<< back

Paul Vögler, 2012-04-20

Mixing two digital audio streams with on the fly Loudness Normalization by Logarithmic Dynamic Range Compression

1 Index

2 Preface

When I first started to experiment with different CAPTCHA methods, I came across the problem to provide a method for visually impaired people. A common practice is to concatenate the spelling of the codeword. To make things more difficult, I first added some white gaussian noise to the audio file. Thinking some more about it, I realized, that that might not be enough.

I wanted to add some background noise by exploiting the Cocktail Party Effect. It was then that I first came across the problem of mixing two digital audio files. Knowing only a little about it, I tried to educate myself and had to realize, that the problem is not as simple as I first thought.

As using the built in audio mixing capabilities of today's operating system is not an option for a web service, I tried the most common proposed ways I will explain later. But I was not satisfied with the result. I then found a note by Viktor T. Toth, who engaged the problem over a decade ago. I adopted his formula to my needs and had good results - or so I

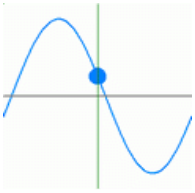
though at least.

As I familiarized myself further with the matter of mixing digital audio, I tried to fully understand his formula. In the end I came to the conclusion that his method also cannot be the solution. I will explain why at the end of this document.

I then developed the method described in the title. This document summarizes some of my research and explains the process of developing the Logarithmic Dynamic Range Compressor.

3 Basics

3.1 Sound Waves



Sound waves are mechanical waves that propagate through the medium (air) by displacement. When observed from a fixed point, sound waves produce continuous and varying levels of positive and negative pressure above the mean pressure. The rate over time at which this happens is called the Frequency, the pressure difference, the Amplitude. Therefore sound waves are a bit different from the "normal" waves familiar in water, but behave much in the same way.

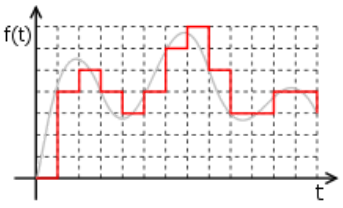
The higher the frequency, the higher the pitch of the sound wave is perceived. And the higher the amplitude, the louder that sound wave is perceived.

3.2 Analogue Audio Signals

An analogue audio signal is the continuous electrical representation of sound waves. Let's consider an ordinary microphone. When sound reaches the membrane, it pushes and pulls on this membrane because of the arriving pressure differences.

These differences are converted into different voltage levels, let's say from -1V for the maximum detectable negative pressure, though 0V for mean pressure, to +1V for the maximum detectable positive pressure. All voltages in between are possible and reflect the amount of pressure difference. A continouos voltage of 0V means there is no pressure difference over time, hence it's silent.

3.3 Digital Audio Signals



Digital audio is the representation of that signal in binary format. To achieve this, samples of the electrical signal are taken at a fixed rate, called the Sampling Rate. The read level is then converted to a binary number. This step is called Quantization, as not every possible reading can be encoded with full accuracy. There is only a limited number of levels available. That number is determined by the bit depth with which the sampling occurs, called the Resolution. The error made hereby is called Quantization Error, resulting in a worse Signal to Noise Ratio (SNR).

For 8 bit samples that means there are just 256 (2 to the power of 8) different levels that can be represented. About half of that for positive levels plus one level for 0, and the other half for negative levels. The hardware device which does this is called the Analogue-Digital-Converter (ADC). The method is called Pulse Code Modulation (PCM), where each sample is represented by the same amount of bits (e.g. 8).

3.4 Digital Audio Formats

The digital audio format describes the way the digital audio signal is stored and transmitted. There are many digital audio formats around, some of which use additional compression techniques to reduce the amount of data, like MP3.

Another common format is the Microsoft WAV-Format. In most cases it is used to store uncompressed PCM values. The most common PCM resolutions are 8, 16 and 32 bit, with 8 and 16 bit being encoded as integer values (whole numbers) and 32 bit as floating point numbers (fractions).

Integer values above 8 bits per sample are stored as signed integer values, i.e. they have a positive and negative range with the mid point at 0. 8 bits and below are special in that they are stored as unsigned values. That is, levels from 0 to half the range represent negative amplitudes (0 being the highest negative amplitude) and above for positive amplitudes. For 8 bit that means 0-127 are negative amplitudes, 128 is the mid point and 129 to 255 are positive amplitudes. They can easyly be transformed into signed values by substracting half the range (-128 for 8 bit).

PCM floating point numbers are in the range of -1 to 1 with 0 as the mid point and fractions in between.

One format can be transformed into another format (and back) without loss, as long as the new format's range is at least as big as the source's. That is why mixing and mastering is often done in 32 bits exclusively for higher accuracy,

and only the result is then published in 16 bits (e.g. Audio CD).

For all practical purposes to come, every integer PCM value can be transformed into a floating point PCM value in the range of -1 to 1 and back without loss of information or introduction of additional noise.

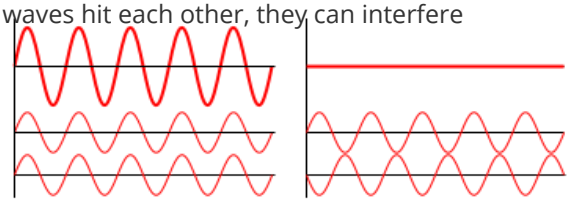
3.5 Playing back Digital Audio Signals

The reverse process is done by the Digital-Analogue-Converter (DAC). It tries to restore the original electrical voltage levels continously and as accurate and smooth as possible. The amplifier then does it's job, and the speakers attached to it's output create positive and negative pressure waves, thus replaying the sound.

4 Mixing Sound

4.1 Mixing two Sound Waves

Mixing two sound waves is basically observing their interference. When waves hit each other, they can interfere constructively or destructively. That is, two wave crests form a bigger wave crest, a crest and a valley cancel each other out and two valleys form a bigger valley. If wave crests are defined as being above 0 and valleys being below 0, the result is simply adding both values.



More complex sounds, like music or speech, are merely the result of mixing sound waves of different frequencies, amplitudes and phases (left or right shift).

That is why a choire is the louder, the more people are singing and the better they are synchronized. Active noise cancellation devices try doing the opposite by producing a "counter wave", that cancels out as much of the unwanted sound as possible.

The reason why we don't perceive the two waves as exactly twice as loud as each individual wave, is for the complicated way we perceive loudness. In reality, our perception of loudness not only varies non-linearly with the amplitude, but also with the frequency and duration of a sound. There are models that try to formalize this, but this is beyond this article.

The important thing to note here is, that in general we are more perceptible to changes of small amplitudes than we are of large amplitudes.

4.2 Mixing two Digital Sound Sources

As seen above, mixing two digital sound sources of the same sample rate and resolution that encode positive pressure as positive values and negative pressure as negative values, is as simple as adding the pair of sample values.

$$C = A + B$$

Where A and B are the samples of either source at the same time and C is the mixed sample at that time.

