SPATIAL CHORUS WITH FRACTAL MODULATION
LAB REPORT 1

Robert W Taylor 311123392 (SID)

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

INTRODUCTION

Chorus is a member of the family of 'delay based effects'. Like *flanging* and *vibrato* the resultant effect is based on the principals of comb filtering and pitch variance due to dynamic modulation. In this lab report I will be discussing the processes undertaken to achieve a chorus effect using a fractal modulator, within the MatLab environment.

1. THE CONSTRUCTION OF THE CHORUS EFFECT IN MATLAB

1.1. FIR comb filter

This spatial chorus function was developed originally from a vibrato algorithm sourced from the DAFx text [1]. In this case a signal was periodically modulated in time to produce variance in pitch.

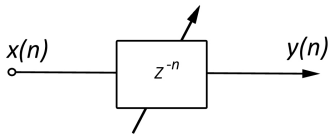

Figure 1. Vibrato

What disseminates a chorus and flange effect from vibrato is the inclusion of a direct signal and a delayed version creating an FIR comb filter.

In the next stage of the function construction, a feed forward signal was implemented to the vibrato, which essentially creates a FIR comb filter, hence the *flange effect*. Gain control was also added to both the delay line and feed forward to control the relative amplitude of the delayed and original signal. [2]

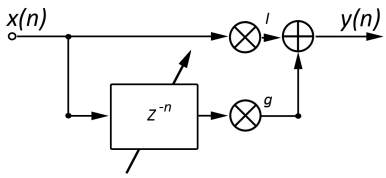

Figure 2. A FIR comb filter. A vibrato with feed forward will create a basic flange effect.

The difference equation and transfer function are given by: [3]
Eq.1

$$\mathcal{Y}(n) = \mathcal{X}(n) + gx(n - \mathcal{M})$$

$$with \quad M = \tau / \mathcal{F}_s$$

$$\mathcal{H}(z) = 1 + gz^{-M}$$

'For positive values of g, the filter amplifies all frequencies that are multiples of $1/\tau$ and attenuates all frequencies that lie in between. The transfer function of such a filter shows a series of spikes and it looks like a comb. For negative values of g, the filter attenuates frequencies that are multiples of $1/\tau$ and amplifies those that lie in between. The gain varies between 1 + g and 1 – g.'[4] (Figure 3)

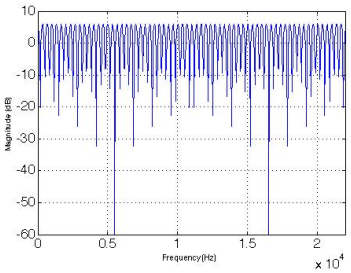

Figure 3. Magnitude response of a 3ms delay line/1kHz tone. An example of FIR comb filtering, and as you can see it creates a 'comb' like response.

1.2. Interpolation

Interpolation is necessary to process non-integer values of the sampling frequency due to variable delay times. 3 algorithms were included in the original DAFx vibrato function, which were allocated to 'cases' in a switch. Linear, Spline and All-pass.

'The choice of algorithm depends on the specific application' [5] and in this situation a spline interpolation was chosen primarily due to the audible quality of the output signal. This was an arbitrary decision, however the immediate concerns of the level of distortion and overall quality led to this selection. Further investigation will be required.

Spline interpolation:
Eq. 2

$$y(n) = x(n - [M + 1] \cdot \frac{frac^3}{6}$$

$$+ x(n - M) \cdot \frac{(1 + frac)^3 - 4 \cdot frac^3}{6}$$

$$+ x(n - [M - 1]) \cdot \frac{(2 - frac)^3 - 4(1 - frac)^3}{6}$$

$$+ x(n - [M - 2]) \cdot \frac{(1 - frac)^3}{6}$$

1.3. Modulation signal

Vibrato and flange both use smooth sinusoidal modulation of the delay time to create their respective effects. Chorus on the other hand uses a random modulation via white noise as the modulating signal. One of the main aims of this exercise was to experiment using a fractal noise algorithm. A separate function to create this fractal modulation signal was developed by Assoc. Professor William Martens for this project. [frand64.m] [Figure 4] [6]

With the use of a 'switch', both a fractal and white noise modulator [randsig.m] could be selected giving an opportunity to compare the results.

$$x_1 = [a_{11}, a_{12}, a_{13}, a_{14}, a_{15}, a_{16}, a_{17}, a_{18}]$$
$$x_2 = [a_{11}, a_{12}, a_{13}, a_{14}]$$
$$x_3 = [a_{11}, a_{12}]$$

$$x_{Conc} = \begin{bmatrix} x_1 \\ x_2\ x_2 \\ x_3\ x_3\ x_3\ x_3 \end{bmatrix}$$

$x_{frac} = [a_{11}, a_{12}, a_{13}, a_{14}, a_{15}, a_{16}, a_{17}, a_{18}]$
(where the a_{ij} entry is the sum of the $a_1 - n$ rows of x_{Conc})

Figure 4. An example of an 8 sample fractal algorithm when each entry is a random number.

