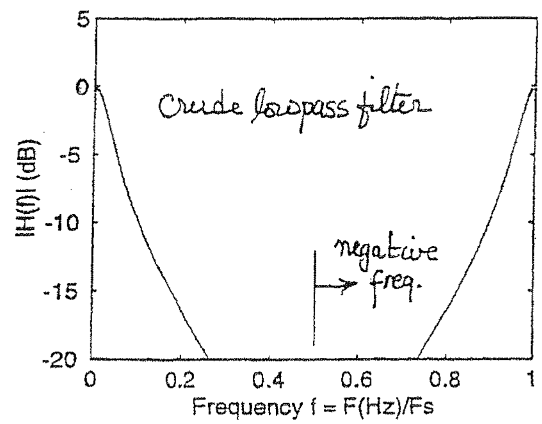
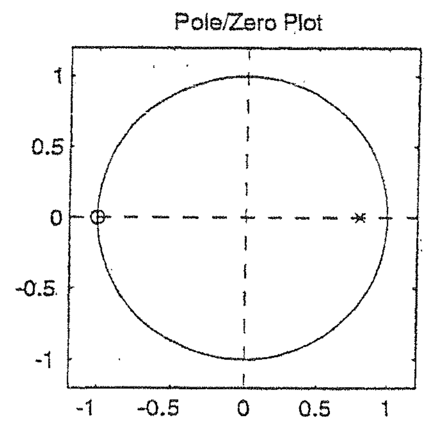
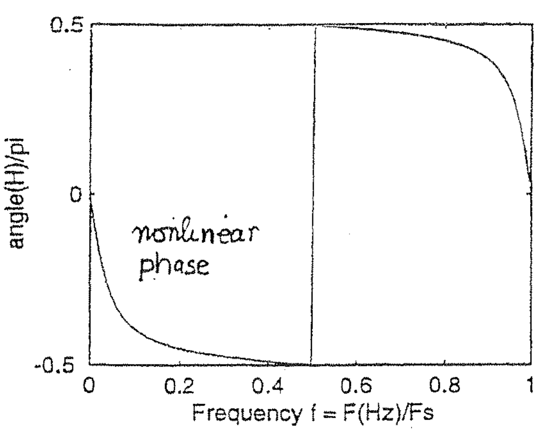
Digital IIR Filters: Analysis Examples

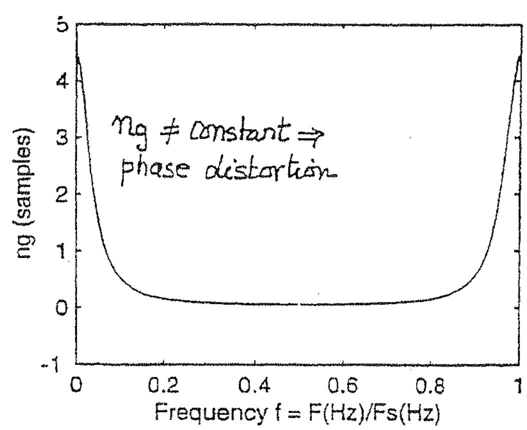
```
% Script File: IIR_Anal_Ex1.m
%
% Crude Lowpass filter
% ------------------------
%          b0 (1 + z^-1)
% H(z) = ---------------
%          (1 - c1 z^-1)
% c1 = 0.8
% b0= ((1-c1)/2 is a gain factor so that H(z=1, or w=0) = 1
%
c1=0.8;
b0=(1-c1)/2;
b=b0*[1    1];
a=[1  -c1];
% plot results
L=[ -1.2 -20 -0.5  -1];
U=[ 1.2   5   0.5  5];
[c d]= sys(b,a,L,U);
```



