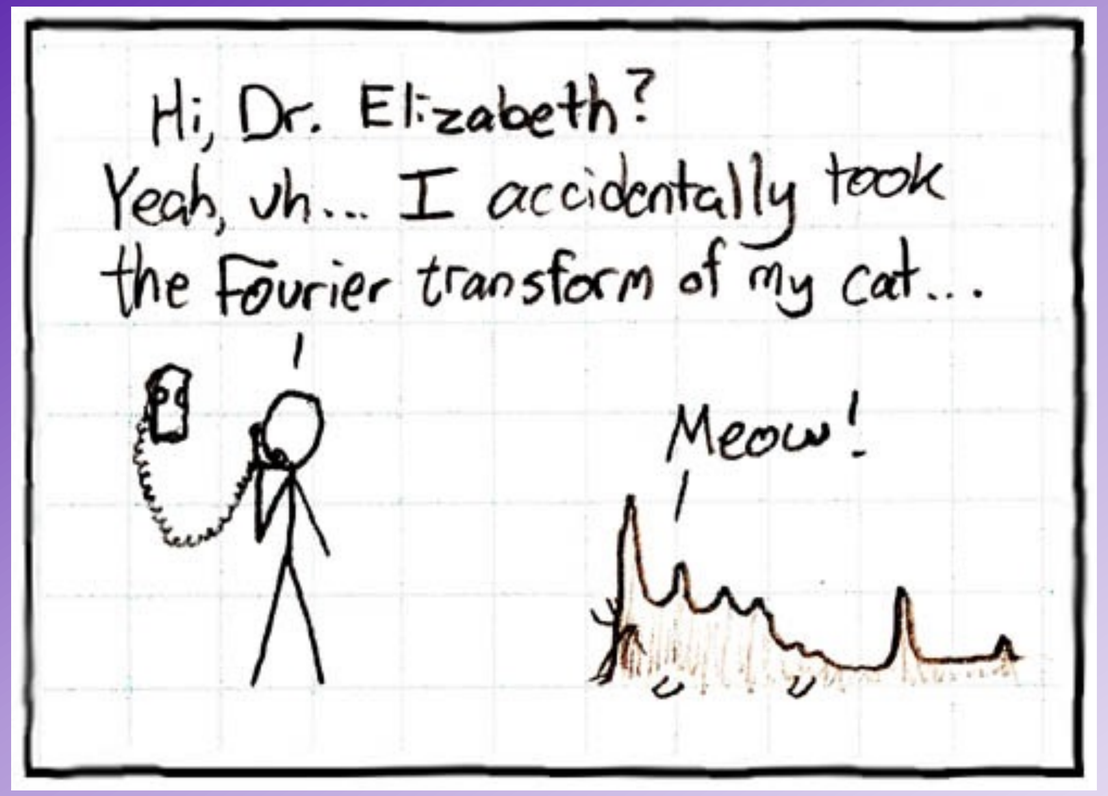


Lecture 11: Fourier Transform

ENAE 380 Flight Software Systems
October 9, 2024

Today

- Fourier Series Overview
- Discrete Fourier Transform
- Fast Fourier Transform

Signal Decomposition

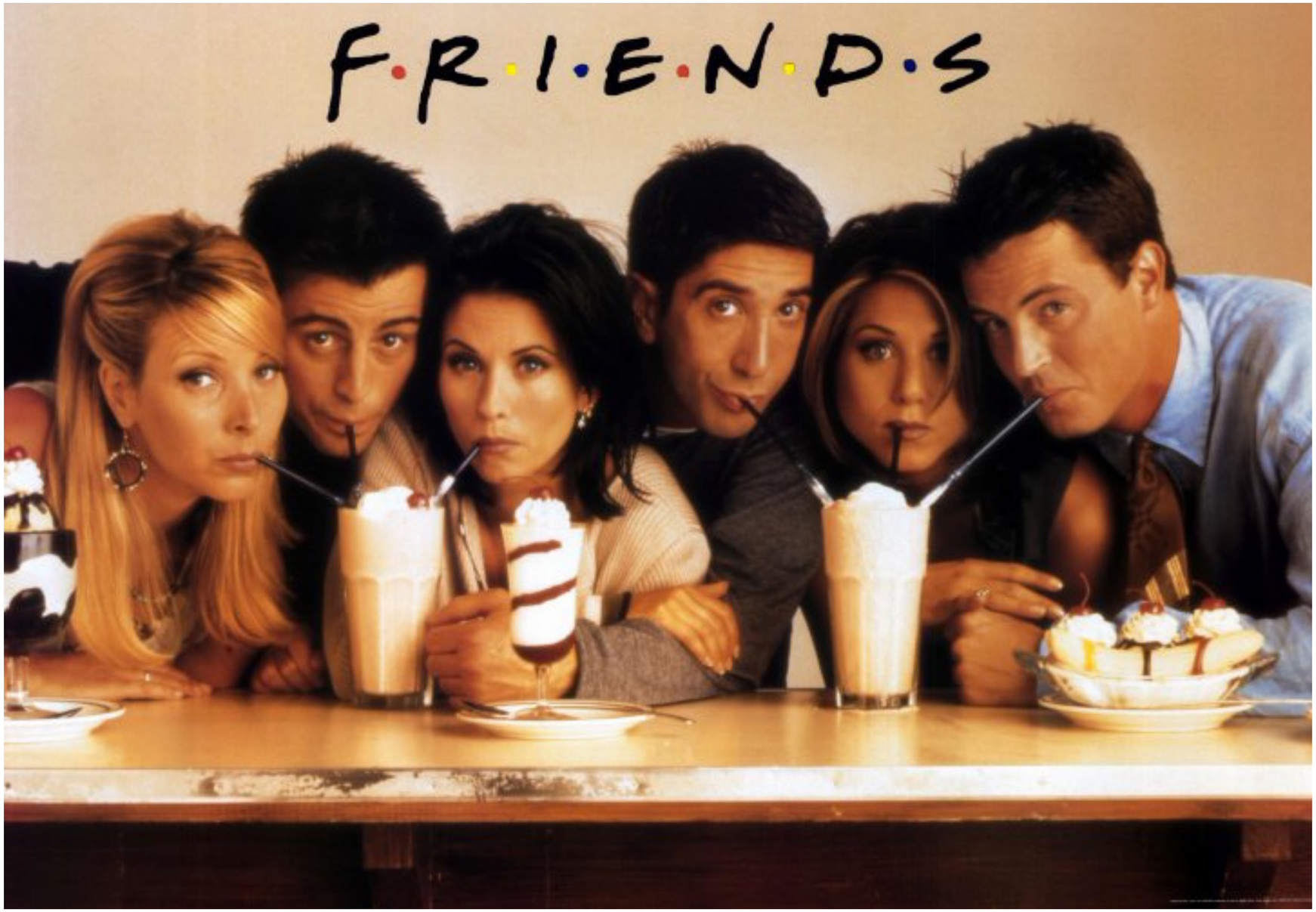
- Two main decompositions
  - Impulse decomposition: convolution
  - Fourier decomposition

Superposition is the key to digital signal processing

Fourier Transform

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i 2\pi k n / N}$$

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot e^{i 2\pi k n / N}$$


F·R·I·E·N·D·S

Cookies

What does Fourier Transform do?

Finds the recipe for the cookie

How?

Run cookie through filters to extract each ingredient

Why?

Recipes are easier to analyze, compare, and modify

How do we get cookie back?

Mix the ingredients

Transformation: change of perspective

“What did I see?” becomes “How was it made?”

Cookie to Recipe

	Flour Filter	2 cups flour
	Butter Filter	1 cup butter
	Sugar Filter	1.5 cups sugar
	Chocolate Filter	2 cups choc. chips
Consumer	Transformation	Producer

We can reverse engineer the cookie if:

1. Filters are independent. Flour filter only captures flour and nothing else. Adding more sugar doesn’t affect flour reading.
2. Filters must be complete. We won’t get the real recipe if we leave out a filter (walnuts!)
3. Ingredients must be combineable. Cookies can be separated and recombined without issue. The ingredients, when separated and combined in any order, must make the same result.

Fourier Transform: What if any signal could be filtered into a bunch of circular paths?

Earthquake vibrations	Design buildings to avoid interacting with strongest vibrations
Sound waves	Boost parts we care about, noise, bass, treble
Radio waves	Filter to a particular channel
Computer Data	Lossy compression

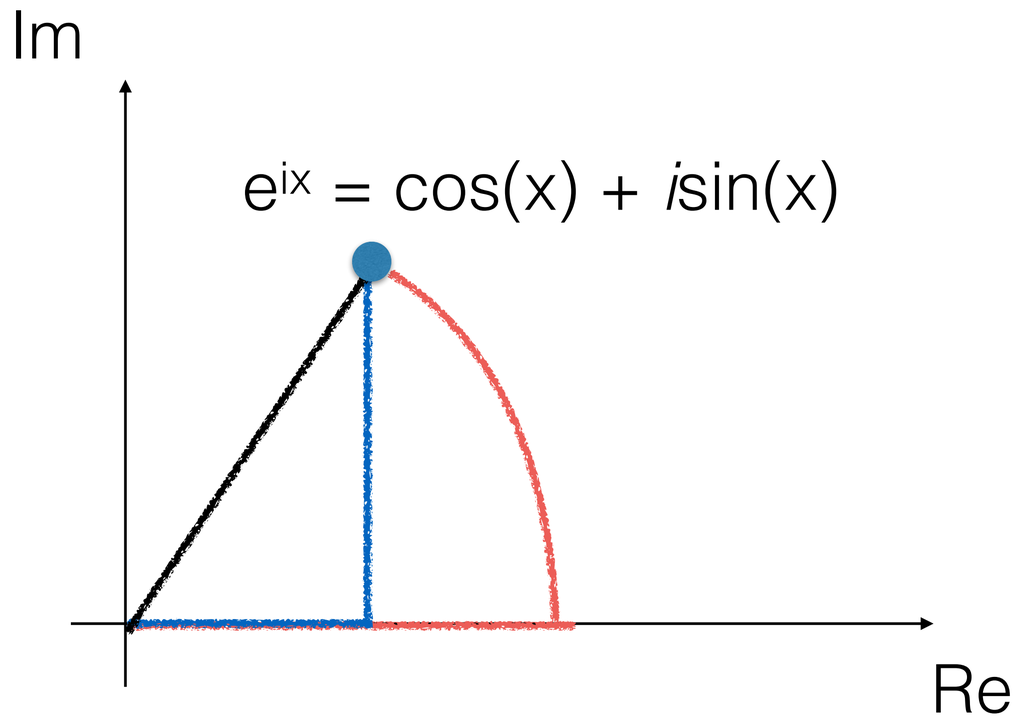
The Fourier Transform is a metaphor for finding
the root causes behind an observed effect.

