San Francisco State University
Engineering 451

Laboratory #6 – FIR Filters II

6.1 Outline

In the last lab, we investigated the some window-based FIR filters and applied the Kaiser filter to the task of separating the row and column DTMF tones. The purpose of this lab is to investigate spline, frequency sampled and optimum FIR filters and compare their properties with those of the window-based filters we looked at in the last lab. Then we apply them to the task of separating DTMF tones.

6.2 Background

Window-based filters

The window-based rectangular, Hamming and Kaiser filters we studied in the previous lab were all based on the principle of multiplying the response of the ideal lowpass filter,

$$h_{ideal}[n] = \frac{\omega_c}{\pi}\operatorname{sinc}\omega_c n\ ,$$

by a window function, $w[n]$, that shapes the response so that the resulting impulse response of the FIR filter is

$$h[n] = h_{ideal}[n]\cdot w[n]\ .$$

The simplest case is that of the rectangular-window filter, in which

$$w[n] = \operatorname{rect}_N[n] = 1, \qquad |n| < (N-1)/2$$

is a rectangular pulse of width, *N*, that essentially simply truncates $h_{ideal}[n]$ symmetrically. The Hamming window is one of a family of raised cosine windows, and has the equation

$$w[n] = \left(0.54 + 0.46\cos\left(\frac{2\pi n}{N-1}n\right)\right)\operatorname{rect}_N(n), \quad |n| \le (N-1)/2\ .$$

For these two filters, we can control the size of the transition zone, $\Delta\omega$, by increasing the length of the filter, *N*. But the maximum ripple in the passband, δ_p and the minimum attenuation in the stopband, δ_s are fixed properties of the window and we can't control them.

For the Kaiser filter, we are able to specify $\Delta\omega$ and $\delta \triangleq \min(\delta_p, \delta_s)$. Then, the Kaiser window is given by a Bessel function,

$$w[n] = \frac{I_o\left(\beta\left(1-\left(\frac{2n}{N-1}\right)^2\right)^{\frac{1}{2}}\right)}{I_o(\beta)}, \qquad |n| \le \frac{N-1}{2}\ ,$$

The filter length, *N*, is

$$N = \left\lceil \frac{A-8}{2.285\Delta\omega} \right\rceil,$$

where

$$A = -20\log_{10}\delta$$

and β is a shape parameter that determines the ratio between the width of the main lobe and the minimum attenuation of the side-lobes:

$$\beta = \begin{cases} 0.1102(A-8.7), & A > 50 \\ 0.5842(A-21)^{0.4} + 0.07886(A-21), & 21 \le A \le 50 \\ 0, & A < 21 \end{cases}.$$

Matlab implementation of window-based filters

In this lab, you don't have to write your own implementations of the window-based filters. You can use Matlab's `fir1` function if you like. The syntax is

Rectangular:	`h = fir1(N-1, wc, rectwin(N));`
Hamming:	`h = fir1(N-1, wc, hamming(N));`
Kaiser:	`h = fir1(N-1, wc, kaiser(N, beta));`

Note that `wc` must be normalized by π, so it ranges from 0 to 1.

Spline filters

Window-based filters start with the impulse response of an ideal lowpass filter and then attempt to modify this response to reduce the problems that arise from the infinitely narrow transition zone. Spline filters address the problem of the transition zone in another way. Here, we base the design of a realizable filter on a non-ideal filter that has a transition zone of finite width. Figure 6-1 shows how this is done. Figure 6-1a shows the frequency response of the ideal lowpass filter, $H_0(\omega)$, with a cutoff frequency, ω_c. We start by specifying a transition zone of width, $\Delta\omega$, centered on ω_c.

