

Lab Report 2 – 3-Band Multiband Compressor implementation in MATLAB

Tom Brickhill S/N: 420088651

Digital Audio Systems, DESC9115, Semester 1 2013
Graduate Program in Audio and Acoustics
Faculty of Architecture, Design and Planning, The University of Sydney

1. Introduction

When setting out to build a digital multiband compressor there are a number of different components and concerns to take into account. The purpose of this particular implementation is to focus on use for vocal content. Multiband compression is widely used in broadcast, studio and live audio engineering for numerous reasons. When it comes to vocal content there are a few reasons one might use such a device. Some people may have a particularly strong resonant frequency (formant) that becomes more prominent as they increase the volume at which they're speaking. A multiband compressor could be used to target that formant and reduce it's presence. It could also be used to increase sillabance without increasing hiss as much as a shelving EQ and decrease 'mud' without the 'gutting' feeling of a bell EQ.

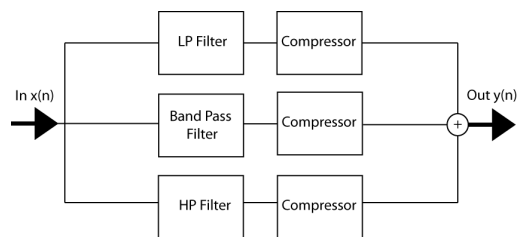


Fig. 1

In this implementation I have chosen to create a 3-band multiband compressor. The block diagram of the operation is shown in fig. 1. At first I intended for the crossover points to be adjustable, but due to the complications experienced during the filter design process in trying to keep the filters stable as the crossover points are moved, I decided to settle with fixed crossover points of 200 Hz and 2000 Hz. I have chosen these points because it splits the signal into bands that are tonally useful to alter for vocal content. These are also points where the

filters are the most stable and only cause minimal ripple at the crossover points.

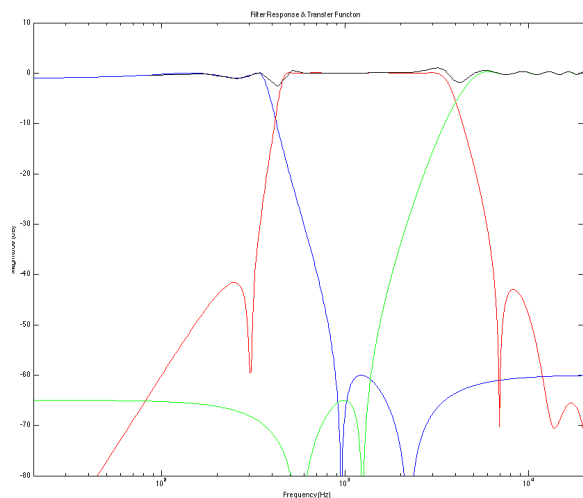


Fig. 2

2. Filter Design

Figure 2 shows the frequency response of each filter. Blue being lowpass, red as bandpass and green as the highpass. The black line is the transfer function of the system when the three band limited signals are combined back together after the filters. The stimulus signal is an exponential swept sine wave from 20 Hz to 20 KHz. It can be observed from this figure that there is a small amount of ripple at the crossover points between filters. This amounts ripple at it's maximum is about 2dB. After carrying out some critical listening of vocal and musical content I can safely say that there is little to no discernable colouration of the sound caused by these ripples. After experimenting with a number of design methods within MATLAB I decided to utilise the fdesign toolbox to design the filters:

```
D1 = fdesign.lowpass()
D2 = fdesign.bandpass()
D3 = fdesign.highpass()
```

These functions allow the programmer to input the various parameters they desire the filter to fulfill and then attempts to build the filter within those parameters. The filtering method is then chosen and “designed”:

```
design(D1, 'IIR')
design(D2, 'IIR')
design(D3, 'FIR')
```

In this case I have chosen a combination of IIR and FIR filters. IIR for the lowpass and bandpass. FIR for the highpass. This came down to experimentation with different parameters to get the most stable filters and flattest magnitude response. One of the most sensitive parameters for the stability of the filters is the distance between the stopband and passband frequencies. The closer they are together the less likely the filter will be stable.

