Carnegie Mellon DSP Electrical & Computer ENGINEERING

Digital Signal Processing (18-491/18-691)
Spring Semester, 2024

Problem Set 9

Issued: 4/14/24

Due: 4/21/24 at midnight on Gradescope

Note: This is the last problem set of the semester (!).

Reminder: Quiz 3 (the final exam) will be Monday, May 6, from 8:30 to 11:30 am :-(, in SH 234 (our home!). It will be comprehensive in nature, covering through the material on this problem set, with a somewhat greater emphasis on topics since the last exam (*i.e.* IIR design, FIR design). The exam will be in a similar format as Quiz 2, and will be closed book with three sheets of notes permitted. **Note:** You can use the arithmetic and transcendental functions on your calculator or telephone, but your phone must be in airplane mode. Credit will not be granted for numerical calculations unless all the work is shown.

Reading: During the past two weeks, we have been discussing the three most important approaches to FIR filter design: windowing, frequency sampling, and the optimum equiripple approach known as the Parks-McClellan algorithm. Our discussions on FIR filter design are covered in Secs. 7.5 through 7.9 in the text. The frequency sampling FIR design method is described in Sec. 5.6.1 of Oppenheim and Schafer (1975), which is reprinted on the course website.

Problem 9.1:

In Problem 8.3 you implemented a highpass filter using the lowpass-to-highpass bilinear transformation. As you will recall, the overall system is intended to have the following specifications (expressed in *continuous time*):

- Passband frequencies: 5 to 8 kHz
- Stopband frequencies: 0 to 4.5 kHz
- Passband ripple: -1 dB $\leq |H(j\Omega)| \leq 0$ dB, $\Omega_p \leq |\Omega| \leq 8$ kHz
- Stopband ripple: $|H(j\Omega)| \leq -60$ dB, $|\Omega| \leq \Omega_s$

Assume that the sampling frequency remains 16 kHz.

In this problem we would like to implement the same filter but using an FIR implementation with one of the "classic" window designs with performance summarized in OSYP Table 7.2. Let the magnitude of the desired frequency response for the filter be

$$H_d(e^{j\omega}) = \begin{cases} 1, & \frac{\omega_s+\omega_p}{2} < |\omega| \leq \pi \\ 0, & 0 \leq |\omega| \leq \frac{\omega_s+\omega_p}{2} \end{cases}$$

where ω_s and ω_p are the stopband and passband cutoff frequencies, respectively.

(a) What is the ideal unit sample response $h_d[n]$? (Don't forget that we want the filter to be causal and linear phase.)

(b) Which of the five windows of Table 7.2 would be the best to use? Be sure to specify the window type and the window length parameter M.

(c) Using your answers from parts (a) and (b), write an expression for $h[n]$, the unit sample response of the filter.

(d) Using MATLAB 's `freqz` or some similar command, verify that the specifications are met. What are the actual passband, stopband, and ripple specifications of the filter you hsve designed?

Problem 9.2:

In Problem 9.1 you implemented the highpass filter from last week using an FIR implementation with one of the classic window designs with performance summarized in OSYP Table 7.2. In this problem we will repeat the window design process for a filter of the same specifications, but using a Kaiser window instead of one of the "classic" windows.

1. Using the design equations that are found in OSYP Sec. 7.5.3, determine the minimum length $N = M + 1$ of the impulse response and the value of the Kaiser window parameter β for a filter that meets the specifications.
2. What is the delay of the filter?
3. Compare the order of the filter designed using the Kaiser window with that of the filter designed using the bilinear transform as in Problem 8.3 and with that of the order of the filter that you designed in Problem 9.1.

Problem 9.3: Problem 7.31 in OSYP, parts (a) through (g) only.

Problem 9.4: Problem 7.36 in OSYP

Problem 9.5: Problem 7.63 in OSYP, parts (a) through (e) only

MATLAB Problems

Reminder: As we noted in the last assignment, the signal processing package in MATLAB itself contains a number of functions that facilitate the design of IIR digital filters including the following:

- Functions that perform complete filter design from discrete-time specifications: `butter, cheby1, cheby2, ellip`
- Functions that estimate the order of the prototype continuous-time filter that will be needed or realize a particular set of specifications: `buttord, cheb1ord, cheb2ord, ellipord`
- Functions that enable you to design "by hand" the continuous-time prototype filters: `buttap, cheb1ap, cheb2ap, ellipap`
- Functions that convert continuous-time filters into discrete-time filters: `bilinear, impinvar`
- Functions that perform frequency transformations: `lp2lp, lp2hp, lp2bp, lp2bs`
- The design aids `filterDesigner` and `fvtool` that display responses and parameters for IIR and FIR filters. (You can use these tools to confirm your design, but you should design the filters by hand according to the procedures we discussed in class.)

