EPRO/IAS

Bachelor Studiengang Medientechnik
Fachhochschule St.Pölten

Patrik Lechner

Wien, December 15, 2016

Introduction

Contents

Part I.

Semester 3

1. Sampling, Waveshaping, und nicht Linearität

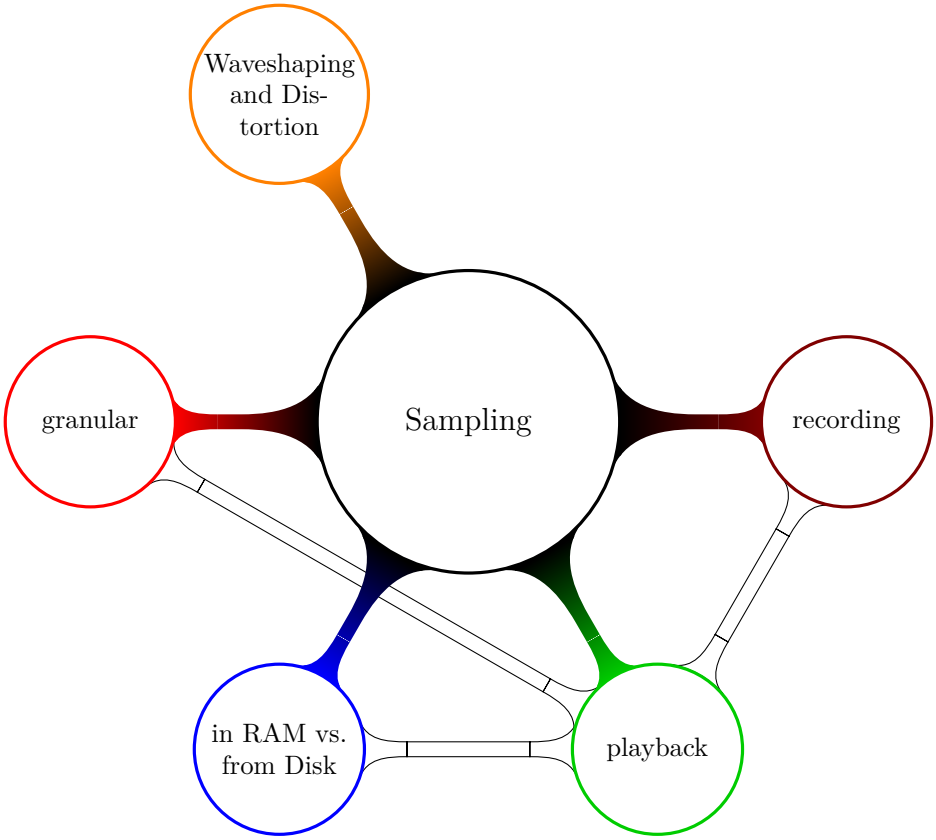

Figure 1.1.: Lecture Contents

1.1. Notizen

Waveshaping generell.
Evt. Subtraktiver Synth am ende.
Zusammenhang waveshaping, distortion. lookup table, sampling, wavescanning.
Waveshaping = modulation (Farnell, 2010, p. 257)
chebyshev

„intermod. distortion “ , vgl. miller puckette,
http://msp.ucsd.edu/techniques/latest/book-html/node78.html
middle term! ((a+b)**2 = a**2 +2ab +b**2)
beispiele: sinus vllt:
dyn. non-linear functions

1.2. Uebersicht

Zunächst: Sampling, dann lineare transfer function.

1. übersicht: fragen wies geht, wie ist es mit Max/MSP?
2. HÜ ankündigen
3. neue abstraction: z-1
4. Raetsel
5. Wo letztes mal stehngeblieben? Poly syth fertig?
6. linearität erklären. Nun Nichtlinearität.
7. wavetable/lookup wiederholung
8. Lookup als waveshaping/distortion
9. waveshaping durch processing vs. durch lookuptable
10. durchrechnen
11. chebyshev
12. Sampling (Ram nicht Ram)
13. send and receive von GUI f. Hausübung

1.3. Waveshaping

Wikipedia quote, page “wavesahper”:
„The mathematics of non-linear operations on audio signals is difficult, and not well understood.“ Waveshaping means distortion. It add overtones, take a look at figure 1.2.

