command for different values of ω
Compute the magnitude using abs command
Compute the phase using angle command
Plot the magnitude and phase response
Stop

`

DATE:13/08/2024

NAME: ARUNKARTHICK R

REG.NO:412522106013

Expt. No. 5d. DESIGN OF ANALOG IIR CHEBYSHEV BAND STOP FILTER

AIM:

To write a program in MATLAB to plot the magnitude and phase response of analog IIR

Chebyshev band stop filter.

SOFTWARE REQUIRED:

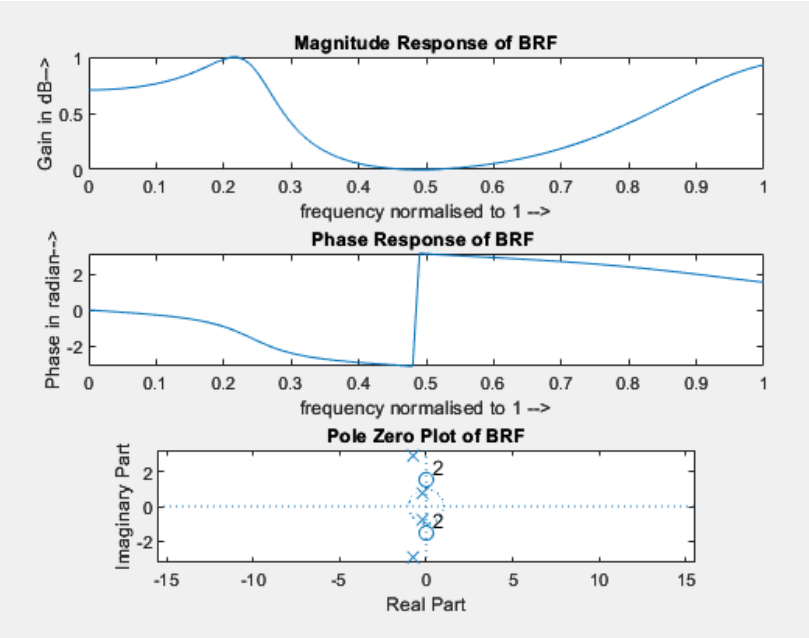
MATLAB Software

ALGORITHM:

1. Clear the command window.
2. Get the first stopband edge frequency (fs1), first passband edge frequency (fp1), second
   passband edge frequency (fp2) and second stopband edge frequency(fs2).
3. Get sampling frequency greater than the nyquist frequency
4. Get the attenuation values for passband and stop band.
5. Compute the digital frequency 'ω_p' and 'ω_s'by using the formula ω=2*pi*f/fs_sf
6. Group the passband and
   stopband edge frequencies as
   wp = [wp1 wp2] and ws =
   [ws1 ws2]
7. Compute the order of the analog filter using 'cheb1ord' function with 's' option
8. Compute IIR analog bandpass filter coefficients using
   'cheby1' command with 'stop'option.
9. Compute frequency response of analog filter using 'freqs' command
   for different values of ω
10. Compute the magnitude using abs command.
11. Compute the phase using angle command.
12. Plot the magnitude and phase response.

`

OUTPUT:


Magnitude Response of BRF
Gain in dB-->
frequency normalised to 1 -->
Phase Response of BRF
Phase in radian-->
frequency normalised to 1 -->
Pole Zero Plot of BRF
Imaginary Part
Real Part
2
2

`

PROGRAM:

```
%Program for Butterworth IIR Bandstop analog filter
clc;
close all;
clear all;
fprintf('Program for Butterworth IIR Bandstop analog filter\n\n');
%We get the passedge, stopedge and sampling frequencies in Hz
fp1=input('Enter the pass edge frequency1: ');
fs1=input('Enter the stop edge frequency1: ');
fs2=input('Enter the stop edge frequency2: ');
fp2=input('Enter the pass edge frequency2: ');
%fs_min should be twice the maximum frequency. Here, fs_min = 2*fp2.
fs_min = 2*fp2;
fprintf('\nEnter the sampling frequency greater than %d\n',fs_min');
fs_sf=input('Enter the sampling frequency: ');
%We get the attenuation in dB. rp will be around 0 to 3 dB
%rs will be around 30 to 50 dB
rp = input('\nEnter the passband ripple in dB: ');
rs = input('Enter the stopband attenuation in dB: ');
rp1=20*log10(rp)
rs1=20*log10(rs)
%We need to normalise wp,ws to pi. It is similar to finding the digital
%frequency digital omega = analog omega* Ts = analog omega/fs_sf
wp1=2*pi*fp1/fs_sf;
ws1=2*pi*fs1/fs_sf;
ws2=2*pi*fs2/fs_sf;
wp2=2*pi*fp2/fs_sf;
%The normalised frequencies are wp and ws
fprintf('\nwp1 is %d\n',wp1);
fprintf('ws1 is %d\n',ws1);
fprintf('ws2 is %d\n',ws2);
fprintf('wp2 is %d\n',wp2);
%Computing the order(N) and cutoff frequency(wc) using wp,ws,rp,rs using
%the cheb1ord command with 's' option for analog filter.
%Finding the coefficients of filter [b a] using cheby command with 'stop' and 's' option.
wp = [wp1 wp2];
ws = [ws1 ws2];
[N wc]=cheb1ord(wp,ws,rp1,rs1,'s');
[b a]=cheby1(N,rp,wc,'stop','s');
%Computing the frequency response using freqs command
%Computing h for specific values of w i.e. 0, pi/100, 2*pi/100... till pi
%and storing the corresponding the w values
w=0:(pi/100):pi;
[h omega] = freqs(b,a,w);
```

`

COMMAND PROMPT:

```
Program for Butterworth IIR Bandstop analog filter

Enter the pass edge frequency1: 1000
Enter the stop edge frequency1: 4000
Enter the stop edge frequency2: 6000
Enter the pass edge frequency2: 9000

Enter the sampling frequency greater than 18000
Enter the sampling frequency: 20000

Enter the passband ripple in dB: 3
Enter the stopband attenuation in dB: 50

wp1 is 3.141593e-01
ws1 is 1.256637e+00
ws2 is 1.884956e+00
wp2 is 2.827433e+00
  tf with properties:

       Numerator: {[0.7079 0 3.3538 0 3.9721]}
     Denominator: {[1 1.8124 10.3289 4.2930 5.6107]}
        Variable: 's'
         IODelay: 0
      InputDelay: 0
     OutputDelay: 0
       InputName: {''}
       InputUnit: {''}
      InputGroup: [1×1 struct]
      OutputName: {''}
      OutputUnit: {''}
     OutputGroup: [1×1 struct]
           Notes: [0×1 string]
        UserData: []
            Name: ''
              Ts: 0
        TimeUnit: 'seconds'
    SamplingGrid: [1×1 struct]
```

`

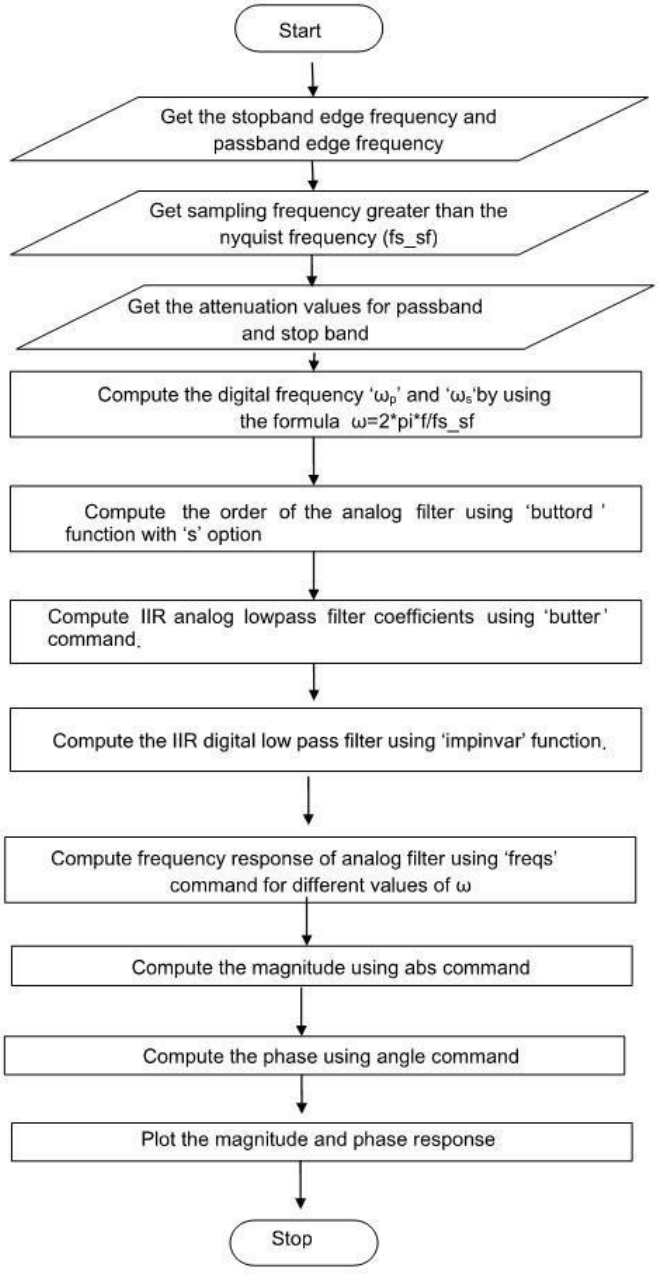
```
hf=tf(b,a)
disp(hf)
[z p]=tf2zp(b,a)
%Finding the magnitude response. Note: log10 should be used.
%Plotting magnitude versus omega.
mag_h=abs(h);

figure(1);
subplot(3,1,1);
plot(omega/pi,mag_h);
xlabel('frequency normalised to 1 -->');
ylabel('Gain in dB-->');
title('Magnitude Response of BRF');
%Finding the phase response.
%Plotting phase versus omega.
angle_h = angle(h);
subplot(3,1,2);
plot(omega/pi,angle_h)
xlabel('frequency normalised to 1 -->');
ylabel('Phase in radian-->');
title('Phase Response of BRF ');
subplot(3,1,3)
zplane(z,p,'k')
title('Pole Zero Plot of BRF')
```

RESULT:

Thus, the magnitude and phase response of the analog IIR Chebyshev band stop filter is plotted using MATLAB.

`

FLOWCHART:


Start
Get the stopband edge frequency and passband edge frequency
Get sampling frequency greater than the nyquist frequency (fs_sf)
Get the attenuation values for passband and stop band
Compute the digital frequency 'ωp' and 'ωs'by using the formula ω=2*pi*f/fs_sf
Compute the order of the analog filter using 'buttord ' function with 's' option
Compute IIR analog lowpass filter coefficients using 'butter' command.
Compute the IIR digital low pass filter using 'impinvar' function.
Compute frequency response of analog filter using 'freqs' command for different values of ω
Compute the magnitude using abs command
Compute the phase using angle command
Plot the magnitude and phase response
Stop

`

DATE:20/08/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

Expt. No. 6a. DESIGN OF DIGITAL IIR BUTTERWORTH LOW PASS FILTER USING IMPULSE INVARIANT TRANSFORMATION

AIM:

To write a program in MATLAB to plot the magnitude and phase response of digital IIRButterworth low pass filter using Impulse Invariant Transformation.

SOFTWARE REQUIRED:

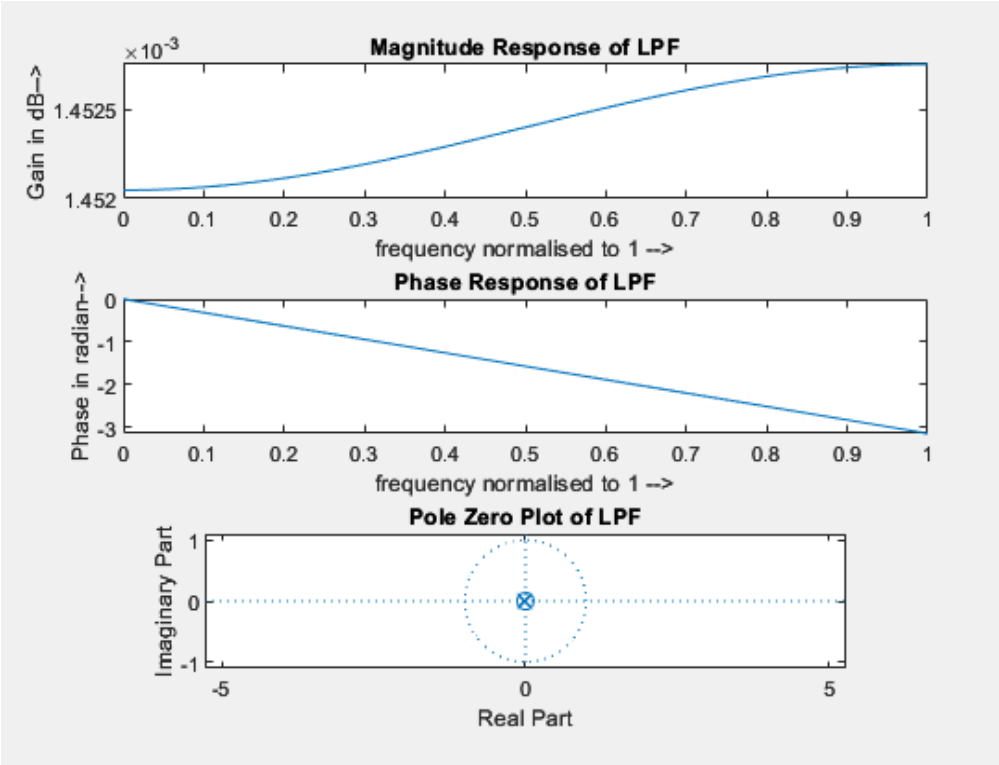
MATLAB Software

ALGORITHM:

1. Clear the command window.
2. Get the passband edge frequency and stopband edge frequency
3. Get sampling frequency greater than the nyquist frequency(fs_sf)
4. Get the attenuation values for passband and stop band.
5. Compute the digital frequency 'ωp' and 'ωs'by using the formula ω=2*pi*f/fs_sf
6. Compute the order of the analog filter using 'buttord' function with 's' option
7. Compute IIR analog lowpass filter coefficients using 'butter' command.
8. Compute the IIR digital lowpass filter using 'impinvar' function.
9. Compute frequency response of digital filter using 'freqz' command for different values ofω
10. Compute the magnitude using abs command.
11. Compute the phase using angle command.
12. Plot the magnitude and phase response.

`

OUTPUT:


Magnitude Response of LPF
×10^-3
Gain in dB-->
1.4525
1.452
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
frequency normalised to 1 -->
Phase Response of LPF
Phase in radian-->
0
-1
-2
-3
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
frequency normalised to 1 -->
Pole Zero Plot of LPF
Imaginary Part
1
0
-1
-5
0
5
Real Part

`

PROGRAM:

```
clc;
close all;
clear all;
fprintf('Program for Digital IIR Butterworth Low Pass Filter using Impulse Invariant
Transformation\n\n');
%We get the passedge, stopedge and sampling frequencies in Hz
fp=input('Enter the passband edge frequency: ');
fs=input('Enter the stopband edge frequency: ');
%fs_min should be twice the maximum frequency. Here, fs_min = 2*fs.
fs_min = 2*fs;
fprintf('\nEnter the sampling frequency greater than %d\n',fs_min');
fs_sf=input('Enter the sampling frequency: ');
%We get the attenuation in dB. rp will be around 0 to 3 dB
%rs will be around 30 to 50 dB
rp = input('\nEnter the passband ripple in dB: ');
rs = input('Enter the stopband attenuation in dB: ');
rp1=20*log10(rp)
rs1=20*log10(rs)
%We need to normalise wp,ws to pi. It is similar to finding the digital
%frequency; digital omega = analog omega* Ts = analog omega/fs_sf
wp=2*pi*fp/fs_sf;
ws=2*pi*fs/fs_sf;
analog_wp=wp*fs_sf;
analog_ws=ws*fs_sf;
%The normalised frequencies are wp and ws
fprintf('\nwp is %d\n',wp);
fprintf('ws is %d\n',ws);
%Computing the order(N) and cutoff frequency(wc) using wp,ws,rp,rs using
%the buttord command with 's' option for analog filter.
%Finding the coefficients [b a] of filter using butter command with 's' option.
[N wc]=buttord(analog_wp,analog_ws,rp1,rs1,'s')
[b a]=butter(N,wc,'s');
%Finding the digital filter coefficients [c d] using impinvar command.
[c d]=impinvar(b,a,fs_sf);
%Computing the frequency response using freqz command
%Computing h for specific values of w i.e. 0, pi/100, 2*pi/100... till pi
%and storing the corresponding the w values
w=0:(pi/100):pi;
[h omega] = freqz(c,d,w);
disp('Analog Filter Transfer Function - Unnormlised: ')
hf=tf(b,a)
disp('Digital Filter Transfer Function: ')
hf1=tf(c,d,fs_sf)
disp(hf)
```

`

COMMAND PROMPT:

```
Program for Digital IIR Butterworth Low Pass Filter using Impulse Invariant Transformation

Enter the passband edge frequency: 2*pi
Enter the stopband edge frequency: 7*pi

Enter the sampling frequency greater than 4.398230e+01
Enter the sampling frequency: 5

Enter the passband ripple in dB: .707
Enter the stopband attenuation in dB: .2

wp is 7.895684e+00
ws is 2.763489e+01

Analog Filter Transfer Function - Unnormlised:
Digital Filter Transfer Function:
  tf with properties:

      Numerator: {[0 0 3.8972e+03]}
    Denominator: {[1 88.2856 3.8972e+03]}
       Variable: 's'
        IODelay: 0
     InputDelay: 0
    OutputDelay: 0
      InputName: {''}
      InputUnit: {''}
     InputGroup: [1×1 struct]
     OutputName: {''}
     OutputUnit: {''}
    OutputGroup: [1×1 struct]
          Notes: [0×1 string]
       UserData: []
           Name: ''
             Ts: 0
       TimeUnit: 'seconds'
   SamplingGrid: [1×1 struct]

  tf with properties:

      Numerator: {[0 0.0015 0]}
    Denominator: {[1 2.4243e-04 2.1459e-08]}
       Variable: 'z'
        IODelay: 0
     InputDelay: 0
    OutputDelay: 0
      InputName: {''}
      InputUnit: {''}
     InputGroup: [1×1 struct]
     OutputName: {''}
     OutputUnit: {''}
    OutputGroup: [1×1 struct]
          Notes: [0×1 string]
       UserData: []
           Name: ''
             Ts: 5
       TimeUnit: 'seconds'
   SamplingGrid: [1×1 struct]
```

