Digital Signal Processing

Basic Concepts

Part 1: Sampling

written in June/July 2023 by chn

Website: http://www.chn-dev.net/

Mastodon: https://mastodon.art/@chn/

Github: https://github.com/chn-dev/

Email: chn@chn-dev.net

Introduction

Designing and implementing a software synthesizer or audio effect is a fascinating and rewarding challenge. The maths involved may seem a little daunting for many people, though. This series of short articles summarizes what I have learned over the years about the topic. I'm writing them in order to consolidate my own knowledge and to help other people trying to do some audio/music programming of their own. The concepts will be demonstrated with example programs in C++ using SDL2 and/or JUCE, however the focus is on the fundamental principles. We'll start by discussing what digital audio is mathematically, and work our way up to the point where we can design and implement our own digital filters.

1. Sampling

Before we can apply any computations to audio, we need to get audio data into our computer. That can be done by either directly generating audio data within the computer itself with algorithms or by recording and thereby digitizing sounds from the "real world". Either way, the result is always sequence of quantized numbers in the computer's memory, whereby each number represents the strength of the audio signal at a certain point in time. That's what we call a **time-quantized** and **value-quantized** signal.

1.1 Time Quantization

It can be shown (not here, not now) that all signals, including noise, can be constructed from a combination of sine signals of different frequencies, amplitudes and phases. So it makes sense to sine signals when demonstrating signal processing concepts.

Here's a basic sine function as a time continuous signal:

$x(t)=\sin(\omega t), t\in\mathbb{R}$, with

$\omega=2\pi f_{signal}$ being the angular frequency

The time parameter t is a real number, i.e. within every time interval, there is an infinite number of points with no gaps between them. We put round brackets around the parameter t to make clear that the function defines a time-continuous signal.

When going from continuous time to discrete time, we take sample measurements of the continuous signal at regular time intervals. The resulting discrete-time signal is thereby only defined at specific points in time with a fixed distance between them. That distance between the samples is called the sampling rate T_s. Its inverse is the sampling frequency $f_s=\frac{1}{T_s}$.

Through the process of sampling, a continuous sine signal $x(t)$ becomes a time-discrete signal $x[n]$:

$$x[n]=x(T_s n)=\sin(\omega T_s n)=\sin(2\pi f_{signal} T_s n)=\sin(2\pi \frac{f_{signal}}{f_s} n), n\in\mathbb{N},$$

We put square brackets around the discrete parameter n to make clear that this is a time-discrete signal.

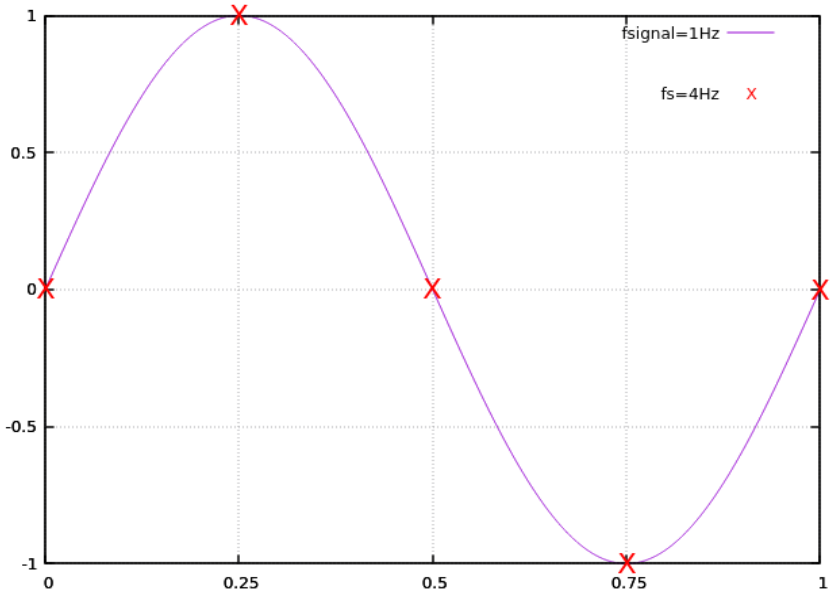
As an example, let's consider sampling a sine signal with the signal frequency $f_{signal}=1\,Hz$ at a sampling frequency $f_s=4\,Hz$. This translates the time-continuous signal

