# Scilab Manual for
# Digital Signal Processing
# by Prof K Jeevan Reddy
# Electronics Engineering
# Sreenidhi Institute Of Science And Technology[1]

Solutions provided by
Prof K Jeevan Reddy
Electronics Engineering
Sreenidhi Institute Of Science And Technology

May 19, 2026

[1]Funded by a grant from the National Mission on Education through ICT, http://spoken-tutorial.org/NMEICT-Intro. This Scilab Manual and Scilab codes written in it can be downloaded from the "Migrated Labs" section at the website http://scilab.in

# Contents

# List of Experiments

# Experiment: 1

# Analog To Digital Filter Conversion using impulse invariance method

**Scilab code Solution 1.1** Analog to Digital Filter Conversion

```
//Filter Conversion using Impulse Invariance Method
//OS:Windows 10
//Scilab 5.5.2

clear all;
clc;
close;

s=%s;
T =1;
HS =(2)/(s^2+3*s+2) ;
elts = pfss(HS);
disp(elts,'Factorized HS=');

//The poles associated are -2 and -1
p1 = -2;
p2 = -1;
```

```
z=%z;
HZ =(2/(1-%e^(p2*T)*z^(-1)))-(2/(1-%e^(p1*T)*z^(-1)))
disp(HZ,'HZ=');
```

# Experiment: 2

# Analog To Digital Filter Conversion using bilinear transformation method

**Scilab code Solution 2.2** Analog to Digital Filter Conversion

```
//Filter Conversion using Bilinear Transformation
   Method
//OS:Windows 10
//Scilab 5.5.2

clear all;
clc;
close;

ap = input('Enter value of ap in dB')
as = input('Enter value of as in dB')
fp = input('Enter value of fp in Hz')
fs = input('Enter value of fs in Hz')
f = input('Enter value of f')

T=1/f;
wp =2*%pi*fp;
```

```
ws =2*%pi*fs;
op =2/T*tan(wp*T/2);
os =2/T*tan(ws*T/2);
N= log(sqrt((10^(0.1*as)-1)/(10^(0.1*ap)-1)))/log(op
   /os);
disp(ceil(N),' Order of the filter , N =');
s=%s;
HS =1/(s+1) // Transfer Function for N=1
oc=op //rad/sec
HS1 = horner(HS,oc/s);
disp(HS1,'Normalized Transfer Function , H(s) =');
z=%z;
HZ= horner(HS,(2/T)*(z-1)/(z+1));
disp(HZ,'H(z)=');

//Example Input
//Enter value of ap in dB 3
//Enter value of as in dB 10
//Enter value of fp in Hz 1000
//Enter value of fs in Hz 350
//Enter value of f 5000
//Order of the filter , N =    1.
// Normalized Transfer Function , H(s) =
//          s
//    ----------------
//
//    7265.4253 + s
// H(z)=
//        1 + z
//    -----------------
//
//    -9999 + 10001z
```

# Experiment: 3

# Prototype Analog Low Pass Filter to High Pass Filter Conversion

**Scilab code Solution 3.1** Analog LPF to HPF Conversion

```
// To Convert Analog LPF to High Pass
//Using Analog Filter Transformations
//For the cutoff frequency Wc = 500 Hz
//OS:Windows 10
//Scilab 5.5.2

clear all;
clc;
close;

omegap =  500;
omegas =  1000;
delta1_in_dB = -3;
delta2_in_dB = -40;
delta1 = 10^(delta1_in_dB/20)
delta2 = 10^(delta2_in_dB/20)

