M-Band Graphic Equalizer

Prof. Ch.Srinivasa Kumar
Prof. and Head of department.
Electronics and communication
Nalanda Institute of technology
Guntur.,

Dr. P. Mallikarjuna Rao
Prof. , Andhra University.
Electronics and communicationVizag.

Abstract

This paper deals with the concept of filter banks. Filter banks are a group of band-pass filters connected in parallel. Each parallel connection forms a channel for different frequency-bands present in the input signal. The output of the filter bank is formed by merging these channels. The main theme behind the use of filter banks is to boost or attenuate different individual bands of frequencies present in a signal, without affecting other bands. There are several ways of realizing an filter bank and one such way is the Cosine Modulated filter bank. Instead of designing band-pass filters for different bands, the cosine modulated filter banks, modulates a low-pass filter. In simple words, it simply shifts the pass band of the low pass filter towards higher frequencies, and there by covering the entire band of frequencies of the input signal. Filter banks have many applications in the field of signal processing. Perhaps one of the biggest applications is Graphic Equalizer, where in, the user can 'graphically' modify different bands. The design and implementation of a filter bank as M-Band Graphic Equalizer is covered in this paper.

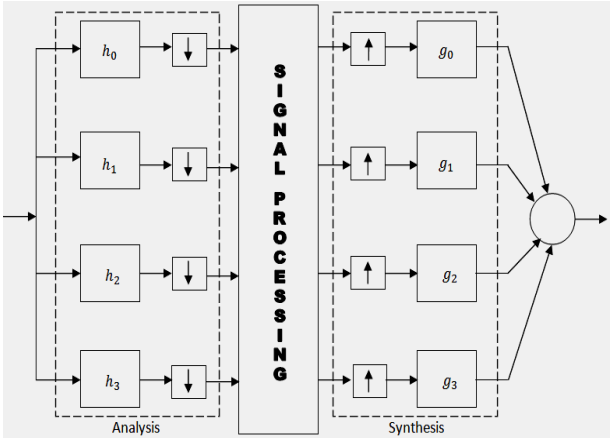
1. Introduction

Filter Bank

A filter bank is an array of band-pass filters (either FIR or IIR) that separates the input signal into multiple components, each one carrying single frequency sub-band of the original signal. These collection of filters are divided in two groups – analysis filters and the synthesis filters. At the analysis side, down-sampling is also done and at the synthesis side, up-sampling. This is illustrated with the help of the following figure.

Filter Bank Realizations: Cosine Modulated

There are many ways of constructing filter banks. Some of them are polyphase quadrature filtering (PQF), quadrature mirror filtering (QMF), etc. The method what is used in this paper is the cosine modulation. In this method, each filter of the analysis part has an impulse response of a cosine modulated version of a prototype low-pass filter with cutoff frequency at $F_s/_{2M}$ where M is the number of bands in the filter bank and F_s is the sampling frequency of the signal. The length of impulse response of each filter is $L = 2M$.



Block diagram of a 4-band filter bank

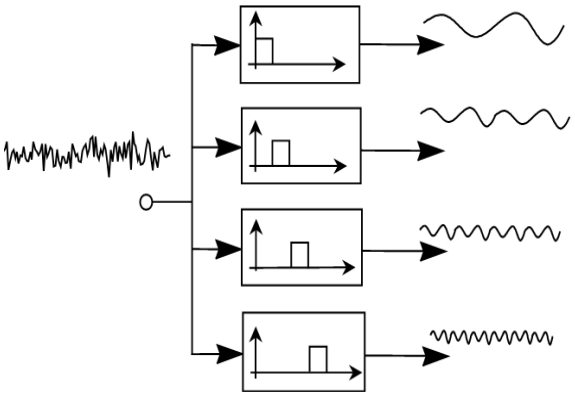


Illustration of the analysis part of a 4-band filter bank

Let h_{lp} be the impulse response of the prototype low-pass filter. Then the impulse response of each analysis filter is given by:

$$h_k(n) = 2h_{lp}\cos\left(\frac{2k+1}{4M(2n+1-M)\pi}\right)$$

where $k = 0, 1, 2, \ldots\ldots M \quad 1$

The synthesis filter responses are matched to that of the analysis filters. They are given by:

$$g_k(n) = h_k(L-1-n)$$

where $n = 0, 1, 2, \ldots\ldots L-1$

As, a result of cosine modulation, the total frequency bandwidth of the input signal is covered by all the bandwidths of all the analysis filters put together, with each filter having a bandwidth of $1/M$ of the total bandwidth of the input signal. The following figure illustrates this for the case of a 4 band filter bank.

