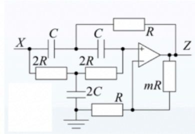
1. In the lab we analyzed filtering 60 Hz power-line noise from ECG signal using a digital (signal processing) filter. Now let's try to an analog (circuit) filter approach to remove the 60 Hz line-noise. Following is an active twin-T notch filter with transfer function:

$$H(\omega) = \frac{Z(\omega)}{X(\omega)} = \frac{(1+m)\left((2j\omega RC)^2+1\right)}{(2j\omega RC)^2+4(1-m)j\omega RC+1}$$



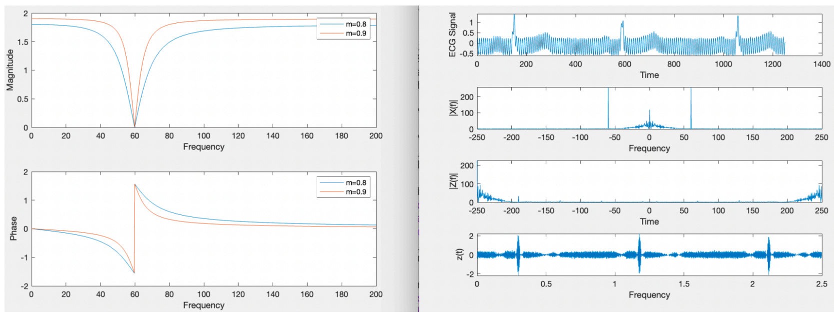
Here `m` is the ratio of the two feedback resistance which determines the gain and quality for the filter. The drop frequency of this twin-T notch filter is $f_{drop}=1/4\pi RC$. For designing a 60 Hz drop filter, let's use `R=10 kΩ` and `C=133 nF`.

(a) For `m={0.8, 0.9}` plot the magnitude and phase response of `H(ω)` with a range of `f=ω/2π=[0, 200 Hz]`.

(b) Consider the ECG signal used during the class (`ecg_signal.mat`) as the input (`x(t)=ecg`) of a 60 Hz twin-T notch filter with `m=0.9`. Using the functions `fft()` and `ifft()`, determine the `X(ω)`, `Z(ω)`, and `z(t)`[`z(t)` is the output signal from the twin-T notch filter]. Plot `x(t)`, `X(f)`, `Z(f)`, and `z(t)` in a 4x1 subplot for the range of $-250 \le f \le 250$ and $0 \le t \le 2.5$.
[Please pay attention to the proper use of `fftshift()` and `ifftshift()` while solving this problem.]

Summary:
We define H(w) as above for m = 0.8 and 0.9. We plot the phase and magnitude of each. We then plot the ecg signal x(t) and X(f), Z(f), and z(t) using fft() and ifft().

Results:



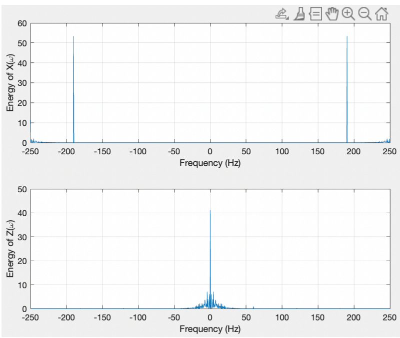
Code:
```
R = 10000; % define variables and ranges
C = 133e-9;
F = linspace(0, 200, 1000);
W = 2*pi*F;
M = 0.8;
H1 = ((1+M).*((2*1j*W*R*C).^2+1))./((2*1j*W*R*C).^2+(4*(1-M).*(1j*W*R*C))+1); % h for m = 0.8
M = 0.9;
H2 = ((1+M).*((2*1j*W*R*C).^2+1))./((2*1j*W*R*C).^2+(4*(1-M).*(1j*W*R*C))+1);% h for m = 0.9
subplot(2, 1, 1)
plot(F, abs(H1), 'DisplayName', 'm=0.8') % plot magnitude for m = 0.8
hold on;
plot(F, abs(H2), 'DisplayName', 'm=0.9') % plot magnitude for m = 0.9
legend show;
ylabel('Magnitude')
xlabel('Frequency')
subplot(2, 1, 2)
plot(F, angle(H1), 'DisplayName', 'm=0.8') % plot phase for m = 0.8
hold on;
plot(F, angle(H2), 'DisplayName', 'm=0.9') % plot phase for m = 0.9
legend show;
ylabel('Phase')
xlabel('Frequency')
load ecg_signal.mat; % load for part B
F = linspace(-250, 250, length(ecg)); % define new variables
W=2*pi*F;
H = ((1+M).*((2*1j*W*R*C).^2+1))./((2*1j*W*R*C).^2+(4*(1-M).*(1j*W*R*C))+1);
figure;
subplot(4, 1, 1)
plot(ecg) % plot ecg signal
ylabel('ECG Signal')
xlabel('Time')
subplot(4, 1, 2)
plot(F, abs(fftshift(fft(ecg)))); % plot magnitude of X(f)
ylabel('|X(f)|')
xlabel('Frequency')
subplot(4, 1, 3)
plot(F, abs(fftshift(fftshift(fft(ecg)).*H))) % plot magnitude of Z(f)
ylabel('|Z(f)|')
xlabel('Time')
subplot(4, 1, 4)
plot(t, real(ifft(fftshift(fft(ecg)).*H))) % plot z(t)
ylabel('z(t)')
xlabel('Frequency')
```

2. Calculate the energy of time domain signal `x(t)` and `z(t)` for the range of $0 \le t \le 2.5$. Also calculate the energy of these signals in frequency domain using Parseval's theorem. Plot `Energy(X)` and `Energy(Z)` as a function of frequency `f` in a 2x1 subplot (Energy vs frequency plot is know as energy spectrum of a signal).

Summary:

We define H as in problem 1, X using fft(ecg), and Z using fft(X).*H. We calculate the energies of x and z in the time and frequency domains and print. We then plot Energy vs frequency for Z(w) and X(w).

Results:




```
Energy of x(t) in the time domain
  156.2210

Energy of z(t) in the time domain
  178.4664

Energy of X(w) in the frequency domain
  156.2210

Energy of Z(w) in the frequency domain
  178.4664
```

Code:

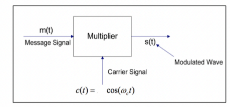
```
load ecg_signal.mat;
R = 10000; % define constants
C = 133e-9;
M = 0.9;
F = linspace(-250, 250, length(ecg)); % define range
W = 2 * pi * F;
H = ((1 + M).*((2*1j*W*R*C).^2+1))./((2*1j*W*R*C).^2+(4*(1-M).*(1j*W*R*C))+1); % calc freq response
X = fft(ecg); % time domain output signal
Z = fftshift(X).*H; % filtered signal Z(f)
z = ifft(Z);
energyx = sum(abs(ecg).^2); % calc energies
energyz = sum(abs(z).^2);
disp("Energy of x(t) in the time domain");
disp(energyx);
disp("Energy of z(t) in the time domain");
disp(energyz);
frequencyEnergyX = sum(abs(X).^2)/length(X);
disp("Energy of X(w) in the frequency domain")
disp(frequencyEnergyX);
frequencyEnergyZ = sum(abs(Z).^2)/length(Z);
disp("Energy of Z(w) in the frequency domain");
disp(frequencyEnergyZ);
frequencyEX = (abs(X).^2)/length(X); % array of the values computed for the energy
frequencyEZ = (abs(Z).^2)/length(Z);
figure(1); % plot energy spectrums
subplot(2,1,1);
plot(F,frequencyEX);
xlabel("Frequency (Hz)")
ylabel("Energy of X(\omega)");
grid on;
subplot(2,1,2);
plot(F, frequencyEZ);
xlabel("Frequency (Hz)")
ylabel("Energy of Z(\omega)")
grid on;
```