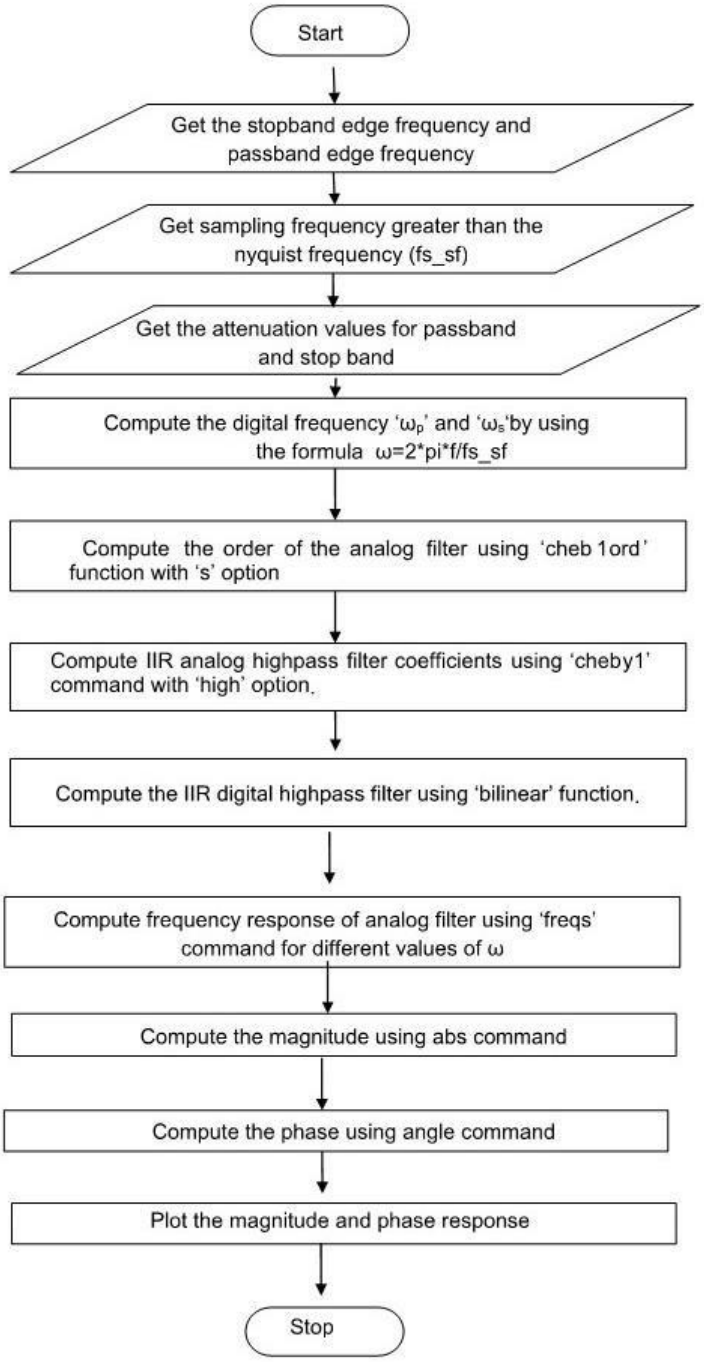
```
disp(hf1)
[z p]=tf2zp(c,d)
%Plotting magnitude versus omega.
mag_h=abs(h);
figure(1);
subplot(3,1,1);
plot(omega/pi,mag_h);
xlabel('frequency normalised to 1 -->');
ylabel('Gain in dB-->');
title('Magnitude Response of LPF');
%Finding the phase response.
%Plotting phase versus omega.
angle_h = angle(h);
subplot(3,1,2);
plot(omega/pi,angle_h)
xlabel('frequency normalised to 1 -->');
ylabel('Phase in radian-->');
title('Phase Response of LPF');
%Plotting poles in the Z plane.
subplot(3,1,3)
zplane(z,p)
title('Pole Zero Plot of LPF')
```

RESULT:

Thus, the magnitude and phase response of the digital IIR Butterworth low pass filter using Impulse Invariant Transformation is plotted using MATLAB.

`

FLOWCHART:


Start
Get the stopband edge frequency and passband edge frequency
Get sampling frequency greater than the nyquist frequency (fs_sf)
Get the attenuation values for passband and stop band
Compute the digital frequency 'ωp' and 'ωs'by using the formula ω=2*pi*f/fs_sf
Compute the order of the analog filter using 'cheb 1ord' function with 's' option
Compute IIR analog highpass filter coefficients using 'cheby1' command with 'high' option.
Compute the IIR digital highpass filter using 'bilinear' function.
Compute frequency response of analog filter using 'freqs' command for different values of ω
Compute the magnitude using abs command
Compute the phase using angle command
Plot the magnitude and phase response
Stop

`

DATE: 20/08/2024

NAME: ARUNKARTHICK R
REG.NO: 412522106013

Expt. No. 6b. DESIGN OF DIGITAL IIR CHEBYSHEV HIGH PASS FILTER USING BILINEAR TRANSFORMATION

AIM:

To write a program in MATLAB to plot the magnitude and phase response of digital IIRChebyshev High pass filter using Bilinear Transformation.

SOFTWARE REQUIRED:

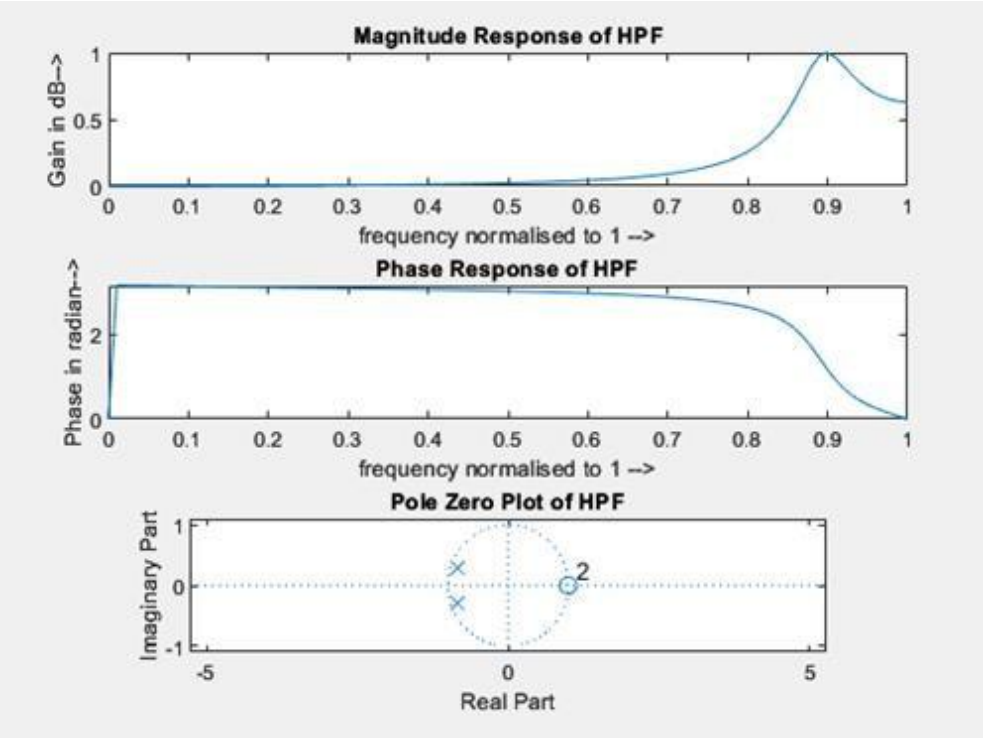
MATLAB Software

ALGORITHM:

1. Clear the command window.
2. Get the passband edge frequency and stopband edge frequency
3. Get sampling frequency greater than the nyquist frequency(fs_sf)
4. Get the attenuation values for passband and stop band.
5. Compute the digital frequency 'ωp' and 'ωs'by using the formula ω=2*pi*f/fs_sf
6. Compute the order of the analog filter using 'cheb1ord' function with 's' option
7. Compute IIR analog highpass filter coefficients using 'cheby1' command with 'high'option.
8. Compute the IIR digital highpass filter using 'bilinear' function.
9. Compute frequency response of digital filter using 'freqz' command for different values ofω
10. Compute the magnitude using abs command.
11. Compute the phase using angle command.
12. Plot the magnitude and phase response.

`

OUTPUT:


Magnitude Response of HPF
Gain in dB-->
frequency normalised to 1 -->
Phase Response of HPF
Phase in radian-->
frequency normalised to 1 -->
Pole Zero Plot of HPF
Imaginary Part
Real Part
2

`

PROGRAM:

```
clc;
close all;
clear all;
fprintf('Program for Digital IIR Chebyshev High Pass Filter using Bilinear Transformation\n\n');
%We get the passedge, stopedge and sampling frequencies in Hz
fs=input('Enter the stopband edge frequency: ');
fp=input('Enter the passband edge frequency: ');

%fs_min should be twice the maximum frequency. Here, fs_min = 2*fs.
fs_min = 2*fp;
fprintf('\nEnter the sampling frequency greater than %d\n',fs_min');
fs_sf=input('Enter the sampling frequency: ');
%We get the attenuation in dB. rp will be around 0 to 3 dB
%rs will be around 30 to 50 dB
rp = input('\nEnter the passband ripple in dB: ');
rs = input('Enter the stopband attenuation in dB: ');
rp1=20*log10(rp)
rs1=20*log10(rs)
%We need to normalise wp,ws to pi. It is similar to finding the digital
%frequency; digital omega = analog omega* Ts = analog omega/fs_sf
wp=2*pi*fp/fs_sf;
ws=2*pi*fs/fs_sf;
analog_wp=2*fs_sf*(tan(wp/2));
analog_ws=2*fs_sf*(tan(ws/2));
%The normalised frequencies are wp and ws
fprintf('\nwp is %d\n',wp);
fprintf('ws is %d\n',ws);
%Computing the order(N) and cutoff frequency(wc) using wp,ws,rp,rs using
%thecheb1ord command with 's' option for analog filter.
%Finding the coefficients of filter [b a] using cheby1 command with 's' and ‘high’ options.
[N wc]=cheb1ord(analog_wp,analog_ws,rp1,rs1,'s')
[b a]=cheby1(N,rp,wc,'high','s');
%Finding the digital filter coefficients [c d] using bilinear command.
[c d]=bilinear(b,a,fs_sf);
%Computing the frequency response using freqz command
%Computing h for specific values of w i.e. 0, pi/100, 2*pi/100... till pi
%and storing the corresponding the w values
w=0:(pi/100):pi;
[h omega] = freqz(c,d,w);
disp('Analog Filter Transfer Function - Unnormlised: ')
hf=tf(b,a)
disp('Digital Filter Transfer Function: ')
hf1=tf(c,d,fs_sf)
disp(hf)
disp(hf1)
[z p]=tf2zp(c,d)
%Plotting magnitude versus omega.
```

`

COMMAND PROMPT:

```
Analog Filter Transfer Function - Unnormlised:
Digital Filter Transfer Function:
  tf with properties:

       Numerator: {[0.6310 0 0]}
     Denominator: {[1 5.116le+04 5.8378e+09]}
        Variable: 's'
         IODelay: 0
      InputDelay: 0
     OutputDelay: 0
       InputName: {''}
       InputUnit: {''}
      InputGroup: [1×1 struct]
      OutputName: {''}
      OutputUnit: {''}
     OutputGroup: [1×1 struct]
           Notes: [0×1 string]
        UserData: []
            Name: ''
              Ts: 0
        TimeUnit: 'seconds'
    SamplingGrid: [1×1 struct]

tf with properties:

     Numerator: {[0.0183 -0.0366 0.0183]}
   Denominator: {[1 1.6716 0.7878]}
      Variable: 'z'
       IODelay: 0
    InputDelay: 0
   OutputDelay: 0
     InputName: {''}
     InputUnit: {''}
    InputGroup: [1×1 struct]
    OutputName: {''}
    OutputUnit: {''}
   OutputGroup: [1×1 struct]
         Notes: [0×1 string]
      UserData: []
          Name: ''
            Ts: 7000
      TimeUnit: 'seconds'
  SamplingGrid: [1×1 struct]
```

`

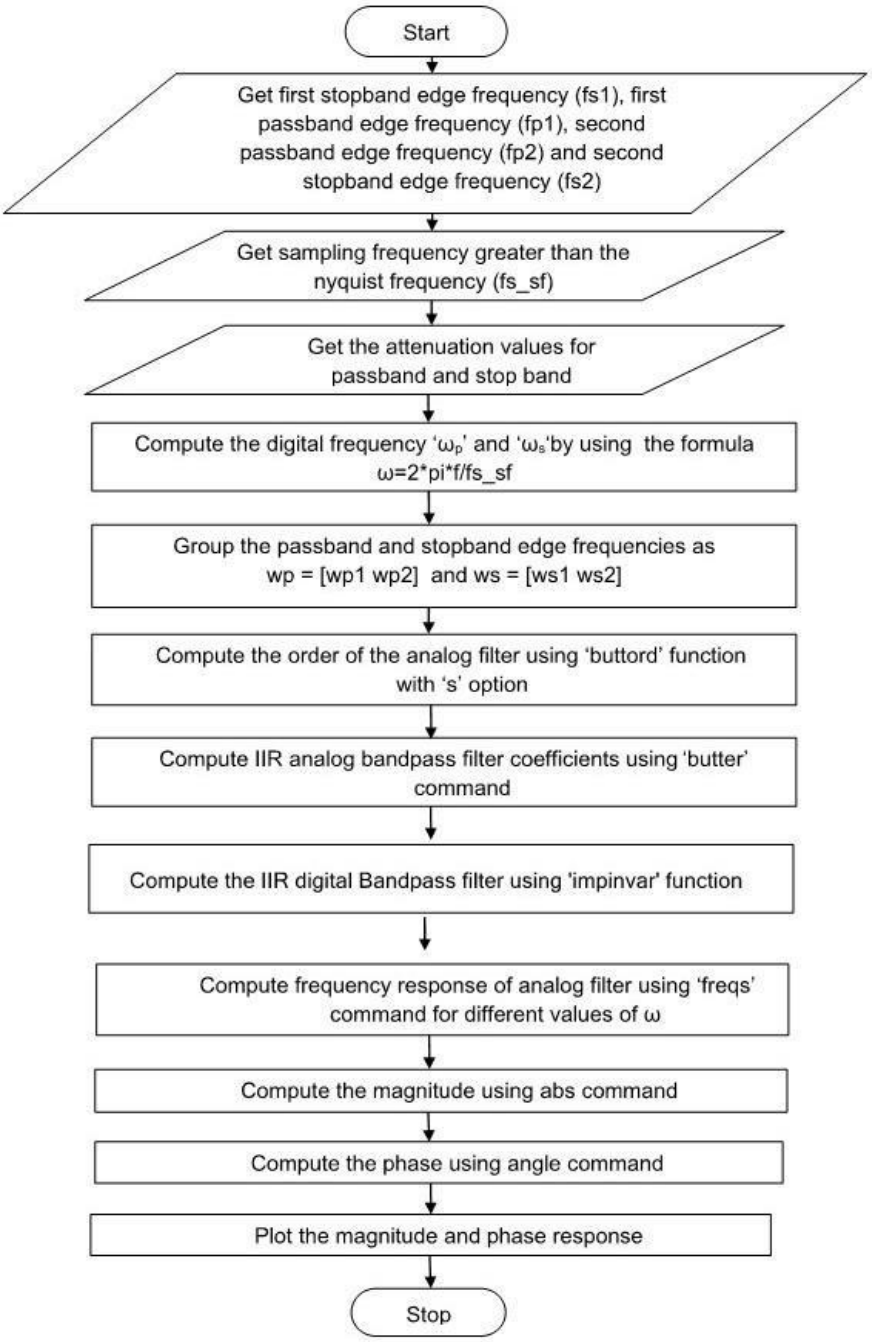
```
mag_h=abs(h);
figure(1);
subplot(3,1,1);
plot(omega/pi,mag_h);
xlabel('frequency normalised to 1 -->');
ylabel('Gain in dB-->');
title('Magnitude Response of HPF');
%Finding the phase response.
%Plotting phase versus omega.
angle_h = angle(h);
subplot(3,1,2);
plot(omega/pi,angle_h)
xlabel('frequency normalised to 1 -->');
ylabel('Phase in  radian-->');
title('Phase Response of HPF');
%Plotting poles in the Z plane.
subplot(3,1,3)
zplane(z,p)
title('Pole Zero Plot of HPF')
```

RESULT

Thus, the magnitude and phase response of the digital IIR Chebyshev High passfilter using Bilinear Transformation is plotted using MATLAB

`

FLOWCHART:


Start
Get first stopband edge frequency (fs1), first passband edge frequency (fp1), second passband edge frequency (fp2) and second stopband edge frequency (fs2)
Get sampling frequency greater than the nyquist frequency (fs_sf)
Get the attenuation values for passband and stop band
Compute the digital frequency 'ωp' and 'ωs'by using the formula ω=2*pi*f/fs_sf
Group the passband and stopband edge frequencies as wp = [wp1 wp2] and ws = [ws1 ws2]
Compute the order of the analog filter using 'buttord' function with 's' option
Compute IIR analog bandpass filter coefficients using 'butter' command
Compute the IIR digital Bandpass filter using 'impinvar' function
Compute frequency response of analog filter using 'freqs' command for different values of ω
Compute the magnitude using abs command
Compute the phase using angle command
Plot the magnitude and phase response
Stop

`

DATE:20/08/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

Expt. No. 6c. DESIGN OF DIGITAL IIR BUTTERWORTH BAND PASS FILTER USING IMPULSE INVARIANT TRANSFORMATION

AIM:

To write a program in MATLAB to plot the magnitude and phase response of analog IIR
Chebyshev band pass analog IIR filter.

SOFTWARE REQUIRED:

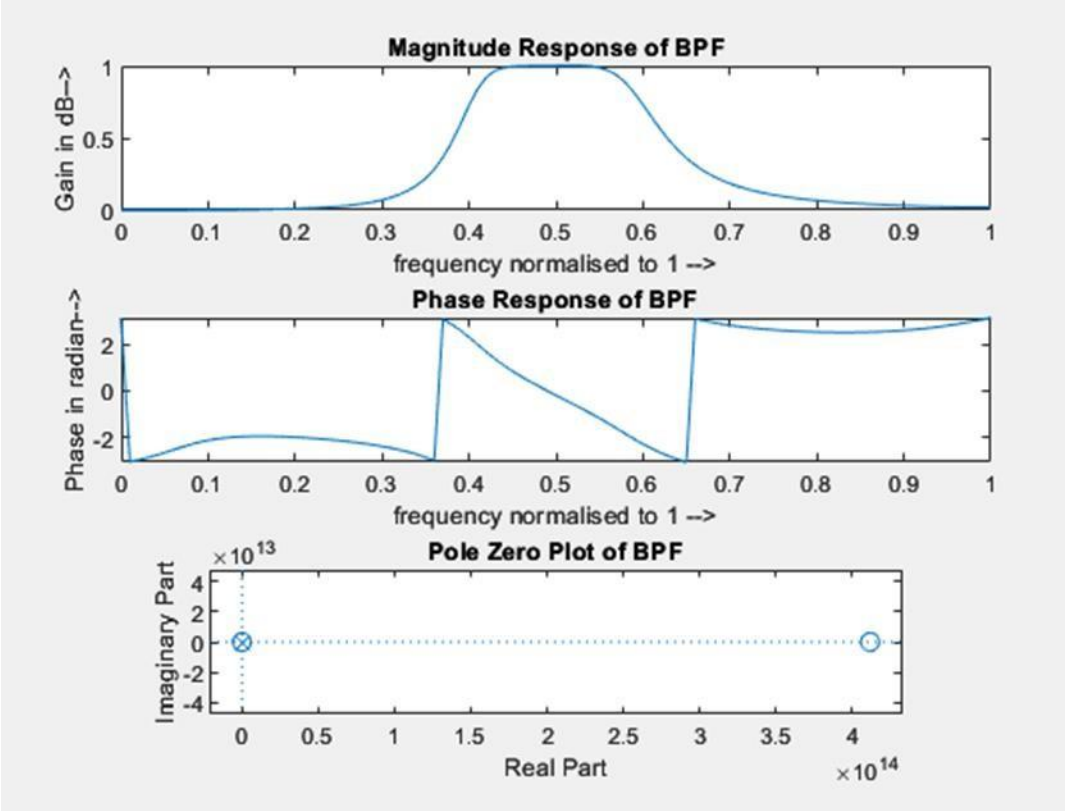
MATLAB Software

ALGORITHM:

1. Clear the command window.
2. Get the first stopband edge frequency (fs1), first passband edge frequency (fp1), second passband edge frequency (fp2) and second stopband edge frequency(fs2).
3. Get sampling frequency greater than the nyquist frequency
4. Get the attenuation values for passband and stop band.
5. Compute the digital frequency 'ω_p' and 'ω_s'by using the formula ω=2*pi*f/fs_sf
6. Group the passband and stopband edge frequencies as wp = [wp1 wp2] and ws = [ws1 ws2]
7. Compute the order of the analog filter using 'buttord' function with 's' option
8. Compute IIR analog bandpass filter coefficients using 'butter' command.
9. Compute the IIR digital band pass filter using 'impinvar' function.
10. Compute frequency response of digital filter using 'freqz' command for different values ofω
11. Compute the magnitude using abs command.
12. Compute the phase using angle command.
13. Plot the magnitude and phase response.

