2ND ORDER BUTTERWORTH LOWPASS FILTER

Lab report 1

EunJu Gong 311050557

Digital Audio Systems, DESC9115, Semester 1 2012
Graduate Program in Audio and Acoustics
Faculty of Architecture, Design and Planning, The University of Sydney

1. INTRODUCTION

Developments of a filter began with the exploration of the simple low pass filter using delay and add processing, which we experimented in MatLab session in week 2. This led to exploring more complex filters such as the IIR 2nd order Butterworth filter. The ideal low pass filters can be described as a system that allows through a certain frequency components or octave of an original signal above cutoff frequency fc (the passband), and rejects other frequency range (the stopband). [1]

A butterworth low pass filter (2nd order) is often referred one that has almost maximally flat (no ripple) filter. This is because the filter is optimized to provide the sharpest roll off (drop in amplitude) to have flat frequency response with no ripples in the pass band (Figure1). [2]

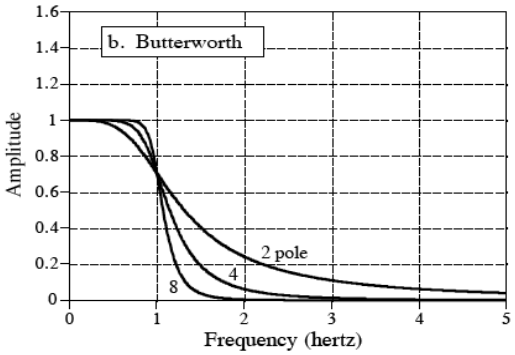

Figure 1. *The butterworth filter provides the flattest passband.*

The Butterworth filter will be designed with the exploration of a simple low pass FIR Filter using delay and add processing, and its technical overview. The MatLab code was sourced from the MatLab tutorials from the MathWorks website and YouTube tutorials. The scripting and function were based from a simple code that was expanded.

2. FIR FILTER

FIR Filters are digital filters with finite impulse response. They are known as non-recursive digital filters as they do not have the feedback loops. [3] With the exploration of the simple low pass filter using delay and add filter that uses basic comb filtering as the process is the first step of this project.

As shown in Fig.2, the MatLab session in week 2 in which we experimented in MatLab using a signal with an exact copy of itself superimposed with a very short delay of 1 sample.

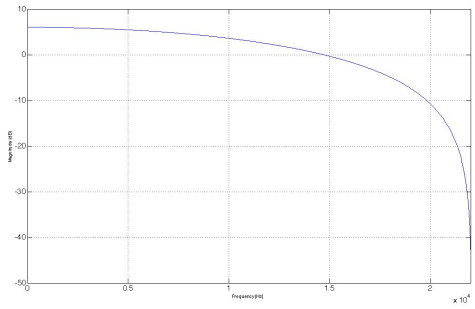

Figure 2. *The simple low pass filter using delay and add processing, magnitude response of the frequency spectrum. X= Frequency, Y= Magnitude.*

2.1 FIR low pass filters

It is almost impossible to get an ideal FIR low pass filter. This is because that the impulse response required to implement the perfect filter is infinitely long. Finite length approximations to the ideal impulse response lead to the presence of ripples in both the pass band and the stopband of the filter. [5]

2.2 Technical overview of a FIR filter

Finite Impulse Response (FIR) filters are one of the primary types of digital filters in Digital Signal Processing, the other type being Infinite Impulse Response (IIR) filters. Input and output relation from the FIR is described by the equation below.

$$y(n)=\sum_{k=0}^{N-1}h(k)x(n-k),\quad H(z)=\sum_{k=0}^{N-1}h(k)z^{-k}$$

Equation 1. *Input and output relation from the FIR filters.*

As shown in Fig.3, FIR filters are easily designed to be linear phase and simple to implement. Multiplying the impulse response *g(0),g(1),g(2)* resulting from the tab delay line with delay z^{-1} and sum all the values. [4]

