PAT 464/564 (Winter 2026)

Generative AI for Music & Audio Creation

Lecture 4: Audio Processing Fundamentals

Instructor: Hao-Wen Dong

UNIVERSITY OF MICHIGAN

How do we process audio on a computer?

Four Representative Music Representations

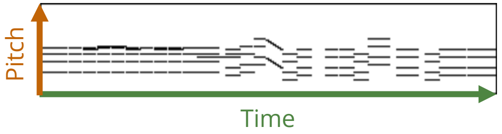
Symbolic music representations

Text-based

```
Program_change_0,
Note_on_60, Time_shift_2, Note_off_60,
Note_on_60, Time_shift_2, Note_off_60,
Note_on_76, Time_shift_2, Note_off_67,
Note_on_67, Time_shift_2, Note_off_67,
...
```

MIDI

Image-based



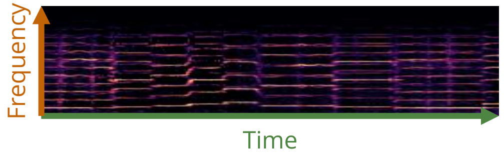
Piano roll

Audio-domain music representations

Time series-based

Waveform

Image-based

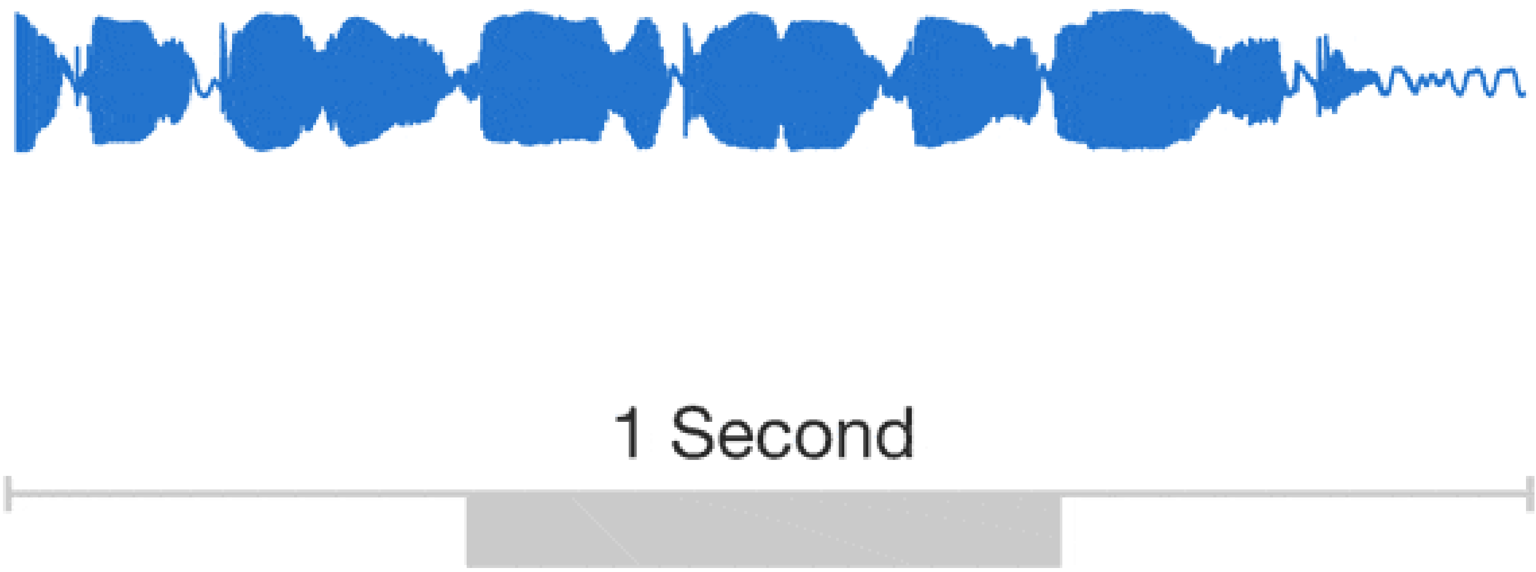


Spectrogram

Today's topic!

Digital Audio

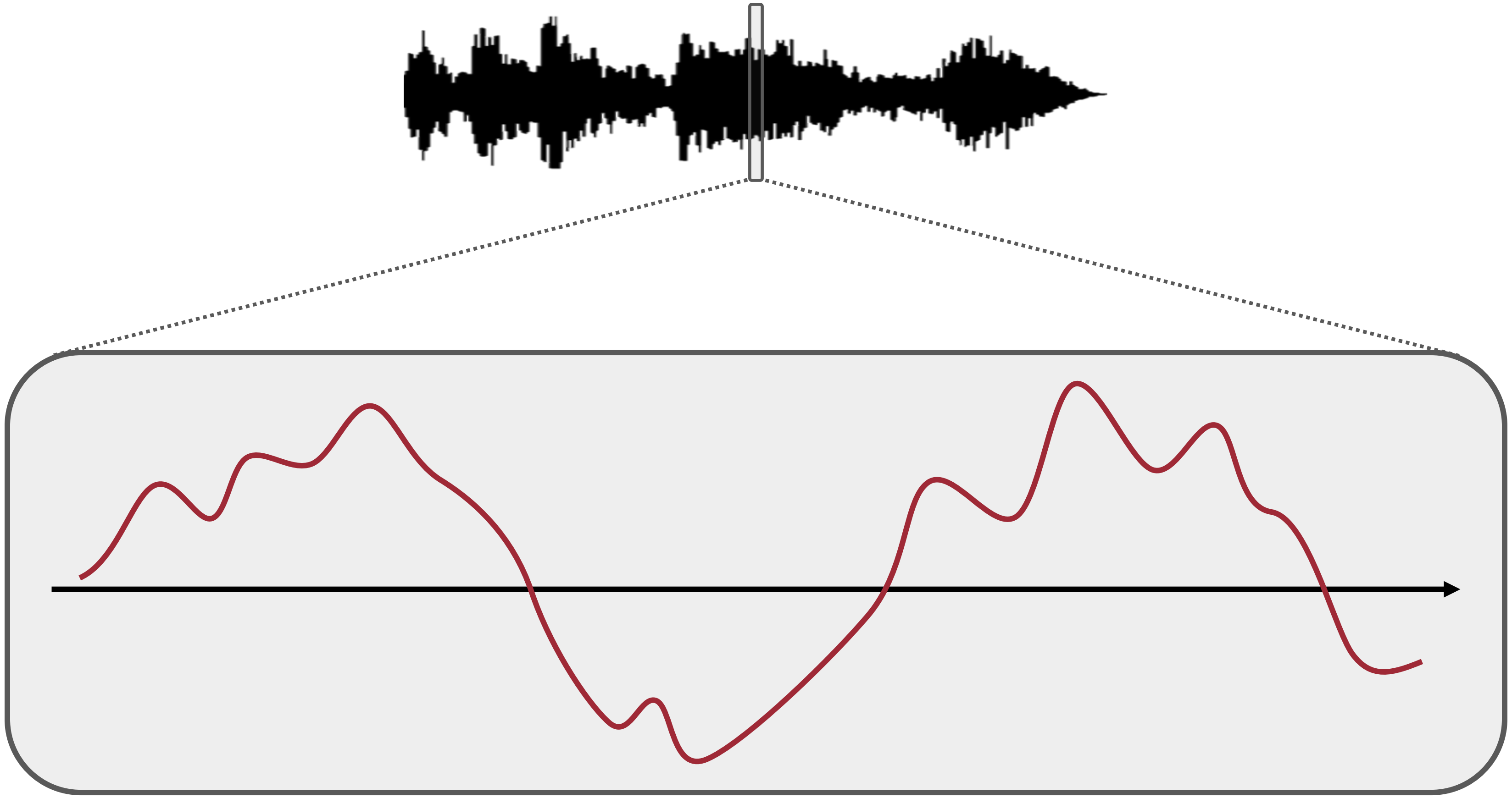
Digital Audio



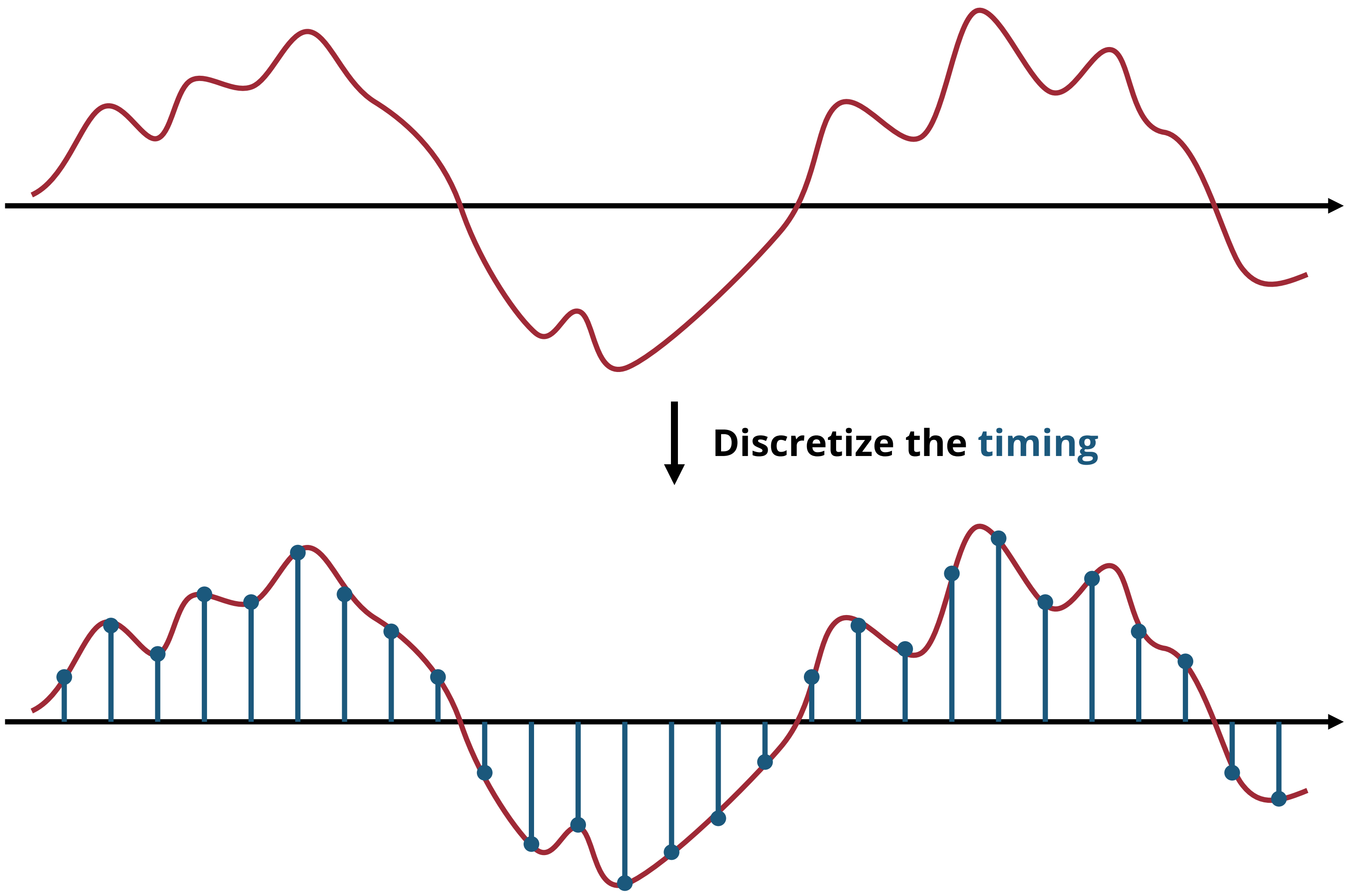
(Source: van den Oord et al., 2016)

Aaron van den Oord and Sander Dieleman, "WaveNet: A generative model for raw audio," *DeepMind Blog*, September 8, 2016.

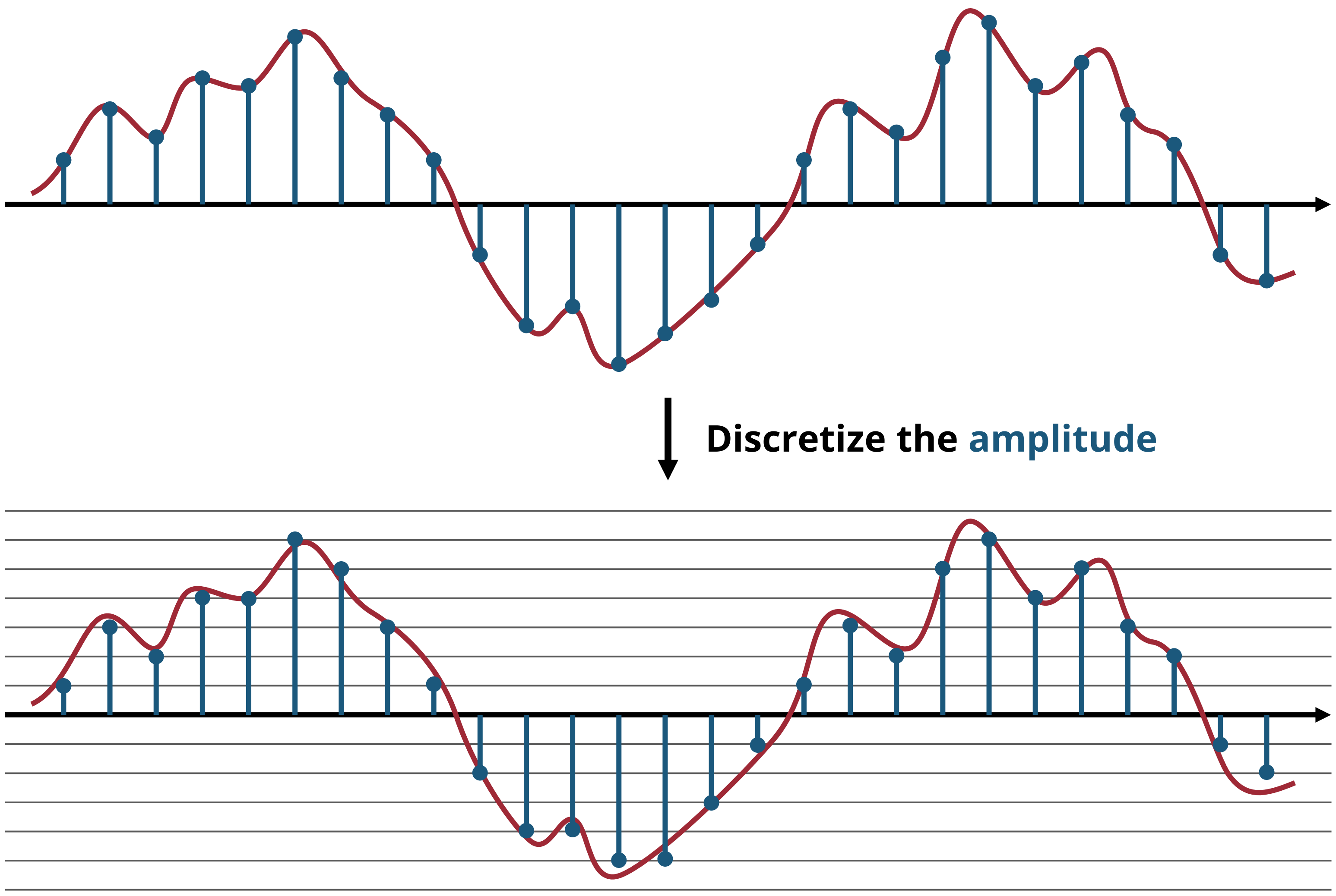
Waveform



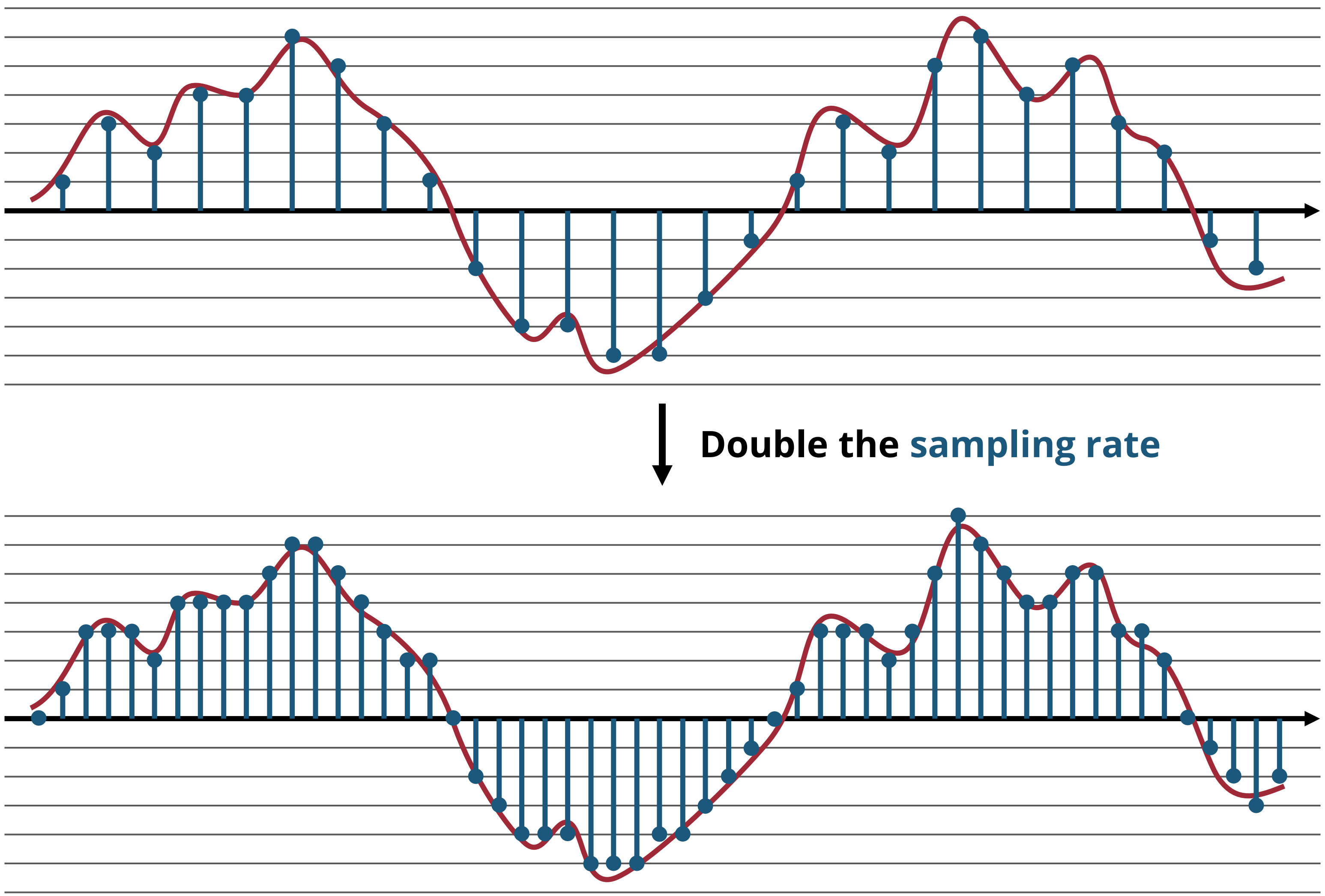
Digitalizing Audio: Timing



Digitalizing Audio: Amplitude



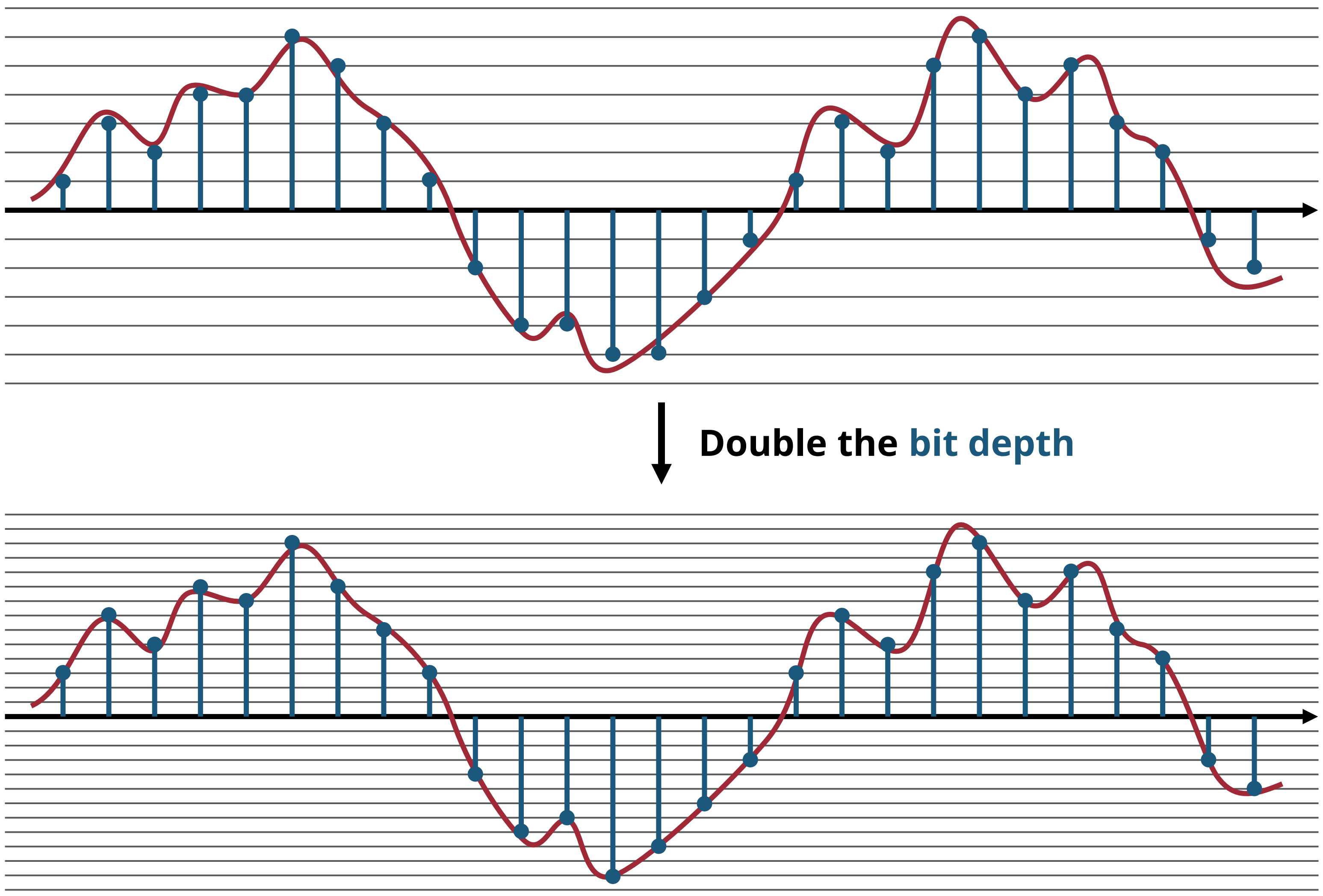
Resolution: Sampling Rate



Sampling Rate

- **Definition**: **Number of samples per second**
  - How many times the "sound pressure" is measured per second
  - The higher the sampling rate, the lower the timing distortion
- **Common sampling rates**
  - **Telephone**: 8 kHz
  - **CD**: 44.1 kHz
  - **DVD**: 48 kHz
  - **Modern audio interfaces & DAWs**: 96 kHz, 192 kHz

Resolution: Bit Depth



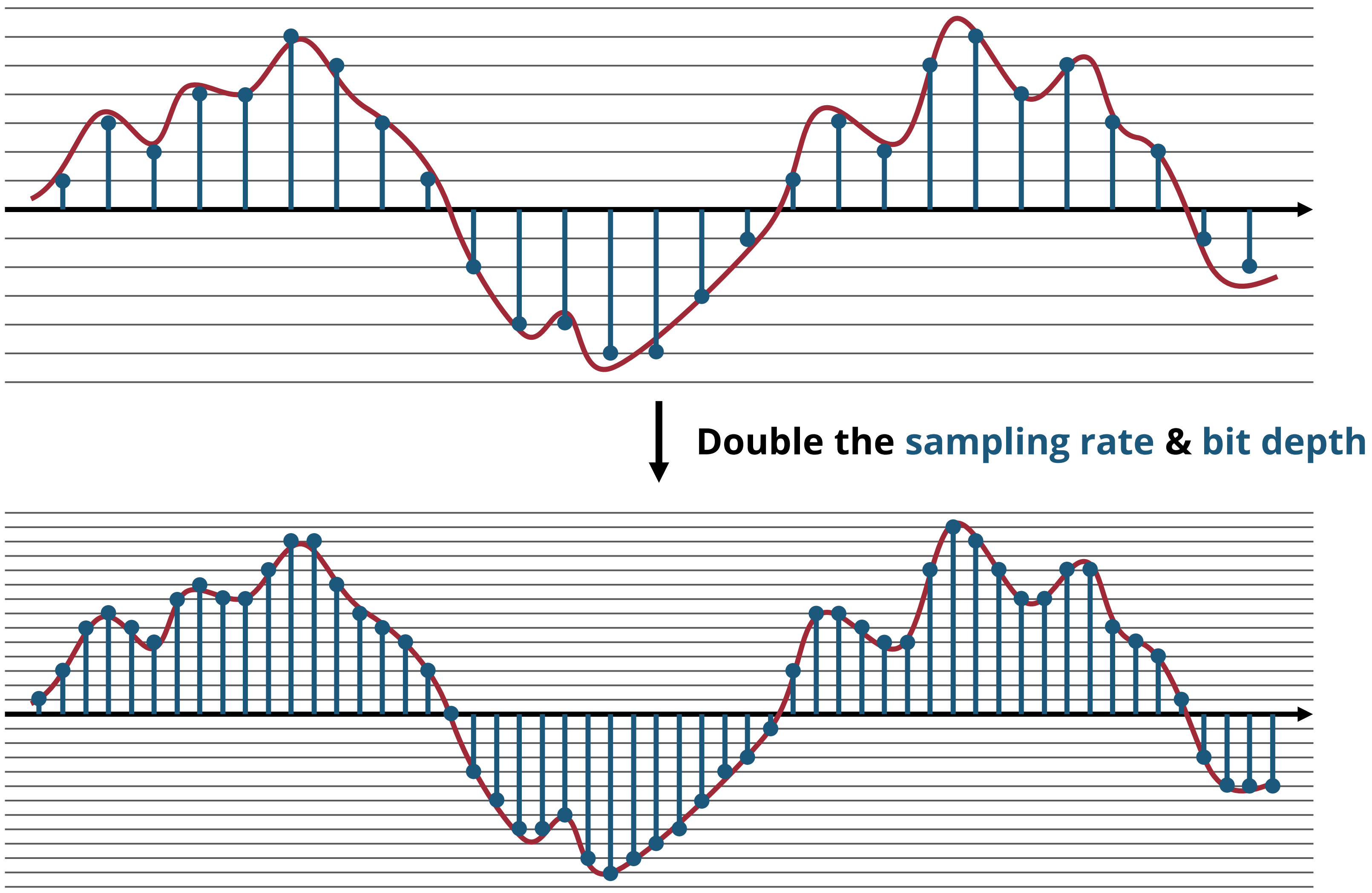
Bit Depth

- **Definition**: **Number of bits used to store each sample**
  - How many bits used to store the amplitude
  - The higher the bit depth, the lower the amplitude distortion
- **Common bit depth**
  - **Chiptunes**: 8 bit
  - **CD**: 16 bit
  - **Modern audio interfaces & DAWs**: 24 bit, 32 bit

Bit Depth

- **8 bit**: -128 to 127
- **16 bit**: -32,768 to 32,767
- **24 bit**: -8,388,608 to 8,388,607
- **32 bit**: 32-bit floating numbers

Resolution: Sampling Rate & Bit Depth



Bit Depth ≠ Bit Rate

- **Bit Depth**: **Number of bits used to store each sample**
  - Example: **CD quality** is **16bit/44.1kHz**

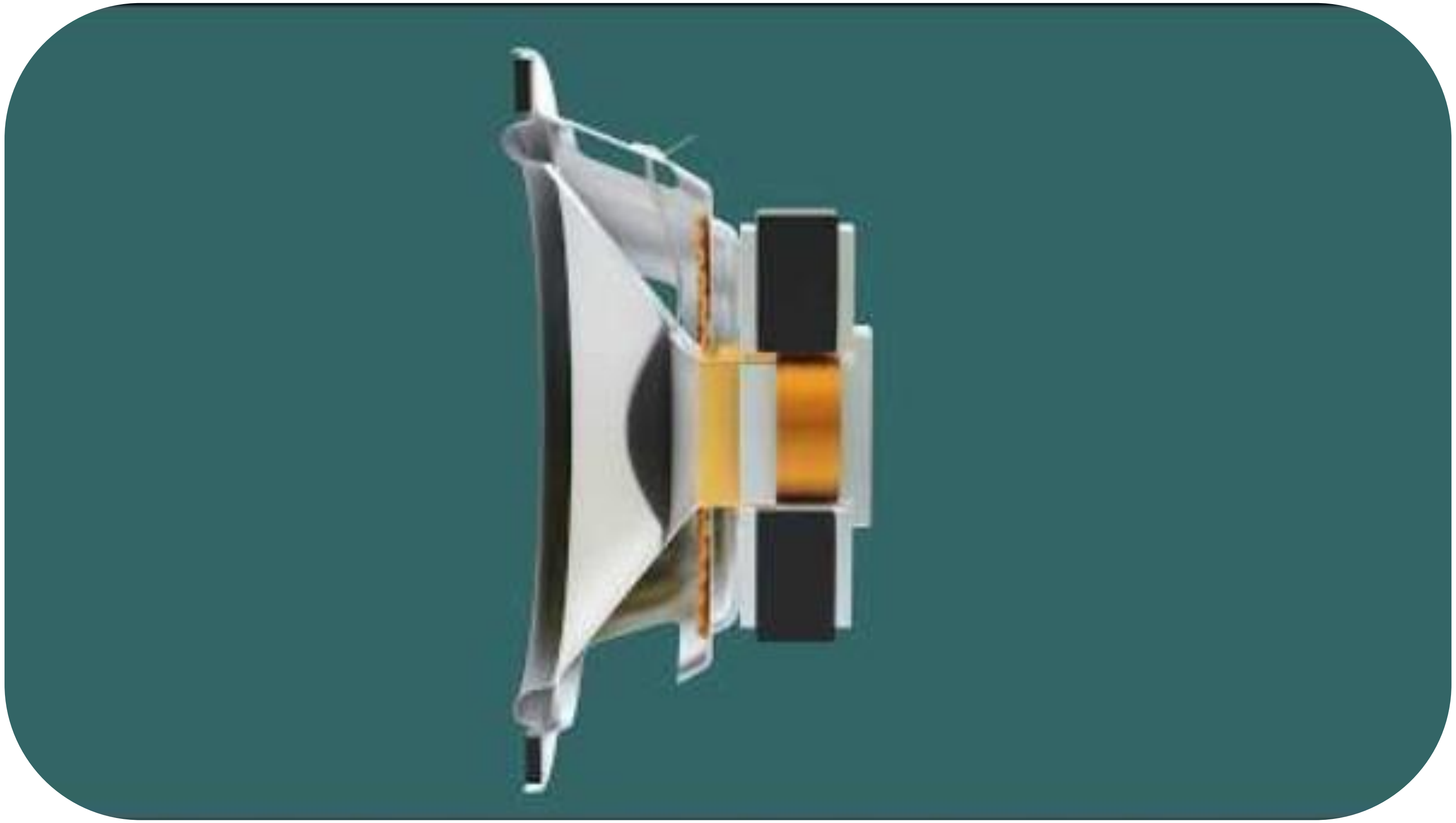
- **Bit Rate**: **Amount of data transferred per second** (unit: bits/sec)
  - Example: **320K MP3** files → **320kbps** (320,000 bits per second)
  - Example: **YouTube** recommendation → **128 kbps** for mono and **384 kbps** for stereo
  - Determines the file size!

📖 Reading: Microphones: Measuring Sound Pressure

youtu.be/d_crXXbuEKE

Into The Ordinary, "How do microphones work? - Recording Explained," *YouTube*, June 24, 2015.

