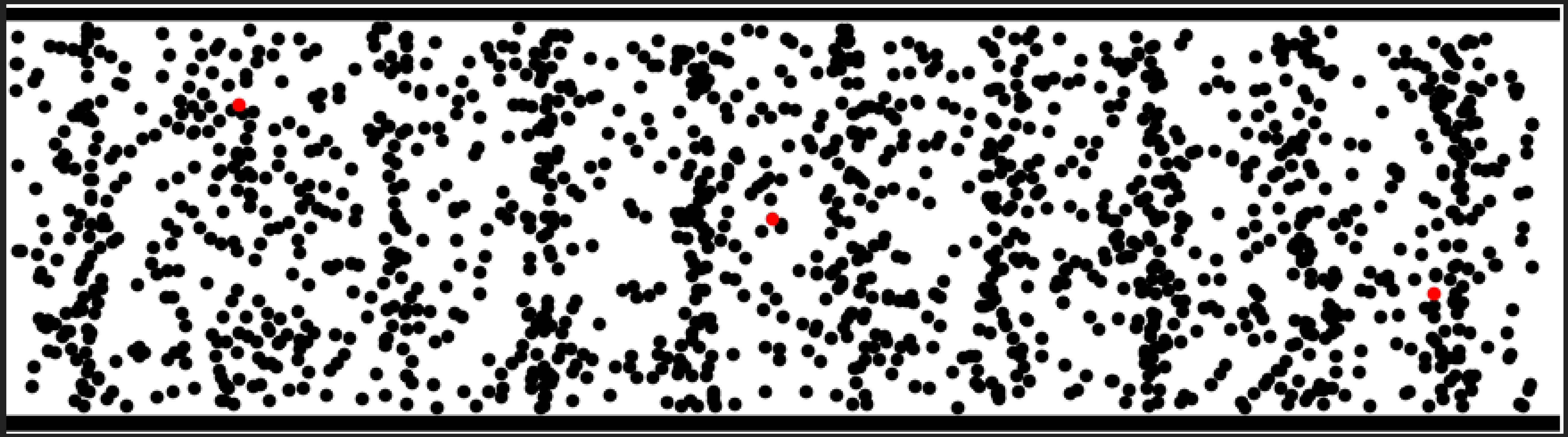
Digital Signal Processing for Music

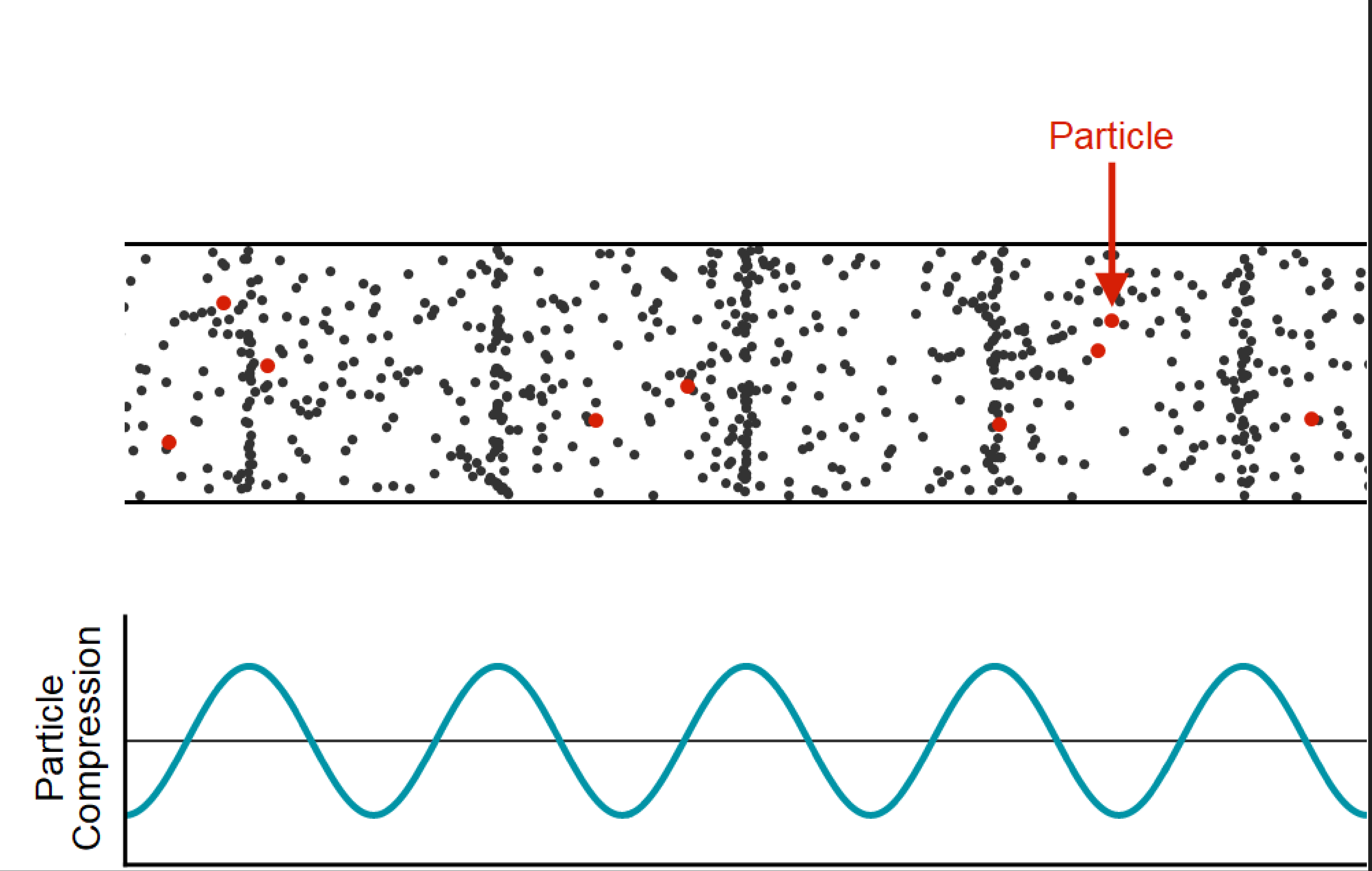
Part 2: Signals

Andrew Beck

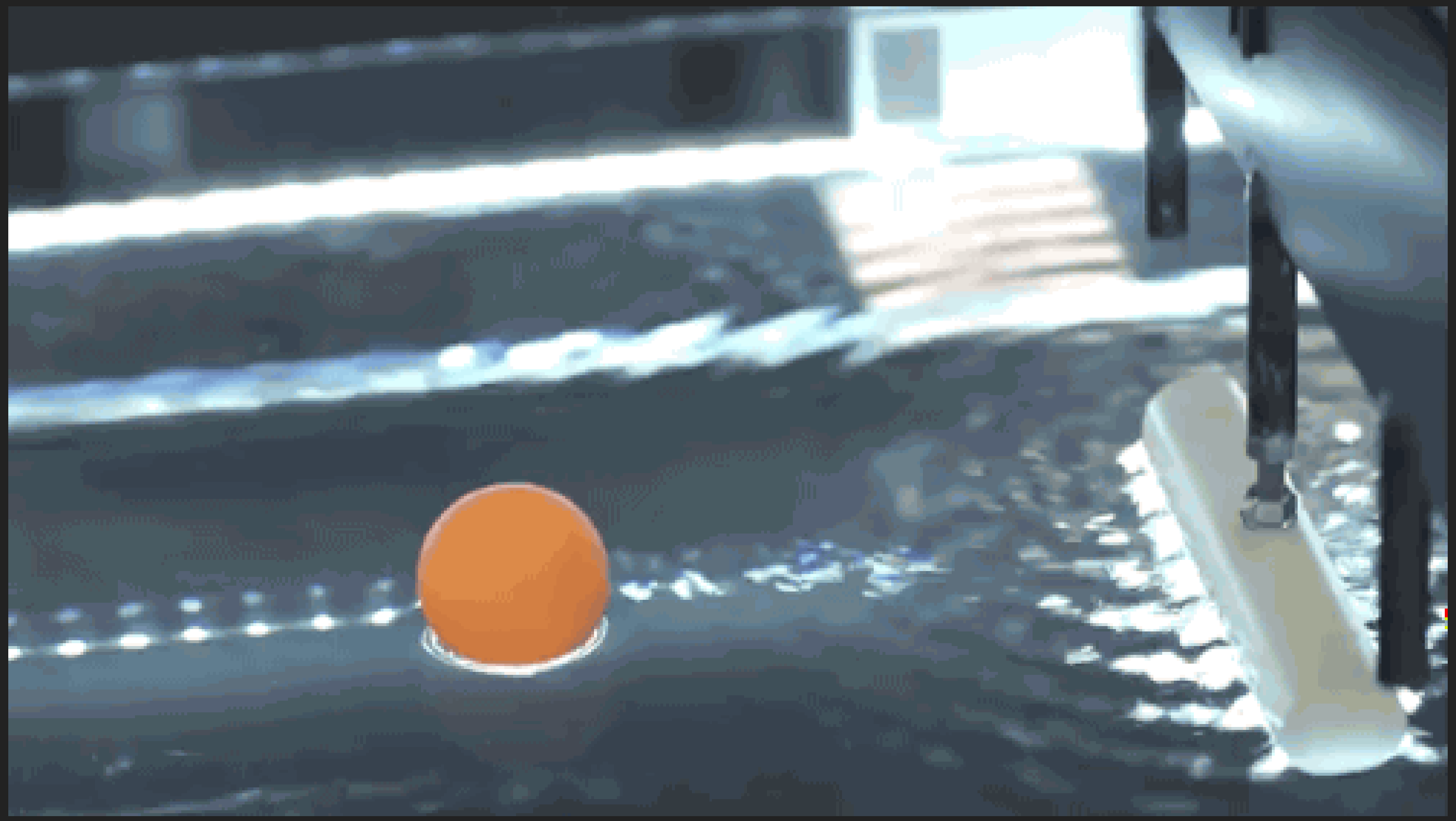
Sound is a vibration **propagating through a medium.**



The *audio signal* is a measure of the compression of the medium at a given point

