Lab Report 2

Carl Edser
Student # - 430488700

DESC9115 – Digital Audio Systems
William Martens
Assessment Task 3
Due Tuesday Week 12, Semester 1, 2013

Table Of Contents

ABSTRACT

For the second lab report in digital signal processing, I have chosen to discover, develop and implement a sample-based limiter. The DAFX textbook, *Chapter 5 – Nonlinear Processing*, written by P. Dutilleux and U. Zölzer, will be referred to in this report and has provided guidance for the development of limiter function (limiter.m) and script (call_limiter.m) in MatLab.

1. INTRODUCTION

A limiter is a type of dynamics processor that can automatically control the amplitude of the input signal based on its detected input gain (Dutilleux & Zölzer 2002, p.95). This device can be used to limit audio peaks, whilst also maximising the amplitude of the input signal without resulting in the distortion of the output. A limiter can be used in all production phases of audio, but is most commonly called upon in the post-production, mastering process (applied for multichannel processing) and live sound applications. Not only should audio engineers use this device to aid the dynamic qualities of the input signal, but also implement the device to protect a systems output equipment from overload and from causing damage to the listener's ears. Other devices that are considered to be dynamic processors are compressors, expanders and noise gates.

A block diagram explaining the signal flow of the limiter has been provided below, along with a description of its components.

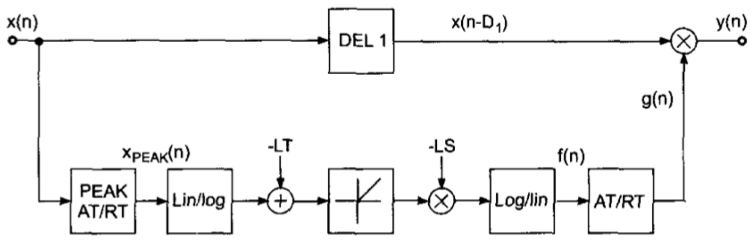

Figure 1: Block diagram of a limiter (Zölzer, 2002, p.99)

$x(n)$	= input signal	$x(n-D_1)$	= delayed input signal
$x_{PEAK}(n)$	= peak detector of the input signal	-LT	= limiting threshold
-LS	= limiting slope	$f(n)$	= output factor
$g(n)$	= limiter output	$y(n)$	= output

2. EXPERIMENTAL

In order to process the audio through the limiter function, the syntax, along with several control variables of the limiter must first be defined. The following syntax was used;

```
function dataout = limiter(datain, fs, att, rel, compfac, compthresh, limthresh)
```

where;

```
%    datain      = input signal
%    fs          = sampling frequency in Hz
%    att         = attack time of the limiter
%    rel         = release time of the limiter
%    compfac     = compression factor
%    compthresh  = compression threshold
%    limthresh   = limiter threshold
%    dataout     = output where limiter has been applied to the input
                   signal
```

The limiter must make use of peak level measurement and should react very quickly to the extensions of the limiter threshold (Dutilleux & Zölzer 2002, p.99). This means that it is a requirement of the limiter to detect the peaks of the input signal before it reaches the output, so that the peaks aren't further amplified by the limiter, potentially resulting in distortion of the output signal. However this must be imposed when the DSP detects a certain variation in the difference between peak and RMS values. This has been accounted for in the function `limit_react` using the formula;

```
limit_react = (limthresh - compthresh) * compfac + compthresh;
```

It is a requirement of the limiter to respond to the input signal in a matter of microseconds/samples. This is due to the inability of a DSP to 'look ahead' of time, particularly when referring to live sound or other input applications. The attack time (`att`) parameter provided in my code allows the limiter's response to the input to trigger within 1×10^{-5} and 2 seconds. The sample calculation of the attack parameter is carried out in MatLab by;

```
att = 1-exp(-2.2* Fs/att);
```

This is the same for the release parameter, however `rel` replaces `att`. The release time parameter (`rel`) determines the time it takes for the limiter to release the compression hold on the input signal. If this is set to a short time, the affect of the limiter may almost go unnoticed, provided the threshold amounts are not set too low. This could be considered ideal by capturing only the peaks and imposing minimal affect on the input RMS.

A range of parameters in the function has been devised to constrain the use of the limiter so that it can be used appropriately. These are under the title 'Potential error messages from invalid input values in script' seen in Appendix 1. If the parameters in the script are exceeded by the user, then an error message will confirm what value in the script has been exceeded.

3. RESULTS

An unprocessed audio signal (top, 'Time To Relax.wav') was inserted into the script and processed using the limiter function and script parameters written in MatLab (bottom, *see figure 1*). The input and limited output were then plotted in figure 2. The bottom image demonstrates that the limiter function has increased the RMS gain of the input signal, whilst also removing the peak that initially occurs in the top image (at approx. $2x10^4$ samples).

The bottom example was achieved by using the variables defined in the high order script. These were implemented as follows;

```
att         = 0.0001;
rel         = 0.5;
compfac     = 1;
compthresh  = -17;
limthresh   = -17;
```

