Fast Fourier Transform (FFT) and Short-Time Fourier Transform (STFT)

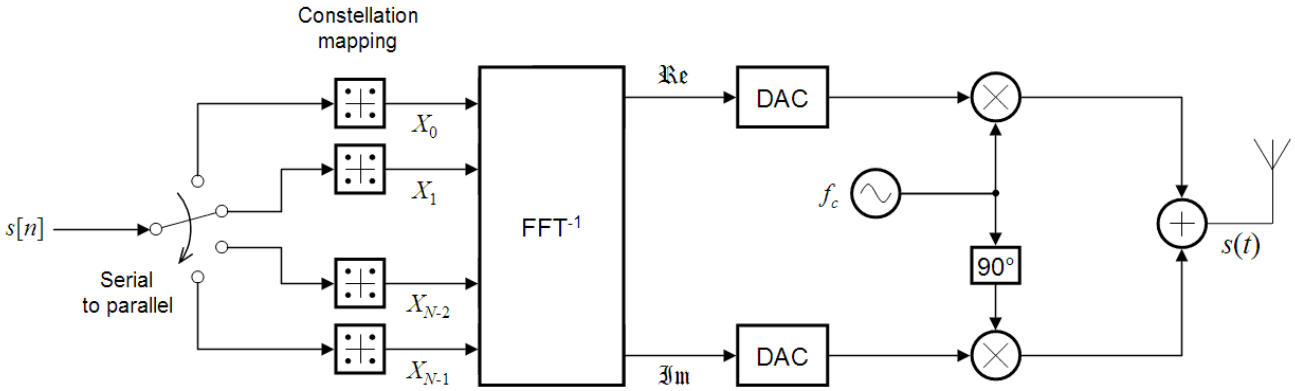
Digital Signal Processing

February 21, 2023

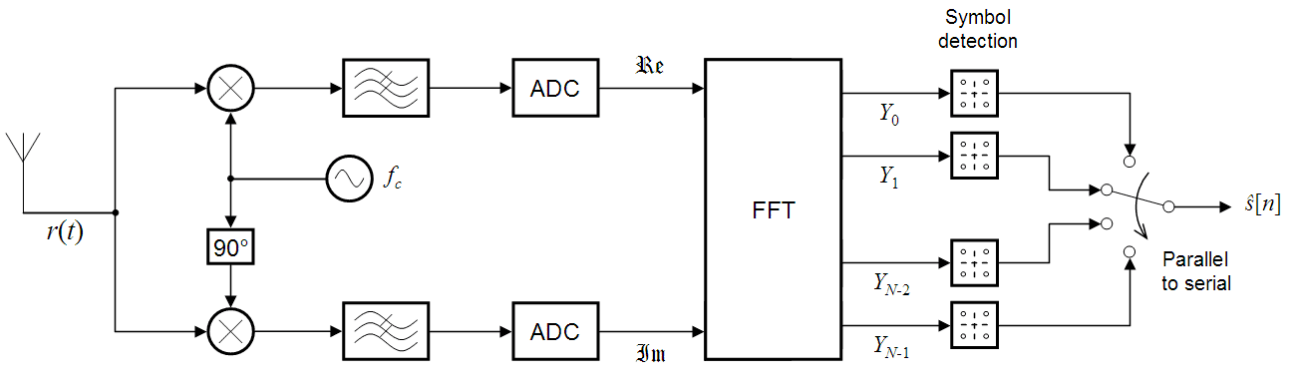
Application of FFT: Communications

Orthogonal Frequency Division Multiplexing (OFDM)

Transmitter:



Receiver:



Technology behind most digital wireless communication! (WiFi, 4G, 5G, HD Radio)

The Fast Fourier Transform (FFT)

Naïve DFT Algorithm

Recall the DFT equation:

