Using transform to keep some important features from the data

MATH 4280

Lecture Notes 2: Fourier and Wavelet transforms

Fourier series

- The Fourier series of a 2π-periodic real-valued function is

Think of these cos(kx) and sin(kx) are new basis,
and the amptitudes of cosine and sine function is the value of "projection"

$$f(x) = \boxed{\frac{a_0}{2}} + \sum_{k=1}^{\infty} (a_k \cos(kx) + b_k \sin(kx))$$

where

derived by using orthogonality

$$a_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(kx) dx$$

a_k, b_k are the coordinate of the original function

$$b_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(kx) dx$$

Projection

- It represents the projection of the function onto the orthogonal basis

$$\{\cos(kx), \sin(kx)\}_{k=0}^{\infty}$$

- The coefficients a_k and b_k are the coordinates in the new basis

Complex fourier series

- The Fourier series of a 2π-periodic complex-valued function is

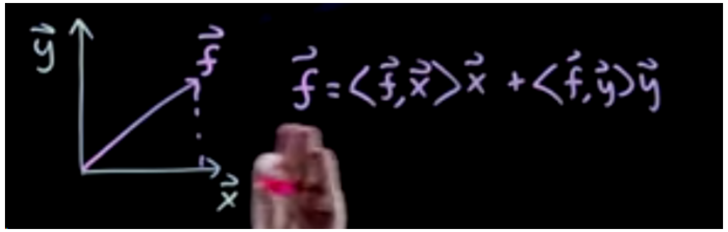
$$f(x) = \sum_{k=-\infty}^{\infty} c_k \boxed{e^{ikx}}$$

= cos(kx) + i sin(kx)

- The functions $\boxed{\psi_k = e^{ikx}}$ provide a basis for this class of functions
- These functions are orthogonal since

$$\langle \psi_j, \psi_k \rangle = \int_{-\pi}^{\pi} e^{ijx} e^{-ikx} dx = \int_{-\pi}^{\pi} e^{i(j-k)x} dx = \left[\frac{e^{i(j-k)x}}{i(j-k)} \right]_{-\pi}^{\pi} = \begin{cases} 0 & \text{if } j \neq k \\ 2\pi & \text{if } j = k \end{cases}$$

- The Fourier series is a change of coordinates



$$f(x) = \sum_{k=-\infty}^{\infty} c_k \psi_k(x) = \frac{1}{2\pi} \sum_{k=-\infty}^{\infty} \langle f(x), \psi_k(x) \rangle \psi_k(x).$$

where $\|\psi_k\|^2 = 2\pi$ and $\langle f(x), g(x) \rangle = \int_a^b f(x) \bar{g}(x)\, dx$

Inner product in Hilbert space

Fourier transform

This expression could be referred to 2.11 eqn of the book

- The Fourier transform pair is given by

original coordinate new basis

$$f(x) = \mathcal{F}^{-1}\left(\hat{f}(\omega)\right) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \boxed{\hat{f}(\omega)}\boxed{e^{i\omega x}}\, d\omega$$

Inverse Fourier transform

$$\hat{f}(\omega) = \mathcal{F}\left(f(x)\right) = \int_{-\infty}^{\infty} f(x)e^{-i\omega x}\, dx.$$

Fourier transform

where the integrals exist when $f, \hat{f} \in L^1(-\infty, \infty)$

- Derivatives of functions:

Integrating by parts

$$\begin{aligned}
\mathcal{F}\left(\frac{d}{dx}f(x)\right) &= \int_{-\infty}^{\infty} \overbrace{f'(x)}^{dv}\,\overbrace{e^{-i\omega x}}^{u}\, dx \\
&= \left[\underbrace{f(x)e^{-i\omega x}}_{uv}\right]_{-\infty}^{\infty} - \int_{-\infty}^{\infty} \underbrace{f(x)}_{v}\left[\underbrace{-i\omega e^{-i\omega x}}_{du}\right] dx \\
&= i\omega \int_{-\infty}^{\infty} f(x)e^{-i\omega x}\, dx \\
&= i\omega\mathcal{F}(f(x)).
\end{aligned}$$

- Linearity of Fourier transform

$$\mathcal{F}(\alpha f(x) + \beta g(x)) = \alpha \mathcal{F}(f) + \beta \mathcal{F}(g)$$

$$\mathcal{F}^{-1}(\alpha \hat{f}(\omega) + \beta \hat{g}(\omega)) = \alpha \mathcal{F}^{-1}(\hat{f}) + \beta \mathcal{F}^{-1}(\hat{g})$$

- Parseval's Theorem

$$\int_{-\infty}^{\infty} |\hat{f}(\omega)|^2 \, d\omega = 2\pi \int_{-\infty}^{\infty} |f(x)|^2 \, dx$$