The problem now is, that when adding the two values, the result may over- or underflow the range available. The sum of A and B can be double the maximum (or minimum) value within the resolution's range in a worst-case-scenario.

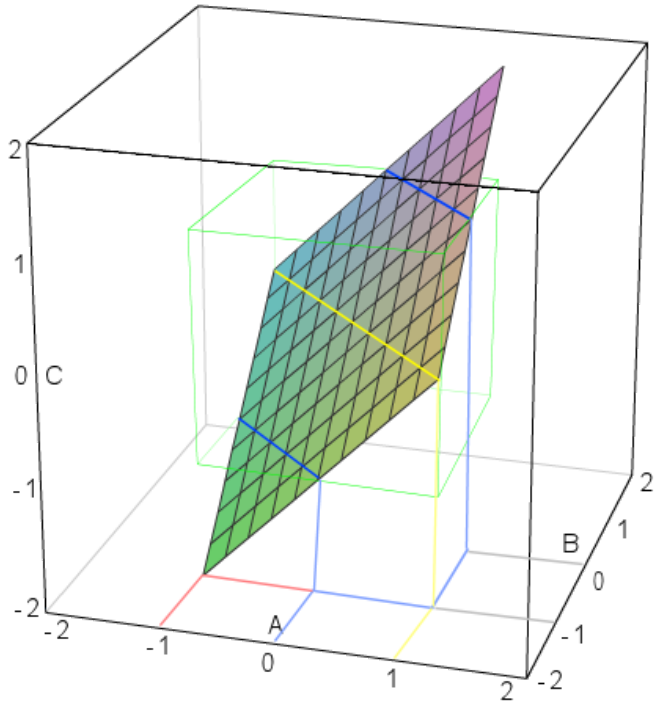
The graph shows the result of A + B.

Most possible combinations are between the dark blue lines, which mark the boundary for C being below -1 or above 1.

The yellow line marks the 0-level, where all combinations of A + B = 0 are.

The light green box marks the valid rage for A, B and C and is the area shown in all other 3D graphs below.

All graphs assume working with floating point PCM values in the range of -1 to 1.



5 Normalization

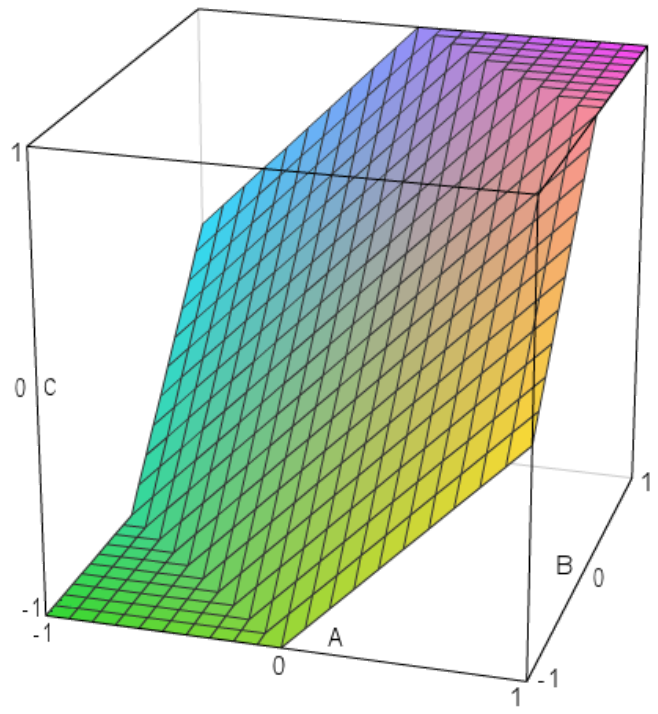
To account for the problem above, some sort of method has to be applied to get the values of the resulting signal back into the valid range. This process is called normalization.

5.1 Clipping

The simplest way to do this is by clipping. Values out of range are simply cut off and encoded using the maximum or minimum value possible. As a result "clicking" and "popping" may occur, as the original sound wave cannot be reconstructed properly.

$$C = min(C_{max}, max(C_{min}, A + B))$$

Where C_{max} is the maximum, and C_{min} the minimum value within the range of the resolution.



The graph shows the result of A + B with all sums below -1 or above 1 clipped.

Compared to the first graph, the flaps below and above the blue lines are sharply folded, so that they don't exceed the valid range.

Although the middle part looks as it should be, it is not hard to imagine why the result is far from optimal, as many combinations get the same value for C.

This would only then not be a problem, if there are no combinations of A + B that are below -1 or above 1.

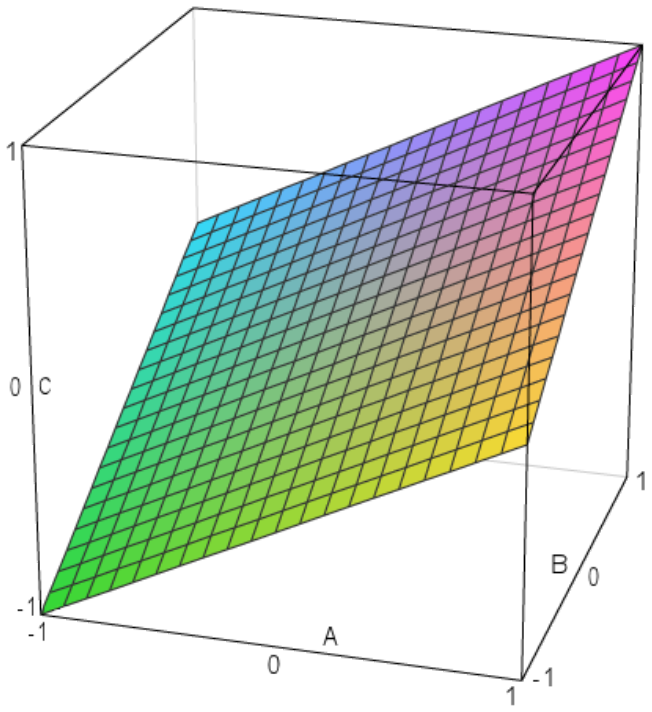
5.2 Linear Attenuation

When mixing two audio sources without prior knowledge of their maximum range used, and without a second pass, one way to be sure to stay within the resolution's limits is, if the sources volumes are adjusted considering a worst-case-scenario. An easy way to achieve this is by simply dividing the result by 2.

$$C = \frac{A + B}{2}$$

This way the mixed sample is assured to stay within range. The problem with this method is, that in addition to reducing the resolution of each source to half of what it used to be (if the output range is the same as the input range), and thus increasing the quantization error, each individual source is attenuated (it's loudness reduced) as well.

That is why, when using this method, the perception of the result is to not have enough volume. The most obvious case is when one source is just silence. The result would be the other source with all amplitudes halfed, a reduction in volume by about 6 dB.



The graph shows the result of A + B with the result normalized by dividing by 2.

Compared to the first graph, the plane is squeezed together from both ends beyond the blue line.

The yellow line is acting as an anchor, fixing the width and position of the plane, but allowing it to tilt so that both corners are in the extreme position.

