2015

High Precision Graphic Equaliser

Senior Honours Project

Supervisor: Dr Stefan Bilbao

WEBBER Jacob

University of Edinburgh

4/7/2015

Abstract

The following report describes the programming and theory behind the creation of a High Precision Graphic Equaliser that utilises a system of parallel digital filters to alter the frequency content of an audio signal according to user defined controls. The poles of these digital filters are fixed, while the numerator components of the summed parallel filter transfer function are jointly optimised to find a least squares match with the target magnitude response that is derived from the user controls. The method is based upon the work of Rämö et al and is described in their paper of the same name as this. The recreation of this work was successful, with highly accurate equalising achieved. Attempts to improve on this method, however, were not successful.

Declaration

I declare that this project and report are my own work.

Signed: Date:

Acknowledgements

I would like to thank my supervisor, Dr Stefan Bilbao, who was a tremendous help throughout, providing warm encouragement and assistance. I would also like to thank my Mummy, who assured me that she would still love me even if I failed my degree and who, along with my father, consistently tells me that I am clever. I am so grateful to you all.

Contents

Section I – Introduction

The following paper outlines the recreation of a High Precision Parallel Graphic Equaliser using the methods of digital filter design outlined in the paper by Rämö et al of the same name [1].
The aim of balancing frequency content in recorded sound predates even the use of electronics in sound recording. In the earliest analogue recordings the bass instruments were typically promoted to the front of the band, nearer the phonograph, to make up for the low sensitivity that primitive recording techniques had to lower frequencies [2].

Equalisers to this day are still used to improve the performance of systems that have frequency responses that are not flat. This does not just apply to recordings, equalisers can also correct non-flat frequency responses of speakers, headphones and amplifiers [3] [4].

Equalisers are not just used for correction, they are also used by musicians to modify sound output to create a range of effects - from the EQing done by dance music DJs, [3] to softening the sound on a jazz musician's electric guitar.

More fundamentally, a simple practical description of an audio equaliser is given by Hodgson in *Understanding Records,* where an audio equaliser is described as a series of "frequency-specific volume knobs" [5]. Where the frequency band assigned to each knob can be controlled then the equaliser is said to be parametric. The other case is dealt with in this paper, where the frequency bands at which the volumes are adjusted are fixed. This class of system is known as a graphic equaliser.

Graphic equalisers work by passing a signal through a number of filters that all affect different areas of the frequency spectrum. In an equaliser that works in cascade, the signal is passed through each of these filters in turn. In the case described in this paper, where a parallel configuration is used, the signal is passed through each of the filters separately. The outputs from these parallel filters are then summed together to get the outputted equalised signal.

Both these classes of system therefore require different ideal constituent filters. In the cascade system, the ideal filter affects the particular frequency it is there to adjust, leaving the others untouched. In the parallel system the ideal constituent filter would zero out all frequency content that was not within a narrow band around that filters centre frequency. Both these ideal filters are in practice impossible to realise and therefore there is some leakage into other bands. This results in inaccuracies in the frequency response of an equaliser.

The principles discussed thus far would apply equally to an equaliser with analogue or digital constituent filters. The method outlined here operates digitally, using a system of delay and feedback lines to change the frequency content of a signal. The frequency response of the total system can then be calculated and optimised using z-transform theory and matrix optimisation theory, specifically the Moore-Penrose pseudo-inverse [6].

It is worth noting at this point that, although equalising is the stated aim of the techniques exhibited in the paper by Rämö et al, the same filter design techniques could have many applications in other fields of audio technology. Indeed, the design of filters with accurate frequency responses is a field of audio technology in itself.

The methods outlined in the rest of this paper can broadly be split into four sections.

In the first a target magnitude and phase response for our parallel filters is computed. In the second section z-transform theory will be used to build a matrix that can be multiplied with a vector of our filter coefficients to give an output frequency response.

The third section uses the target response and previously built matrix to calculate a set of parallel filter coefficients that give a least squares match to the target frequency and magnitude response. The fourth and final section of methodology will address weighting, which was used to increase the accuracy of the equaliser.

The rest of the paper will analyse the results before the paper is concluded.

Section II – Methods

A. Target Computation

The following target computation was done in two parts. First a magnitude response was calculated, this being the magnitude adjustment of each frequency. Then the phase response was calculated, this being the phase delay on each frequency.

A graphic equaliser takes its user input in the form of a range of slider positions, with each slider corresponding to a particular frequency, spaced logarithmically, and each position corresponding to a magnitude in decibels. Once these slider positions have been inputted, the frequency response for frequencies that do not have sliders on them must be computed using a method of interpolation. In total the number of interpolated points which our equaliser is optimised to is ten times the number of command points. So, where P is the number of command points, in this case 31, the number of points optimised to will be $10P$. Here piecewise cubic interpolation is used. A comparison is made in fig. 1 between the pchip and spline functions in Matlab that could be used for this interpolation. The pchip function is shown to be more suitable as it is less prone to overshooting between command points.

