# Dynamic compression

**Seminar Handout** – 2024-Jan-25

This document was provided for the Hearing Systems Seminar belonging to the Audio Signal Processing & Audio Systems Lectures, which were held for the first time in winter semester 2020 as part of the Audio Systems Technology lecture series at TU Ilmenau.

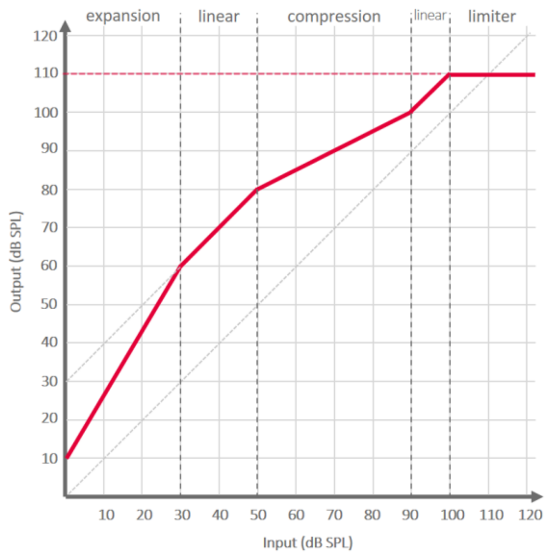
Please contact Dr.-Ing. Tamas Harczos (tamas.harczos@audifon.com) or Dr. rer. nat. Iko Pieper (iko.pieper@audifon.com) if you have questions.

## Introduction

A compressor can be imagined as an automaton adjusting the gain (or volume) knob of an amplifier. It's decision on increasing (turning the knob right) or decreasing (turning the knob left) the gain is always based on what it has heard recently at the input of the amplifier. "Recently" typically means the past few to few hundred milliseconds. Furthermore, the knob should be imagined as having its own weight and inertia, so that it cannot be turned infinitely fast. In fact, it can usually be turned faster to the left (towards minimum) than to the right (towards maximum).

Source: freepik.com

As discussed during the lecture, the characteristic of a dynamic compressor can be defined in different ways. Today, we will stick to the kind of characterization presented on the right. The only difference will be that we don't use the physical unit dB SPL, but will take dB FS (RMS) as a unit.



You will find **multiple ZIP files** on the seminar homepage (moodle2.tu-ilmenau.de/course/view.php?id=4426 or hearingsystems.github.io). Please download these and have them ready by the time the seminar starts. Some of the ZIP files are password protected; the password will be shared during the seminar.

Please also make sure to have MATLAB 2018b (or newer) or GNU Octave 6.1 (or newer) installed on your system. Furthermore, it would be useful to have an audio editor software installed. This can be Audacity (www.audacityteam.org), Wavosaur (www.wavosaur.com) or any other piece software, which you are familiar with.

## Notices about GNU Octave

- If you use GNU Octave for your work, don't forget to load the signal processing package first by issuing the command `pkg load signal`.
- The 1D-interpolation function `interp1()`, used in the `hss_compressor()` function seems to be very slow in the Octave implementation. Be patient during calculations.

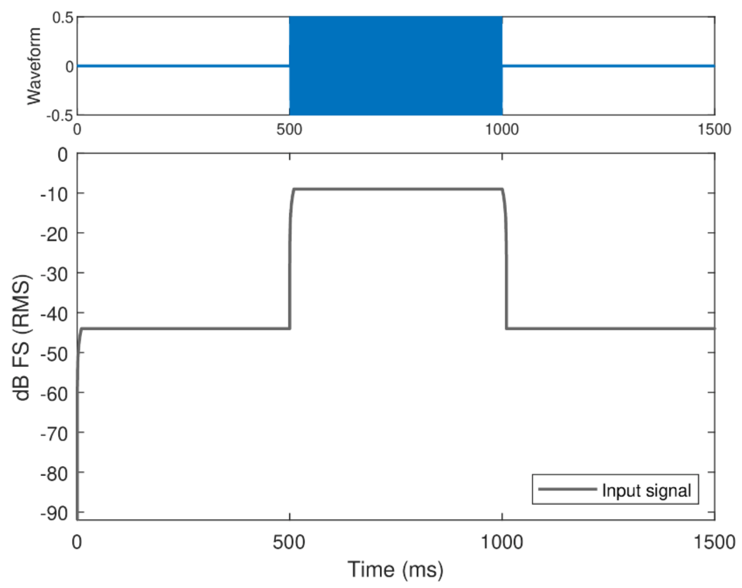
## Workflow

### Task 1

Please first extract **`HSS1.zip`** into a (preferable empty) directory of your choice. Open up MATLAB (or GNU Octave – for the sake of simplicity I will use MATLAB for either software), navigate to the directory where you've extracted the contents of the ZIP file.

- Inspect the files **`hss_main.m`** and **`hss_rms.m`** in the MATLAB editor.
- Inspect the audio files in the **`sounds`** subdirectory with your audio editor software and listen to them.
- Edit **`hss_rms.m`** in a way that it conforms its description in the function header. Suggested steps:
  - Recall definition of root-mean-square (RMS).

    $$RMS = \sqrt{\frac{1}{N}\sum_{n=1}^{N} x_n^2}$$
  - Calculate the size of the sliding window in samples, based on its specified length in ms and on the sampling rate.
  - Loop through the input samples and calculate the RMS value for the current and its preceding samples filling exactly one window.
  - The most troublesome part may be the beginning of each audio vector, since there you will not have enough samples to fill a window for the RMS calculation. Use zeros here instead of the missing samples.
  - Pre-allocate variables whenever possible (e.g. create vector filled with zeros before you write single values to the vector).
  - Test regularly by calling **`hss_main`** until you get a result like below for the test file **`Ansi.wav`** for an RMS window size of 10 ms.