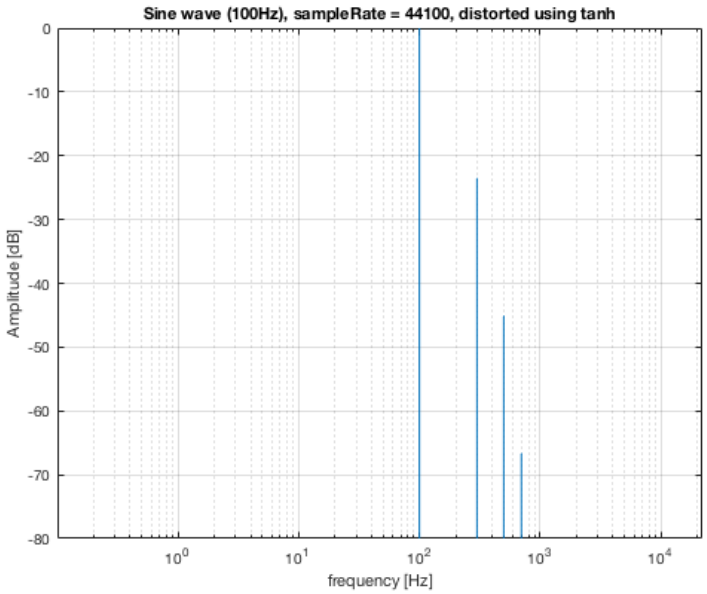

Figure 1.2.: A sine wave has been generated and waveshaping was applied to add overtones.

1.3.1. The simples case: a linear Transfer function.

See 1.3. A linear transfer function is used as a lookup table for a sinosoidal input.

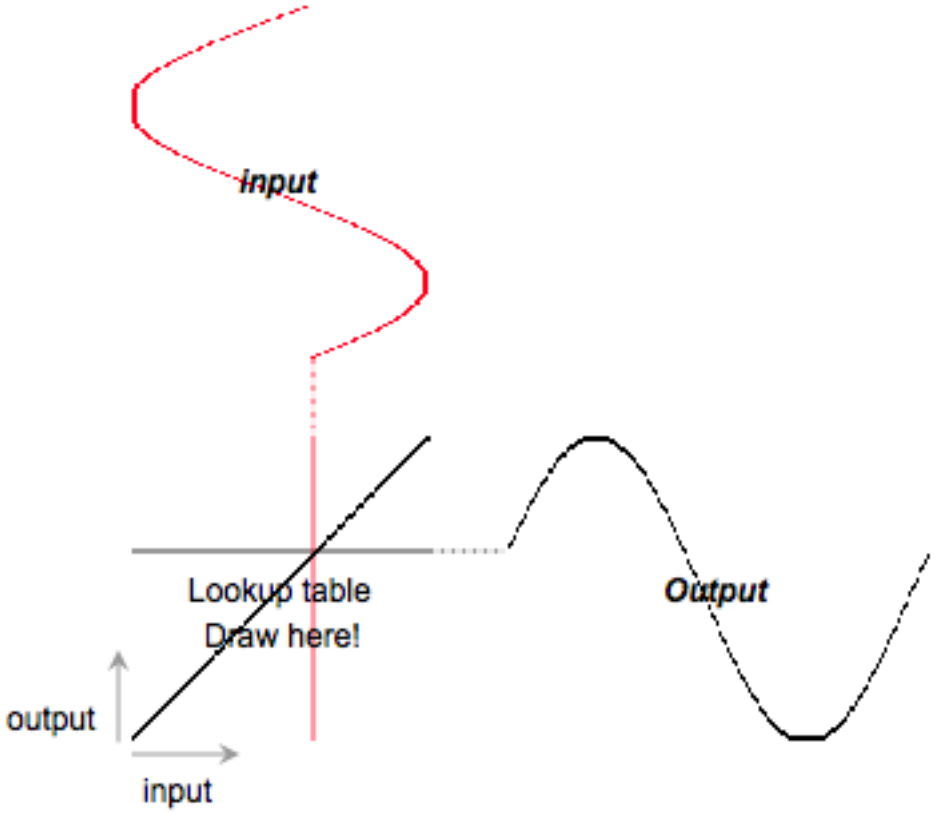

Figure 1.3.: Linear Transfer function

A transfer function in the sense of a waveshaper (a "transfer function" might also mean frequency response in other contexts) is a simple look-up function. Waveshaping means to use an input wave *look up* values in a table or function. A linear transfer function, let's call it l, results in no change, since, by definition, it returns $l(x) = x$. This means, that whatever value we pass in, we get the same value out. Non-liner transfer functions behave differently. They map they input to other values, such as $f(x) = x^2$. It may seem trivial, but if we put 2 into f we get 4 as an output. If we now plot this function we get:

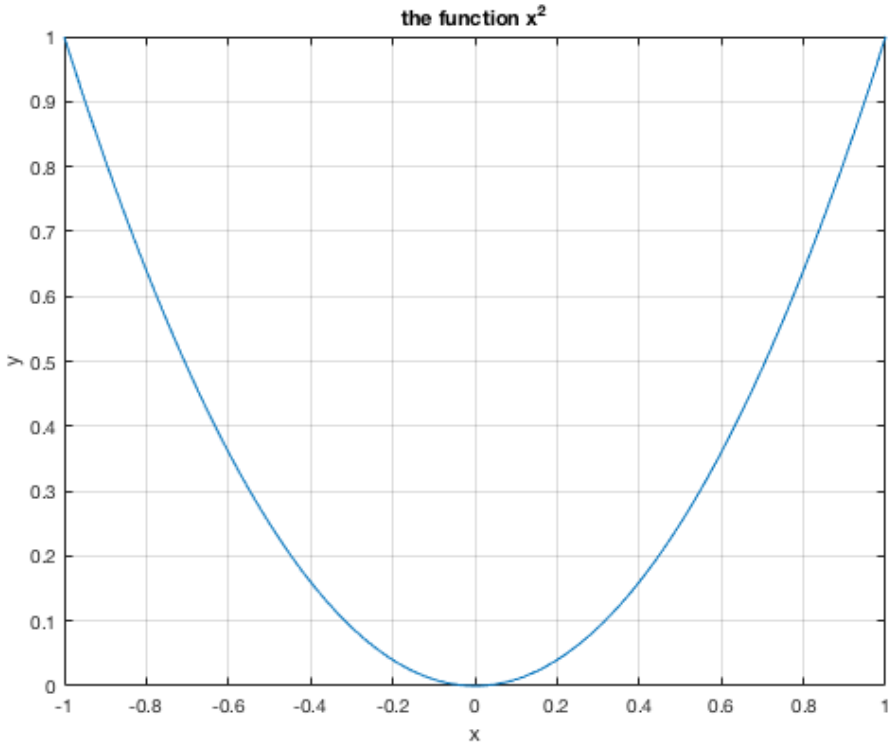

Figure 1.4.: The function $f(x) = x^2$

1.3.2. Simple non-linearity: X^2

A 2nd order polynomial shall be analyzed. The function

$$f(x) = x^2 \tag{1.1}$$

is used and also plotted in figure 1.4.

We can simply plot what happened is we apply the function before trying to understand analytically:

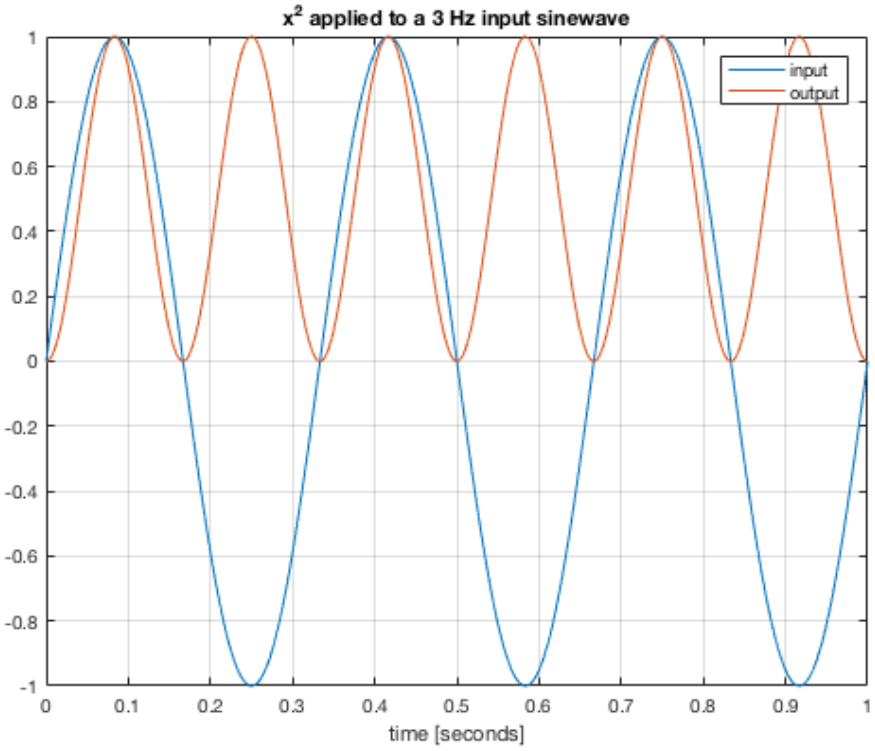

Figure 1.5.: Applying the square function to an input sine wave.