A sinusoid is a back and forth pattern

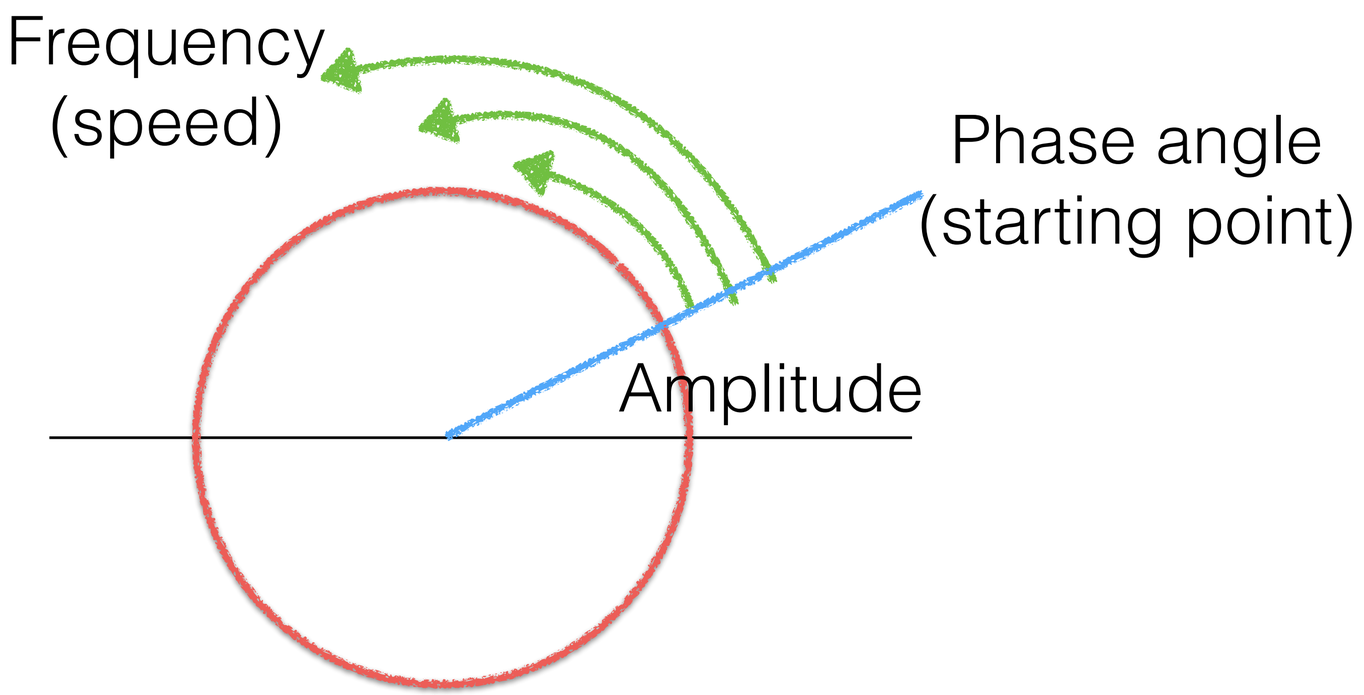
A circle is a 2D pattern

A circular path is a complex sinusoid

Think with circles, not with sinusoids

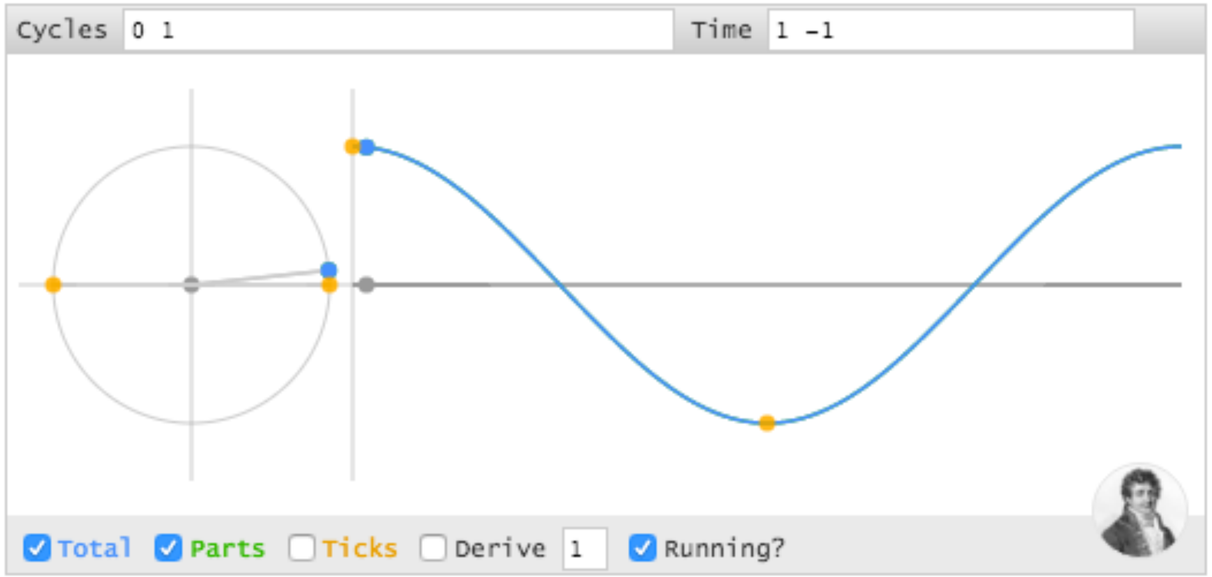


Following circular paths

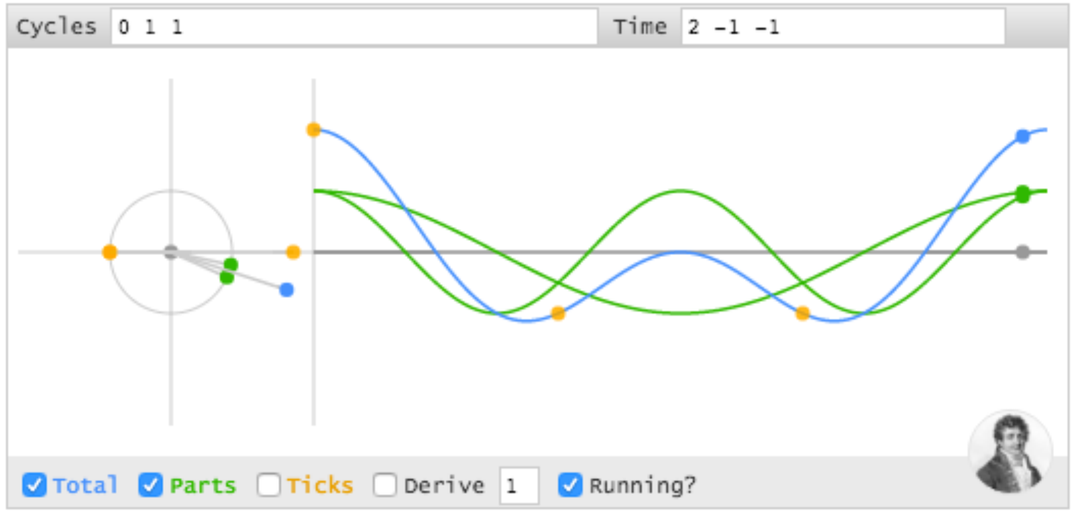


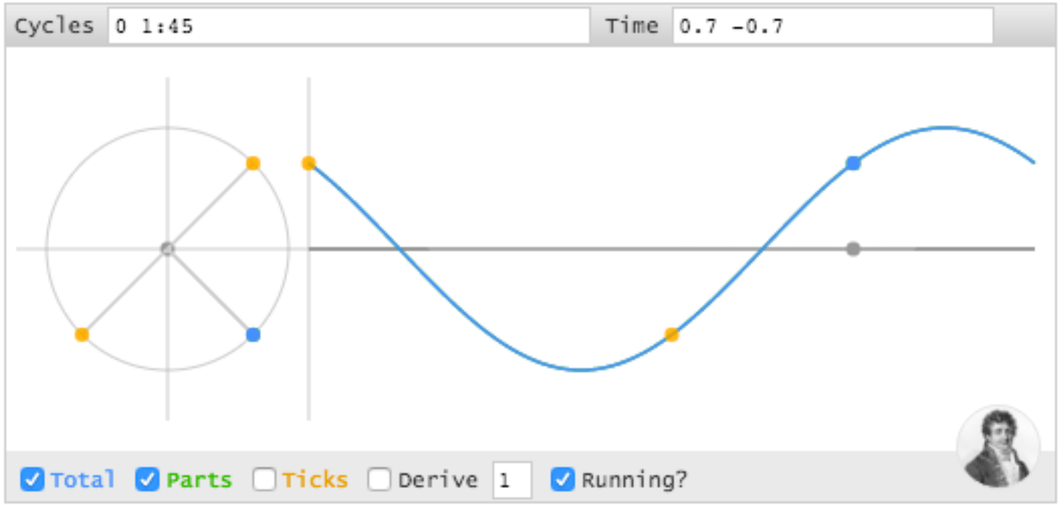
Following circular paths