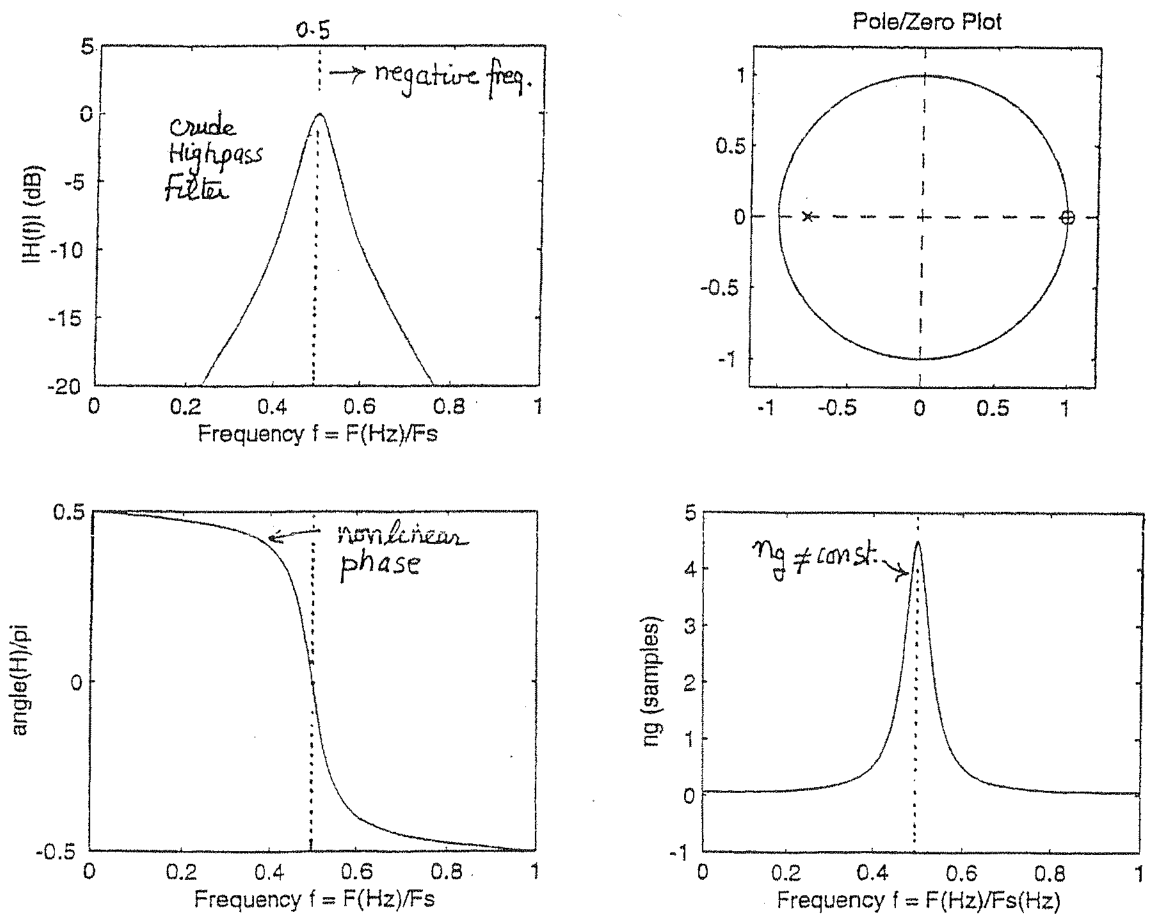


IIR ⇒ at least one pole not at z=0

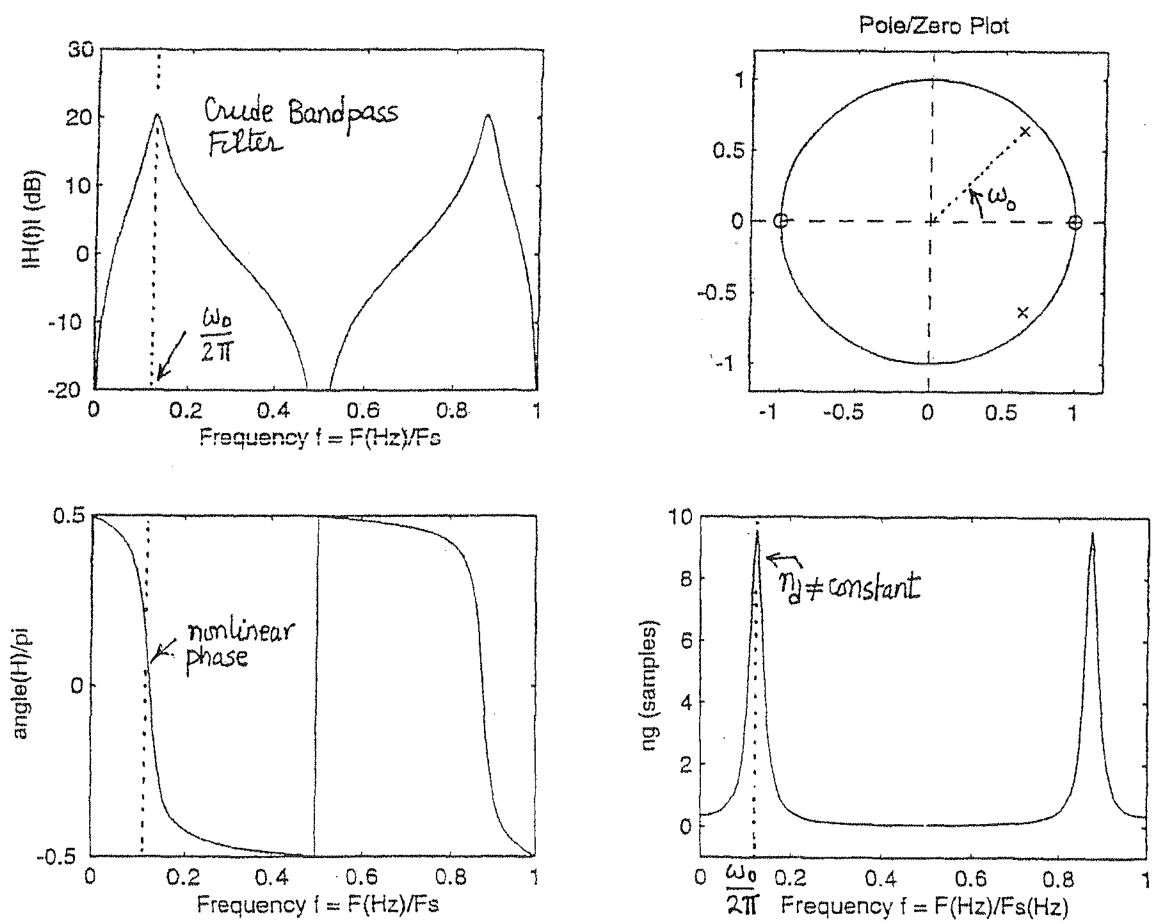




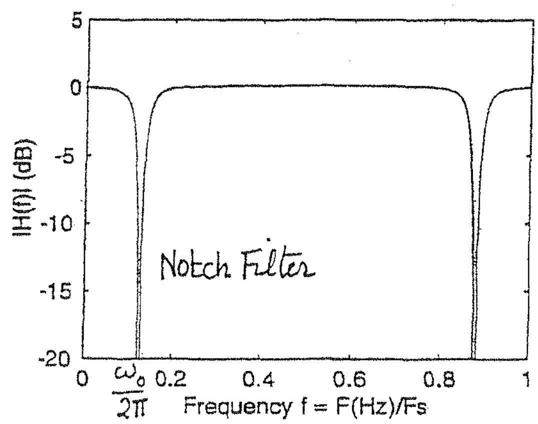
```
% Script File: IIR_Anal_Ex2.m
%
% Crude Highpass filter
% ----------------------
%
%            b0  (1 - z^-1)
% H(z) = ---------------
%            (1 - c1 z^-1)
%
%  c1 = -0.8
% b0= ((1+c1)/2 is a gain factor so that H(z=-1, or w=pi) = 1
%

c1=-0.8;
b0=(1+c1)/2;
b=b0*[1    -1];
a=[1  -c1];
% plot results
L=[ -1.2 -20 -0.5  -1];
U=[ 1.2   5   0.5  5];
[c d]= sys(b,a,L,U);
```

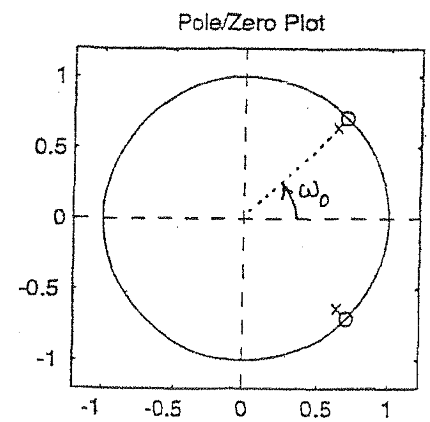


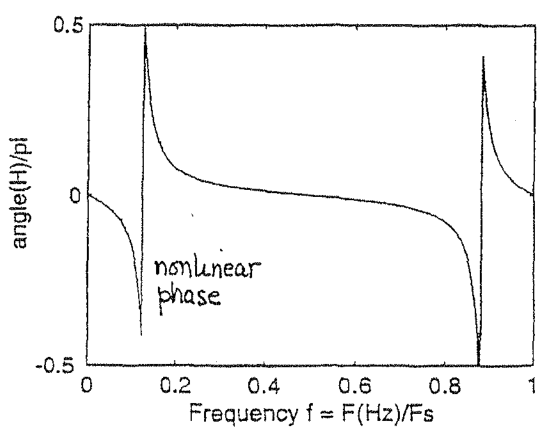
```
% Script File: IIR_Anal_Ex3.m
%
% Crude Bandpass filter
% ---------------------
%
%            b0  (1+z^-1) (1-z^-1)/
% H(z) = ----------------------------
%            (1-c1 z^-1) (1-c2 z^-1))
%
% c1,2 = 0.9 exp(±j*pi/4)
% b0 = 1 is assumed below
%
j=sqrt(-1);