Keep in mind that the MATLAB routines `poly` and `roots` can be extremely helpful in multiplying and factoring polynomials. You are strongly encouraged to look over the `help` files for these functions to see how they work. In working the MATLAB problems, turn in a printout of your results, a copy of the MATLAB source code you developed to work the problem, as well as any additional comments you'd like to add.

In addition to the IIR routines, the MATLAB signal processing package also contains a number of functions that facilitate the design of FIR digital filters including the following ....

- Functions that enable you to design FIR filters using the window method: `fir1, fir2`
- Functions that realize particular window shapes: `bartlett, blackman, boxcar, chebwin, hamming, hanning, kaiser, triang`
- A function that implements the Parks-McClellan algorithm: `firpm`
- Functions that perform frequency transformations: `lp2lp, lp2hp, lp2bp, lp2bs`

- Complete signal processing design environments: `filterDesigner`

You are encouraged to look over the `help` files of these functions to see how they work.

FIR filter design is, of course, covered in Chapter 7 of OSYP. OSYP also describes the design of ideal differentiators and phase shifters (which are also called Hilbert transformers), which we do not discuss in much detail in the lectures.

Please remember: The notational conventions used by OSYP are somewhat different from those used by MATLAB . In discussing FIR design, OSYP uses the constant M to indicate the order of the polynomial used to generate the filter, which is also the maximum number of delays. As you will recall, $M = N - 1$, where N is the total "length" of the FIR filter's unit sample response. MATLAB itself uses the variable M to indicate the total length of the filter, denoted by N in OSYP. Finally please note that (lamentably), the length parameter used in MATLAB routines is defined inconsistently. For example, the length parameter used in MATLAB 's function `firpm` refers to OSYPs M, while the length parameter used in MATLAB 's function `kaiser` refers to OSYP's N. So, please do be careful, read the `help` files, and check what you're doing!

Please also note: It has been mentioned in passing in class that the total length of an FIR filter (*i.e.* OSYP's N) must be odd if the filter is a highpass filter or a bandstop filter. (Why?) Please keep this in mind in developing your solutions to problems.

Problem C9.1:

Using the MATLAB routine `firpm`, implement a filter that has approximates the following desired response:

$$H_d(e^{j\omega}) = \begin{cases} 1, & 0 \le |\omega| < 0.4\pi \\ 2, & 0.45\pi \le |\omega| < 0.55\pi \\ 3, & 0.6\pi \le |\omega| < 0.7\pi \\ 1, & 0.8\pi \le |\omega| \le \pi \end{cases}$$

The filter response is unconstrained for those frequencies outside the bands specified above. The filter specifications call for the ripple parameter δ to be equal to ± 0.05 dB in all four bands about their nominal values.

Determine the minimum order of the filter needed to realize these specifications. Turn in a print-out of the filter coefficients, and a plot of the magnitude of the filter's frequency response in decibels.

Problem C9.2: In this problem we will design and implement a simple lowpass filter using several of the techniques we have talked about during the course of this class. Please use the MATLAB routines specified in each part of this problem rather than the filter design tool `filterDesigner` for now.

- Passband cutoff frequency: $\omega_p = 0.6\pi$
- Stopband cutoff frequency: $\omega_s = 0.65\pi$
- Maximum passband ripple: $1 - \delta_1 \geq -1$ dB
- Minimum stopband attenuation: $\delta_2 \leq -75$ dB

(a) Find the *order* of the filter (*i.e.* the number of delays) needed to realize these specifications using the following filter design approaches. (You do **not** need to develop the whole filter design.)

1. FIR design using a carefully-selected Kaiser window. (Use the same procedure that you used in working Problem 9.2.)
2. FIR design using the Parks-McClellan algorithm. (Use `firpm` as you did in Problem C9.1.)
3. IIR design using the bilinear transform in conjunction with an elliptical prototype filter. (Use the appropriate frequency warping functions and the MATLAB routine `ellipord`. See also the help file for `ellip`.)

(b) Design each of the above filters using the appropriate MATLAB routines. Briefly describe how you did your implementations.

For each design, turn in a printout of the numerator and denominator coefficients of the filters system function, and a plot of the magnitude of the frequency response in decibels.