📖 Reading: Speakers: Reproducing Sound Pressure



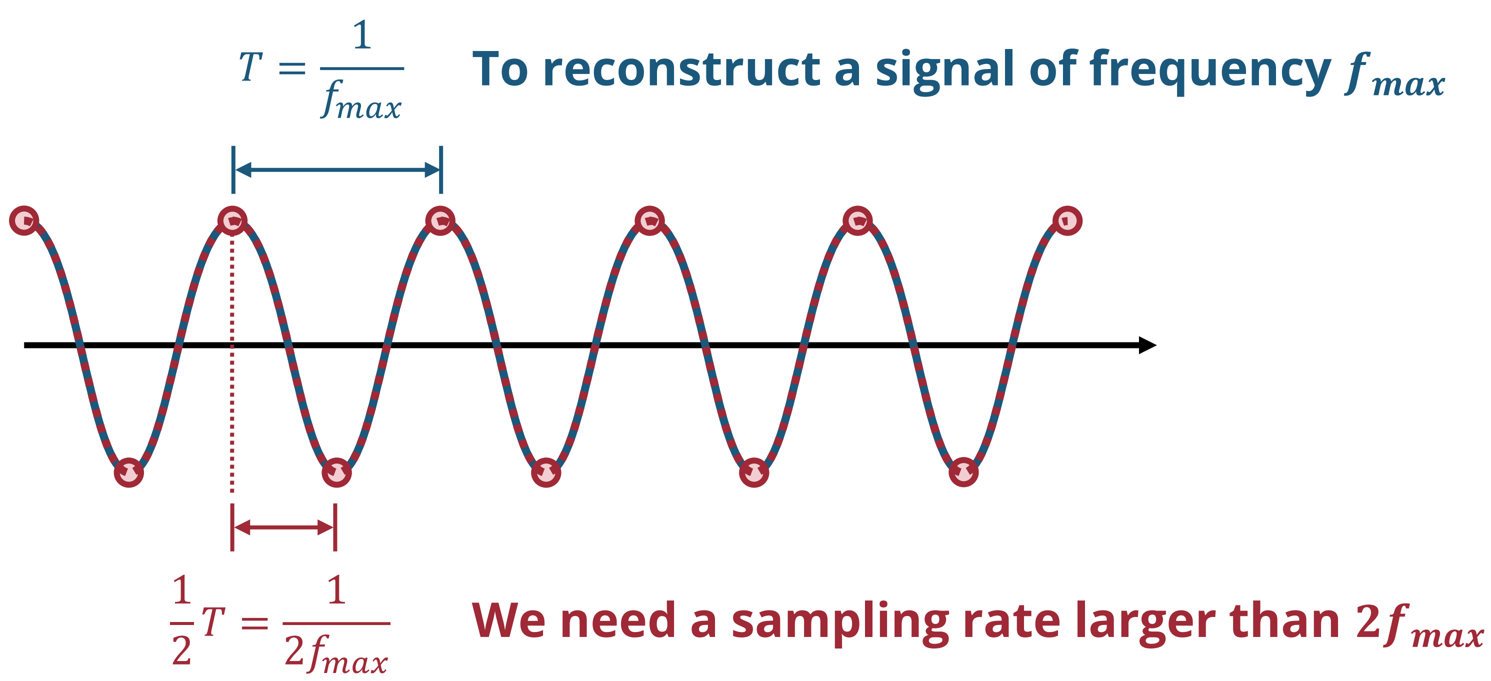
youtu.be/RxdFP31QYAg

Animagraffs, "How Speakers Make Sound," *YouTube*, December 16, 2020.

Sampling Theorem

Nyquist–Shannon Sampling Theorem

- **Theorem**: If a signal contains no frequencies higher than f_{max}, then the signal can be perfectly reconstructed when sampled at a rate $f_s > 2f_{max}$
  - $2f_{max}$ is usually referred to as the **Nyquist rate**

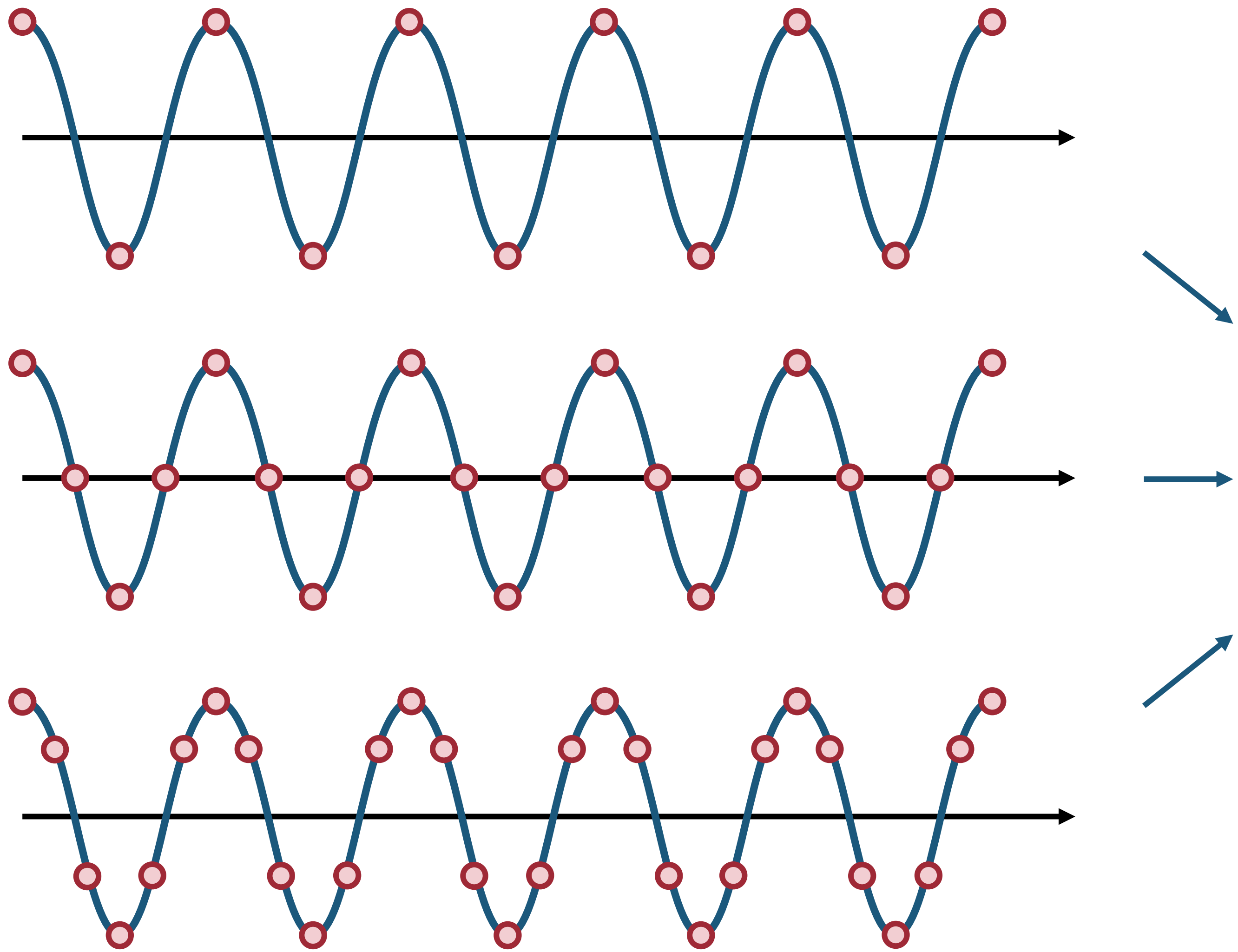


Sampling Theorem: Oversampling

Critically sampled
$(f_s = 2f_{max})$

Oversampled
$(f_s = 4f_{max})$

Oversampled
$(f_s = 6f_{max})$



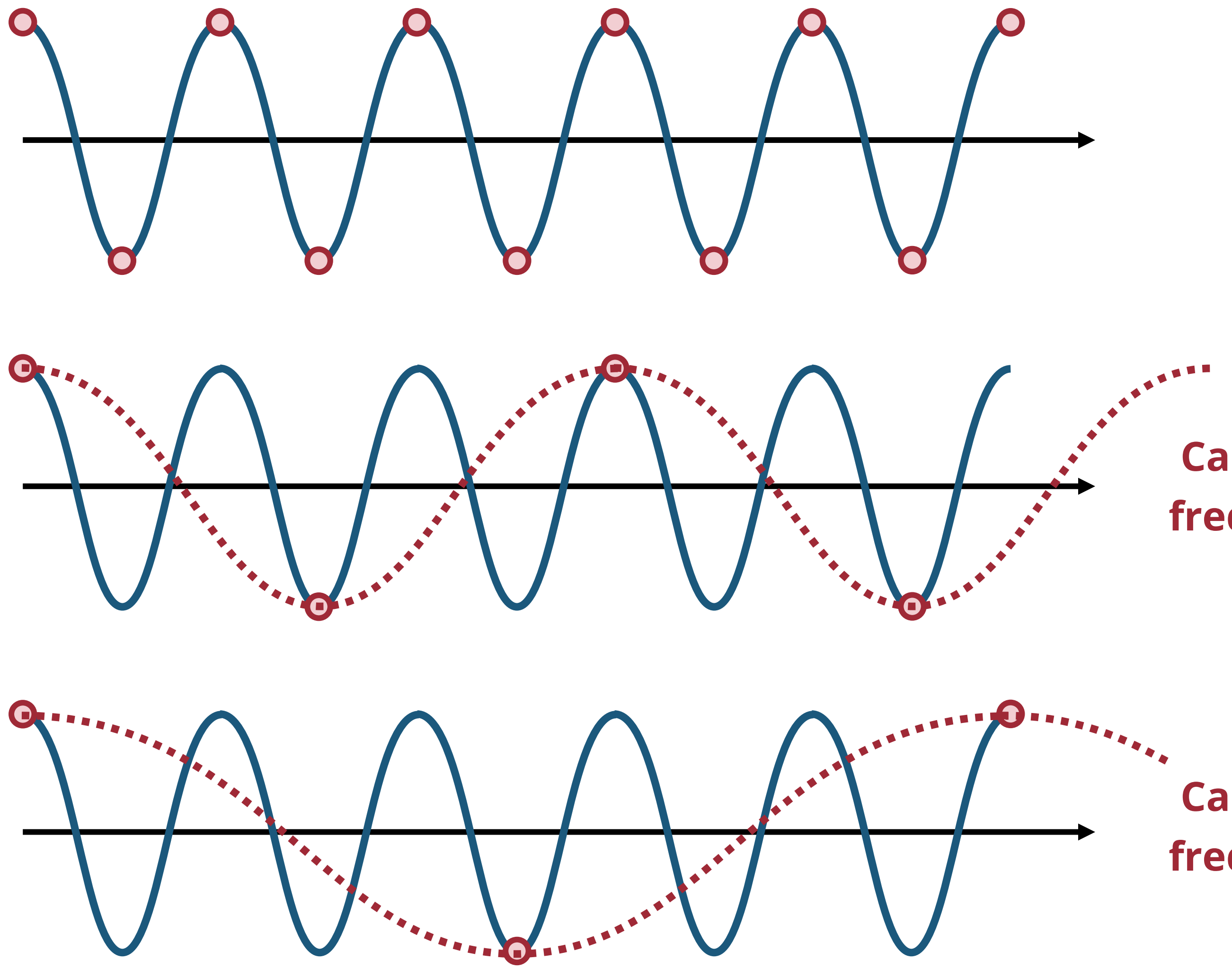
Reconstruction is possible!

Sampling Theorem: Undersampling

Critically sampled
$(f_s = 2f_{max})$

Undersampled
$(f_s = \frac{2}{3}f_{max})$

Undersampled
$(f_s = \frac{2}{5}f_{max})$



Can only reconstruct frequency up to $\frac{1}{3}f_{max}$

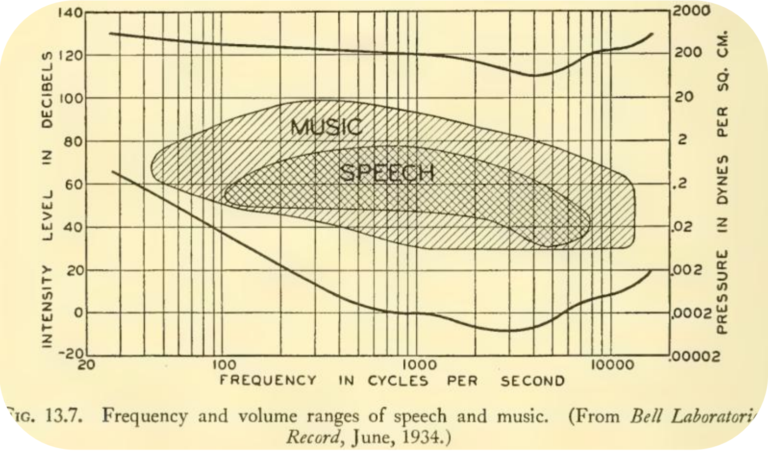
Can only reconstruct frequency up to $\frac{1}{3}f_{max}$

🤔 Sampling Theorem

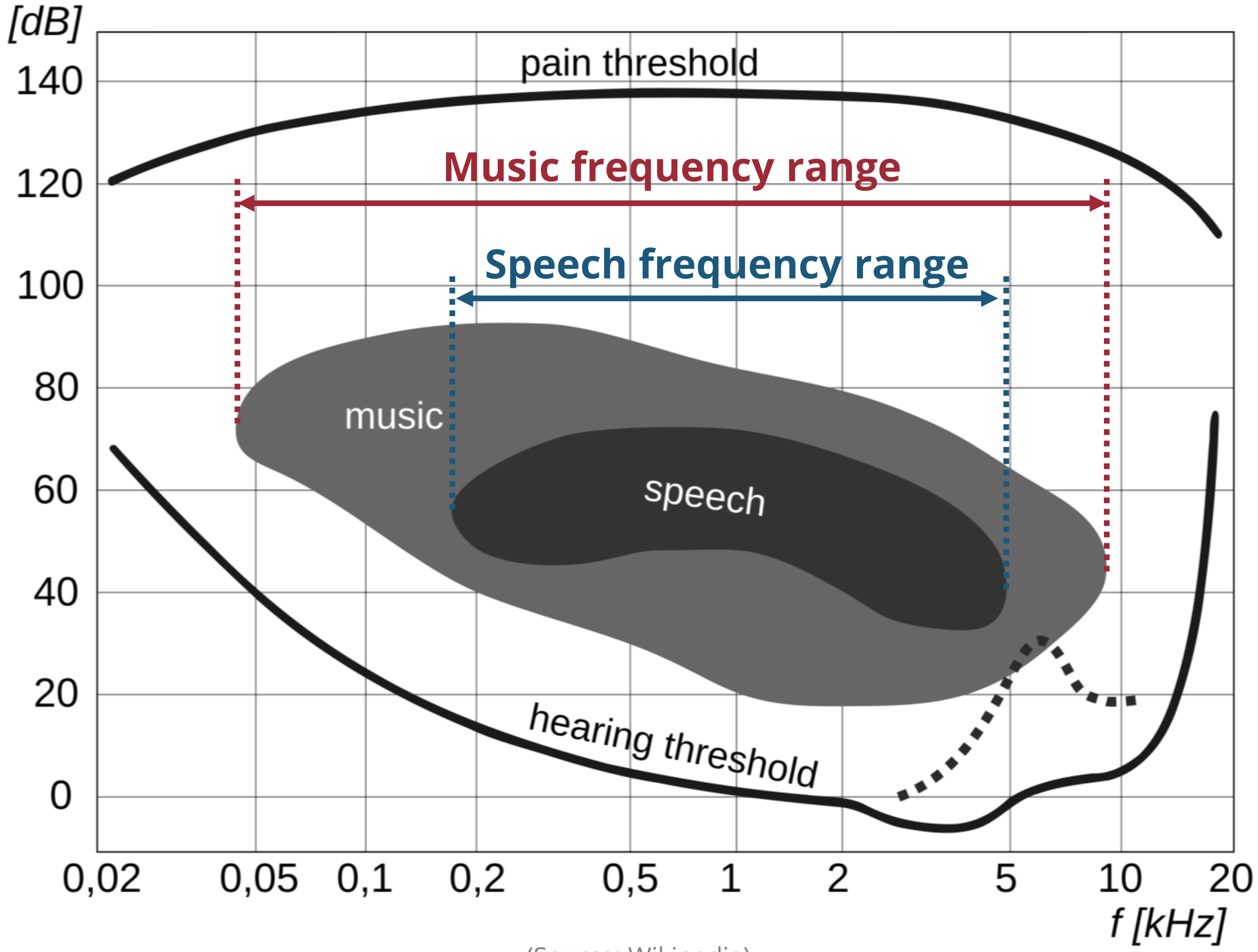
- **Telephone audio** is sampled at **8 kHz**. What is the maximum frequency it can reconstruct?
  - **4 kHz**

- To cover the **human hearing range of 20 Hz to 20 kHz**, what is the minimum sampling rate required?
  - **40 kHz**

Sampling Rate & Frequency Range



(Source: Bell Laboratories Record 1934 & Olson 1947)



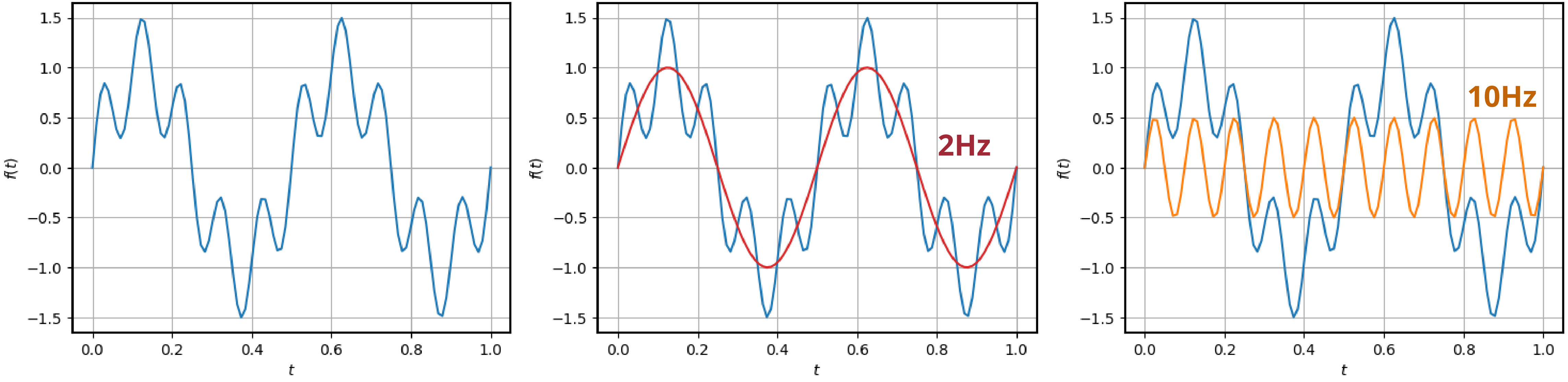
(Source: Wikipedia)

Bell Laboratories Record, 12(6):314, 1934.
Harry Ferdinand Olson, "Speech, Music and Hearing," *Elements of acoustical engineering Hardcover*, p. 326, 1947.
en.wikipedia.org/wiki/Hearing_range

Spectral Analysis

Spectral Analysis

- **Goal**: Analyze the **frequency components** of a signal

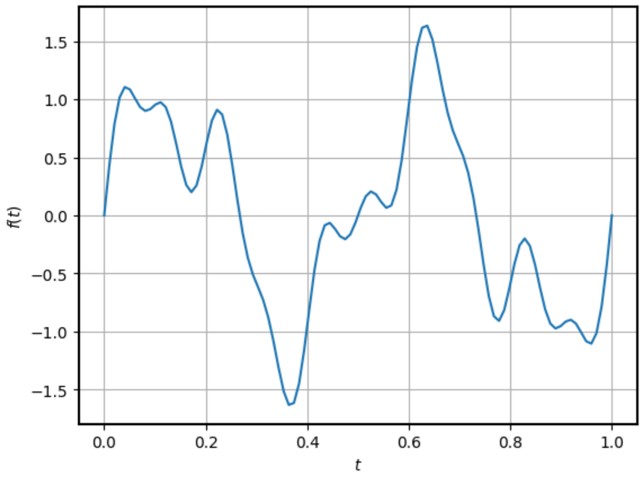


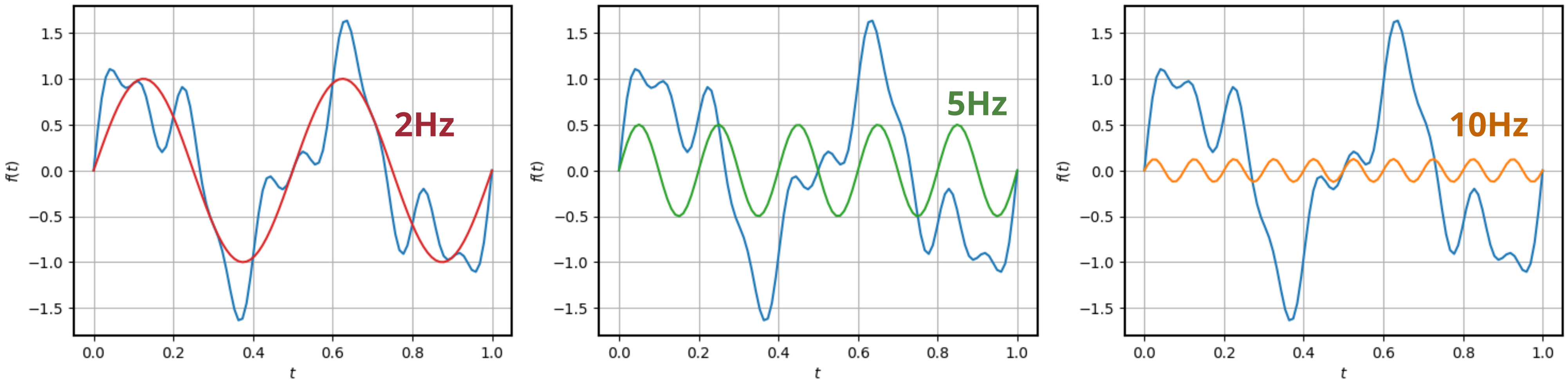
$$\sin(2 \cdot 2\pi t) + \frac{1}{2}\sin(10 \cdot 2\pi t)$$

$$\sin(2 \cdot 2\pi t)$$

$$\frac{1}{2}\sin(10 \cdot 2\pi t)$$

Spectral Analysis

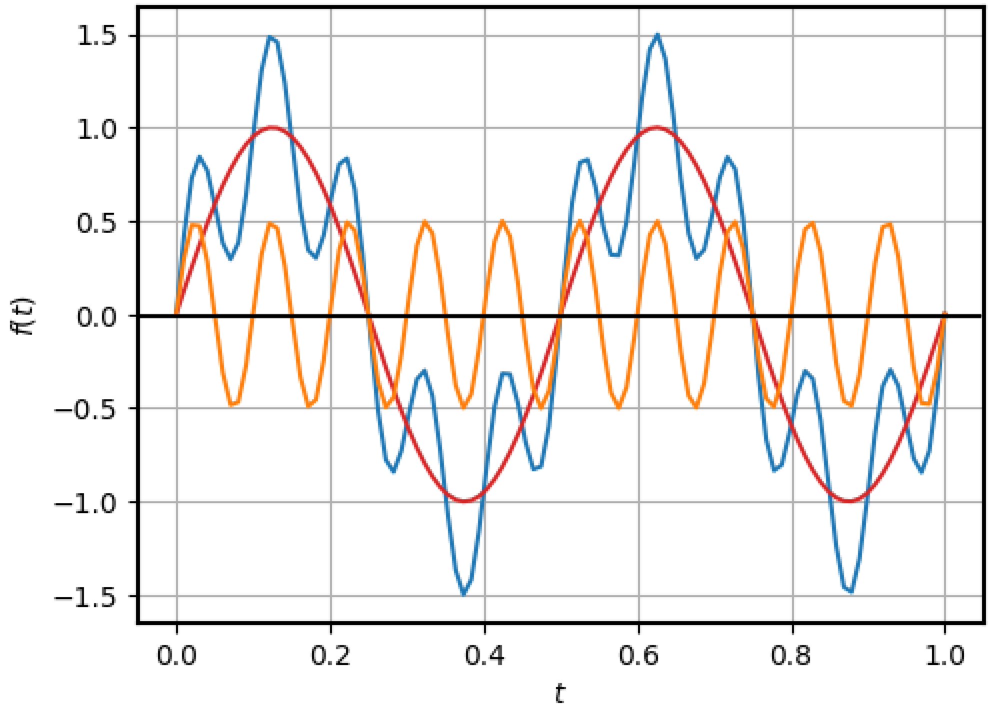




Fourier Transform

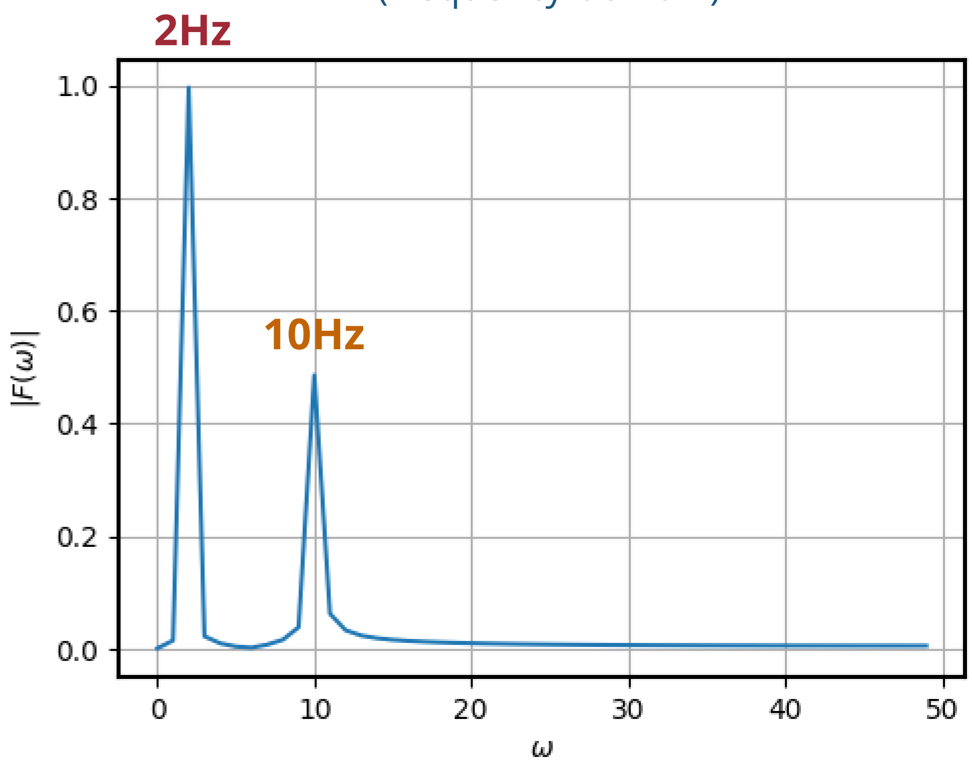
Signal

(time-domain)



Spectrum

(frequency-domain)



Fourier Transform

- **Intuition**: Decompose time-domain signals into **frequency components**
- Math formulation:

Output spectrum

Input signal

$$F(\omega) = \int_{-\infty}^{\infty} f(t)\, e^{-j\omega t}\, dt$$

Frequency

Sum over all t

Demystifying Fourier Transform

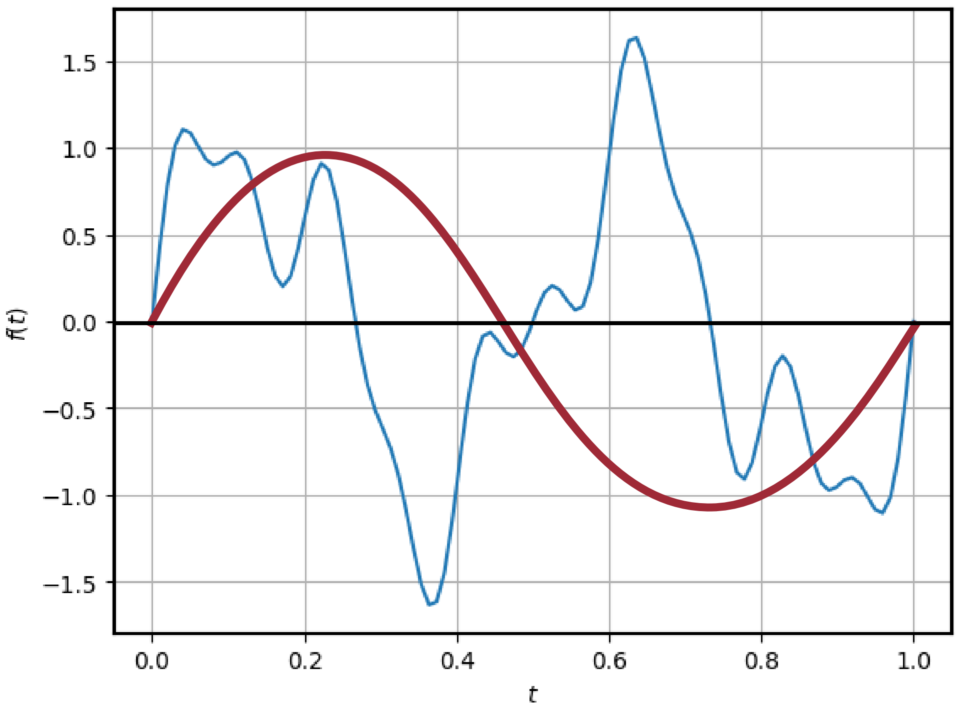
$$F(\omega) = \int_{-\infty}^{\infty} f(t)\, \boxed{e^{-j\omega t}}\, dt$$

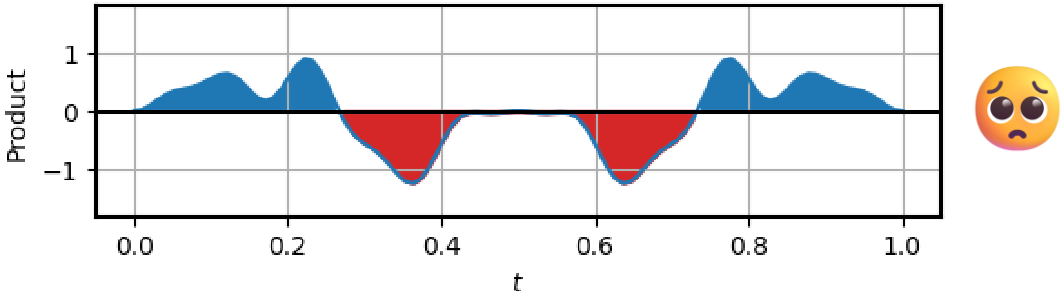
Euler's formula

$$e^{-j\theta} = \cos\theta + j\sin\theta$$

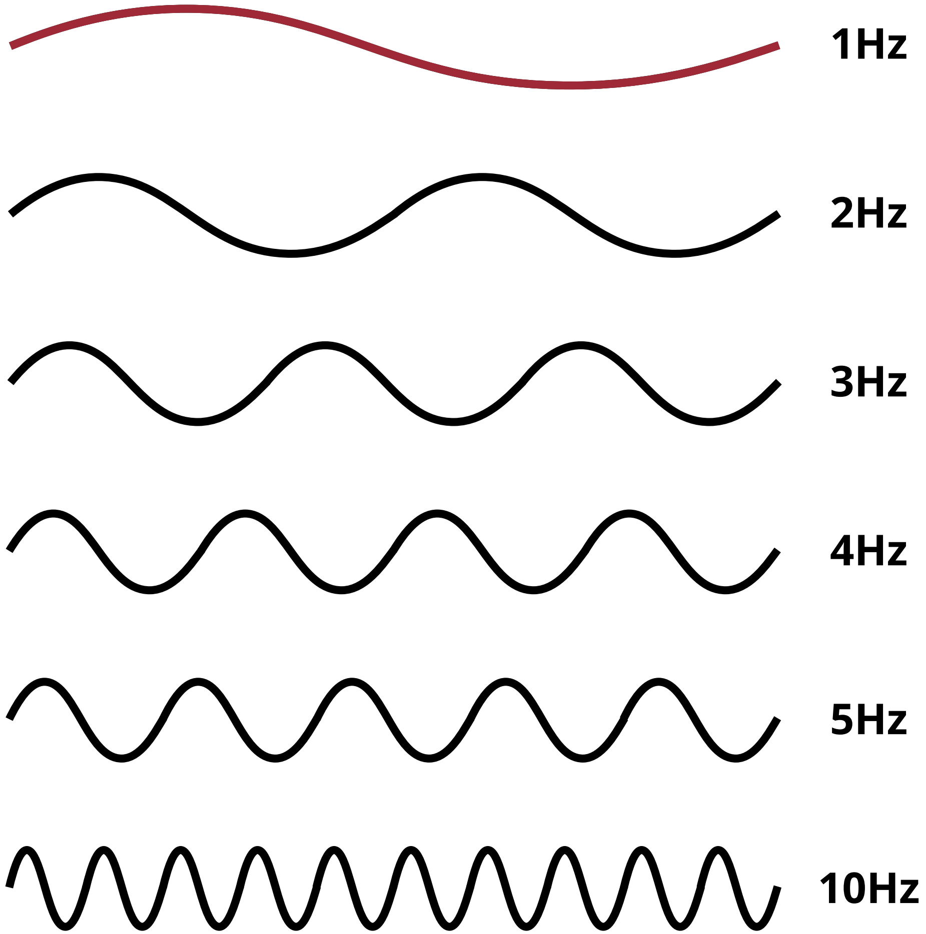
$$F(\omega) = \int_{-\infty}^{\infty} f(t)\boxed{\cos(-\omega t)} + j\, f(t)\boxed{\sin(-\omega t)}\, dt$$

