FAUST

Digital Audio Processing and Synthesis

AGENDA

October 2025

Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

November 2025

Su	Mo	Tu	We	Th	Fr	Sa
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

Monday, October 13th 2025

02:30PM – 06:30PM (4h)

- ~~Sound~~
- ~~Audio Signal~~
- ~~From Analog to Digital~~
- ~~Sampling~~
- ~~Quantization~~
- ~~Digital Audio Formats~~
- ~~Digital Audio Processing and Synthesis~~
- ~~The Faust programming language~~
- ~~Faust playground!~~

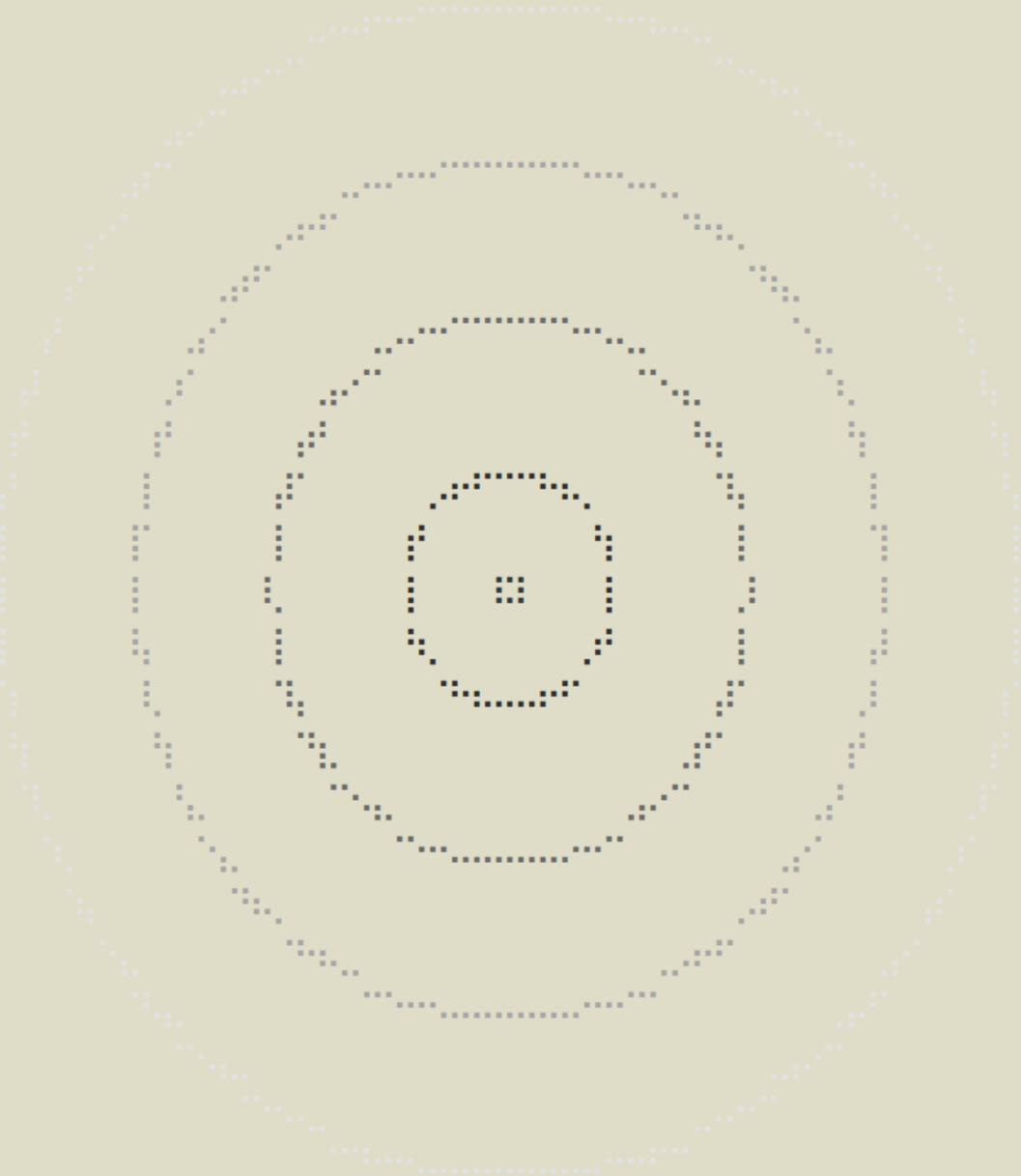
SOUND

[↲] Sound is a ***pressure wave*** that propagates through a **medium** (*gas*, *liquid* or *solid*).

[↲] Propagation is caused by the **oscillation** (*vibration*) of the medium's *particles*, around their ***equilibrium*** positions.

• Sound has the following **properties:**

- **Speed:** ~343 m/s in **air**
- **Amplitude:** in *Pascals* (***Pa***) or *Decibels* (***dB***)
- **Period:** time between two oscillations
- **Wavelength:** distance between two oscillations
- **Frequency:** cycles/sec., in *Hertz* (***Hz***, ***kHz***, ***MHz***)
- **Spectrum**, or *Timbre*



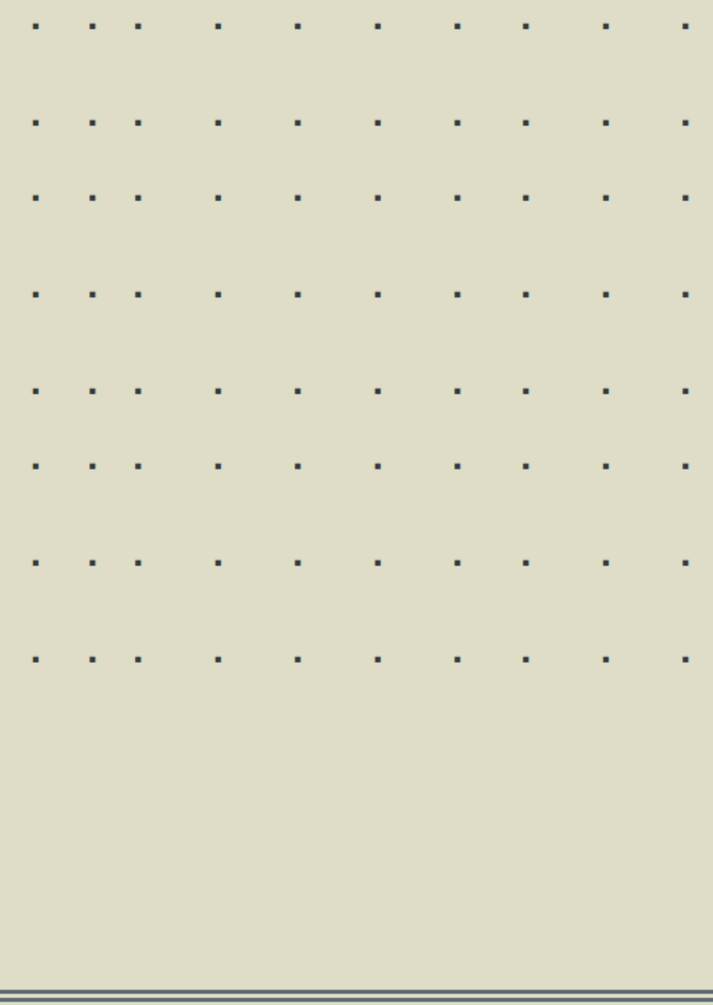
SOUND

[↲] Sound is a ***pressure wave*** that propagates through a **medium** (*gas*, *liquid* or *solid*).

[↲] Propagation is caused by the **oscillation** (*vibration*) of the medium's *particles*, around their ***equilibrium*** positions.

• Sound has the following **properties:**

- **Speed:** ~343 m/s in **air**
- **Amplitude:** in *Pascals* (***Pa***) or *Decibels* (***dB***)
- **Period:** time between two oscillations
- **Wavelength:** distance between two oscillations
- **Frequency:** cycles/sec., in *Hertz* (***Hz***, ***kHz***, ***MHz***)
- **Spectrum**, or *Timbre*

