DIGITAL AUDIO SYSTEMS

LAB REPORT 2

Nick Frost SID: 308211448

Creating and implementing a Stereo ‘Wonder-Wah’ Filter

The Problem:

A musician has written a chord progression for a song he/she would like to record digitally. Currently this section of the song has been deemed to “shallow” and the musician has said that the chords need to be “brought to life” through the use of movement in the sound. The musician must to be able to adjust the amount of movement to create different atmospheres and wants the chords to sing at certain frequencies as if the sound was originating from a human mouth.

The Solution:

The creation of a custom ‘Wonder Wah’ filter that uses a modulated stable state filter to bring life and movement to the music. In order to enhance some portions of the spectrum of the music I propose to implement a modulated band pass filter. This filter will allow a certain band of frequencies through, while attenuating the other frequencies. By modulating the band pass filter over time, as the filter sweeps through the frequencies it will resemble the way in which a human mouth can filter a sound source.

The challenge of this development is to implement dual band pass filters, the parameters of which can be individually controlled to create a custom stereo output. The construction of the ‘Wonder Wah’ can be broken up into three parts;
-Creating a triangular waveform to modulate the band pass
-Building the state variable filter and linking it to the triangle wave
-Splitting the output audio into two individual filters on each channel

The first step in creating this digital signal processor in MATLAB is to create a function that implements one of the two ‘wahs’. I have named this wah_wah_function.m. To call the .wav input audio file into the function I use the command:

```
[ x, Fs, N ] = wavread(infile);
```

Next, the sampling frequency and damping factor are set. The damping factor is the width of the pass band. Higher values will give a broader pass band, while smaller values will give tighter bands and sound more articulate.

```
Fs = 44100

damp = 0.05;
```

Now the filter must be programmed so that the pass bands will modulate up and down between two low and high cut offs that will be programmed later. This is achieved by constructing a triangular waveform that will oscillate the filter. The speed or rate of the effect is divided by the sampling frequency. A faster rate will result in the "wah” cycling up and down between the low cut and the high cut frequencies more rapidly.

```
DELTA = rate/Fs;
```

Then a vector of is created and the centre frequency Fc will modulate the band pass filter. DELTA is a slope created between the low cut and the high cut frequencies. So the band pass will start the low cut and sweep up linearly to the high cut.

```
Fc=lowcut:DELTA:highcut;
```

To loop these sweeps up and down repeatedly the length of the vector is made less than the length of the input. The band pass will now modulate from the high cut down to the low cut at a slope of -DELTA. Then once it is at the low cut, the Fc will sweep back up to the high cut at a slope of DELTA.

```
while(length(Fc) < length(x) )
    Fc= [ Fc (highcut:-DELTA:lowcut) ];
    Fc= [ Fc (lowcut:DELTA:highcut) ];
End
```

Then the code ensures that the length of the triangle wave that has been created is adjusted so that the centre frequency does not stop modulating at some point during the output audio.

```
Fc = Fc(1:length(x));
```

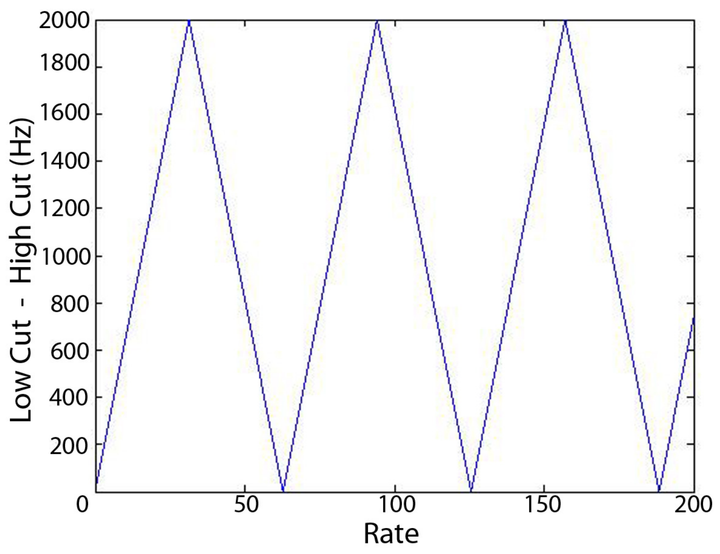

Figure 1: Triangle wave that oscillates between a low cut and a high cut

Finally some more code calculates the tuning values each time the centre frequency changes. Then a matrix of zeros or empty samples the sample length as the input is constructed for each of the difference equations (highpass, bandpass, lowpass).

```
F1 = 2*sin((pi*Fc(1))/Fs);
Q1 = 2*damp;


yh=zeros(size(x));
yb=zeros(size(x));
yl=zeros(size(x));
```