$$x(t)=\sin(2\pi\,1\,Hz\,t)$$

to the time-discrete signal

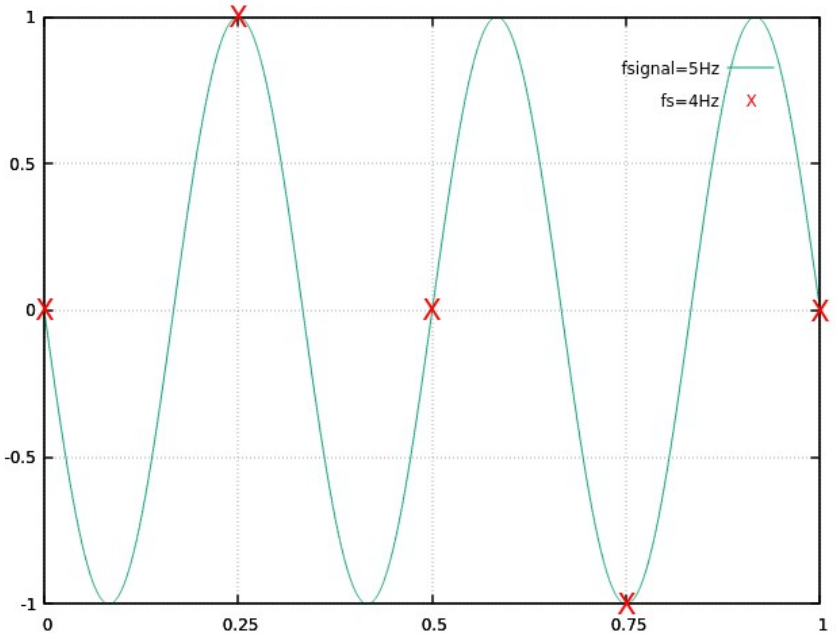
$$x[n]=\sin(2\pi\frac{1\,Hz}{4\,Hz}n)=[0,1,0,-1,0,1,0,-1,..],$$

as illustrated in the following plot:



These four samples are sufficient to reconstruct the original, time-continuous signal.

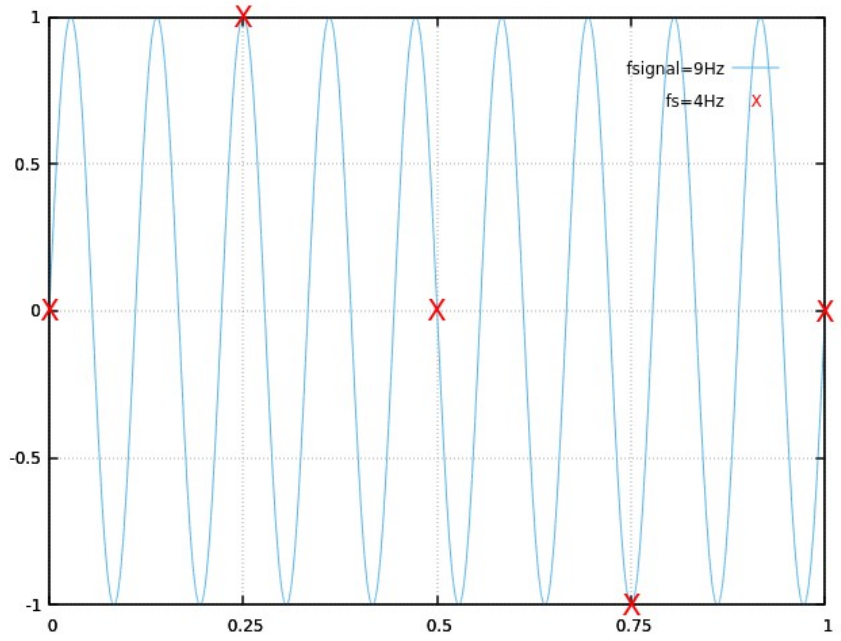
Now let's consider sampling the time-continous signal $x(t)=\sin(2\pi 5\,hz\,t)$ at the same sampling frequency $f_s=4\,Hz$, as shown in the following plot:



The sampled, time-discrete signal is the same as before, namely

$$x[n]=\sin(2\pi\frac{5\,Hz}{4\,Hz}n)=[0,1,0,-1,0,1,0,-1,..].$$

The same is true at $f_s=9\,Hz$:

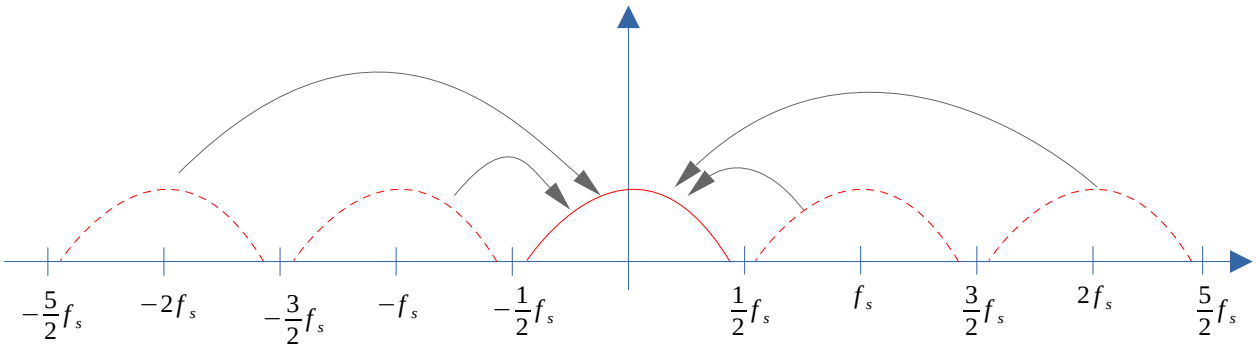


Time-continuous signals with different frequencies can obviously result in the same time-discrete signal when using the same sampling frequency.

Or more generally, all time-continuous signals

$$x_m(t) = \sin(2\pi(f_{signal} + m f_s)t), m \in \mathbb{Z}, t \in \mathbb{R}$$

result in the same time-discrete signal $x[n]$ when sampling them with the sampling frequency f_s, as shown in the following diagram:



The frequency spectrum becomes periodic, i.e. adding a whole multiple of the sampling frequency f_s to the signal frequency f_{signal} results in the same time-discrete signal. Or in mathematical terms:

$$x[n] = x_m(\frac{n}{f_s}) = \sin(2\pi\frac{f_{signal} + m f_s}{f_s} n), n \in \mathbb{Z}, m \in \mathbb{Z}$$

$$\rightarrow x[n] = \sin(2\pi n\frac{f_{signal}}{f_s} + 2\pi n m\frac{f_s}{f_s}) = \sin(2\pi n\frac{f_{signal}}{f_s} + 2\pi n m) = \sin(2\pi n\frac{f_{signal}}{f_s}), n \in \mathbb{Z}, m \in \mathbb{Z}$$

Frequencies beyond $\pm\frac{1}{2}f_s$ are shifted back into the frequency spectrum between $-\frac{1}{2}f_s$ and $\frac{1}{2}f_s$.

That's what we call **aliasing**.

It should be noted that negative signal frequencies can be regarded just the same way as signals with the same absolute frequency played backwards or with a negative sign:

$$\sin(-\omega t) = -\sin(\omega t)$$

The absolute frequency is the same - there's no audible difference.

