BELMONT UNIVERSITY, CURB COLLEGE OF ENTERTAINMENT AND MUSIC BUSINESS

AET 5420 Homework 1

Audio Signal Processing

Due: January 31, 2024

1 MID-SIDE DISTORTION

1.1 HARD-CLIPPING DISTORTION EFFECT

Hard-clipping is a type of distortion effect where the amplitude of a signal is limited to a maximum (and minimum) amplitude. This effect is created in analog circuits when a transistor is pushed to a maximum amplitude. At this amplitude, the transistor saturates and cannot output a signal above a specific level, so the input signal is hard clipped.

This type of distortion effect can also be created using software. This is accomplished by detecting when the amplitude of a signal goes above a specified threshold. If the amplitude of the input signal goes above the threshold, then the amplitude of the output signal clipped.

Don't forget that audio signals have positive and negative amplitude (for compression and rarefaction). In the case of hard-clipping, there should also be a minimum specified amplitude. If the amplitude of the input signal goes below the negative of the threshold, then the amplitude of the output signal is also clipped.

An equation to determine the amplitude of a processed or clipped, output signal is the following:

$$output(n) = \begin{cases} thresh & \text{if } input(n) \text{ is > thresh} \\ input(n) & \text{if -thresh < } input(n) \text{ < thresh} \\ -thresh & \text{if } input(n) \text{ is < -thresh} \end{cases}$$

1.1.1 Problem

Create and save a **function** (m-file) in MATLAB that performs the following steps:

- Name the function: hardClip.m
- The function should have the following input/output variables:
  - inputSF (input variable representing input signal)
  - thresh (input variable representing clipping amplitude)
  - outputSF (output variable representing processed signal)
- Use the following recommended commands within the function:
  - Create a 'for' loop to step through each sample of the *inputSF*.
    * for sample = 1:length(inputSF)
    * If the amplitude of inputSF(sample,1) is greater than *thresh*, then set the amplitude of outputSF(sample,1) equal to *thresh*.
    * If the amplitude of inputSF(sample,1) is less than -*thresh*, then set the amplitude of outputSF(sample,1) equal to -*thresh*.
    * Otherwise (*else*), the amplitude of the outputSF(sample) should equal the amplitude of inputSF(sample,1).
    * Remember to *end* your loop
- Use the test script, *distortionTest.m* to ensure proper performance of your function.
- If your function works correctly, your hard-clipping effect should produce the results shown in Fig. 1.1.

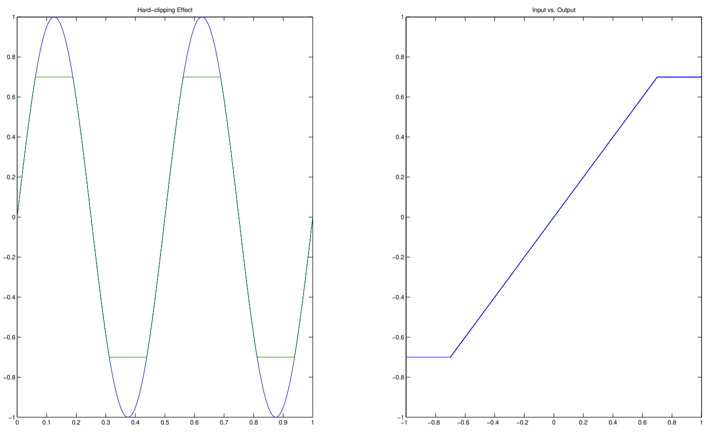


Figure 1.1: Hard-clipping Effect

Remember to add comments to your code to explain what each command is accomplishing. For this problem, you will submit the function file - **hardClip.m**

1.2 Mid-Side Processing

Mid-side processing is based on the decomposition of a stereo signal into separate *mid* and *side* signals. After the signals have been separated, each can be processed independently. During the mastering process of a recorded song, it is common to process the instruments or vocals in the center of a mix differently than the instruments on the side. An example of this would be filtering out low frequencies from the side channels to focus the bass guitar and kick drum in the center of a mix. Another example would be compressing the center of a mix separate from the sides to prevent the stereo field from being compressed asymmetrically.