b0=1;                                   % unity gain factor assumed
b=b0* conv([1 1],[1 -1]);               % conv performs poly multiplication
%
c1=0.9*exp(j*pi/4);
c2=conj(c1);
a=conv([1 -c1],[1 -c2]);                % conv performs poly multiplication
%
% plot results
L=[ -1.2 -20 -0.5  -1];
U=[ 1.2   30  0.5  10];
[c d]= sys(b,a,L,U);
```

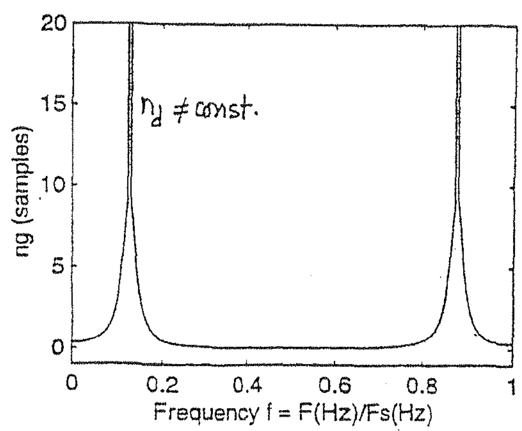


```
% Script File: IIR_Anal_Ex4.m
%
% Crude Notch (or bandstop) filter
% ----------------------------------------
%
%             b0  (1-d1 z^-1) (1-d2 z^-1)
% H(z) = ---------------------------------
%             (1-c1 z^-1) (1-c2 z-1);
%
% d1 = exp(j*pi/4); c1 =  0.9 exp(j*pi/4)
% b0 = (1-c1)(1-c2)/(1-d1)(1-d2)

j=sqrt(-1);
c1=0.9*exp(j*pi/4); c2=conj(c1);
d1=   exp(j*pi/4); d2=conj(d1);
b0=(1-c1)*(1-c2)/((1-d1)*(1-d2));
b=b0*conv([1 -d1],[1 -d2]);
a=conv([1 -c1],[1 -c2]);

% plot results
L=[ -1.2 -20 -0.5  -1];
U=[ 1.2   5 0.5  20];
[c d]= sys(b,a,L,U);
```

