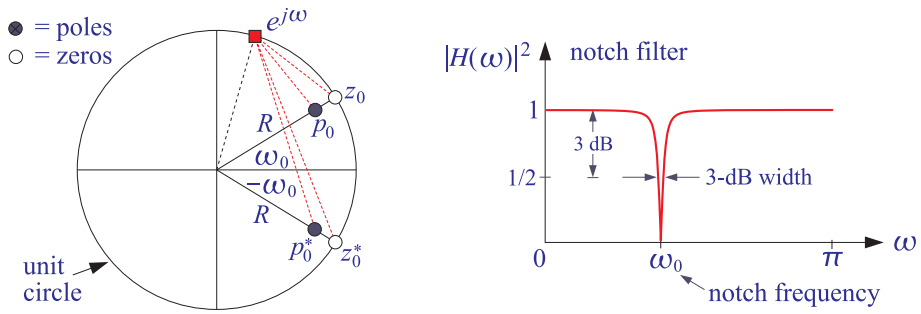
Notch Filter Design Examples – S. J. Orfanidis
Audio Demo and ECG Interference Removal

Notch Filter Design by Pole-Zero Placement

A simple way to design a filter with a notch at some frequency ω_0 (rads/sample) is to construct a transfer function with a zero at $z_0 = e^{j\omega_0}$, and a pole behind the zero at $p_0 = Re^{j\omega_0}$, with $0 < R < 1$. The complex conjugates z_0^*, p_0^* must also be included to make the filter coefficients real-valued. The geometry is depicted below.



The transfer function is constructed by,

$$H(z) = G\,\frac{(1 - e^{j\omega_0}z^{-1})\,(1 - e^{-j\omega_0}z^{-1})}{(1 - Re^{j\omega_0}z^{-1})\,(1 - Re^{-j\omega_0}z^{-1})} = G\,\frac{1 - 2\cos\omega_0\, z^{-1} + z^{-2}}{1 - 2R\cos\omega_0\, z^{-1} + R^2 z^{-2}} \tag{1}$$

The gain factor G may be adjusted to normalize the transfer function to *unity* gain at DC, that is, setting $z = 1$ and requiring $H(z)\,|_{z=1} = 1$, we obtain,

$$H(1) = G\,\frac{1 - 2\cos\omega_0 + 1}{1 - 2R\cos\omega_0 + R^2} = 1 \quad \Rightarrow \quad G = \frac{1 - 2R\cos\omega_0 + R^2}{2\,(1 - \cos\omega_0)} \tag{2}$$

Thus, the transfer function and filter coefficients are,

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_2 z^{-1} + a_2 z^{-2}}\,, \qquad \begin{aligned} \mathbf{b} &= [b_0\,,\ b_1\,,\ b_2] = [G\,,\ -2G\cos\omega_0\,,\ G] \\ \mathbf{a} &= [1\,,\ a_1\,,\ a_2] = [1\,,\ -2R\cos\omega_0\,,\ R^2] \end{aligned} \tag{3}$$

The basic tradeoff for such design is that as the pole radius R gets closer to the unit circle (from inside), the notch becomes sharper, but at the expense of a longer transient response. This tradeoff can be quantified by the following approximate formulas for the 3-dB width Δf in Hz and the effective 40-dB time constant τ_{eff} in seconds, where f_s is the sampling rate in Hz:

$$\Delta f = \frac{f_s}{\pi}(1 - R)\,, \qquad \tau_{\text{eff}} = \frac{1}{f_s}\,\frac{\ln(10^{-2})}{\ln(R)} \approx \frac{1}{f_s}\,\frac{2\ln(10)}{1 - R} \quad \Rightarrow \quad \tau_{\text{eff}}\,\Delta f = \frac{2\ln(10)}{\pi} \approx 4.6 \tag{4}$$

where the approximation is valid for $R \lesssim 1$, very near 1. Eq. (4) is also an example of the uncertainty principle for time-bandwidth product.

