EE269
Signal Processing and Quantization for Machine Learning

Lecture 3 Part II

Instructor : Mert Pilanci

Stanford University

Outline

- Short Time Fourier Transform
- Examples

Recap: Continuous Time vs Discrete Fourier Transform

Continuous Time Fourier Transform

$$X_c(f) = \int e^{-j2\pi ft} dt$$

Discrete Fourier Transform

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-2\pi jkn/N}$$

Short Time Fourier Transform (STFT)

$$X(f,t) = \int w(t-\tau)x(\tau)e^{-j2\pi f\tau}d\tau$$

- $w(t)$: window signal
- Discrete STFT

$$X_{nm} = DFT\{w[nD-k]x[k]\}$$

- D: hop length

Why STFT? (DFT is “global”)

- The DFT assumes the signal is (approximately) **stationary** over the whole block:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}nk}.$$

- Many real signals are **nonstationary**: speech, music, chirps, transient events.
- Idea: compute “local spectra” by applying a **window** and sliding it in time.
- STFT produces a **time-frequency representation**:

$$\text{time index} \times \text{frequency bin} \Rightarrow X[\text{time}, \text{freq}].$$

STFT definition (discrete-time)

Let $x[n]$ be a discrete-time signal and $g[n]$ a window of length L.

STFT (sample-shift form):

$$X_x[\tau, k] \;=\; \sum_{n=-\infty}^{\infty} x[n]\, g[n-\tau]\, e^{-j\frac{2\pi}{N}kn}, \quad k = 0, \ldots, N-1$$

where τ is the window center (in samples), and N is the FFT size.

Frame/hop form (typical in code): take $\tau = mH$ with hop size H,

$$X[m, k] \;=\; \sum_{n=0}^{L-1} x[n+mH]\, g[n]\, e^{-j\frac{2\pi}{N}kn}.$$

- L (window length) controls **time localization**
- N (FFT size) controls **frequency grid** (via zero-padding if $N > L$)
- H (hop) controls **redundancy / overlap**

Spectrogram and axis mapping

- The most common visualization is the **spectrogram**:

  $$S[m,k] = |X[m,k]|^2 \qquad \text{(often shown in dB: } 10\log_{10}(S[m,k]+\epsilon)\text{)}.$$

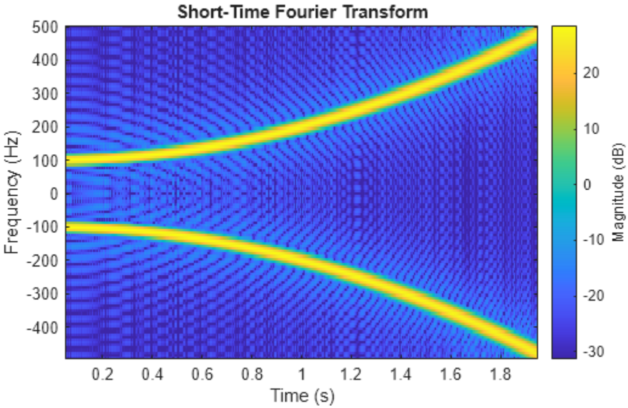
- If sampling rate is f_s (Hz), then

  $$t_m = \frac{mH}{f_s} \text{ seconds}, \qquad f_k = \frac{kf_s}{N} \text{ Hz}.$$

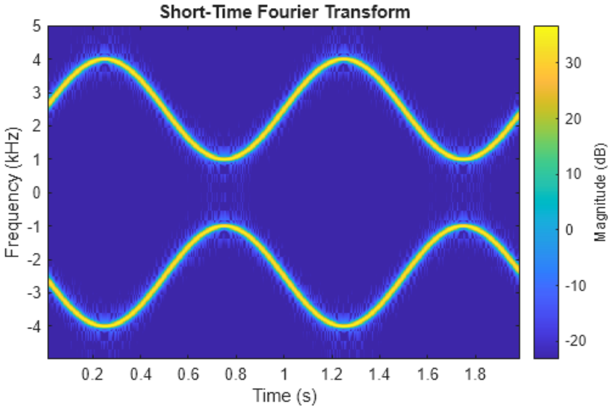
- For **real** signals, you often store only $k = 0, \dots, \lfloor N/2 \rfloor$ due to conjugate symmetry (numpy `rfft` does this).