https://betterexplained.com/examples/fourier/

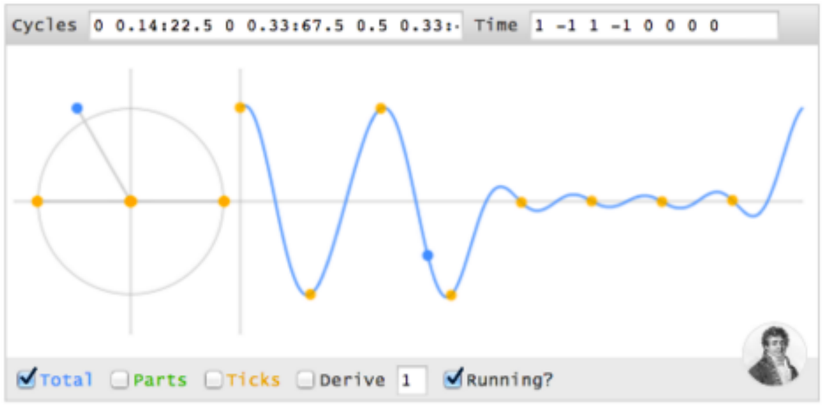


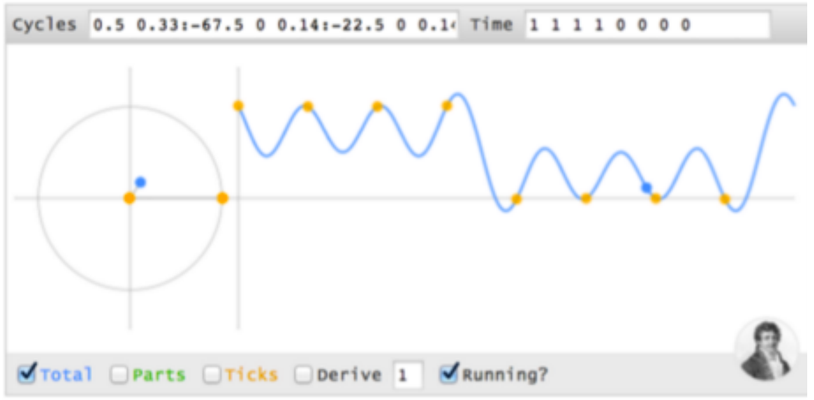
Following circular paths



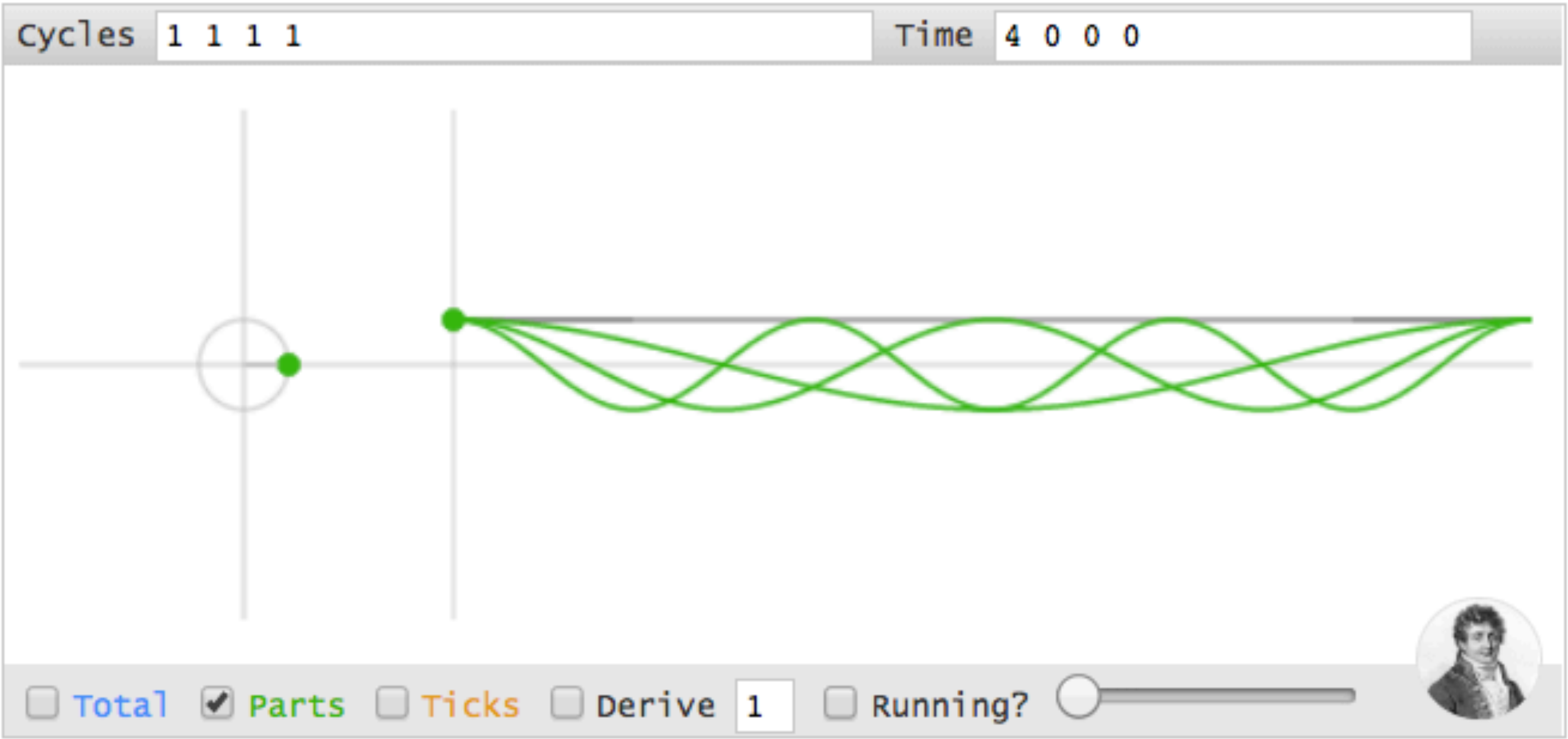


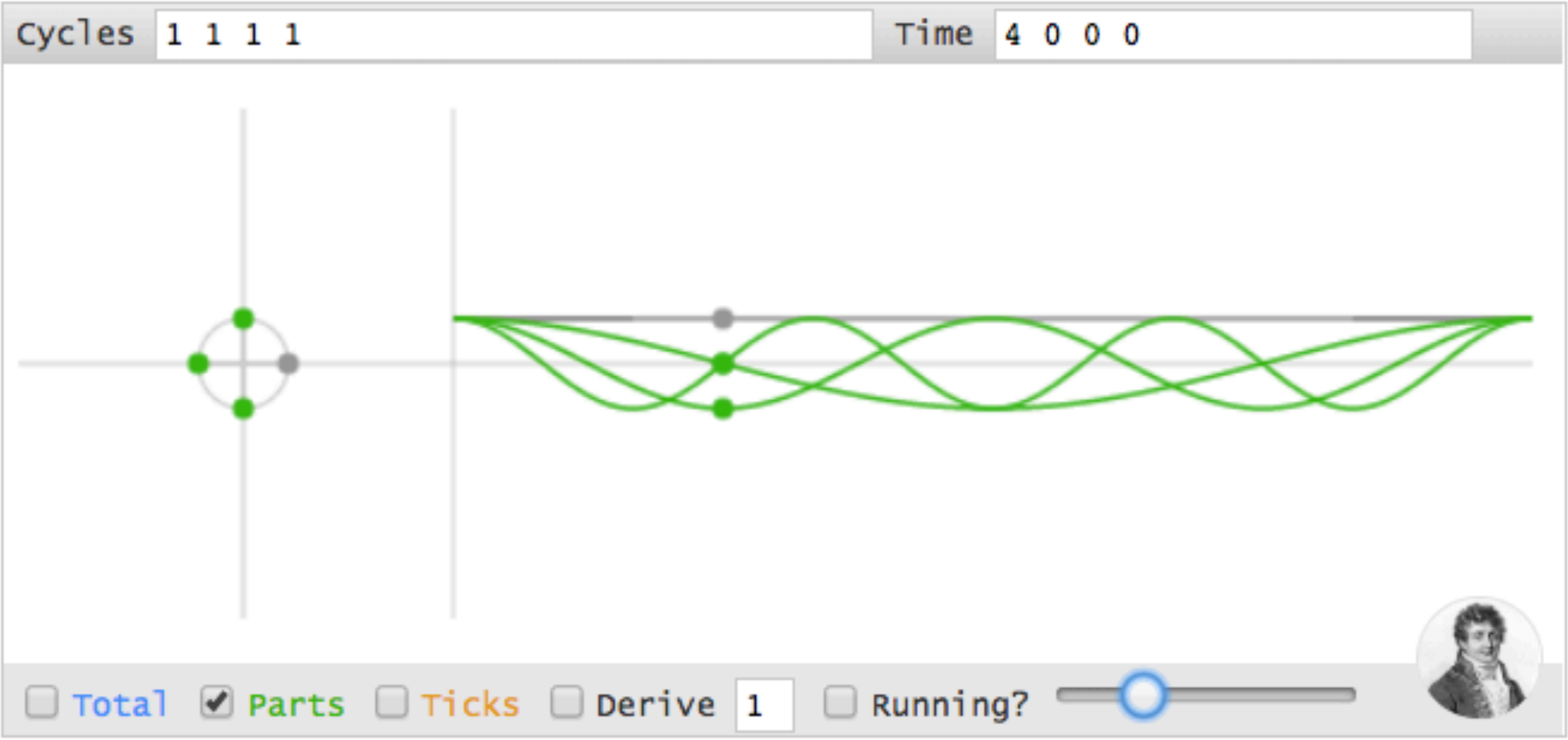
The Fourier Transform finds the set of cycle speeds, strengths, and phases to match any time signal