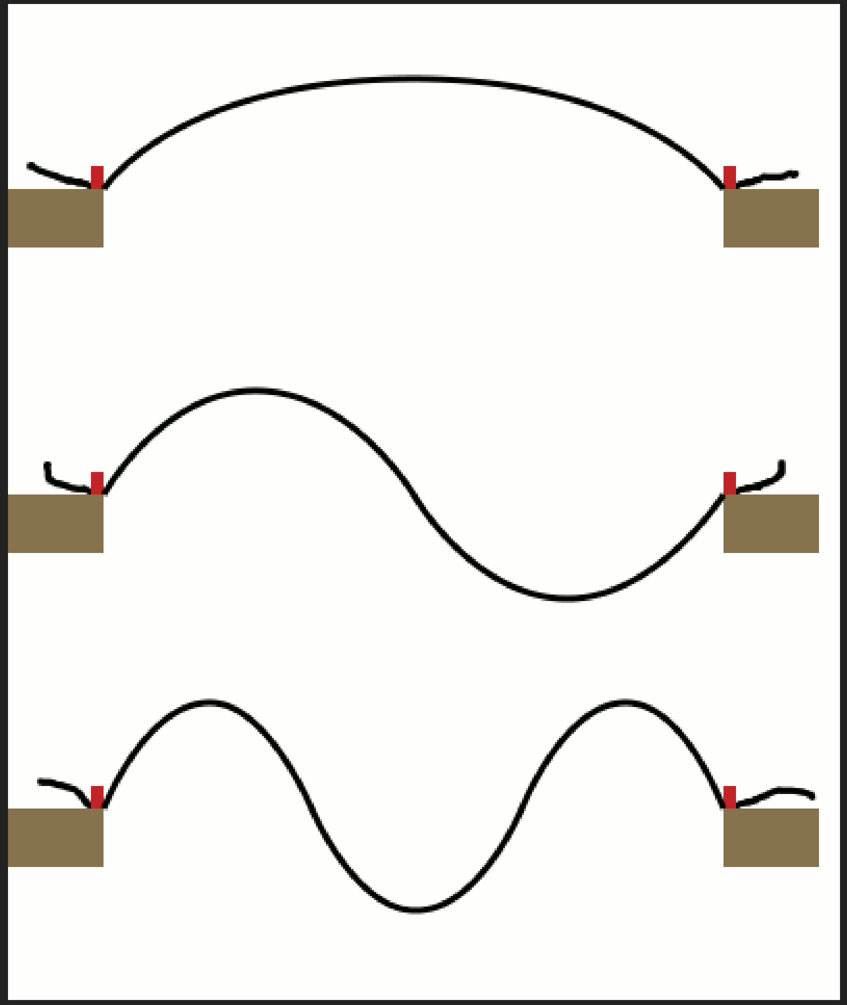


Vibration in medium is caused by an objects motion

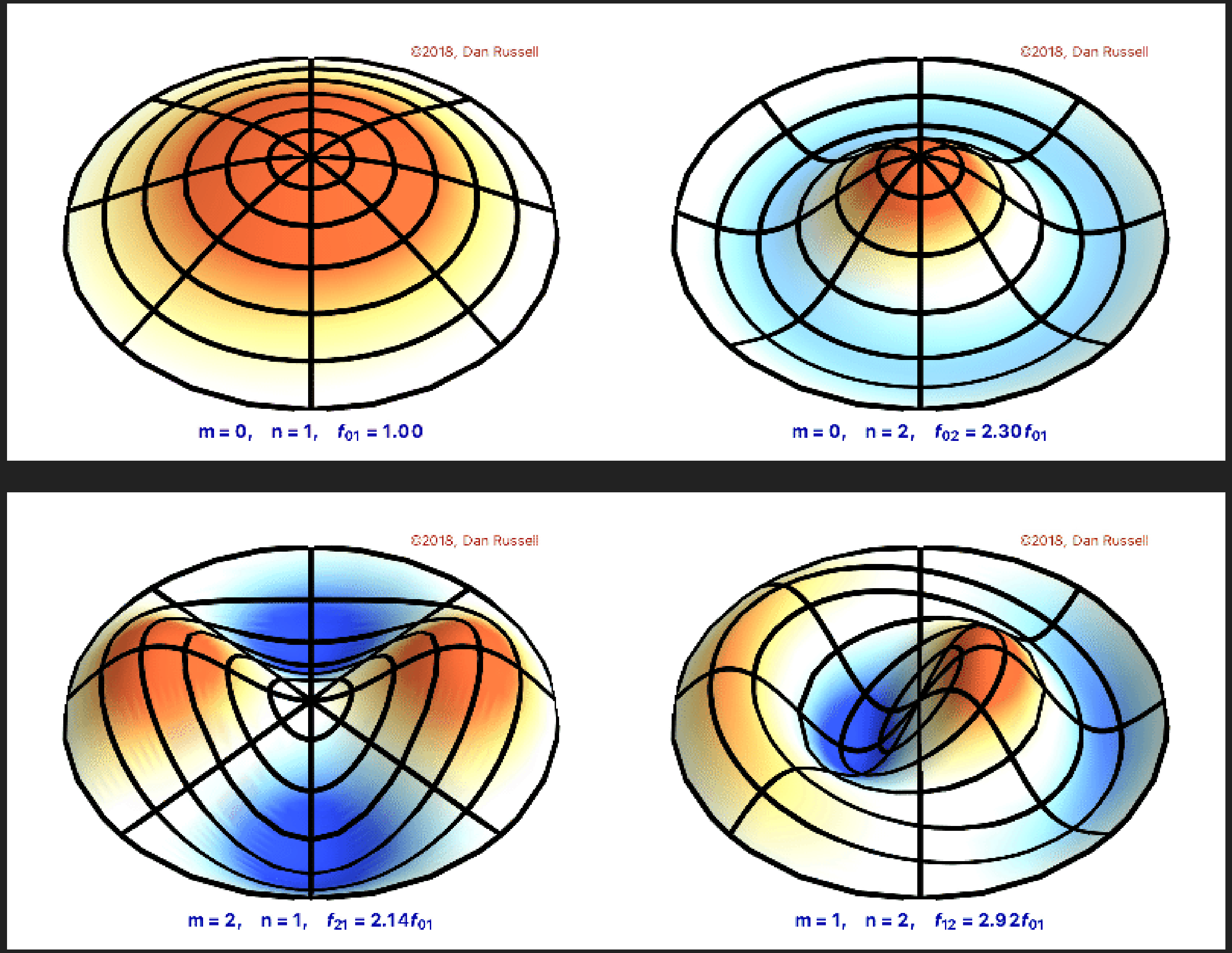


Objects vibrate in many different modes simultanously

As Integer Multiples



Or inharmonically

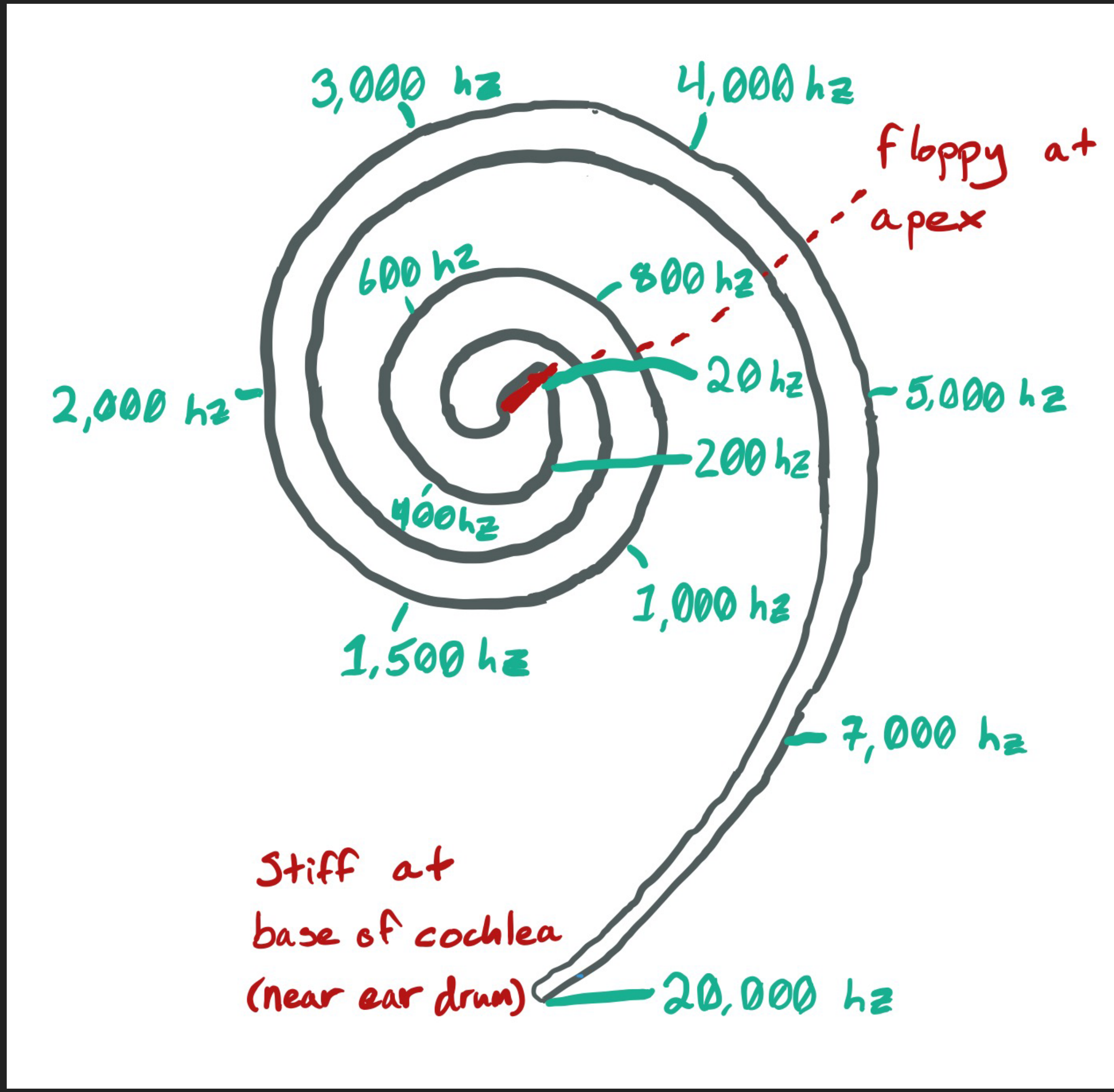


- **Partials**: a set of frequencies comprising a (pitched) sound
- **Overtones**: as partials but without the fundamental frequency
- **Harmonics**: integer multiples of the fundamental frequency, including the fundamental frequency

Physical Properties of Sound Production

- Larger objects produce larger sine waves (lower frequencies)
- The relative strength of various partials indicate different materials

Physical Properties of the Ear



- The cochlea resonates via thickness and stiffness across our hearing spectrum
- In a sense, our inner ear mirrors the way sound resonates in object modes

Deterministic Signals	*Predictable*: future shape of the signal can be known (example: sinusoidal)
Random Signals	*Unpredictable*: no knowledge can help to predict what is coming next (example: white noise)

Every "real-world" audio signal can be modeled as time-varying combination of

- (Quasi-)periodic parts
- (Quasi-)random parts

Properties of Real-World Signals

- Real-Valued
- Finite Energy
- Finite Bandwidth
  (aka smooth)

Amplitude:

$$max|x(t)| < \infty$$

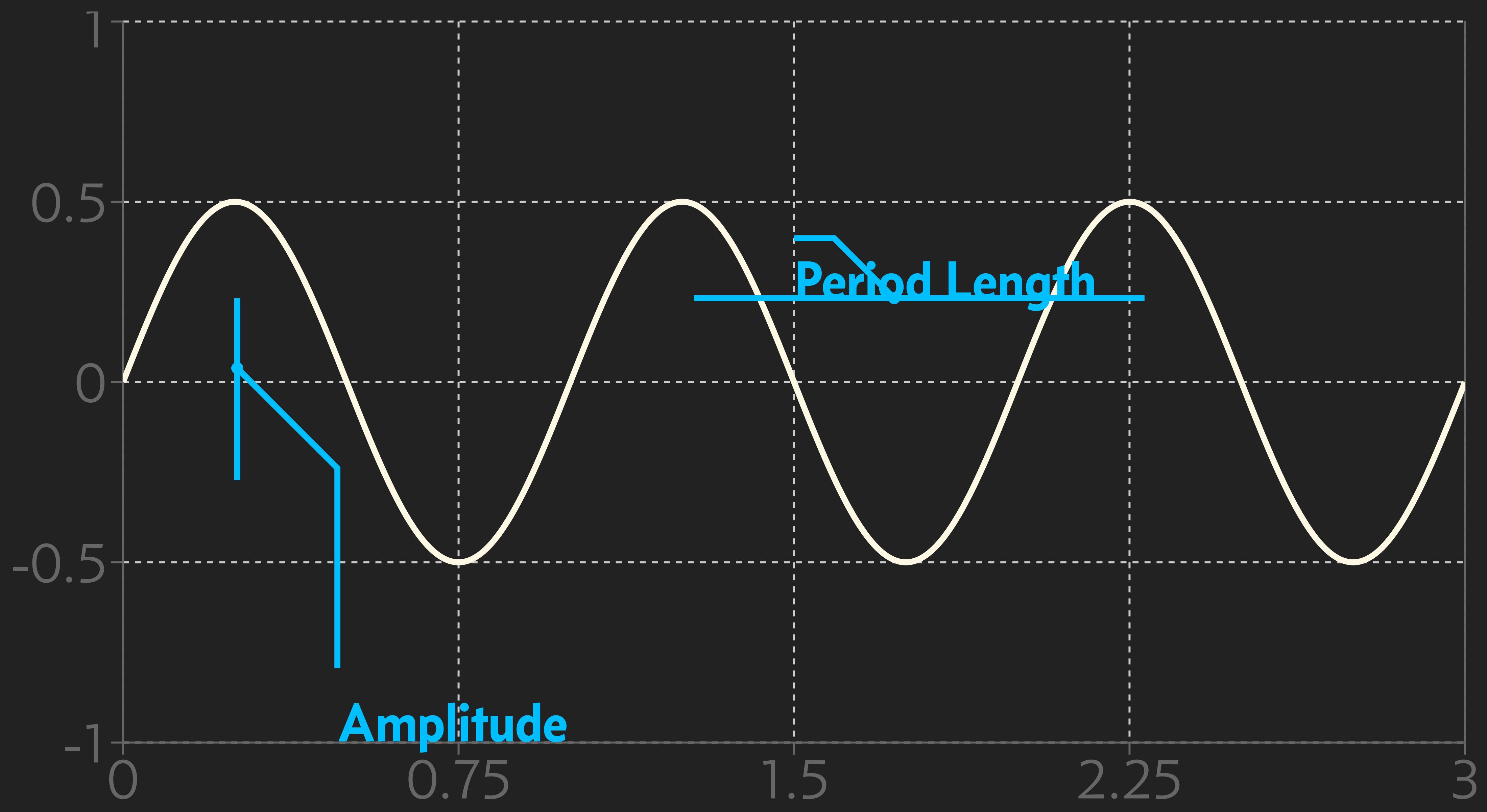
Energy:

$$E = \int_{-\infty}^{\infty} x^2(t)dt$$

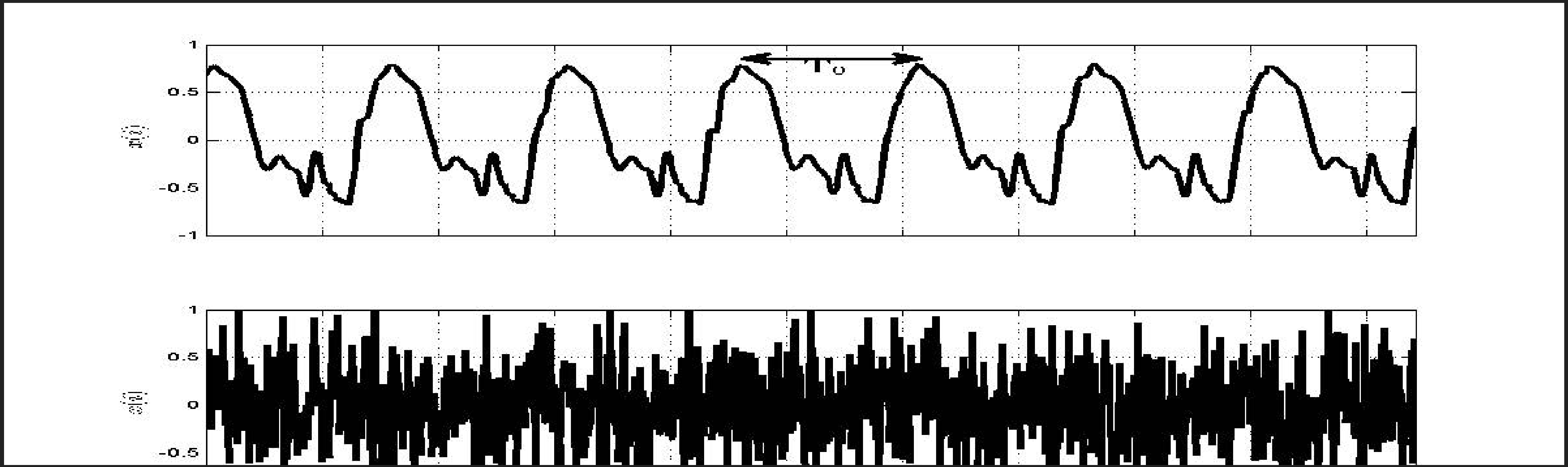
$$P = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} x^2(t)dt$$

Periodic Signals

$$x(t) = x(t + T_0) \qquad f_0 = \frac{1}{T_0} \qquad \omega_0 = \frac{2\pi}{T_0}$$



Real-World Example of Periodicity



Reconstruction

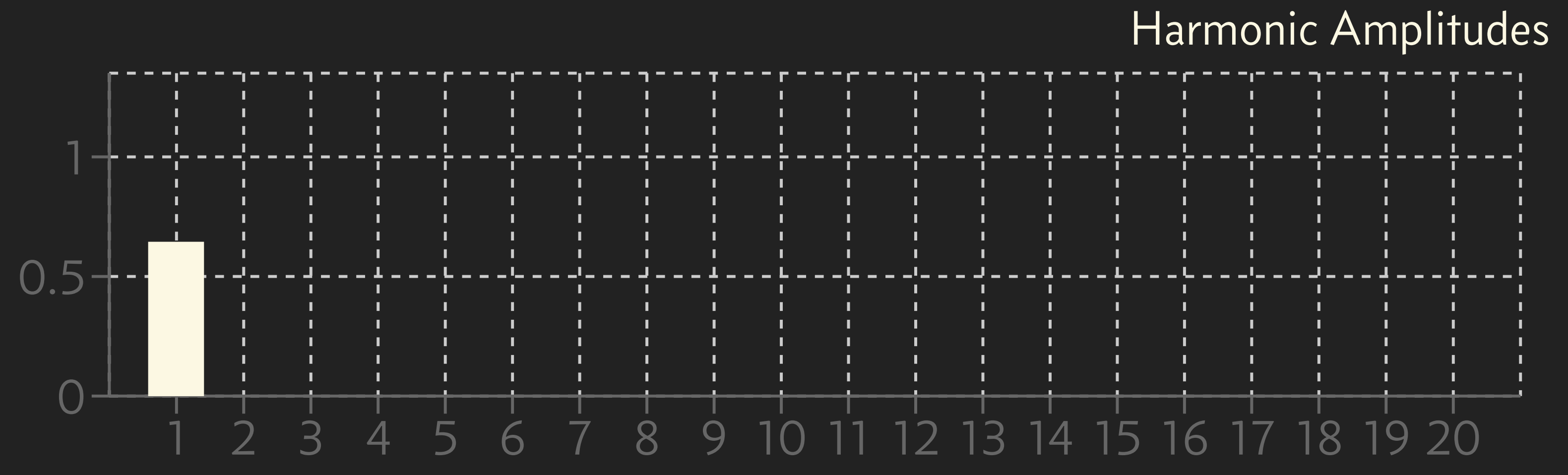
Periodic Signals can be reconstructed through a sum of sinusoidals at frequencies $k \cdot \omega$

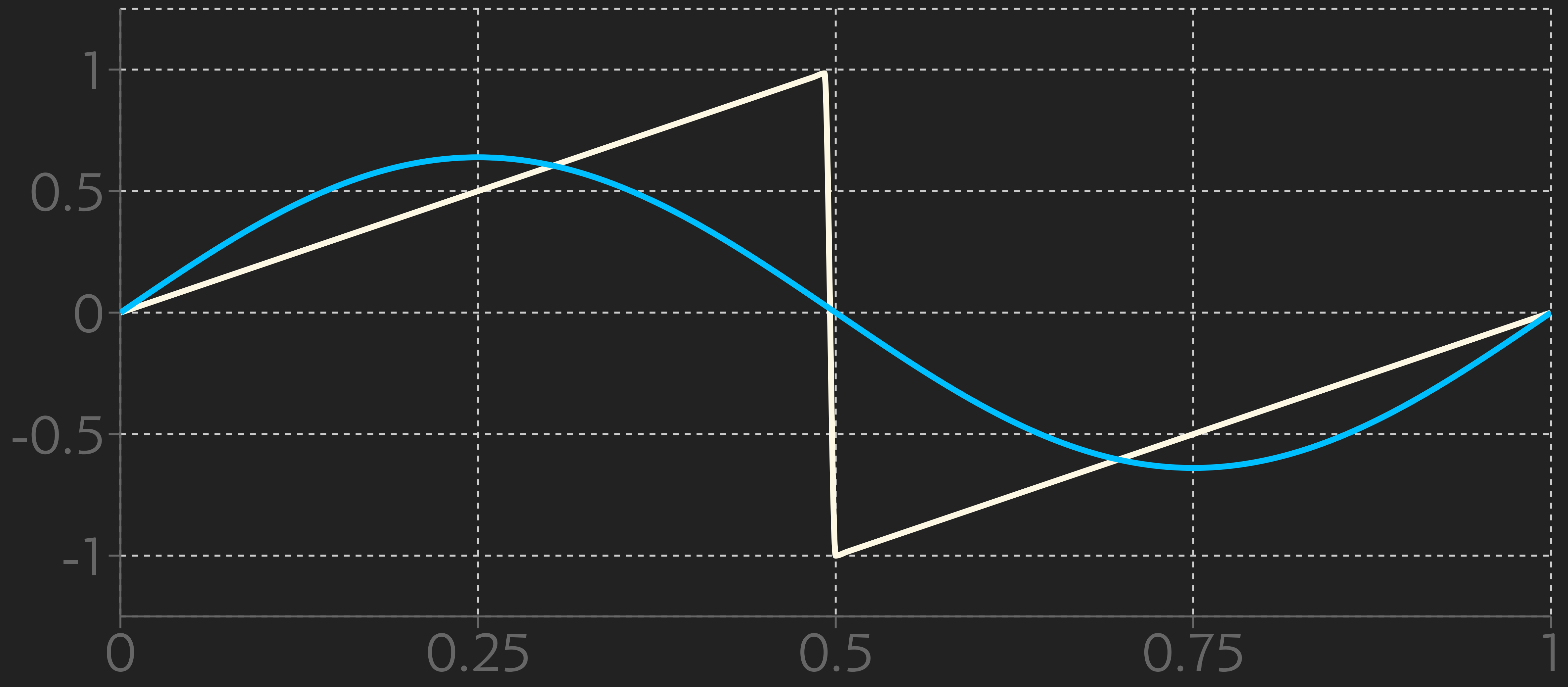
$$\hat{x}(t) = a_1 \cdot sin(\omega_0 t) + a_2 \cdot sin(2 \cdot \omega_0 t) + \ldots + a_3 \cdot sin(n \cdot \omega_0 t)$$

