Karol Wadolowski
ECE-413 Music and Engineering

Homework #4 – Effects

In this assignment, various types of musical effects were implemented. The implementation will be explained as well as the results and my opinion of the effect.

Compressor

The compressor function takes four inputs: the signal to compress, the threshold, the slope, and the averaging length. If the average power in the signal is over the threshold then the gain of the signal over some future samples will be decreased, how much depends on the slope value.

The MATLAB code for the compressor first averages the (let's say averaging length is 50 samples) first 50 samples. This averaging is over the signal power and the signal is treated as a voltage value, i.e. the mean of the squared samples is taken as the average power. If the average power is greater than the threshold than samples 26 to 75 have there gain decreased. So, the average power of samples 1 to 50 affects the gain of 26 to 75. The next averaging is over the 26^{th} to 75^{th} samples and the outcome affects the gain for the 51^{st} to 100^{th} samples. This continues till the end of the signal. By default, the gain of all the samples starts at 1 and can only decrease with this scheme.

The gain is applied exponentially over the 50 samples that are having their gain decreased. This gives a smoother transition in the signal instead of a sudden jump. An example of a signal being compressed can be seen in Fig. 1. When listening to the original and compressed versions it is quite easy to hear the difference (with the values I have set in the MATLAB code).

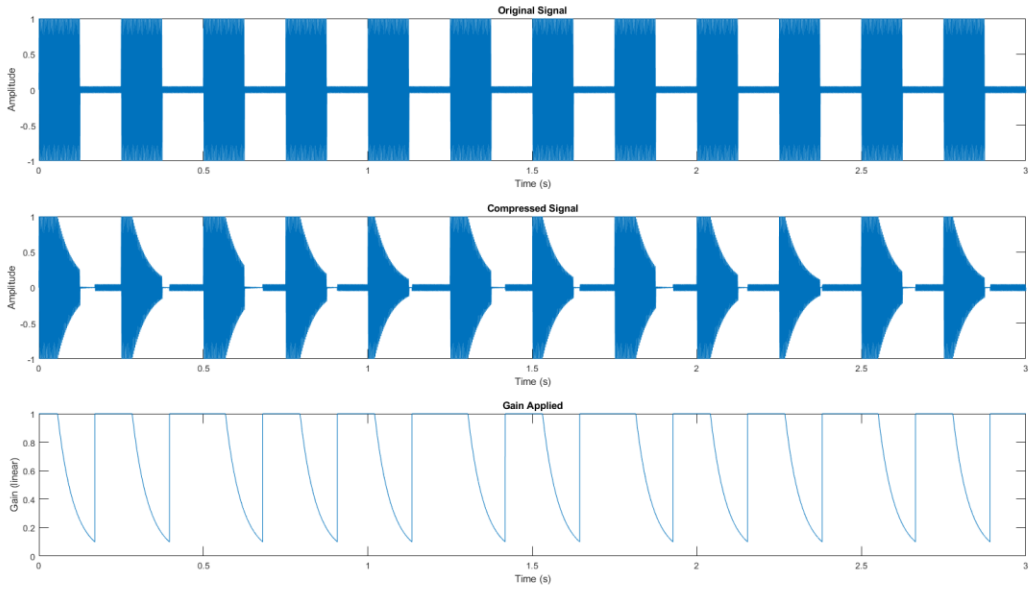

Figure 1: The original signal on top, the compressed signal in the middle, and the corresponding gain for each sample on the bottom.

Ring Modulator

The ring modulator function takes three inputs: the signal to modulate, the modulation frequency, and the mix depth. The input signal gets mixed with a sine wave at the modulation frequency. Depth then controls how much of that mixed signal is added back on to the original.

The MATLAB code uses the signal vector to find the duration that the sine needs to be and generates the mixing sine wave signal. The input signal and sine wave are then mixed. An amount, specified by the depth, is then added back onto the original signal.

Though this effect does sound bad for most configurations, I managed to find one that I think worked well on the clip I chose. I enjoyed hearing the beats that came from the mixing on the specific instance in the MATLAB script.