Interpretation: each column (fixed m) is a DFT of a *windowed segment* of $x[n]$.

Examples of Short Time Fourier Transforms



Examples of Short Time Fourier Transforms



Choosing parameters: time–frequency tradeoff

- Approximate time resolution (seconds):

$$\Delta t \approx \frac{L}{f_s}.$$

- Approximate frequency-bin spacing (Hz):

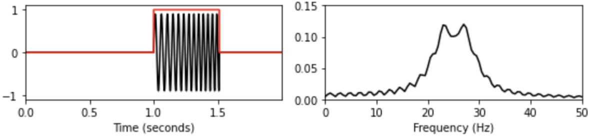
$$\Delta f = \frac{f_s}{N}.$$

- Longer window ($L \uparrow$): better frequency resolution, worse time resolution.
- Shorter window ($L \downarrow$): better time localization, more spectral leakage.
- Typical audio/speech heuristic: $L \approx 20$–40 ms, overlap 50–75%.
- Window choice matters (leakage vs mainlobe width): Hann, Hamming, Blackman, ...

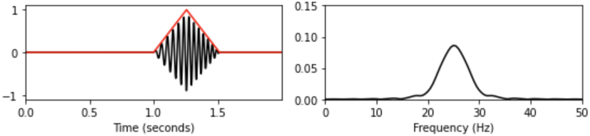
Key point: STFT is *not* a perfect representation; it is a controlled compromise.

Choosing the window type

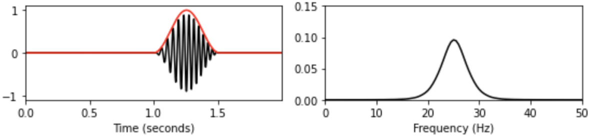
`Rectangular window:`



`Triangular window:`



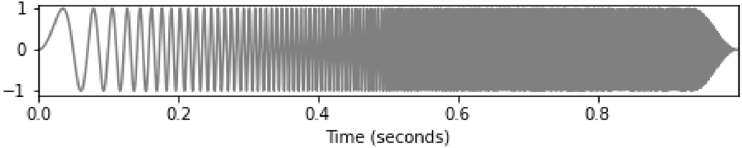
`Hann window:`



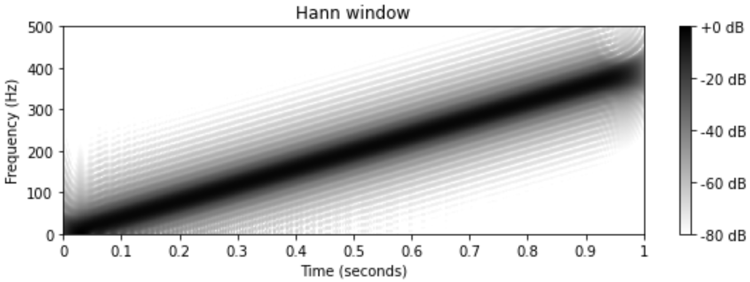
- Hann window: $g[n] = \frac{1}{2}\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right)$

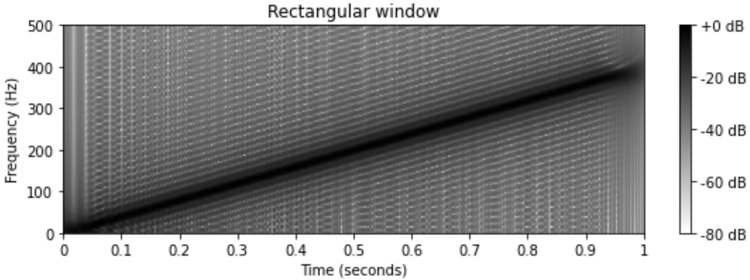
Ripple artifacts

- simple chirp: $x(t) = \sin(400\pi t^2)$ sampled at $f_s = 4000$



Ripple artifacts

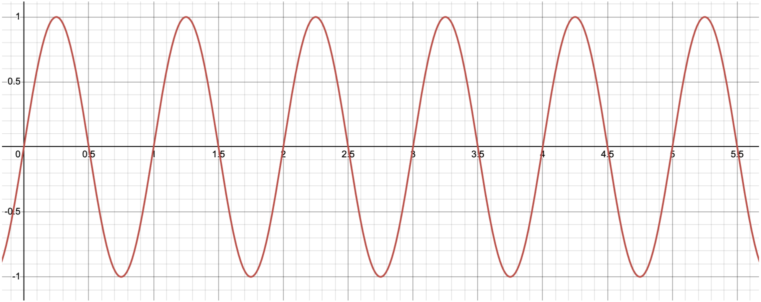




- rectangular window leads to stronger artifacts (diagonals)