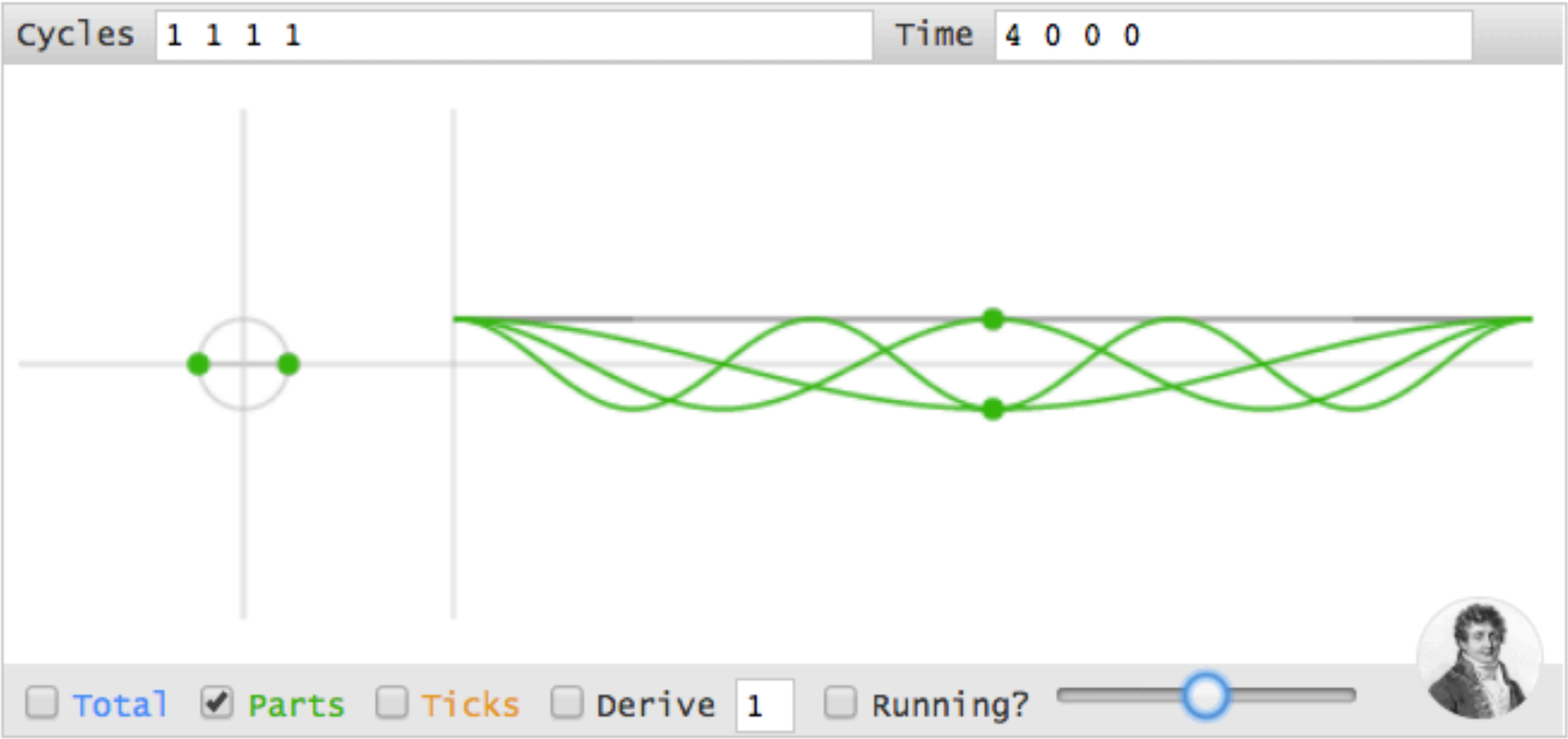


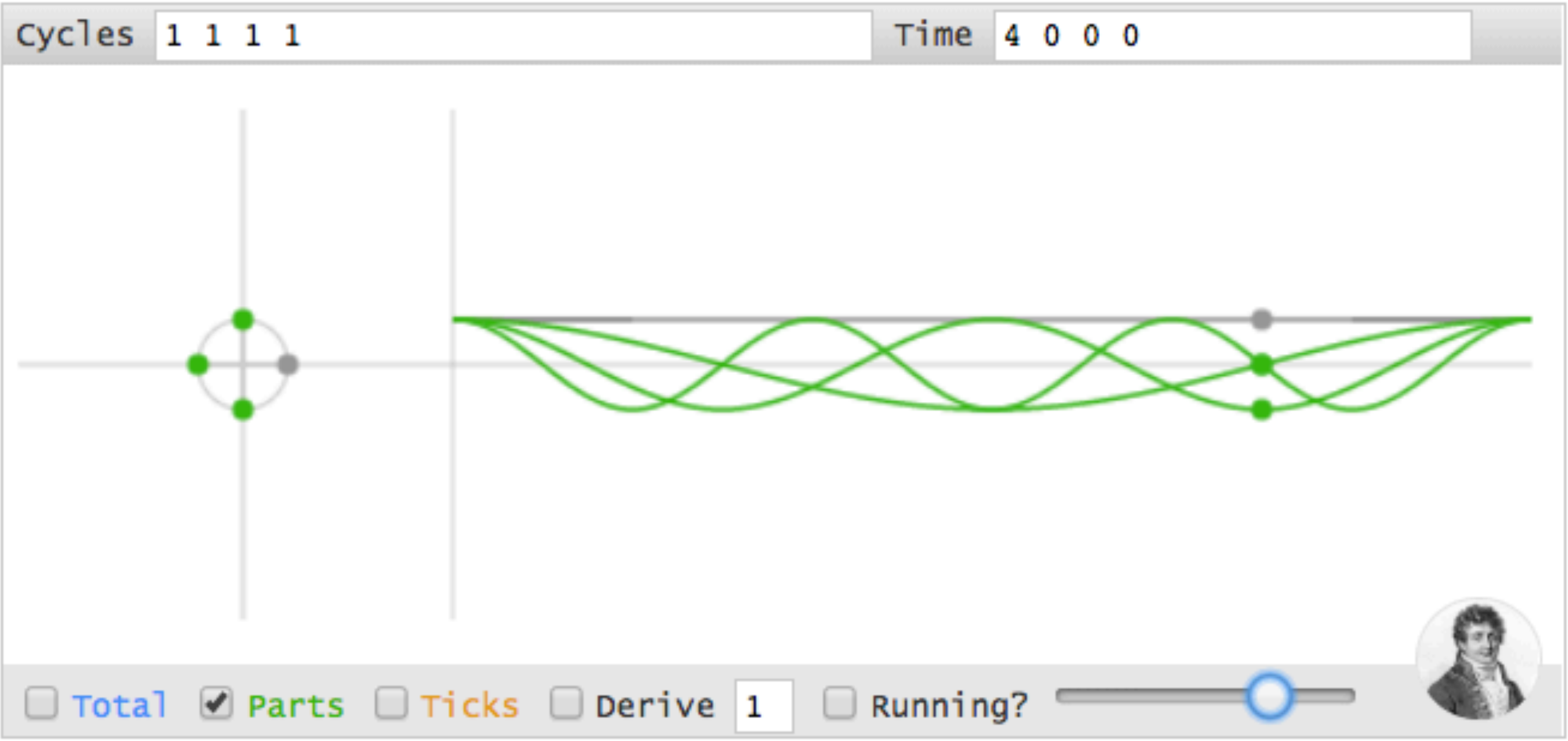


Making a spike in time

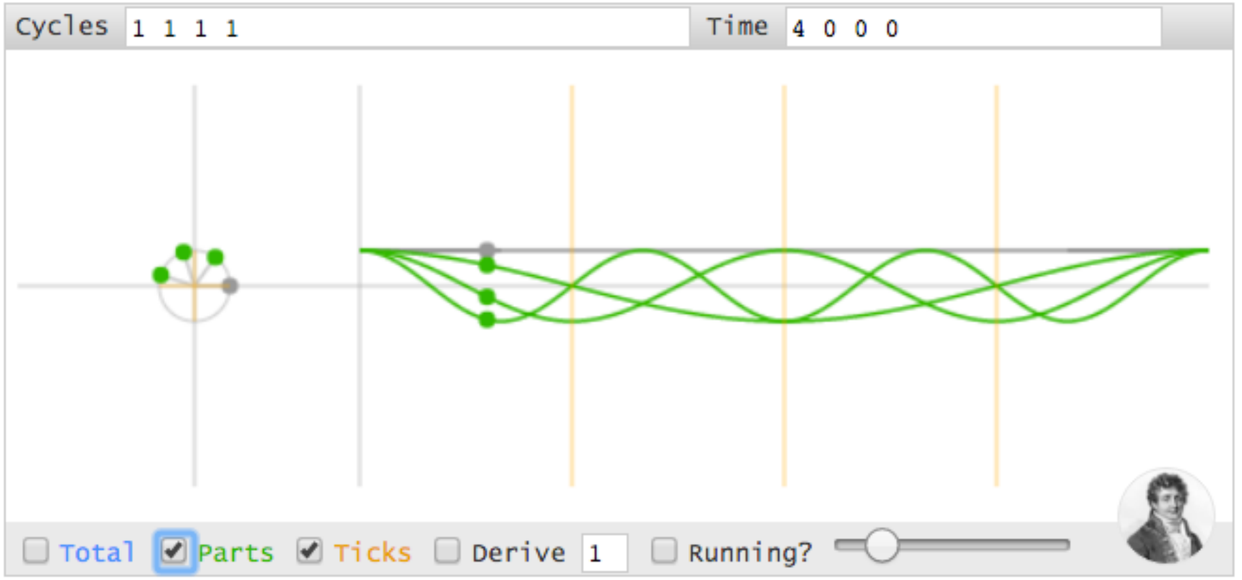


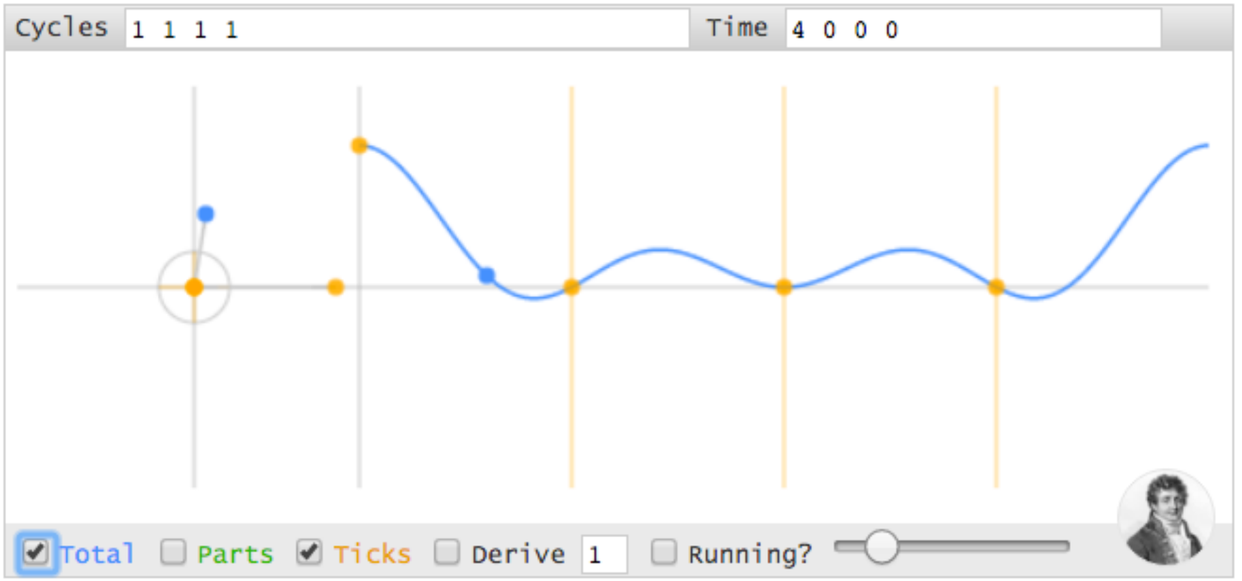






Making a spike in time

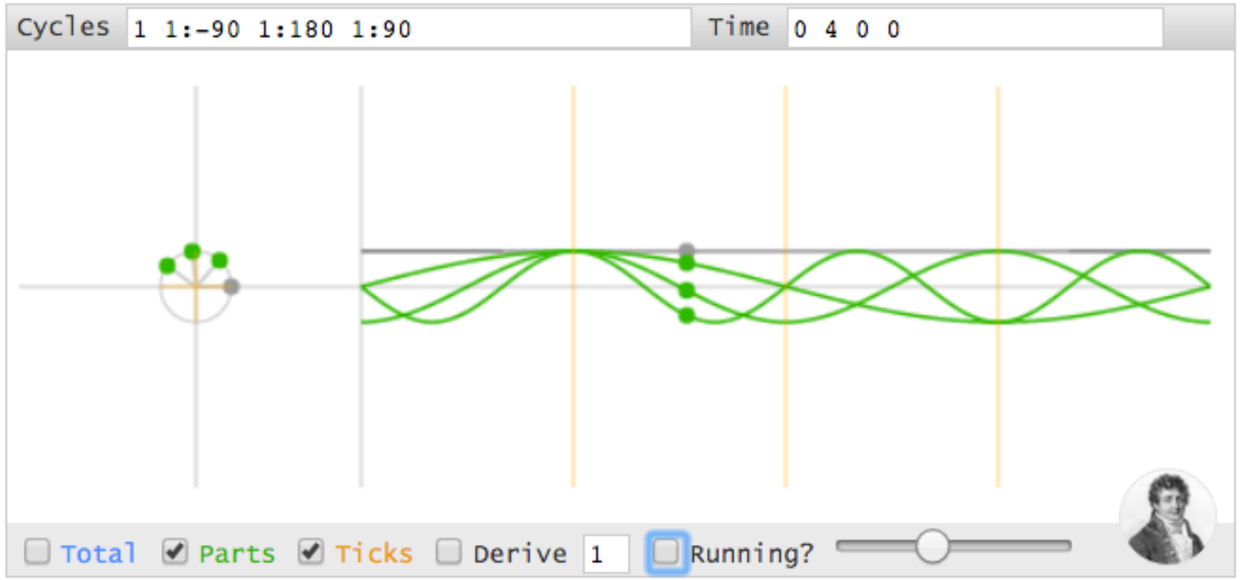


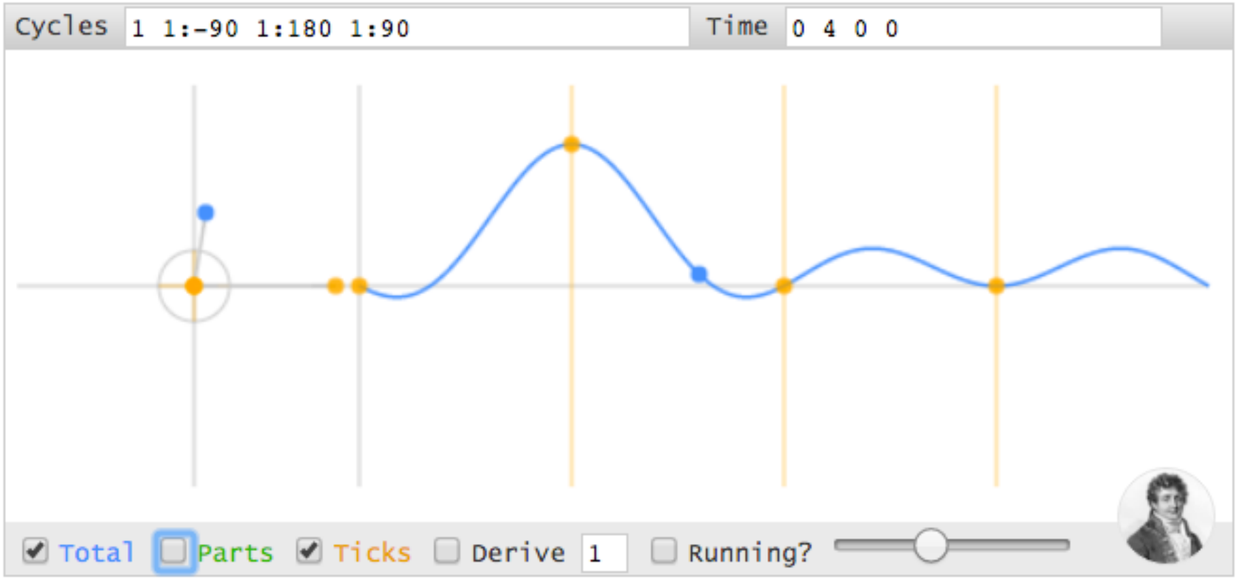


Moving the Time Spike

https://www.youtube.com/watch?v=jy5UDtRmbEA

Moving the Time Spike





Discovering the Full Transform

(a b c d)

a/4 (no offset)
\+ b/4 (1 second offset)
\+ c/4 (2 second offset)