Stereo Tremolo

The stereo tremolo function takes five inputs: the signal, the oscillator type (LFOtype), oscillator frequency (LFOrate), lag, and depth. The oscillator type affects the waveform of the low frequency oscillator (LFO). The available types are sin, triangle, and square. LFOrate

controls the frequency of the oscillations. The lag controls when the LFO turns on and starts taking affect. The depth is the same as it was in the Ring Modulator.

The MATLAB code works by treating the time up to the lag as time zero (oscillator output is zero until the lag time is up). After that point time resumes and the appropriate LFO is generated. The signal is then mixed with the LFO signal and that result is added onto the input signal by an amount controlled by the depth.

I choose to test this on a bell sound. Before the affect only one distinct peak is present. After the affect you can hear the bell dying out (sort of like an underdamped system in a way). The sound is pretty cool and more interesting than the original.

Distortion

The distortion function takes three inputs: the signal, the gain, and the tone. The gain controls how much the signal is amplified before it goes into a clipping function. The tone controls the cutoff frequency of the low pass filter (LPF) that the clipped signal is fed into.

For my implementation I made the clipper the function in Fig. 2. The input is amplified before being fed into the clipper function. After being clipped the signal is fed into a LPF specified by the tone. I designed the clipper like this so it would be less linear than a simple single slope with outer thresholds. These extra discontinuities add more places were more harmonics can appear which will distort the signal more.

Unfortunately, I wasn't able to make my own LPF that I was happy with (the remnants of one of my attempts is purposely commented out in the MATLAB code). So, I used the built MATLAB function that generates a lowpass at a specific cutoff frequency and filters the input signal. The function is called lowpass.

I obtained 4 spectrograms corresponding to 4 different gain settings. As more gain is added you can more clearly see more frequencies becoming larger in magnitude. These spectrograms can be seen in Fig. 3.

I found a cool distortion on my bell sound that gave it a weird, slightly more metallic and high frequency sound. It also sounds like the main lobe is being dragged out longer.

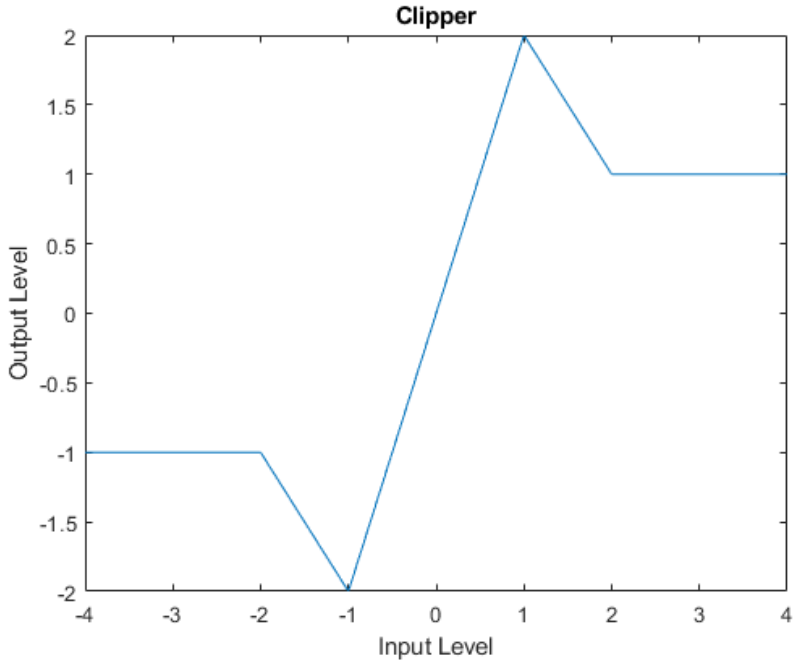

Figure 2: The clipper waveform for the distortion affect.

