ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
School of Computer and Communication Sciences

Handout 20 Signal Processing for Communications
Solutions to Homework 9 May 17, 2009

Problem 1.

1.

$$H(e^{i\omega}) = \begin{cases} 1 & |\omega| < \omega_c, \\ 0 & \text{else} \end{cases}$$

$$\begin{aligned} h[n] &= \frac{1}{2\pi}\int_{-\omega_c}^{\omega_c} H(e^{j\omega n})e^{j\omega n}d\omega \\ &= \frac{\sin(\omega_c n)}{\pi n} = \frac{\omega_c}{\pi}\text{sinc}(\frac{n\omega_c}{\pi}) \end{aligned}$$

2. We want to minimize $\|H(e^{j\omega}) - H_K(e^{j\omega})\|_2$. Using the Parseval's Theorem, we have

$$\begin{aligned} \|H(e^{j\omega}) - H_K(e^{j\omega})\|_2^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi} |H(e^{j\omega}) - H_K(e^{j\omega})|^2 d\omega &= \sum_n |h[n] - h_K[n]|^2 \\ &= \sum_{n:|n|\le K} |h[n] - h_K[n]|^2 + \sum_{n:|n|>K} |h[n] - h_K[n]|^2 \\ &= \sum_{n:|n|\le K} |h[n] - h_K[n]|^2 + \sum_{n} |h[n]|^2. \end{aligned}$$

Note that the second term does not depend on the choice of $h_K[n]$, and in order to minimize the first term, we choose $h_K[n] = h[n]$ (to make all terms in the summation equal to zero). Thus,

$$h_K[n] = \begin{cases} h[n] & |n| \le K \\ 0 & \text{else} \end{cases}.$$

3.
```
function [h,H] = wind(K,window\_type,fr)
window = char(window\_type);
n= -K:K;


if (nargin == 2)
    % if we want to plot different windows on top of each other.
    % Question 1.5
    h = ones(1,2*K+1);
else
    h = sinc(n*fr/pi)*fr/pi;
end
switch lower(window)
    case('rectangular')
```

```
            win = ones(1,2*K+1);
        case('triangular')
            win = conv(ones(1,K+1),ones(1,K+1))/(K+1);
        case('hamming')
            win = 0.54 - 0.46*cos(2*pi*(n+K)/(2*K));
        case('blackman')
            win = 0.42 - 0.5*cos(2*pi*(n+K)/(2*K)) + 0.08*cos(4*pi*(n+K)/(2*K));
    end
    h = h.*win;
    w = linspace(-pi,pi,2048);
    % implement fft
    H = fftshift(fft(h,2048));
    % another way to implement
    % H = 0;
    % for k=-K:K
    % H=H+h(k+K+1)*exp(-j*k*w);
    % end
    if(nargin ==3)
        figure;plot(w,abs(H));
        title(['Window Type =', window , ' and Length =',num2str(2*K+1)])
        xlabel('frequency [-pi,pi]');
        ylabel('|H(e^{(jw)})|')
    end
    end
```

4. Fig. 1 shows the frequency response of the filter for different length. It can be seen that the larger length FIR filter gives a better approximation of $H(e^{j\omega})$. The length of transition band decreases. However, we can never get rid of the ripples. We can see the jumps (ripples) at the cut-off frequencies ($\omega = \pm\pi/4$), which corresponds to the Gibbs phenomenon. You can think of it as approximating a discontinuity with linear combination of continuous functions, which is impossible for any finite K.

5.
```
%  Question 1.5
[r,R] = wind(12,'rectangular');
[tr,TR] = wind(12,'triangular');
[h,H] = wind(12,'hamming');
[b,B] = wind(12,'blackman');

plot(r,'-*');
hold on
plot(tr,'-o');
plot(h,':');
plot(b);
```

6. Already implemented in 3.

7.
```
% Question 1.7
[r,R] = wind(12,'rectangular',pi/4);
[tr,TR] = wind(12,'triangular',pi/4);
[h,H] = wind(12,'hamming',pi/4);
```