\+ d/4 (3 second offset)

Frequency Recipe

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi kn/N}$$

Contributions to this frequency, from each time spike

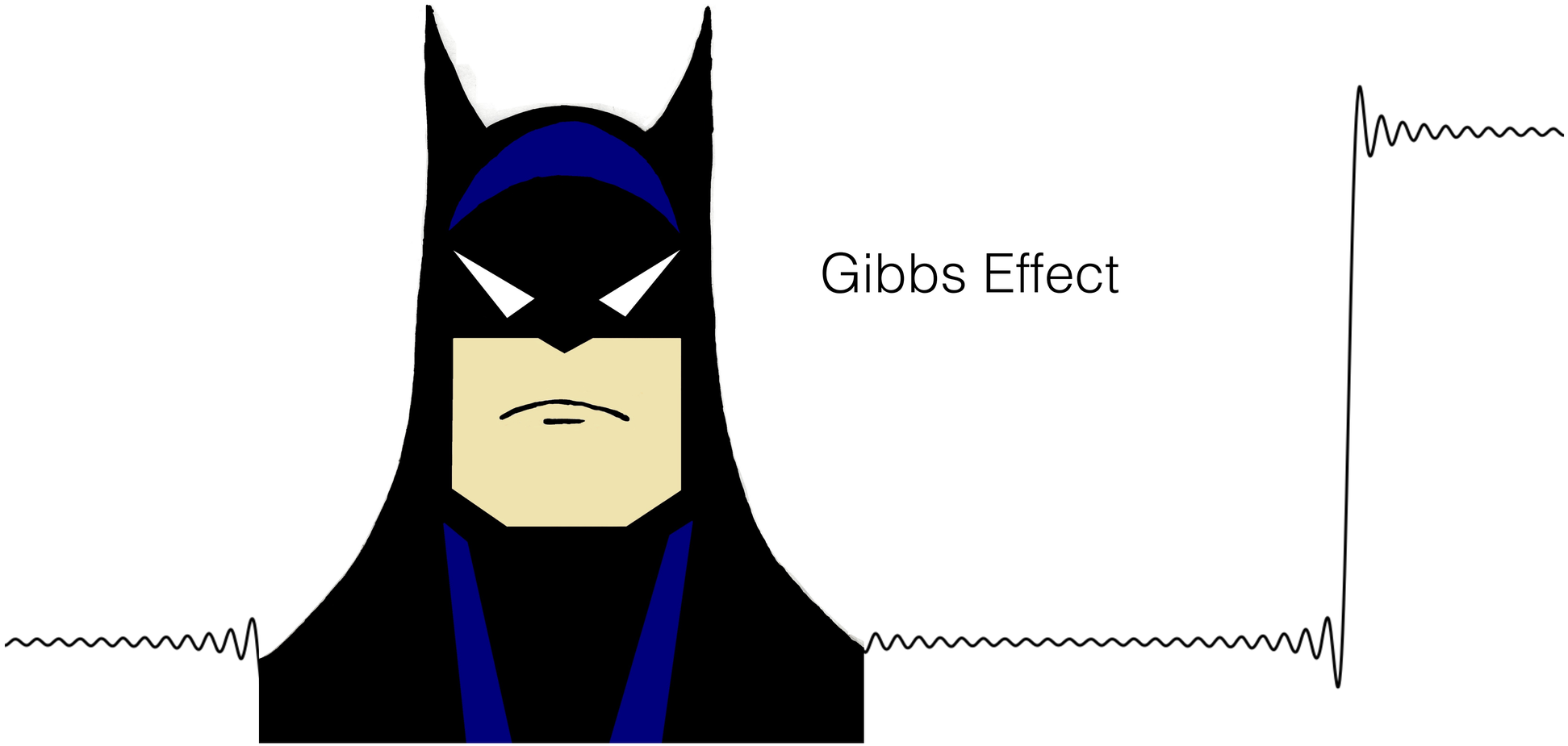
Time Point

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot e^{i2\pi kn/N}$$

Contributions to this time point from each frequency

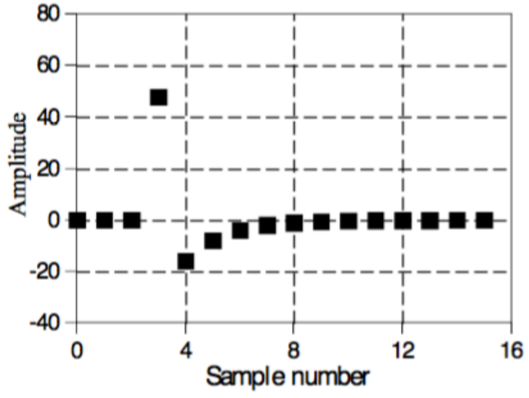
Fourier Transform Family
Jean Baptiste Joseph Fourier 1807
Heat Propagation

Lagrange versus Fourier
Who was right?