$$X[k] = \frac{1}{\sqrt{L}} \sum_{n=0}^{L-1} e^{-i\omega_0 nk} x[n]$$

Naïve DFT Algorithm

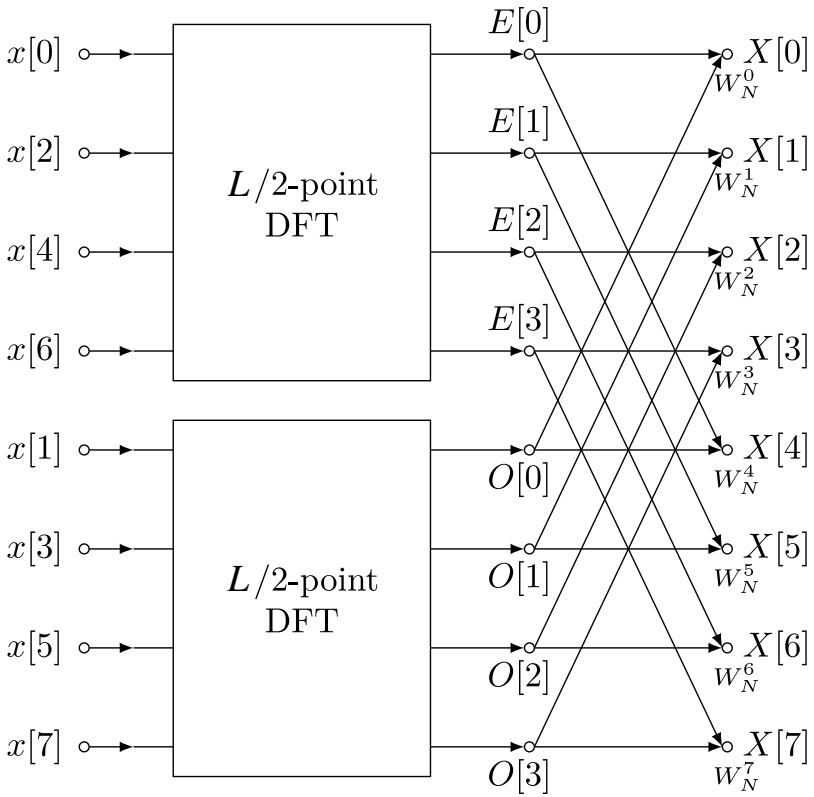
Loop over $k = 0, \ldots, L-1$

Compute $X[k]$ by sum (loop) over $n = 0, \ldots, L-1$

Complexity is $O(L^2)$

Fast Fourier Transform

Cooley-Tukey Algorithm



Divide-and-Conquer

Radix-2 version:

1. compute “even” DFT
2. compute “odd” DFT
3. combine and reuse results

Recursively apply to each $L/2$ block

Complexity is $O(L \log L)$

Fast Fourier Transform

- More general radix-p FFT breaks the DFT into p blocks, where p is a prime factor of the signal length L
- Recursively applied to each L/p block
- Recursion stops when the remaining block lengths are prime numbers (can't be factored any further)
- **Bottom line:** The FFT is most efficient when the input signal length has small prime factors, preferrably L is a power of 2.
- Sometimes it is more efficient to pad a signal with zeros to get a good prime factorization.

First Half of FFT

Compute $X[k]$ for $k = 0, 1, \ldots, \frac{L}{2} - 1$,

$$X[k] = \frac{1}{\sqrt{L}} \sum_{n=0}^{L-1} e^{-i\omega_0 nk} x[n]$$

$$= \frac{1}{\sqrt{L}} \sum_{m=0}^{L/2-1} e^{-i\omega_0 2mk} x[2m] + \frac{1}{\sqrt{L}} \sum_{m=0}^{L/2-1} e^{-i\omega_0 (2m+1)k} x[2m+1]$$

$$= \underbrace{\frac{1}{\sqrt{L}} \sum_{m=0}^{L/2-1} e^{-i\omega_0 2mk} x[2m]}_{E[k]=\text{sum of even terms}} + e^{-i\omega_0 k} \underbrace{\frac{1}{\sqrt{L}} \sum_{m=0}^{L/2-1} e^{-i\omega_0 2mk} x[2m+1]}_{O[k]=\text{sum of odd terms}}$$

$$= E[k] + e^{-i\omega_0 k} O[k]$$

Second Half of FFT

Compute $X[k+\frac{L}{2}]$ for $k = 0, 1, \ldots, \frac{L}{2} - 1$,

$$X\left[k+\frac{L}{2}\right] = \frac{1}{\sqrt{L}} \sum_{m=0}^{L/2-1} e^{-i\omega_0 2m(k+\frac{L}{2})} x[2m] + \\ + e^{-i\omega_0(k+\frac{L}{2})} \frac{1}{\sqrt{L}} \sum_{m=0}^{L/2-1} e^{-i\omega_0 2m(k+\frac{L}{2})} x[2m+1]$$

$$= E[k] - e^{-i\omega_0 k} O[k]$$

Using $e^{-i\omega_0 2m(k+\frac{L}{2})} = e^{-i\omega_0 2mk}$ and $e^{-i\omega_0(k+\frac{L}{2})} = -e^{-i\omega_0 k}$