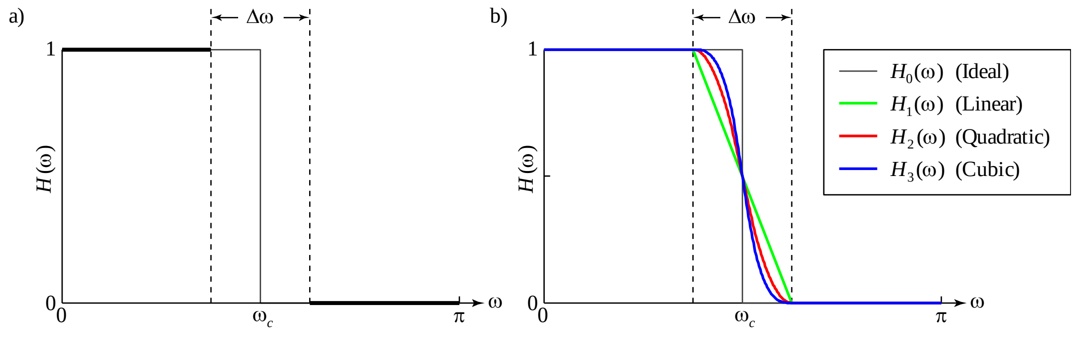

Figure 6-1: Spline filters

Figure 6-1b shows the frequency response of a number of filters created by connecting the right edge of the pass-band of the ideal filter (i.e., $H_0(\omega_c - \Delta\omega/2) = 1$) to the left edge of the stop-band (i.e., $H_0(\omega_c + \Delta\omega/2) = 0$) with *splines* (polynomials in ω) of different orders. The simplest spline is a straight line, resulting in a filter with a linear transition zone, $H_1(\omega)$, shown in green. Figure 6-1b also shows filters with frequency responses $H_2(\omega)$ and $H_3(\omega)$, whose transition zones are second-order and third-order splines respectively. However, in this lab, we will only deal with first-order splines.

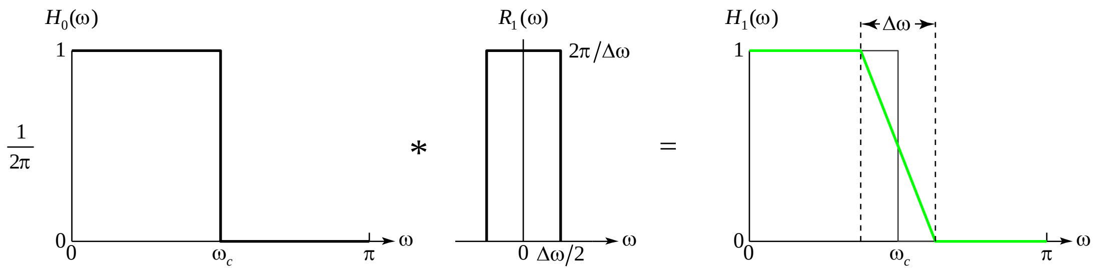

Figure 6-2: Construction of first-order spline filter

Figure 6-2 shows a simple way of deriving the impulse response of a first-order spline filter, $h_1[n]$. First, we derive an expression for the frequency response, $H_1(\omega)$. To do so, we create a rectangular function, $R_1(\omega)$, of width, $\Delta\omega$, and height $2\pi/\Delta\omega$, so it has constant area, 2π:

$$R_1(\omega) = \begin{cases} \dfrac{2\pi}{\Delta\omega}, & -\Delta\omega/2 \le \omega \le \Delta\omega/2 \\ 0, & \text{otherwise} \end{cases}.$$

Then, $H_1(\omega)$ is the frequency-domain convolution of $H_0(\omega)$ with $R_1(\omega)$,

$$H_1(\omega) = \frac{1}{2\pi} H_0(\omega) * R_1(\omega).$$

The impulse response of this filter is just

$$h_1[n] = h_0[n] \cdot r_1[n]$$

where

$$h_0[n] = \frac{\omega_c}{\pi} \operatorname{sinc} \omega_c n$$

and

$$r_1[n] = \operatorname{sinc} \frac{\Delta\omega}{2} n.$$

Then,

$$h_1[n] = \left(\frac{\omega_c}{\pi} \operatorname{sinc} \omega_c n \right) \operatorname{sinc} \frac{\Delta\omega}{2} n.$$

This is just another window-based filter, where the window is a sinc function. Because $\operatorname{sinc} n$ falls off as $1/n$, it tapers the ideal impulse response, $h_0[n]$, dramatically.

Frequency sampling filter design

The frequency sampling method of designing a linear-phase FIR filter gives finer control of the shape of the frequency response, but there are (of course) tradeoffs to consider. This method produces filter whose frequency response, $H(\omega)$, matches a desired frequency response, $H_d(\omega)$, at selected points.