`

OUTPUT:



COMMAND PROMPT:

```
Program for Digital IIR Butterworth Band Pass Filter using Impulse Invariant Transformation

Enter the stop edge frequency1: 1000
Enter the pass edge frequency1: 4000
Enter the pass edge frequency2: 6000
Enter the stop edge frequency2: 9000

Enter the sampling frequency greater than 18000
Enter the sampling frequency: 20000

Enter the passband ripple in dB: 3
Enter the stopband attenuation in dB: 30

ws1 is 3.141593e-01
wp1 is 1.256637e+00
wp2 is 1.884956e+00
ws2 is 2.827433e+00
```

`

PROGRAM:

```
clc;
close all;
clear all;
fprintf('Program for Digital IIR Butterworth Band Pass Filter using Impulse Invariant
Transformation\n\n');
%We get the passedge, stopedge and sampling frequencies in Hz
fs1=input('Enter the stop edge frequency1: ');
fp1=input('Enter the pass edge frequency1: ');
fp2=input('Enter the pass edge frequency2: ');
fs2=input('Enter the stop edge frequency2: ');
%fs_min should be twice the maximum frequency. Here, fs_min = 2*fs2.
fs_min = 2*fs2;
fprintf('\nEnter the sampling frequency greater than %d\n',fs_min');
fs_sf=input('Enter the sampling frequency: ');
%We get the attenuation in dB. rp will be around 0 to 3 dB
%rs will be around 30 to 50 dB
rp = input('\nEnter the passband ripple in dB: ');
rs = input('Enter the stopband attenuation in dB: ');
rp1=20*log10(rp)
rs1=20*log10(rs)
%We need to normalise wp,ws to pi. It is similar to finding the digital
%frequency digital omega = analog omega* Ts = analog omega/fs_sf
ws1=2*pi*fs1/fs_sf;
wp1=2*pi*fp1/fs_sf;
wp2=2*pi*fp2/fs_sf;
ws2=2*pi*fs2/fs_sf;
analog_ws1=ws1*fs_sf;
analog_wp1=wp1*fs_sf;
analog_wp2=wp2*fs_sf;
analog_ws2=ws2*fs_sf;
%The normalised frequencies are wp and ws
fprintf('\nws1 is %d\n',ws1);
fprintf('wp1 is %d\n',wp1);
fprintf('wp2 is %d\n',wp2);
fprintf('ws2 is %d\n',ws2);
%Computing the order(N) and cutoff frequency(wc) using wp,ws,rp,rs using
%the buttord command with 's' option for analog filter.
%Finding the coefficients of filter [b a] using butter command with 's' option.
wp = [analog_wp1 analog_wp2];
ws = [analog_ws1 analog_ws2];
[N wc]=buttord(wp,ws,rp1,rs1,'s');
[b a]=butter(N,wc,'s');
%Finding the digital filter coefficients [c d] using impinvar command.
```

`

```
Analog Filter Transfer Function - Unnormlised:
Digital Filter Transfer Function:
  tf with properties:

       Numerator: {[0 0 0 2.1016e+12 0 0 0]}
     Denominator: {[1 2.5618e+04 3.1706e+09 5.0647e+13 3.0041e+18 2.2998e+22 8.5058e+26]}
        Variable: 's'
         IODelay: 0
      InputDelay: 0
     OutputDelay: 0
       InputName: {''}
       InputUnit: {''}
      InputGroup: [1×1 struct]
      OutputName: {''}
      OutputUnit: {''}
     OutputGroup: [1×1 struct]
           Notes: [0×1 string]
        UserData: []
            Name: ''
              Ts: 0
        TimeUnit: 'seconds'
    SamplingGrid: [1×1 struct]

      tf with properties:

           Numerator: {[-9.0949e-17 0.0375 -0.0937 0.0369 0.0461 -0.0290 0]}
         Denominator: {[1 -0.1704 1.7715 -0.1734 1.1807 -0.0585 0.2778]}
            Variable: 'z'
             IODelay: 0
          InputDelay: 0
         OutputDelay: 0
           InputName: {''}
           InputUnit: {''}
          InputGroup: [1×1 struct]
          OutputName: {''}
          OutputUnit: {''}
         OutputGroup: [1×1 struct]
               Notes: [0×1 string]
            UserData: []
                Name: ''
                  Ts: 20000
            TimeUnit: 'seconds'
        SamplingGrid: [1×1 struct]
```

`

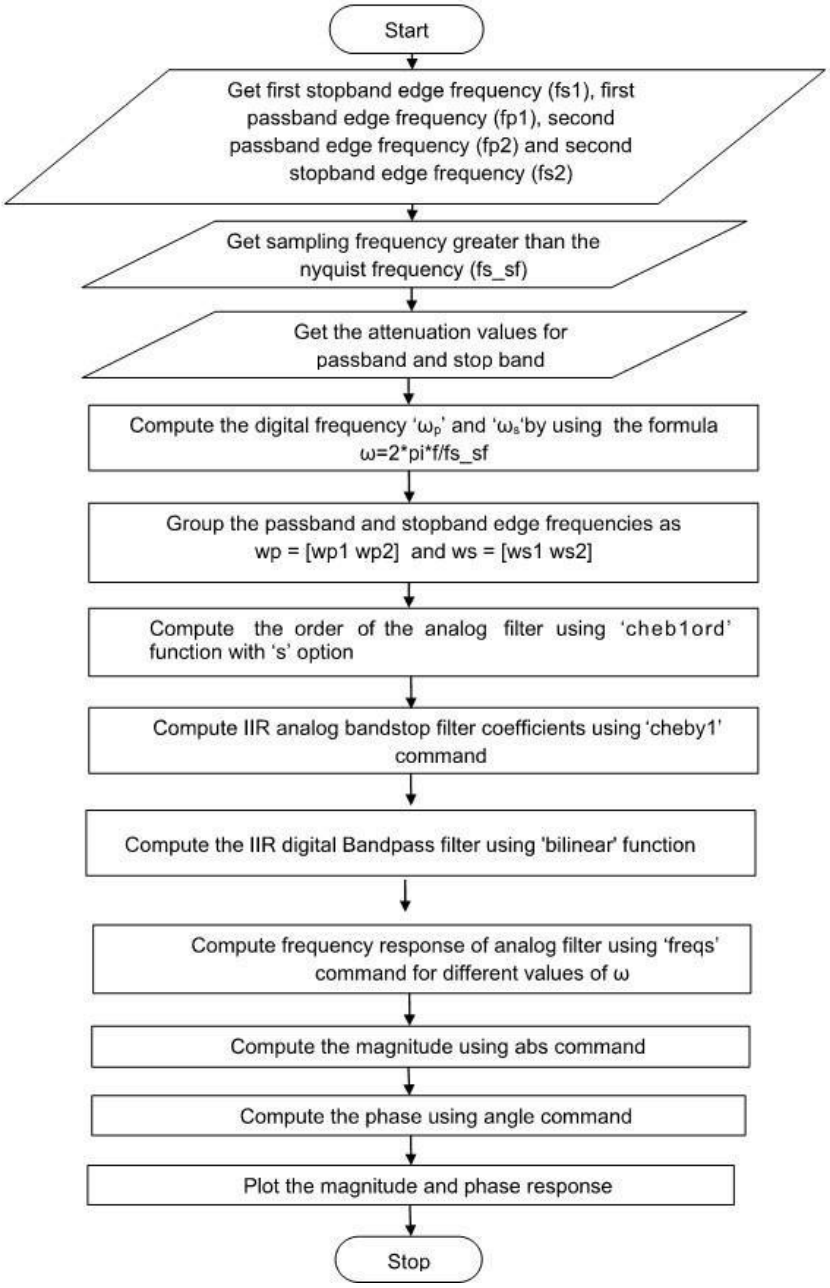
```
[c d]=impinvar(b,a,fs_sf);
%Computing the frequency response using freqz command
%Computing h for specific values of w i.e. 0, pi/100, 2*pi/100... till pi
%and storing the corresponding the w values
w=0:(pi/100):pi;
[h omega] = freqz(c,d,w);
disp('Analog Filter Transfer Function - Unnormlised: ')
hf=tf(b,a)
disp('Digital Filter Transfer Function: ')
hf1=tf(c,d,fs_sf)
disp(hf)
disp(hf1)
[z p]=tf2zp(c,d)
%Plotting magnitude versus omega.
mag_h=abs(h);
figure(1);
subplot(3,1,1);
plot(omega/pi,mag_h);
xlabel('frequency normalised to 1 -->');
ylabel('Gain in dB-->');
title('Magnitude Response of BPF');
%Finding the phase response.
%Plotting phase versus omega.
angle_h = angle(h);
subplot(3,1,2);
plot(omega/pi,angle_h)
xlabel('frequency normalised to 1 -->');
ylabel('Phase in radian-->');
title('Phase Response of BPF');
%Plotting poles in the Z plane.
subplot(3,1,3)
zplane(z,p)
title('Pole Zero Plot of BPF')
```

RESULT:

Thus, the magnitude and phase response of the digital IIR Butterworth Band Pass filter using Impulse Invariant Transformation is plotted using MATLAB.

`

FLOWCHART:


Start
Get first stopband edge frequency (fs1), first passband edge frequency (fp1), second passband edge frequency (fp2) and second stopband edge frequency (fs2)
Get sampling frequency greater than the nyquist frequency (fs_sf)
Get the attenuation values for passband and stop band
Compute the digital frequency 'ωp' and 'ωs'by using the formula ω=2*pi*f/fs_sf
Group the passband and stopband edge frequencies as wp = [wp1 wp2] and ws = [ws1 ws2]
Compute the order of the analog filter using 'cheb1ord' function with 's' option
Compute IIR analog bandstop filter coefficients using 'cheby1' command
Compute the IIR digital Bandpass filter using 'bilinear' function
Compute frequency response of analog filter using 'freqs' command for different values of ω
Compute the magnitude using abs command
Compute the phase using angle command
Plot the magnitude and phase response
Stop

`

DATE:20/08/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

Expt. No. 6d. DESIGN OF DIGITAL IIR CHEBYSHEV BAND STOP FILTER USINGBILINEAR TRANSFORMATION

AIM:

To write a program in MATLAB to plot the magnitude and phase response of digital IIR Chebyshev Band Stop filter using Bilinear Transformation.

SOFTWARE REQUIRED:

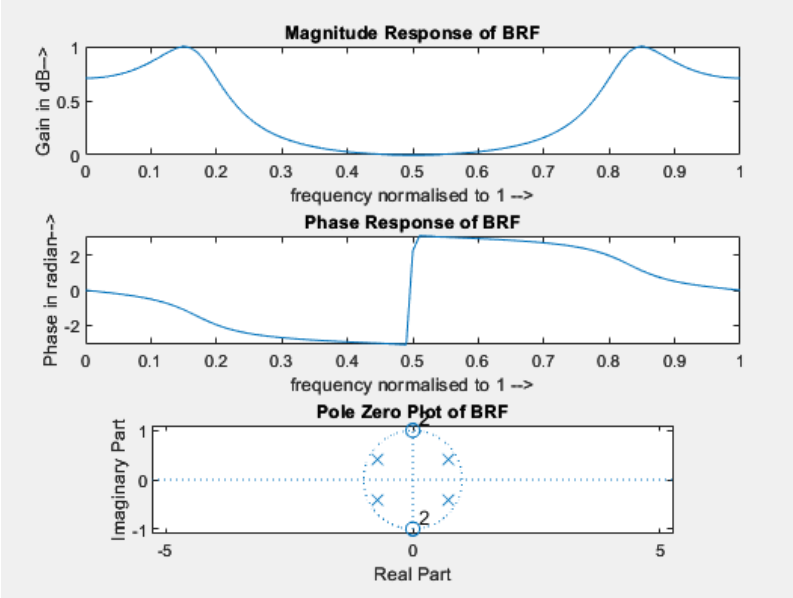
MATLAB Software

ALGORITHM:

1. Clear the command window.
2. Get the first stopband edge frequency (fs1), first passband edge frequency (fp1), second passband edge frequency (fp2) and second stopband edge frequency(fs2).
3. Get sampling frequency greater than the nyquist frequency
4. Get the attenuation values for passband and stop band.
5. Compute the digital frequency 'ωp' and 'ωs'by using the formula ω=2*pi*f/fs_sf
6. Group the passband and stopband edge frequencies as wp = [wp1 wp2] and ws = [ws1 ws2]
7. Compute the order of the analog filter using 'cheb1ord' function with 's' option
8. Compute IIR analog bandpass filter coefficients using 'cheby1' command with 'stop'option.
9. Compute the IIR digital band stop filter using 'bilinear' function.
10. Compute frequency response of digital filter using 'freqz' command for different values ofω
11. Compute the magnitude using abs command.
12. Compute the phase using angle command.
13. Plot the magnitude and phase response.

`

OUTPUT:



COMMAND PROMPT:

```
Program for Digital IIR Chebyshev Band Stop Filter using Bilinear Transformation

Enter the pass edge frequency1: 1000
Enter the stop edge frequency1: 2000
Enter the stop edge frequency2: 3000
Enter the pass edge frequency2: 4000

Enter the sampling frequency greater than 8000
Enter the sampling frequency: 10000

Enter the passband ripple in dB: 3
Enter the stopband attenuation in dB: 30

ws1 is 1.256637e+00
wp1 is 6.283185e-01
wp2 is 2.513274e+00
ws2 is 1.884956e+00
```

`

PROGRAM:

```
clc;
close all;
clear all;
fprintf('Program for Digital IIR Chebyshev Band Stop Filter using Bilinear Transformation\n\n');
%We get the passedge, stopedge and sampling frequencies in Hz
fp1=input('Enter the pass edge frequency1: ');
fs1=input('Enter the stop edge frequency1: ');
fs2=input('Enter the stop edge frequency2: ');
fp2=input('Enter the pass edge frequency2: ');
%fs_min should be twice the maximum frequency. Here, fs_min = 2*fs2.
fs_min = 2*fp2;
fprintf('\nEnter the sampling frequency greater than %d\n',fs_min');
fs_sf=input('Enter the sampling frequency: ');
%We get the attenuation in dB. rp will be around 0 to 3 dB
%rs will be around 30 to 50 dB
rp = input('\nEnter the passband ripple in dB: ');
rs = input('Enter the stopband attenuation in dB: ');
rp1=20*log10(rp)
rs1=20*log10(rs)
%We need to normalise wp,ws to pi. It is similar to finding the digital
%frequency digital omega = analog omega* Ts = analog omega/fs_sf
wp1=2*pi*fp1/fs_sf;
ws1=2*pi*fs1/fs_sf;
ws2=2*pi*fs2/fs_sf;
wp2=2*pi*fp2/fs_sf;
analog_wp1=2*fs_sf*(tan(wp1/2));
analog_ws1=2*fs_sf*(tan(ws1/2));
analog_ws2=2*fs_sf*(tan(ws2/2));
analog_wp2=2*fs_sf*(tan(wp2/2));
%The normalised frequencies are wp and ws
fprintf('\nws1 is %d\n',ws1);
fprintf('wp1 is %d\n',wp1);
fprintf('wp2 is %d\n',wp2);
fprintf('ws2 is %d\n',ws2);
%Computing the order(N) and cutoff frequency(wc) using wp,ws,rp,rs using
%the cheb1ord command with 's' option for analog filter.
%Finding the coefficients of filter [b a] using cheby command with 's' and ‘stop’ option.
wp = [analog_wp1 analog_wp2];
ws = [analog_ws1 analog_ws2];
[N wc]=cheb1ord(wp,ws,rp1,rs1,'s');
[b a]=cheby1(N,rp,wc,'stop','s');
%Finding the digital filter coefficients [c d] using bilinear command.
[c d]=bilinear(b,a,fs_sf);
%Computing the frequency response using freqz command
%Computing h for specific values of w i.e. 0, pi/100, 2*pi/100... till pi
%and storing the corresponding the w values
```

`

```
Analog Filter Transfer Function - Unnormlised:
Digital Filter Transfer Function:
  tf with properties:

      Numerator: {[0.7079 0 5.6636e+08 0 1.1327e+17]}
    Denominator: {[1 5.0152e+04 5.0815e+09 2.0061e+13 1.6000e+17]}
       Variable: 's'
        IODelay: 0
     InputDelay: 0
    OutputDelay: 0
      InputName: {''}
      InputUnit: {''}
     InputGroup: [1×1 struct]
     OutputName: {''}
     OutputUnit: {''}
    OutputGroup: [1×1 struct]
          Notes: [0×1 string]
       UserData: []
           Name: ''
             Ts: 0
       TimeUnit: 'seconds'
   SamplingGrid: [1×1 struct]



tf with properties:

     Numerator: {[0.1436 2.9391e-09 0.2872 2.9391e-09 0.1436]}
   Denominator: {[1 6.7543e-09 -0.6799 1.5490e-09 0.4913]}
      Variable: 'z'
       IODelay: 0
    InputDelay: 0
   OutputDelay: 0
     InputName: {''}
     InputUnit: {''}
    InputGroup: [1×1 struct]
    OutputName: {''}
    OutputUnit: {''}
   OutputGroup: [1×1 struct]
         Notes: [0×1 string]
      UserData: []
          Name: ''
            Ts: 10000
      TimeUnit: 'seconds'
  SamplingGrid: [1×1 struct]
```

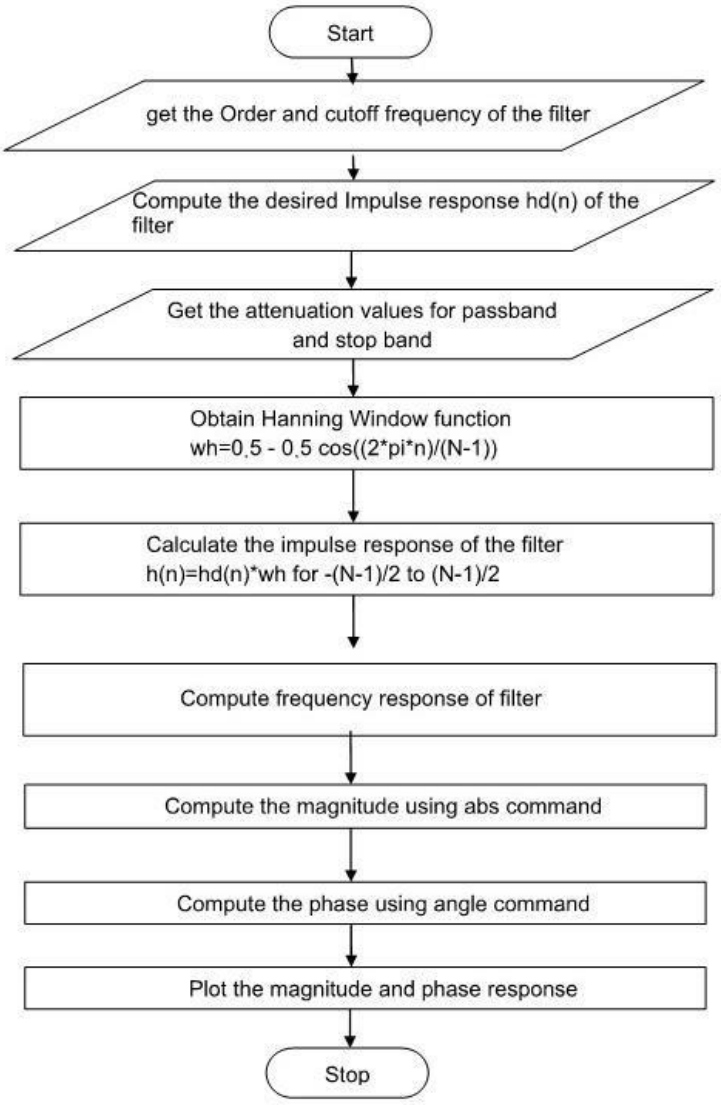
```
w=0:(pi/100):pi;
[h omega] = freqz(c,d,w);
disp('Analog Filter Transfer Function - Unnormlised: ')
hf=tf(b,a)
disp('Digital Filter Transfer Function: ')
hf1=tf(c,d,fs_sf)
disp(hf)
disp(hf1)
[z p]=tf2zp(c,d)
%Plotting magnitude versus omega.
mag_h=abs(h);
figure(1);
subplot(3,1,1);
plot(omega/pi,mag_h);
xlabel('frequency normalised to 1 -->');
ylabel('Gain in dB-->');
title('Magnitude Response of BRF');
%Finding the phase response.
%Plotting phase versus omega.
angle_h = angle(h);
subplot(3,1,2);
plot(omega/pi,angle_h)
xlabel('frequency normalised to 1 -->');
ylabel('Phase in radian-->');
title('Phase Response of BRF');
%Plotting poles in the Z plane.
subplot(3,1,3)
zplane(z,p)
title('Pole Zero Plot of BRF')
```

RESULT:

Thus, the magnitude and phase response of the digital IIR Chebyshev Band Stop filter using Bilinear Transformation is plotted using MATLAB.