To control the correlation between the left and right modulators, a simple mixing system was implemented as shown below [pcor_sigs.m]. This is an important spatial component considering that when de-correlated it will enhance the 'stereo width' effect associated with ITD (Inter-aural time delay). [y.wav] When a mono modulated signal is fed to both left and right, this effect is reduced. [y1.wav]

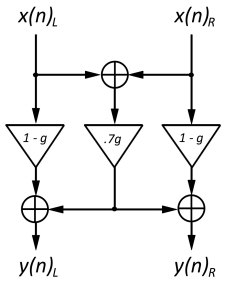

Figure 5. Mixing algorithm to control correlation between left and right modulators.

A lowpass filter was then added to the modulation signal to control sub-audio modulation rates. A second order filter was created in a separate function and implemented into the main chorus function. [M_fq2coef.m] [7]

Finally, the modulating signal is normalized as the signal can go up to high levels due to the filtering. This makes sure that the depth actually makes sense as the multiplier of the signal and the percentage of the maximum deviation from the base delay. The following standard normalizing code is used:

```
resampled_noiseL = resampled_noiseL./max(abs(resampled_noiseL));
resampled_noiseR = resampled_noiseR./max(abs(resampled_noiseR));
```

1.4. Memory allocation

It was important to allocate adequate memory for the output vectors of the delay. Calculations are made for any possible length of delay line when the parameters are at extremes, and zeros created using the following code;

```
L = ceil(6 + DELAYMAX + (DELAYMAX*WidthMax));
% length of the entire delay
DelaylineL=zeros(L,1);
DelaylineR=zeros(L,1);
% memory allocation for delay
yL=zeros(length(x),1);
yR=zeros(length(x),1);
% memory allocation for output vector
```

1.5. Gain structure and output

It is typical of an effects system to provide a function to mix the 'dry' and 'wet' signals at the output stage. A simple additive algorithm was included to combine the original signal and effected signal. In this case parameters were set to provide choices between, -1 (100% dry), 0 (50/50), 1 (100% wet). It was implemented using the following code:

```
mixnorm = mix + 1;
mixnorm = mixnorm./2;
drylevel = 1 - mixnorm;
wetlevel = mixnorm;
x_gain_corrected = x .* drylevel;
yL = yL .* wetlevel;
yR = yR .* wetlevel;
```

Finally, a two-channel matrix for output and the mixed output signal normalized, including a gain control to provide the final output amplitude between 0 and 1.

```
y = [yL yR];
y = (y./max(max(abs(y)))).*output_level;
```

2. FUNCTION PARAMETERS

```
function y=Spatial_chorus(x,SAMPLERATE, WidthL,
WidthR, DelayL, DelayR, ModRate, ModCorr,
ModType, ModFilterF, ModFilterQ, Interpolation,
mix, fx_shift, output_level); [Spatial_chorus.m]
```

- 'x' – wave input
- 'SAMPLERATE' – Sample rate of sound source in Hz.
- 'WidthL, WidthR' – Peak deviation from base delay. Value between 0 (less width) and 1 (more width). [y3.wav] [y4.wav]
- 'DelayL, DelayR' – Delay of delay line in milliseconds. [y5.wav] [y7.wav]
- 'ModRate' – Control rate of modulation signal in samples/sec. [y7.wav]
- 'ModCorr' – Correlation of left and right modulation signals. Value between 0 (no correlation) and 1 (correlation). [y.wav] [y1.wav]
- 'ModType' – To select the type of modulation signal, either fractal based or white noise. Value either 1 (fractal), or 0 (white noise). [y.wav] [y2.wav]