Demystifying Fourier Transform

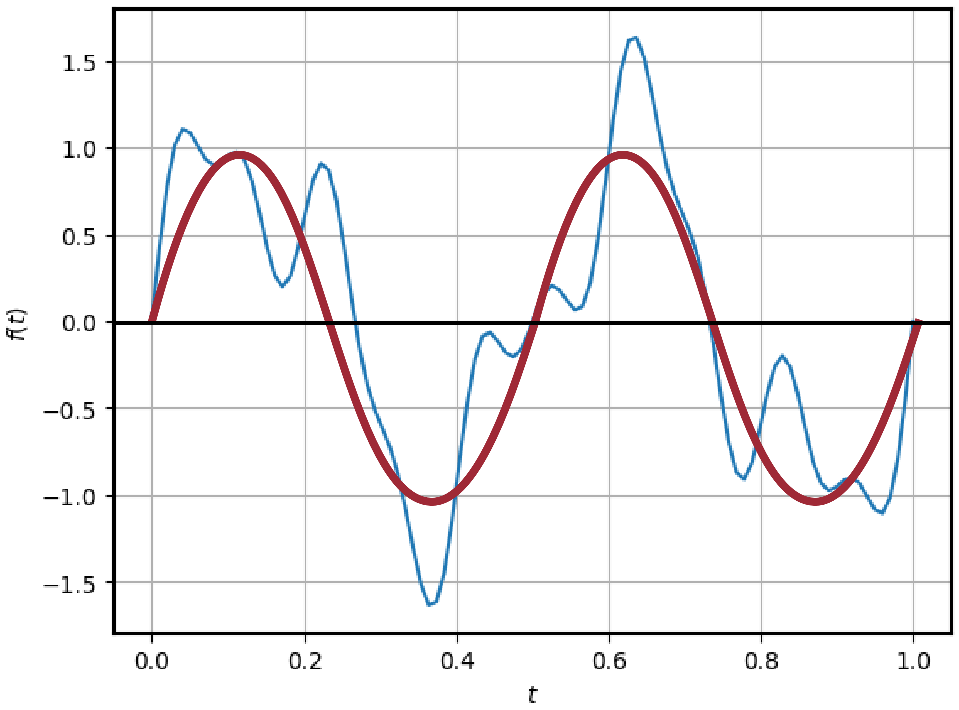


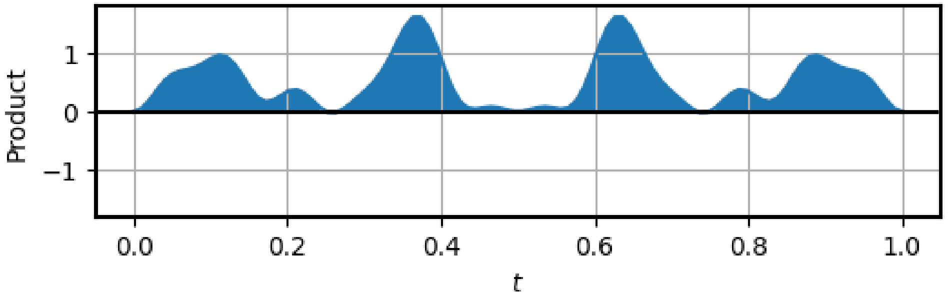


Candidate frequency components

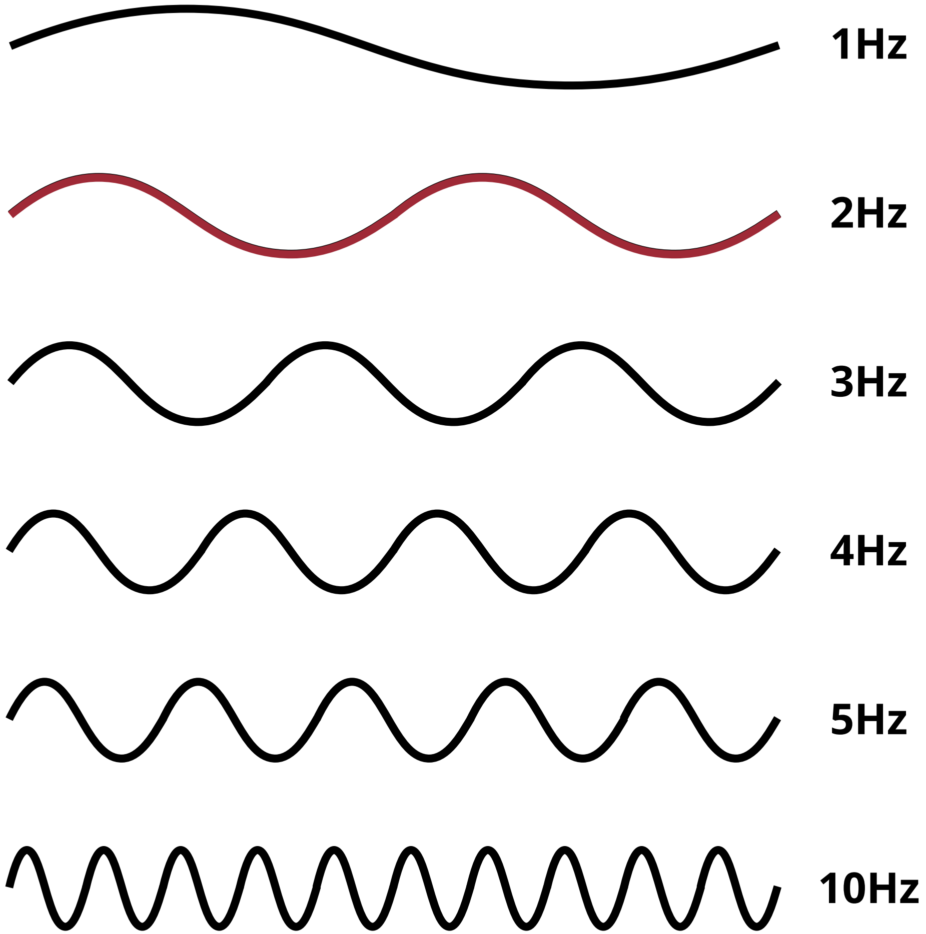


Demystifying Fourier Transform

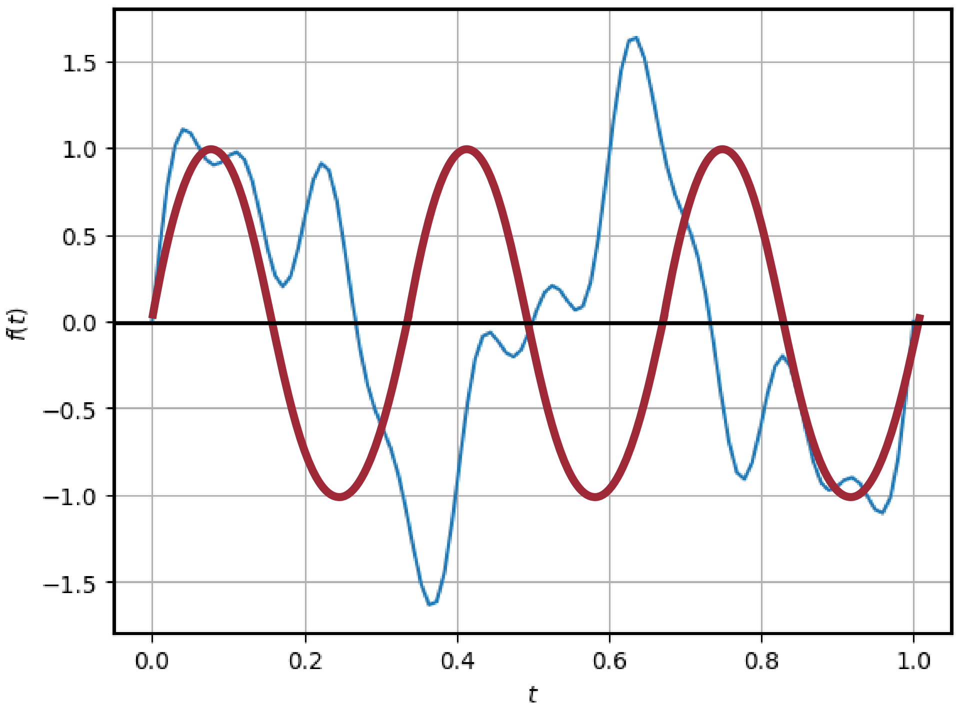


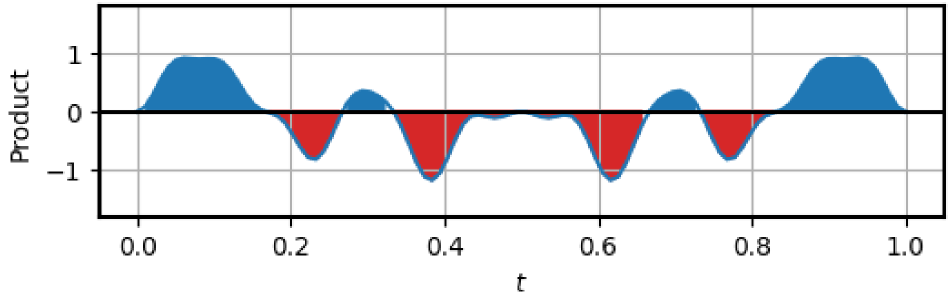


Candidate frequency components

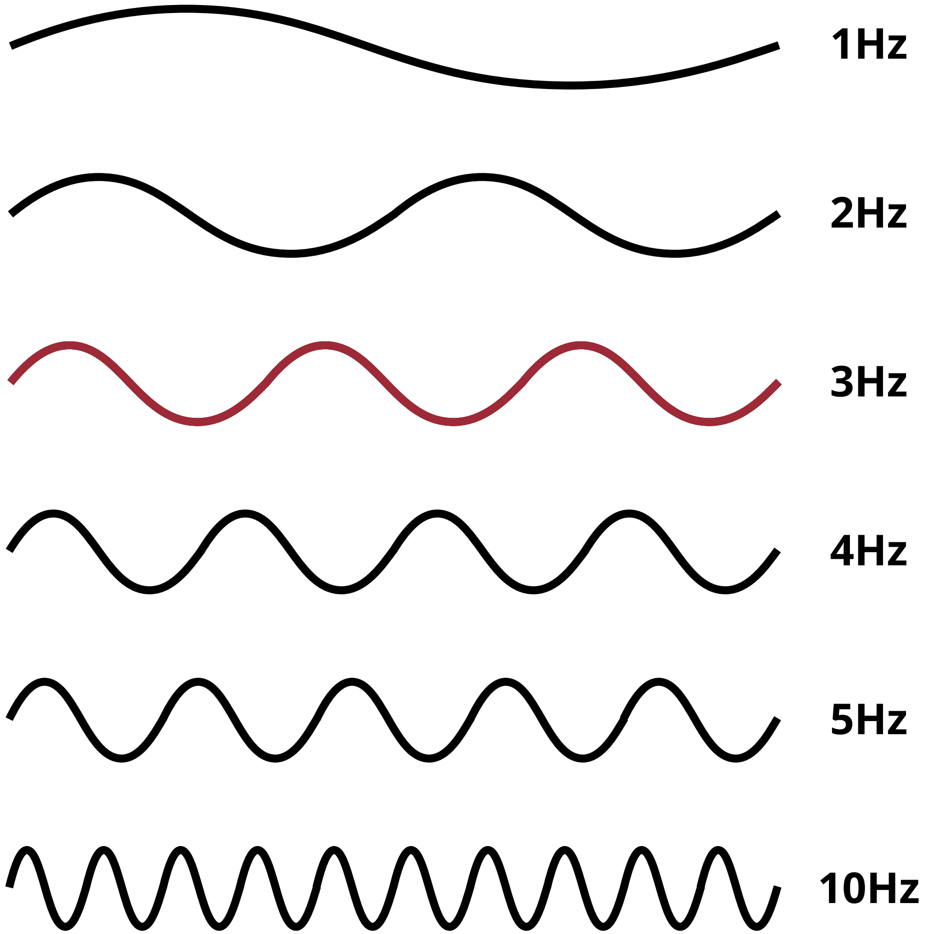


Demystifying Fourier Transform

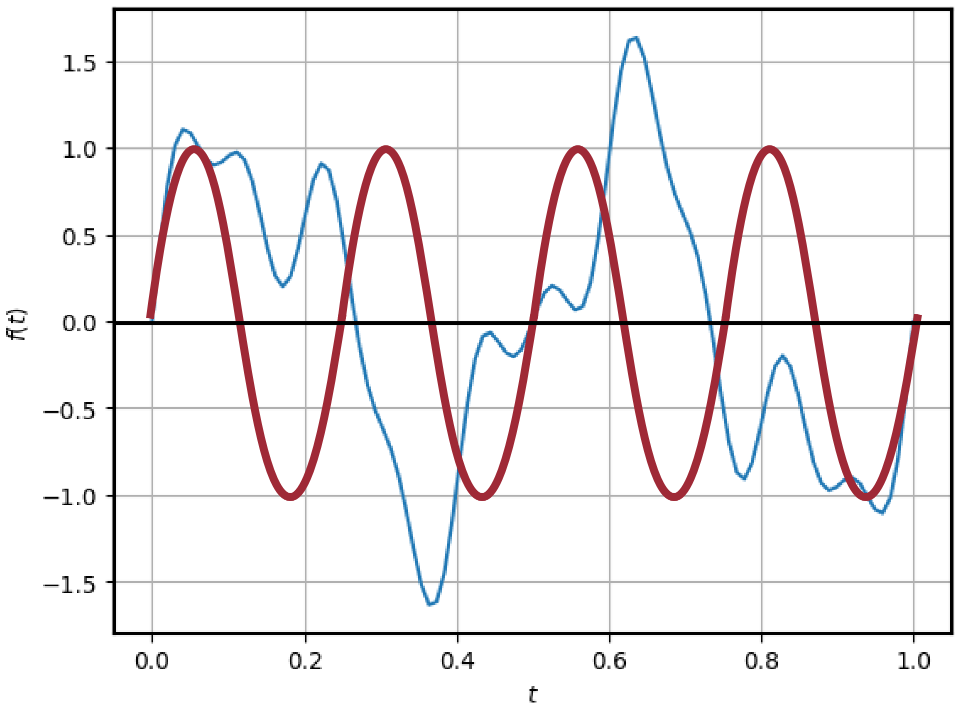


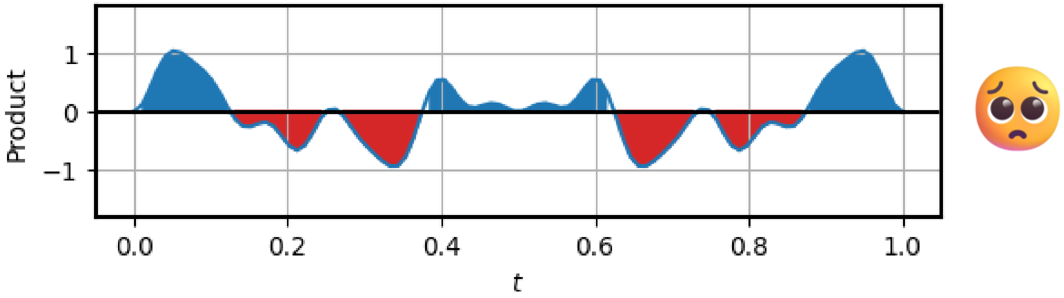


Candidate frequency components

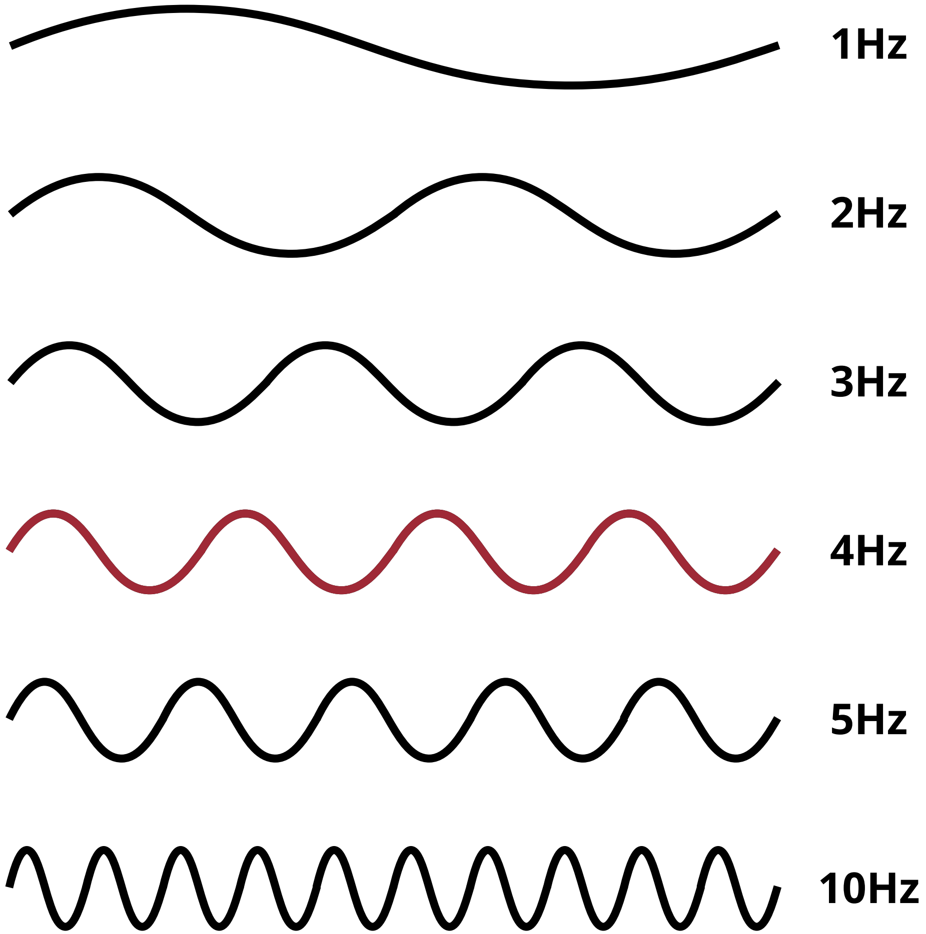


Demystifying Fourier Transform

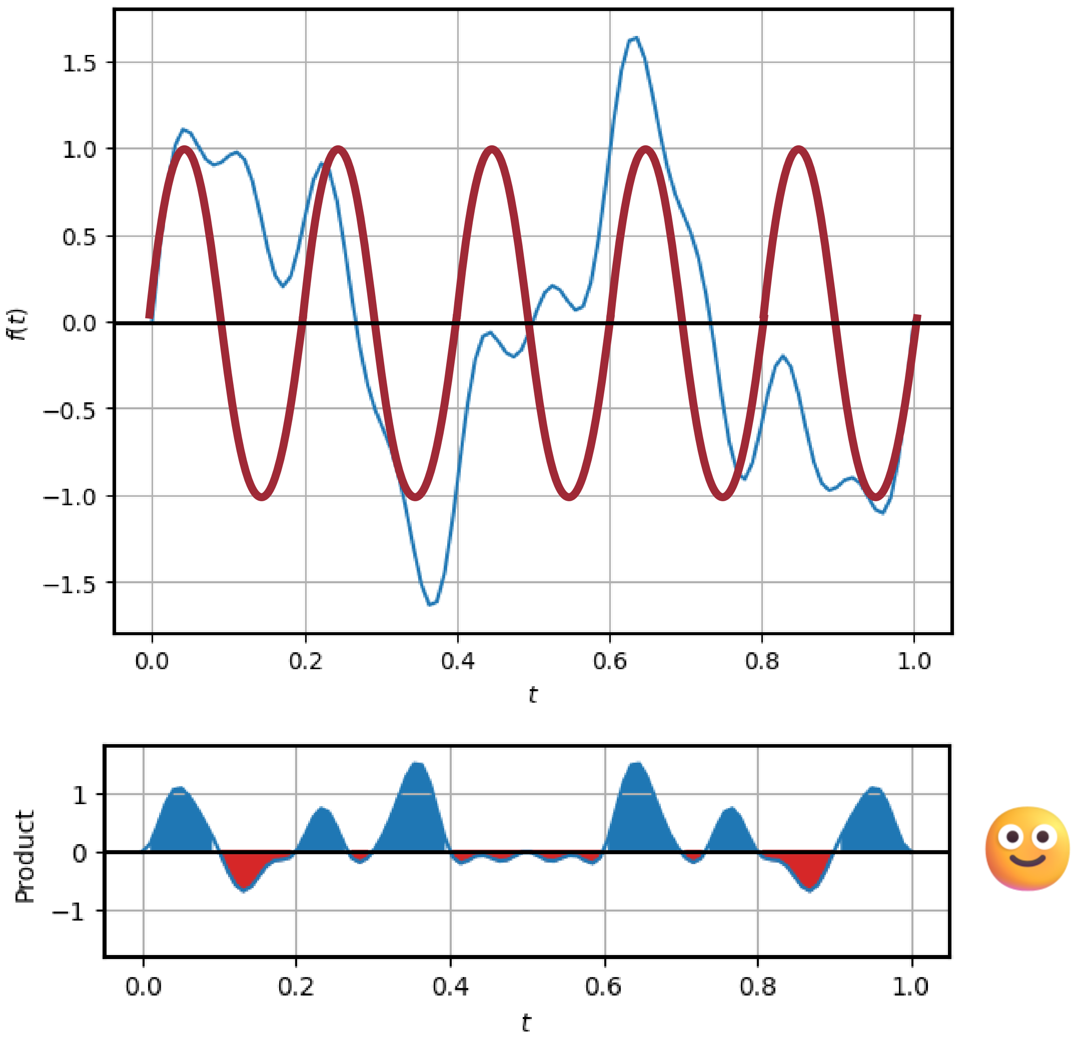




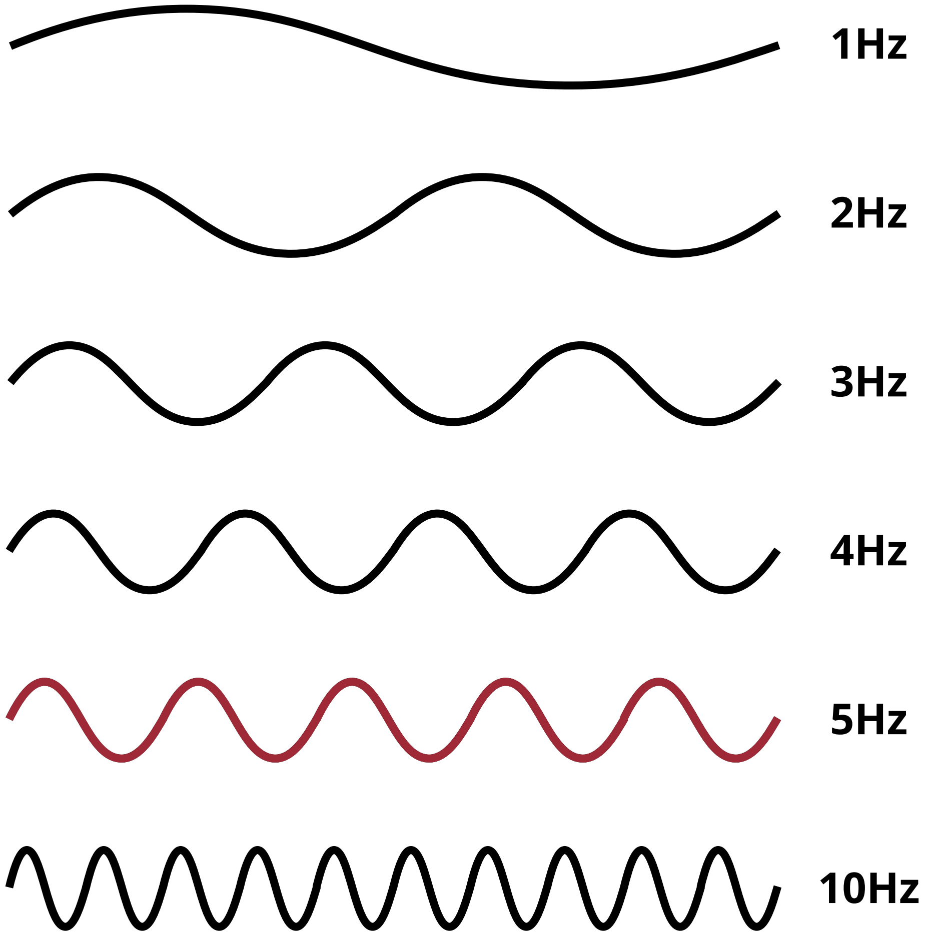
Candidate frequency components



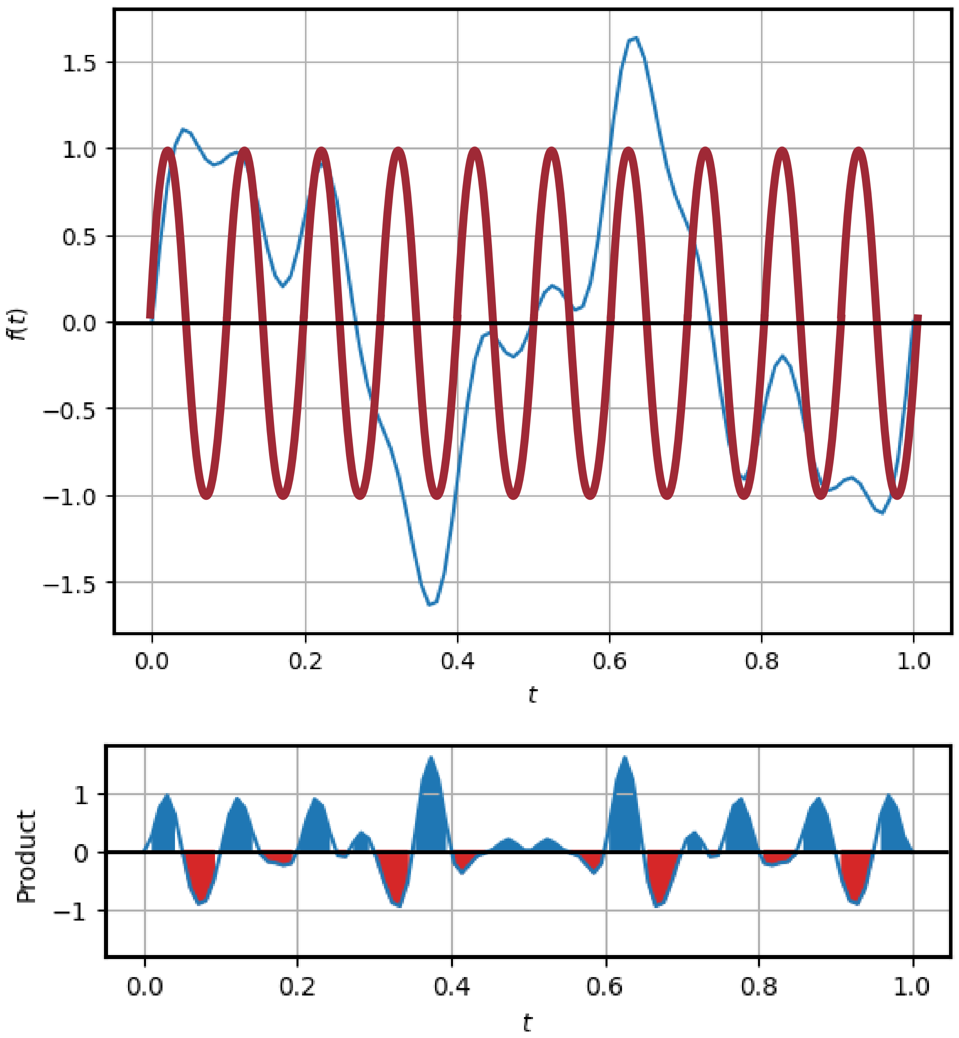
Demystifying Fourier Transform



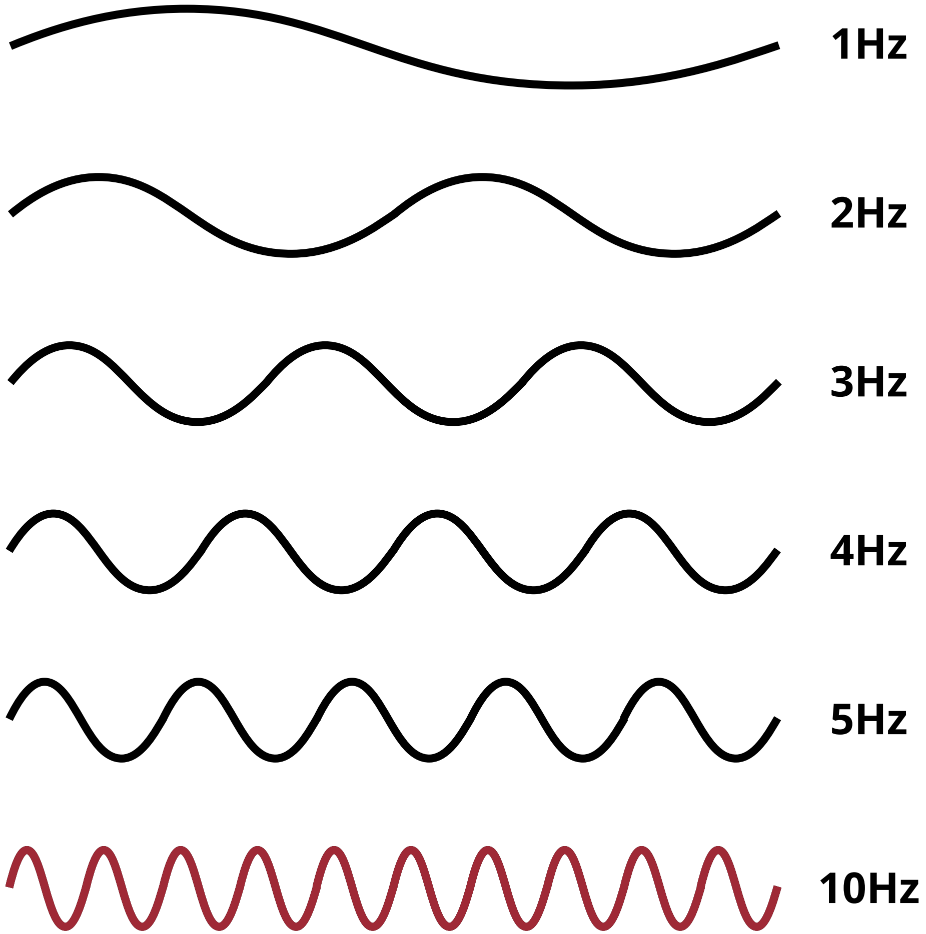
Candidate frequency components



Demystifying Fourier Transform



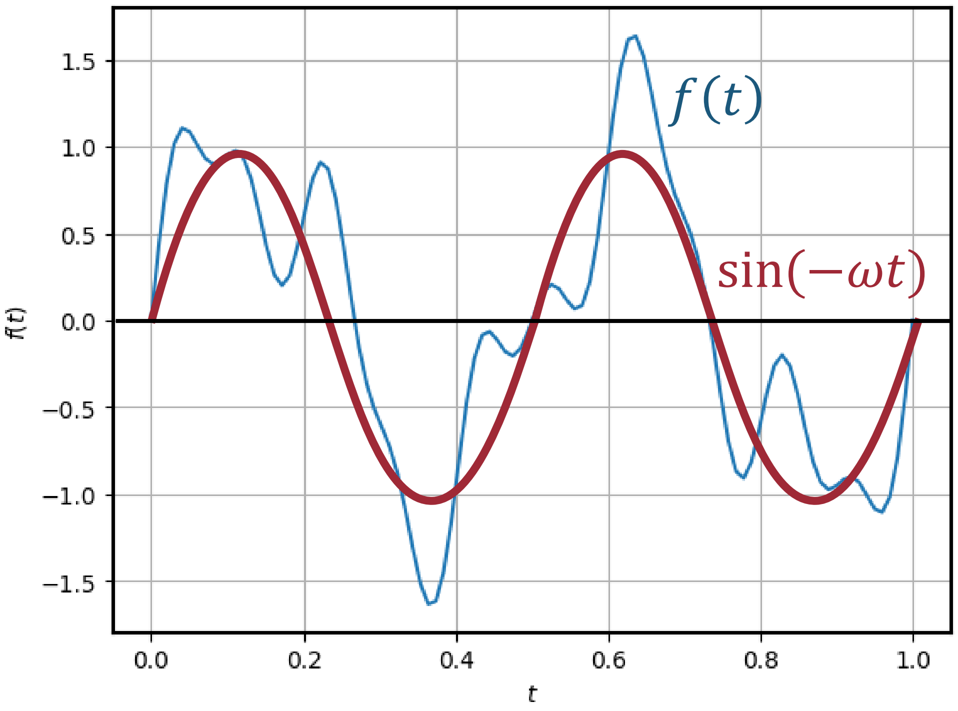
Candidate frequency components



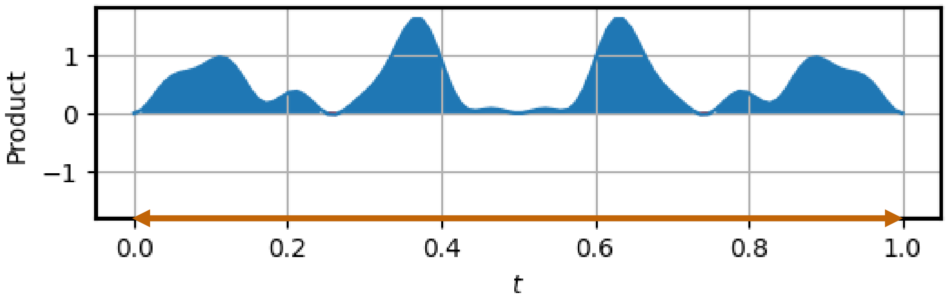
Demystifying Fourier Transform

$$F(\omega) = \int_{-\infty}^{\infty} f(t)\cos(-\omega t) + j\, f(t)\sin(-\omega t)\, dt$$

Sum over all t



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

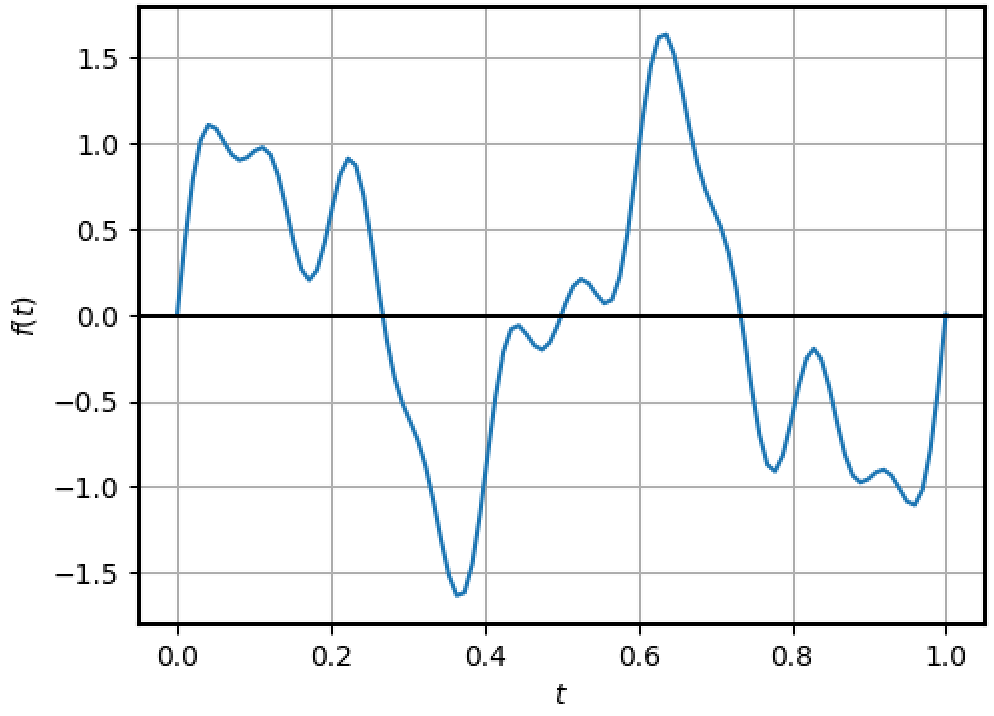


Sum = 0.495

Demystifying Fourier Transform

Signal

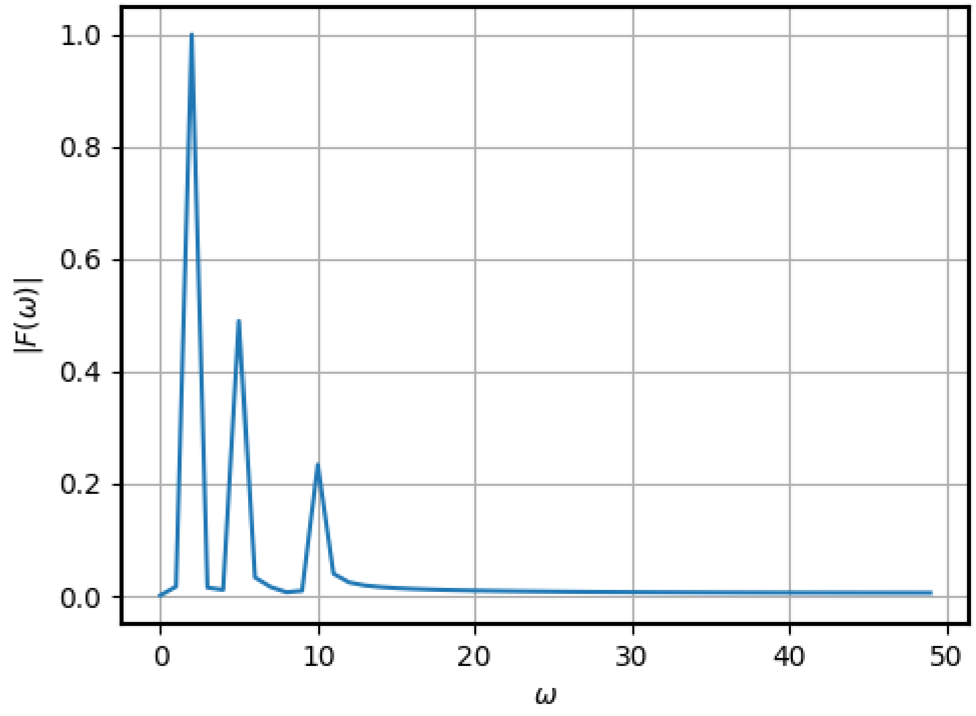
(time-domain)



Fourier Transform →

Spectrum

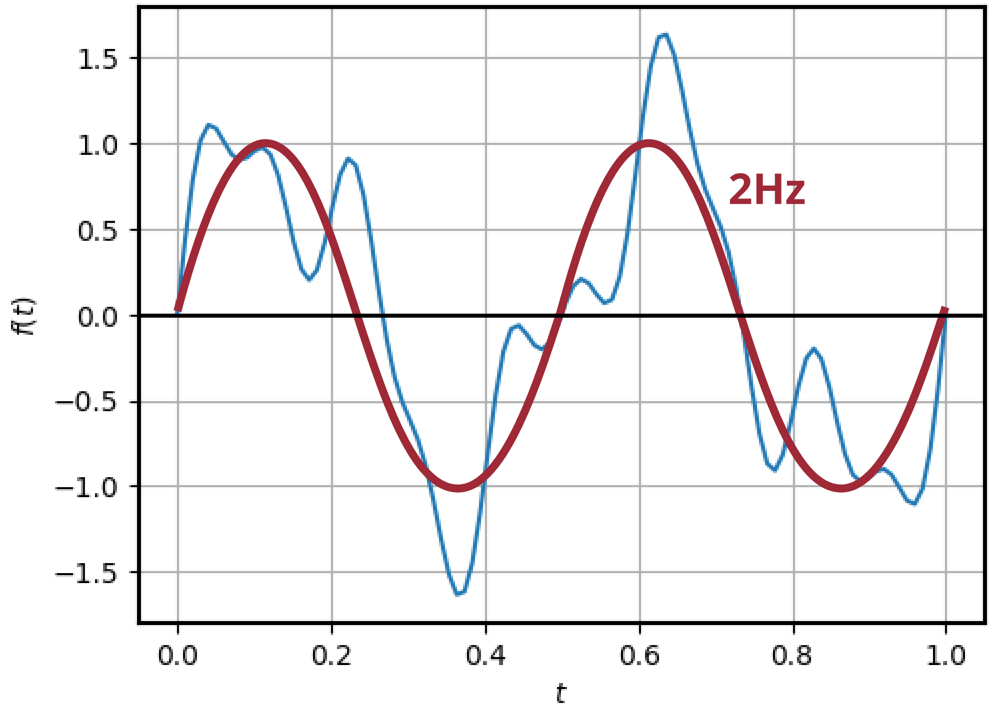
(frequency-domain)



Demystifying Fourier Transform

Signal

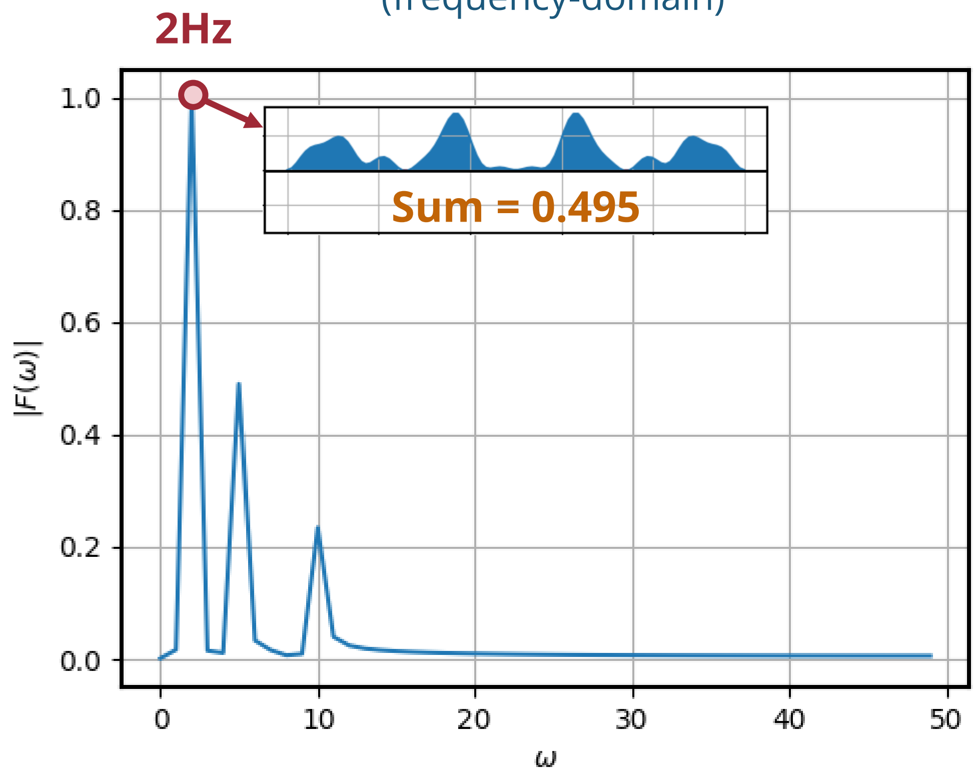
(time-domain)



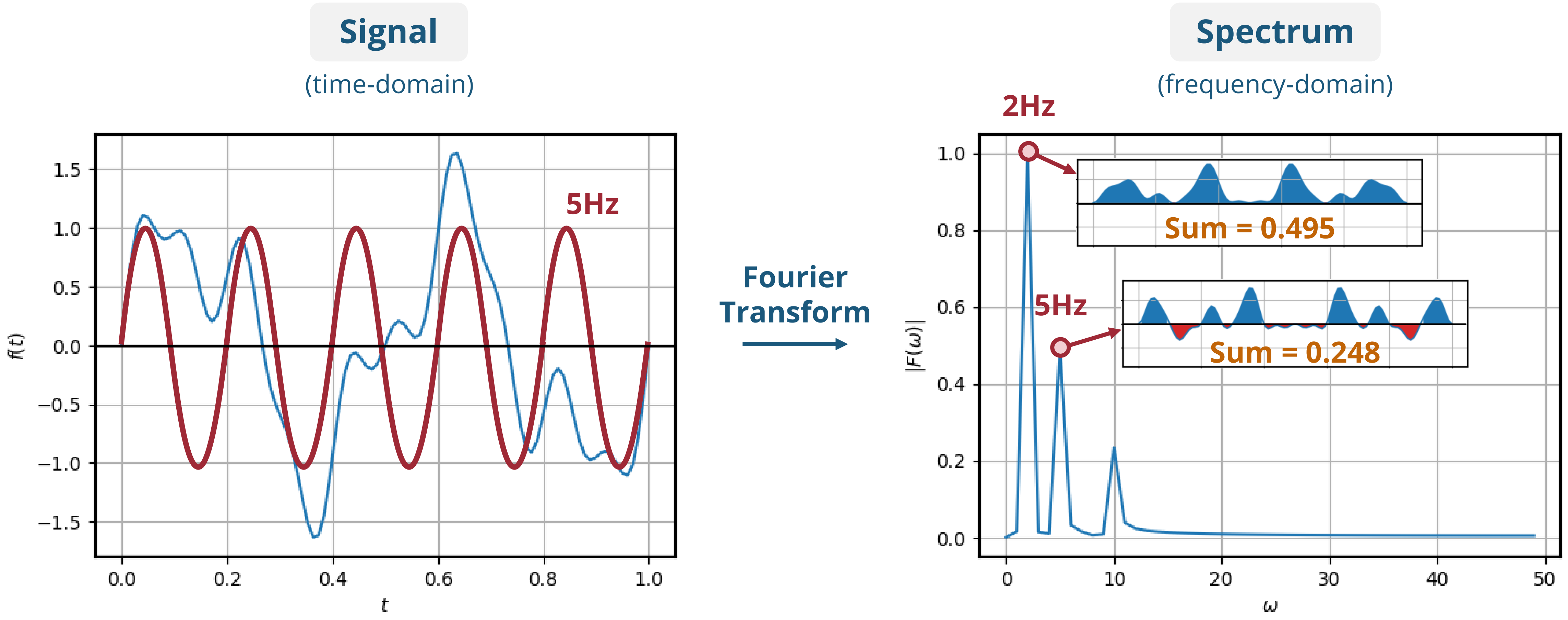
Fourier Transform

Spectrum

(frequency-domain)



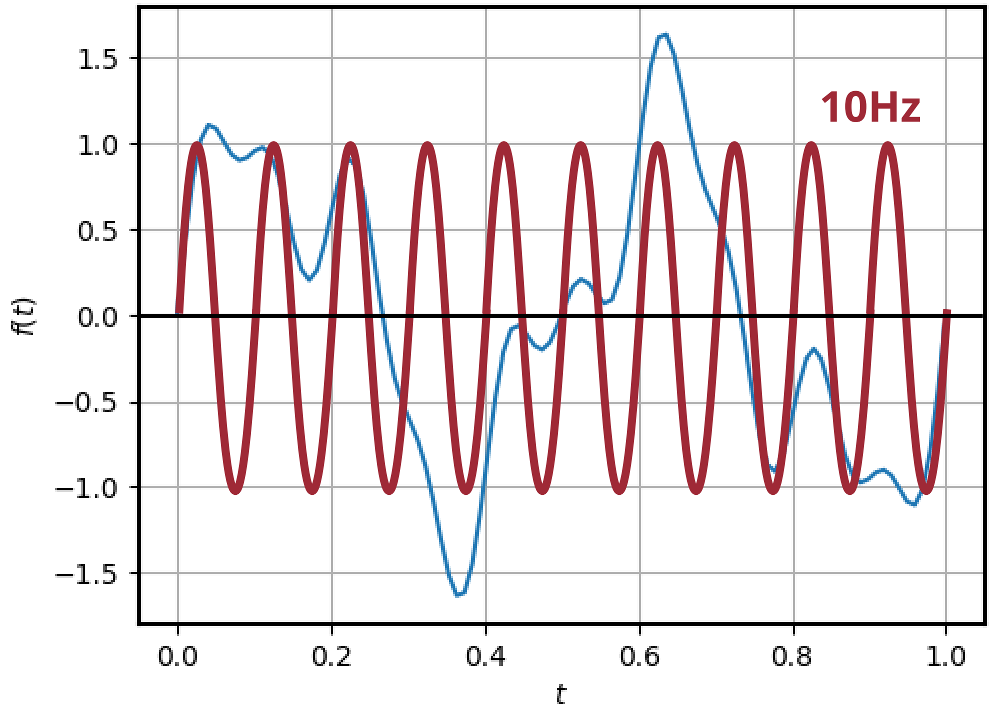
Demystifying Fourier Transform



Demystifying Fourier Transform

Signal

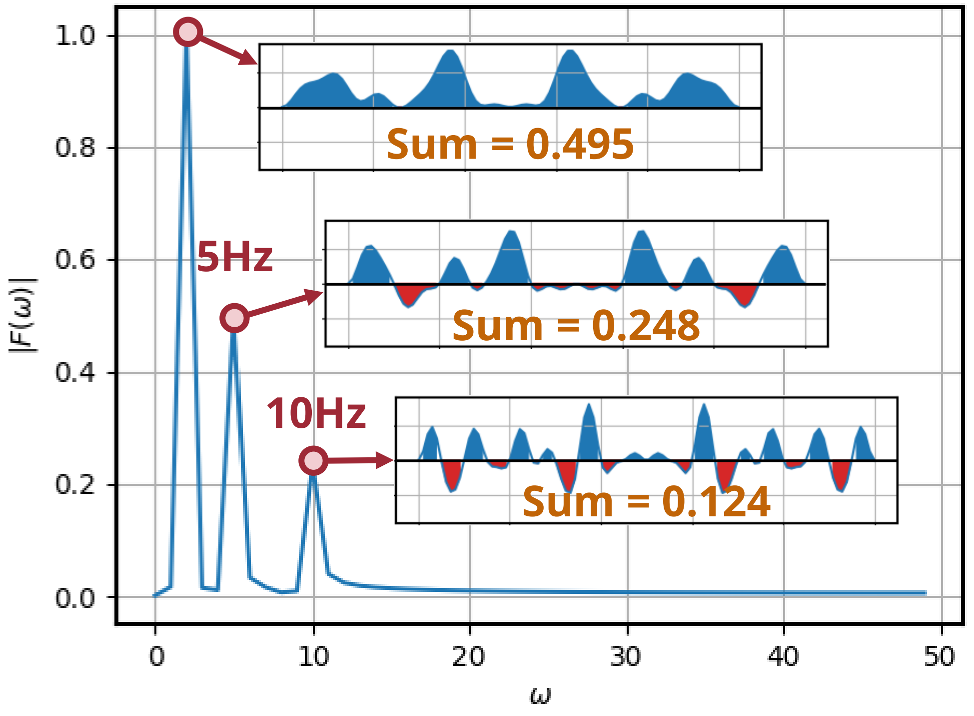
(time-domain)



Fourier Transform →

Spectrum

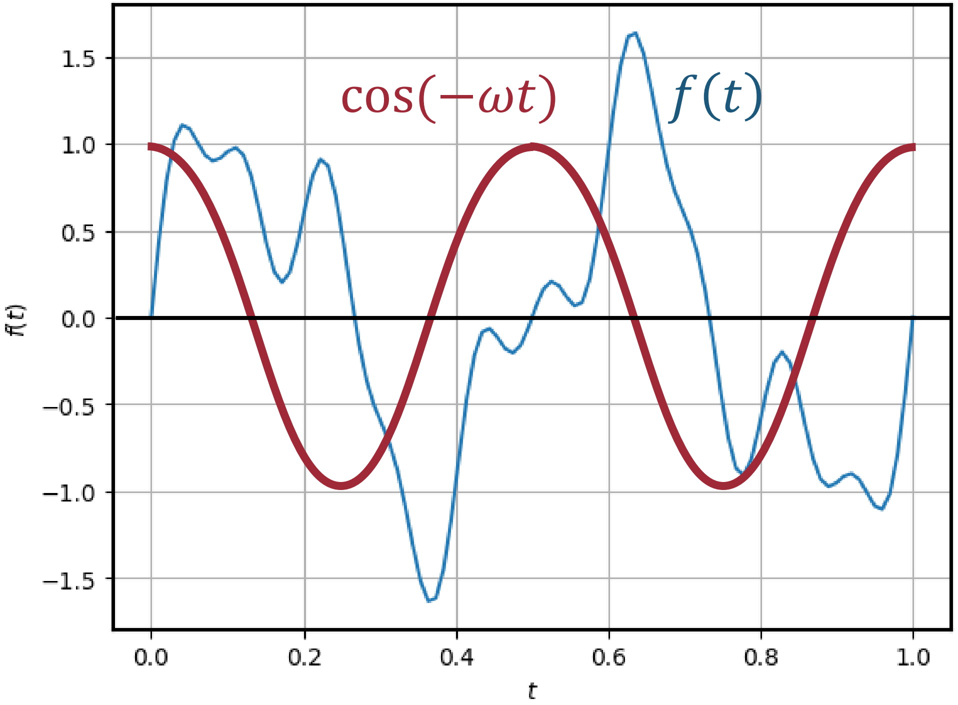
(frequency-domain)



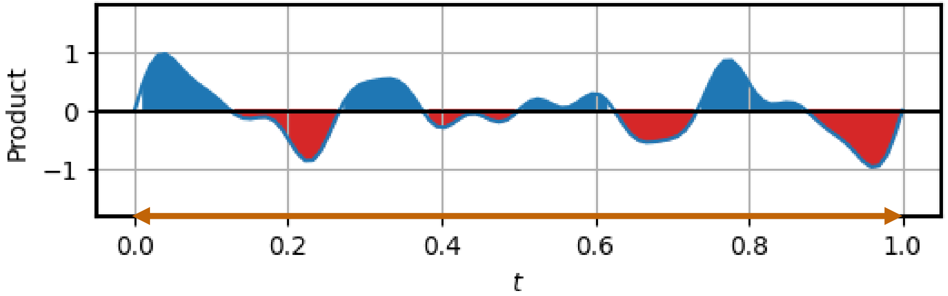
Demystifying Fourier Transform

$$F(\omega) = \int_{-\infty}^{\infty} f(t)\cos(-\omega t) + j\, f(t)\sin(-\omega t)\; dt$$

Sum over all t

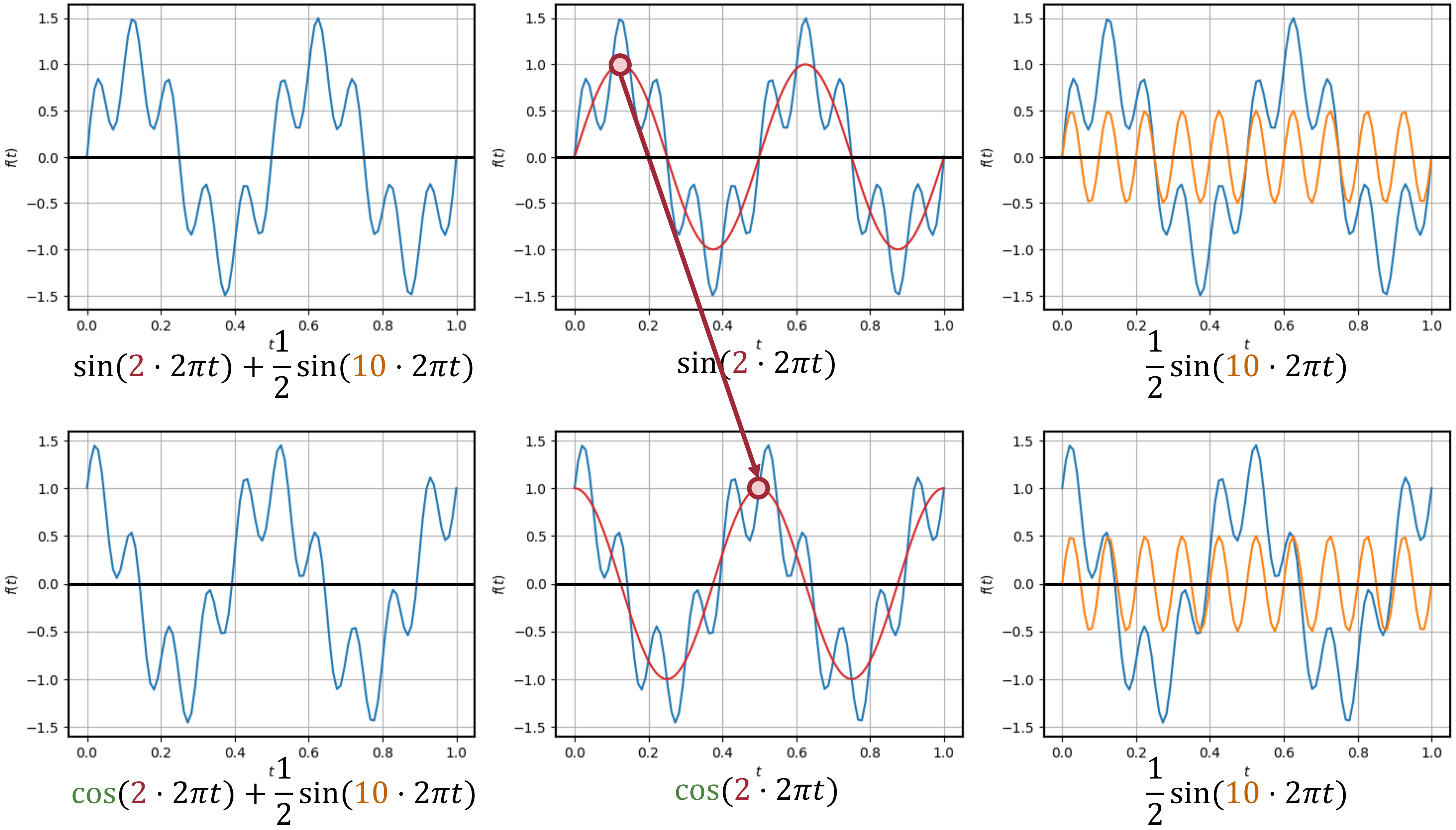


$f(t)\cos(-\omega t)$



Sum = 0

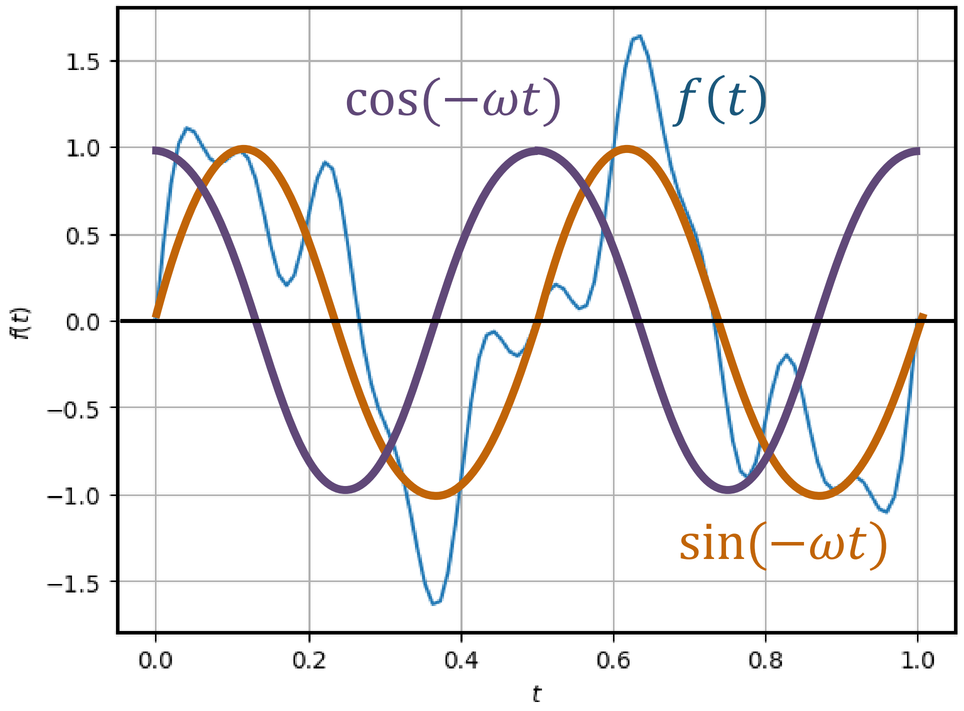
Phase


$\sin(2 \cdot 2\pi t) + \frac{1}{2}\sin(10 \cdot 2\pi t)$
$\sin(2 \cdot 2\pi t)$
$\frac{1}{2}\sin(10 \cdot 2\pi t)$
$\cos(2 \cdot 2\pi t) + \frac{1}{2}\sin(10 \cdot 2\pi t)$
$\cos(2 \cdot 2\pi t)$
$\frac{1}{2}\sin(10 \cdot 2\pi t)$
f(t)
t

Demystifying Fourier Transform

Real part **Imaginary part**

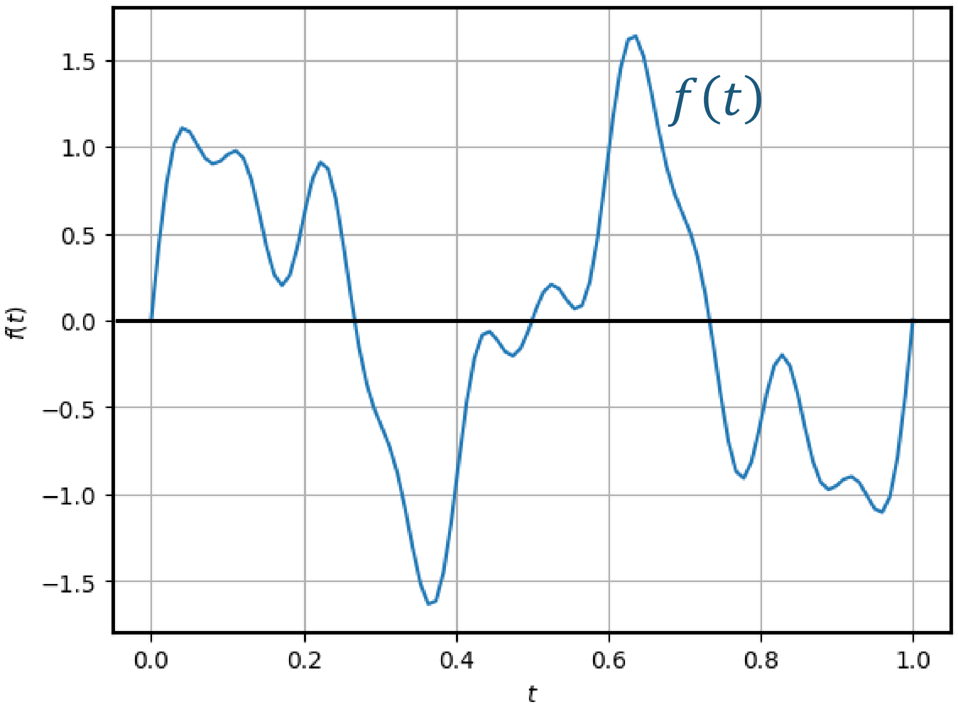
$$F(\omega) = \int_{-\infty}^{\infty} f(t)\cos(-\omega t) + j f(t)\sin(-\omega t)\, dt$$

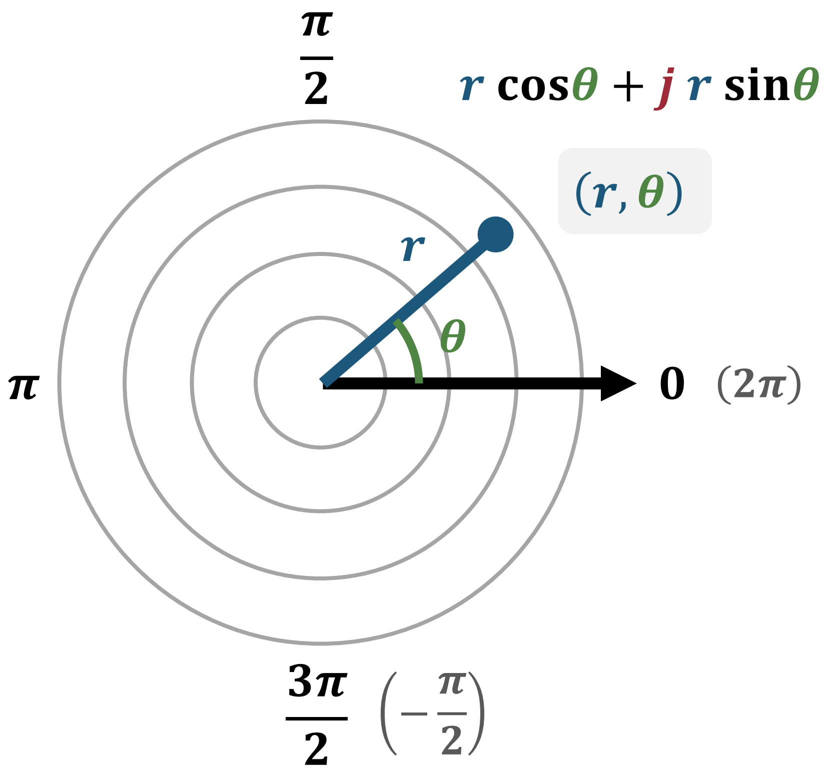


Demystifying Fourier Transform

Real part — **Imaginary part**

$$F(\omega) = \int_{-\infty}^{\infty} \boxed{f(t)\cos(-\omega t)} + \boxed{j\, f(t)\sin(-\omega t)}\, dt$$

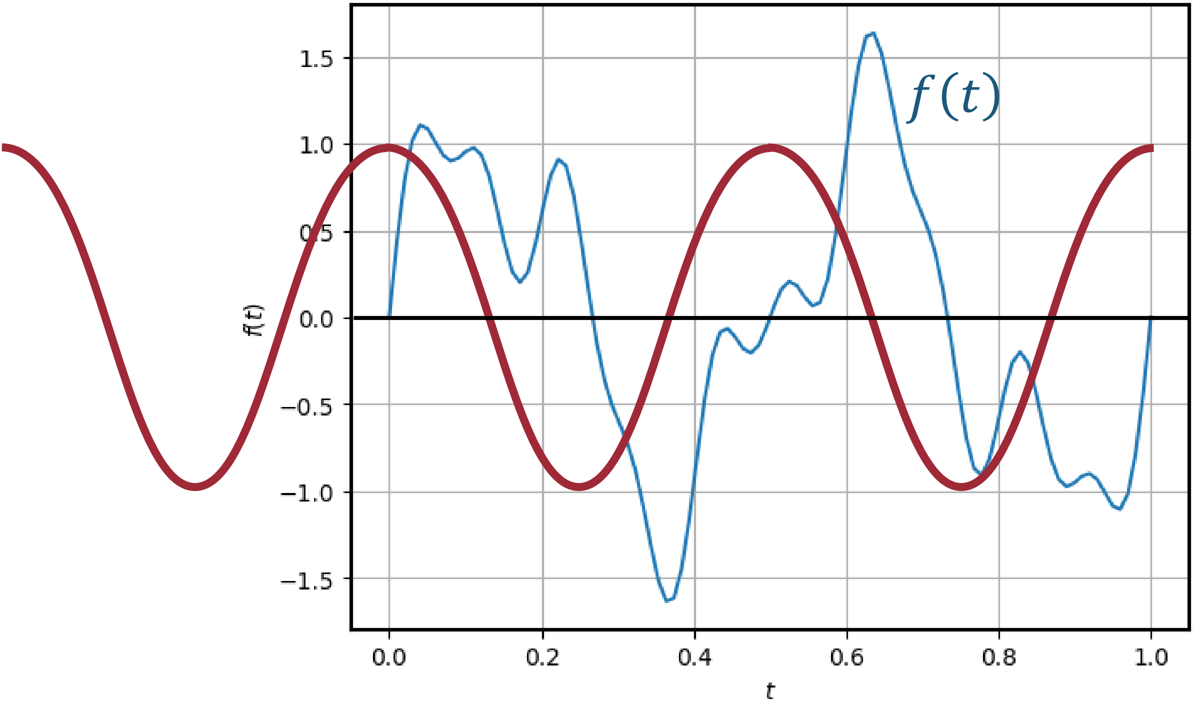


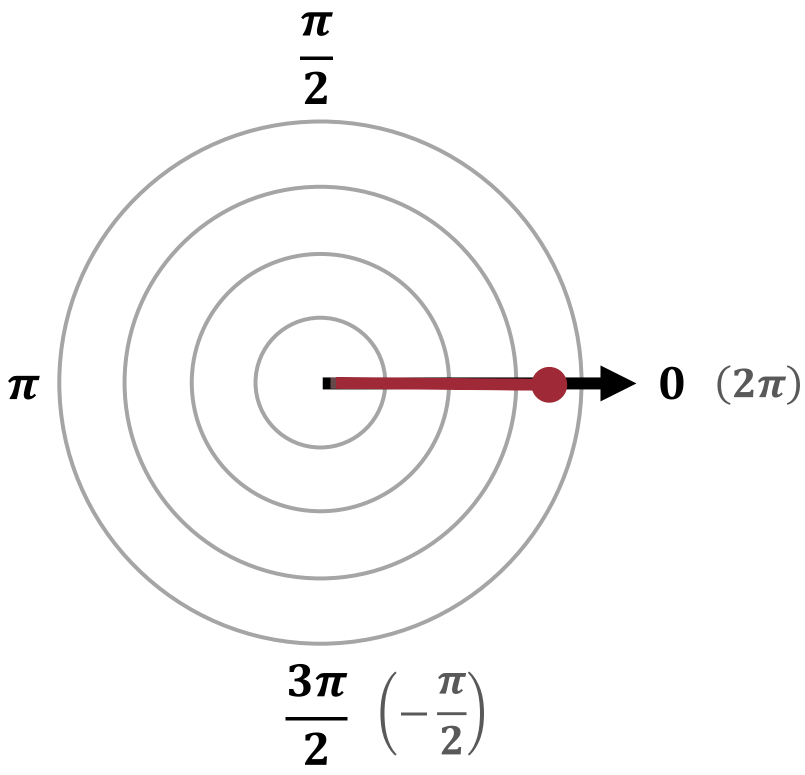


Demystifying Fourier Transform

Real part | **Imaginary part**

$$F(\omega) = \int_{-\infty}^{\infty} \boxed{f(t)\cos(-\omega t)} + \boxed{j\, f(t)\sin(-\omega t)}\, dt$$

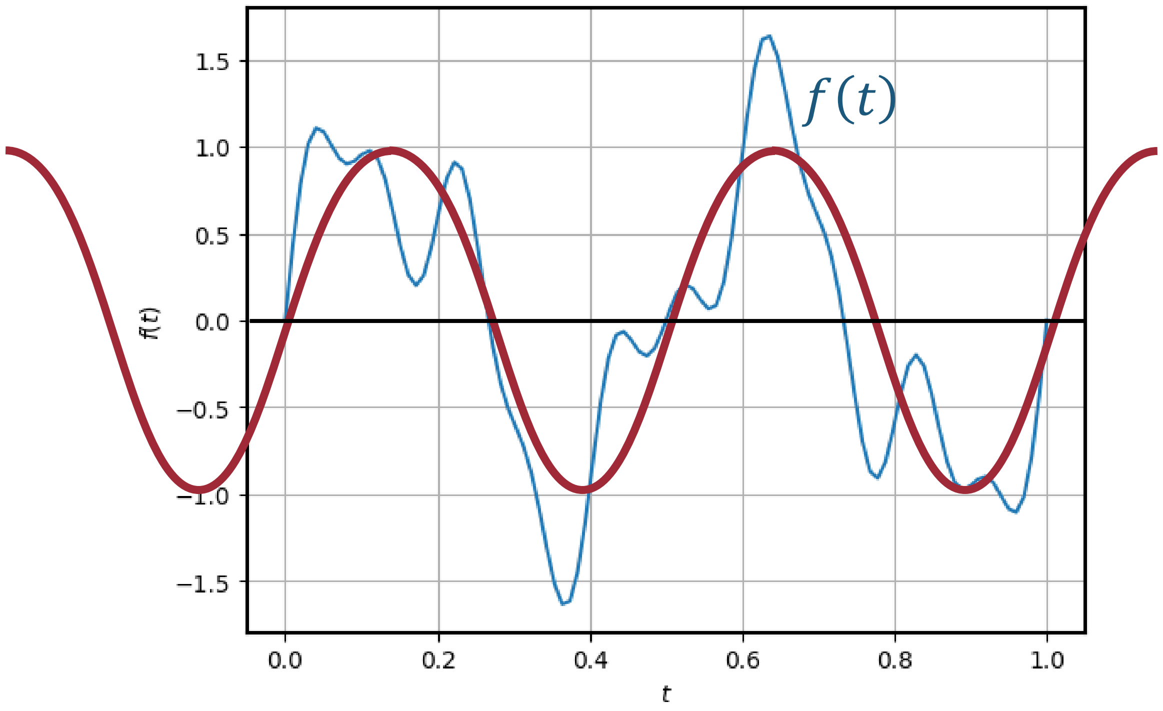


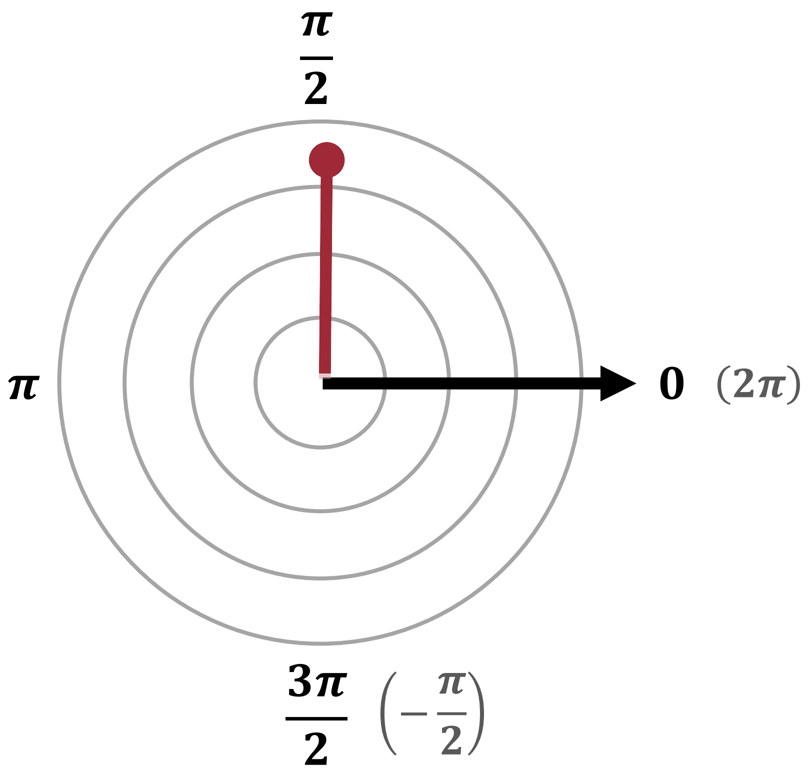


Demystifying Fourier Transform

Real part   **Imaginary part**

$$F(\omega) = \int_{-\infty}^{\infty} f(t)\cos(-\omega t) + j f(t)\sin(-\omega t)\, dt$$

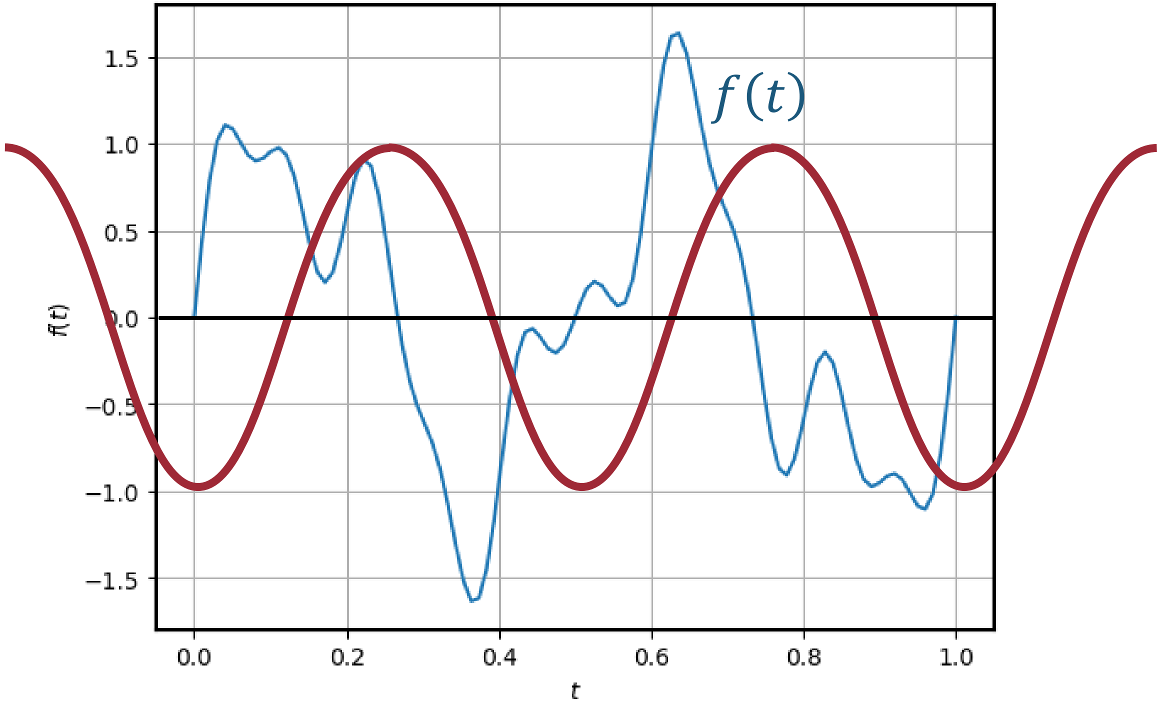


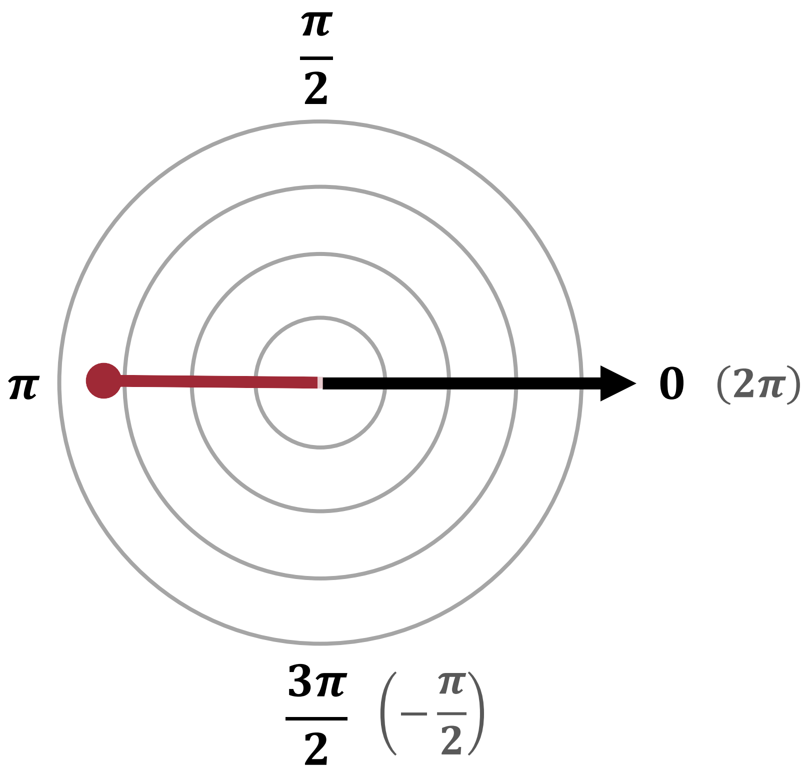


Demystifying Fourier Transform

Real part | **Imaginary part**

$$F(\omega) = \int_{-\infty}^{\infty} f(t)\cos(-\omega t) + j\, f(t)\sin(-\omega t)\, dt$$

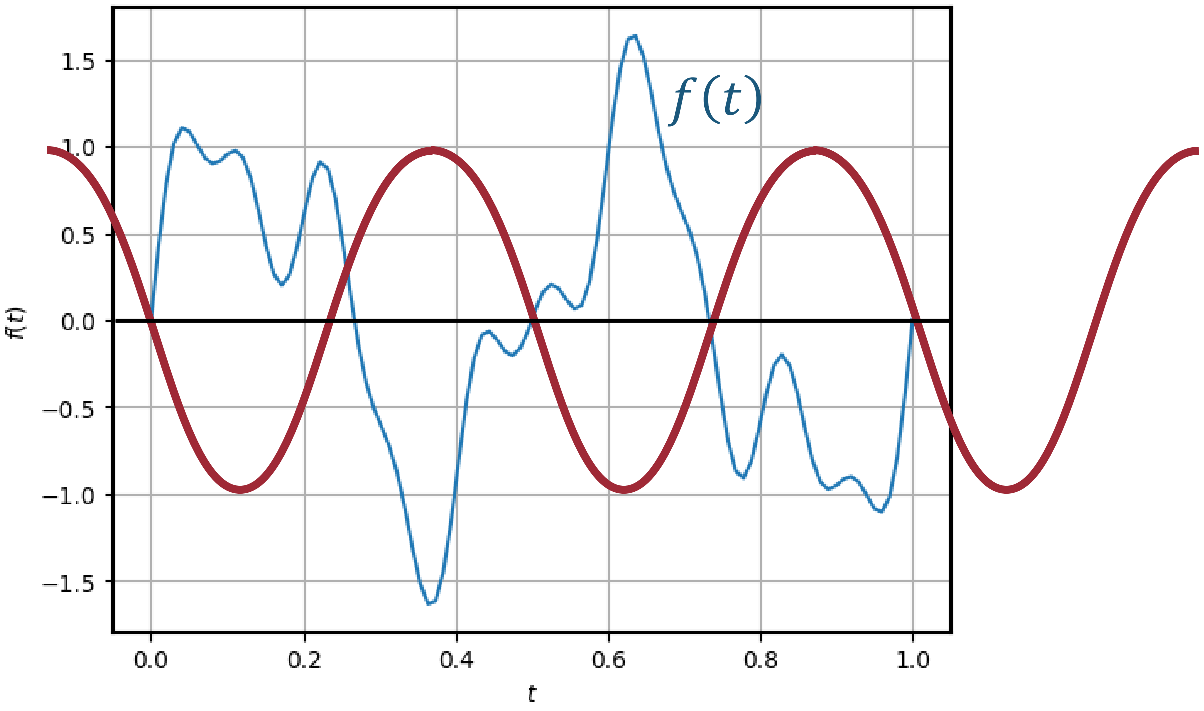


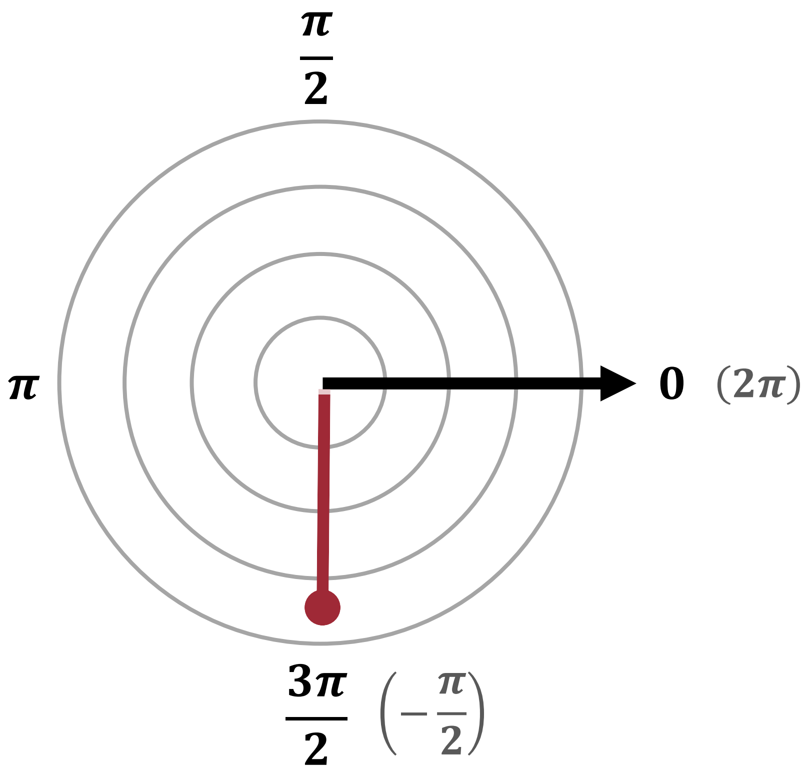


Demystifying Fourier Transform

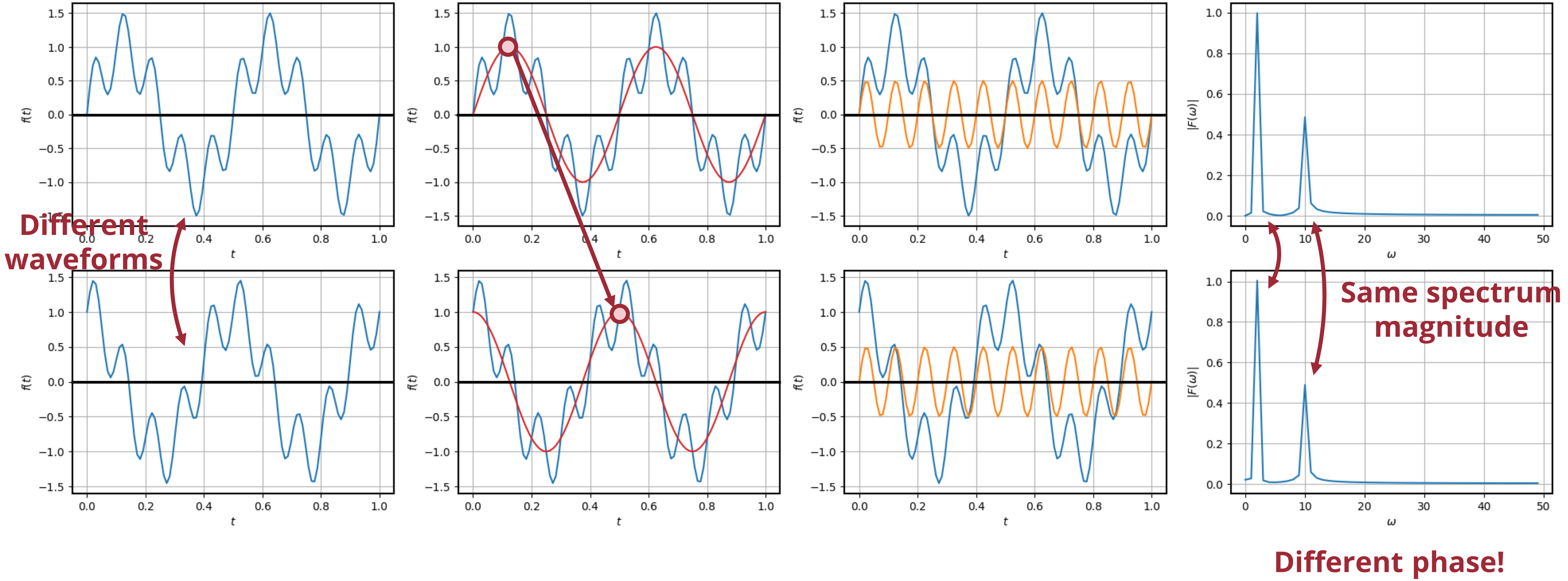
Real part — **Imaginary part**

$$F(\omega) = \int_{-\infty}^{\infty} \boxed{f(t)\cos(-\omega t)} + \boxed{j\, f(t)\sin(-\omega t)}\, dt$$





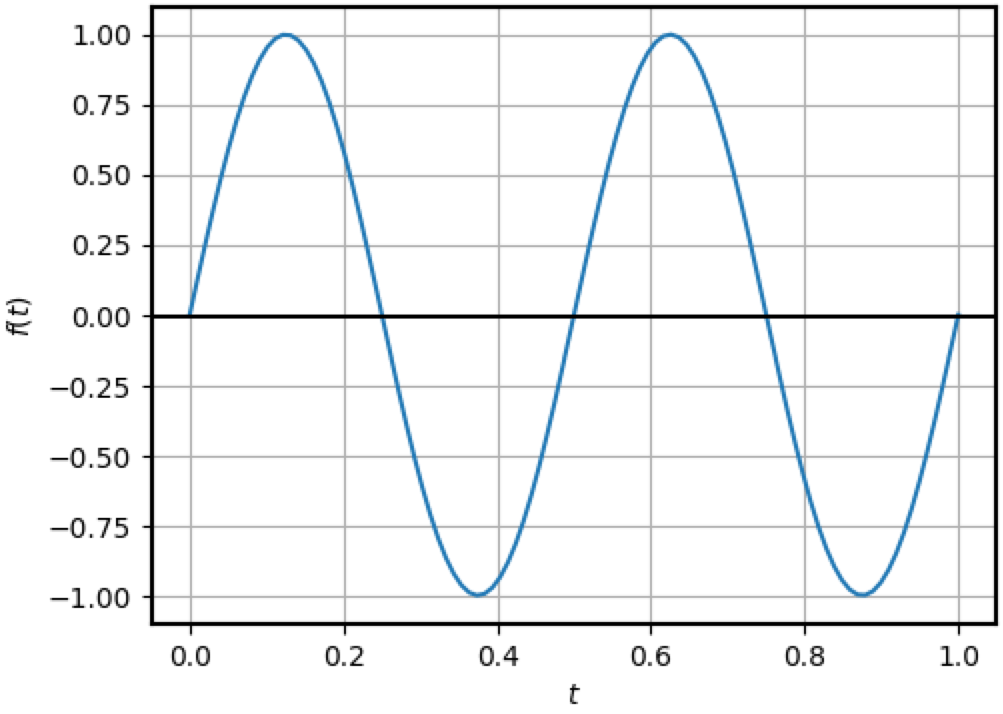
Magnitude & Phase



Example: A 2Hz Sine Wave

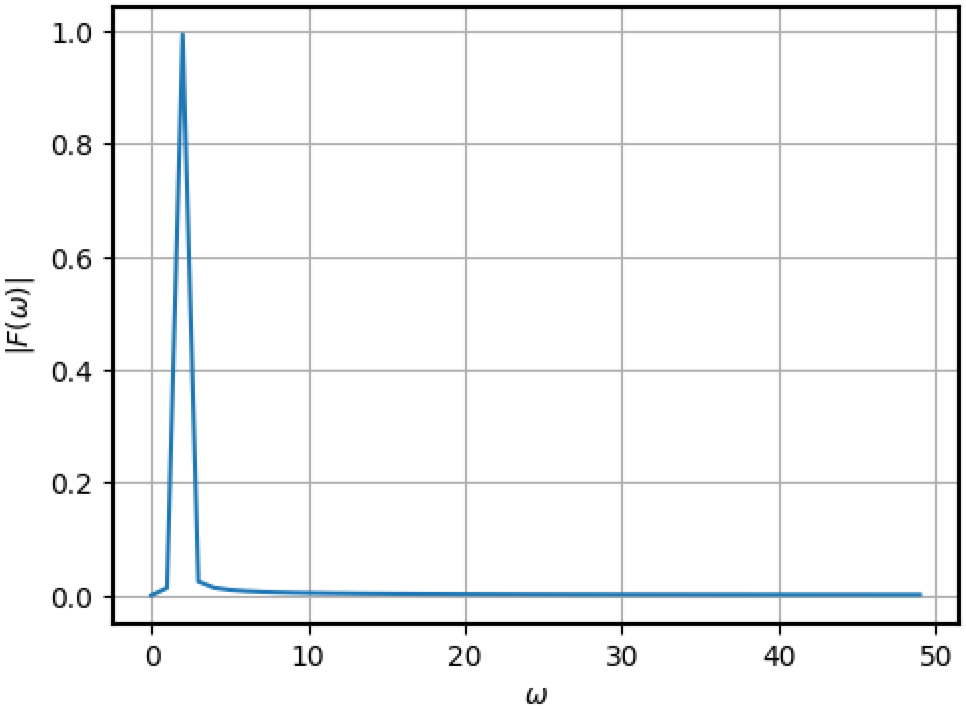
Signal

(time-domain)



Spectrum

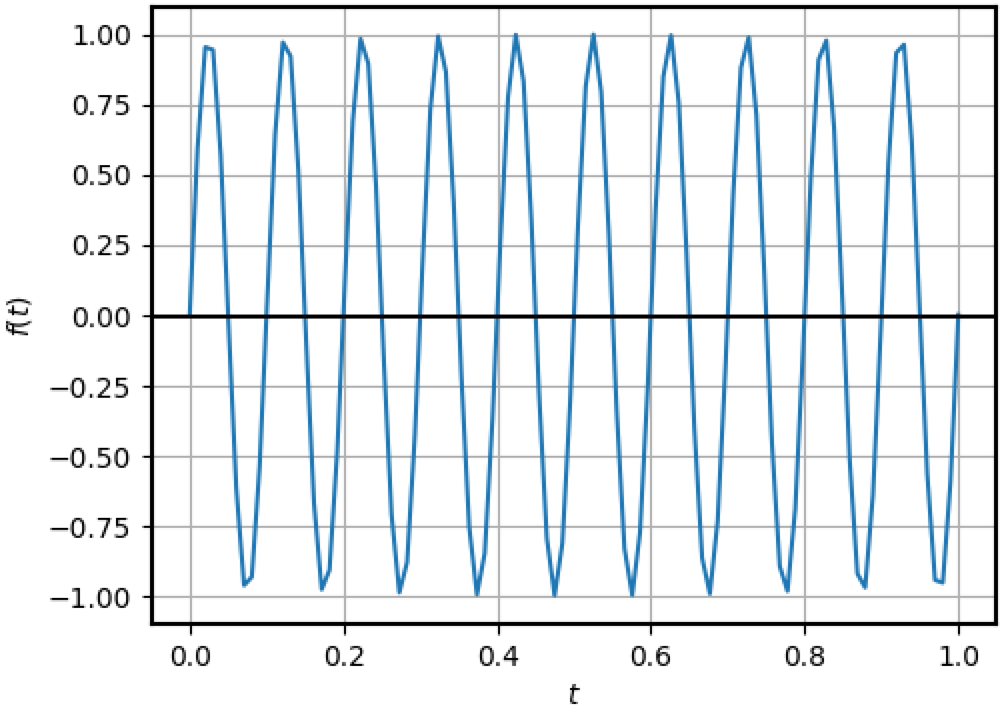
(frequency-domain)



Example: A 10Hz Sine Wave

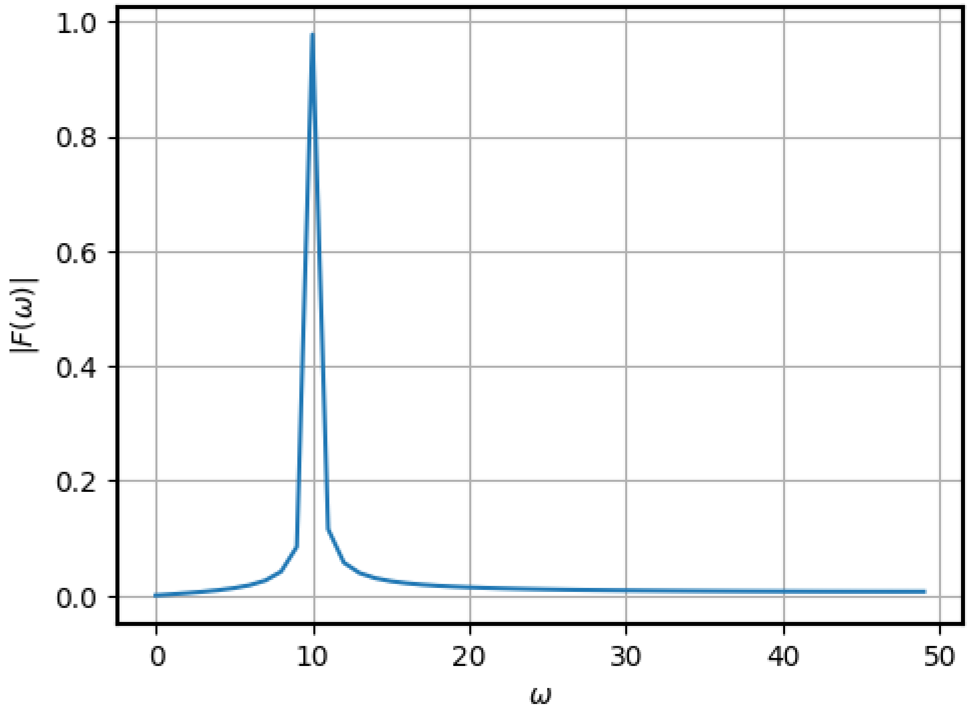
Signal

(time-domain)



Spectrum

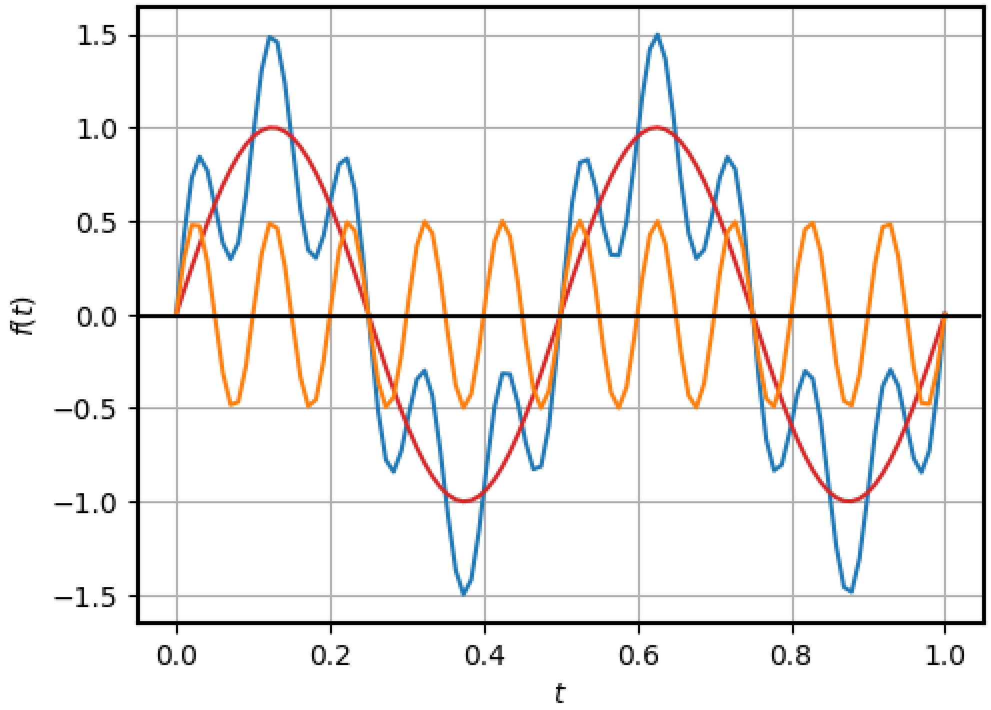
(frequency-domain)



Example: Sum of 2Hz & 10Hz Sine Waves

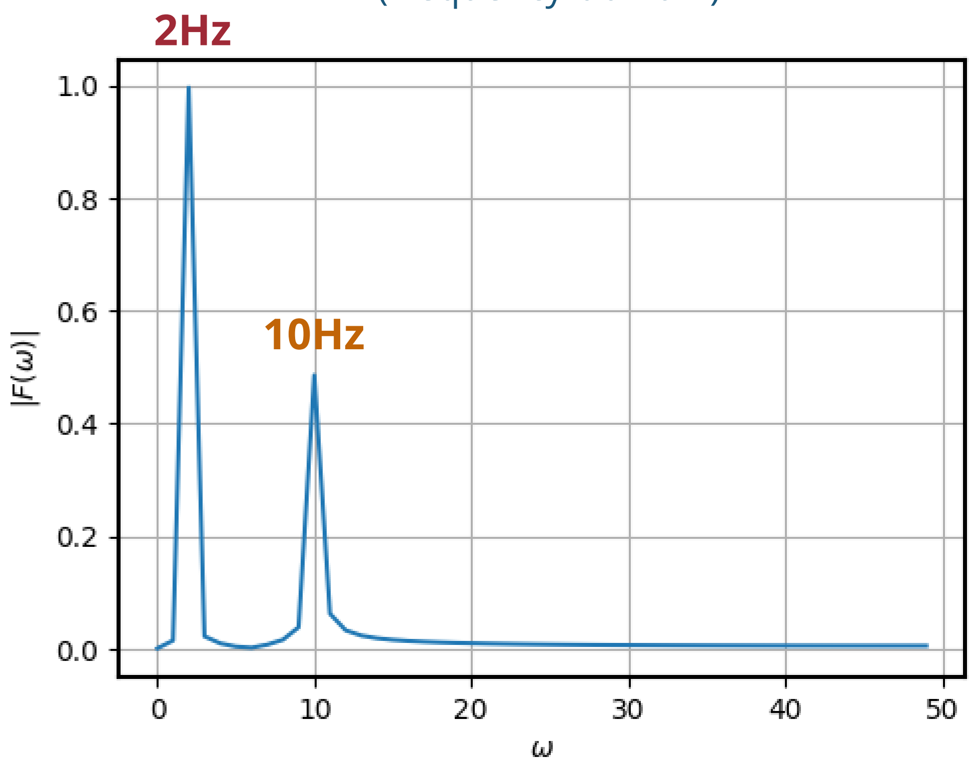
Signal

(time-domain)



Spectrum

(frequency-domain)



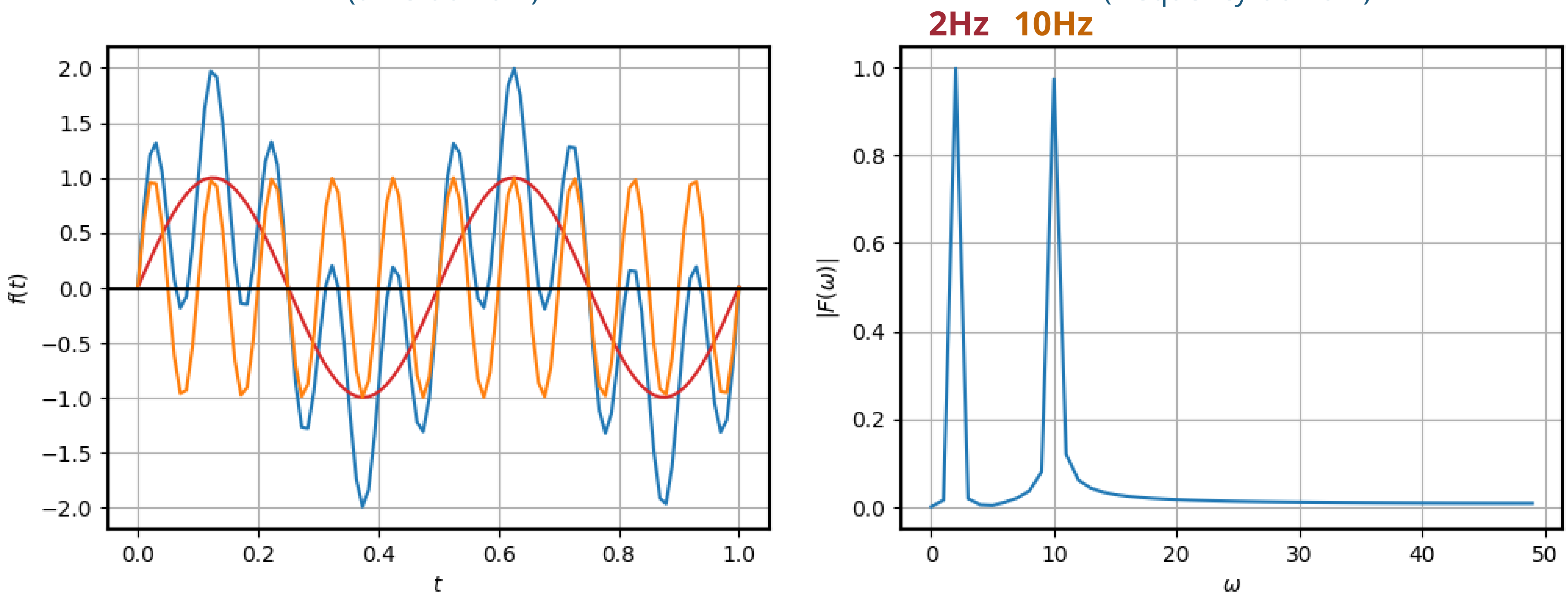
How about this?

Signal

(time-domain)

Spectrum

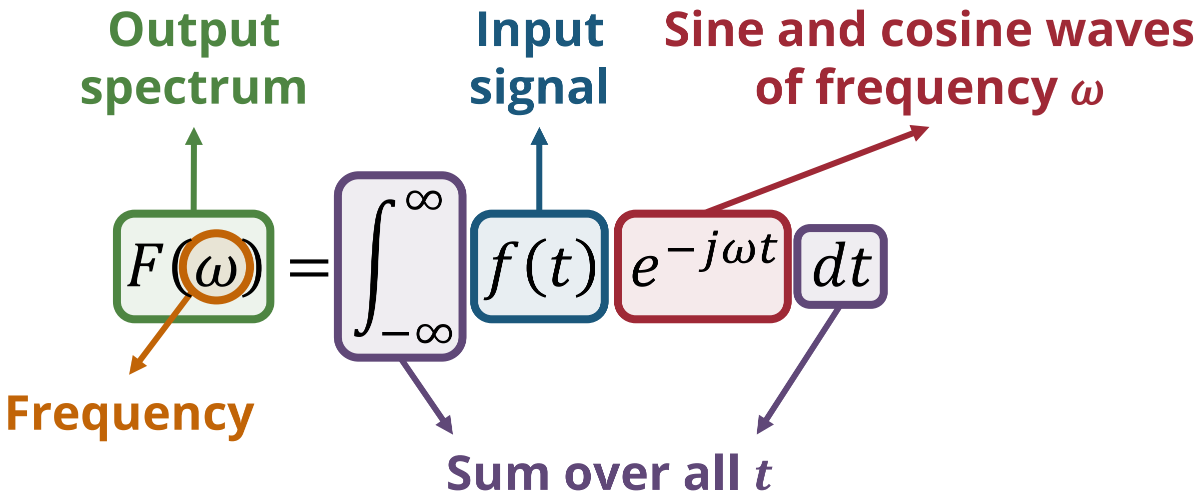
(frequency-domain)



Fourier Transform

- **Intuition**: Decompose time-domain signals into **frequency components**
- Math formulation:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)\, e^{-j\omega t}\, dt$$



Fourier Transform

- **Intuition**: **Analysis** through **resynthesis**!

Analysis

$$F(\omega) = \int_{-\infty}^{\infty} f(t)\, e^{-j\omega t}\, dt$$

Synthesis

Discrete Fourier Transform (DFT)

- **Intuition**: Fourier transform with **discrete time and frequency**
  - Used for **digital audio** → we cannot achieve an infinite sampling rate…
- Math formulation:

$$X_k = \sum_{n=0}^{N-1} x_n \; e^{-j2\pi\frac{k}{N}n}$$

Fourier Transform vs. Discrete Fourier Transform

Fourier Transform

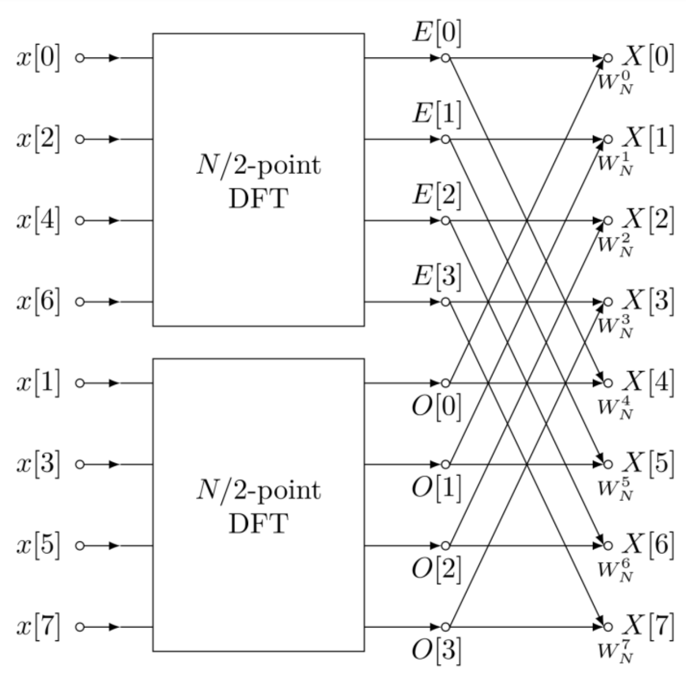
$$F(\omega) = \int_{-\infty}^{\infty} f(t)\, e^{-j\omega t}\, dt$$

Discrete Fourier Transform

$$X_k = \sum_{n=0}^{N-1} x_n\, e^{-j2\pi\frac{k}{N}n}$$

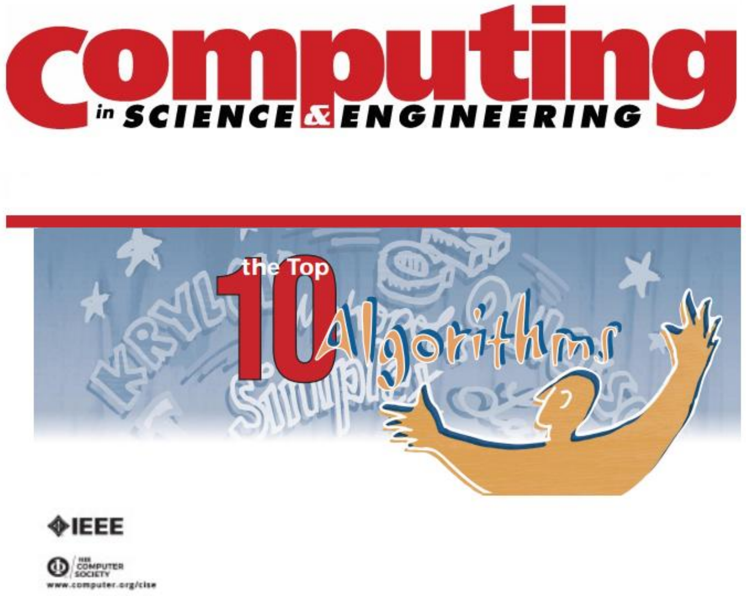
In Practice: Fast Fourier Transform (FFT)

- An **efficient implementation** of discrete Fourier transform
  - Reduce the complexity from $O(n^2)$ to $O(n \log n)$



(Source: Yangwenbo99 via Wikimedia)

Top 10 algorithms from the 20th century

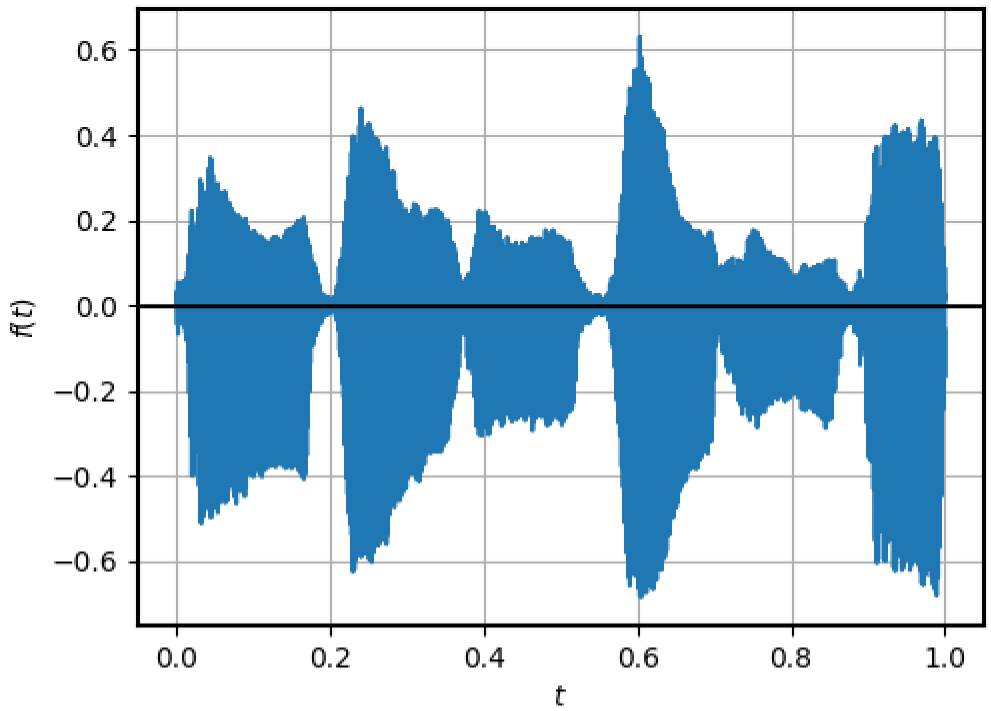


Time-Frequency Analysis

Fourier Transform of a **Trumpet** Sound

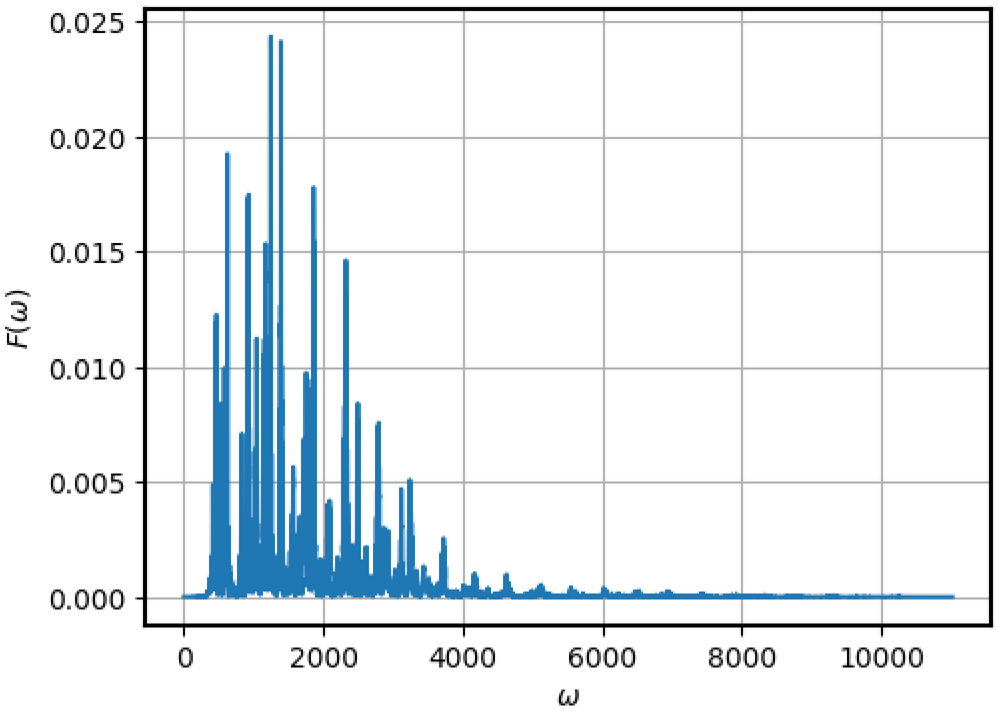
Signal

(time-domain)



Spectrum

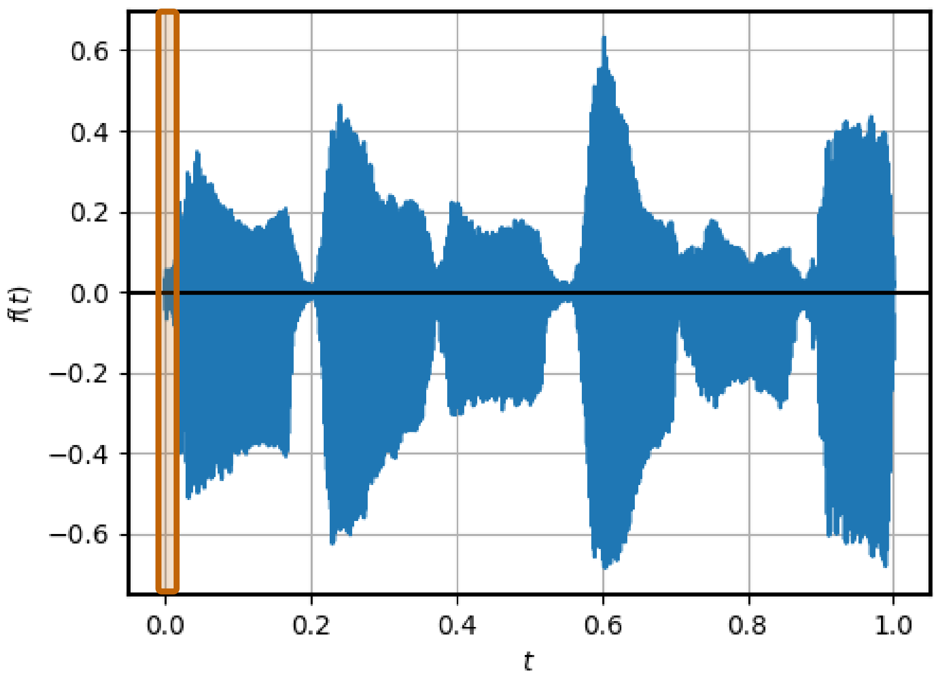
(frequency-domain)



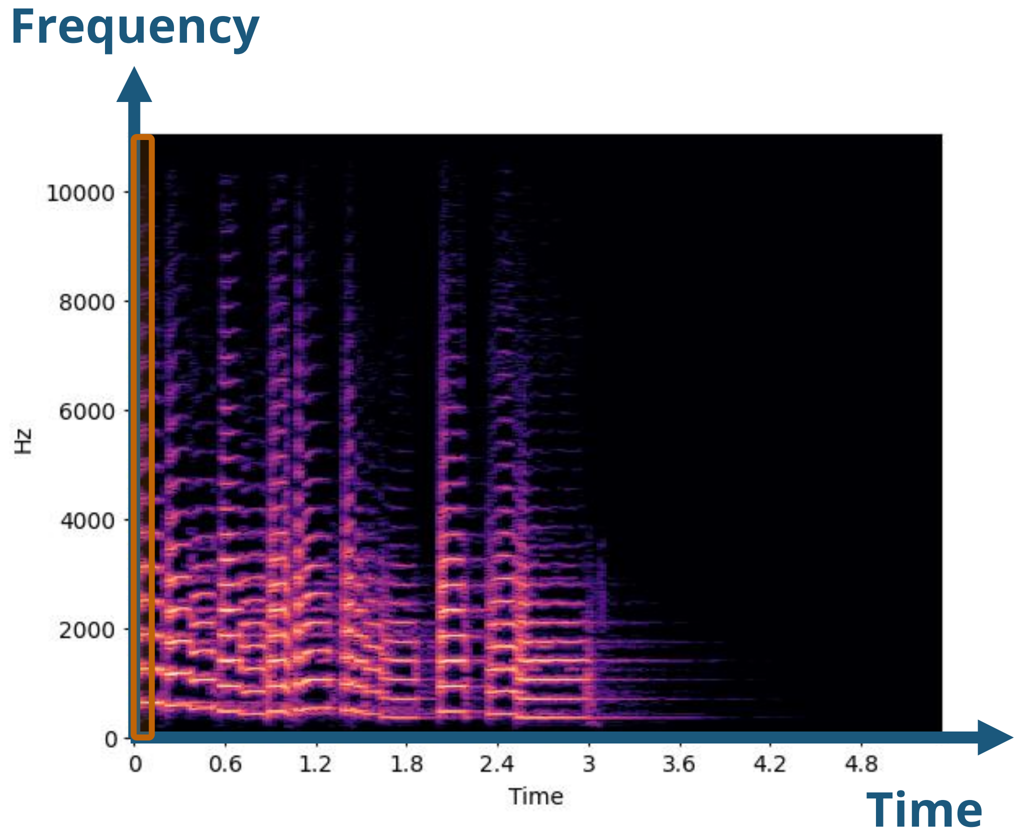
Fourier Transform cannot localize! 🥺

Short-Time Fourier Transform (STFT)

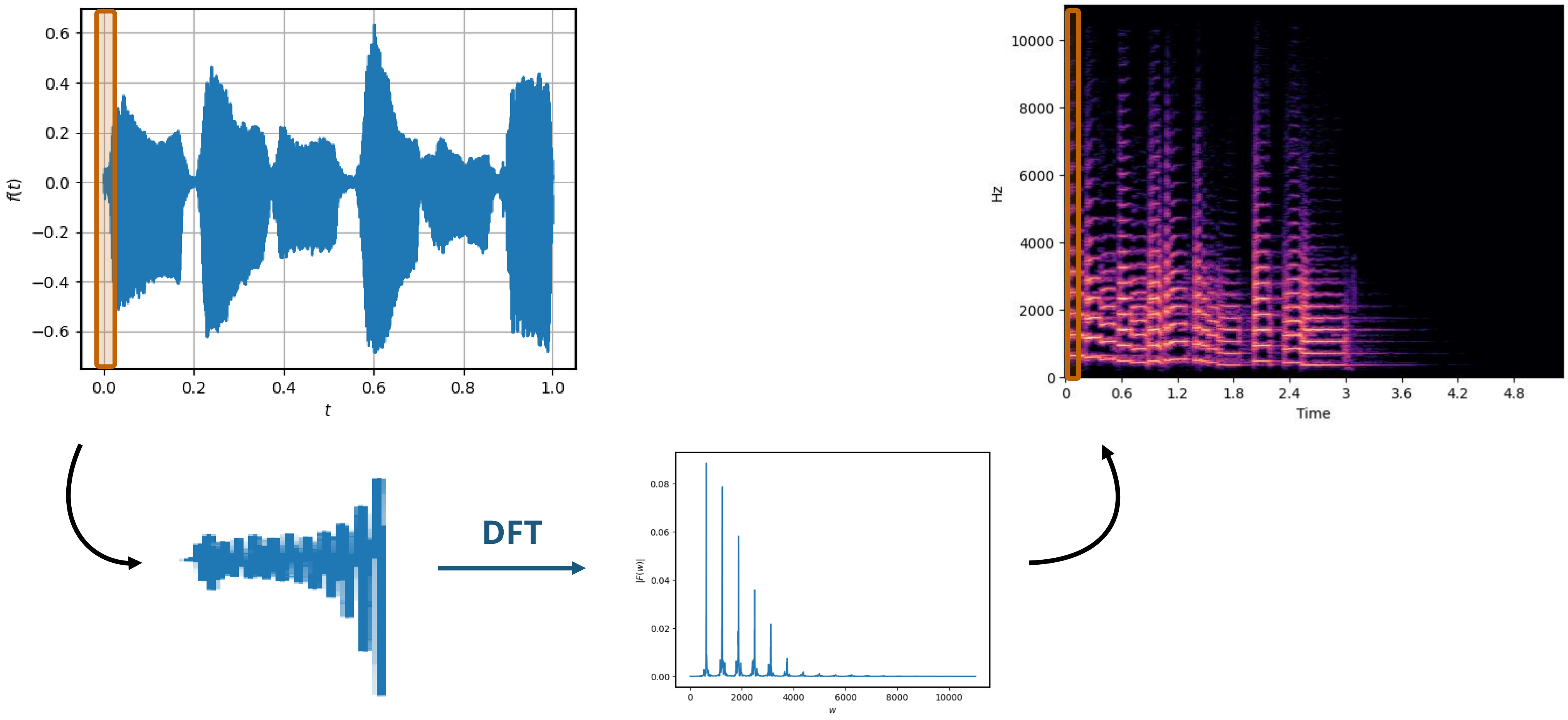
- **Intuition**: Slice the audio into chunks and apply Fourier transform



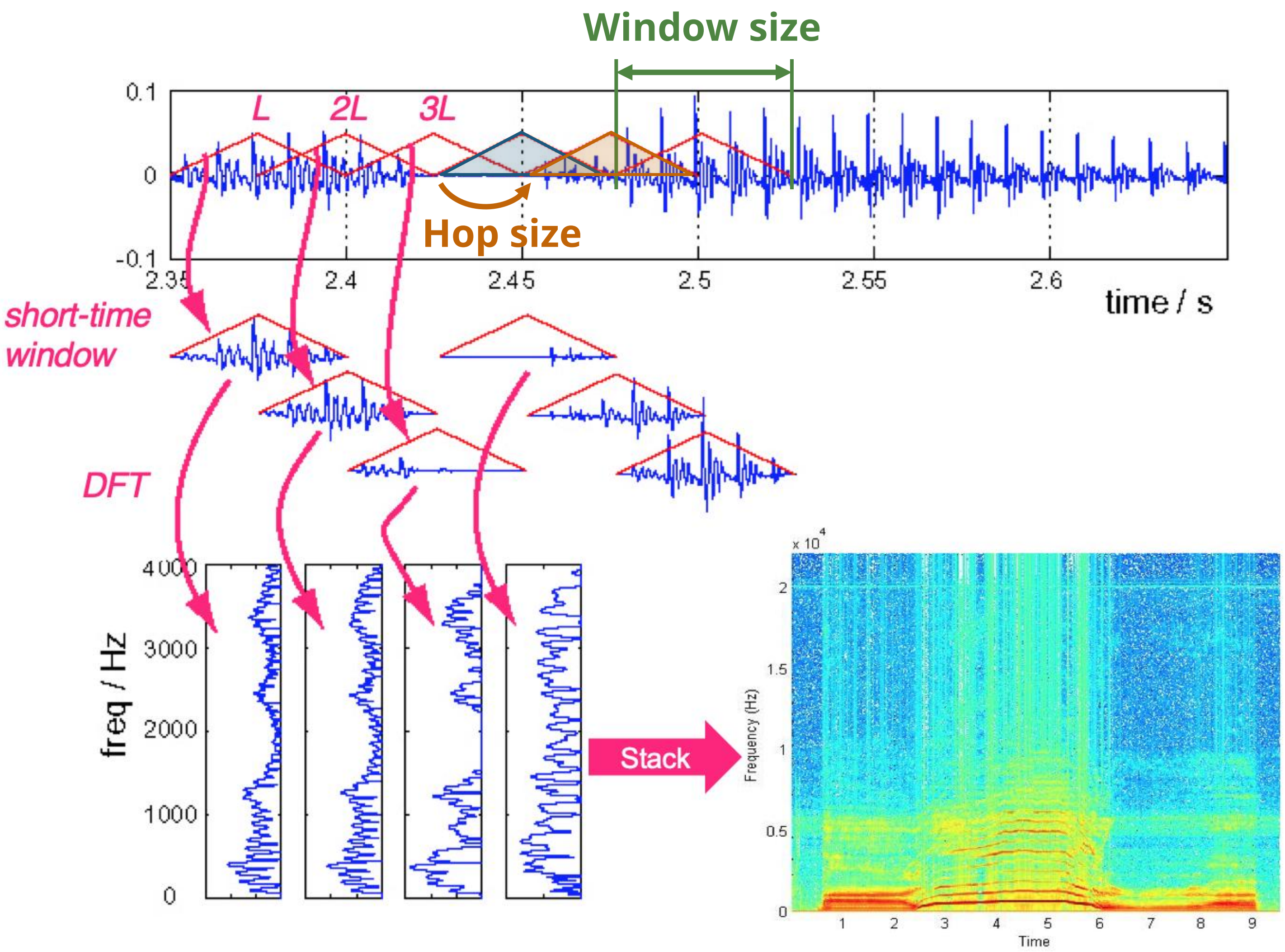
STFT



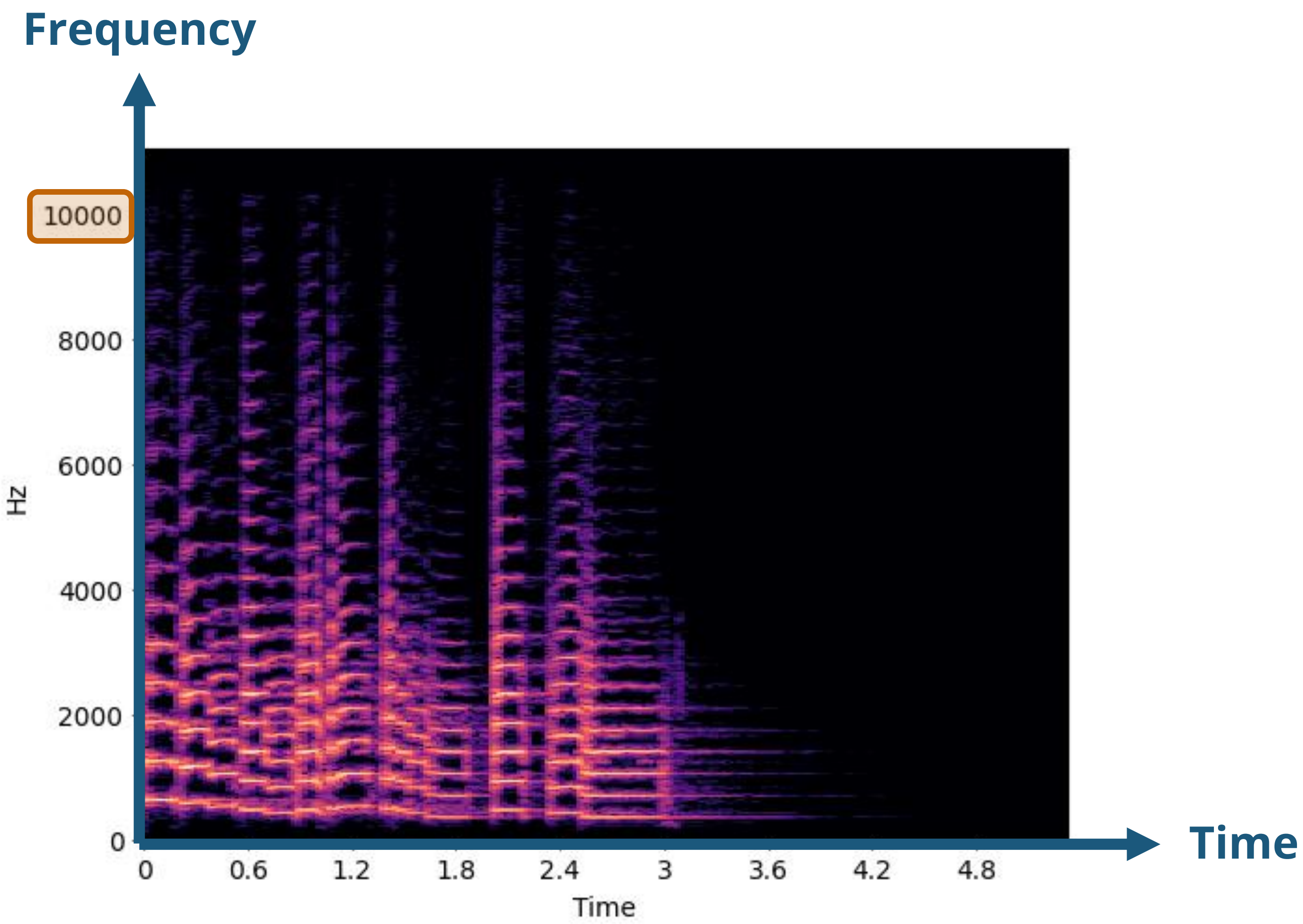
Short-Time Fourier Transform (STFT)



