DESC9115: DIGITAL AUDIO SYSTEMS
2013 LAB REPORT 2
Instructor: William L. Martens
Student: Ben Beverley
SID: 420170037
Date: Tuesday 28th May, 2013

DSP Application: Broadcast Loudness Corrector

Note: This report refers to the following attached Matlab files: broadcast_level_corrector.m, CALL_broadcast_level_corrector.m, loudness_itu.m

In January 2013, new broadcast audio delivery standards that had already been implemented in the USA and Europe came into effect in Australia. The new standards, outlined in the Free TV Australia Operational Practice OP-59 (2010), are based on the International Telecommunication Union recommendation ITU-R BS.1770-3 (2012). The ITU recommendation determines methods and specifications of audio metering to ensure TV viewers experience a more even loudness of content across different programmes and commercials.

There has been a number of digital audio applications released to aid in accurate loudness monitoring by audio professionals. These plugins and standalone applications can be time consuming, expensive and difficult to operate, particularly for non-audio professionals. It is common for video editors with limited knowledge of audio processing and metering to be required to deliver content direct to broadcast. Television networks too must implement "tech checks" on all delivered content, often rejecting deliveries based on inappropriate audio and loudness levels or even digital clipping. And at the consumer level, there is a distinct lack of any audio standards at all for amateur filmmakers providing videos to websites such as YouTube and Vimeo.

The Broadcast Loudness Corrector (BLC) is a DSP application that analyses, processes and exports single or multi-channel wave files to the latest audio broadcast specifications in ITU-R BS.1770-3. Through the use of true-peak measurement, a built-in limiter and automatic LKFS loudness matching, the BLC can accurately ensure that audio deliverables match loudness criteria with merely a few clicks.

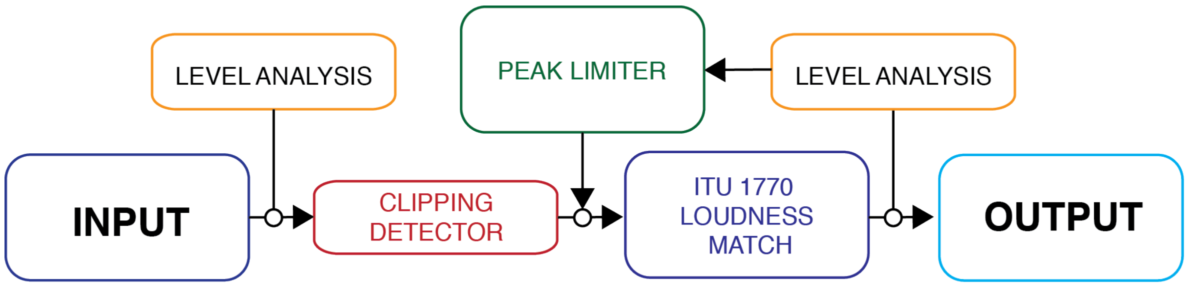


Figure 1. *Broadcast Loudness Corrector signal flow*

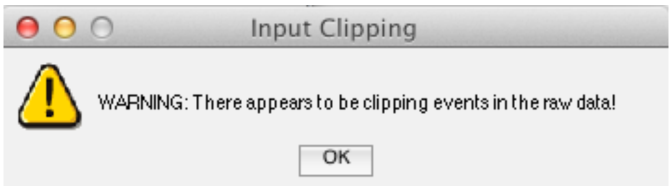
Digital Signal Processing:

The BLC processor follows the signal flow seen in Fig. 1. The signal's input levels are first assessed. This Level analysis occurs multiple times throughout the processing, as it is used to provide onscreen feedback to the user, as well as provide data for graph plotting. The peak levels are analysed by finding the maximum value of the following equation:

$$dB = 20\log_{10}\left|\sqrt{x^2}\right| \tag{1}$$

where x is the discrete data signal

A clipping detector (Abdulla, 2007) is implemented through a process that seeks out whether any same sample value occurs ten times in a row. If clipping is detected in the input signal, a warning box will appear.



Input Clipping

WARNING: There appears to be clipping events in the raw data!

OK

Figure 2. *Input clipping warning box*

The ITU 1770 Loudness matching calls an external function 'loudness_itu.m' (Marui, 2012), which calculates the current LKFS value of the signal using equations and constants outlined in ITU-R BS.1770-3. Fig. 3 is a block diagram showing the signal flow of this LKFS loudness matching function for up to a 5.1 channel audio file (please note: only mono and stereo processing is currently supported the BLC function).