There are several ways of designing frequency sampled filters. In this lab, we will use a method based on solving simultaneous equations. This will allow us to control the selected frequencies very accurately. We'll just consider the case of a Type I (odd-length, symmetric) filter of length, N, and impulse response, $h[n]$. For this filter, the derivation in the chapter shows that amplitude response is given by

$$A(\omega) = \sum_{m=0}^{M} b[m]\cos m\omega \ ,$$

where $M = (N-1)/2$ and the sequence, $b[m]$, is related to the impulse response of the filter by the relation

$$b[m] = \begin{cases} h[(N-1)/2], \quad m = 0 \\ 2h[(N-1)/2-m], & 1 \le m \le (N-1)/2 \end{cases} .$$

To design the frequency sampled filter, we create a sequence, $A[k]$, which is just samples of $A(\omega)$ at $M+1$ frequencies, ω_k, that span the interval, $0 \le \omega < \pi$, not necessarily uniformly:

$$A[k] = A(\omega_k) = \sum_{m=0}^{M} b[m]\cos m\omega_k \ , 0 \le k \le M \ .$$

This results a set of $M+1$ linearly independent equations in $M+1$ unknowns that can be solved for $b[n]$. Formulated this a matrix problem, we have $\mathbf{A} = \mathbf{Cb}$,

$$\underbrace{\begin{bmatrix} A[0] \\ A[1] \\ \vdots \\ A[M-1] \\ A[M] \end{bmatrix}}_{\mathbf{A}} = \underbrace{\begin{bmatrix} 1 & \cos\omega_0 & \cos 2\omega_0 & \cdots & \cos M\omega_0 \\ 1 & \cos\omega_1 & \cos 2\omega_1 & \cdots & \cos M\omega_1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \cos\omega_M & \cos 2\omega_M & \cdots & \cos M\omega_M \end{bmatrix}}_{\mathbf{C}} \underbrace{\begin{bmatrix} b[0] \\ b[1] \\ \vdots \\ b[M] \end{bmatrix}}_{\mathbf{b}}$$

C is a square $(M+1)\times(M+1)$ matrix, whose inverse can be found by a number of standard methods (assuming it is well-conditioned), leading to a unique solution for the parameter vector, $\mathbf{b} = \mathbf{C}^{-1}\mathbf{A}$. Finally, all N values of $h[n]$ can be computed from $b[n]$ by "reversing" the equation for $b[n]$:

$$h[n] = \begin{cases} \tfrac{1}{2} b[M-n], & 0 \le n \le M-1 \\ b[0], & n = M \\ \tfrac{1}{2} b[n-M], & M+1 \le n \le 2M \end{cases} .$$

The frequencies samples of $A(\omega)$ can be either uniformly or nonuniformly distributed over the interval $0 \le \omega < \pi$. To sample uniformly, we would have we $\omega_k = k\omega_0$, where $\omega_0 \triangleq 2\pi/N$, so the equation for $A[k]$ becomes

$$A[k] = \sum_{m=0}^{M} b[m]\cos mk\omega_0 \ , 0 \le k \le M \ .$$

As an example, Figure 6-3 shows (in blue) the frequency response of frequency-selective lowpass filter of length $N = 17$ with cutoff, $\omega_c = 0.5\pi$, designed by sampling the ideal lowpass filter uniformly. There are $M+1 = (N+1)/2$ samples in the frequency range $0 \le \omega \le \pi$. Here is the code:

```
N = 17;
```

```
wc = 0.5*pi;
M = (N-1)/2;
k = 0:M;
w = 2*pi*k/N;
Ak = double(w<wc)';
C = cos(w'*k);
b = C \ Ak;
h = [0.5*b(end:-1:2); b(1); 0.5*b(2:end)];
```

Note that there are still fairly large oscillations in the pass- and stopbands due to the sharp transition at $\omega_c = 0.5\pi$. In red is shown the response of a frequency selective filter where the two frequency samples nearest the discontinuity at $\omega_c = \pi/2$ are moved away from their initial positions by a small amount ($\Delta\omega = \pi/40$). As a result, the peak amplitude of the ripples in the pass-band and stop-band are attenuated.