The effect of this squeezing is, that every amplitude is attenuated.

5.3 Pre or Post Mixing Normalization

If both sources and their amplitudes at any given time are known in advance, or knowing the maximum absolute amplitude after mixing, the volume of the sources can be adjusted to ensure they stay within range after mixing. Implementation constraints aside, adjusting before, after or at mixing is equivalent.

$$C = \frac{A + B}{\frac{max(\{|A_i + B_i|\})}{\frac{C_{range}}{2}}} \quad \forall i \in \mathbb{N}, i \leq T$$

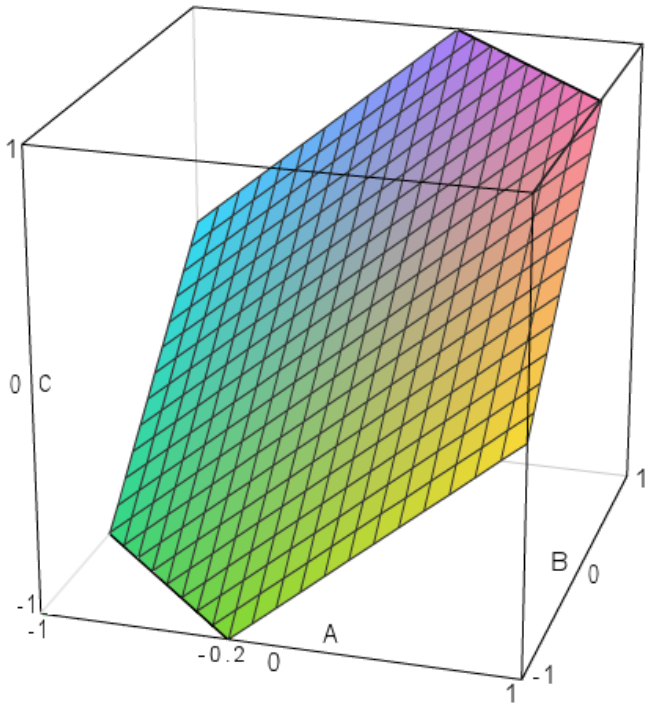
Where T is the length of the signals in number of samples, i any sample number, A_i and B_i a pair of samples at the same time and C_{range} the distance between the minimum and the maximum values (float) or the number of levels in the resolution (integer).

This can be interpreted as follows. To get a mixed sample C, divide every A + B by the following divisor:
Find the maximum absolute amplitude (positive or negative) that can result from adding a pair of samples A_i and B_i within the entire length of the signals to be mixed and set it in relation to the maximum absolute value possible to encode (C_{range} / 2).

i.e. If the range was from -1 to 1, C_{range} would be 2 and C_{range} / 2 = 1. If the maximum absolute amplitude of any A + B was 1.2 (either +1.2 or -1.2), each sample pair would have to be divided by 1.2 (=1.2 / 1).

This should probably only be done if the maximum absolute amplitude found is above C_{range} / 2. Otherwise the mixed

signal is boosted. This may or may not be a desired effect.



The Graph shows the result of A + B with the result normalized by dividing by 1.2.

There are no flaps in this graph, as the maximum absolute amplitude for any sum of A_i and B_i found was 1.2.

Compared to the first graph, the plane is slightly squeezed (and tilted) until all amplitudes fit within the green box.

One example would be A = -0.2 and B = -1. The sum of both is -1.2 and after normalizing with 1.2 the result for C is -1.

The effect is, that the mixed sound uses the maximum possible resolution (the whole range).

6 Dynamic Range Compression

Another way to make sure to stay within range but at the same time preserve the fidelity of low- to mid-level amplitudes, is to attenuate the level after mixing only if it exceeds a certain threshold. This way, depending on the setting of the threshold, most of the possible combinations of adding A and B retain their original mixed value. Only the (very) loud values above that threshold get compressed into a smaller space. Because we are not as perceptibe to amplitude changes in the upper range, the effect is less noticeable.

6.1 Linear Dynamic Range Compression

Amplitudes above the threshold are linearly compressed into the space left by a compression factor. This factor is determined by the threshold to ensure every combination of A + B fits within the resolution's range.

The following formulas assume working with floating point values in the range of -1 to 1.

$$x \in \mathbb{R} \land -2 \leq x \leq 2;\ t \in \mathbb{R} \land 0 \leq t < 1$$

$$f_i(x;t) = \begin{cases} x & -t \leq x \leq t; \\ \frac{x}{|x|} \cdot \left(t + \frac{1-t}{2-t} \cdot (|x| - t)\right) & |x| > t \end{cases}$$

Where x is the mixed sample of A and B to be normalized, and t the threshold.

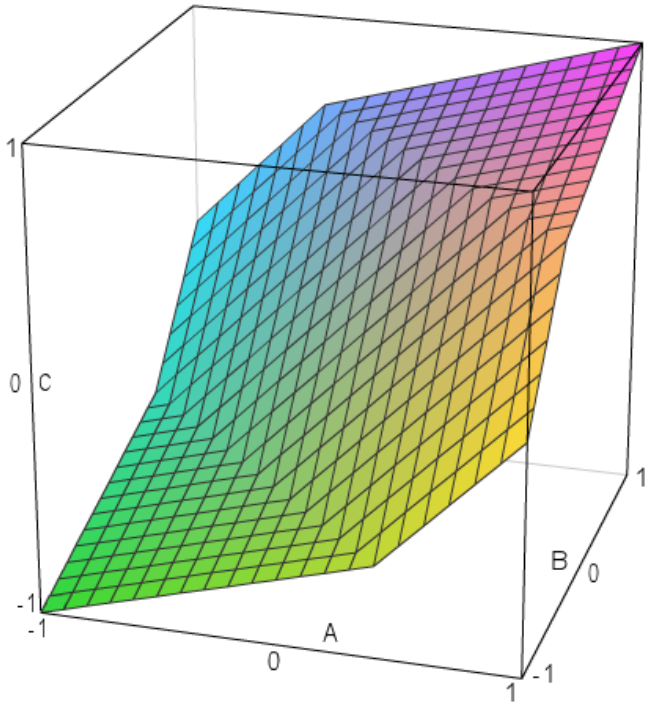
The graph shows C = f_i(A + B; 0.6).

C is the sum of A and B linearly compressed by the function f_i with a threshold of 0.6.

All (absolute) amplitudes below 0.6 retain their original value. From 0.6 to 2, amplitudes are linearly attenuated to fit within the range.

The effect is that all possible amplitudes fit within the range. Low and medium sound levels have good volume.

Because high amplitudes are suddenly attenuated, the result may sound like there was an unexpected drop in volume when switching from low or medium sound to loud.



