ECE 348: Digital Signal Processing Lab
Lab 4 (Spring 2020)

Waheed U. Bajwa

March 20, 2020

General Instructions

Please submit a report that carefully answers the problems posed in here, and include all graphs and Matlab code. Examples of all the graphs that you need to generate in this lab and include in your report are contained in this handout. The reports must be uploaded to Sakai under Assignments within the specified time frame. The written report will count *only if* you had attended *and* worked on the lab during the *full* double-period lab session in which you are registered.

1 Filter Realization Examples

Consider the second-order IIR filter:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}. \tag{1}$$

The time-domain operation of this filter can be structured in different ways corresponding to different block-diagram realizations. See for example Figures 8.13 and 8.14 of the text. In the *Direct Form I* realization, the calculation of the output signal $y[n]$ from the input signal $x[n]$ is implemented by the difference equation:

$$y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] - a_1 y[n-1] - a_2 y[n-2]. \tag{2}$$

Assuming zero initial conditions, $x[-1] = x[-2] = y[-1] = y[-2] = 0$, the first two output samples are computed by

$$\begin{aligned} y[0] &= b_0 x[0], \\ y[1] &= b_0 x[1] + b_1 x[0] - a_1 y[0]. \end{aligned} \tag{3}$$

Afterward, (2) can be iterated for $n \geq 2$.

Another realization is the *transposed* (of the *Direct Form II*) realization, which is used internally by the MATLAB function **filter**, and is based on the system of difference equations, for $n \geq 0$,

$$\begin{aligned} y[n] &= b_0 x[n] + v_1[n] \\ v_1[n+1] &= b_1 x[n] - a_1 y[n] + v_2[n] \\ v_2[n+1] &= b_2 x[n] - a_2 y[n], \end{aligned} \tag{4}$$

where $v_1[n]$ and $v_2[n]$ are internal state-variables. Note that (4) is usually initialized at $v_1[0] = v_2[0] = 0$. The built-in function **filter** also keeps track of a similar internal state-variable vector and its more general usage is as follows:

```
[y,vout] = filter(b,a,x,vin),
```

where $\mathbf{v}_{\text{in}}$ and $\mathbf{v}_{\text{out}}$ are the initial and final state vectors; that is, if (4) was iterated for $0 \le n \le N-1$, then the last computed output and states were $y[N-1], v_1[N], v_2[N]$, so that

$$\mathbf{v}_{\text{in}} = [v_1[0]; v_2[0]]\,, \qquad \mathbf{v}_{\text{out}} = [v_1[N]; v_2[N]]\,.$$

Problem 1.1 (15 points)**.** Write two functions, **direct.m** (7.5 points) and **tran.m** (7.5 points), that implement (2) and (4) with usages:

```
y = direct(b,a,x);              % Direct Form I

y = tran(b,a,x);                % Transposed
[y,vout] = tran(b,a,x,vin);     % with arbitrary vin
[y,vout] = tran(b,a,x);         % with vin = [0; 0]
```

Problem 1.2 (5 points)**.** Test your functions on the following example:

$$\mathbf{x} = [1,\,1,\,2,\,1,\,2,\,2,\,1,\,1]'\,, \qquad H(z) = \frac{4 + 2.4z^{-1} - 1.6z^{-2}}{1 - 0.5z^{-1} + 0.6z^{-2}}\,,$$

and compare their outputs to that of **filter** (2.5 points). Assuming $\mathbf{v}_{\text{in}} = [0;\,0]$, verify (2.5 points) that the output state vector $\mathbf{v}_{\text{out}}$ is the same for **tran** and **filter**.