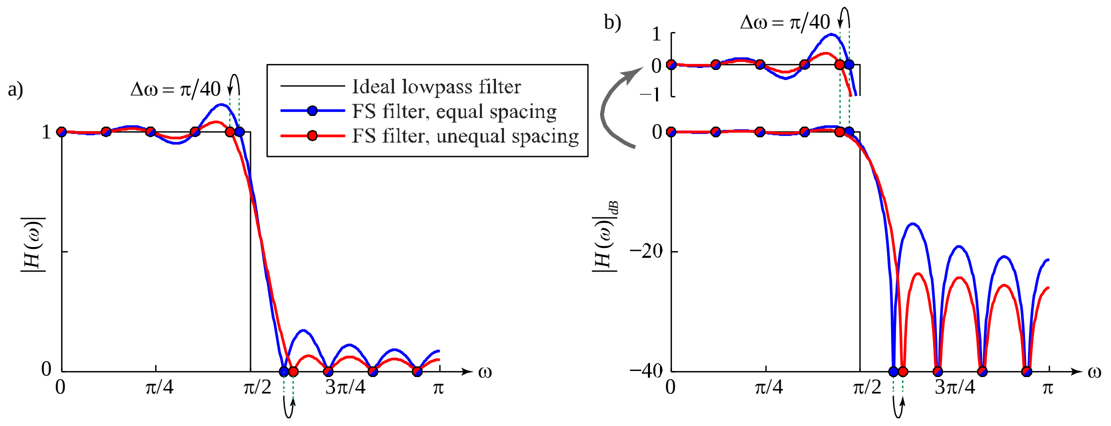

Figure 6-3: Frequency-sampled filter with equal and unequal frequency distributions

Optimum filter design

You'll have to read Section 5.8 of the notes to understand the theory of optimal filter design. But, for the purposes of this lab, we will just use the appropriate Matlab functions to design these filters. Design of optimal FIR filters with Matlab is a two-step process. Matlab implements the Parks-McClellan algorithm using a couple of functions: `firpmord` and `firpm`. Normally, you first use `firpmord`, which takes as input values of the band edges, magnitudes and ripple amplitudes, and outputs the filter order, as well as other parameters. The syntax is

```
[N,f0,A0,W] = firpmord(f,a,delta).
```

The input parameters are

`f` is a vector of normalized cutoff frequencies between 0 and half the sampling frequency, `fs`.
`a` is a vector that specifies the amplitudes corresponding to the bands defined by the cutoff frequencies.
`delta` is a vector of band tolerances corresponding to the bands defined by the cutoff frequencies.

The output parameters are

`N` is the estimated filter order;
`f0` is a vector of frequencies normalized between 0 (i.e., $\omega = 0$) and 1 (i.e., $\omega = \pi$) that correspond to the beginning and frequencies of each band of the input vector, `f`;
`A0` is a vector of amplitudes that correspond to the beginning and frequencies of each band of the input;
`W` is a vector of weights derived from the input vector, `delta`.

Having obtained values of `N,f0,A0,W` from `firpmord`, you then use `firpm` to generate the impulse response. The syntax is

```
    h = firpm(N,f0,A0,W)
```

where the input parameters are equivalent to the output parameters of `firpm`, and `h` is the impulse response of the resulting optimum filter.

Highpass filters

It turns out to be easier to hear the problems of the filters when you listen to a filter that passes the column tones and attempts to stop the row tones. Accordingly, we will need to make highpass filters as well as lowpass filters. Remember from the last lab that it is easy to make a highpass filter. First you design the appropriate lowpass filter with impulse response $h_{LP}[n]$, and then use the fact that

$$h_{HP} = \delta[n] - h_{LP}[n] .$$

In Matlab, that can be done with two simple lines:

```
hhp = -hlp;
hhp(M+1) = hhp(M+1) + 1;
```

6.3 Assignment

There are several parts to the assignment. The object of all parts of the lab is to compare the ability of various FIR filters to separate the row and column DTMF phone tones we used in the previous lab. Recall that the row tones occur at frequencies at 697, 770, 852 and 941 Hz. The frequencies of the column tones are 1209, 1339 and 1447 Hz. You can use your `magdb` function from the previous lab to make your plots if you wish.

Rectangular-window filter

Start by designing a rectangular-window lowpass filter of length $N = 73$. Set the corner frequency of the filter, ω_c, to be exactly midway between the frequencies of the highest row tone and the lowest column tone. Plot the frequency response of the filter over the frequency range $0 \le \omega \le \pi/2$ on a dB scale ranging from -60 dB to +10 dB. Recall that for a sampled data system with a sampling frequency, f_S, the mapping between the continuous-time frequency, f, and discrete-time frequency, ω, is given by

$$\omega = 2\pi \frac{f}{f_S} .$$

Your lowpass result should be something like that shown in Figure 6-4a and your highpass filter like that in Figure 6-b.