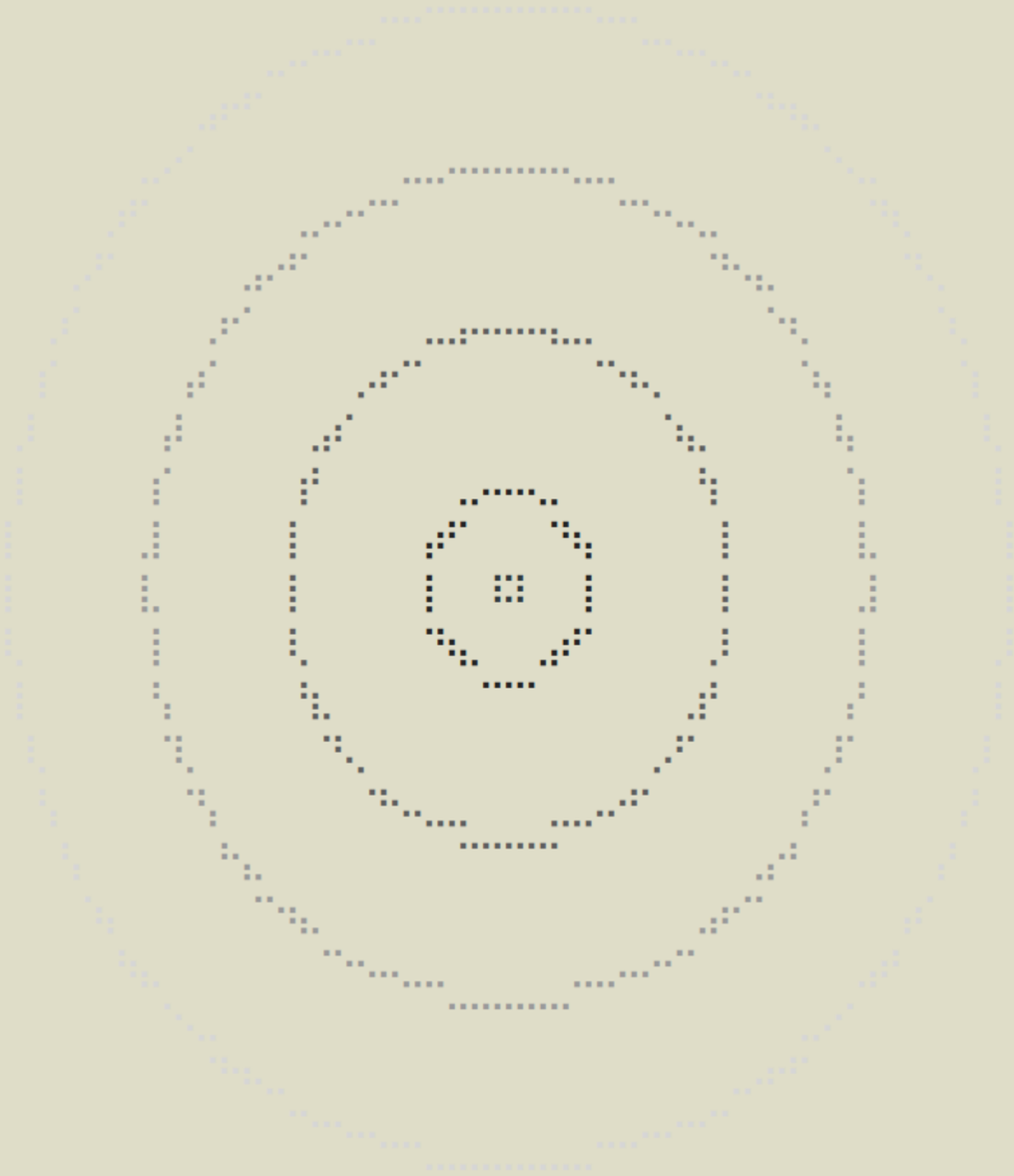


SOUND

[↲] Sound is a ***pressure wave*** that propagates through a **medium** (*gas*, *liquid* or *solid*).

[↲] Propagation is caused by the **oscillation** (*vibration*) of the medium's *particles*, around their ***equilibrium*** positions.

- Sound has the following **properties:**
  - **Speed:** ~343 m/s in **air**
  - **Amplitude:** in *Pascals* (***Pa***) or *Decibels* (***dB***)
  - **Period:** time between two oscillations
  - **Wavelength:** distance between two oscillations
  - **Frequency:** cycles/sec., in *Hertz* (***Hz***, ***kHz***, ***MHz***)
  - **Spectrum**, or *Timbre*



SOUND

• Our ***perception*** of sound is made from the conversion of the vibrations reaching our ***eardrums*** to a *signal of nerve impulses*, transmitted and interpreted by **the brain.**

• Human ears can typically identify sounds ***from 20 Hz to 20 kHz.***

- **Bat:** 2000 to 110,000 Hz
- **Porpoise:** 75 to 150,000 Hz
- **Cat:** 45 to 64,000 Hz
- **Dog:** 67 to 45,000 Hz
- **Chicken:** 125 to 2,000 Hz

SIGNAL

• A ***signal*** describes the evolution of data *over time*. In our case, the vibration of an entity (like the *membrane* of a ***microphone***).

• Just like sound waves turning into nerve impulses, the analyzed data usually needs to be first converted to another *physical unit*, or *domain* (***transduction***) in order to adapt to measurement/processing tools.

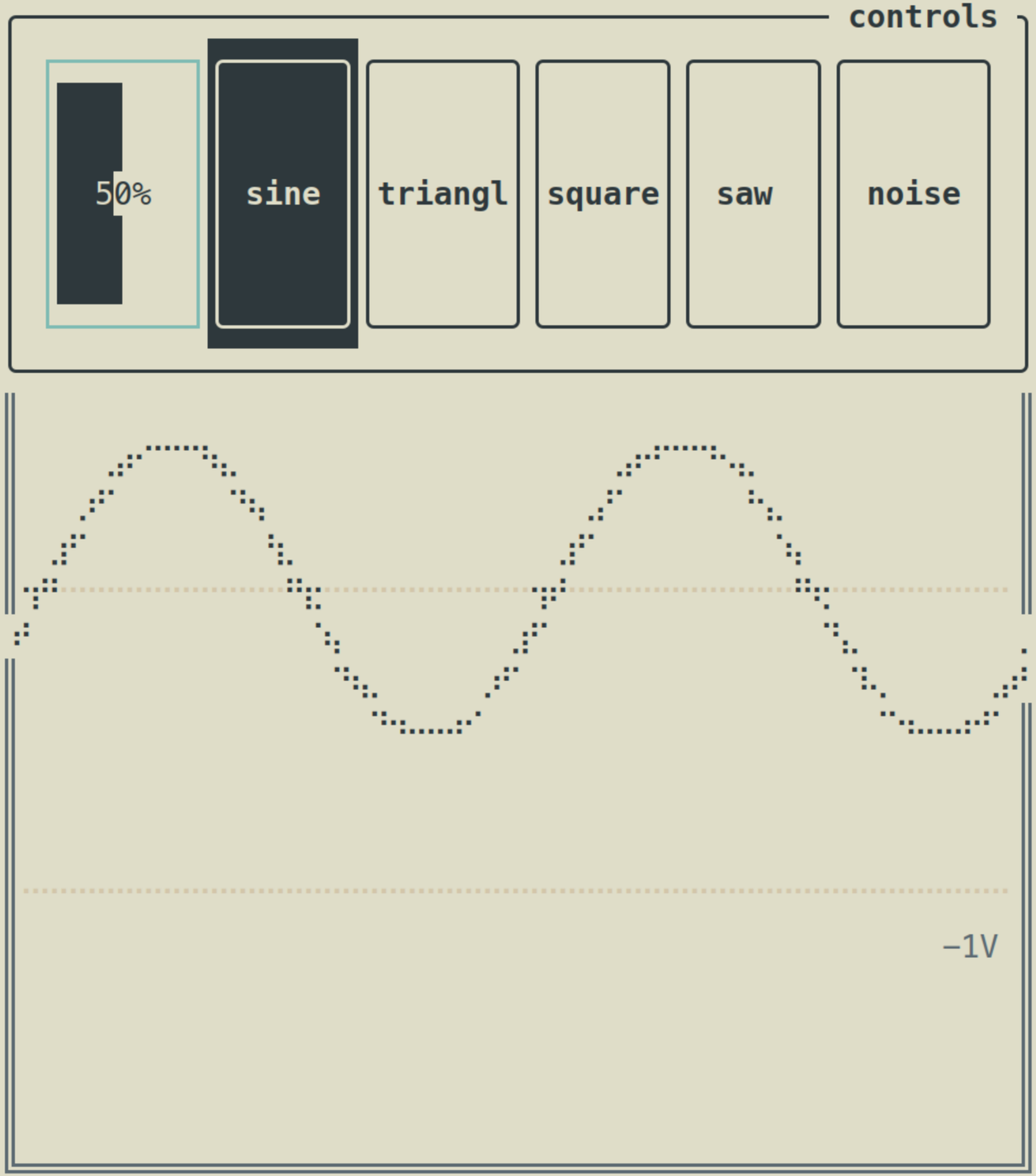
• The vibration of a microphone's membrane is, for instance, usually converted to ***continuous electrical current***, before it can be processed and/or analyzed. In this case, the signal is said to be "***analog***".