Sawtooth Wave

Num Harmonics

← 1 →

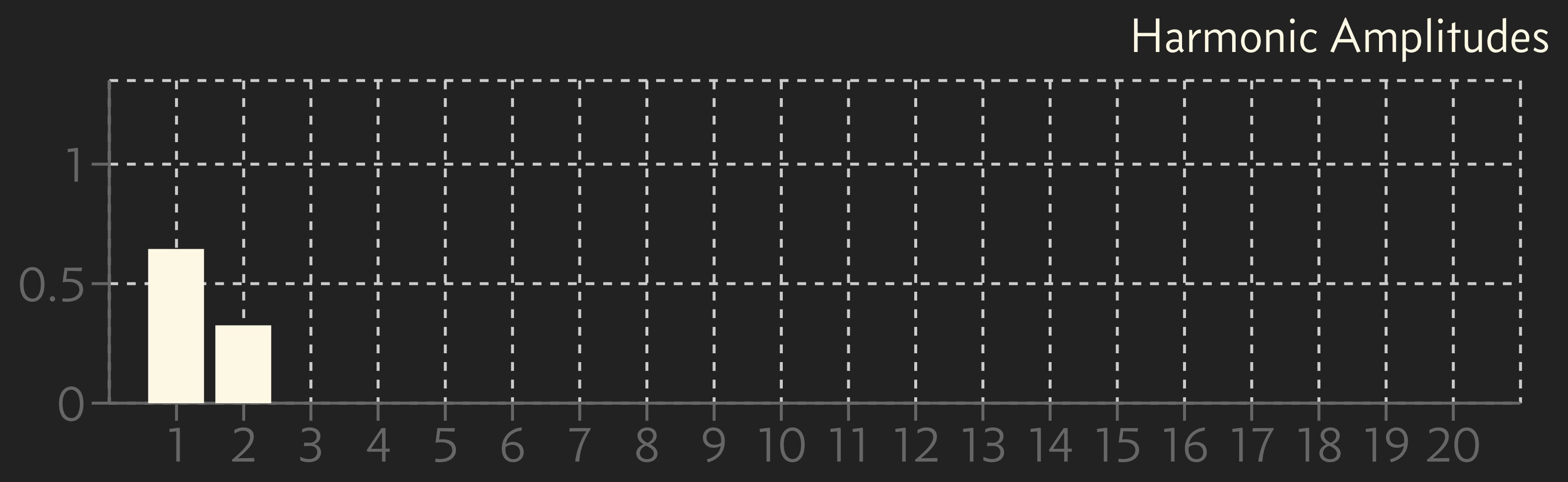


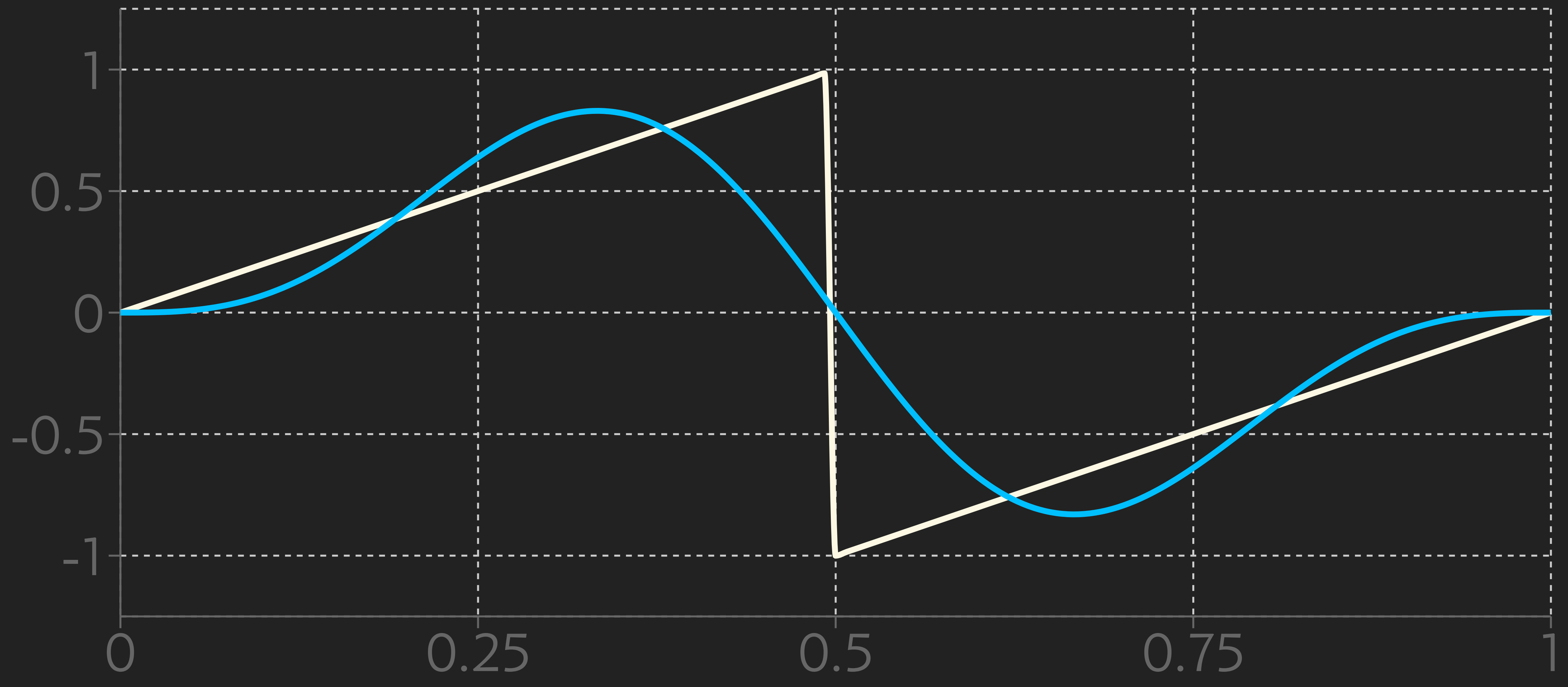


Sawtooth Wave

Num Harmonics

2

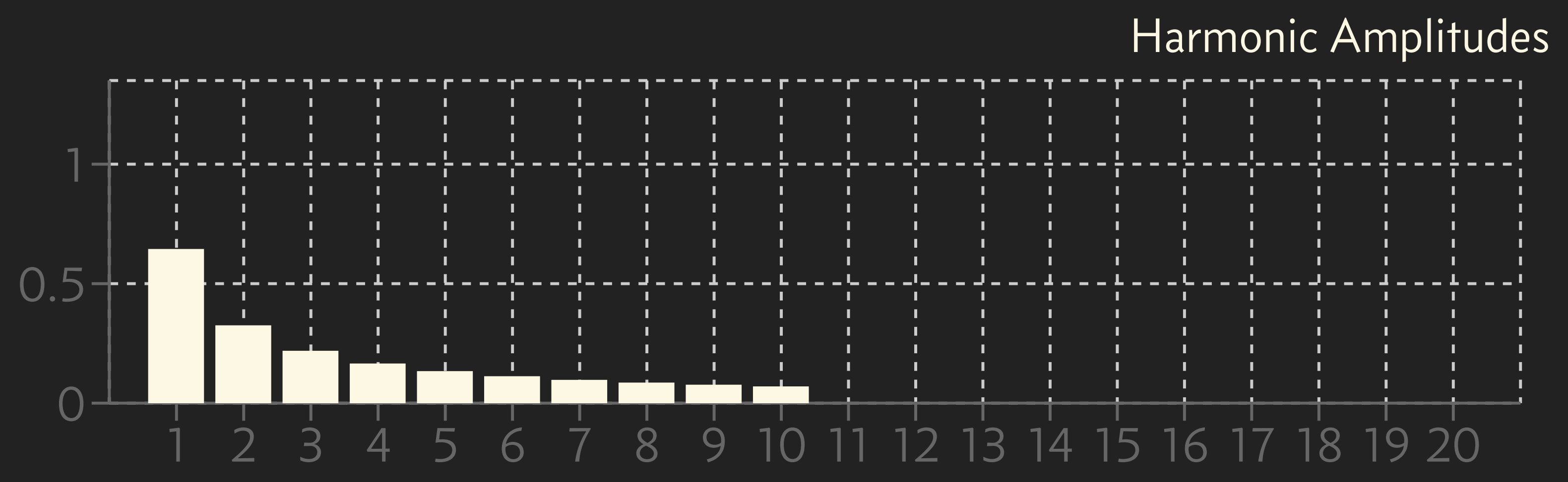


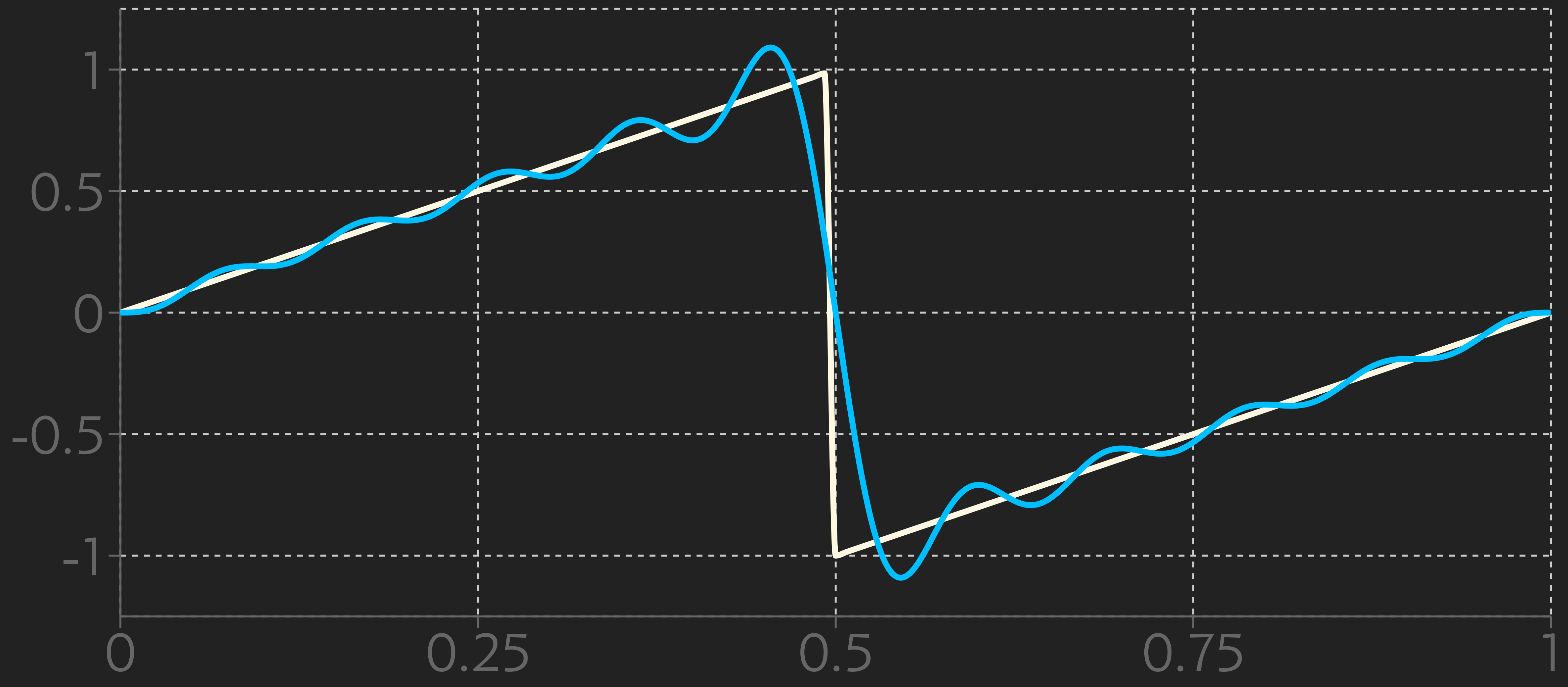


Sawtooth Wave

Num Harmonics

10

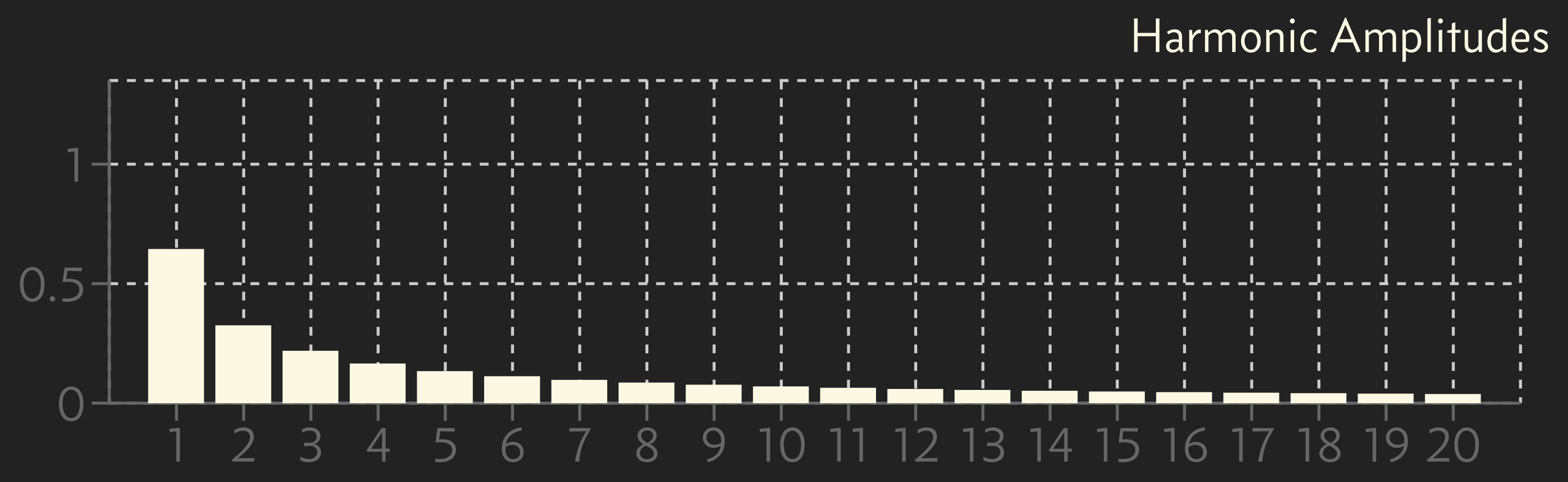


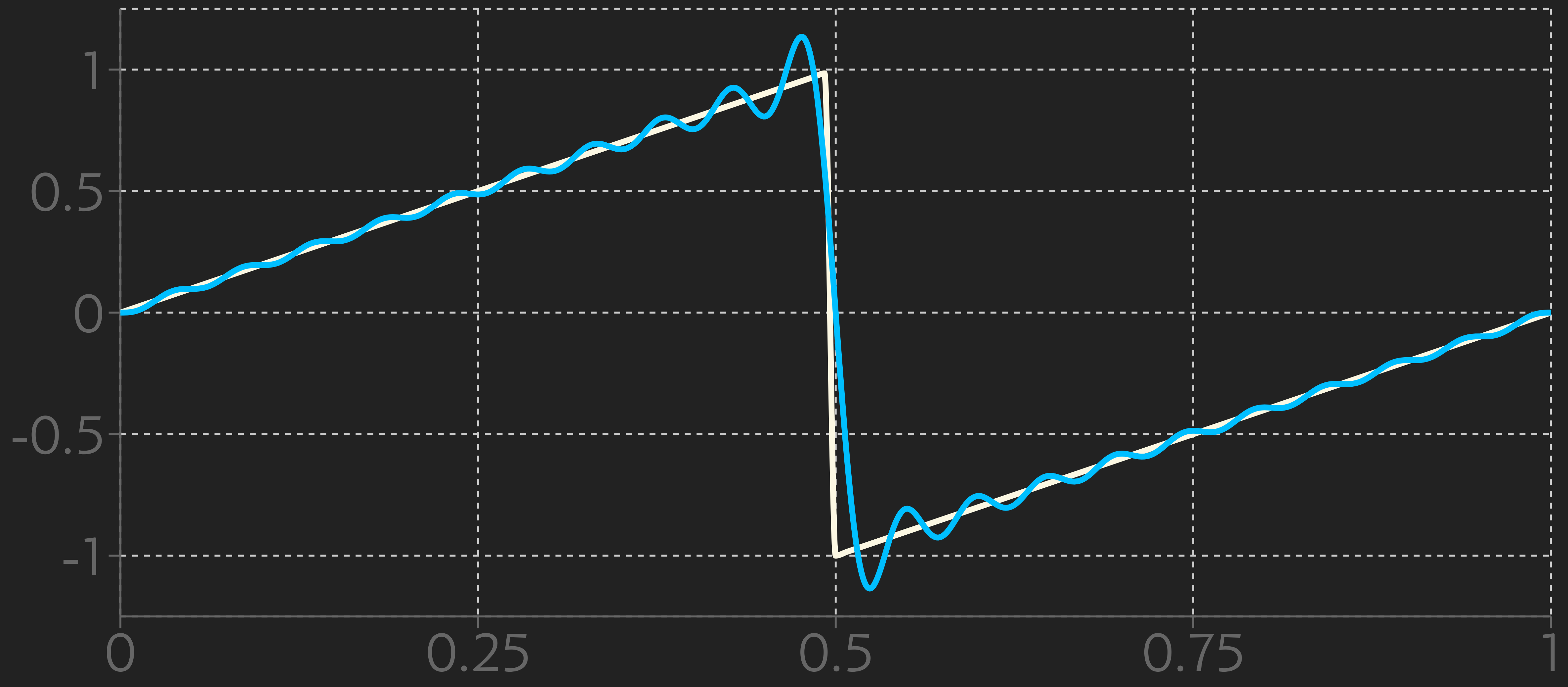


Sawtooth Wave

Num Harmonics

← 20 →

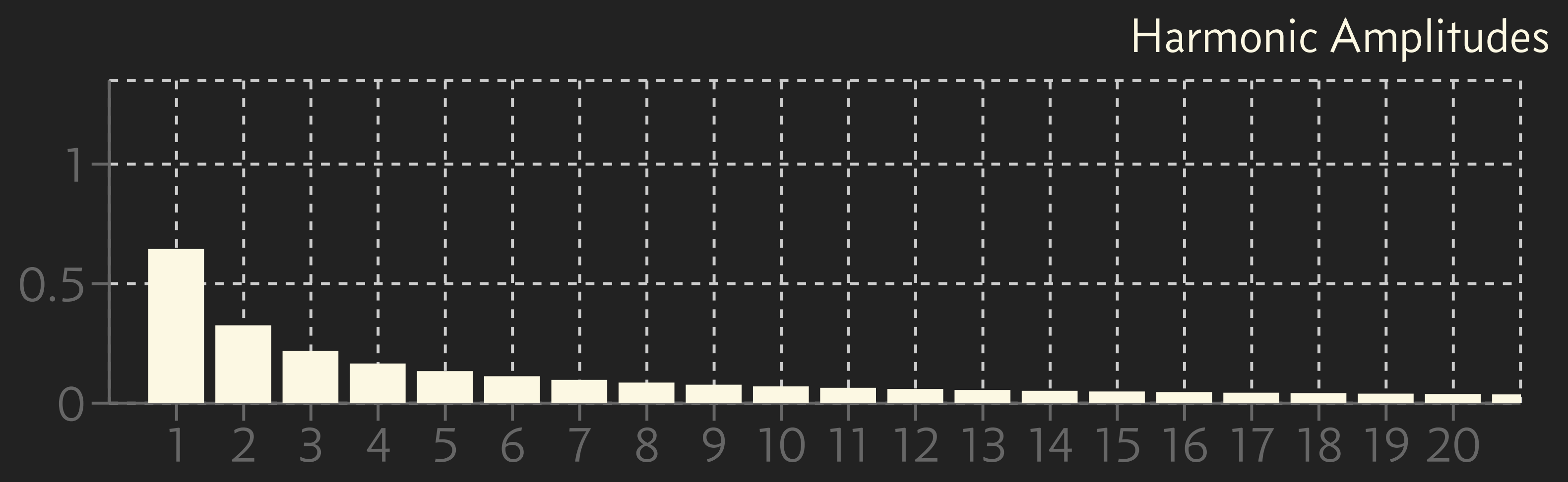