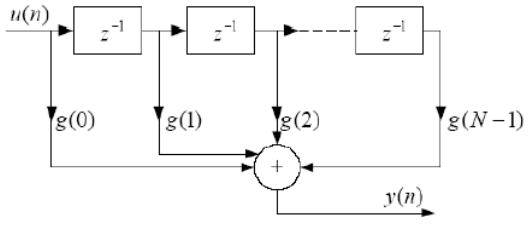

Figure 3. *The diagram of FIR Filters and its implementation.*

3. IIR – 2ND ORDER BUTTERWORTH LOW PASS FILTER

3.1. Technical aspects of IIR Filters

Infinite Impulse Response (IIR) filters consist of a feedback path and it contributes to the output signal. Each output is a sum of new input signal: *x[n], x[n-1], x[n-3]…,* with previously calculated values of the output signal: *y[n-1], y[n-2], y[n-3]….*, and summation of earlier output values, all multiplied by their respective coefficients: a_0, a_1, a_2 …. . [6]

$$y[n] = a_0 x[n] + a_1 x[n-1] + a_2 x[n-2] + a_3 x[n-3] + \cdots + b_1 y[n-1] + b_2 y[n-2] + b_3 y[n-3] + \cdots$$

Equation 2. *The recursion equation. In this equation x[ ] is input signal, y[ ] is the output signal, and the a's and b's are coefficients.*

As mention in the introduction, a Butterworth low pass filter (2nd order) is often referred one that has almost maximally flat (no ripple) filter. When the ripple is set to 0%, the filter is called Butterworth filter. [8] Because of this reason, choosing the Butterworth filter for this project is the next process from the simple low pass FIR Filter.

3.2. Butterworth Filters

A British engineer S. Butterworth first described Butterworth filter in 1930. Its frequency response is given as below.

$$|H(j\omega)|^2 = \frac{1}{1 + \varepsilon \left(\frac{\omega}{\omega_p}\right)^{2n}}$$

Equation 3. *Where the generalized equation representing a "nth" Order Butterworth filter*

n represents the filter order, Omega ω is equal to $2\pi f$ and Epsilon ε is the maximum pass band gain, (Amax). $\varepsilon\omega_P$ is the 3dB cut off frequency.

3.3. Making the filter

The basic MatLab code of the filter is obtained from tutorial from the MathWorks website and also watching YouTube tutorials provided for using MatLab.

The procedure to design a Butterworth filter is to determine the four specifications.

1) ω_p = The passband edge.
2) ω_s = The stopband edge.
3) Passband ripple, in dB. This value is the maximum permissible passband loss in dB.
4) Stopband attenuation, in dB. This value is the number of decibels the stopband is down from the passband.

The cut off frequency is 1000 Hz in this project. The main part of the filter is the filter coefficients, which are derived from;

```
[b,a]=butter(n,w,'s');
```

N is the filter order therefore, if n is 2, it will design the 2nd order Butterworth filter. W is the 3dB cut off frequency, *num* is 1x2 vector numerator coefficients, and *dom* is a 1x2 vector of denominator coefficients. [9]

```
[b,a]=butter(2,fc/fn);
```

The numerator is being the frequency cut-off and the denominator being the Nyquist frequency, fn. From there I applied the filter to the input wave using the IIR Filter coefficients, b=numerator, a=denominator.

```
y=filter(b,a,x);
```

After getting IIR Filter coefficients, the next step is that apply filter to x, which is an input wave file. *Filter* is an inbuilt function of Matlab whereas "Butterworth filter" usually comes with MatLab signal processing toolbox. To complete the filter function, using two coefficients obtained from the previous process is essential. Apply this filter to the input signal, x, then simply can listen the filtered output signal, y.

The script was created to implement the function, including importing the wave file, determining the parameters and finally writing the filtered wave file to disc. [Sound file 'yeah_filtered.wav']

3.4. Parameters

The parameters for the MatLab function are:
'fn' – Nyquist frequency
'y' – output of wave file
'x' – input of wave file
'fs' – Sample rate
'fc' – cut-off Frequency (Cut off Frequency applies to an edge in low pass filters and it characterizes a boundary between a passband and a stopband.). Therefore, filters select low frequencies up to the cut-off frequency fc and attenuate frequencies higher than fc [7].

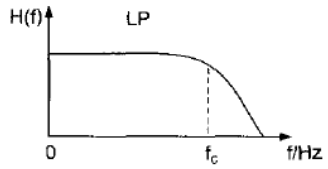

Figure 4. *Low pass filter with cut off frequency fc.*

4. RESULTS

As you can see from the diagrams below the filter response as a result of the filter was very successful. The high frequencies above 1kHz frequency cut-off, selected for this experiment have been eliminated. [Figure 5 and 6] In a more complex filter other parameters such as Q (bandwidth) would be accessible however for this example a fixed slope is more than enough to exhibit the filter standard behavior.

