Lab Report 1
McLean Pierce SID: 311267165
Digital Audio Systems, DESC9115, Semester 1 2012
Graduate Program in Audio and Acoustics
Faculty of Architecture, Design and Planning
The University of Sydney

Introduction

The objective of this exercise was to explore the use of existing Matlab code related to the implementation of audio dynamics compression. As such, I auditioned compression code provided in two different editions of the book *DAFX – Digital Audio Effects* and developed a Matlab script to call the compression function and generate a set of figures to graphically demonstrate the results of the effect.

Technology Review

I found the compression code from the first edition of *DAFX* (M-file 5.3) to be insufficient and very limited compared to most hardware and software compression implementations. The first edition code did not include any implementation of attack and decay time, nor any clear method of setting a compressor threshold. Using this code resulted in a highly distorted output signal. As such, I went looking for other Matlab implementations of dynamics compression and found that the second edition of *DAFX* included an updated compression code (M-file 4.2) that was more sophisticated and included attack and decay time parameters as well as time-averaged rms level detection and a clear method for setting the compression threshold level and compression slope. As such, I took this code as the base of my compression function.

Methods and Results

The code was originally written by M. Holters so I built a function called holtcomp.m that includes only the compression part of the function (the original included compression and expansion) and an additional means for taking attack and decay time as function inputs.

The holtcomp.m function takes five arguments as inputs and is called as follows: y=holtcomp(x, CT, CS, at, rt). 'x' is the input sound vector, 'CT' is the compression threshold specified in (-)dB, 'CS' is the compression slope (a value between 0 and 1) that is applied to signal above the threshold , 'at' is the attack time in seconds and 'rt' is the release time in seconds. The function returns a single output 'y'

which is a vector containing the resulting compressed sound file. The function acts by detecting the rms level of the input signal and then applying a gain reduction factor that is determined for each sample based on attack and decay times and a time averaging factor.

The primary work of the holtcomp function is carried out by the gain equation which is G=CS(CT-X) if X>CT. Effectively this equation generates a negative value when the signal level (X) exceeds the compressor threshold (CT). The difference of the signal level and the threshold is then multiplied by the compressor slope. Next, this value G is fed into the equation f=10^(G/20) to determine the multiplying factor to reduce the level of the input sample. All of this is implemented with a time averaging and smoothing factor. Signal flow in the system is represented in Figure 1.

To control this function I built a script called compscript.m which calls the function holtcomp.m and implements a number of other tasks useful in evaluating the function. This script loads the input sound file into a vector, cuts it down to a manageable size, and normalizes it. Then the script takes the input values and calls the holtcomp function. The script then normalizes the holtcomp output and stores it as a wav file. Finally, the script generates two figures that are useful in determining the operation of the function. First, it generates a figure showing a graph of input vs output values for the function, which allows us to see a typical compression curve (see Figure 2). Second, the script generates a time vs amplitude plot for the original sound file, the compressed sound, and the compressed and normalized sound (see Figure 3) which is useful in visualizing the effect that different compression settings have on the sound. Figures 2 and 3 use a threshold setting (CT) of -20dB and a slope (CS) of 0.75. We can clearly see that the compressed sound with makeup gain (normalization) has more power than the original uncompressed sound, and the graph of the compressed but un-normalized sound allows us to see how the compressor functions in reducing the peaks of the original sound.

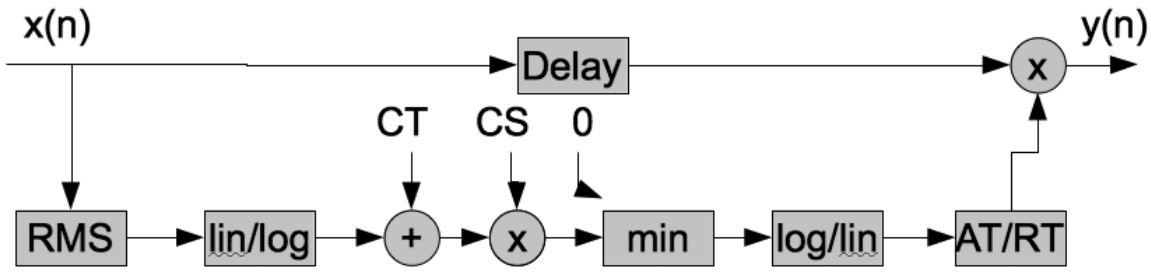

Figure 1: Compression Signal Flow (adapted from DAFx 1st ed.)