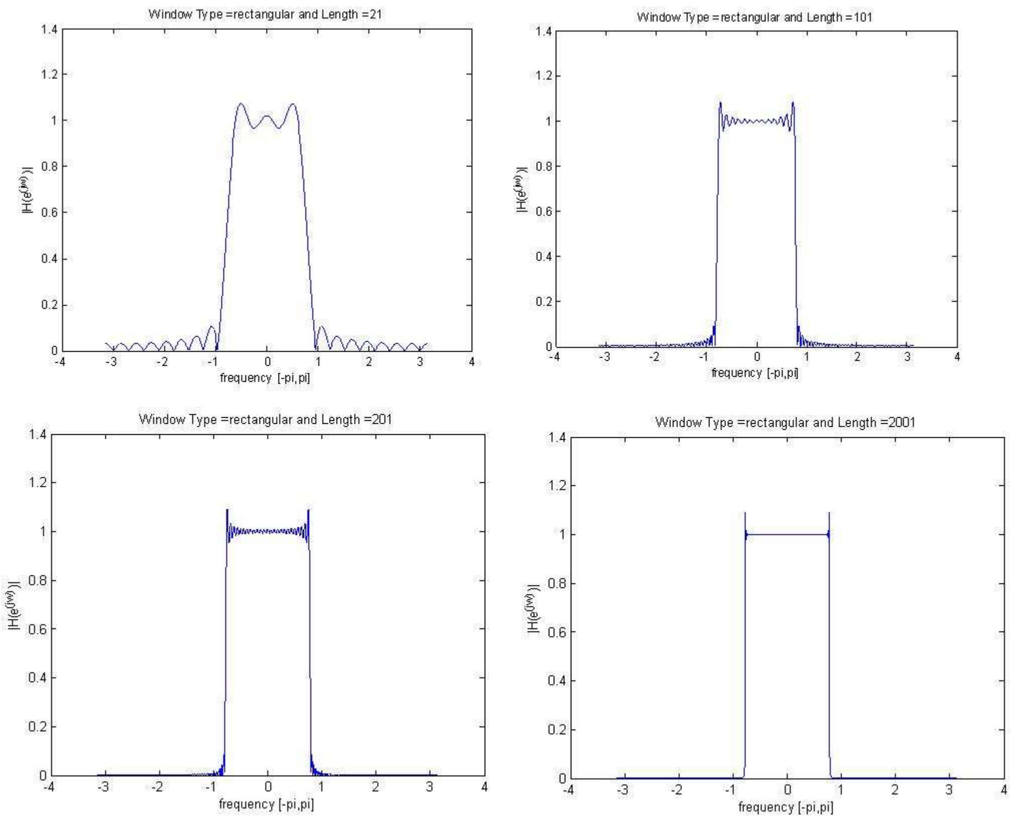


Figure 1: Approximation of low pass filter with rectangular function of different lengths

```
[b,B] = wind(12,'blackman',pi/4);


plot(abs(R),'-r');
hold on
plot(abs(TR),'g');
plot(abs(H),'c');
plot(abs(B));

w = linspace(-pi,pi,length(R));
plot(w,20*log10(abs(R)),'-r');
hold on
plot(w,20*log10(abs(TR)),'g');
plot(w,20*log10(abs(H)),'c');
plot(w,20*log10(abs(B)));
xlabel('frequency, [-pi,pi]')
```

The choice of the window to be used depends on the application. There is a trade off between the transition width and the side lobe amplitude(error). We can observe in Figure 3 that rectangular window has the narrowest transition band whereas it has the maximum rippling artifacts. On the other hand blackman window has less attenuation in the stop band but it has a wider transition band. For example, if we want to separate two sinusoidal signal which have similar frequencies then we can use rectangular windows as it has narrower transition band(Note that, the amplitudes of

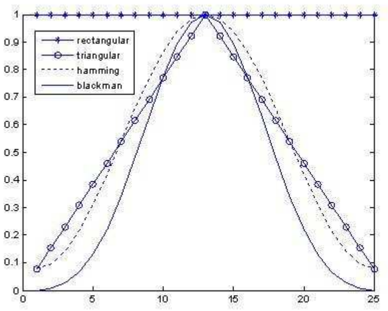


Figure 2: Different Window Types

the two signals should be comparable to each other). If two sinusoidal have different frequencies, then using a wider transition band(lower ripples) may be more efficient. In summary, the choice of the window type is the trade off between separating signals with similar frequencies(and comparable amplitudes) and separating signals with dissimilar frequencies(and different amplitudes).