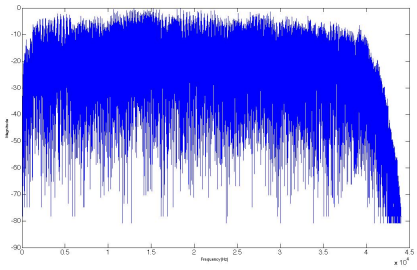

Figure 5. *A FFT frequency spectral response of the original signal. X= Frequency, Y= Magnitude.*

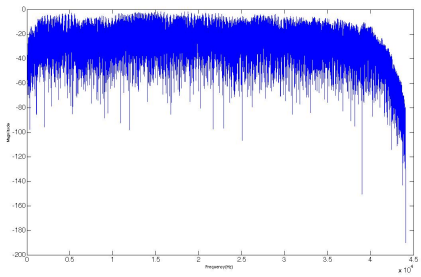

Figure 6. *A FFT frequency spectral response of the filtered signal. X = Frequency, Y= Magnitude.*

As shown in Fig. 7 and 8, the diagrams are obtained by using the *myspectrogram* function in MatLab (week 5 tutorial), the window size of 4096. As illustrated below, the frequency response has changed and eliminated above the cut off frequency 1000Hz.

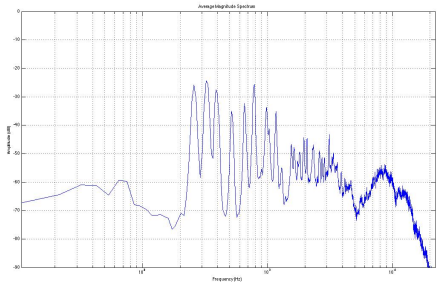

Figure 7. *The average magnitude spectrum of the original file.* X= Frequency, Y= Magnitude.

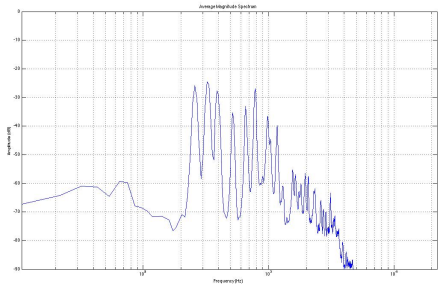

Figure 8. *The average magnitude spectrum of the filtered file.* X= Frequency, Y= Magnitude.

It is evident from these diagrams that the filter has changed the frequency response effectively.

5. CONCLUSION.

I drew upon many aspects of the lab sessions for the creation of this MatLab function, and derived a logical progression from a basic FIR filter (as from week 2) to a working and functional low-pass Butterworth filter.

6. ADDITIONAL FILES

MatLab files: - delay_add_filter.m
- butterworth_filter.m
- butterworth_filter_script.m

Audio files: - Yeah.wav
- Yeah_filtered.wav

Diagrams: - Average magnitude spectrum x
- Average magnitude spectrum y
- Delay add filter
- FFT x
- FFT y

7. REFERENCES

[1] Digital Audio Effects, Edited by Udo Zölzer, John Wiley & Sons, 2002, ISBN: 0-471-49078-4.

[2] The scientist and engineer's guide to digital signal processing, Steven W. Smith, 1997.

[3] Digital Audio Effects, Edited by Udo Zölzer, John Wiley & Sons, 2002, ISBN: 0-471-49078-4.

[4] Digital Audio Effects, Edited by Udo Zölzer, John Wiley & Sons, 2002, ISBN: 0-471-49078-4

[5] Practical FIR filter Design in MatLab, revision1.0, Ricardo A Losada, The MathWorks, Inc. March 31, 2003.

[6] The scientist and engineer's guide to digital signal processing, Steven W. Smith. 1997

[7] Digital Audio Effects, Edited by Udo Zölzer, John Wiley & Sons, 2002, ISBN: 0-471-49078-4

[8] The scientist and engineer's guide to digital signal processing, Steven W. Smith. 1997

[9] The MathWorks, Inc. 1984-2012, Product Documentation, R2012a Documentation- Signal Processing Toolbox, <http://www.mathworks.com.au/help/toolbox/signal/ref/buttord.html;jsessionid=9d15614f8ed617ed81676ea44bed>