6.2 Logarithmic Dynamic Range Compression

Amplitudes above the threshold are logarithmically compressed into the space left by a dynamic compression factor. The aim is to smoothen out the effect at the threshold as well as exploiting our non linear loudness perception. Amplitudes further away from the threshold are comressed with an ever increasing factor.

$$x \in \mathbb{R} \wedge -2 \leq x \leq 2;\ t \in \mathbb{R} \wedge 0 \leq t < 1$$

$$f_l(x;t) = \begin{cases} x & -t \leq x \leq t; \\ \frac{x}{|x|} \cdot \left(t + (1-t) \cdot \frac{\ln\left(1+f_\alpha(t) \cdot \frac{|x|-t}{2-t}\right)}{\ln\left(1+f_\alpha(t)\right)} \right) & |x| > t \end{cases}$$

Where x is the mixed sample of A and B to be normalized, and t the threshold. See below for a definition of the function f_α.

The graph shows C = f_l(A + B; 0.6).

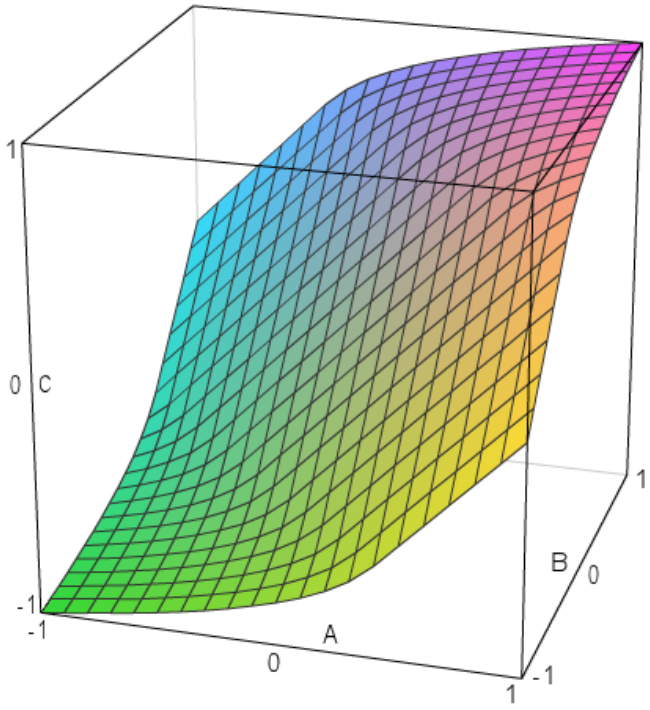
C is the sum of A and B logarithmically compressed by the function f_l with a threshold of 0.6.

All (absolute) amplitudes below 0.6 retain their original value. From 0.6 to 2, amplitudes are logarithmically attenuated to fit within the range.

The effect is that all possible amplitudes fit within the range. Low to high sound levels have good volume.

Only very high amplitudes have less resolution and are compressed into a smaller space compared to linear compression.

Because high amplitudes are only gradually attenuated with a smooth crossing, the attenuation is barely noticeable.



6.2.1 Derivation of f_l and f_α

6.2.1.1 Preliminary considerations

The goal is to find a suitable method to normalize a mixture of two audio streams on the fly with as little noticeable effect as possible. The two most common recommended methods when browsing the web on that subject are clipping and linear attenuation by dividing by 2. The former is prone to produce distortion in the form of clicking and popping, the latter noticeably attenuates the volume.

There are methods like Dynamic Range Compression (DRC) that are used in broadcasting and production environments. These systems often adjust volume levels by monitoring the output with a feedback or feed-forward loop.

The goal here is to find a function that projects the input range (-2 to 2) to the output range (-1 to 1) on the fly and independent of the stream behind or ahead while avoiding disturbing noises and still having a decent output volume in all possible situations.

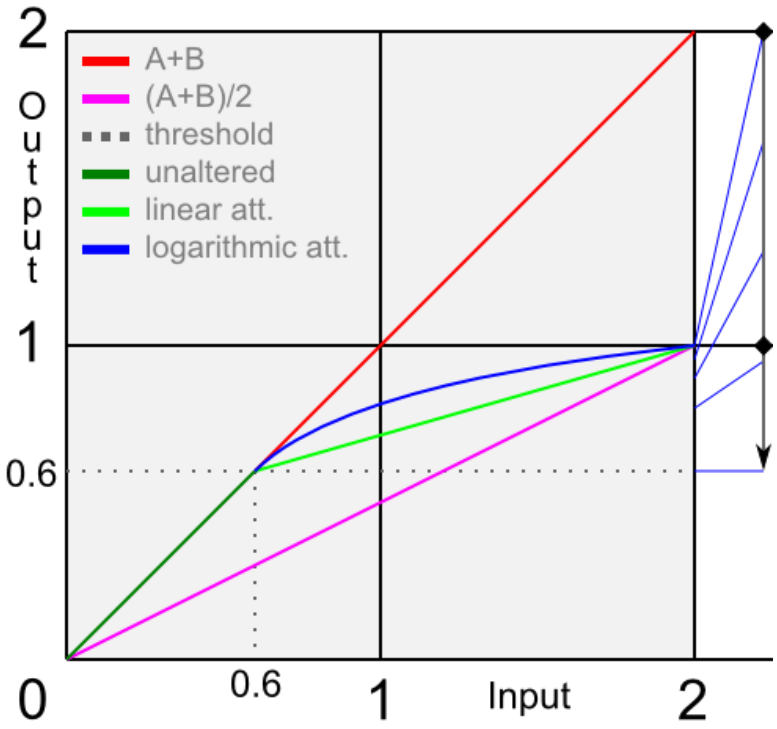
The graph to the right shows different projections from input (A + B) to output values. Because the negative and positive range are exactly the same only with opposite sign, it is sufficient to only look at the positive range for now.

If no means for normalization are applied, the result is the red (and dark green) line, which overflows the desired target range from 0 to 1.

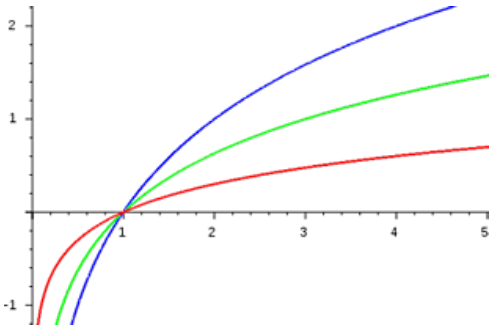
The magenta line shows the result if all output levels are normalized by dividing by 2.

The light green line shows the result of linearly compressing, i.e. with a fixed compression factor, the values above the threshold at 0.6 into the target range.

The effect of the blue cuve is compressing ever higher values with an ever increasing compression factor, thus leaving the relatively lower amplitudes more space (resolution) to be compressed in.



6.2.1.2 Developing the function



A logarithmic function seems to be a logical choice here, because it has all the properties we are looking for. It is strictly monotonically increasing at an ever sligther rate, thus slighly but steadily increasing the compression factor.

Therefore, to find the function that describes the blue line above, we begin with the basic logarithmic function $f(x) = log_b(x)$.

The graph to the right shows the log function for base 2 (blue), 3 (green) and 10 (red).

We are only interested in the positive part of the log function. Because all log functions cross from positive to negative values at 1, we add +1 inside the log function: $f(x) = log_b(1 + x)$.