- Convolution

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x - \xi) g(\xi) \, d\xi$$

Definition of convolution

$$\mathcal{F}^{-1}\left(\hat{f}\hat{g}\right)(x) = f * g$$

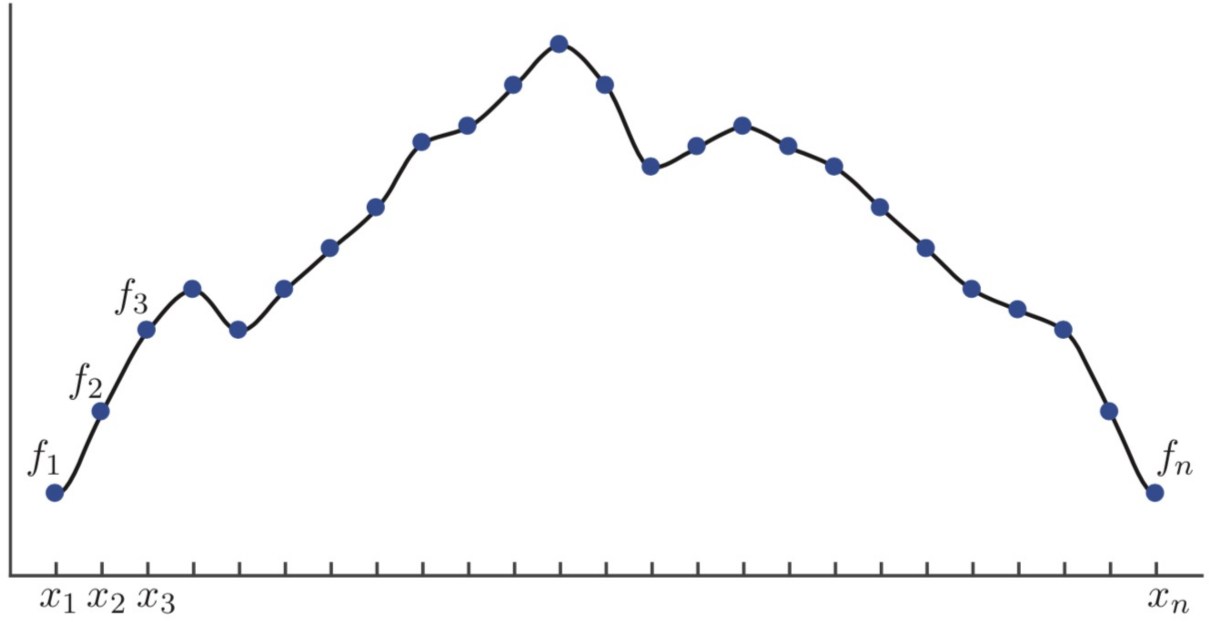
Discrete Fourier transform (DFT)

- In most applications, we work with discrete data
- The DFT is a discrete version of Fourier transform for vectors of data

$$\boxed{\mathbf{f} = \begin{bmatrix} f_1 & f_2 & f_3 & \cdots & f_n \end{bmatrix}^T}$$

The set of measurements

obtained by discretizing the function f(x) at a regular spacing



- The DFT and the inverse DFT (iDFT) are given respectively by

DFT $\boxed{\hat{f}_k = \sum_{j=0}^{n-1} f_j e^{-i2\pi jk/n}}$ iDFT $f_k = \frac{1}{n} \sum_{j=0}^{n-1} \hat{f}_j e^{i2\pi jk/n}$

- The DFT can be represented by a matrix multiplication

$$\begin{bmatrix} \hat{f}_1 \\ \hat{f}_2 \\ \hat{f}_3 \\ \vdots \\ \hat{f}_n \end{bmatrix}\begin{array}{l} 0 \\ \\ \\ \\ n\text{-}1 \end{array} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \omega_n^2 & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \cdots & \omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \cdots & \omega_n^{(n-1)^2} \end{bmatrix} \overset{f}{\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_n \end{bmatrix}}\begin{array}{l} 0 \\ \\ \\ \\ n\text{-}1 \end{array}$$

where $\omega_n = e^{-2\pi i/n}$

determine f_i cost n operations, complete all f_i n times, thus, it is O(n^2)

Fast Fourier transform (FFT)

- The DFT requires $O(n^2)$ operations
- The FFT requires $O(n \log n)$ operations, a significant improvement
- The basic idea of FFT is the fact that the DFT can be implemented efficiently when n is a power of 2

- Example: when $n = 1024 = 2^{10}$, the DFT matrix F_{1024} is

$$\hat{\mathbf{f}} = \mathbf{F}_{1024}\mathbf{f} = \begin{bmatrix} \mathbf{I}_{512} & \mathbf{D}_{512} \\ \mathbf{I}_{512} & -\mathbf{D}_{512} \end{bmatrix} \begin{bmatrix} \mathbf{F}_{512} & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_{512} \end{bmatrix} \begin{bmatrix} \mathbf{f}_{\text{even}} \\ \mathbf{f}_{\text{odd}} \end{bmatrix}$$

where

$$\mathbf{D}_{512} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & \omega & 0 & \cdots & 0 \\ 0 & 0 & \omega^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \omega^{511} \end{bmatrix}$$

and $\omega = e^{-2\pi i/n}$

Example: denoising

- Consider a signal with two dominant frequencies $f_1 = 50$ and $f_2 = 120$

$$f(t) = \sin(2\pi f_1 t) + \sin(2\pi f_2 t)$$

- Add some Gaussian noise to the signal
- Use FFT and look at the signal in the frequency domain
- Truncate those frequencies with small amplitude
- Apply iFFT to obtain a denoised signal