(c) We would now like to compare the total computational efficiency of "efficient" implementations of each of the designs. Assume that you have realized the filters using the following implementations:

1. The FIR design using windowing techniques is realized using the "linear-phase" implementation (with half the number of delays and multiplications compared to the direct form).
2. The FIR design using the Parks McClellan algorithm is also realized using the standard linear-phase implementation.
3. The IIR design using the lowpass-to-highpass bilinear transform and an elliptical prototype filter is realized using the Direct Form II implementation.

(d) Using each of the above implementations, what is the total number of multiplications required when the filter is used to process an arbitrarily long input sequence directly in the time domain? Express your answers in terms of multiplications *per sample of input.*

1. Estimate the number of multiplications per sample of input that would be required if you were implement the convolution of a long input sequence with the FIR filters using DFTs and the overlap-add method. Assume that you are using the equiripple filter design provided by the Parks-McClellan algorithm and a DFT size of 2048 points.

2. Compare this result with the number of multiplications that you would incur implementing the most efficient FIR filter completely in the time domain (presumably the equiripple FIR design) with the IIR filter in the time domain.

Problem C9.3:

Re-design the FIR and IIR filters with specifications from Problem C9.2, but using the MATLAB tool `filterDesigner`. Include the FIR window design method using Kaiser windows, the FIR equiripple method, and the IIR method using bilinear transformation with an elliptic prototype. Don't forget to use the normalized frequency scale (0 to 1).

(a) What is the order of each of the filters obtained using MATLAB ?

(b) Are there any major differences between the frequency responses obtained using `filterDesigner` and the "manual" design approaches considered in Problem C9.2?

Problem C9.4:

In this problem we will implement a simple FIR filter using the *frequency sampling* method. We will design a simple lowpass filter of size $N = 63$ and compare its performance to a equipripple filters with similar specifications.

(a) Consider a naively-designed lowpass filter with DFT coefficients $H[k]$:

$$H[k] = \begin{cases} 1, & 0 \leq k \leq 7 \text{ and } 56 \leq k \leq 62 \\ 0, & \text{otherwise} \end{cases}$$

Note that this is a symmetric Type I lowpass filter, and that the frequency response is real and even. The discrete-time frequencies corresponding to the coefficient values are $2\pi k/63$.

1. What is the discrete-time frequency that corresponds to the nominal cutoff frequency of this lowpass filter? Remember that the nonzero passband DFT coefficients are between 0 and 9, that the filter has even symmetry, and that the DFT coefficients are implicitly periodic with period N.

2. What is the minimum stopband attenuation in dB observed for this filter when implemented in its original form? **Hint:** You are encouraged to use code very similar to the MATLAB routine `FreqSampDemo2.m` that was part of the Lecture 24 on April 10 and is included on the Lectures page of the course website to obtain solutions for this and subsequent parts of the problem.

(b) Now choose other values for the samples representing $n = 8$ and $n = 55$ (which should be the edges of the original passband of the filter, and would occur with MATLAB 's origin 1 at indices 10 and 55) between 0 and 1.

1. What is the value of samples $H[8]$ and $H[55]$ that provides the greatest minimum stopband attenuation?

2. Using your value of samples $H[8]$ and $H[55]$ that provides the greatest minimum stopband attenuation, determine for the resulting filter using `freqz` or any other convenient approach the passband and stop band edge frequencies (ω_p and ω_s) and the passband and stopband ripple (δ_p and δ_s) expressed in decibels.

(c) Using the MATLAB routine `filterDesigner`, determine the length of the filter designed using Kaiser windows that acchieves the specifications ω_p, ω_s, δ_p, and δ_s.

(d) Using the MATLAB routine `filterDesigner`, determine the length of the filter designed using the Parks-McClellan algorithm that acchieves the specifications ω_p, ω_s, δ_p, and δ_s.

(e) Obtain the number of multiplys *per output point* that would be obtained when convolving a long input sequence with the FIR filter $h[n]$ using the following design and implementation approaches:

1. Direct convolution using your frequency-sampled FIR filter of length $N = 63$, implemented using the direct form.

2. Direct convolution using your frequency-sampled design, and using the frequency-sampled implementation method you developed in Problem 7.5.

3. Direct convolution using the equivalent FIR filter designed using the Parks-McClellan algorithm, implemented in the linear phase form (which results in a further reduction of multiplys by a factor of 2).

4. Convolution using the equivalent FIR filter designed using the Parks-McClellan algorithm implemented using the overlap-save algorithm with DFTs of size 2048. Assume about 30,000 input samples.