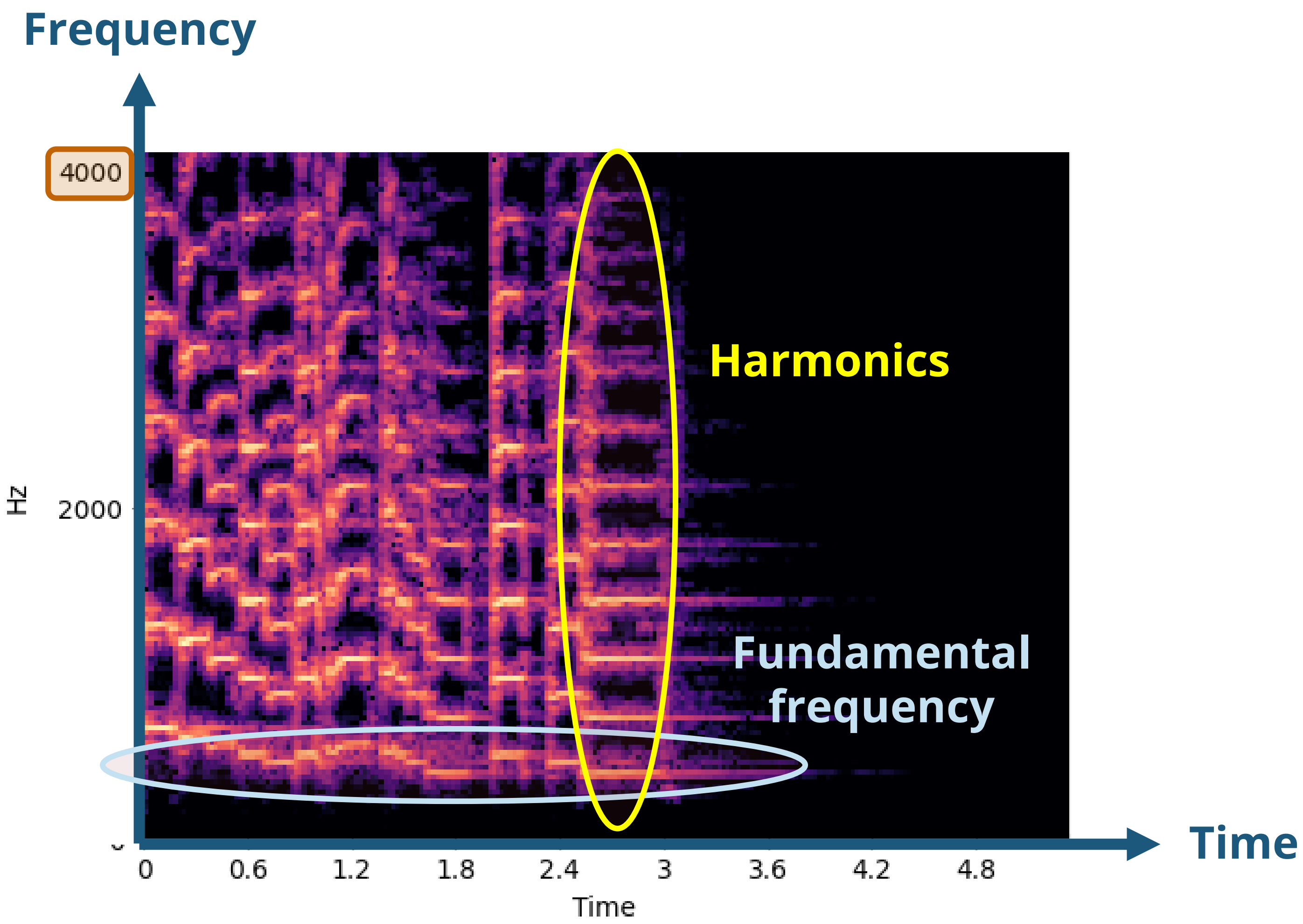
Short-Time Fourier Transform (STFT)



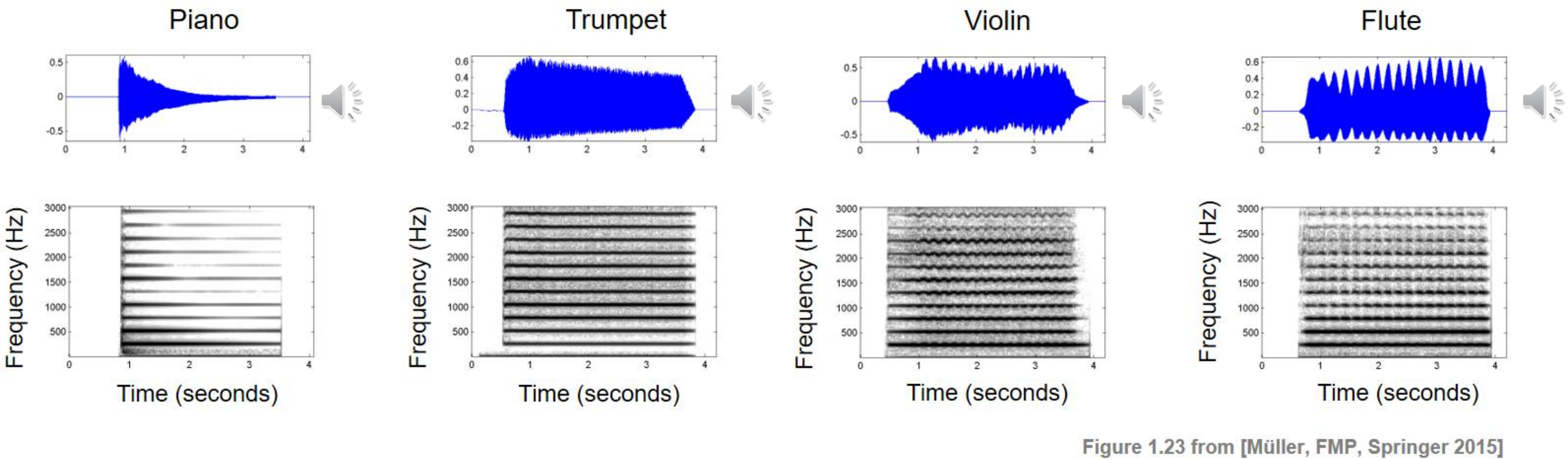
Spectrogram



Spectrogram



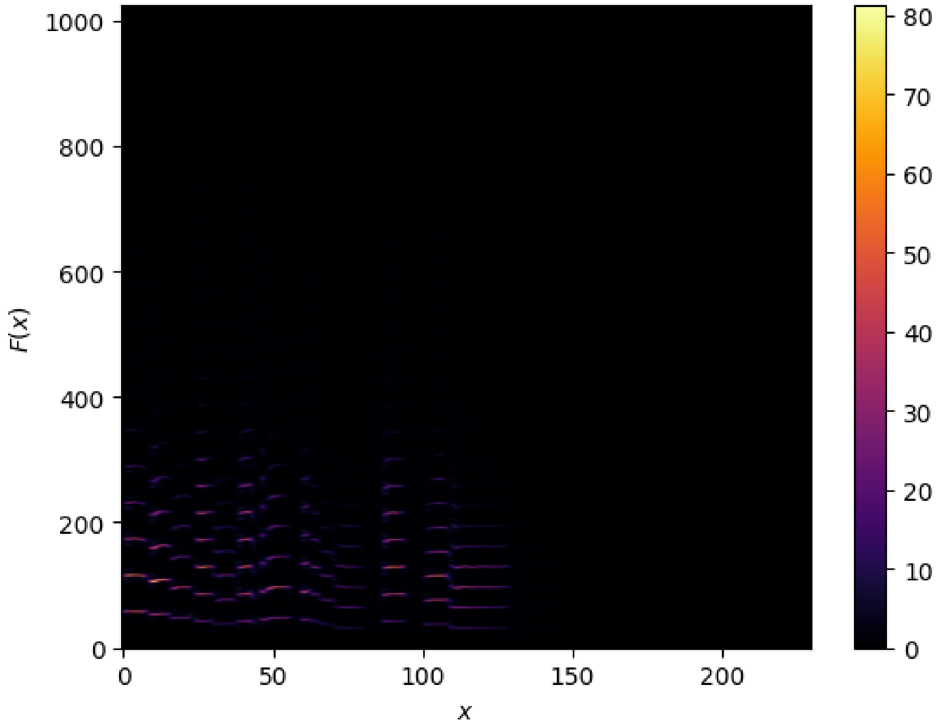
Timbre



(Source: Müller et al., 2021)

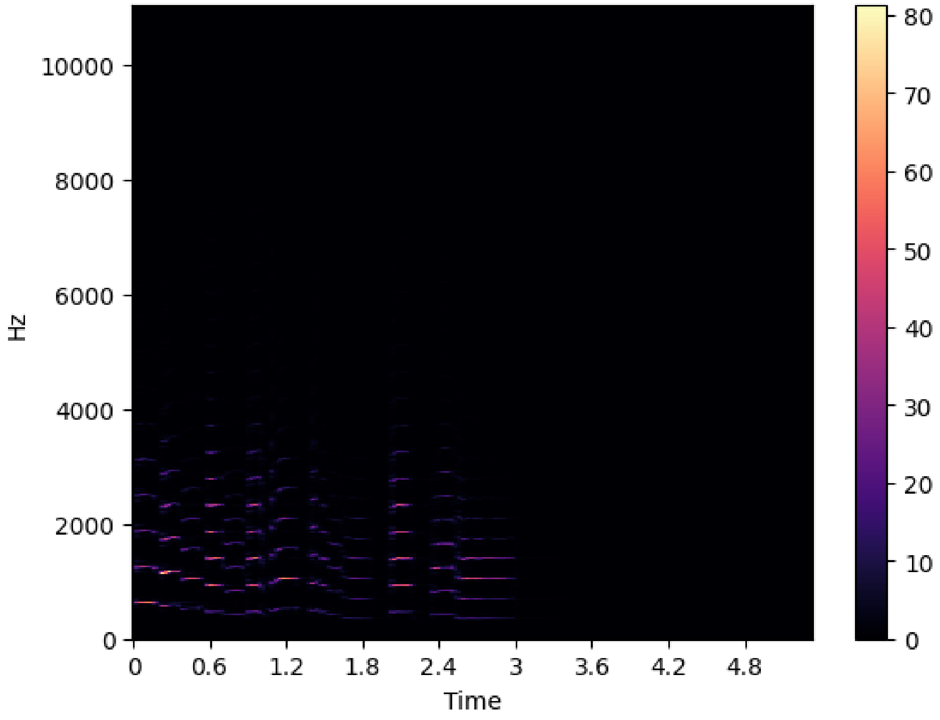
Meinard Müller, "Fundamentals of Music Processing – Using Python and Jupyter Notebooks," 2nd edition, *Springer Verlag*, 2021.

Example: librosa.stft



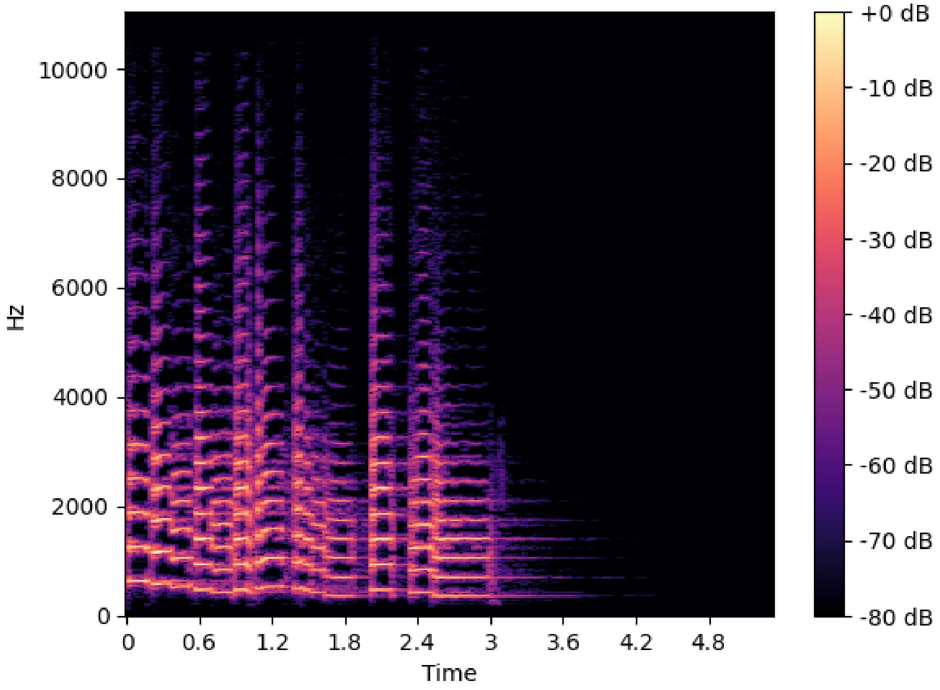

```python
# Load the example audio in librosa
y, sr = librosa.load(librosa.example("trumpet"))

# Compute the spectrogram
S = np.abs(librosa.stft(y))

# Plot the spectrogram
im = plt.imshow(S, cmap="inferno", aspect="auto",
                origin="lower")
plt.colorbar(im)
plt.xlabel("Time (sec)")
plt.ylabel("Frequency (Hz))")
plt.show()
```

Example: librosa.display.specshow




```
# Load the example audio in librosa
y, sr = librosa.load(librosa.example("trumpet"))

# Compute the spectrogram
S = np.abs(librosa.stft(y))

# Plot the spectrogram
im = librosa.display.specshow(S, x_axis="time",
                               y_axis="linear")
plt.colorbar(im)
plt.show()
```

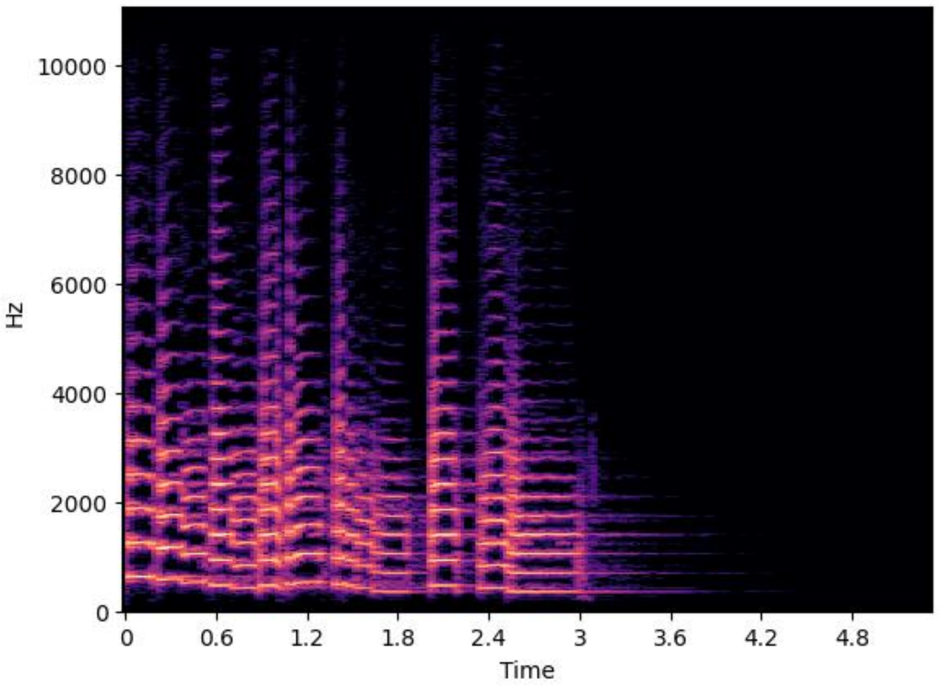
Example: librosa.amplitude_to_db



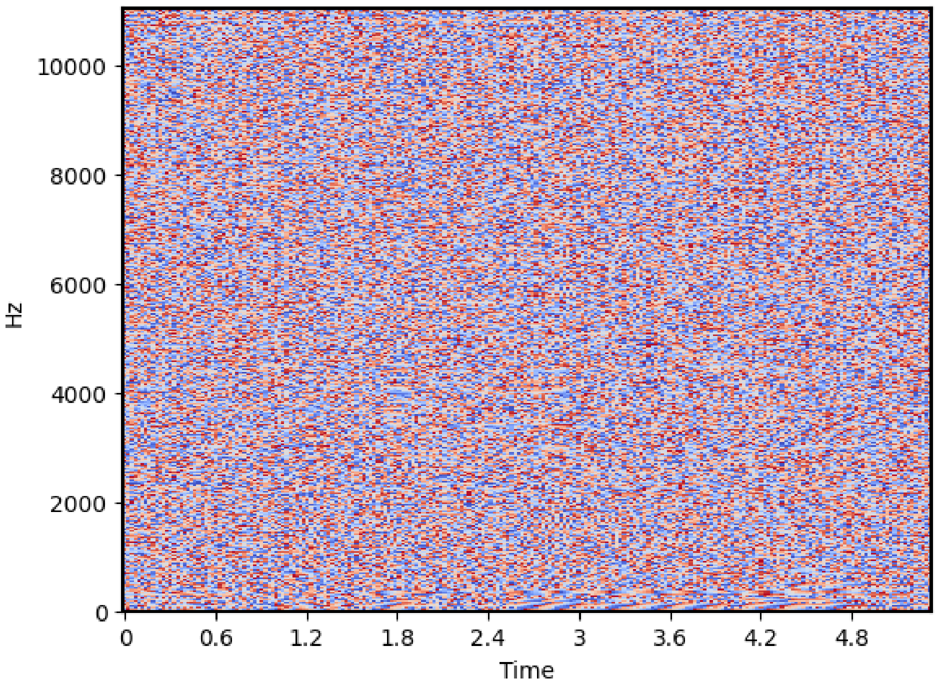

```python
# Load the example audio in librosa
y, sr = librosa.load(librosa.example("trumpet"))

# Compute the spectrogram
S = np.abs(librosa.stft(y))
S_db = librosa.amplitude_to_db(S, ref=np.max)

# Plot the spectrogram
im = librosa.display.specshow(S_db, x_axis="time",
                              y_axis="linear")
plt.colorbar(im, format="%+2.0f dB")
plt.show()
```

Example: Magnitude & Phase

Magnitude

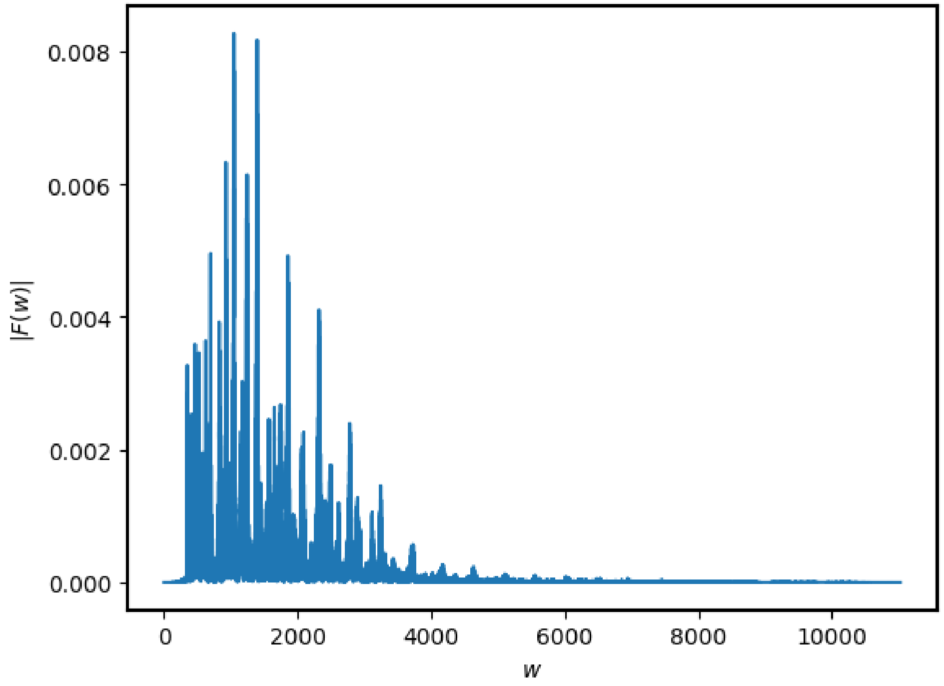


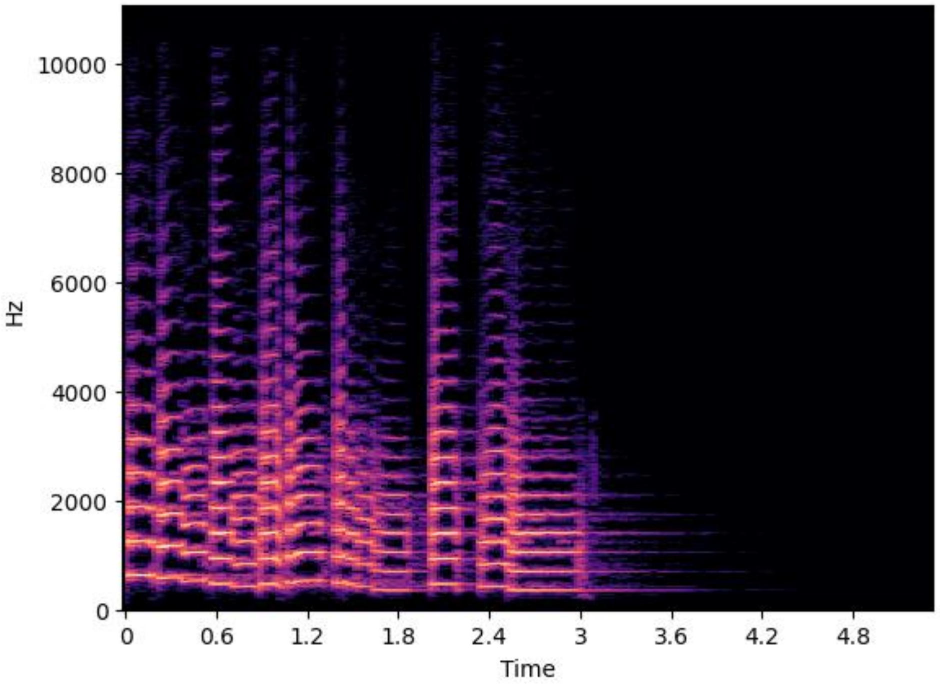
Phase



🔥 **PA2**: Spectral Analysis

- Use **[librosa]** to process audio files
  - Fast Fourier transform (**FFT**)
  - Short-time Fourier transform (**STFT**)

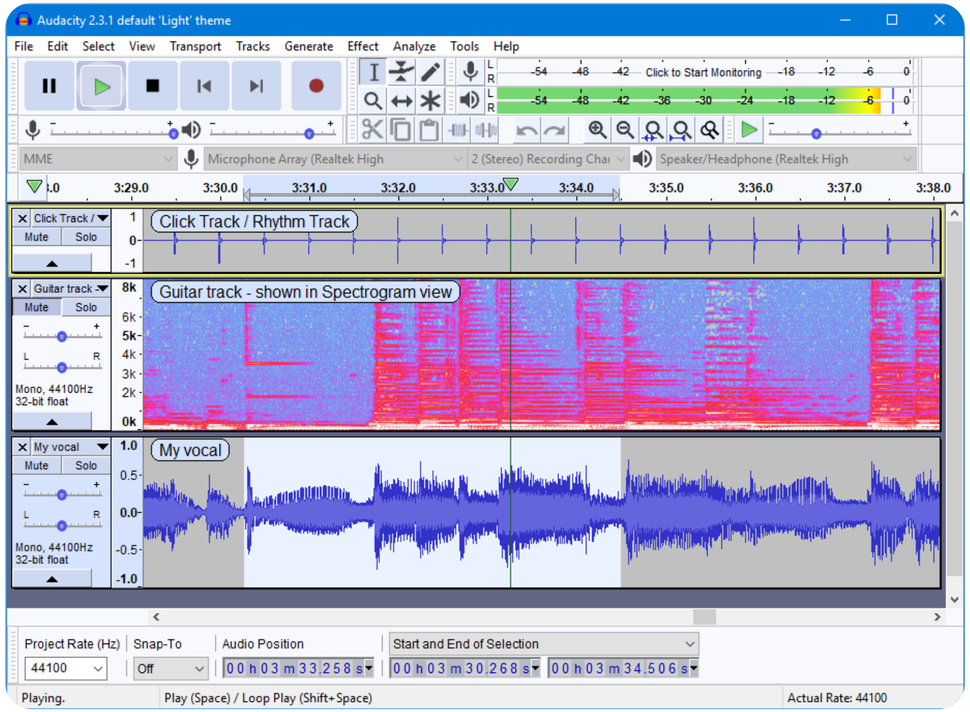




librosa.org/doc/latest/index.html

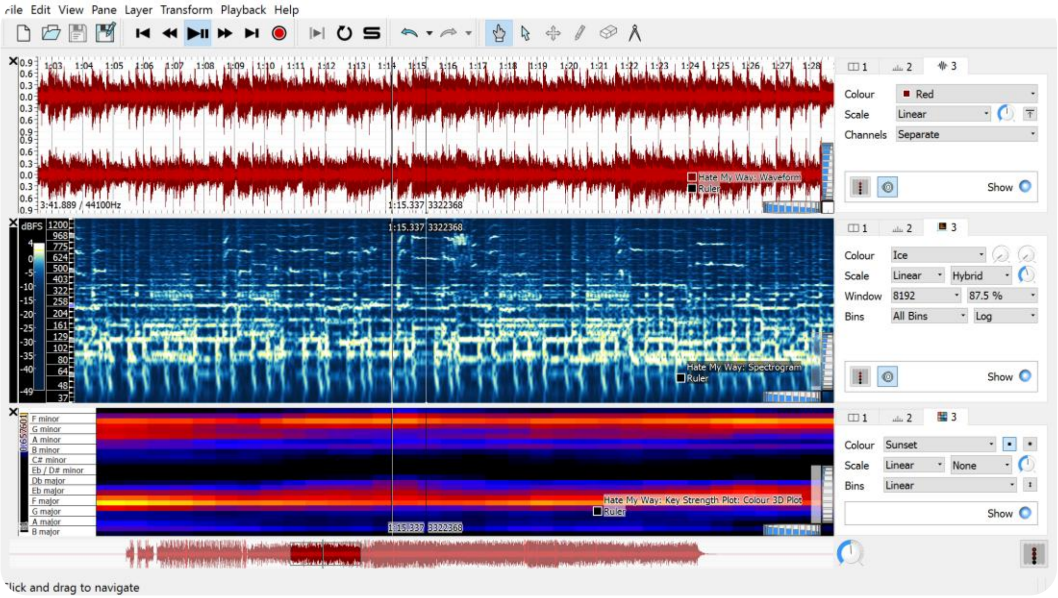
Software

Audacity



(Source: audacity-2.3.1 via Internet Archive)

Sonic Visualiser

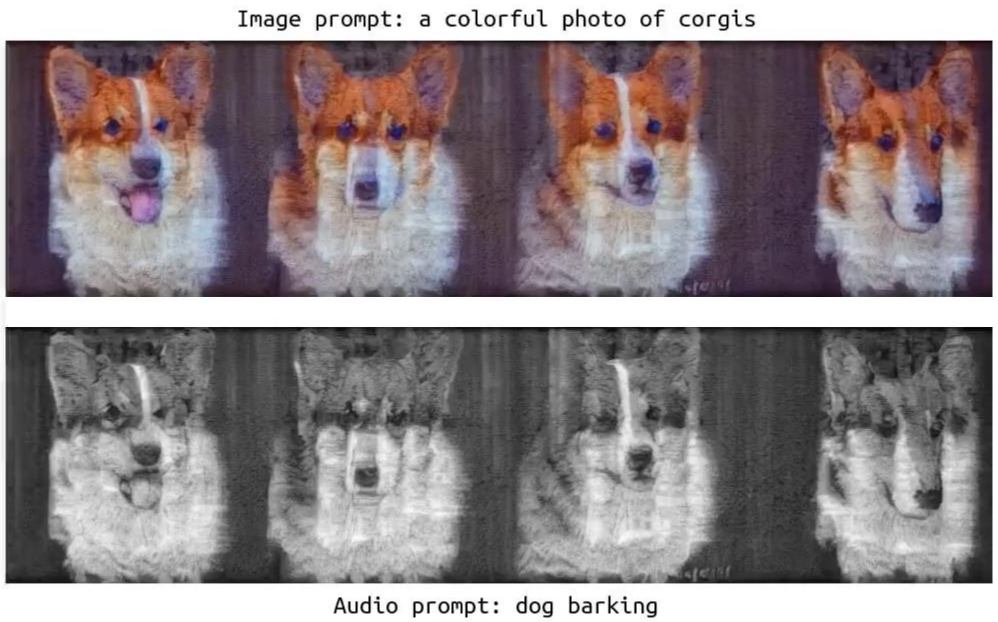


(Source: sonicvisualizer.org)

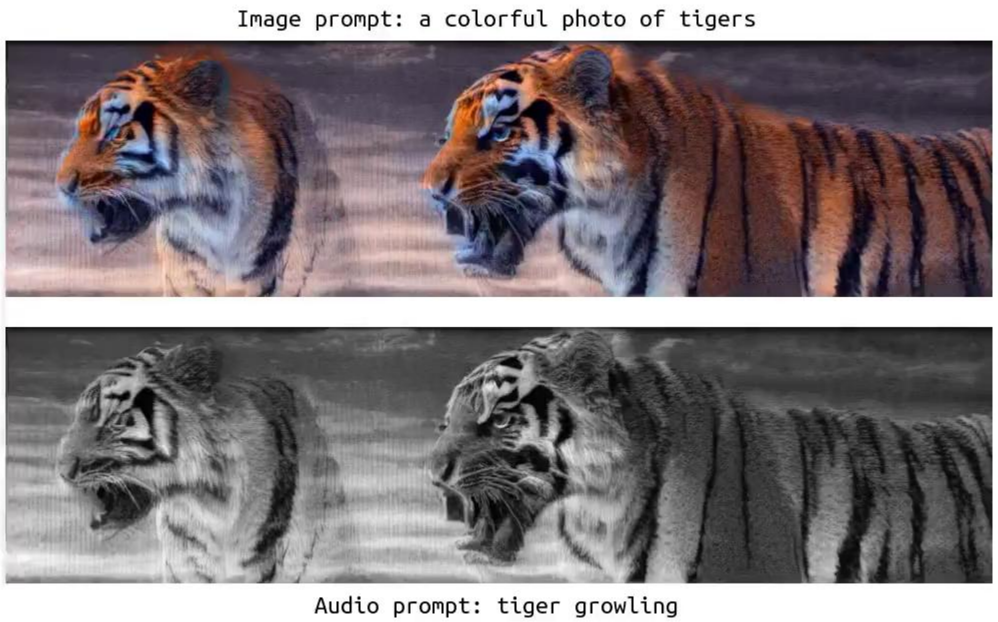
archive.org/details/audacity-2.3.1
sonicvisualiser.org

Images that Sound (Chen et al., 2024)

Using diffusion models to generate visual spectrograms that look like images but can also be played as sound.



(Source: Chen et al., 2024)

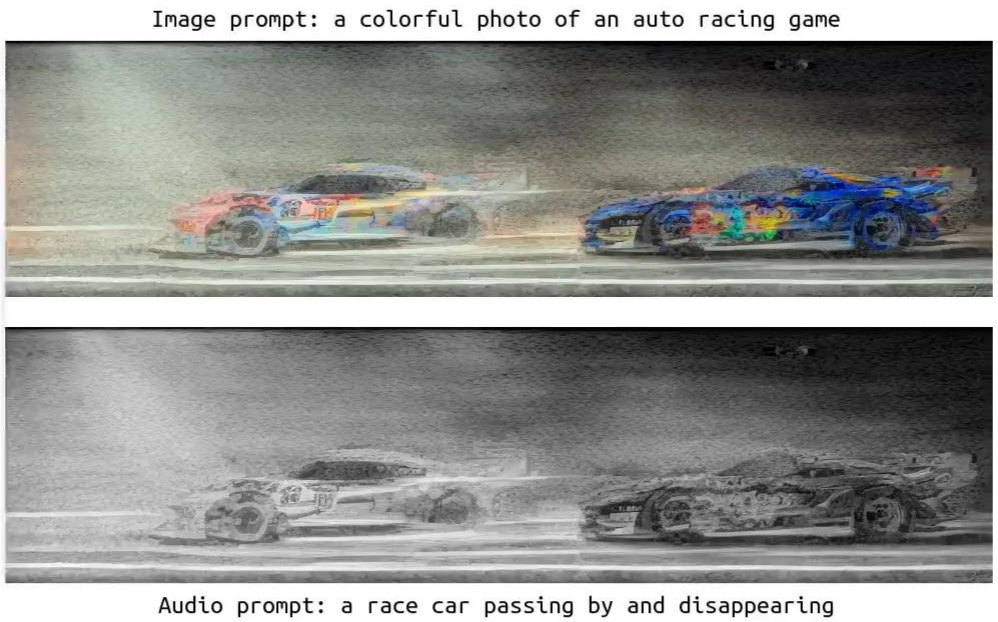


(Source: Chen et al., 2024)

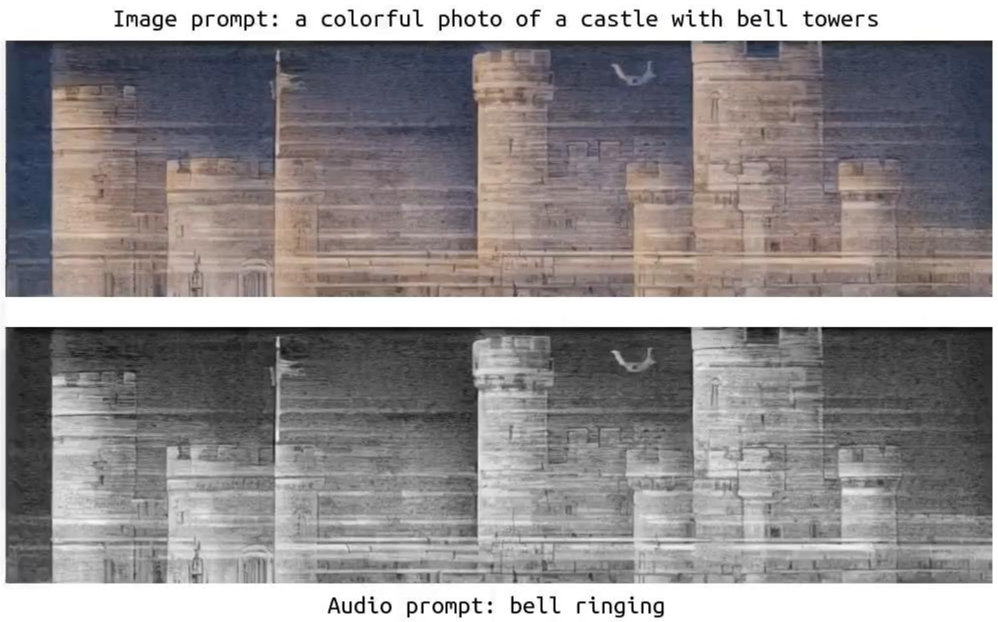
Ziyang Chen, Daniel Geng, and Andrew Owens, "Images that Sound: Composing Images and Sounds on a Single Canvas," *NeurIPS*, 2024.