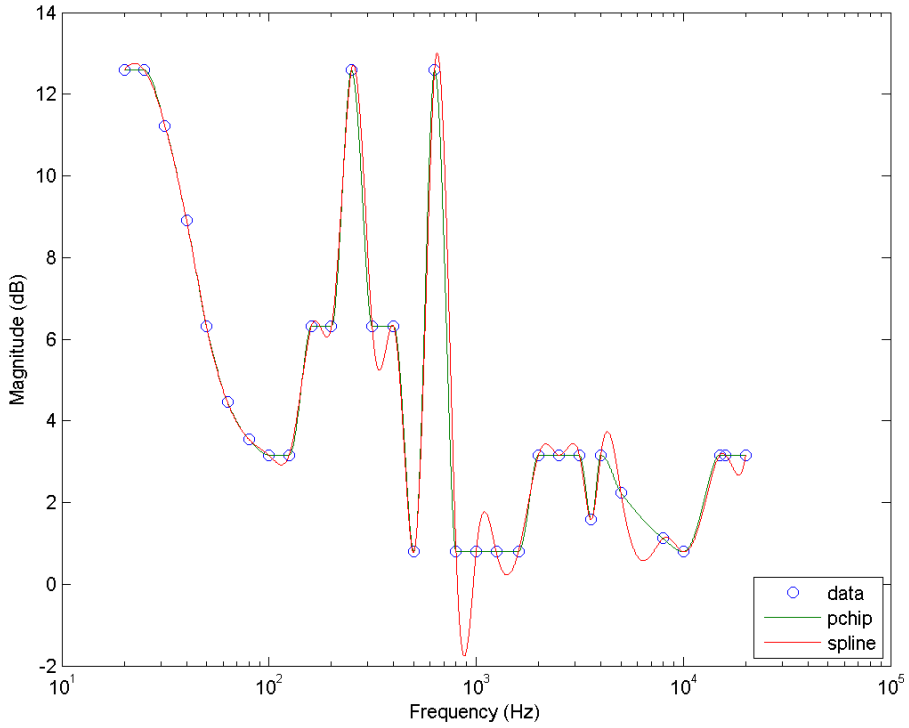


Figure 1 Comparison between pchip and spline functions.

It was not enough for the graphic equalizer to have just the correct magnitude response. It is conceivable that an array of delay and feedback lines could give the correct magnitude response, but completely garble the audio signal in the process. The aim of the graphic equaliser is to minimise any audible effect other than adjusting the frequency content. Minimum phase theory allows us to calculate the minimum phase delay for each frequency that is optimised to that it is possible while achieving a particular magnitude response at those frequencies. It can be thought of as the

theoretical lower limit of change to the phase content of a signal, while making a specific adjustment to the magnitude content.

Minimum phase was therefore seen as a natural choice. Analogue graphic equalisers also have a minimum phase response [7]. As minimum phase systems concentrate the energy near the start of the impulse response, the minimum phase solution is also easiest to optimise to with a parallel filter array of lower order, a lower order parallel filter array being one with fewer delay lines per filter.

This minimum phase target response of the system was computed using a Hilbert transform. For this to be done accurately, the magnitude response is resampled on a linear frequency scale with a high number of data points. This high number of data points was chosen to be 2^{15}. The choice of a number that is a power of 2 increases the efficiency of computing a Hilbert transform as the Hilbert transform is computed by taking a Fourier transform and zeroing out negative frequencies before performing an inverse Fourier transform. A number of data points that is a power of 2 allows for the fft and ifft functions in Matlab to operate inexpensively. The Hilbert transform is performed on the logarithm of the rescaled magnitude response.
Where we define our complete complex target, with phase and magnitude components to be:

$$H(z) = |H(z)|e^{j\arg[H(z)]} \tag{1}$$

Then where the complex logarithm of $H(z)$ is

$$\widehat{H}(z) = \log[H(z)] = log|H(z)| + j\arg[H(z)] \tag{2}$$

It can be shown that where the magnitude response of the system, here $|H(z)|$, is real, then for a causal, stable and real sequence, the minimum phase system would have $\log\left[H\left(e^{j\omega}\right)\right]$ and $\arg[H\left(e^{j\omega}\right)]$ as Hilbert transforms of each other [8]. Therefore a suitable phase response is computed by taking the imaginary part of the log of the Hilbert transform of the rescaled magnitude response. Once this has been found the phase response is scaled back down to a logarithmic scale with ten times as many data points as there are command points. Our total rescaled target transfer function can then be calculated trivially.

B. Fixed Poles Parallel Filter

Once a target transfer function has been computed, a set of parallel filters must be constructed that have a transfer function that matches this target as closely as possible. The novel approach proposed by B. Bank in 2007 [9] and applied to graphic equaliser design here [1] does this by combining an array of filters with fixed IIR components and with FIR components that have their coefficients optimised to match the target transfer function.
Digital filters work by combining a series of delay lines with a series of feedback lines. For a second order IIR filter the samples of the output, $y[n]$, are defined by the following equation:

$$y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] - a_1y[n-1] + a_2y[n-2] \tag{3}$$

Where $x[n]$ is the input signal.

Using Z transform theory, the transfer function of this system can be calculated. There are $N = 10P$ optimisation angular frequencies and K filters working in parallel. The transfer function is therefore given by:

$$H(Z) = b_0 + \sum_{k=1}^{K} \frac{b_{k,1} + b_{k,2}Z^{-1}}{1 + a_{k,1}Z^{-1} + a_{k,2}Z^{-2}} \tag{4}$$

For this equaliser the a coefficients, that being those on the denominator, are fixed, regardless of the target transfer function. This means that the complex values of Z, where the denominator is equal to zero, are fixed as well. These points are known as the poles of a filter, as they are where the transfer function tends towards infinity. As they must be preselected a sensible choice must be made here as to those pole positions. As demonstrated in the paper [1] and shown later in the results section, the best results were achieved when $K = 2P$. With K being the number of filters in parallel and therefore also the number of poles. Before a suitable choice of poles can be calculated, a connection between the transfer function and the audio output must be established. The transfer function $H(Z)$ exists in complex space. The frequency spectrum of the signal is altered by what $H(Z)$ is equal to on the unit circle of the complex/real plane. This is found where $Z = e^{i\omega_n}$.