MATLAB implementation

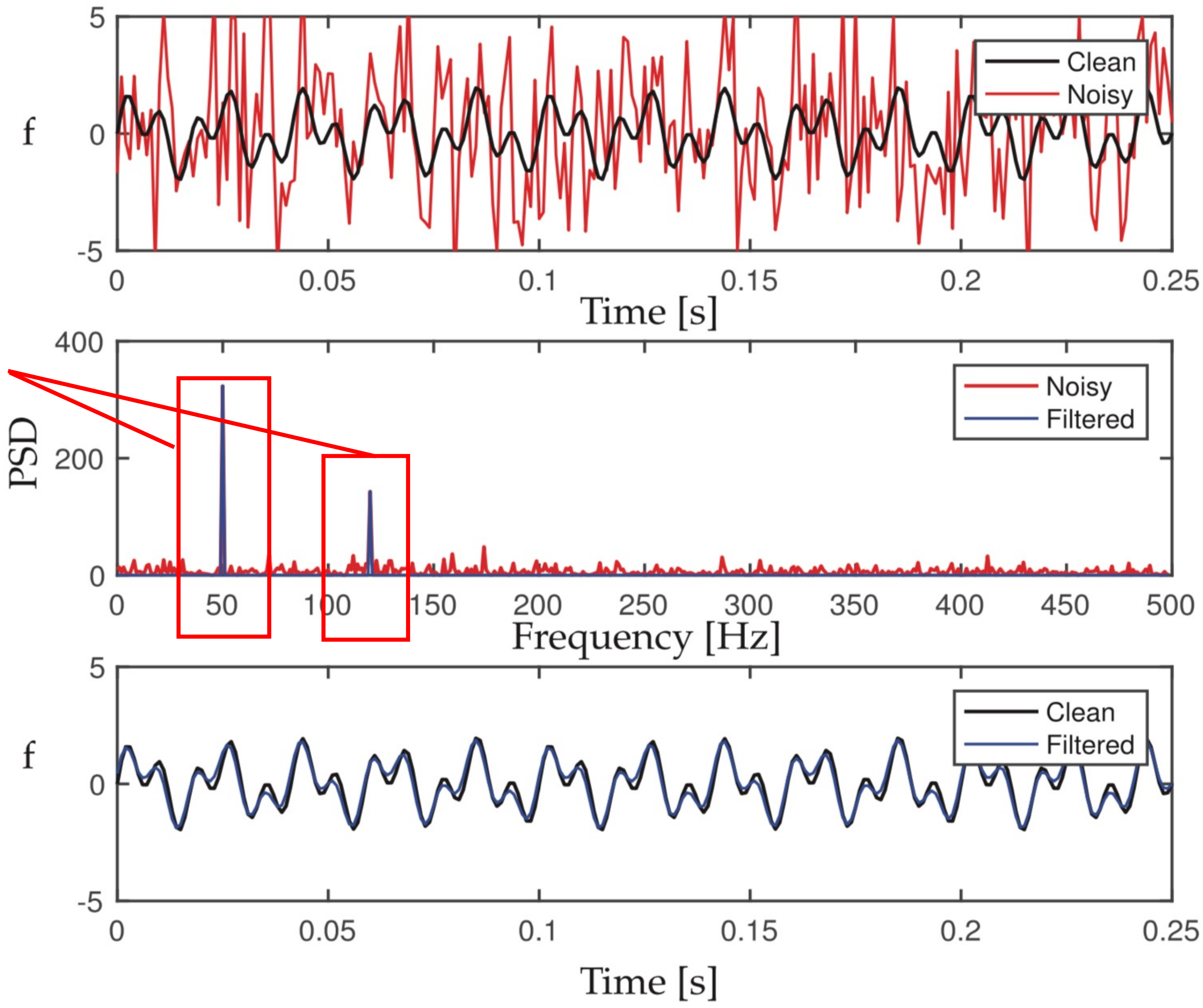
modulus

```
dt = .001;
t = 0:dt:1;
f = sin(2*pi*50*t) + sin(2*pi*120*t); % Sum of 2 frequencies
f = f + 2.5*randn(size(t));   %  Add some noise

%% Compute the Fast Fourier Transform FFT
n = length(t);
fhat = fft(f,n);          % Compute the fast Fourier transform
PSD = fhat.*conj(fhat)/n; % Power spectrum (power per freq)
freq = 1/(dt*n)*(0:n); % Create x-axis of frequencies in Hz
L = 1:floor(n/2);     % Only plot the first half of freqs

%% Use the PSD to filter out noise
indices = PSD>100;   % Find all freqs with large power
PSDclean = PSD.*indices;   % Zero out all others
fhat = indices.*fhat;   % Zero out small Fourier coeffs. in Y
ffilt = ifft(fhat); % Inverse FFT for filtered time signal
```

Keep large enough numbers

Keeping large values

Results



Original and noisy signals

Signals in frequency domain

Denoised signal

Fast computation of derivative of function

- Recall the property $\mathcal{F}(df/dx) = i\omega\mathcal{F}(f)$
- Steps of computing derivatives:

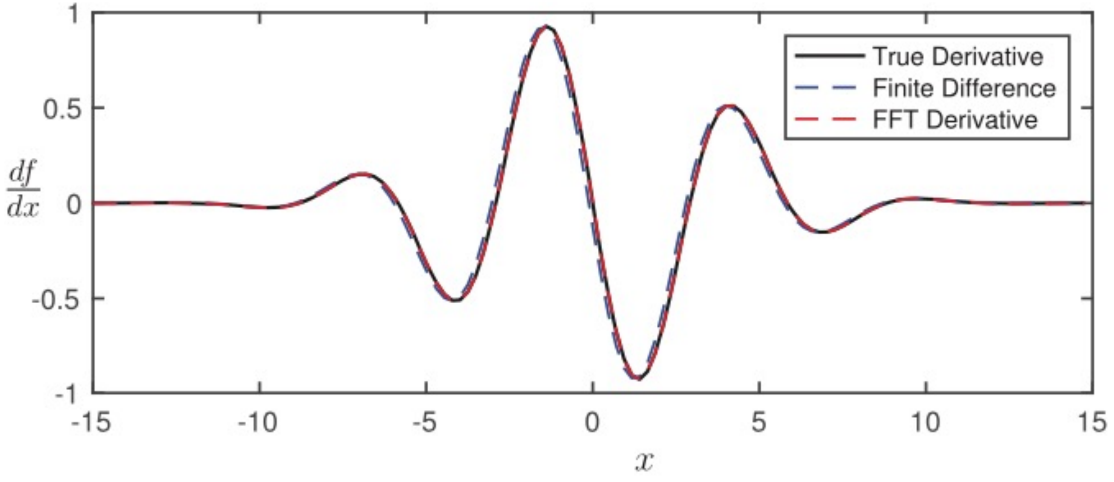
1. Compute the Fourier transform $\hat{f}$ $$\hat{f}_k = \sum_{j=0}^{n-1} f_j e^{-i2\pi jk/n}$$