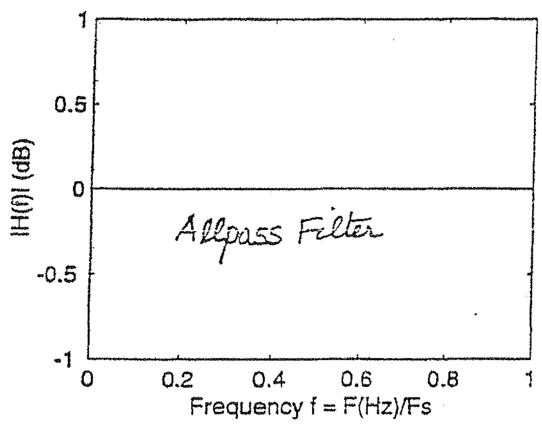


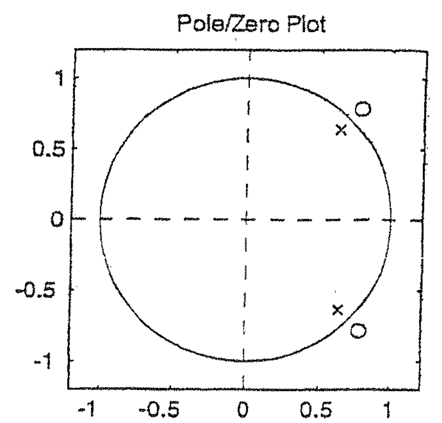


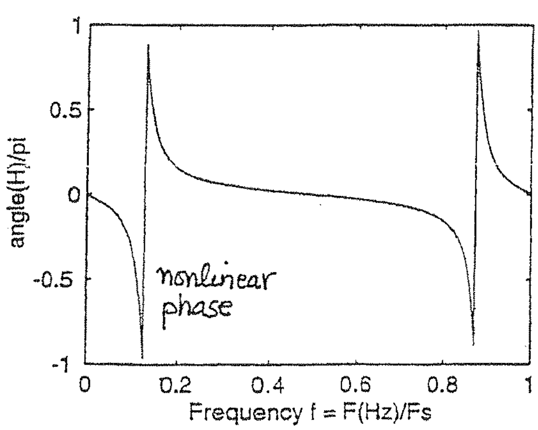


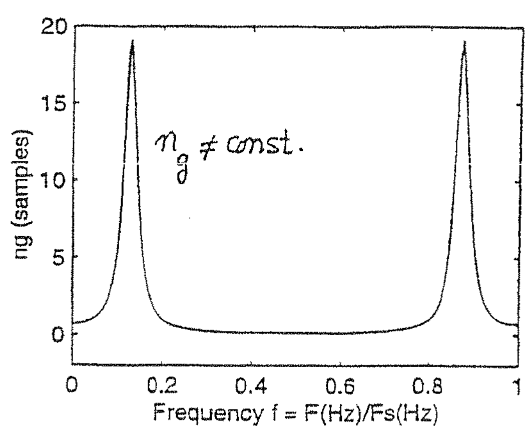


```
% Script File IIR_Anal_Ex5.m
%
% An Allpass Filter (from O&S)
% ----------------------------------------
%
%            (z^-1 - c1) (z^-1 - c2)       a2 + a1 z^-1 +    z^-2
% H(z) = ------------------------------ = ------------------------------
%            (1-c1 z^-1) (1-c2 z^-1)       1 + a1 z^-1 + a2 z^-2
%
j=sqrt(-1);
c=[ 0.9*exp(j*pi/4)   0.9*exp(-j*pi/4)];
a=real(poly(c));
%
b=a(length(a):-1:1);                          % "b" is "a" written backwards
L=[-1.2 -1  -1  -2];
U=[1.2  1   1   20];
[c,d]=sys(b,a,L,U);
```