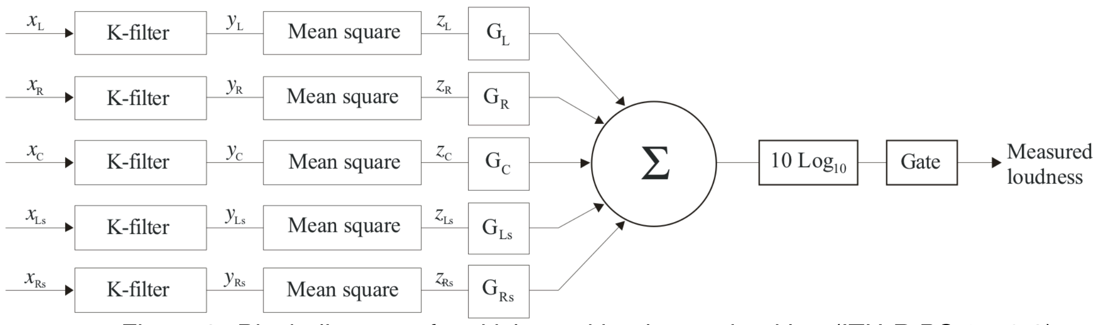


Figure 3. *Block diagram of multichannel loudness algorithm (ITU-R BS.1770-3)*

In the Matlab implementation, firstly the constants are set:

```
blockSize = 400;   % in ms
overlapSize = 0.75;   % in percentage
channelWeights = [1 1 1 0 sqrt(2) sqrt(2)];   % in L, R, C, LFE, Ls, Rs order
absoluteThreshold = -70;   % in dB
relativeThreshold = -10;   % in dB
```

The signal is then downsampled back to 48kHz using Matlab's 'resample.m' function, since this is the sampling frequency for which the ITU-R filter coefficients are defined. The channels are weighted according to the ITU-R specifications: the

Left, Centre and Right are all unweighted (1), Left Surround and Right Surround are weighted by $\sqrt{2}$, and the LFE is excluded (0).

A 2-stage pre-filtering is next applied using ITU-R specified FIR coefficients. The first stage of the K-filter is a shelving filter, which accounts for the acoustic effects of the head, whilst the second stage is a simple high-pass filter that serves to negatively weight the lower frequencies, somewhat matching the frequency response of the human auditory system. The mean square is then calculated by the following code:

```
for n1 = 1:length(j)
  yyy = yy(blockSize*j(n1)*(1-overlapSize)+1:blockSize*(j(n1)*(1-
overlapSize)+1)+1,:);
  z(n1,:) = sum(yyy .^ 2) / blockSize;
end
loudness_K = -0.691 + 10*log10(sum(repmat(chwat,length(z),1) .* z, 2));
```

The code uses a "for" loop to measure the mean square value for separate time intervals determined by the variable 'blocksize'. The final calculation for loudness over the measurement time (L_K) is calculated:

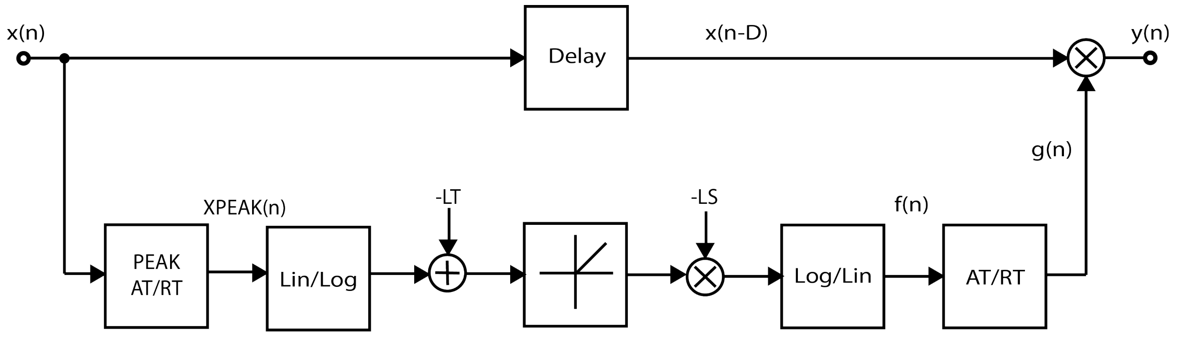
$$L_K = -0.691 + 10\log_{10}\sum_i G_i \cdot z_i \tag{2}$$

where G_i are the weighting coefficients for each channel and z_i is the mean square of the filtered input signal for a measured time interval (ITU-R BS.1770-3)