1.3 Mid-Side Background

When a listener perceives a stereo signal played back over stereo speakers, they do not simply hear the sound coming out of each speaker. A listener can perceive a "phantom center," when the sound appears to be produced between the two speakers. The "phantom center" occurs when the same signal is presented to both ears simultaneously. Because the signal is perceived in the center, it is said that the signal is in the *middle*, or between the speakers. If a signal is only presented from the left speaker or only presented from the right speaker, then a listener perceives the sound as coming from the same *side* as the speaker.

A stereo signal is created from a mono signal by panning the mono signal to create a stereo signal. If an engineer would like to have a signal be perceived by a listener on the *side*, they will pan the signal to the left channel or the right channel. In this simple case, the mono signal will only be presented in one speaker, but will not be presented to the other speaker (more importantly: each speaker receives a different signal). If an engineer would like to have a signal be perceived by a listener in the *middle*, they will pan the signal to the center. In this case, the identical mono signal will be presented to both the left and right speakers simultaneously (more importantly: both speakers receive the same signal).

In other words: if a signal is perceived on the *side*, the left and right channels are different. If a signal is perceived in the *center*, the left and right channels are identical.

By subtracting the signal on the right channel from the signal on the left channel, any identical signals on both channels will cancel out by *destructive interference*. Therefore, any signals that were panned to the center will be removed, leaving only the signal on the *sides*. By adding the signal on the right channel to the signal on the left channel, any identical signals on both channels will increase in amplitude by *constructive interference*. Therefore, any signals that were panned to the center will be louder.

1.4 Encoding

To encode the *mid* and *side* signals from a conventional stereo file with left and right channels, the following equations can be used:

$$\begin{aligned} side &= \frac{1}{2}(left - right) \\ mid &= \frac{1}{2}(left + right) \end{aligned} \tag{1.1}$$

1.5 Decoding

To decode, or perfectly recover, the original *left* and *right* channels from the *mid* and *side* signals, the following equations can be used:

$$\begin{aligned} left &= mid + side \\ right &= mid - side \end{aligned} \tag{1.2}$$

Remember:

$$\begin{aligned} left &= mid + side = (\frac{1}{2}L + \frac{1}{2}R) + (\frac{1}{2}L - \frac{1}{2}R) = \frac{1}{2}L + \frac{1}{2}L \\ right &= mid - side = (\frac{1}{2}L + \frac{1}{2}R) - (\frac{1}{2}L - \frac{1}{2}R) = \frac{1}{2}R + \frac{1}{2}R \end{aligned} \tag{1.3}$$

Therefore, a stereo signal can be decomposed into *mid* and *side* signals, and still be perfectly reconstructed.

1.6 Mid-side Processing

Engineers can process the *mid* signal independently of the *side* signal by using audio effects (EQ, Compression, Distortion, etc) after encoding the *mid* and *side* channels, but before decoding the *left* and *right* channels.

As one example, an engineer could add hard-clipping distortion to the *mid* channel without adding it to the *sides* channel.

1.7 Problem

Create and save a **script** (m-file) in MATLAB that performs the following steps:

- Name the script: **msDistortion.m**
- Import the audio file: stereoDrums.wav
  - Next, separate the stereo signal into a mono, 'left' signal and a mono, 'right' signal
- Perform Mid-side Encoding

  - Use equations 4.1 to create new variables *mid* and *sides*
- Perform hard-clipping distortion on the *mid* channel
  - Use your **hardClip.m** function
  - Use a threshold of 0.4
  - Store the output signal as a new variable
- Perform Mid-side Decoding
  - Use equations 4.2 to create new variables *leftNew* and *rightNew*
  - Make sure to use the distorted version of the *mid* signal
- Create a single variable to store the stereo audio signal
- Write this stereo audio signal to a new file - **distDrums.wav**

Remember to add comments to your code to explain what each command is accomplishing. For this problem, you will submit the script file - **msDistortion.m** and the sound file - **distDrums.wav**.