Second Half of FFT

Compute $X[k+\frac{L}{2}]$ for $k = 0, 1, \ldots, \frac{L}{2} - 1$,

$$X\left[k+\frac{L}{2}\right] = \frac{1}{\sqrt{L}} \sum_{m=0}^{L/2-1} e^{-i\omega_0 2m(k+\frac{L}{2})} x[2m] +$$

$$+ e^{-i\omega_0(k+\frac{L}{2})} \frac{1}{\sqrt{L}} \sum_{m=0}^{L/2-1} e^{-i\omega_0 2m(k+\frac{L}{2})} x[2m+1]$$

$$= E[k] - e^{-i\omega_0 k} O[k] \qquad \text{Reused!}$$

Using $e^{-i\omega_0 2m(k+\frac{L}{2})} = e^{-i\omega_0 2mk}$ and $e^{-i\omega_0(k+\frac{L}{2})} = -e^{-i\omega_0 k}$

FFT Historical Trivia

- FFT actually invented by Gauss in 1805! (but lost)
- Re-invented by Cooley and Tukey in 1965
- Tukey coined the term “bit” (for “binary digit”) and was first to use the term “software” in writing

Carl Friedrich Gauss

John Tukey

The Short-Time Fourier Transform (STFT)

STFT Definition

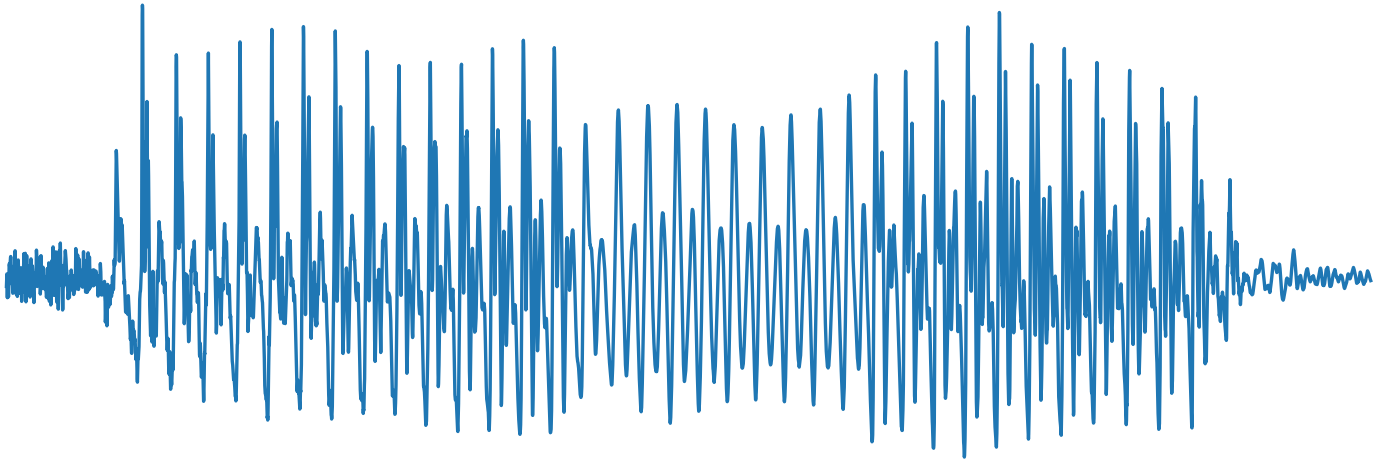
The STFT of a signal, $x[n]$, is a function of frequency, k, and time, n, given by:

$$X[k,n] = \frac{1}{\sqrt{W}} \sum_{m=0}^{W-1} x[m+nh]w[m]e^{-\frac{i2\pi km}{W}},$$

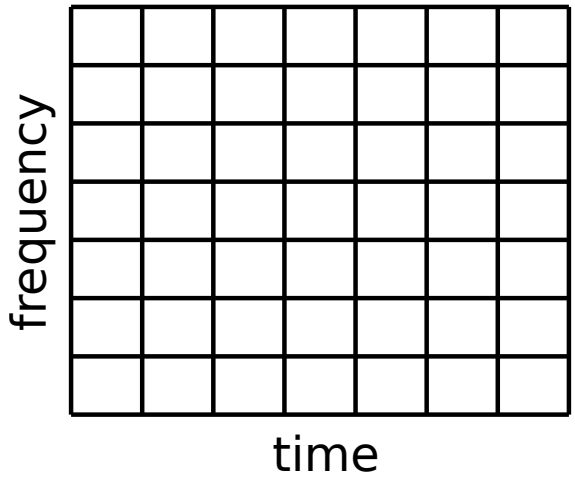
STFT Procedure

Multiply signal $x[n]$ by a sliding window $w[n]$ and take FFT.