We now have to adjust our input value x to be 0 for this function when we are at the threshold: $f(x) = log_b(1 + (x - t))$. This in effect shifts the log function on the horizontal axis.

We then shift the function vertically to take the value of the threshold when x is at the threshold: $f(x) = t + log_b(1 + (x - t))$.

We now have to bend the function such that it takes the value 1 when x is 2. We do this by first normalizing our input value to be in the range of 0 to 1 independent of the threshold: $f(x) = t + log_b(1 + \frac{x-t}{2-t})$.

For a given base b = 2 the log function would now return the value 0 for x = t and 1 for x = 2 because $\log_2(1) = 0$ and $\log_2(2) = 1$. We now have to squeeze the log function vertically to return values in the range of 0 to (1 - t), because we already added t to our function and thus that's the range we want the log function to work in: $f(x) = t + (1 - t) \cdot log_2(1 + \frac{x-t}{2-t})$.

Because we want to have control over the base of the log function, we introduce a new parameter α and use the fact that $\log_b(x) = \ln(x) / \ln(b)$:

$$\alpha \in \mathbb{R} \wedge 0 < \alpha;\ t \in \mathbb{R} \wedge 0 \leq t < 1;\ x \in \mathbb{R} \wedge t \leq x \leq 2$$

$$f(x) = t + (1 - t) \cdot \frac{\ln(1 + \alpha \cdot \frac{x-t}{2-t})}{\ln(1 + \alpha)}$$

Where x is the positive input value above the threshold t and α determines the base for the logarithmic function.

6.2.2 Determinig alpha

The parameter α controls the rate at which the log function grows, or in other words the amount of curveature.

The graph to the right shows three different values for α with a threshold t = 0.5. For ogange is α = 1, for blue α = 4 and for purple α = 20.

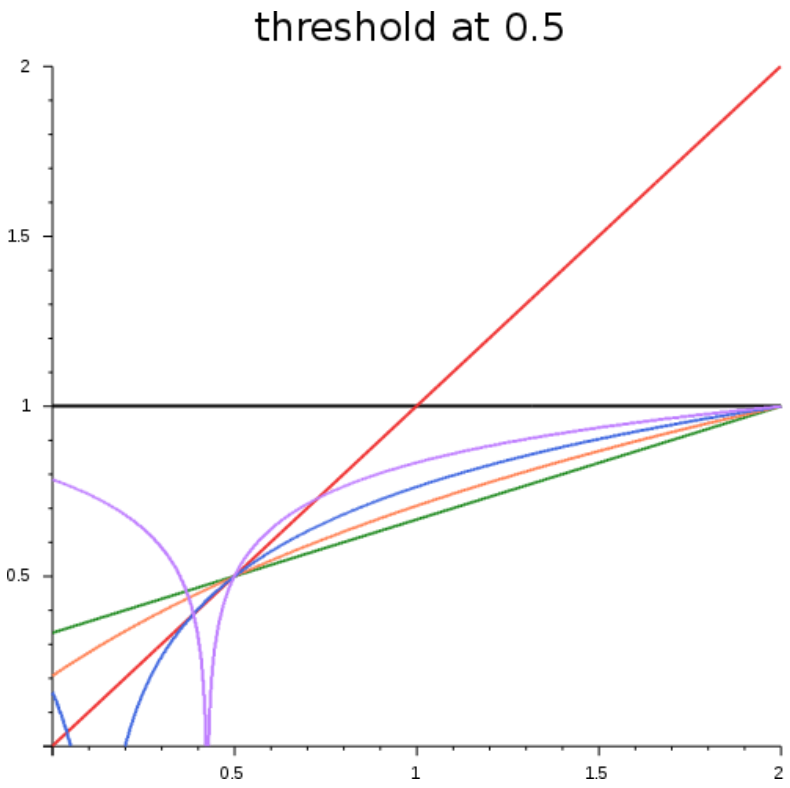
A suitable choice for α would be the one that has the smoothest transition from the red line into the curve at the threshold.

It is easy to see that α must be dependant on t, the threshold.

The smoothest transition possible is, when the slope of the curve matches the slope of the red line at the threshold.

The slope of the red line is always equal to 1.

For the slope of the function f(x) from above to be 1 at the threshold, the first derivative of the function has to be 1 at the threshold, where x = t.



6.2.2.1 Solving for alpha

When finding the first derivative f'(x) for f(x) and solving for α at the point $x_0 = t$ we get:

$$f(x) = t + (1-t) \cdot \frac{\ln(1+\alpha \cdot \frac{x-t}{2-t})}{\ln(1+\alpha)}$$

$$f'(x_0) = \frac{\alpha \cdot (1-t)}{\ln(1+\alpha) \cdot (2-t+\alpha x_0 - \alpha t)}$$

$$f'(t) \stackrel{!}{=} 1 \Rightarrow$$

$$1 = \frac{\alpha \cdot (1-t)}{\ln(1+\alpha) \cdot (2-t)}$$

$$\sqrt[\alpha]{1+\alpha} = e^{\frac{1-t}{2-t}}$$

Unfortunately we cannot solve for α directly here, so we define the function:

$$f_\alpha(t) := \left\{ \alpha : \sqrt[\alpha]{1+\alpha} = e^{\frac{1-t}{2-t}} \mid t \in \mathbb{R} \wedge 0 \leq t < 1 ; \alpha \in \mathbb{R} \wedge \alpha > 0 \right\}$$

$$f_\alpha(0) \approx 2.51286;\ f_\alpha(0.5) \approx 5.71144;\ \lim_{t \to 1} f_\alpha(t) = \infty$$

6.2.2.2 Calculating alpha

The equasion $\sqrt[\alpha]{1+\alpha} = e^{\frac{1-t}{2-t}}$ can be solved with an iterative method:

1. Pick a threshold t.
2. Calculate the right side of the equasion.
3. Take an initial guess for a lower bound and an upper bound for alpha. 2.5 is a safe lower bound.
4. Calculate the left side of the equasion for the lower bound and the upper bound of alpha.
5. If the lower bound solution is below the right side, start over at step 3 with a lower lower bound. The old lower bound is now a safe upper bound.

```
6.  If the upper bound solution is above the right side, start over at step 3 with a higher upper bound.
    The old upper bound is now a safe lower bound.
7.  If the difference between the upper bound and lower bound is smaller than the desired precision,
    approximate alpha linerarly and stop.
8.  Calculate the left side of the equasion for the mid point between lower and upper bound.
9.  If the mid point solution is above the right side,
    make the mid point the new lower bound and go to step 4.
10. Make the mid point the new upper bound and go to step 4.
```

Calculate a list of values for alpha with 20 steps for t and a precision of 2 decimal places: GO!