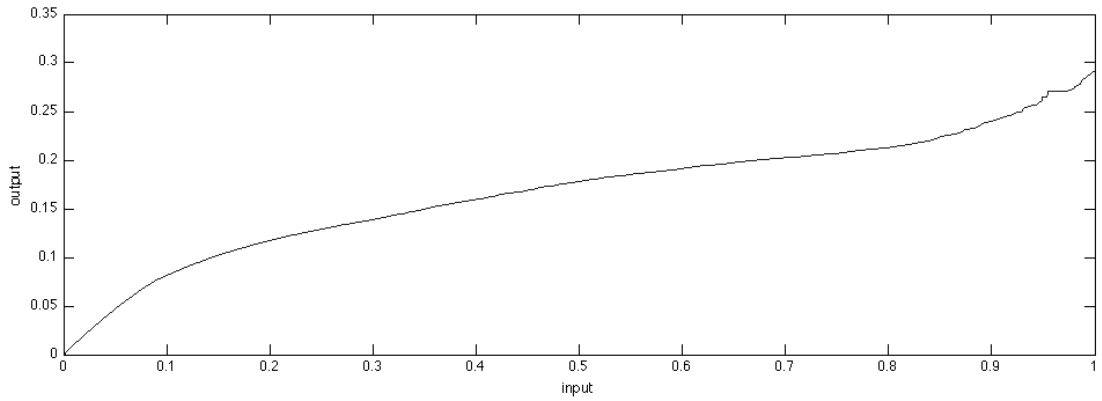

Figure 2: Compression Curve

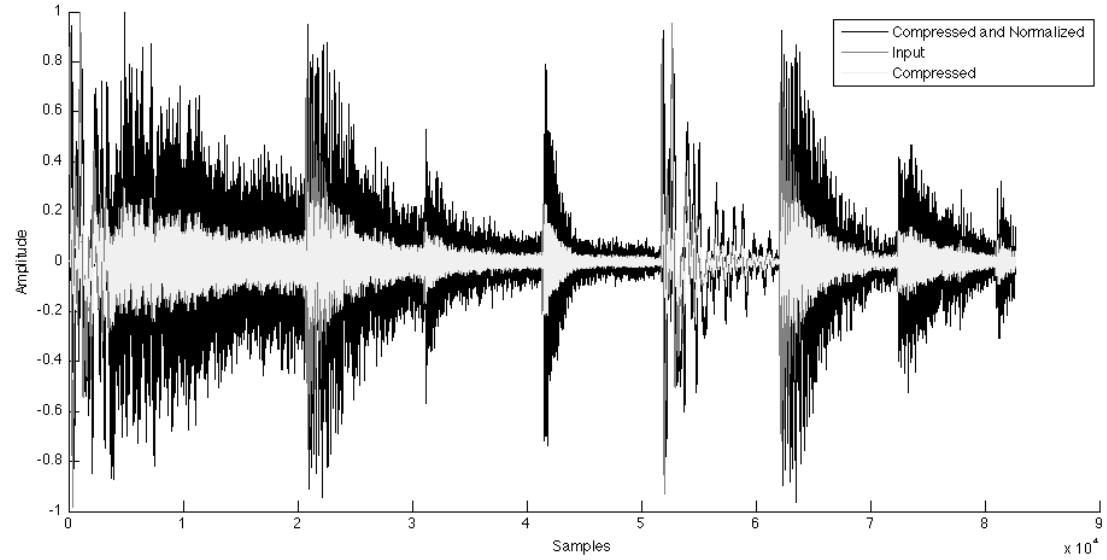

Figure 3: Input and Output Data

Bibliography

Zoelzer, Udo, and Xavier Amatriain. "Chapter 4.2.2 Compressor and Expander" *DAFx: Digital Audio Effects*. 2nd ed. Chichester, England: Wiley, 2011. Print.

Zoelzer, Udo, and Xavier Amatriain. "Chapter 5.2.2 Compressor and Expander" *DAFx: Digital Audio Effects*. 1st ed. Chichester, England: Wiley, 2002. Print.