$x[n]$



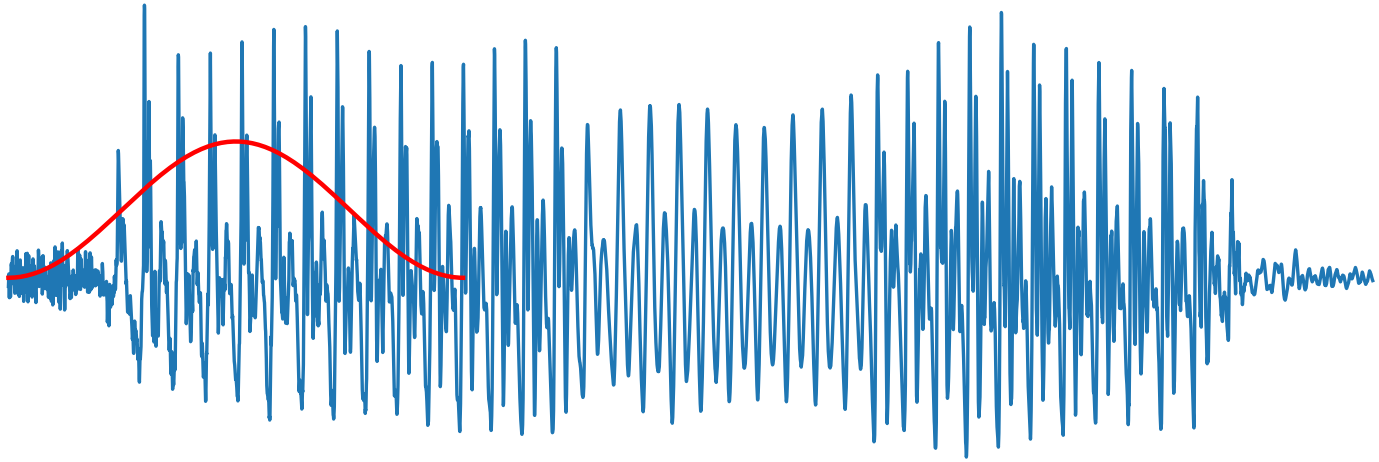
$X[k, n]$



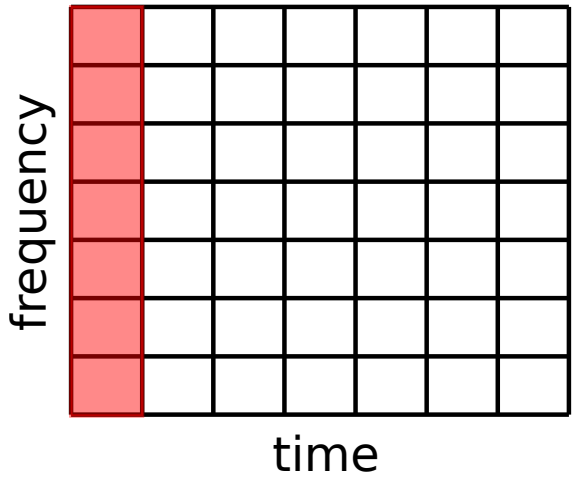
STFT Procedure

Multiply signal $x[n]$ by a sliding window $w[n]$ and take FFT.