SIGNAL

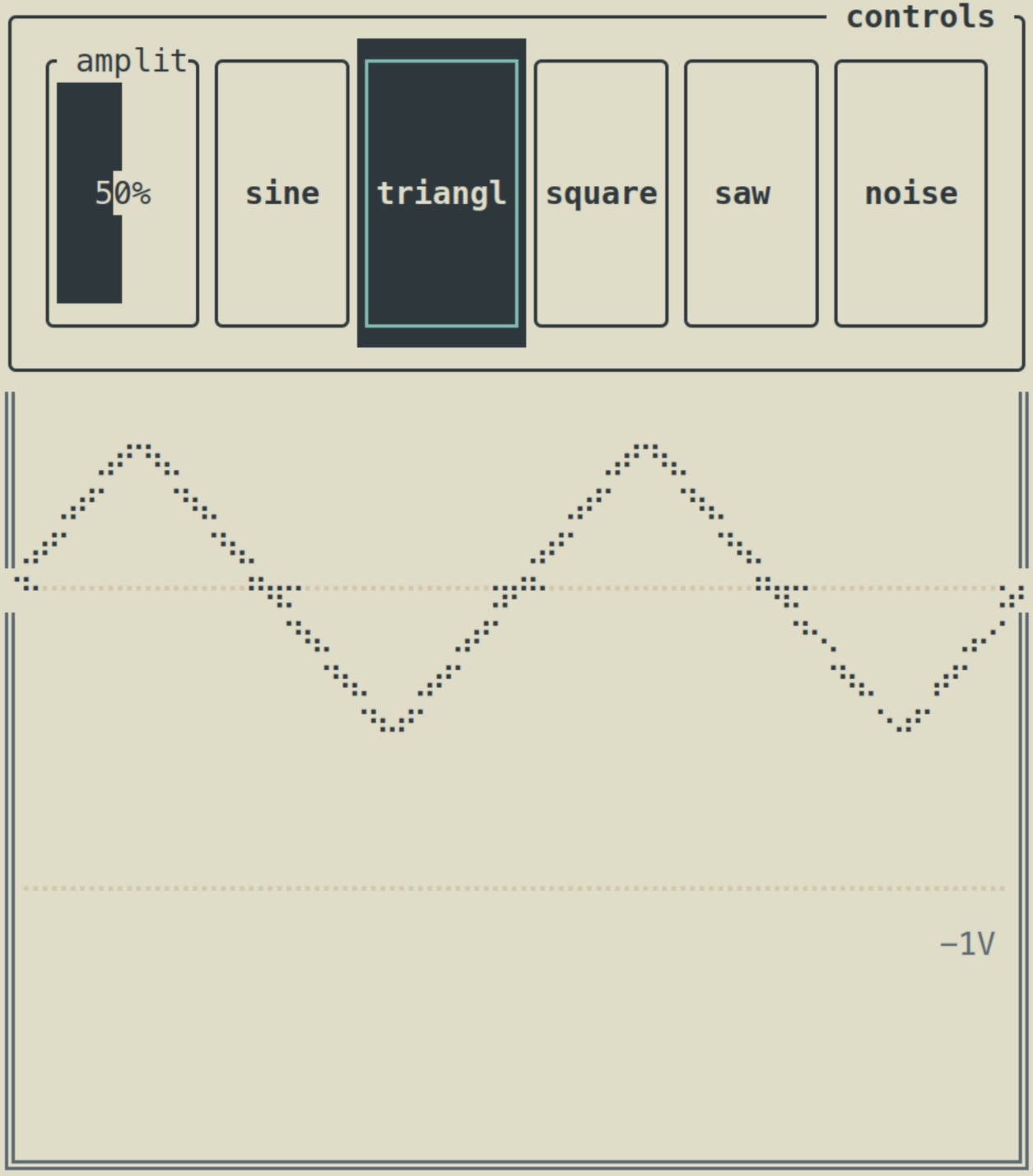
[↲] With an oscilloscope, we can measure the **amplitude** of a signal at a given *point in time* (***time-domain***), through the visualisation of a ***waveform***.

• An analog signal can already be processed as it is, with ***analog effects***: *tape delay*, *distortion*, *chorus*, *reverberation (spring, plate)*, etc.

[↲] On the other hand, it is difficult to extract precise information about ***frequency*** and ***spectrum***. For this purpose, it's far more efficient to switch to the ***frequency domain***, which requires the analog signal to be turned into a ***digital signal***...



SIGNAL



[↲] With an oscilloscope, we can measure the **amplitude** of a signal at a given *point in time* (***time-domain***), through the visualisation of a ***waveform***.

• An analog signal can already be processed as it is, with ***analog effects***: *tape delay, distortion, chorus, reverberation (spring, plate)*, etc.

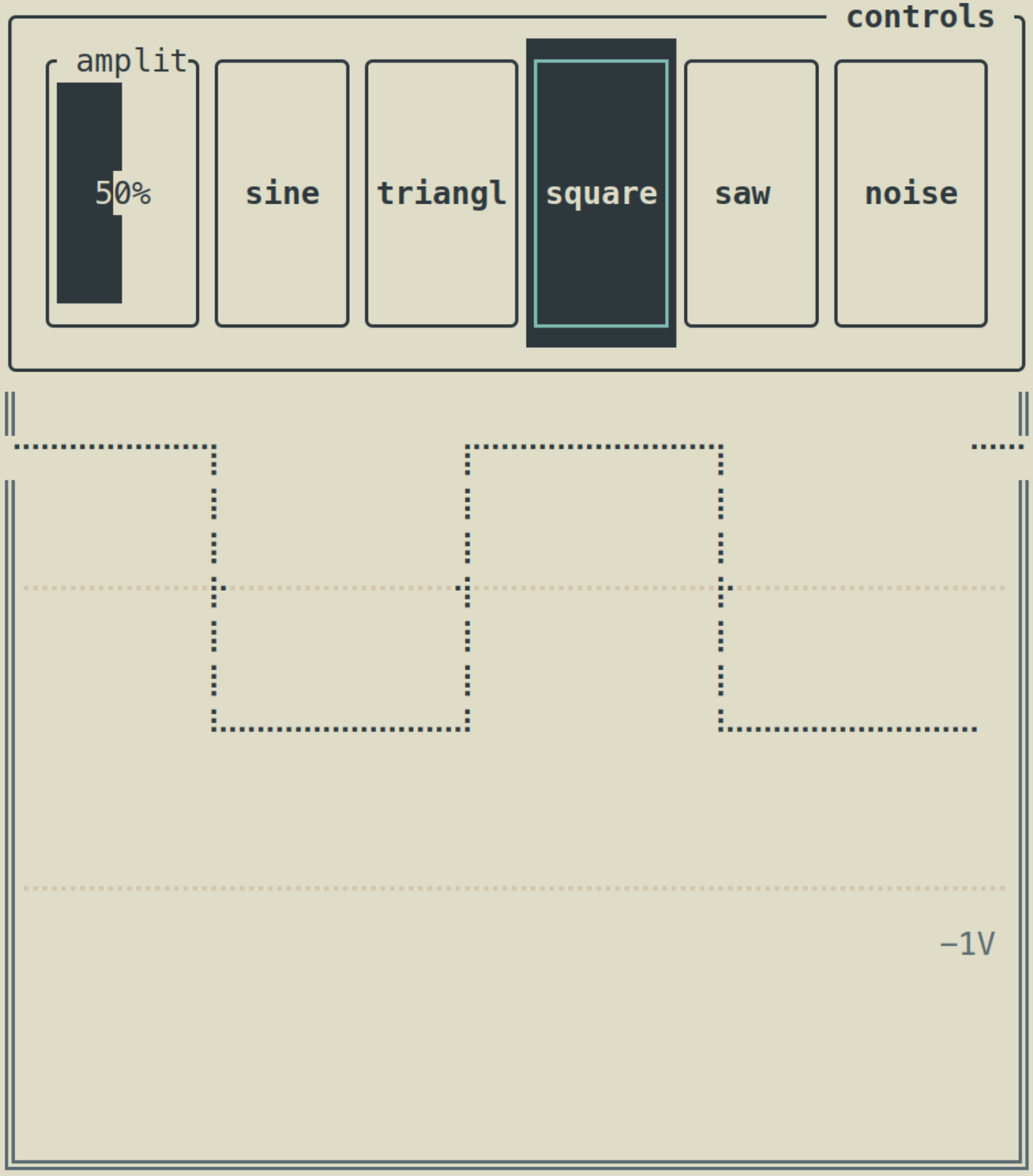
[↲] On the other hand, it is difficult to extract precise information about ***frequency*** and ***spectrum***. For this purpose, it's far more efficient to switch to the ***frequency domain***, which requires the analog signal to be turned into a ***digital signal***...

SIGNAL

[↲] With an oscilloscope, we can measure the **amplitude** of a signal at a given *point in time* (***time-domain***), through the visualisation of a ***waveform***.

• An analog signal can already be processed as it is, with ***analog effects***: *tape delay, distortion, chorus, reverberation (spring, plate)*, etc.

[↲] On the other hand, it is difficult to extract precise information about ***frequency*** and ***spectrum***. For this purpose, it's far more efficient to switch to the ***frequency domain***, which requires the analog signal to be turned into a ***digital signal***...