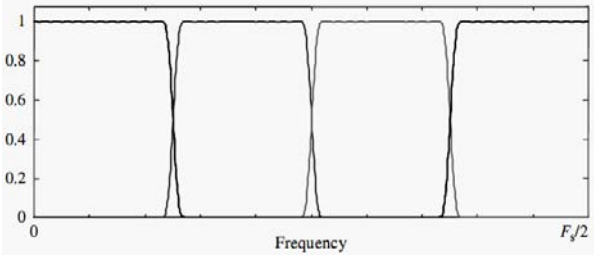


Figure depicting the cosine modulated version of the frequency bands

Non-Uniform Filter Banks: Octave Structured

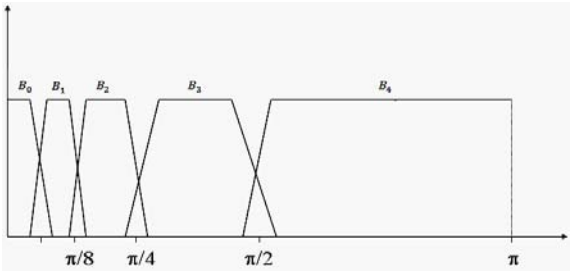
If the bandwidth of each filter in the filter bank is constant, that is, of the same size, then it is called *uniform structured* or simply, *uniform filter banks*. If the bandwidths differ for different filters, then we have a *non-uniform filter bank.*

Non-uniform filter banks are either fixed or dynamic depending on the input. Fixed non-uniform filter banks have bandwidths which are tree structured. Dynamic ones are having bandwidths that are adaptive depending on the type of input signal. Such filter banks can be used in real time applications. In this paper, the filter bank is uses an octave structure. In the octave structured filter bank, the bandwidth of the current filter is twice the size of the size of the filter below its order.

For example, h_0, h_1, h_2 and h_3 are the analysis filters, then bandwidth of h_2 is twice that of h_1 and bandwidth of h_3 is twice that of h_2, while, h_0, and h_1 have the same bandwidth. An octave structured filter bank is a suitable choice for a graphic equalizer. In fact, the octave bands can be formed by merging the uniform bandwidths. If, K is the number of uniform bands, then the number of non-uniform octave bands M can be represented as:

$$M = 1 + \log_2 K$$

If b_0, b_1, b_2 and b_3 are the uniform sub bands, then a 3 band non-uniform sub bands can be formed as $B_0 = b_0, B_1 = b_1$ and B_2 = combination b_2 and b_3. Since the uniform filter-bank consists of a parallel connection of the K sub-band filter, the impulse responses, merging bands is easily accomplished by simply adding together the respective impulse responses. The octave structured sub-bands for a 5 band octave structured filter bank is as shown in the figure.



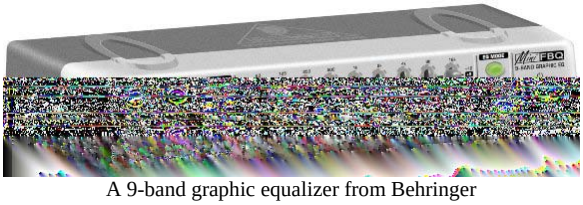
Octave structured sub-bands for 5-band filter bank

Applications of Filter Banks: Graphic Equalizer

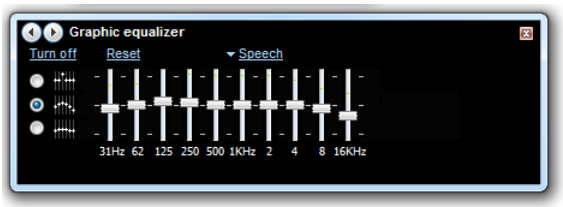
Filter banks have wide number of applications. One of the traditional applications of the filter banks is graphic equalizer which attenuate bands specified by the user. Other applications include, sub-band coding of the speech or music signals, data compressions, spectral analysis of signals, Auto-Regressive Moving Average (ARMA) modeling and many more.