A two-stage process is then used to make the gated measurement:

```
% Gating (absolute) -
Jg = loudness_K > absoluteThreshold; % when loudness exceeds absolute threshold
zz = repmat(channelWeights(1:numch),length(z),1) .* z; % replicates and tile
arrays zeros according to number of channels
L_KG = -0.691 + 10*log10(chwat * sum(zz(Jg,:), 1)' / sum(Jg)); % calculation
for gated loudness amount
Gamma_r = L_KG + relativeThreshold; % sum of gated loudness and relative
threshold
Jg = loudness_K > Gamma_r; % represents when loudness exceeds relative thresh
L_KG = -0.691 + 10*log10(chwat * sum(zz(Jg,:), 1)' / sum(Jg)); % calculation
for relative loudness amount
```

Now that the signal has been analysed for apparent loudness, a brickwall feed-forward peak limiter (Zölzer, 2002) is applied. The limiter itself is designed to hard-limit any signal exceeding the limiter threshold (LT). As the plugin is designed for broadcast and the levels will have already been adjusted to reach the LKFS target (default -24dB), it is not likely the limiter will be required apart from to tame the occasional transient peak. The recommended maximum peak for broadcast deliveries in Australia is -2dBFS, which ensures there is a small amount of headroom to account for any inaccuracies in metering. This is the default value for the limiter if the user defines no other amount.



AT = Attack Time
RT = Release Time

LT = Limiter Threshold
LS = Limiter Slope

Figure 4. *Block diagram of a limiter*

The limiter's signal processing, seen in Fig. 4, is a feed-forward design in which the input signal *x(n)* is split into a delayed component *x(n-D)* and a side chain path. The side-chain processing is done using the following code:

```
for n=2:length(x_lkfs1)
   a=abs(x_lkfs1(n,1))-x_peak(n-1); % delays signal by n-1
   if a<0, a=0; end;
   x_peak(n) = x_peak(n-1)*(1-RT) + AT*a; % Peak detector depending on AT/RT
   if x_peak(n)>lt, % if the signal exceeds the threshold...
       % ...run the limiter processor:
      f_1(n)=10^(-LS*(log10(x_peak(n))-log10(lt))); % linear calculation of…
   else f_1(n)=1; % ...otherwise let the gain factor remain as 1
   end;
end;
limit_out(:,1)=x_lkfs1(:,1).*f_1; % apply gain adjustment
```

Firstly the variable (`n`) is defined by the input (`x_lkfs1`). The absolute value of the signal is found and then delayed by 1. The peak detector algorithm, which is dependent on whether the limiter is in attack (`AT`) or release (`RT`) mode, is then applied. If the signal is greater than the threshold (`LT`) then the limiter processing is performed to determine the gain factor (`f`). The processing of the gain factor is the linear calculation of the following equation:

$$f(n) = 10^{-LS(X-LT)} \tag{3}$$

where X is the logarithmic value of x (Zölzer, 2002)

The final calculation of the limiter is the multiplication of the input signal by the gain factor. The processing is then repeated if a second channel is present.

If any limiting is applied then the overall LKFS value of the signal will have changed; meaning that the signal must undergo another level match to ensure the target LKFS value is met. This second matching process only runs if the analysis determines that the limiter input signal's peaks are greater than the limiter output.

Further Development:

There is further room for development of this processor in the future: the first and most obvious being the design of a graphical user interface (GUI) containing various preset settings that ensure the user experience is both fast and intuitive.

Support for different file formats could be easily implemented, along with better support for surround sound files. Up to 5.1 channels processing is in fact already implemented in the 'loudness_itu.m' function. However testing revealed that Matlab's 'wavread.m' function does not currently support more than 2 channels of audio; an issue that could be easily overcome with more development.

With further research, a more transparent limiter could be introduced that has intelligent auto-release for less audible transient control. This feature is not too necessary for the current application, but would be ideal if the processor were to be developed for real-time analysis in DAWs and not just post-export corrections.

References:

Abdulla, K., 2007. *Matlab Central: Clipping Detection*. The Mathworks Inc. [ONLINE] Available at: http://www.mathworks.com.au/matlabcentral/newsreader/view_thread/159887. [Accessed 15 May 13].

Free TV Australia 2010, Operational Practice OP-59: Measurement and Management of Loudness in Soundtracks for Television Broadcasting, Issue 1, Sydney.