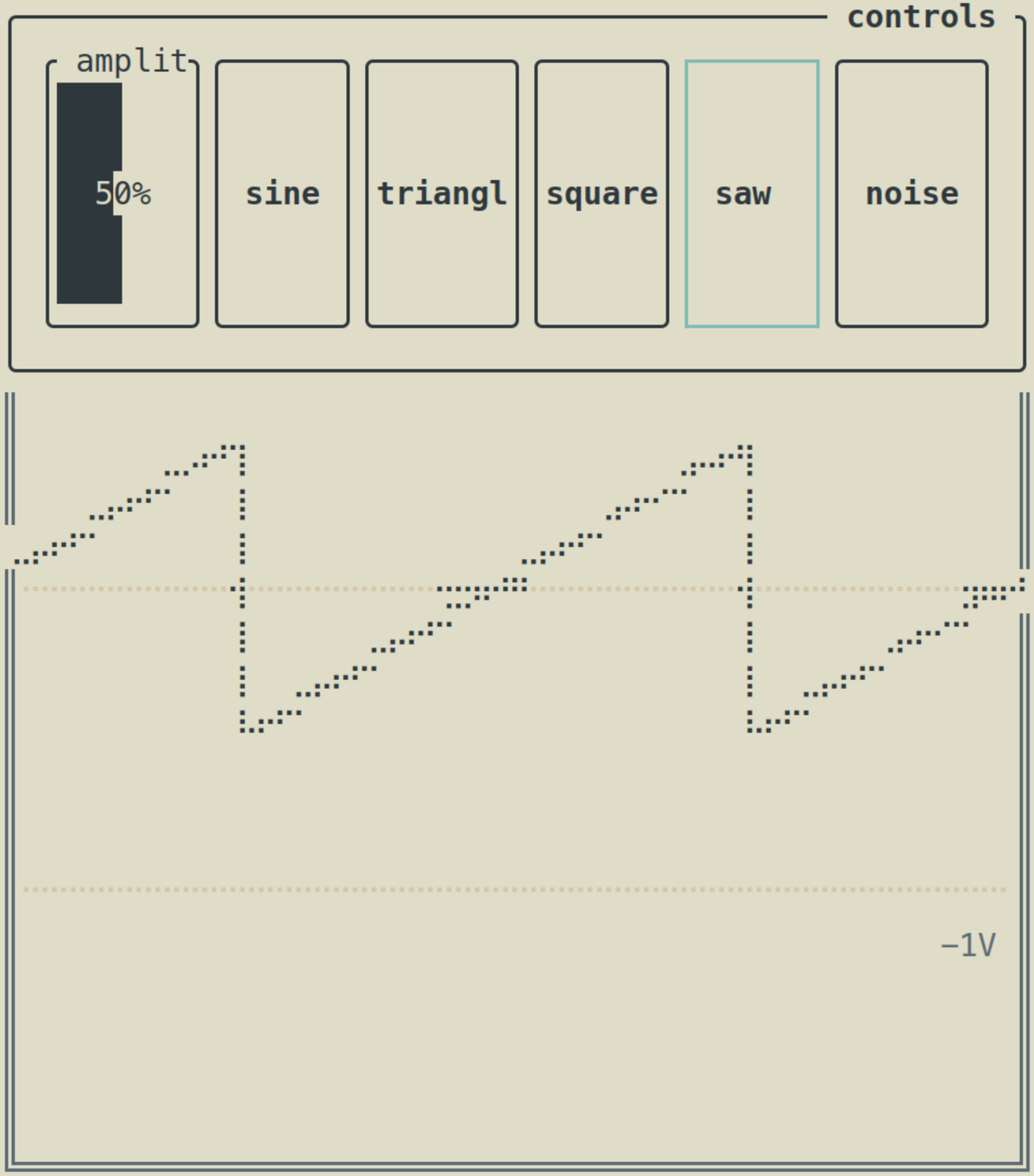


SIGNAL

[↲] With an oscilloscope, we can measure the **amplitude** of a signal at a given *point in time* (***time-domain***), through the visualisation of a ***waveform***.

• An analog signal can already be processed as it is, with ***analog effects***: *tape delay*, *distortion*, *chorus*, *reverberation (spring, plate)*, etc.

[↲] On the other hand, it is difficult to extract precise information about ***frequency*** and ***spectrum***. For this purpose, it's far more efficient to switch to the ***frequency domain***, which requires the analog signal to be turned into a ***digital signal***...



SIGNAL

[↲] With an oscilloscope, we can measure the **amplitude** of a signal at a given *point in time* (***time-domain***), through the visualisation of a ***waveform***.

• An analog signal can already be processed as it is, with ***analog effects***: *tape delay, distortion, chorus, reverberation (spring, plate)*, etc.

[↲] On the other hand, it is difficult to extract precise information about ***frequency*** and ***spectrum***. For this purpose, it's far more efficient to switch to the ***frequency domain***, which requires the analog signal to be turned into a ***digital signal***...

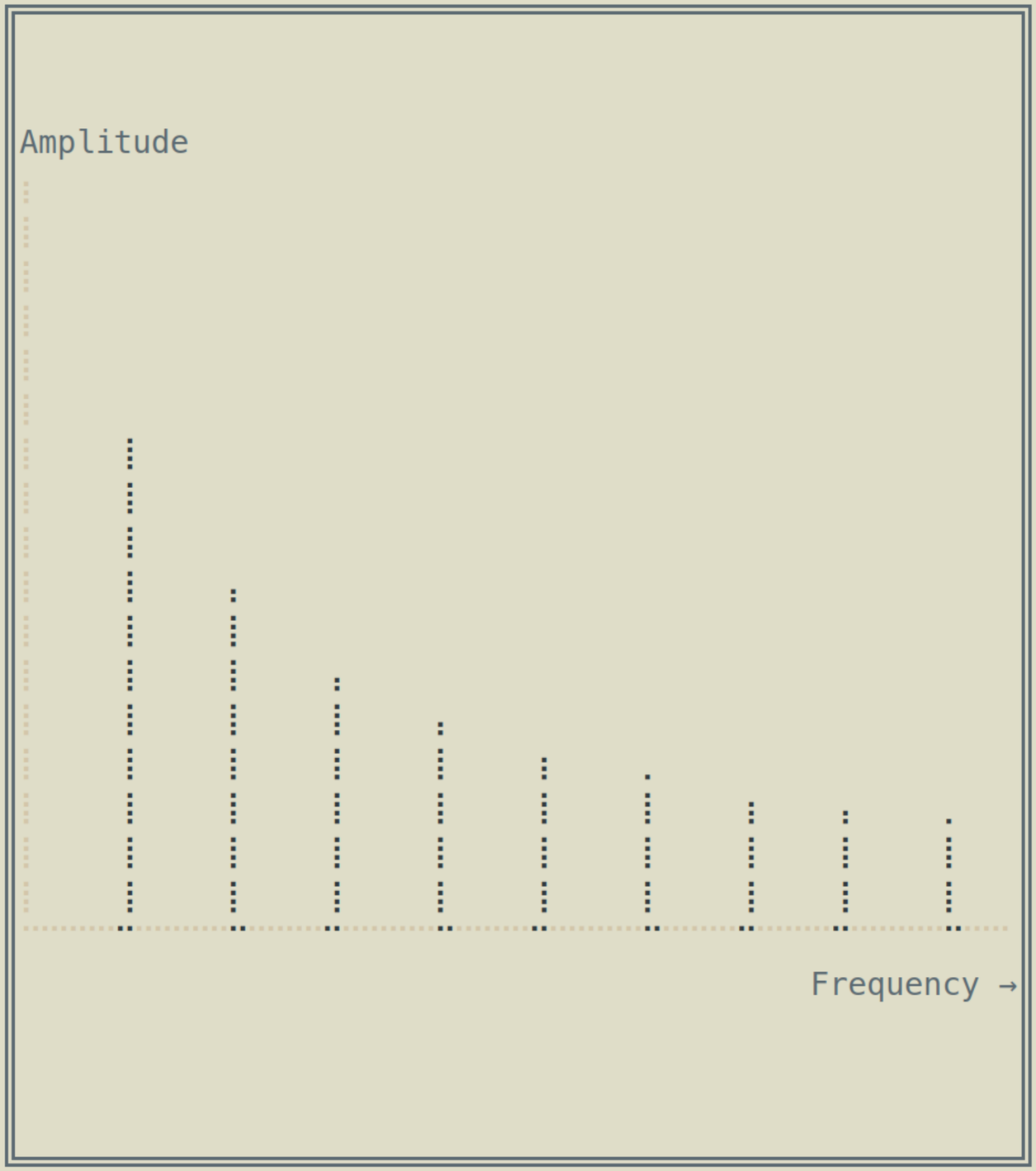
@pchdev – **Emeraude** – *Inria, INSA-Lyon, CITI Lab*

SIGNAL

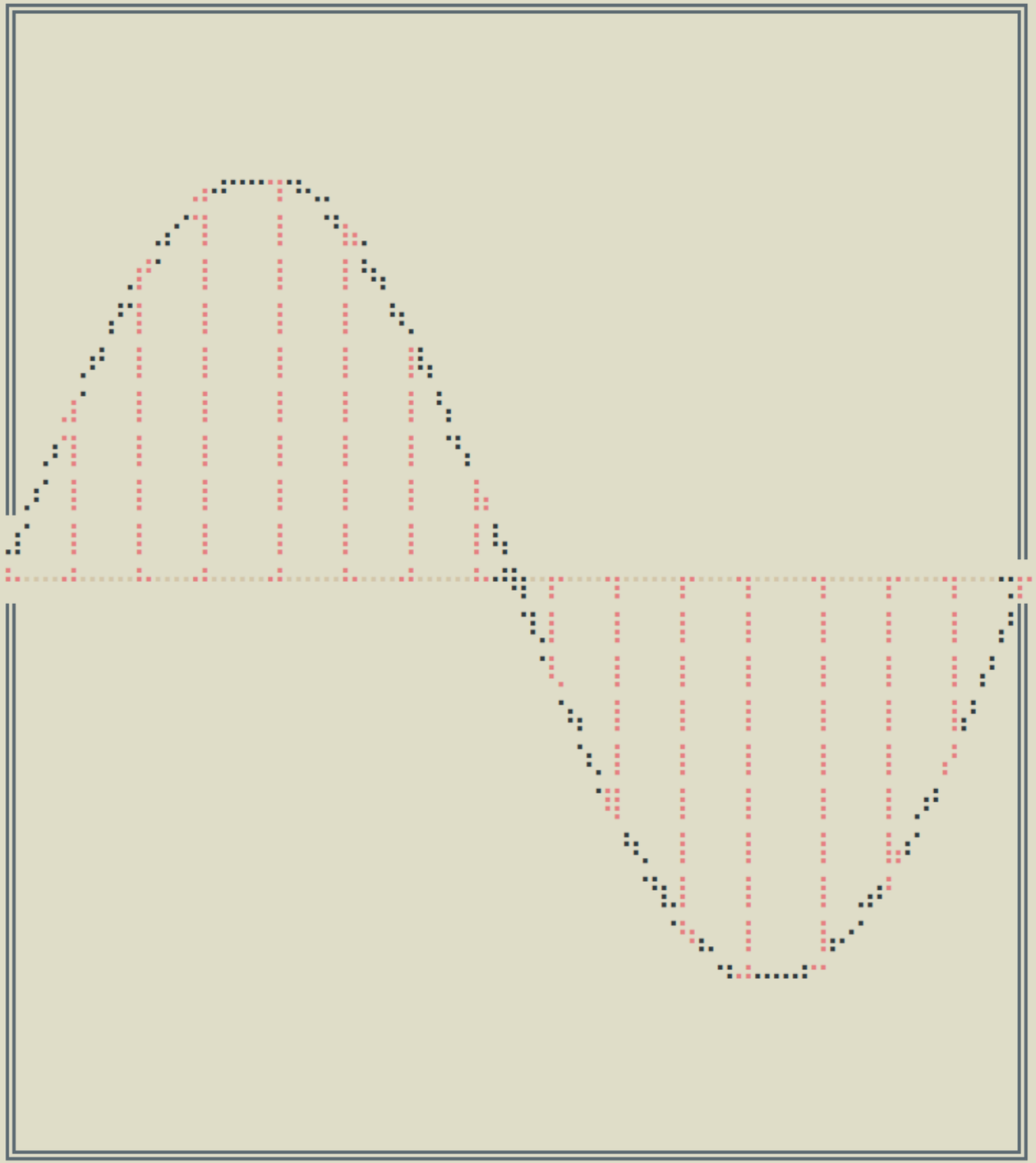
[↲] With an oscilloscope, we can measure the **amplitude** of a signal at a given *point in time* (***time-domain***), through the visualisation of a ***waveform***.