An exact design with notch frequency f_0 and arbitrary 3-dB notch width Δf can be found in ch.11 of the I2SP text, and is given by,

$$H(z) = \left(\frac{1}{1 + \beta}\right)\frac{1 - 2\cos\omega_0\, z^{-1} + z^{-2}}{1 - 2\left(\dfrac{\cos\omega_0}{1 + \beta}\right)z^{-1} + \left(\dfrac{1 - \beta}{1 + \beta}\right)z^{-2}} \tag{5}$$

where

$$\omega_0 = \frac{2\pi f_0}{f_s}\,, \quad \Delta\omega = \frac{2\pi\,\Delta f}{f_s}\,, \quad \beta = \tan\left(\frac{\Delta\omega}{2}\right) \tag{6}$$

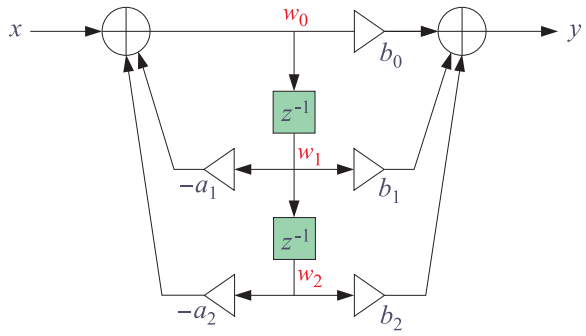
The notching behavior of Eq. (1) can be understood geometrically by looking at the frequency response of the filter, obtained by replacing z by $e^{j\omega}$ in $H(z)$,

$$H(\omega) = G\,\frac{(z-e^{j\omega_0})\,(z-e^{-j\omega_0})}{(z-Re^{j\omega_0})\,(z-Re^{-j\omega_0})}\bigg|_{z=e^{j\omega}} \quad\Rightarrow\quad |H(\omega)| = G\cdot\frac{|e^{j\omega}-z_0|}{|e^{j\omega}-p_0|}\cdot\frac{|e^{j\omega}-z_0^*|}{|e^{j\omega}-p_0^*|}$$

It can be seen from the above figure that, if R is very near the unit circle, then the pole p_0 is very near the zero z_0, and therefore, as soon as the movable point $e^{j\omega}$ on the unit circle moves away from the immediate vicinity of the zero, the distances from $e^{j\omega}$ to z_0 and to p_0 become practically equal, i.e., $|e^{j\omega}-z_0| \approx |e^{j\omega}-p_0|$, and similarly, $|e^{j\omega}-z_0^*| \approx |e^{j\omega}-p_0^*|$, and the magnitude response becomes constant as a function of frequency.

Sample-by-Sample Processing

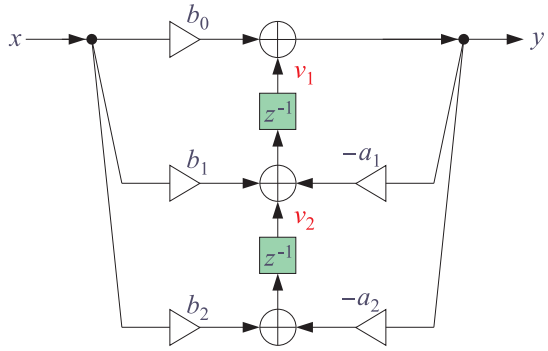
Such filters may be realized in their canonical form (also known as direct-form-2) using linear delay-line buffers as shown below:



for each x do:
$w_0 = x - a_1 w_1 - a_2 w_2$
$y = b_0 w_0 + b_1 w_1 + b_2 w_2$
$w_2 = w_1$
$w_1 = w_0$

(7)

The transposed (of the canonical) form is also used,



for each x do:
$y = b_0 x + v_1$
$v_1 = b_1 x - a_1 y + v_2$
$v_2 = b_2 x - a_2 y$

(8)

These realizations were discussed in class and a small numerical example was given.