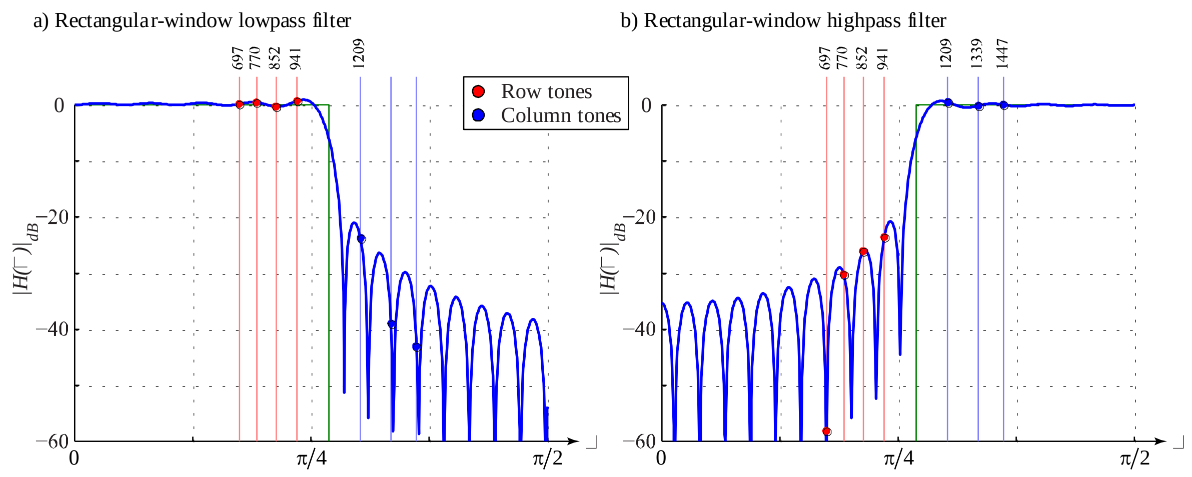

Figure 6-4: Rectangular-window filters

Also, plot on the curve seven dots that correspond to the frequencies of the phone tones, ω_{tones} . In Matlab-speak:

```
wtones = 2 * pi * [697 770 852 941 1209 1339 1447] / fs;
```

I've plotted the points corresponding to the row tones in red and the column tones in blue. I've also plotted a green line that corresponds to the frequency response of an ideal low-pass filter with a corner frequency of ω_c . As you can see, the SNR (signal-to-noise ratio), defined as the separation of highest frequency row tone and the lowest frequency column tone, is only about 20 dB, which isn't very good at all.

Download the .wav file of phone tones and listen to the sound files processed through this filter, particularly the highpass filter. You *want* the filter to pass only the column tones, but if you listen on a reasonable pair of headphones, you can clearly hear the row tones, which indicates that this isn't a very good filter.

Hamming-window filter

Now design Hamming-window lowpass and highpass filters of the same length, namely $N = 73$. The results should be equivalent to those shown in Figure 6-5.

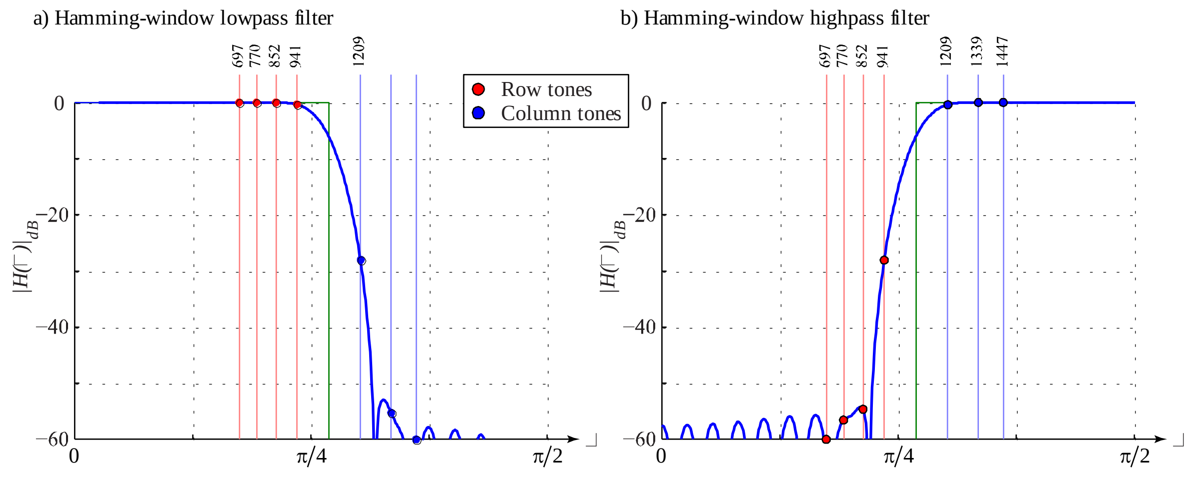

Figure 6-5: Hamming-window filters

Listen to the response of the highpass filter. You can clearly hear the row tone at 941 Hz leaking through on digits, * 0 #.