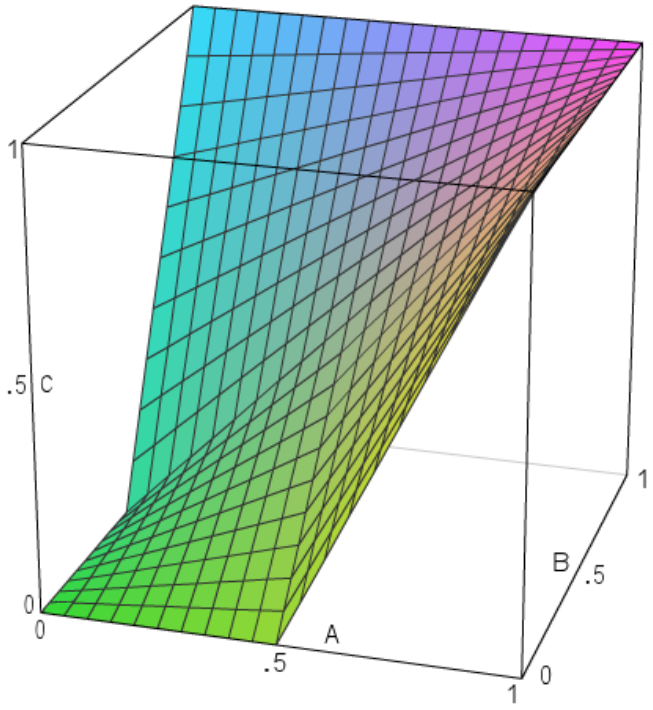
7 Viktor T. Toth's method

His formula was developed for unsigned amplitude values with an input / output range from 0 to 1 and a baseline of 0.5. The formula can be easily transformed to handle ranges from -1 to 1 equivalently.

He came up with the formula by this thought process:

> *"If we have two signals, A and B, and either A or B has a value at the midpoint (representing silence), we want the other to appear on the output in unchanged form. If either A or B has an extremal value (minimum or maximum) we want that extremal value to appear on the output. If both A and B are below the midpoint, the result should be further below the midpoint than either A and B; if both are above the midpoint then similarly, the result should be higher than either of them. Lastly, if A and B are on opposite sides of the midpoint, the result should represent the fact that the two signals are cancelling each other out to some extent."* [Mixing digital audio; Viktor T. Toth; 2000]

$$f(A;B) = \begin{cases} 2 \cdot A \cdot B & A < 0.5 \wedge B < 0.5 \\ 2 \cdot (A+B) - 2 \cdot A \cdot B - 1 & A \geq 0.5 \vee B \geq 0.5 \end{cases}$$



The thing that is most obvious in this graph is, that the graph is not symmetric. Ideally the result should be symmetric when you invert all values.

Because the function has a different formula in case both values are below the baseline versus the other two possibilities, there is a coarse transition in the surface of the combined function.

If at least one value for A or B is at a maximum or minimum, the other value has no effect on the output. In this image this is visible by the graph being clamped to the axes at the top and bottom.

Overall, the graph is curved towards high values, which also affects the signals "*cancelling each other out to some extent*".

The results can be observed in the wave form comparison below in the missing symmetry, the sharp and sudden changes in values and the missing wave crests and valleys compared to the natural mix.

In short, the mixed result with this function is far from the original.

8 Wave form comparison of different normalization methods

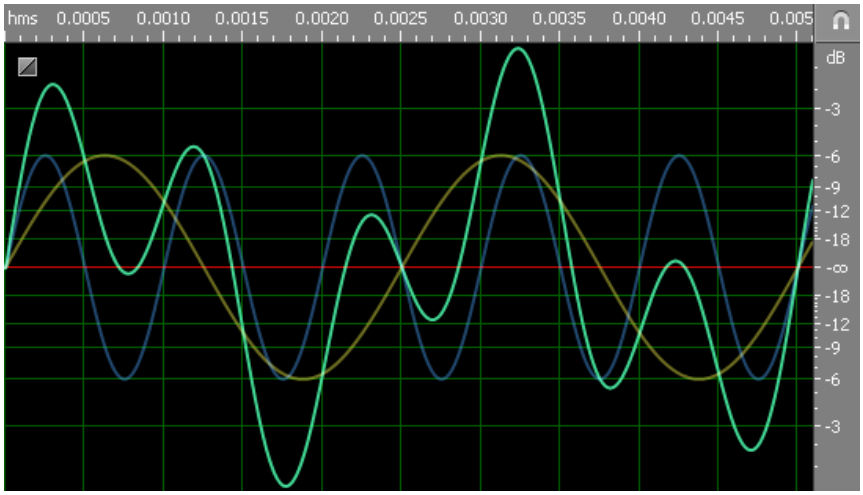
8.1 Source waves and mixed result

The blue line is a sine wave at the frequency of 1000Hz.

The yellow line is a sine wave at the frequency of 400Hz.

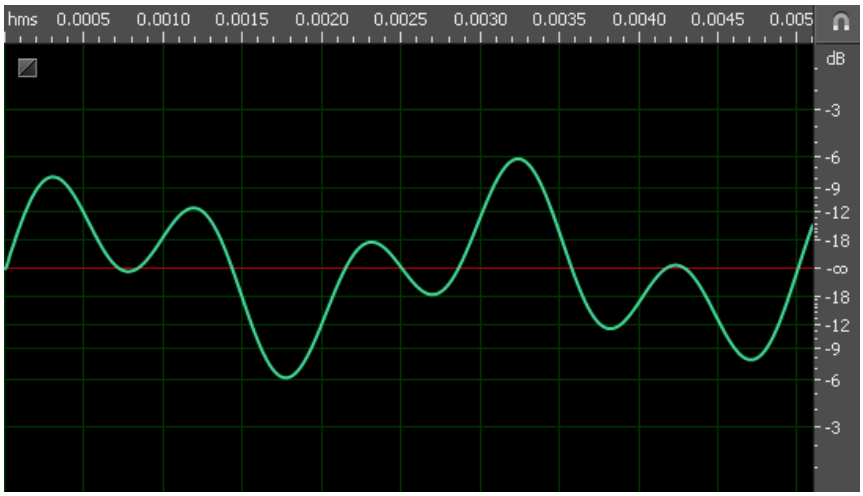
The green line is the mixed result of both waves.

The source waves use half the dynamic range, up to -6dB.



The result uses the full dynamic range and technically would not need to be normalized, albeit it may be perceived as overdriven.

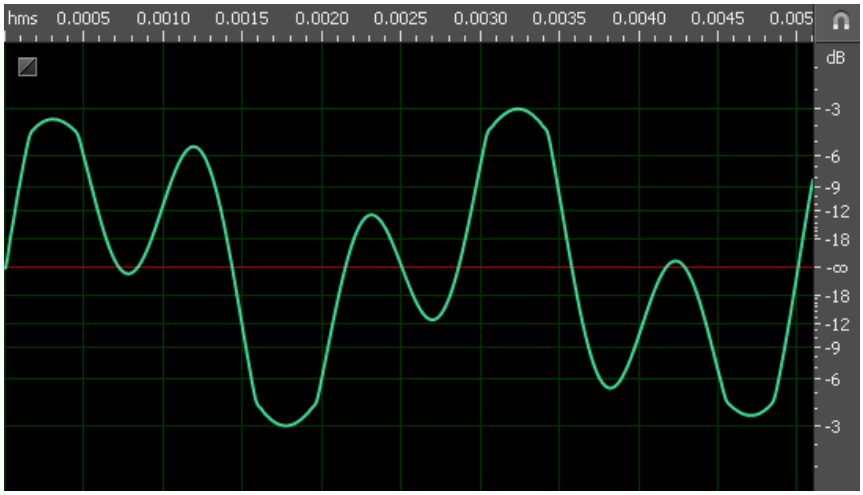
8.2 Normalization by dividing by 2



The compression factor is 2:1 for every sample.