Graphic Equalizers

Graphic Equalizers can an instrument or an software, that can attenuate different bands of frequencies based on the user input. The graphic equalizers used in music industry or recording companies are the M-band octave equalizer. The user can simply turn the 'knobs' that either boosts the gain or attenuate the selected band. Depending on the number of bands, the hardware requirements and the complexity increases. There are graphic equalizers for even 30+ bands. Now-a-days, graphic equalizations have become a common feature of every media player software.



A 9-band graphic equalizer from Behringer



A 10-band graphic equalizer of Microsoft Windows Media Player

2. Design of M-Band Octave Structured Graphic Equalizer

Filter design

Filters are the core parts of the Graphic Equalizer. Hence its design is an important consideration as it affects the overall quality. The filters used are FIR Low Pass filters, which are cosine modulated. This FIR Low Pass filters are linearly phased, and hence, they have the filter coefficients which are usually symmetrical. The ideal Low Pass filter with cutoff frequency f_c Hz has the following impulse response.

$$h_{ideal} = 2f_cT\,sinc\left(2f_cT\left(n - \frac{L-1}{2}\right)\right)$$

Where the filter is delayed by $L-1/2$ samples. Here L is the length of the filter and T is the sampling period. This impulse response is clearly non-causal and infinite in duration. Hence, we have to multiply this one with a window function. Doing so will restrict the number of samples to the length of the window. The first obvious choice of window function would be a rectangular window as show below.

$$w(n) = \begin{cases} 1, & n = 0, 1, 2 \ldots\ldots M-1 \\ 0, & otherwise \end{cases}$$

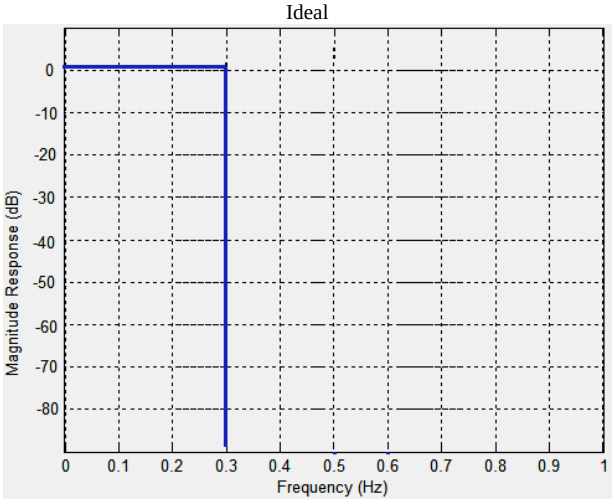
But the infamous rectangular window is known for its side lobes to increase as the length of the window increases. This is mainly because of the abrupt truncation or in other words, abrupt change in the values. Hence, we need to use windows whose values gradually change from 1 to 0, from both sides. There are several windows and each have their merits and demerits. The window that is chosen in this paper is Dolph – Chebyshev window. Its equation is given as below.

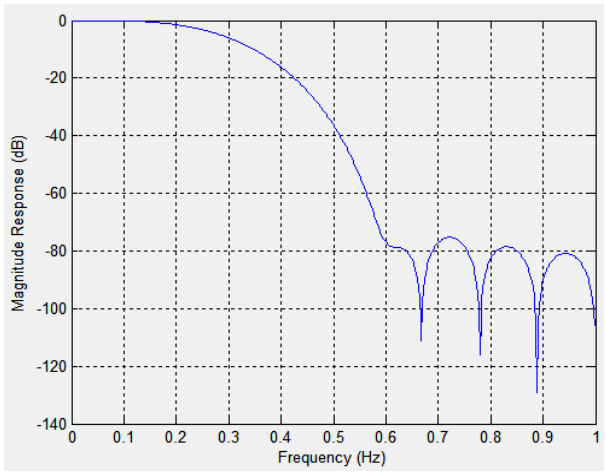
$$\hat{W}(k) = (-1)^k \frac{\cos[N\cos^{-1}[\beta\cos(\pi k / N)]]}{\cosh[N\cosh^{-1}(\beta)]} \quad 0 \le k \le N-1$$

where, $\beta = \cos[1/N\cosh^{-1}(10^{\alpha})]$ and N is the length of the sequence