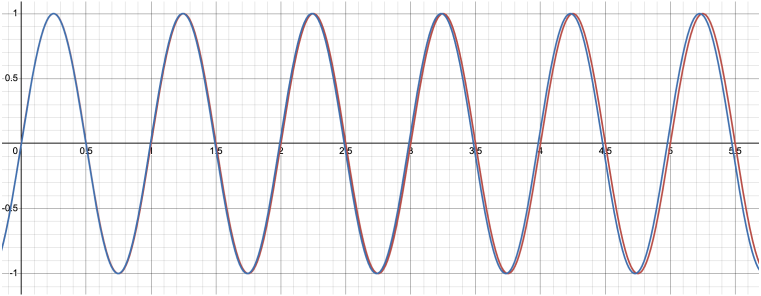
Two sinusoids

- $x_1(t) = \sin(2\pi t f_1)$ where $f_1 = 440\text{Hz}$



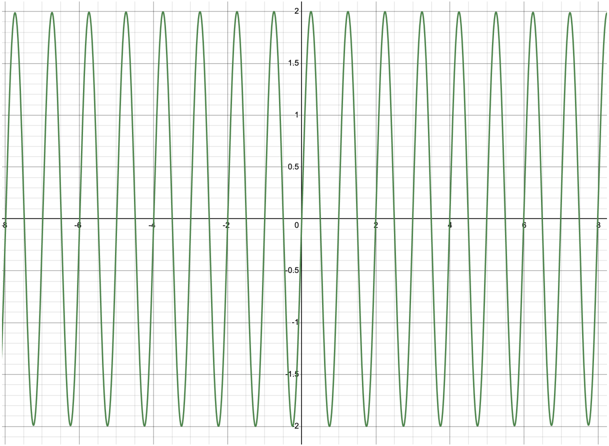
Two sinusoids

- $x_1(t) = \sin(2\pi t f_1)$ where $f_1 = 440\text{Hz}$
- $x_2(t) = \sin(2\pi t f_2)$ where $f_2 = 442\text{Hz}$



Sum of two sinusoids sinusoids

- $x_1(t) = \sin(2\pi t f_1)$ where $f_1 = 440$Hz
- $x_2(t) = \sin(2\pi t f_2)$ where $f_2 = 442$Hz
- $x_1(t) + x_2(t) = ?$



(not the full range - zoomed in)

A trigonometric formula

- Trigonometric identity:

$$\sin a + \sin b = 2 \sin\left(\frac{a+b}{2}\right) \cos\left(\frac{a-b}{2}\right)$$

Sum of two sinusoids

- When two sinusoids with nearby frequencies are added:

$$\sin(\omega_1 t) + \sin(\omega_2 t) = 2\sin\left(\frac{\omega_1 + \omega_2}{2}t\right)\cos\left(\frac{\omega_1 - \omega_2}{2}t\right)$$

- This can be viewed as a high-frequency carrier

$$\sin\left(\frac{\omega_1 + \omega_2}{2}t\right)$$

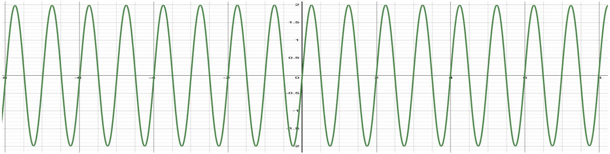
  modulated by a slowly varying envelope