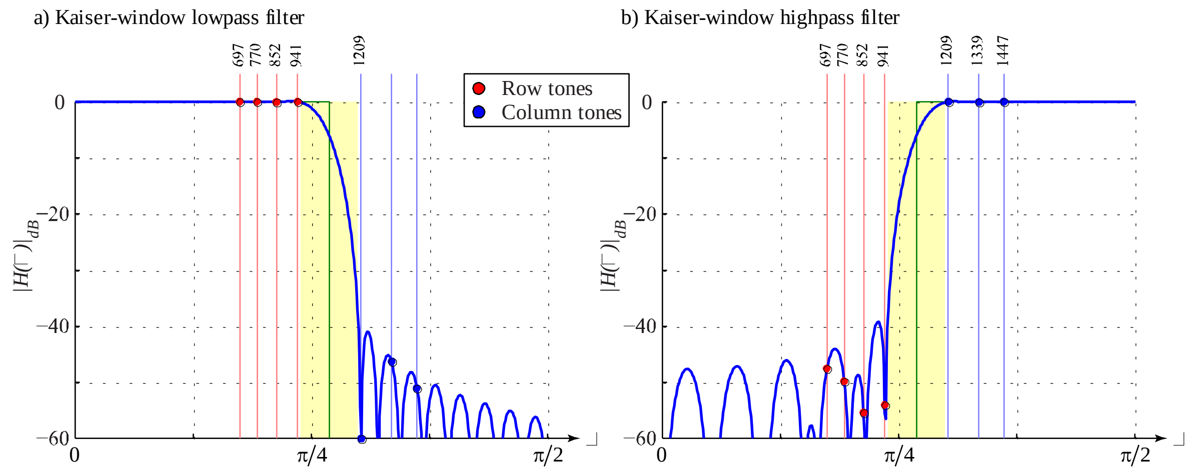

Figure 6-6: Kaiser-window filters

Kaiser filter

Now design Kaiser lowpass and highpass filter to do the same task. In this case, again choose $N = 73$. The only parameter left to calculate is β. To find this, choose the width of the transition, $\Delta\omega$, to be 90% of the frequency difference of the highest row tone and the lowest column tone; namely,

```
dw = 0.9*(wtones(5)-wtones(4));
```

Your responses should look like those in Figure 6-6. The yellow region shows that the transition zone. The SNR is about 47dB. What is the resulting β of your filter? If you listen to the response of the highpass filter, you'll probably hear the first two row tones sneaking through (i.e. digits 1-6), even though they are attenuated by almost 50 dB with respect to the column tones! The human ear is really amazingly sensitive.

Spline filter

Make lowpass and highpass spline filters. The best filters I am able to make have $\Delta\omega$ selected such that

```
dw = 0.77*(wtones(5)-wtones(4));
```

You should see the responses of Figure 6-7.

