Computational imaging & learning: from physics to data-driven methods

Luca Calatroni
MaLGa center, DIBRIS, UniGe
MMS, Istituto Italiano di Tecnologia
Genoa, Italy

Mini-Corso Data Science @ UniPD

Università degli studi di Padova

May 11-14 2026

Practical information

My academic ID & Contacts

Luca Calatroni:
PI of the **Computational Imaging & Learning** team, MaLGa, UniGe.

- **MSc in Applied Mathematics**, 2011.
- **PhD in Applied Mathematics**, University of Cambridge, 2015.
- **Marie-Curie Post-doc** University of Genova, 2015-2016.
- **Post-doctoral fellowship**, *Lecteur Hadamard*: CMAP, École Polytechnique, France, 2016-2019.
- **CNRS researcher**, I3S lab, Sophia-Antipolis, France, 2019-2024.
- **Professore Associato**, Computer Science department, Genoa, Italy.

Contacts

- **e-mail**: luca.calatroni@unige.it
- **web**: `https://sites.google.com/view/lucacalatroni/home`

Course objectives and organisation

Objectives: understanding basic notions on forward/inverse modelling of exemplar computational imaging applications in biomedical imaging.

Approach: theory (8h) & Python labs (4h), on your laptop using DeepInverse library.

Syllabus

- **May 11, 14:30-16:30**, Theory: Introduction, forward/inverse imaging problems, regularisation.
- **May 12, 8:30-10:30**, Theory: Regularisation concepts, some practical computational imaging problems.
- **May 12, 10:30-12:30**, Lab: Regularisation for CT imaging.
- **May 13, 14:30-16:30**, Theory: Optimisation tools for computational imaging.
- **May 13, 16:30-18:30**, Lab: Optimisation-driven learned regularisation for CT imaging.
- **May 14, 8:30-10:30**, Theory: Advanced optimisation-driven learning for image reconstruction.

Main references

Books

C. A. Bouman, *Foundations of Computational Imaging: A Model-Based Approach*, SIAM, 2022.

G. Aubert, P. Kornprobst, *Mathematical Problems in Image Processing: PDEs and the COV*, II edition, Springer, 2000.

Tony F. Chan, J. Shen, *Image Processing and Analysis: variational, PDE, wavelet and stochastic methods*, SIAM, 2005.

Blogs

Tristan van Leeuwen: https://tristanvanleeuwen.github.io/IP_and_Im_Lectures/what_is.html

Johannes Meyer: https://computational-imaging.de/book/intro.html

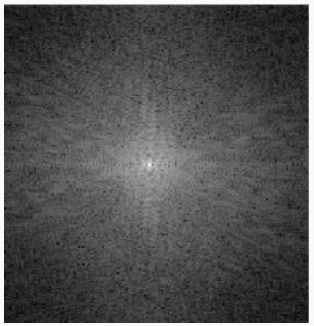
Computational resources

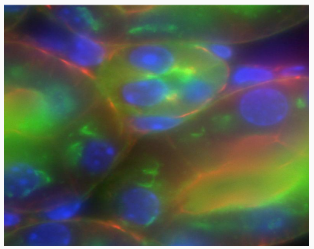
- Numerical tours by G. Peyré: http://www.numerical-tours.com/
- DeepInverse tutorials: https://deepinv.github.io/deepinv/auto_examples/index.html

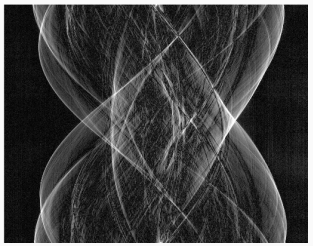
What is computational imaging ?

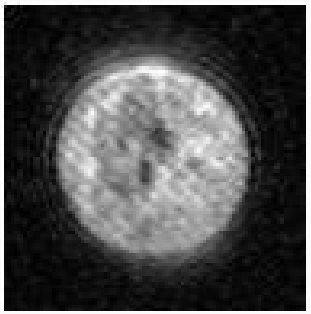
Microscopy

Medical Imaging





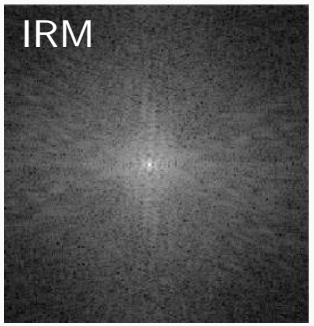


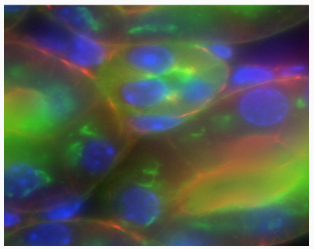


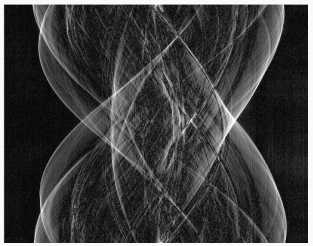
What is computational imaging ?

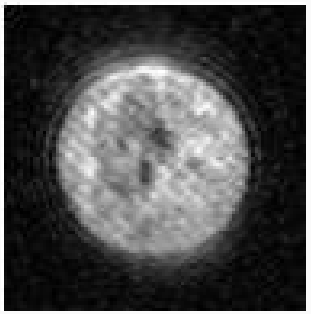
Microscopy

Medical Imaging





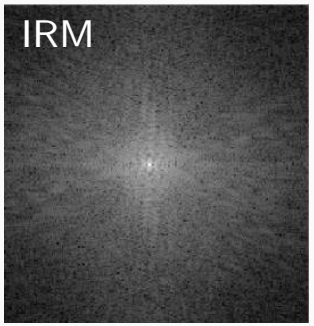


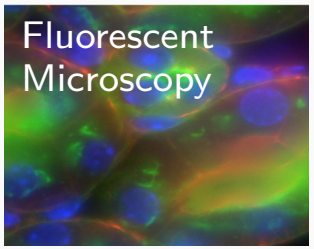


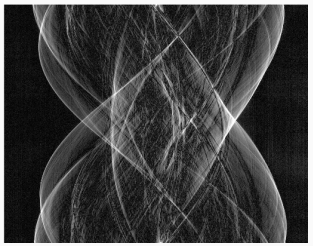
What is computational imaging ?

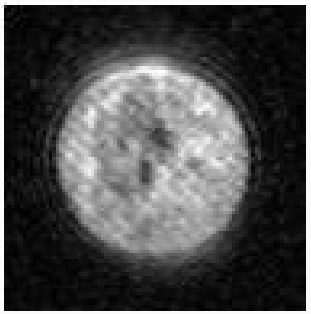
Microscopy

Medical Imaging





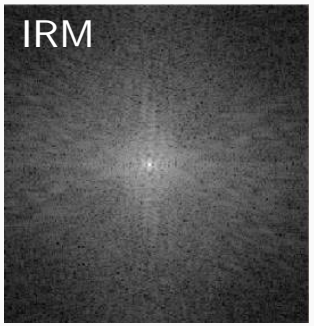


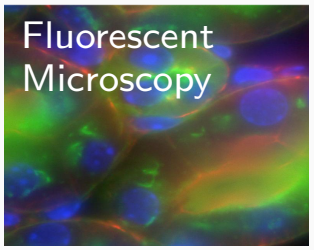


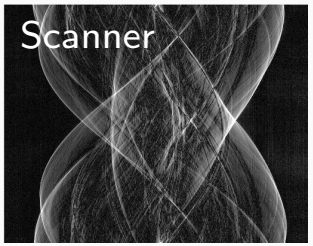
What is computational imaging ?

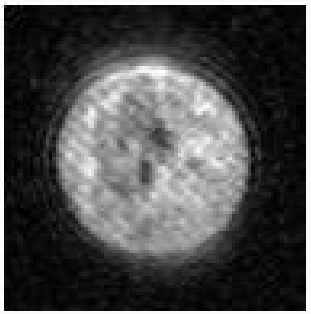
Microscopy

Medical Imaging





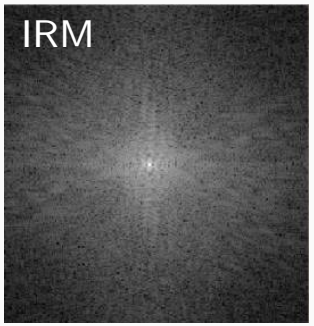


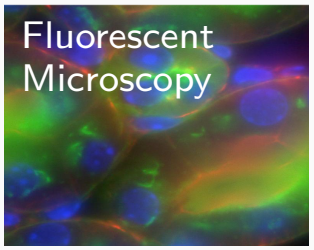


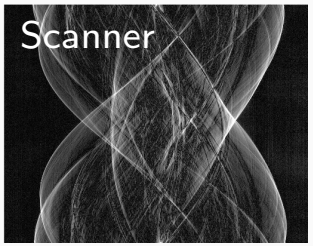
What is computational imaging ?

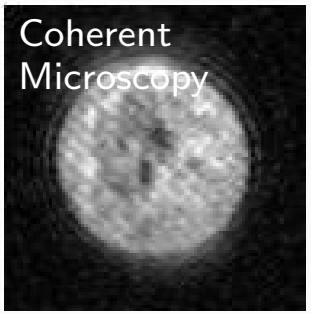
Microscopy

Medical Imaging





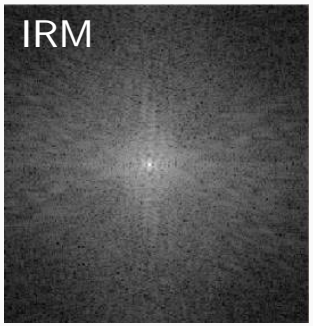


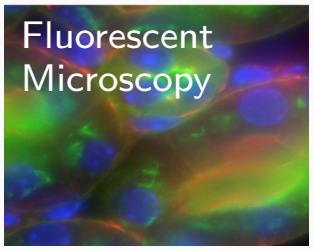


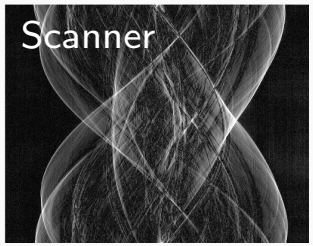
What is computational imaging ?

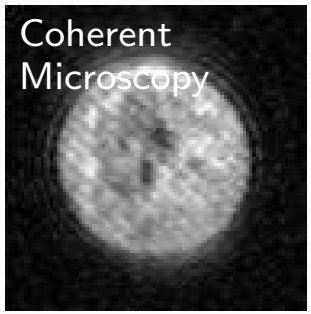
Microscopy

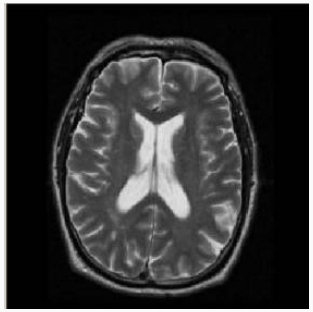
Medical Imaging







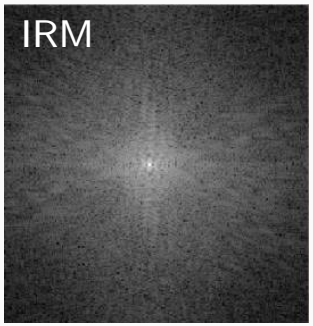


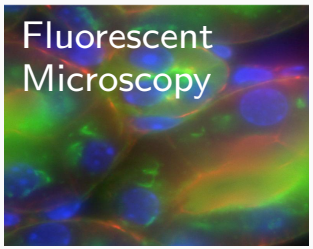


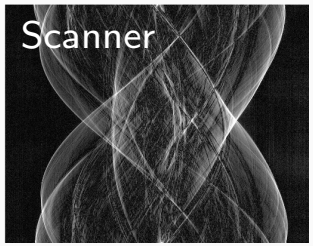
What is computational imaging ?

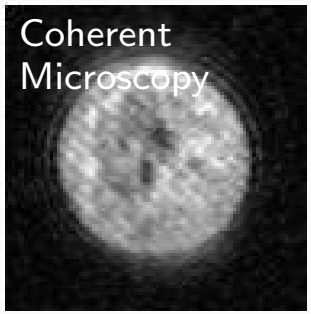
Microscopy

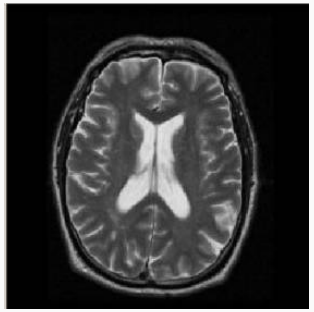
Medical Imaging

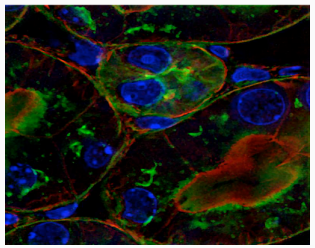








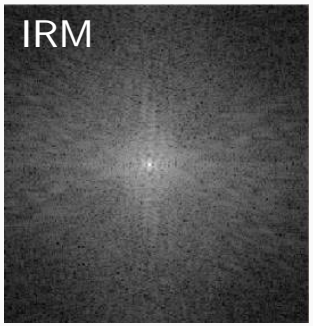


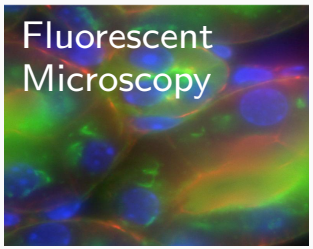


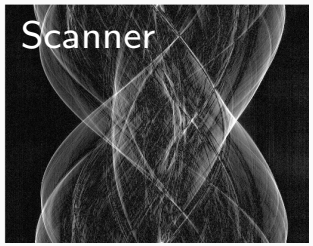
What is computational imaging ?

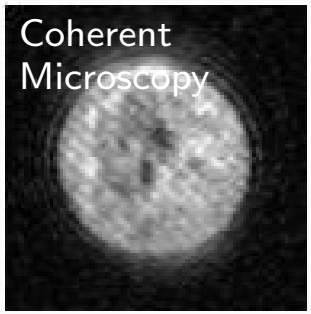
Microscopy

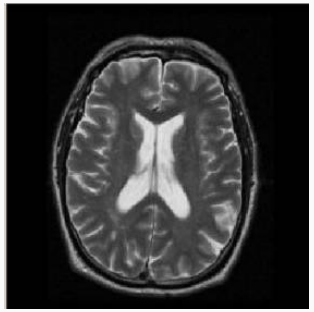
Medical Imaging

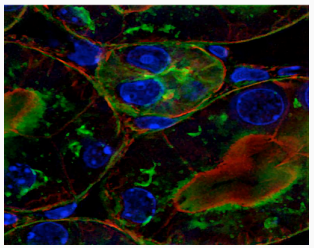


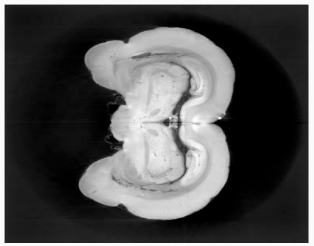








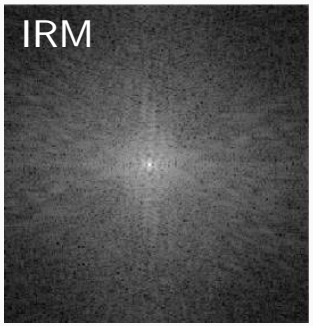


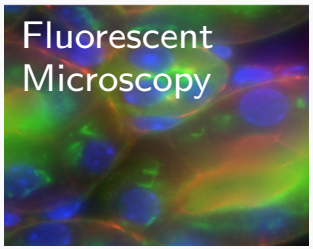


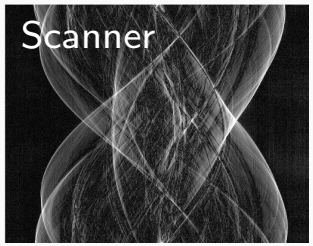
What is computational imaging ?

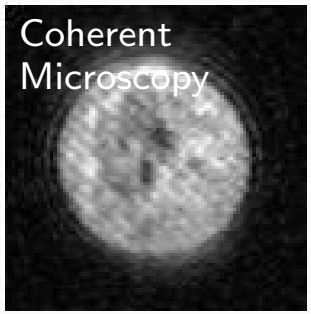
Microscopy

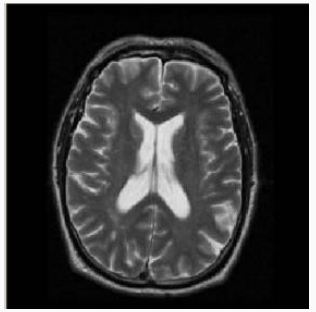
Medical Imaging

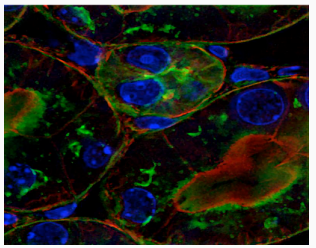


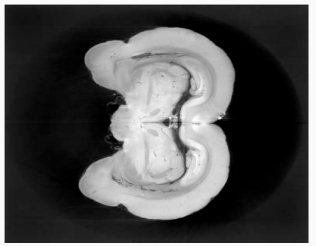


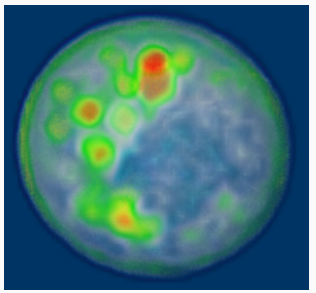








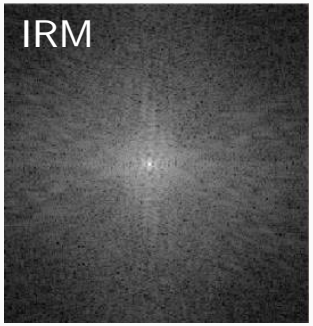


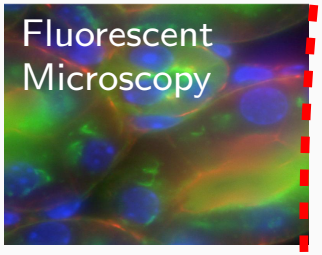


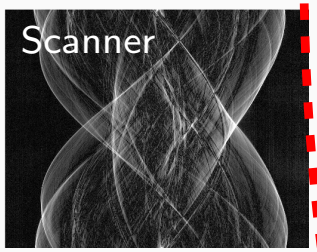
What is computational imaging ?