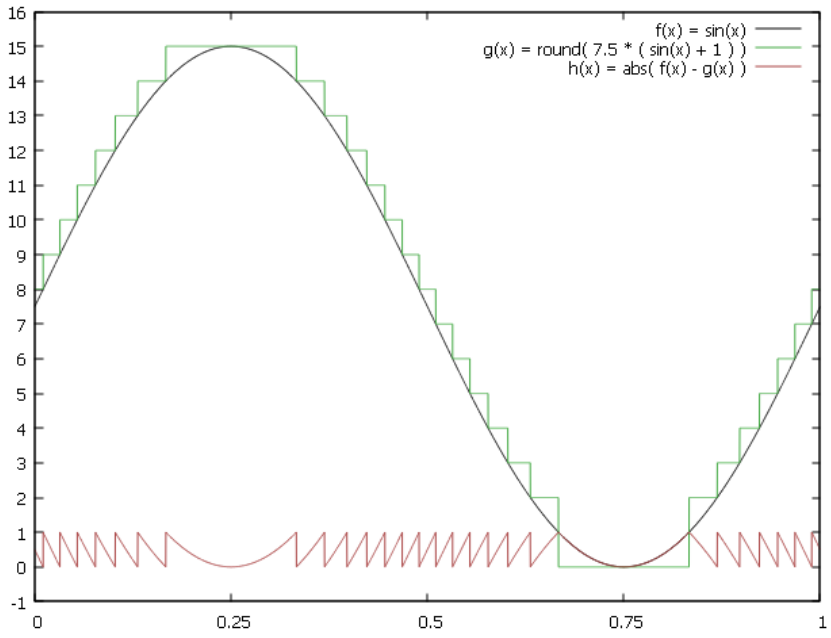
1.2 Value Quantization

The second thing that happens when digitizing a signal is value quantization. Each measurement of the signal at a specific point in time can only have exactly one value from a finite set of possible values after quantization. For example, if you encode a measurement (or sample) as an 8bit integer value, each resulting value can have only exactly one of $2^8 = 256$ possible values ranging from 0 to 255.

Let's again start with a simple time-continuous sine signal ranging from -1 to 1 at a signal freqency of 1Hz:

$$x(t) = \sin(\omega t) = \sin(2\pi\, 1\, Hz\, t)$$

We're going to quantize the values of that signal as 4bit integers, giving us a value range from 0 to 15. Here's a plot of the results of that:



The black line shows the original, (time and value) continuous signal. The green line shows the original signal after quantization to a value range from 0 to 15, representing a 4bit integer.

Through quantization we introduce measurement errors which can be audible as so-called quantization noise, shown by the red line in the plot as the absolute difference between the continuous signal and the quantized signal. Assuming an ideal quantizer, the error introduced is at most 1 in absolute terms and $\frac{1}{2^b}$ in relation to the maximum possible amplitude of the signal.

Quantizing a signal with 4bits introduces an error of $100\% \times \frac{1}{2^4} = 6.25\%$. In audio, we usually express a relative signal strength in terms of dB or Decibel:

$$L_P = 10 \log_{10} \frac{1}{2^b} dB$$

When quantizing with 4 bits, the quantization noise as a power of

$$L_P = 10 \log_{10} \frac{2}{1^4} dB = -12.04 dB.$$

1.3 Summary

When digitizing a signal, its time as well as its values are quantized. **Time quantization** introduces aliasing, an effect where the frequency spectrum of the signal becomes periodic. **Value quantization** introduces digital noise which becomes stronger with fewer number of bits used for representing the values.

The result of the digizing process (or sampling) is a time-discrete, value-quantized signal $x[n]$ which can be used for applying signal processing algorithms in a computer.

Exercise:

Create a small program or plugin which demonstrates the audible effects of aliasing and quantization noise.