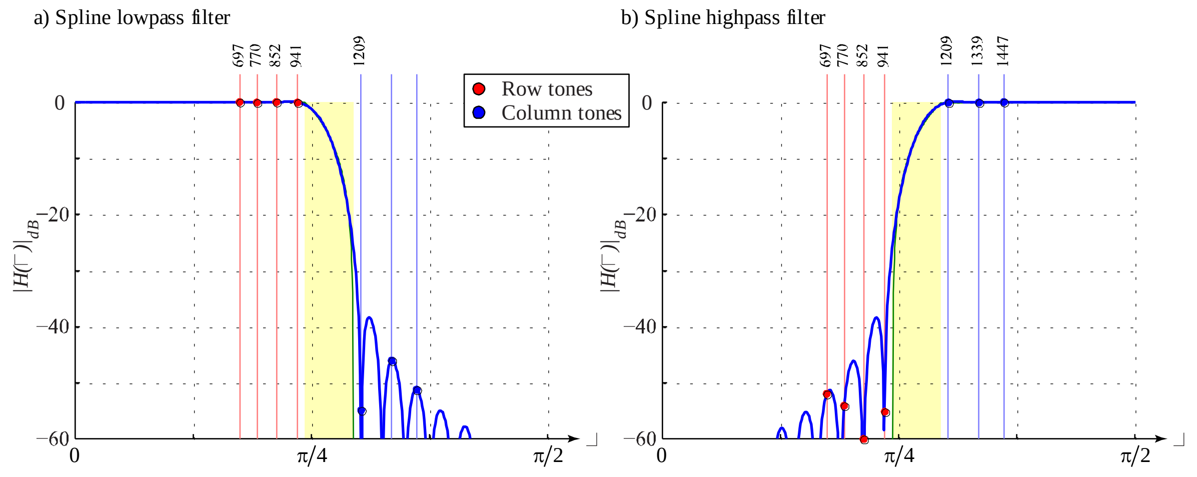

Figure 6-7: Spline filters

If you listen to the response of the highpass filter you'll probably still hear the first two row tones. Still not good enough! Onward!

Frequency-sampled filter

Uniform frequency spacing

Now design a frequency sampled filter with frequencies that occur at uniform multiples of $2\pi/N$, where $N = 73$. Your amplitude function, $A[k]$, should be set such that all frequencies above ω_c have an amplitude of one and all frequencies below have an amplitude of zero; namely,

$$A(\omega) = \begin{cases} 1, & \omega > \omega_c \\ 0, & \omega < \omega_c \end{cases}.$$

Mark the sampling frequencies on your curves with small 'X's. The results should look like those in Figure 6-6.

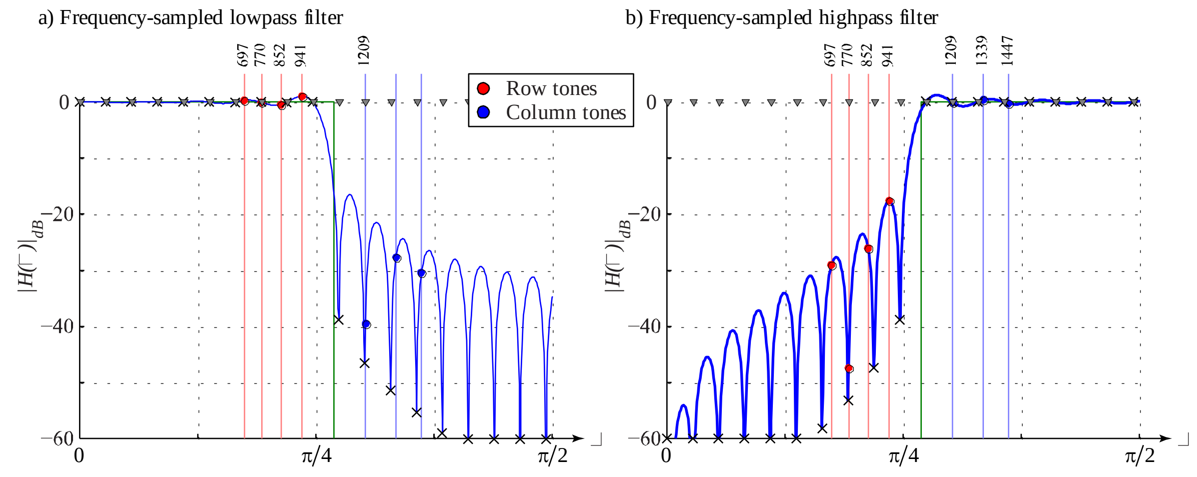

Figure 6-7: Frequency-sampled filters, uniform spacing

In my figures, I've also marked the sampling frequencies with small inverted triangles on the top of the plot, but you don't have to do that. Notice how crummy this frequency-sampled filter is. The SNR is only 17 dB, even worse than the rectangular filter. Why bother with this filter? It certainly seems like a step backwards. Well, let's modify it a bit…

Non-uniform frequency spacing

We now adjust the sampling frequencies so that seven of the sampled frequencies occur exactly at the frequencies of the row and column tones. Specifically, list for yourself the $M+1=37$ sampled frequencies, `w`, in your code. Pick the seven frequencies that are closest to the seven tone frequencies, `wtones`, and replace those values of `w` with `wtones`. Now recomputed the matrix, `C` and redesign the filter. The results should look something like those of Figure 6-7.