$$H\left(e^{i\omega_n}\right) = b_0 + \sum_{k=1}^{K} \frac{b_{k,1} + b_{k,2}e^{-i\omega_n}}{1 + a_{k,1}e^{-i\omega_n} + a_{k,2}e^{-i2\omega_n}} \tag{5}$$

Each pole therefore has a corresponding frequency and radius, from which the a coefficients can be calculated according to the following equations:

$$\theta_k = \frac{2\pi f_k}{f_s} \tag{6}$$

This θ_k is the angle in the complex/real plane that corresponds to the pole frequency, f_k, and f_s is the sample frequency. The poles are therefore found at

$$p_k = e^{-\frac{\Delta\theta_k}{2}} e^{j\theta_k} \tag{7}$$

Where $\Delta\theta_k$ is the bandwidth defined to be:

$$\Delta\theta_k = \frac{\theta_{k+1} - \theta_{k-1}}{2} \tag{8}$$

With special cases for $\Delta\theta_{k=1}$ and $\Delta\theta_{k=K}$:

$$\Delta\theta_1 = \theta_2 - \theta_1 \tag{9}$$

$$\Delta\theta_K = \theta_K - \theta_{K-1} \tag{11}$$

The coefficients are then given by:

$$a_{k,1} = -2|p_k|\cos\theta_k \tag{12}$$

$$a_{k,2} = |p_k|^2 \tag{13}$$

It has been shown that, to get to these coefficients, we must have pole frequencies. The command frequencies themselves were obvious choices to make up half of these pole frequencies but the equaliser was shown to be most accurate when there were $K = 2P$ poles. The other pole frequencies were therefore spaced logarithmically between the command frequencies.
With the denominator coefficients calculated, a matrix must then be constructed that when multiplied with a column vector of the numerator coefficients:

$$\boldsymbol{p} = [b_0, b_{k,1}, b_{k,2}, \dots, b_{K,1}, b_{K,2}]^T \tag{14}$$

will yield the transfer function, $H(e^{i\omega_n})$, for our system. The first column in this matrix must be ones to account for the position of b_0 outside the fraction. The rows of the matrix, $\boldsymbol{M}$, will be vectors, with the n^{th} row corresponding to the angular frequencies, ω_n, that are optimised to. There are $N = 10P$ such angular frequencies which can be calculated trivially by:

$$\omega_n = \frac{2\pi f_n}{f_s} \tag{15}$$

We therefore see that the N rows of $\boldsymbol{M}$ take the form of the following vector:

$$\boldsymbol{M}_n = [\,1, {1}/{(1 + a_{1,1}e^{-i\omega_n} + a_{1,2}e^{-i2\omega_n})}, {e^{-i\omega_n}}/{(1 + a_{1,1}e^{-i\omega_n} + a_{1,2}e^{-i2\omega_n})}, \dots, {1}/{(1 + a_{K,1}e^{-i\omega_n} + a_{K,2}e^{-i2\omega_n})}, {e^{-i\omega_n}}/{(1 + a_{K,1}e^{-i\omega_n} + a_{K,2}e^{-i2\omega_n})}\,] \tag{16}$$

This $N \times (2K + 1)$ matrix would therefore give us our complex target transfer function, $\boldsymbol{H}_t$, in vector form, with an entry for every $10P$ frequencies that are optimised to if $\boldsymbol{M}$ were multiplied with a vector of ideal numerator coefficients, a vector that will be defined as $\boldsymbol{p}$.

$$\boldsymbol{H}_t = \boldsymbol{M}\boldsymbol{p} \tag{17}$$

Such a combination of ideal coefficients will often not exist. The description of how an optimum solution for these coefficients, $\boldsymbol{p}_{opt}$, can be found when $\boldsymbol{H}_t$ and $\boldsymbol{M}$ are known is discussed in the next section.

C. Matrix Optimisation

The matrix equation which was arrived at in Section II-C. was optimised according to the least squares solution to a system of linear equations using a Moore-Penrose pseudo-inverse [6]. This gives us a solution where $||\boldsymbol{M}\boldsymbol{p}_{opt} - \boldsymbol{H_t}||^2$ is minimised [10]. The optimal result is therefore achieved by the following equation:

$$\boldsymbol{p}_{opt} = \boldsymbol{M}^{+}\,\boldsymbol{H}_t \tag{18}$$

Where the Moore-Penrose pseudo inverse is $\boldsymbol{M}^{+}$, defined in [6] and [10] as:

$$\boldsymbol{M}^{+} = (\boldsymbol{M}^{H}\boldsymbol{M})^{-1}\boldsymbol{M}^{H} \tag{19}$$

The output transfer function $\boldsymbol{H}_{out}$ is then given by:

$$\boldsymbol{H}_{out} = \boldsymbol{M}\,\boldsymbol{p}_{opt} \tag{20}$$

The accuracy of the filter system can then be measured by comparing the output transfer function with the target transfer function. The magnitude of these functions gives the magnitude response for each frequency while the complex angle gives the phase delay for each frequency. While reasonably good matches are achieved between $\boldsymbol{H}_{out}$ and $\boldsymbol{H}_t$ using the methods described so far, there is one huge problem. The filter coefficients in $\boldsymbol{p}_{opt}$ will be complex. This means that the output signal from the filter will also be complex, and, as the human ear cannot discern imaginary pressure changes, this makes the filter coefficients derived by this method useless for an audio application. A method is therefore required that achieves optimal results while constraining the constituent values of $\boldsymbol{p}_{opt}$ to be real. Such a method was put forward by Balázs Bank in *Logarithmic frequency scale parallel filter design with complex and magnitude-only specifications*, [11] and is described here.

The real and imaginary parts of the matrix $\boldsymbol{M}$ and vector $\boldsymbol{H_t}$ are calculated and then horizontally concatenated as follows:

$$\boldsymbol{H}_{t,r} = \begin{pmatrix} Im\{\boldsymbol{H_t}\} \\ Re\{\boldsymbol{H_t}\} \end{pmatrix} \tag{21}$$

$$\boldsymbol{M}_r = \begin{pmatrix} Im\{\boldsymbol{M}\} \\ Re\{\boldsymbol{M}\} \end{pmatrix} \tag{22}$$

These concatenated versions therefore only have real components. When they are optimised to find a least squares solution for $\boldsymbol{p}_{opt}$ as defined by the following equation, then the real version of the Moore-Penrose method may be used:

$$\boldsymbol{p}_{opt} = \boldsymbol{M}_r^{+}\,\boldsymbol{H}_{t,r} \tag{23}$$

Where

$$\boldsymbol{M}_r^+ = (\boldsymbol{M}_r^T \boldsymbol{M}_r)^{-1} \boldsymbol{M}_r^T \tag{24}$$

Not only does this elegant method guarantee real values in $\boldsymbol{p}_{opt}$, while still optimally matching both the real and imaginary parts of our output and input transfer functions but it also reduces the number of floating point operations required to calculate this pseudo-inverse. This is because complex floating point multiplications are necessarily more expensive than multiplications of floating point numbers that exist solely in real or imaginary space.

It should be noted that, in this form, the pseudo-inverse is fixed whatever the input slider positions and therefore values of $\boldsymbol{H}_{t,r}$, are being used. This means that a change in slider positions would mean that only one matrix multiplication would have to be calculated in order to find a collection of coefficients, $\boldsymbol{p}_{opt}$, for a given slider configuration. This means that, once reasonable choices of pole positions and target frequencies have been made, the pseudo-inverse, $\boldsymbol{M}_r^+$, can be precomputed. This can be flashed to a DSP chip leaving only relatively inexpensive operations to be done for the equaliser to work effectively, even when the target transfer function is frequently changed. The method of weighting, as described below, increases the accuracy of the system at the expense of removing precomputing the pseudo-inverse as an option. With the weighting method, the pseudo-inverse is a function of the target transfer function and is therefore different for every configuration of slider positions.

D. Weighting

To increase the accuracy of the equaliser a system of weighting was implemented. This weighting compensates for an inadequacy of the least squares solution to a system of linear equations. By minimising absolute error, a least squares solution will always tend to minimise fractional error where the values of $H(\omega_n)$ are large at the expense of minimising fractional error when they are small. Optimising to reduce the fractional error is done by multiplying each column of $\boldsymbol{M}_r$ and element of $\boldsymbol{H}_{t,r}$ by $1/|H(\omega_n)|^2$, where ω_n is the relevant frequency for each row or element.

Doing this weighting results in fits that look much better but, as mentioned before, weighting in this way means that a pseudo-inverse must be calculated every time the slider positions are changed. This may be impossible to do in real time and would reduce the equaliser's utility in active automatic equalisation where the latency caused by this issue may render the parallel equaliser less useful than a cruder but less computationally expensive system.

Section III – Outcomes

A. Results

As has been previously discussed, an exact solution to certain sets of linear equations does not exist. In these cases a least squares solution is found. However, there are special cases where an exact solution to the system of linear equations does exist. The most obvious of these is where all the sliders are positioned at 0dB. In this case it can intuitively be seen that the ideal result is for $b_0 = 1$ and all other coefficients to equal 0. Using (3) we see that when these coefficients are chosen, then:

$$y[n] = x[n] \tag{25}$$

as required.

This system, where the solution is obvious, forms a good test for the equaliser. The results were encouraging.

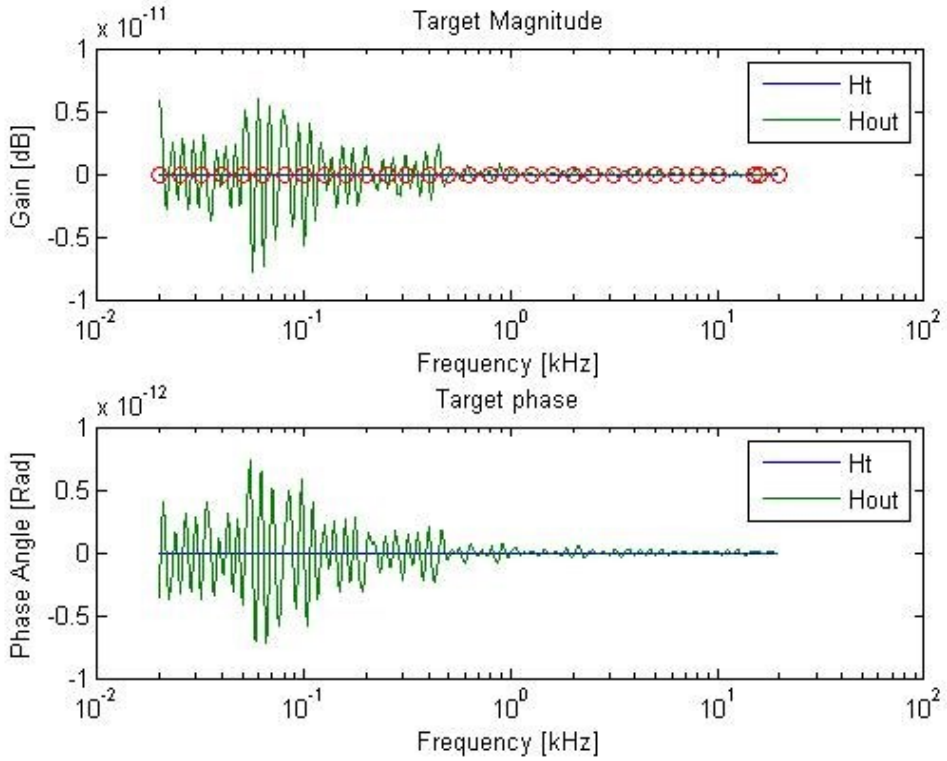


Figure 2 Magnitude and phase response of the equaliser

The magnitude and phase response, in dB and radians respectively, were in the order 10^{-12}, with the ideal being zero for both. The largest delay coefficient was of order 10^{-14}, with b_0 similarly only differing from 1 by the same level of machine inaccuracy. Almost identical results were achieved when all the sliders were raised or lowered by the same amount, the only difference being that b_0 was scaled accordingly. Whether weighting, as described in Section II-D, was used did not make a significant difference to the results in these cases.

A comparison between results achieved with and without this weighting is more easily done with a step configuration as in Figures 3 and 4.

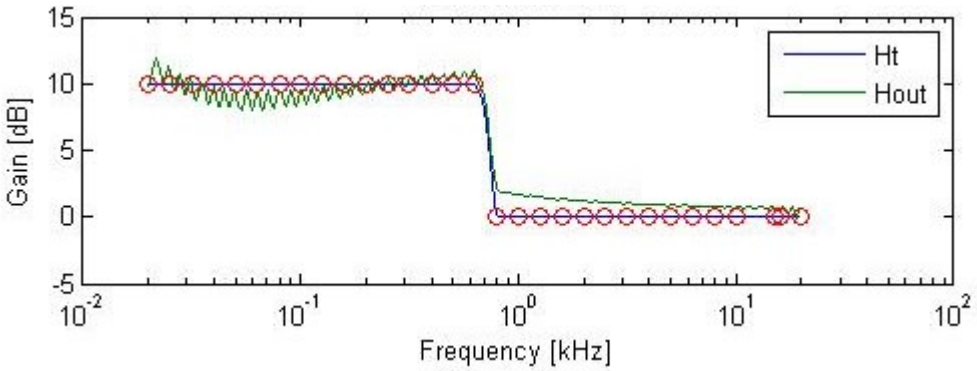


Figure 3 Magnitude response for step configuration without weighting. Red circles are slider positions.

The results of without weighting as shown in Figure 3 are clearly not as good a match as in Figure 4, where a system of weighting was implemented.

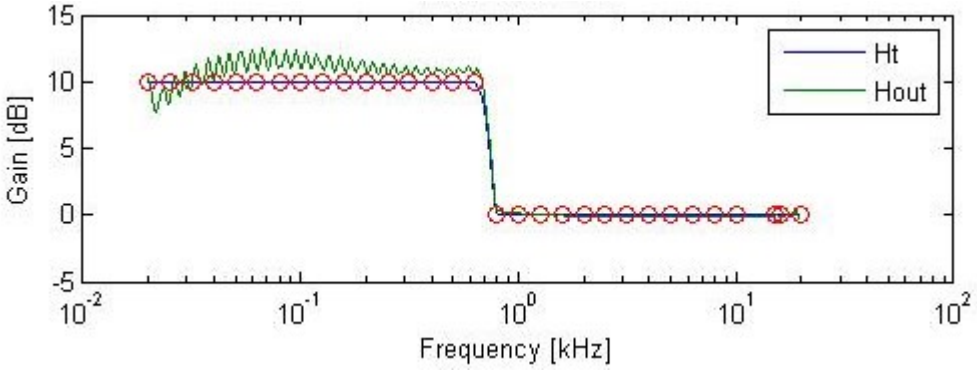
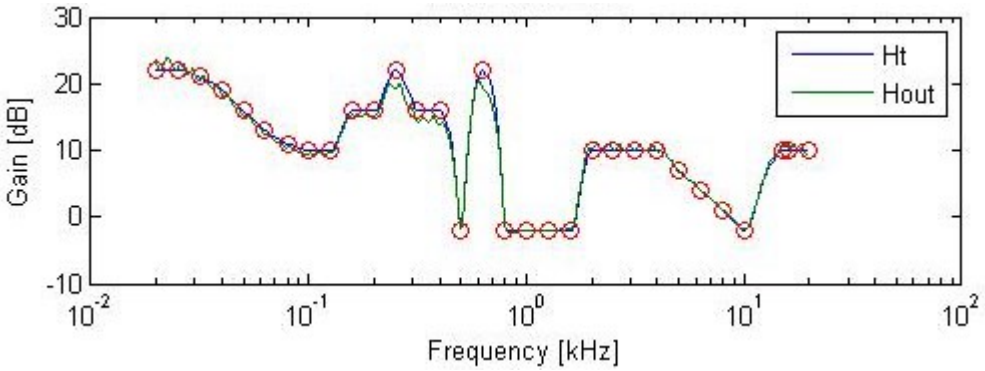


Figure 4 Magnitude response for step configuration with weighting. Red circles are slider positions.

As previously theorised, where weighting is not used, errors where the target response is low are under prioritised as the error magnitudes will be low, even when fractional error is high. The system with weighting would be considered more appropriate for most audio applications where the reduction of error as a ratio of the target magnitude would be seen as more sensible than the reduction of absolute error.

Results were then recorded with weighting for a variety of aggressive slider combinations.



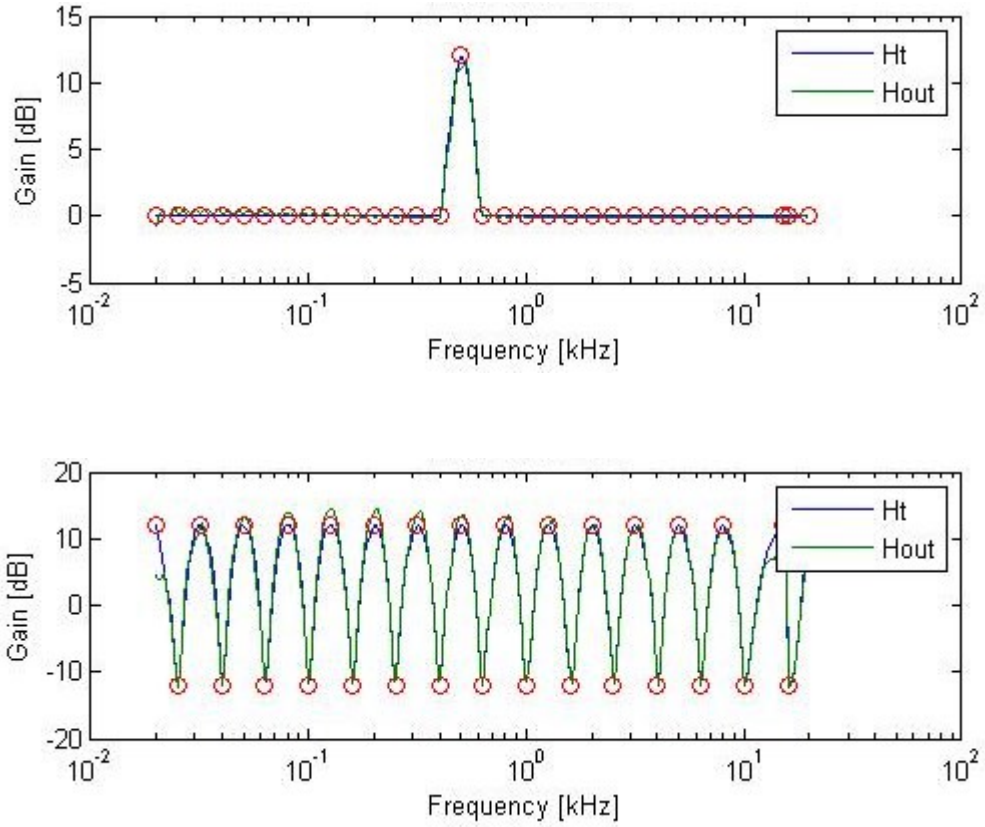


Figure 5, 6 and 7: Magnitude responses for aggressive slider positions

B. Evaluation

The following section evaluates the success of the project, including outlining abortive attempts to improve on the system designed by Rämö et al.

The fits achieved and demonstrated in section III-A were good, and showed the merits of this equaliser. They also confirmed that the system was superior with weighting implemented.

The problem of recalculating the pseudo-inverse for any change in slider position was one that was given significant thought. One theory that might have been used to reduce the computational complexity of doing this was the Sherman-Morrison formula. This formula allows a simplified calculation of the pseudo inverse to be done when a matrix, of which the pseudo-inverse is known, is only adjusted in a small number of columns. However, even when only one slider position is changed, there must realistically be at least 18 columns of the matrix adjusted as all the interpolated points either side of the slider position must also be adjusted. This means that this theory does not allow a significant reduction to the computational complexity of recalculating the inverse. The Sherman-Morrison method also introduces an error that accumulates every time it is applied. This effect is shown in Figure 6 and, although it is initially small, it would become significant over a period of seconds. Attempts this way to improve the methods speed using more advanced matrix optimisation techniques were therefore unsuccessful as they were both inefficient and added an unacceptable level of error.

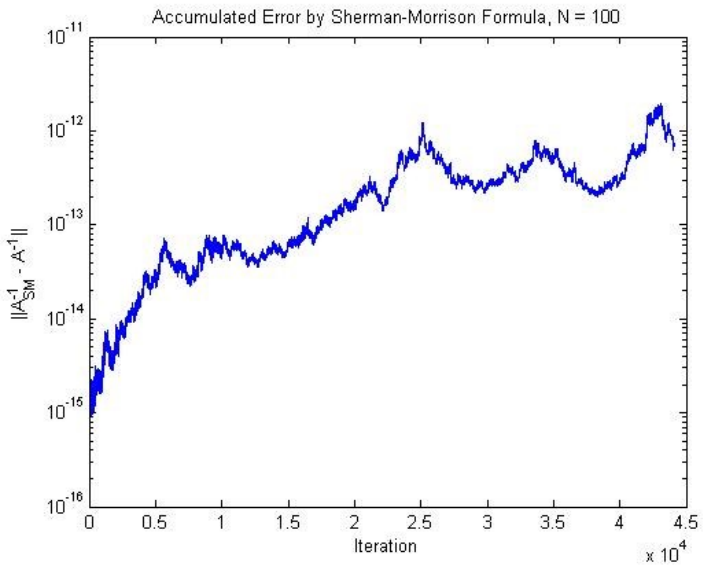


Figure 6 accumulated error in Sherman-Morrison Formula for $N \times N$ matrix

Another technique that was considered as a way of increasing the precision of the equaliser was the addition of an extra delay line. However it was noted that although this meant that a large amount more memory and processing power were required, no difference was made to the precision. One reason this may be the case is the optimisation to match a minimum phase delay, as discussed in Section II-A. This concentrates energy near the start of the impulse response of a system and means that a longer delay is not used.

It was seen, and has been demonstrated in this paper, that the methods followed in this report yield an extremely precise graphic equaliser. It should be noted, however, that the results that were achieved by Rämö et al were noticeably, albeit not significantly better. The reason for this is not clear but may well be because of the more sophisticated techniques that were used to calculate a target phase response. This is a matter that would have been explored more had time allowed.