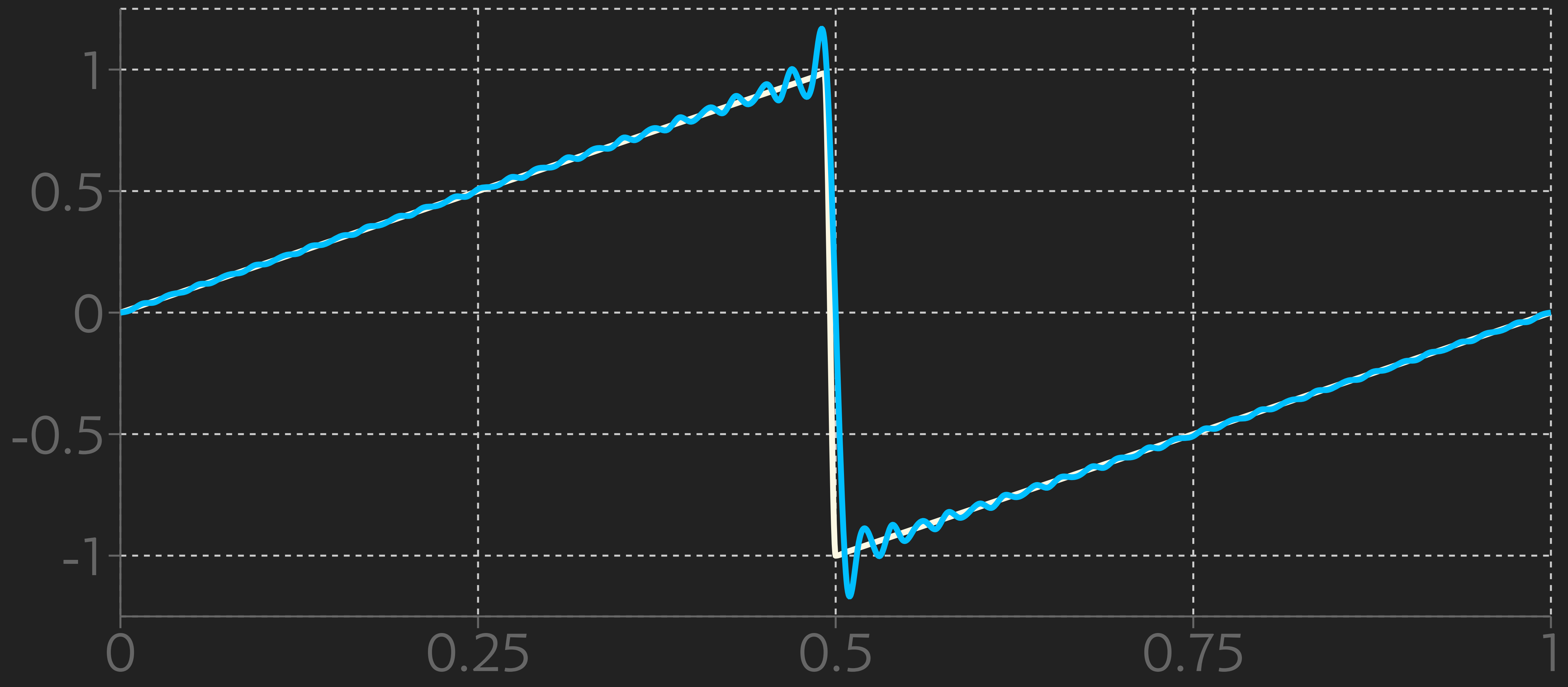


Sawtooth Wave

Num Harmonics

50

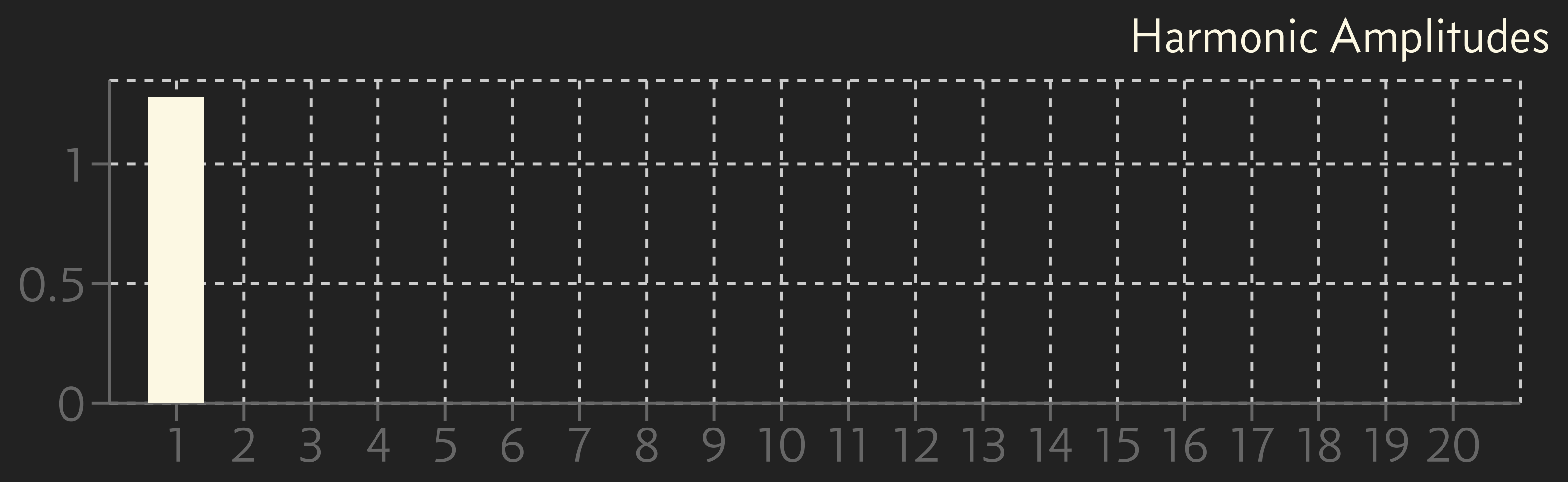


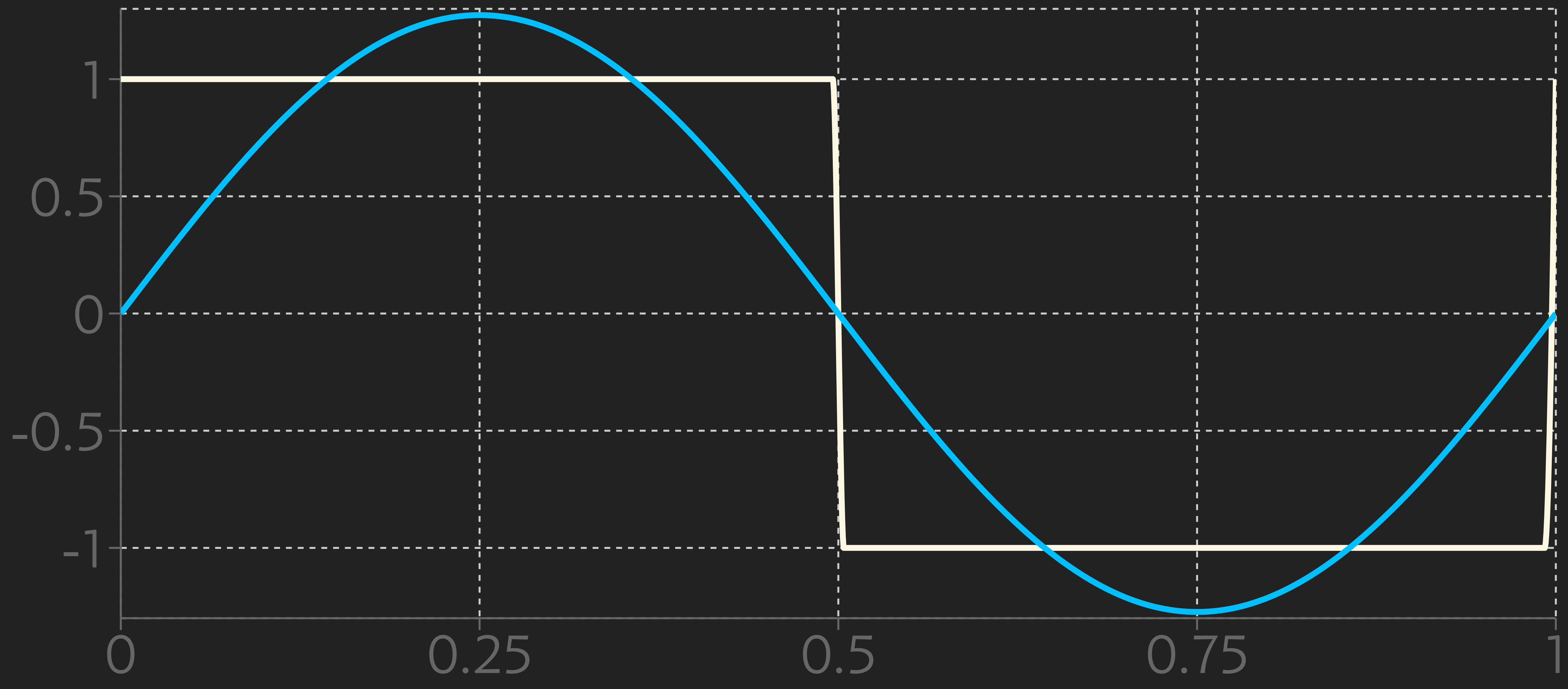


Square Wave

Num Harmonics

← 1 →

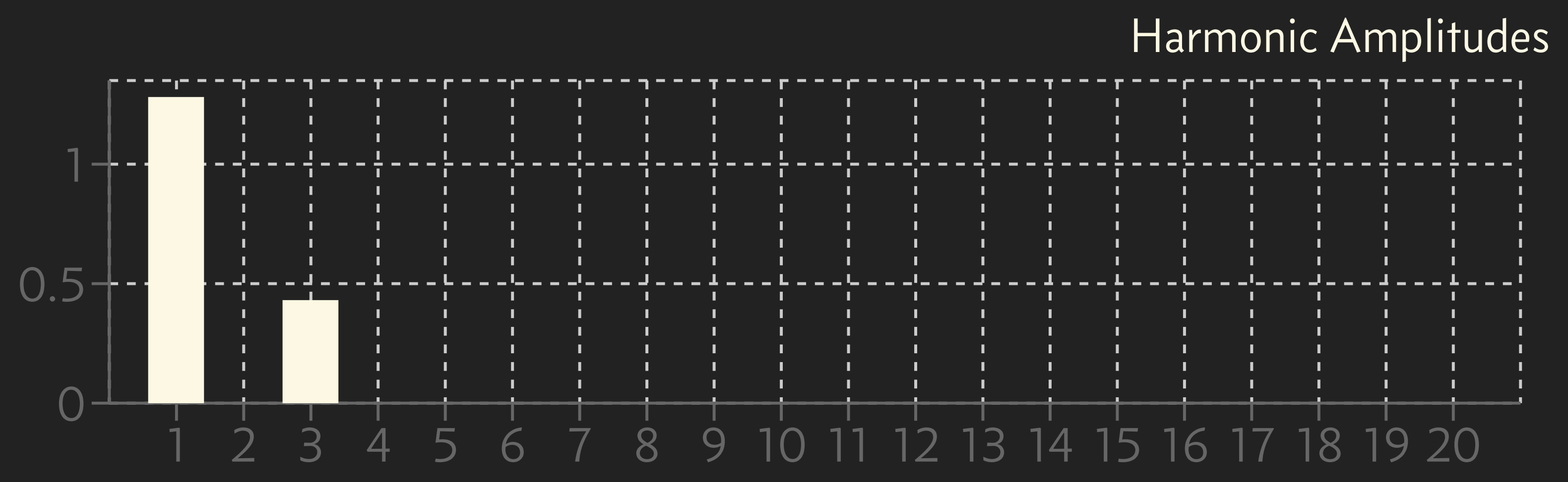


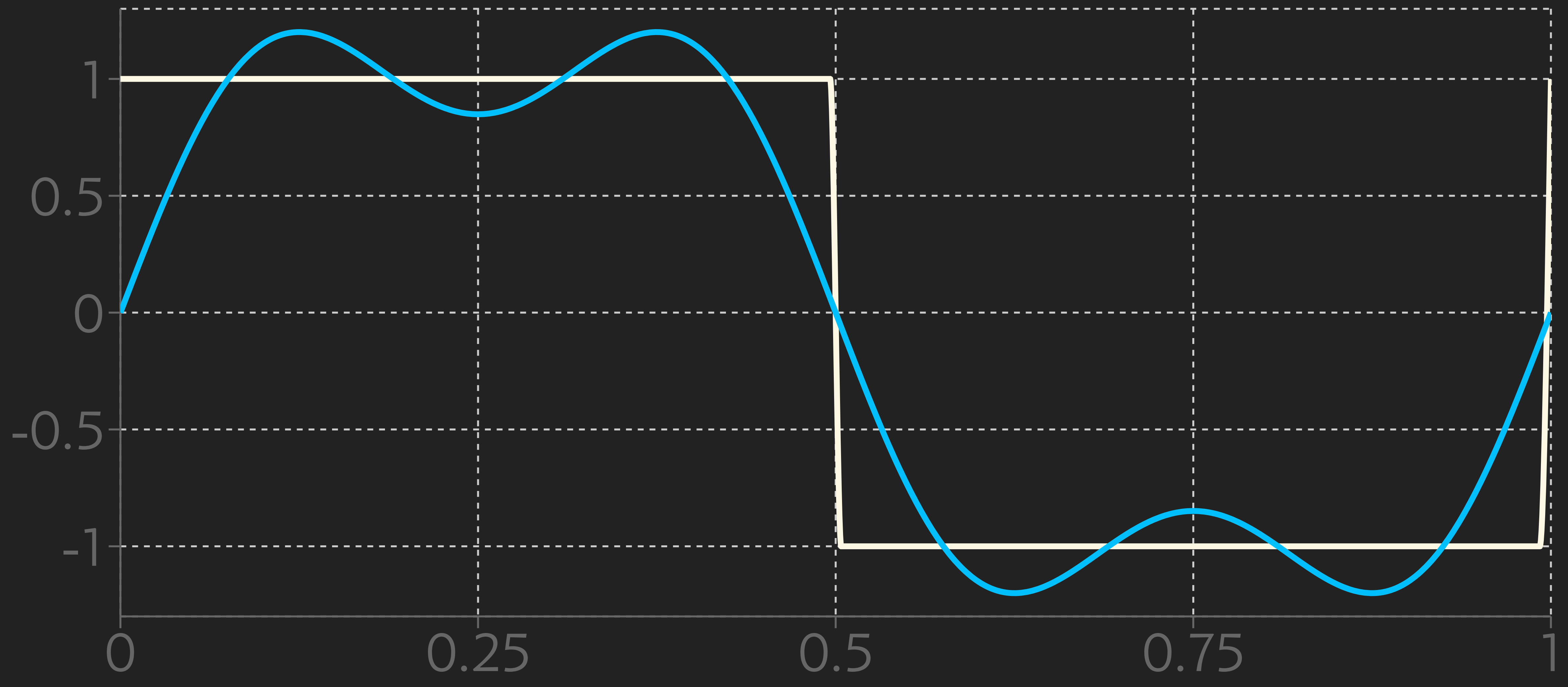


Square Wave

Num Harmonics

← 3 →

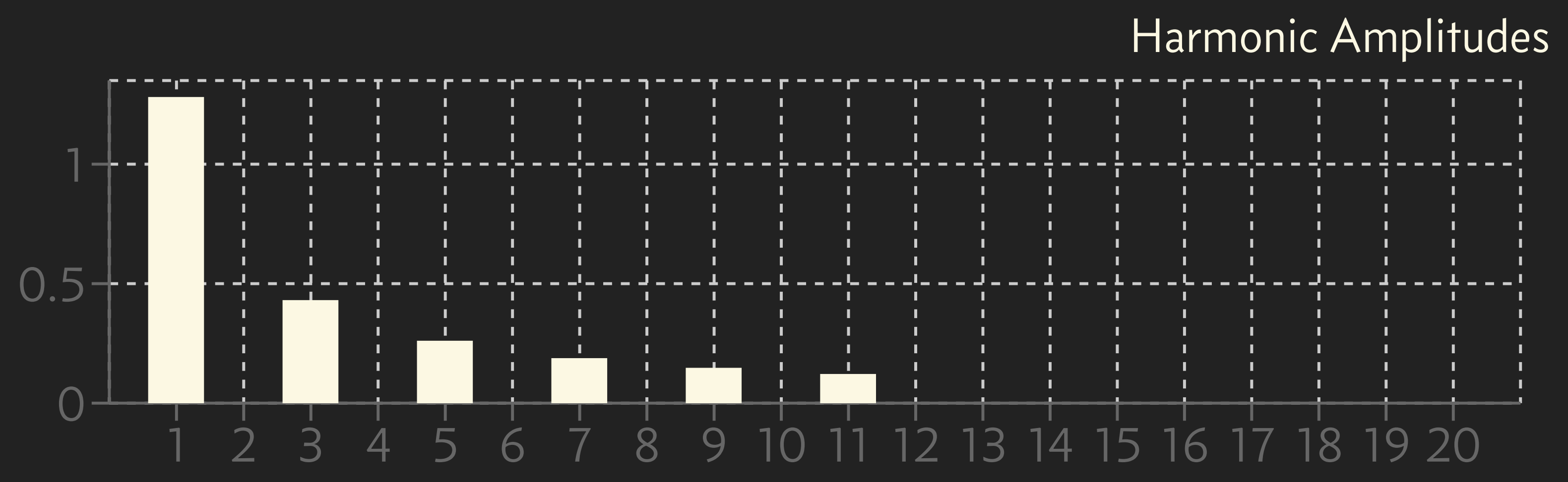




Square Wave

Num Harmonics

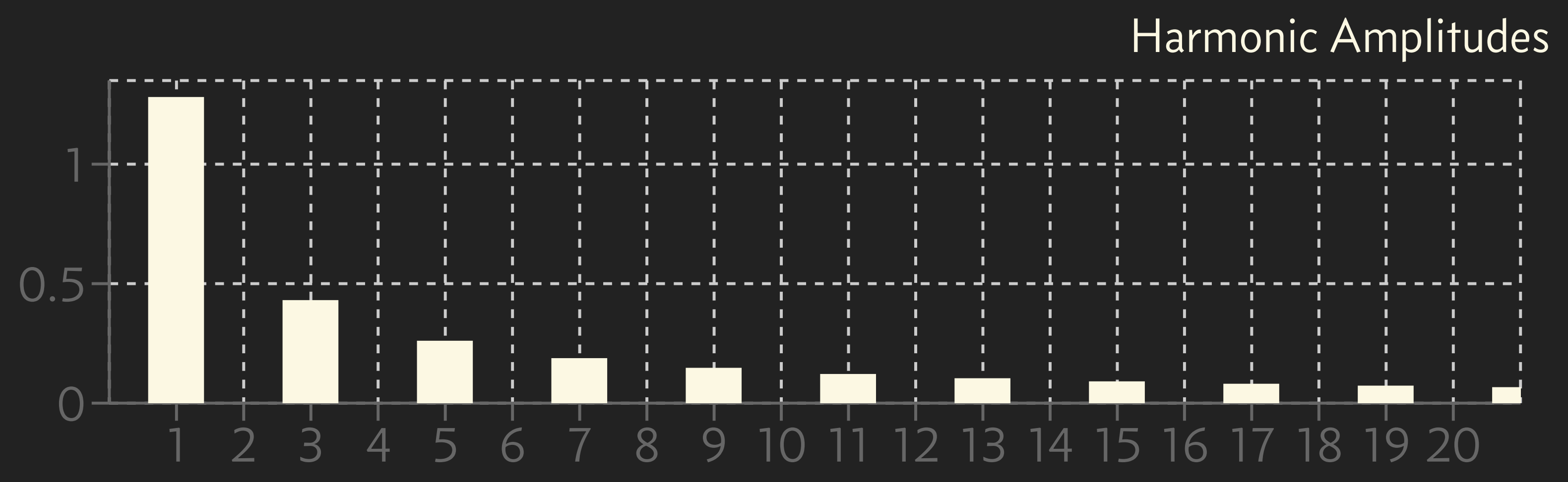
11