Images that Sound (Chen et al., 2024)

Using diffusion models to generate visual spectrograms that look like images but can also be played as sound.



(Source: Chen et al., 2024)



(Source: Chen et al., 2024)

Ziyang Chen, Daniel Geng, and Andrew Owens, "Images that Sound: Composing Images and Sounds on a Single Canvas," *NeurIPS*, 2024.

Recap

Four Representative Music Representations

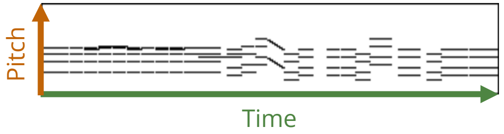
Symbolic music representations

Text-based

Image-based

```
Program_change_0,
Note_on_60, Time_shift_2, Note_off_60,
Note_on_60, Time_shift_2, Note_off_60,
Note_on_76, Time_shift_2, Note_off_67,
Note_on_67, Time_shift_2, Note_off_67,
...
```

MIDI



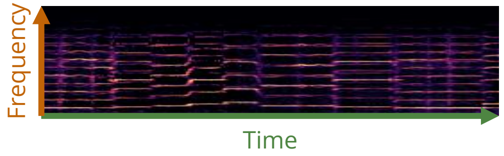
Piano roll

Audio-domain music representations

Time series-based

Image-based

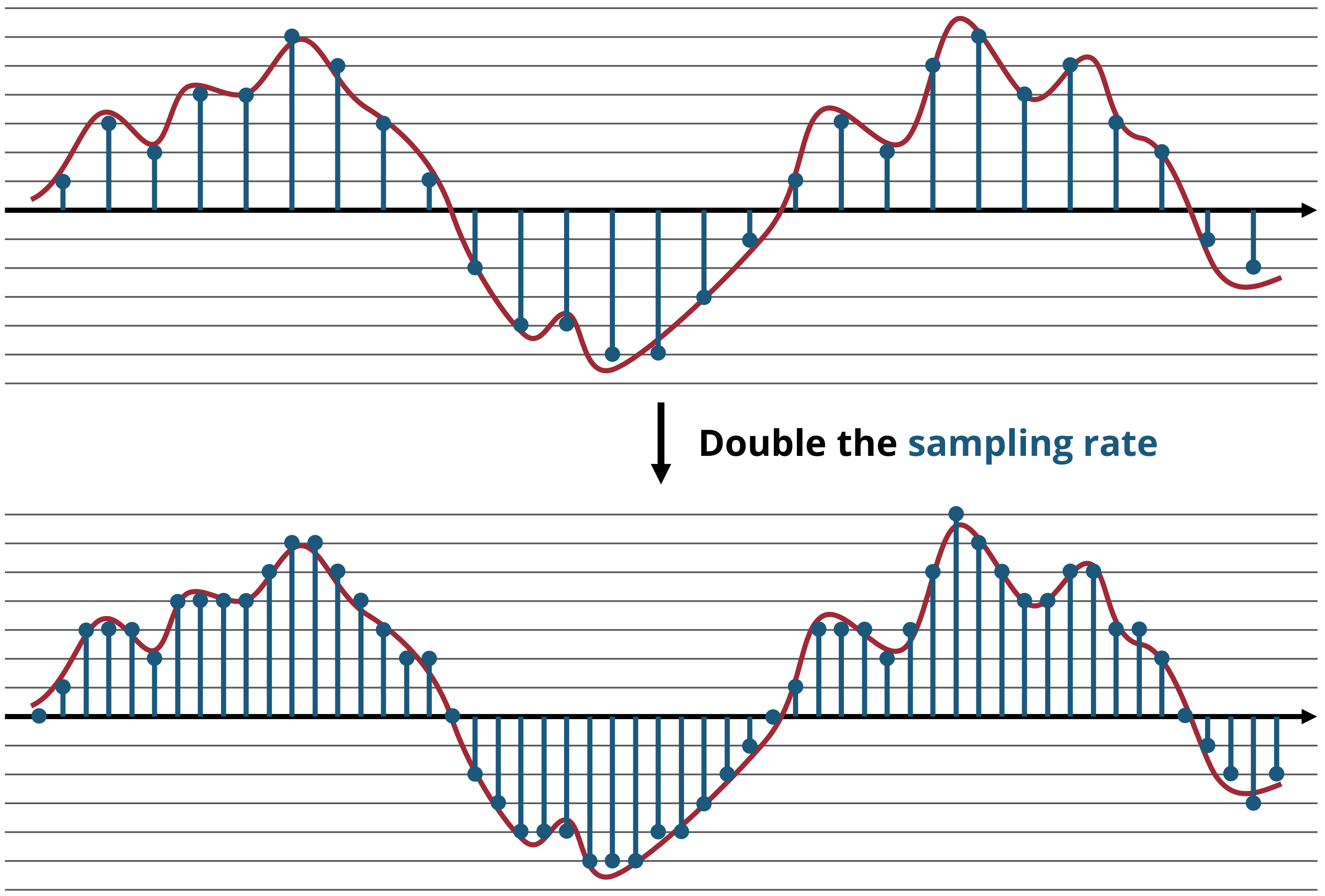
Waveform



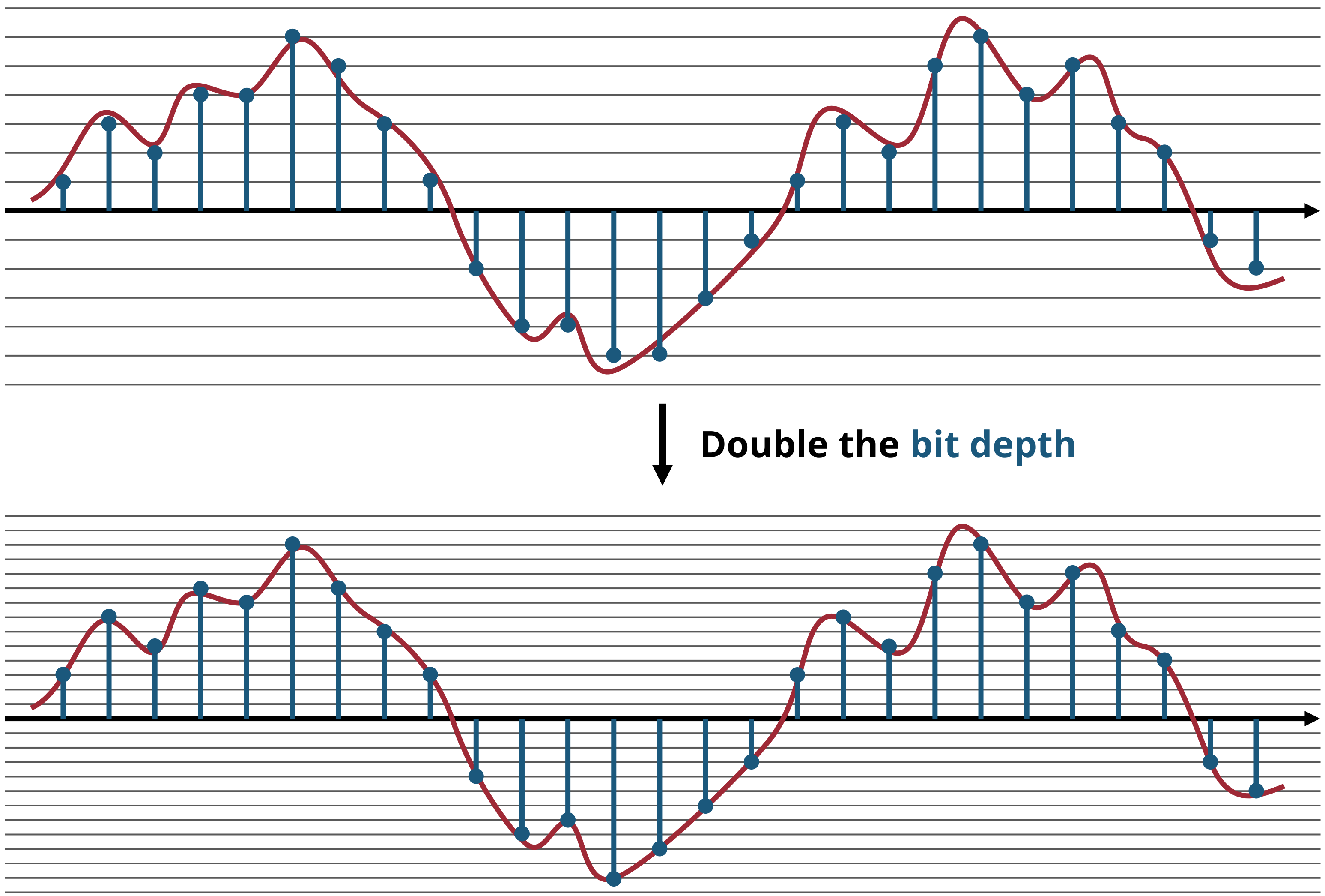
Spectrogram

Today's topic!

Resolution: Sampling Rate



Resolution: Bit Depth

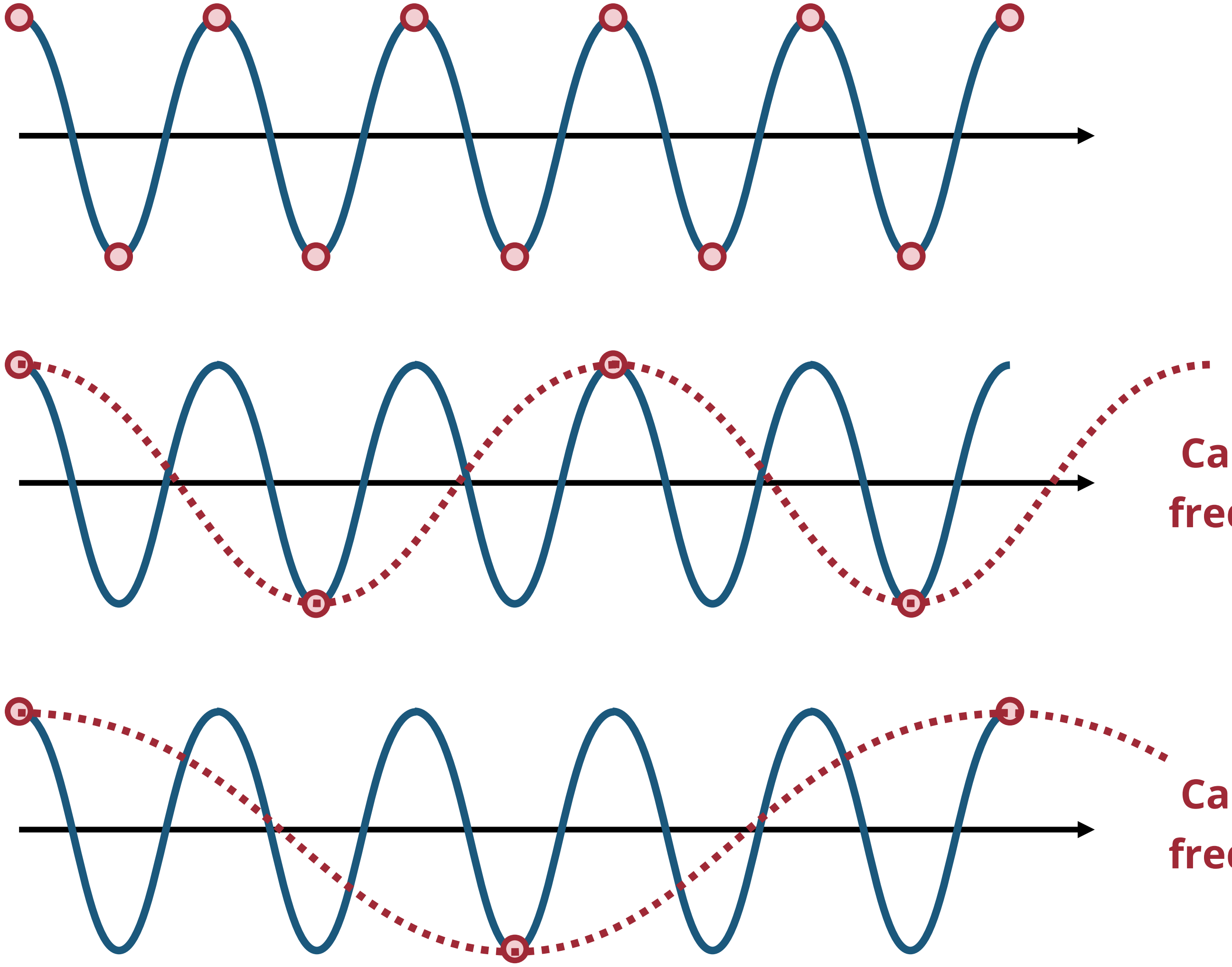


Sampling Theorem: Undersampling

Critically sampled
$(f_s = 2f_{max})$

Undersampled
$(f_s = \frac{2}{3}f_{max})$

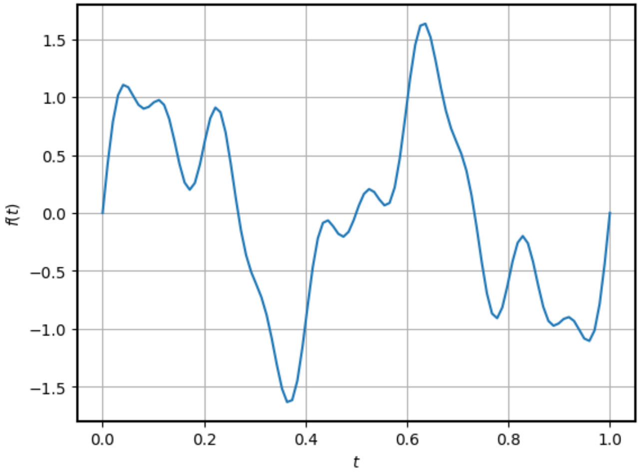
Undersampled
$(f_s = \frac{2}{5}f_{max})$

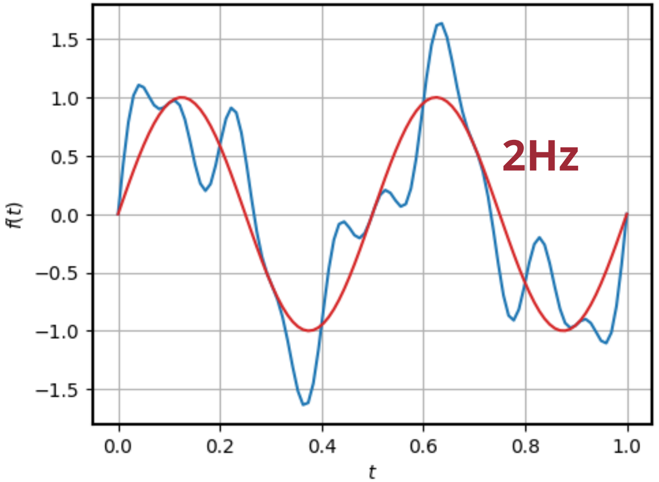


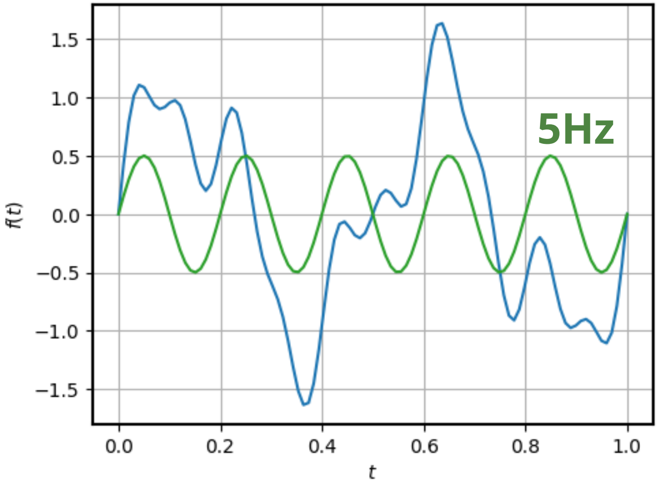
Can only reconstruct frequency up to $\frac{1}{3}f_{max}$

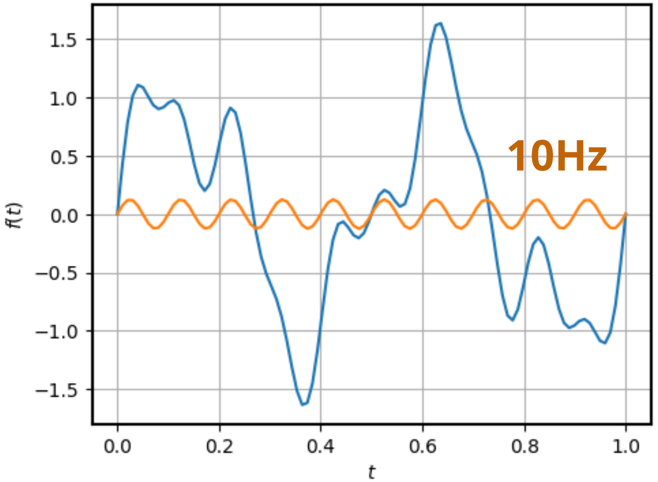
Can only reconstruct frequency up to $\frac{1}{3}f_{max}$

Spectral Analysis



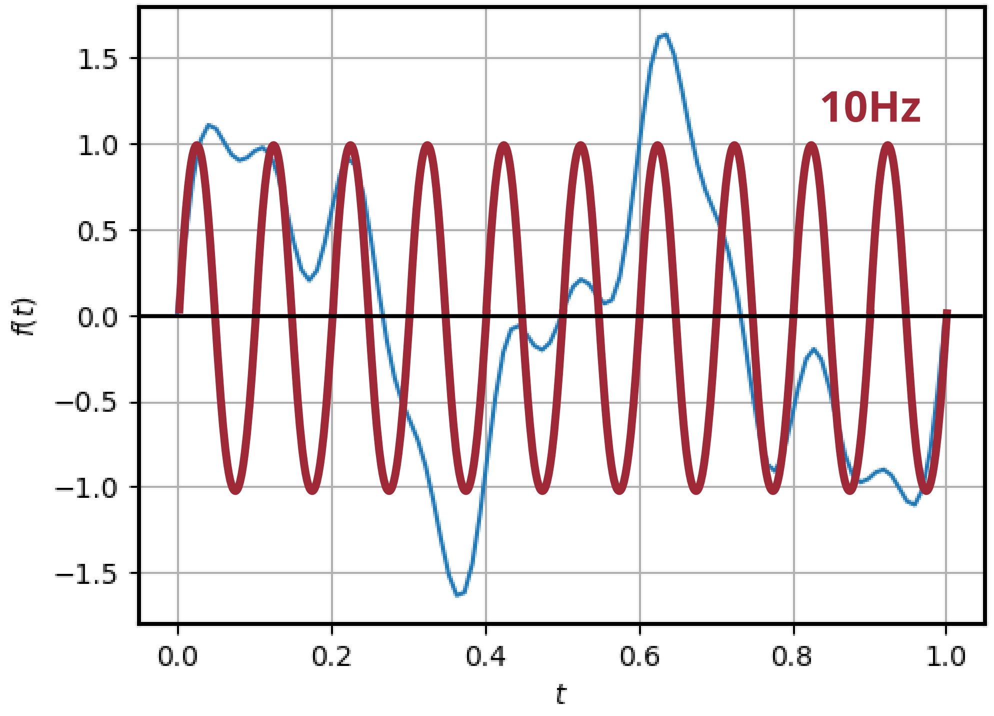






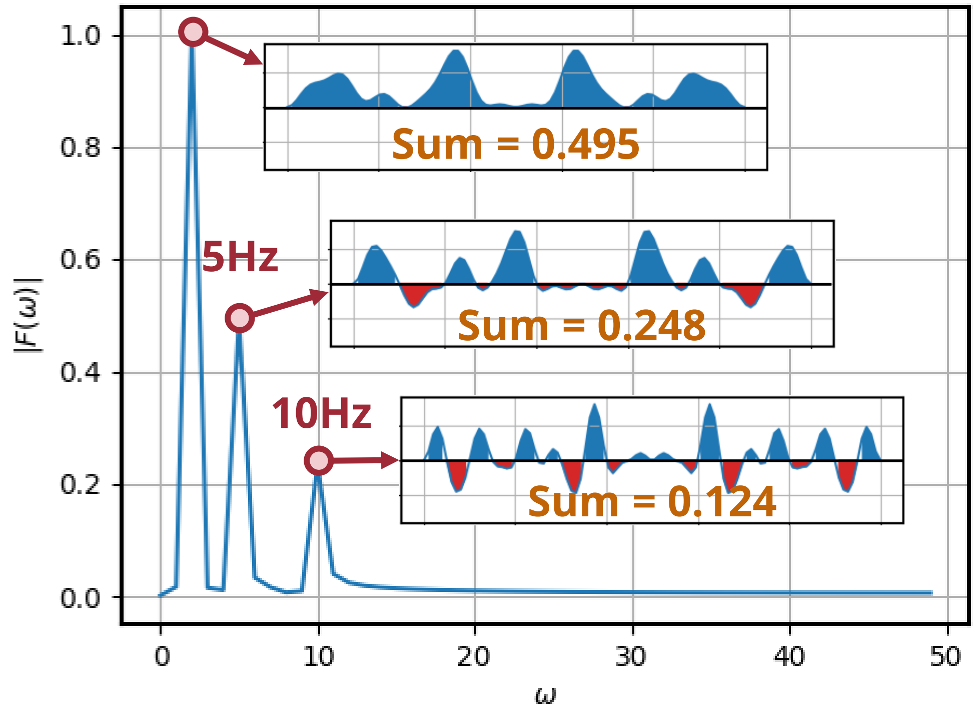
Demystifying Fourier Transform

Signal
(time-domain)



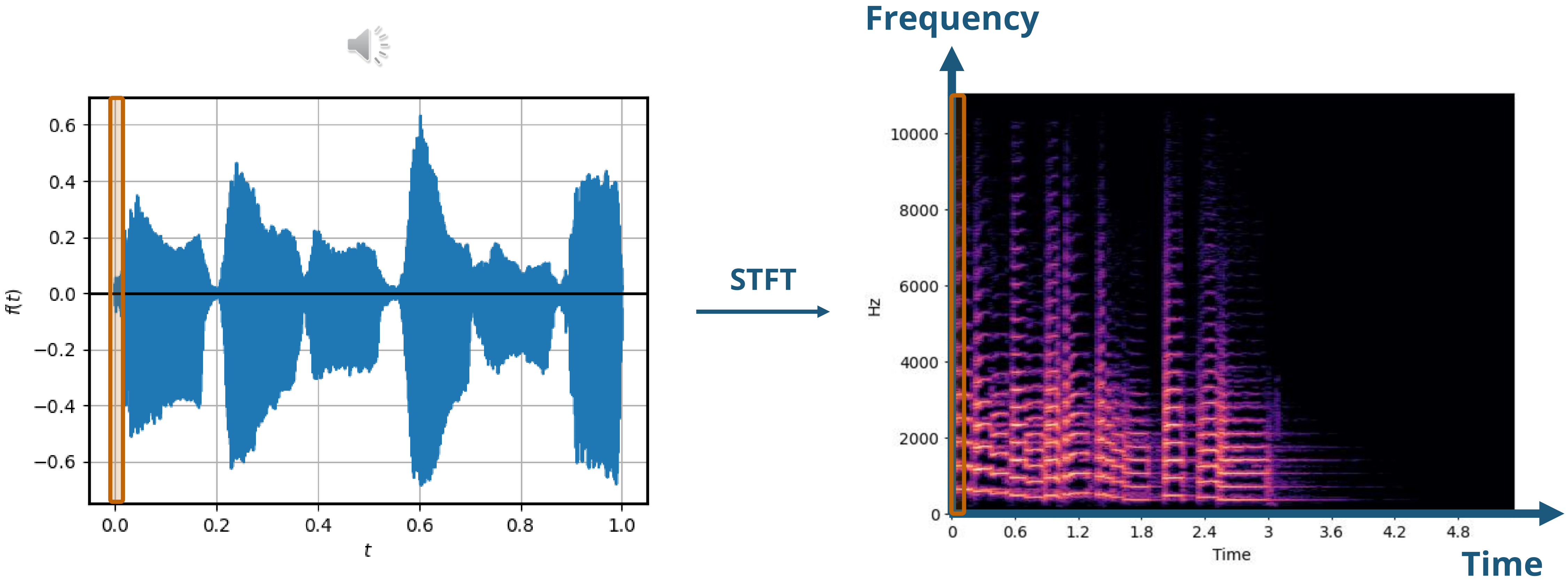
Fourier Transform →

Spectrum
(frequency-domain)

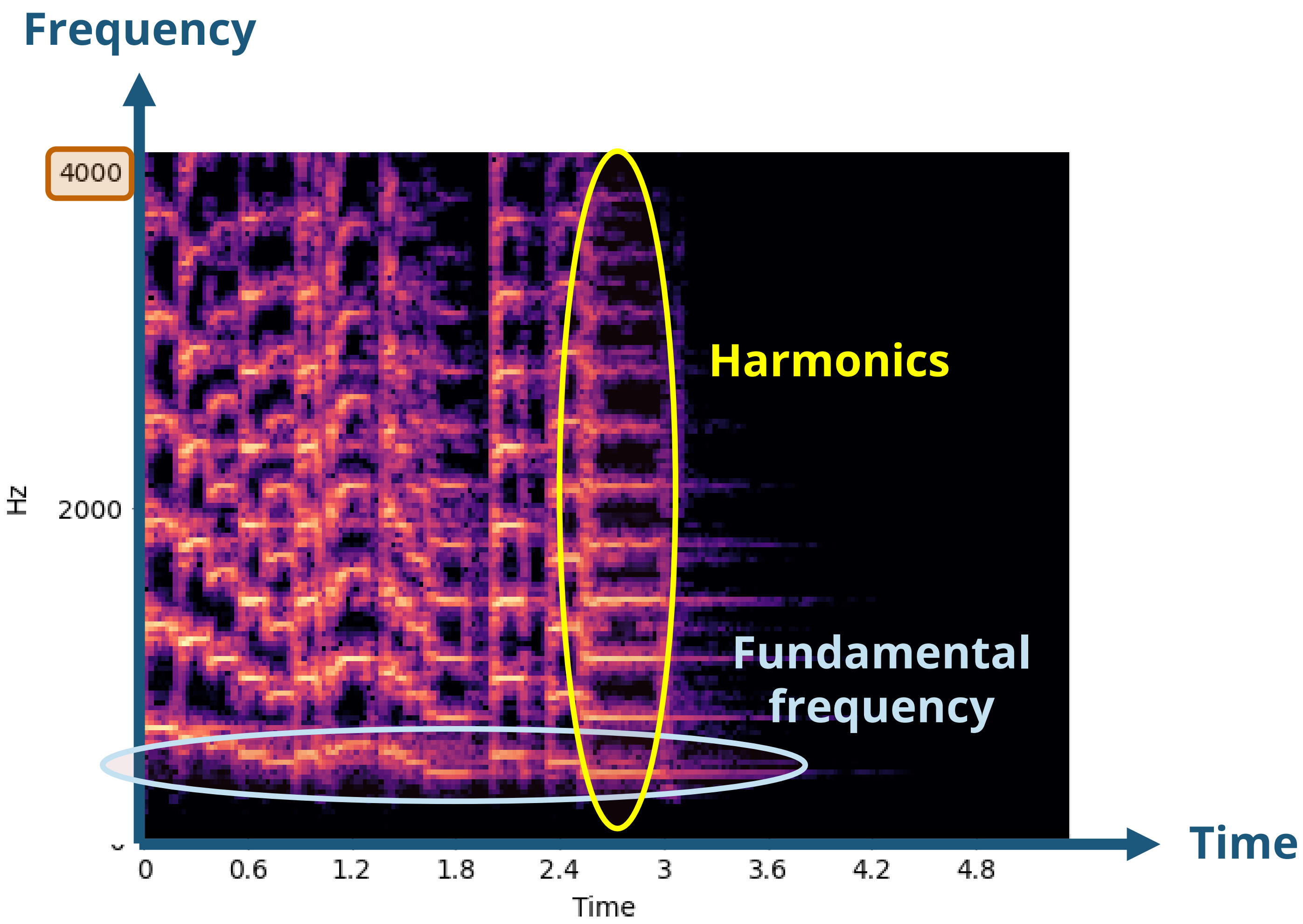


Short-Time Fourier Transform (STFT)

- **Intuition**: Slice the audio into chunks and apply Fourier transform

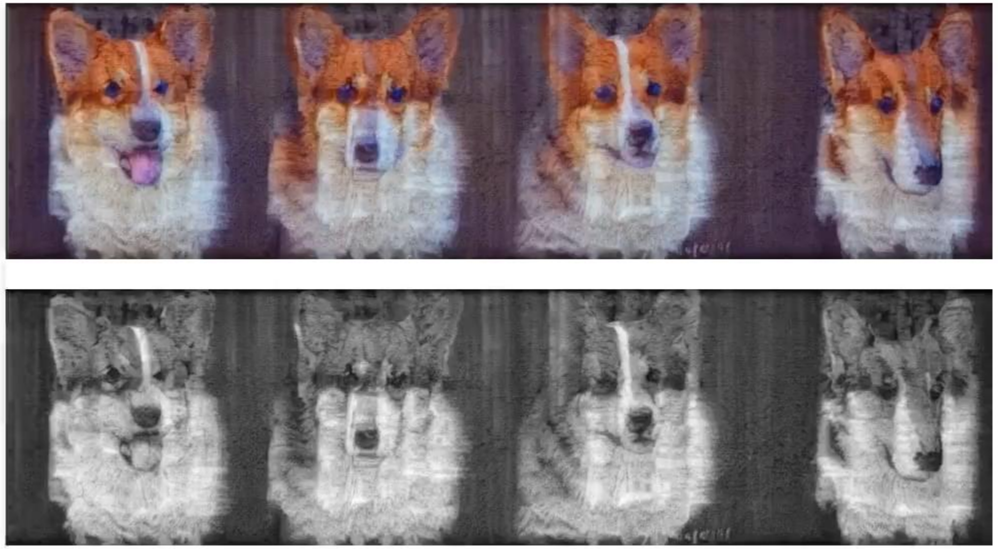


Spectrogram



Images that Sound (Chen et al., 2024)

Using diffusion models to generate visual spectrograms that look like images but can also be played as sound.



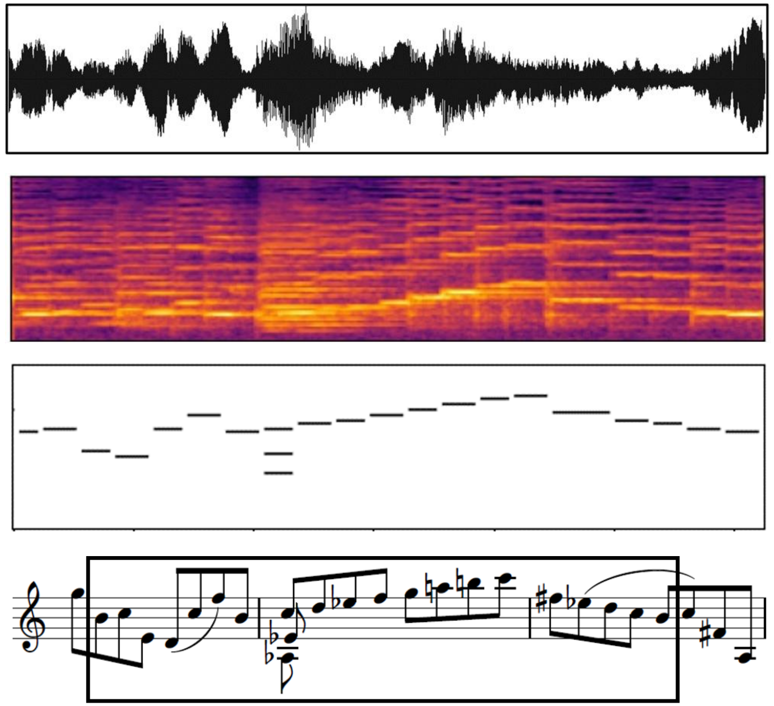
(Source: Chen et al., 2024)

(Source: Chen et al., 2024)

Ziyang Chen, Daniel Geng, and Andrew Owens, "Images that Sound: Composing Images and Sounds on a Single Canvas," *NeurIPS*, 2024.

Next Lecture

Music Analysis



(Source: Dong et al., 2022)