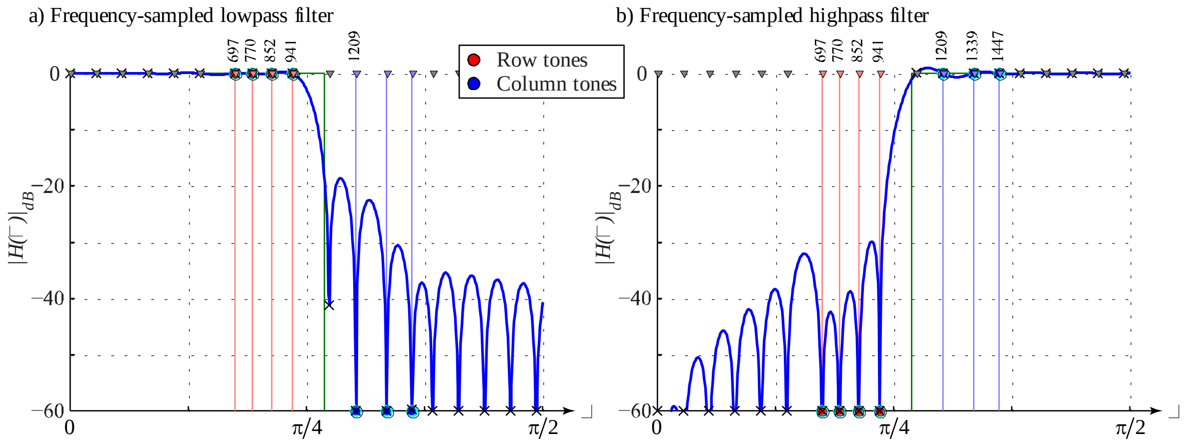

Figure 6-8: Frequency-sampled filter, non-uniform spacing

Wow! Even though many of the sidelobes are terrible, the filter is doing the task that we require: attenuating the row or column tones almost completely. The SNR is very large, perhaps >300 dB. Listen to the result for yourself.

Optimum filter

Lastly, design an optimal filter of length, $N = 73$ using `firpm`. For example, for the lowpass filter the syntax will simply be:

```
firpm(N-1, [0 wtones([4 5])/pi 1], [1 1 0 0])
```

The results should look like those in Figure 6-8. Listen to the highpass filter's response to the phone tones. The order of the optimum filter needs to be quite a bit higher if it is to match the perceived performance of the frequency-sampled filter. Try to design several optimum high-pass filters of increasing length, and see how large a filter you need to get before you stop being able to hear the row tones.

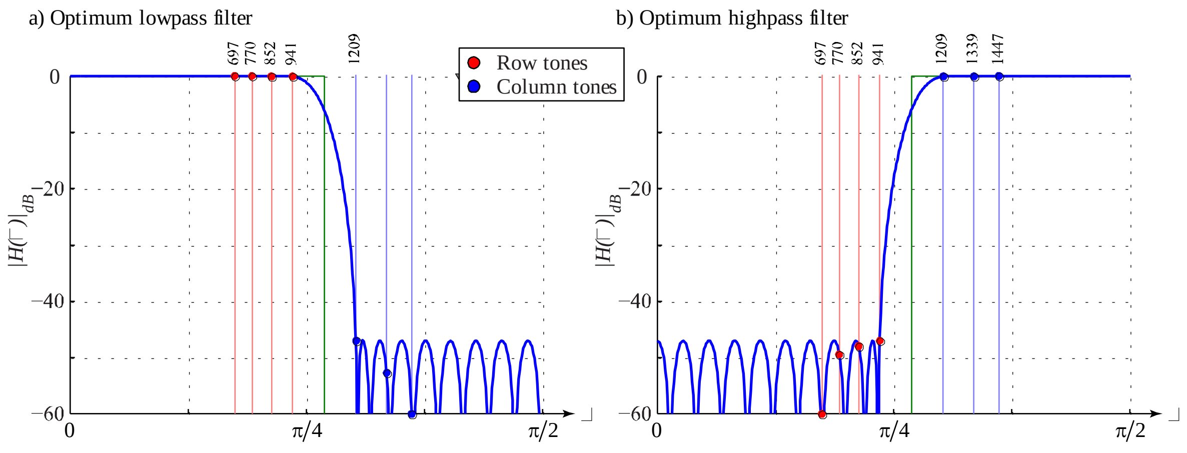

Figure 6-9: Optimum filter

The deliverables for this lab are eight pictures, like those of Figures 6-6 to 6-9.