- ‘MofFilterF’ – Modulation signal filter frequency selection in Hz
- ‘ModFilterQ’ – Modulation signal filter bandwidth in Hz.
- ‘Interpolation’ – selection of interpolation algorithm. Value either 1 (spline) or 2 (all-pass).
- ‘Mix’ – mix level between dry/wet signals. Values between -1 (100% dry), 0 (50/50), 1 (100% wet).
- ‘fx_shift’ – Delay of the modulated channel path in milliseconds.
- ‘output_level’ – Overall level of final output signal

3. RESULTS

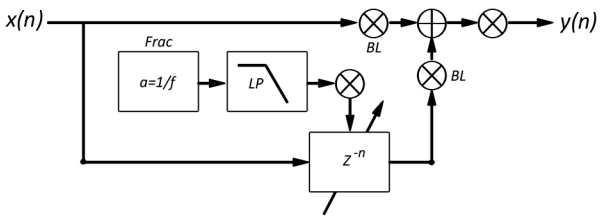

Figure 6. A diagram of the finished ‘Spatial chorus with fractal modulation’.

The resulting digital audio effect is a working ‘Spatial chorus with fractal modulator’. [Spatial_chorus.m] The parameters included are designed to give the operator maximum control over all aspects of this chorus effect, more than would be available in a commercial product. Due to the experimental nature of the design, it is necessary at this stage to provide such parameters until conclusions can be made to their effectiveness.

It must be considered that some parameters could be set to a default setting or in fact not included at all. An example of these would be the parameter for selection of ‘interpolation’, which after evaluation only the one considered most appropriate would be included.

The fractal noise created for this function, although does provide a usable alternative to white noise as a random modulation signal, requires further evaluation and subject testing before any further conclusions can be made. There is a obvious slight audible differentiation between the fractal and white noise modulation to the outcome, so in my opinion further evaluation is warranted.

4. ADDITIONAL FILES

MatLab Files:

[Spatial_chorus_script_LabReport1.m] – The MatLab operational script to run the effect.

[Spatial_chorus.m] – The main MatLab function of the effect.

[frand64.m] – The MatLab function containing the fractal modulator signal.

[randsig.m] – The MatLab function containing the white noise modulator signal.

[pcor_sigs.m] – The MatLab function containing the mixing system of the left/right modulator signals.

[M_fq2coef.m] – The MatLab function containing the second order filter of the modulator signal.

Sound files:

- [x.wav] – original file
- [y.wav] – reference parameters (as listed below)

```
% Reference parameters:
if nargin < 15, output_level = 0.999; end
if nargin < 14, fx_shift = 0; end
if nargin < 13, mix = 0; end
if nargin < 12, Interpolation = 1; end
if nargin < 11, ModFilterQ = 2; end
if nargin < 10, ModFilterF = 8; end
if nargin < 9, ModType = 1; end
if nargin < 8, ModCorr = 0; end
if nargin < 7, ModRate = 10; end
if nargin < 6, DelayR = 10; end
if nargin < 5, DelayL = 8; end
if nargin < 4, WidthR = 0.09; end
if nargin < 3, WidthL = 0.05; end
if nargin < 2, SAMPLERATE = 44100; end
```

The following files include the reference parameters as base settings plus the parameters listed.

- [y1.wav] - Maximum correlation between left and right modulators.
- [y2.wav] – white noise modulator signal.
- [y3.wav] – WidthL 0.04, WidthR 0.02.
- [y4.wav] – WidthL 0.6, WidthR 0.4
- [y5.wav] – DelayL 15ms, DelayR 20ms
- [y6.wav] - DelayL 25ms, DelayR 30ms
- [y7.wav] – ModRate 40Hz, ModFilterF 25Hz.
- [y8.wav] - ModRate 40Hz, ModFilterF 25Hz and white noise modulation signal

5. REFERENCES

[1] Digital Audio Effects, Edited by Udo Zölzer, John Wiley & Sons, 1st edition 2002, ISBN: 0-471-49078-4. (Chap 3, Page 69).
[2] Op. cit. (Chap 3, Page 69).
[3] Ibid (Chap 3, Page 65).
[4] Ibid (Chap 3, page 64).
[5] Ibid (Chap 3, page 68).
[6] Evangelista, Gianpaolo. *“Fractal modulation effects”*. Proc. of the 9th Int. Conference on Digital Audio Effects (DAFx-06), Montreal, Canada, 2006 (Page 106)
[7] Zölzer, Udo. *“Digital Audio Signal Processing”*. John Wiley and Sons, 2nd edition, 2008, ISBN 978-0-470-99785-7. (Chap. 5, Page 130).