Problem 1.3 (5 points)**.** Using the same example, run your function **tran** on a sample-by-sample basis (i.e. within a for-loop), printing with **fprintf** the quantities $x[n], y[n], v_1[n], v_2[n]$, and generating a table as shown below, where you must replace the * entries by their computed values.

```
 n    x       y          v1          v2
------------------------------------------
 0    1    *.*****    0.000000    0.000000
 1    1    *.*****    *.******    *.******
 2    2   **.*****    *.******    *.******
 3    1    *.*****    *.******    *.******
 4    2    *.*****    *.******    *.******
 5    2    *.*****    *.******    *.******
 6    1    *.*****    *.******    *.******
 7    1    *.*****    *.******    *.******
 8    -       -      -2.879575   -2.485510
```

2 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; see Fig. 1.

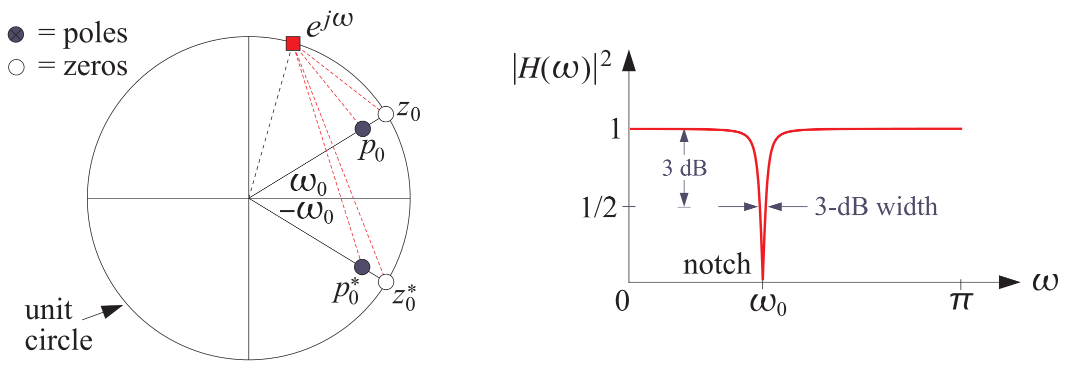


Fig. 1: Notch filter design by pole-zero placement.

Thus, the transfer function is given by:

$$H(z) = G\,\frac{(z - e^{j\omega_0})(z - e^{-j\omega_0})}{(z - Re^{j\omega_0})(z - Re^{-j\omega_0})} = G\,\frac{1 - 2\cos(\omega_0)\,z^{-1} + z^{-2}}{1 - 2R\cos(\omega_0)\,z^{-1} + R^2 z^{-2}}. \tag{5}$$

Note that the gain factor G may be adjusted to normalize the transfer function to *unity* gain at DC.

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:

$$\Delta f_a = \frac{f_s}{\pi}(1 - R)\,, \qquad \tau_{\text{eff}} = \frac{1}{f_s}\,\frac{\ln(0.01)}{\ln(R)}. \tag{6}$$

In this lab you will design two such notch filters and study their time-domain filtering properties, as well as demonstrate the above tradeoff and the validity of the 3-dB width approximation formula. During this lab, you may find the following anonymous MATLAB functions useful, where we note that **freqz** was avoided because it cannot compute at a single frequency point, requiring always a frequency vector of length at least two:

```
fresp = @(b,a,w) polyval(flip(b),exp(-j*w))./polyval(flip(a),exp(-j*w));
mag   = @(b,a,w) abs(fresp(b,a,w));
```

Problem 2.1 (30 points)**.** Design two notch filters (15 points) operating at a rate of $f_s = 200$ Hz with a notch frequency at $f_0 = 4$ Hz, for the following values of R:

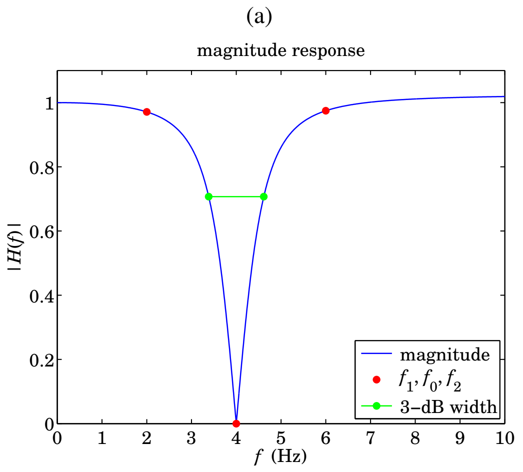
$$R = 0.980 \quad \text{(filter 1)},$$
$$R = 0.995 \quad \text{(filter 2)}.$$