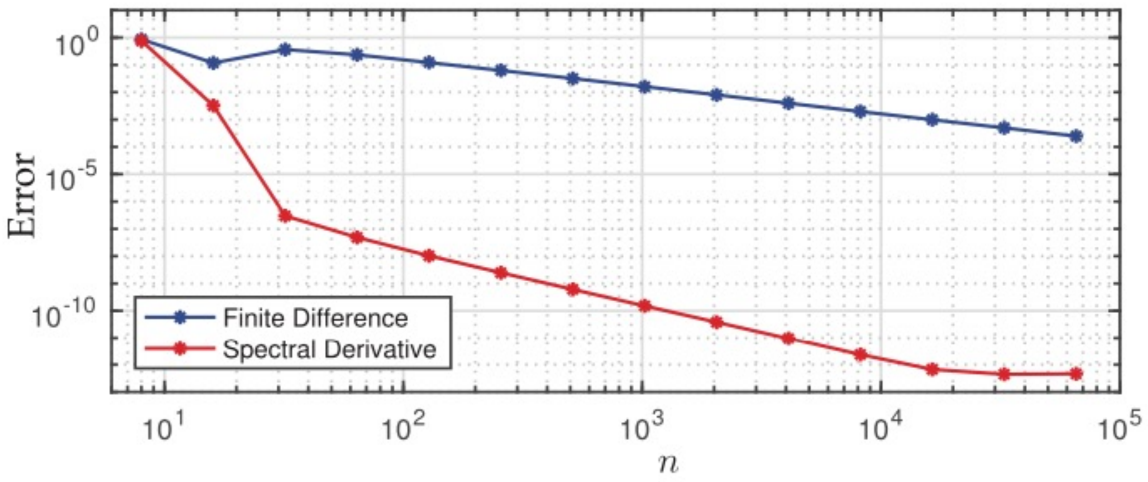
mimic the derivative of the functio

2. Multiply the k-th component of $\boxed{\hat{f} \text{ by i } 2\pi k/n}$

3. Compute the inverse Fourier transform $$f_k = \frac{1}{n}\sum_{j=0}^{n-1} \hat{f}_j e^{i2\pi jk/n}$$

- Let $f(x) = \cos(x)e^{-x^2/25}$
- Compare with finite difference approximation $\frac{df}{dx}(x_k) \approx \frac{f(x_{k+1}) - f(x_k)}{\Delta x}$





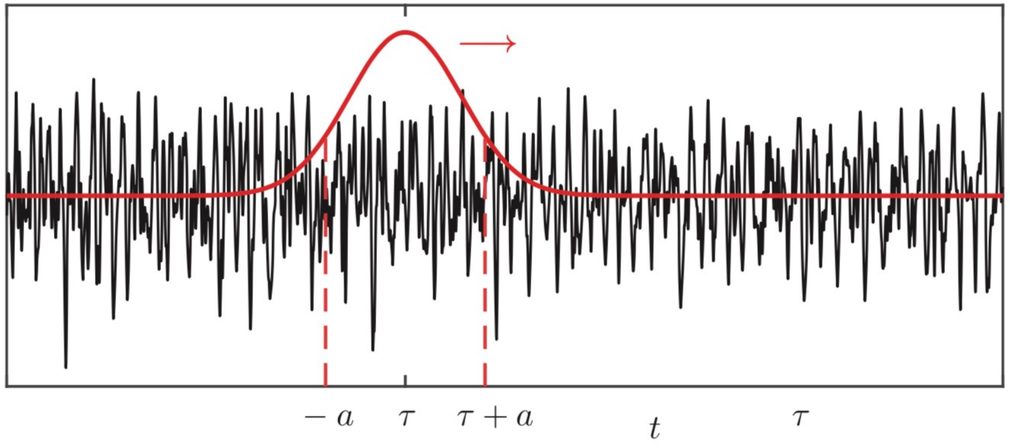
n=128

Comparison for various n

Gabor transform

Localize the fourier transform

- Fourier transform works for stationary signals (frequency does not change with time)
- For nonstationary signals, it is important to characterize the frequency and its evolution in time
- The Gabor transform is a windowed Fourier transform in a moving window (also called short-time Fourier transform)
- It allows localization of frequency content



The Gabor transform is defined as

$$\mathcal{G}(f)(t,\omega) = \hat{f}_g(t,\omega) = \int_{-\infty}^{\infty} f(\tau)e^{-i\omega\tau}\bar{g}(\tau - t)\,d\tau = \langle f, \boxed{g_{t,\omega}\rangle}$$

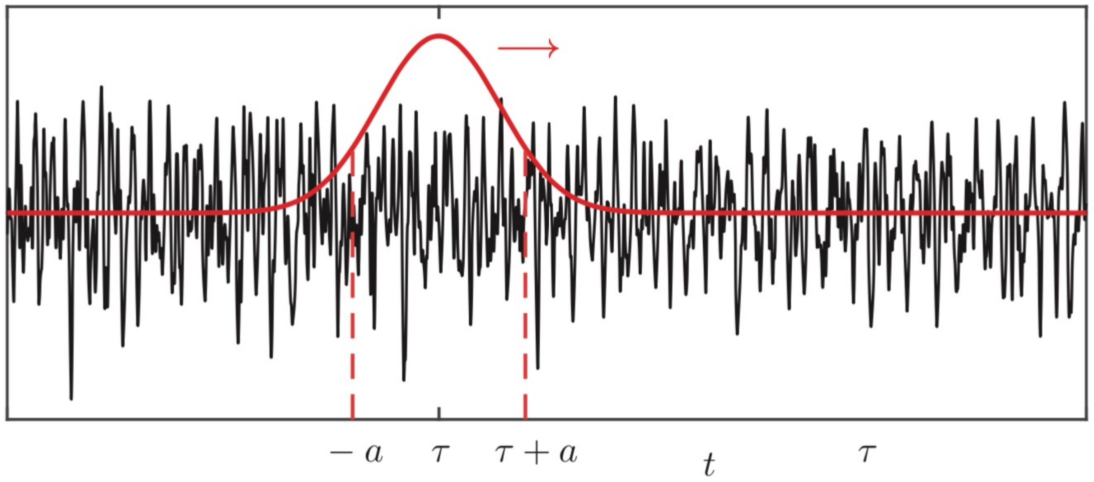
where

Key: Know about how the transform is localized

The detail of the transformation is not important

$$g_{t,\omega}(\tau) = e^{i\omega\tau}\boxed{g}(\tau - t)$$

$$g(t) = e^{-(t-\tau)^2/a^2}$$



The constant a determines the spread of the window, and τ determines the center of the window