`

FLOWCHART:


Start
get the Order and cutoff frequency of the filter
Compute the desired Impulse response hd(n) of the filter
Get the attenuation values for passband and stop band
Obtain Hanning Window function
wh=0.5 - 0.5 cos((2*pi*n)/(N-1))
Calculate the impulse response of the filter
h(n)=hd(n)*wh for -(N-1)/2 to (N-1)/2
Compute frequency response of filter
Compute the magnitude using abs command
Compute the phase using angle command
Plot the magnitude and phase response
Stop

`

DATE:17/09/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

Expt. No. 7a. DESIGN OF FIR LOW PASS FILTER USING HANNING WINDOW TECHNIQUE

AIM:

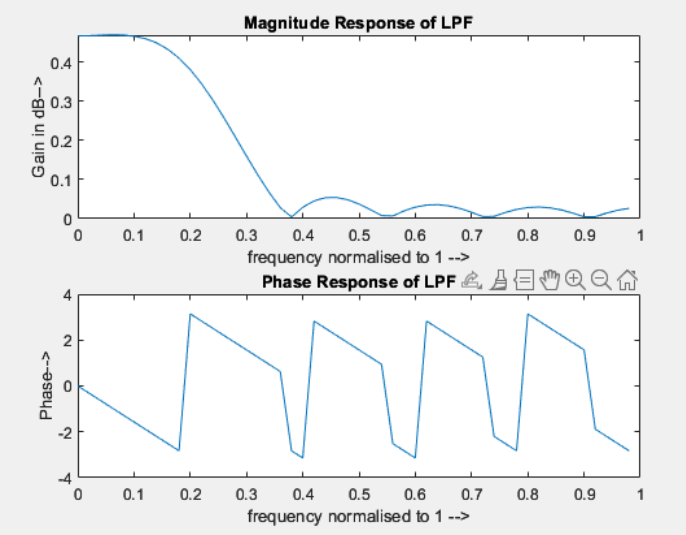
To write a program in MATLAB to plot the magnitude and phase response of FIR low pass filter using Hanning Window technique.

ALGORITHM:

1. Clear the command window.
2. Get the order and cut off frequency of the filter.
3. Compute the desired impulse response $h_d(n)$ of the filter.
4. Obtain different window functions
   - Rectangular window : wh =1
   - Hamming Window : wh = 0.54 – 0.46 cos(2*pi*n/(N-1))
   - Hanning Window : wh = 0.5 – 0.5 cos(2*pi*n/(N-1))
   - Blackman Window : wh = 0.42 – 0.5 cos(2*pi*n/(N-1)) +0.08 cos(4*pi*n/(N-1)) 5. Calculate the impulse response of the filter h(n) = $h_d(n)$ * wh for n= -(N-1)/2 to (N+1)/2 6. Compute the frequency response of the filter.
7. Compute the magnitude using abs command.
8. Compute the phase using angle command.
9. Plot the magnitude and phase response.

`

OUTPUT:



COMMAND PROMPT:

```
Program for FIR Low Pass filter using windowing technique

Enter the order of the filter: 11
Enter the cut off frequency: 1200

Enter the sampling frequency greater than 2400
Enter the sampling frequency:9000

hd =

   -0.0551

hd =

   -0.0551   -0.0165

hd =

   -0.0551   -0.0165    0.0624

hd =

   -0.0551   -0.0165    0.0624    0.1583

hd =

   -0.0551   -0.0165    0.0624    0.1583    0.2366

hd =

   -0.0551   -0.0165    0.0624    0.1583    0.2366    0.2667

hd =

   -0.0551   -0.0165    0.0624    0.1583    0.2366    0.2667    0.2366

hd =

   -0.0551   -0.0165    0.0624    0.1583    0.2366    0.2667    0.2366    0.1583

hd =

   -0.0551   -0.0165    0.0624    0.1583    0.2366    0.2667    0.2366    0.1583    0.0624

hd =

   -0.0551   -0.0165    0.0624    0.1583    0.2366    0.2667    0.2366    0.1583    0.0624   -0.0165
```

`

PROGRAM:

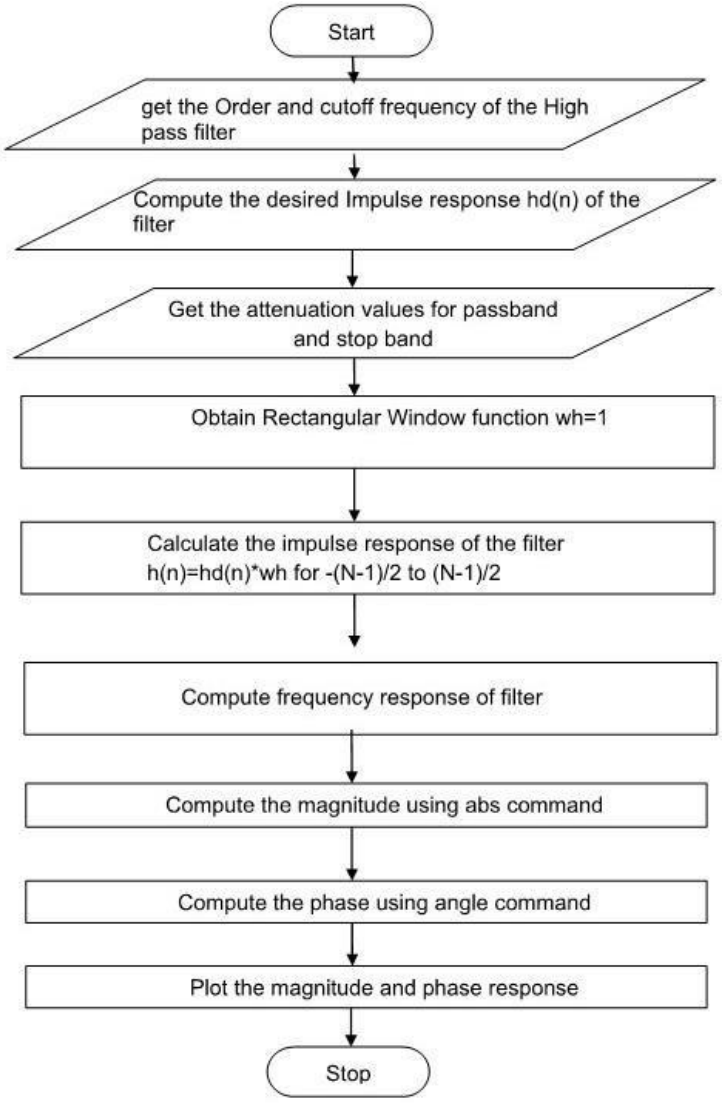
```
clc
close all
clear all

fprintf('Program for FIR Low Pass filter using windowing technique\n\n');
N =input('Enter the order of the filter: ');
fc=input('Enter the cut off frequency: ');
fs_min = 2 * fc;
fprintf('\nEnter the sampling frequency greater than %d\n',fs_min');
fs_sf=input('Enter the sampling frequency:');
wc=2*pi*fc / fs_sf;
alp = (N-1)/2;
for n = 1 : 1 : N
 if (n - 1) == alp
 hd(n) = wc / pi
 else
 hd(n) = (sin((n-1-alp) * wc)) / (pi*(n-1-alp))
 end
hannwin(n) = 0.5 - 0.5 * (cos(2*pi*n) / (N-1));
 end
 hw = hd .* hannwin;

 [h omega] = freqz(hw,1,50);
 mag_h = abs(h);
 figure(1);
 subplot(2,1,1);
 plot(omega/pi,mag_h);

 xlabel('frequency normalised to 1 -->');
 ylabel('Gain in dB-->');
 title('Magnitude Response of LPF');
 angle_h = angle(h);
 subplot(2,1,2);
 plot(omega/pi,angle_h)
 xlabel('frequency normalised to 1 -->');
 ylabel('Phase-->');
 title('Phase Response of LPF');
```

RESULT:

Thus the magnitude and phase response of the FIR low pass filter using Hanning Window technique is plotted using MATLAB.

`

FLOWCHART:


Start
get the Order and cutoff frequency of the High pass filter
Compute the desired Impulse response hd(n) of the filter
Get the attenuation values for passband and stop band
Obtain Rectangular Window function wh=1
Calculate the impulse response of the filter h(n)=hd(n)*wh for -(N-1)/2 to (N-1)/2
Compute frequency response of filter
Compute the magnitude using abs command
Compute the phase using angle command
Plot the magnitude and phase response
Stop

`

DATE:17/09/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

Expt. No. 7b. DESIGN OF FIR HIGH PASS FILTER USING RECTANGULAR WINDOW TECHNIQUE

AIM:

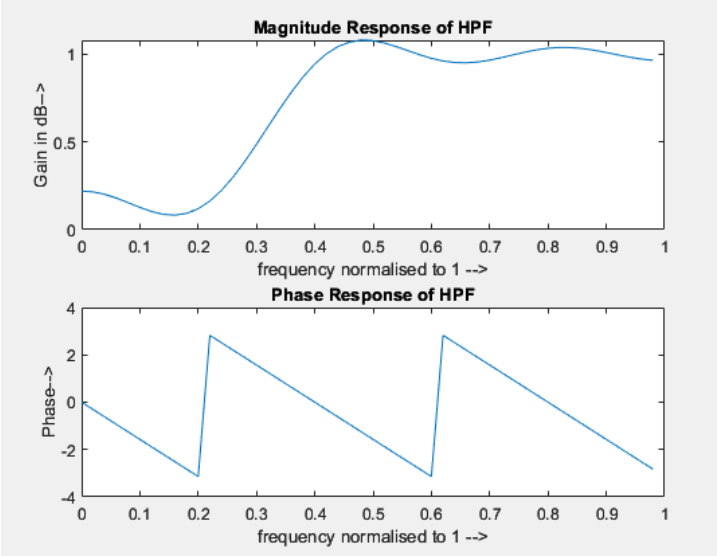
To write a program in MATLAB to plot the magnitude and phase response of FIR High pass filter using Rectangular Window Technique.

ALGORITHM:

1. Clear the command window.
2. Get the order and cut off frequency of the filter.
3. Compute the desired impulse response $h_d(n)$ of the filter.
4. Obtain different window functions
   - Rectangular window : wh =1
   - Hamming Window : wh = 0.54 – 0.46 cos(2*pi*n/(N-1))
   - Hanning Window : wh = 0.5 – 0.5 cos(2*pi*n/(N-1))
   - Blackman Window : wh = 0.42 – 0.5 cos(2*pi*n/(N-1)) +0.08 cos(4*pi*n/(N-1)) 5. Calculate the impulse response of the filter h(n) = $h_d(n)$ * wh for n= -(N-1)/2 to (N+1)/2 6. Compute the frequency response of the filter.
7. Compute the magnitude using abs command.
8. Compute the phase using angle command.
9. Plot the magnitude and phase response.

`

OUTPUT:



COMMAND PROMPT:

```
Program for FIR High pass filter using windowing technique

Enter the order of the filter: 11
Enter the cut off frequency: 1200

Enter the sampling frequency greater than 2400
Enter the sampling frequency:9000

hd =

    0.0511


rect_win =

     1


hd =

    0.0511    0.0505


hd =

    0.0511    0.0505   -0.0125


rect_win =

     1     1     1


hd =

    0.0511    0.0505   -0.0125   -0.1212


rect_win =

     1     1     1     1


hd =

    0.0511    0.0505   -0.0125   -0.1212   -0.2244


rect_win =

     1     1     1     1     1     1     1     1     1


hd =

    0.0511    0.0505   -0.0125   -0.1212   -0.2244    0.7333   -0.2244   -0.1212   -0.0125    0.0505


rect_win =

     1     1     1     1     1     1     1     1     1     1


hd =

    0.0511    0.0505   -0.0125   -0.1212   -0.2244    0.7333   -0.2244   -0.1212   -0.0125    0.0505    0.0511


rect_win =

     1     1     1     1     1     1     1     1     1     1     1
```

`

PROGRAM:

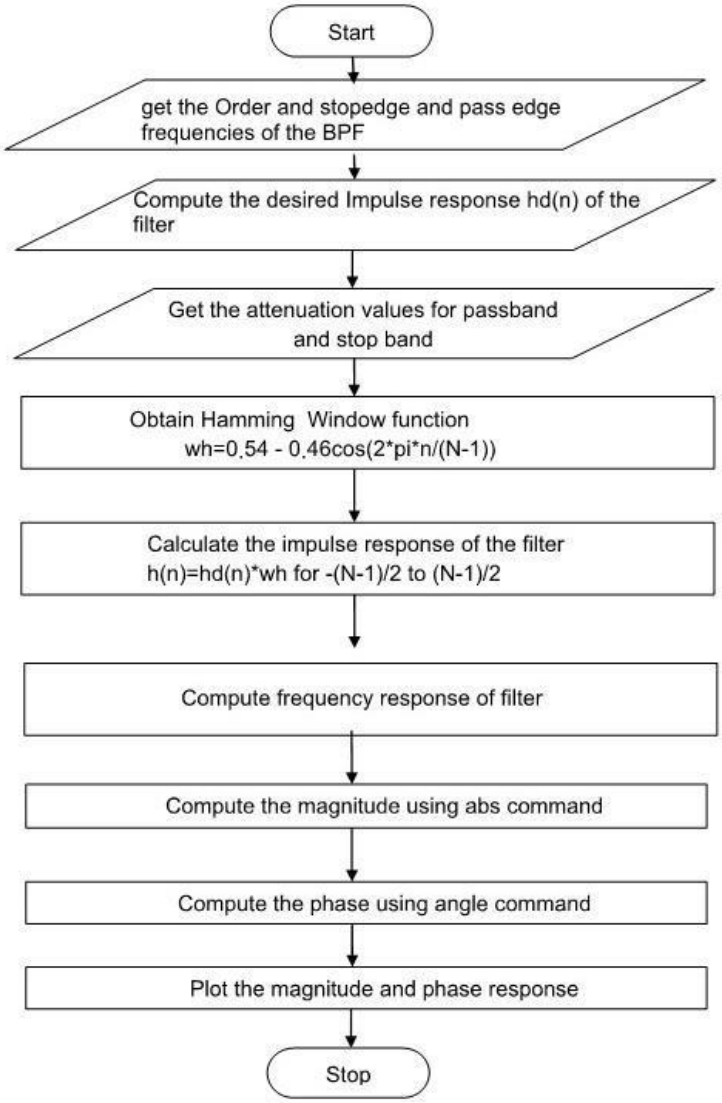
```
clc
close all
clear all
fprintf('Program for FIR High pass filter using windowing technique\n\n');
N =input('Enter the order of the filter: ');
fc=input('Enter the cut off frequency: ');
fs_min = 2 * fc;
fprintf('\nEnter the sampling frequency greater than %d\n',fs_min');
fs_sf=input('Enter the sampling frequency:');
wc=2*pi*fc / fs_sf;
alp = (N-1)/2;
for n = 1 : 1 : N
 if (n - 1) == alp
 hd(n) = (pi - wc) / pi
 else
 hd(n)=(sin((n-1-alp)*pi))-(sin(n-1-alp)*wc)/(pi*(n-1-alp))
 end
 rect_win(n)=1
end

hw=hd .* rect_win

[h omega] = freqz(hw,1,50);
mag_h = abs(h);
figure(1);
subplot(2,1,1);
plot(omega/pi,mag_h);

xlabel('frequency normalised to 1 -->');
ylabel('Gain in dB-->');
title('Magnitude Response of HPF');
angle_h = angle(h);
subplot(2,1,2);
plot(omega/pi,angle_h)
xlabel('frequency normalised to 1 -->');
ylabel('Phase-->');
title('Phase Response of HPF');
```

RESULT:

Thus, the magnitude and phase response of the FIR High pass filter using Rectangular Window technique is plotted using MATLAB.

`

FLOWCHART:


Start
get the Order and stopedge and pass edge frequencies of the BPF
Compute the desired Impulse response hd(n) of the filter
Get the attenuation values for passband and stop band
Obtain Hamming Window function
wh=0.54 - 0.46cos(2*pi*n/(N-1))
Calculate the impulse response of the filter
h(n)=hd(n)*wh for -(N-1)/2 to (N-1)/2
Compute frequency response of filter
Compute the magnitude using abs command
Compute the phase using angle command
Plot the magnitude and phase response
Stop

`

DATE:17/09/2024
R

NAME: ARUNKARTHICK

REG.NO:412522106013

Expt. No. 7c. DESIGN OF FIR BAND PASS FILTER USING HAMMING WINDOW TECHNIQUE

AIM:

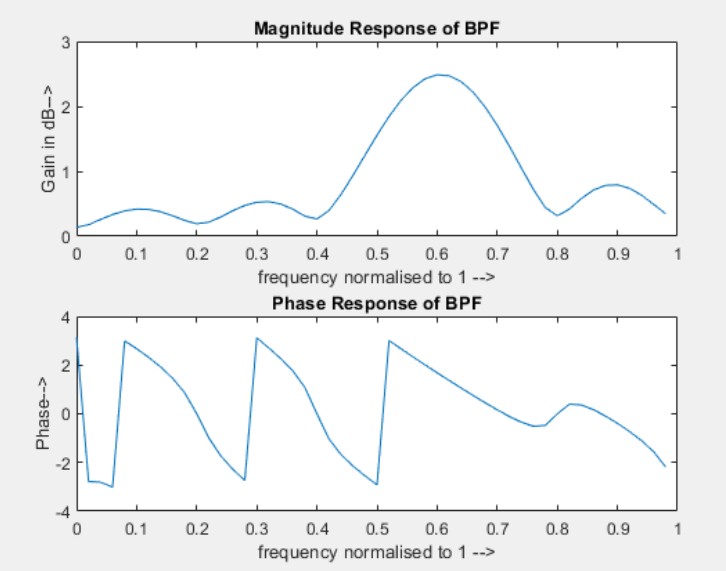
To write a program in MATLAB to plot the magnitude and phase response of FIR band pass filter using Hamming Window technique.