With the help of **fprintf**, make a table of filter coefficients (5 points) as follows, replacing the * entries with the computed values:

```
filter 1
--------------------------------------
b = [ *.******    *.******    *.****** ]
a = [ *.******    *.******    *.****** ]

filter 2
--------------------------------------
b = [ *.******    *.******    *.****** ]
a = [ *.******    *.******    *.****** ]
```

Make a plot of the magnitude responses $|H(f)|$ of these filters over the range $0 \le f \le 10$ Hz (10 points). See example graphs in Fig. 2, where additional features have been added as asked in Problem 2.2.



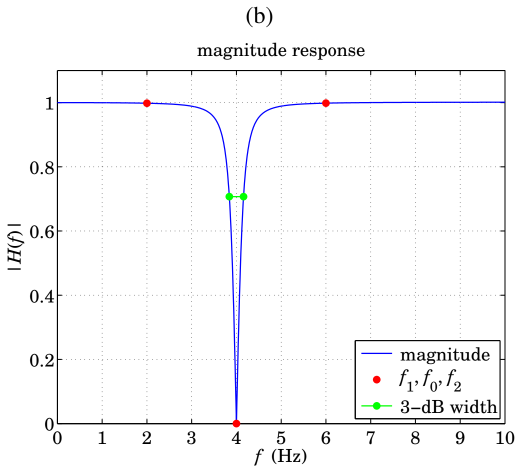


Fig. 2: Plots for Problem 2.1 and Problem 2.2.

Problem 2.2 (17.5 points)**.** Since the responses were normalized to unity gain at DC, the left and right 3-dB frequencies f_L, f_R can be obtained by solving the following equation with respect to f:

$$|H(f)|^2 = \frac{1}{2} \quad \Rightarrow \quad |H(f)| = \frac{1}{\sqrt{2}}. \tag{7}$$

For each filter, solve this equation with the help of **fzero** (5 points). You may choose the initial search point for **fzero** to be the approximate left and right 3-dB frequencies obtained using the approximation of (6), that is,

$$\Delta f_a = \frac{f_s}{\pi}(1-R) \quad \Rightarrow \quad f_{La} = f_0 - \frac{1}{2}\Delta f_a\,, \quad f_{Ra} = f_0 + \frac{1}{2}\Delta f_a.$$

Hint: Pass the following function handle into **fzero**, assuming that **b**, **a**, f_s were already defined:

```
F = @(f) mag(b,a,2*pi*f/fs) - 1/sqrt(2).
```

For each filter, calculate the exact and approximate 3-dB frequencies in Hz and the corresponding percentage error (<u>5 points</u>), and make a table (<u>2.5 points</u>) like the one shown below (for the case of filter #2):

```
       exact   approx
--------------------
fL = 3.8409   3.8408
fR = 4.1591   4.1592
-----------------------
percent error = 0.0011%
```

The percent error may be defined in terms of the L_1 norms:

$$\text{error} = 100\,\frac{|f_L - f_{La}| + |f_R - f_{Ra}|}{|f_L| + |f_R|}.$$

Moreover, add the exact 3-dB frequency points to the graphs of Problem 2.1 at the 3-dB level (<u>5 points</u>) (see Fig. 2).

Problem 2.3 (27.5 points)**.** In this part we study the time properties of the filters. In order to observe the notching behavior as well as the transients of the filters, we construct an input signal consisting of a 4-sec segment of the sinusoid f_0, sandwiched between 4-sec segments of two sinusoids of frequencies $f_1 = 2$ Hz, and $f_2 = 6$ Hz. Because f_1, f_2 lie in the passband of the filters, they will pass through, whereas the middle f_0 segment will be notched out after the transients die out.