As the loop of the triangle oscillator starts at n=2, each of the output filters is set to 1 in order to make up for the very first sample in the output.

```
yh(1) = x(1);
yb(1) = F1*yh(1);
yl(1) = F1*yb(1);
```

The second stage in the MATLAB code is to create a digital, state variable filter that will be used for the band pass. A band pass filter is simply a low pass filter combined with a high pass filter. It will allow a defined "band" of frequencies through, while it attenuates at all other frequencies. The state variable filter will give the user much more control over the cut-off frequency (Fc) and damping factor (C). This state variable filter was described in Hal Chamberlin's *Musical Applications of Microprocessors.* The filter acts upon its inputs x (n) (where n is the sample number), to produce its outputs y(n). Because the state variable filter can produce three outcomes they are named yh(n), yb(n), and yl(n). These signify the high pass, band pass, and low pass outputs respectfully. It uses the following formula:

$$yh(n) = x(n) - yl(n-1) - Q1*yb(n-1)$$

$$yb(n) = F1*yh(n) + yb(n-1)$$

$$yl(n) = F1*yb(n) + yl(n-1)$$

The formula uses the tuning coefficients:

$$F1 = 2*\sin((pi*Fc(1))/Fs) \quad \text{and} \quad Q1 = 2*C$$

Where Fc is the cutoff frequency and Fs is the sampling frequency and C is the damping factor. The difference equation coefficients have been taken from chapter 2 of the DAFX. The formula is illustrated in the following signal flow diagram:

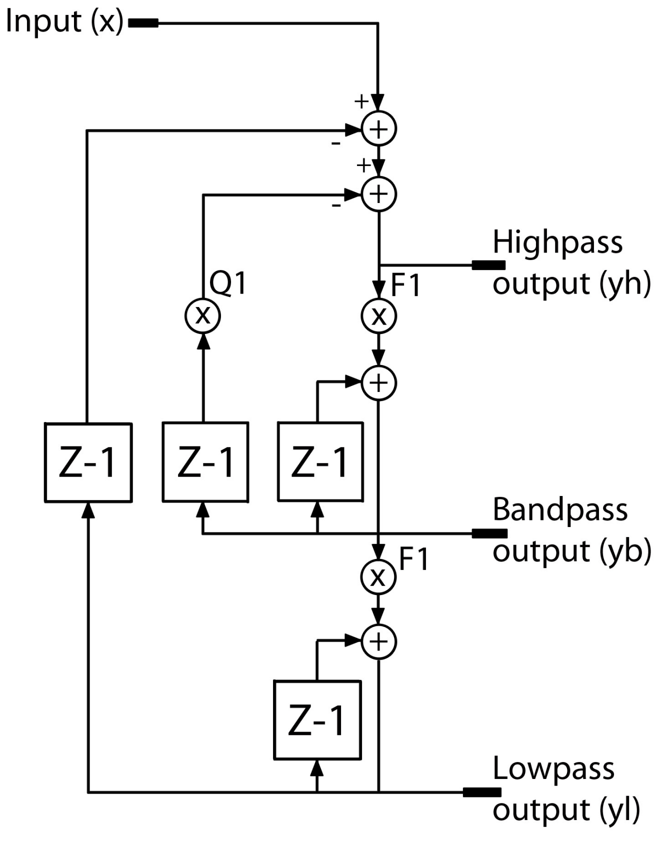

Figure 2: State variable filter used to create the band pass output for the Wonder Wah

Now that a 'wah' filter function has been created, I use a second function DOUBLEWAH_stereo.m to apply the filter individually to the same audio input, and split the two outputs into two channels (left and right) to create a moving, stereo wah. The first line defines the input arguments

```
WONDER_WAH = DOUBLEWAH_stereo(infile, lowcutL, highcutL,
rateL, lowcutR, highcutR, rateR);
```

Then the parameters are applied to two separate filters.

```
LEFTWAH = wah_wah_function (infile, lowcutL, highcutL,
rateL);
RIGHTWAH = wah_wah_function (infile, lowcutR, highcutR,
rateR);
```

Finally the function WONDER_WAH will play the two filtered outputs through the left and right channels simultaneously.

```
WONDER_WAH = [LEFTWAH, RIGHTWAH];
```

Now that the digital signal processor is complete, it can be called using the script:

```
DUAL_STEREO_WAH = DOUBLEWAH_stereo(infile, lowcutL,
highcutL, rateL, lowcutR, highcutR, rateR);
```

Where the input audio and all input arguments can be adjusted at will. The following waveforms show the original input and then how the 'Wonder Wah' has split filtered each side and created a modulated stereo output.