2 Transistor Clipping

The transistor is a circuit component used in many analog audio systems. It is one of the components in a circuit that can distort an in-coming signal, if the signal's amplitude is great enough. In many ways the transistor behaves like a hard-clipper. When the amplitude of the in-coming signal is relatively low, the transistor behaves linearly over this operating region. When the signal becomes too high, the transister clips the signal off abruptly.

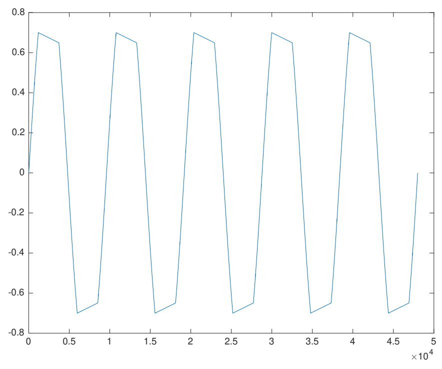


Figure 2.1: Wave of Signal with Transistor Clipping Effect

In many analog circuits, the transistor(s) show a unique behavior where the threshold of clipping is **not** constant. Instead, the threshold starts relatively high and then slowly decreases over time. This creates a gradual slope to the clipping effect, rather than a flat, constant level. This can behavior is modeled in the AIR Distortion plug-in with the "Hard" setting. The waveform of a sine-wave signal after being distorted in this manner is shown in Figure 2.1.

To model this effect digitally, there are several parameters to control. First is the threshold, similar to the hard-clipping distortion. Second is the rate at which the threshold decreases as long as the signal stays in the distortion region. You can use a variable which decreases the threshold by a very small number whenever the amplitude of the input signal is greater than the threshold (or less than -threshold). When the amplitude of the signal is within the region of linear operation, the threshold level should reset to its original value.

2.1 PROBLEM

Two sound files are provided for you to use in this problem. One file is a bass DI track recorded without distortion. The second file is the distorted bass signal created by sending the DI track through an LA-3A with the Gain at 10 and Peak Reduction at 0. The goal of this problem is to create an effect which replicates this distortion.

Create a **function** and test **script** (m-files) in MATLAB that perform the following steps:

- Name the function: transistorClipping.m
- The function should have the following input/output variables:
  - in (input variable representing input signal)
  - thresh (input variable representing clipping amplitude)
  - out (output variable representing processed signal)
- Create a variable within the function to control the slope of the threshold
- Experiment with different values for this variable until the distortion resembles the example files
- Follow the recommend approach above for modifying the hard-clipping effect (loop with conditional statements) to achieve the transistor-type distortion
- Write your own test script for the function to demonstrate proper performance

Remember to add comments to your code to explain what each command is accomplishing. For this problem, you will submit the function file - **transistorClipping.m** and the **test script**.

3 Parallel Distortion

A common feature of some software distortion effects is the ability to blend between the “dry”, unprocessed signal and the “wet”, distorted signal. This feature can be created by using parallel distortion. The relative amplitude of each path is controlled by the gain scalars, g_1 and g_2. An example of the block diagram for this effect is shown in Fig. 3.1.

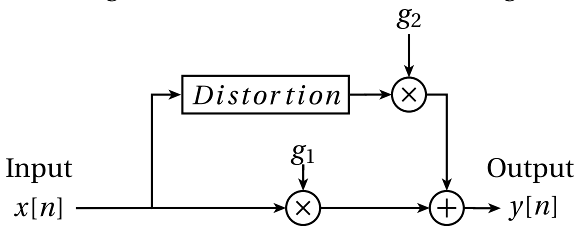
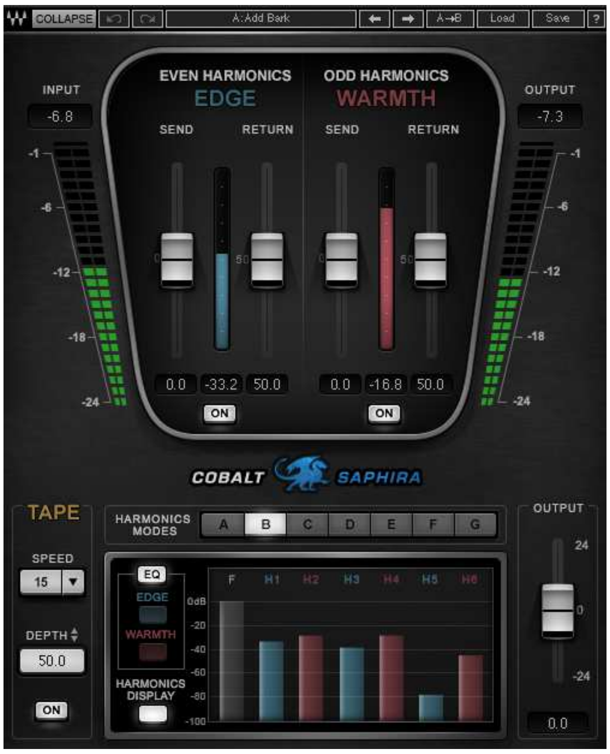


