Signal Processing
from Fourier to Machine Learning
Part 2 : Digital Signal Processing

R. Flamary

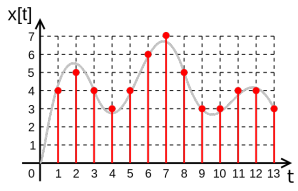
November 21, 2024

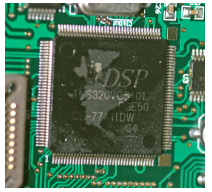
Full course overview

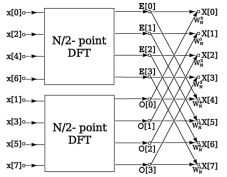
1. **Fourier analysis and analog filtering**
   - 1.1 Fourier Transform
   - 1.2 Convolution and filtering
   - 1.3 Applications of analog signal processing
2. **Digital signal processing**
   - 2.1 Sampling and properties of discrete signals
   - 2.2 z Transform and transfer function
   - 2.3 Fast Fourier Transform
3. **Random signals**
   - 3.1 Random signals, stochastic processes
   - 3.2 Correlation and spectral representation
   - 3.3 Filtering and linear prediction of stationary random signals
4. **Signal representation and dictionary learning**
   - 4.1 Non stationary signals and short time FT
   - 4.2 Common signal representations (Fourier, wavelets)
   - 4.3 Source separation and dictionary learning
   - 4.4 Signal processing with machine learning

Course overview

Digital Signal Processing



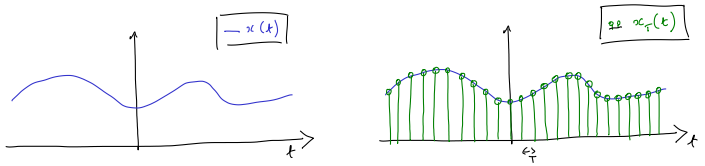




Digital Signal Processing

- Microprocessors widely available and cheap since the 70s.
- Analog-to-digital converter (ADC) and digital-to-analog converter (DAC).
- Standard signal processing : ADC→DSP→DAC.
- DSP more robust/stationary.
- Analog SP is faster but sensitive to physics (temperature).
- Digital Signal Processing can be done on dedicated hardware or processors.

Sampling



Principle

- Sampling is the reduction of a continuous-time signal to a discrete-time signal.
- A discrete signal sampled for period T can be expressed as

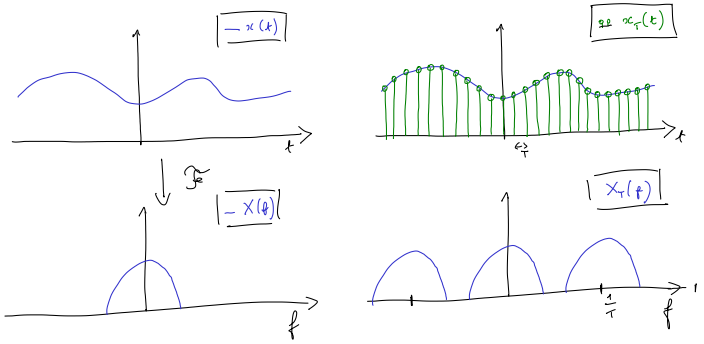
$$x_T(t) = \sum_{n=-\infty}^{\infty} x(nT)\delta(t - nT) \tag{1}$$

- T is the sampling period (or interval), $f_s = \frac{1}{T}$ is the sampling frequency.
- Due to the properties of the dirac δ the sampled signal is equal to

$$x_T(t) = x(t) \sum_{n=-\infty}^{\infty} \delta(t - nT) = x(t)\text{Ш}_T(t) \tag{2}$$

where $\text{Ш}_T(t)$ is the dirac comb of period T.

Sampling in the Fourier domain



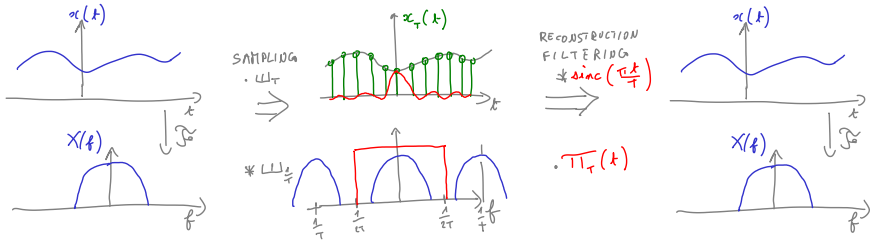
Fourier transform of sampled signal

- If $x(nT)$ is bounded for $n \in \mathbb{Z}$, $x_T(t)$ is a tempered distribution.
- The FT of $x_T(t)$ can be expressed as a function of $X(f) = \mathcal{F}[x(t)]$:

$$\mathcal{F}[x_T(t)] = \mathcal{F}[x(t)\text{Ш}_T(t)] = X(f) \star \frac{1}{T}\text{Ш}_{\frac{1}{T}}(f) = \frac{1}{T}\sum_{n=-\infty}^{\infty} X\left(f - \frac{n}{T}\right) \tag{3}$$

- The regular sampling leads to a periodization in the Fourier domain.

Nyquist/Shannon sampling Theorem



Theorem [Shannon, 1949][Nyquist, 1928]

Let $x(t)$ be a signal of Fourier transform $X(f)$ that has a support in $[-\frac{1}{2T}, \frac{1}{2T}]$. Then the signal $x(t)$ can be reconstructed from its sampling with

$$x(t) = \sum_{n=-\infty}^{\infty} h_T(t - nT)x(nT) = x_T(t) \star h_T(t) \tag{4}$$

where

$$h_T(t) = \text{sinc}\left(\frac{\pi t}{T}\right) = \frac{\sin(\frac{\pi t}{T})}{\frac{\pi t}{T}} \tag{5}$$

For a signal of frequency support $[-B, B]$, B is often called the Nyquist frequency (half the sampling rate necessary for reconstruction).

Nyquist/Shannon sampling Theorem

Proof

Let $x_T(t) = x(t)\sum_{n=-\infty}^{\infty}\delta(t - nT)$ be the sampled signal. Its Fourier Transform is

$$X_T(t) = \frac{1}{T}\sum_{n=-\infty}^{\infty} X\left(f - \frac{1}{T}\right)$$

Since we know that X is of support $[-\frac{1}{2T}, \frac{1}{2T}]$ it means that $\forall f \in [-\frac{1}{2T}, \frac{1}{2T}]$ we have $X_T(f) = \frac{1}{T}X(f)$.
Now if we want to reconstruct the signal we can multiply in the Fourier domain by the ideal filter:

$$H(f) = \begin{cases} T & \text{if } |f| < \frac{1}{2T} \\ 0 & \text{else} \end{cases}$$

Using the bounded support of X we have now $\forall f$

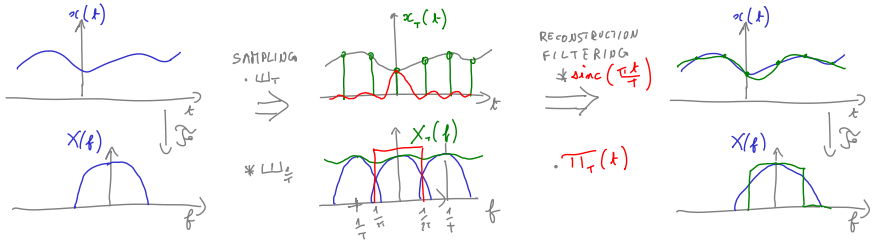
$$X_T(f)H(f) = X(f)$$

Which in the temporal domain means

$$x(t) = x_T(t) \star h(t) = h(t) \star \sum_{n=-\infty}^{\infty} x(nT)\delta(t - nT) = \sum_{n=-\infty}^{\infty} x(nT)h(t - nT)$$

where $h(t) = \text{sinc}\left(\frac{\pi t}{T}\right)$ is the inverse TF of H.

Aliasing



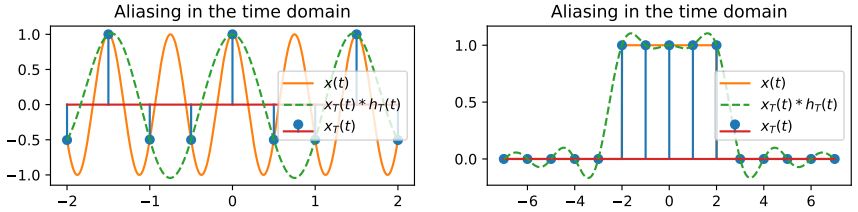
Aliasing

- The FT of $x_T(t)$ is a weighted sum of $X(f) = \mathcal{F}[x(t)]$:

$$\mathcal{F}[x_T(t)] = \mathcal{F}[x(t)\mathrm{III}_T(t)] = X(f) \star \frac{1}{T}\mathrm{III}_{\frac{1}{T}}(f) = \frac{1}{T}\sum_{n=-\infty}^{\infty} X\left(f - \frac{n}{T}\right) \quad (6)$$

- When the support of $X(f)$ is not in $[-\frac{1}{2T}, \frac{1}{2T}]$ the repeated shapes will overlap in frequency.
- In this case the signal cannot be reconstructed and some information is lost.

Example of aliasing



- Let $x(t) = \cos(2\pi f_0 t)$ be a signal that we want to sample.
- We suppose that $\frac{f_s}{2} < f_0 < f_s = f_s$.
- We have

$$X_T(f) = \frac{1}{T}\sum_k \frac{1}{2}(\delta(f - f_0 - kf_s) + \delta(f + f_0 - kf_s))$$

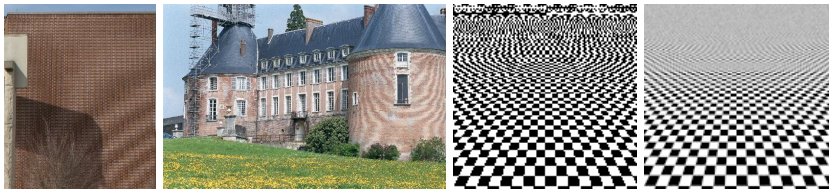
- The only components of the spectrum in $[-\frac{f_s}{2}, \frac{f_s}{2}]$ are:

$$X_T(f)H(f) = \frac{1}{2}(\delta(f - f_0 + f_s) + \delta(f + f_0 - f_s)$$

- Reconstructed signal:

$$x(t) = \cos(2\pi(f_s - f_0)t)$$

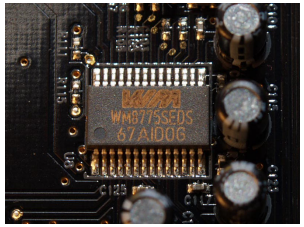
Aliasing in real life

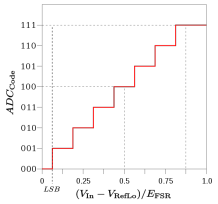


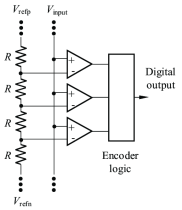
Aliasing

- When sampling high frequency real life signals.
- Always needs a low-pass filter (analog) before sampling.
- Can be solved by oversampling (followed by filtering then subsampling).
- Anti-aliasing filters in graphic cards (and digital cameras).

Analog to Digital Conversion







ADC circuits

- Sampling frequency has to be twice the maximum frequency in the signal.
- Low pass filtering before sampling (analog).
- Several sources of noise : jitter (non perfect clock), non-linearity,
- For images CCD or CMOS (smartphones) sensors count photons.

Quantization

- Computers are discrete, digital signal are discrete both in time and value.
- Quantization is the conversion from continuous value to a finite bit format.
- Number of bits has an important impact on SNR after reconstruction.

Discrete signal (1)

Notations

- $x(t)$ with $t \in \mathbf{R}$ is the analog signal.
- $x_T(t)$ with $t \in \mathbf{R}$ is the sampled signal of period (T) but still continuous time:

$$x_T(t) = \sum_{n=-\infty}^{\infty} x(nT)\delta(t - nT)$$

- $x[n]$ with $n \in \mathbb{Z}$ is the discrete signal sampled with period T such that:

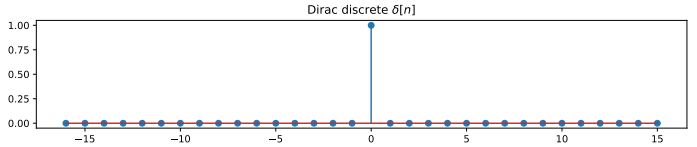
$$x[n] = x(nT)$$

- Obviously one can recover $x_T(t)$ from $x[n]$ with

$$x_T(t) = \sum_{n=-\infty}^{\infty} x[n]\delta(t - nT)$$

- In order to simplify notations we will suppose $T = 1$ in the following.
- In this course we suppose that $|x[n]|$ is bounded.

Discrete signal (2)



Discrete dirac

We note the discrete dirac $\delta[n]$ defined as

$$\delta[n] = \begin{cases} 1 & \text{for } n = 0 \\ 0 & \text{else} \end{cases} \tag{7}$$

Discrete signal

Any discrete signal $x[n]$ can be decomposed as a sum of translated discrete diracs:

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n - k] \tag{8}$$

The discrete diracs are an orthogonal basis of $L_2(\mathbb{Z})$ of scalar product and corresponding norm

$$< x[n], h[n] >= \sum_{k=-\infty}^{\infty} x[k]h^*[k], \qquad \|x[n]\|^2 =< x[n], x[n] >= \sum_{k=-\infty}^{\infty} |x[k]|^2.$$

Discrete Convolution

Convolution between discrete signals

Let $x[n]$ and $h[n]$ two discrete signals. The convolution between them is expressed as:

$$x[n] \star h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n - k] \tag{9}$$

Digital filter properties

Let the discrete system/operator/filter L described by its impulse response $h[n]$.

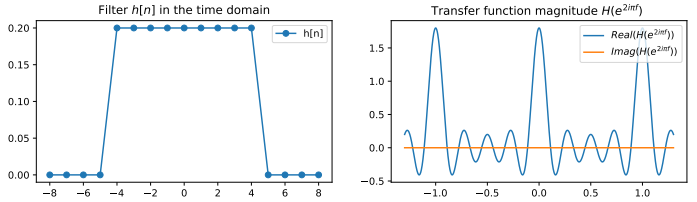
- **Causality** L is causal if $h[n] = 0,\ \forall n < 0$. L is causal if

$$h[n] = h[n]\Gamma[n], \qquad \text{where} \qquad \Gamma[n] = \begin{cases} 1 & \text{for } n \geq 0 \\ 0 & \text{else} \end{cases} \tag{10}$$

- **Stability** A system is stable if the output of a bonded input is bounded. A necessary and sufficient condition is that

$$\sum_{n=-\infty}^{\infty} |h[n]| < \infty \tag{11}$$

Transfer function



Transfer function of a discrete filter

- Let L be a digital filter of impulse response $h[n]$.
- For an input $e_f[k] = e^{i2\pi fk}$ the output of the filter is

$$Le_f[n] = \sum_{k=-\infty}^{\infty} e^{i2\pi f(n-k)}h[k] = e^{i2\pi fn} \sum_{k=-\infty}^{\infty} e^{-i2\pi fk}h[k] \tag{12}$$

- $e_f[k] = e^{i2\pi fk}$ are then the eigenvectors of the discrete convolution operator.
- The **Transfer function** of the filter is defined as the following Fourier series:

$$H(e^{i2\pi f}) = \sum_{k=-\infty}^{\infty} e^{-i2\pi fk}h[k] \tag{13}$$

This actually corresponds to the Fourier transform of the signal.

Discrete Time Fourier Transform (DTFT)

Fourier transform

The Discrete Time Fourier Transform of the discrete signal $x[n]$ is defined as

$$X(e^{i2\pi f}) = \sum_{k=-\infty}^{\infty} e^{-i2\pi fk} x[k] \tag{14}$$

- It is periodic and equivalent to the Fourier transform of $x_T(t)$.
- For a tempered distribution ($x[n]$ bounded) all the FT properties are preserved.

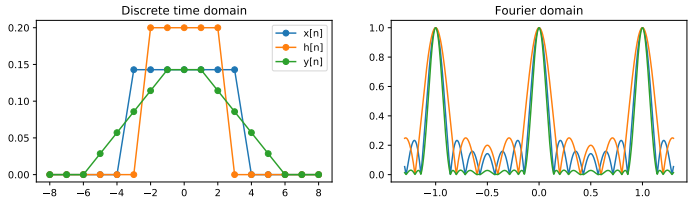
Orthonormal basis of $L_2([0,1]))$

- The FT of a discrete signal is periodic (of period $T = 1$) and can be expressed as a Fourier series.
- $e^{i2\pi fn}$ defines an orthogonal basis of $L_2([0,1]))$ with scalar product $< a(f), b(f) >= \int_0^1 a(f) b*(f) df$.
- The coefficients can be recovered using the scalar product:

$$x[n] =< X(e^{i2\pi f}), e^{-i2\pi fn} >= \int_0^1 X(e^{i2\pi f}) e^{i2\pi fn} df \tag{15}$$

- Conservation of energy implies that $\sum_{k=-\infty}^{\infty} |x[n]|^2 = \int_0^1 |X(e^{i2\pi f})|^2 df$

Fourier Transform and discrete convolution



Theorem

Let $x[n] \in L_2(\mathbb{Z})$ and $h[n] \in L_2(\mathbb{Z})$ two discrete signals. The Fourier Transform of $y[n] = x[n] \star h[n]$ is

$$Y(e^{i2\pi f}) = X(e^{i2\pi f}) H(e^{i2\pi f}) \tag{16}$$

- Similarly to continuous signal, the FT of the convolution is a pointwise multiplication.
- This shows similarly that the filter will have an effect (amplification and attenuation) on the individul frequency components.
- The FT of a temporal multiplication $y[n] = x[n]h[n]$ can also be expressed as

$$Y(e^{i2\pi f}) = \int_0^1 X(e^{i2\pi u}) H(e^{i2\pi (f-u)}) du = X(e^{i2\pi f}) * H(e^{i2\pi f})$$

Ideal filter

Ideal low pass filter

- The Fourier transform of the ideal low pass filter with $f_c < \frac{1}{2}$ is

$$H_0(e^{i2\pi f}) = \begin{cases} 1 & \text{for } |f| < f_c \\ 0 & \text{else} \end{cases} \tag{17}$$

- Using Eq. 15 one can recover the impulse response

$$h[n] = \int_{-\frac{1}{2}}^{\frac{1}{2}} H_0(e^{i2\pi f}) e^{i2\pi fn} df = \frac{\sin(2\pi f_c n)}{\pi n} = 2 f_c \text{sinc}(2\pi f_c n) \tag{18}$$

  That is a regular sampling of the continuous impulse response.
- This filter is non causal and cannot be implemented in practice (infinite sum).
- In practice one has to approximate this filter using Finite impulse response or Infinite impulse response filters.

Digital filter

Recurent filter formulation

- We want to design an implementable filter (finite number fo computations).
- We define the relation between the input $x[n]$ and the output $y[n]$ as a difference equation :

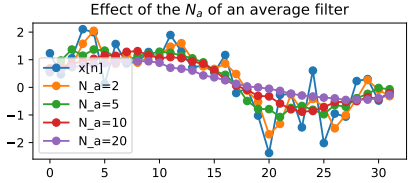
$$\sum_{k=0}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k] \tag{19}$$

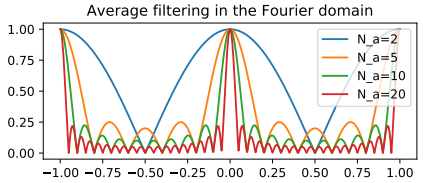
  where a_k and b_k are reals and $a_0 \neq 0$.
- The sample $y[n]$ can be expressed as

$$y[n] = \frac{1}{a_0} \left(\sum_{k=0}^{M} b_k x[n-k] - \sum_{k=1}^{N} a_k y[n-k] \right) \tag{20}$$

- Can be computed only from the past (causal) and with $M + N$ multiply/accumulate.
- Called Infinite Impulse response (IIR) filter when $N > 1$ because the recurrence imply that $y[n]$ depends on all the values of $x[k]$ for $k \leq n$.

Special cases





Finite Impulse Filter (FIR) when $N = 0$

$$y[n] = \sum_{k=0}^{M} \frac{b_k}{a_0} x[n-k] = x[n] \star h[n] \quad (21)$$

The impulse response is

$$h[n] = \begin{cases} \frac{b_n}{a_0} & \text{for } 0 \leq n \leq M \\ 0 & \text{else} \end{cases} \quad (22)$$

Example: Average filter

$$h[n] = \begin{cases} \frac{1}{N_a} & \text{for } 0 \leq n \leq N_a \\ 0 & \text{else} \end{cases} \quad (23)$$

Autoregressive model (AR) when $M = 1$

$$y[n] = \sum_{k=1}^{N} \frac{b_k}{a_0} y[n-k] \quad (24)$$

The output depends only on initial (in time) condition of the output. It is not a filter.

Transfer function of an IIR filter

Transfer function

- We recall that the relations between input $x[n]$ and output $y[n]$ is defined as

$$\sum_{k=0}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k]$$

- Taking the Fourier transform of both terms in the equality we find

$$\sum_{k=0}^{N} a_k \, e^{-i2\pi fk} Y(e^{i2\pi f}) = \sum_{k=0}^{M} b_k \, e^{-i2\pi fk} X(e^{i2\pi f})$$

- The Transfer function of the filter is then

$$H(e^{i2\pi f}) = \frac{Y(e^{i2\pi f})}{X(e^{i2\pi f})} = \frac{\sum_{k=0}^{M} b_k \, e^{-i2\pi fk}}{\sum_{k=0}^{N} a_k \, e^{-i2\pi fk}}. \quad (25)$$

- The transfer function is a rational function of polynomials of $e^{-i2\pi f}$.

Factorization of the transfer function

Zero/poles factorization

- The transfer function of a recurrent filter is

$$H(e^{i2\pi f}) = \frac{Y(e^{i2\pi f})}{X(e^{i2\pi f})} = \frac{\sum_{k=0}^{M} b_k \, e^{-i2\pi fk}}{\sum_{k=0}^{N} a_k \, e^{-i2\pi fk}}. \quad (26)$$

- It can be factorized as

$$H(e^{i2\pi f}) = \frac{b_0}{a_0} \frac{\prod_{k=1}^{M}(1 - c_k e^{-i2\pi f})}{\prod_{k=1}^{N}(1 - d_k e^{-i2\pi f})}. \quad (27)$$

- c_k are the zeros and d_k are the poles of the transfer function.
- Modulus and phase are easier to interpret in with the factorization.
- Easier to make a bode plot by treating each pole/zero independently.

Frequency response for discrete signals

Modulus and Gain

The modulus of the transfer function as a function of $w = 2\pi f$ is:

$$|H(e^{iw})| = \frac{|b_0|}{|a_0|} \frac{\prod_{k=1}^{M} |1 - c_k e^{-iw}|}{\prod_{k=1}^{N} |1 - d_k e^{-iw}|}$$

The gain in dB can be expressed as $G(w) = 20 \log(|H(e^{iw})|)$

$$G(w) = 10 \log_{10} \frac{|b_0|^2}{|a_0|^2} + \sum_{k=1}^{M} 10 \log_{10} |1 - c_k e^{-iw}|^2 - \sum_{k=1}^{N} 10 \log_{10} |1 - d_k e^{-iw}|^2.$$

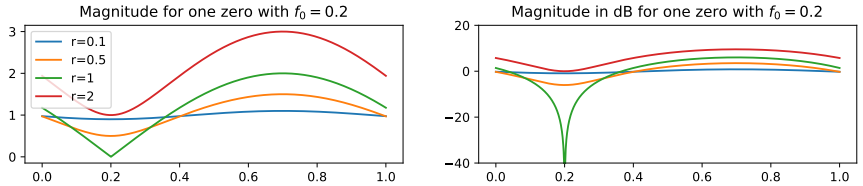
where the difference between poles and zeros is only a sign.

Phase

The phase of the transfer function can be expressed similarly

$$Arg(H(e^{iw})) = Arg(\frac{b_0}{a_0}) + \sum_{k=1}^{M} Arg(1 - c_k e^{-iw}) - \sum_{k=1}^{N} Arg(1 - d_k e^{-iw}).$$

Example for one pole/zero



- We study a transfer function with a unique zero $c_1 = re^{i2\pi f_0}$:

$$H(e^{2i\pi f}) = (1 - re^{i2\pi f_0}e^{-i2\pi f})$$

- The gain in dB can be expressed as

$$G_dB(f) = 10\log_{10}|1 - re^{i2\pi f_0}e^{-i2\pi f}|^2$$

- The magnitude reaches a minimum for $f = f_0$ and a maximum in $f = f_0 + \frac{1}{2}$.
- The attenuation in $f = f_0$ is perfect when $r = 1$.
- The Phase fo the transfer function is:

$$Arg(H(e^{i2\pi f})) = \arctan\left[\frac{r\sin(2\pi(f - f_0))}{1 - r\cos(2\pi(f - f_0))}\right].$$

The Z-transform (1)

Definition

The Z-transform is a generalization of the Fourier transform for discrete signals. It can be computed as the Laurent series:

$$H(z) = \mathcal{Z}(h[n]) = \sum_{n=-\infty}^{+\infty} h[n]z^{-n}. \tag{28}$$

where $z \in \mathbb{C}$ is a complex.

Region of Convergence (ROC)

- The Z-transform is always associated to its region of convergence.
- The Laurent series is said to be convergent if

$$\sum_{n=-\infty}^{+\infty} |h[n]|\,|z|^{-n} < +\infty.$$

- The Region of Convergence is defined as

$$ROC(H) = \left\{z : \left|\sum_{n=-\infty}^{\infty} h[n]z^{-n}\right| < \infty\right\}$$

- This region depends only on $|z|$.

The Z-transform (2)

Region of Convergence (2)

- There always exists ρ_1 and ρ_2 such that $H(z)$ is convergent for $\rho_1 < |z| < \rho_2$ and divergent for $|z| < \rho_1$ or $|z| > \rho_2$.
- When $H(z)$ converges for $|z| = 1$ we recover the Discrete Time Fourier Transform of the signal.
- For a causal filter, if $H(z)$ converges with $|z| = \rho$, it converges with $|z| > \rho$ and $\rho_2 = \infty$.
- If a filter is stable ($\sum_n |h[n]| < \infty$) the $H(z)$ is convergent for $|z| = 1$, if it is causal and stable it is convergent for $|z| \geq 1$

Example

Let $h[n] = \Gamma[n]\phi^n$ with $\phi > 0$ be a causal impulse response of a filter.

$$\sum_{n=-\infty}^{\infty} h[n]z^{-n} = \sum_{n=0}^{\infty} \phi^n z^{-n} = \sum_{n=0}^{\infty}\left(\frac{\phi}{z}\right)^n = \frac{1}{1 - \phi z^{-1}}.$$

Note that the series converges for $|\phi z^{-1}| < 1$ hence the region of convergence is $|z| > \phi$

Inverse Z-transform

Definition

The inverse Z-transform depends on the ROC and can be expressed as

$$x[n] = \mathcal{Z}^{-1}\{X(z)\} = \frac{1}{2\pi j}\oint_C X(z)z^{n-1}dz \tag{29}$$

where C is a counterclockwise closed path encircling the origin and entirely in the $ROC(H)$. Usually solved using Cauchy's residue theorem.

- When $|z| = 1$ is in the $ROC(H)$ one can compute the inverse Fourier transform for discrete signals:

$$x[n] = \int_0^1 X(e^{i2\pi f})e^{i2\pi fn}df$$

Example

Let $H(z) = \frac{1}{1-\phi z^{-1}}$ and $ROC(H) = \{z\,|\,|z| > \rho\}$.
This ROC means that $h[n]$ is causal and we recover

$$H(z) = \frac{1}{1 - \phi z^{-1}} = \sum_{n=0}^{+\infty}\phi^n z^{-n} = \sum_{n=-\infty}^{+\infty}\phi^n\Gamma[n]z^{-n} \quad \rightarrow \quad h[n] = \phi^n\Gamma[n]$$

Properties of Z-transform

Some properties

- **Linearity** : $\mathcal{Z}[a_1 x_1[n] + a_2 x_2[n]] = a_1 X_1(z) + a_2 X_2(z)$
- **Time reversal** : $\mathcal{Z}[x[-n]] = X(z^{-1})$
- **Time delay** : $\mathcal{Z}[x[n - n_0]] = X(z) z^{-n_0}$
- **Differentiation** : $\mathcal{Z}[nx[n]] = -z \frac{dX(z)}{dz}$
- **Convolution** : $\mathcal{Z}[x[n] \star h[n]] = X(z)H(z)$, and $ROC = ROC(X) \cap ROC(H)$
- **Scaling in the z-domain** : $\mathcal{Z}[a^n x[n]] = X(a^{-1}z)$ (also scales the ROC)
- **Accumulation** : $\mathcal{Z}[\sum_{k=-\infty}^{n} x[k]] = X(z) \frac{1}{1-z^{-1}}$

Examples of Z-transform

- Dirac δ: $\mathcal{Z}[\delta[n - n_0]] = z^{-n_0}$, and $ROC = \{z | 0 < |z| < \infty\}$
- Unitary step function : $\mathcal{Z}[\Gamma[n]] = \frac{1}{1-z^{-1}}$, and $ROC = \{z | |z| > 1\}$
- $\mathcal{Z}[a^n \Gamma[n]] = \frac{1}{1-az^{-1}}$, and $ROC = \{z | |z| > |a|\}$
- $\mathcal{Z}[\cos(w_0 n)\Gamma[n]] = \frac{1-z^{-1}\cos(w_0)}{1-2z^{-1}\cos(w_0)+z^{-2}}$, and $ROC = \{z | |z| > 1\}$

Z-transform of recurrent filters

- The Z-transform of a recurrent filter can be expressed as

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{\sum_{k=0}^{N} a_k z^{-k}}.$$

- By using a polynomial identification when d_k are the simple poles of $H(z)$ one can reformulate this transform as

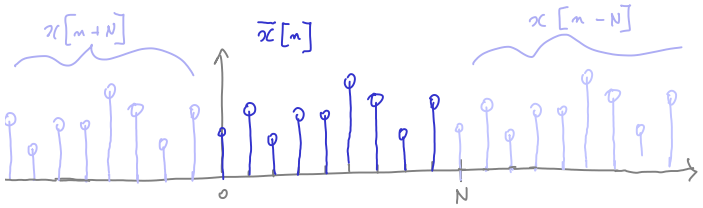
$$\hat{h}(z) = \sum_{r=0}^{M-N} B_r z^{-r} + \sum_{k=0}^{N} \frac{A_k}{1 - d_k z^{-1}}.$$

- The causal filter corresponding to this Z-transform has then the following impulse response

$$h[n] = \sum_{r=0}^{M-N} B_r \delta[n - r] + \sum_{k=0}^{N} A_k (d_k)^n \Gamma[n].$$

  If $H(z)$ has multiple poles then the decomposition is done with the exponent.
- Note that a filter is causal and stable if and only if all its poles $|d_k| < 1$.

Finite discret signals



Finite discrete signals

- Most of the theoretical results seen up to now correspond to signals $x[n]$ where $n \in \mathbb{Z}$.
- In practice recordings are only done for a finite amount of time resulting to only N samples.
- We defined $\tilde{x}[n]$ a finite signal of N samples with $n \in \{0, \ldots, N-1\}$.
- We use in the following the periodization of $\bar{x}[n]$

$$x[n] = \tilde{x}[n \bmod N]$$

  where $\bmod$ is the modulo operator.

Circular convolution

Discrete convolution of finite signals

The convolution between $\tilde{x}[n]$ and $\tilde{h}[n]$ both finite signals of N samples can be expressed as:

$$\tilde{y}[n] = \tilde{x}[n] \star \tilde{h}[n] = \sum_{p=-\infty}^{+\infty} \tilde{x}[p]\tilde{h}[n-p] \tag{30}$$

- It requires values for the signals outside of the sampling widow.
- One common approach consists in having $\tilde{x}[n]$ and $\tilde{h}[n]$ equal to 0 outside the sampling interval. Other choices can be done (see next slides)

Circular convolution

When using the periodic version of the signals the circular convolution can be computed on a unique period of size N:

$$x \circledast h[n] = \sum_{p=0}^{N-1} x[p]h[n-p].$$

The circular convolution is rarely appropriate in real life images due to border effects.

Discrete convolution as matrix multiplication

Vector representation and convolution matrix

- Finite signal x of N samples can be represented as a vector $\mathbf{x} \in \mathbb{C}^N$.
- The convolution operator is linear and can be expressed as:

$$\mathbf{y} = \mathbf{x} \star \mathbf{h} = \mathbf{C_h x}$$

Where $\mathbf{C_h} \in \mathcal{M}_{\mathbb{C}}(N, N)$ is a convolution matrix parametrized by vector $\mathbf{h}$.

Discrete convolution

The convolution operator when the values outside the support are 0 can be expressed as

$$\mathbf{C_h} = \begin{bmatrix} h[0] & 0 & \cdots & 0 \\ h[1] & h[0] & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ h[N-1] & h[N-2] & \cdots & h[0] \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & h[N-1] \end{bmatrix}$$

where $\mathbf{C_h} \in \mathcal{M}_{\mathbb{C}}(2 * N - 1, N)$ is a Toeplitz matrix.

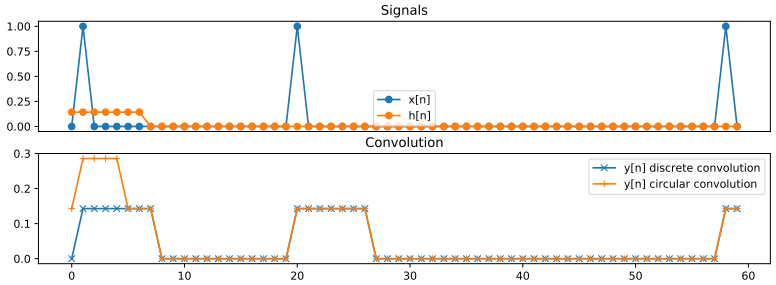
Circular convolution

The circular convolution operator can be expressed as

$$\mathbf{C_h} = \begin{bmatrix} h[0] & h[N-1] & \cdots & h[1] \\ h[1] & h[0] & \cdots & h[2] \\ \vdots & \vdots & \ddots & \vdots \\ h[N-1] & h[N-2] & \cdots & h[0] \end{bmatrix}$$

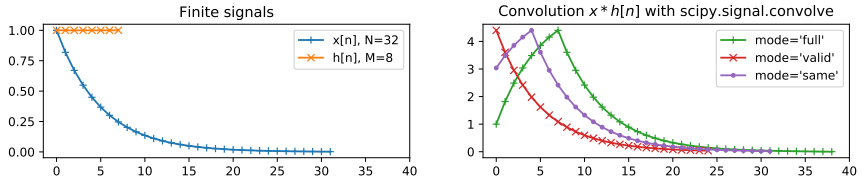
where $\mathbf{C_h} \in \mathcal{M}_{\mathbb{C}}(N, N)$ is a circulant Toeplitz matrix.

Border effects example



- Convolution between diracs $\tilde{x}[n]$ and a shape $h[n]$ will repeat the shape at the diracs position.
- A dirac at the end of the signal will cut the shape for discrete convolution where the outside of the sampling is 0.
- With circular convolution the shape is repeated t the beginning of the signal.
- One can remove border effects by creating virtual periodic signal with zeros (zero padding, see fast convolution).

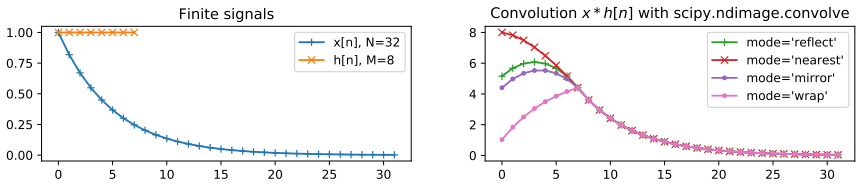
Discrete convolution in practice



The Scipy `scipy.signal.convolve` function:

- Convolution between two signals of support respectively N and M samples supposing that their values are 0 outside of the support.
- The third parameter is `mode` that change the size of the output :
  - `mode='full'` returns a signal of support $N + M - 1$ (default).
  - `mode='valid'` returns a signal of support $|N - M| + 1$ with only the samples that do not rely on zeros padding of the larger signal.
  - `mode='same'` returns a signal of the same size as the first input.
- Parameter `method` allows to choose between `'direct'` computation and `'fft'` and selects the most efficient by default.

Discrete convolution in practice (2)



The Scipy `scipy.ndimage.convolve` function:

- Always return the same size as the first parameter by default.
- The `mode` parameter allows selecting the borders of a signal $x = (abcd)$:
  - `mode='reflect'` : $(dcba|abcd|dcba)$ (default)
  - `mode='constant'` : $(kkkk|abcd|kkkk)$
  - `mode='nearest'` : $(aaaa|abcd|dddd)$
  - `mode='mirror'` : $(dcb|abcd|cba)$
  - `mode='wrap'` : $(abcd|abcd|abcd)$ (circular convolution)
- Parameter `origin` allows to select the origin of the filter h.

Discrete Fourier Transform (DFT)

Definition

The discrete Fourier Transform of a periodic signal $x[n]$ can be expressed as

$$X[k] = \sum_{p=0}^{N-1} x[p] e^{\frac{-i2\pi k}{N} p}. \tag{31}$$

- The DFT of periodic discrete signal is also periodic and discrete.
- This means that both the signal and its Fourier Transform can be stored in memory in a size N vector.
- The frequency domain is sampled regularly between 0 and $\frac{N-1}{N}$.
- The Inverse Discrete Fourier Transform (IDFT) can be computed as

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{\frac{i2\pi n}{N} k}. \tag{32}$$

- The complexity for computing naively the DFT is $\mathcal{O}(N^2)$.

Finite signal and vector space

Vector space of finite signals

- The space of finite signals is a finite space of scalar product and norm

$$< \mathbf{x}, \mathbf{h} >= \sum_{k=0}^{N-1} x[k] h^*[k], \quad \|\mathbf{x}\|^2 =< \mathbf{x}, \mathbf{x} >= \sum_{k=0}^{N-1} |x[k]|^2$$

- The family of discrete exponentials $(e_k[n])_{0 \leq k \leq N-1}$ such that $e_k[n] = e^{\frac{i2\pi k}{N} n}$, is an orthogonal basis of the space of finite discrete signals of period N.

DFT as a change of basis

- A signal $x[n]$ can be decomposed on the basis of complex exponentials:

$$x[n] = \sum_{k=0}^{N-1} \frac{< \mathbf{x}, \mathbf{e}_k >}{\|e_k\|^2} e_k[n]. \tag{33}$$

- The DFT can be computed as : $X[k] =< \mathbf{x}, \mathbf{e}_k >$
- Since $\|\mathbf{e}_k\|^2 = N$ we can recover the IDFT as

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{\frac{i2\pi k}{N} n}. \tag{34}$$

Properties of DFT

Parseval-Plancherel identity for finite discrete signals

Since the family $(e_k[n])_{0 \leq k \leq N-1}$ is orthogonal, we can recover the Plancherel identify for discrete signals as

$$\sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X[k]|^2. \tag{35}$$

Circular convolution

The circular convolution $y[n] = x[n] \circledast h[n]$ is a signal of period N and its Discrete Fourier Transform can be expressed as:

$$Y[k] = X[k] H[k] \tag{36}$$

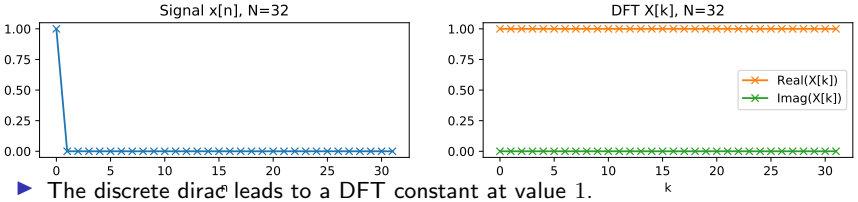
This will be used for fast convolution with FFT.

Border effect

- The DFT supposes that the signal is periodic.
- When the signal is recorded, there is no reason for it to be periodic, $x[N-1]$ and $x[0]$ can be very different.
- This can introduce some high frequencies in practice.

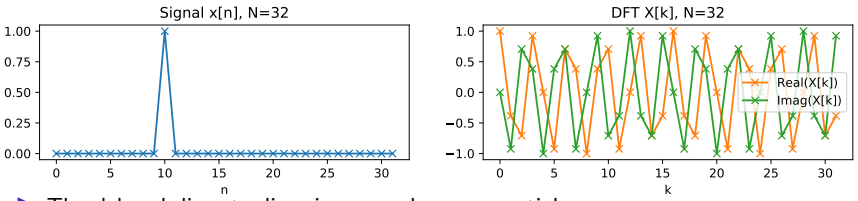
Examples of DFT (1)

$x[n] = \delta[n]$ with period N



- The discrete dirac leads to a DFT constant at value 1.
- It does not depends o N.

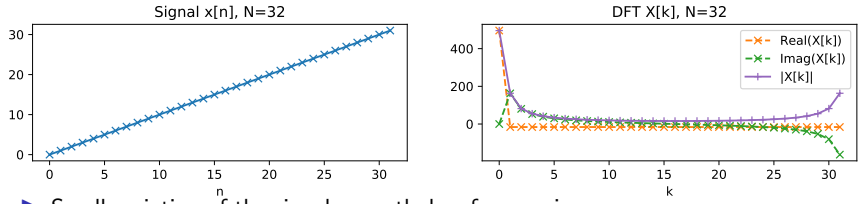
$x[n] = \delta[n - n_0]$ with period N



- The delayed discrete dirac is a complex exponential.
- DFT magnitude $|X[k]|$ constant at value 1.

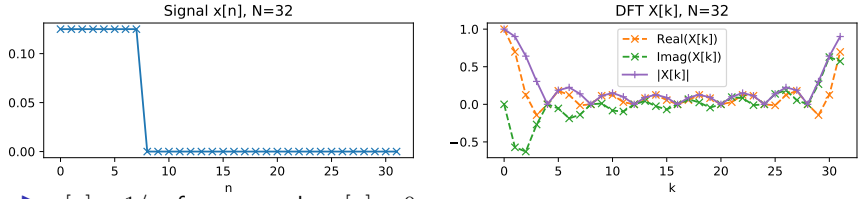
Examples of DFT (2)

$x[n] = n$ **with period** N



- Small variation of the signal : mostly low frequencies.
- The large change due to the periodicity requires high frequencies.

Average filtering

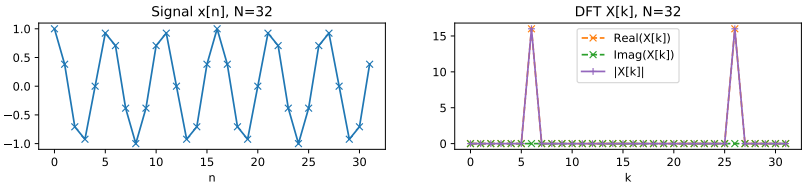


- $x[n] = 1/m$ for $n < m$ else $x[n] = 0$.
- Recognize the periodic sinc corresponding to rectangular function.

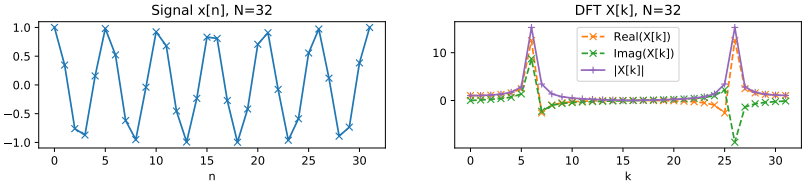
Examples of DFT (3)

$x[n] = \cos(2\pi f_0 n)$ **with period** N

- When $f_0 = \frac{6}{N} = \frac{k}{N}$ (one of the sampled frequencies):



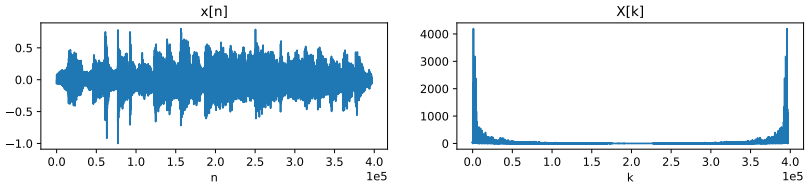
- When $f_0 = \frac{6.2}{N} \neq \frac{k}{N}$:



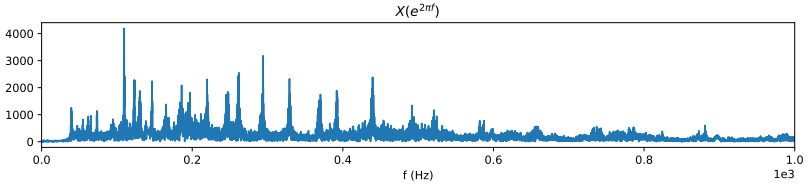
Examples of DFT (4)

Stairway to Heaven

- First 10 seconds of "Stairway to Heaven" from Led Zeppelin sampled at 44100Hz.



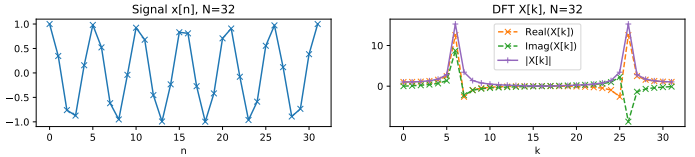
- Zoom to $[1, 1000]$ Hz and real life frequency:



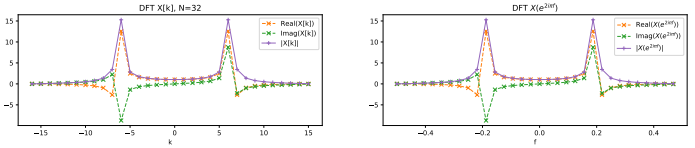
Plotting the DFT and fftshift

- DFT is defined on a finite number of frequencies $\frac{f_s k}{N}$ and periodic.
- When sampling at frequency f_s we suppose that the signal has filtered to avoid aliasing.
- All relevant frequencies can be expressed between $\frac{f_s k}{N}$ with $k = -N/2, \ldots, N/2 - 1$.
- One often use what is called fftshift to center the 0 frequency.

DFT of cosine



When doing fftshift:



Fourier Transform as matrix multiplication

$$\mathbf{x} = \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \vdots \\ x[N-1] \end{bmatrix}, \quad \mathbf{F}_N = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & e^{-\frac{i2\pi}{N}} & e^{-\frac{i4\pi}{N}} & \dots & e^{\frac{-i2\pi(N-1)}{N}} \\ 1 & e^{-\frac{i4\pi}{N}} & \ddots & & e^{-\frac{i4\pi(N-1)}{N}} \\ \vdots & \vdots & & \ddots & \vdots \\ 1 & e^{-\frac{i2\pi(N-1)}{N}} & e^{-\frac{i4\pi(N-1)}{N}} & \dots & e^{-\frac{i2\pi(N-1)(N-1)}{N}} \end{bmatrix}$$

Vector representation

- Periodic signal x can be represented as a vector $\mathbf{x} \in \mathbb{C}^N$.
- The DFT is a Linear operator that can be represented as a matrix $\mathbf{F}_N = [\mathbf{e}_0, \mathbf{e}_1, \dots, \mathbf{e}_{N-1}]^T \in \mathcal{M}_{\mathbb{C}}(N, N)$ of components

$$F_{k,p} = e^{-\frac{i2\pi(p-1)}{N}(k-1)}$$

DFT and IDFT

- The DFT of $\mathbf{x}$ can be computed as

$$\mathbf{x}^{DFT} = \mathbf{F}\mathbf{x}$$

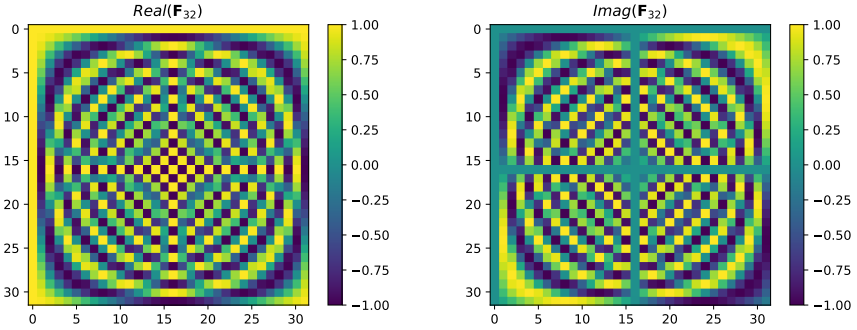
- The IDTF can be computed with

$$\mathbf{x} = \mathbf{F}^{-1}\mathbf{x}^{DFT} = \frac{1}{N}\mathbf{F}^H\mathbf{x}^{DFT}$$

where $\mathbf{F}^H$ is the conjugate transpose of $\mathbf{F}$.

Fourier Transform matrix

$$\mathbf{F}_1 = [1], \quad \mathbf{F}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \mathbf{F}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$$



- Each line in the matrix is a basis function $e_k[n]$ ordered by increasing frequency.
- The lines of the matrix contain the eigenvectors of all circular convolution matrices (hence the point-wise multiplication in Fourier domain).

Fast Fourier Transform (1)

- DFT for a finite signal of size N can be computed with

$$X[k] = \sum_{p=0}^{N-1} x[p] e^{\frac{-i2\pi k}{N}p}. \tag{37}$$

- It is of complexity $\mathcal{O}(N^2)$ since it requires for each element $X[k]$ the computation of $\mathcal{O}(N)$ operations.
- Reorganizing the computation, one can decrease greatly the computational complexity.

History of FFT

- Traced back to Carl Friedrich Gauss in 1805 who used it to interpolate trajectories of asteroids Pallas and Juno
- Rediscovered several time until publication by [Cooley and Tukey, 1965].
- Considered one of the most important algorithm in history.

Fast Fourier Transform (2)

Principle of radix-2 decimation-in-time (DIT)

First one can decompose the DFT between even and odd part of the sum:

$$\begin{aligned} X[k] &= \sum_{p=0}^{N-1} x[p] e^{\frac{-i2\pi k}{N}p} \\ &= \sum_{p=0}^{N/2-1} x[2p] e^{\frac{-i2\pi k}{N}2p} + \sum_{p=0}^{N/2-1} x[2p+1] e^{\frac{-i2\pi k}{N}(2p+1)} \\ &= \sum_{p=0}^{N/2-1} x[2p] e^{\frac{-i2\pi k}{N/2}p} + e^{\frac{-i2\pi k}{N}} \sum_{p=0}^{N/2-1} x[2p+1] e^{\frac{-i2\pi k}{N/2}p} \\ &= E[k] + e^{\frac{-i2\pi k}{N}} O[k] \end{aligned}$$

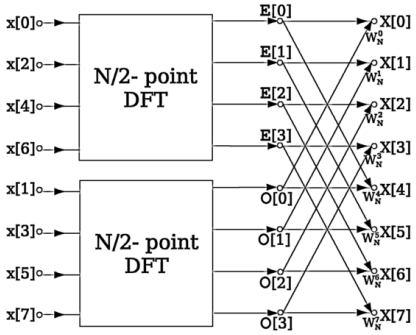
where $E[k]$ is the DFT of $e[n] = x[2n]$ on even samples of x and $O[k]$ the DFT of $o[n] = x[2n+1]$ on odd samples of x.
Note that using the periodicity of the complex exponential on can also recover:

$$X[k + N/2] = E[k] + e^{\frac{-i2\pi(k+N/2)}{N}} O[k] = E[k] - e^{\frac{-i2\pi k}{N}} O[k]$$

Fast Fourier Transform (3)

Cooley–Tukey FFT algorithm [Cooley and Tukey, 1965]

Algorithm : $FFT(N, x[n])$
if $N = 1$ **then**
  return $x[0]$
else
  $E[k] = FFT(N/2, x[2n])$
  $O[k] = FFT(N/2, x[2n+1])$
  for $k = 0$ **to** N/2 **do**
    $X[k] = E[k] + e^{\frac{-2i\pi k}{N}} O[k]$
    $X[k+N/2] = E[k] - e^{\frac{-2i\pi k}{N}} O[k]$
  end for
  return $X[k]$
end if



- Classical divide and conquer approach.
- Recurrent algorithm often called butterfly due to the crossing of even/odds.
- Can be implemented with limited memory use only $\mathcal{O}(N)$
- Has been extended to general N through factorization.
- The inverse FFT can be computed with $IFFT(X[k]) = \frac{1}{N} FFT(X^*[k])^*$

Fast Fourier Transform (4)

Complexity of FFT

The complexity $C(N)$ of $FFT(N, x[n])$ car be obtained from the recurrence:

$$C(N) = 2C(\frac{N}{2}) + 4N$$

Let us suppose that $L = \log_2(N)$ is an integer, then we have:

$$\frac{C(N)}{N} = \frac{C(\frac{N}{2})}{\frac{N}{2}} + K$$

where $K = 4$. With $T(L) = \frac{C(N)}{N}$ we can see that

$$T(L) = T(L-1) + K = T(0) + \sum_{l=1}^{L-1} K = \mathcal{O}(KL + T(0))$$

Since $T(0) = 0$ we recover the final complexity:

$$C(N) = \mathcal{O}(KN \log_2(N))$$

Fast convolution

Convolution for finite signals

When two signals $x[n]$ and $h[n]$ have a support on $[0, N-1]$ their convolution is

$$y[n] = x[n] \star h[n] = \sum_{k=0}^{N-1} x[k]h[n-k]$$

and the support of the convolution is in $[0, 2N-1]$. Computing the values on the support is $\mathcal{O}(N^2)$.

Fast convolution with zero-padding

- In order to avoid border effect we define two signals $a[n]$ and $b[n]$ of periodicity $2N$ such that:

$$a[n] = \begin{cases} x[n] & \text{for } 0 \leq n < N \\ 0 & \text{for } N \leq n < 2N \end{cases}, \quad b[n] = \begin{cases} h[n] & \text{for } 0 \leq n < N \\ 0 & \text{for } N \leq n < 2N \end{cases}$$

- This procedure is called zero-padding and ensures that the circular convolution $c[n] = a \circledast b[n]$ can recover the regular convolution:

$$c[n] = y[n] \quad \text{for } 0 \leq n < 2N. \tag{38}$$

- The complexity of Fast convolution with zero-padding is $\mathcal{O}(N \log_2(N))$.

Finding the most efficient convolution

Convolution between two signals of different support

- In applications we often have two signal $x[n]$ and $h[n]$ that have different lengths N and M.
- Complexity for the different convolution approaches is:
  - Direct computation: $\mathcal{O}(NM)$
  - FFT computation : $\mathcal{O}(\max(M,N) \log_2(\max(M,N)))$
- Some applications such as convolutional neural networks have a very small M and direct implementation is more efficient.

`choose_conv_method` Scipy function

- Scipy convolution `scipy.signal.convolve` automatically selects the more efficient (and precise enough) approach.
- Function `choose_conv_method` in module `scipy.signal.signaltools`.
- Compare the number of operations for FFT and direct implementation.
- Can also do a quick benchmark to find which is the fastest in practice.

FFT implementations

FFT in the real world

- FFT is a standard algorithm implemented in numerous numerical libraries.
- Implementations for all types of input (float, double, complex).
- Implementation for all devices (X86, ARM, GPU, DSP).
- Implemented in Numpy and Scipy.

Common implementations

- **FFTPACK** FORTRAN [Swarztrauber, 1982] (public domain).
- **GNU scientific library** [Galassi et al., 1996] (GPL).
- **Intel Math Kernel Library** (proprietary).
- **FFTW** Fastest Fourier Transform in The West [Frigo and Johnson, 1998] (GPL).
- **pyFFTW** a python wrapper for FFTW (GPL).
- **CuFFT** Runs on GPU [NVIDIA, 2013], C and Python (proprietary).

Applications of Digital Signal Processing

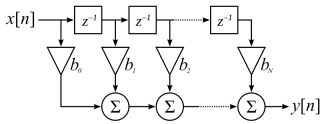
Digital Signal Processing

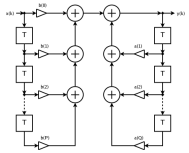
- Processing of signals after Analog to Digital Conversion.
- Can be implemented on any general purpose computer or on dedicated devices/chips for more efficiency.
- Objectives of DSP include denoising, signal reconstruction and source separation.

Applications discussed in the following

- Digital filter design.
- Interpolation.
- Digital Image Processing.
- Convolutional neural networks.

Digital filter design



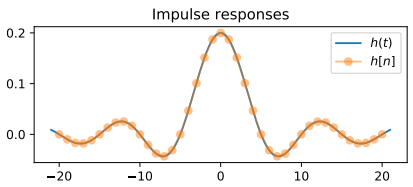


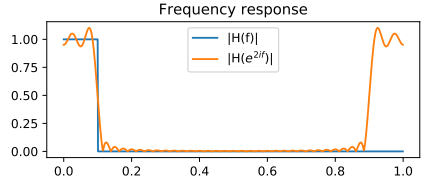
- Digital Filter design follows the same objective as Analog filter designs.
- For finite signal already in memory ideal filtering is possible with FFT.
- When filter has to be applied in real time one needs to find the parameters of a recursive IIR filter or FIR filter.
- Scipy provides several digital filter design functions.

Approaches for digital filter design

- **Window design method (FIR)**: Sample and truncate the impulse response of an analog filter.
- **Least Mean Square Error method (FIR)**: Find the coefficients of the FIR that approximates the best in the square error sens the objective filter in frequency.
- **Bilinear transform (IIR)**: non-linear approximation of the transfer function on an analog filter.

Window design method





Principle

$$h[n] = \begin{cases} h(nT) & \text{for } -M \leq n \leq N \\ 0 & \text{else} \end{cases}$$

- The sampling can lead to aliasing, the truncation leads to ripples in frequencies.
- Works better for low frequencies and large M, N.
- Can be used to approximate the ideal filter:

$$h(t) = 2f_c \frac{\sin(2\pi f_c t)}{2\pi f_c t} = 2f_c \text{sinc}(2\pi f_c t)$$

- Example above with $f_c = 0.1$, $M = 19, N = 20$.

Bilinear transform

Principle

- For a continuous time transfer function that can be expressed as

$$H(f) = H_0(2i\pi f)$$

- The bilinear transformation use the change of variable:

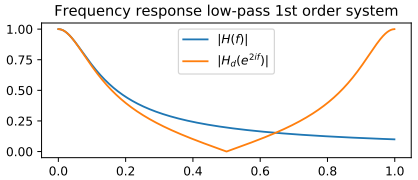
$$2i\pi f \to F(e^{2i\pi f}) = \frac{2}{T}\frac{1-e^{-2i\pi f}}{1+e^{-2i\pi f}} = \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}} = \frac{2}{T} i\tan(\pi f)$$

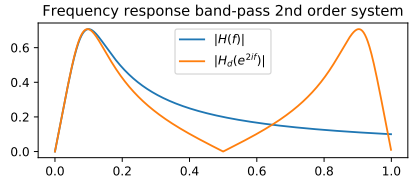
  T is a parameter that can be used to change the cutoff frequency.
- The digital Filer z-transform can be expressed as:

$$H_d(z) = H_0\left(\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}\right)$$

- Null frequency in continuous maps to null frequency in discrete.
- Infinite frequency in continuous maps to $1/2$ in discrete.
- Bilinear transform preserves stability properties.

Examples of bilinear transform





First order system

- The transfer function for a first order low-pass filter is of the form:

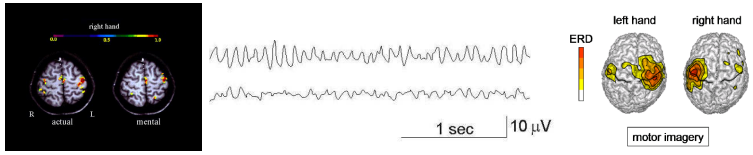
$$H(f) = \frac{1}{1+\frac{i2\pi f}{2\pi f_0}}$$

- The discrete transfer function can be recovered from

$$H_d(z) = \frac{1}{1+\frac{2}{2\pi f_0 T}\frac{1-z^{-1}}{1+z^{-1}}} = \frac{1-z^{-1}}{1+\frac{2}{2\pi f_0 T}+\left(1-\frac{2}{2\pi f_0 T}\right)z^{-1}}$$

- Comparison above for first order low-pass and second order Butterworth band-pass filter.

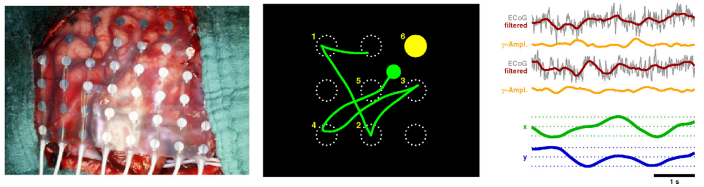
Application : motor imagery BCI



Motor imagery on ElectroEncephaloGraphy

- Motor Imagery is a paradigm of Brain Computer Interfaces (BCI).
- Uses the μ rhythms in the motor cortex (Band [8-40]Hz).
- The subject concentrates on a movement (right or left hand).
- The contralateral area of the motor cortex sees a desynchronization (lower energy in the band) similar to a real movement [Roth et al., 1996].
- Butterworth filter of order 5 used to perform filtering [Lotte and Guan, 2010].
- Used to control a cursor or a movement of a machine.

Application : motor control BCI



Prediction of arm movement from (EcoG) [Pistohl et al., 2008]

- ElectroCorticoGram implanted on patient suffering from pharmaco-resistant epilepsy.
- Recording of simultaneous arm movement and EcoG signals from patients.
- Use a model to predict arm movement from EcoG signals.
- Potential applications in prosthetics and robotics.
- Signals filtered using a discrete a Savitzky-Golay filter (order 3,win 0.75 sec).

Limited band signal interpolation

Sinc interpolation

- Interpolation of a sampled signal with the Nyquist/Shannon sampling Theorem 4 can be very expensive since the sinus cardinal has an infinite support.
- Computing M samples from N will be $\mathcal{O}(MN)$.

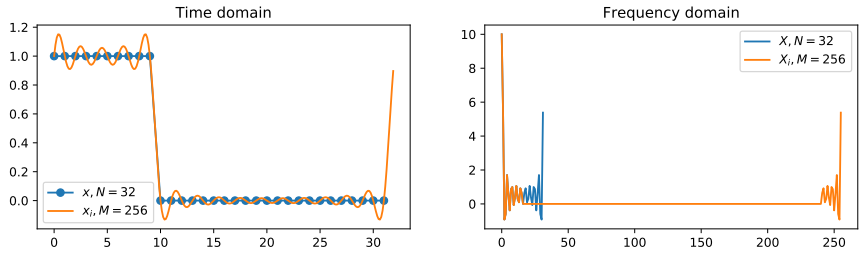
Fast interpolation

- Fast interpolation from N to M can be done using FFT with the Following steps:

1. $X[n] \leftarrow$ Compute FFT of $x[n]$ of size N.
2. $X_i[k] \leftarrow \begin{cases} X[k]\frac{M}{N} & \text{for } 0 \leq k < N//2 \\ X[k-M+N]\frac{M}{N} & \text{for } M - N/2 \leq k < M \text{ (zero padding)} \\ 0 & \text{else} \end{cases}$
3. $x_i[n'] \leftarrow$ Compute IFFT of $X_i[k]$ of size M.

- The $\frac{M}{N}$ is because the FFT is not normalized.
- For a sampling frequency of $T = 1$ of $x[n]$, sample $x_i[n']$ corresponds to $t = n'\frac{N}{M}$.
- For $M > N$ complexity for the interpolation is $\mathcal{O}(M \log_2(M))$.

Fast interpolation example



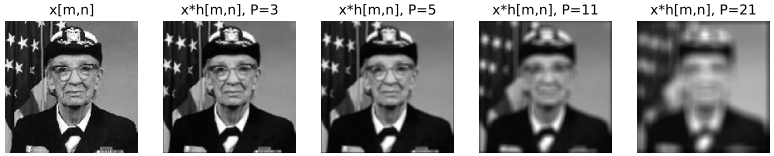
Interpolation example

- Interpolation of rectangular function.
- $N = 32, M = 256$.

Python code

```
X=np.fft.fft(x) # FFT
Xi=np.zeros(M)+0*1j # Initalize complex vector at 0
Xi[:N//2]=X[:N//2] # Positive low frequencies
Xi[-N//2:]=X[N//2:]# Negative low frequencies
xi=np.fft.ifft(X2*M/N) # Scaling + IFFT
ti=np.arange(M)*N/M # position in time of the interpolated samples
```


Digital Image Processing



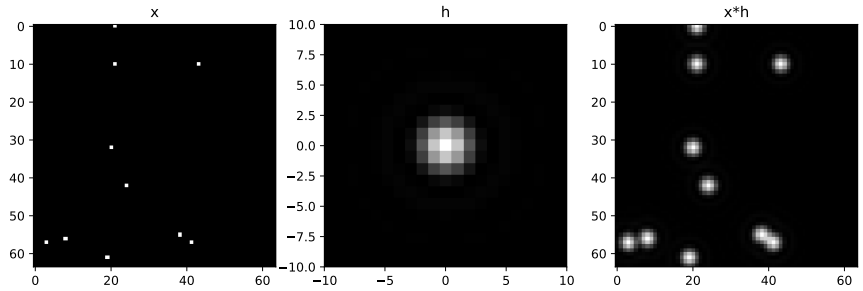
Digital images

- We now all have a digital camera in our pockets.
- Image capture has several limits:
  - Noise (especially at low illumination).
  - Motion blur on camera.
  - Limited resolution (apperture and sensor).
- Images are stored in memory as a matrix of integers (or float).

Signal processing in 2D

- All results seen up to now can be extended to 2D.
- In images one does not need causality (whole image available).
- Filters in 2D are often FIR filter.
- Small filters of the building blocks of Convolutional Neural Networks (CNN).

Image convolution

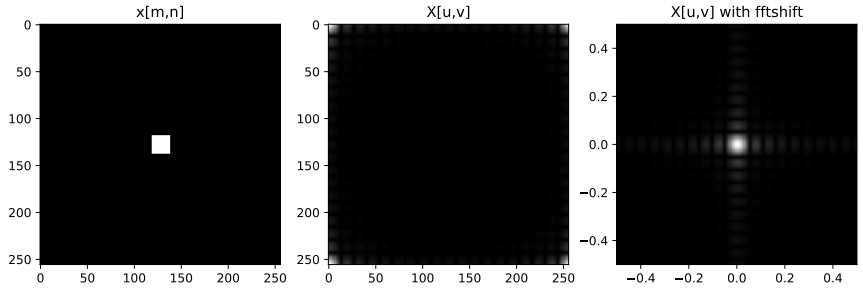


2D convolution

$$x \star h[m,n] = \sum_{k,l} x[k,l]h[m-k,n-l]$$

- In practice $x[m,n]$ is often a large image and the filter $h[m,n]$ a small filter (also called kernel).
- $h[m,n]$ is often of odd size and its support centered around $(0,0)$.
- Similarly to finite discrete signal there exists a circular 2D convolution that can be computed with FFT.

2D Fast Fourier Transform



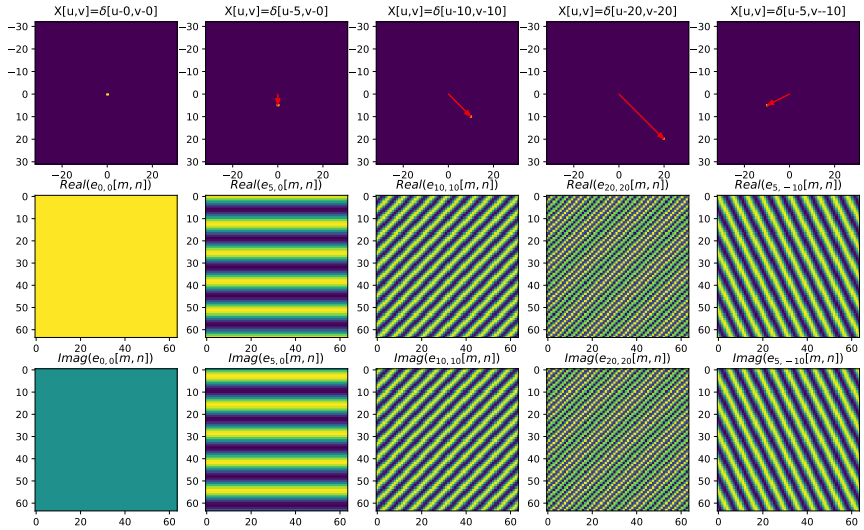
2D FFT

- let $x[m,n]$ be a finite image of size (M,N).
- The 2D DFT of the image is:

$$X[u,v] = \sum_{m=0,n=0}^{M,N} x[m,n] e^{-2i\pi\left(\frac{um}{M}+\frac{vn}{N}\right)} = \sum_{m=0}^{M}\left(\sum_{n=0}^{N} x[m,n] e^{-2i\pi\frac{vn}{N}}\right) e^{-2i\pi\frac{um}{M}}$$

- The DFT for each index/direction of the image are separable.
- Can be performed in $\mathcal{O}(MN\log_2(N)+NM\log_2(M))$.

2D basis functions

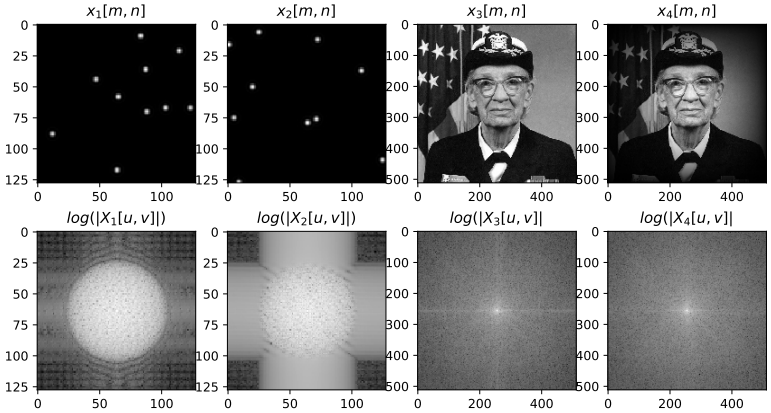


The basis functions

$$e_{u,v}[m,n] = e^{-2i\pi\left(\frac{um}{M}+\frac{vn}{N}\right)}$$

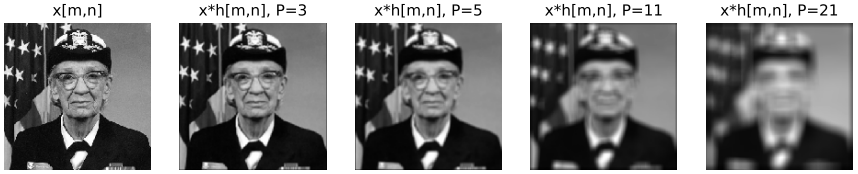
correspond to a dirac in the frequency domain.

Border effects on FFT



- DFT/FFT supposes that an image is periodic in space.
- Large differences between values at opposite borders can lead to high frequencies not existing in the original image.
- This can be attenuated using "windowing" or apodization that consists in attenuation the borders of the images.
- This spatial multiplication corresponds to a convolution (filtering) in the frequency domain (see next course).

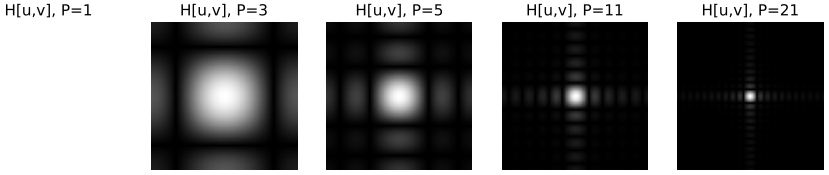
Classical 2D filters (1)



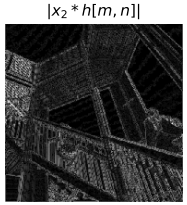
Average filtering

$$h_{a,P}[m,n] = \begin{cases} \frac{1}{P^2} & \text{for } |m| < P/2 \text{ and } |n| < P/2 \\ 0 & \text{else} \end{cases}, \quad h_{a,3} = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

- Classical low pass filter (with P odd).
- Parameter P defines its width and cutoff frequency
- Its FT is a multiplication of sinc in both directions (non isotropic):

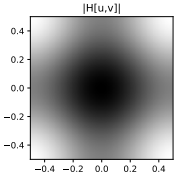


Classical 2D filters (2)



Spatial high pass

$$h = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



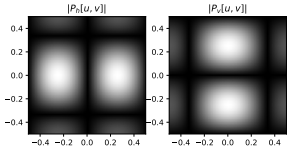
- Compute de difference between the center pixel and its neighbors.
- Non isotropic filter.
- Use absolute value of the output to detect area with large changes.

Classical 2D filters (3)

Edge detectors

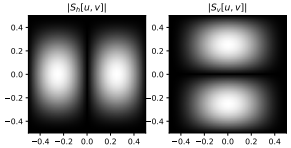
- Prewitt edge detector

$$p_h = \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \quad p_v = \frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$



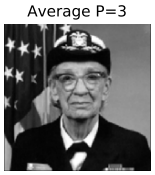
- Sobel edge detector

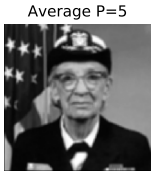
$$s_h = \frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad s_v = \frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

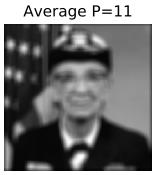


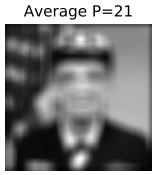
- Combination of a low pass in one direction and high-pass in the other.

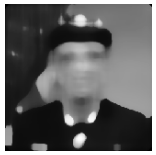
Classical 2D filters (4)











Median filtering

- Compute de median in a window of width P.
- Robust version of the average filter
- Non linear (no impulse response or FT representation).
- Bette able to preserve high frequencies and large objects details.
- There exists a whole family of more complex non-linear morphological filter.

Total Variation (TV)

- In images of man-made objects, we often have near constant parts in the image.
- One way to measure the presence of those constant parts is called the Total Variation that can be expressed in its anisotropic version as:

$$TV(x) = \sum_{m,n} |x[m+1,n] - x[m,n]| + |x[m,n+1] - x[m,n]|$$

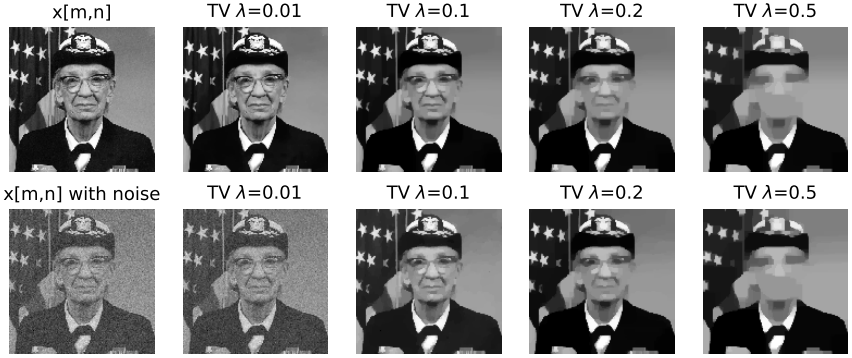
- It can be reformulated (forgetting border effects) as:

$$TV(x) = \|x * d_v\|_1 + \|x * d_h\|_1, \quad \text{with} \quad d_h = \begin{bmatrix} -1 & 1 \end{bmatrix}, \quad d_v = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

where d_v and d_h are finite differences derivations and $\|y\|_1 = \sum_{m,n} |y[m,n]|$.

- The presence of noise tends to introduce high frequencies and to greatly increase the TV of an image.

Total Variation denoising



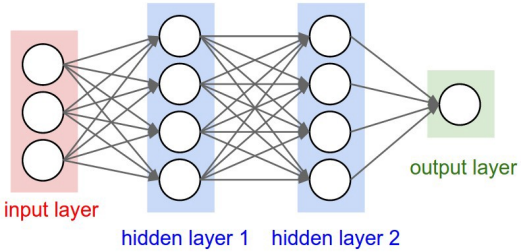
- Since TV increases with noise, one would want to use it for denoising.
- Total Variation denoising of image y can be expressed as

$$\min_x \quad \|x - y\|_F^2 + \lambda TV(x)$$

where $\|\cdot\|_F$ is the Fobenius norm of a matrix.

- We want the denoised image x to be close to y but also have a small TV.
- It is a convex but non-smooth optimization problem.

Deep learning



Deep neural network [LeCun et al., 2015]

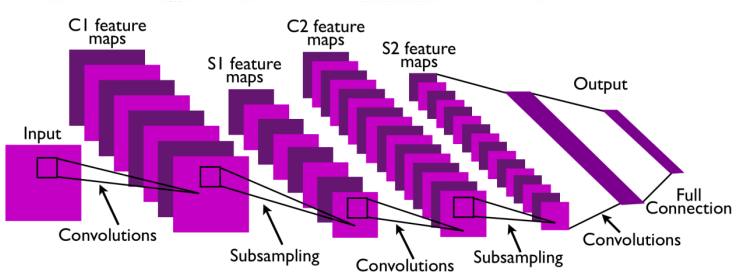
$$f(x) = f_K(f_{K-1}(...f_1(\mathbf{x})...)) \tag{39}$$

- f is a composition of basis functions f_k of the form:

$$f_k(x) = g_k(\mathbf{W}_k\mathbf{x} + \mathbf{b}_k) \tag{40}$$

- $\mathbf{W}_k$ is a linear operator and $\mathbf{b}_k$ is a bias for layer k.
- g_k is a non-linear activation function for layer k.
- Function f parameters $\{\mathbf{W}_k, \mathbf{b}_k\}_k$ are learned from the data.

Convolutional Neural Network (CNN)



- Replace the linear operator by a convolution [LeCun et al., 2010].
- Reduce image dimensionality with sub-sampling or max pooling.
- Number of parameters depends on the size fo the filter, not the image.
- Recent deep CNN use Relu activation [Glorot et al., 2011]: $g(\mathbf{x}) = \max(0, \mathbf{x})$.

Bibliography

- Discrete-time signal processing [Oppenheim and Shafer, 1999].
- Signals and Systems [Haykin and Van Veen, 2007].
- Signals and Systems [Oppenheim et al., 1997].
- Signal Analysis [Papoulis, 1977].
- Fourier Analysis and its applications [Vretblad, 2003].

- Polycopiés from Stéphane Mallat and Éric Moulines [Mallat et al., 2015].
- Théorie du signal [Jutten, 2018].
- Distributions et Transformation de Fourier [Roddier, 1985]

References I

[Cooley and Tukey, 1965] Cooley, J. W. and Tukey, J. W. (1965).
An algorithm for the machine calculation of complex fourier series.
Mathematics of computation, 19(90):297–301.

[Frigo and Johnson, 1998] Frigo, M. and Johnson, S. G. (1998).
Fftw: An adaptive software architecture for the fft.
In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, volume 3, pages 1381–1384. IEEE.

[Galassi et al., 1996] Galassi, M., Davies, J., Theiler, J., Gough, B., Jungman, G., Alken, P., Booth, M., and Rossi, F. (1996).
Gnu scientific library.
No. Release, 2.

[Glorot et al., 2011] Glorot, X., Bordes, A., and Bengio, Y. (2011).
Deep sparse rectifier neural networks.
In *Aistats*, volume 15, page 275.

[Haykin and Van Veen, 2007] Haykin, S. and Van Veen, B. (2007).
Signals and systems.
John Wiley & Sons.

References II

[Jutten, 2018] Jutten, C. (2018).
Théorie di signal.
Univ. Grenoble Alpes - Polytech' Grenoble.

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015).
Deep learning.
Nature, 521(7553):436–444.

[LeCun et al., 2010] LeCun, Y., Kavukcuoglu, K., and Farabet, C. (2010).
Convolutional networks and applications in vision, pages 253–256.

[Lotte and Guan, 2010] Lotte, F. and Guan, C. (2010).
Regularizing common spatial patterns to improve bci designs: unified theory and new algorithms.
IEEE Transactions on biomedical Engineering, 58(2):355–362.

[Mallat et al., 2015] Mallat, S., Moulines, E., and Roueff, F. (2015).
Traitement du signal.
Polycopié MAP 555, École Polytechnique.

[NVIDIA, 2013] NVIDIA, C. (2013).
Fast fourier transform library (cufft).

References III

[Nyquist, 1928] Nyquist, H. (1928).
Certain topics in telegraph transmission theory.
Transactions of the American Institute of Electrical Engineers, 47(2):617–644.

[Oppenheim and Shafer, 1999] Oppenheim, A. V. and Shafer, R. W. (1999).
Discrete-time signal processing.
Prentice Hall Inc.

[Oppenheim et al., 1997] Oppenheim, A. V., Willsky, A. S., and Nawab, S. H. (1997).
Signals and systems prentice hall.
Inc., Upper Saddle River, New Jersey, 7458.

[Papoulis, 1977] Papoulis, A. (1977).
Signal analysis, volume 191.
McGraw-Hill New York.

[Pistohl et al., 2008] Pistohl, T., Ball, T., Schulze-Bonhage, A., Aertsen, A., and Mehring, C. (2008).
Prediction of arm movement trajectories from ecog-recordings in humans.
Journal of neuroscience methods, 167(1):105–114.

[Roddier, 1985] Roddier, F. (1985).
Distributions et transformée de fourier.

References IV

[Roth et al., 1996] Roth, M., Decety, J., Raybaudi, M., Massarelli, R., Delon-Martin, C., Segebarth, C., Morand, S., Gemignani, A., Décorps, M., and Jeannerod, M. (1996).
Possible involvement of primary motor cortex in mentally simulated movement: a functional magnetic resonance imaging study.
Neuroreport, 7(7):1280–1284.

[Shannon, 1949] Shannon, C. E. (1949).
Communication in the presence of noise.
Proceedings of the IRE, 37(1):10–21.

[Swarztrauber, 1982] Swarztrauber, P. N. (1982).
Vectorizing the ffts, parallel computations (new york)(g. rodrigue, ed.).

[Vretblad, 2003] Vretblad, A. (2003).
Fourier analysis and its applications, volume 223.
Springer Science & Business Media.