Although the waveform is perfect in every way, the result is a loss of loudness for each individual sound by 6dB.

8.3 Normalization by linear compression

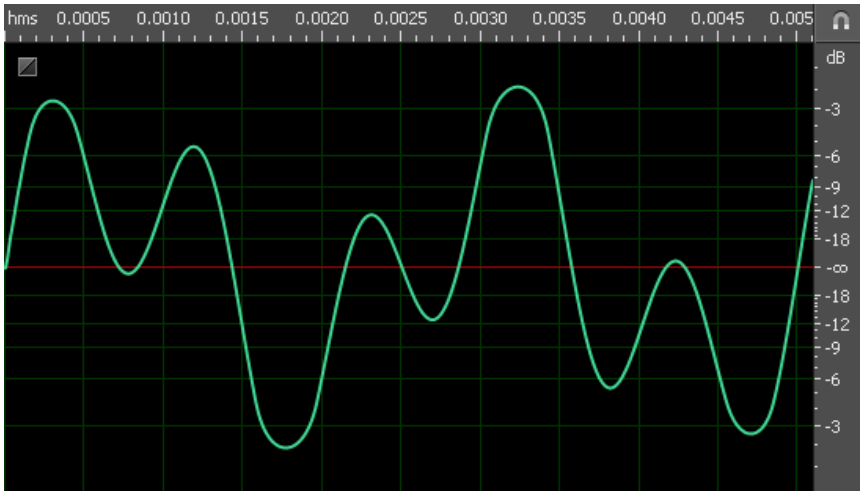


The threshold used is 0.6 (around -4.5dB).

When the threshold is reached, amplitudes are suddenly compressed with a compression factor of 3.5.

The result is an unsmooth transition, seen in the sharp edges near the top of the largest wave crests and valleys.

8.4 Normalization by logarithmic compression with a threshold

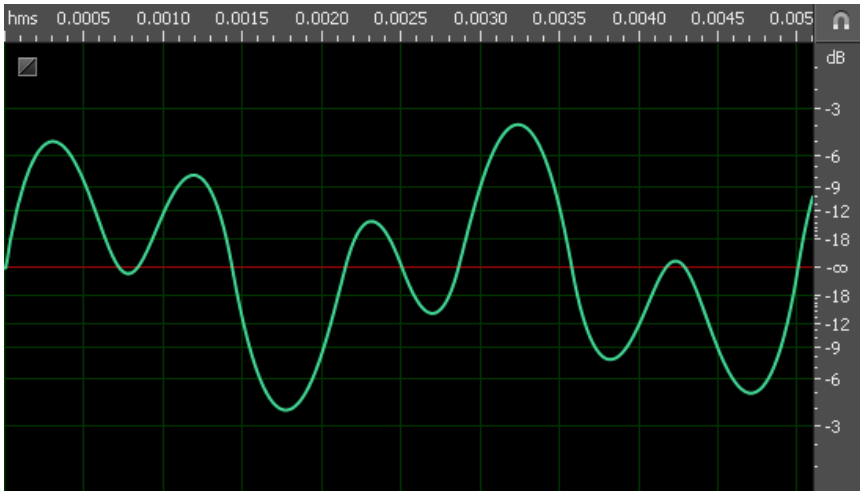


The threshold used is 0.6 (around -4.5dB).

Beyond the threshold, amplitudes are compressed with increasing compression factors from 1 to about 8.5 at maximum for increasing amplitudes.

The result is a smooth transition to compressed amplitudes at the threshold while leaving more resolution to lower amplitudes than higher amplitudes.

8.5 Normalization by full range logarithmic compression

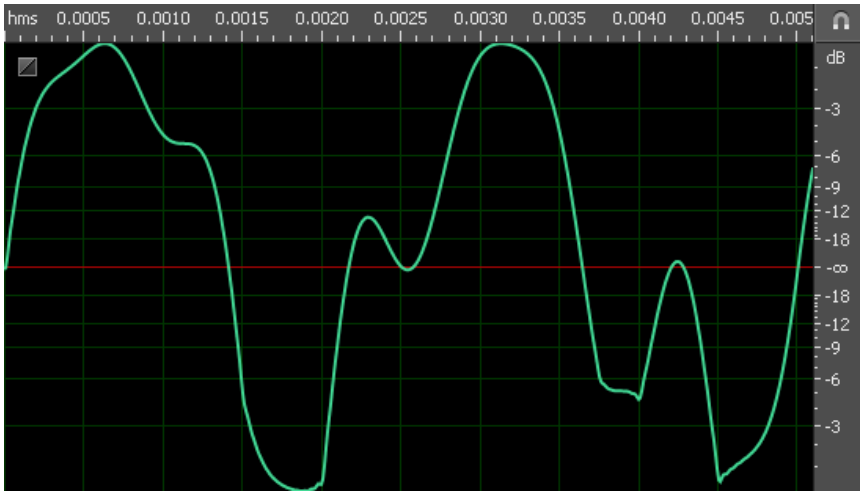


The threshold used is 0 (baseline).

The full dynamic range is compressed.

The initial compression factor is 1, meaning no compression. It slightly increases with increasing amplitudes to about 3.5 at the maximum.

8.6 Normalization and mixing with Viktor T. Toth's formula



Although the amplitudes stay within range, the wave form is distorted beyond recognition.

The wave form is not symmetric.