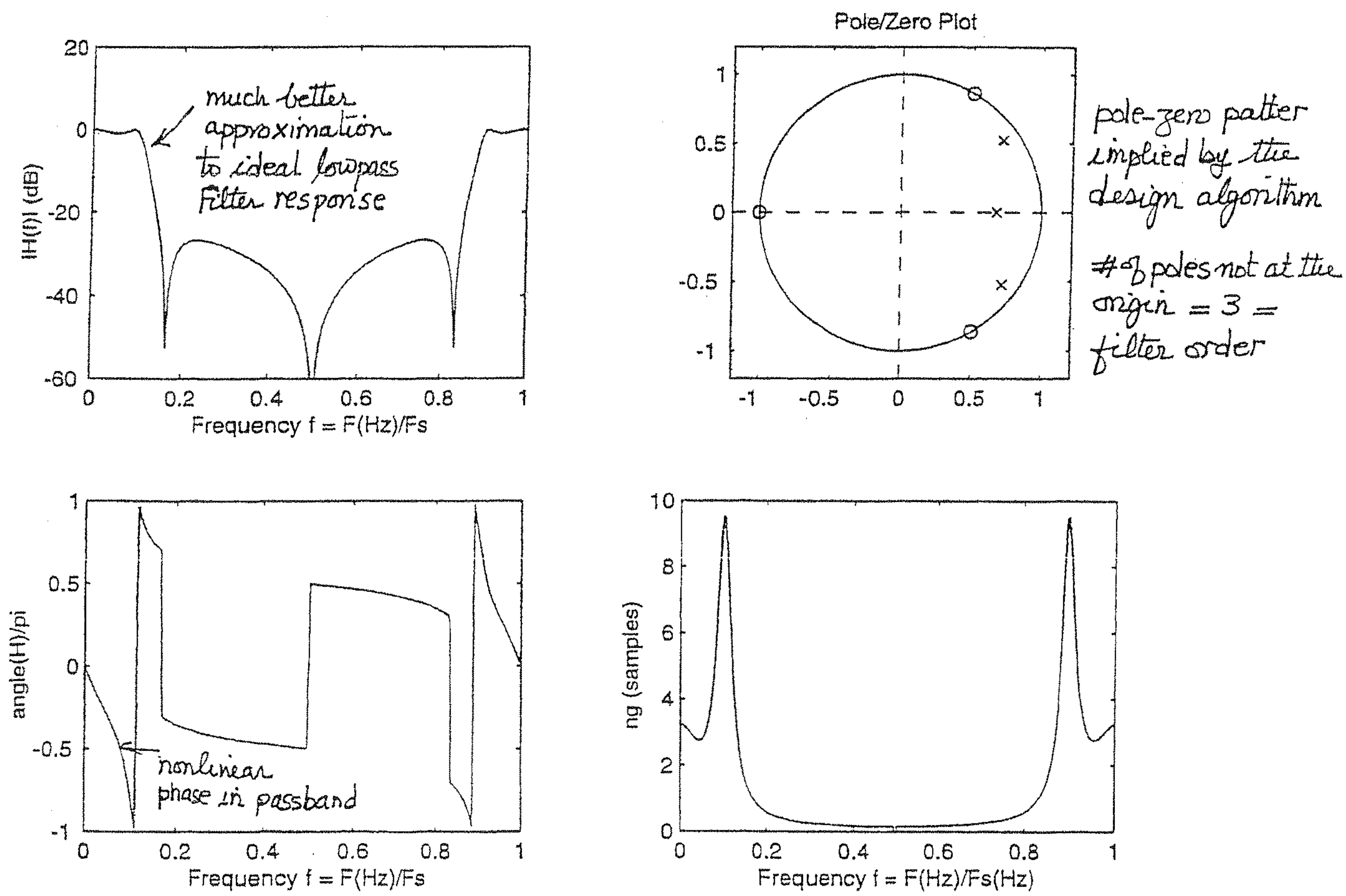








```
% A "good" lowpass filter (from O&S)
% -----------------------------------------------
%
%              0.05643 (1+z^-1)  (1- 1.0166 z^-1 + z^-2)
% H(z) =   ------------------------------------------------------
%              (1 - 0.6830 z^-1)  (1 -  1.4461 z^-1 + 0.7957 z^-2)
%
b0=0.05643;
b1=[1   1];                          % coefficients of the first poly in the numerator
b2=[1  -1.0166   1];                 % coeff of the second poly
b=conv(b1,b2);                       % coeffs of the product poly
b=b0*b;
%
a1=[1  -0.6830];                     % repeat for the denomenator
a2=[1  -1.4461  0.7957];
a=conv(a1,a2);
%
L=[-1.2  -60  -1   0];
U=[1.2    20   1   10];
[c,d] = sys(b,a,L,U);
```



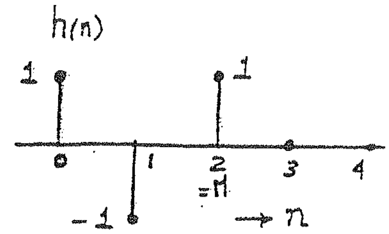
```
function [c,d]=sys(b,a,L,U)
%
% system plots the pole-zero diagram, the frequency response, and
% the group delay of discrete-time LTI systems; a and b are the
% denomenator and numerator coefficients of H(z).
% L and U are arrays containing the lower and upper limits for
% axes in plotted figures (pole-zer, H_mag, H_ARG, and ng, respectively)


% poles and zeros (other than those at the origin)
% ------------------------------------------------
d=roots(b);
c=[];
if(length(a) > 1) c=roots(a); end
%
clf
% pole-zeo plot (could also use the function zplane)
% ------------------------------------------------
subplot(2,2,2),
j=sqrt(-1);
z=exp(j*2*pi*[0:360]/260);
plot(real(z),imag(z),'-'), hold on
if(length(a) > 1)  plot(real(c),imag(c),'x'), end
plot(real(d),imag(d),'o')
plot([0 0],[L(1) U(1)],'--r');
plot([L(1) U(1)],[0 0],'--r');
axis([L(1)  U(1)  L(1)  U(1)]);
title('Pole/Zero Plot')
axis('square');
%
% Frequency Response and Group Delay
% ------------------------------------------------
[H,w]=freqz(b,a,256,'whole');
H_mag=20*log10(abs(H));
H_ARG=angle(H);
ng=grpdelay(b,a,256,'whole');
%
tpi=2*pi;
subplot(2,2,1),plot(w/tpi,H_mag,'-r');
axis([0  1  L(2)  U(2)]);
axis('normal');
xlabel('Frequency f = F(Hz)/Fs'), ylabel('|H(f)| (dB)')
%
subplot(2,2,3),plot(w/tpi,H_ARG/pi,'-r');
axis([0  1  L(3) U(3)]);
xlabel('Frequency f = F(Hz)/Fs'), ylabel('angle(H)/pi')
%
subplot(2,2,4),plot(w/tpi,ng,'-');
axis([0  1  L(4)  U(4)]);
xlabel('Frequency f = F(Hz)/Fs(Hz)'), ylabel('ng (samples)')
axis; subplot
```

see also the Matlab function "zplot"