• An analog signal can already be processed as it is, with ***analog effects***: *tape delay*, *distortion*, *chorus*, *reverberation (spring, plate)*, etc.

[↲] On the other hand, it is difficult to extract precise information about ***frequency*** and ***spectrum***. For this purpose, it's far more efficient to switch to the ***frequency domain***, which requires the analog signal to be turned into a ***digital signal***...



DIGITAL SIGNAL



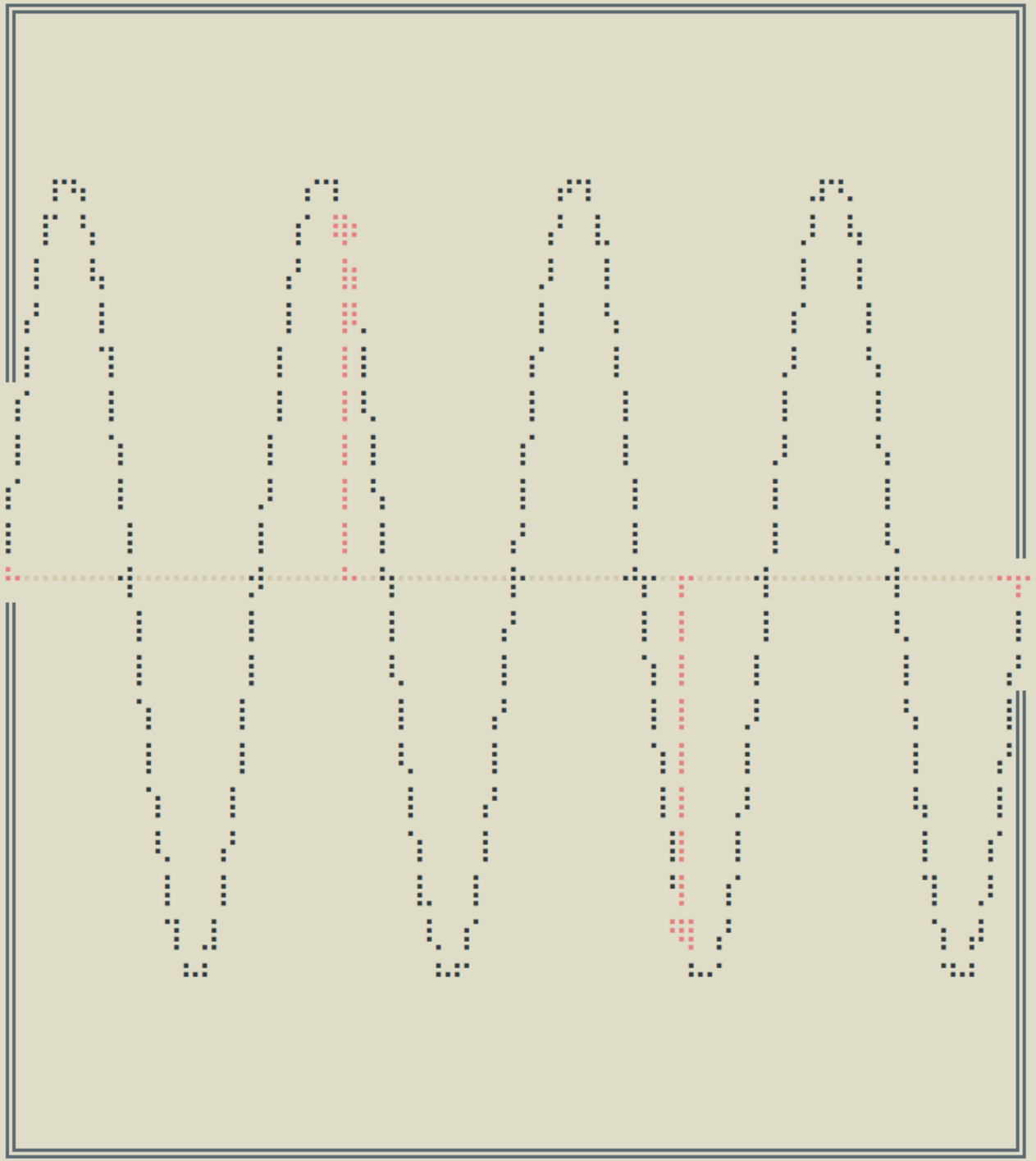
• To ***digitize*** a continuous signal implies *discretizing* it. This is made possible by an *Analog-to-Digital Conversion* (**ADC**) process, which implies two key elements: ***sampling*** and ***quantization.***

[↲] ***Sampling*** means taking a sample of a signal at a certain frequency/rate (***sample rate***).

- **Audio CD:** *44.1 kHz*
- **Pro Audio:** *48/96 kHz*
- **MP3:** *320/256/128/96/64* **kbps**

[↲] • Because of ***aliasing***, the *sampling rate* must be **at least two times superior** to the **highest frequency** we want to represent (*Nyquist-Shannon*).

DIGITAL SIGNAL



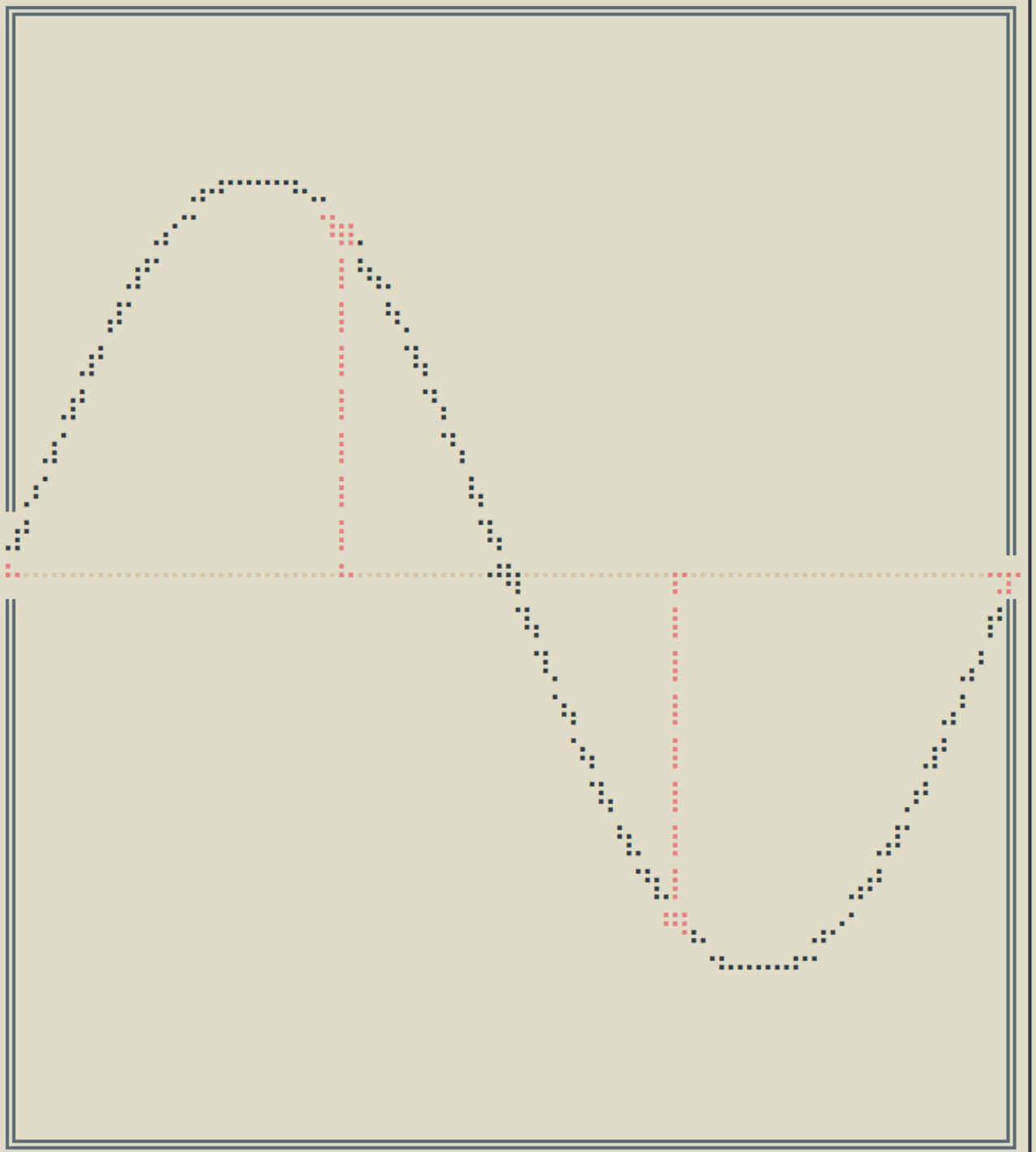
• To ***digitize*** a continuous signal implies *discretizing* it. This is made possible by an *Analog-to-Digital Conversion* (**ADC**) process, which implies two key elements: ***sampling*** and ***quantization.***

[↲] ***Sampling*** means taking a sample of a signal at a certain frequency/rate (***sample rate***).