ALGORITHM:

1. Clear the command window.
2. Get the order and cut off frequency of the filter.
3. Compute the desired impulse response $h_d(n)$ of the filter.
4. Obtain different window functions
   - Rectangular window : wh =1
   - Hamming Window : wh = 0.54 – 0.46 cos(2*pi*n/(N-1))
   - Hanning Window : wh = 0.5 – 0.5 cos(2*pi*n/(N-1))
   - Blackman Window : wh = 0.42 – 0.5 cos(2*pi*n/(N-1)) +0.08 cos(4*pi*n/(N-1))
5. Calculate the impulse response of the filter h(n) = $h_d(n)$ * wh for n= -(N-1)/2 to (N+1)/2
6. Compute the frequency response of the filter.
7. Compute the magnitude using abs command.
8. Compute the phase using angle command.
9. Plot the magnitude and phase response.

`

OUTPUT:



COMMAND PROMPT:

```
Enter the order of the filter: 11
Enter the stop edge frequency1:1000
Enter the pass edge frequency1:3000
Enter the pass edge frequency2:6000
Enter the stop edge frequency2:9000

Enter the sampling frequency greater than 18000
Enter the sampling frequency:20000

Enter the passband ripple in dB:3
Enter the stopband attenuation in dB:60

hd =

    0.0575


hd =

    0.0575   -0.8943


hd =

    0.0575   -0.8943    0.5737


hd =

    0.0575   -0.8943    0.5737    0.4514   -1.2035    0.3000    0.6986


hd =

    0.0575   -0.8943    0.5737    0.4514   -1.2035    0.3000    0.6986   -0.7242


hd =

    0.0575   -0.8943    0.5737    0.4514   -1.2035    0.3000    0.6986   -0.7242   -0.6019


hd =

    0.0575   -0.8943    0.5737    0.4514   -1.2035    0.3000    0.6986   -0.7242   -0.6019    1.0078


hd =

    0.0575   -0.8943    0.5737    0.4514   -1.2035    0.3000    0.6986   -0.7242   -0.6019    1.0078    0.0575
```

`

PROGRAM:

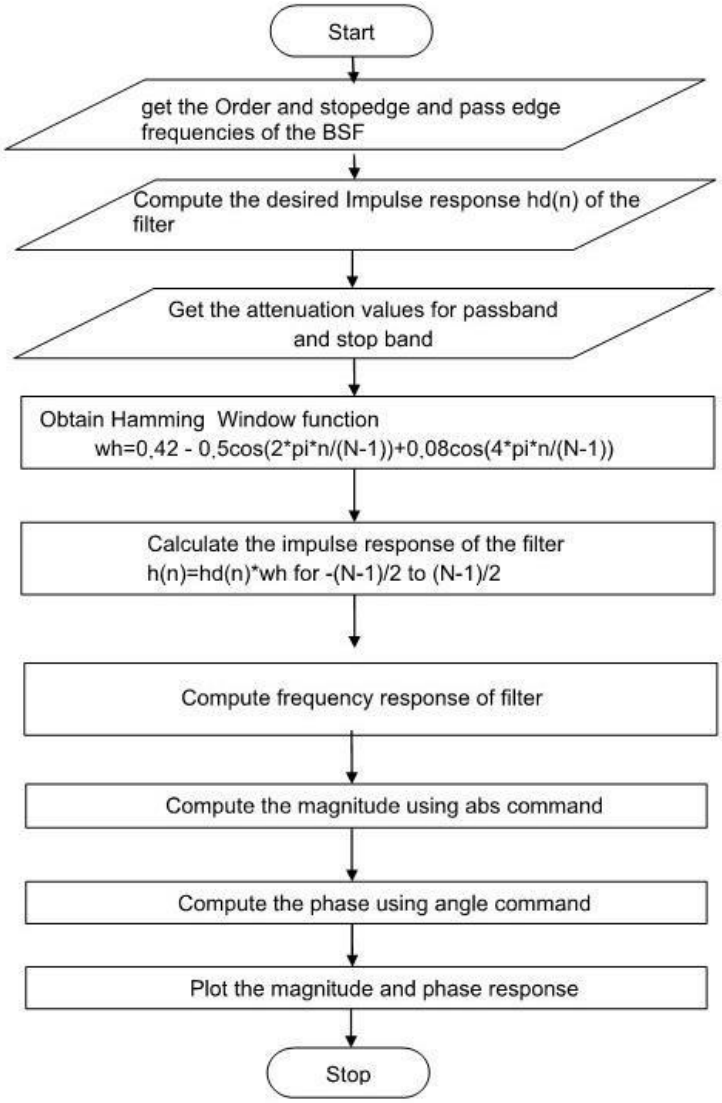
```
clc;
clear all;
close all;
N =input('Enter the order of the filter: ');
fs1=input('Enter the stop edge frequency1:');
fc1=input('Enter the pass edge frequency1:');
fc2=input('Enter the pass edge frequency2:');
fs_min = 2*fs2;
fprintf('\nEnter the sampling frequency greater than %d\n',fs_min');
fs_sf=input('Enter the sampling frequency:');
rp = input('\nEnter the passband ripple in dB:');
rs = input('Enter the stopband attenuation in dB:');
ws1=2*pi*fs1/fs_sf;
wc1=2*pi*fc1/fs_sf;
wc2=2*pi*fc2/fs_sf;
ws2=2*pi*fs2/fs_sf;
alp = (N-1)/2;
for n = 1 : 1 : N
 if (n - 1) == alp
 hd(n) = (wc2-wc1) / pi
 else
 hd(n)=(sin((n-1-alp)*wc2))-(sin(n-1-alp)*wc1)/(pi*(n-1-alp))
 hammwin(n) = 0.54 - 0.46 * (cos(2*pi*n) / (N-1));
end
hw = hd .* hammwin;
[h omega] = freqz(hw,1,50);
mag_h = abs(h);
figure(1);
subplot(2,1,1);
plot(omega/pi,mag_h);
xlabel('frequency normalised to 1 -->');
ylabel('Gain in dB-->');
title('Magnitude Response of BPF');
angle_h = angle(h);
subplot(2,1,2);
plot(omega/pi,angle_h)
xlabel('frequency normalised to 1 -->');
ylabel('Phase-->');
title('Phase Response of BPF');
```

RESULT:

Thus, the magnitude and phase response of the FIR Band Pass filter using Hamming Window technique is plotted using MATLAB.

`

FLOWCHART:


Start
get the Order and stopedge and pass edge frequencies of the BSF
Compute the desired Impulse response hd(n) of the filter
Get the attenuation values for passband and stop band
Obtain Hamming Window function
wh=0.42 - 0.5cos(2*pi*n/(N-1))+0.08cos(4*pi*n/(N-1))
Calculate the impulse response of the filter
h(n)=hd(n)*wh for -(N-1)/2 to (N-1)/2
Compute frequency response of filter
Compute the magnitude using abs command
Compute the phase using angle command
Plot the magnitude and phase response
Stop

`

DATE:17/09/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

Expt. No. 7d. DESIGN OF FIR BAND STOP FILTER USING BLACKMAN WINDOW TECHNIQUE

AIM:

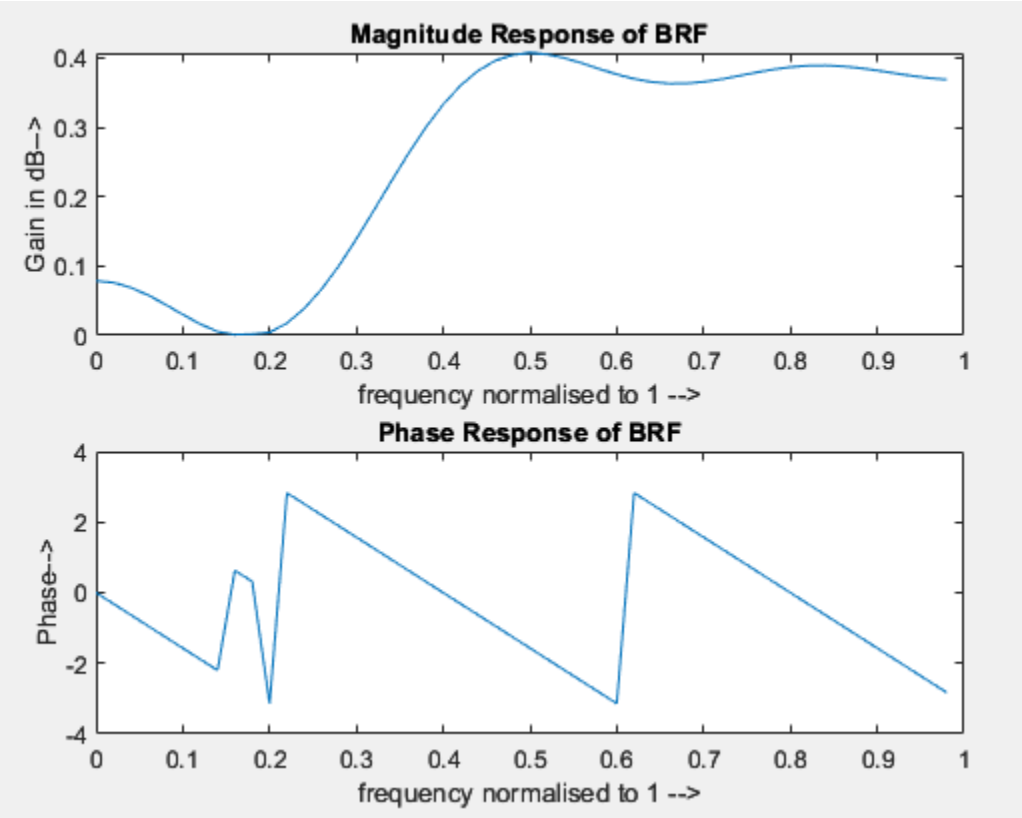
To write a program in MATLAB to plot the magnitude and phase response of FIR Band Stop filter using Blackman Window Technique.

ALGORITHM:

1. Clear the command window.
2. Get the order and cut off frequency of the filter.
3. Compute the desired impulse response $h_d(n)$ of the filter.
4. Obtain different window functions
   - Rectangular window : wh =1
   - Hamming Window : wh = 0.54 – 0.46 cos(2*pi*n/(N-1))
   - Hanning Window : wh = 0.5 – 0.5 cos(2*pi*n/(N-1))
   - Blackman Window : wh = 0.42 – 0.5 cos(2*pi*n/(N-1)) +0.08 cos(4*pi*n/(N-1))
5. Calculate the impulse response of the filter h(n) = $h_d(n)$ * wh for n= -(N-1)/2 to (N+1)/2
6. Compute the frequency response of the filter.
7. Compute the magnitude using abs command.
8. Compute the phase using angle command.
9. Plot the magnitude and phase response.

`

OUTPUT:




```
Enter the order of the filter: 11
Enter the pass edge frequency1:1000
Enter the stop edge frequency1:3000
Enter the stop edge frequency2:6000
Enter the pass edge frequency2:9000

Enter the sampling frequency greater than 18000
Enter the sampling frequency:20000

Enter the passband ripple in dB:3
Enter the stopband attenuation in dB:60
```

`

PROGRAM:

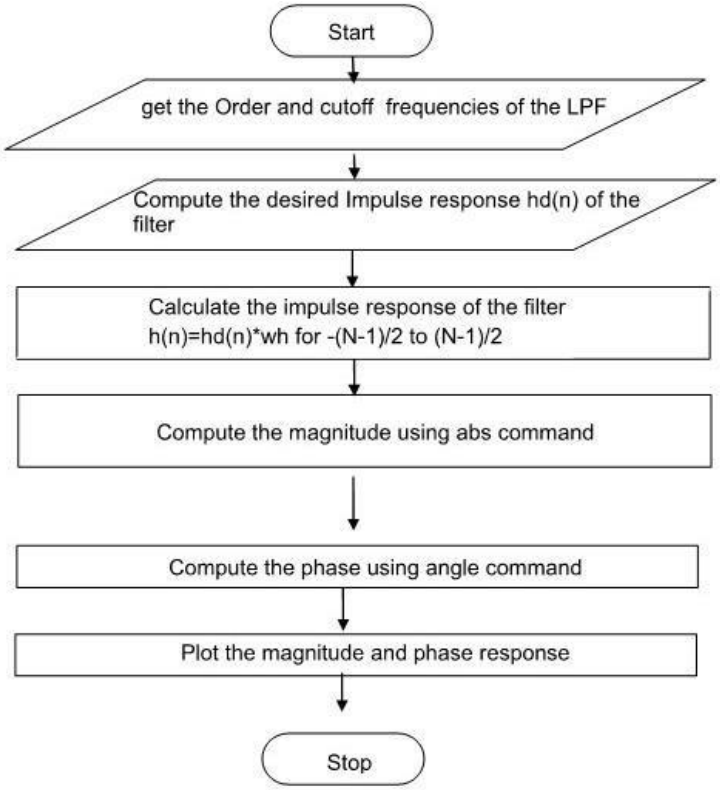
```
clc;
clear all;
close all;
N =input('Enter the order of the filter: ');
fp1=input('Enter the pass edge frequency1:');
fc1=input('Enter the stop edge frequency1:');
fc2=input('Enter the stop edge frequency2:');
fp2=input('Enter the pass edge frequency2:');
fs_min = 2*fp2;
fprintf('\nEnter the sampling frequency greater than %d\n',fs_min');
rp = input('\nEnter the passband ripple in dB:');
rs = input('Enter the stopband attenuation in dB:');
wp1=2*pi*fp1/fs_sf;
wc1=2*pi*fc1/fs_sf;
wc2=2*pi*fc2/fs_sf;
wp2=2*pi*fp2/fs_sf;
alp = (N-1)/2;
for n = 1 : 1 : N
 if (n - 1) == alp
 hd(n) =1 - ((wc2-wc1 )/ pi)
 else
 hd(n)=((sin((n-1-alp)*wc1))-(sin(n-1-alp)*wc2)+ sin((n-1-alp)*pi))/(pi*(n-1-alp))
 end
 blackwin(n) = 0.42 - 0.5 * (cos(2*pi*n) / (N-1)) + 0.08 * (cos(4*pi*n) / (N-1));
hw = hd .* blackwin;
[h omega] = freqz(hw,1,50);
mag_h = abs(h);
figure(1);
subplot(2,1,1);
plot(omega/pi,mag_h);
xlabel('frequency normalised to 1 -->');
ylabel('Gain in dB-->');
title('Magnitude Response of BRF');
angle_h = angle(h);
subplot(2,1,2);
plot(omega/pi,angle_h)
xlabel('frequency normalised to 1 -->');
ylabel('Phase-->');
title('Phase Response of BRF');
```

RESULT:

Thus, the magnitude and phase response of the FIR Band Stop filter using Blackman Window technique is plotted using MATLAB.

`

FLOWCHART:


Start
get the Order and cutoff frequencies of the LPF
Compute the desired Impulse response hd(n) of the filter
Calculate the impulse response of the filter
h(n)=hd(n)*wh for -(N-1)/2 to (N-1)/2
Compute the magnitude using abs command
Compute the phase using angle command
Plot the magnitude and phase response
Stop

`

DATE:17/09/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

Expt. No. 8a. DESIGN OF FIR LOW PASS FILTER USING FOURIER SERIES METHOD

AIM:

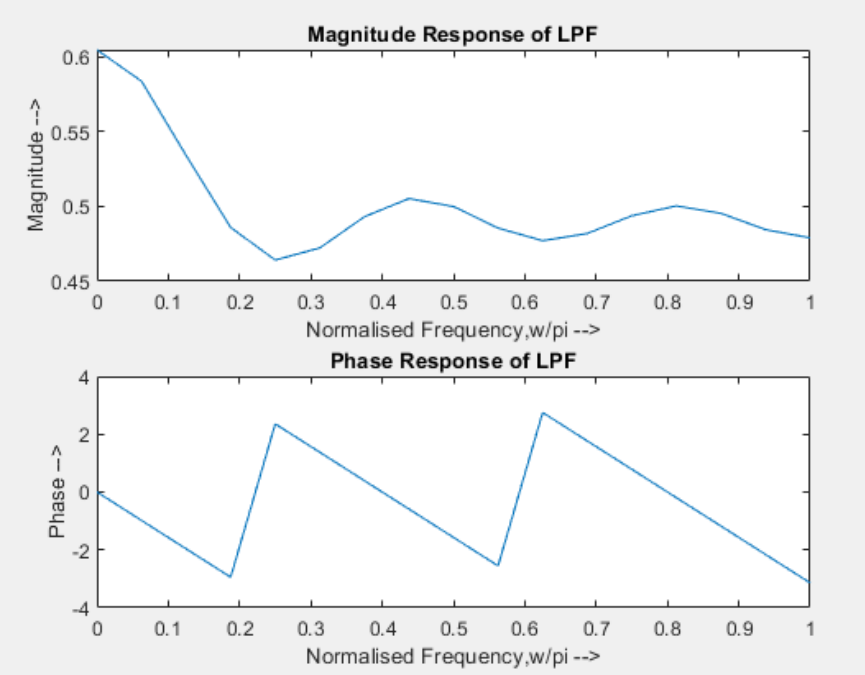
To write a program in MATLAB to plot the magnitude and phase response of FIR low pass filter using Fourier Series method.

ALGORITHM:

10. Clear the command window.
11. Get the order and cut off frequency of the filter.
12. Compute the desired impulse response $h_d(n)$ of the filter.
13. Calculate the impulse response of the filter $h_n(n) = h_d(n)$ for n= -(N-1)/2 to (N+1)/2 14. Compute the magnitude using abs command.
15. Compute the phase using angle command.
16. Plot the magnitude and phase response.

`

OUTPUT:


Magnitude Response of LPF
Magnitude -->
0.6
0.55
0.5
0.45
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Normalised Frequency,w/pi -->
Phase Response of LPF
Phase -->
4
2
0
-2
-4
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Normalised Frequency,w/pi -->

`

PROGRAM:

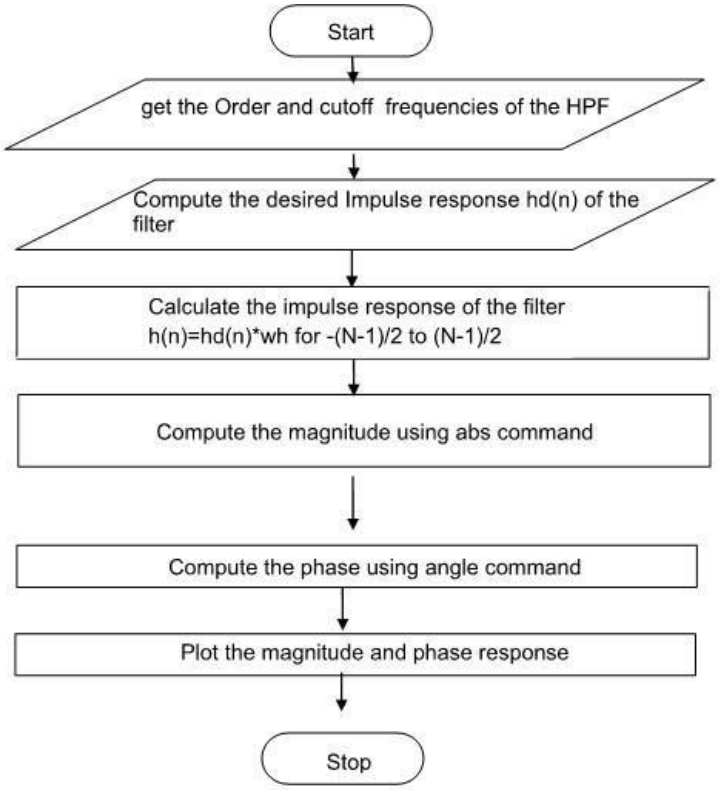
```
clc
clear all
close all
wc=.5*pi;
N=11;
hd=zeros(1,N);
hd(1)=wc/pi;
n = 1:1:((N-1)/2)+1;
hd(n+1) = (sin(wc*n)) . / (pi*n);
hn(n) = hd(n)
a=(N-1)/2;
w=0: pi/16:pi;
Hw1=hn(1)*exp(-j*w*a);
Hw2=0;

for m = 1:1:a;
 Hw3 = hn(m+1) * ((exp(j*w*(m-a))) + exp(-j*w*(m+a)));
 Hw2 = Hw2 + Hw3;
end
Hw = Hw2 + Hw1

mag_h=abs(Hw)
subplot(2,1,1);
plot(w/pi,mag_h);
xlabel('Normalised Frequency,w/pi -->');
ylabel('Magnitude -->');
title('Magnitude Response of LPF');
angle_h=angle(Hw);
subplot(2,1,2);
plot(w/pi,angle_h);
xlabel('Normalised Frequency,w/pi -->');
ylabel('Phase -->');
title('Phase Response of LPF');
```

RESULT:

Thus the magnitude and phase response of the FIR low pass filter using Fourier Series method is plotted using MATLAB.