Weird, the input seems to double in frequency. Let's try to understand what's happening.
So we calculate what happens if we send a cosine through this function, so let's take:

$$x = cos(\omega) \tag{1.2}$$

wit arbitrary ω. We can just ignore ω here for a while. Usually, there should be some indexing variable in the cosine function if we want to describe an oscillator that moves over time, but let's also skip that.
So applying our square function we of course get:

$$y = cos(\omega)^2 \tag{1.3}$$

This again results in:

$$y = cos(\omega) \cdot cos(\omega) \tag{1.4}$$

So far so trivial. Note that a multiplication of two oscillators is called *Amplitude Modulation* (actually, in this case we encounter "Ring Modulation", but let's ignore that also), and we know things about Amplitude modulation, namely:

When multiplying two oscillators, we get sum and difference of the two input frequencies. (And the whole output is attenuated by 6dB)

The above statement in equation form:

$$cos(a) \cdot cos(b) = \frac{cos(a+b) + cos(a-b)}{2} \tag{1.5}$$

We could also have looked up this *trigonometric identity*. This means for our experiment with our cosine squared:

$$y = \frac{cos(\omega+\omega) + cos(\omega-\omega)}{2} \tag{1.6}$$

So:

$$y = \frac{cos(2\cdot\omega) + cos(0)}{2} = \frac{cos(2\cdot\omega)}{2} + \frac{1}{2} \tag{1.7}$$

We arrive at the same result! **But is this true for every input? That would mean we just built a frequency shofter, did we? No.** Waveshaping is much more complicated, which is immediately obvious when we try to do the same t two oscillators:

$$x = cos(\omega_1) + cos(\omega_2) \tag{1.8}$$

then

$$y = (cos(\omega_1) + cos(\omega_2))^2 \tag{1.9}$$

$$y = cos(\omega_1)^2 + cos(\omega_2)^2 + 2\cdot cos(\omega_1)\cdot cos(\omega_2) \tag{1.10}$$

And finally:

$$y = \frac{cos(2\cdot\omega_1)}{2} + \frac{1}{2} + \frac{cos(2\cdot\omega_2)}{2} + \frac{1}{2} + 2\cdot(\frac{cos(\omega_1+\omega_2) + cos(\omega_1-\omega_2)}{2}) \tag{1.11}$$

1.3.3. How is Waveshaping related to other techniques?

Sampling

If we take a look at figure 1.11, we see that we play a sound file by accessing a buffer (wavetable) using an index, an oscillator. This is effectively the same setup as we would build for distorting an input sound. Also take a look at figure 1.6, which showing us that waveshaping and wavetable synthesis are identical.

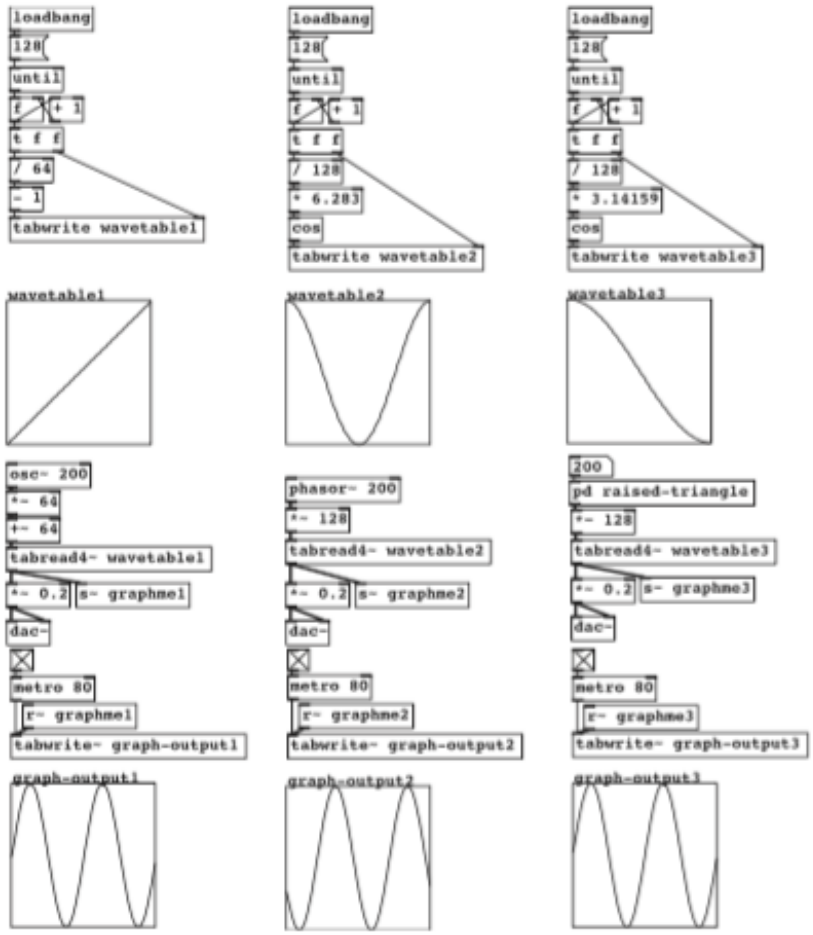

Figure 1.6.: Picture taken from Farnell (2010), showing the identity of waveshaping and wavetable synthesis