- **Audio CD:** *44.1 kHz*
- **Pro Audio:** *48/96 kHz*
- **MP3:** *320/256/128/96/64* **kbps**

[↲] • Because of ***aliasing***, the *sampling rate* must be **at least two times superior** to the **highest frequency** we want to represent (*Nyquist-Shannon*).

DIGITAL SIGNAL



• To ***digitize*** a continuous signal implies *discretizing* it. This is made possible by an *Analog-to-Digital Conversion* (**ADC**) process, which implies two key elements: ***sampling*** and ***quantization.***

[↲] ***Sampling*** means taking a sample of a signal at a certain frequency/rate (***sample rate***).

- **Audio CD:** *44.1 kHz*
- **Pro Audio:** *48/96 kHz*
- **MP3:** *320/256/128/96/64* **kbps**

[↲] • Because of ***aliasing***, the *sampling rate* must be **at least two times superior** to the **highest frequency** we want to represent (*Nyquist-Shannon*).

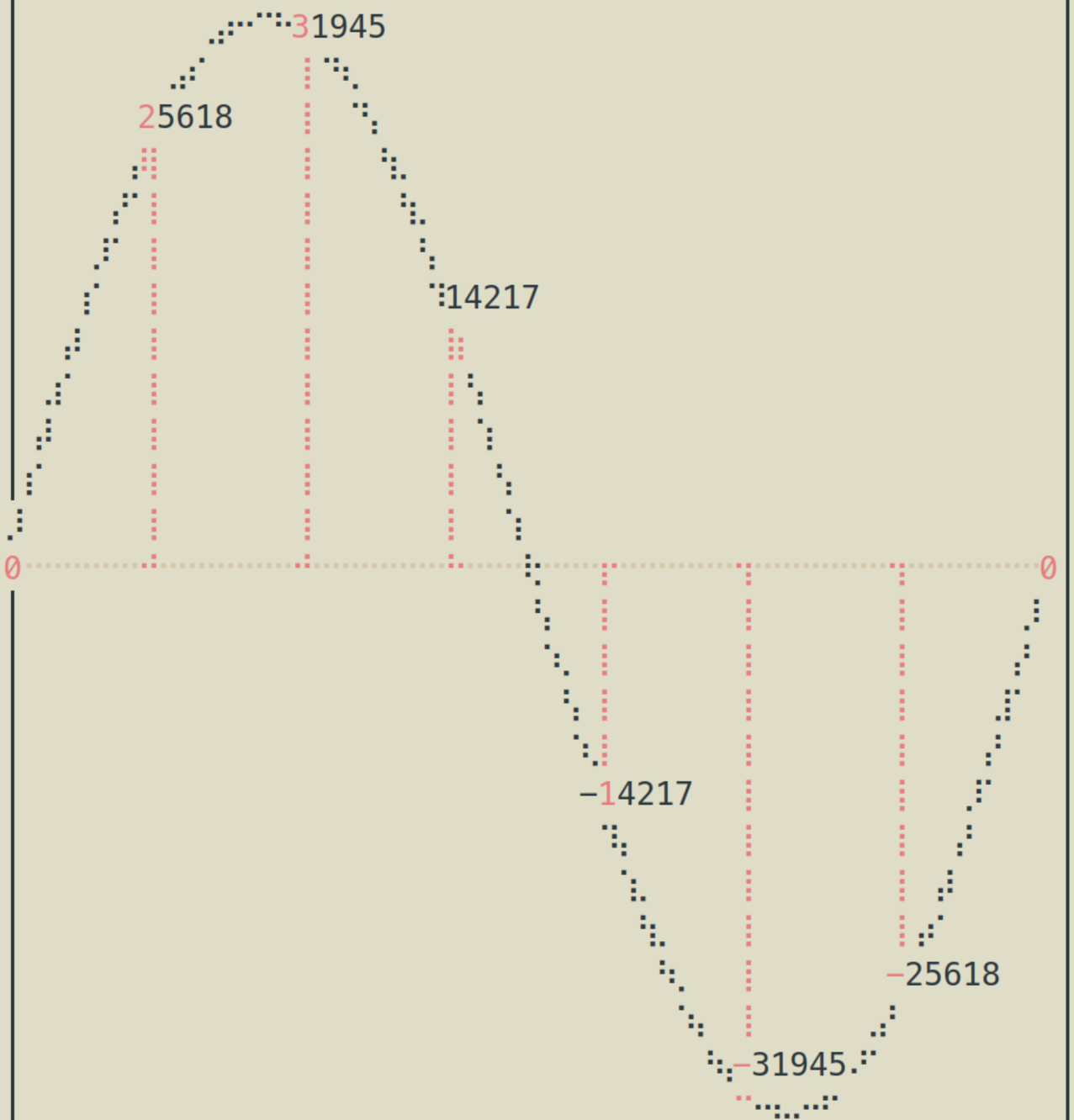
DIGITAL SIGNAL

[↲] • Once we take a sample of a signal at a given time, we need to determine the ***scale of its value***, this is called ***quantization***. Increasing the scale implies reducing the ***quantization noise*** (*quality vs. storage tradeoff*).

↳ • **Audio CD**: 16-bits (65,536 values, 98 dB SNR)
  • **Pro Audio**: 24-bits (~16,7 mil., 146 dB SNR)
  • ***DSP***: 32/64-bits float (~4,3 bil., 194 dB SNR)

• For **DSP**, it is easier to make computations in ***floating-point*** (decimal), and *normalize* the signal between *-1.0* and *1.0*.

• Finally, sending a digital signal to audio speakers involves the inverse process of an **ADC**: *Digital-to-Analog Conversion* (**DAC**).