`

FLOWCHART:


Start
get the Order and cutoff frequencies of the HPF
Compute the desired Impulse response hd(n) of the filter
Calculate the impulse response of the filter h(n)=hd(n)*wh for -(N-1)/2 to (N-1)/2
Compute the magnitude using abs command
Compute the phase using angle command
Plot the magnitude and phase response
Stop

`

DATE:17/09/2024

NAME: ARUNKARTHICK R

REG.NO:412522106013

Expt. No. 8b. DESIGN OF FIR HIGH PASS FILTER USING FOURIER SERIES METHOD

AIM:

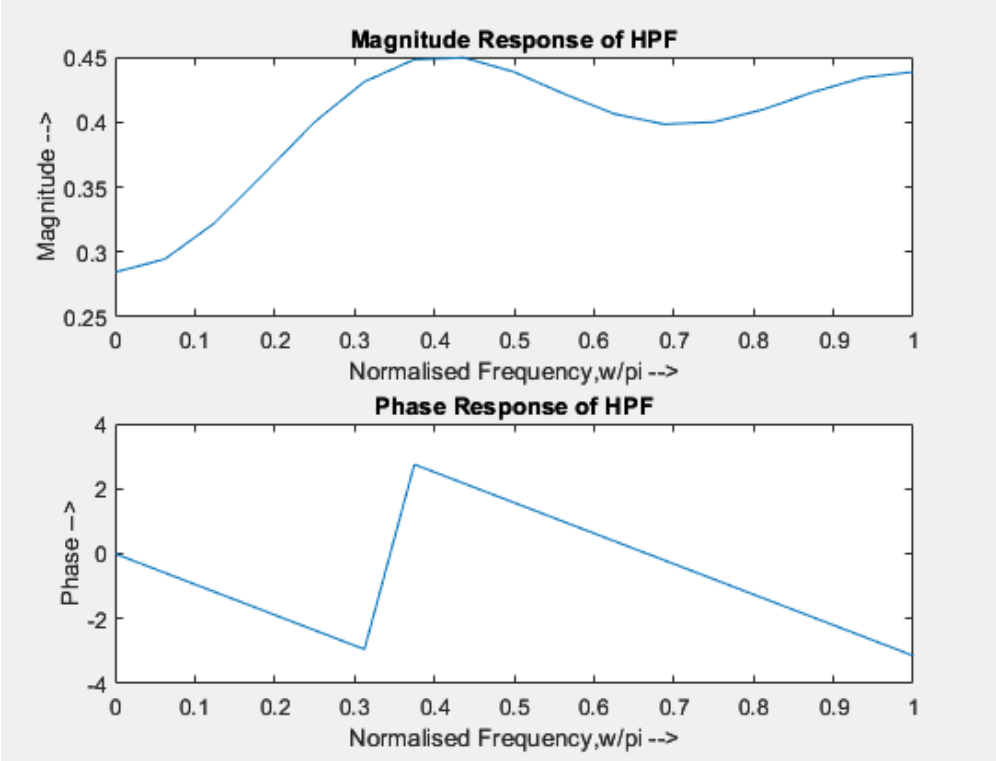
To write a program in MATLAB to plot the magnitude and phase response of FIR high pass filter using Fourier Series method.

ALGORITHM:

1. Clear the command window.
2. Get the order and cut off frequency of the filter.
3. Compute the desired impulse response $h_d(n)$ of the filter.
4. Calculate the impulse response of the filter $h_n(n) = h_d(n)$ for n= -(N-1)/2 to (N+1)/2
5. Compute the magnitude using abs command.
6. Compute the phase using angle command.
7. Plot the magnitude and phase response.

`

OUTPUT:


Magnitude Response of HPF
Magnitude -->
0.45
0.4
0.35
0.3
0.25
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalised Frequency,w/pi -->
Phase Response of HPF
Phase -->
4
2
0
-2
-4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalised Frequency,w/pi -->

`

PROGRAM:

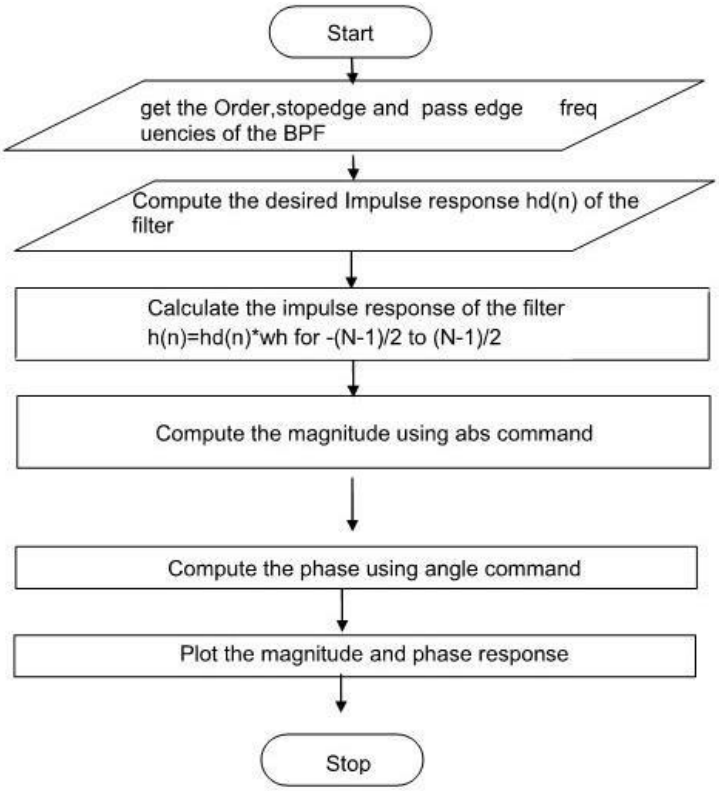
```
clc
clear all
close all
wc=.6*pi;
N=7;
hd=zeros(1,N);
hd(1)=1-(wc/pi);
n = 1:1:((N-1)/2)+1;
hd(n+1) = (-sin(wc*n)) . / (pi*n);
hn(n) = hd(n)
a=(N-1)/2;
w=0: (pi/16):pi;
Hw1=hn(1)*exp(-j*w*a);
Hw2=0;

for m = 1:1:a;
 Hw3 = hn(m+1) * ((exp(j*w*(m-a))) + exp(-j*w*(m+a)));
 Hw2 = Hw2 + Hw3;
end
Hw = Hw2 + Hw1

mag_h=abs(Hw)
subplot(2,1,1);
plot(w/pi,mag_h);
xlabel('Normalised Frequency,w/pi -->');
ylabel('Magnitude -->');
title('Magnitude Response of HPF');
angle_h=angle(Hw);
subplot(2,1,2);
plot(w/pi,angle_h);
xlabel('Normalised Frequency,w/pi -->');
ylabel('Phase -->');
title('Phase Response of HPF');
```

RESULT:

Thus the magnitude and phase response of the FIR high pass filter using Fourier Series method is plotted using MATLAB.

`

FLOWCHART:


Start
get the Order,stopedge and pass edge freq uencies of the BPF
Compute the desired Impulse response hd(n) of the filter
Calculate the impulse response of the filter h(n)=hd(n)*wh for -(N-1)/2 to (N-1)/2
Compute the magnitude using abs command
Compute the phase using angle command
Plot the magnitude and phase response
Stop

`

DATE:17/09/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

Expt. No. 8c. DESIGN OF FIR BAND PASS FILTER USING FOURIER SERIES METHOD

AIM:

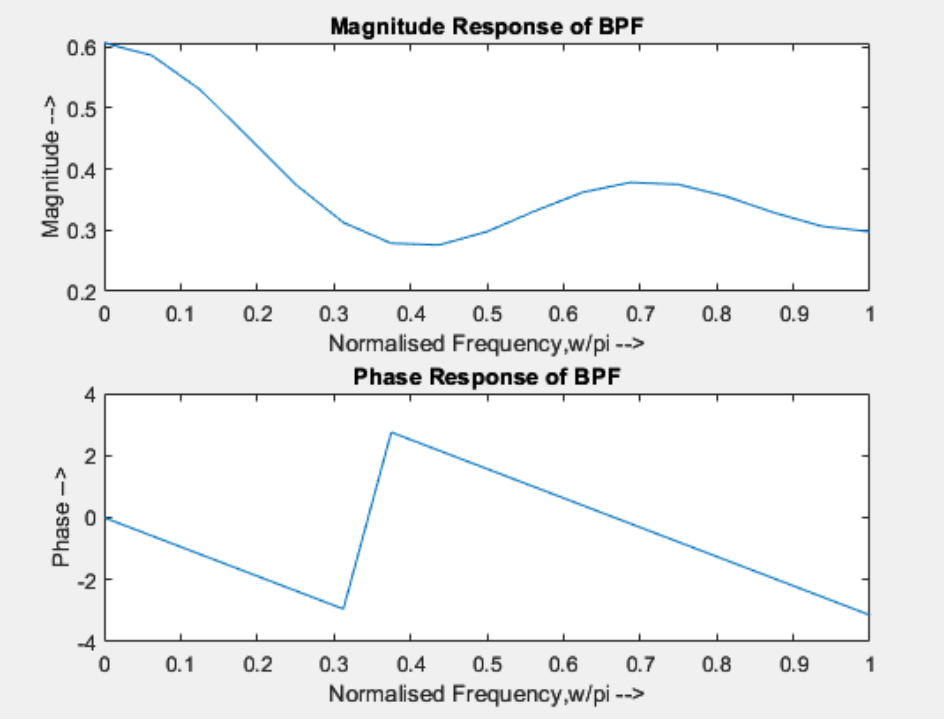
To write a program in MATLAB to plot the magnitude and phase response of FIR band pass filter using Fourier Series method.

ALGORITHM:

1. Clear the command window.
2. Get the order and cut off frequency of the filter.
3. Compute the desired impulse response $h_d(n)$ of the filter.
4. Calculate the impulse response of the filter $h_n(n) = h_d(n)$ for n= -(N-1)/2 to (N+1)/2
5. Compute the magnitude using abs command.
6. Compute the phase using angle command.
7. Plot the magnitude and phase response.

`

OUTPUT:


Magnitude Response of BPF
Magnitude -->
0.6
0.5
0.4
0.3
0.2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalised Frequency,w/pi -->
Phase Response of BPF
Phase -->
4
2
0
-2
-4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalised Frequency,w/pi -->

`

PROGRAM:

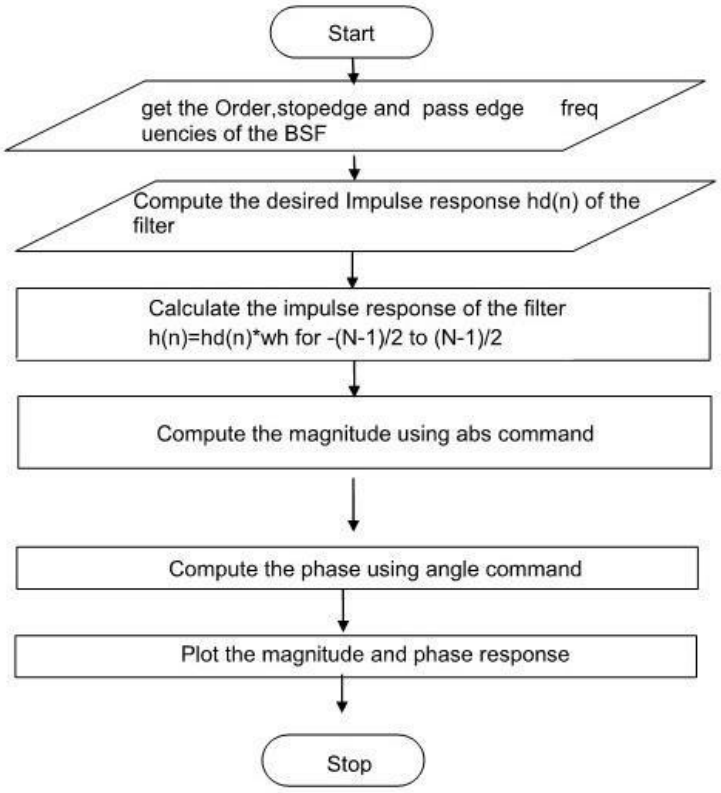
```
Clc
clear all
close all
wc1=.375*pi;
wc2=.75*pi;
N=7;
hd=zeros(1,N);

hd(1)=(wc2-wc1)/pi;
n = 1:1:((N-1)/2)+1;
hd(n+1) = ((sin(wc2*n))-(sin(wc1*n))) . / (pi*n);
hn(n) = hd(n)
a=(N-1)/2;
w=0: (pi/16):pi;
Hw1=hn(1)*exp(-j*w*a);
Hw2=0;

for m = 1:1:a;
 Hw3 = hn(m+1) * ((exp(j*w*(m-a))) + exp(-j*w*(m+a)));
 Hw2 = Hw2 + Hw3;
end
Hw = Hw2 + Hw1

mag_h=abs(Hw)
subplot(2,1,1);
plot(w/pi,mag_h);
xlabel('Normalised Frequency,w/pi -->');
ylabel('Magnitude -->');
title('Magnitude Response of BPF');
angle_h=angle(Hw);
subplot(2,1,2);
plot(w/pi,angle_h);
xlabel('Normalised Frequency,w/pi -->');
ylabel('Phase -->');
title('Phase Response of BPF');
```

RESULT:

Thus the magnitude and phase response of the FIR band pass filter using Fourier Series method is plotted using MATLAB.

`

FLOWCHART:


Start
get the Order,stopedge and pass edge freq uencies of the BSF
Compute the desired Impulse response hd(n) of the filter
Calculate the impulse response of the filter h(n)=hd(n)*wh for -(N-1)/2 to (N-1)/2
Compute the magnitude using abs command
Compute the phase using angle command
Plot the magnitude and phase response
Stop

`

DATE:17/09/2024

NAME: ARUNKARTHICK R

REG.NO:412522106013

Expt. No. 8d. DESIGN OF FIR BAND STOP FILTER USING FOURIER SERIES METHOD

AIM:

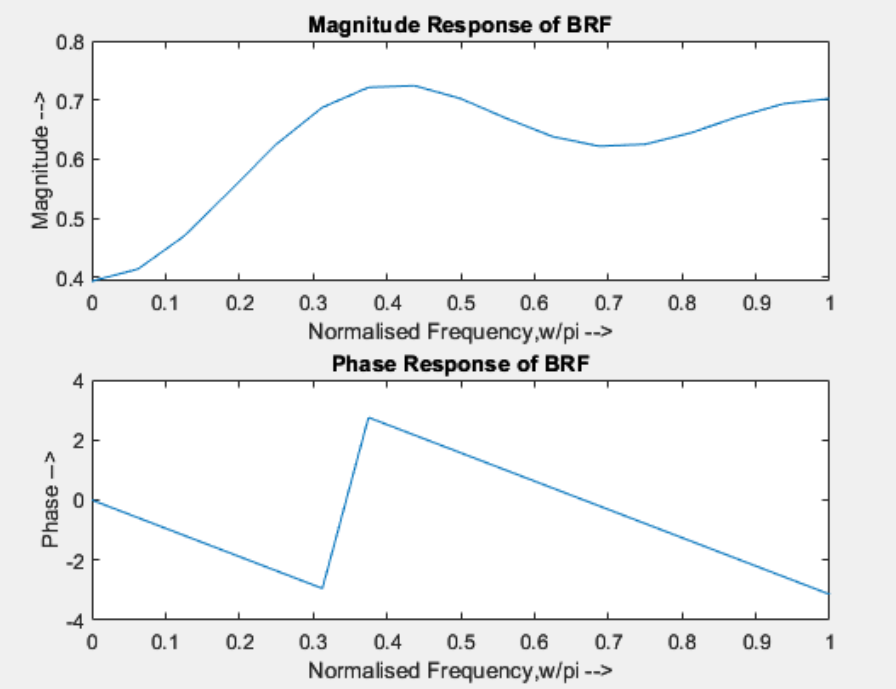
To write a program in MATLAB to plot the magnitude and phase response of FIR band stop filter using Fourier Series method.

ALGORITHM:

1. Clear the command window.
2. Get the order and cut off frequency of the filter.
3. Compute the desired impulse response $h_d(n)$ of the filter.
4. Calculate the impulse response of the filter $h_n(n) = h_d(n)$ for n= -(N-1)/2 to (N+1)/2
5. Compute the magnitude using abs command.
6. Compute the phase using angle command.
7. Plot the magnitude and phase response.

`

OUTPUT:


Magnitude Response of BRF
Magnitude -->
0.8
0.7
0.6
0.5
0.4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalised Frequency,w/pi -->
Phase Response of BRF
Phase -->
4
2
0
-2
-4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalised Frequency,w/pi -->

`

PROGRAM:

```
clc
clear all
close all
wc1=.375*pi;
wc2=.75*pi;
N=7;
hd=zeros(1,N);

hd(1)=1-((wc2-wc1)/pi);
n = 1:1:((N-1)/2)+1;
hd(n+1) = (((sin(wc1*n))-(sin(wc2*n))) . / (pi*n));
hn(n) = hd(n)
a=(N-1)/2;
w=0: (pi/16):pi;
Hw1=hn(1)*exp(-j*w*a);
Hw2=0;

for m = 1:1:a;
 Hw3 = hn(m+1) * ((exp(j*w*(m-a))) + exp(-j*w*(m+a)));
 Hw2 = Hw2 + Hw3;
end
Hw = Hw2 + Hw1

mag_h=abs(Hw)
subplot(2,1,1);
plot(w/pi,mag_h);
xlabel('Normalised Frequency,w/pi -->');
ylabel('Magnitude -->');
title('Magnitude Response of BRF');
angle_h=angle(Hw);
subplot(2,1,2);
plot(w/pi,angle_h);
xlabel('Normalised Frequency,w/pi -->');
ylabel('Phase -->');
title('Phase Response of BRF');
```

RESULT:

Thus the magnitude and phase response of the FIR band stop filter using Fourier Series method is plotted using MATLAB.

