EE 473 HW 5 (Fall 2019)

1) Homework is due **December 3, Tuesday!**
2) This one will be longer than usual, so start early. Accordingly, grading will be out of 300; i.e., it will count as three homeworks.
3) Be neat and well-organized with your submission and coding. Sloppy homeworks, including hand-written ones, will be rewarded with a 50-point deduction.
4) Do NOT exceed 6 pages. Additional pages will not be graded, or even looked at.

1: *Reading.* The week of Nov. 19 we covered Chapter 6, Oppenheim and Schafer/2e, up to and including Section 6.5, as well as Section 7.2. Homework 5 corresponds to Sections 6.6 and 6.7.

2: *Elliptic filter.* The function `ellip` can be used to design discrete-time or continuous-time elliptic filters. Elliptic frequency-selective filters have equiripple in both passband and stopband. That is, the magnitude of the frequency response oscillates in the interval $[1-\delta_1, 1+\delta_1]$ in passband, and in $[0, \delta_2]$ in stopband.

(a) The command

```
>> [b,a] = ellip(4,.2,40,[.41 .47]);
```

returns the vectors `b` and `a` that are the transfer function coefficients of an eighth-order elliptic filter with 0.2 dB ripple in the passband $0.41\pi \leq |\omega| \leq 0.47\pi$, and has 40 dB attenuation in the stopband. Use `[H,w] = freqz(b,a,4096)` to compute 4096 samples of the frequency response of the filter for $0 \leq \omega \leq \pi$. Plot the magnitude of `H` (in dB) against `w`/π by executing

```
>> plot(w/π,20*log10(abs(H)));
>> axis([0 1 -80 10]);
```

(b) Use `axis` to zoom in on the passband of the filter and verify that it is equiripple in this region. Note that the filter returned by `ellip` is scaled such that the passband oscillates between 1 and $1-2\delta_1$. How could you change the filter coefficients to make the frequency response oscillate in the interval $[1-\Delta_1, 1+\Delta_1]$, for some $\Delta_1 > 0$, in the passband and still be equiripple in the stopband? Is $\Delta_1 = \delta_1$?

(c) Using `filter` compute 4096 samples of the impulse response of the original filter given by the coefficients in `b` and `a` above, and store the result in vector `h`. Plot the first 200 samples of the impulse response. What features of the impulse response indicate that this is a bandpass filter?

We see three distinct implementations of a discrete-time filter in Fig. 1. When the coefficients of a discrete-time filter is quantized, the resulting system is what we call a digital filter. Digital filters are generally implemented using fixed-point arithmetic on digital signal processing (DSP) chips (or consecutive memory registers in case of software). The coefficients of the digital filter must be quantized to the number of bits available on the DSP chip. The structures in Fig. 1 react differently when the coefficients are quantized, which is what you will investigate next.