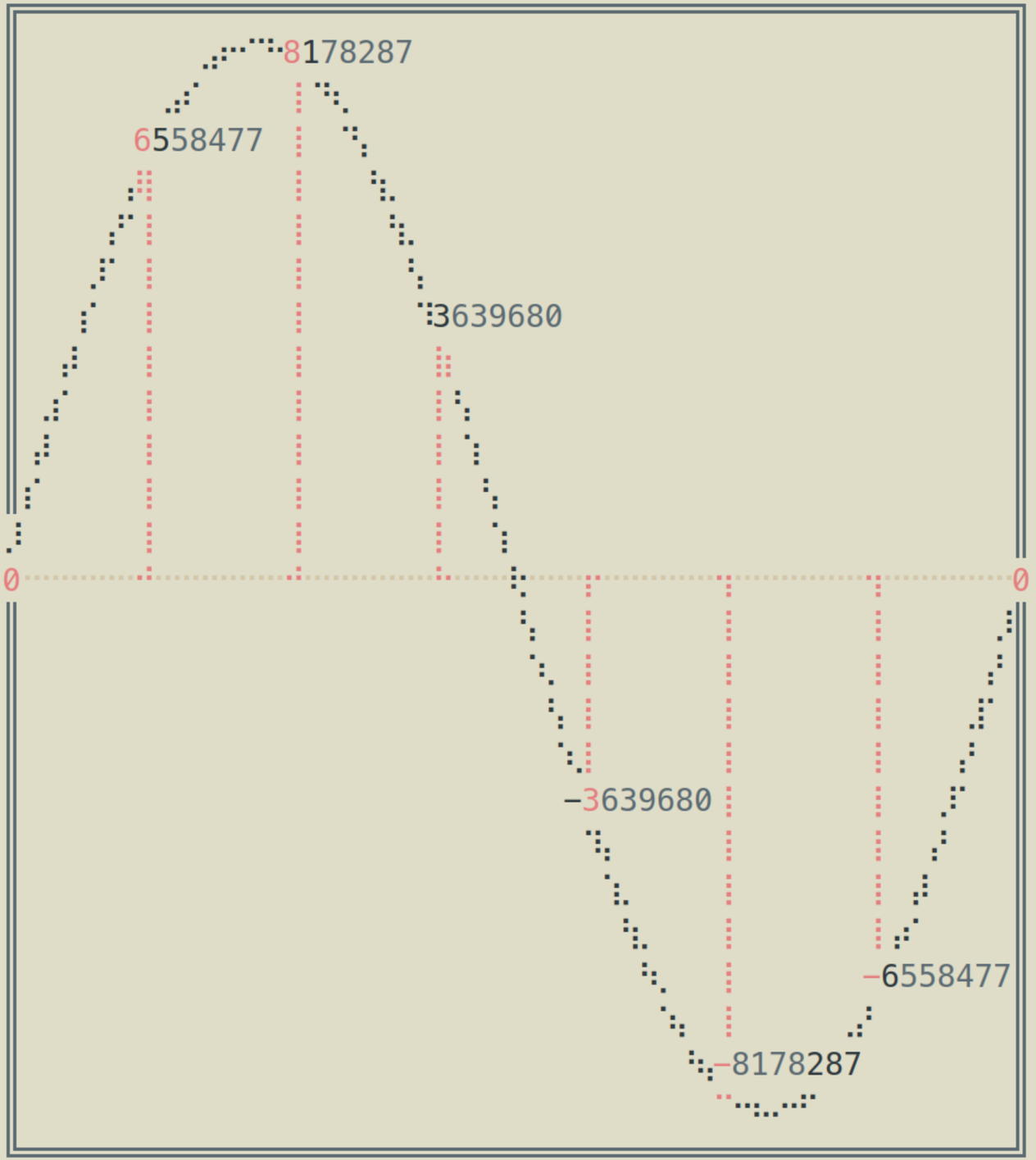
DIGITAL SIGNAL

[↲] • Once we take a sample of a signal at a given time, we need to determine the ***scale of its value***, this is called ***quantization***. Increasing the scale implies reducing the ***quantization noise*** (*quality vs. storage tradeoff*).

- **Audio CD**: 16-bits (65,536 values, 98 dB SNR)
- **Pro Audio**: 24-bits (~16,7 mil., 146 dB SNR)
- ***DSP***: 32/64-bits float (~4,3 bil., 194 dB SNR)

• For **DSP**, it is easier to make computations in ***floating-point*** (decimal), and *normalize* the signal between *-1.0* and *1.0*.

• Finally, sending a digital signal to audio speakers involves the inverse process of an **ADC**: *Digital-to-Analog Conversion* (**DAC**).



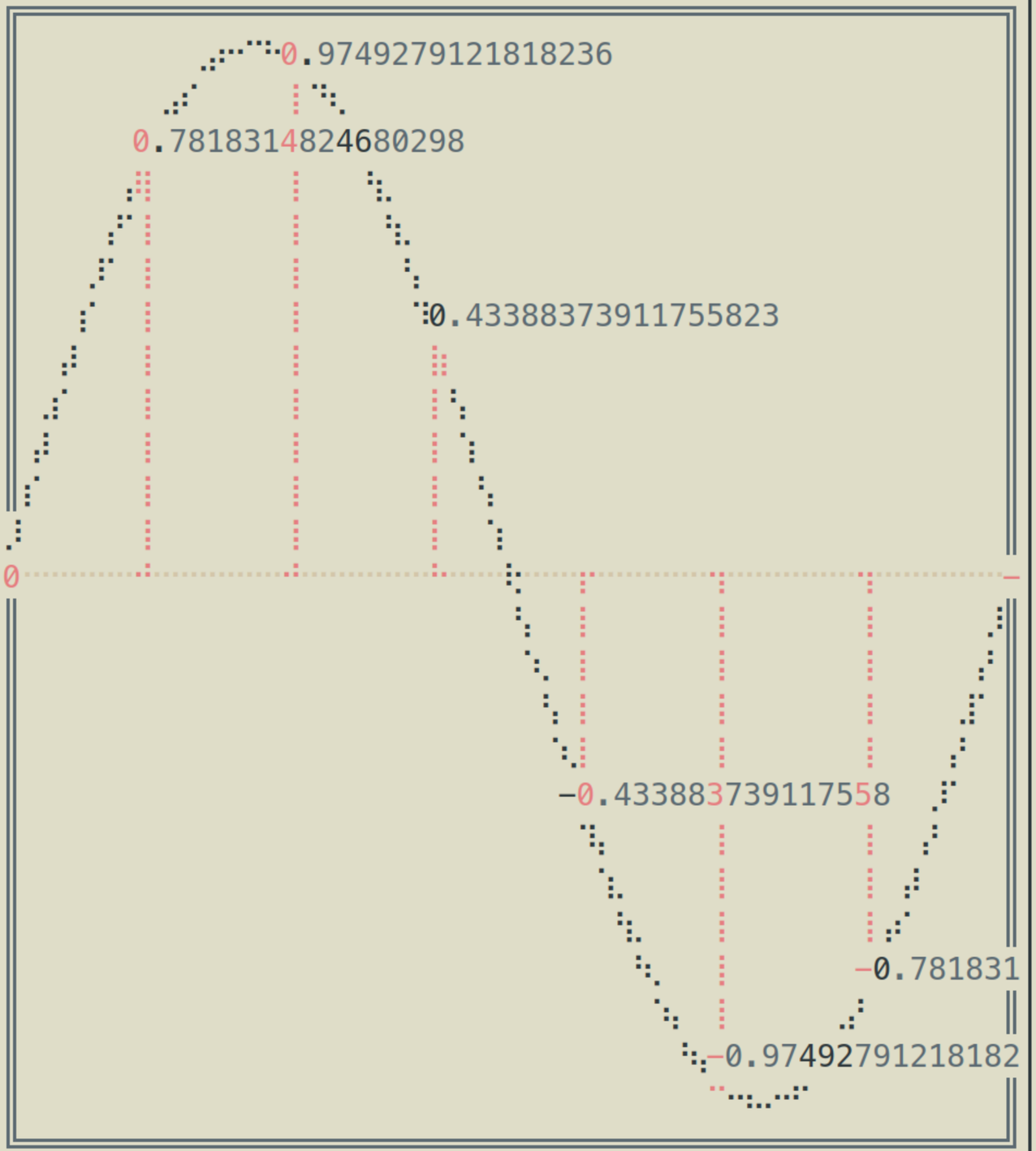
DIGITAL SIGNAL

[↲] • Once we take a sample of a signal at a given time, we need to determine the ***scale of its value***, this is called ***quantization***. Increasing the scale implies reducing the ***quantization noise*** (*quality vs. storage tradeoff*).

- **Audio CD**: 16-bits (65,536 values, 98 dB SNR)
- **Pro Audio**: 24-bits (~16,7 mil., 146 dB SNR)
- ***DSP***: 32/64-bits float (~4,3 bil., 194 dB SNR)

• For **DSP**, it is easier to make computations in ***floating-point*** (decimal), and *normalize* the signal between *-1.0* and *1.0*.

• Finally, sending a digital signal to audio speakers involves the inverse process of an **ADC**: *Digital-to-Analog Conversion* (**DAC**).



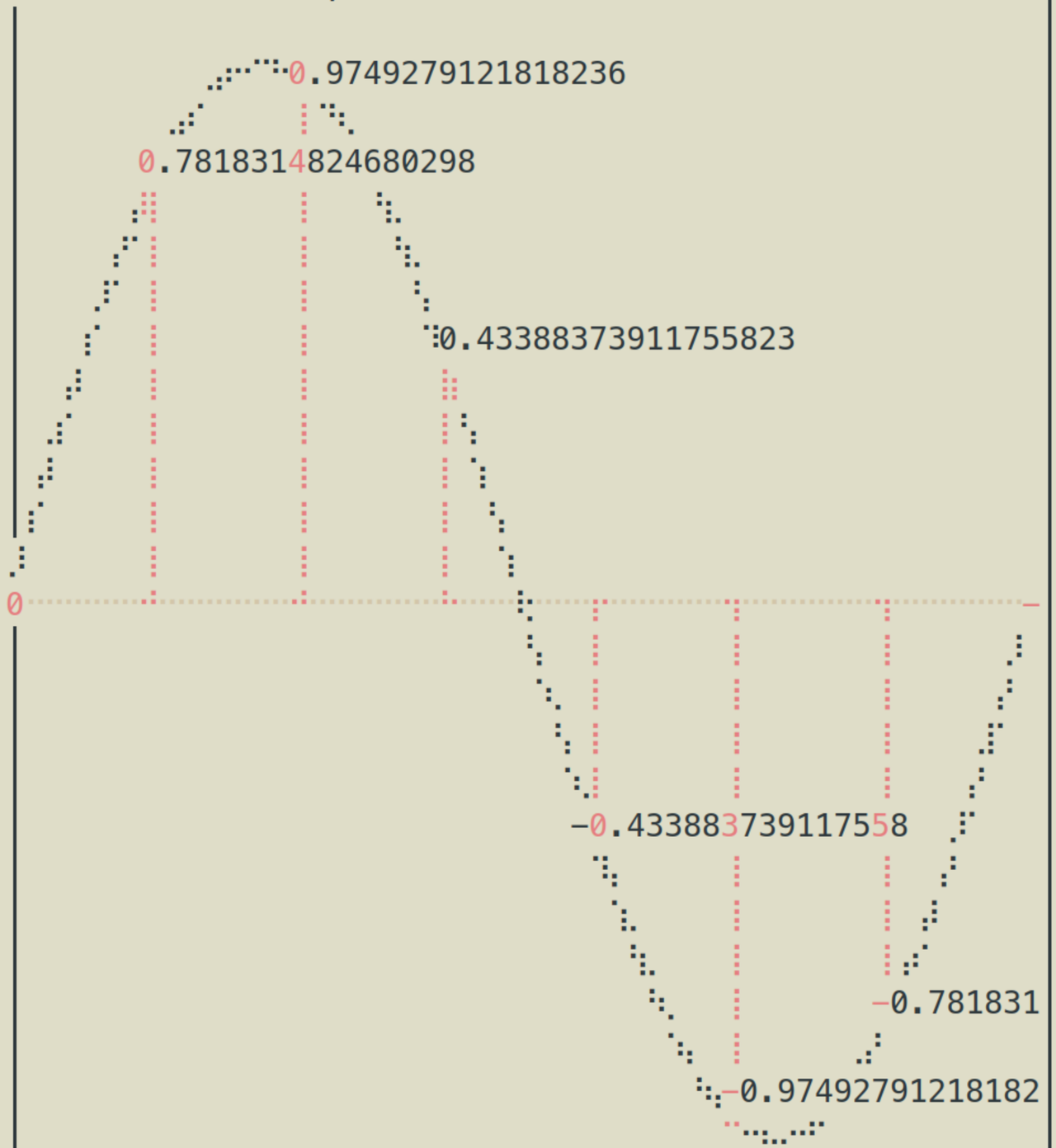
DIGITAL SIGNAL

[↲] • Once we take a sample of a signal at a given time, we need to determine the ***scale of its value***, this is called ***quantization***. Increasing the scale implies reducing the ***quantization noise*** (*quality vs. storage tradeoff*).