The Inverse Gabor transform is given by

$$f(t) = \mathcal{G}^{-1}\left(\hat{f}_g(t,\omega)\right) = \frac{1}{2\pi\|g\|^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \hat{f}_g(\tau,\omega)g(t-\tau)e^{i\omega t}\,d\omega\,dt.$$

Discrete Gabor transform

- We discretize both in time and frequency

$$\nu = j\Delta\omega$$
$$\tau = k\Delta t.$$

- The discretized kernel function is

$$g_{j,k} = e^{i2\pi j\Delta\omega t} g(t - k\Delta t)$$

- The discrete Gabor transform is

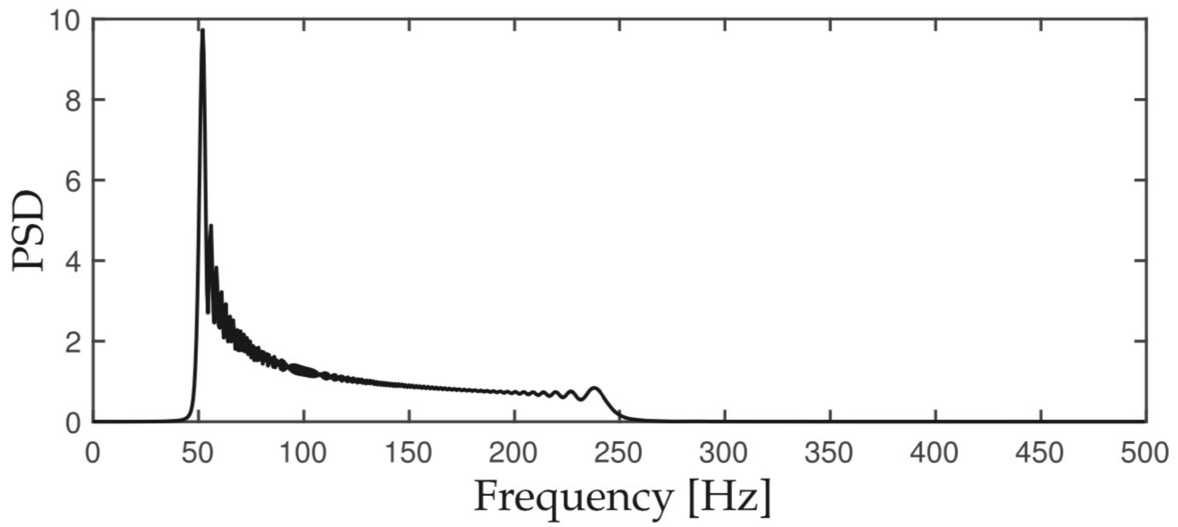
$$\hat{f}_{j,k} = \langle f, g_{j,k}\rangle = \int_{-\infty}^{\infty} f(\tau)\bar{g}_{j,k}(\tau)\, d\tau$$

Example: a quadratic chirp

- We consider the signal

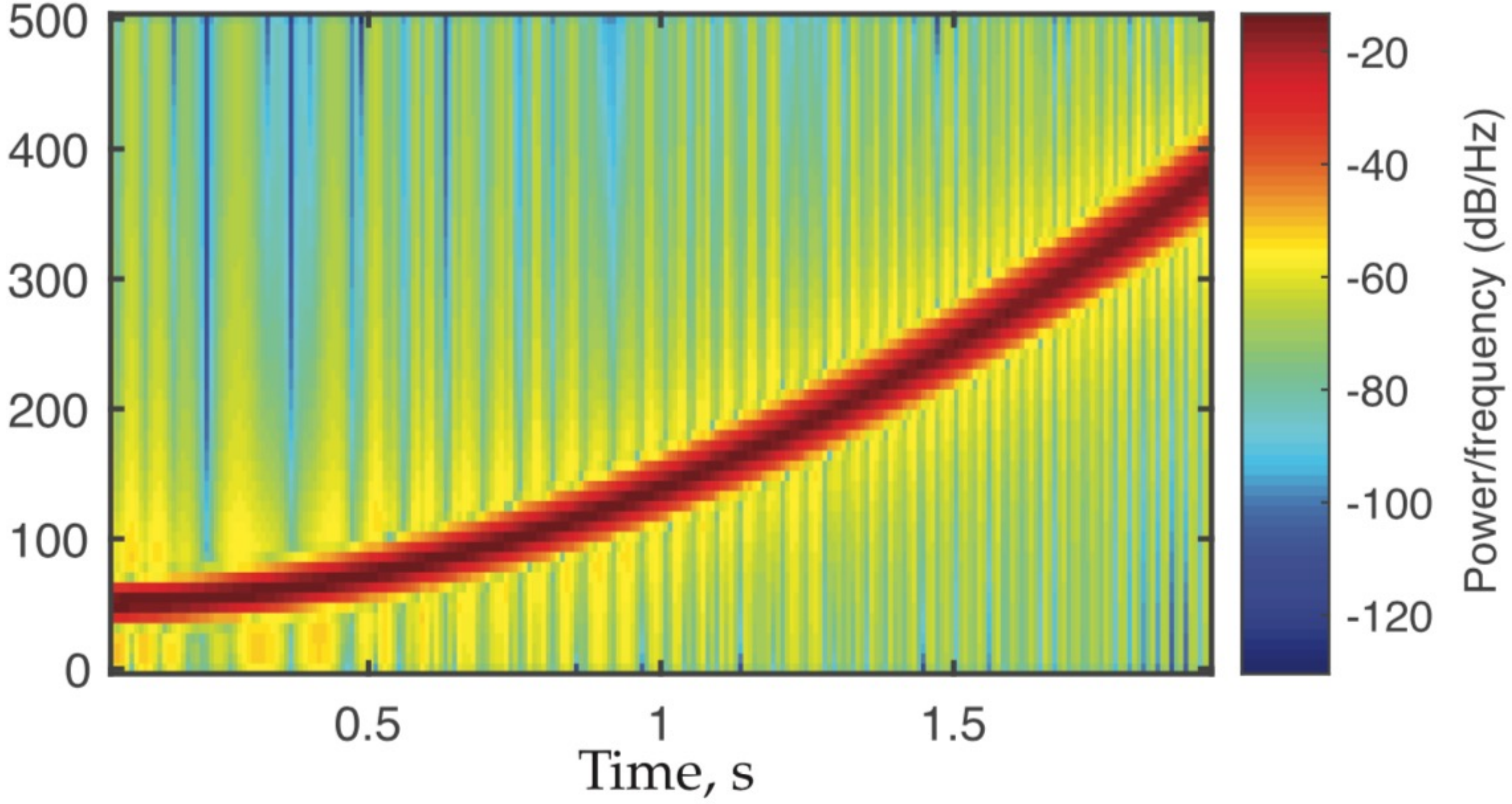
$$\boxed{f(t) = \cos(2\pi t\omega(t)) \quad \text{where} \quad \omega(t) = \omega_0 + (\omega_1 - \omega_0)t^2/3t_1^2}$$

- The frequency changes from ω_0 to ω_1 from $t = 0$ to $t = t_1$
- Take $\omega_0 = 50$ and $\omega_1 = 250$ and $t_1 = 2$



Fourier transform does not see the change in the frequency

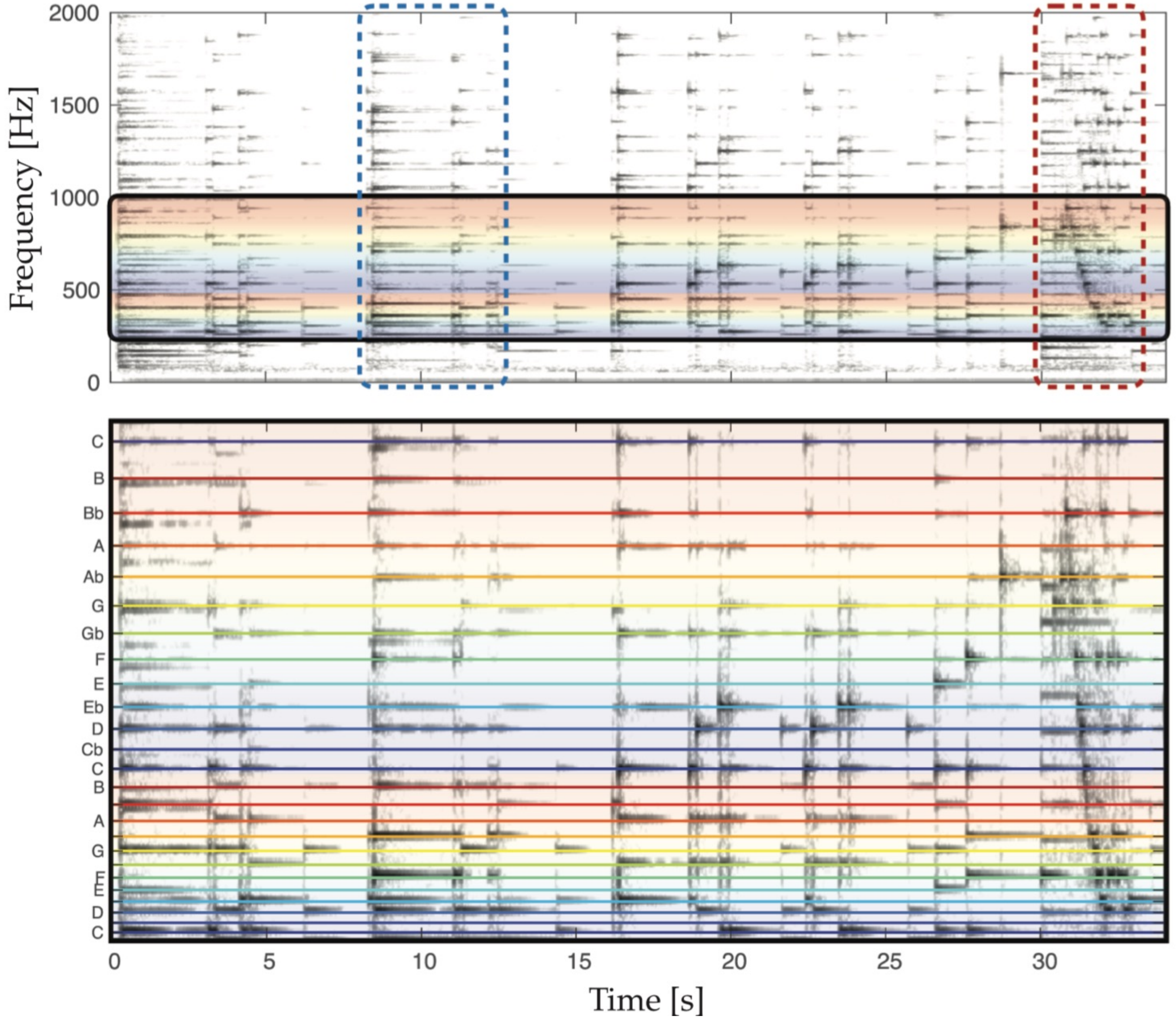
- Using the Gabor transform, we can plot the following spectrogram



- We see clearly frequency shifts in time

Example: music

- The spectrogram can be used to analyze music
- Identify key markers for classification



Uncertainty principles

- It limits the ability to simultaneously attain high resolution in both the time and frequency domains
- Mathematically,

variane of f

variance of f head

one of these number must be large

$$\left(\int_{-\infty}^{\infty} x^2 |f(x)|^2 \, dx\right) \left(\int_{-\infty}^{\infty} \omega^2 |\hat{f}(\omega)|^2 \, d\omega\right) \geq \frac{1}{16\pi^2}$$

- For real-valued functions, this is the second moment, which measures the variance of Gaussian functions
- Thus, a function and its Fourier transform cannot both be arbitrarily localized

Wavelet transform

- Wavelets extend the concept of Fourier analysis to more general orthogonal bases
- They can partially overcome the limitation resulting from the uncertainty principle by using a multi-resolution decomposition
- The idea starts with a function $\psi(t)$, called the mother wavelet
- A family of scaled and translated functions can be generated

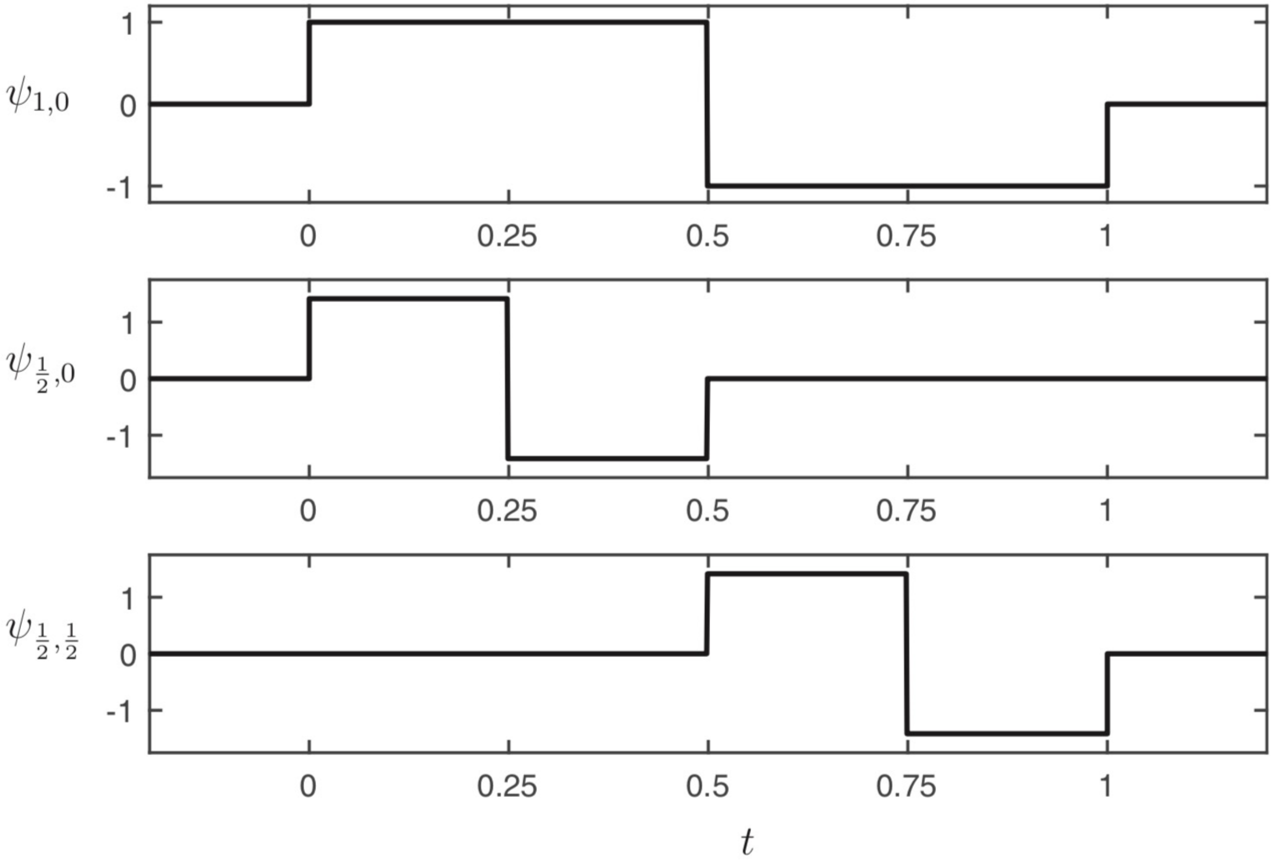
$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right)$$

Haar wavelet

- The simplest and the earliest example is the Haar wavelet

$$\psi(t) = \begin{cases} 1 & 0 \leq t < 1/2 \\ -1 & 1/2 \leq t < 1 \\ 0 & \text{otherwise.} \end{cases}$$

- Three Haar wavelets are shown
- Choosing the next higher frequency layer using a bisection, the resulting Haar wavelets are orthogonal



Continuous wavelet transform (CWT)

- The continuous wavelet transform (CWT) is given by

$$\mathcal{W}_\psi(f)(a,b) = \langle f, \psi_{a,b} \rangle = \int_{-\infty}^{\infty} f(t)\bar{\psi}_{a,b}(t)\,dt$$

- The inverse continuous wavelet transform (iCWT) is given by

$$f(t) = \frac{1}{C_\psi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathcal{W}_\psi(f)(a,b)\,\psi_{a,b}(t)\,\frac{1}{a^2}\,da\,db$$

where $C_\psi = \int_{-\infty}^{\infty} \frac{|\hat{\psi}(\omega)|^2}{|\omega|}\,d\omega$

Discrete wavelet transform (DWT)

- The discrete wavelet transform (DWT) is given by

$$\mathcal{W}_\psi(f)(j,k) = \langle f, \psi_{j,k} \rangle = \int_{-\infty}^{\infty} f(t)\bar{\psi}_{j,k}(t)\,dt$$

- Here $\psi_{j,k}(t)$ is a discrete family of wavelets

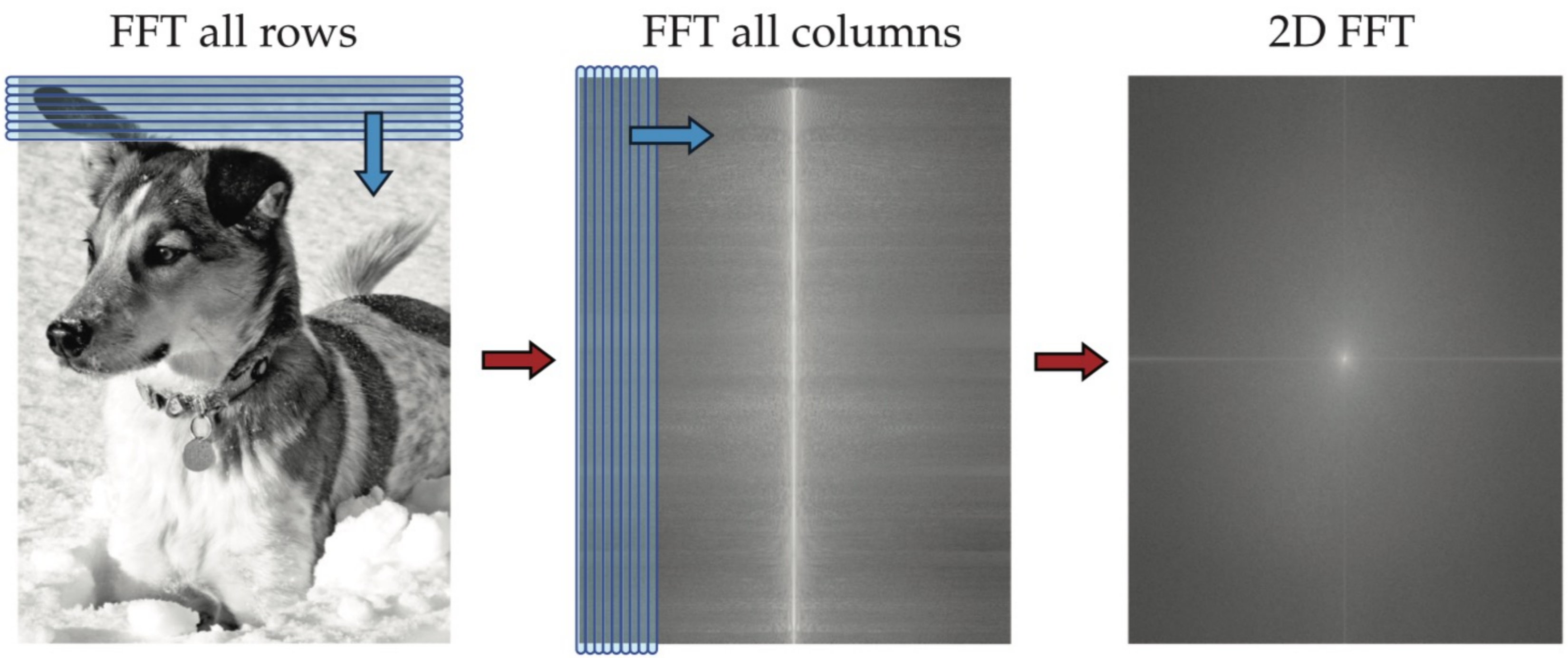
$$\psi_{j,k}(t) = \frac{1}{a^j}\psi\left(\frac{t-kb}{a^j}\right)$$

- If the family is orthonormal, then we can expand a function as follows

$$f(t) = \sum_{j,k=-\infty}^{\infty} \langle f(t), \psi_{j,k}(t) \rangle \psi_{j,k}(t)$$

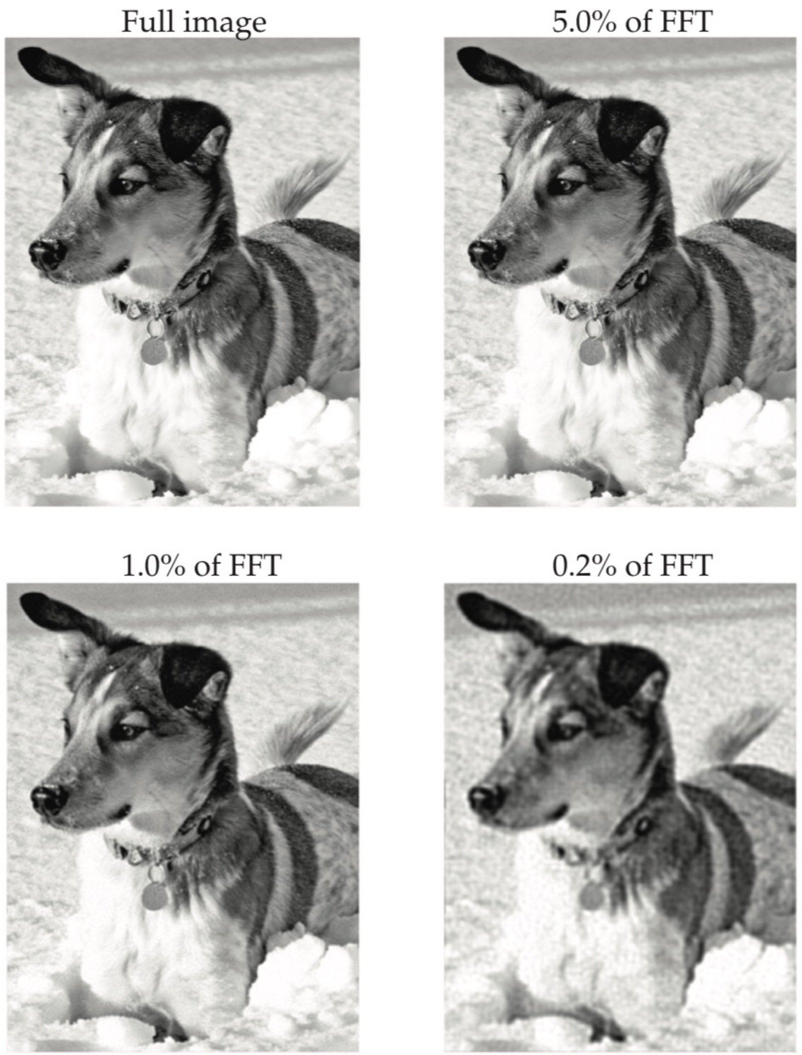
2D Fourier transform

- Given a matrix X, its Fourier transform is obtained by applying the 1D Fourier transform to each row, then applying to each column
- The order is irrelevant



Example: image compression by FFT

Recall:
FFT decompose an image into a lot of terms. (Descending)



Keeping 5% of the largest Fourier coefficients

Keeping 1% of the largest Fourier coefficients

Of course, you need to perform inverse transform to get back the original images

Keeping 0.2% of the largest Fourier coefficients