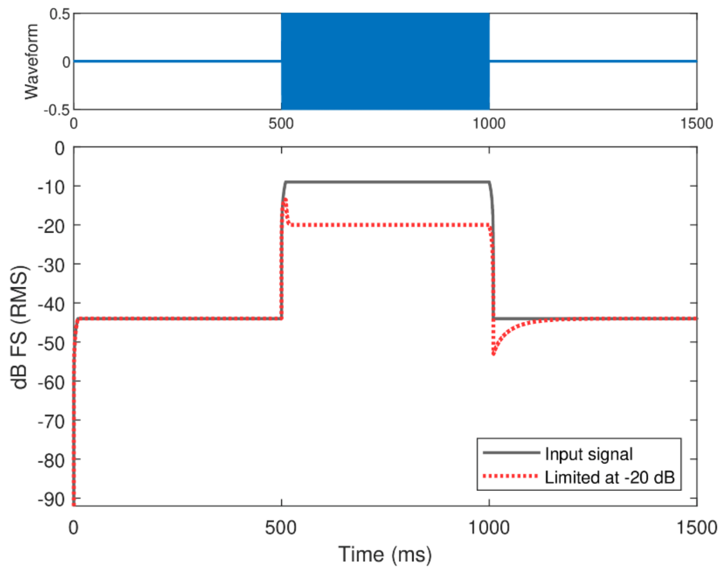


  - Investigate how different RMS window sizes affect the plot.

## Task 2

Extract **`HSS2.zip`** into a (preferable empty) directory of your choice. In MATLAB navigate to the directory where you've extracted the contents of the ZIP file.

- Inspect the provided **`hss_rms.m`** file and compare it with your implementation.
- Inspect the newly provided **`hss_limiter.m`** file and try to comprehend its inner workings. You may also try it by calling **`hss_main`** with various test audio files.
- You may want to look for this line in **`hss_main.m`**:
  ```
  % playback_signal= audio_limited;
  ```
  and uncomment it to hear how the processing affects the sound.
- For **`Ansi.wav`**, with 20 ms attack time, 100 ms release time and -20 dB FS (RMS) threshold level you should see a plot like this:

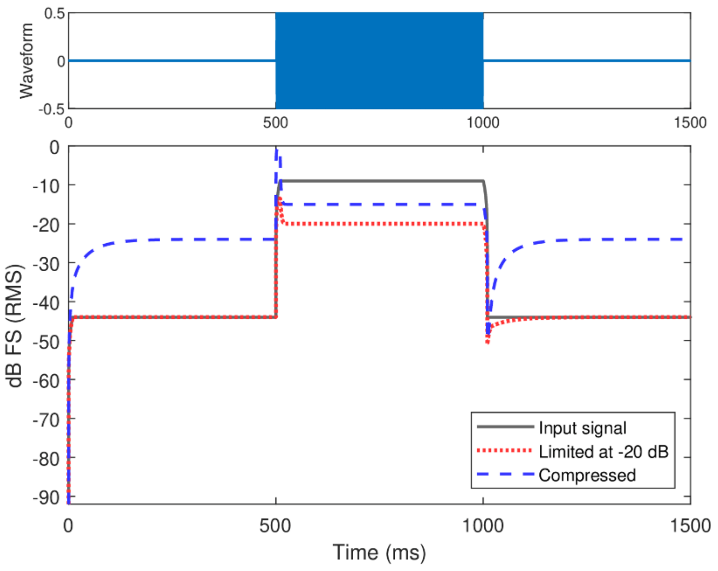


- Notice the **over-** and **undershoot** at time instances where the limiter changes the gain. Those are typical artifacts with algorithms changing dynamics of the sound.
- Test how various attack and release times as well as threshold levels affect the output!

## Task 3

Extract **`HSS3.zip`** into a directory of your choice. In MATLAB navigate to the directory where you've extracted the contents of the ZIP file.

- Inspect the newly provided **`hss_compressor.m`** file and try to comprehend its inner workings. You may also try it by calling **`hss_main`** with various test audio files.
- You may want to look for this line in **`hss_main.m`**:
  `% playback_signal= audio_compressed;`
  and uncomment it to hear how the processing affects the sound.
- For **`Ansi.wav`**, with all the default parameters you should see a plot like this:



- Notice the over- and undershoot at time instances where the compressor changes the gain.
- Test how various attack and release times and knee point definitions affect the output!
- Let the MATLAB script plot the dynamic characteristic by uncommenting the following lines in the code:

```
% figure;
% plot(compressor_knees_db(2:end,1),compressor_knees_db(2:end,2),'o-');
% line([compressor_knees_db(2,1),0],[compressor_knees_db(2,1),0],'Lin-
eStyle',':','Color',[1 0 0]);
% xlabel('Input (dB FS RMS)'); ylabel('Output (dB FS RMS)');
% return;
```

## Task 4

Extract **`HSS4.zip`** into a (preferably empty) directory of your choice. In MATLAB navigate to the directory where you've extracted the contents of the ZIP file.

- You may now explore the newly provided **`hss_main_multiband.m`** file and try multi-band dynamic compression.
- Notice that you now have a **`sounds\processed`** subfolder with the reference outputs of the algorithms.
- Observe how much "louder" the output sound of the multi-band processing is.
- We will elaborate on multi-band compression together.