International Telecommunication Union 2012, Recommendation ITU-R BS.1770-3: Requirements for loudness and true-peak indicating meters, International Telecommunication Union, Geneva.

International Telecommunication Union 2012, Recommendation ITU-R BS.1771-1: Algorithms to measure audio programme loudness and true-peak audio level, International Telecommunication Union, Geneva.

Marui, A., 2012. *Loudness calculation based on Rec. ITU-R BS.1770-2*. [ONLINE] Available at: http://www.geidai.ac.jp/~marui/matlab/node42.html [Accessed 22 April 13]

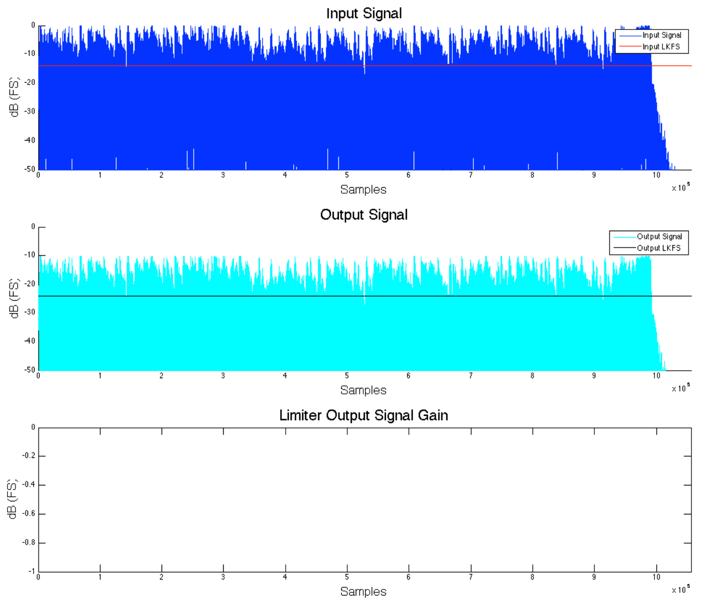
Zölzer, U, 2002. *DAFX - Digital Audio Effects*. 1st ed. West Sussex, England: John Wiley & Sons, Ltd.

Zölzer, U, 2008. *Digital Audio Signal Processing*. 2nd ed. West Sussex, England: John Wiley & Sons, Ltd.

Appendix:
Please find the following supplementary audio files attached:

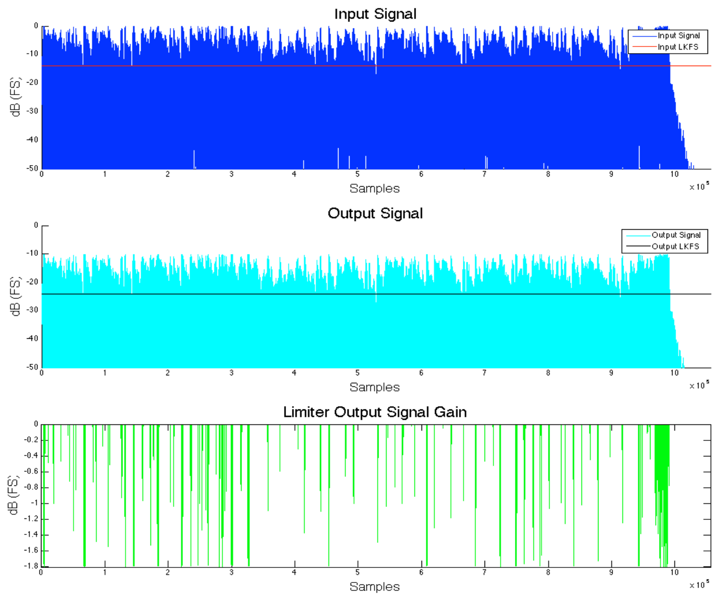
Input Files	Channels	Output Files	Broadcast Level Corrector Settings	
			Target LKFS (dB)	**Limiter Threshold (dB)**
input_eg1_film_stereo.wav	Stereo	output_eg1_film_stereo.wav	-24	-2
input_eg2_music_mono.wav	Mono	output_eg2a_music_mono.wav	-24	-2
		output_eg2b_music_mono.wav	-24	-10

Table 2. *Input audio files vs. output audio files and the BLC settings used during processing*