$x[n]$



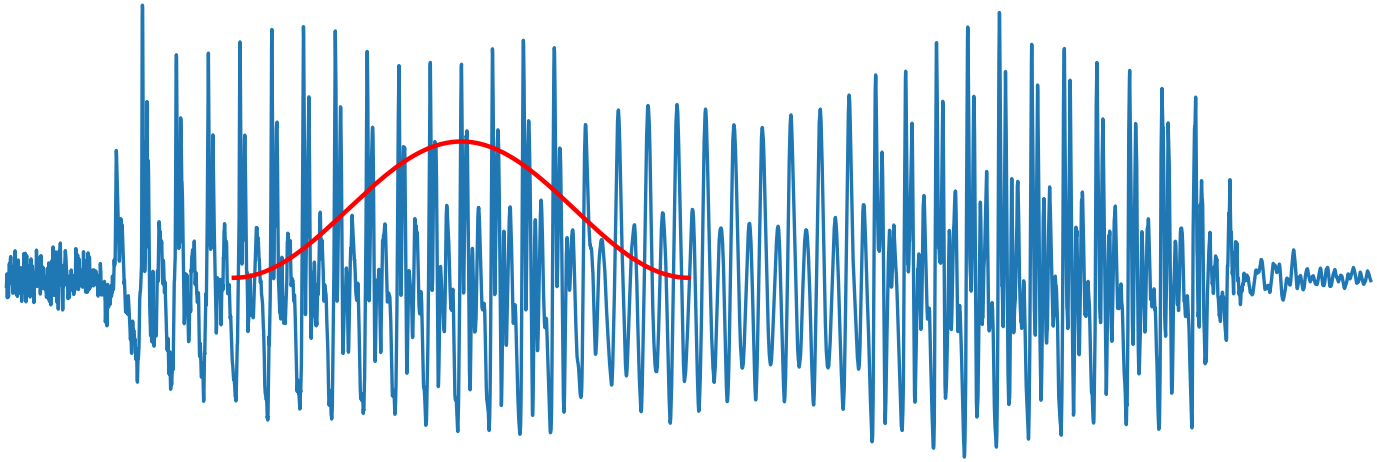
$X[k, n]$



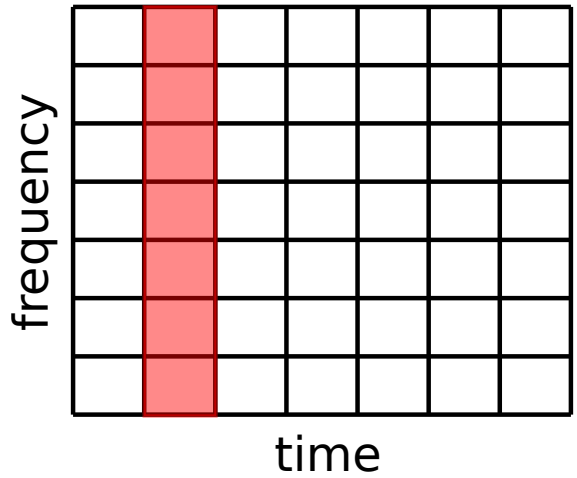
STFT Procedure

Multiply signal $x[n]$ by a sliding window $w[n]$ and take FFT.