First, add the two frequency points f_1, f_2 to the magnitude response graphs of Problem 2.1 in Fig. 2 (<u>2.5 points</u>). Then construct the following signal of duration of 12 sec:

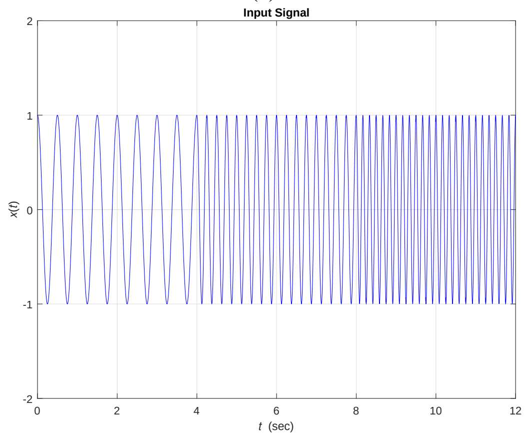
$$x(t) = \begin{cases} \cos(2\pi f_1 t), & 0 \le t < 4 \text{ sec}, \\ \cos(2\pi f_0 t), & 4 \le t < 8 \text{ sec}, \\ \cos(2\pi f_2 t), & 8 \le t \le 12 \text{ sec}. \end{cases}$$

and sample it a rate of f_s, that is, replace t by $t_n = nT_s$, $n = 0, 1, 2\ldots$, where $T_s = 1/f_s$ (<u>5 points</u>). Plot $x(t)$ versus $0 \le t \le 12$ sec (<u>2.5 points</u>).

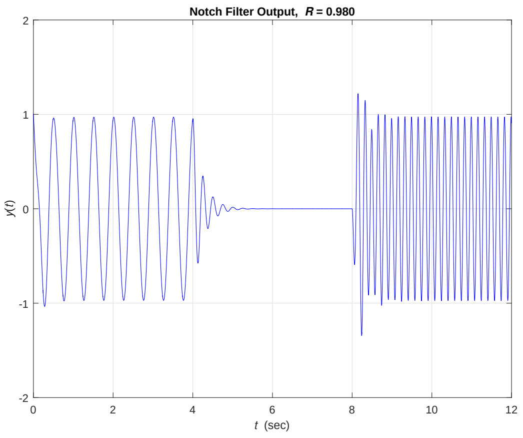
Then, using your function **tran** of Problem 1.1, filter this signal through both notch filters (<u>5 points</u>) and, on two separate graphs, plot the corresponding output signals (<u>2.5 points</u>). Moreover, calculate the effective time constants given in (6) and discuss whether they are consistent with what you observe in your output graphs (<u>7.5 points</u>). Make a table of calculated values as shown below (<u>2.5 points</u>):

```
  R        t_eff
--------------------
0.980   *.**** (sec)
0.995   *.**** (sec)
```

(a)



(b)



(c)

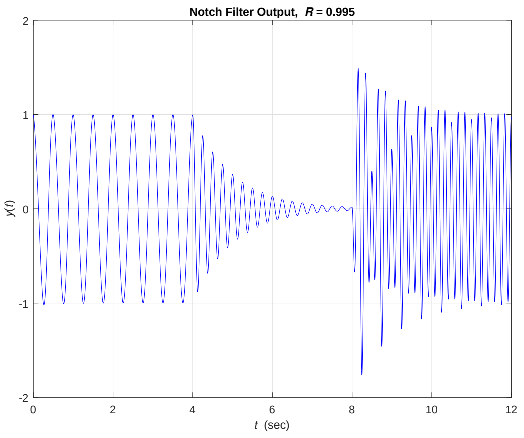


Fig. 3: Plots for Problem 2.3.