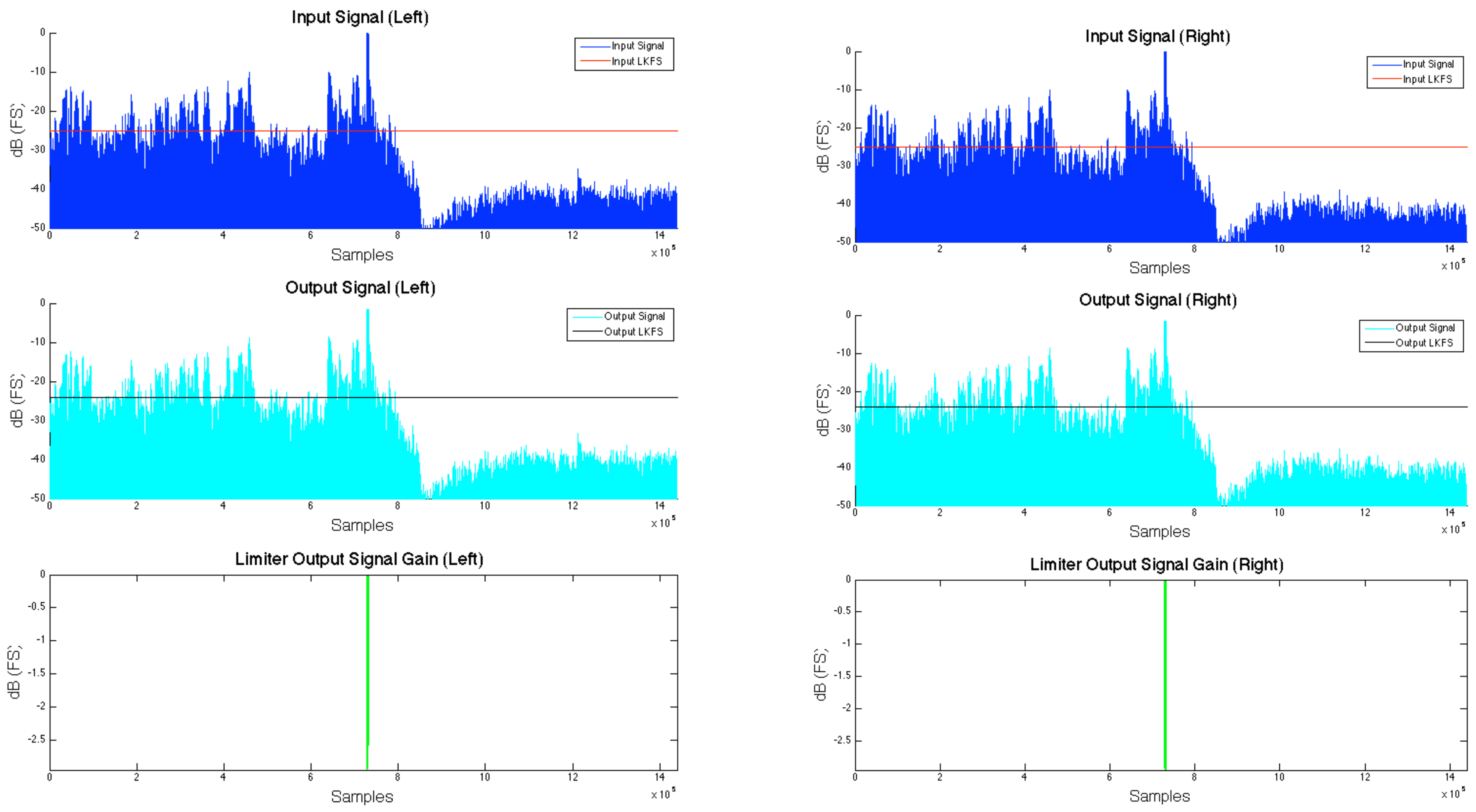
Input file: input_eg2_music_mono.wav

Figure 5. *Plotted graph showing the processing of a loud mono music input signal. The lines across the audio show the LKFS level of the input and output files. As is evident from the limiter graph, no limiting incurred. This file did however have clipping in the raw data, prompting the Clipping Warning in Matlab.*



Input file: input_eg2_music_mono.wav

Figure 6. *The same input as Fig. 6 is shown here with the target loudness level still -24 dB (LKFS), but the limiter threshold set to -10dBFS.*



input_eg1_film_stereo.wav

Figure 7. *Plotted graphs showing a stereo audio signal pre-processing and post-processing. The input signal was a 30 second snippet from a short film. The bottom graph shows the Limiter Output Signal Gain. This signal had a distinct peak toward the middle, which after the LKFS gain increase would have resulted in clipping if the limiter were not present.*