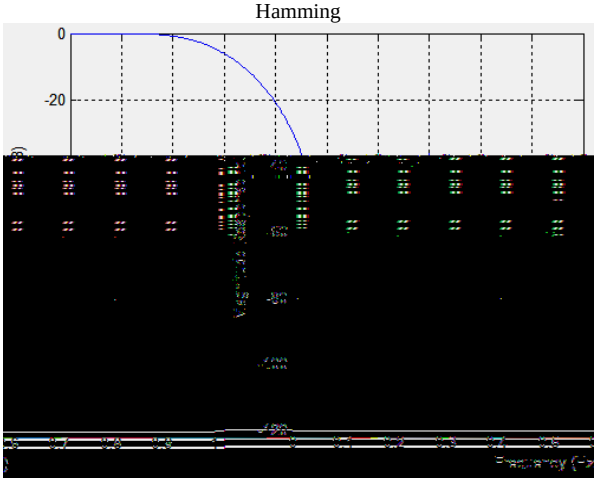
The reason for this window being used is that the level of the side lobes can be controlled by the parameter α. Suppose α = 5, then there will be an attenuation of 100dB on the side lobes. There is always a tradeoff between the size of the main lobe and the attenuation of side lobes. If the attenuation of the side lobe is very large, then there will be an increase in the main lobe, and thus deviating from the ideal characteristics.

The length of the filter also has an effect on the widow characteristics. If the length is small, then the side lobe would be large enough, deviating hugely from the ideal characteristics. The below figures depict this. The comparison of Hamming, Blackman and Dolph-Chebyshev windows in the design of a low-pass filter with cut-off frequency and 0.3 (normalized frequency) and length 20 is shown below.

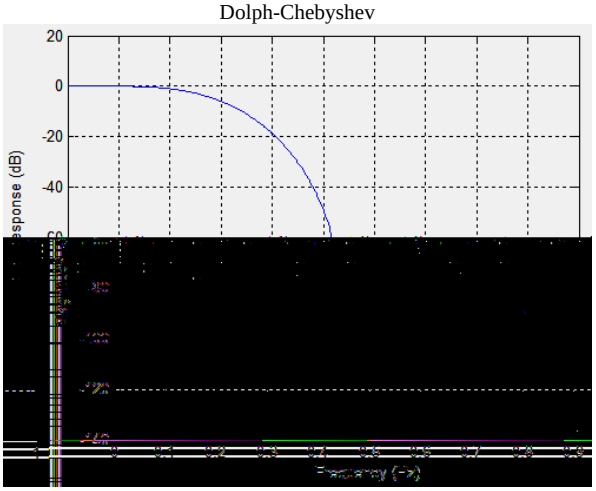




Hamming



Dolph-Chebyshev



As one can observe that length and attenuation of the side lobes have an effect of the width of the main lobe. Since, the attenuation factor of the windows functions like Blackman and Hamming are fixed, the width of their main lobes are decided by their lengths. Since Hamming window has the side lobes restricted to -41dB, it has the smaller main lobe. Blackman attenuates side lobes to -57dB, and hence it has a larger main lobe.

Dolph-Chebyshev, on the other hand has main lobe is slightly more than that of Hamming with the side lobes suppressed to a level of -60dB. If the lengths get smaller, then the main lobe size of the respective windows increases, but by controlling the level attenuation level of Dolph-Chebyshev window, we can get a somewhat closer approximation of the main lobe to the pass band of the ideal low pass filter. That is, if for smaller lengths, we can decrease the attenuation level of the side lobes under some 'tolerable' limits, such that the height of the side lobes cannot be more than this limit. This can be used for the cases of the 3, 4, 5 and 6 band filter banks, where in the lengths ($L = 2M$) of each analysis filters are 6, 8 and 10 respectively.

The Matlab function for the generating the dolph window sequence is CHEBWIN(N,R), where N is the length of the sequence to be generated and R is the decibels of relative side lobe attenuation. In this paper, the functions is implemented as CHEBWIN(L, 3*L), where L is the length of the filter sequence and the window sequence. The attenuation factor R is a multiple of the length L and thereby making it dynamic. The scalar that is multiplied is decided empirically. Hence we get smaller main lobe compared to other windows even for smaller L, but at the cost of increase in the magnitude of the side lobes, which are under some tolerable levels, depending on the application. But for larger lengths, the attenuation goes high. For example, in the case of a 5 band octave filter bank, we get length of 128 and we get the attenuation factor as R = 3 x 128 = 384dB. This causes the main lobe to increase drastically, causing undesirable results. Hence, we restrict the attenuation to a maximum of 100dB. The filter coefficients are determined in Matlab by using the function fir1(.).