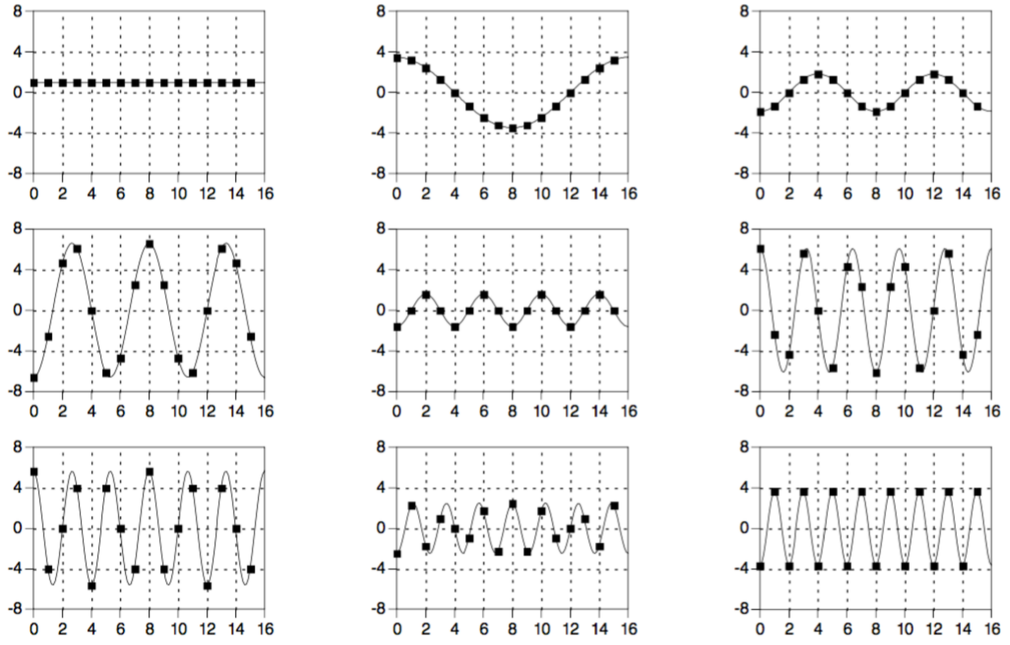


Gibbs Effect

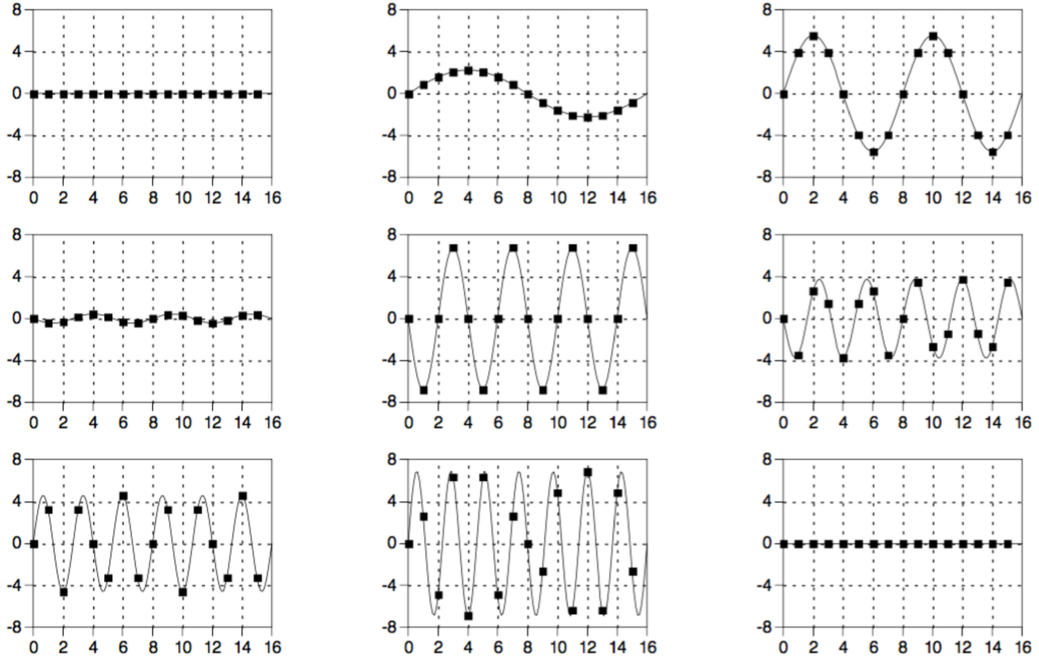
Discrete Fourier Transform



Cosine Waves



Sine Waves



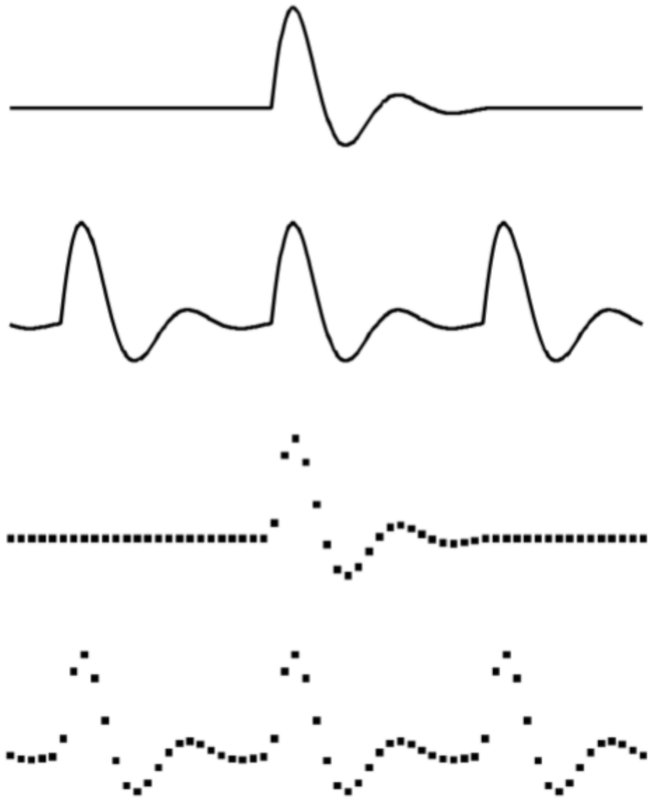
Fourier Transform

Aperiodic-Continuous: Fourier Transform

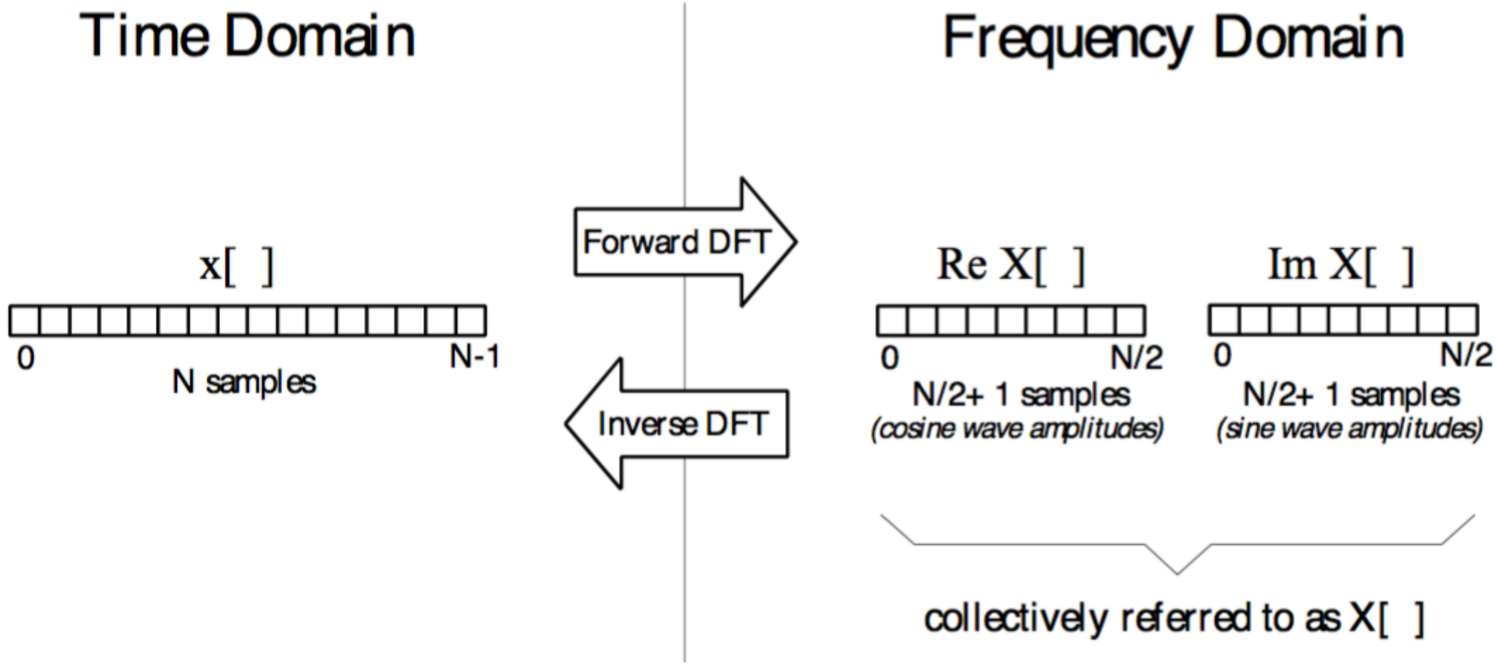
Periodic-Continuous: Fourier Series

Aperiodic-Discrete: Discrete Time Fourier Transform

Periodic-Discrete: Discrete Fourier Transform



Notation and Format
aka “keeping it real”



Solving the Real DFT

$$X_k = \sum_{n=0}^{N-1} x_n \cdot (\cos(-2\pi k \frac{n}{N}) + j \sin(-2\pi k \frac{n}{N})), \quad n \in \mathbb{Z}$$

N = number of time samples we have

n = current sample we're considering (0...N-1)

$\mathbf{x_n}$ = value of the signal at time n

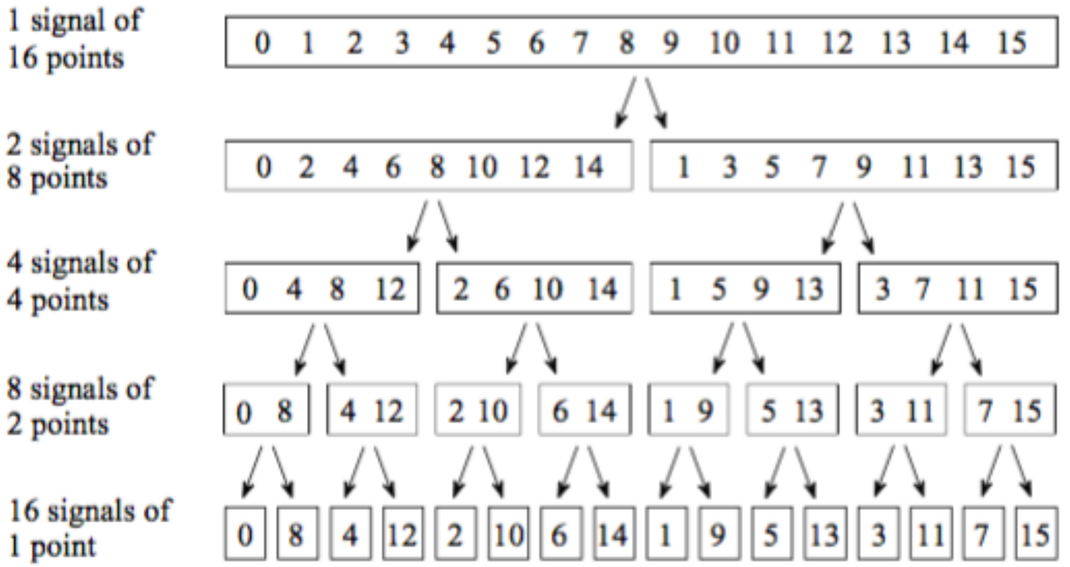
k = current frequency we're considering (0 Hertz up to N-1 Hertz)

$\mathbf{X_k}$ = amount of frequency k in the signal (Amplitude and Phase, a complex number)

1. Simultaneous Equations
2. Correlation
3. Fast Fourier Transform

Fast Fourier Transform

Time Domain Decomposition



$$f = (f[1], f[2], \ldots, f[n]), n = 2^k$$

$$\omega[p, q] = e^{2\pi i q/p}$$

$$\mathcal{F}f[m] = \sum_{k=0}^{n-1} f[k]\omega[n, -km]$$

$$\mathcal{F}f[m] = \sum_{k=0}^{\frac{n}{2}-1} f[2k]\omega[n, -2km] + \sum_{k=0}^{\frac{n}{2}-1} f[2k+1]\omega[n, -(2k+1)m]$$