Modulation

While we will talk about modulation in a separate chapter, let's loosely define amplitude modulation (AM) as the multiplication od two oscillators and frequency modulation (FM) as the varying the frequency of an oscilator using another oscillator. So, as we have also seen above, AM looks like this:

$$y = sin(a) \cdot sin(b) \tag{1.12}$$

and FM looks like this:

$$y = sin(sin(a)) \tag{1.13}$$

in practice, the a and b terms are a bit more complicated, but we will look at this later. That certain cases of AM are identical to waveshaping has been shown above, think

about the square function again. This of course does not mean that waveshaping can do everything AM can do and it does not mean that AM can achieve everything that waveshaping can. This should just show that we can understand the techniques from the perspective of another. What about FM? Well if our lookup function we use for waveshaping is a sine wave, we arrive at the exact same equation as how we defined FM above. Again, practically speaking, the results we get with these two techniques are very different, but we can see the connections.

1.3.4. Why is Waveshaping useful?

The output spectrum is dependent on the input amplitude. This makes it easy to create complex evolving spectra.

1.3.5. What are the problems with waveshaping?

Waveshaping adds overtones. When we build a waveshaper, we have to be aware of aliasing. Take a look at figures 1.7 and 1.8.

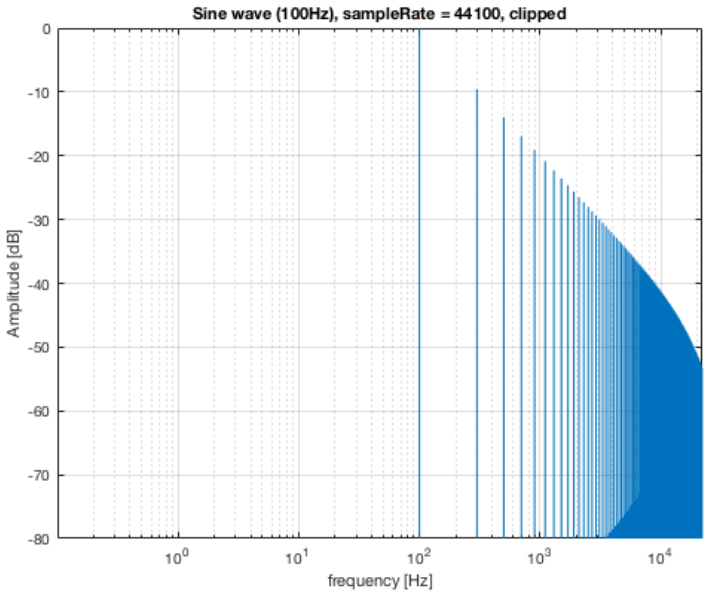

Figure 1.7.: A sine wave was generated and clipped. Clipping is a form of waveshaping which adds many overtones. Note how high frequencies fold back into the lower parts of the spectrum because they exceed the Nyquist-rate.

The problem of aliasing is usually treated by over-sampling. This does not solve the problem but lessens it significantly resulting in cleaner, arguably better sound. Over-sampling means that, if we work at a sample-rate of 44.1kHz, the input is up-sampled, essentially interpolated, to be at a sample-rate of 88.2kHz. Then the waveshaping is applied, leaving room for high frequencies up to 44.1kHz. Using a lowpass filter, high

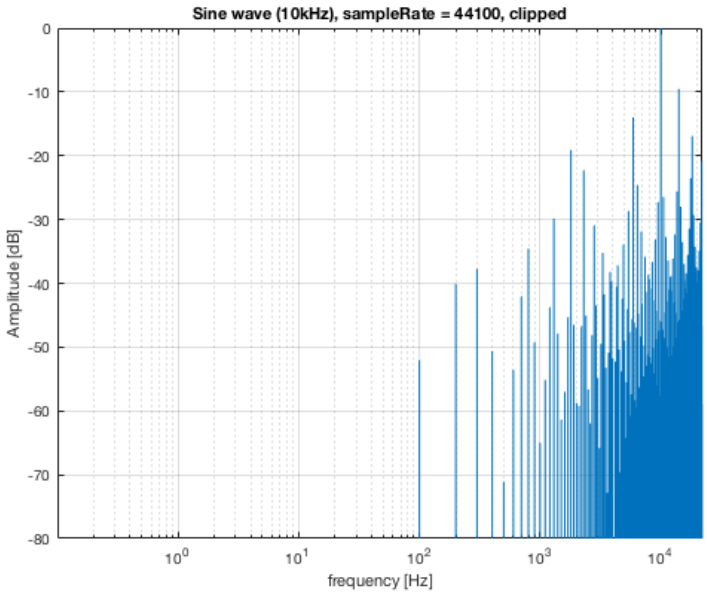

Figure 1.8.: Again, a sine wave, this time with a higher frequency to begin with. Extreme clipping has been applied by boosting the input amplitude. The aliased overtones are all over the place, even below the input frequency.