`

`

DATE:08/10/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

9. STUDY OF ARCHITECTURE OF TMS320C5x PROCESSOR

This chapter provides an overview of the architectural structure of the ‟C5x, which consists of the buses, on-chip memory, central processing unit (CPU), and on-chip peripherals. The ‟C5x uses an advanced, modified Harvard-type architecture based on the ‟C25 architecture and maximizes processing power with separate buses for program memory and data memory. The instruction set supports data transfers between the two memory spaces. The diagram shows a functional block diagram of the ‟C5x. All ‟C5x DSPs have the same CPU structure; however, they have different on-chip memory configurations and on-chip peripherals.

Bus Structure

Separate program and data buses allow simultaneous access to program instructions and data, providing a high degree of parallelism. For example, while data is multiplied, a previous product can be loaded into, added to, or subtracted from the accumulator and, at the same time, a new address can be generated. Such parallelism supports a powerful set of arithmetic, logic, and bit-manipulation operations that can all be performed in a single machine cycle. In addition, the ‟C5x includes the control mechanisms to manage interrupts, repeated operations, and function calling. The ‟C5x architecture is built around four major buses:

· Program bus (PB)
· Program address bus (PAB)
· Data read bus (DB)
· Data read address bus (DAB)

The PAB provides addresses to program memory space for both reads and writes. The PB also carries the instruction code and immediate operands from program memory space to the CPU. The DB interconnects various elements of the CPU to data memory space. The program and data buses can work together to transfer data from on-chip data memory and internal or external program memory to the multiplier for single-cycle multiply/accumulate operations.

Central Processing Unit (CPU)

The ‟C5x CPU consists of these elements:

· Central arithmetic logic unit (CALU)
· Parallel logic unit (PLU)
· Auxiliary register arithmetic unit (ARAU)
· Memory-mapped registers
· Program controller

The ‟C5x CPU maintains source-code compatibility with the ‟C1x and ‟C2x generations while achieving high performance and greater versatility. Improvements include a 32-bit accumulator buffer, additional scaling capabilities, and a host of new instructions. The instruction set exploits the additional hardware features and is flexible in a wide range of applications. Data management has been improved through the use of new block move instructions and memory-mapped register instructions. See Chapter 3, Central Processing Unit (CPU).

`

Central Arithmetic Logic Unit (CALU)

The CPU uses the CALU to perform 2s-complement arithmetic. The CALU consists of these elements: · 16-bit x 16-bit multiplier

· 32-bit arithmetic logic unit (ALU)

· 32-bit accumulator (ACC)

· 32-bit accumulator buffer (ACCB)

· Additional shifters at the outputs of both the accumulator and the product register (PREG)

Parallel Logic Unit (PLU)

The CPU includes an independent PLU, which operates separately from, but in parallel with, the ALU. The PLU performs Boolean operations or the bit manipulations required of high-speed controllers. The PLU can set, clear, test, or toggle bits in a status register, control register, or any data memory location. The PLU

provides a direct logic operation path to data memory values without affecting the contents of the ACC or PREG. Results of a PLU function are written back to the original data memory location.

Auxiliary Register Arithmetic Unit (ARAU)

The CPU includes an unsigned 16-bit arithmetic logic unit that calculates indirect addresses by using inputs from the auxiliary registers (ARs), index register (INDX), and auxiliary register compare register (ARCR). The ARAU can auto-index the current AR while the data memory locationis being addressed and can index either by ±1 or by the contents of the INDX. As a result, accessingdata does not require the CALU for address manipulation; therefore, the CALU is free for other operations in parallel.

Memory-Mapped Registers

The ‟C5x has 96 registers mapped into page 0 of the data memory space. All ‟C5x DSPs have 28 CPU registers and 16 input/output (I/O) port registers but have different numbers of peripheral and reserved registers. Since the memory-mapped registers are a component of the data memory space, they can be written to and read from in the same way as any other data memory location. The memory-mapped registers are used for indirect data address pointers, temporary storage, CPU status and control, or integer arithmetic processing through the ARAU.

Program Controller

The program controller contains logic circuitry that decodes the operational instructions, manages the CPU pipeline, stores the status of CPU operations, and decodes the conditional operations. Parallelism of architecture lets the ‟C5x perform three concurrent memory operations in any given machine cycle: fetch an instruction, read an operand, and write an operand.

The program controller consists of these elements:

· Program counter

· Status and control registers

· Hardware stack

· Address generation logic

· Instruction register

`

On-Chip Memory

The ‟C5x architecture contains a considerable amount of on-chip memory to aid in system performance and integration:

· Program read-only memory (ROM)
· Data/program dual-access RAM (DARAM)
· Data/program single-access RAM (SARAM)

The ‟C5x has a total address range of 224K words x 16 bits. The memory space is divided into four individually selectable memory segments: 64K-word program memory space, 64K-word local data memory space, 64K-word input/ output ports, and 32K-word global data memory space.

Program ROM

All ‟C5x DSPs carry a 16-bit on-chip maskable programmable ROM. The ‟C50 and ‟C57S DSPs have boot loader code resident in the on-chip ROM, all other ‟C5x DSPs offer the boot loader code as an option. This memory is used for booting program code from slower external ROM or EPROM to fast on-chip or external RAM. Once the custom program has been booted into RAM, the boot ROM space can be removed from program memory space by setting the MP/MC bit in the processor mode status register (PMST). The on-chip ROM is selected at reset by driving the MP/MC pin low. If the on-chip ROM is not selected, the ‟C5x devices start execution from off-chip memory. The on-chip ROM may be configured with or without boot loader code. However, the on-chip ROM is intended for your specific program.

Data/Program Dual-Access RAM

All ‟C5x DSPs carry a 1056-word
· 16-bit on-chip dual-access RAM (DARAM).

The DARAM is divided into three individually selectable memory blocks: 512-word data or program DARAM block B0, 512-word data DARAM block B1, and 32-word data DARAM block B2. The DARAM is primarily intended to store data values but, when needed, can be used to store programs as well. DARAM blocks B1 and B2 are always configured as data memory; however, DARAM block B0 can be configured by software as data or program memory. The DARAM can be configured in one of two ways:

· All 1056 words
· 16 bits configured as data memory
· 544 words
· 16 bits configured as data memory and 512 words × 16 bits configured as program memory DARAM improves the operational speed of the ‟C5x CPU. The CPU operates with a 4-deep pipeline. In this pipeline, the CPU reads data on the third stage and writes data on the fourth stage. Hence, for a given instruction sequence, the second instruction could be reading data at the same time the first instruction is writing data. The dual data buses (DB and DAB) allow the CPU to read from and write to DARAM in the same machine cycle.

Data / Program Single-Access RAM

All ‟C5x DSPs except the ‟C52 carry a 16-bit on-chip single-access RAM (SARAM) of various sizes (see Table 1–1). Code can be booted from an offchip ROM and then executed at full speed, once it is loaded into the on-chip SARAM. The SARAM can be configured by software in one of three ways: · All SARAM configured as data memory

`

· All SARAM configured as program memory
· SARAM configured as both data memory and program memory

The SARAM is divided into 1K- and/or 2K-word blocks contiguous in address memory space. All ‟C5x CPUs support parallel accesses to these SARAM blocks. However, one SARAM block can be accessed only once per machine cycle. In other words, the CPU can read from or write to one SARAM block while accessing another SARAM block. When the CPU requests multiple accesses, the SARAM schedules the accesses by providing a not-ready condition to the CPU and executing the multiple accesses one cycle at a time. SARAM supports more flexible address mapping than DARAM because SARAM can be mapped to both program and data memory space simultaneously.However, because of simultaneous program and data mapping, an instruction fetch and data fetch that could be performed in one machine cycle with DARAM may take two machine cycles with SARAM.

On-Chip Memory Protection

The ‟C5x DSPs have a maskable option that protects the contents of on-chip memories. When the related bit is set, no externally originating instruction can access the on-chip memory spaces.

On-Chip Peripherals

All ‟C5x DSPs have the same CPU structure; however, they have different onchip peripherals connected to their CPUs. The ‟C5x DSP on-chip peripherals available are:

· Clock generator
· Hardware timer
· Software-programmable wait-state generators
· Parallel I/O ports
· Host port interface (HPI)
· Serial port
· Buffered serial port (BSP)
· Time-division multiplexed (TDM) serial port
· User-maskable interrupts

Clock Generator

The clock generator consists of an internal oscillator and a phase-locked loop (PLL) circuit. The clock generator can be driven internally by a crystal resonator circuit or driven externally by a clock source. The PLL circuit can generate an internal CPU clock by multiplying the clock source by a specific factor, so you can use a clock source with a lower frequency than that of the CPU.

Hardware Timer

A 16-bit hardware timer with a 4-bit prescaler is available. This programmable timer clocks at a rate that is between 1/2 and 1/32 of the machine cycle rate (CLKOUT1), depending upon the timer‟s divide-down ratio. The timer can be stopped, restarted, reset, or disabled by specific status bits.

Software-Programmable Wait-State Generators

Software-programmable wait-state logic is incorporated in ‟C5x DSPs allowing wait-state generation without any external hardware for interfacing with slower off-chip memory and I/O

devices. This feature consists of multiple wait state generating circuits. Each circuit is user-programmable to operate in different wait states for off-chip memory accesses.

Parallel I/O Ports

A total of 64K I/O ports are available, sixteen of these ports are memory-mapped in data memory space. Each of the I/O ports can be addressed by the IN or the OUT instruction. The memory-mapped I/O ports can be accessed with any instruction that reads from or writes to data memory. The IS signal indicates a read or

write operation through an I/O port. The ‟C5x can easily interface with external I/O devices through the I/O ports while requiring minimal off-chip address decoding circuits.

Host Port Interface (HPI)

The HPI available on the ‟C57S and ‟LC57 is an 8-bit parallel I/O port that provides an interface to a host processor. Information is exchanged between the DSP and the host processor through on-chip memory that is accessible to both the host processor and the ‟C57.

Serial Port

Three different kinds of serial ports are available: a general-purpose serial port, a time-division multiplexed (TDM) serial port, and a buffered serial port (BSP). Each ‟C5x contains at least one general-purpose, high speed synchronous, full-duplexed serial port interface that provides direct communication with serial devices such as CODECs, serial analog-to-digital (A/D) converters, and other serial systems. The serial port is capable of operating at up to one-fourth the machine cycle rate (CLKOUT1). The serial port transmitter and receiver are double-buffered and individually controlled by maskable external interrupt signals. Data is framed either as bytes or as words.

Buffered Serial Port (BSP)

The BSP available on the ‟C56 and ‟C57 devices is a full-duplexed, double buffered serial port and an auto buffering unit (ABU). The BSP provides flexibility on the data stream length. The ABU supports high-speed data transfer and reduces interrupt latencies.

TDM Serial Port

The TDM serial port available on the ‟C50, ‟C51, and ‟C53 devices is a fullduplexed serial port that can be configured by software either for synchronous operations or for time-division multiplexed operations. The TDM serial port is commonly used in multiprocessor applications.

User-Maskable Interrupts

Four external interrupt lines (INT1–INT4) and five internal interrupts, a timer interrupt and four serial port interrupts, are user maskable. When an interrupt service routine (ISR) is executed, the contents of the program counter are saved on an 8-level hardware stack, and the contents of eleven specific CPU registers are automatically saved (shadowed) on a 1-level-deep stack. When

a return from interrupt instruction is executed, the CPU registers‟ contents are restored.

Assembly Language Instructions

`

The „C5x instruction set supports numerically intensive signal processing operations as well as general-purpose applications, such as multiprocessing and high-speed control. The instruction set is a superset of the „C1x and „C2x instruction sets and is source-code compatible with both devices. The „C5x instructions are summarized within the following functional groups.

· Accumulator memory reference instructions.
· Auxiliary registers and data memory page pointer instructions
· Parallel Logi Unit(PLU) instructions
· TREG0, PREG and multiply instructions
· Branch and Call instructions

· I/O and data memory operation instructions
· Control instructions

The number of words that an instruction occupies in program memory is specified in the words column of the table. Several instructions specify the words column because different forms of the instruction occupy a different number of words. For example, the ADD instruction occupies one word when the operand is a short immediate value or two words if the operand is a long immediate value. The number of cycles that an instruction requires to execute is in the cycles column of the table. The tables assume that all instructions are executed from internal program memory (ROM) and internal data memory (RAM).

Mnemonic	Description
1. Accumulator Memory Reference Instructions	
ABS	Absolute value of ACC; zero carry bit
ADCB	Add ACCB and carry bit to ACC
ADD	Add data memory value with left shift, to ACC Add data memory value, with left shift of 16, to ACC Add short immediate to ACC Add long immediate with left shift, to ACC
ADDB	Add ACCB to ACC
ADDC	Add data memory value and carry bit to ACC with sign extension suppressed
ADDS	Add data memory value to ACC with sign extension suppressed
ADDT	Add data memory value with left shift specified by TREG1 to ACC
AND	AND data memory value with ACCL; ,zero ACCH AND long immediate with left shift with ACC AND long immediate with left shift of 16 with ACC
ANDB	AND ACCB with ACC

`

BSAR	Barrel-shift ACC right
2. Accumulator Memory Reference Instructions	
CMPL	1s complement ACC
CRGT	Store ACC in ACCB if ACC>ACCB
CRLT	Store ACC in ACCB if ACC < ACCB
EXAR	Exchange ACCB with ACC
LACB	Load ACC to ACCB
LACC	Load data memory value, with left shift to ACC; Load Long immediate with left shift to ACC Load data memory value, with left shift of 16 to ACC
LACL	Load data memory value to ACCL; zero ACCH Load short immediate to ACCL; zero ACCH
LACT	Load data memory value with left shift specified by TREG1, to ACC
LAMM	Load contents of memory mapped register to ACCL; zero ACCH
NEG	Negate(2"s complement) ACC
NORM	Normalize ACC
OR	OR data memory value with ACCL OR long immediate with left shift with ACC

	OR long immediate with left shift of 16 with ACC
ORB	OR ACCB with ACC
ROL	Rotate ACC left 1 bit
ROLB	Rotate ACCB and ACC left 1 bit
ROR	Rotate ACC right 1 bit
RORB	Rotate ACCB and ACC right 1 bit
SACB	Store ACC in ACCB
SACH	Store ACCH, with left shift, in data memory location.

`

SACL	Store ACCL, with left shift in data memory location.
SAMM	Store ACCL in memory mapped register.
SATH	Barrel-shift ACC right 0 or 16 bits as specified by TREG1
SATL	Barrel-shift ACC right as specified by TREG1
SBB	Subtract ACCB from ACC
SBBB	Subtract ACCB and logical inversion of carry bit from ACC
SFL	Shift ACC left 1 bit.
SFLB	Shift ACCB and ACC left 1 bit
SFR	Shift ACC right 1 bit
SFRB	Shift ACCB and ACC right 1 bit.
SUB	Subtract data memory value, with left shift, from ACC Subtract data memory value with left shift of 16 from ACC Subtract short immediate from ACC Subtract long immediate with left shift from ACC
SUBB	Subtract data memory value and logical inversion of carry bit from ACC with sign extension suppressed.
SUBC	Conditional subtract
SUBS	Subtract data memory value from ACC with sign extension suppressed.
SUBT	Subtract data memory value with left shift specified by TREG1, from ACC
XOR	Exclusive-OR data memory value with ACCL Exclusive-OR long immediate with left shift of 16 with ACC Exclusive-OR long immediate with left shift with ACC
XORB	Exclusive-OR ACCB with ACC
ZALR	Zero ACCL and load ACCH with rounding
ZAP	Zero ACC and PREG
ADRK	Add short immediate to AR
CMPR	Compare AR with ARCR as specified by CM bits
LAR	Load data memory value to ARx, Load short immediate to ARx, Load long immediate to ARx.

`

LDP	Load data memory value to DP bits Load short immediate to DP bits
MAR	Modify AR
SAR	Store ARx in data memory location
SBRK	Subtract short immediate from AR
3. Parallel Logic Unit(PLU) Instructions	
APL	AND data memory value with DBMR and store result in data memory location. AND data memory value with long immediate and store result in data memory location.
CPL	Compare data memory value with DBMR Compare data memory value with long immediate
OPL	OR data memory value with DBMR and store result in data memory location. OR data memory value with long immediate and store result in data memory location.

SPLK	Store long immediate in data memory location
XPL	Exclusive-OR data memory value with DBMR and store result in data memory location. Exclusive-OR data memory value with long immediate and store result in data memory location.
LPH	Load data memory value to PREG high byte
LT	Load data memory value to TREG0
4. TREG0, PREG and Multiply Instructions	
LTA	Load data memory value to TREG0; add PREG, with shift specified by PM bits, to ACC
LTD	Load data memory value to TREG0; add PREG, with shift specified by PM bits, to ACC; and move data
LTP	Load data memory value to TREG0; store PREG, with shift specified by PM bits, in ACC.

`

LTS	Load data memory value to TREG0; subtract PREG, with shift specified by PM bits, from ACC.
MAC	Add PREG, with shift specified by PM bits, to ACC; load data memory value to TREG0; multiply data memory value by program memory value and store result in PREG.
MACD	Add PREG, with shift specified by PM bits, to ACC; load data memory value to TREG0; multiply data memory value by program memory value and store result in PREG; and move data.
MADD	Add PREG, with shift specified by PM bits, to ACC; load data memory value to TREG0; multiply data memory value by value specified in BMAR and store result in PREG; and move data
MADS	Add PREG, with shift specified by PM bits, to ACC; load data memory value to TREG0; multiply data memory value by value specified in BMAR and store result in PREG.
MPY	Multiply data memory value by TREG0 and store result in PREG. Multiply short immediate by TREG0 and store result in PREG. Multiply long immediate by TREG0 and store result in PREG.
MPYA	Add PREG, with shift specified by PM bits, to ACC; multiply data memory value by TREG0 and store result in PREG.
MPYS	Subtract PREG, with shift specified by PM bits, from ACC; multiply data memory value by TREG0 and store result in PREG.
MPYU	Multiply unsigned data memory value by TREG0 and store result in PREG.
PAC	Load PREG, with shift specified by PM bits to ACC
SPAC	Subtract PREG, with shift specified by PM bits, from ACC
SPH	Store PREG high byte, with shift specified by PM bits, in data memory location.
SPL	Store PREG low byte, with shift specified by PM bits, in data memory location
SPM	Set product shift mode(PM) bits
SQRA	Add PREG, with shift specified by PM bits to ACC; load data memory value to TREG0;square value and store result in PREG.
SQRS	Subtract PREG, with shift specified by PM bits, from ACC; load data memory value to TREG0; square value and store result in PREG.