Digital FIR Filters: Analysis Examples

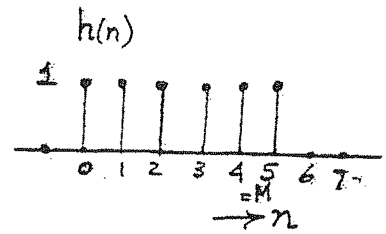
```
% Script File FIR_Anal.m
%
% Type I:  h(n) even, M even
% ---------------------------
% Crude Notch Filter
% h(n)=delta(n)-delta(n-1)+delta(n-2)
% H(z) = 1 - z^-1 + z^-2;
%
a=1;  b=[1 -1 1];
L= [-1.2 -40 -1 0]; U=[1.2 20 1 4];
[c d]=sys(b,a,L,U)
pause
```

```
h(n)   = h(M-n)
H(z^-1)= z^M H(z)

% M=2
```




```
% 2- Type II:  h(n) even, M odd
% ------------------------------
% Boxcar Filter
% h(n)=delta(n)+delta(n-1)+...+delta(n-5) ;
% H(z) = 1+z^-1+z^-2+z^-3+z^-4+z^-5;
%      =(1-z^-6)/(1-z^-1)
%
a=1;  b=[1 1 1 1 1 1];
L= [-1.2 -40 -1 0]; U=[1.2 20 1 4];
[c d]=sys(b,a,L,U)
pause
```

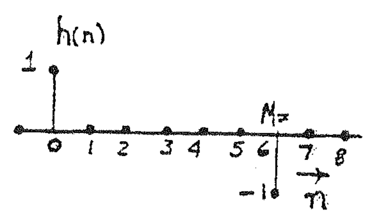
```
h(n)    = h(M-n)
H(z^-1) = z^M H(z)

% M=5
```



zero @ $z = -1$

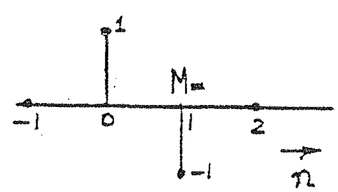
```
% 3- Type III:  h(n) odd, M even
% ------------------------------
% Comb Filter
% h(n)=delta(n)-delta(n-6) ;
% H(z) = 1 - z^-6;
%
a=1;  b=[1 0 0 0 0 0  -1];
L= [-1.2 -30 -1 0]; U=[1.2 10 1 4];
[c d]=sys(b,a,L,U)
pause
%
% 4- Type IV:  h(n) odd, M odd
% ----------------------------
% Crude Differentiator (Crude Highpass Filter)
% h(n)=delta(n)-delta(n-1) ;
% H(z) = 1 - z^-1;
%
a=1;  b=[1 -1];
L= [-1.2 -40 -1 0]; U=[1.2 20 1 4];
[c d]=sys(b,a,L,U)
```

```
h(n)    = -h(M-n)
H(z^-1) = -z^M H(z)

% M=6
```

