Inverse Problems 1: convolution and deconvolution

Lesson 2: matrix representation and naïve inversion

Luca Ratti

September 4, 2019

University of Helsinki

Table of contents

Mathematical description of 1D convolution

Convolution formula

1D-signal: consider a vector $f \in \mathbb{R}^n$.

Point Spread Function: take $p \in \mathbb{R}^m$, $m = 2\nu + 1$.

Convolution formula

1D-signal: consider a vector $f \in \mathbb{R}^n$.

Point Spread Function: take $p \in \mathbb{R}^m$, $m = 2\nu + 1$.

The **convolution** between f and p is a vector $p * f \in R^n$ such that:

Convolution formula

$$(p * f)_j = \sum_{\ell=-\nu}^{\nu} p_\ell f_{j-\ell}$$

Convolution formula

1D-signal: consider a vector $f \in \mathbb{R}^n$.

Point Spread Function: take $p \in \mathbb{R}^m$, $m = 2\nu + 1$.

The **convolution** between f and p is a vector $p * f \in R^n$ such that:

Convolution formula

$$(p * f)_j = \sum_{\ell=-\nu}^{\nu} p_\ell f_{j-\ell}$$

Open problem: the formula requires the knowledge of the values $f_{-\nu+1}, f_{-\nu+2}, \ldots, f_0$ and also $f_{n+1}, \ldots, f_{n+\nu}$.

Continuous version and discretization

Discrete model (DM)

$f \in \mathbb{R}^n,\ f = (f_1, \ldots, f_n);$

$p \in \mathbb{R}^{2\nu+1},\ p = (p_\nu, \ldots, p_{-\nu}),$

$$\sum_{\ell=-\nu}^{\nu} p_\ell = 1; \quad p * f \in \mathbb{R}^n,$$

$$(p * f)_i = \sum_{\ell=-\nu}^{\nu} p_\ell f_{i-\ell}$$

Continuous version and discretization

Discrete model (DM)

$$f \in \mathbb{R}^n,\ f = (f_1, \ldots, f_n);$$
$$p \in \mathbb{R}^{2\nu+1},\ p = (p_\nu, \ldots, p_{-\nu}),$$
$$\sum_{\ell=-\nu}^{\nu} p_\ell = 1; \quad p * f \in \mathbb{R}^n,$$
$$(p * f)_i = \sum_{\ell=-\nu}^{\nu} p_\ell f_{i-\ell}$$

Continuous model (CM)

$$f : \mathbb{R} \to \mathbb{R},\ f = f(x);$$
$$p : [-\gamma, \gamma] \to \mathbb{R},$$
$$\int_{-\gamma}^{\gamma} p(x)dx = 1; \quad p * f : \mathbb{R} \to \mathbb{R},$$
$$(p * f)(x) = \int_{-\gamma}^{\gamma} p(y)f(x-y)dy$$

Continuous version and discretization

Discrete model (DM)

$f \in \mathbb{R}^n,\ f = (f_1, \dots, f_n);$

$p \in \mathbb{R}^{2\nu+1},\ p = (p_\nu, \dots, p_{-\nu}),$

$$\sum_{\ell=-\nu}^{\nu} p_\ell = 1; \quad p * f \in \mathbb{R}^n,$$

$$(p * f)_i = \sum_{\ell=-\nu}^{\nu} p_\ell f_{i-\ell}$$

Continuous model (CM)

$f : \mathbb{R} \to \mathbb{R},\ f = f(x);$

$p : [-\gamma, \gamma] \to \mathbb{R},$

$$\int_{-\gamma}^{\gamma} p(x)dx = 1; \quad p * f : \mathbb{R} \to \mathbb{R},$$

$$(p * f)(x) = \int_{-\gamma}^{\gamma} p(y)f(x - y)dy$$

- the real physical model is (CM): signals vary continuously in time or space, there is no 'minimum scale' for convolution;

Continuous version and discretization

Discrete model (DM)

$f \in \mathbb{R}^n,\ f = (f_1, \ldots, f_n);$

$p \in \mathbb{R}^{2\nu+1},\ p = (p_\nu, \ldots, p_{-\nu}),$

$$\sum_{\ell=-\nu}^{\nu} p_\ell = 1; \quad p * f \in \mathbb{R}^n,$$

$$(p * f)_i = \sum_{\ell=-\nu}^{\nu} p_\ell f_{i-\ell}$$

Continuous model (CM)

$f : \mathbb{R} \to \mathbb{R},\ f = f(x);$

$p : [-\gamma, \gamma] \to \mathbb{R},$

$$\int_{-\gamma}^{\gamma} p(x)dx = 1; \quad p * f : \mathbb{R} \to \mathbb{R},$$

$$(p * f)(x) = \int_{-\gamma}^{\gamma} p(y)f(x-y)dy$$

- the real physical model is (CM): signals vary continuously in time or space, there is no 'minimum scale' for convolution;
- discretization, the passage from (CM) to (DM), is mandatory for applications: sampling of signals, finite algebra of computers;

Continuous version and discretization

Discrete model (DM)

$f \in \mathbb{R}^n,\ f = (f_1, \dots, f_n);$

$p \in \mathbb{R}^{2\nu+1},\ p = (p_\nu, \dots, p_{-\nu}),$

$$\sum_{\ell=-\nu}^{\nu} p_\ell = 1; \quad p * f \in \mathbb{R}^n,$$

$$(p * f)_i = \sum_{\ell=-\nu}^{\nu} p_\ell f_{i-\ell}$$

Continuous model (CM)

$f : \mathbb{R} \to \mathbb{R},\ f = f(x);$

$p : [-\gamma, \gamma] \to \mathbb{R},$

$$\int_{-\gamma}^{\gamma} p(x)dx = 1; \quad p * f : \mathbb{R} \to \mathbb{R},$$

$$(p * f)(x) = \int_{-\gamma}^{\gamma} p(y)f(x-y)dy$$

- the real physical model is (CM): signals vary continuously in time or space, there is no 'minimum scale' for convolution;
- **discretization**, the passage from (CM) to (DM), is mandatory for applications: sampling of signals, finite algebra of computers;
- how to discretize: introduce a grid of points $x_1, \dots, x_n$ to represent (a segment of) the real line - represent f and p as vectors (interpolation, fitting, splines,...) - approximate integrals with sums (quadrature formulas).

Historical background

Many famous mathematicians contributed to the definition and the accurate description of convolution

Many famous mathematicians contributed to the definition and the accurate description of convolution

L. Euler
introduced convolution to solve ordinary differential equations

Historical background

Many famous mathematicians contributed to the definition and the accurate description of convolution

L. Euler
introduced convolution to solve ordinary differential equations

A.-L. Cauchy
introduced the discrete convolution formula

Historical background

Many famous mathematicians contributed to the definition and the accurate description of convolution

L. Euler
introduced convolution to solve ordinary differential equations

A.-L. Cauchy
introduced the discrete convolution formula

V. Volterra
introduced the notation with the symbol $*$

Extended points

In order to define the value of

$$(p * f)_1, \ldots (p * f)_\nu \text{ and } (p * f)_{n-\nu}, \ldots (p * f)_n$$

we need to prescribe the values of the extended signal points,

$$f_{-\nu+1}, \ldots f_0 \text{ and } f_{n+1}, \ldots f_{n+\nu}.$$

Two main possibilities:

Extended points

In order to define the value of

$$(p * f)_1, \ldots (p * f)_\nu \text{ and } (p * f)_{n-\nu}, \ldots (p * f)_n$$

we need to prescribe the values of the **extended signal points**,

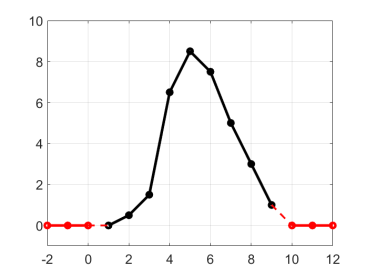
$$f_{-\nu+1}, \ldots f_0 \text{ and } f_{n+1}, \ldots f_{n+\nu}.$$

Two main possibilities:

zero padding:

$$f_{-\nu+1} = \ldots = f_0 = 0;$$

$$f_{n+1} = \ldots = f_{n+\nu} = 0;$$



Extended points

In order to define the value of

$$(p * f)_1, \dots (p * f)_\nu \text{ and } (p * f)_{n-\nu}, \dots (p * f)_n$$

we need to prescribe the values of the **extended signal points**,

$$f_{-\nu+1}, \dots f_0 \text{ and } f_{n+1}, \dots f_{n+\nu}.$$

Two main possibilities:

zero padding:

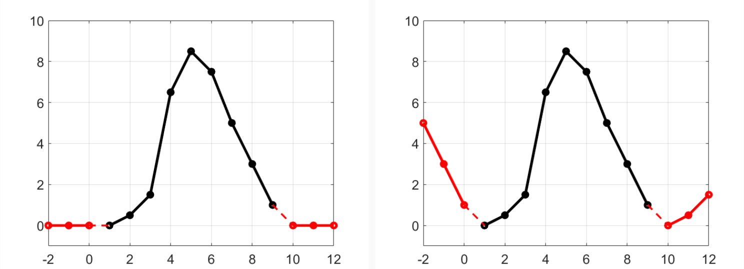
$$f_{-\nu+1} = \dots = f_0 = 0;$$

$$f_{n+1} = \dots = f_{n+\nu} = 0;$$

periodic extension:

$$f_{-\nu+1} = f_{n-\nu}, \dots, f_0 = f_n;$$

$$f_{n+1} = f_1, \dots, f_{n+\nu} = f_\nu;$$



Matrix representation

Matrix - vector product

Claim

The convolution of the PSF p with a vector in $f \in \mathbb{R}^n$ can be represented as the product between a suitable matrix A (depending on p) and the vector f.

Claim

The convolution of the PSF p with a vector in $f \in \mathbb{R}^n$ can be represented as the product between a suitable matrix A (depending on p) and the vector f.

Why?

Rigorous argument:

The convolution with p is a linear operator from $\mathbb{R}^n$ to $\mathbb{R}^n$ (indeed, $p * (f + g) = p * f + p * g$ and $p * (\lambda f) = \lambda p * f$). By the matrix representation theorem, there exists a matrix $A \in \mathbb{R}^{n \times n}$ such that $p * f = Af$ for all $f \in \mathbb{R}^n$.

Matrix - vector product

Claim

The convolution of the PSF p with a vector in $f \in \mathbb{R}^n$ can be represented as the product between a suitable matrix A (depending on p) and the vector f.

Why?

Rigorous argument:

The convolution with p is a linear operator from $\mathbb{R}^n$ to $\mathbb{R}^n$ (indeed, $p * (f + g) = p * f + p * g$ and $p * (\lambda f) = \lambda p * f$). By the matrix representation theorem, there exists a matrix $A \in \mathbb{R}^{n \times n}$ such that $p * f = Af$ for all $f \in \mathbb{R}^n$.

Intuitive argument:

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\ A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} \\ A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} \\ A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

Sizes of the problem

- $f \in \mathbb{R}^n$

f_1	f_2	f_3	f_4	f_5	f_6

Sizes of the problem

- $f \in \mathbb{R}^n$

f_1	f_2	f_3	f_4	f_5	f_6

- $p \in \mathbb{R}^m$, $m = 2\nu + 1$

p_2	p_1	p_0	p_{-1}	p_{-2}

Sizes of the problem

- $f \in \mathbb{R}^n$

f_1	f_2	f_3	f_4	f_5	f_6

- $p \in \mathbb{R}^m$, $m = 2\nu + 1$

p_2	p_1	p_0	p_{-1}	p_{-2}

- extension of f in $\mathbb{R}^{n+2\nu}$

f_{-1}	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8

Sizes of the problem

- $f \in \mathbb{R}^n$

$$\begin{array}{|c|c|c|c|c|c|} \hline f_1 & f_2 & f_3 & f_4 & f_5 & f_6 \\ \hline \end{array}$$

- $p \in \mathbb{R}^m$, $m = 2\nu + 1$

$$\begin{array}{|c|c|c|c|c|} \hline p_2 & p_1 & p_0 & p_{-1} & p_{-2} \\ \hline \end{array}$$

- extension of f in $\mathbb{R}^{n+2\nu}$

$$\begin{array}{|c|c|c|c|c|c|c|c|c|c|} \hline f_{-1} & f_0 & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & f_8 \\ \hline \end{array}$$

- $c = p * f \in \mathbb{R}^n$

$$\begin{array}{|c|c|c|c|c|c|} \hline c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \\ \hline \end{array}$$

Sizes of the problem

- $f \in \mathbb{R}^n$

$$\begin{array}{|c|c|c|c|c|c|}\hline f_1 & f_2 & f_3 & f_4 & f_5 & f_6 \\ \hline\end{array}$$

- $p \in \mathbb{R}^m$, $m = 2\nu + 1$

$$\begin{array}{|c|c|c|c|c|}\hline p_2 & p_1 & p_0 & p_{-1} & p_{-2} \\ \hline\end{array}$$

- extension of f in $\mathbb{R}^{n+2\nu}$

$$\begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline f_{-1} & f_0 & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & f_8 \\ \hline\end{array}$$

- $c = p * f \in \mathbb{R}^n$

$$\begin{array}{|c|c|c|c|c|c|}\hline c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \\ \hline\end{array}$$

- $A \in \mathbb{R}^{n \times n}$

Sizes of the problem

- $f \in \mathbb{R}^n$

f_1	f_2	f_3	f_4	f_5	f_6

- $p \in \mathbb{R}^m$, $m = 2\nu + 1$

p_2	p_1	p_0	p_{-1}	p_{-2}

- extension of f in $\mathbb{R}^{n+2\nu}$

f_{-1}	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8

- $c = p * f \in \mathbb{R}^n$

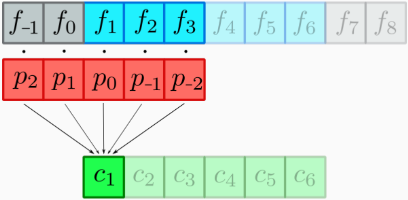
c_1	c_2	c_3	c_4	c_5	c_6

- $A \in \mathbb{R}^{n \times n}$

Remember the formulas:

$$(Af)_i = \sum_{j=1}^{n} A_{i,j} f_j; \qquad (p * f)_i = \sum_{\ell=-\nu}^{\nu} p_\ell f_{i-\ell}$$

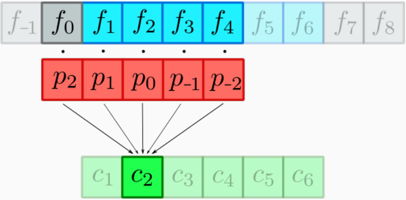
Matrix: zero padding case



$$
\begin{aligned}
c_1 &= p_2 f_{-1} + p_1 f_0 + p_0 f_1 + p_{-1} f_2 + p_{-2} f_3 \\
&= p_2 0 + p_1 0 + p_0 f_1 + p_{-1} f_2 + p_{-2} f_3 \\
&= p_0 f_1 + p_{-1} f_2 + p_{-2} f_3
\end{aligned}
$$

$$
\begin{bmatrix} p_0 & p_{-1} & p_{-2} & 0 & 0 & 0 \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \end{bmatrix}
\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix}
=
\begin{bmatrix} c_1 \\ \\ \\ \\ \\ \\ \end{bmatrix}
$$

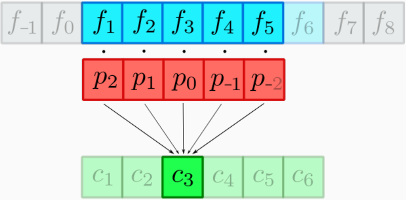
Matrix: zero padding case



$$
\begin{aligned}
c_2 &= p_2 f_0 + p_1 f_1 + p_0 f_2 + p_{-1} f_3 + p_{-2} f_4 \\
&= p_2 0 + p_1 f_1 + p_0 f_2 + p_{-1} f_3 + p_{-2} f_4 \\
&= p_1 f_1 + p_0 f_2 + p_{-1} f_3 + p_{-2} f_4
\end{aligned}
$$

$$
\begin{bmatrix}
p_0 & p_{-1} & p_{-2} & 0 & 0 & 0 \\
p_1 & p_0 & p_{-1} & p_{-2} & 0 & 0 \\
 & & & & & \\
 & & & & & \\
 & & & & & \\
 & & & & &
\end{bmatrix}
\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix}
=
\begin{bmatrix} c_1 \\ c_2 \\ \\ \\ \\ \\ \end{bmatrix}
$$

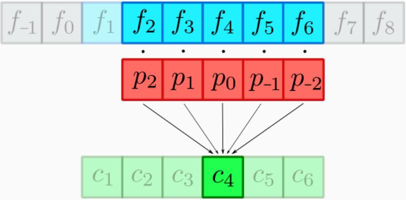
Matrix: zero padding case



$$c_3 = p_2 f_1 + p_1 f_2 + p_0 f_3 + p_{-1} f_4 + p_{-2} f_5$$

$$\begin{bmatrix} p_0 & p_{-1} & p_{-2} & 0 & 0 & 0 \\ p_1 & p_0 & p_{-1} & p_{-2} & 0 & 0 \\ p_2 & p_1 & p_0 & p_{-1} & p_{-2} & 0 \\ & & & & & \\ & & & & & \\ & & & & & \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \\ \\ \end{bmatrix}$$

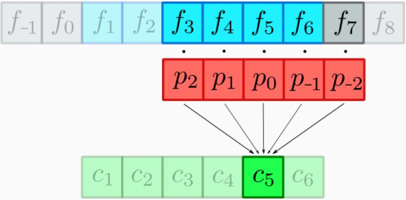
Matrix: zero padding case



$$c_4 = p_2 f_2 + p_1 f_3 + p_0 f_4 + p_{-1} f_5 + p_{-2} f_6$$

$$\begin{bmatrix} p_0 & p_{-1} & p_{-2} & 0 & 0 & 0 \\ p_1 & p_0 & p_{-1} & p_{-2} & 0 & 0 \\ p_2 & p_1 & p_0 & p_{-1} & p_{-2} & 0 \\ 0 & p_2 & p_1 & p_0 & p_{-1} & p_{-2} \\ & & & & & \\ & & & & & \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ \\ \end{bmatrix}$$

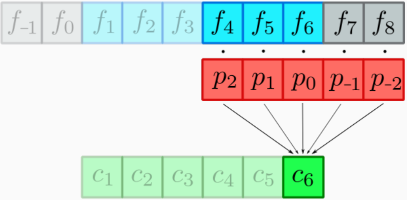
Matrix: zero padding case



$$\begin{aligned} c_5 &= p_2 f_3 + p_1 f_4 + p_0 f_5 + p_{-1} f_6 + p_{-2} f_7 \\ &= p_2 f_3 + p_1 f_4 + p_0 f_5 + p_{-1} f_6 \end{aligned}$$

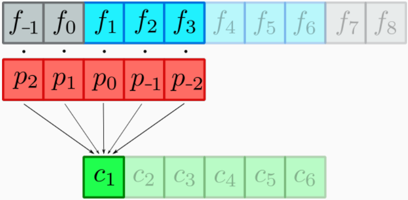
$$\begin{bmatrix} p_0 & p_{-1} & p_{-2} & 0 & 0 & 0 \\ p_1 & p_0 & p_{-1} & p_{-2} & 0 & 0 \\ p_2 & p_1 & p_0 & p_{-1} & p_{-2} & 0 \\ 0 & p_2 & p_1 & p_0 & p_{-1} & p_{-2} \\ 0 & 0 & p_2 & p_1 & p_0 & p_{-1} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix}$$

Matrix: zero padding case



$$\begin{aligned} c_6 &= p_2 f_4 + p_1 f_5 + p_0 f_6 + p_{-1} f_7 + p_{-2} f_8 \\ &= p_2 f_4 + p_1 f_5 + p_0 f_6 \end{aligned}$$

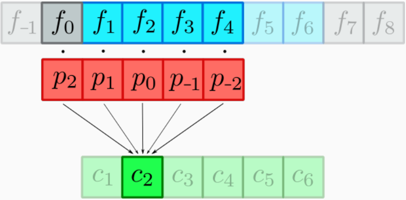
$$\begin{bmatrix} p_0 & p_{-1} & p_{-2} & 0 & 0 & 0 \\ p_1 & p_0 & p_{-1} & p_{-2} & 0 & 0 \\ p_2 & p_1 & p_0 & p_{-1} & p_{-2} & 0 \\ 0 & p_2 & p_1 & p_0 & p_{-1} & p_{-2} \\ 0 & 0 & p_2 & p_1 & p_0 & p_{-1} \\ 0 & 0 & 0 & p_2 & p_1 & p_0 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{bmatrix}$$



$$\begin{aligned} c_1 &= p_2 f_{-1} + p_1 f_0 + p_0 f_1 + p_{-1} f_2 + p_{-2} f_3 \\ &= p_2 f_5 + p_1 f_6 + p_0 f_1 + p_{-1} f_2 + p_{-2} f_3 \end{aligned}$$

$$\begin{bmatrix} p_0 & p_{-1} & p_{-2} & 0 & p_2 & p_1 \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix} = \begin{bmatrix} c_1 \\ \\ \\ \\ \\ \\ \end{bmatrix}$$

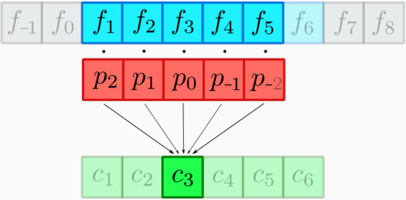
Matrix: periodic extension



$$\begin{aligned} c_2 &= p_2 f_0 + p_1 f_1 + p_0 f_2 + p_{-1} f_3 + p_{-2} f_4 \\ &= p_2 f_6 + p_1 f_1 + p_0 f_2 + p_{-1} f_3 + p_{-2} f_4 \end{aligned}$$

$$\begin{bmatrix} p_0 & p_{-1} & p_{-2} & 0 & p_2 & p_1 \\ p_1 & p_0 & p_{-1} & p_{-2} & 0 & p_2 \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \\ \\ \\ \\ \end{bmatrix}$$

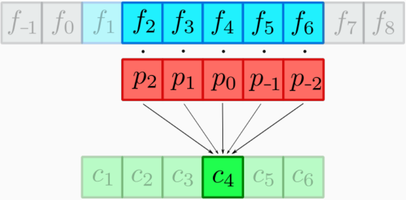
Matrix: periodic extension



$$c_3 = p_2 f_1 + p_1 f_2 + p_0 f_3 + p_{-1} f_4 + p_{-2} f_5$$

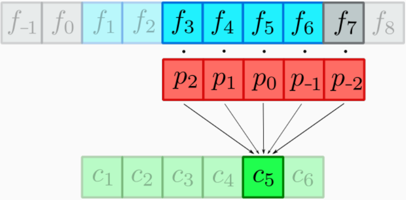
$$\begin{bmatrix} p_0 & p_{-1} & p_{-2} & 0 & p_2 & p_1 \\ p_1 & p_0 & p_{-1} & p_{-2} & 0 & p_1 \\ p_2 & p_1 & p_0 & p_{-1} & p_{-2} & 0 \\ & & & & & \\ & & & & & \\ & & & & & \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \\ \\ \end{bmatrix}$$

Matrix: periodic extension



$$c_4 = p_2 f_2 + p_1 f_3 + p_0 f_4 + p_{-1} f_5 + p_{-2} f_6$$

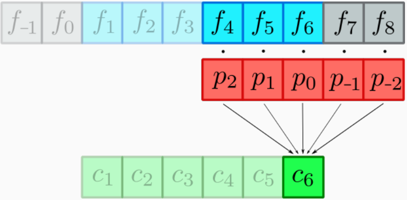
$$\begin{bmatrix} p_0 & p_{-1} & p_{-2} & 0 & p_2 & p_1 \\ p_1 & p_0 & p_{-1} & p_{-2} & 0 & p_1 \\ p_2 & p_1 & p_0 & p_{-1} & p_{-2} & 0 \\ 0 & p_2 & p_1 & p_0 & p_{-1} & p_{-2} \\ & & & & & \\ & & & & & \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ \\ \end{bmatrix}$$



$$\begin{aligned} c_5 &= p_2 f_3 + p_1 f_4 + p_0 f_5 + p_{-1} f_6 + p_{-2} f_7 \\ &= p_2 f_3 + p_1 f_4 + p_0 f_5 + p_{-1} f_6 + p_{-2} f_1 \end{aligned}$$

$$\begin{bmatrix} p_0 & p_{-1} & p_{-2} & 0 & p_2 & p_1 \\ p_1 & p_0 & p_{-1} & p_{-2} & 0 & p_1 \\ p_2 & p_1 & p_0 & p_{-1} & p_{-2} & 0 \\ 0 & p_2 & p_1 & p_0 & p_{-1} & p_{-2} \\ p_{-2} & 0 & p_2 & p_1 & p_0 & p_{-1} \\ & & & & & \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ \\ \end{bmatrix}$$

Matrix: periodic extension



$$\begin{aligned} c_6 &= p_2 f_4 + p_1 f_5 + p_0 f_6 + p_{-1} f_7 + p_{-2} f_8 \\ &= p_2 f_4 + p_1 f_5 + p_0 f_6 + p_{-1} f_1 + p_{-2} f_2 \end{aligned}$$

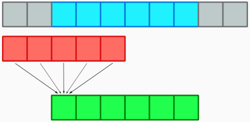
$$\begin{bmatrix} p_0 & p_{-1} & p_{-2} & 0 & p_2 & p_1 \\ p_1 & p_0 & p_{-1} & p_{-2} & 0 & p_2 \\ p_2 & p_1 & p_0 & p_{-1} & p_{-2} & 0 \\ 0 & p_2 & p_1 & p_0 & p_{-1} & p_{-2} \\ p_{-2} & 0 & p_2 & p_1 & p_0 & p_{-1} \\ p_{-1} & p_{-2} & 0 & p_2 & p_1 & p_0 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{bmatrix}$$

In Matlab® it is possible to compute the convolution of a PSF $p \in \mathbb{R}^m$ and a signal $f \in \mathbb{R}^n$ with the following commands:

Implementation

In Matlab® it is possible to compute the convolution of a PSF $p \in \mathbb{R}^m$ and a signal $f \in \mathbb{R}^n$ with the following commands:

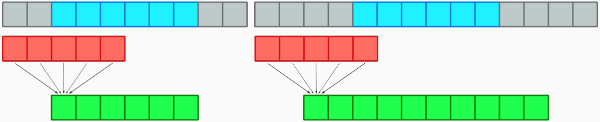
- conv:
  `c1 = conv(f,p,'same')` returns a vector of size $n \times 1$ obtained by zero padding of f;
  `c2 = conv(f,p,'full')` returns a vector of size $(n+m-1) \times 1$ obtained by a larger zero padding of f; (this is the default option)
  `c3 = conv(f,p,'valid')` returns a vector of size $(n-m+1) \times 1$ obtained without extending f;



Implementation

In Matlab® it is possible to compute the convolution of a PSF $p \in \mathbb{R}^m$ and a signal $f \in \mathbb{R}^n$ with the following commands:

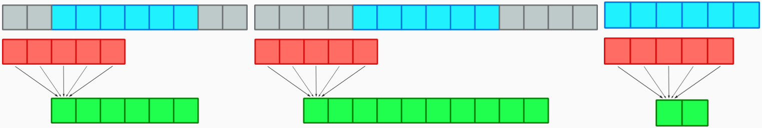
- conv:
  `c1 = conv(f,p,'same')` returns a vector of size $n \times 1$ obtained by zero padding of f;
  `c2 = conv(f,p,'full')` returns a vector of size $(n+m-1) \times 1$ obtained by a larger zero padding of f; (this is the default option)
  `c3 = conv(f,p,'valid')` returns a vector of size $(n-m+1) \times 1$ obtained without extending f;



Implementation

In Matlab® it is possible to compute the convolution of a PSF $p \in \mathbb{R}^m$ and a signal $f \in \mathbb{R}^n$ with the following commands:

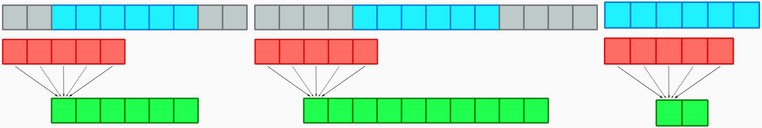
- conv:
  `c1 = conv(f,p,'same')` returns a vector of size $n \times 1$ obtained by zero padding of f;
  `c2 = conv(f,p,'full')` returns a vector of size $(n+m-1) \times 1$ obtained by a larger zero padding of f; (this is the default option)
  `c3 = conv(f,p,'valid')` returns a vector of size $(n-m+1) \times 1$ obtained without extending f;



Implementation

In Matlab® it is possible to compute the convolution of a PSF $p \in \mathbb{R}^m$ and a signal $f \in \mathbb{R}^n$ with the following commands:

- conv:
  `c1 = conv(f,p,'same')` returns a vector of size $n \times 1$ obtained by zero padding of f;
  `c2 = conv(f,p,'full')` returns a vector of size $(n+m-1) \times 1$ obtained by a larger zero padding of f; (this is the default option)
  `c3 = conv(f,p,'valid')` returns a vector of size $(n-m+1) \times 1$ obtained without extending f;



- convmtx: `A = convmtx(p,n)` returns a matrix of size $(n+m-1) \times 1$ such that `A*f = c2`, the full vector. To obtain `c1` one must extract `A = A(1+nu:n+nu,:)`.

Exercise

Example 1

1. define a vector $f \in \mathbb{R}^{100}$ such that $f_i = 1$ if $i \in \{25, \dots, 75\}$, $f_i = 0$ otherwise. Plot the signal f;
2. define a point spread function $p \in \mathbb{R}^5$ obtained by normalizing the vector $[1, 2, 3, 2, 1]'$;
3. compute $p * f$ with zero padding by the command `conv`;
4. compute $p * f$ with zero padding by the command `convmtx`;
5. compare the results of the previous two points with the original signal.
6. (Extra): create the convolution matrix by hand and compare it with the one obtained by `convmtx`. *(The following commands can be useful:* `diag` *and* `spy`*)*

Inverse problem

Formulation of the inverse problem

The convolution between a signal f and a point spread function p can be written by means of a matrix A (whose expression depends on p)

Formulation of the inverse problem

The convolution between a signal f and a point spread function p can be written by means of a matrix A (whose expression depends on p)

Convolution

$$c = Af; \qquad f \in \mathbb{R}^n, c \in \mathbb{R}^n, A \in \mathbb{R}^{n \times n}$$

Formulation of the inverse problem

The convolution between a signal f and a point spread function p can be written by means of a matrix A (whose expression depends on p)

Convolution

$$c = Af; \qquad f \in \mathbb{R}^n, c \in \mathbb{R}^n, A \in \mathbb{R}^{n \times n}$$

The **inverse problem** consists in trying to do the opposite: suppose you only have information on $m = p * f$, can you find f?

Deconvolution

$$\text{given } m = p * f = Af \in \mathbb{R}^n, \text{ find } f \in \mathbb{R}^n$$

Since the mapping from f to m is linear, this is called a **linear inverse problem**.

Formulation of the inverse problem

The convolution between a signal f and a point spread function p can be written by means of a matrix A (whose expression depends on p)

Convolution

$$c = Af; \qquad f \in \mathbb{R}^n, c \in \mathbb{R}^n, A \in \mathbb{R}^{n\times n}$$

The **inverse problem** consists in trying to do the opposite: suppose you only have information on $m = p * f$, can you find f?

Deconvolution

$$\text{given } m = p * f = Af \in \mathbb{R}^n, \text{ find } f \in \mathbb{R}^n$$

Since the mapping from f to m is linear, this is called a **linear inverse problem**. Main questions:

1. is this problem meaningful?
2. how can we solve it? (spoiler: A^{-1})

Is this problem meaningful?

J. Hadamard introduced the following criteria a problem must satisfy in order to be defined as well-posed:

Is this problem meaningful?

J. Hadamard introduced the following criteria a problem must satisfy in order to be defined as well-posed:

a) existence of a solution: $\forall m \in \mathbb{R}^n, \; \exists f \in \mathbb{R}^n : Af = m$

Is this problem meaningful?

J. Hadamard introduced the following criteria a problem must satisfy in order to be defined as well-posed:

a) existence of a solution: $\forall m \in \mathbb{R}^n,\ \exists f \in \mathbb{R}^n : Af = m$
b) uniqueness of the solution: $\forall f, g \in \mathbb{R}^n,\ Af = Ag = m \Rightarrow f = g$

Is this problem meaningful?

J. Hadamard introduced the following criteria a problem must satisfy in order to be defined as well-posed:

a) existence of a solution: $\forall m \in \mathbb{R}^n,\ \exists f \in \mathbb{R}^n : Af = m$
b) uniqueness of the solution: $\forall f, g \in \mathbb{R}^n,\ Af = Ag = m \Rightarrow f = g$
c) stability of the solution: consider $\varepsilon \in \mathbb{R}^n$ s.t. $\|\varepsilon\| \leq \delta$, $m_\delta = m + \varepsilon$ and let f_δ be the solution associated to m_δ. Then $f_\delta \to f$ as $\delta \to 0$.

Is this problem meaningful?

J. Hadamard introduced the following criteria a problem must satisfy in order to be defined as well-posed:

a) existence of a solution: $\forall m \in \mathbb{R}^n,\ \exists f \in \mathbb{R}^n : Af = m$
b) uniqueness of the solution: $\forall f, g \in \mathbb{R}^n,\ Af = Ag = m \Rightarrow f = g$
c) stability of the solution: consider $\varepsilon \in \mathbb{R}^n$ s.t. $\|\varepsilon\| \leq \delta$, $m_\delta = m + \varepsilon$ and let f_δ be the solution associated to m_δ. Then $f_\delta \to f$ as $\delta \to 0$.

A stronger version of stability (although less formal) is known as well-conditioning: *small perturbations on the datum m cause small perturbations on the solution f.*

Is this problem meaningful?

J. Hadamard introduced the following criteria a problem must satisfy in order to be defined as well-posed:

a) existence of a solution: $\forall m \in \mathbb{R}^n,\ \exists f \in \mathbb{R}^n : Af = m$
b) uniqueness of the solution: $\forall f, g \in \mathbb{R}^n,\ Af = Ag = m \Rightarrow f = g$
c) stability of the solution: consider $\varepsilon \in \mathbb{R}^n$ s.t. $\|\varepsilon\| \leq \delta$, $m_\delta = m + \varepsilon$ and let f_δ be the solution associated to m_δ. Then $f_\delta \to f$ as $\delta \to 0$.

A stronger version of stability (although less formal) is known as well-conditioning: *small perturbations on the datum m cause small perturbations on the solution f.*

Warning: real-world measurements are always some perturbation m_δ of the expected exact measurements m. Stability matters!

Is this problem meaningful?

J. Hadamard introduced the following criteria a problem must satisfy in order to be defined as well-posed:

a) existence of a solution: $\forall m \in \mathbb{R}^n,\ \exists f \in \mathbb{R}^n : Af = m$
b) uniqueness of the solution: $\forall f, g \in \mathbb{R}^n,\ Af = Ag = m \Rightarrow f = g$
c) stability of the solution: consider $\varepsilon \in \mathbb{R}^n$ s.t. $\|\varepsilon\| \leq \delta$, $m_\delta = m + \varepsilon$ and let f_δ be the solution associated to m_δ. Then $f_\delta \to f$ as $\delta \to 0$.

A stronger version of stability (although less formal) is known as well-conditioning: *small perturbations on the datum m cause small perturbations on the solution f.*

Warning: real-world measurements are always some perturbation m_δ of the expected exact measurements m. Stability matters!

The problem of approximating f from $m_\delta = Af + \varepsilon$ is usually referred to as **deconvolution and denoising**.

Excursus: real data and synthetic data

In real-world inverse problems, the vector m comes from **real measurements**. This could be caused by several issues:

- m could be affected by measurement noise;

Excursus: real data and synthetic data

In real-world inverse problems, the vector m comes from **real measurements**. This could be caused by several issues:

- m could be affected by measurement noise;
- even disregarding the noise, the model linking f to m could be different from the one that we expect (e.g. the point spread function is slightly different, there is something more than convolution...)

Excursus: real data and synthetic data

In real-world inverse problems, the vector m comes from **real measurements**. This could be caused by several issues:

- m could be affected by measurement noise;
- even disregarding the noise, the model linking f to m could be different from the one that we expect (e.g. the point spread function is slightly different, there is something more than convolution...)
- the original model is a continuous one. Discretization can cause an additional approximation.

Excursus: real data and synthetic data

In real-world inverse problems, the vector m comes from **real measurements**. This could be caused by several issues:

- m could be affected by measurement noise;
- even disregarding the noise, the model linking f to m could be different from the one that we expect (e.g. the point spread function is slightly different, there is something more than convolution...)
- the original model is a continuous one. Discretization can cause an additional approximation.

In our case studies (and also in many applications) we do not use real measurements, but **synthetic data**: we suppose to know the exact solution f and we generate the measurement m from it. Then, we forget about f and aim at reconstructing it by m.

Excursus: real data and synthetic data

In real-world inverse problems, the vector m comes from **real measurements**. This could be caused by several issues:

- m could be affected by measurement noise;
- even disregarding the noise, the model linking f to m could be different from the one that we expect (e.g. the point spread function is slightly different, there is something more than convolution...)
- the original model is a continuous one. Discretization can cause an additional approximation.

In our case studies (and also in many applications) we do not use real measurements, but **synthetic data**: we suppose to know the exact solution f and we generate the measurement m from it. Then, we forget about f and aim at reconstructing it by m.

Is this fair and helpful in real-world applications?

Excursus: real data and synthetic data

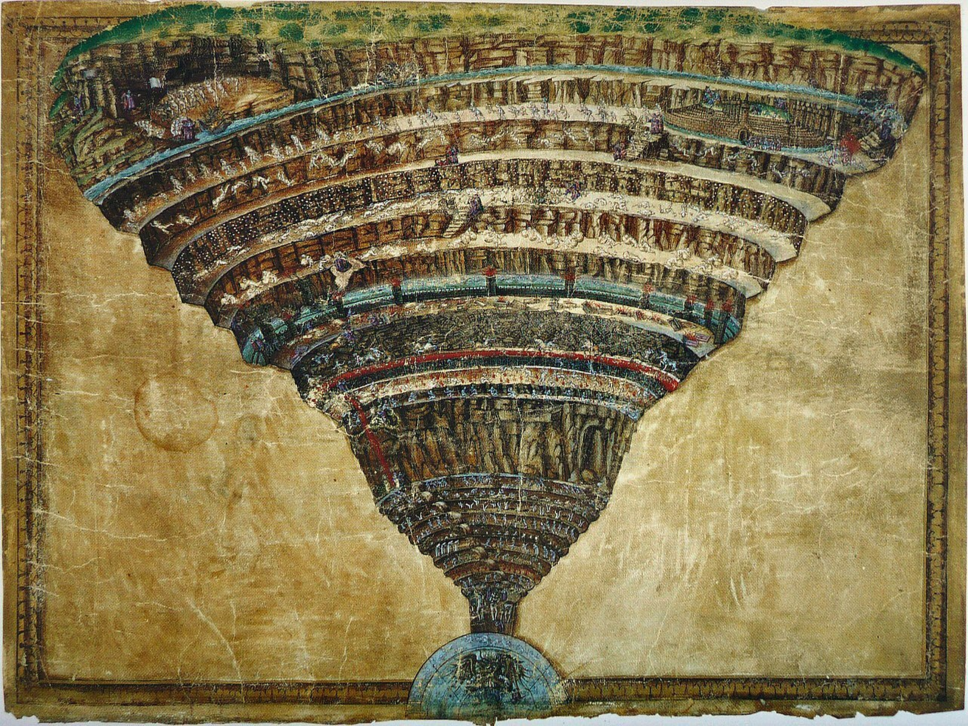
In real-world inverse problems, the vector m comes from **real measurements**. This could be caused by several issues:

- m could be affected by measurement noise;
- even disregarding the noise, the model linking f to m could be different from the one that we expect (e.g. the point spread function is slightly different, there is something more than convolution...)
- the original model is a continuous one. Discretization can cause an additional approximation.

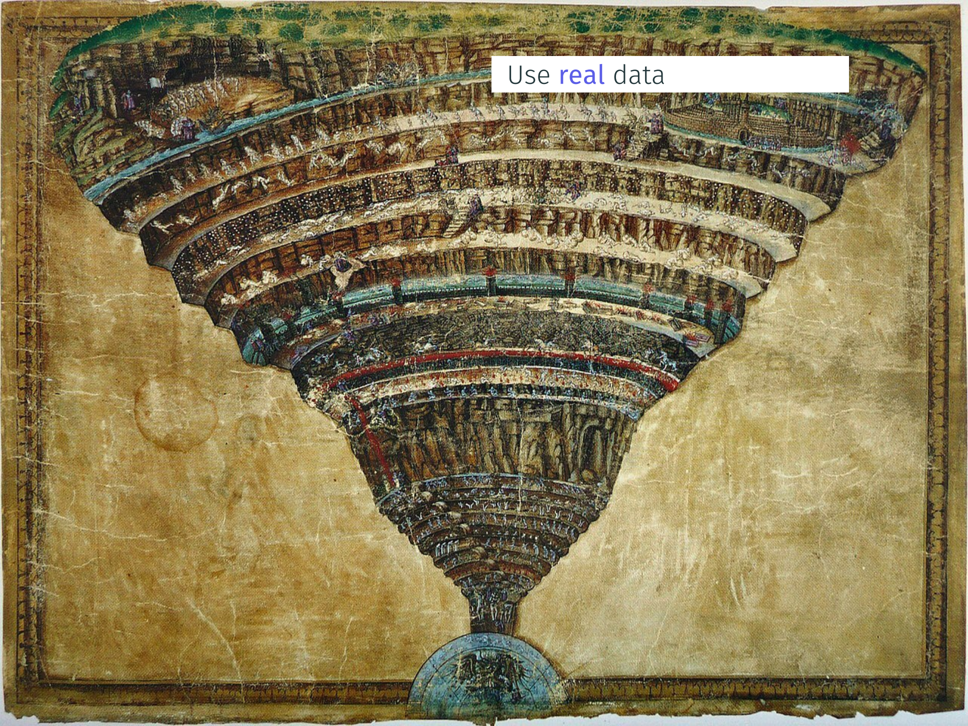
In our case studies (and also in many applications) we do not use real measurements, but **synthetic data**: we suppose to know the exact solution f and we generate the measurement m from it. Then, we forget about f and aim at reconstructing it by m.

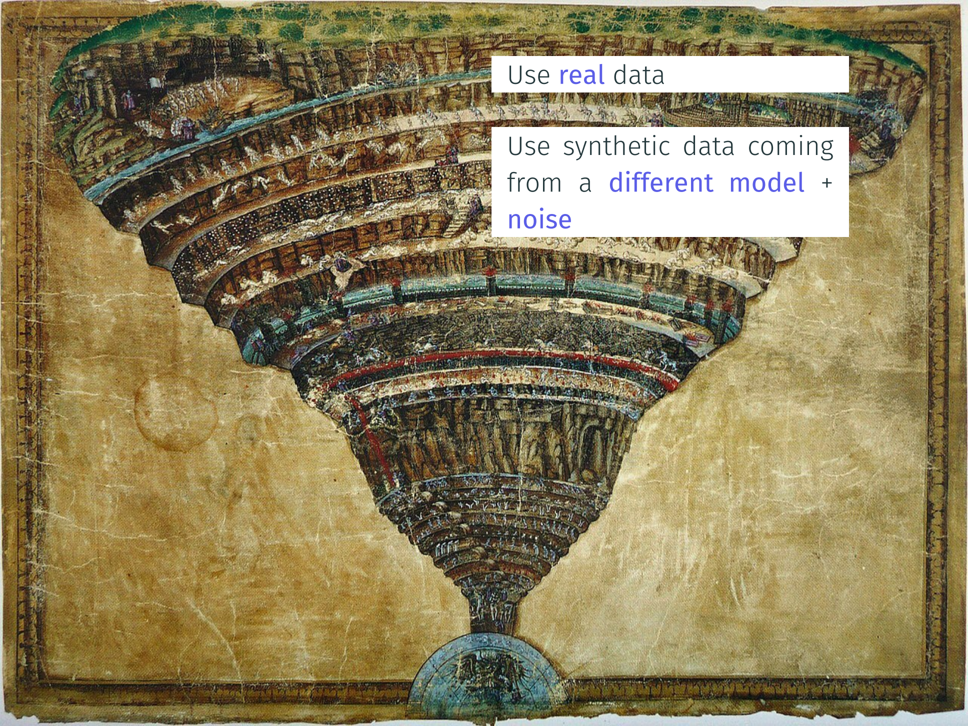
Is this fair and helpful in real-world applications?

Inverse Crimes



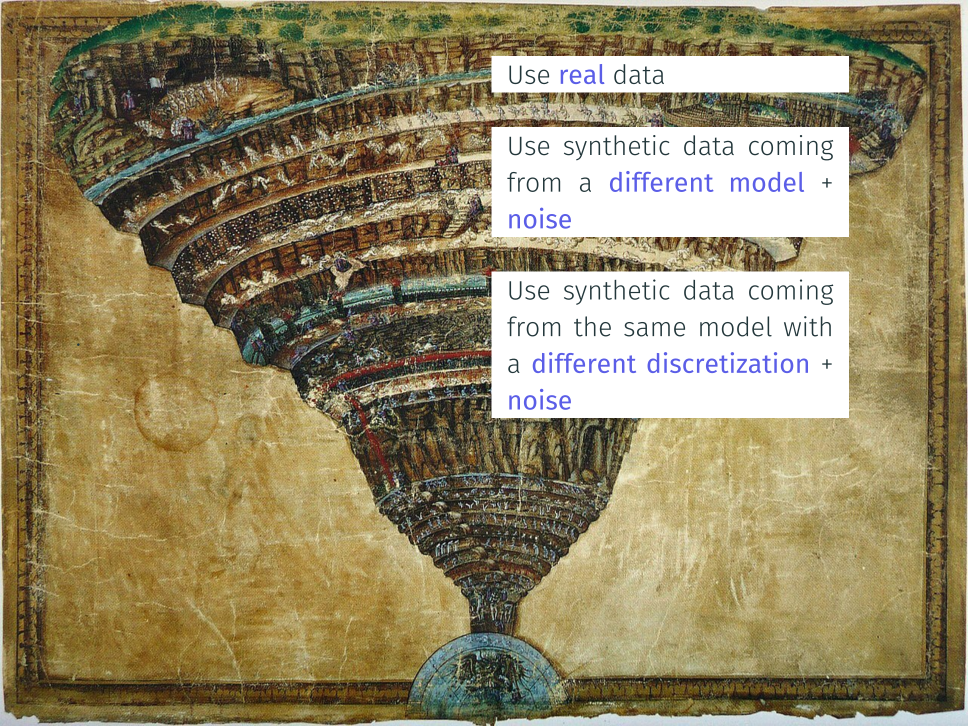
Use real data





Use real data

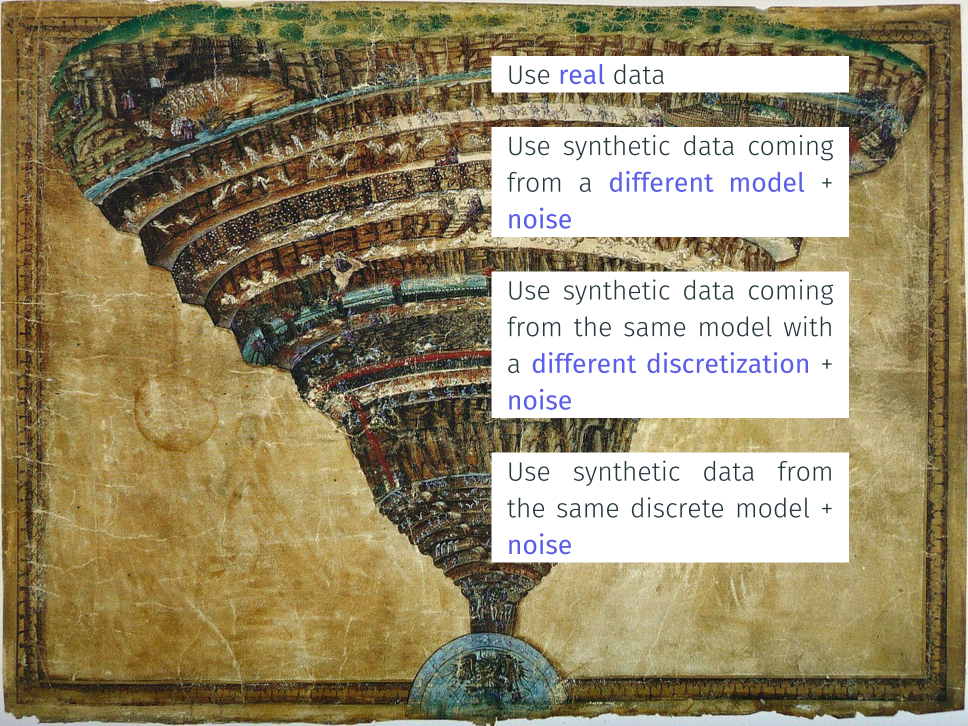
Use synthetic data coming from a different model + noise



Use real data

Use synthetic data coming from a different model + noise

Use synthetic data coming from the same model with a different discretization + noise

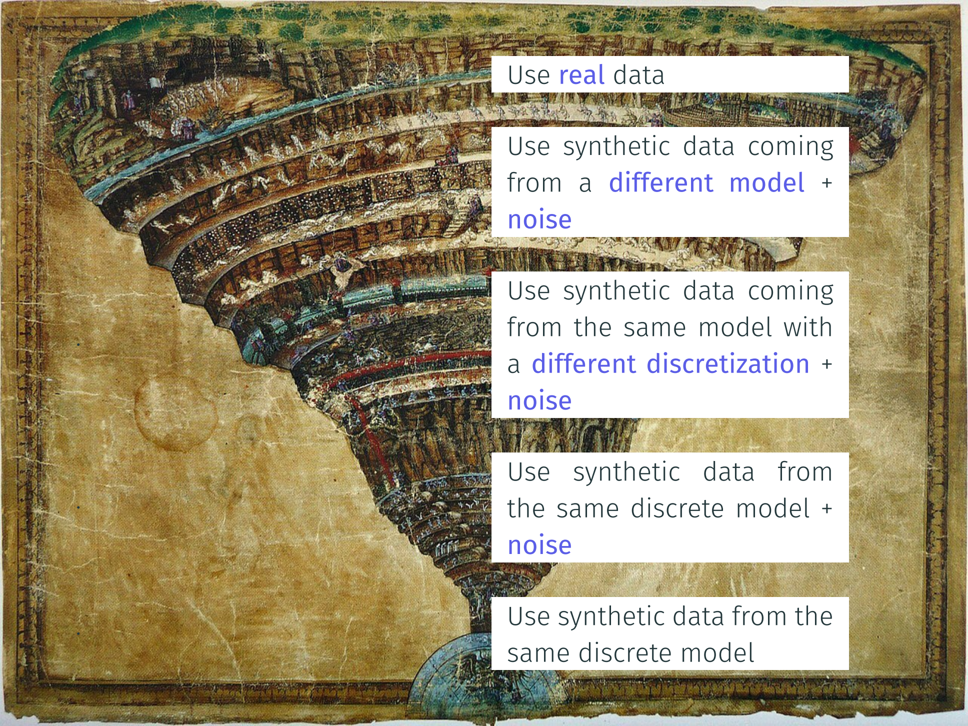


Use real data

Use synthetic data coming from a different model + noise

Use synthetic data coming from the same model with a different discretization + noise

Use synthetic data from the same discrete model + noise



Use real data

Use synthetic data coming from a different model + noise

Use synthetic data coming from the same model with a different discretization + noise

Use synthetic data from the same discrete model + noise

Use synthetic data from the same discrete model

Naïve inversion

How do we solve the Deconvolution problem?

Recall the deconvolution problem: given $m = Af$, find f.

What is the easiest idea to study and solve the problem?

How do we solve the Deconvolution problem?

Recall the deconvolution problem: given $m = Af$, find f.
What is the easiest idea to study and solve the problem?

Naïve inversion

Suppose the matrix A is invertible and compute A^{-1}. Then,

$$f = A^{-1}m$$

is the unique solution of the deconvolution problem.

How do we solve the Deconvolution problem?

Recall the **deconvolution** problem: given $m = Af$, find f.
What is the easiest idea to study and solve the problem?

Naïve inversion

Suppose the matrix A is invertible and compute A^{-1}. Then,

$$f = A^{-1}m$$

is the unique solution of the deconvolution problem.

Problem 1: what if A is not invertible? How can we tell that?
Problem 2: is this solution stable and well conditioned? Remember: the real world application is a combination of **deconvolution and denoising**, i.e., recover a good approximation of f from $m_\delta = Af + \varepsilon$, $\|\varepsilon\| \leq \delta$.

Is $\widetilde{f_\delta} := A^{-1}m_\delta$ a good approximation of f?

Matlab example

Example 2

1. Consider a signal f as in Exercise 1, and a point spread function obtained from $p = [1, 1, 1, 1, 1]$, convolved with itself and normalized.
2. Compute the convolution matrix A.
3. Generate the synthetic datum $m = Af$. Compute $A^{-1}m$ and compare it to f by plotting both and by computing the norm of their difference. *(use the command \ and not* `inv`*)*
4. Generate the synthetic datum $m_\delta = Af + \varepsilon$, being ε a Gaussian vector of null mean and standard deviation $\delta = 10^{-4}$. Compute $\widetilde{f_\delta} = A^{-1}m_\delta$ and compare it to f as above.
5. Repeat the previous point with $\delta = 0.01$.
6. Repeat the previous point considering a different point spread function obtained from $p = [1, 1, 1, 1, 1]$, convolved with itself three times and normalized.

Why does naïve inversion fail?

A little computation shows that

$$\widetilde{f_\delta} = A^{-1}m_\delta = A^{-1}Af + A^{-1}\varepsilon = f + A^{-1}\varepsilon$$

This implies that

$$\|\widetilde{f_\delta} - f\| \leq \|A^{-1}\|\|\varepsilon\| \leq \|A^{-1}\|\delta.$$

Why does naïve inversion fail?

A little computation shows that

$$\widetilde{f_\delta} = A^{-1}m_\delta = A^{-1}Af + A^{-1}\varepsilon = f + A^{-1}\varepsilon$$

This implies that

$$\|\widetilde{f_\delta} - f\| \leq \|A^{-1}\|\|\varepsilon\| \leq \|A^{-1}\|\delta.$$

- The noise ε is amplified by A^{-1}. In convolution matrices, $\|A^{-1}\|$ can easily reach huge values (in the previous exercise, as it can be computed by the command `norm(inv(A))`, it was $\approx 4 \cdot 10^{11}$.

Why does naïve inversion fail?

A little computation shows that

$$\widetilde{f}_\delta = A^{-1} m_\delta = A^{-1} A f + A^{-1} \varepsilon = f + A^{-1} \varepsilon$$

This implies that

$$\|\widetilde{f}_\delta - f\| \leq \|A^{-1}\| \|\varepsilon\| \leq \|A^{-1}\| \delta.$$

- The noise ε is amplified by A^{-1}. In convolution matrices, $\|A^{-1}\|$ can easily reach huge values (in the previous exercise, as it can be computed by the command `norm(inv(A))`, it was $\approx 4 \cdot 10^{11}$.
- (An extra detail: the term $A^{-1}Af$ is theoretically identical to f but, in the inexact arithmetic of the computer, small errors can occur during the operations, and they can be highly amplified by A^{-1})

Why does naïve inversion fail?

A little computation shows that

$$\widetilde{f_\delta} = A^{-1} m_\delta = A^{-1} A f + A^{-1} \varepsilon = f + A^{-1} \varepsilon$$

This implies that

$$\|\widetilde{f_\delta} - f\| \leq \|A^{-1}\| \|\varepsilon\| \leq \|A^{-1}\| \delta.$$

- The noise ε is amplified by A^{-1}. In convolution matrices, $\|A^{-1}\|$ can easily reach huge values (in the previous exercise, as it can be computed by the command `norm(inv(A))`, it was $\approx 4 \cdot 10^{11}$.
- (An extra detail: the term $A^{-1}Af$ is theoretically identical to f but, in the inexact arithmetic of the computer, small errors can occur during the operations, and they can be highly amplified by A^{-1})

The harsh reality

The deconvolution problem is ill-posed, or at least (in this discrete context) ill-conditioned. Thus we cannot just naïvely invert A.