- **Audio CD**: 16-bits (65,536 values, 98 dB SNR)
- **Pro Audio**: 24-bits (~16,7 mil., 146 dB SNR)
- ***DSP***: 32/64-bits float (~4,3 bil., 194 dB SNR)

• For **DSP**, it is easier to make computations in ***floating-point*** (decimal), and *normalize* the signal between *-1.0* and *1.0*.

• Finally, sending a digital signal to audio speakers involves the inverse process of an **ADC**: *Digital-to-Analog Conversion* (**DAC**).



ENTER FAUST

[↲] **Faust** (*Functional Audio Stream*) is a programming language specifically made for ***audio DSP and synthesis***. It was created by **Yann Orlarey, Dominique Fober** & **Stéphane Letz** at **GRAME** in 2002.

+ *Functional* paradigm
+ Declarative, math-like syntax
+ Produces optimized code for many architectures
- Not recommended (yet) for multi-rate (FFT)

- **Plugins:** VST, CLAP, AudioUnit
- **Software:** Max, Pd, SuperCollider, Csound
- **OS:** Linux, macOS, Windows, Android, iOS
- **Embedded:** Bela, Teensy, Daisy, ESP32, FPGA
- **Code:** C/C++, Rust, WASM, Java...

[↲] Tutorials & documentation available at:
https://faust.grame.fr

[↲] Dedicated online IDE:
https://faustide.grame.fr

@pchdev – **Emeraude** – *Inria*, *INSA-Lyon*, *CITI Lab*

editor control graph wave spectrum

faust

```
import("stdfaust.lib");

process = os.oscrc(440) * 0.25;
```

Normal

BASICS

[↲] Basic program: ***import*** and ***process*** statements.

[↲] Use a ***Sinewave Oscillator*** from the library.

[↲] In DSP, we represent an audio signal with ***floating-point*** (decimal) numbers and *scale* its values ***between -1.0 and +1.0***.

[↲] ***Mixing*** two signals can be done using the '+' operator.

[↲] When multiplying two signals together, we can already make a simple DSP effect: ***Amplitude Modulation*** (or ***Ring Modulation***).

[↲] Faust can process **multiple channels**, we can use the ',' symbol (***Parallel Operator***) to put signals *in parallel*.

[↲] We can declare **custom variables** (***functions***).

editor control graph wave spectrum

faust

```
// Import statement:
// we import all of the faust libraries.
import("stdfaust.lib");

// 'process' statement:
// what your program will be processing/running:
process = 0;

// /!\ IMPORTANT:
// All the code expressions in Faust
// end with a semicolon ';'
```

Normal

BASICS

[↲] Basic program: ***import*** and ***process*** statements.

[↲] Use a ***Sinewave Oscillator*** from the library.

[↲] In DSP, we represent an audio signal with ***floating-point*** (decimal) numbers and *scale* its values ***between -1.0 and +1.0***.

[↲] ***Mixing*** two signals can be done using the '+' operator.

[↲] When multiplying two signals together, we can already make a simple DSP effect: ***Amplitude Modulation*** (or ***Ring Modulation***).

[↲] Faust can process **multiple channels**, we can use the ',' symbol (***Parallel Operator***) to put signals *in parallel*.

[↲] We can declare **custom variables** (***functions***).

editor control graph wave spectrum

faust

```
// Import statement:
// we import all of the faust libraries.
import("stdfaust.lib");

// Here, we use the 'oscrc' function
// from the 'os' (oscillators) library:
process = os.oscrc(440) * 0.25;
```

Normal

BASICS

[↲] Basic program: ***import*** and ***process*** statements.

[↲] Use a ***Sinewave Oscillator*** from the library.

[↲] In DSP, we represent an audio signal with ***floating-point*** (decimal) numbers and *scale* its values ***between -1.0 and +1.0***.

[↲] ***Mixing*** two signals can be done using the '+' operator.

[↲] When multiplying two signals together, we can already make a simple DSP effect: ***Amplitude Modulation*** (or ***Ring Modulation***).

[↲] Faust can process **multiple channels**, we can use the ',' symbol (***Parallel Operator***) to put signals *in parallel*.

[↲] We can declare **custom variables** (***functions***).

editor control graph wave spectrum

faust

```
import("stdfaust.lib");

// We run the oscillators at 440Hz (A440),
// we multiply its amplitude by 0.25,
// which means we divide its volume by 4:
process = os.oscrc(440) * 0.25;

// Which would be the same as:
process = os.oscrc(440) / 4;
```

Normal

BASICS

[↲] Basic program: ***import*** and ***process*** statements.

[↲] Use a ***Sinewave Oscillator*** from the library.

[↲] In DSP, we represent an audio signal with ***floating-point*** (decimal) numbers and *scale* its values ***between -1.0 and +1.0***.

[↲] ***Mixing*** two signals can be done using the '+' operator.

[↲] When multiplying two signals together, we can already make a simple DSP effect: ***Amplitude Modulation*** (or ***Ring Modulation***).

[↲] Faust can process **multiple channels**, we can use the ',' symbol (***Parallel Operator***) to put signals *in parallel*.

[↲] We can declare **custom variables** (***functions***).

editor control graph wave spectrum

faust

```
import("stdfaust.lib");

// Mixing two signals means adding them together,
// which can be done by using the '+' operator
process = (os.oscrc(440) + os.oscrc(880)) * 0.25;

// Here, we used parentheses because of
// operator precedence.
// Multiplications are always evaluated before
// additions or substractions.
```

Normal

BASICS

[↲] Basic program: ***import*** and ***process*** statements.

[↲] Use a ***Sinewave Oscillator*** from the library.

[↲] In DSP, we represent an audio signal with ***floating-point*** (decimal) numbers and *scale* its values ***between -1.0 and +1.0***.

[↲] ***Mixing*** two signals can be done using the '+' operator.

[↲] When multiplying two signals together, we can already make a simple DSP effect: ***Amplitude Modulation*** (or ***Ring Modulation***).

[↲] Faust can process **multiple channels**, we can use the ',' symbol (***Parallel Operator***) to put signals *in parallel*.

[↲] We can declare **custom variables** (***functions***).

editor control graph wave spectrum

faust

```
import("stdfaust.lib");

// Basic amplitude modulation.
// no need for parentheses here, since we only
// use multiplications:
process = os.oscrc(440) * os.oscrc(10) * 0.25;
```

Normal

BASICS

[↲] Basic program: ***import*** and ***process*** statements.

[↲] Use a ***Sinewave Oscillator*** from the library.

[↲] In DSP, we represent an audio signal with ***floating-point*** (decimal) numbers and *scale* its values ***between -1.0 and +1.0***.

[↲] ***Mixing*** two signals can be done using the '**+**' operator.

[↲] When multiplying two signals together, we can already make a simple DSP effect: ***Amplitude Modulation*** (or ***Ring Modulation***).

[↲] Faust can process **multiple channels**, we can use the '**,**' symbol (***Parallel Operator***) to put signals *in parallel*.

[↲] We can declare **custom variables** (***functions***).

editor control graph wave spectrum

faust

```
import("stdfaust.lib");

// Here, we use the Parallel Operator ','
// to put two signals in parallel:
process = (os.oscrc(440) * 0.25), (os.oscrc(880) *
0.25);
```

Normal

BASICS

[↲] Basic program: ***import*** and ***process*** statements.

[↲] Use a ***Sinewave Oscillator*** from the library.

[↲] In DSP, we represent an audio signal with ***floating-point*** (decimal) numbers and *scale* its values ***between -1.0 and +1.0***.

[↲] ***Mixing*** two signals can be done using the '**+**' operator.

[↲] When multiplying two signals together, we can already make a simple DSP effect: ***Amplitude Modulation*** (or ***Ring Modulation***).

[↲] Faust can process **multiple channels**, we can use the '**,**' symbol (***Parallel Operator***) to put signals *in parallel*.

[↲] We can declare **custom variables** (***functions***).

editor control graph wave spectrum

faust

```
import("stdfaust.lib");

// We can declare our custom functions:
osc440 = os.oscrc(440) * 0.25;
osc880 = os.oscrc(880) * 0.25;

// And call them from anywhere:
process = osc440, osc880;
```

Normal