frequencies over 22.1kHz are then attenuated as much as possible, in order to be able to down-sample again to reach our initial sample-rate of 44.1kHz.
To state it more simple: Waveshaping is usually encapsulated in a process that runs at higher sampling rates in order to lessen aliasing.

1.4. Sampler

Work in progress.

Figure 1.9.: simpleSampler

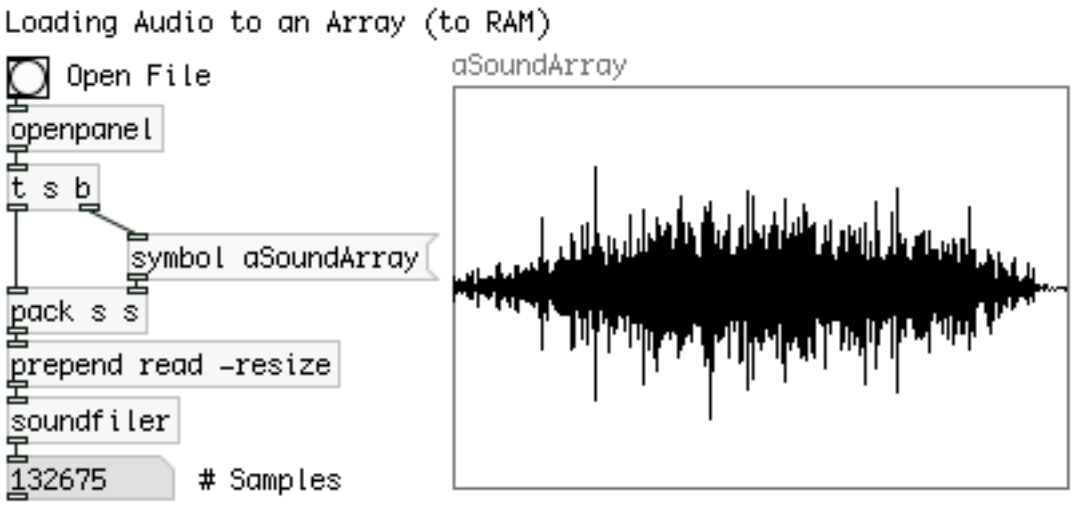