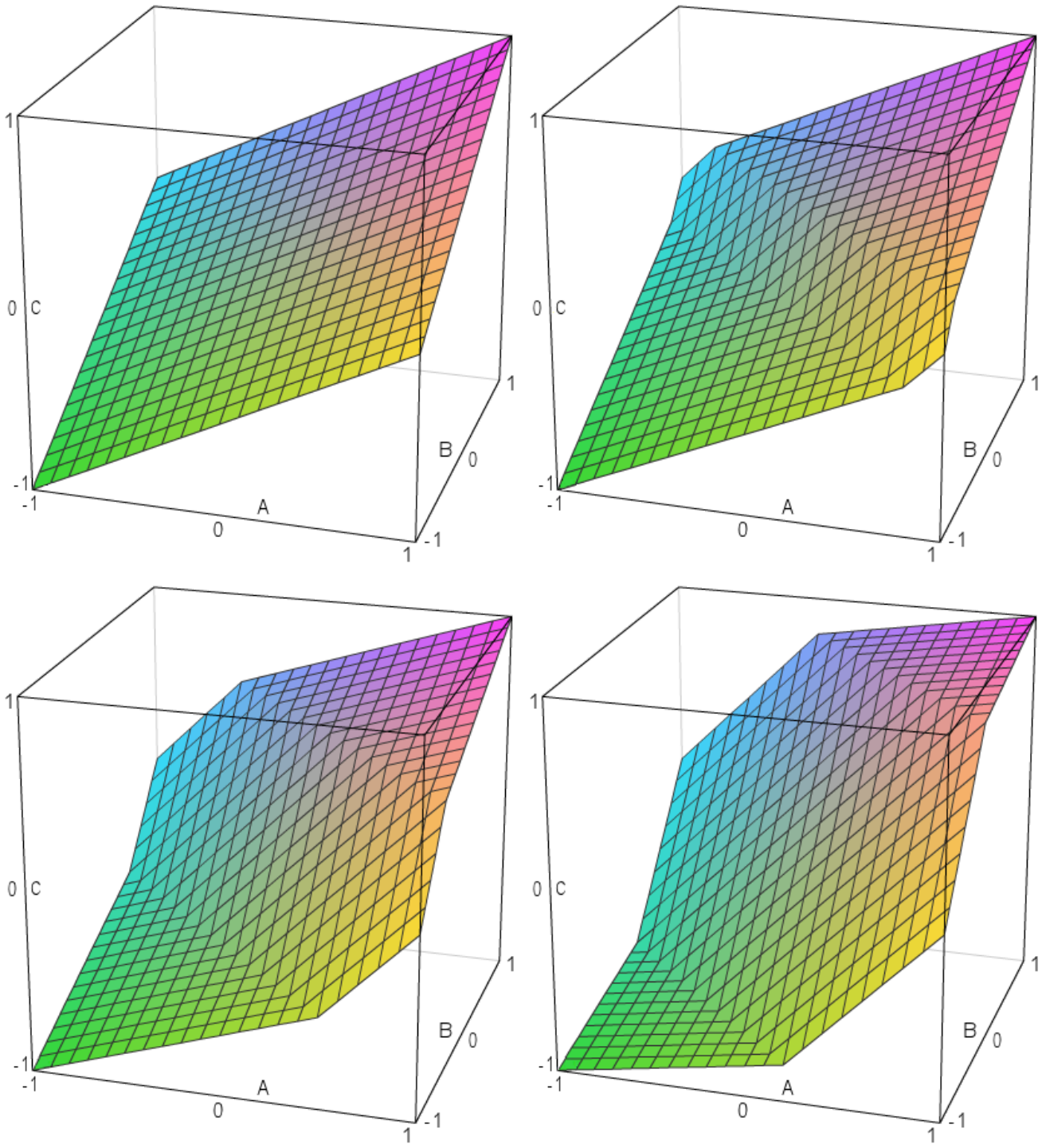
There are coarse and sudden changes is values.

High values dominate.

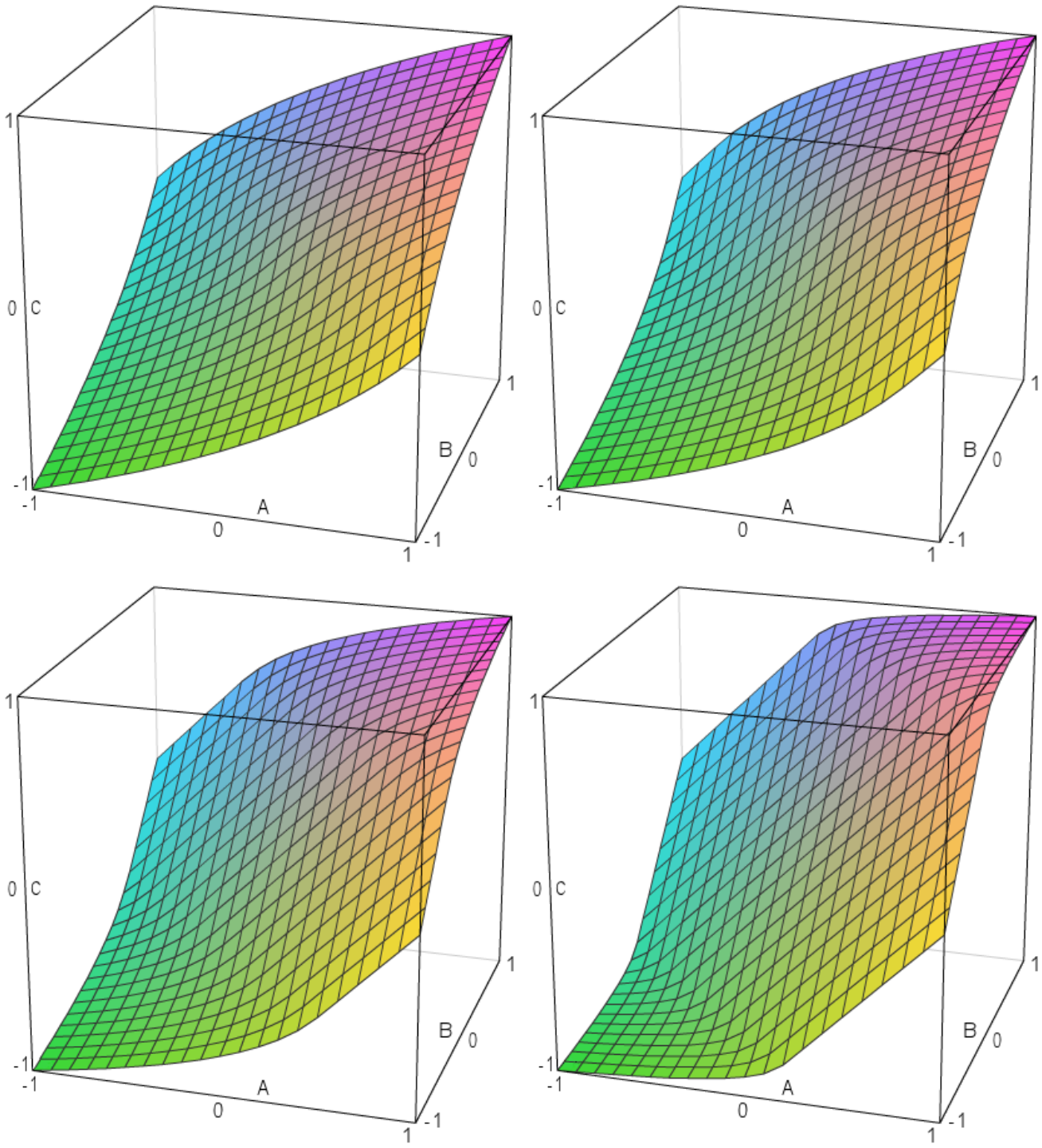
Some transitions between increasing and decreasing pressures are lost.

9 Appendix A

9.1 Linear Dynamic Range Compression with thresholds 0, 0.2, 0.5 and 0.8



9.2 Logarithmic Dynamic Range Compression with thresholds 0, 0.2, 0.5 and 0.8



last update 2012-08-29

<< back >> give feedback