$$\pm 2\cos\left(\frac{\omega_1 - \omega_2}{2}t\right)$$

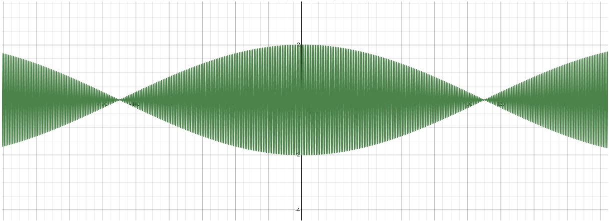
- The sign is perceptually irrelevant: two tones with the same frequency but different phase sound identical
- What we hear is:
  - the **average frequency** $\frac{\omega_1+\omega_2}{2}$
  - with an **amplitude modulation** (beats) at frequency $\frac{|\omega_1-\omega_2|}{2\pi}$
- This amplitude ripple is the origin of audible beating and visual ripple artifacts in time-frequency representations

Sum of two sinusoids sinusoids

- $x_1(t) + x_2(t)$ zoomed in

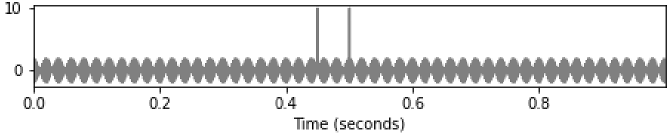


- $x_1(t) + x_2(t)$ zoomed out

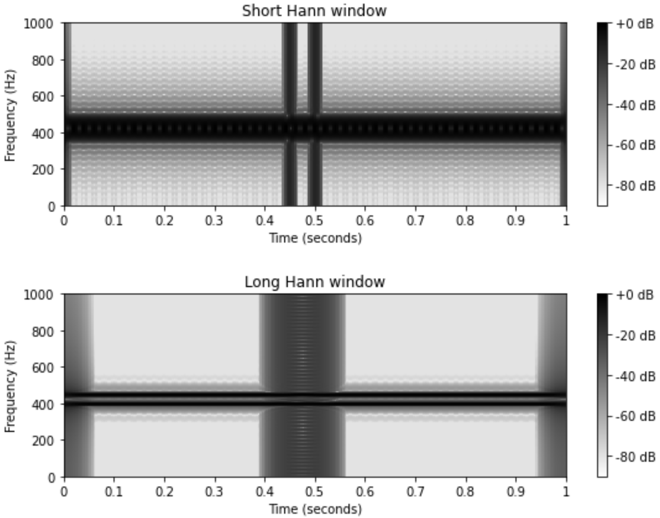


Choosing the window duration

- sinusoids and impulses:
  $x(t) = f(t) = \sin(800\pi t) + \sin(900\pi t) + \delta(t - 0.45) + \delta(t - 0.5)$.
  sampled at $f_s = 4000$



Choosing the window duration



- short window separates the impulses not the sinusoids
- long window separates the sinusoids not the impulses

Useful STFT properties (good to remember)

Using the sample-shift definition
$X_x[\tau, k] = \sum_n x[n] g[n-\tau] e^{-j\frac{2\pi}{N}kn}$:

- **Linearity:** $X_{\alpha x+\beta y}[\tau, k] = \alpha X_x[\tau, k] + \beta X_y[\tau, k]$.
- **Time shift:** if $y[n] = x[n-n_0]$ then

$$X_y[\tau, k] = e^{-j\frac{2\pi}{N}kn_0}\ X_x[\tau - n_0, k].$$

- **Modulation / frequency shift:** if $y[n] = x[n]e^{j\frac{2\pi}{N}k_0 n}$ then

$$X_y[\tau, k] = X_x[\tau, (k-k_0) \mod N].$$

- **Real-signal symmetry:** if $x[n]$ and $g[n]$ are real,

$$X_x[\tau, N-k] = X_x[\tau, k]^* \quad \Rightarrow \quad |X| \text{ is symmetric.}$$

- **Windowing = smoothing in frequency:** multiplication by g in time corresponds to convolution with G in frequency (explains leakage).

STFT as inner products with time–frequency atoms

Define the **time–frequency atom**

$$g_{m,k}[n] \triangleq w[n - mR]\, e^{+j\frac{2\pi}{N}kn}.$$

Then the STFT coefficient is an inner product:

$$\boxed{X[m,k] = \langle x, g_{m,k} \rangle \; = \; \sum_n x[n]\, g_{m,k}[n]^*}$$

- Each $g_{m,k}$ is a **localized sinusoid**: a windowed complex exponential.
- The collection $\{g_{m,k}\}$ is a **Gabor system** (time shifts + frequency shifts).