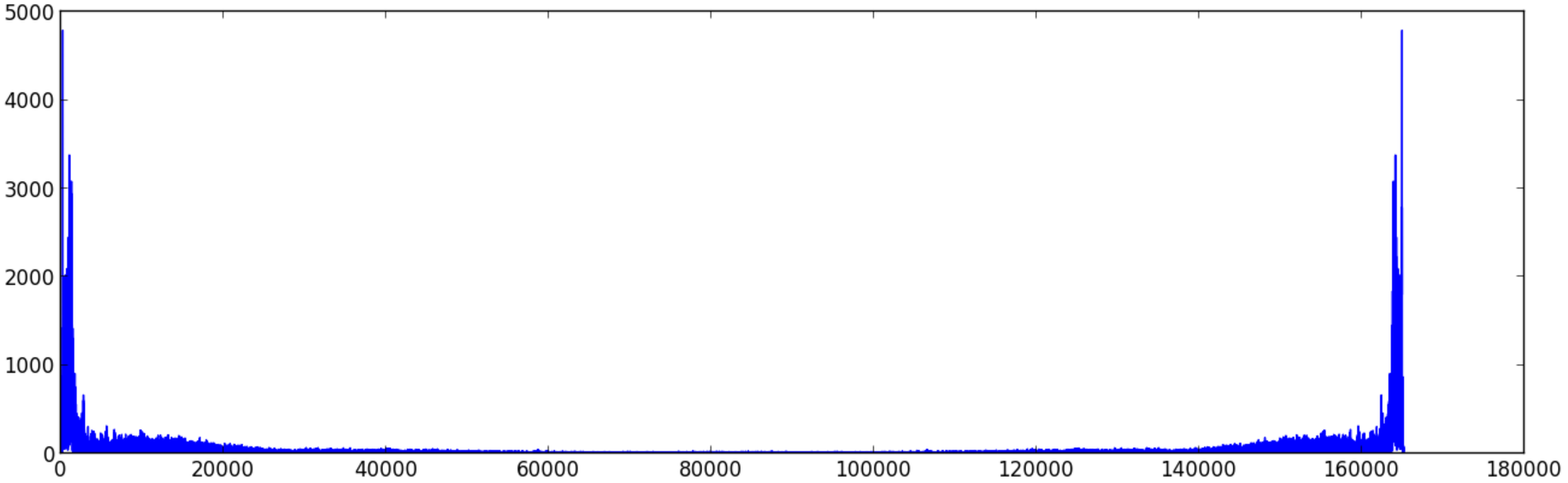
$$\mathcal{F}f[m] = \mathcal{F}f_{\text{even}}[m] + \omega[n, -m]\mathcal{F}f_{\text{odd}}[m]$$

$$\mathcal{F}f[m + n/2] = \mathcal{F}f_{\text{even}}[m] - \omega[n, -m]\mathcal{F}f_{\text{odd}}[m]$$

Implementation in Python

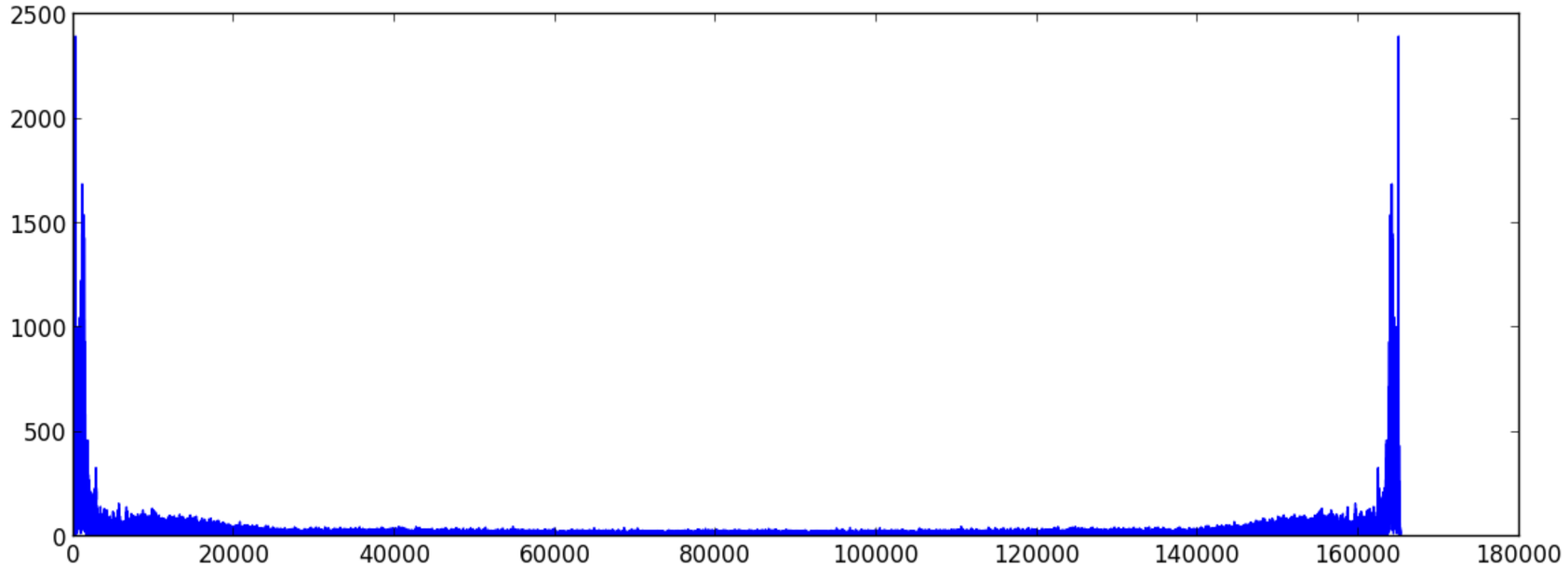
```
def omega(p, q):
    return cmath.exp((2.0 * cmath.pi * 1j * q) / p)


def fft(signal):
    n = len(signal)
    if n == 1:
        return signal
    else:
        Feven = fft([signal[i] for i in xrange(0, n, 2)])
        Fodd = fft([signal[i] for i in xrange(1, n, 2)])

        combined = [0] * n
        for m in xrange(n/2):
            combined[m] = Feven[m] + omega(n, -m) * Fodd[m]
            combined[m + n/2] = Feven[m] - omega(n, -m) * Fodd[m]

        return combined
```

“Tree, I am no tree, I am an ent.”



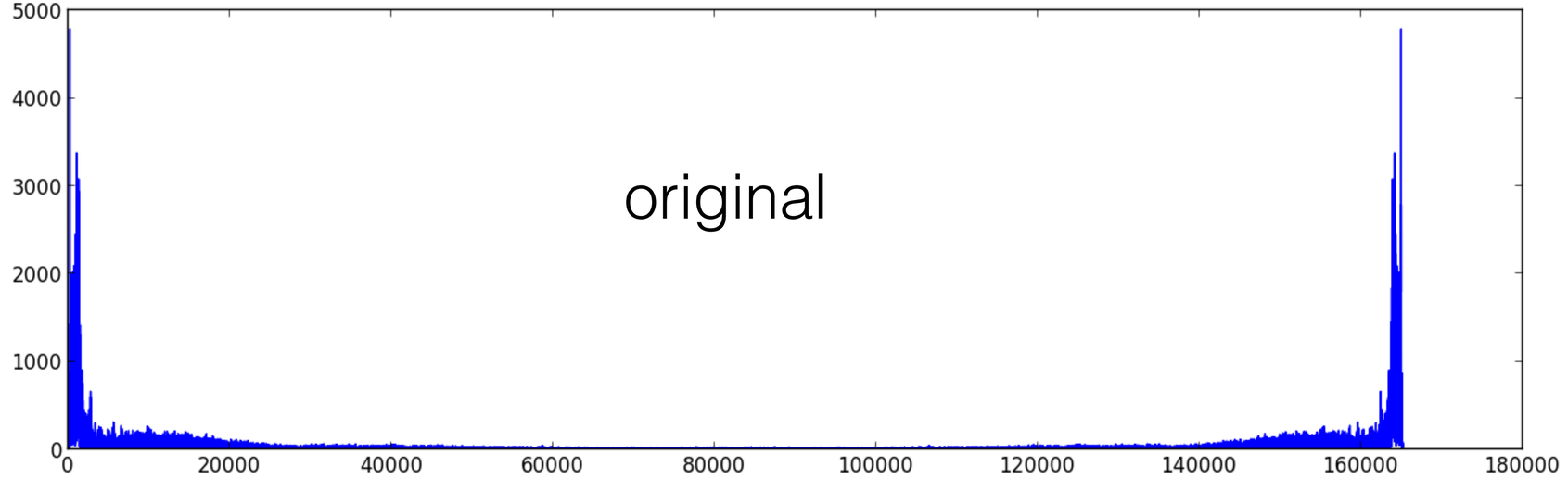
noise added

“Tree, I am no tree, I am an ent.”

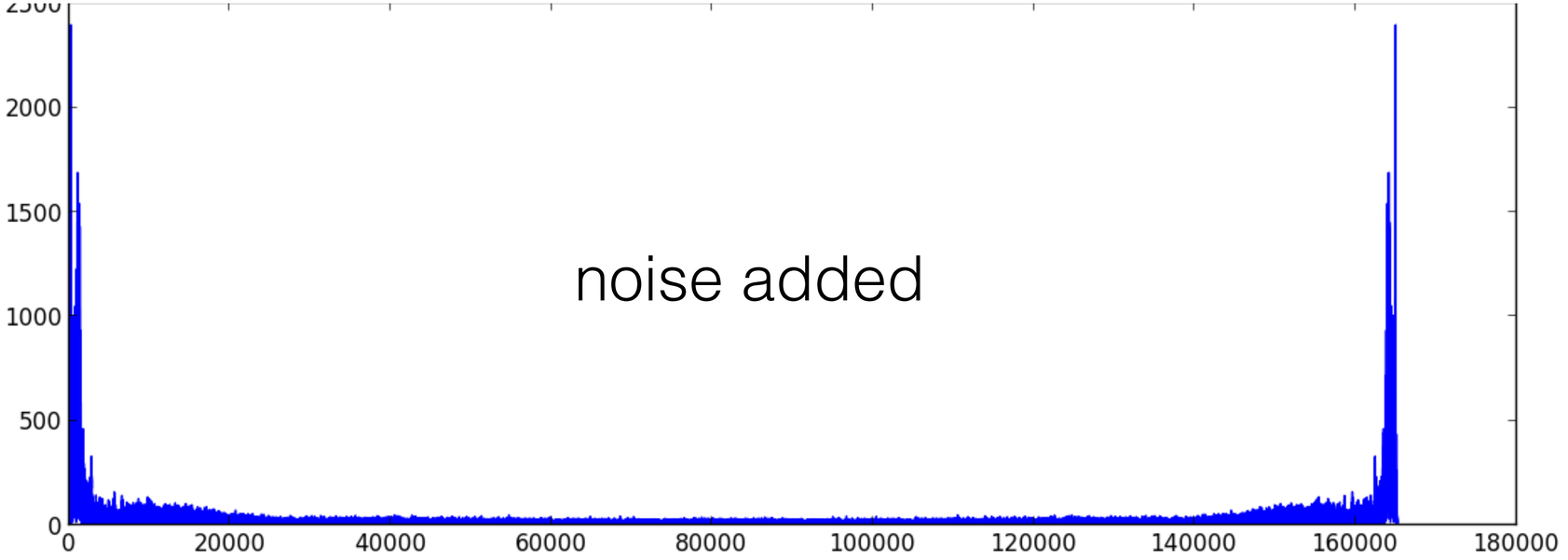



```
import random
inputSignal = [x/2.0 + random.random()*0.1 for x in inputSignal]
```