PROBLEM 2. We will do everything on the file `corrupt_l.wav`, the solutions for the other files (`corrupt_o.wav`, `corrupt_r.wav` and `corrupt_t.wav`) being similar.

1. We first read the samples of `corrupt_l.wav`:

   ```
   >> [corr, fs] = wavread('corrupt\_l.wav');
   ```

   In the above, `corr` holds the samples, and `fs` holds the sampling frequency (44100 Hz in this case). We can now listen to the file:

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

   On top of the piano, there is a higher frequency noise. Plot the DFT of the samples to locate the noise:

   ```
   >> plot(abs(fftshift(fft(corr))))
   ```

   By visually inspecting the DFT plot, we see that a significant portion of the original music lies roughly in the frequency range $[-0.4, 0.4]$ (that is, $[-0.127\pi, 0.127\pi]$), and the positive portion of the noise DFT is centered at $\pi/4 \approx 0.785$, with a bandwidth of roughly $\pi/5 \approx 0.63$. We need a lowpass filter to get rid of the noise, whose passband is the frequncy range where the original music lies, i.e., roughly $[-0.127\pi, 0.127\pi]$.

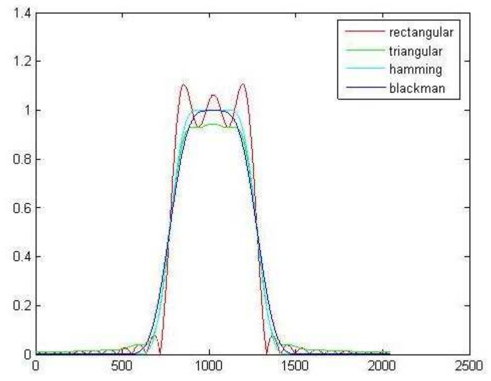
2. The ideal lowpass filter $h[n]$ with passband $[-0.127\pi, 0.127\pi]$ has coefficients

   $$h[n] = \frac{\sin(0.127\pi n)}{\pi n}, \qquad n \in \mathbb{Z}.$$

   Using a rectangular window of width 1001, we obtain the filter $h_{500}[n]$ with

   $$h_{500}[n] = \begin{cases} h[n] & \text{if } |n| \le 500 \\ 0 & \text{otherwise} \end{cases}.$$

   We will do the filtering in the frequency domain. First generate the filter in the time domain:



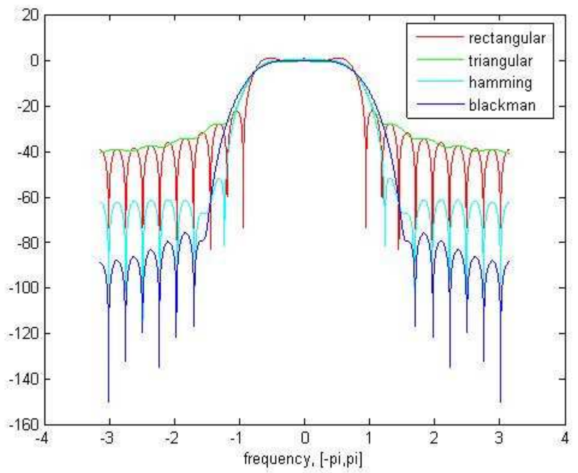


Figure 3: Frequency Responses in normal(up) and log scale(bottom)