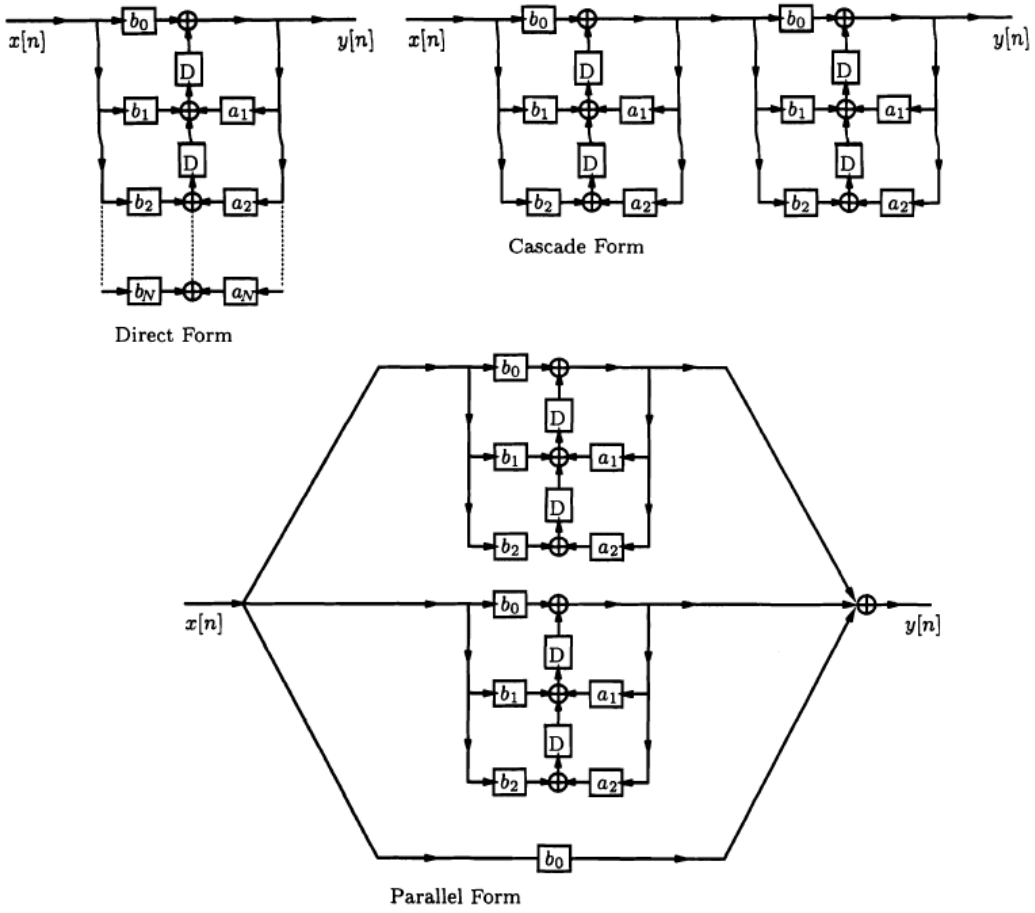

Figure 1. Three structures for implementing discrete-time filters. Note that the direct form in this figure is not identical to direct forms I and II described in Oppenheim and Schafer.

3: *Effect of quantization: Direct form.* The function `y=filter(b,a,x)` uses a transposed direct form II structure to implement the LTI causal system represented by the difference equation

`y(n)=b(1)*x(n)+b(2)*x(n-1)+...+b(nb+1)*x(n-nb)-a(2)*y(n-1)-...-a(na+1)*y(n-na)`

where `a(1)=1`, and `na+1` and `nb+1` are the lengths of the vectors `a` and `b`, respectively. When the difference equation is realized on hardware, the direct form can be sensitive to coefficient quantization. The function `quant(x,N,M)`, attached to this homework, will quantize the coefficients in the vector `x` to `N` bits, where `M` is the maximum possible amplitude of each element.

(a) Set `M=max(abs([b a]))` and use `quant` to quantize the coefficients in `b` and `a` to 16 bits, storing the results in `a16` and `b16`. What is the maximum amount any of the coefficients is changed by quantization? Use `freqz` to plot the frequency response magnitude of the quantized filter as in **2**(a) above, and zoom in on the passband as in **2**(b).

(b) Use the attached `dpzplot` function to generate and plot a pole-zero diagram for the filter described by the coefficients in `a16` and `b16`.

(c) Repeat parts **3**(a) and **3**(b) by quantizing `a` and `b` to 12 bits, and storing the results in `a12` and `b12`.

(d) Based on the pole-zero plots, are the filters whose frequency responses you have plotted both causal and stable? Why or why not?

(e) Use `filter` to generate and plot 4096 samples of the impulse response of the filter described by `a12` and `b12`. Is the filter whose impulse response you have plotted stable? Why or why not?

4: *Effect of quantization: Cascade form.* Next, you will examine the sensitivity of a cascade of four second-order direct form subsections to coefficient quantization.

(a) The function `[bc,ac]=df2cf(b,a)`, which is attached to this homework, transforms the filter described in parts **2**(a) and **2**(b) into a cascade of second-order subsections. Each row of `ac` and `bc` contains the coefficients of one of the second-order subsections. Use `df2cf` with the original unquantized coefficients in `b` and `a` to create the cascade system parameters and store them in `bc` and `ac`, respectively.

