MUS424:
Signal Processing Techniques for Digital Audio Effects Handout #4
Jonathan Abel, David Berners April 5, 2012

Homework #1: Basic Compression
Due Date: April 12, 2012

Submission Instructions

Submit via coursework's DropBox. Create **a single compressed file** (`.zip`, `.tar` or `.tar.gz`) containing all your submitted files. Place the compressed file at the top level of your Drop Box. Name the file using the following convention:

```
<suid>_hw<number>.zip
```

where `<suid>` is your Stanford username and `<number>` is the homework number. For example, for Homework #1 my own submission would be named `jorgeh_hw1.zip`.

Each problem should be in its own subfolder, ideally named `problemX`. Therefore, if different problems ask to to modify the same code, you'll have to turn in different projects, one for each problem.

For coding problems (either C++ or Matlab code), submit all the files necessary to compile/run your code, including instructions on how to do it. In case of theory problems, submit the solutions in **PDF format only**. LaTeX or other equation editors are preferred, but scans are also accepted. In case of scanned handwriting, make sure the scan is legible.

Problem 1. [50 points]

In this lab you will modify a basic compressor and experiment with the level detector. The file `Lab1_<OS>_VST.zip` contains code for a basic compressor—a feed-forward compressor with a peak detector and a gain computer configured as a limiter. The code implements several sliders,

- Input Gain, Output Gain and Threshold,

which are used to control the input and output levels and threshold of compression for the basic compressor, and

- Attack Time and Release Time,

which define the time constants in the peak detector. Two additional sliders are present, but not used in the basic compressor,

- Detector Exponent and Compression Ratio.

You will use these sliders to develop more sophisticated compressors later in this lab and in the next lab.

1(a). [20 points] Configure your development environment and digital audio workstation, perhaps with help from Jorge or fellow students. Compile the basic compressor, and listen to its effect on the audio tracks provided, a short drum track, guitar track and a test signal called `tdiff1.wav`. To give you confidence that your plug-in is working as intended, the plug-in `BasicLimiter` is a running version of the basic compressor.

Note: due to licensing restrictions we cannot distribute the VST SDK, so the code won't compile out of the box. You'll need to get the VST SDK 2.4 from Steinberg's website (`http://www.steinberg.net/nc/en/company/developer/sdk_download_portal.html`) and place the needed files inside the project's `vst_sdk` folder (`Readme.txt` files are provided inside that folder with more details).

1(b). [20 points] The test signal `tdiff1.wav` contains a 1.0 kHz sine wave with a stepped amplitude, starting out at a level of 0.09 for one second, jumping (at a zero crossing of the sine wave) to a level of 0.9 for one second, and then back to a level of 0.09 for one second.

Set the Input Gain and Threshold of compression (which is defined with respect to the input signal, scaled by the input gain) so that the initial level of the test signal, 0.09, is at the threshold of compression. Set the Attack Time to 10 ms and the Release Time to 200 ms. Capture the output of the basic compressor in response to the test signal, and make two Matlab plots,

- one showing the output signal around the time of the attack, with the attack time labeled, and

- one showing the output signal around the time of the release, with the release time labeled.

Make a rough sketch showing the envelope of the input signal and the detected level as a function of time for the test signal input.

Finally, set the Attack Time to 1 ms and the Release Time to 50 ms, and make another pair of plots showing the attack and release.

1(c). [10 points] Using the drum signal as your input, setting the Release Time to 200 ms, and setting the Threshold so that there is about 6 dB of compression indicated on the gain reduction meter, adjust the Attack Time from 0.1 ms to 100 ms, and listen to the result. At what values of attack time does the compressor make the drums sound

- smooth,

- tinky,

- thuddy,

- transparent?

Now, set the Attack Time to 0.1 ms, and adjust the Release time from 20 ms to 200 ms. At what values of release time does the compressor make the drums sound

- buzzy,
- roomy,
- even (each hit having equal loudness)?

Note that references for these values can be found in the Course Notes, but the idea is for you to evaluate your own compressor. Maybe your perception differs from the Course Notes authors!

Problem 2. [**50 points**]

Modify the basic compressor by replacing the peak detector with an RMS detector. Use the Release Time slider to set the time constant of the leaky integrator. (The Attack Time slider setting will be unused.) Plot the output of the RMS compressor in response to the `tdiff1.wav` test signal with the Release Time set to 200 ms. How does this output envelope compare to that of the attack/release basic compressor? Very briefly, how does this compressor sound on drums or guitar compared to the basic compressor when the attack time of the basic compressor is set to 0.1 ms, and the release time of the RMS compressor and basic compressor are the same? (Note that you might want to use the `BasicLimiter` plug-in for comparison. to the peak detector case.)

Problem 3. [**60 points**]

Now, modify the RMS detector of the previous problem to implement a running p-norm of the input. In other words, replace your RMS detector with a detector that forms the pth root of the smoothed pth power of the absolute value of the input signal. Here, the smoothing is a leaky integrator described in the notes, having transfer function

$$H(z) = \frac{1-a}{1-az^{-1}}, \tag{1}$$

where $a \in [0, 1)$ is determined by the Release Time slider value. Use the Exponent slider to determine the value for $p \in [1, 10]$.

3(a). [**30 points**] Submit your modified code, and plots of the response of your modified compressor to the test signal for three cases, $p = [2, 5, 10]$, for a Release Time of 100 ms.

3(b). [**30 points**] Note the similarity between this "RMp" compressor and a compressor with a peak detector having separate attack time and release time controls. With larger values of p, the attack becomes quicker and the release becomes slower. In this way, a compressor with separate attack and release time controls can be made using a linear time-invariant filter and memoryless nonlinear elements. The advantage is that the inner loop of the process doesn't have any conditional statements, slowing the computation. The disadvantage is that outside a limited range of exponents p (which controls the ratio of release time to attack time), there are numerical difficulties.

Let's define the time constant of the leaky integrator, τ, as the time taken for the leaky integrator to reach $1 - 1/e$ of its target value. More precisely, if the input to the leaky integrator was, for a long time, at a level μ so that the state of the leaky integrator $\lambda(n)$ was also μ, and at a given point in time, the level of the leaky integrator input became ν, then the time constant of the leaky integrator is the time taken for the leaky integrator output $\lambda(n)$, starting at μ to reach

$$\lambda(n) = \mu + (1 - 1/e)(\nu - \mu). \tag{2}$$

For a given exponent p and leaky integrator time constant τ, derive expressions for the attack and release time constants of the RMp detector.