```

```
//Calculation of Filter Order
N = log10((1/(delta2^2))-1)/(2*log10(omegas/omegap))
N = ceil(N)
omegac = omegap;

//Poles and Gain Calculation
[pols,gain]=zpbutt(N,omegac);
N =1;

omega_LPF = omegap;  //Analog LPF Cutoff frequency
omega_HPF = omega_LPF;  //Analog HPF Cutoff frequency
omega2 = 600;     //Upper Cutoff frequency
omega1 = 300;      //Lower Cutoff Frequency
omega0 = (omega2*omega1);
BW =  omega2 -  omega1;  //Bandwidth
disp('Analog LPF Transfer function')
[hs,pols,zers,gain] = analpf(N,'butt',[0,0],omega_LPF)
hs_LPF = hs;
hs_LPF(2) = hs_LPF(2)/500;
hs_LPF(3)= hs_LPF(3)/500;
s =poly(0,'s');
disp('Analog HPF Transfer function')
h_HPF = horner(hs_LPF,omega_LPF*omega_HPF/s)
num = (s^2)+omega0
den = BW*s
h_BPF = horner(hs_LPF,omega_LPF*(num/den))

//Plotting Low Pass Filter Frequency Response
figure
fr=0:.1:1000;
hf=freq(hs_LPF(2),hs_LPF(3),%i*fr);
hm=abs(hf);
plot(fr,hm)
xgrid(1)
xtitle('Magnitude Response of LPF Filter Cutoff frequency = 500Hz','Analog Frequency--->','
```

```
Magnitude');

//Plotting High Pass Filter Frequency Response
figure
fr=0:.1:1000;
hf_HPF=freq(h_HPF(2),h_HPF(3),%i*fr);
hm_HPF=abs(hf_HPF);
plot(fr,hm_HPF)
xgrid(1)
xtitle('Magnitude Response of HPF Filter Cutoff
    frequency = 500Hz','Analog Frequency--->','
    Magnitude');
```

# Experiment: 4

# Prototype Analog Low Pass Filter to Band Pass Filter Conversion

**Scilab code Solution 4.1** Analog LPF to BPF Conversion

```
// To Convert Analog LPF to Band Pass IIR
   Butterworth Filter
//Using Analog Filter Transformations
//For the cutoff frequency Wc = 500 Hz
//OS:Windows 10
//Scilab 5.5.2

clear all;
clc;
close;

omegap =  500;
omegas =  1000;
delta1_in_dB = -3;
delta2_in_dB = -40;
delta1 = 10^(delta1_in_dB/20)
delta2 = 10^(delta2_in_dB/20)
```

```

//Calculation of Filter Order
N = log10((1/(delta2^2))-1)/(2*log10(omegas/omegap))
N = ceil(N)
omegac = omegap;

//Poles and Gain Calculation
[pols,gain]=zpbutt(N,omegac);
N =1;

omega_LPF = omegap;  //Analog LPF Cutoff frequency
omega_HPF = omega_LPF;  //Analog HPF Cutoff frequency
omega2 = 600;     //Upper Cutoff frequency
omega1 = 300;      //Lower Cutoff Frequency
omega0 = (omega2*omega1);
BW =  omega2 -  omega1;  //Bandwidth
disp('Analog LPF Transfer function')
[hs,pols,zers,gain] = analpf(N,'butt',[0,0],omega_LPF)
hs_LPF = hs;
hs_LPF(2) = hs_LPF(2)/500;
hs_LPF(3)= hs_LPF(3)/500;
s =poly(0,'s');
h_HPF = horner(hs_LPF,omega_LPF*omega_HPF/s)
disp('Analog BPF Transfer function')
num = (s^2)+omega0
den = BW*s
h_BPF = horner(hs_LPF,omega_LPF*(num/den))

//Plotting Low Pass Filter Frequency Response
figure
fr=0:.1:1000;
hf=freq(hs_LPF(2),hs_LPF(3),%i*fr);
hm=abs(hf);
plot(fr,hm)
xgrid(1)
xtitle('Magnitude Response of LPF Filter Cutoff
```

```
    frequency = 500Hz','Analog Frequency——>','
    Magnitude');

//Plotting Band Pass Filter Frequency Response
figure
fr=0:.1:1000;
hf_BPF=freq(h_BPF(2),h_BPF(3),%i*fr);
hm_BPF=abs(hf_BPF);
plot(fr,hm_BPF)
xgrid(1)
xtitle('Magnitude Response of BPF Filter Upper
    Cutoff frequency = 600Hz & Lower Cutoff frequency
     = 300Hz','Analog  Frequency——>','Magnitude');
```