“Tree, I am no tree, I am an ent.”



original

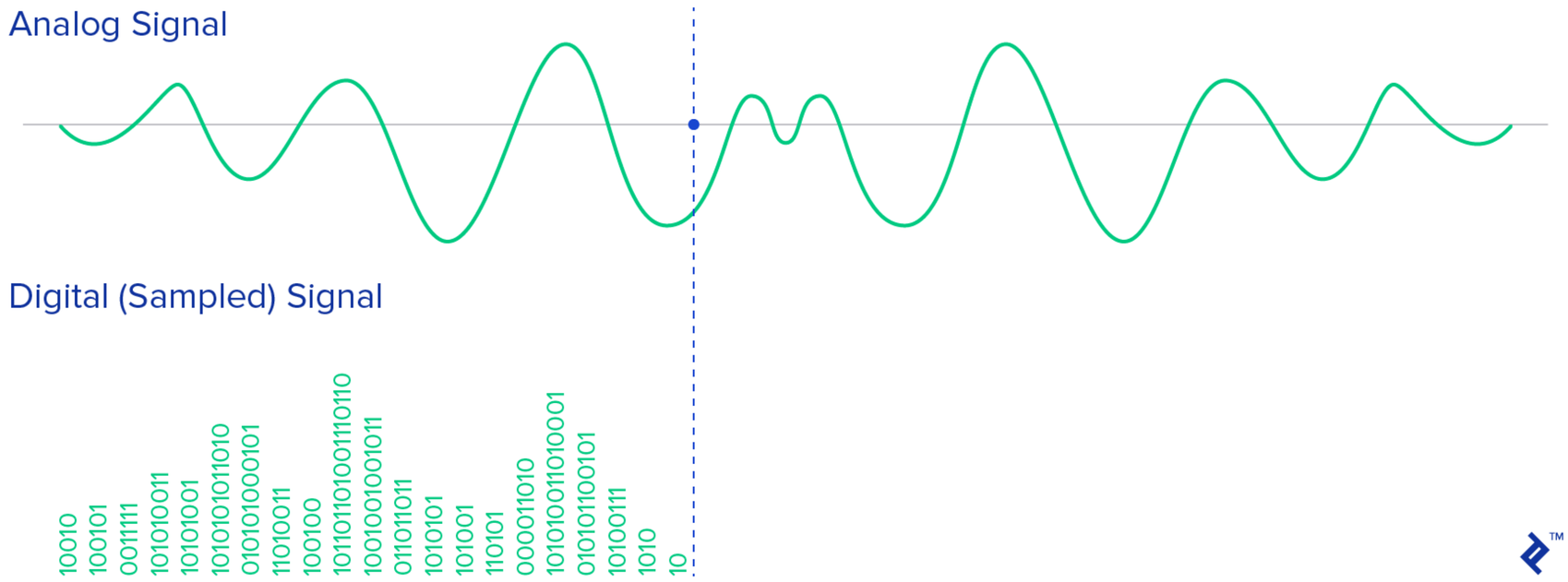


noise added

```
def frequencyFilter(signal):
    for i in range(20000, len(signal)-20000):
        signal[i] = 0
```

Shazam

1. Analog to Digital Conversion



20 Hz and 20,000 Hz

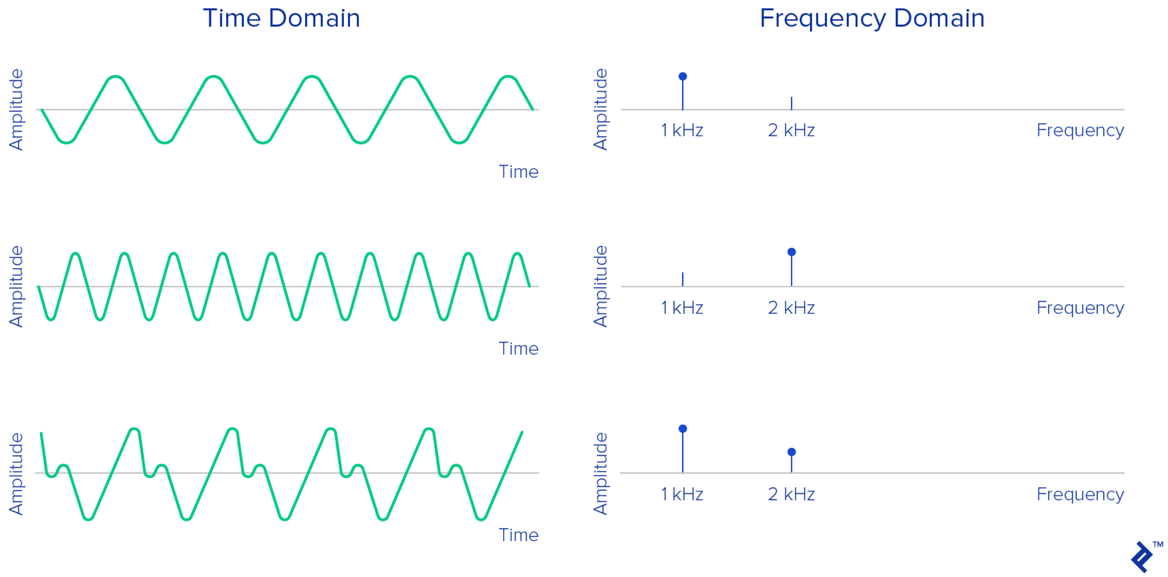
44,100 Hz

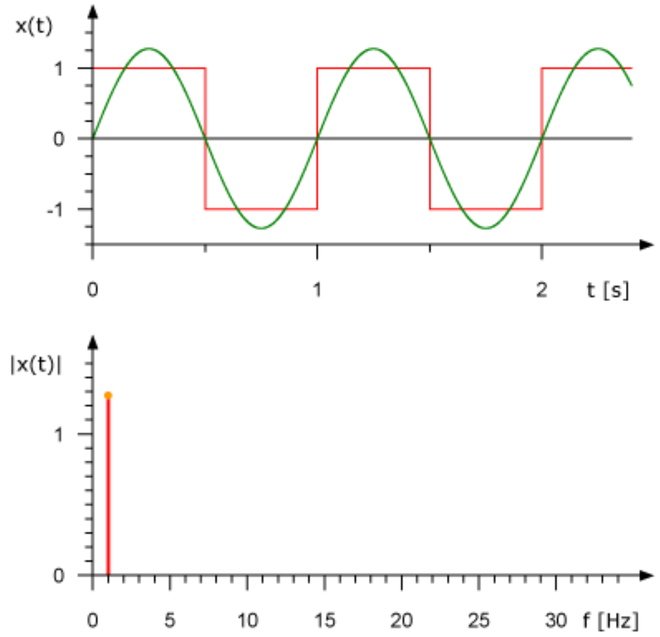
2. Recording - Capturing the Sound

Python

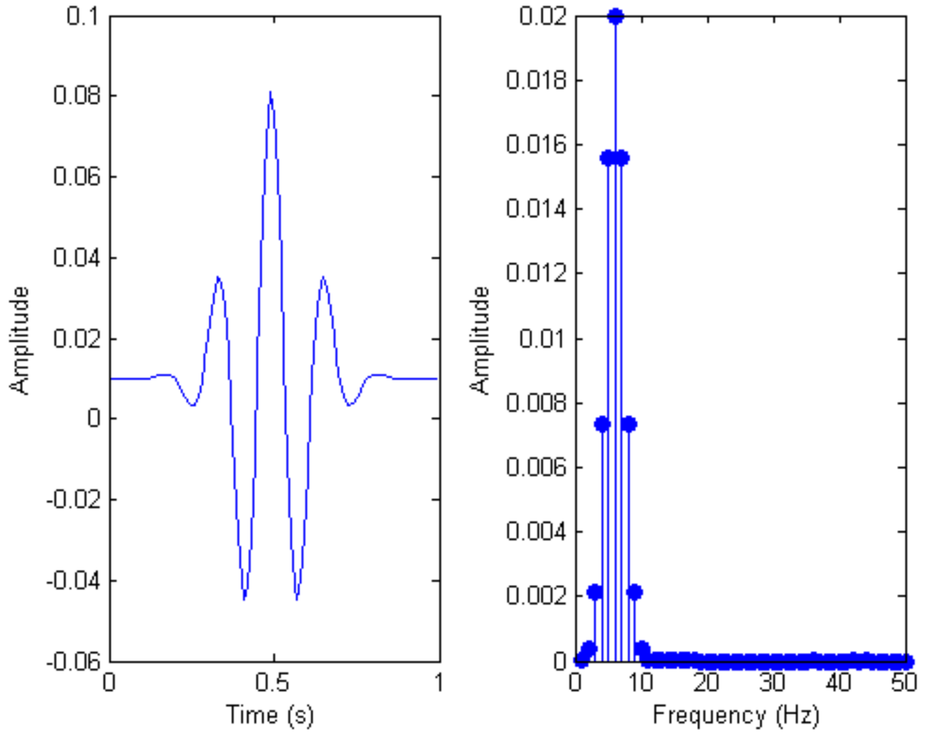
```python
import sounddevice as sd
from scipy.io.wavfile import write

fs = 44100  # Sample rate
seconds = 3  # Duration of recording

myrecording = sd.rec(int(seconds * fs), samplerate=fs, channels=2)
sd.wait()  # Wait until recording is finished
write('output.wav', fs, myrecording)  # Save as WAV file
```

3. Time Domain and Frequency Domain





4. Discrete Fourier Transform



5. Music Recognition: Fingerprinting a Song

Lose information about timing

Sliding window (chunks of data)

16 bit samples at 44,100 Hz
1 second of sound = 44,100 samples * 2 bytes* 2 channels ~ 176 kB
4kB for a chunk
44 chunks of data every second

5. Music Recognition: Fingerprinting a Song

Lose information about timing

Sliding window (chunks of data)

Hash Tag	Time in Seconds	Song
`30 51 99 121 195`	53.52	Song A by artist A
`33 56 92 151 185`	12.32	Song B by artist B
`39 26 89 141 251`	15.34	Song C by artist C
`32 67 100 128 270`	78.43	Song D by artist D
`30 51 99 121 195`	10.89	Song E by artist E
`34 57 95 111 200`	54.52	Song A by artist A
`34 41 93 161 202`	11.89	Song E by artist E

6. Music Algorithm: Song Identification