Analysis Filters

Analysis filters are formed basically using a cosine modulated low pass filter, designed using dolph window function. Multiplying the filter coefficients with the cosine function described previously, decided the shift of the pass band of the low pass filter. This can be easily performed in Matlab by multiplying the low pass filter sequence with the sequence formed by the cosine function. The bandwidth of each analysis filter is formed

using the octave structuring. In Matlab, octave bands are formed by according adding the impulse responses or filter sequences of 2^{M-1} filters of the 2^{M-1} band uniform filter bank.

Synthesis Filters

The synthesis filters have impulse response that are matched with the respective analysis filters, but of opposite phase, so that we can get a perfect reconstruction of the input signal. In Matlab, the order of the filter coefficients of each analysis filter is reversed and then assigned to the respective synthesis filters.

Gain Control

The output sequence from each of the synthesis filters is multiplied with a scalar, which specifies the gain. In Matlab, The values of the gain for respective channels is entered in to a gain vector during the execution of the program.

3. The Matlab Code

The Matlab code is written for both uniform and non-uniform octave M-band graphic equalizer.

Uniform M-Band Graphic Equalizer

mbandge.m

```
function mbandge(file, M, gain);

fprintf('Processing...');

%Read the input WAV file
[xi, Fs] = wavread(file);
x = transpose(xi);
lx = length(x);

%//////////////////////////FIR filter design//////////////////////////
L = 2*M;
%Cutoff frequency
fc = Fs/(2*M);
%Normalized cutoff frequencies
fcn = fc/(Fs/2);
if L > 100
 slripple = 100;
else
 slripple = 2*L;
end
hlp = fir1(L-1, fcn, chebwin(L, slripple));
%////////////////////////////////////////////////////////////////////
%///////////////////////Cosine modulated filter bank/////////////////
%Analysis filters
for ka = 1:M
 for n = 1:L
 h(ka, n) = 2*hlp(n)*cos(((2*(ka-1) + 1)/(4*M))*(2*(n-1) + 1 - M)*pi);
 end
end
%Synthesis filters
for ks = 1:M
 g(ks, 1:L) = h(ks, L:-1:1);
end
%DC gain estimation
sumc = [zeros(1, 2*L - 1)]; %Initial
for kd = 1:M
 sumc = sumc + conv(h(kd, 1:L), g(kd, 1:L));
end
a = sum(sumc);
%Output of the analysis filters
for kw = 1:M
 w(kw, 1:(lx + L - 1)) = conv(x, h(kw, 1:L));
end
%Output of the synthesis filters
```

```
for kv = 1:M
 v(kv, 1:(lx + 2*L - 2)) = conv(w(kv, 1:(lx + L - 1)), g(kv, 1:L));
end

%Gain adjustment
for ky = 1:M
 y(ky, 1:length(v(ky, 1:(lx + 2*L - 2)))) = gain(ky)*v(ky, 1:(lx + 2*L - 2));
end
%//////////////////////////////////////////////////////////////////
fprintf('Done\n');
%//////////////////////////////////Output//////////////////////////////
Y = sum(y);
%Removing DC gain
Y = Y/a;
%//////////////////////////////////////////////////////////////////
%////////////////////////Playing Output//////////////////////////////
fprintf('Playing...');
wavplay(Y, Fs);
fprintf('Done\n');
wavwrite(Y, Fs, 'dolp');
%//////////////////////////////////////////////////////////////////
%//////////////////PLOTTING: ONLY FOR 5 Band Filter Bank////////////////
if M == 5
 H1 = freqz(h(1, 1:L), 1);
 mag1 = 20*log10(abs(H1));
 H2 = freqz(h(2, 1:L), 1);
 mag2 = 20*log10(abs(H2));
 H3 = freqz(h(3, 1:L), 1);
 mag3 = 20*log10(abs(H3));
 H4 = freqz(h(4, 1:L), 1);
 mag4 = 20*log10(abs(H4));
 H5 = freqz(h(5, 1:L), 1);
 mag5 = 20*log10(abs(H5));
 q = 0:1/length(mag1):1 - 1/length(mag1);
 plot(q, mag1, q, mag2, q, mag3, q, mag4, q, mag5), grid on;
 xlabel('Frequency (Hz)');
 ylabel('Magnitude Response (dB)');
end
%//////////////////////////////////////////////////////////////////
```