`

ZPR	Zero PREG.
5. Branch and Call Instructions	
B	Branch unconditionally to program memory location
BACC	Branch to program memory location specified by ACCL
BACCD	Delayed branch to program memory location specified by ACCL
BANZ	Branch to program memory location if AR not zero

BANZD	Delayed branch to program memory location if AR not zero
BCND	Branch conditionally to program memory location.
BCNDD	Delayed branch conditionally to program memory location
BD	Delayed branch unconditionally to program memory location.
CALA	Call to subroutine addressed by ACCL
CALAD	Delayed call to subroutine addressed by ACCL
CALL	Call to subroutine unconditionally
CALLD	Delayed call to subroutine unconditionally
CC	Call to subroutine conditionally
CCD	Delayed call to subroutine conditionally
INTR	Software interrupt that branches program control to program memory location
NMI	Non-maskable interrupt and globally disable interrupts(INTM =1)
RET	Return from subroutine
RETC	Return from subroutine conditionally
RETCD	Delayed return from subroutine conditionally.
RETD	Delayed return from subroutine
RETE	Return from interrupt with context switch and globally enable interrupts(INTM=0)

`

RETI	Return from interrupt with context switch;
TRAP	Software interrupt that branches program control to program memory location 22h
XC	Execute next instruction(s) conditionally
6. I/O and Data Memory Operation Instructions	
BLDD	Block move from data to data memory. Block move from data to data memory with destination address long immediate Block move from data to data memory with source address in BMAR. Block move from data to data memory with destination address in BMAR
BLDP	Block move from data to program memory with destination address in BMAR.
BLPD	Block move from program to data memory with source address in BMAR. Block move from program to data memory with source address long immediate
DMOV	Move data in data memory
IN	Input data from I/O port to data memory location
LMMR	Load data memory value to memory- mapped register
OUT	Output data from data memory location to I/O port
SMMR	Store memory-mapped register in data memory location
TBLR	Transfer data from program to data memory with source address in ACCL
TBLW	Transfer data from data to program memory with destination address in ACCL
BIT	Test bit
BITT	Test bit specified by TREG2
CLRC	Clear overflow mode(OVM) bit. Clear sign extension mode(SXM) bit Clear hold mode(HM) bit. Clear test/control (TC) bit Clear carry(C) bit Clear Configuration control(CNF) bit Clear interrupt mode (INTM) bit Clear external flag(XF) pin.

`

IDLE	Idle until non-maskable interrupt or reset

IDLE2	Idle until non-maskable interrupt or reset-low-power mode
LST	Load data memory value to ST0, Load data memory value to ST1.
NOP	No operation
POP	Pop top of stack to data memory location
PSHD	Push data memory value to top of stack
PUSH	Push ACCL to top of stack
RPT	Repeat next instruction specified by data memory value. Repeat next instruction specified by short immediate. Repeat next instruction specified by long immediate
RPTB	Repeat block of instructions specified by BRCR
RPTZ	Clear ACC and PREG; repeat next instruction specified by long immediate
SETC	Set overflow mode(OVM)bit, Set Sign extension mode(SXM) bit, Set hold mode(HM) bit, Set test/control (TC) bit, Set carry(C) bit, Set external flag(XF) pin high, Set configuration control(CNF)bit Set interrupt mode(INTM) bit
SST	Store ST0 in data memory location Store ST1 in data memory location.

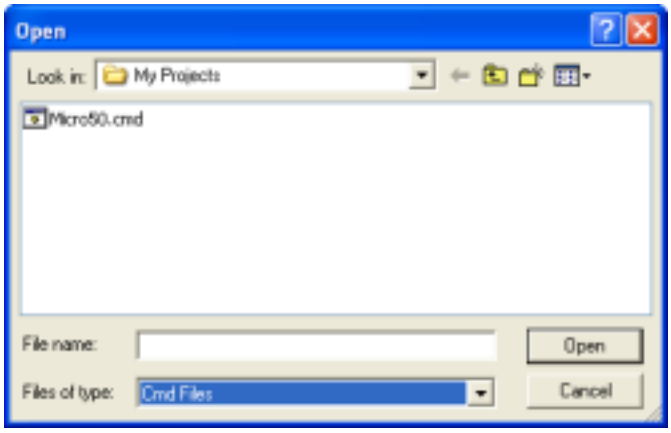
Instructions to execute TMS Processor Assembly Program:

1. Open C50 Debugger.
2. Select Serial□Port Settings.
3. Click Autodetect.
4. If the kit is not detected, it will display as follows:

5. Reset the kit by clicking the reset button and then type **SM** followed by Enter key in the keyboard attached to the kit.

`

6. If the kit is detected, you will get a message as follows: “Vi DSP Hardware found in COM1”
7. Go to Project□New Project.
You will get a dialog box.
Enter a meaningful Project Name and click Save.
8. In the left pane you will see 3 folders in the order
· Assembly
· Cmd Files
· C programs
9. Select File □ New □Assembly File and type the assembly program and select File□Save. Change the **Save as type** to **Assembly files**.



Select the file Micro50.cmd and click Open.
12. Go to Project□Build. If there are errors, project will not be built. Debug and remove the errors. When the project is built successfully you will get a message as follows: “Completed Sucessfully”
13. Go to Serial□Load program

`

Using Browse, select the **.asc** file and click OK. you will get a msg like "57 bytes successfully downloaded".

14. Go to Serial□Communication Window. A window will be displayed with a # symbol

15. If you need to load data in some memory location, say, for example 8000 , type SD 8000

It will display as

Substitute 8000:0A90-

where 0A90 is the data present at 8000.

If you want to store 0001, type it as

Substitute 8000:0A90-**0001**

and press Enter,

It will display

Substitute 8001:0210-

Type in the data to be stored at 8001 location as

Substitute 8001:0210-**0002**

and press Enter.

It will display

Substitute 8002:0140-

If you do not want to store any more information, type "**.**" (dot) at the end as **Substitute 8002:0140-.**

If you had made any mistake while typing, press "." to get back to "#" prompt and then type SD followed by memory address where you want to store data.

16. To execute the program,

type **GO C000**

The window will display "**Executing**…."

17. After a few seconds, Reset the kit by clicking the reset button in the kit and then type SM followed by Enter key in the keyboard attached to the kit.

18. To view the output stored in some memory location say 8002.,

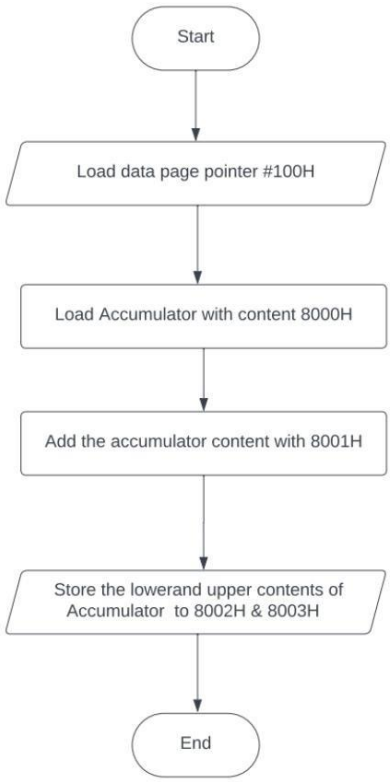
type **SD 8002**

It will display as

Substitute 8002:0003-

`

FLOWCHART: DIRECT ADDRESSING MODE



INPUT:

8000 – 05

8001 – 06

OUTPUT:

8002 -- 0B

8003 -- 00

`

DATE:22/10/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

10a. MAC OPERATION USING DIRECT ADDRESSING MODE

AIM:

To perform MAC operation using the direct addressing mode of TMS320C50 processor.

ALGORITHM:

1. Start the program
2. Load data page pointer with #100H
3. Load accumulator with content of 8000H
4. Add accumulator content with content of 8001H
5. Store lower content of accumulator to 8002H
6. Store higher content of accumulator to 8003H
7. Terminate the program

PROGRAM LISTING:

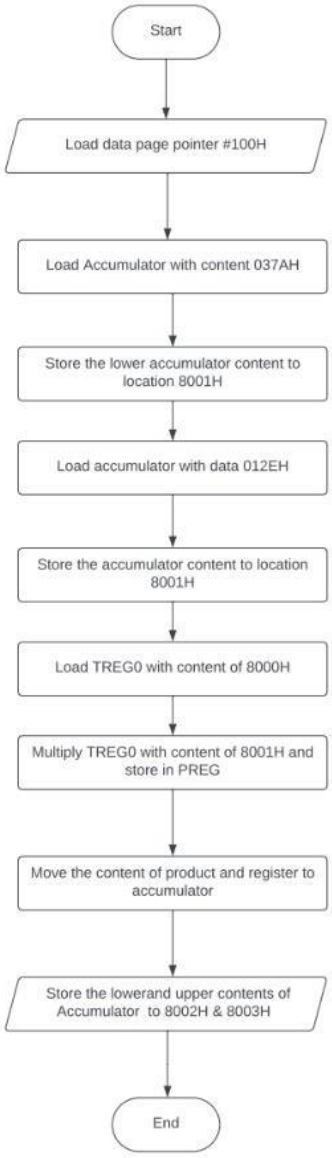
ADDRESS	LABEL	OPCODE	COMMENT
C000		LDP #100H	Load data page pointer with #100H
C001		LACC 0	Load accumulator with content of 8000H
C002		ADD 1	Add accumulator content with content of 8001H
C003		SACL 2	Store lower content of accumulator to 8002H
C004		SACH 3	Store higher content of accumulator to 8003H
C005	R:	B R	Terminate the program

RESULT:

Thus, the MAC operation using the Direct Addressing mode of TMS320C50 processor is performed

`

FLOWCHART: IMMEDIATE ADDRESSING MODE



OUTPUT:

8002 – 19EC
8003 – 0004

`

DATE:22/10/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

10b. MAC OPERATION USING IMMEDIATE ADDRESSING MODE

AIM:

To perform MAC operation using the immediate addressing mode of TMS320C50 processor.

ALGORITHM:

1. Start the program
2. Load Data Page pointer with data #100H
3. Load Accumulator with the data 037AH
4. Store lower accumulator content to location 8000H
5. Load accumulator with data 012EH
6. Store the accumulator content to location 8001H
7. Load Treg0 with content of 8000H [037AH]
8. Multiply Treg0 content with content of 8001H and store result in P reg
9. Move the content of Product and register to accumulator
10. Store lower accumulator content to location 8002H
11. Store higher accumulator content to location 8003H
12. Terminate the program

PROGRAM:

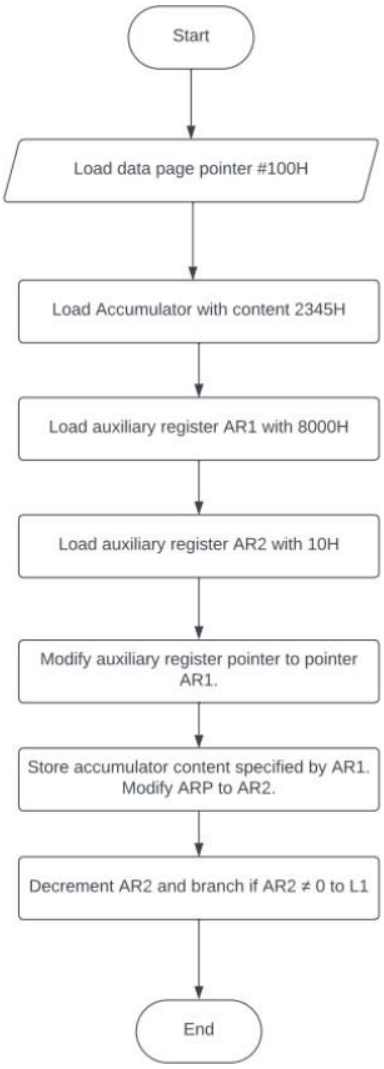
ADDRESS	LABEL	OPCODE	COMMENT
C000		LDP #100H	Load Data Page pointer with data #100H
C001		LACC #037AH,0	Load Accumulator with the data 037AH
C002		SACL 0	Store lower accumulator content to location 8000H
C003		LACC #012EH,0	Load accumulator with data 012EH
C005		LT 0	Load Treg0 with content of 8000H [037AH]
C006		MPY 1	Multiply Treg0 content with content of 8001H and store result in P reg
C007		PAC	Move the content of Product register to accumulator
C008		SACL 2	Store lower accumulator content to location 8002H
C009		SACH 3	Store higher accumulator content to location 8003H
C00A	R:	B R	Terminate the program

RESULT:

Thus, the MAC operation using the Immediate Addressing mode of TMS320C50 processor is performed.

`

FLOWCHART: INDIRECT ADDRESSING MODE



OUTPUT:

8000 – 2345H
8001 – 2345H
8002 – 2345H
8003 – 2345H
8004 – 2345H
8005 – 2345H
8006 – 2345H
8007 – 2345H
8008 – 2345H
8009 – 2345H
800A – 2345H
800B – 2345H
800C – 2345H
800D – 2345H
800E – 2345H
800F – 2345H

`

DATE:22/10/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

10c. MAC OPERATION USING INDIRECT ADDRESSING MODE

AIM:

To perform MAC operation using the indirect addressing mode of TMS320C50 processor.

ALGORITHM:

1. Start the Program
2. Load accumulator with data 2345H
3. Load auxiliary register AR1 with 8000H
4. Load auxiliary register AR2 with 10H
5. Modify auxiliary register pointer to pointer AR1.
6. Store accumulator content specified by AR1. Modify ARP to AR2.
7. Decrement AR2 and branch if AR2 $\neq 0$ to L1
8. Stop the program

PROGRAM:

ADDRESS	LABEL	OPCODE	COMMENT
C000		LACC #2345H	Load accumulator with immediate value #2345
C002		LAR AR1,#8000H	Load auxiliary register AR1 with 8000H
C004		LAR AR2,#10H	Load auxiliary register AR2 with 10H
C006	L1:	MAR *,1	Modify auxiliary register pointer to point AR1
C007		SACL *+,0,2	Store accumulator content specified by AR1.Increment AR1 by 1. Modify ARP to point to AR2
C008		BANZ L1,*-	Decrement AR2 and Branch to L1 if AR2 $\neq 0$
C009	H:	B H	Stop the program

RESULT

Thus, the MAC operation using the Indirect Addressing mode of TMS320C50 processor is performed.

`

`

DATE:29/10/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

EXPT. No. 11 WAVEFORM GENERATION
EXPT. No. 11a. TRIANGULAR WAVEFORM GENERATION

AIM:

To write a program to generate a triangular waveform using TMS320C50 processor

PROGRAM:

```
        TXD .SET 0H
        STS .SET 1H
        DATA .SET 2H
        TEMP .SET 3H
        B3 .SET 0F000H
        B2 .SET 0F00H
        B1 .SET 00F0H
        B0 .SET 000FH
        .MMREGS
        .TEXT
START:
        LDP #100H
REP:
        SPLK #0, DATA
        LAR AR0,#60
REPH:
        OUT DATA, 04H
        LACC DATA
        ADD  #40H
        SACL DATA
        BANZ REPH,*-
        LAR AR0, #60
REPL:
        OUT DATA, 04H
        LACC DATA
        SUB #40H
        SACL DATA
        MAR *, AR0
        BANZ REPL,*-
        B REP
HLT: B HLT
```

RESULT:

Thus a triangular waveform is generated using TMS320C50 processor.

`

`

DATE:29/10/2024

NAME: ARUNKARTHICK R
REG.NO: 412522106013

EXPT. No. 11b. SAWTOOTH WAVEFORM GENERATION

AIM:

To write a program to generate a sawtooth waveform using TMS320C50 processor

PROGRAM:

```
        LDP #100H
        LAR AR0,#50
CIRCULAR:
        LAR AR1,#80
        LAR AR0,#0FFFH
UP:
        OUT 0,04H
        LACC 0
        ADD #85
        SACL 0
        MAR *,AR0
        BANZ UP,*-
        SPLK #0FFFH,0
        LAR AR0,#0FFFH
DOWN:
        OUT 0,04H
        LACC 0
        SUB #85
        SACL 0
        MAR *,AR0
        MAR *,AR1,DOWN,*-
        BANZ CIRCULAR,*-
H: B H
```

RESULT:

Thus a sawtooth waveform is generated using TMS320C50 processor.

`

`

DATE:29/10/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

EXPT. No. 11c. SQUARE WAVEFORM GENERATION

AIM:

To write a program to generate a square waveform using TMS320C50 processor

PROGRAM:

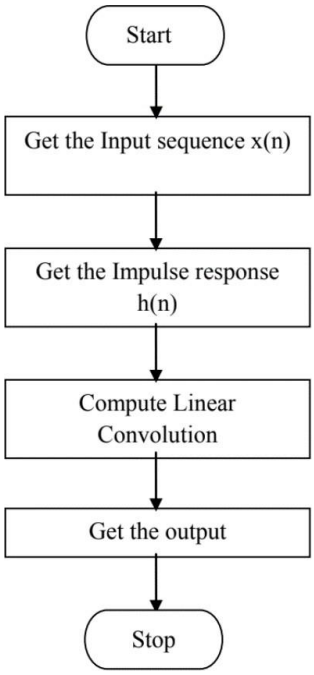
```
START:
LDP #140H
LACC #0FFFH
SACL  0H
OUT 0,04H
NOP
LAR AR3, #0000
DELAY1:
MAR *, AR3
RPT #07FFFH
NOP
BANZ DELAY1, *-
LACC #0H
SACL 0H
OUT 0,04H
NOP
LAR AR3, #0000
DELAY2:
MAR *, AR3
RPT #07FFFH
NOP
BANZ DELAY2, *-
B START
```

RESULT:

Thus, a square waveform is generated using TMS320C50 processor.

`

FLOWCHART: LINEAR CONVOLUTION



`

DATE:29/10/2024

NAME: ARUNKARTHICK R
REG.NO:412522106013

EXPT. No. 12. LINEAR CONVOLUTION

AIM:

To write a processor program to perform linear convolution of two sequences using TMS320C50 processor.

ALGORITHM:

1. Start the program
2. Enter the input sequence x (n).
3. Enter the impulse function h (n).
4. Perform linear convolution of the two sequences.
5. Get the output.
6. Stop the execution

`

INPUT:

x(n): {1,1,2,1,2}

h(n): {0,1,1,3,2}

Data Memory Address	Data
8100	1
8101	1
8102	2
8103	1
8104	2
8200	0
8201	1
8202	1
8203	3
8204	2

OUTPUT:

y(n) = {0,1,2,6,8,9,9,8,4}

Data Memory Address	Data
8300	0
8301	1
8302	2
8303	6
8304	8
8305	9
8306	9
8307	8
8308	4

`

PROGRAM:

ADDRESS	LABEL	OPCODE	COMMENT
C000		LDP #02H	Load data pointer with data 02H
C001		LAR AR1,#8100H	Load AR1 with 8100H
C002		LAR AR0,#8200H	Load AR0 with 8200H
C003		LAR AR3,#8300H	Load AR3 with 8300H
C004		LAR AR4,#0007H	Load AR4 with 0007H
C005		LAR AR0,#8203H	Load AR0 with 08203H
C006		LACC #C100H	Load long immediate 0C100 with left shift to accumulator
C007		MAR *,AR0	Modify the ARP register to point to AR0
C008		RPT #3	Repeat next instruction 4 times
C009		TBLW*-	Transfer data from 8203H to C100H
C00A		LAR AR6,#8104	Load AR6 with 8104H
C00B		MAR *,AR6	Modify ARP register 8203 with AR6
C00C		LACC #0H	Load long immediate 0H with left shift to accumulator
C00D		RPT #3H	Repeat next instruction 4 times
C00E		SACL*+	Store acc lower content to location 8104
C00F	LOP:	MAR *,AR1	Modify ARP register 8203 with AR1
C010		LACC *+	Load accumulator with data in 8203
C011		SACL 50H	Store lower acc content at 050H
C012		LAR AR2,#0153H	Load AR2 with immediate data 0153H
C013		MAR *,AR2	Modify ARP register 8203 with AR2
C014		ZAP	Zero accumulator and PREG
C015		RPT #03H	Repeat next instruction 4 times
C016		MACD C100H	Add PREG with 0c100H to acc, load data mem. value to TREG0, multiply data mem. value by program mem. value and store result in PREG

`

`

\

C017		APAC	Load PREG with
C018		MAR*,AR3	Modify ARP register 8203 with AR3
C019		SACL*+	Store acc lower content to location 8104
C01A		MAR*,AR0	Modify ARP register 8203 with AR0
C01B		BANZ LOP*-	Branch to C00F if AR not zero
C01C	H:	B H	Terminate program

RESULT:

Thus, the program to perform linear convolution of two sequences is executed using TMS320C50 processor.