Noise filtering using FIR and IIR filters

Erik Sandström

March 15th 2018

1 Introduction

The objective of this project is to filter out noise at a specific frequency of an arbitrary audio signal using finite impulse response (FIR) and infinite impulse response (IIR) notch filters. The filters are designed, tested and compared. The MatLab implementation is also compared with an equivalent implementation in C. The problem is relevant in order to use digital signal processing techniques in order to process data after sampling.

Corrupting an audio signal

An arbitrary audio mono wav file was chosen. The original signal was corrupted by a sinusoid at 2.4 kHz and the resulting spectrograms of the original signal and the corrupted version are displayed in figure 1 and 2 respectively. Signal x and y refers to the uncorrupted and corrupted signals respectively.

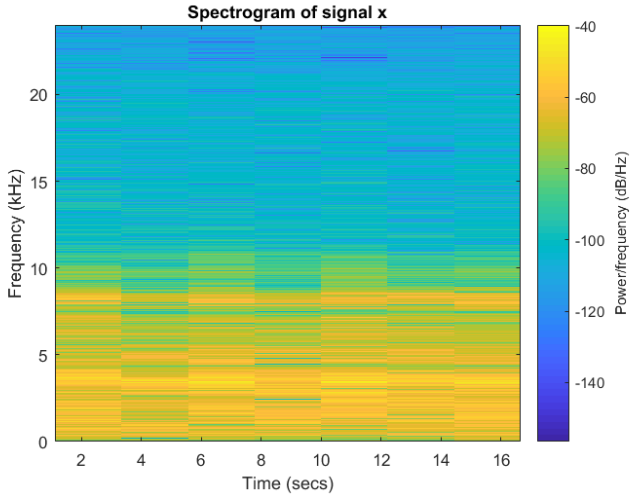

Figure 1: Original signal.

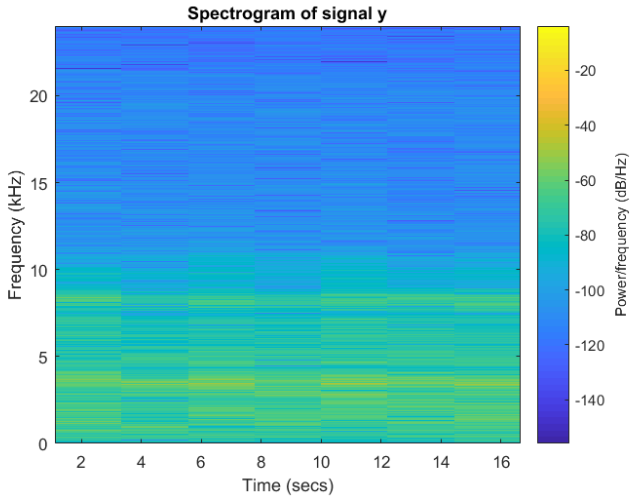

Figure 2: Corrupted signal with corruption frequency at 2.4 kHz.

The following lines of code describe the implementation.

```
f = 2400;
v = samplefunc(length(x), f, Fs);
y = x + v';
y = y/max(y);
filename3 = 'Task2.wav';
audiowrite(filename3, y, Fs); %Data is clipped because interval is [-1,1) so 1 becomes
%closest possible value instead.
spectrogram(y,[],[],[],Fs,'yaxis');
title('Spectrogram of signal y');
figure
spectrogram(x,[],[],[],Fs,'yaxis');
title('Spectrogram of signal x');
```

FIR notch filter

Given the frequency response of an ideal low pass filter

$$X(e^{jw}) = \begin{cases} e^{-j\alpha\omega}, & \text{if} \quad 0 \leq \omega \leq \omega_c \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

the filter is displayed in figure 3

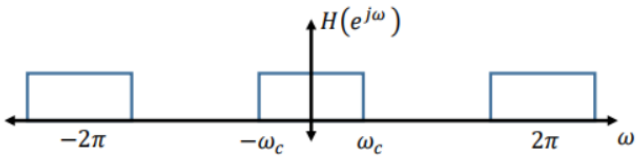

Figure 3: Ideal low pass filter.

The inverse Fourier transform of equation 1 yields

$$x[n] = \frac{sin(\omega_c(n-\alpha))}{\pi(n-\alpha)}, \tag{2}$$

which is an IIR filter. Truncating $x[n]$ using a rectangular window $\omega[n]$ yields the FIR approximating low pass filter

$$h[n] = x[n]\omega[n], \tag{3}$$

where

$$\omega[n] = \begin{cases} 1, & \text{if } 0 \leq n \leq M \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

where M is the length of the filter. A filter with order 14 was assumed to be a good compromise between a not too simply model and a not too complex one. This yields $M = 14$ and a line of symmetry at $\alpha = M/2 = 7$ for $h[n]$ which is a type I linear phase system. The sequence of the FIR low pass filter is

$$h[n] = \sum_{k=0}^{M} h[k]\delta[n-k], \tag{5}$$

where

$$h[k] = x[k] \quad 0 \leq k \leq M. \tag{6}$$

Choosing $\omega_c = 2\pi\frac{2400}{48000}$ for a time domain cut-off frequency of 2.4 kHz and plotting the frequency response of the FIR low pass filter given by equation 5 yields figure 4 which confirms the implementation.

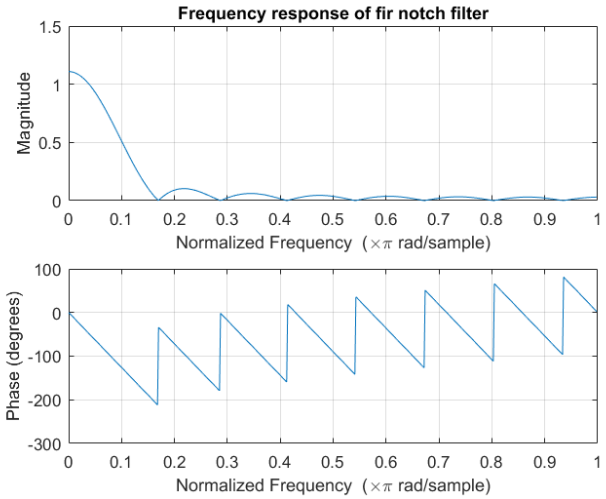

Figure 4: FIR low pass filter.

Shifting the amplitude response of the FIR low pass filter by -1/2 and scaling with 2 yields the first order notch FIR filter $G(e^{j\omega})$ given by

$$G(e^{j\omega}) = 2e^{-j7\omega}(H_{amp} - \frac{1}{2}), \tag{7}$$

where $H(e^{j\omega}) = e^{-j7\omega}H_{amp}$ is the frequency response of the FIR low pass filter. It can now be concluded that the sequence $g[n]$ of the FIR notch filter is given by

$$g[n] = \frac{sin(\omega_c(n-7))}{\pi(n-7)} \quad 0 \leq n \leq 6 \tag{8}$$

$$g[7] = \frac{\omega_c}{\pi} - \frac{1}{2} \tag{9}$$

$$g[n] = g[2\alpha - n] = g[14 - n] \tag{10}$$

Plotting the frequency response of $G(e^{j\omega})$ yields figure 5.

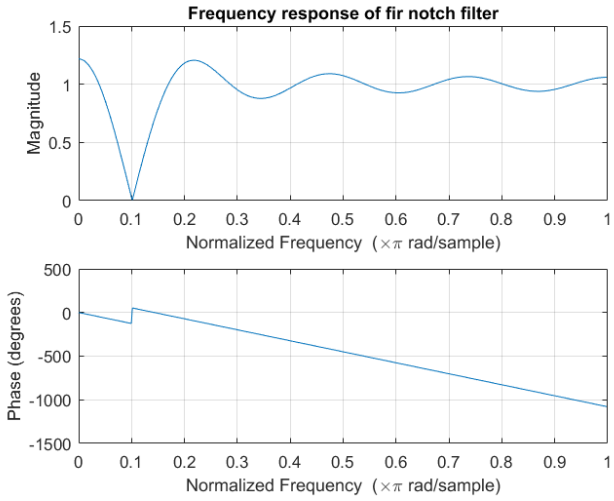

Figure 5: FIR notch filter.

Filtering using FIR notch filter

The corrupted speech signal was run through the FIR notch filter and the spectrogram is given by figure 6.

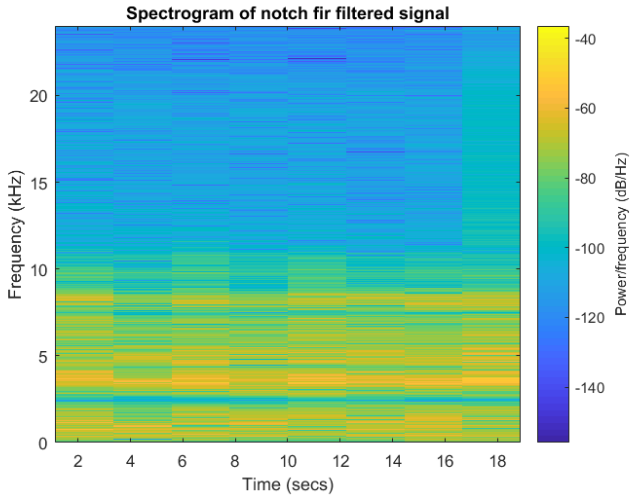

Figure 6: Spectrogrram of corrupted signal after being processed by the FIR notch filter.

It is clear, by looking at the spectrogram in figure 6, that a specific region at

2.4 kHz is greatly attenuated as a result of the FIR notch filter. The function that applies the filter above is given by the following lines of code.

```
function output = fir(input, filtercoeff)
output = zeros(1,length(input)+length(filtercoeff)-1);
input = [zeros(length(filtercoeff)-1,1)' input' zeros(length(filtercoeff)-1,1)'];
for i = 1:length(output)
   output(i) = fliplr(input(i:i+length(filtercoeff)-1))*filtercoeff';
end

end
```

IIR notch filter

The system function of a second order IIR notch filter is of the form

$$H(z) = \frac{(-e^{j\omega_0} + z^{-1})(-e^{-j\omega_0} + z^{-1})}{(1 - re^{-j\omega_0}z^{-1})(1 - re^{j\omega_0}z^{-1})} \tag{11}$$

$$= \frac{1 - 2cos(\omega_0)z^{-1} + z^{-2}}{1 - 2rcos(\omega_0)z^{-1} + r^2z^{-2}} \tag{12}$$

where ω_0 denotes the notch frequency which is removed when the filter is applied to a given signal. The poles are located at distance r from the origin and at angles $\pm\omega_0$. The zeros are located on the unit circle at angles $\pm\omega_0$. To make the filter causal and stable, r is chosen to a number less than one. A second order notch filter with notch frequency 2.4 kHz means that

$$\omega_0 = 2\pi\frac{2.4}{48} = 0.1\pi, \tag{13}$$

where 48 denotes the sampling frequency in our case. The function that applies the filter above is given by the following lines of code.

```
function output = iirnotch(input, wc)
output = zeros(1,length(input)+2);
input = [zeros(1,2) input];
r = 0.99;
for  i = 3:length(output)
   output(i) = 2*r*cos(wc)*output(i-1)-r^2*output(i-2)...
   +input(i)-2*cos(wc)*input(i-1)+input(i-2);
end
output = output(3:end);
end
```

The frequency response of the IIR filter shows a correct implementation.

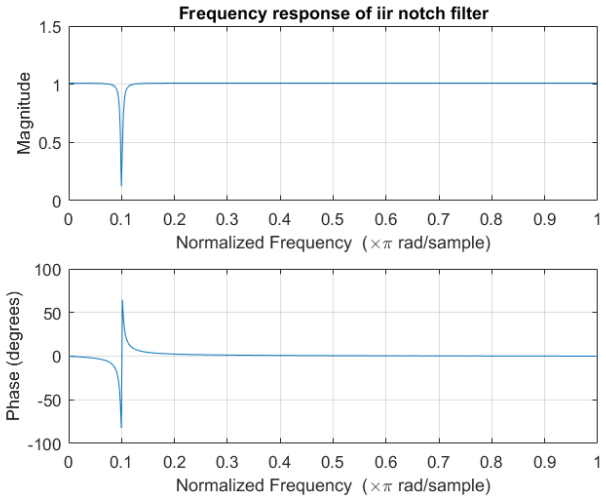

Figure 7: Frequency spectra of iir notch filter.

The corrupted signal was run through the second order IIR notch filter resulting in the spectrogram given by figure 8.

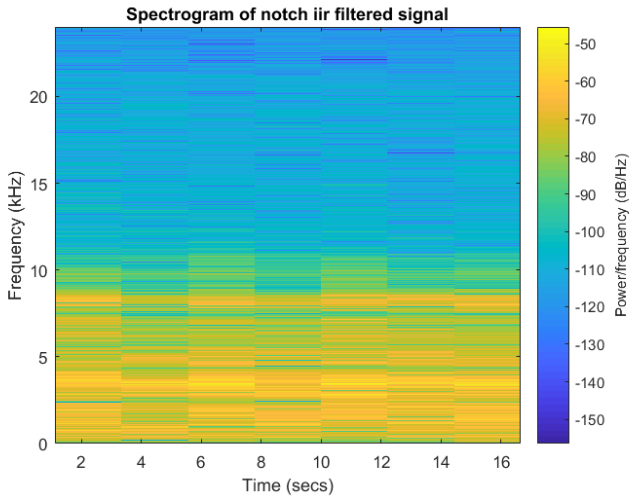

Figure 8: Spectrogram of corrupted signla run through the IIR notch filter.

Zooming in on the frequency 2.4 kHz in figure 8, it is clear that the notch filter performs very well with regards to eliminating the specific frequency.

Implementing the FIR filter in C

A comparison between the difference mathematically between the signal created using matlab and C-code was conducted by calculating the norm squared error normalized with the norm of the matlab signal. This resulted in

$$NormError_{FIRnotch} = 0.0035 \tag{14}$$

One can question the correctness of the implementation of the FIR notch filter in C. The difference between the two implementations is believed to be due to floating point round off errors. If there, against the odds, is an error made during the implementation, the error does not affect the result in a real world setting since no subjective difference when listening to the two versions can be heard.

Implementing the IIR filter in C

A comparison between the difference mathematically between the signal created using matlab and C-code was conducted by calculating the norm squared error normalized with the norm of the matlab signal. This resulted in

$$NormError_{IIRnotch} = 7.4802 * 10^{-7} \tag{15}$$

No subjective difference when listening to the two versions can be heard.