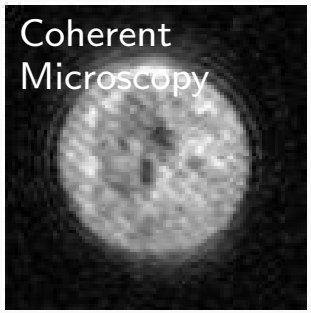
Microscopy

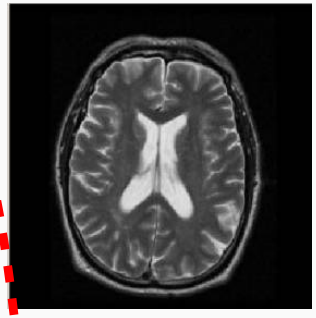
Medical Imaging

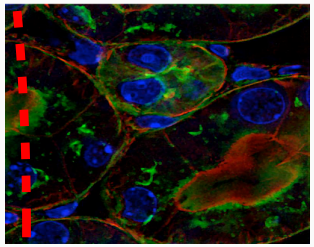


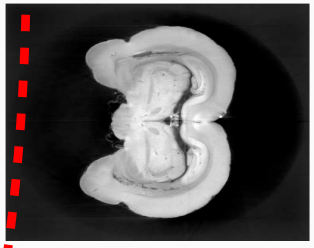


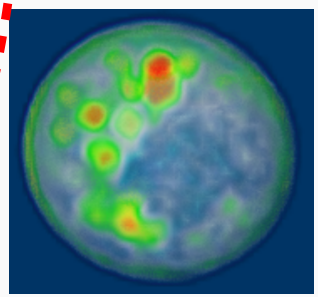












Outline

Practical information

From continuous to discretised image formation models

Ill-posedness

Maximum likelihood, MAP and MMSE

Regularisation: from priors to discrete models

From continuous to discretised image formation models

The Physics of Imaging – Image formation model

Direct/Forward model

$$\mathbf{y} = P(SHu)$$

- $u \in \mathcal{U}(\mathbb{R}^d)$ is the imaged (*continuous*) object,
- $\mathbf{y} \in \mathbb{R}^M$ is the vector of (*discrete*) measurements,
- $P : \mathbb{R}^M \to \mathbb{R}^M$ is a degradation operator (e.g., quantisation, noise),
- $H : \mathcal{U}(\mathbb{R}^d) \to \mathcal{Y}(\mathbb{R}^{d'})$ models the physics between the object and the signal to be recorded by the detector,
- $S : \mathcal{Y}(\mathbb{R}^{d'}) \to \mathbb{R}^M$ models the integration on the detector.

Direct/Forward model

$$\mathbf{y} = P(SHu)$$

Definitions of H and S

H and S are assumed to be **linear integral** operators,

$$\begin{aligned} H : \mathcal{U}(\mathbb{R}^d) &\longrightarrow \mathcal{Y}(\mathbb{R}^{d'}) \\ u &\longmapsto y(\,\cdot\,) = \int_{\mathbb{R}^d} h(\,\cdot\,, \mathbf{x}) u(\mathbf{x}) \,\mathrm{d}\mathbf{x}, \end{aligned}$$

where $h : \mathbb{R}^{d'} \times \mathbb{R}^d \to \mathbb{R}$ denote the **kernel** of H, and

$$\begin{aligned} S : \mathcal{Y}(\mathbb{R}^{d'}) &\longrightarrow \mathbb{R}^M \\ v &\longmapsto \mathbf{y} = \left[\int_{\mathbb{R}^{d'}} \psi_m(\mathbf{z}) y(\mathbf{z}) \,\mathrm{d}\mathbf{z} \right]_{m=1}^{M}, \end{aligned}$$

where $\{\psi_m\}_{m \in [M]}$ is a family of **sampling functions**.

Table 1: Kernels of basic linear integral operators.

Operator	$h(\mathbf{z}, \mathbf{x})$
Pointwise multiplication with function w	
Convolution with kernel $k(\cdot - \cdot)$	
Fourier transform	

Table 1: Kernels of basic linear integral operators.

Operator	$h(\mathbf{z}, \mathbf{x})$
Pointwise multiplication with function w	$w(\mathbf{z})\delta(\mathbf{z} - \mathbf{x})$
Convolution with kernel $k(\cdot - \cdot)$	
Fourier transform	

Table 1: Kernels of basic linear integral operators.

Operator	$h(\mathbf{z}, \mathbf{x})$
Pointwise multiplication with function w	$w(\mathbf{z})\delta(\mathbf{z} - \mathbf{x})$
Convolution with kernel $k(\cdot - \cdot)$	$k(\mathbf{x} - \mathbf{z})$
Fourier transform	

Table 1: Kernels of basic linear integral operators.

Operator	$h(\mathbf{z}, \mathbf{x})$
Pointwise multiplication with function w	$w(\mathbf{z})\delta(\mathbf{z} - \mathbf{x})$
Convolution with kernel $k(\cdot - \cdot)$	$k(\mathbf{x} - \mathbf{z})$
Fourier transform	$\exp(-\mathrm{j}\mathbf{z}^T\mathbf{x})$

Table 1: Kernels of basic linear integral operators.

Operator	$h(\mathbf{z}, \mathbf{x})$
Pointwise multiplication with function w	$w(\mathbf{z})\delta(\mathbf{z} - \mathbf{x})$
Convolution with kernel $k(\cdot - \cdot)$	$k(\mathbf{x} - \mathbf{z})$
Fourier transform	$\exp(-\mathrm{j}\mathbf{z}^T\mathbf{x})$

Table 2: Examples of sampling functions

Sampling functions	$\psi_m(\mathbf{z})$
Sampling at points $\{\mathbf{x}_m\}_{m\in[M]}$	
Pixel integration (size $\mathbf{s} \in \mathbb{R}^{d'}_{>0}$ at $\{\mathbf{x}_m\}_{m\in[M]}$)	

Table 1: Kernels of basic linear integral operators.

Operator	$h(\mathbf{z}, \mathbf{x})$
Pointwise multiplication with function w	$w(\mathbf{z})\delta(\mathbf{z} - \mathbf{x})$
Convolution with kernel $k(\cdot - \cdot)$	$k(\mathbf{x} - \mathbf{z})$
Fourier transform	$\exp(-\mathrm{j}\mathbf{z}^T\mathbf{x})$

Table 2: Examples of sampling functions

Sampling functions	$\psi_m(\mathbf{z})$
Sampling at points $\{\mathbf{x}_m\}_{m\in[M]}$	$\delta(\mathbf{z} - \mathbf{x}_m)$
Pixel integration (size $\mathbf{s} \in \mathbb{R}^{d'}_{>0}$ at $\{\mathbf{x}_m\}_{m\in[M]}$)	

Table 1: Kernels of basic linear integral operators.

Operator	$h(\mathbf{z}, \mathbf{x})$
Pointwise multiplication with function w	$w(\mathbf{z})\delta(\mathbf{z}-\mathbf{x})$
Convolution with kernel $k(\cdot - \cdot)$	$k(\mathbf{x}-\mathbf{z})$
Fourier transform	$\exp(-\mathrm{j}\mathbf{z}^T\mathbf{x})$

Table 2: Examples of sampling functions

Sampling functions	$\psi_m(\mathbf{z})$
Sampling at points $\{\mathbf{x}_m\}_{m\in[M]}$	$\delta(\mathbf{z}-\mathbf{x}_m)$
Pixel integration (size $\mathbf{s} \in \mathbb{R}^{d'}_{>0}$ at $\{\mathbf{x}_m\}_{m\in[M]}$)	$\mathbb{1}_{[-1,1]^{d'}}\left(\frac{\mathbf{z}-\mathbf{x}_m}{\mathbf{s}/2}\right)$

Table 1: Kernels of basic linear integral operators.

Operator	$h(\mathbf{z}, \mathbf{x})$
Pointwise multiplication with function w	$w(\mathbf{z})\delta(\mathbf{z}-\mathbf{x})$
Convolution with kernel $k(\cdot - \cdot)$	$k(\mathbf{x}-\mathbf{z})$
Fourier transform	$\exp(-\mathrm{j}\mathbf{z}^T\mathbf{x})$

Table 2: Examples of sampling functions

Sampling functions	$\psi_m(\mathbf{z})$
Sampling at points $\{\mathbf{x}_m\}_{m\in[M]}$	$\delta(\mathbf{z}-\mathbf{x}_m)$
Pixel integration (size $\mathbf{s}\in\mathbb{R}^{d'}_{>0}$ at $\{\mathbf{x}_m\}_{m\in[M]}$)	$\mathbb{1}_{[-1,1]^{d'}}\left(\frac{\mathbf{z}-\mathbf{x}_m}{\mathbf{s}/2}\right)$

Physical phenomena $\approx$ compositions of basic linear operators

The Physics of Imaging – Blur

From diffraction to point spread function (PSF)

The finite aperture of optical elements induces diffraction (light blur).

From diffraction to point spread function (PSF)

The finite aperture of optical elements induces diffraction (light blur).

A point source is not imaged as a point, but as a **point spread function** k

From diffraction to point spread function (PSF)

The finite aperture of optical elements induces diffraction (light blur).

A point source is not imaged as a point, but as a **point spread function** k

For a **spatially invariant system** : $H\delta(\cdot - \mathbf{x}) = (H\delta)(\cdot - \mathbf{x})$, $\forall \mathbf{x}$, we have

$$Hu \leftrightarrow k * u \qquad \text{(i.e., } h(\mathbf{z}, \mathbf{x}) = k(\mathbf{x} - \mathbf{z})\text{)}$$

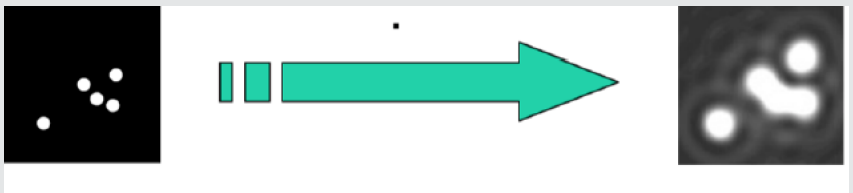
From diffraction to point spread function (PSF)

The finite aperture of optical elements induces diffraction (light blur).
A point source is not imaged as a point, but as a **point spread function** k

For a **spatially invariant system** : $H\delta(\cdot - \mathbf{x}) = (H\delta)(\cdot - \mathbf{x})$, $\forall \mathbf{x}$, we have

$$Hu \leftrightarrow k * u \qquad \text{(i.e., } h(\mathbf{z}, \mathbf{x}) = k(\mathbf{x} - \mathbf{z})\text{)}$$

Limitations in image resolution



X-Ray transform

For $d \geq 2$, the X-ray transform is defined by the line integral

$$Hu(\mathbf{s}, \mathbf{w}) = \int_{L_{\mathbf{s},\mathbf{w}}} u \, \mathrm{d}l = \int_{\mathbb{R}} u(\mathbf{s} + t\mathbf{w}) \mathrm{d}t,$$

- for all compactly supported $u : \Omega \subset \mathbb{R}^d \to \mathbb{R}$
- $L_{\mathbf{s},\mathbf{w}} := \{\mathbf{x} = \mathbf{s} + t\mathbf{w}, \ \forall t \in \mathbb{R}\} \subset \mathbb{R}^d$: lines through $\mathbf{s} \in \mathbb{R}^d$ with direction $\mathbf{w} \in \mathbb{R}^d$.

Radon transform

For $d \geq 2$, the Radon transform is defined by the line integral

$$Ru(\boldsymbol{\theta}, \rho) = \int_{L_{\boldsymbol{\theta},\rho}} u \, \mathrm{d}\sigma = \int_{\mathbf{x} \cdot \boldsymbol{\theta} = \rho} u(\mathbf{x}) \, \mathrm{d}\sigma(\mathbf{x}),$$

- for all hyperplanes

  $$L_{\theta,\rho} := \{\mathbf{x} \in \mathbb{R}^d : \mathbf{x} \cdot \boldsymbol{\theta} = \rho\},$$

  with normal direction $\boldsymbol{\theta} \in \mathbb{S}^{d-1}$ and distance $\rho \in \mathbb{R}^+$ from the origin.
- In $d = 2$, this gives integration over lines with

  $$\boldsymbol{\theta} = (\cos\varphi, \sin\varphi), \qquad \boldsymbol{\theta} \perp \mathbf{w}.$$

The Physics of Imaging – X-ray transforms

Central to many imaging modalities

- Computed tomography (CT, PET) in medical imaging
- Electronic microscopy
- Radio astronomy
- ...

X-Ray transform

For $d \geq 2$, the X-ray transform is defined by the line integral

$$Hu(\mathbf{s}, \mathbf{w}) = \int_{L_{\mathbf{s},\mathbf{w}}} u \, \mathrm{d}l = \int_{\mathbb{R}} u(\mathbf{s} + t\mathbf{w}) \mathrm{d}t,$$

- for all compactly supported $u : \Omega \subset \mathbb{R}^d \to \mathbb{R}$
- $L_{\mathbf{s},\mathbf{w}} := \{\mathbf{x} = \mathbf{s} + t\mathbf{w}, \ \forall t \in \mathbb{R}\} \subset \mathbb{R}^d$: lines through $\mathbf{s} \in \mathbb{R}^d$ with direction $\mathbf{w} \in \mathbb{R}^d$.

Description within the general model

For $\mathbf{z} = (\mathbf{s}, \mathbf{w}) \in \mathbb{R}^d \times \mathbb{R}^d$, $\mathbf{x} \in \mathbb{R}^d$

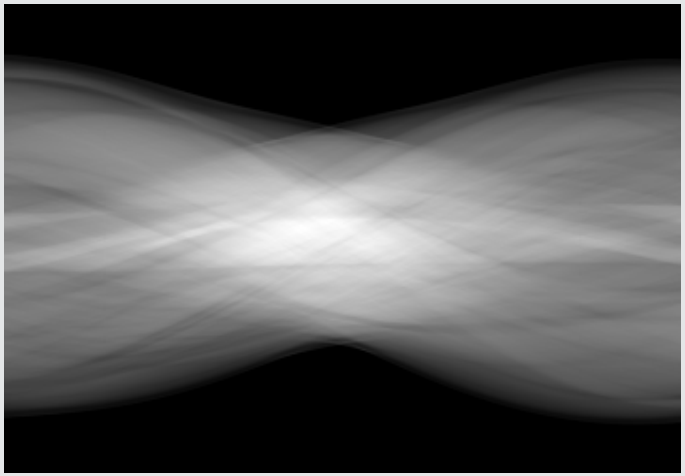
$$h(\mathbf{z}, \mathbf{x}) = h\left((\mathbf{s}, \mathbf{w}), \mathbf{x}\right) = \delta\left(\|\mathbf{P}_{\mathbf{w}^\perp}(\mathbf{x} - \mathbf{s})\|_2\right),$$

- $\mathbf{P}_{\mathbf{w}^\perp} \in \mathbb{R}^{(d-1)\times d}$ is the projection onto the subspace orthogonal to $\mathbf{w}$.

Link with Radon transform

In dimension $d = 2$, X-ray transform $\equiv$ Radon transform.

Object u



Sinogram $\mathbf{y}$ (Radon transform)

Different types of detector

- **photosensitive detectors**: in microscopy/tomography,
- **pressure sensors**: ultrasound imaging,
- **antennas**: magnetic resonance imaging (MRI, Fourier measurements).

Different types of detector

- **photosensitive detectors**: in microscopy/tomography,
- **pressure sensors**: ultrasound imaging,
- **antennas**: magnetic resonance imaging (MRI, Fourier measurements).

A "general" description of a detector

A finite number of **compact domains** $(A_m)_{m\in[M]}$, $A_m \subset \mathbb{R}^{d'}$ over which the received signal is *integrated* with $\psi_m = f_m \mathbb{1}_{A_m}$ for function f_m (often constant).

When A_m is sufficiently small, the approximation $\psi_m = \delta_{\mathbf{x}_m}$ can be used so that $\mathbf{y}[m] = Hu(\mathbf{x}_m)$ (sampling operator).

From continuous to discrete – Objectives

Generic direct model

$$\mathbf{y} = P(SHu)$$

From continuous to discrete – Objectives

Generic direct model

$$\mathbf{y} = P(SHu)$$

$\longrightarrow$ link the **continuous** (unknown) object u to the **discrete** measurements $\mathbf{y}$

From continuous to discrete – Objectives

Generic direct model

$$\mathbf{y} = P(SHu)$$

$\longrightarrow$ link the **continuous** (unknown) object u to the **discrete** measurements $\mathbf{y}$

$\longrightarrow$ not suitable for numerical treatment (life is discrete!)

From continuous to discrete – Objectives

Generic direct model

$$\mathbf{y} = P(SHu)$$

$\longrightarrow$ link the **continuous** (unknown) object u to the **discrete** measurements $\mathbf{y}$

$\longrightarrow$ not suitable for numerical treatment (life is discrete!)

Discretized model

Define a **discrete representation** $\mathbf{u} \in \mathbb{R}^N$ ($N \in \mathbb{N}$) of $u \in \mathcal{U}(\mathbb{R}^d)$ (e.g., through sampling, basis expansion) and **matrix** $\mathbf{A} \in \mathbb{R}^{M \times N}$ such that

- the approximation error $Ad\mathbf{u} \approx SHu$
- fast algorithms can be used to compute $\mathbf{Au}$ without building $\mathbf{A}$.

From continuous to discrete – General principle

Approximation space for u

Let $\{\phi_n\}_{n\in[N]} \subset \mathcal{U}(\mathbb{R}^d)$ be a family of linearly independent functions of $\mathcal{U}(\mathbb{R}^d)$.
Define:

$$\mathcal{U}_N(\mathbb{R}^d) = \mathrm{span}(\{\phi_n\}_{n\in[N]}) = \left\{ \mathbf{\Phi u} : \mathbf{u} \in \mathbb{R}^N \right\}$$

where $\mathbf{\Phi} : \mathbb{R}^N \longrightarrow \mathcal{U}(\mathbb{R}^d)$ is defined by

$$\mathbf{\Phi u} = \sum_{n\in[N]} u_n \phi_n.$$

From continuous to discrete – General principle

Approximation space for u

Let $\{\phi_n\}_{n\in[N]} \subset \mathcal{U}(\mathbb{R}^d)$ be a family of linearly independent functions of $\mathcal{U}(\mathbb{R}^d)$.
Define:

$$\mathcal{U}_N(\mathbb{R}^d) = \mathrm{span}(\{\phi_n\}_{n\in[N]}) = \left\{ \mathbf{\Phi u} : \mathbf{u} \in \mathbb{R}^N \right\}$$

where $\mathbf{\Phi} : \mathbb{R}^N \longrightarrow \mathcal{U}(\mathbb{R}^d)$ is defined by

$$\mathbf{\Phi u} = \sum_{n\in[N]} u_n \phi_n.$$

$\longrightarrow$ Any $u \in \mathcal{U}_N(\mathbb{R}^d)$ is **fully described** by a vector $\mathbf{u} = (u_1, \ldots, u_N) \in \mathbb{R}^N$

From continuous to discrete – General principle

Approximation space for u

Let $\{\phi_n\}_{n\in[N]} \subset \mathcal{U}(\mathbb{R}^d)$ be a family of linearly independent functions of $\mathcal{U}(\mathbb{R}^d)$. Define:

$$\mathcal{U}_N(\mathbb{R}^d) = \mathrm{span}(\{\phi_n\}_{n\in[N]}) = \left\{ \mathbf{\Phi u} : \mathbf{u} \in \mathbb{R}^N \right\}$$

where $\mathbf{\Phi} : \mathbb{R}^N \longrightarrow \mathcal{U}(\mathbb{R}^d)$ is defined by

$$\mathbf{\Phi u} = \sum_{n\in[N]} u_n \phi_n.$$

$\longrightarrow$ Any $u \in \mathcal{U}_N(\mathbb{R}^d)$ is **fully described** by a vector $\mathbf{u} = (u_1, \ldots, u_N) \in \mathbb{R}^N$

Finite-dimensional projection

Consider then $P_{\mathcal{U}_N}(u)$, the orthogonal projection of $u \in \mathcal{U}(\mathbb{R}^d)$ on $\mathcal{U}_N(\mathbb{R}^d)$

$$\mathbf{u} = P_{\mathcal{U}_N}(u) = \arg \min_{\mathbf{v}\in\mathcal{U}_N(\mathbb{R}^d)} \|\mathbf{v} - u\|^2_{\mathcal{U}(\mathbb{R}^d)}.$$

From continuous to discrete – General principle

Usual choice of the family $\{\phi_n\}_{n\in[N]}$

$$\phi_n = \phi(\cdot - \mathbf{x}_n)$$

- $\phi \in \mathcal{U}(\mathbb{R}^d)$ a **generating function** (e.g., sinc, spline, wavelet),
- $\Omega = \{\mathbf{x}_n\}_{n\in[N]}$ **grid** of N points of $\Omega \subset \mathbb{R}^d$ evenly spaced,
- such that the family is **linearly independent**.

From continuous to discrete – General principle

Discrete forward model

Let $u \in \mathcal{U}_N(\mathbb{R}^d)$ and its discrete representation $\mathbf{u} \in \mathbb{R}^N$, then $\forall m \in [M]$,

$$[SHu]_m = \sum_{n \in [N]} u_n [SH\phi_n]_m, \qquad \text{(apply only to elements } \phi_n\text{)}$$

Defining $[\mathbf{A}_0]_{m,n} = [SH\phi_n]_m$ we get

$$SHu = \mathbf{A}_0 \mathbf{u}$$

From continuous to discrete – General principle

Discrete forward model

Let $u \in \mathcal{U}_N(\mathbb{R}^d)$ and its discrete representation $\mathbf{u} \in \mathbb{R}^N$, then $\forall m \in [M]$,

$$[SHu]_m = \sum_{n \in [N]} u_n [SH\phi_n]_m, \qquad \text{(apply only to elements } \phi_n\text{)}$$

Defining $[\mathbf{A}_0]_{m,n} = [SH\phi_n]_m$ we get

$$SHu = \mathbf{A}_0 \mathbf{u}$$

Practical challenges

1. Coefficients $[\mathbf{A}_0]_{m,n}$ may not be computed analytically (depend on ϕ_n)
2. $\mathbf{A}_0 \in \mathbb{R}^{M \times N}$ is too large to be **stored** and to perform $O(MN)$ **matrix-vector products** $\mathbf{A}_0\mathbf{u}$.

From continuous to discrete – General principle

Discrete forward model

$$SHu = \mathbf{A}_0\mathbf{u}$$

Practical challenges

1. Coefficients $[\mathbf{A}_0]_{m,n}$ may not be computed analytically
2. $\mathbf{A}_0 \in \mathbb{R}^{M\times N}$ is too large to be **stored** and to perform $O(MN)$ **matrix-vector products** $\mathbf{A}_0\mathbf{u}$.

From continuous to discrete – General principle

Discrete forward model

$$SHu = \mathbf{A}_0\mathbf{u}$$

Practical challenges

1. Coefficients $[\mathbf{A}_0]_{m,n}$ may not be computed analytically
2. $\mathbf{A}_0 \in \mathbb{R}^{M\times N}$ is too large to be **stored** and to perform $O(MN)$ **matrix-vector products** $\mathbf{A}_0\mathbf{u}$.
   - 500 GFlops processor,
   - M = N = 10 millions (image 2D $\approx 3162 \times 3162$ or 3D $\approx 500 \times 500 \times 40$),
   - 200 seconds to perform a standard matrix-vector product,
   - 800 To to store $\mathbf{A}_0$!

From continuous to discrete – General principle

Discrete forward model

$$SHu = \mathbf{A}_0\mathbf{u}$$

Practical challenges

1. Coefficients $[\mathbf{A}_0]_{m,n}$ may not be computed analytically
2. $\mathbf{A}_0 \in \mathbb{R}^{M\times N}$ is too large to be **stored** and to perform $O(MN)$ **matrix-vector products** $\mathbf{A}_0\mathbf{u}$.
   - 500 GFlops processor,
   - M = N = 10 millions (image 2D $\approx 3162 \times 3162$ or 3D $\approx 500 \times 500 \times 40$),
   - 200 seconds to perform a standard matrix-vector product,
   - 800 To to store $\mathbf{A}_0$!

Alternative : numerical integration + structured approximation

From continuous to discrete – Numerical integration

Quadrature rule

$$[\mathbf{A}_0]_{m,n} = \int_{\mathbb{R}^{d'}} \int_{\mathbb{R}^d} \psi_m(\mathbf{z})\, h(\mathbf{z},\mathbf{x})\, \phi_n(\mathbf{x})\, \mathrm{d}\mathbf{x}\, \mathrm{d}\mathbf{z}$$

$$\approx [\widehat{\mathbf{A}}]_{m,n} = \sum_{p\in[P]} \sum_{q\in[Q]} w_{p,q} \psi_m(\mathbf{z}_p) h(\mathbf{z}_p, \mathbf{x}_q) \phi_n(\mathbf{x}_q),$$

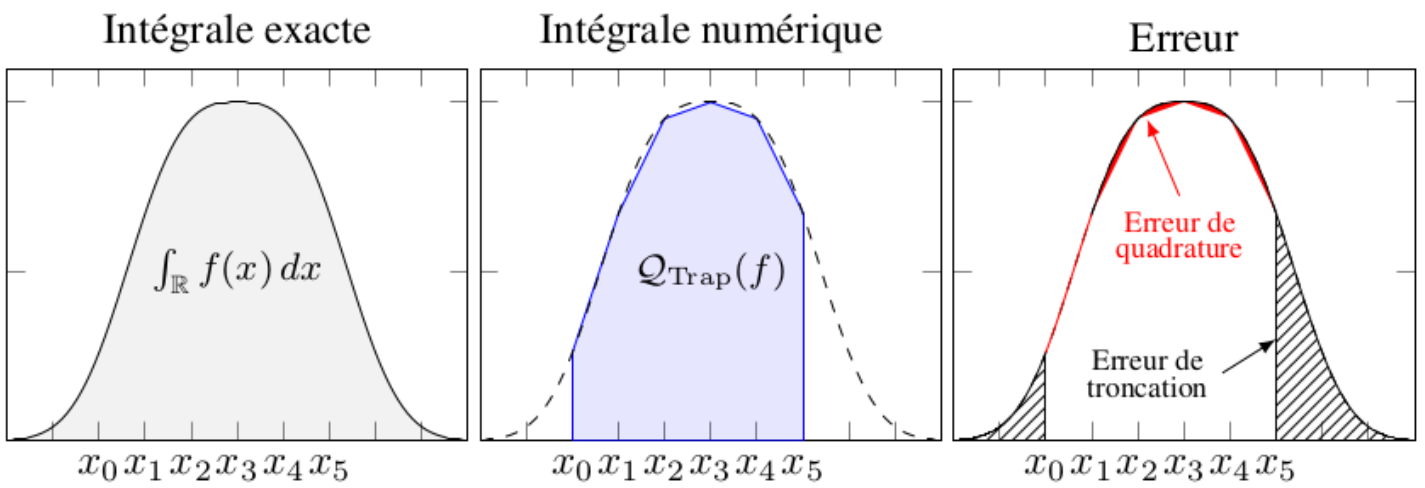
where the nodes $\{\mathbf{z}_p\}_{p\in[P]}$ and $\{\mathbf{x}_q\}_{q\in[Q]}$, as well as the coefficients $\{w_{p,q}\}_{(p,q)\in[P]\times[Q]}$, depend on the desired accuracy and the used quadrature.

Quadrature rule

$$[\mathbf{A}_0]_{m,n} = \int_{\mathbb{R}^{d'}} \int_{\mathbb{R}^d} \psi_m(\mathbf{z})\, h(\mathbf{z},\mathbf{x})\, \phi_n(\mathbf{x})\, \mathrm{d}\mathbf{x}\, \mathrm{d}\mathbf{z}$$

$$\approx [\widehat{\mathbf{A}}]_{m,n} = \sum_{p\in[P]} \sum_{q\in[Q]} w_{p,q} \psi_m(\mathbf{z}_p) h(\mathbf{z}_p,\mathbf{x}_q) \phi_n(\mathbf{x}_q),$$

where the nodes $\{\mathbf{z}_p\}_{p\in[P]}$ and $\{\mathbf{x}_q\}_{q\in[Q]}$, as well as the coefficients $\{w_{p,q}\}_{(p,q)\in[P]\times[Q]}$, depend on the desired accuracy and the used quadrature.



How to consider now **structured matrices A**? What is a structured matrix?

Sparse matrices

A matrix $\mathbf{B} \in \mathbb{R}^{M \times N}$ is **sparse** if it **contains many zeros**,

$$\mathbf{B} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & \bullet & 0 & 0 & \bullet \\ 0 & \bullet & 0 & 0 & 0 & 0 & 0 & 0 & \bullet \\ 0 & 0 & 0 & 0 & \bullet & 0 & 0 & 0 & 0 \\ 0 & 0 & \bullet & 0 & 0 & \bullet & 0 & 0 & 0 \\ \bullet & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

where $\bullet$ are non-zero elements.

From continuous to discrete – Structured matrices

How to consider now **structured matrices A**? What is a structured matrix?

Sparse matrices

A matrix $\mathbf{B} \in \mathbb{R}^{M \times N}$ is **sparse** if it **contains many zeros**,

$$\mathbf{B} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & \bullet & 0 & 0 & \bullet \\ 0 & \bullet & 0 & 0 & 0 & 0 & 0 & 0 & \bullet \\ 0 & 0 & 0 & 0 & \bullet & 0 & 0 & 0 & 0 \\ 0 & 0 & \bullet & 0 & 0 & \bullet & 0 & 0 & 0 \\ \bullet & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

where $\bullet$ are non-zero elements.

Memory and matrix-vector product complexities

If the K non-zeros are such that $K < (M(N-1)-1)/2$, the **Compressed Sparse Row** allows for

- memory/storage cost: $O(2K + M)$
- matrix vector product cost: $O(K)$

Sparse approximation

In practice $\widehat{\mathbf{A}}$ may not be sparse but **contains many small coefficients**
$\longrightarrow$ We can define a sparse approximation $\mathbf{A} = T_\epsilon(\widehat{\mathbf{A}})$ of $\widehat{\mathbf{A}}$ where

$$\forall \mathbf{B},\ \forall (m,n), \quad [T_\epsilon(\mathbf{B})]_{m,n} = \begin{cases} 0 & \text{ifi } |b_{m,n}| < \epsilon, \\ b_{m,n} & \text{otherwise.} \end{cases}$$

$\epsilon > 0$: **trade-off** between **approximation and computational complexity**.

Band/Multi-diagonal matrices

A square matrix $\mathbf{B} \in \mathbb{R}^{N \times N}$ is **multi-diagonal** if its non-zero entries are concentrated around the diagonal,

$$\mathbf{B} = \begin{pmatrix} \bullet & \bullet & 0 & 0 & 0 \\ \bullet & \bullet & \bullet & 0 & 0 \\ 0 & \bullet & \bullet & \bullet & 0 \\ 0 & 0 & \bullet & \bullet & \bullet \\ 0 & 0 & 0 & \bullet & \bullet \end{pmatrix}.$$

Band/Multi-diagonal matrices

A square matrix $\mathbf{B} \in \mathbb{R}^{N \times N}$ is **multi-diagonal** if its non-zero entries are concentrated around the diagonal,

$$\mathbf{B} = \begin{pmatrix} \bullet & \bullet & 0 & 0 & 0 \\ \bullet & \bullet & \bullet & 0 & 0 \\ 0 & \bullet & \bullet & \bullet & 0 \\ 0 & 0 & \bullet & \bullet & \bullet \\ 0 & 0 & 0 & \bullet & \bullet \end{pmatrix}.$$

Memory and matrix-vector product complexities

Only the L bands needs to be stored, leading to a memory and matrix vector product complexity of $O(LN)$.

Decaying kernels

Band matrices are well adapted to approximate matrices resulting from the discretization of integral operators with **decaying kernels** such that

$$h(\mathbf{z}, \mathbf{x}) \leq \frac{c}{\|\mathbf{z} - \mathbf{x}\|_2^{d+\alpha}}$$

for some $c > 0$ and $\alpha > 0$.
$\longrightarrow$ An approximation of $\widehat{\mathbf{A}}$ is given by $\mathbf{A} = T_\beta(\widehat{\mathbf{A}})$ with

$$\forall \mathbf{B}, \forall (m, n), \quad [T_\beta(\mathbf{B})]_{m,n} = \begin{cases} b_{m,n} & \text{if } |m - n| \leq \beta, \\ 0 & \text{otherwise.} \end{cases}$$

$\beta > 0$: **trade-off** between **approximation and computational complexity**.

Circulant matrices

A square matrix $\mathbf{B} \in \mathbb{R}^{N \times N}$ is **circulant** if there exists $\mathbf{c} \in \mathbb{R}^N$ such that $b_{i,j} = c_{\mathrm{mod}(j-i,N)+1}$,

$$\mathbf{B} = \begin{pmatrix} c_1 & c_2 & \dots & c_{N-1} & c_N \\ c_N & c_1 & c_2 & \dots & c_{N-1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ c_2 & \dots & c_{N-1} & c_N & c_1 \end{pmatrix}.$$

Circulant matrices

A square matrix $\mathbf{B} \in \mathbb{R}^{N\times N}$ is **circulant** if there exists $\mathbf{c} \in \mathbb{R}^N$ such that $b_{i,j} = c_{\mathrm{mod}(j-i,N)+1}$,

$$\mathbf{B} = \begin{pmatrix} c_1 & c_2 & \dots & c_{N-1} & c_N \\ c_N & c_1 & c_2 & \dots & c_{N-1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ c_2 & \dots & c_{N-1} & c_N & c_1 \end{pmatrix}.$$

Circulant matrices can be **diagonalised in the Fourier basis**

$$\mathbf{B} = \mathbf{F}^{-1}\mathbf{diag}\left(\mathbf{F}\mathbf{c}\right)\mathbf{F},$$

Circulant matrices

A square matrix $\mathbf{B} \in \mathbb{R}^{N\times N}$ is **circulant** if there exists $\mathbf{c} \in \mathbb{R}^N$ such that $b_{i,j} = c_{\mathrm{mod}(j-i,N)+1}$,

$$\mathbf{B} = \begin{pmatrix} c_1 & c_2 & \dots & c_{N-1} & c_N \\ c_N & c_1 & c_2 & \dots & c_{N-1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ c_2 & \dots & c_{N-1} & c_N & c_1 \end{pmatrix}.$$

Circulant matrices can be **diagonalised in the Fourier basis**

$$\mathbf{B} = \mathbf{F}^{-1}\mathbf{diag}\left(\mathbf{F}\mathbf{c}\right)\mathbf{F},$$

Memory and matrix-vector product complexities

Exploiting fast Fourier transform algorithms (FFT), we have

- matrix vector product cost : $O(2N\log N + N)$

Circulant matrices

A square matrix $\mathbf{B} \in \mathbb{R}^{N \times N}$ is **circulant** if there exists $\mathbf{c} \in \mathbb{R}^N$ such that $b_{i,j} = c_{\mathrm{mod}(j-i,N)+1}$,

$$\mathbf{B} = \begin{pmatrix} c_1 & c_2 & \dots & c_{N-1} & c_N \\ c_N & c_1 & c_2 & \dots & c_{N-1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ c_2 & \dots & c_{N-1} & c_N & c_1 \end{pmatrix}.$$

Circulant matrices can be **diagonalised in the Fourier basis**

$$\mathbf{B} = \mathbf{F}^{-1}\mathbf{diag}\left(\mathbf{Fc}\right)\mathbf{F},$$

Remarks

- Special case of **sparse matrices** (in the Fourier basis)
- Discretisation of **convolution** operators with **periodic boundary conditions**

From continuous to discrete – Structured matrices

Low rank matrices

A matrix $\mathbf{B} \in \mathbb{R}^{M \times N}$ is **low rank** if its rank R is such that $R \ll \min(M, N)$.

Low rank matrices

A matrix $\mathbf{B} \in \mathbb{R}^{M \times N}$ is **low rank** if its rank R is such that $R \ll \min(M, N)$.

Memory and matrix-vector product complexities

Exploiting the singular value decomposition (SVD), $\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$ (with $\mathbf{U} \in \mathbb{R}^{M \times R}$, $\mathbf{\Sigma} \in \mathbb{R}^{R \times R}$ et $\mathbf{V} \in \mathbb{R}^{N \times R}$), we have

- memory cost : $O(R(M + N))$
- matrix vector product cost : $O(R(M + N))$

Table 3: Processor 500 GFlops per second.

	Complexit Thorique		Ex. $M = N = 10^7$	
Structure	Time	Memory	Time	Memory
K-Sparse	$O(K)$	$O(2K + M)$ *(format CSR)*	0,2 ms	1.68 Go $(K = 10^{-6}MN)$
L-Diagonal	$O(LN)$	$O(LN)$	0,06 ms	240 Mo $(L = 3)$
Circulant	$O(2N \log N + N)$	$O(2N)$	0,3 ms	160 Mo
Toeplitz	$O(4N \log(2N) + 2N)$	$O(4N)$	0,6 ms	320 Mo
Low rank (R)	$O(R(M + N))$	$O(R(M + N))$	0,4 ms	1.6 Go $(R = 10)$
Prod. standard	$O(MN)$	$O(MN)$	200 s	800 To

Outline

Practical information

From continuous to discretised image formation models

Ill-posedness

Maximum likelihood, MAP and MMSE

Regularisation: from priors to discrete models

Ill-posedness

Deconvolution example (square matrix A)

$$\mathbf{y} = \mathbf{A}\mathbf{u} + \mathbf{n}$$

with $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_N)$ and $\mathbf{A} \in \mathbb{R}^{N \times N}$ a **convolution operator (periodic BC)** with kernel $\mathbf{h}$,

$$(\mathbf{A}\mathbf{u})[\mathbf{m}] = (\mathbf{h} \star \mathbf{u})[\mathbf{m}] = \sum_{\mathbf{l}} \mathbf{h}[\mathbf{m} - \mathbf{l}]\mathbf{u}[\mathbf{l}]$$

Deconvolution example (square matrix A)

$$\mathbf{y} = \mathbf{A}\mathbf{u} + \mathbf{n}$$

with $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_N)$ and $\mathbf{A} \in \mathbb{R}^{N \times N}$ a **convolution operator (periodic BC)** with kernel $\mathbf{h}$,

$$(\mathbf{A}\mathbf{u})[\mathbf{m}] = (\mathbf{h} \star \mathbf{u})[\mathbf{m}] = \sum_{\mathbf{l}} \mathbf{h}[\mathbf{m} - \mathbf{l}]\mathbf{u}[\mathbf{l}]$$

As $\mathbf{A}$ is **circulant**, we have $\mathbf{A} = \mathbf{F}^{-1}\mathbf{diag}(\mathbf{F}\mathbf{h})\mathbf{F}$

Deconvolution example (square matrix A)

$$\mathbf{y} = \mathbf{A}\mathbf{u} + \mathbf{n}$$

with $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_N)$ and $\mathbf{A} \in \mathbb{R}^{N \times N}$ a **convolution operator (periodic BC)** with kernel $\mathbf{h}$,

$$(\mathbf{A}\mathbf{u})[\mathbf{m}] = (\mathbf{h} \star \mathbf{u})[\mathbf{m}] = \sum_{\mathbf{l}} \mathbf{h}[\mathbf{m} - \mathbf{l}]\mathbf{u}[\mathbf{l}]$$

As $\mathbf{A}$ is **circulant**, we have $\mathbf{A} = \mathbf{F}^{-1}\mathbf{diag}(\mathbf{F}\mathbf{h})\mathbf{F}$

$$\mathbf{A} \text{ invertible} \iff (\mathbf{F}\mathbf{h})[\mathbf{l}] \neq 0 \text{ for all } \mathbf{l}$$

Deconvolution example (square matrix A)

$$\mathbf{y} = \mathbf{A}\mathbf{u} + \mathbf{n}$$

with $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_N)$ and $\mathbf{A} \in \mathbb{R}^{N \times N}$ a **convolution operator (periodic BC)** with kernel $\mathbf{h}$,

$$(\mathbf{A}\mathbf{u})[\mathbf{m}] = (\mathbf{h} \star \mathbf{u})[\mathbf{m}] = \sum_{\mathbf{l}} \mathbf{h}[\mathbf{m} - \mathbf{l}]\mathbf{u}[\mathbf{l}]$$

As $\mathbf{A}$ is **circulant**, we have $\mathbf{A} = \mathbf{F}^{-1}\mathbf{diag}(\mathbf{F}\mathbf{h})\mathbf{F}$

$$\mathbf{A} \text{ invertible} \iff (\mathbf{F}\mathbf{h})[\mathbf{l}] \neq 0 \text{ for all } \mathbf{l}$$

Direct inversion

$$\mathbf{u}_* = \mathbf{A}^{-1}\mathbf{y} = \mathbf{F}^{-1}\mathbf{diag}(\mathbf{F}\mathbf{h})^{-1}\mathbf{F}\mathbf{y}.$$

- When $\mathbf{n} = \mathbf{0}$, we have $\mathbf{u}_* = \mathbf{A}^{-1}\mathbf{A}\mathbf{u} = \mathbf{u}$
- When $\mathbf{n} \neq \mathbf{0}$, $\mathbf{u}_* = \mathbf{u} + \mathbf{F}^{-1}\mathbf{diag}(\mathbf{F}\mathbf{h})^{-1}\mathbf{F}\mathbf{n}$

Deconvolution example (square matrix A)

$$\mathbf{y} = \mathbf{A}\mathbf{u} + \mathbf{n}$$

with $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_N)$ and $\mathbf{A} \in \mathbb{R}^{N \times N}$ a **convolution operator (periodic BC)** with kernel $\mathbf{h}$,

$$(\mathbf{A}\mathbf{u})[\mathbf{m}] = (\mathbf{h} \star \mathbf{u})[\mathbf{m}] = \sum_{\mathbf{l}} \mathbf{h}[\mathbf{m} - \mathbf{l}]\mathbf{u}[\mathbf{l}]$$

As $\mathbf{A}$ is **circulant**, we have $\mathbf{A} = \mathbf{F}^{-1}\mathbf{diag}(\mathbf{F}\mathbf{h})\mathbf{F}$

$$\mathbf{A} \text{ invertible} \iff (\mathbf{F}\mathbf{h})[\mathbf{l}] \neq 0 \text{ for all } \mathbf{l}$$

Direct inversion

$$\mathbf{u}_* = \mathbf{A}^{-1}\mathbf{y} = \mathbf{F}^{-1}\mathbf{diag}(\mathbf{F}\mathbf{h})^{-1}\mathbf{F}\mathbf{y}.$$

- When $\mathbf{n} = \mathbf{0}$, we have $\mathbf{u}_* = \mathbf{A}^{-1}\mathbf{A}\mathbf{u} = \mathbf{u}$
- When $\mathbf{n} \neq \mathbf{0}$, $\mathbf{u}_* = \mathbf{u} + \mathbf{F}^{-1}\mathbf{diag}(\mathbf{F}\mathbf{h})^{-1}\mathbf{F}\mathbf{n}$

Deconvolution example (square matrix A)

$$\mathbf{y} = \mathbf{A}\mathbf{u} + \mathbf{n}$$

with $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_N)$ and $\mathbf{A} \in \mathbb{R}^{N \times N}$ a **convolution operator (periodic BC)** with kernel $\mathbf{h}$,

$$(\mathbf{A}\mathbf{u})[\mathbf{m}] = (\mathbf{h} \star \mathbf{u})[\mathbf{m}] = \sum_{\mathbf{l}} \mathbf{h}[\mathbf{m} - \mathbf{l}]\mathbf{u}[\mathbf{l}]$$

As $\mathbf{A}$ is **circulant**, we have $\mathbf{A} = \mathbf{F}^{-1}\mathbf{diag}(\mathbf{F}\mathbf{h})\mathbf{F}$

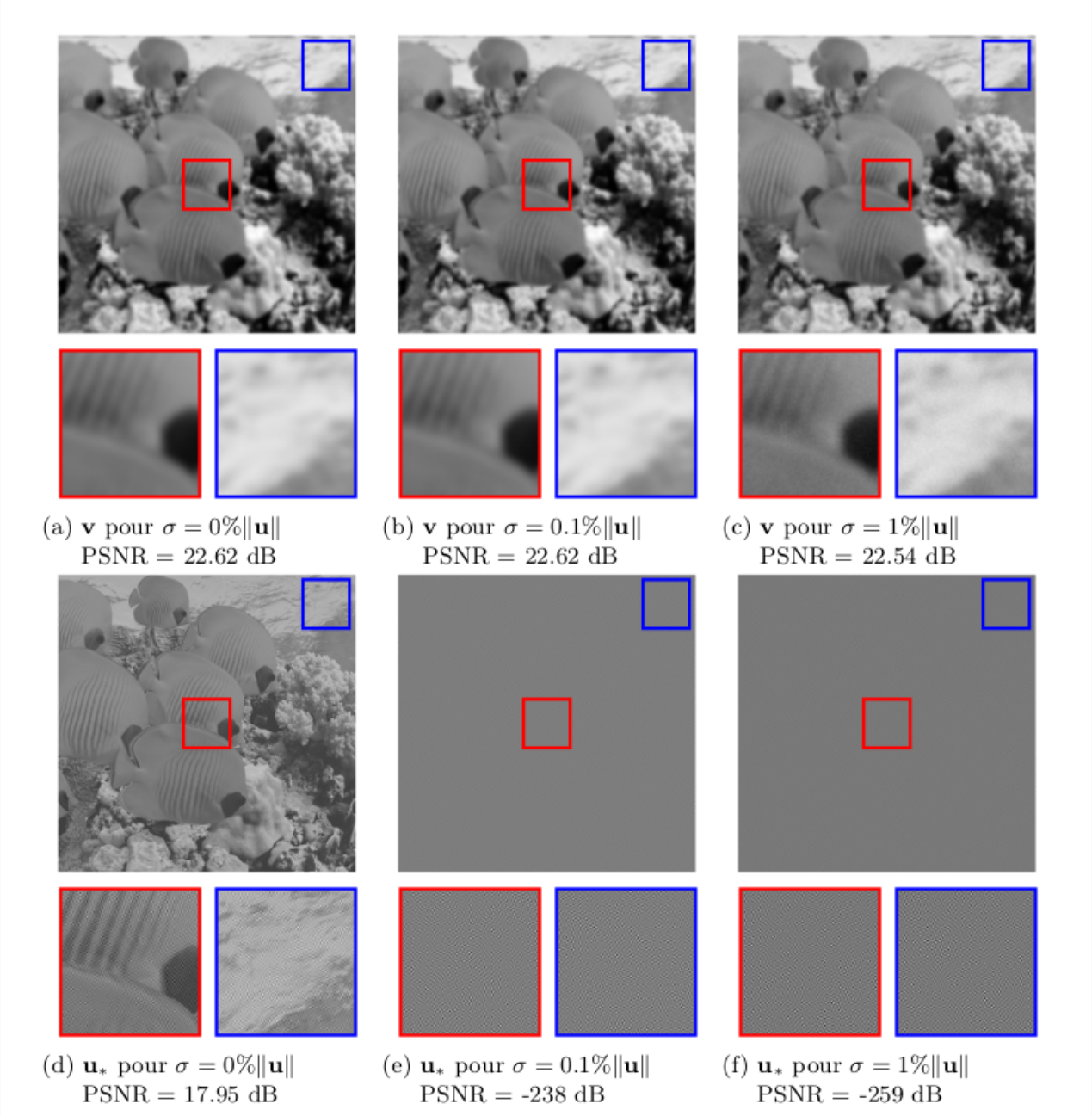
$$\mathbf{A} \text{ invertible} \iff (\mathbf{F}\mathbf{h})[\mathbf{l}] \neq 0 \text{ for all } \mathbf{l}$$

Direct inversion

$$\mathbf{u}_* = \mathbf{A}^{-1}\mathbf{y} = \mathbf{F}^{-1}\mathbf{diag}(\mathbf{F}\mathbf{h})^{-1}\mathbf{F}\mathbf{y}.$$

- When $\mathbf{n} = \mathbf{0}$, we have $\mathbf{u}_* = \mathbf{A}^{-1}\mathbf{A}\mathbf{u} = \mathbf{u}$
- When $\mathbf{n} \neq \mathbf{0}$, $\mathbf{u}_* = \mathbf{u} + \mathbf{F}^{-1}\mathbf{diag}(\mathbf{F}\mathbf{h})^{-1}\mathbf{F}\mathbf{n}$

Ill-posed inverse problems – Failure of direct inversions



(a) $\mathbf{v}$ pour $\sigma = 0\%\|\mathbf{u}\|$
PSNR = 22.62 dB

(b) $\mathbf{v}$ pour $\sigma = 0.1\%\|\mathbf{u}\|$
PSNR = 22.62 dB

(c) $\mathbf{v}$ pour $\sigma = 1\%\|\mathbf{u}\|$
PSNR = 22.54 dB

(d) $\mathbf{u}_*$ pour $\sigma = 0\%\|\mathbf{u}\|$
PSNR = 17.95 dB

(e) $\mathbf{u}_*$ pour $\sigma = 0.1\%\|\mathbf{u}\|$
PSNR = -238 dB

(f) $\mathbf{u}_*$ pour $\sigma = 1\%\|\mathbf{u}\|$
PSNR = -259 dB

Ill-posed inverse problems – Definition

Well conditioned problem

A discrete inverse problem is **well conditioned** if a solution

- **exists** : $\mathbf{y} \in \mathrm{Im}(\mathbf{A})$
- is **unique** : $\mathbf{A}$ injective (i.e., $\ker(\mathbf{A}) = \{\mathbf{0}\}$)
- is **stable** to **perturbations**: $\|\mathbf{y}^\delta - \mathbf{y}\| \leq \delta$, then $\|\mathbf{u}^\delta - \mathbf{u}\| \leq C(\delta)$

Ill-posed inverse problems – Definition

Well conditioned problem

A discrete inverse problem is **well conditioned** if a solution

- **exists** : $\mathbf{y} \in \mathrm{Im}(\mathbf{A})$
- is **unique** : $\mathbf{A}$ injective (i.e., $\ker(\mathbf{A}) = \{\mathbf{0}\}$)
- is **stable** to **perturbations**: $\|\mathbf{y}^\delta - \mathbf{y}\| \leq \delta$, then $\|\mathbf{u}^\delta - \mathbf{u}\| \leq C(\delta)$

If one of these three conditions fails $\longrightarrow$ **ill-conditioned problem**.

Least-squares solution

Alternative to non-existence (i.e., $\mathbf{y} \notin \mathrm{Im}(\mathbf{A})$)

$$\mathbf{u}_* \in \arg \min_{\mathbf{u} \in \mathbb{R}^N} \frac{1}{2} \|\mathbf{A}\mathbf{u} - \mathbf{y}\|_2^2.$$

- When $\mathbf{y} \in \mathrm{Im}(\mathbf{A})$, we have $\mathbf{A}\mathbf{u}_* = \mathbf{y}$,
- The least-squares problem **always admits a solution**, whatever $\mathbf{A}$ (also rectangular, $\mathbf{A} \in \mathbb{R}^{M \times N}$) and $\mathbf{y}$
- But the solution **may not be unique** if $\ker(\mathbf{A}) \neq \{\mathbf{0}\}$, since for all $\mathbf{w} \in \ker(\mathbf{A}) \backslash \{\mathbf{0}\}$ and all $\alpha \in \mathbb{R}$, $(\mathbf{u}_* + \alpha \mathbf{w})$ **is solution**.

Least-squares solution

Alternative to non-existence (i.e., $\mathbf{y} \notin \mathrm{Im}(\mathbf{A})$)

$$\mathbf{u}_* \in \arg\min_{\mathbf{u}\in\mathbb{R}^N} \frac{1}{2}\|\mathbf{A}\mathbf{u}-\mathbf{y}\|_2^2.$$

- When $\mathbf{y} \in \mathrm{Im}(\mathbf{A})$, we have $\mathbf{A}\mathbf{u}_* = \mathbf{y}$,
- The least-squares problem **always admits a solution**, whatever $\mathbf{A}$ (also rectangular, $\mathbf{A} \in \mathbb{R}^{M\times N}$) and $\mathbf{y}$
- But the solution **may not be unique** if $\ker(\mathbf{A}) \neq \{\mathbf{0}\}$, since for all $\mathbf{w} \in \ker(\mathbf{A})\backslash\{\mathbf{0}\}$ and all $\alpha \in \mathbb{R}$, $(\mathbf{u}_* + \alpha\mathbf{w})$ **is solution**.

Least-squares solution

Alternative to non-existence (i.e., $\mathbf{y} \notin \mathrm{Im}(\mathbf{A})$)

$$\mathbf{u}_* \in \arg \min_{\mathbf{u} \in \mathbb{R}^N} \frac{1}{2} \|\mathbf{A}\mathbf{u} - \mathbf{y}\|_2^2.$$

- When $\mathbf{y} \in \mathrm{Im}(\mathbf{A})$, we have $\mathbf{A}\mathbf{u}_* = \mathbf{y}$,
- The least-squares problem **always admits a solution**, whatever $\mathbf{A}$ (also rectangular, $\mathbf{A} \in \mathbb{R}^{M \times N}$) and $\mathbf{y}$
- But the solution **may not be unique** if $\ker(\mathbf{A}) \neq \{\mathbf{0}\}$, since for all $\mathbf{w} \in \ker(\mathbf{A}) \backslash \{\mathbf{0}\}$ and all $\alpha \in \mathbb{R}$, $(\mathbf{u}_* + \alpha \mathbf{w})$ **is solution**.

Least-squares solution

Alternative to non-existence (i.e., $\mathbf{y} \notin \mathrm{Im}(\mathbf{A})$)

$$\mathbf{u}_* \in \arg \min_{\mathbf{u} \in \mathbb{R}^N} \frac{1}{2} \|\mathbf{A}\mathbf{u} - \mathbf{y}\|_2^2.$$

- When $\mathbf{y} \in \mathrm{Im}(\mathbf{A})$, we have $\mathbf{A}\mathbf{u}_* = \mathbf{y}$,
- The least-squares problem **always admits a solution**, whatever $\mathbf{A}$ (also rectangular, $\mathbf{A} \in \mathbb{R}^{M \times N}$) and $\mathbf{y}$
- But the solution **may not be unique** if $\ker(\mathbf{A}) \neq \{\mathbf{0}\}$, since for all $\mathbf{w} \in \ker(\mathbf{A}) \backslash \{\mathbf{0}\}$ and all $\alpha \in \mathbb{R}$, $(\mathbf{u}_* + \alpha \mathbf{w})$ **is solution**.

Minimum norm solution

Let $\mathcal{S} \subseteq \mathbb{R}^N$ be the set of least-squares solutions (possibly many values). The **minimum norm solution** is given by

$$\mathbf{u}_* = \arg \min_{\mathbf{u} \in \mathcal{S}} \|\mathbf{u}\|_2,$$

Minimum norm solution

Let $\mathcal{S} \subseteq \mathbb{R}^N$ be the set of least-squares solutions (possibly many values). The **minimum norm solution** is given by

$$\mathbf{u}_* = \arg \min_{\mathbf{u} \in \mathcal{S}} \|\mathbf{u}\|_2,$$

It can be expressed as $\mathbf{u}_* = \mathbf{A}^\dagger \mathbf{y}$, with $\mathbf{A}^\dagger$ **Moore-Penrose pseudo inverse** :

$$\mathbf{A}^\dagger = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^* \tag{1}$$

where $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$ is a SVD of $\mathbf{A}$.

Note that here we consider the "compact" SVD

- $\mathbf{U} \in \mathbb{R}^{M \times R}$ (with $\mathbf{U}^*\mathbf{U} = \mathbf{I}_R$);
- $\mathbf{V} \in \mathbb{R}^{N \times R}$ (with $\mathbf{V}^*\mathbf{V} = \mathbf{I}_R$);
- $\mathbf{\Sigma} = \mathbf{diag}\left(\sigma_1(\mathbf{A}) \cdots \sigma_R(\mathbf{A})\right) \in \mathbb{R}_+^{R \times R}$ (with $\sigma_1(\mathbf{A}) \geq \cdots \geq \sigma_R(\mathbf{A}) > 0$).

Stability and conditioning

Let denote

- $\bar{\mathbf{y}} = \mathbf{A}\bar{\mathbf{u}}$ the noiseless data,
- $\bar{\mathbf{u}}$ the associated minimal norm solution ($\bar{\mathbf{u}} = \mathbf{A}^{\dagger}\bar{\mathbf{y}}$),
- $\mathbf{y} = \mathbf{A}\bar{\mathbf{u}} + \mathbf{n}$ the noisy data.

Stability and conditioning

Let denote

- $\bar{\mathbf{y}} = \mathbf{A}\bar{\mathbf{u}}$ the noiseless data,
- $\bar{\mathbf{u}}$ the associated minimal norm solution ($\bar{\mathbf{u}} = \mathbf{A}^\dagger \bar{\mathbf{y}}$),
- $\mathbf{y} = \mathbf{A}\bar{\mathbf{u}} + \mathbf{n}$ the noisy data.

Then we have

$$\|\bar{\mathbf{y}}\| = \|\mathbf{A}\bar{\mathbf{u}}\| \leq \|\mathbf{A}\|\|\bar{\mathbf{u}}\|$$
$$\|\mathbf{u}_* - \bar{\mathbf{u}}\| = \|\mathbf{A}^\dagger \mathbf{y} - \mathbf{A}^\dagger \bar{\mathbf{y}}\| \leq \|\mathbf{A}^\dagger\|\|\mathbf{y} - \bar{\mathbf{y}}\| = \|\mathbf{A}^\dagger\|\|\mathbf{n}\|,$$

Stability and conditioning

Let denote

- $\bar{\mathbf{y}} = \mathbf{A}\bar{\mathbf{u}}$ the noiseless data,
- $\bar{\mathbf{u}}$ the associated minimal norm solution ($\bar{\mathbf{u}} = \mathbf{A}^\dagger \bar{\mathbf{y}}$),
- $\mathbf{y} = \mathbf{A}\bar{\mathbf{u}} + \mathbf{n}$ the noisy data.

Then we have

$$\|\bar{\mathbf{y}}\| = \|\mathbf{A}\bar{\mathbf{u}}\| \leq \|\mathbf{A}\|\|\bar{\mathbf{u}}\|$$
$$\|\mathbf{u}_* - \bar{\mathbf{u}}\| = \|\mathbf{A}^\dagger \mathbf{y} - \mathbf{A}^\dagger \bar{\mathbf{y}}\| \leq \|\mathbf{A}^\dagger\|\|\mathbf{y} - \bar{\mathbf{y}}\| = \|\mathbf{A}^\dagger\|\|\mathbf{n}\|,$$

From which we get

$$\frac{\|\mathbf{u}_* - \bar{\mathbf{u}}\|}{\|\bar{\mathbf{u}}\|} \leq \kappa(\mathbf{A}) \frac{\|\mathbf{n}\|}{\|\bar{\mathbf{y}}\|},$$

with $\kappa(\mathbf{A})$ the **condition number** of $\mathbf{A}$,

$$\kappa(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^\dagger\| = \frac{\sigma_1(\mathbf{A})}{\sigma_R(\mathbf{A})},$$

which could be potentially $\gg 1$ if $\sigma_R(\mathbf{A})$ is small...

Outline

Practical information

From continuous to discretised image formation models

Ill-posedness

Maximum likelihood, MAP and MMSE

Regularisation: from priors to discrete models

Maximum likelihood, MAP and MMSE

Bayesian formulation

Bayesian formalism

- $\mathbf{u}$ is a realisation of a random vector $\mathbb{u}$ with distribution $\pi_{\mathbb{u}}$
- $\mathbf{y}$ is a realisation of the random vector $\mathbb{y} = \mathcal{P}(\mathbf{A}\mathbb{u})$
- **Posterior distribution**[a]: $\pi_{\mathbb{u}|\mathbb{y}}(\mathbf{u}|\mathbf{y}) \propto \pi_{\mathbb{y}|\mathbb{u}}(\mathbf{y}|\mathbf{u})\pi_{\mathbb{u}}(\mathbf{u})$

[a] Stuart, Taeb, and Sanz-Alonso, '23

Bayesian formulation

Bayesian formalism

- $\mathbf{u}$ is a realisation of a random vector $\mathbb{u}$ with distribution $\pi_{\mathbb{u}}$
- $\mathbf{y}$ is a realisation of the random vector $\mathbb{y} = \mathcal{P}(\mathbf{A}\mathbb{u})$
- **Posterior distribution**[a]: $\pi_{\mathbb{u}|\mathbb{y}}(\mathbf{u}|\mathbf{y}) \propto \pi_{\mathbb{y}|\mathbb{u}}(\mathbf{y}|\mathbf{u})\pi_{\mathbb{u}}(\mathbf{u})$

[a] Stuart, Taeb, and Sanz-Alonso, '23

Maximum A Posteriori (MAP) estimator

$$\hat{\mathbf{u}}_{\mathrm{MAP}}(\mathbf{y}) \overset{\mathrm{def}}{=} \arg\max_{\mathbf{u}} \pi_{\mathbb{u}|\mathbb{y}}(\mathbf{u}|\mathbf{y})$$

$\longrightarrow$ *Best* **point** *estimator*

Bayesian formulation

Bayesian formalism

- $\mathbf{u}$ is a realisation of a random vector $\mathbb{u}$ with distribution $\pi_{\mathbb{u}}$
- $\mathbf{y}$ is a realisation of the random vector $\mathbb{y} = \mathcal{P}(\mathbf{A}\mathbb{u})$
- **Posterior distribution**[a]: $\pi_{\mathbb{u}|\mathbb{y}}(\mathbf{u}|\mathbf{y}) \propto \pi_{\mathbb{y}|\mathbb{u}}(\mathbf{y}|\mathbf{u})\pi_{\mathbb{u}}(\mathbf{u})$

[a] Stuart, Taeb, and Sanz-Alonso, '23

Maximum A Posteriori (MAP) estimator

$$\hat{\mathbf{u}}_{\mathrm{MAP}}(\mathbf{y}) \overset{\mathrm{def}}{=} \arg\max_{\mathbf{u}} \pi_{\mathbb{u}|\mathbb{y}}(\mathbf{u}|\mathbf{y})$$

$\longrightarrow$ *Best* **point** *estimator*

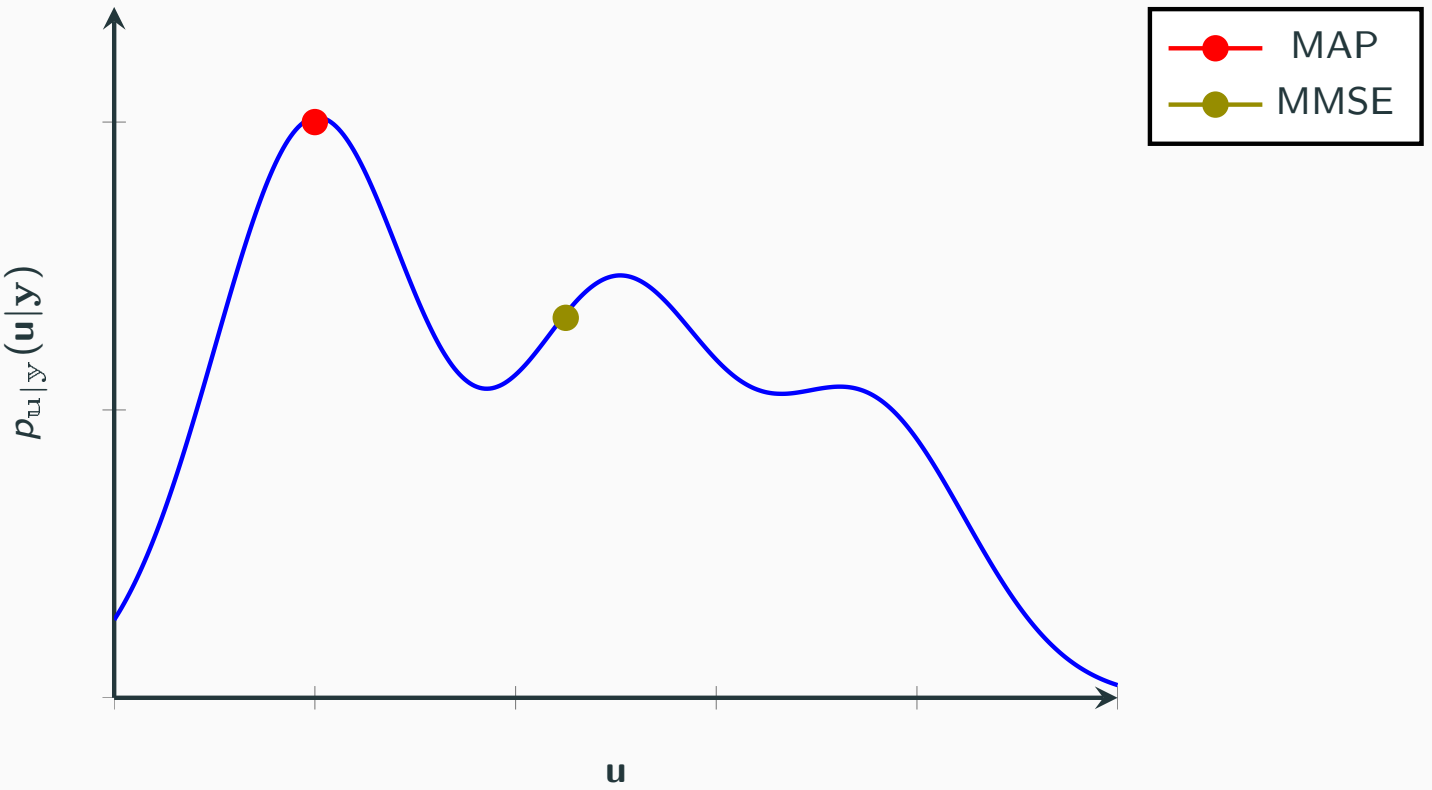
Minimum Mean Square Error (MMSE) estimator

$$\hat{\mathbf{u}}_{\mathrm{MMSE}} \overset{\mathrm{def}}{=} \arg\min_{\phi} \mathbb{E}_{(\mathbb{u},\mathbb{y})}\left[\|\phi(\mathbb{y}) - \mathbb{u}\|_2^2\right]$$

There holds: $\hat{\mathbf{u}}_{\mathrm{MMSE}}(\mathbf{y}) = \mathbb{E}\left[\mathbb{u}|\mathbb{y} = \mathbf{y}\right] = \int \mathbf{u}\pi_{\mathbb{u}|\mathbb{y}}(\mathbf{u}|\mathbf{y})\, d\mathbf{u}$

$\longrightarrow$ *Best estimator* **in average** *(according to posterior distribution)*

Bayesian formulation



Bayesian formulation – Parametric statistic viewpoint

Definitions and assumptions

- $\mathcal{P} \equiv P_\theta : \mathbb{R}^M \to \mathbb{R}^M$ is a **stochastic perturbation** parametrized by $\theta \in \mathbb{R}^s$
- $\mathbf{y}$ is a realization of a random vector with **law** $\pi_{\mathbb{y}}$ and parameters $(\theta, \mathbf{Au})$
- $\mathbf{u}$ is a realization of a random vector from the **a priori** law $\pi_{\mathbb{u}}$.

Definitions and assumptions

- $\mathcal{P} \equiv P_\theta : \mathbb{R}^M \to \mathbb{R}^M$ is a **stochastic perturbation** parametrized by $\theta \in \mathbb{R}^s$
- $\mathbf{y}$ is a realization of a random vector with **law** $\pi_{\mathbb{y}}$ and parameters $(\theta, \mathbf{Au})$
- $\mathbf{u}$ is a realization of a random vector from the **a priori** law $\pi_{\mathbb{u}}$.

White Gaussian noise	$\mathbf{y} = \mathbf{Au} + \mathbf{n}$	$\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma_\eta^2 \mathbf{I}_M)$	$\theta = \sigma_\eta$
Correlated Gaussian noise	$\mathbf{y} = \mathbf{Au} + \mathbf{n}$	$\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_\eta)$	$\theta = \mathbf{\Sigma}_\eta$
Poisson noise	$\mathbf{y} \sim \mathcal{P}(\mathbf{Au})$		
Poisson-Gaussian noise	$\mathbf{y} = \mathbf{z} + \mathbf{n}$	$\mathbf{z} \sim \mathcal{P}(\mathbf{Au})$ $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma_\eta^2 \mathbf{I})$	$\theta = \sigma_\eta$

Table 4: Examples of parametric stochastic perturbations P_θ

Bayesian formulation – Parametric statistic viewpoint

Maximum A Posteriori (MAP)

Maximize the probability of having $\mathbf{u}$ knowing $\mathbf{y}$ (posterior law),

$$\mathbf{u}_* \in \arg\max_{\mathbf{u}\in\mathbb{R}^N} \pi(\mathbf{u}|\mathbf{y})$$

Maximum A Posteriori (MAP)

Maximize the probability of having $\mathbf{u}$ knowing $\mathbf{y}$ (posterior law),

$$\mathbf{u}_* \in \arg\max_{\mathbf{u}\in\mathbb{R}^N} \pi(\mathbf{u}|\mathbf{y})$$

From Bayes formula $\pi(\mathbf{u}|\mathbf{y}) = \pi(\mathbf{y}|\mathbf{u})\pi(\mathbf{u})/\pi(\mathbf{y})$ and taking the negative log,

$$\mathbf{u}_* \in \arg\min_{\mathbf{u}\in\mathbb{R}^N} -\log\pi(\mathbf{y}|\mathbf{u}) - \log\pi(\mathbf{u})$$

- the **data fidelity** term $D(\mathbf{Au}, \mathbf{y}) \propto -\log\pi(\mathbf{y}|\mathbf{u})$ associated to the statistic of the perturbations
- the **regularization** term $R(\mathbf{u}) \propto -\log\pi(\mathbf{u})$ governed by the **a priori** distribution imposed on $\mathbf{u}$ (e.g., Gaussian law $\rightarrow$ quadratic)

Bayesian formulation – Examples of likelihood functions

Gaussian noise

$\mathcal{P}$ models an additive Gaussian noise $\mathbf{y} = \mathbf{A}\mathbf{u} + \mathbf{n}$ where $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$.

Bayesian formulation – Examples of likelihood functions

Gaussian noise

$\mathcal{P}$ models an additive Gaussian noise $\mathbf{y} = \mathbf{A}\mathbf{u} + \mathbf{n}$ where $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$.
Hence, $\mathbf{y} \sim \mathcal{N}(\mathbf{A}\mathbf{u}, \mathbf{\Sigma})$ and the likelihood function is given by

$$\pi(\mathbf{y}|\mathbf{u}) = \frac{1}{(2\pi)^{M/2}\det(\mathbf{\Sigma})^{1/2}} \mathrm{e}^{-\frac{1}{2}(\mathbf{y}-\mathbf{A}\mathbf{u})^T\mathbf{\Sigma}^{-1}(\mathbf{y}-\mathbf{A}\mathbf{u})}$$

Bayesian formulation – Examples of likelihood functions

Gaussian noise

$\mathcal{P}$ models an additive Gaussian noise $\mathbf{y} = \mathbf{A}\mathbf{u} + \mathbf{n}$ where $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$.
Hence, $\mathbf{y} \sim \mathcal{N}(\mathbf{A}\mathbf{u}, \boldsymbol{\Sigma})$ and the likelihood function is given by

$$\pi(\mathbf{y}|\mathbf{u}) = \frac{1}{(2\pi)^{M/2} \det(\boldsymbol{\Sigma})^{1/2}} \mathrm{e}^{-\frac{1}{2}(\mathbf{y}-\mathbf{A}\mathbf{u})^T \boldsymbol{\Sigma}^{-1}(\mathbf{y}-\mathbf{A}\mathbf{u})}$$

Data fidelity term

$$D(\mathbf{A}\mathbf{u}, \mathbf{y}) = \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{u}\|^2_{\boldsymbol{\Sigma}^{-1}}$$

where $\|\cdot\|^2_{\mathbf{W}} = \langle \mathbf{W}\cdot, \cdot \rangle$.

Bayesian formulation – Examples of likelihood functions

Gaussian noise

$\mathcal{P}$ models an additive Gaussian noise $\mathbf{y} = \mathbf{A}\mathbf{u} + \mathbf{n}$ where $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$.
Hence, $\mathbf{y} \sim \mathcal{N}(\mathbf{A}\mathbf{u}, \boldsymbol{\Sigma})$ and the likelihood function is given by

$$\pi(\mathbf{y}|\mathbf{u}) = \frac{1}{(2\pi)^{M/2} \det(\boldsymbol{\Sigma})^{1/2}} \mathrm{e}^{-\frac{1}{2}(\mathbf{y}-\mathbf{A}\mathbf{u})^T \boldsymbol{\Sigma}^{-1}(\mathbf{y}-\mathbf{A}\mathbf{u})}$$

Data fidelity term

$$D(\mathbf{A}\mathbf{u}, \mathbf{y}) = \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{u}\|^2_{\boldsymbol{\Sigma}^{-1}}$$

where $\|\cdot\|^2_{\mathbf{W}} = \langle \mathbf{W}\cdot, \cdot \rangle$.

Case of **white Gaussian noise**, $\boldsymbol{\Sigma} = \sigma^2 \mathbf{I}_M$ and $D(\mathbf{A}\mathbf{u}, \mathbf{y}) = \frac{1}{2\sigma^2}\|\mathbf{y} - \mathbf{A}\mathbf{u}\|_2^2$.

Bayesian formulation – Examples of likelihood functions

Gaussian noise

$\mathcal{P}$ models an additive Gaussian noise $\mathbf{y} = \mathbf{A}\mathbf{u} + \mathbf{n}$ where $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$.
Hence, $\mathbf{y} \sim \mathcal{N}(\mathbf{A}\mathbf{u}, \boldsymbol{\Sigma})$ and the likelihood function is given by

$$\pi(\mathbf{y}|\mathbf{u}) = \frac{1}{(2\pi)^{M/2} \det(\boldsymbol{\Sigma})^{1/2}} \mathrm{e}^{-\frac{1}{2}(\mathbf{y}-\mathbf{A}\mathbf{u})^T \boldsymbol{\Sigma}^{-1}(\mathbf{y}-\mathbf{A}\mathbf{u})}$$

Data fidelity term

$$D(\mathbf{A}\mathbf{u}, \mathbf{y}) = \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{u}\|^2_{\boldsymbol{\Sigma}^{-1}}$$

where $\|\cdot\|^2_{\mathbf{W}} = \langle \mathbf{W}\cdot, \cdot \rangle$.

Case of **white Gaussian noise**, $\boldsymbol{\Sigma} = \sigma^2 \mathbf{I}_M$ and $D(\mathbf{A}\mathbf{u}, \mathbf{y}) = \frac{1}{2\sigma^2}\|\mathbf{y} - \mathbf{A}\mathbf{u}\|_2^2$.

Properties of $D(\mathbf{A}\cdot, \mathbf{y})$

- infinitely differentiable
- convex
- L-Lipschitz gradient with $L = \|\mathbf{A}^*\mathbf{A}\|$.

Bayesian formulation – Examples of likelihood functions

Poisson noise

$\mathcal{P}$ models a Poisson noise[a] (counting process) $\mathbf{y} \sim \mathcal{P}(\mathbf{Au})$ **for u s.t.** $\mathbf{Au} > 0$.

[a] Bertero, Boccacci, '98

Poisson noise

$\mathcal{P}$ models a Poisson noise[a] (counting process) $\mathbf{y} \sim \mathcal{P}(\mathbf{Au})$ **for u s.t.** $\mathbf{Au} > 0$.

The likelihood function is given by

$$\pi(\mathbf{y}|\mathbf{u}) = \prod_{m \in [M]} \frac{([\mathbf{Au}]_m)^{y_m} \mathrm{e}^{-[\mathbf{Au}]_m}}{y_m!}$$

[a] Bertero, Boccacci, '98

Poisson noise

$\mathcal{P}$ models a Poisson noise[a] (counting process) $\mathbf{y} \sim \mathcal{P}(\mathbf{Au})$ **for u s.t.** $\mathbf{Au} > 0$.

The likelihood function is given by

$$\pi(\mathbf{y}|\mathbf{u}) = \prod_{m \in [M]} \frac{([\mathbf{Au}]_m)^{y_m} \mathrm{e}^{-[\mathbf{Au}]_m}}{y_m!}$$

[a] Bertero, Boccacci, '98

Data fidelity term

$$D(\mathbf{Au}, \mathbf{y}) = \mathrm{KL}(\mathbf{Au}, \mathbf{y}) = \sum_{m \in [M]} (\mathbf{Au})_m - y_m \log((\mathbf{Au})_m)$$

known as the **Kullback-Leibler** divergence: not Lipschitz-smooth in $\mathbf{0}$.

Poisson noise

$\mathcal{P}$ models a Poisson noise[a] (counting process) $\mathbf{y} \sim \mathcal{P}(\mathbf{Au})$ **for u s.t.** $\mathbf{Au} > 0$.

The likelihood function is given by

$$\pi(\mathbf{y}|\mathbf{u}) = \prod_{m \in [M]} \frac{([\mathbf{Au}]_m)^{y_m} \mathrm{e}^{-[\mathbf{Au}]_m}}{y_m!}$$

[a] Bertero, Boccacci, '98

Data fidelity term

$$D(\mathbf{Au}, \mathbf{y}) = \mathrm{KL}(\mathbf{Au}, \mathbf{y}) = \sum_{m \in [M]} (\mathbf{Au})_m - y_m \log((\mathbf{Au})_m)$$

known as the **Kullback-Leibler** divergence: not Lipschitz-smooth in $\mathbf{0}$.

Smoothed version: $\mathrm{KL}(\mathbf{Au} + \boldsymbol{\epsilon}, \mathbf{y})$ where $\boldsymbol{\epsilon} \in \mathbb{R}^M_{>0}$.

Poisson noise

$\mathcal{P}$ models a Poisson noise[a] (counting process) $\mathbf{y} \sim \mathcal{P}(\mathbf{Au})$ **for u s.t.** $\mathbf{Au} > 0$.

The likelihood function is given by

$$\pi(\mathbf{y}|\mathbf{u}) = \prod_{m \in [M]} \frac{([\mathbf{Au}]_m)^{y_m} \mathrm{e}^{-[\mathbf{Au}]_m}}{y_m!}$$

[a] Bertero, Boccacci, '98

Data fidelity term

$$D(\mathbf{Au}, \mathbf{y}) = \mathrm{KL}(\mathbf{Au}, \mathbf{y}) = \sum_{m \in [M]} (\mathbf{Au})_m - y_m \log((\mathbf{Au})_m)$$

known as the **Kullback-Leibler** divergence: not Lipschitz-smooth in $\mathbf{0}$.

Smoothed version: $\mathrm{KL}(\mathbf{Au} + \boldsymbol{\epsilon}, \mathbf{y})$ where $\boldsymbol{\epsilon} \in \mathbb{R}^M_{>0}$.

Properties of $\mathrm{KL}(\mathbf{A} \cdot + \boldsymbol{\epsilon}, \mathbf{y})$ **on** $\mathbb{R}^N_{\geq 0}$

- infinitely differentiable
- convex
- L-Lipschitz gradient with $L = \|\mathbf{A}\|^2 \max_m y_m / \epsilon_m^2$

Poisson denoising

$$\mathbf{y} = \mathcal{P}(\alpha \mathbf{u} / \|\mathbf{u}\|_\infty)$$

where $\alpha > 0$ is an estimate of the expected number of photons.

Standard quality metrics:

$$\mathrm{MSE}(\mathbf{u}, \widehat{\mathbf{u}}) = \frac{1}{N} \sum_{n=1}^{N} (u_n - \widehat{u}_n)^2 , \qquad \in [0, +\infty) \downarrow$$

$$\mathrm{PSNR}(\mathbf{u}, \widehat{\mathbf{u}}) = 10 \log_{10} \left(\frac{u_{\max}^2}{\mathrm{MSE}(\mathbf{u}, \widehat{\mathbf{u}})} \right) , \qquad \in (-\infty, +\infty) \uparrow$$

$$\mathrm{SSIM}\,[1](\mathbf{u}, \widehat{\mathbf{u}}) = \frac{(2\mu_{\mathbf{u}}\mu_{\widehat{\mathbf{u}}} + C_1)(2\sigma_{\mathbf{u}\widehat{\mathbf{u}}} + C_2)}{(\mu_{\mathbf{u}}^2 + \mu_{\widehat{\mathbf{u}}}^2 + C_1)(\sigma_{\mathbf{u}}^2 + \sigma_{\widehat{\mathbf{u}}}^2 + C_2)}, \qquad \in [0, 1] \uparrow$$

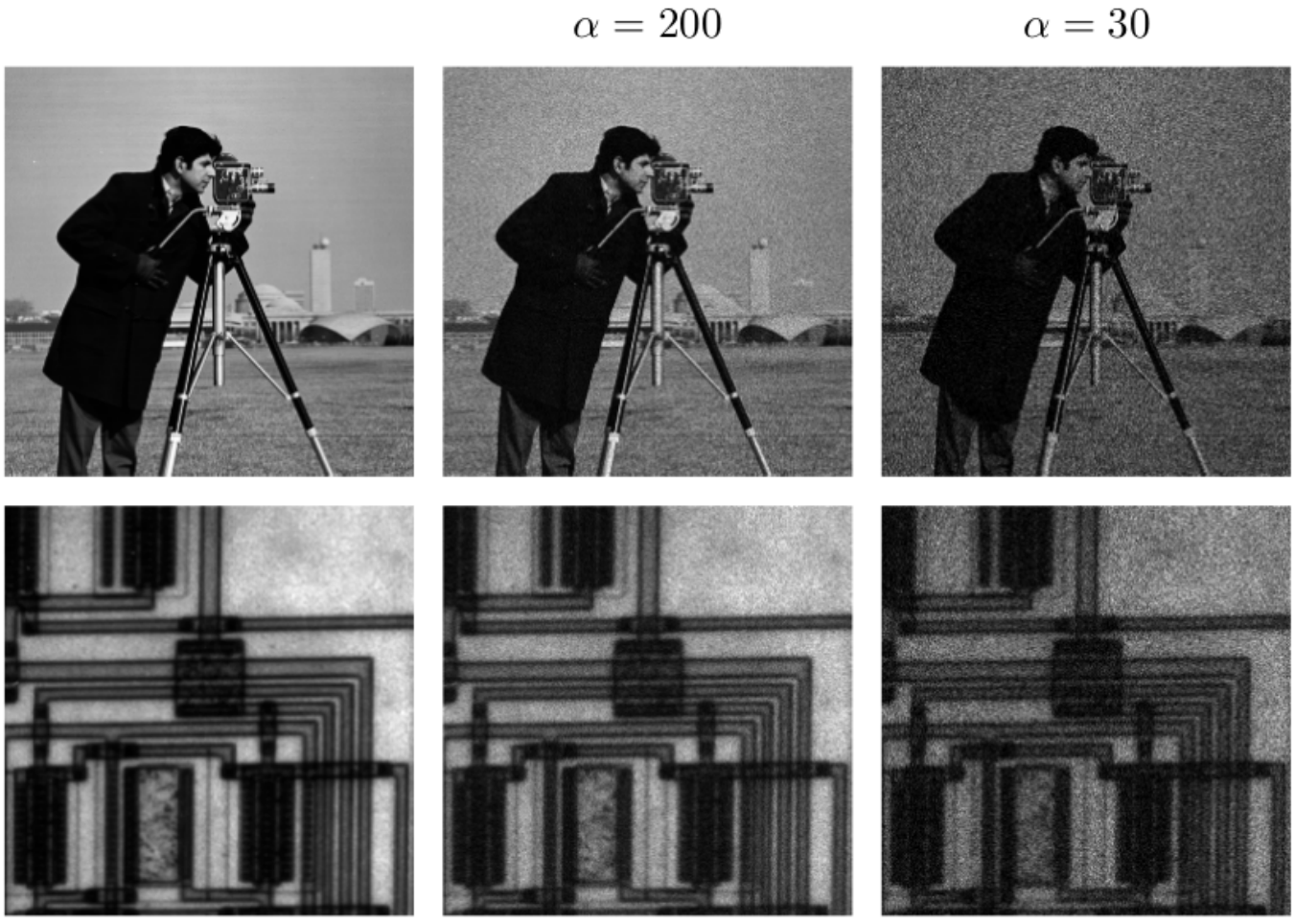
[1] Wang, Bovik, Sheikh, Simoncelli, '01

Bayesian formulation – Numerical examples

Poisson denoising

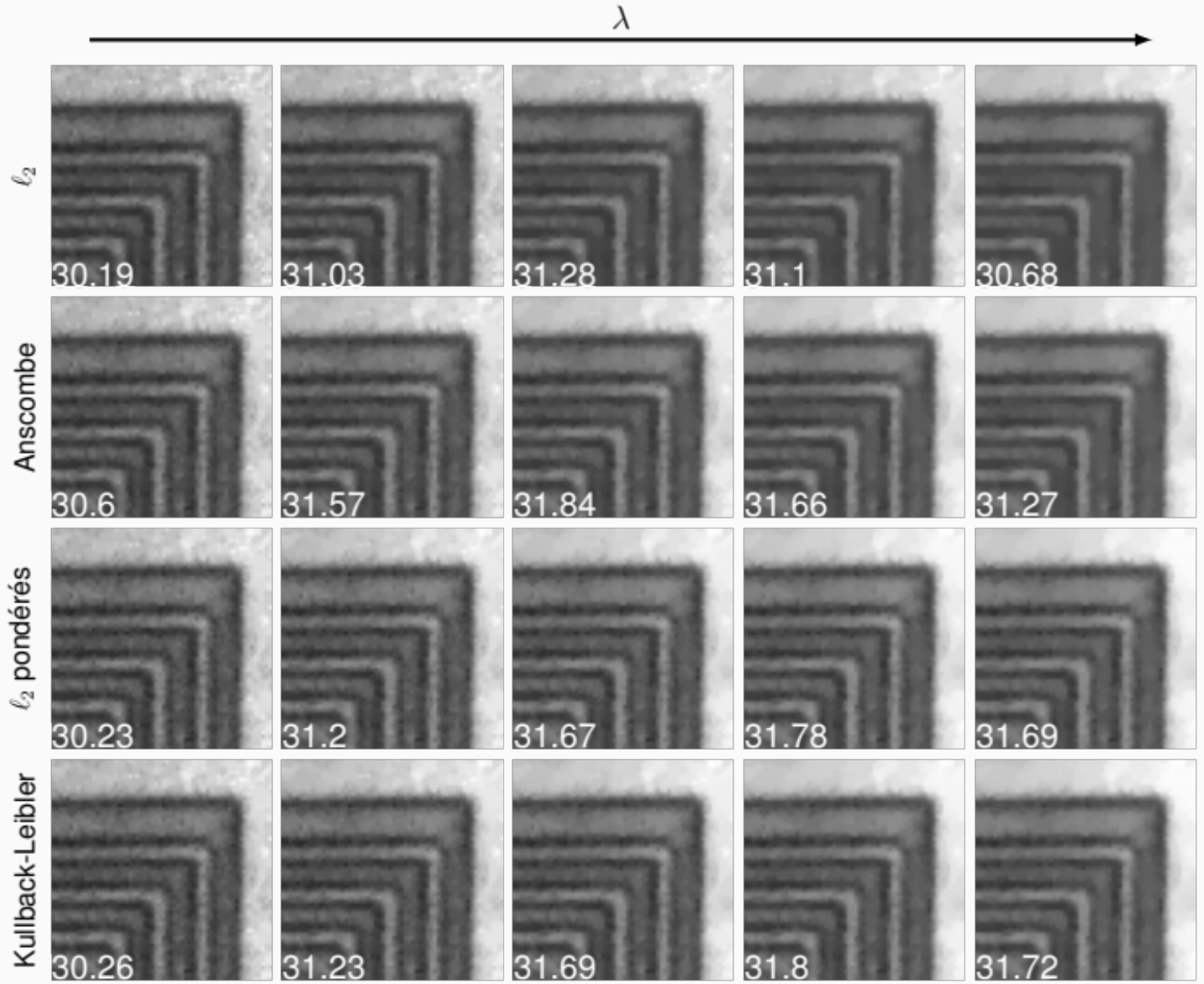
$$\mathbf{y} = \mathcal{P}(\alpha \mathbf{u} / \|\mathbf{u}\|_\infty)$$

where $\alpha > 0$ is an estimate of the expected number of photons.



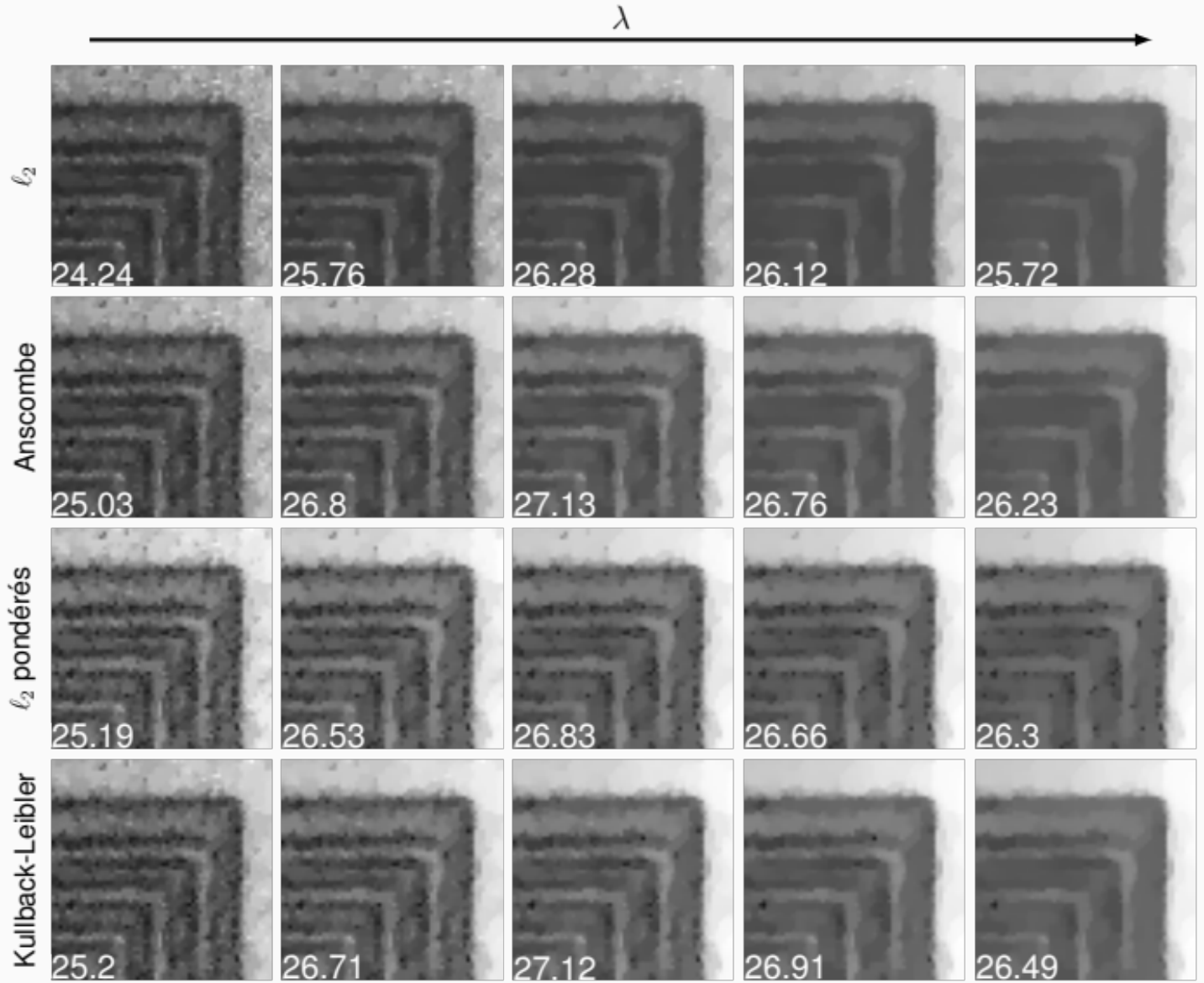
Bayesian formulation – Numerical examples

Bayesian formulation – Numerical examples



Bayesian formulation – Numerical examples

Bayesian formulation – Numerical examples



Outline

Practical information

From continuous to discretised image formation models

Ill-posedness

Maximum likelihood, MAP and MMSE

Regularisation: from priors to discrete models

Regularisation: from priors to discrete models

Regularisation idea– Why regularisation?

Inverse problems need prior information

Given measurements

$$\mathbf{y} \simeq SHu,$$

lack of injectivity: many objects may explain the same data.

Regularisation: among all plausible reconstructions, select the one that has desirable image features.

Generic variational model

$$u_* \in \arg\min_u D(SHu, \mathbf{y}) + \lambda R(u).$$

- $D(SHu, \mathbf{y})$: data fit with measurements.
- $R(u)$: prior knowledge on the image (smoothness, regularity...).
- $\lambda > 0$: tradeoff between data fidelity and prior.

Examples of priors

- small energy: $R(u) = \|u\|_2^2$,
- smoothness: $R(u) = \|\nabla u\|_2^2$,
- piecewise-constant structure: $R(u) = \mathrm{TV}(u)$,
- sparsity: $R(u) = \|u\|_1$.

Regularisation idea – Representer theorems (intuition)

Informal idea

A representer theorem says that, although the unknown image may live in a very large space, the regularised solution has a simple structure.

regularisation restricts the shape of admissible solutions.

Typical result

The solution has the following structure[a]:

$$u_* \approx \sum_{m=1}^{M} \alpha_m f_m.$$

- The coefficients α_m come from the data.
- The functions f_m depend on $\mathbf{y}$ and R

[a] Bredies, Carioni, '20

Why useful?

- It explains what kind of images a regulariser prefers.
- It suggests how to discretise the continuous model.

Regularisation idea – Structure of solutions for typical regularisers

Regulariser	Preferred structure
$\|u\|_2^2$	small-energy image
$\|\nabla u\|_2^2$	smooth image
$\mathrm{TV}(u)$	piecewise-smooth / edge-preserving image
$\|\alpha\|_1$	few non-zeros

Intuition

- Quadratic penalties spread energy smoothly.
- TV allows jumps and therefore preserves edges.
- ℓ^1 penalties select only a few active components.

Regularisation idea – Quadratic regularisation

Model

$$u_* \in \arg\min_u D(SHu, \mathbf{y}) + \frac{\lambda}{2}\|Lu\|_2^2,$$

with $L : \mathcal{U} \to \mathcal{U}'$. Typical choices are:

$$L = \mathbf{Id} \qquad \text{or} \qquad L = \nabla.$$

The resulting solution is usually smooth and globally distributed.

- $L = \mathbf{Id}$: penalises large image values.
- $L = \nabla$: penalises large variations.

Consequence

Quadratic regularisation tends to oversmooth sharp transitions, but it is convenient.

Regularisation idea – Discretised formulation

Discrete deconvolution problem

- $\mathbf{y} = \mathbf{A}\mathbf{u} + \mathbf{n}$
- $\mathbf{A} \in \mathbb{R}^{N \times N}$ convolution with kernel $\mathbf{h}$
- $\mathbf{n}$ Gaussian noise

Least-squares + quadratic (so-called Tikhonov/Sobolev) regularisation:

$$\mathbf{u}_* \in \arg \min_{\mathbf{u} \in \mathbb{R}^N} \frac{1}{2} \|\mathbf{A}\mathbf{u} - \mathbf{y}\|_2^2 + \frac{\lambda}{2} \|\mathbf{D}\mathbf{u}\|_2^2.$$

Two basic choices

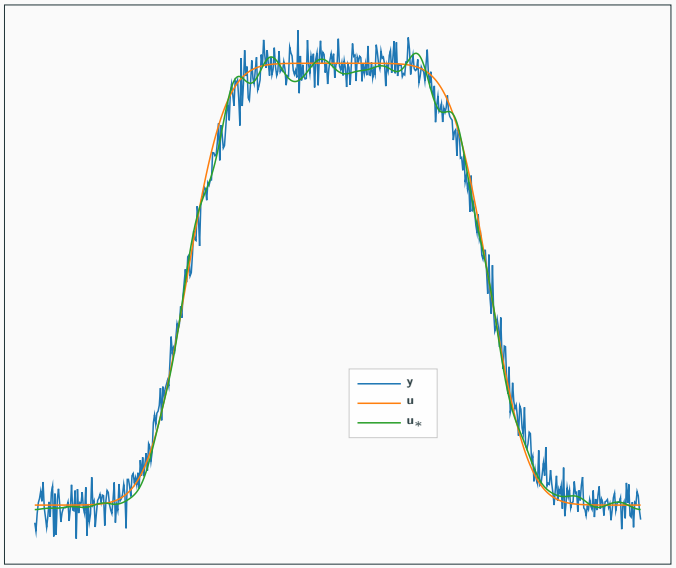
- $\mathbf{D} = \mathrm{Id}$: penalise the image itself.
- $\mathbf{D} = \nabla$: penalise finite differences, i.e. discrete gradients.

Regularisation idea – Tikhonov with $\mathbf{D} = \mathrm{Id}$

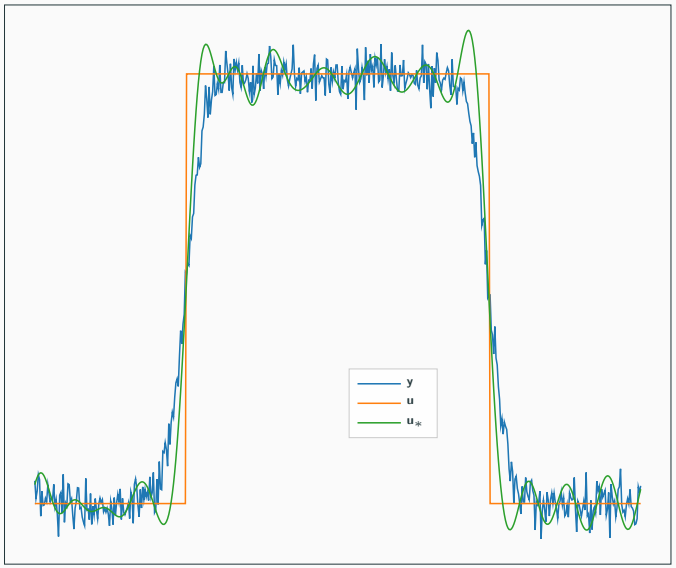
Interpretation

$$\mathbf{u}_* \in \arg\min_{\mathbf{u}\in\mathbb{R}^N} \frac{1}{2}\|\mathbf{A}\mathbf{u} - \mathbf{y}\|_2^2 + \frac{\lambda}{2}\|\mathbf{u}\|_2^2.$$

- Keeps the reconstruction energy small.
- Stabilises the inversion.
- Does not explicitly preserve edges.

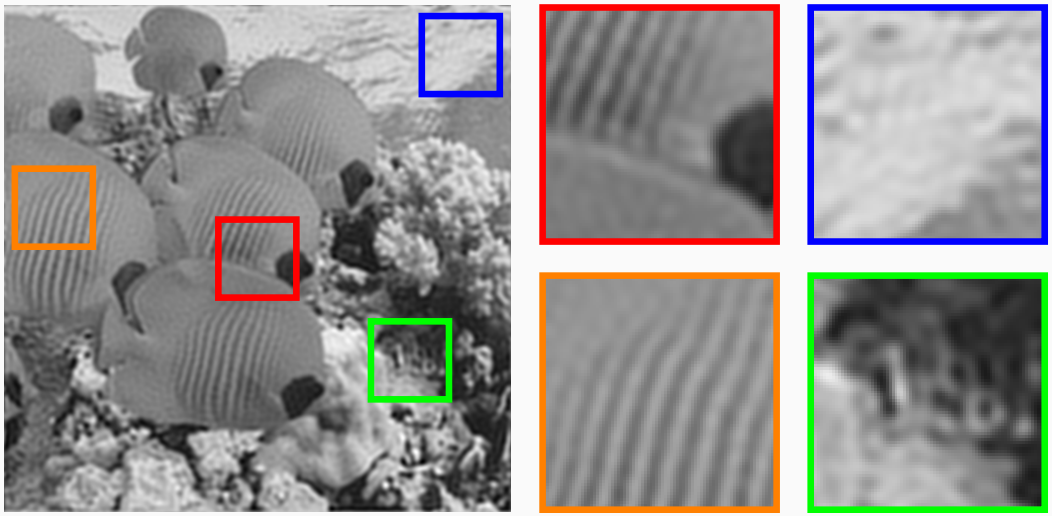


$\mathrm{PSNR}(\mathbf{u}_*) = 34.9\mathrm{dB}$; $\lambda = 0.01$



$\mathrm{PSNR}(\mathbf{u}_*) = 21.2\mathrm{dB}$; $\lambda = 0.005$

Regularisation idea – Tikhonov with $\mathbf{D} = \mathrm{Id}$: image example



$\mathbf{u}_*$ $(\mathbf{D} = \mathbf{Id})$ with $\sigma = 1\%$, PSNR $= 24.7$ dB and $\lambda = 10^{-2}$.

The reconstruction is stabilised, but sharp details may be smoothed.

Regularisation idea – Tikhonov/Sobolev regularisation, $\mathsf{D} = \nabla$

Gradient regularisation

$$\mathbf{u}_* \in \arg\min_{\mathbf{u}\in\mathbb{R}^N} \frac{1}{2}\|\mathbf{A}\mathbf{u} - \mathbf{y}\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{\nabla}\mathbf{u}\|_2^2.$$

- Penalises oscillations.
- Encourages smooth reconstructions.
- Strong discontinuities are still penalised.

Finite-difference discretisation

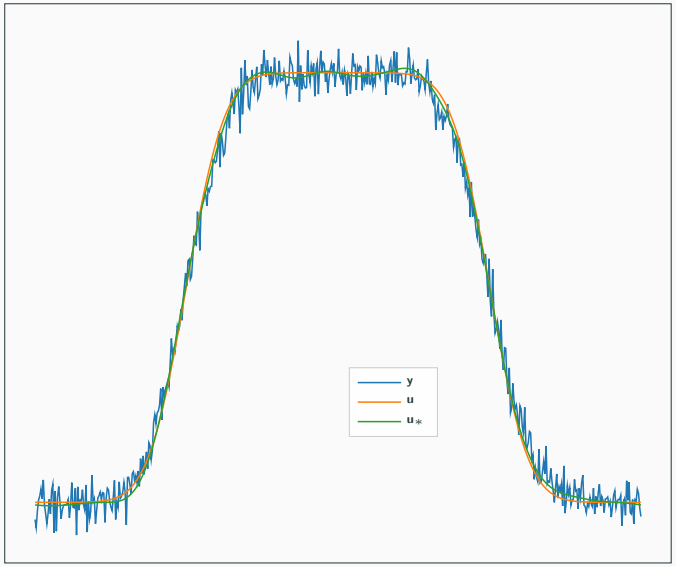
For a pixel grid,

$$\partial_1 u(\mathbf{x_n}) \simeq \mathbf{u}[\mathbf{n} + \mathbf{e}_1] - \mathbf{u}[\mathbf{n}] = \partial_1 \mathbf{u}[\mathbf{n}].$$

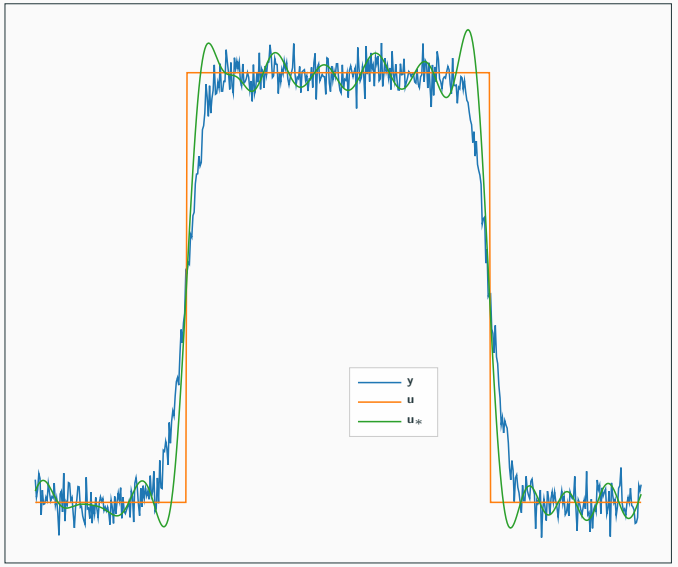
Thus,

$$\|\nabla u\|_2^2 \quad \rightsquigarrow \quad \sum_{\mathbf{n}} \|\boldsymbol{\nabla}\mathbf{u}[\mathbf{n}]\|_2^2.$$

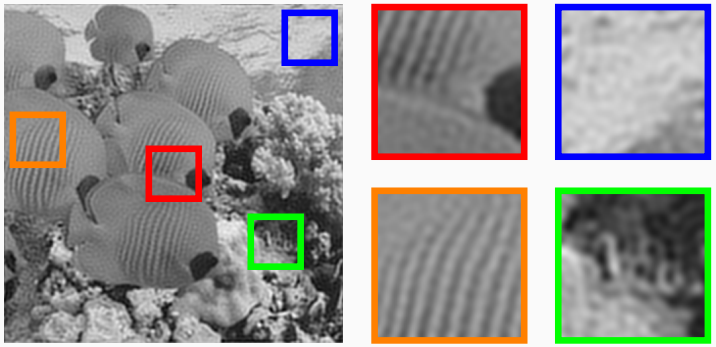
Regularisation idea – Tikhonov with $\mathbf{D} = \nabla$: examples



$\text{PSNR}(\mathbf{u}_*) = 40.7\text{dB}$; $\lambda = 10$



$\text{PSNR}(\mathbf{u}_*) = 21.3\text{dB}$; $\lambda = 0.3$



$\mathbf{u}_*$ $(\mathbf{D} = \boldsymbol{\nabla})$ with $\sigma = 1\%$, PSNR $= 24.6$ dB and $\lambda = 5.10^{-3}$.

Motivation

Natural images often contain:

- homogeneous regions,
- edges and contours.

Quadratic smoothness penalises edges too strongly.

TV regularisation

Instead of penalising squared gradients, TV penalises gradient magnitudes[a]:

$$\mathrm{TV}(\mathbf{u}) = \sum_{\mathbf{n}} \|\boldsymbol{\nabla}\mathbf{u}[\mathbf{n}]\|_2.$$

[a] Ruder, Osher, Fatemi, '92

Intuition

TV allows a few large gradients, hence it can preserve edges.

Total variation – TV representer

Preferred structure for TV solutions

$$\text{TV prior} \quad \Longrightarrow \quad \text{favours piecewise-constant images.}$$

- Flat regions are cheap.
- Edges are allowed.
- Too many oscillations are penalised.

Total variation – Drawback: staircasing effect

Remarks

- TV is edge-preserving.
- It promotes sparse gradients.
- It may transform smooth transitions into piecewise-constant plateaus.

Total variation – TV discretisation

Discrete image model

For a discrete image $\mathbf{u}$ represented on a pixel grid:

$$\mathbf{u} \in \mathbb{R}^N, \qquad N = N_1 \times N_2,$$

The discrete gradient is computed by finite differences:

$$\boldsymbol{\nabla} \mathbf{u}[\mathbf{n}] = (\partial_1 \mathbf{u}[\mathbf{n}], \partial_2 \mathbf{u}[\mathbf{n}]) \,.$$

Discrete total variation

$$\mathrm{TV}(\mathbf{u}) = \sum_{\mathbf{n} \in [N]} \|\boldsymbol{\nabla} \mathbf{u}[\mathbf{n}]\|_2 .$$

Total variation – TV deconvolution

In the deconvolution problem

- $\mathbf{y} = \mathbf{A}\mathbf{u} + \mathbf{n}$
- $\mathbf{A} \in \mathbb{R}^{N \times N}$ convolution with kernel $\mathbf{h}$
- $\mathbf{n}$ Gaussian noise

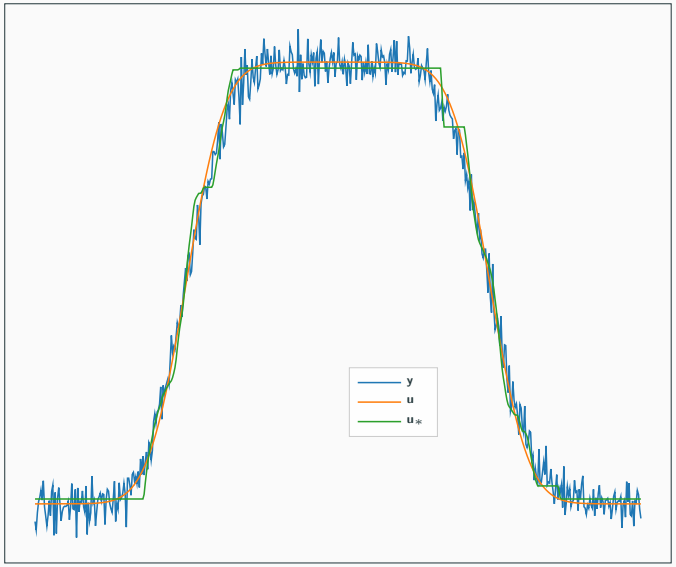
Least-squares TV:

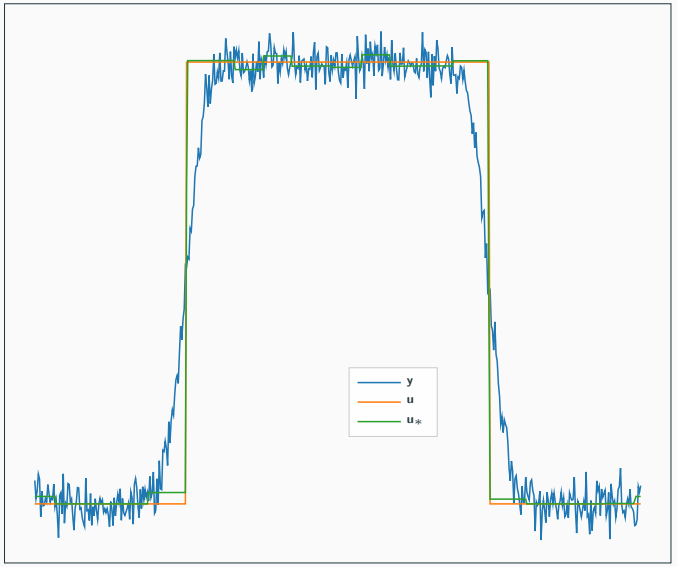
$$\mathbf{u}_* = \arg \min_{\mathbf{u} \in \mathbb{R}^N} \frac{1}{2} \|\mathbf{A}\mathbf{u} - \mathbf{y}\|_2^2 + \lambda \mathrm{TV}(\mathbf{u}).$$

Take-away

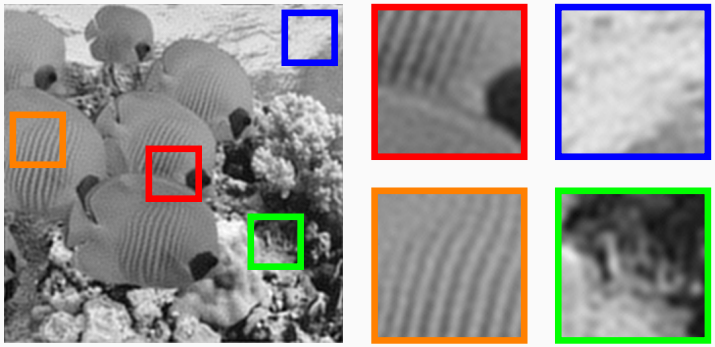
TV replaces the smoothness prior by an edge-preserving prior.

Total variation – TV deconvolution: examples





$\text{PSNR}(\mathbf{u}_*) = 32.3\text{dB}$; $\lambda = 0.67$ $\text{PSNR}(\mathbf{u}_*) = 30.2\text{dB}$; $\lambda = 0.5$



$\mathbf{u}_*$ $(L_2 - \mathrm{TV})$ with $\sigma = 1\%$, PSNR $= 25$ dB and $\lambda = 10^{-2}$.

Discrete gradient

For a discrete image $\mathbf{u} \in \mathbb{R}^{N_1 \times N_2}$, define the forward finite differences

$$(\nabla \mathbf{u})_{i,j} = \begin{bmatrix} u_{i+1,j} - u_{i,j} \\ u_{i,j+1} - u_{i,j} \end{bmatrix}.$$

Then

$$\nabla : \mathbb{R}^{N_1 \times N_2} \longrightarrow \left(\mathbb{R}^{N_1 \times N_2}\right)^2.$$

Total variation – Discretization of differential operators

Discrete gradient

For a discrete image $\mathbf{u} \in \mathbb{R}^{N_1 \times N_2}$, define the forward finite differences

$$(\nabla \mathbf{u})_{i,j} = \begin{bmatrix} u_{i+1,j} - u_{i,j} \\ u_{i,j+1} - u_{i,j} \end{bmatrix}.$$

Then

$$\nabla : \mathbb{R}^{N_1 \times N_2} \longrightarrow \left(\mathbb{R}^{N_1 \times N_2}\right)^2.$$

Discrete adjoint

Using discrete summation by parts,

$$\langle \nabla \mathbf{u}, \mathbf{v} \rangle = \langle \mathbf{u}, \nabla^* \mathbf{v} \rangle,$$

where

$$(\nabla^* \mathbf{v})_{i,j} = -(v^1_{i,j} - v^1_{i-1,j}) - (v^2_{i,j} - v^2_{i,j-1}).$$

Hence, the adjoint of the discrete gradient is the negative discrete divergence:

$$\nabla^* = -\operatorname{div}.$$

New viewpoint

Instead of constraining the image itself, we can constrain its representation in a suitable basis or dictionary. Assume the image can be represented as

$$u \simeq \sum_m \alpha[m]\psi_m.$$

- $(\psi_m)_m$: atoms (wavelets, Fourier, learned filters, ...)
- $\alpha[m]$: coefficients

Sparse representation

Natural images often admit such representations with only a few coefficients different from zero:

Sparse representation: idea

After discretisation, we write

$$\mathbf{u} \simeq \Psi\boldsymbol{\alpha}, \qquad \Psi \in \mathbb{R}^{N\times S}$$

where only a few coefficients of $\boldsymbol{\alpha} \in \mathbb{R}^S$ are non-zero.

- Ψ: dictionary, wavelet basis, learned basis, etc.
- $\boldsymbol{\alpha}$: coefficients.

Ideal sparse regularisation

$$\boldsymbol{\alpha}^* \in \arg\min_{\boldsymbol{\alpha}} D(\mathbf{A}\Psi\boldsymbol{\alpha}, \mathbf{y}) + \lambda\|\boldsymbol{\alpha}\|_0,$$

where the term counting the non-zero terms is:

$$\|\boldsymbol{\alpha}\|_0 = \#\{i : \boldsymbol{\alpha}_i \neq 0\}.$$

Problem

The ℓ^0 problem is combinatorial and generally hard to solve.

Sparsity – Relaxation of ℓ^0 problem

Convex relaxation

Replace the counting penalty by the ℓ^1 norm:

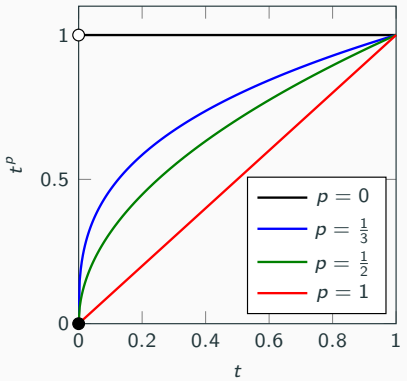
$$\|\boldsymbol{\alpha}\|_1 = \sum_i |\boldsymbol{\alpha}_i|.$$

The reconstruction problem becomes:

$$\boldsymbol{\alpha}_* \in \arg\min_{\boldsymbol{\alpha}} D(\mathbf{A}\boldsymbol{\Psi}\boldsymbol{\alpha}, \mathbf{y}) + \lambda\|\boldsymbol{\alpha}\|_1.$$

Why ℓ^1?

- It promotes sparse coefficients.
- It is convex and computationally tractable.



Sparsity – ℓ^1 representer intuition

Informal representer intuition

The ℓ^1 penalty favours solutions defined in terms of a small number of atoms[a]:

$$\boldsymbol{\alpha}_* \text{ has many zero coefficients.}$$

Thus,

$$\mathbf{u}_* = \Psi \boldsymbol{\alpha}_*$$

is reconstructed from a few selected elements of the dictionary.

[a] Candés, Wakin, '08

Preferred structure

$$\ell^1 \text{ prior} \quad \Longrightarrow \quad \text{sparse representation.}$$

- The active atoms are selected from the data.
- The amplitudes are estimated by optimisation.
- The dictionary encodes the type of patterns we expect.

Sparsity – Geometry of sparse reconstruction

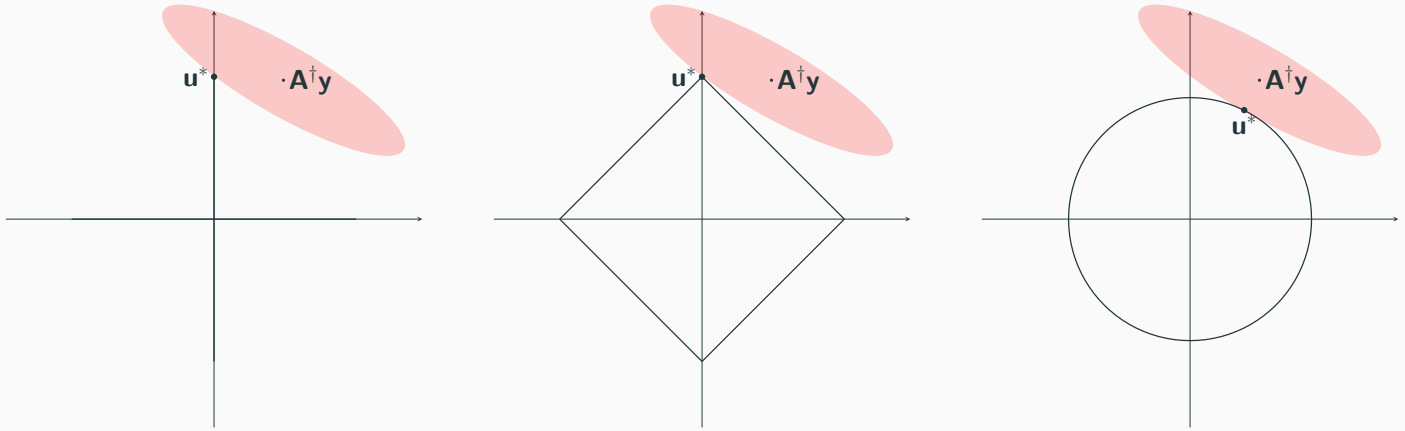
Convex proxy

The ℓ^1 problem is a useful proxy for the ideal ℓ^0 problem.

When does it work well?

This is studied in compressed sensing:

- exact recovery conditions,
- restricted isometry properties,
- sparse recovery guarantees.



ℓ_0 ℓ_1 ℓ_2

Sparsity – Summary: regularisers as image models

Main message

Regularisation is not only a numerical trick. It is an image model.

Regulariser	Promotes	Typical effect
$\|\mathbf{u}\|_2^2$	small energy	ringing
$\|\nabla \mathbf{u}\|_2^2$	smoothness	oversmoothing
$\mathrm{TV}(\mathbf{u})$	sparse gradients	good edges/staircasing
$\|\boldsymbol{\alpha}\|_1$	sparse coefficients	localised reconstructions

Bridge to algorithms

All these models lead to optimisation problems of the form

$$\min_{\mathbf{u}} D(\mathbf{A}\mathbf{u}, \mathbf{y}) + \lambda R(\mathbf{u}).$$

The next question is: how do we solve them efficiently?

Thank you!
Questions?

luca.calatroni@unige.it