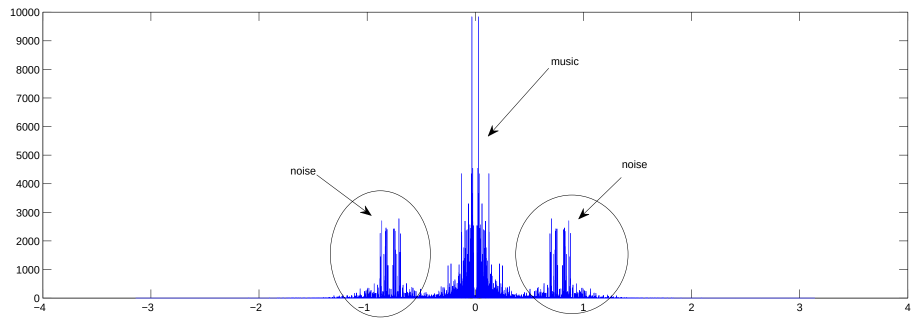


Figure 4:

```
>> n = -500:500;
>> filter = sin(0.127*pi*n)./(pi*n);
```

Set the value of filter for $n = 0$:

```
>> filter(501) = 0.127;
```

Pad the filter to generate a filter vector of the correct length (equal to the length of the audio file)

```
>> filter  = [zeros(1, (length(corr') - length(filter))/2), filter,
                     zeros(1, (length(corr') - length(filter))/2)];
```

Now plot the DFT:

```
>> plot(N, abs(fftshift(fft(filter))))
```

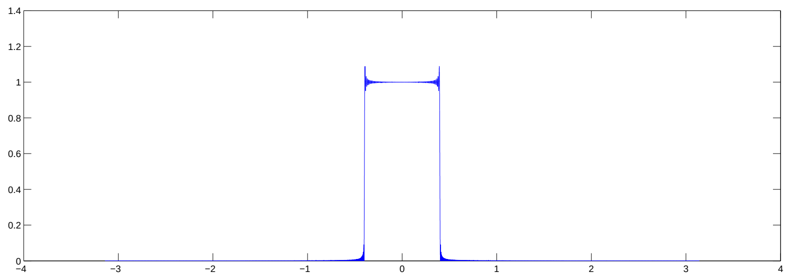


Figure 5:

```
>> FILTER = fft(filter);
>> CORR = fft(corr');
>> CLN = FILTER .* CORR;
>> plot(fftshift(abs(CLN)))
```

Recover the filtered signal:

```
>> cln=fftshift(ifft(CLN));
```

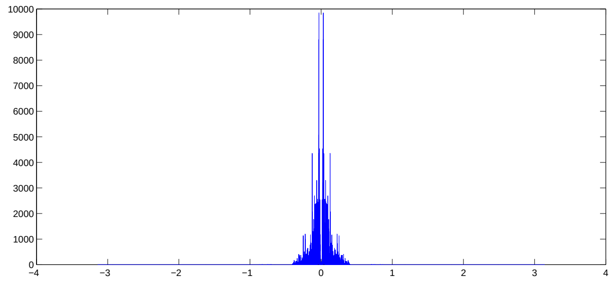


Figure 6: DFT of the filtered signal

Upon listening to the resulting signal (after taking the inverse DFT, of course), one will note that most of the noise content has been removed. However, one can still hear the low frequency content of the noise in the background.

3. Keeping in mind that the cutoff frequency of the target filter is 0.127π, one way of generating a filter of length 800 for our purpose is the following:

```
>> [lowpass,e] = firpm(800, [0 0.125 0.135 1], [1 1 0 0], [1 10]);
```

The DFT of the filter `lowpass` is given in the following figure:

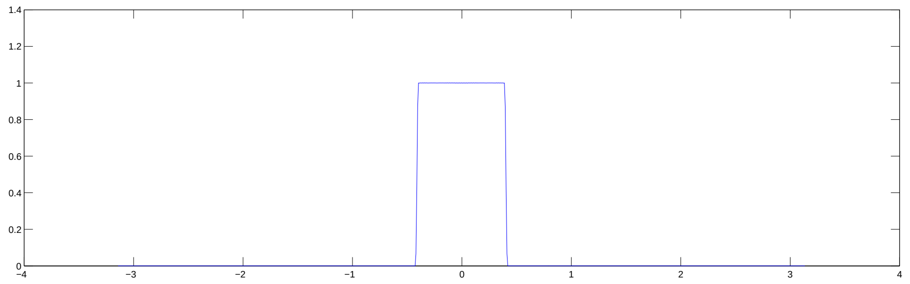


Figure 7:

Now filter out the noise in the time domain:

```
>> cln = conv(lowpass, corr);
```

The DFT of the filtered signal is given in Figure 8. Upon listening to the filtered sound file, one will note again that the low frequency content of the noise is still present in the background.