Figure 1.10.: sound in Ram

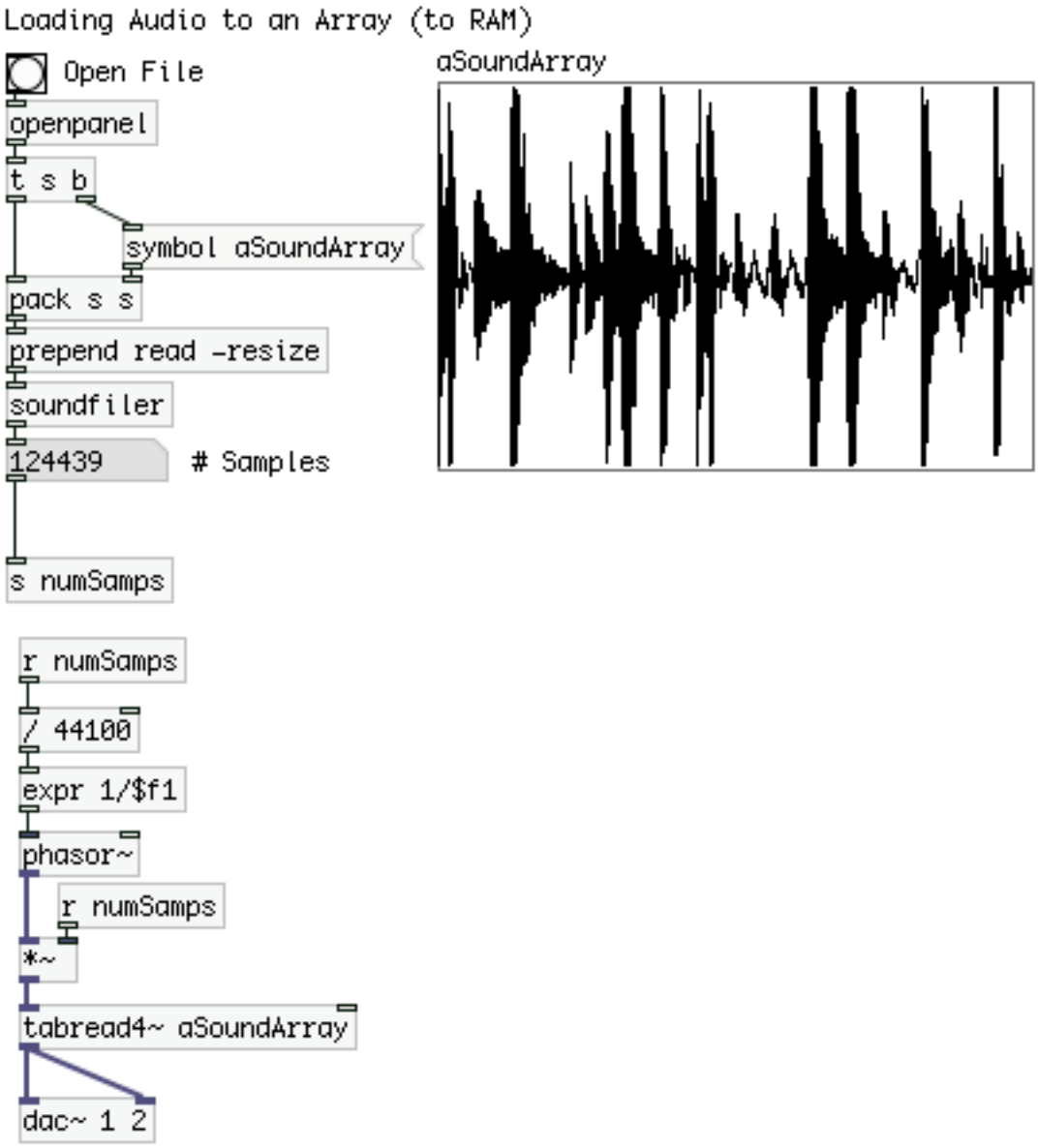

Figure 1.11.: RamFilePlayback

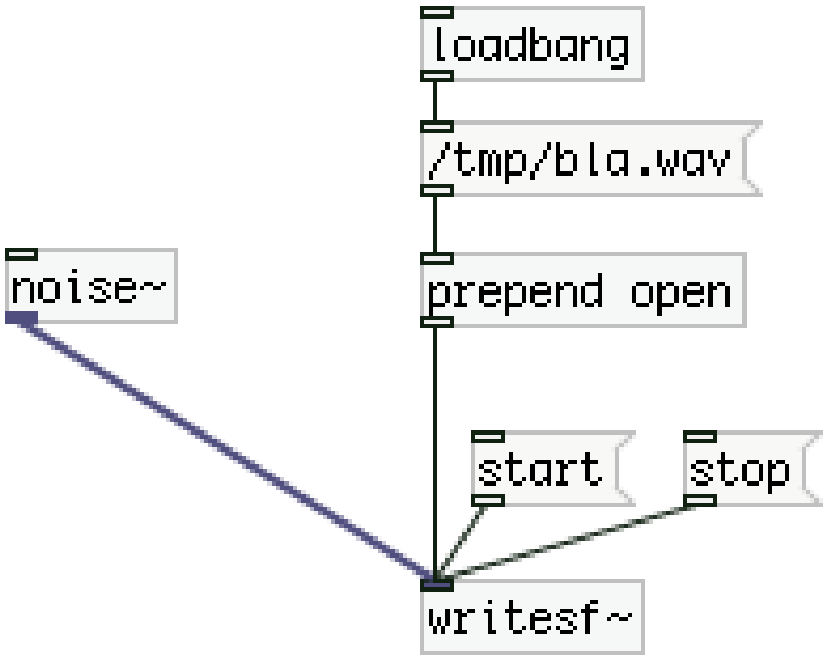

Figure 1.12.: writing Audio to disk

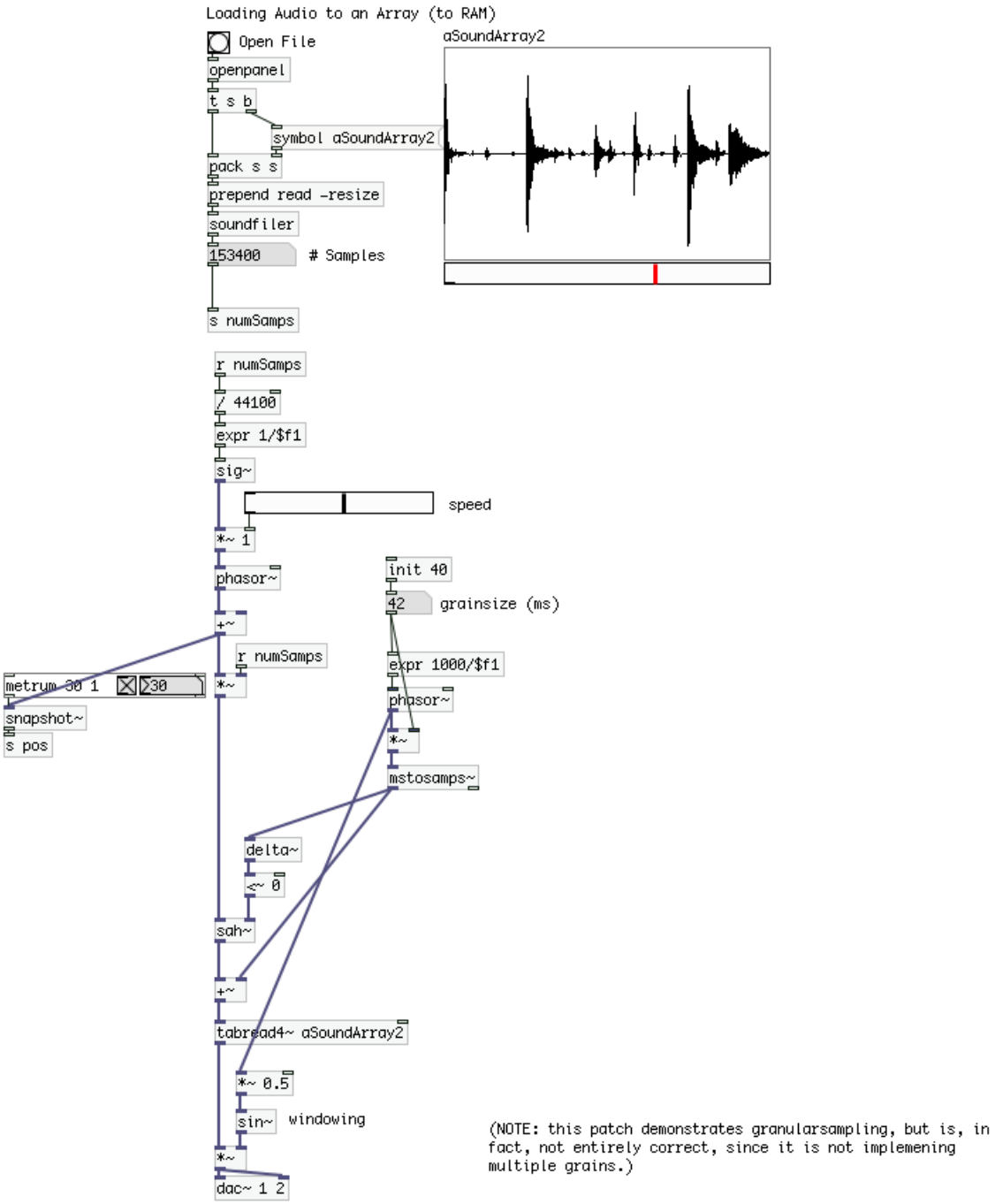

Figure 1.13.: moreSamplimg.pd, a simplified version of granular sampling

1.5. Hausübung

1.5.1. Testmodul

baue ein audio Testmodul mit folgener spezifikation:

- Ein audio output