(b) Use successive calls to `filter` with each row of `ac` and `bc` to generate 4096 samples of the impulse response of the cascade system, and store the result in `hc`. Plot the first 200 samples of `hc` and compare this result to the plot of `h` to verify that you have implemented the filter correctly. If you have, then `max(hc-h)` should be roughly `3e-13`.

(c) Use `quant` to quantize the coefficients in `bc` and `ac` to 16 bits and store the results in `bcq16` and `acq16`. Repeat part **4**(b) to generate 4096 samples of the impulse response of the quantized cascade system and store the result in `hc16`.

(d) Use `freqz(hc16,1,4096)` to generate the plots of the frequency response magnitude in dB for the cascade system and zoom in on the passband as you did in **2**(b) above. Use also `dpzplot` to make a pole-zero plot for each of the second-order subsections described by the rows of `bcq16` and `acq16`. Are each of the causal second-order subsections stable? How do the poles and zeros of the second-order subsections correspond to the poles and zeros of the overall cascade system? How does the magnitude response of the 16-bit quantized cascade filter compare to that of the 16-bit quantized direct form implementation?

(e) Repeat parts **4**(c) and **4**(d) by quantizing `bc` and `ac` to 12 bits.

5: *Effect of quantization: Parallel form.* Finally you will consider a parallel implementation of the filter using a combination of second-order subsections.

(a) Use the function `[r,p,k]=residue(b,a)` to factor the system function into its partial fraction expansion (PFE). Beware that `residue` carries out the expansion as

$$\frac{b_M z^M + \ldots + b_1 z + b_0}{a_N z^N + \ldots + a_1 z + a_0} = \sum_{n=1}^{N} \frac{r_n}{(z-p_n)^{d_n}} + \sum_{m=0}^{M-N} k_m z^m,$$

where some of he roots p_n may be repeated. The vector `k` returned by `residue` contains the coefficients of the second sum, k_m, and it is a null vector if $M \leq N$. Using `residue` on polynomials in z^{-1} requires the coefficients of the filter to be in the reverse of the order required by `filter`. In order to implement the filter with a parallel combination of second-order subsections, you will need to combine pairs of the first-order sections from the PFE. To simplify computation, each complex pole should be combined with its complex conjugate so that the

second-order subsections will have real-valued coefficients. Select four pairs of first-order terms and use `residue` to recombine these first-order terms into second-order subsections. If there is any `k` term in the PDE corresponding to a term of the form $k\delta[n]$ in the impulse response, be sure to include `k` in one of the calls to `residue`. Store the coefficients of the resulting second-order subsections in the matrices `bp` and `ap`, with each row corresponding to one second-order system.

(b) Use repeated calls to `filter` to implement the parallel system and generate 4096 samples of the impulse response of the system, and store the result in `hp`. Plot the first 200 samples of `hp` and compare this result to the plot of `h` to verify that you have implemented the filter correctly. If you have, then `max(hp-h)` should be roughly `2e-13`.

(c) Use `quant` to quantize the coefficients in `bp` and `ap` to 16 bits, and store the results in `bpq16` and `apq16`. Use `residue` on each column of `bpq16` and `apq16` to obtain the PFE of the filter that was quantized in parallel form. Now with a single call to `residue`, recombine the terms to obtain the single set of coefficients, `bp16` and `ap16`, for the overall system function. Note that although the system function is given as a single difference equation, the coefficients were quantized with the system in parallel form.

(d) Use repeated calls to `filter` to simulate the filter in quantized parallel form and generate 4096 samples of the impulse response, storing the result in `hp16`. Use `[H,w]=freqz(hp16,1,4096)` to generate plots of the frequency response magnitude in dB. In addition, use `freqz(bp16,ap16,4096)` to generate the plots of the frequency response magnitude to verify that the system function described by `bp16` and `ap16` is indeed the same as that of the quantized parallel form system.

(e) Use `dpzplot` to make a pole-zero plot for the filter described by `bp16` and `ap16`. Is this filter both causal and stable? How does the magnitude response of the 16-bit quantized parallel filter compare to that of the 16-bit quantized cascade and direct form implementations?

(f) Repeat parts **5**(c), **5**(d) and **5**(e) by quantizing `bp` and `ap` to 12 bits.