It should be noted that designing an accurate parallel filter system has many useful applications beyond equalisation. An accurate filter might be used in room reverberation simulation. When sound reflects off surfaces, the frequency content is altered. Taking accurate frequency responses for reflections of different materials, the methods described here might be incorporated into a room reverberation model, such as the image source model. A filter with such a precise frequency response, but with relatively low computational complexity, might also be useful in physical modelling synthesis. It could be built in to a waveguide model of a wind instrument, modelling the effect that the bell has on the frequency content of the synthesised signal. Similarly, these methods of filter design might be used to emulate the frequency response of a violin body, after the signal at the bridge has been calculated using a finite difference scheme.

Section IV – Conclusion

In conclusion, a highly precise graphic equalizer was successfully built. The equalizer utilised an array of digital filters in built in parallel. The poles, and therefore the feedback coefficients, of these filters were fixed and the zeros, and therefore the delay coefficients, of these filters were jointly optimised to a target function that was designed based on the users choice of magnitude responses at a range of frequencies. The results were highly accurate for a range of trivial and non-trivial user inputted slider positions. There were two failed attempts at improving the system. One tried to speed up the matrix optimisation using the Sherman-Morrison formula, and the other aimed to increase the accuracy by adding an extra delay line and therefore more coefficients that can be optimised.

Bibliography

[1] V. V. a. B. B. Jussi Rämö, “High-Precision Parallel Graphic Equalizer,” *IEEE/ACM TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING,* vol. 22, no. 03, pp. 1894-1903, 2014.

[2] S. Schoenherr, “Recording Technology History,” 31 03 2005. [Online]. Available: http://www.aes.org/aeshc/docs/recording.technology.history/notes.html.

[3] G. S. John, in *Performance, Victor Turner and Contemporary Cultural Performance*, London, Berghahn Books, 2008, p. 167.

[4] R. Berkovitz, “Digital equalization of audio signals,” in *Proc. AES 1st int. conf.:Digital audio*, 1982.

[5] J. Hodgson, Understanding Records, London: Bloomsbury Academic, 2010.

[6] R. Penrose, “A generalized inverse for matrices,” in *Proceedings of the Cambridge Philosophical Society* , Cambridge, 1955.

[7] R. A. G. a. M. Schoessow, “ Design aspects of graphic equalizers,” *J. Audio Eng. Soc,* vol. 31, no. 6, pp. 394-407, 1983.

[8] O. a. Schaffer, Digital Signal Processing, New Jersey: Prentice-Hall, 1975.

[9] B. Bank, “Direct design of parallel second-order filters for instrument modelling,” in *Proc. Int. Computer Music Conf*, Copenhagen, 2007.

[10] M. Planitz, “Inconsistent systems of linear equations,” *Mathematical Gazette,* vol. 62, pp. 181-185, 1979.

[11] B. Bank, “Logarithmic frequency scale parallel filter design with complex and magnitude only components,” *IEEE Signal Process. Lett.,* vol. 18, no. 2, pp. 138-141, 2011.