Non-Uniform M-Band Octave Graphic Equalizer

This function has a special input when the M is entered as 8. If 0 is entered for the gain vector for M = 8, then some common presets are presented. The user simply has to type in the preset exactly and press enter.

moctbandge.m

```
function moctbandge(file, M, sgain);

if (length(sgain) == 1) && (M == 8)
 if sgain == 0
 fprintf('Presets Available: 1.treble 2.bass 3.full treble 4.full bass\n ');
 preset = input('Type a preset: ','s');
 switch preset
 case 'treble'
 gain = [0.5, 0.5, 0.5, 0.5, 0.5, 1.5, 2, 3];
 case 'bass'
 gain = [3, 2, 1.5, 0.5, 0.5, 0.5, 0.5, 0.5];
 case 'full treble'
 gain = [0.1, 0.1, 0.1, 0.1, 0.5, 0.5, 2, 3];
 case 'full bass'
 gain = [3, 2, 0.5, 0.5, 0.1, 0.1, 0.1, 0.1];
 end
 else
```

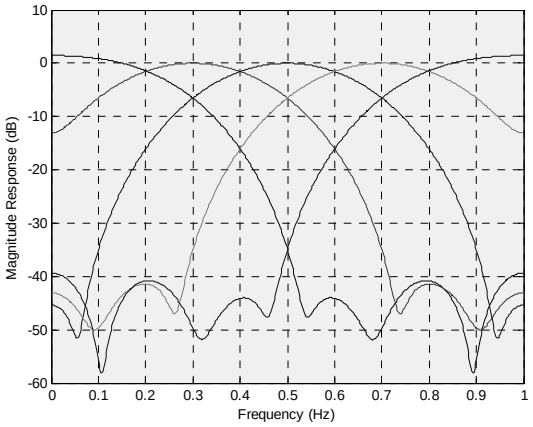
```
 error('Invalid input');
 end
else
 if length(sgain) == M
 gain = sgain;
 else
 if (length(M) ~= 8) && (sgain == 0)
 error('Sorry. Presets available only for 8 band');
 else
 error('Invalid input');
 end
 end
end

fprintf('Processing...');

%Read the input WAV file
[xi, Fs] = wavread(file);
x = transpose(xi);
lx = length(x);
%////////////////////////FIR filter design//////////////////////////
K = 2^(M-1);
L = 2*K;
%Cutoff frequency
fc = Fs/(2*K);
%Normalized cutoff frequencies
fcn = fc/(Fs/2);
if L > 100
 slripple = 100;
else
 slripple = 2*L;
end
bhlp = fir1(L-1, fcn, chebwin(L, slripple));
%////////////////////////////////////////////////////////////////////
%//////////////////////Cosine modulated Octave filter bank/////////////////
%Analysis filters
for p = 1:K
 for kb = 1:L
 b(p, kb) = 2*bhlp(kb)*cos(((2*(p-1) + 1)/(4*K))*(2*(kb-1) + 1 - K)*pi);
 end
end

i1 = 3;
for ka = 0:M-1
 if ka < 2
 h(ka+1, 1:L) = b(ka+1, 1:L);

 else
 i2 = i1 + 2^(ka-1) - 1;
 h(ka+1, 1:L) = sum(b(i1:i2, 1:L));
 i1 = i2 + 1;
 end
end
%Synthesis filters
for ks = 1:M
 g(ks, 1:L) = h(ks, L:-1:1);
end
%DC gain estimation
sumc = [zeros(1, 2*L - 1)]; %Initial
for kd = 1:M
```

```
 sumc = sumc + conv(h(kd, 1:L), g(kd, 1:L));
end
a = sum(sumc);
%Output of the analysis filters
for kw = 1:M
 w(kw, 1:(lx + L - 1)) = conv(x, h(kw, 1:L));
end
%Output of the synthesis filters
for kv = 1:M
 v(kv, 1:(lx + 2*L - 2)) = conv(w(kv, 1:(lx + L - 1)), g(kv, 1:L));
end
%Gain adjustment
for ky = 1:M
 y(ky, 1:length(v(ky, 1:(lx + 2*L - 2)))) = gain(ky)*v(ky, 1:(lx + 2*L - 2));
end
%////////////////////////////////////////////////////////////////
%//////////////////////////////////Output/////////////////////////////
Y = sum(y);
%Removing DC gain
Y = Y/a;
%////////////////////////////////////////////////////////////////
fprintf('Done\n');
%//////////////////////////Playing Output/////////////////////////////
fprintf('Playing...');
wavplay(Y, Fs);
fprintf('Done\n');
%//////////////////PLOTTING: ONLY FOR 5 Band Filter Bank////////////////
if M == 5
 H1 = freqz(h(1, 1:L), 1);
 mag1 = 20*log10(abs(H1));
 H2 = freqz(h(2, 1:L), 1);
 mag2 = 20*log10(abs(H2));
 H3 = freqz(h(3, 1:L), 1);
 mag3 = 20*log10(abs(H3));
 H4 = freqz(h(4, 1:L), 1);
 mag4 = 20*log10(abs(H4));
 H5 = freqz(h(5, 1:L), 1);
 mag5 = 20*log10(abs(H5));
 q = 0:1/length(mag1):1 - 1/length(mag1);
 plot(q, mag1, q, mag2, q, mag3, q, mag4, q, mag5), grid on;
 xlabel('Frequency (Hz)');
 ylabel('Magnitude Response (dB)');
end
```