Computation: the core algorithm

Let number of frames be $M = \left\lfloor \frac{N_x - L}{H} \right\rfloor + 1$.

- For each frame $m = 0, \ldots, M-1$:
  1. Extract segment: $x_m[n] = x[n + mH]$, $n = 0, \ldots, L-1$
  2. Window: $\tilde{x}_m[n] = x_m[n]\, g[n]$
  3. Zero-pad to length N if needed
  4. FFT: $X[m,k] = \sum_{n=0}^{N-1} \tilde{x}_m[n] e^{-j\frac{2\pi}{N}kn}$
- Complexity: M FFTs $\Rightarrow$ $\mathcal{O}(M\, N \log N)$.

Memory view: STFT is a matrix $X \in \mathbb{C}^{M \times N}$ (or $M \times (N/2+1)$ for real FFT).

Python: STFT in practice (NumPy / SciPy)

```
import numpy as np
from scipy.signal import stft

fs = 16000                  # sampling rate
L  = 512                    # window length
H  = 128                    # hop (overlap = L - H)
N  = 512                    # FFT size (can set N > L for zero-pad

# x: 1D numpy array, shape (Nx,)
f, t, Z = stft(x, fs=fs, nperseg=L, noverlap=L-H, nfft=N,
               window="hann", boundary=None, padded=False)

S = np.abs(Z)**2            # spectrogram (power)
S_db = 10*np.log10(S + 1e-10)

# Z has shape: (freq_bins, time_frames)
# f in Hz, t in seconds
```

- Z is complex: magnitude = energy, angle = local phase.
- Most ML pipelines use `log` power (often plus mel-filterbank).

Inverse STFT (iSTFT) and overlap-add

Each frame has an inverse FFT:

$$\tilde{x}_m[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[m,k] e^{j\frac{2\pi}{N}kn}.$$

Reconstruction is done by **overlap-add**:

$$\hat{x}[n] = \frac{\sum_m g[n-mH]\,\tilde{x}_m[n-mH]}{\sum_m g^2[n-mH]+\epsilon}.$$

- Perfect (or near-perfect) reconstruction requires a consistency condition.
- Common practical condition: the shifted windows satisfy a constant-sum property for an analysis/synthesis window pair.

STFT in ML: from waveform to a 2D "image"

Audio ML pipelines often map 1D waveform $\to$ 2D feature map:

$$x[n] \xrightarrow{\text{STFT}} X[m,k] \xrightarrow{\text{magnitude}} |X[m,k]| \xrightarrow{\log} \log(|X| + \epsilon).$$

- The result looks like an image (time $\times$ frequency) $\Rightarrow$ convolutional neural networks and vision transformers work well.
- This is the basic idea behind many speech / audio classification systems.

Mel spectrogram: perceptual frequency warping (details later)

A common variant is the **log-mel spectrogram**:

$$|X[m,k]|^2 \xrightarrow{\text{mel filter bank}} M[m,\ell] \xrightarrow{\log} \log(M[m,\ell]+\epsilon).$$

- Mel filters average energy across frequency bands (roughly perceptual resolution).
- Reduces dimensionality and de-emphasizes very fine harmonic structure.
- We'll see linear system theory to understand mel filters

Interpretation: linear filter bank + downsample in frequency + log compression.

Differentiable STFT (deep learning view)

STFT is a *linear* transform:

$$X = \mathcal{A}x$$

for a suitable linear operator $\mathcal{A}$ (depends on w, N, R).

- In frameworks (PyTorch/JAX), STFT is differentiable.
- You can backpropagate through the feature extractor.
- Used in speech enhancement, audio generation, and self-supervised learning.