# Experiment: 5

# Prototype Analog Low Pass Filter to Band Rejection/Stop Filter Conversion

**Scilab code Solution 5.1** Analog LPF to BSF Conversion

```
//Analog Low Pass Filter to Band Stop Filter Conversion
//OS:Windows 10
//Scilab 6.0.0

clear ;
clc ;
close ;

fc1 = input("Enter Analog lower cutoff freq. in Hz=")
fc2 = input("Enter Analog higher cutoff freq. in Hz=")
fs = input("Enter Analog sampling freq. in Hz=")
M = input("Enter order of filter =")
w1 = (2*%pi)*(fc1/fs);
w2 = (2*%pi)*(fc2/fs);
```

```
disp(w1,'Digital lower cutoff frequency in radians.cycles/samples');
disp(w2,'Digital higher cutoff frequency in radians.cycles/samples');
wc1 = w1/%pi;
wc2 = w2/%pi;
disp(wc1,'Normalized digital lower cutoff frequency in cycles/samples');
disp(wc2,'Normalized digital higher cutoff frequency in cycles/samples');
[wft1,wfm1,fr1]=wfir('lp',M+1,[wc1/2,0],'re',[0,0]);
disp(wft1,'Impulse Response of LPF FIR Filter:h[n]=');

//Plotting the Magnitude Response of LPF FIR Filter
figure
subplot(2,1,1)
plot(2*fr1,wfm1)
xlabel('Normalized Digital Frequency w--->')
ylabel('Magnitude |H(w)|=')
title('Magnitude Response of FIR LPF')
xgrid(1)
subplot(2,1,2)
plot(fr1*fs,wfm1)
xlabel('Analog Frequency in Hz f --->')
ylabel('Magnitude |H(w)|=')
title('Magnitude Response of FIR LPF')
xgrid(1)

[wft,wfm,fr]=wfir('sb',M+1,[wc1/2,wc2/2],'re',[0,0]);
disp(wft,'Impulse Response of BSF FIR Filter:h[n]=');

//Plotting the Magnitude Response of HPF FIR Filter
figure
subplot(2,1,1)
plot(2*fr,wfm)
```

```
xlabel('Normalized Digital Frequency w--->')
ylabel('Magnitude |H(w)|=')
title('Magnitude Response of FIR BSF')
xgrid(1)
subplot(2,1,2)
plot(fr*fs,wfm)
xlabel('Analog Frequency in Hz f --->')
ylabel('Magnitude |H(w)|=')
title('Magnitude Response of FIR BSF')
xgrid(1)


//Example Input
//Enter Analog lower cutoff freq. in Hz=250
//Enter Analog higher cutoff freq. in Hz=600
//Enter Analog sampling freq. in Hz=2000
//Enter order of filter =4
//Digital lower cutoff frequency in radians
// .cycles/samples
//    0.7853982

// Digital higher cutoff frequency in radian
// s.cycles/samples
//    1.8849556

// Normalized digital lower cutoff frequency
//  in cycles/samples
//    0.25

// Normalized digital higher cutoff frequenc
// y in cycles/samples
//    0.6
// Impulse Response of LPF FIR Filter:h[n]=
//          column 1 to 3
//    0.1591549    0.2250791    0.25
//          column 4 to 5
//    0.2250791    0.1591549

```

```
// Impulse Response of BSF FIR Filter:h[n]=
//          column 1 to 3
//    0.2527039   -0.0776516    0.65
//          column 4 to 5
//  -0.0776516    0.2527039
```