- verschiedene klangquellen wählbar:
  1. White Noise
  2. Sinus (freq. einstellbar)
  3. soundfile (file wählbar)
- GUI
- verfügbar(in eurem pfad, und jederzeit abrufbar als abstraction)
- output pegel sichtbar (level meter)

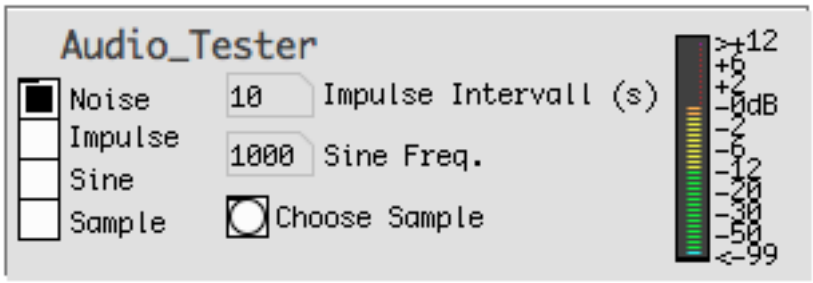

Figure 1.14.: audioTester.pd, zu bauen als Hausübung

List of Figures

Bibliography

Farnell, A. (2010). *Designing sound*. MIT Press, Cambridge, Mass.