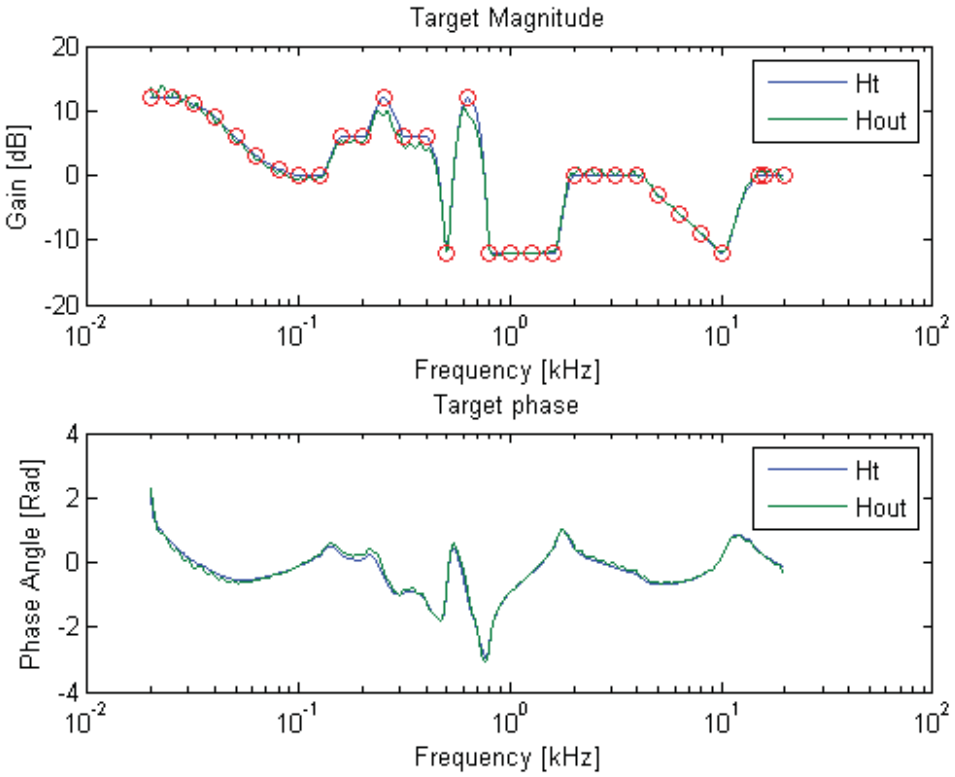
%APPENDIX A - CODE%

```
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~%
%------------------------------------------------------------------------------%
%----------------High Precision Parallel Graphic Equaliser---------------------%
%----------------------------by Jacob Webber-----------------------------------%
%------------------------------------------------------------------------------%
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~%
clear all
close all
%Variables
Fs = 44100;
Ts = 1/Fs;
% First it is necessary to calculate the pole positions.
% Pole frequencies are at the command frequencies and also at 10Hz and the
% band edges.


% Fi is command frequencies.
Fi = [20 25 31.5 40 50 63 80 100 125 160 200 250 315 400 500 630 800 ...
    1000 1250 1600 2000 2500 3150 4000 5000 6300 8000 ...
    10000 15000 16000 20000];

Gains = zeros(1,length(Fi));

Gains(1)=12;
Gains(2)=12;
Gains(3)=11;
Gains(4)=9;
Gains(5)=6;
Gains(6)=3;
Gains(7)=1;
Gains(10:14)=6;
Gains(12)=12;
Gains(15:20)=-12;
Gains(16)=12;
Gains(25)=-3;
Gains(26)=-6;
Gains(27)=-9;
Gains(28)=-12;


Gains = 10.^(Gains/20);


%Fk are my pole frequencies.
Fk = [20 22.4 25 28.2 31.5 35.5 40 44.7 50 56.2 63 70.8 80 89.1 100 ...
    112 125 141 160 178 200 224 250 282 315 355 400 447 ...
    500 562 630 708 800 891 1000 1120 1250 1410 1600 1780 2000 ...
    2240 2500 2820 3150 3550 4000 4470 5000 5620 6300 7080 8000 ...
    8910 10000 11200 12500 14100 16000 17800 20000];
% There are extra ones in between Fi values to give extra acc.
```

```
%Pole angle theta is calculated trivially.
thetak = (2*pi/Fs)*Fk;

% Pole distance derivation is dependent on Deltatheta
K = length(thetak);
deltathetak = zeros(1,K);
deltathetak(1) = thetak(2) - thetak(1);
deltathetak(end) = thetak(end) - thetak(end-1);

for k = 2:K-1;
    deltathetak(k) = 0.5*(thetak(k+1)-thetak(k-1));
end

% Now the pole length can be calc'ed.

Pk = exp(-0.5*deltathetak).*exp(thetak*i);

% Now we can calculate the coefficients of the transfer function.

A1 = -2*abs(Pk).*cos(thetak);
A2 = (abs(Pk)).^2;

% Now I must cunstruct the frequencies that our solution will be matched
% to. These are referred to as OMEGAn in the paper. They should be spaced
% logarithmically and there should be 10P of them where P is th number of
% command points.

loginc = (log10(max(Fi)) - log10(min(Fi)))/(10*length(Gains));
Fn = 10.^(log10(min(Fi)):loginc:log10(max(Fi))-loginc);
N = length(Fn);
% Omega is easily calculated from this.
OMEGAn = 2*pi*Fn;

% Must now construct M which is a (1+2*length(Fk))x(length(OMEGAn))
% Matrix.

M = zeros(N,2*K+1);
M(:,1) = 1;

invZ = exp(-OMEGAn*i*Ts);
for k = (1:K)*2
    for n = 1:N
        M(n,k) = 1/(1+A1(k/2)*invZ(n)+A2(k/2)*invZ(n)^2);
        M(n,k+1) = invZ(n)/(1+A1(k/2)*invZ(n)+A2(k/2)*invZ(n)^2);
    end
end


%-------------------------------------------------------------------------
% Constructing our target vector H
%-------------------------------------------------------------------------

% Interpolate target magnitude
```

```
TargMag = pchip(Fi,Gains,Fn);

%Rescale so that It is linear with 2^15 samples
reslin = (1/(2^15))*(max(Fi)-min(Fi));
Flin = (min(Fi):reslin:max(Fi)-reslin);
TargMagrescaled = pchip(Fn,TargMag,Flin);

logtargrescaled = imag( hilbert( log( TargMagrescaled )));

logTargPhase = pchip(Flin,logtargrescaled,Fn);

H = TargMag.*exp(i*logTargPhase);
H = H';


%------------------------------------------------------------------------
% Weighting
%------------------------------------------------------------------------

W = 1./abs(H).^2;
    siz = size(M);
    for k=1:siz(2),
        M(:,k)=M(:,k).*W;
    end;
    H=H.*W;


%------------------------------------------------------------------------

% Calculate Moore-Penrose inverse of M
Mr=[real(M); imag(M)];
Hr=[real(H); imag(H)];
T=Mr'*Mr;
b=Mr'*Hr;
Popt=T\b;

Hout = M*Popt;
B(:,1) = Popt(2:2:end);
B(:,2) = Popt(3:2:end);
A = [A1;A2];
d = Popt(1);


outGains = 20*log10(abs(Hout));
dBTargMag = 20*log10(TargMag);

%------------------------------------------------------------------------
%------------------------------------------------------------------------
% Plotting our target phase and our target Magnitude.
%------------------------------------------------------------------------
subplot(2,1,1);
semilogx(Fn/1000,dBTargMag,Fn/1000,-outGains,Fi/1000,20*log10(Gains),'o');
legend('Ht','Hout')
xlabel('Frequency [kHz]');
```

```
ylabel('Gain [dB]');
title('Target Magnitude');

subplot(2,1,2);
semilogx(Fn/1000,angle(H),Fn/1000,angle(Hout));
legend('Ht','Hout')
xlabel('Frequency [kHz]');
ylabel('Phase Angle [Rad]');
title('Target phase');
shg
%--------------------------------------------------------------------------
```



Published with MATLAB® R2014a