3. Compression

Utilising a compression script developed from (ZOLZER, Udo, 2008) (ZOLZER, Udo, 2002) the “mycomp.m” script was developed. This compression method uses an RMS detector method defined in equation 1.

$$x_{RMS}^2(n) = (1 - TAV) \cdot x_{RMS}^2(n - 1) + TAV \cdot x^2(n)$$

Eq. 1

The time averaging component ‘TAV’ is defined in equation 2 where ‘ t_M ’ is the averaging time in milliseconds and ‘ T_A ’ is the Nyquist.

$$TAV = 1 - \exp\left(\frac{-2.2T_A}{t_M/1000}\right)$$

Eq. 2

The control function $f(n)$ for a compressor is defined in equation 3 where ‘CS’ is the control slope, ‘CT’ is the compression threshold and ‘CR’ is the compression ratio.

$$f_{dB}(n) = CS(CT - RMS_{dB}(n))$$

where

$$CS = 1 - \left(\frac{1}{CR}\right)$$

Eq. 3

The attack and release times for the compression are defined in equation 4 where ‘at’ and ‘rt’ are the attack and release times in milliseconds.

$$AT = 1 - e^{\frac{-2.2 \times T_A}{at/1000}}$$

$$RT = 1 - e^{\frac{-2.2 \times T_A}{rt/1000}}$$

Eq. 4

The attack and release are applied to $f(n)$ to create the smoothed gain function $g(n)$. This happens after $f(n)$ has been converted back to magnitude from dB. See equation 5.

$$\text{If } f(n) > 0,$$

$$g(n) = (1 - AT) \times g(n - 1) + AT \times f(n)$$

$$\text{If } f(n) < 0,$$

$$g(n) = (1 - RT) \times g(n - 1) + RT \times f(n)$$

Eq. 5

Finally the smoothed gain function is applied to the input signal to give the compressed signal. See equation 6

$$y(n) = x(n) \cdot g(n)$$

Eq. 6

4. Performance and Use

The use of a swept sine wave is useful for testing the performance of the filters in the system however because the compression elements of the system are non-linear in nature a swept sine wave is useless in assessing the performance. The easiest way is to use some actual program material and partake in some critical listening. Using the quintessential Sydney University vocal recording of someone saying “I’m speaking from over here” a number of test were run on the system to test it’s performance limits. These limits have been included in the help at the beginning of the script. The gain limits have been recommended so as not to clip the ‘wavwrite’ output. At first there was a normalization performed on the output before writing to a wave file but it was observed that when a large gain factor was used in a single band it had the effect of

drastically reducing the signal in the other two bands to bring the peaks below clipping. So I have decided that rather than including normalization, instead recommended that smaller gain factors be used.

Within the submitted collection of files there is some examples of possible uses for the compressor. The details of each file are as follows.

Example1.wav

CT1 = 0
CR1 = 1
G1 = 0
CT2 = 0
CR2 = 1
G2 = 0
CT3 = -30
CR3 = 4
G3 = 5

Example2.wav

CT1 = 0
CR1 = 1
G1 = 0
CT2 = -30
CR2 = 3
G2 = 0
CT3 = 0
CR3 = 1
G3 = 0

Example3.wav

CT1 = -40
CR1 = 4
G1 = 0
CT2 = 0
CR2 = 1
G2 = 0
CT3 = 0
CR3 = 1
G3 = 0

Example4.wav

CT1 = -40
CR1 = 4
G1 = 0
CT2 = -30
CR2 = 3
G2 = 0
CT3 = -30
CR3 = 4
G3 = 5

5. Conclusion

The code written is for a basic 3-band multiband compressor. This could be easily expanded out to have more bands with some work on the filter performance to achieve a reasonably flat magnitude response. There are a number of variables that can be modified within the code of the system to change the system's performance. Over time these factors could be fine tuned to result in even better performance. These factors include, but are not limited to the filter parameters, RMS detector time average function (TAV) and compression attack and release times. The compressor could also include expansion factor and a peak detection option. In a full implementation it could be beneficial to make these controls available to the end user to change within set limits.

References

GIANNOULIS, Dimitrios, Michael MASBERG, and Joshua A REISS. 2012. *Digital Digital Dynamic Range Compression Design - A Tutorial and Analysis*. J. Audio Eng. Soc. **60**(6), p.399.

MCNALLY, G.W. 1984. *Dynamic Range Control of Digital Signal*. J. Audio Eng. Soc. **32**(5), pp.317-318.

ZOLZER, Udo. 2002. *DAFX - Digital Audio Effects*. John Wiley & Sons.

ZOLZER, Udo. 2008. *Digital Audio Signal Processing*. Hamburg: John Wiley and Sons.