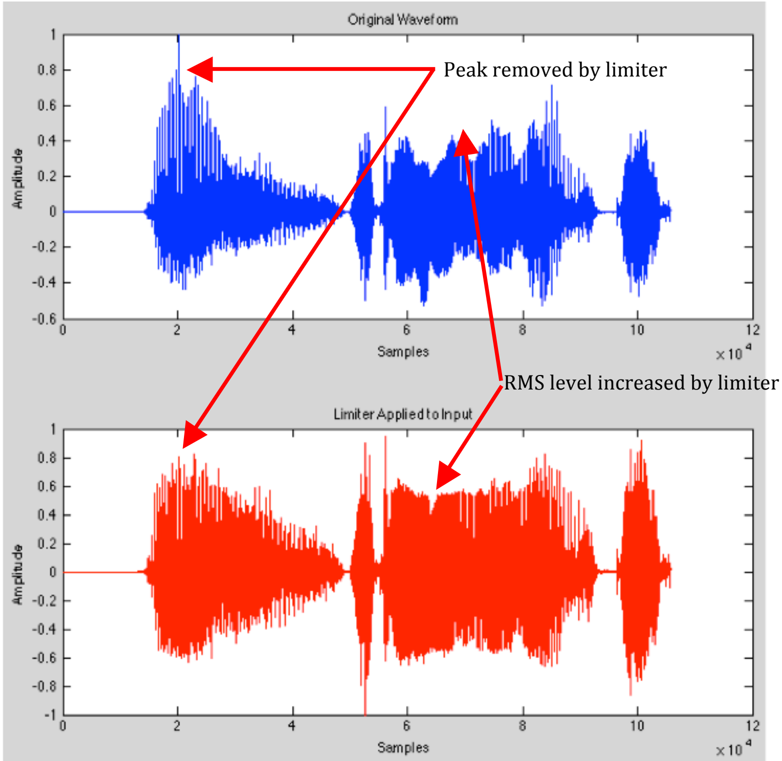

Figure 2: Original input (top) and processed output using limiter function (bottom)

4. CONCLUSION

Although I have designed a limiter that is capable of compressing the peaks and expanding the input RMS level, I feel as though there are several parameters that may still need refining. Firstly, the attack and release parameters defined in the function do not perform as I expected, so there is potentially a function missing here. Also, when adjusting the compressor and limiting thresholds greater than -15, some of the peaks are still able to clip. Again, I feel that this is due to an error in the attack and release components of my function. With this being amended, I am sure the limiter function will work appropriately, however with the time constraints of the assignment, I am unable to complete this.

APPENDIX 1

```
%% LIMITER FUNCTION

function dataout = limiter(datain, Fs, att, rel, compfac, compthresh, limthresh)

%   datain      = input signal
%   Fs          = sampling frequency in Hz
%   att         = attack time of the limiter
%   rel         = release time of the limiter
%   compfac     = compression factor
%   compthresh  = compression threshold
%   limthresh   = limiter threshold
%   dataout     = output where limiter has been applied to the input signal

% Conversion from decibels (dB)

compthresh = 10^(compthresh/20);
limthresh = 10^(limthresh/20);


% Calculate the amplitude at which the limiter is activated

limit_react = (limthresh - compthresh) * compfac + compthresh;


% Attack and release times converted into samples

att = 1-exp(-2.2* Fs/att);
rel = 1-exp(-2.2* Fs/rel);

if size (datain, 2) > size (datain,1)
    datain = datain';
end;


% Initialize peak detector vector and gain factor vector

xd = zeros (length(datain),1);
f = zeros (length(datain),1);


% Analyses the sound vector and compares the signal values with the
% threshold values. Set gain factor vector and multiply

for n = 2:length(datain);

    a = abs (datain(n,1)) - xd (n - 1);
    if a < 0
        a = 0;
    end;
    xd(n) = xd(n - 1) * (1-rel)+ att * a;

    if compthresh < xd(n) && xd(n) < limthresh;
        f(n) = 10^(-limiter * (log10(xd(n))-log10(compthresh)));

    elseif xd(n) >= limthresh;
        f(n) = 10^(-1*(log10(xd(n))-log10(limit_react)));

        else
            f(n) = 1;
    end;
end;


% Multiply with the gain factor vector

dataout (:,1) = datain(:,1).* f;


% Maximize

dataout(:,1) = dataout(:,1)/max(abs(dataout(:,1)));


% Potential error messages from invalid input values in script

if nargin < 7
    error ('Not enough input arguments. Check function parameters')
end;
```

```
if 0.00001 > att || att > 2
    error ('att value must be between 0.00001 and 2 (ten microseconds / two seconds');
end;

if 0.005 > rel || rel > 5
    error ('rel value must be between 0.005 and 5 (five milliseconds / five seconds');
end;

if 0 > compfac || compfac > 1
    error ('compression factor must be set between 0 and 1');
end;

if 20 < compthresh || compthresh < -100
    error ('Compression threshold must be set between -100 and 20');
end;

if 20 < limthresh || limthresh < -100 || limthresh < compthresh
    error ('To use as limiter, limthresh must be set the same as compthresh');
end;


% Plotting of the input signal against processed output signal

subplot(2,1,1);
plot (datain, 'b'), title('Original Waveform'), xlabel('Samples'), ylabel('Amplitude');
subplot(2,1,2);
plot(dataout, 'r'), title('Limiter Applied to Input'), xlabel('Samples'), ylabel('Amplitude');
```

```
%% LIMITER SCRIPT

[datain, Fs] = wavread('Time To Relax.wav');

att          = 0.0001;       % attack time (set between 0.00001 and 2)
rel          = 0.5;          % release time (set between 0.005 and 5)
compfac      = 1;            % compression factor (set between 0 and 1)
compthresh   = -17;          % compression threshold (set between -100 and 20)
limthresh    = -17;          % limiter threshold (set equal to the compthresh
                             % and between -100 and 20)

dataout = limiter (datain, Fs, att, rel, compfac, compthresh, limthresh);

wavwrite(dataout, Fs, 'Time To Relax Limiter.wav');
                             % Writes output as WAV file
```

References

Dutilleux, P & Zolzer, U 2002, *DAFX – Digital Audio Effects*, John Wiley & Sons, Ltd, West Sussex, England.

Lundkvist, A & Oman, P 2009, *S76006E – Lab 4. Compressor/Limiter/Maximizer*, Lulea University Of Technology, Lulea, Sweden.