# Experiment: 6

# Comparision of FIR filter design using windowing method

**Scilab code Solution 6.1** Rectangular Window

```
//Plot Magnitude Response of L.P.F.
//N=7 , fc =1000Hz , F=5000Hz
//FIR filter design using Windowing Method-
   Rectangular Window
//OS:Windows 10
//Scilab 6.0.0

clear ;
clc ;
close ;

N =7;
U =4;
h_Rect =window('re',N);
for n= -3+U :1:3+ U
    if n ==4
        hd(n) =0.4;
    else
        hd(n)=( sin (2* %pi *(n-U) /5) )/( %pi *(n-U
```

```
            ));
    end
    h(n)=hd(n)* h_Rect (n);
end
[hzm,fr]=frmag(h,256) ;
hzm_dB=20*log10(hzm)./max(hzm);
figure
xgrid(2);
plot(2*fr,hzm_dB)
a=gca();
xlabel('Frequency w*pi ');
ylabel('Magnitude in dB ');
title('Frequency Response of FIR LPF with N=7');
xgrid(2)
disp(h,"Filter Co efficients , h(n)=");
```

**Scilab code Solution 6.2** Kaiser Window

```
//Plot Magnitude Response of L.P.F .
//FIR filter design using Windowing Method- Kaiser
   Window
//OS:Windows 10
//Scilab 6.0.0

clear ;
clc ;
close ;

wsf = input("Enter the value of wsf in rad/sec");
ws = input("Enter the value of ws in rad/sec");
wp = input("Enter the value of wp in rad/sec");
as = input("Enter the value of as in dB");
ap = input("Enter the value of ap in dB");

B=ws -wp;
```

```
wc =0.5*(ws+wp);
wc1 =wc*2*%pi/ wsf ;
delta1 =10^(-0.05* as);
delta2 =(10^(0.05* as) -1)/(10^(0.05* as)+1);
delta = min(delta1,delta2);
alphas = -20*log10(delta);
alpha =0.5842*(alphas -21)^0.4+0.07886*(alphas -21)
D=(alphas -7.95)/14.36;
N1=wsf*D/B +1;
N= ceil(N1);
U= ceil(N/2) ;
win_l = window('kr',N,alpha);
for n=- floor(N/2)+U :1: floor(N/2) +U
    if n== ceil(N /2) ;
        hd(n) =0.5;
    else
        hd(n)=(sin(%pi*(n-U)/2))/(%pi*(n-U));
    end
    h(n)=hd(n)* win_l(n);
end
[hzm,fr]=frmag(h,256) ;
hzm_dB = 20*log10(hzm)./max(hzm);
figure
plot(2*fr,hzm_dB )
a= gca();
xlabel('Frequency w*pi ');
ylabel('Magnitude in dB ');
title('Frequency Response of LPF');
xgrid(2);
disp(h,"Filter Co efficients , h(n)=");

//Example Input
//Enter the value of wsf in rad/sec 100
//Enter the value of ws in rad/sec 30
//Enter the value of wp in rad/sec 20
//Enter the value of as in dB 44
//Enter the value of ap in dB 0.1
//Filter Co efficients , h(n)=
```

```
//     0.002441    -3.172D-18    -0.0068491     6.235D-18
    0.0145007    -9.826D-18    -0.027237     1.347D-17
    0.0494369    -1.661D-17    -0.0970495     1.874D-17
    0.3152014     0.5      0.3152014      1.874D-17
    -0.0970495    -1.661D-17     0.0494369     1.347D-17
    -0.027237    -9.826D-18     0.0145007     6.235D-18
    -0.0068491    -3.172D-18     0.002441
```