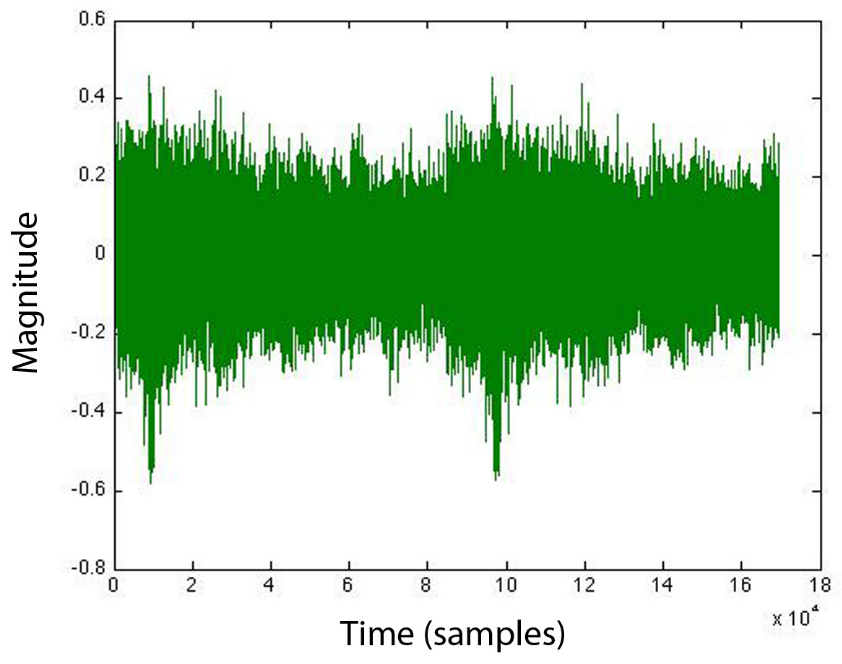

Figure 3: The original waveform of the 'chords.wav' input audio

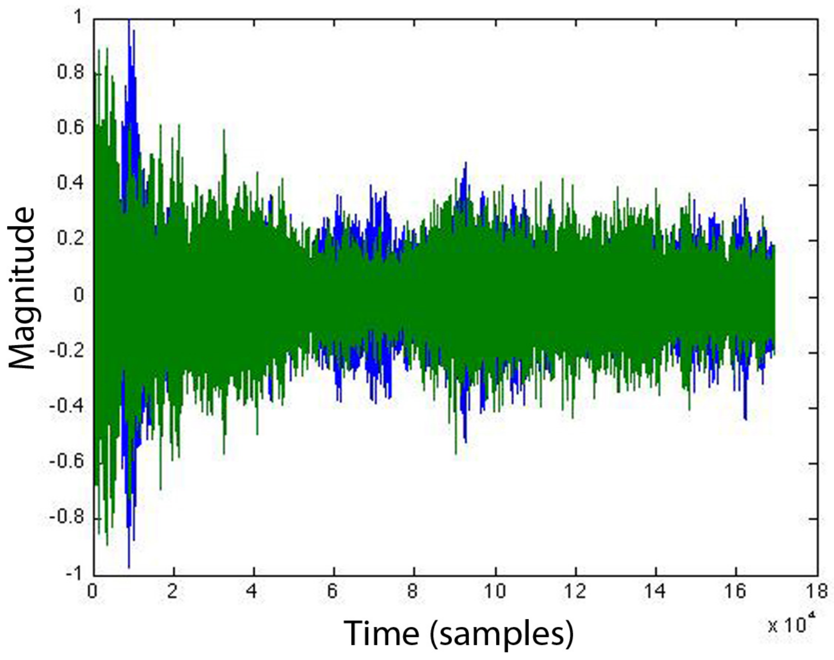

Figure 4: The output waveform after ‘chords.wav’ has been run through Wonder Wah with the settings:

Left low cut = 500
Left high cut = 2000
Left rate = 1600
Right low cut = 1400
Right high cut = 3400
Right rate = 1000

BONUSES OF WONDER WAH:

Creating beautifully filtered stereo wah effects is not all that this code is capable of. Because the function implements a state variable filter the resulting high pass, band pass, and low pass filters can all be adjusted individually. By setting a low rate (eg. 50) in the input arguments of the script, the Wonder Wah can be used as a low pass filter, a high pass filter or can even be used to boost two bands as an equaliser. For example listen to the WONDER_WAH_EQUALISER.wav file to hear how movement in the filter has been reduced and a bandpass emphasises some frequencies more than others. It used the parameters ('chords.wav', 300, 600, 100, 2000, 2500, 100); to achieve these peaks. The following graph demonstrates the peaks in the frequency domain:

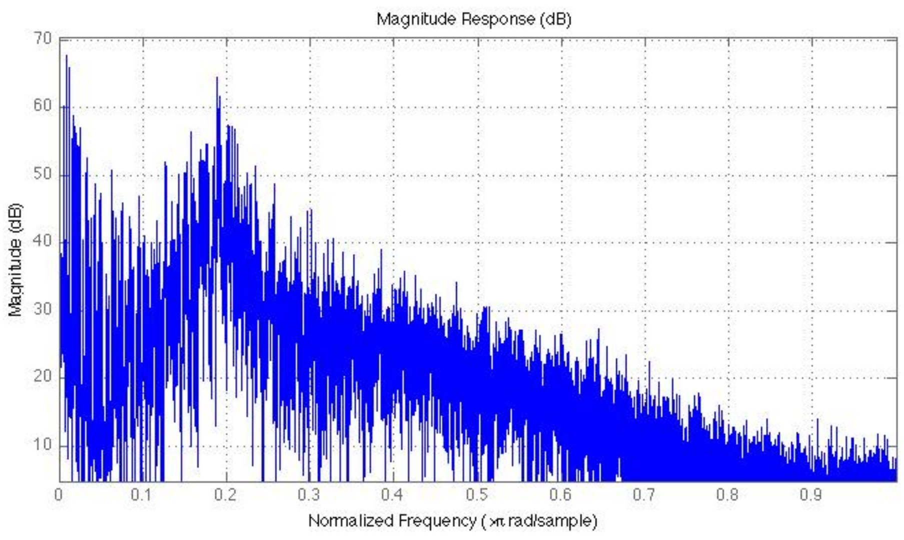

Figure 5: The frequency analysis of WONDER_WAH_EQUALISER.wav. Note the two peaks where the band pass filters are located.

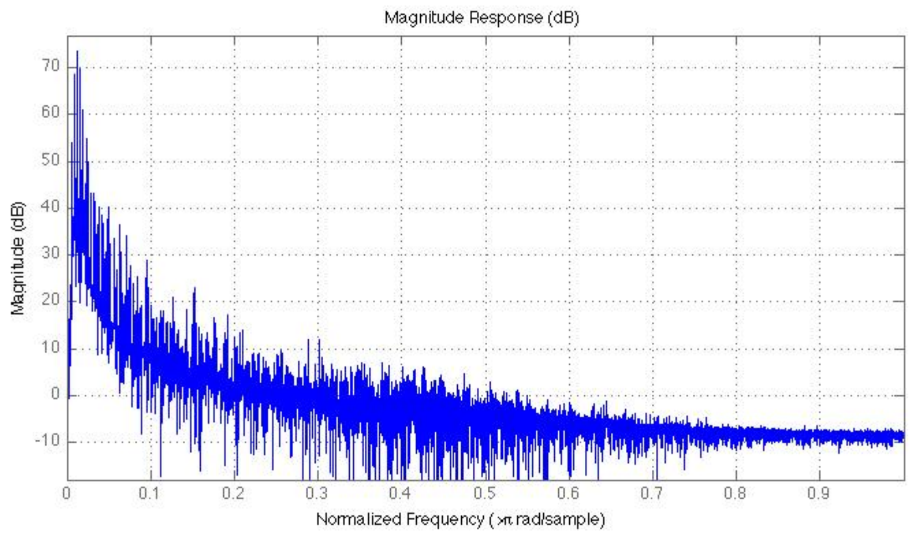

Figure 6: Frequency analysis of the Wonder Wah filter with low pass filter parameters ('chords.wav', 10, 1500, 100, 100, 300, 100). The higher frequencies are attenuated and the low frequncies are passed.

RECOMMENDED PARAMETERS:

('chords.wav', 500, 2000, 1600, 1400, 3400, 1000); this setup will create two “wah” cycles (one per chord) in the low-mid range frequencies with one slow “wah” cycle/sweep in higher frequencies. This sounds quite gradual and smooth and can be heard in the audio file WONDER_WAH.wav.

AVOID HIGH RATES:

For a greater depth between the low and high cut, a very slow rate is recommended. Faster rates sound chaotic and warbled. For example listen to the WONDER_WAH_highrate.wav as an example of how a high rate can give unwanted results. The filter parameters used for this file were ('chords.wav', 1000, 4000, 11000, 100, 3500, 90000);

REFERENCES FOR MATLAB CODES:

Zolzer, Udo, 2002, *DAFX: Digital Audio Effects,* John Wiley & Sons

O'Malley, Ronan, October 2nd 2005, wah_wah.m State variable bandpass filter

http://www.cs.cf.ac.uk/Dave/CM0268/PDF/10_CM0268_Audio_FX.pdf

Chamberlin, Hal, 1980, *Musical Applications of Microprocessors,* Hayden Book Co.