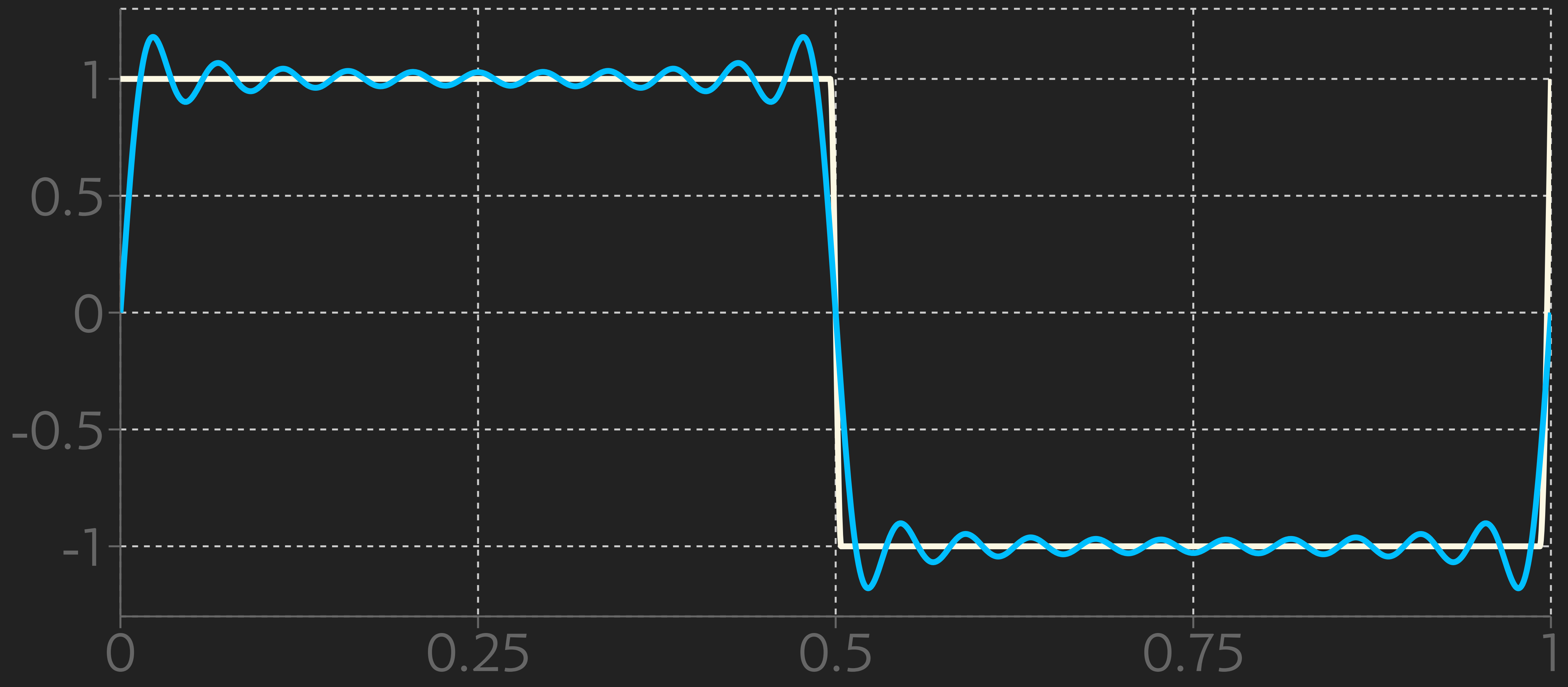


Square Wave

Num Harmonics

21

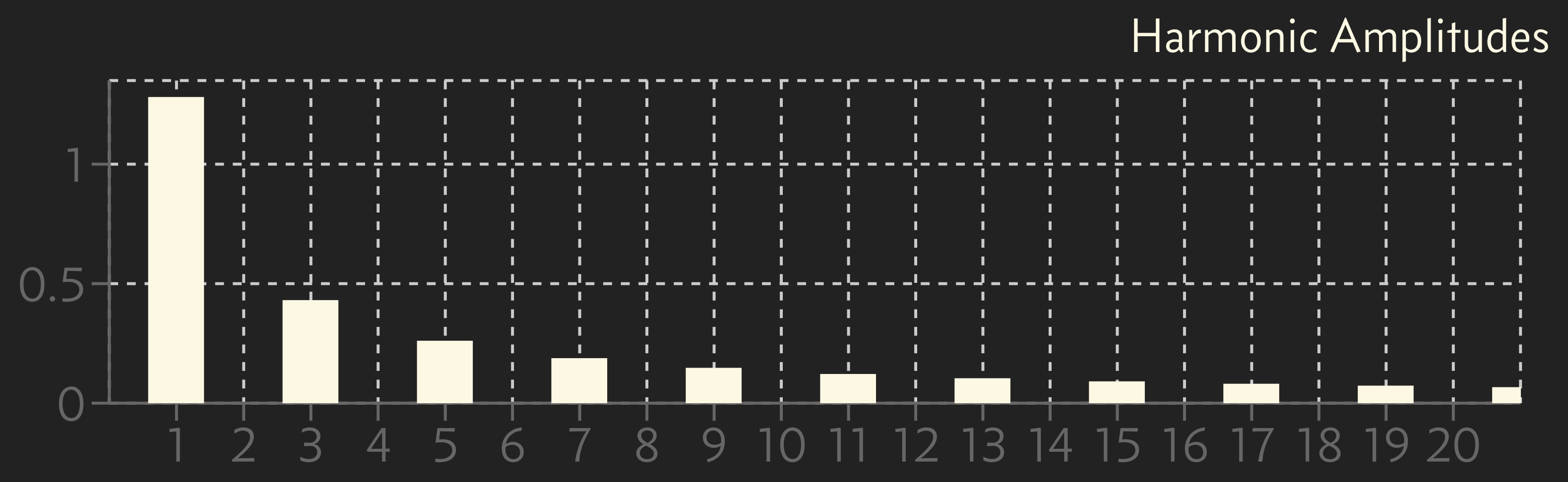


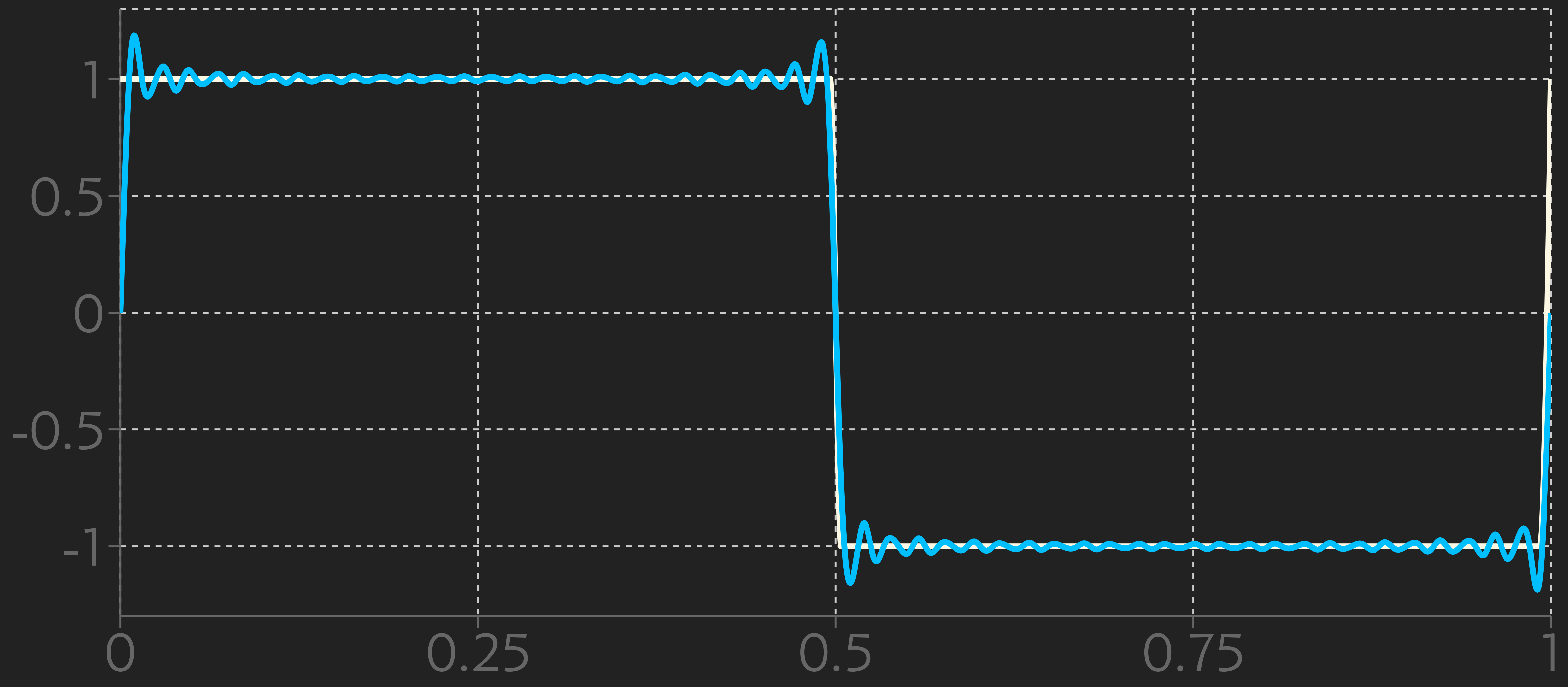


Square Wave

Num Harmonics

← 51 →





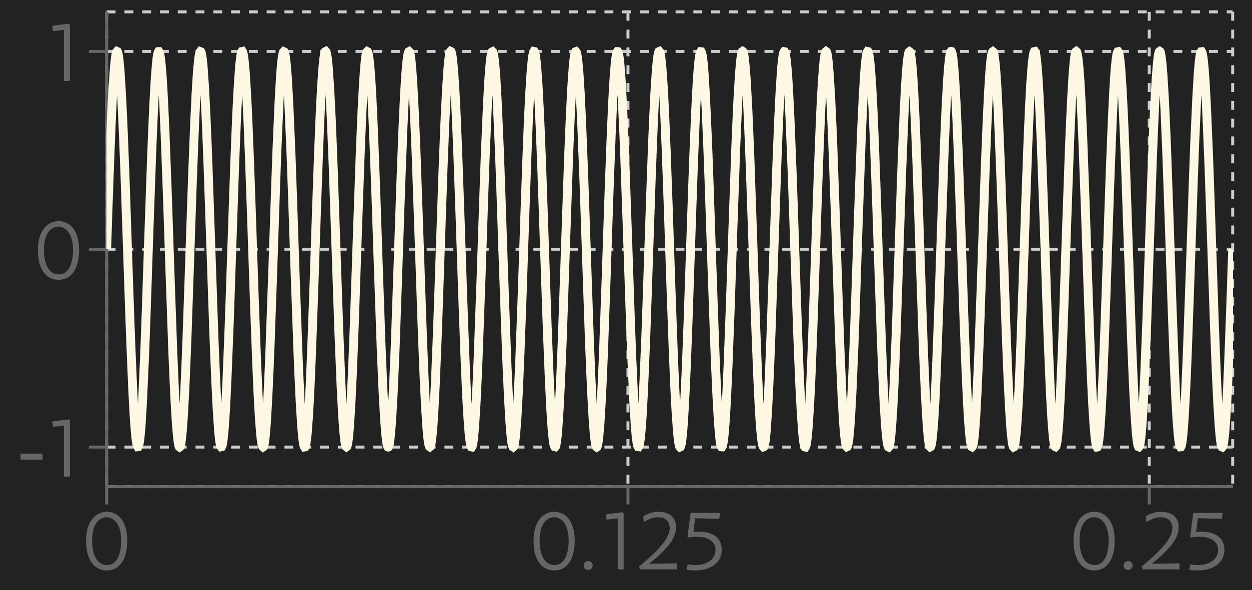
Square wave additive synthesis, try at https://intonal.io/

```
main = {sr: float32 in
  numHarmonics = 25
  blSquare = makeBlAdditiveSquareWave(numHarmonics)
  out = blSquare(440, sr) * 0.25
}

phasor = {hz: float32, sr: float32 in
  out = 0 fby ((prev + (hz/sr)) % 1)
}

PI = 3.14159265358

makeBlAdditiveSquareWave = {numHarmonics: uint64 in
  out = {hz: float32, sr: float32 in
    curHarmonic: float32 = 1 fby prev + 1
    harmonics = render(2 * curHarmonic - 1, numHarmonics) on init

    out = harmonics.multiReduce(0, {prev, harmonic in
      p = phasor(hz * harmonic, sr)
      amp = 4 / (harmonic * PI)
      out = (sin(p * 2 * PI) * amp) + prev
    })
  }
}
```

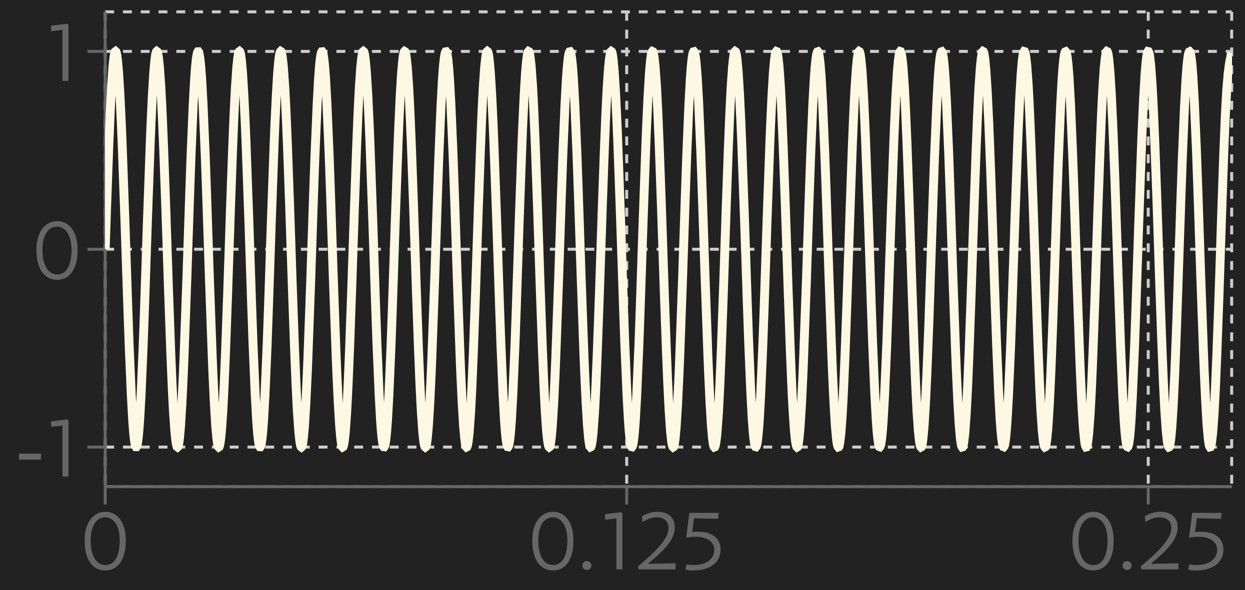
Mechanical Additive Synthesis

https://youtu.be/8KmVDxkia_w

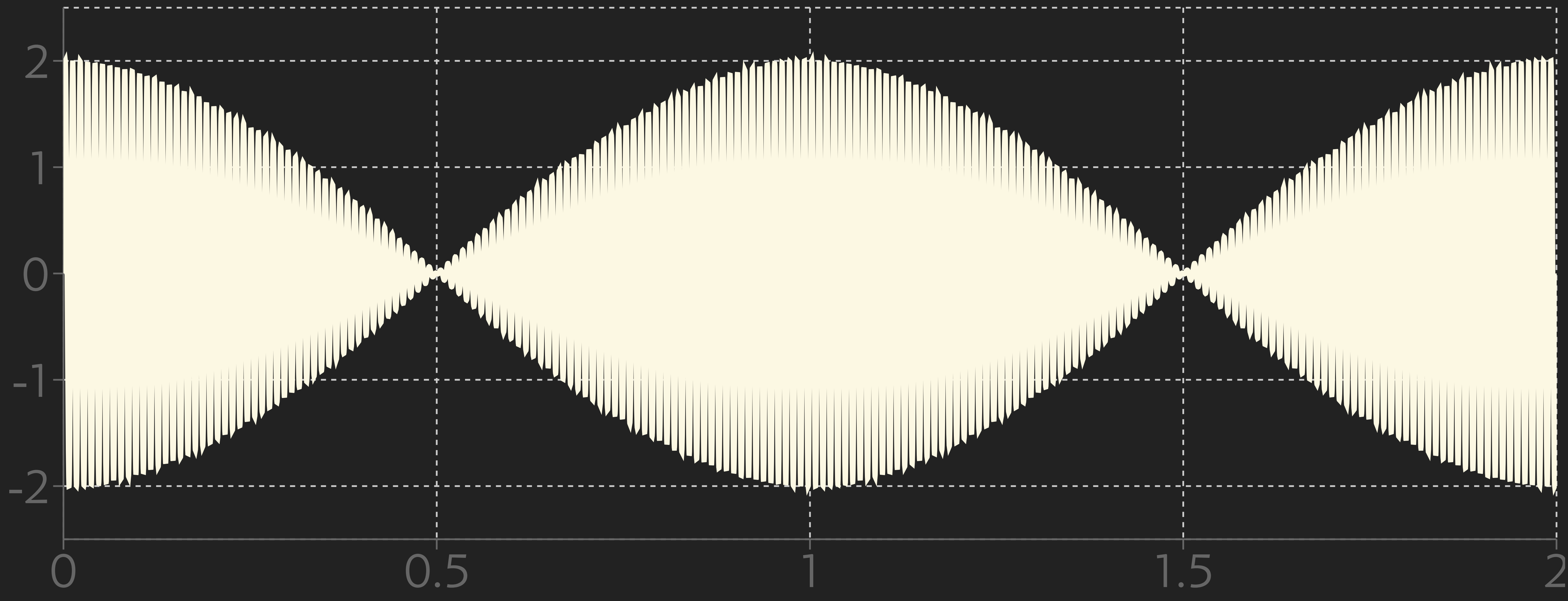
100 hz



101 hz



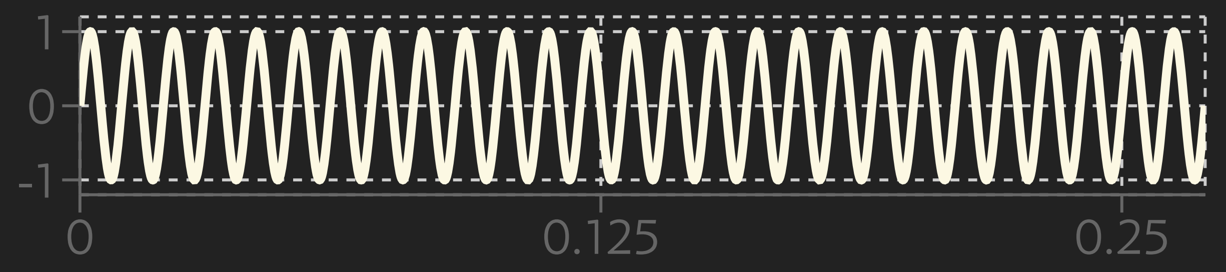
100hz + 101hz



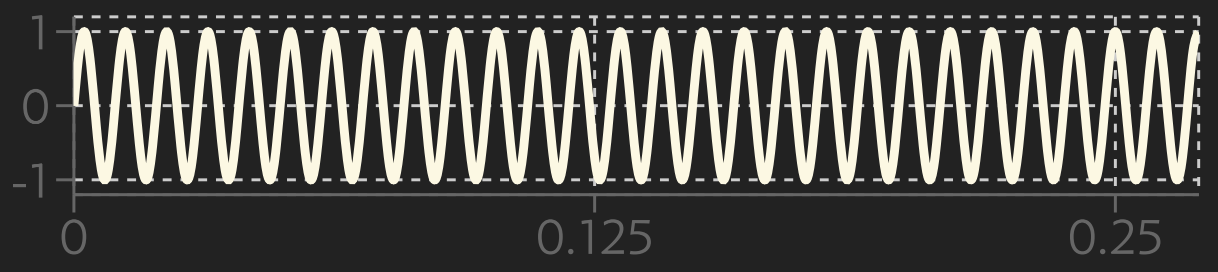
$$y(t) = \underbrace{\sin\left(2\pi(f+\frac{\Delta f}{2})t\right)}_{\sin(2\pi f)\cos\left(2\pi t\frac{\Delta f}{2}\right)+\cos(2\pi f)\sin\left(2\pi t\frac{\Delta f}{2}\right)} + \underbrace{\sin\left(2\pi(f-\frac{\Delta f}{2})t\right)}_{\sin(2\pi f)\cos\left(-2\pi t\frac{\Delta f}{2}\right)+\cos(2\pi f)\sin\left(-2\pi t\frac{\Delta f}{2}\right)}$$

$$= 2\sin\left(2\pi f\right)\cdot\cos\left(2\pi\frac{\Delta f}{2}t\right)$$

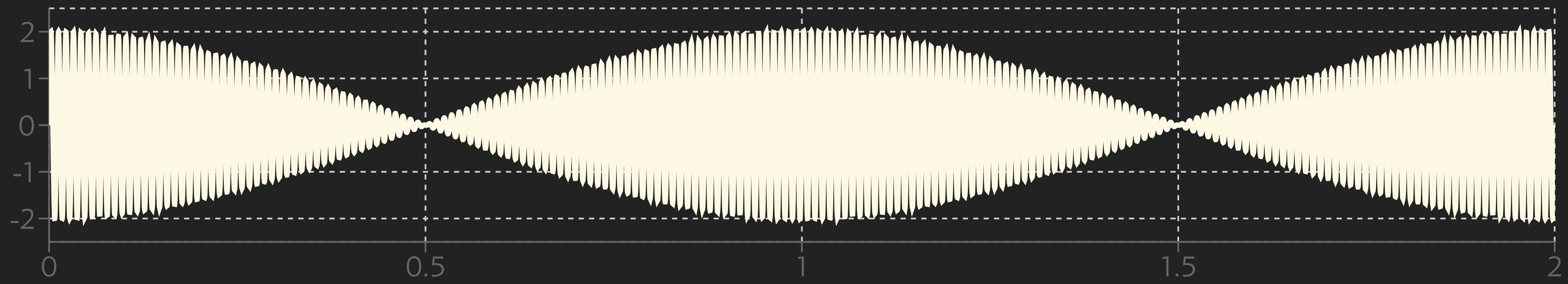
100 hz



101 hz



100hz + 101hz

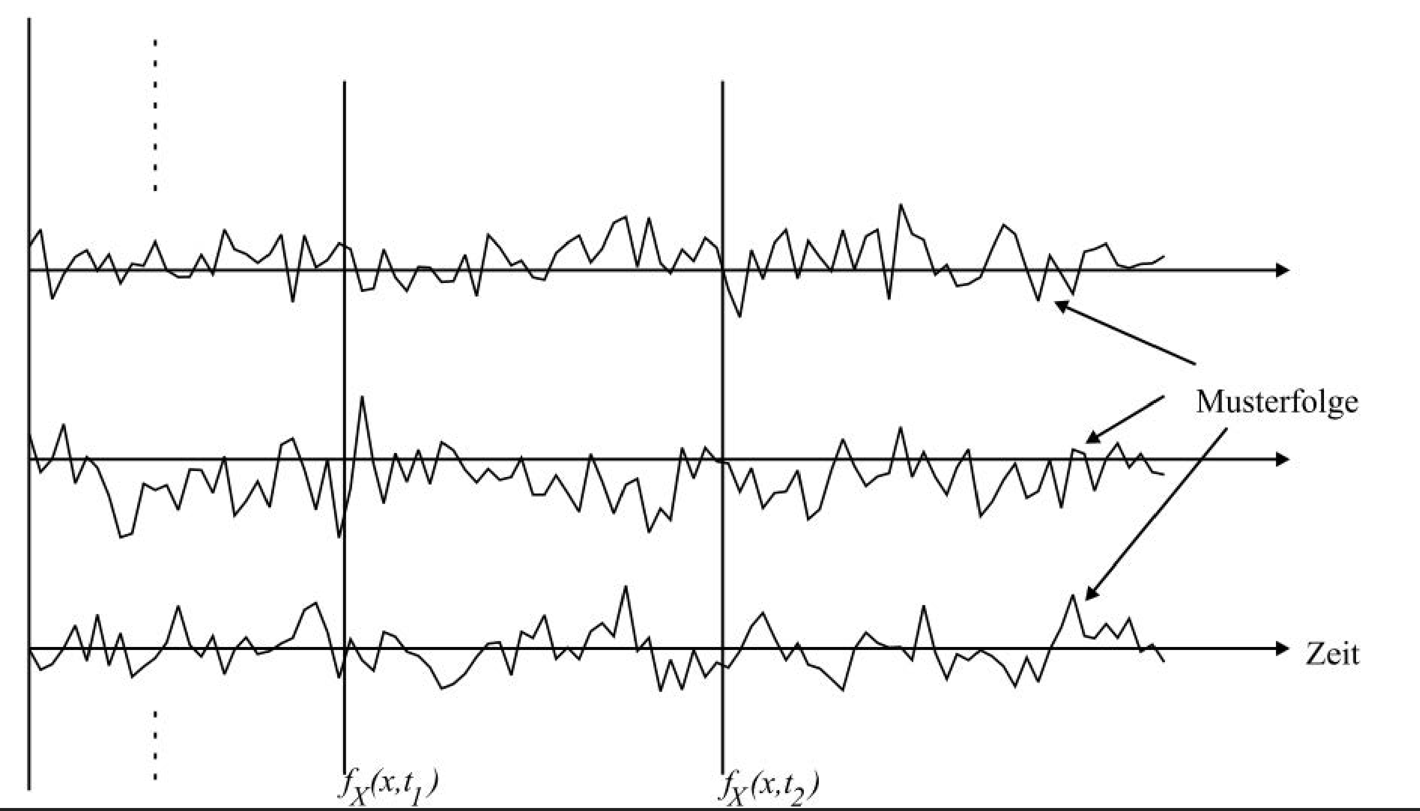


Beating examples, try at https://intonal.io/

```
main = {sr: float32 in
  hzs = [500]
//  hzs = [500, 1000]
//  hzs = [500, 750]
//  hzs = [500, 667]
//  hzs = [500, 625]
//  hzs = [500, 600]
//  hzs = [500, 600, 750]
//  hzs = [500, 530]
//  hzs = [500, 502]
//  hzs = [500, 501]
//  hzs = [500, 500 + playSin(0.01, 100, sr)]

  amp = 0.5 / float32(hzs.len())

  out = hzs
    .multiReduce(0, {prev, hz in
      prev + playSin(hz, amp, sr)
    })
}

playSin = {hz, amp, sr in
  p = phasor(hz, sr)
  out = sin(p * 2 * PI) * amp
}

phasor = {hz: float32, sr: float32 in
  out = 0 fby ((prev + (hz/sr)) % 1)
}

PI = 3.14159265358
```

Random Process

Ensemble of random series



Special Cases:

- **Stationarity:** all parameters (such as the mean) are time invariant
- **Ergodicity:** process with equal time and ensemble mean (implies stationarity)

Common Periodic Signals

Sinusoidal

$$x(t) = \sin(\underbrace{2\pi f}_{\omega} t + \Phi)$$

Sawtooth

$$x(t) = 2\left(\frac{t}{T_0} - \operatorname{floor}\left(\frac{1}{2} + \frac{t}{T_0}\right)\right)$$

Square Wave

$$x(t) = \operatorname{sign}(\sin(\omega t))$$

Common Periodic Signals

DC

$$x(t) = 1$$

Impulse

$$\delta(t) = \begin{cases} \infty & \text{if t} = 0 \\ 0 & \text{if t} != 0 \end{cases}$$

Summary

- Two basic signal classes, **deterministic** and **random**
- *Deterministic* signals can be described by a function and are predictable
- Special case: Periodic signals - sum of sinusoidals with freq. integer ratio
- *Random* signals are not predictable
- Special case: Ergodic signals can be described staticstically