Figure 3.1: Block Diagram of Parallel Distortion

3.1 Waves Cobalt Saphira

The *Waves Cobalt Saphira* plug-in is a distortion effect with the ability to change the relative amount of even and odd harmonic distortion. The interface for the plug-in with independent controls for even harmonics and odd harmonics is shown in Fig. 3.2.



INPUT -6.8

EVEN HARMONICS EDGE: SEND 0.0, -33.2, RETURN 50.0, ON

ODD HARMONICS WARMTH: SEND 0.0, -16.8, RETURN 50.0, ON

OUTPUT -7.3

TAPE: SPEED 15, DEPTH 50.0, ON

HARMONICS MODES: A B C D E F G

EQ, EDGE, WARMTH, HARMONICS DISPLAY

F H1 H2 H3 H4 H5 H6

OUTPUT 0.0

Figure 3.2: Waves Cobalt Saphira Interface

One approach to implementing this type of effect uses parallel distortion. As shown in class, full-wave rectification (i.e. absolute value function) creates even-order harmonic distortion. Additionally, infinite clipping creates odd-order harmonic distortion. Each of these distortion effects can be performed in parallel for independent control of each type. A block diagram for this effect is shown in Fig. 3.3. The gain values (g_1 , g_2, and g_3) are used to change the relative amplitude of each parallel path.

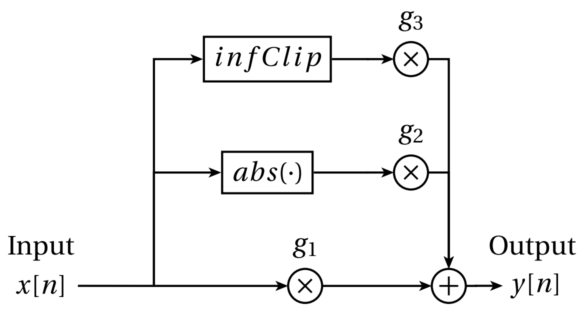


Figure 3.3: Block Diagram of Even and Odd Distortion

3.2 Problem

Create and save a **script** (m-file) in MATLAB that performs the following steps:

- Name the script: **parallelDist.m**
- Choose your own audio file to process
  - For the sake of processing time, use a clip shorter than 15 seconds
  - You will include this file in your homework submission
- Create 3 gain variables (g_1 , g_2, and g_3)
  - Experiment with different values for each variable between 0-1
- Perform the parallel distortion as described in Fig. 3.3.
  - Use full-wave rectification and infinite clipping
  - Then scale the amplitude of the processed signals using g_1 , g_2, and g_3
- Combine all the different paths together to create a single output signal

Remember to add comments to your code to explain what each command is accomplishing. For this problem, you will submit the script file - **parallelDist.m** and the sound file you used.

4 Power Series Distortion

Rewrite the following soft-clipping distortion algorithm by substituting trigonometric identities for the power terms. Consolidate terms in the final expression.

$$y[n] = x[n] - \frac{1}{3}(x[n])^3 + \frac{1}{5}(x[n])^5$$

5 Submission

To submit your homework, create a single zip file that contains your MATLAB files, audio files, and photo of your hand-written work. Name the zip file: xxxxx_AET5420_HW1.zip, where *xxxxx* is your last name. Email the zip file to: eric.tarr@belmont.edu before the start of class on January 31st.