3. Let's say you are using a Double-Sideband Suppressed Carrier Modulation (DSB-SC) scheme to transmit a message `m=[6 0 4 -6 2]` to your friend at San Jose over a communication channel that has good transmission characteristics in the frequency range of 400 kHz to 600 kHz. You decided to modulate your message with carrier signal $c(t)=\cos(1000\text{x}10^3\pi t)$ and encode your message `m(t)` at `1/10` of the carrier frequency (i.e. `50 kHz`). Your friend received the signal you transmitted and demodulated it by multiplying with a local oscillator signal $lo(t)=\cos(1000\text{x}10^3\pi t+\pi/3)$ and then passing the signal through a low-pass filter with transfer function `H(f)` where:

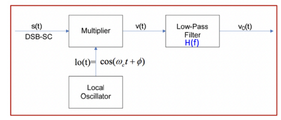
$$H(f) = \begin{cases} 2, & |f| < 500 \times 10^3\ Hz \\ 0, & elsewhere \end{cases}$$

In a 4x1 subplot show the time domain signal transmitted `s(t)`, frequency domain magnitude of the transmitted signal $|S(f)|$, time domain demodulated and low-pass filtered output signal `vo(t)` and corresponding frequency domain spectrum `|Vo(f)|`.

Your modulation scheme:

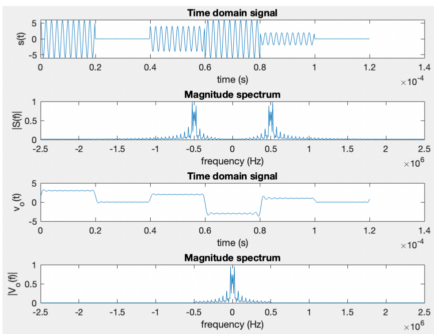


Demodulation scheme of your friend:



Summary:
We define the above constants, signals, and functions. We plot s(t) and its magnitude spectrum and v_o(t) and its magnitude spectrum.

Results:



Code:

```
mFrequency = 5e4; % define the constants and the signals
cFrequency = 5e5;
cTime = 1/10/cFrequency; % Sampling period with 10 samples per
time = 0:cTime:6/mFrequency; % Time vector
message = [6, 0, 4, -6, 2];
mArray = zeros(size(time)); % array for the message signal values
intervals = (0:length(message))/mFrequency;% generate the message signal
for i = 1:length(message) % loop each interval assigning corresponding char
   mArray((time >= intervals(i)) & (time < intervals(i+1))) = message(i);
end
s = mArray.*cos(2*pi*cFrequency*time); % define the DSB-SC signal
v = s.*cos(2*pi*cFrequency*time+pi/3); % define the Demodulated signal
N = length(time);
freq = (-(N-1)/2:N/2)*1/cTime/N; % frequency axis for the fast fourier transform function
fftv = zeros(size(v)); % Performing the low-pass filtering
fftw = fftshift(fft(v));
for i = 1:length(freq)
   if(abs(freq(i))<5e5)
       fftv(i) = 2*fftw(i); % passing low pass frequencies
   end
end
v_out = ifft(ifftshift(fftv));
subplot(4, 1, 1); % plot signals and spectrums
plot(time, s);
xlabel('time (s)');
ylabel('s(t)');
title('Time domain signal');
subplot(4, 1, 2);
plot(freq, fftshift(abs(fft(s)))/N);
xlabel('frequency (Hz)');
ylabel('|S(f)|');
title('Magnitude spectrum');
subplot(4, 1, 3);
plot(time, v_out);
xlabel('time (s)');
ylabel('v_o(t)');
title('Time domain signal');
subplot(4, 1, 4);
plot(freq, fftshift(abs(fft(v_out)))/N);
xlabel('frequency (Hz)');
ylabel('|V_o(f)|');
title('Magnitude spectrum');
```