Audio Demonstration

In this audio demonstration, we use the approximate pole/zero placement design of Eq. (1), and implement it both in the canonical and the transposed realizations. The input signal is chosen to be a sinusoid of frequency $f_0 = 2$ kHz with duration of one second and amplitude 2, sandwiched between two sinusoids each of frequency $f_1 = 1$ kHz and duration also of one second and unity amplitude. The sampling rate is chosen to be 8 kHz. The input signal is defined explicitly as follows over the period $0 \le t \le 3$ sec, and then sampled at the rate f_s, that is, replacing t by $t_n = nT_s$, $n = 0, 1, 2, \ldots$, where $T_s = 1/f_s$,

$$x(t) = \begin{cases} \cos(2\pi f_1 t), & 0 \le t < 1 \text{ sec} \\ 2\cos(2\pi f_0 t), & 1 \le t < 2 \text{ sec} \\ \cos(2\pi f_1 t), & 2 \le t < 3 \text{ sec} \end{cases}$$

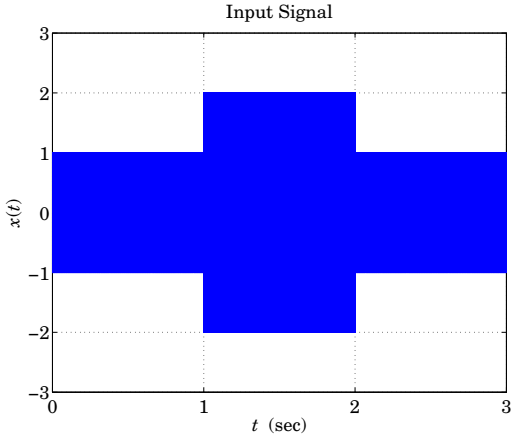
The notch filter is designed to have a notch at f_0, so it will filter out the middle sinusoid. Thus, before filtering, one hears three one-second tones of frequencies 1, 2, 1 kHz, while after filtering, the middle sinusoid is wiped out by the notch filter and one hears silence during the middle second (after the transients die out). By adjusting the value of R, we can make the transients audible.

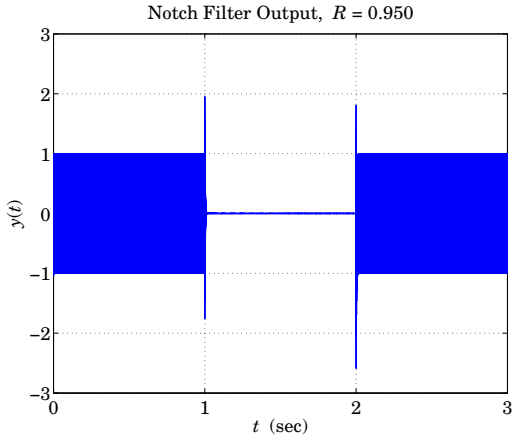
The figures below show the input and output signals, as well as the magnitude response of the notch filter designed for the two values $R = 0.95$ and $R = 0.995$. The corresponding 40-dB time constants, 3-dB widths, and Q-factors are as follows,

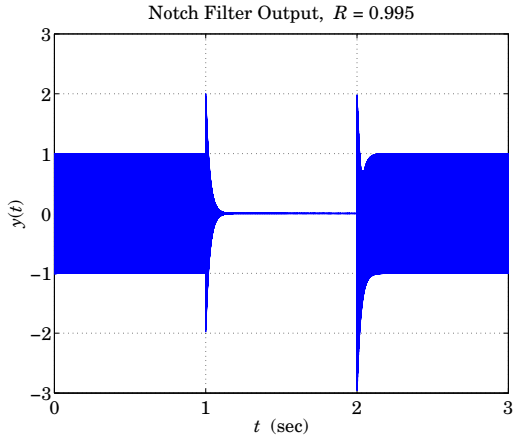
$$R = 0.950\,, \quad \tau_{\text{eff}} = 0.0115 \text{ sec}\,, \quad \Delta f = 127.3240 \text{ Hz}\,, \quad Q = \frac{f_0}{\Delta f} = 15.71$$

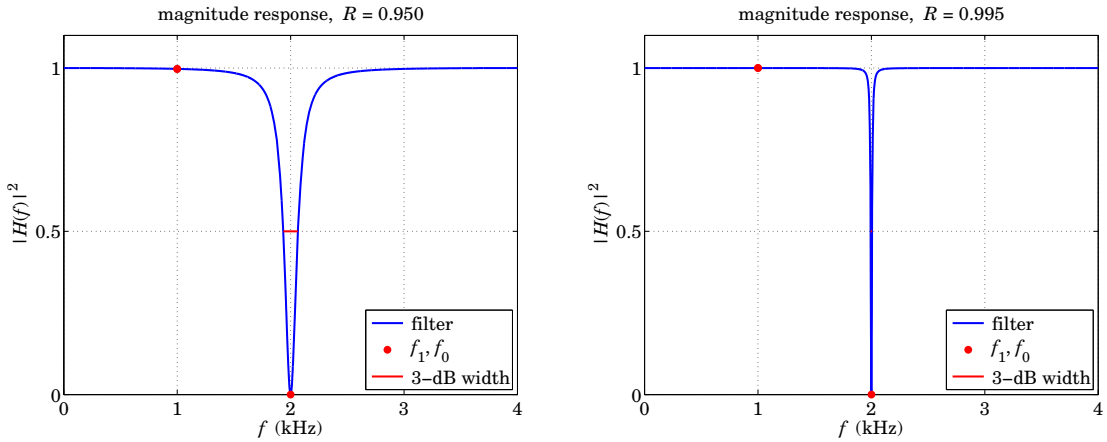
$$R = 0.995\,, \quad \tau_{\text{eff}} = 0.1150 \text{ sec}\,, \quad \Delta f = 12.7324 \text{ Hz}\,, \quad Q = \frac{f_0}{\Delta f} = 157.1$$

The tradeoff between τ_{eff} and Δf is evident. Indeed, because $1 - R$ is equal to 0.05 and 0.005 in the two cases, the time constant is increased by a factor of 10 while the 3-dB width is decreased by a factor of 10 and the Q-factor is increased by a factor of 10.









MATLAB Implementation

```
% notch filter audio demo

fs = 8000;          % sampling rate in samples/sec, or, Hz
Ts = 1/fs;          % Ts = 1/fs = 1/8000 = 0.125 msec
Tot = 3;            % total duration 3 sec (3*8000 = 24000 samples)

t = 0 : Ts : Tot;   % sampled time is steps of Ts = 1/fs

f1 = 1000;          % 1 kHz
f0 = 2000;          % 2 kHz

x =   cos(2*pi*f1*t) .* (t<1) + ...            % sampled input signal
    2*cos(2*pi*f0*t) .* (t>=1 & t<2) + ...
      cos(2*pi*f1*t) .* (t>=2);

% discretized version

R = 0.995; c0 = cos(2*pi*f0/fs);    % may change R to other values

neff = 2*log(10)/(1-R);    % 40-dB time constant in samples
teff = neff/fs             % 40-dB time constant in seconds

Dw = 2*(1-R);              % 3-dB width in rads/sample
Df = fs*Dw/2/pi            % 3-dB width in Hz
Q = f0/Df                  % Q-factor

G = (1 - 2*R*c0 + R^2)/(1-c0)/2;     % gain fctor, G

a1 = -2*R*c0;     % denominator coefficients a1,a2
a2 = R^2;

b0 = G;           % numerator coefficients, b0,b1,b2
b1 = -2*c0*G;
b2 = G;
```

```
% compute output using canonical realization
% ------------------------------------------

w1=0; w2=0;                          % initialize delay registers
for n = 1:length(x)                  % sample processing algorithm
   w0 = x(n) - a1*w1 - a2*w2;
   y(n) = b0*w0 + b1*w1 + b2*w2;
   w2 = w1;
   w1 = w0;
end

% sound(x,fs)             % listen to the input
% pause;
% sound(y,fs)             % listen to the output

% sound([x, zeros(1,2*fs), y], fs)

% compare with output from transposed realization
% ----------------------------------------------

v1=0; v2=0;                          % initialize delay registers
for n = 1:length(x)                  % sample processing algorithm
   yt(n) = b0*x(n) + v1;             % use different symbol for y(n)
   v1 = b1*x(n) - a1*yt(n) + v2;
   v2 = b2*x(n) - a2*yt(n);
end

% compare with output from the built-in FILTER function
% -----------------------------------------------------

b = [b0,b1,b2];        % filter coefficient vectors
a = [1,a1,a2];
yf = filter(b,a,x);    % run FILTER with zero initial conditions

norm(y-yt)     % should be zero, theoretically
norm(yt-yf)    % should be zero, theoretically

% plot input and output signals x(t) and y(t)
% ------------------------------------------

figure; plot(t,x,'b-');
yaxis(-3,3, -3:3); grid;
xaxis(0,3, 0:3);
title('Input Signal');
xlabel('{\itt}  (sec)');
ylabel('{\itx}({\itt})');

figure; plot(t, y, 'b-');
yaxis(-3,3, -3:3); grid;
xaxis(0,3, 0:3);
title(['Notch Filter Output,  {\itR} = ',num2str(R,'%5.3f')]);
xlabel('{\itt}  (sec)');
ylabel('{\ity}({\itt})');

% plot magnitude response |H(f)| vs. f
% -----------------------------------

f = linspace(0,4000,4001);    % frequency in Hz
w = 2*pi*f/fs;                % frequency in rads/sample
```

```
H = abs(freqz(b,a,w)).^2;     % use FREQZ

f10 = [f1,f0];                    % sinusoid frequencies
w10 = 2*pi*f10/fs;
H10 = abs(freqz(b,a,w10)).^2;   % response at sinusoid frequencies

df = fs/pi*(1-R);                 % 3-dB width
fL = f0-df/2; fR = f0+df/2;       % approximate left/right 3-dB frequencies

figure; plot(f/1e3,H, 'b-', 'linewidth',2); hold on     % plot in units of kHz
plot(f10/1e3,H10, 'r.', 'markersize',23);
plot([fL,fR]/1000, [1/2,1/2], 'r-', 'linewidth',2);
title(['magnitude response,  {\itR} = ',num2str(R,'%5.3f')]);
xlabel('{\itf}  (kHz)');
ylabel('|{\itH}({\itf})|^2');
xaxis(0,4, 0:1:4);
yaxis(0,1.1, 0:0.5:1); grid;
legend(' filter', ' {\itf}_1, {\itf}_0', ' 3-dB width', 'location','se')
```

ECG Example – Removing 60-Hz Interference

Another common application of notch filters is the removal of 60-Hz power-frequency interference in electrocardiogram (ECG) recordings. In this demo, there are two ECG recordings, each of duration of two seconds, spanning approximately two heart beats, and recorded at a rate of $f_s = 1000$ Hz. The first is the noise-free ECG included here for reference only, and the second is the ECG superimposed with 60-Hz interference.

To complete the design, we specify the desired 3-dB width Δf in Hz and then estimate the pole radius R using the approximation of Eq. (4),

$$\Delta\omega = \frac{2\pi\Delta f}{f_s} = 2(1-R) \quad \Rightarrow \quad R = 1 - \pi\,\frac{\Delta f}{f_s}$$

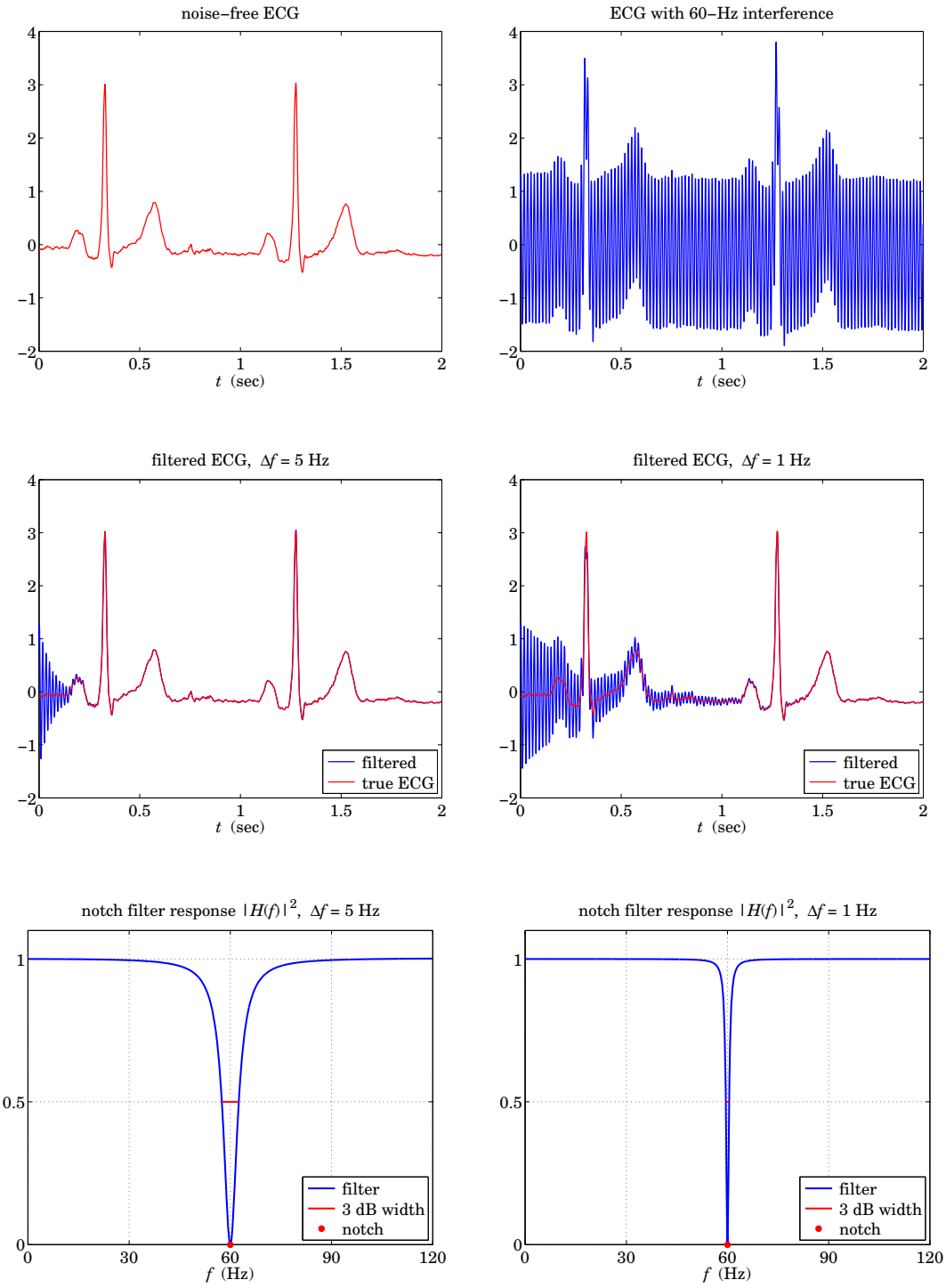
so that with, $f_0 = 60$ Hz, and, $\omega_0 = 2\pi f_0/f_s$, all the parameters of the notch filter $H(z)$ are fixed. Alternatively, we may use the exact design of Eq. (5).

The figures below show the noise-free and noisy input and the corresponding output signals, as well as the magnitude responses of the notch filters designed with the two values of $\Delta f = 5$ and $\Delta f = 1$ Hz. The corresponding 40-dB time constants, pole radii, and Q-factors are as follows,

$$\Delta f = 5 \text{ Hz}, \quad R = 0.9843, \quad \tau_{\text{eff}} = 0.29 \text{ sec}, \quad Q = \frac{f_0}{\Delta f} = 12$$

$$\Delta f = 1 \text{ Hz}, \quad R = 0.9969, \quad \tau_{\text{eff}} = 1.47 \text{ sec}, \quad Q = \frac{f_0}{\Delta f} = 60$$

And, again we observe the tradeoff between narrowness of the notch and time constant. The relevant MATLAB code is included at the end.



MATLAB Implementation

```
% ECG example

x0 = load('ecg0.dat');      % noise-free ECG - for comparison only
x = load('ecg60.dat');      % noisy ECG

fs = 1000; T = 1/fs;        % sampling rate
```

```
N = length(x);
t = (0:N-1)'*T;

f0 = 60;                    % Hz
w0 = 2*pi*f0;               % rad/sec

figure; plot(t,x0,'r-');   % noise-free ECG
title('noise-free ECG');
xaxis(0,2,0:0.5:2);
yaxis(-2,4,-2:1:4);
xlabel('{\itt}  (sec)'); ylabel('{\itx}_0({\itt})');

print -depsc ecgfree.eps

Df = 5;                     % 3-dB width in Hz, try also Df = 1, Df = 10
Dw = 2*pi*Df/fs;            % 3-dB width in rads/sample
R = 1 - Dw/2;               % approximate pole radius
c0 = cos(2*pi*f0/fs);       % defined for convenience

neff = 2*log(10)/(1-R);     % 40-dB time constant in samples
teff = neff/fs              % 40-dB time constant in seconds

Dw = 2*(1-R);               % 3-dB width in rads/sample
Df = fs*Dw/2/pi             % 3-dB width in Hz
Q = f0/Df                   % Q-factor

G = (1 - 2*R*c0 + R^2)/(1-c0)/2;     % normalization gain fctor, G

a1 = -2*R*c0;     % denominator coefficients a1,a2
a2 = R^2;

b0 = G;           % numerator coefficients, b0,b1,b2
b1 = -2*c0*G;
b2 = G;

% compute output using canonical realization
% ------------------------------------------

w1=0; w2=0;                          % initialize delay registers
for n = 1:length(x)                  % sample processing algorithm
   w0 = x(n) - a1*w1 - a2*w2;
   y(n) = b0*w0 + b1*w1 + b2*w2;
   w2 = w1;
   w1 = w0;
end

% compare with output from transposed realization
% -----------------------------------------------

v1=0; v2=0;                          % initialize delay registers
for n = 1:length(x)                  % sample processing algorithm
   yt(n) = b0*x(n) + v1;
   v1 = b1*x(n) - a1*yt(n) + v2;
   v2 = b2*x(n) - a2*yt(n);
end

% compare with output from the built-in FILTER function
% -----------------------------------------------------
```

```
b = [b0,b1,b2];        % filter coefficient vectors
a = [1,a1,a2];
yf = filter(b,a,x');   % run FILTER with zero initial conditions

norm(y-yt)             % should be zero, theoretically
norm(yt-yf)            % should be zero, theoretically

% plot input and output signals x(t) and y(t)
% -------------------------------------------

figure; plot(t,x,'b-');              % noisy ECG
yaxis(-2,4, -2:4);
xaxis(0,2, 0:0.5:2);
title('ECG with 60-Hz interference');
xlabel('{\itt}  (sec)');
ylabel('{\itx}({\itt})');

print -depsc ecg60hz.eps

figure; plot(t,y,'b', t,x0,'r-');    % filtered ECG
xaxis(0,2,0:0.5:2);
yaxis(-2,4,-2:1:4);
title(['filtered ECG,  \Delta{\itf} = ',num2str(Df),' Hz']);
xlabel('{\itt}  (sec)');
ylabel('{\ity}({\itt})');
legend(' filtered', ' true ECG','location','se');

print('-depsc',['ecg',num2str(Df),'.eps']);


% plot notch filter's magnitude frequency response
% ------------------------------------------------

f = linspace(0,120,2401);   % plot over 0 < f < 120 Hz

f1 = f0-Df/2;               % approximate left/right 3-dB frequencies
f2 = f0+Df/2;

w = 2*pi*f/fs;                                      % calculate |H(f)|^2
H2 = abs( (b0 + b1*exp(-j*w) + b2*exp(-2*j*w)) ./ ...
          (1 + a1*exp(-j*w) + a2*exp(-2*j*w)) ).^2;

% H2 = abs(freqz(b,a,w)).^2;        % alternative calculation

figure; plot(f,H2,'b-'); hold on
xaxis(0,120, 0:30:120);
yaxis(0,1.1, 0:0.5:1); grid
xlabel('{\itf}  (Hz)');
plot([f1,f2],[0.5,0.5], 'r-', 'linewidth',2)
plot(f0,0,'r.','markersize',22)
title(['notch filter response |{\itH}({\itf})|^2,  \Delta{\itf} = ',num2str(Df),' Hz']);
legend(' filter', ' 3 dB width', ' notch','location','se');
```