4.

5. Assuming that the sequences `corr` and `cln` hold the original and the filtered signals respectively, the noise signal is obviously equal to `corr` $-$ `cln`. If we wanted to extract the noise from the original signal in the first place, we would use a high pass filter with cutoff frequency equal to 0.127π. Note that such an ideal filters frequency response is given by $1 - H(e^{j\omega})$, where $H(e^{j\omega})$ is the ideal lowpass filter we implemented above (hence, the noise is given by `corr` $-$ `cln`). That is, an implementation of a lowpass filter leads automatically to an implementation of a highpass filter as well.

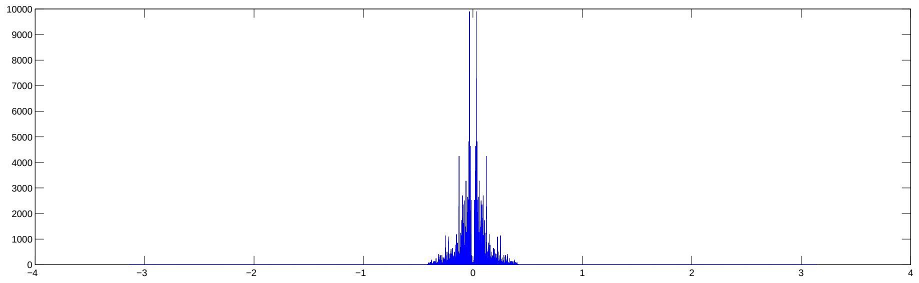


Figure 8:

6. Let $\mathtt{y} = \mathtt{corr} - \mathtt{cln}$. We have

$$y[n] = w[n]\cos(0.25\pi n).$$

We first move $y[n]$ to the baseband:

```
>> y = corr - cln';
>> N = 1:length(y');
>> y_bb = y .* cos(0.25*pi*N');
>> plot(abs(fftshift(fft(y_bb))))
```

We then filter out the high frequency content using a lowpass filter whose cutoff frequency is $\pi/5$, roughly the bandwidth of `y`.

```
>> [lowpass2,e] = firpm(800, [0 0.2 0.21 1], [1 1 0 0], [1 10]);
>> y_bb_cln = conv(y_bb, lowpass2);
>> L= -pi:2*pi/length(y_bb_cln'):pi-2*pi/length(y_bb_cln');
>> plot(abs(fftshift(fft(y_bb))))
>> plot(abs(fftshift(fft(y_bb_cln))))
```

The DFTs of `y_bb` and `y_bb_cln` are plotted in Figure 6.

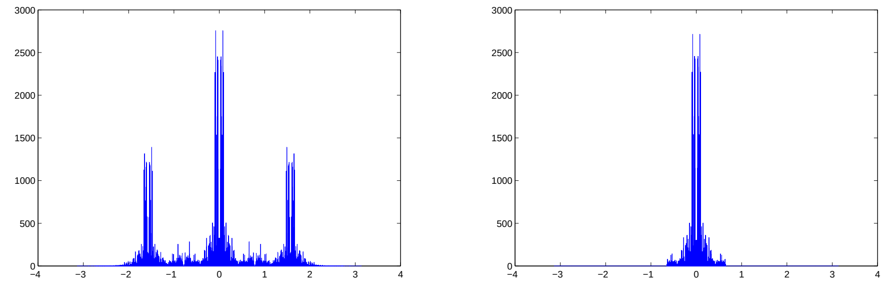


Figure 9:

Listen to `y_bb_cln`:

```
>> soundsc(y_bb_cln', fs);
```

You will hear a cello playing (with some background noise from the original signal remaining). If you go through the above steps for the other files you should be able to hear a piano (`corrupt_o.wav`), a guitar (`corrupt_r.wav`) or a clarinet (`corrupt_t.wav`) playing.