$x[n]$



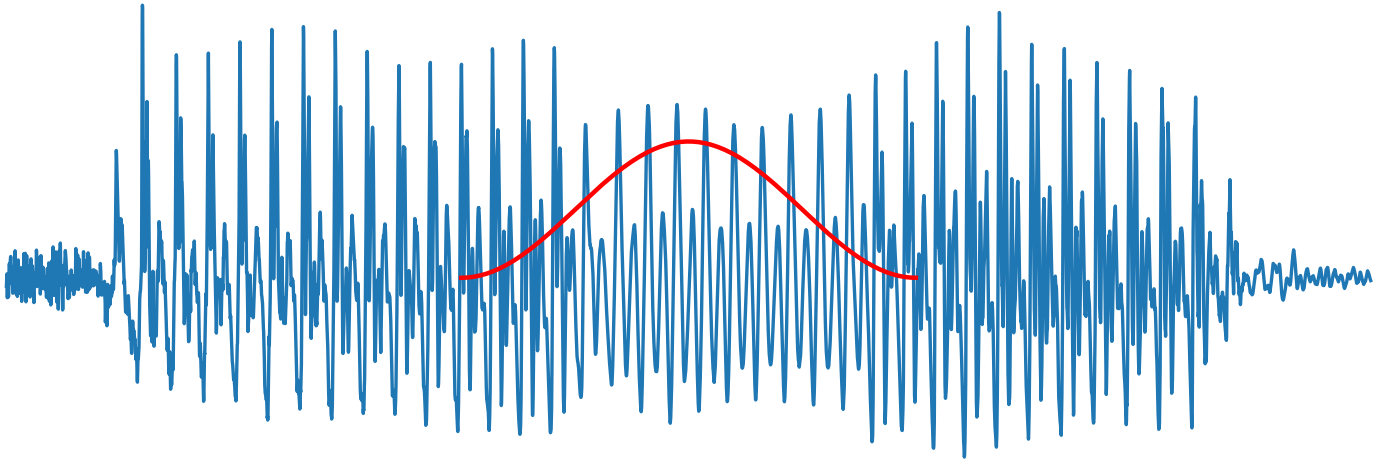
$X[k, n]$



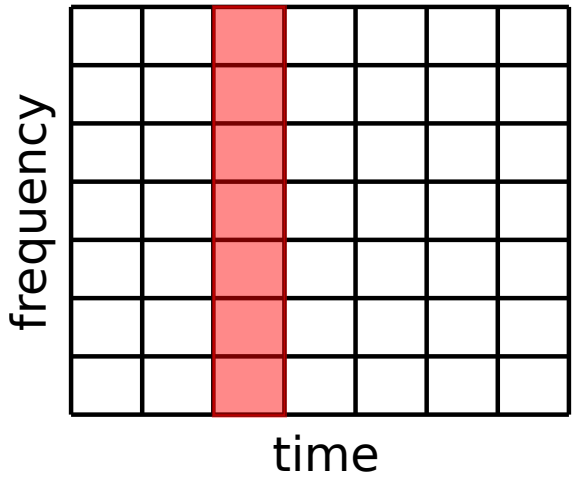
STFT Procedure

Multiply signal $x[n]$ by a sliding window $w[n]$ and take FFT.

$x[n]$



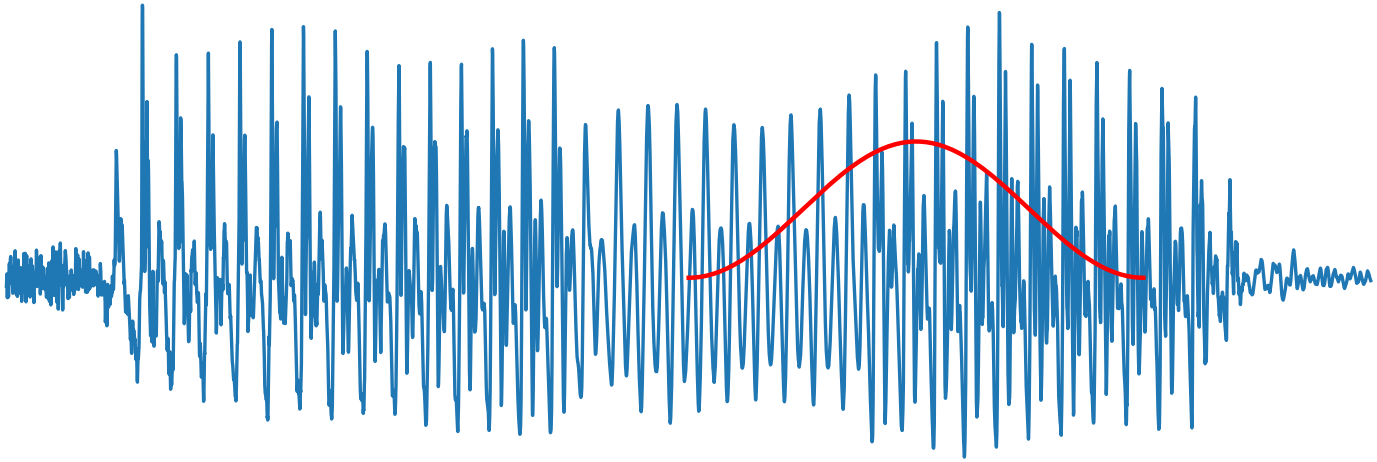
$X[k, n]$



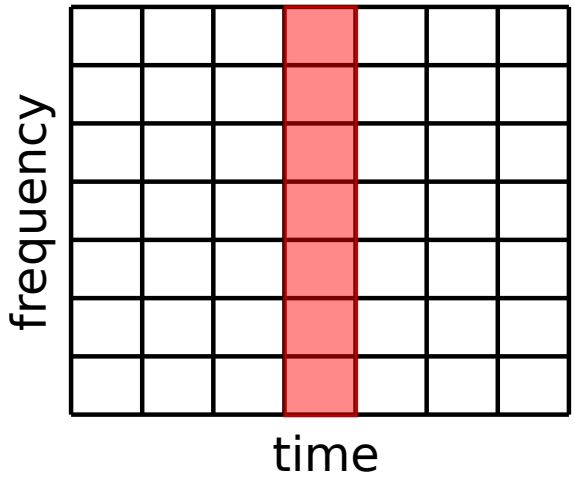
STFT Procedure

Multiply signal $x[n]$ by a sliding window $w[n]$ and take FFT.

$x[n]$



$X[k, n]$



STFT Pseudocode

```
x = input signal
w = window function
H = number of hops
X = output STFT

for n = 0 .. H
    x_clip = x[n*h : n*h + len(w)]
    X[:, n] = FFT(x_clip * w)
```