**Scilab code Solution 6.3** Hamming Window

```
//Plot Magnitude Response of ideal H.P.F.
//wc1=0.25*pi  , N=11
//FIR filter design using Windowing Method-Hamming
    Window
//OS:Windows 10
//Scilab 6.0.0

clear;
clc;
close;

N =11;
U =6;
h_hamm=window('hm',N);
for n= -5+U :1:5+ U
    if n ==6
        hd(n) =0.75;
    else
        hd(n)=(sin(%pi*(n-U))-sin(%pi*(n-U)/4))/(%pi
           *(n-U));
    end
    h(n)= h_hamm(n)*hd(n);
end
[hzm,fr]=frmag(h,256);
hzm_dB=20*log10(hzm)./max(hzm);
```

```
figure
plot(2*fr,hzm_dB)
a=gca();
xlabel('Frequency w*pi');
ylabel('Magnitude in dB');
title('Frequency Response of FIR HPF with N=11 using
    Hamming Window');
xgrid(2);
```

**Scilab code Solution 6.4** Hanning Window

```
//Plot Magnitude Response of ideal H.P.F.
//N=11 ,wc1=0.25* pi
//FIR filter design using Windowing Method—Hanning
    Window
//OS:Windows 10
//Scilab 6.0.0

clear;
clc;
close;

N =11;
U =6;
h_hann = window('hn',N);
for n= -5+U:1:5+U
    if n == 6
        hd(n) = 0.75;
    else
        hd(n)=(sin(%pi*(n-U))-sin(%pi*(n-U)/4))/(%pi
            *(n-U));
    end
    h(n)= h_hann(n)*hd(n);
end
[hzm,fr]= frmag(h,256) ;
```

```
hzm_dB =20*log10(hzm)./max(hzm);
figure
plot(2*fr,hzm_dB)
a=gca();
xlabel('Frequency w*pi ');
ylabel('Magnitude in dB ');
title('Frequency Response of FIR HPF with N=11 using
    Hanning Window ');
xgrid(2);
```

# Experiment: 7

# Detection of DTMF signal

**Scilab code Solution 7.1** DTMF Decoding

```
//DTMF Signal Detection
//OS:Windows 10
//Scilab 5.5.2

clear all;
clc;
close;

row_f1=[800 870 950 990];       // Row Frequency
colum_f1=[1340 1440 1540];      // Column Frequency
fs=8000;                        // Sampling Frequency
N=1:800;                        // Total No. of Samples for each Digit
mobile=[5 6 7 8 9 0 1 2 3 4];
total_signal=[];

figure;

for i=1:length(mobile)
    select mobile(i)
    case 1
```

```
            row_f=1;
            colum_f=1;
        case 2
            row_f=1;
            colum_f=2;
        case 3
            row_f=1;
            colum_f=3;
        case 4
            row_f=2;
            colum_f=1;
        case 5
            row_f=2;
            colum_f=2;
        case 6
            row_f=2;
            colum_f=3;
         case 7
            row_f=3;
            colum_f=1;
        case 8
            row_f=3;
            colum_f=2;
        case 9
            row_f=3;
            colum_f=3;
         case 0
            row_f=4;
            colum_f=2;
        else
            row_f=4;
            colum_f=1;
        end
         y=sin(2*3.14*(row_f1(row_f)/fs)*N)+sin(2*3.14*(colum_f1(colum_f)/fs)*N);  //Time Domain Signal Generation for each Digit
         total_signal=[total_signal y zeros(1,8800)];
         temp(:,:,i)=y(:,:);
```

```
    end
    plot(total_signal);
    title('DTMF Signal','color','blue');
    xlabel("Samples","color", "blue");
    ylabel("Amplitude","color", "blue");
    sound(total_signal,fs);

row_f=[];
col_f=[];

for i=1:10
    n=length(temp(:,:,i));
    p=abs(fft(temp(:,:,i))); // FFT of Signal of respective Digit
    f=(0:n-1)*fs/n;          // Total Frequency Range
    //plot(f,p);
    row=p(2:100);            // Row Frequency separation
    col=p(101:200);          // Column Frequency separation
    [r1 c1]=find(row==max(row));   // Finding the location of peak for Row Frequency
    [r2 c2]=find(col==max(col));   // Finding the location of peak for Column Frequency
    row_f=[row_f 10*c1];             // Array containing peak of Row Freqiency
    col_f=[col_f (10*(c2+100))-10]; // Array containing peak of Column Freqiency
end

mobile_find=[];  // Blank Array to Store Mobile Number
for i=1:10     // Loop forFinding the Number form the Row and Column Frequency
    if(row_f(i)==800 & col_f(i)==1340)
        n0=1;
        elseif(row_f(i)==800 & col_f(i)==1440)
```

```
    n0=2;
    elseif(row_f(i)==800 & col_f(i)==1540)
    n0=3;
    elseif(row_f(i)==870 & col_f(i)==1340)
    n0=4;
    elseif(row_f(i)==870 & col_f(i)==1440)
    n0=5;
    elseif(row_f(i)==870 & col_f(i)==1540)
    n0=6;
    elseif(row_f(i)==950 & col_f(i)==1340)
    n0=7;
    elseif(row_f(i)==950 & col_f(i)==1440)
    n0=8;
    elseif(row_f(i)==950 & col_f(i)==1540)
    n0=9;
    elseif(row_f(i)==990 & col_f(i)==1440)
    n0=0;
end
mobile_find=[mobile_find n0];  // Array containing Decoded Digit of Mobile Number.
end

disp(mobile_find,"Decoded Mobile Number :");
```

# Experiment: 8

# Implementation of Decimation

**Scilab code Solution 8.1** Decimation

```
//Implementation of Decimation
//OS:Windows 10
//Scilab 5.5.2

clear all;
clc;
close;

t=0:0.00025:1;
x=sin(2*%pi*30*t)+sin(2*%pi*60*t);    //original signal
y = x(1:4:length(x)); //downsampled by a factor of 4
subplot(2,1,1)
plot2d3(0:120,x(1:121),-1);
xgrid(1);
xtitle('Original singal')
subplot(2,1,2)
plot2d3(0:30,y(1:31),-1);
xgrid(1);
xtitle('Downsampled Signal');
```

# Experiment: 9

# Implementation of Interpolation

**Scilab code Solution 9.1** Interpolation

```
//Implementation of Interpolation
//OS:Windows 10
//Scilab 5.5.2

clear all;
clc;
close;

t=0:0.00025:1;
x=sin(2*%pi*30*t)+sin(2*%pi*60*t);    //original
    signal
upsampling_x = zeros(1,2*length(x)); //upsampled by
    a factor of 2
upsampling_x(1:2:2*length(x)) = x;
subplot(2,1,1)
plot2d3(0:120,x(1:121),-1);
xgrid(1);
xtitle('Original singal')
subplot(2,1,2)
```

```
plot2d3(0:250,upsampling_x(1:251),-1);
xgrid(1);
xtitle('Upsampled Signal');
```

# Experiment: 10

# FIR Filter using Frequency Sampling Method

**Scilab code Solution 10.1** Frequency Sampling Method

```
//FIR LPF Filter using Frequency Sampling Method
//OS:Windows 10
//Scilab 5.5.2

clc;
clear;
close;

N = input("Enter the value of N:");
U = input("Enter the value of U:");
for n =0+ U :1: N -1+ U
    h(n)=(1+cos(2*%pi*(7-n)/N))/N;
end
[hz,f]=frmag(h,256) ;
hz_dB=20*log10(hz)./max(hz);
figure ;
plot(2*f,hz_dB);
a=gca();
xlabel('Frequency w      pi  ') ;
```

```
ylabel('Magnitude in dB ') ;
title('Frequency Response of FIR LPF') ;
xgrid(2)

//Example input
//Enter the value of N:15
//Enter the value of U:1 , Wc=pi/4
```