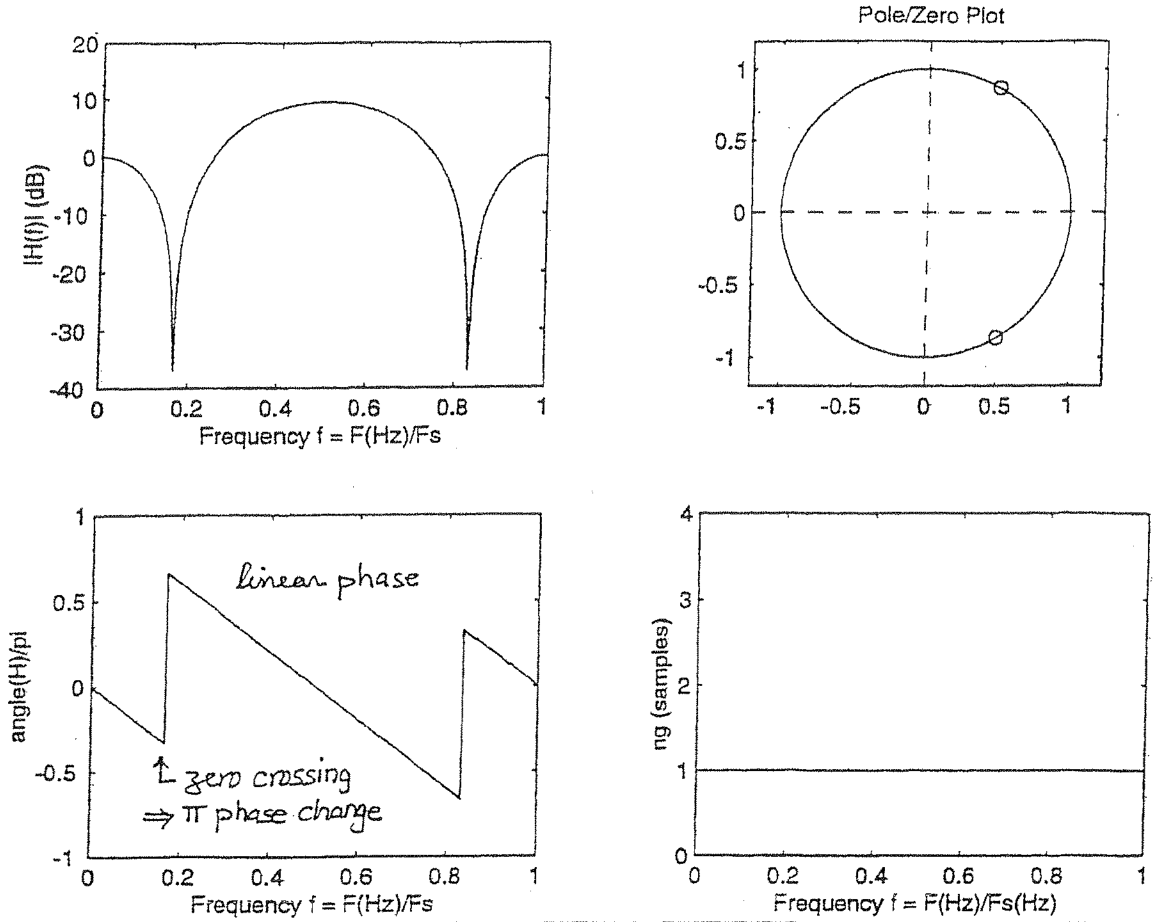


zeros @ $z = \pm 1$

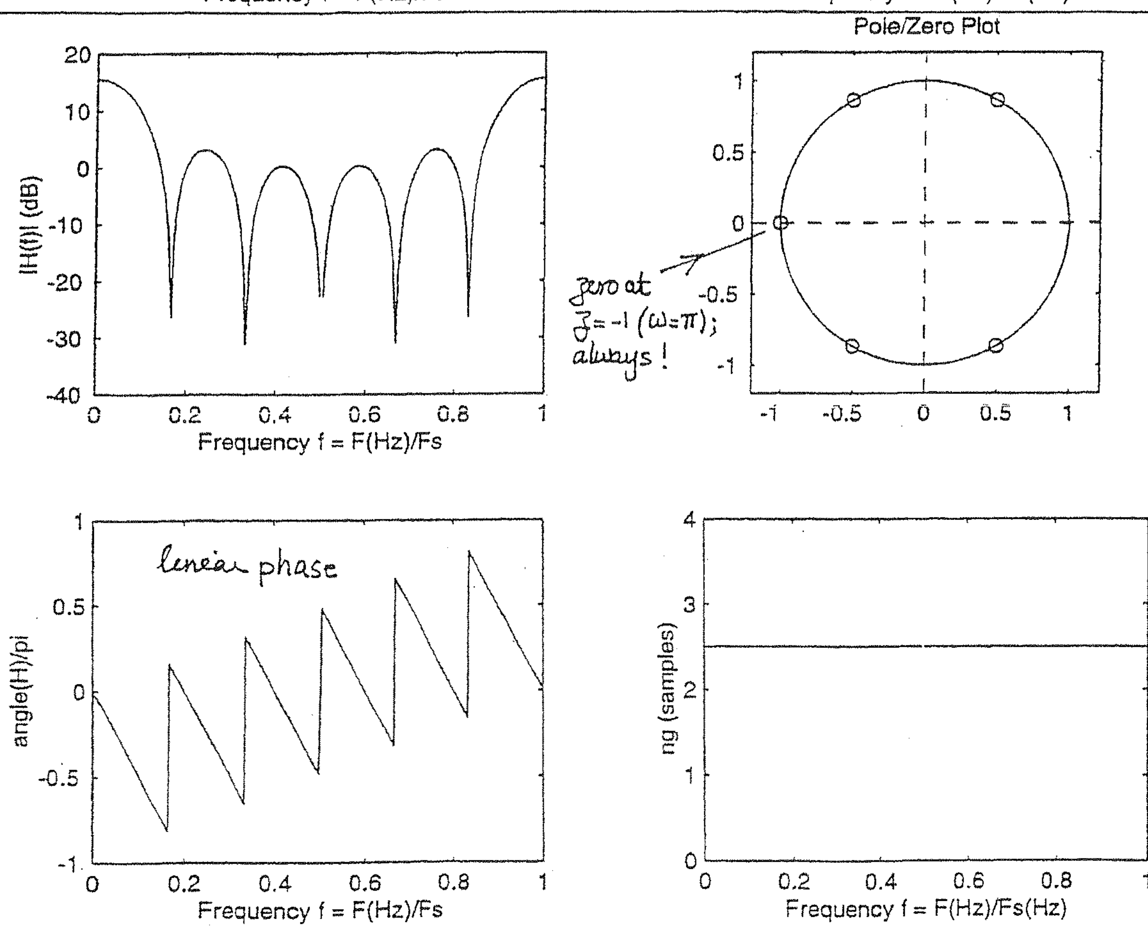
```
h(n)    = -h(M-n)
H(z^-1) = -z^M H(z)

% M=1
```