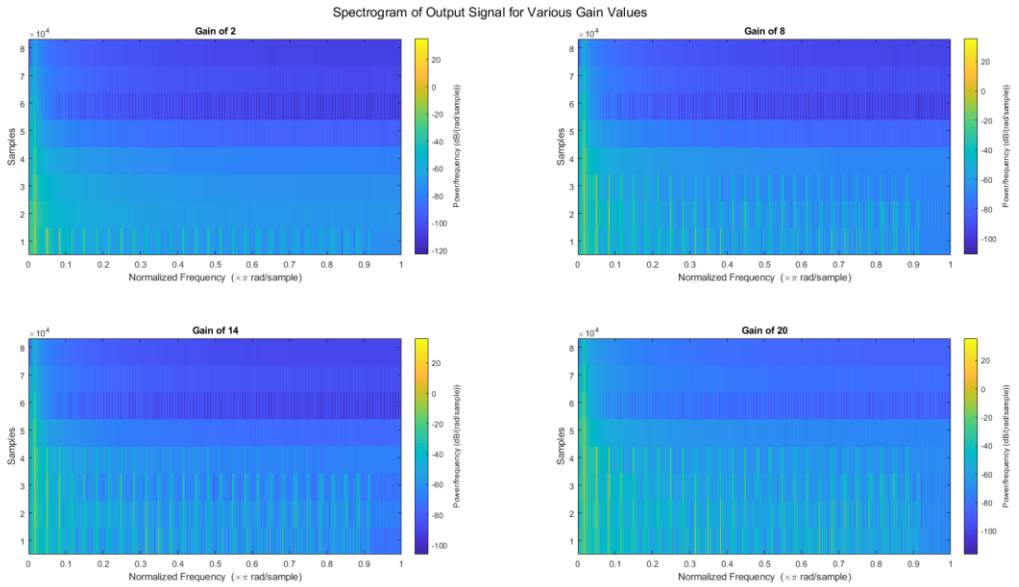

Figure 3: The output signal's spectrogram when a bell sound was distorted. At higher gains more frequencies are present.

Single Tap Delay

The single tap delay function takes 4 inputs: the signal, the depth, delay time, and feedback gain. The delay time is how long the copy of the signal is delayed before being added back on to the original. The feedback gain is how much of the prior samples is added back to the current sample.

In MATLAB, I first convert the delay time to number of samples (for indexing ease). I then set the effect vector equal to the input vector but offset that many samples. Then starting at twice the delay time, I calculate the effect signal by adding the effect signal value from a delay time prior and scaling it by the feedback gain. I also add on the value it was supposed to be if feedback wasn't present.

Slap back: I had the feedback set to zero for this and had the delay time set to a low value as to keep the delayed version almost on top of the first one.

Giant Cavern: I had a feedback gain of a half to volume drop on the 'echoes'. I also had a longer delay time as to be able to distinguish the repeats from themselves.

Delayed Tempo: I chose the feedback and delay such that the delayed versions would seem like extra notes that belonged in the audio clip.

Of these the giant cavern and delayed tempo came out sounding the best. The giant cavern sounded surreal and the delayed tempo made the drum track so much livelier.

Flanger

The flanger function takes six inputs: the signal, depth, delay time, sweep depth, LFOrate, and LFOtype. (Yes, I added LFOtype because it was convenient) The sweep depth is a variable that determines the amplitude of the oscillating delay time. The other inputs are the same as they were in other effects.

In the MATLAB code I created a vector that stored the current delay time. This was calculated by adding on an oscillating waveform to my constant delay time signal. This meant that certain instances there would be more delay than others. Then I looped through the changing delay time vector and created the effect signal. At each instance in the loop I checked if the delay time was shorter than the had already passed in the signal. If the more time had passed than the length of the delay, then the output was equal to the input but delayed that amount.

For example, if the current delay time was 50 samples and we were on sample 60 then the output of the effect at sample 60 was the input at time 10. If the delay time was more than 60 samples than the output of the effect was 0.

At the end a certain ratio of the effect was added back to the input, based on the value of depth.

In the end I choose triangle for my waveform because I prefer the jaggedness it adds in the output more than the smoothness associated with the sine wave flanger. It also seemed to do slightly more on the drums when I tested it.

Chorus

I hope what you heard was chorus because I'm not certain it was.