Homework Set # 6

Problem 1

In this problem we will implement an ideal low-pass filter using the FIR model. Consider the ideal low-pass filter with cut-off frequency $\omega = \frac{\pi}{3}$,

$$H(e^{i\omega}) = \begin{cases} 1 & |\omega| < \frac{\pi}{3} \\ 0 & \text{else.} \end{cases}$$

(a) What is the impulse response of this filter, $h[n]$?

(b) What is the best FIR approximation of length $2K+1$ for $H(e^{j\omega})$? That is an FIR filter $h_K[n]$ for which $h_K[n] = 0$ for $n > K$, and minimizes $\|H(e^{j\omega}) - H_K(e^{j\omega})\|_2$.

(c) Write a Matlab m-file to produce a vector of length $2K+1$ of coefficients of the filter $h_K[n]$, and then take the inverse Fourier transform of the produced vector.

(d) Plot the frequency response of your filter for $K = 10, 50, 100, 1000$. How well is the approximation in comparison to the ideal filter? What happens at the cut-off frequency? How can you explain this phenomenon?

Problem 2 (Weighted Least-Squares Filter Design)

So far you have studied two kinds of filter design in class: Windowing, which minimizes the mean squared error between ideal and actual frequency response, and Minimax filter design, which minimizes the maximum error in select frequency bands.

The goal of this problem is to design a filter that minimizes a weighted mean square error with respect to an ideal highpass filter in the passband and the stopband, with no requirements for the transition band. Just like in the Minimax case, we want the filter to satisfy certain tolerances.

In the following you will design a Type 1 filter $h[n]$ according to the specifications in Figure 1. The passband is $[\omega_p, \pi]$, the stopband is $[0, \omega_s]$, and the transition band is (ω_s, ω_p). The tolerances are, respectively, δ_p for the passband and δ_s for the stopband.

Recall from class that the centered impulse response $h_d[n]$ of a Type 1 filter satisfies

$$H_d(e^{j\omega}) = h_d[0] + 2 \sum_{n=1}^{(M-1)/2} h_d[n] \cos(n\omega),$$

where M is the length of the filter (number of taps). The error function $E(\omega)$ is defined as

$$E(\omega) = W(\omega)[D(\omega) - H_d(e^{j\omega})],$$

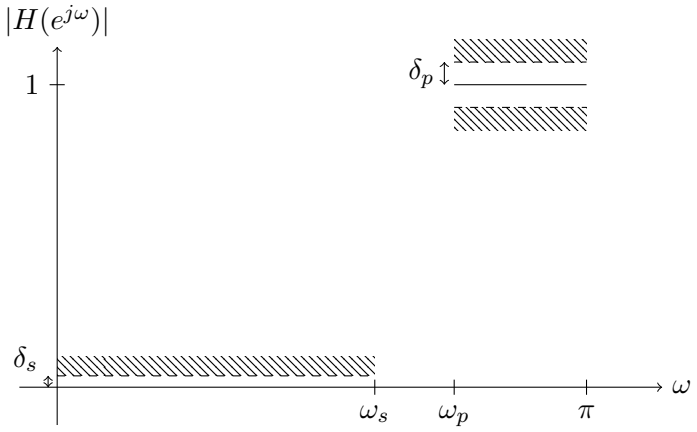

Figure 1: Filter specification for Problem 2.

where

$$D(\omega) = \begin{cases} 0 & \omega \in [0, \omega_s] \\ 1 & \omega \in [\omega_p, \pi] \end{cases}$$

is the desired (ideal) highpass filter frequency response and

$$W(\omega) = \begin{cases} \delta_p/\delta_s & \omega \in [0, \omega_s] \\ 1 & \omega \in [\omega_p, \pi] \\ 0 & \text{otherwise} \end{cases}$$

is the weighting function. We want to find the filter coefficients that minimize the squared error, *i.e.*,

$$\{h_d[n]\}_{n=0}^{(M-1)/2} = \arg\min_{h_d[n]} \left\{ \int_0^\pi |E(\omega)|^2 \mathrm{d}\omega \right\},$$

subject to

$$|E(\omega)| \leq \delta_p \quad \text{for all } \omega.$$

This filter design method is called *weighted least squares* or WLS.

For a discrete set of frequencies the objective function is approximated by

$$\int_0^\pi |E(\omega)|^2 \mathrm{d}\omega \approx \frac{1}{K} \sum_{k=1}^{K} |E(\omega_k)|^2. \tag{1}$$

(a) Let $\mathbf{e} = [E(\omega_1), \cdots, E(\omega_K)]^{\mathrm{T}}$. Then $\sum_{k=1}^{K} |E(\omega_k)|^2 = \mathbf{e}^{\mathrm{T}}\mathbf{e}$. Find the matrices W and U and the vector $\mathbf{d}$ such that we can write

$$\mathbf{e} = W(\mathbf{d} - U\mathbf{h}),$$

where $\mathbf{h} = [h_d[0], h_d[1], \ldots, h_d[(M-1)/2]]^{\mathrm{T}}$ are the coefficients of the centered filter impulse response.

The filter design problem can now be formulated as

$$\min_{\mathbf{h}} \mathbf{e}^{\mathrm{T}}\mathbf{e}$$

subject to

$$|e_k| \leq \delta_p, \quad k = 1, \ldots, K. \tag{2}$$

This kind of problem is called a *nonlinear constrained optimization problem* and is standard in optimization theory.

A general nonlinear constrained optimization problem has the form

$$\begin{array}{ll} \text{minimize} & f(\mathbf{h}) \\ \text{s.t.} & A\mathbf{h} \leq \mathbf{b}, \end{array} \tag{3}$$

where $f(\cdot)$ is a scalar function of a vector argument and where the inequality must be satisfied for all components of the respective vectors. In our case, we have $f(\mathbf{h}) = \mathbf{e}^{\mathrm{T}}\mathbf{e}$.

(b) Find the matrix A and the vector $\mathbf{b}$ such that the constraint (2) can be written as $A\mathbf{h} \leq \mathbf{b}$. Express A and $\mathbf{b}$ in terms of W, U, and $\mathbf{d}$.

To solve a problem of the form (3), MATLAB provides the function `fmincon` (which is part of the *optimization toolbox*). In its basic form, its syntax is

```
[x, fval] = fmincon(f, x0, A, b)
```

Here `f` is a *handle* to the objective function, `x0` is an initial guess for `x`, and `A` and `b` define the inequality constraint $A\mathbf{x} \leq \mathbf{b}$. `fmincon` has two return values: `x` is the argument that minimizes the objective function; `fval` is the function value at that point.

A handle to a function, or function handle, is a way to pass a reference to some function to another function. For example, suppose that you have written a MATLAB function `myfun` that takes a vector `x` and returns the value of the objective function for `x`. Then you would call `fmincon` as

```
fmincon(@myfun, ...)
```

Thus, to create a handle to a function you simply precede the function's name with the `@` sign.

(c) Download the file `wls_skeleton.m` from the course webpage and complete it. Use the values for ω_p, ω_s, etc. defined in the file. Provide a plot of $|H(e^{j\omega})|$ as well as a stem plot of $h[n]$. Also hand in a printout of your Matlab-file `wls.m`.

Problem 3

The aim of this exercise is to find a secret message (different for each student), that is hidden in a music file. Most of the work is done in Matlab, but you will also have to do some parts on paper. In most cases, the Matlab functions that you should use are indicated in *italic* letters. To find out how to use a Matlab function, type

```
>>help function_name
```

(a) Download the file "wave-files.zip" from the course webpage (section "Additional Material"). Unzip it and locate the wave-file intended for you. There is a different wave-file for every student. For instance, the file for Suhas Diggavi would be called `Diggavi_song.wav`. Load the wave-file into your Matlab environment, and listen to the music.

```
>>[song,fs]=wavread('Diggavi_song.wav');
>>soundsc(song,fs);
```

The variable "song" contains now a vector of length $N = 390600$. Every entry of that vector is one sample of the piece of music. Next, use the function *fft* to compute the DFT of the song. Use the functions *abs* and *plot* to plot the magnitude of the DFT. Imagine that the DFT is a sampled version of the DTFT. In that case, the horizontal axis of your plot shows the values of the frequency ω. Print the plot and mark the points 0, π and $-\pi$ on the ω-axis.

(b) A secret message has been hidden in the high frequency range of the music. Can you see it in your frequency-plot? What kind of filter would you need to separate the secret message from the music?

(c) Use the file *wls* from the previous problem to construct such a filter. Do you have to modify the values of ω_s and ω_p in the top section of "wls.m"?

```
>>h=wls();
```

(d) Apply the resulting filter using the function *conv*. (Note that the filter h is much shorter than the music vector "song". Is that a problem?) Let "songFilt" be the filter output. Take the DFT of "songFilt" and plot the magnitude of the frequency representation. Check that the filter did what you wanted.

(e) We tell you now how the message was hidden. A voice signal $m[n]$ has been shifted to very high frequencies using

$$m'[n] = m[n]\cos(0.8\pi n).$$

To understand exactly the effect of this operation, compute (analytically), the DTFT $M'(e^{j\omega})$ of $m'[n]$ as a function of the DTFT $M(e^{j\omega})$ of $m[n]$. In matlab, we actually deal with the DFT. Assuming that the voice message $m[n]$ was of length $N = 390600$, explain the effect of the operation $m'[n] = m[n]\cos(0.8\pi n)$ on the DFT of $m[n]$ in terms of discrete sequences.

(f) Propose an operation that would let us recover $m[n]$ from $m'[n]$. Apply this operation in Matlab to the sequence "songFilt". Call the outcome "songFiltShift". (Hint: use the Matlab function *cos*.) Take the DFT of "songFiltShift" and plots its magnitude. Are you satisfied with the result? Do you observe anything strange?

(g) We know that most of the information in "songFiltShift" is the secret message. Listen to this file.

```
>>soundsc(songFiltShift,fs);
```

What is the secret message?

(h) If you have trouble understanding the secret message, it is probably because of the high-frequency noise that is also visible in the spectrum of "songFiltShift". Why is the noise there? (It was not there in the original message.) What kind of filter would you need to get rid of the noise?

(i) In Problem 1, you have learned how to design FIR filters using an optimization procedure. Matlab also provides much easier tools to design filters. One tool is the function *firpm*, which designs a FIR Parks-McClellan filter. In class, you will only see a brief overview of how this works. The Matlab function, however, is extremely easy to use. With the command

```
>>[hpm, err] = firpm(40, [0 0.6 0.75 1], [0 0 1 1], [10 1]);
```

you construct a high-pass filter that is very similar to the one constructed in Problem 1. The first argument of *firpm* specifies that the resulting filter should be of length 41. The second argument says that there is one frequency band between 0 and 0.6π and a second band between 0.75π and π. The third argument says that the filter frequency response should be close to 0 in the first frequency band and close to 1 in the second frequency band. Between the two bands, we do not care. The fourth argument says that the precision in the first frequency band is 10 times more important than in the second frequency band.

Use the function *firpm* to construct a filter that gets rid of the noise in "songFiltShift". The return value "err" gives you the error (how far the filter is from the ideal filter). You can play a bit with the filter-length parameter to find a good filter.

(j) Apply your filter to remove some noise from "songFiltShift". Call the result "message". Plot the magnitude of the DFT of "message". Are you happy with your result? You can vary the boundaries of the filter bands in *firpm* to try and improve the filter. Play message using

```
>>soundsc(message,fs);
```

What is the message?