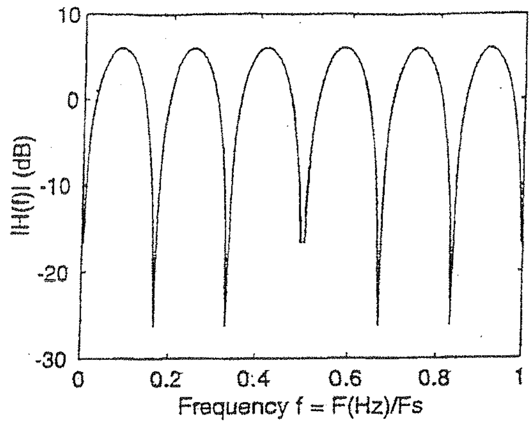
zero @ $z = 1$

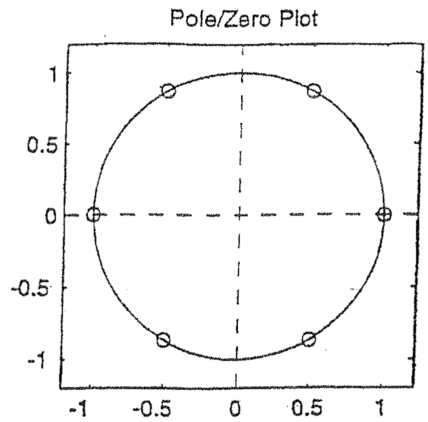


Type I
Crude Notch Filter

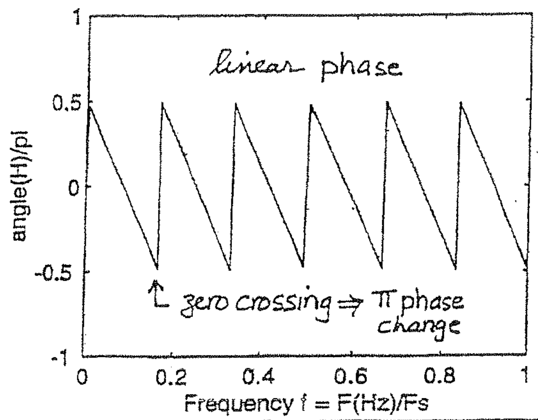


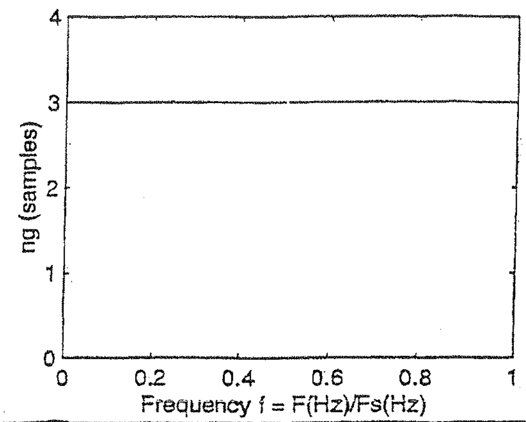
Type II
Lowpass Filter





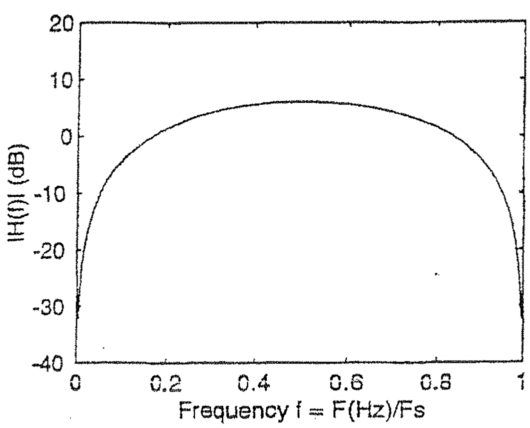
zeros at $z = \pm 1$ $(\omega = 0, \pi)$; <u>always</u>

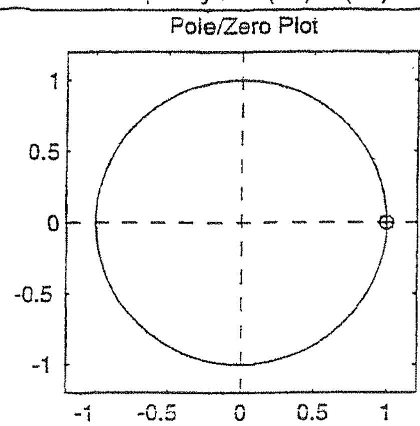




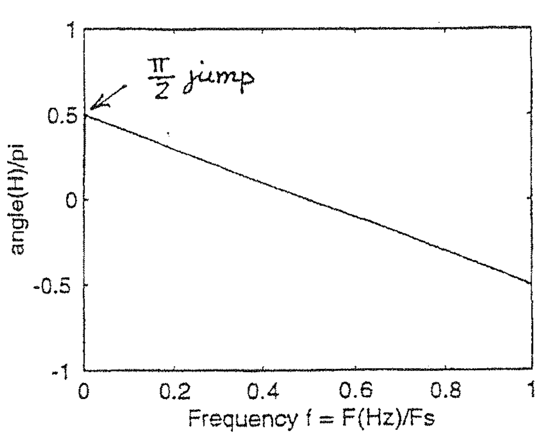
<u>Type III</u>

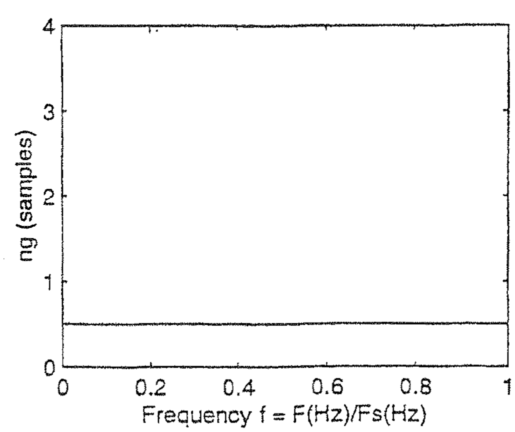
multi-notch comb Filter





zero at $z = 1$ $(\omega = 0)$; always





<u>Type IV</u>

Crude Highpass Filter