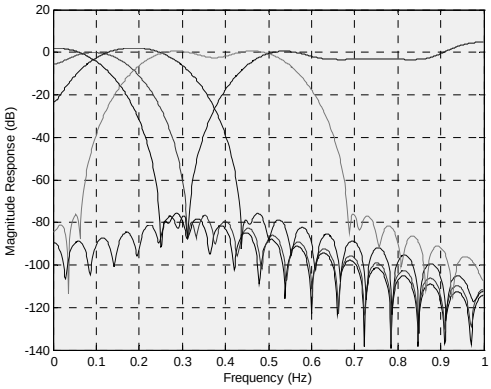
Commands:

```
>> mbandge('Audio2.wav', 5, [0,1,0,0,0,0])
Processing...Done
Playing...Done
>> moctbandge('Audio4.wav', 5, [3,0,0,0,0])
Processing...Done
Playing...Done
```

4. Plots



M-Band Graphic Equalizer



M-Band Octave Structural Graphic Equalizer

5. Conclusion

In this paper we discussed the basic concept of filter banks. There are several ways of realizing a filter bank and one such way is the Cosine Modulated filter bank. We observed the following points.

1. Instead of designing band-pass filters for different bands, the cosine modulated filter banks, modulates a low-pass filter.
2. In simple words, it simply shifts the pass band of the low pass filter towards higher frequencies, and there by covering the entire band of frequencies of the input signal.
3. Filter banks have many applications in the field of signal processing. Perhaps one of the biggest applications is Graphic Equalizer, where in, the user can 'graphically' modify different bands.
4. The design and implementation of a filter bank as M-Band Graphic Equalizer is covered in this paper.

6. References

[1] L.R. Rabiner and B.H. Juang, Fundamentals of Speech Recognition, Prentice-Hall, Englewood Cliffs, N.J., 1993.
[2] L.R Rabiner and R.W. Schafer, Digital Processing of Speech Signals, Prentice-Hall, Englewood Cliffs, N.J., 1978.
[3] José Ramón Calvo de Lara, "A Method of Automatic Speaker Recognition Using Cepstral Features and Vectorial Quantization"10th Iberoamerican Conference on Pattern Recognition, CIARP 2005 Proceedings
[4] S. Furui, "An overview of speaker recognition technology", ESCA Workshop on Automatic Speaker Recognition, Identification and Verification, pp. 1-9, 1994.
[5] F.K. Song, A.E. Rosenberg and B.H. Juang, "A vector quantisation approach to speaker recognition", AT&T Technical Journal, Vol. 66-2, pp. 14-26, March 1987.
[6] Vijay K. Madisetti and Douglas B. Williams, Digital Signal Processing Handbook, Capman & Hall CRCnetBase.
[7] "ECE341 So You Want to Try and do Speech Recognition." A Simple Speech Recognition Algorithm. http://www.eecg.toronto.edu/~aamodt/ece341/speech-recognition
[8] The Mathworks – MATLAB and SIMULINK for Technical Computing. 10 June 2005. http://www.mathworks.com
[9] http://www.vision.caltech.edu/CNS248/Speech/speech95_1.ps.gz.
[10] Spoken Language Input http://cslu.cse.ogi.edu/HLTsurvey/ch1node1.html