Spectral losses: training models using STFT distance

In audio generation/enhancement, a common loss compares STFTs:

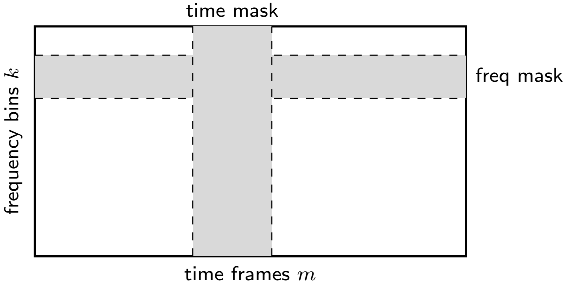
$$\mathcal{L}_{\mathsf{mag}} = \sum_{m,k} \Big|\, |X[m,k]| - |\hat{X}[m,k]| \,\Big|, \qquad \mathcal{L}_{\mathsf{log}} = \sum_{m,k} \Big| \log|X| - \log|\hat{X}| \Big|.$$

- Multi-resolution STFT loss: sum across several (N, R) settings.
- Encourages correct structure at multiple time-frequency scales.

Augmentations in the time–frequency plane: SpecAugment

A simple and effective augmentation for spectrogram-based models:

- Augmentation is a technique that synthetically expands the training data by transformations
- **frequency mask:** zero out (or mean-fill) a random band of frequencies
- **time mask:** zero out a random span of time frames
- intuition: encourages invariance to missing bands / missing time intervals



Mask-based speech enhancement / source separation

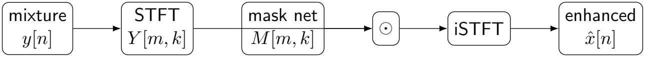
Many systems operate in the STFT domain and predict a *mask*.
For a mixture $y[n] = x[n] + v[n]$:

$$Y[m,k] = \mathsf{STFT}\{y\}, \qquad X[m,k] = \mathsf{STFT}\{x\}.$$

A network predicts $M[m,k]$ and forms

$$\widehat{X}[m,k] = M[m,k] \odot Y[m,k], \qquad \widehat{x}[n] = \mathsf{iSTFT}\{\widehat{X}\}.$$

- Magnitude masks: $M \in [0,1]$ (simplest, ignores phase updates).
- Complex masks: $M \in \mathbb{C}$ (can correct both magnitude and phase).
- iSTFT is differentiable $\Rightarrow$ you can train ML models using gradient descent.



Applications in ML/DL: where STFT appears

The STFT is a workhorse representation whenever a signal is *locally sinusoidal*.

- **Audio classification:** keyword spotting, acoustic scene, music tagging

$$x[n] \;\rightarrow\; \text{log-mel spectrogram} \;\rightarrow\; \text{CNN/Transformer}.$$

- **Automatic speech recognition (ASR):** log-mel + SpecAugment + sequence model.
- **Enhancement / separation:** operate on $X[m,k]$ (or $|X|$), then iSTFT to waveform.
- **Audio generation:** vocoders / diffusion models often use mel features and STFT losses.
- **Multi-mic audio:** STFT phase differences enable beamforming and spatial features.

Beyond speech: time–frequency features for general signals

The same idea (1D signal $\rightarrow$ 2D time–frequency map) appears widely:

- vibration / machinery fault detection (bearing faults, motor monitoring)
- biomedical: ECG/EEG time–frequency patterns
- radar/sonar: micro-Doppler signatures
- seismology: nonstationary events

Once you have a spectrogram-like map, you can reuse vision architectures (CNNs, ViTs, U-Nets).

References

- https://www.mathworks.com/help/audio/ug/spectral-descriptors.html
- Murthy, H.a., F. Beaufays, L.p. Heck, and M. Weintraub. "Robust Text-Independent Speaker Identification over Telephone Channels." IEEE Transactions on Speech and Audio Processing. Vol. 7, Issue 5, 1999, pp. 554–568.
- Peeters, G. "A Large Set of Audio Features for Sound Description (Similarity and Classification) in the CUIDADO Project." Technical Report; IRCAM: Paris, France, 2004.
- Grey, John M., and John W. Gordon. "Perceptual Effects of Spectral Modifications on Musical Timbres." The Journal of the Acoustical Society of America. Vol. 63, Issue 5, 1978, pp. 1493–1500.
- S. Zhang, Y. Guo, and Q. Zhang, "Robust Voice Activity Detection Feature Design Based on Spectral Kurtosis." First International Workshop on Education Technology and Computer Science, 2009, pp. 269–272.
- Hansen, John H. L., and Sanjay Patil. "Speech Under Stress: