Problèmes inverses et optimisation convexe en traitement du signal et des images

Sandrine Anthoine, Valentin Emiya

M2 IAAA - UE SOAP

Sommaire

Programme de l'UE

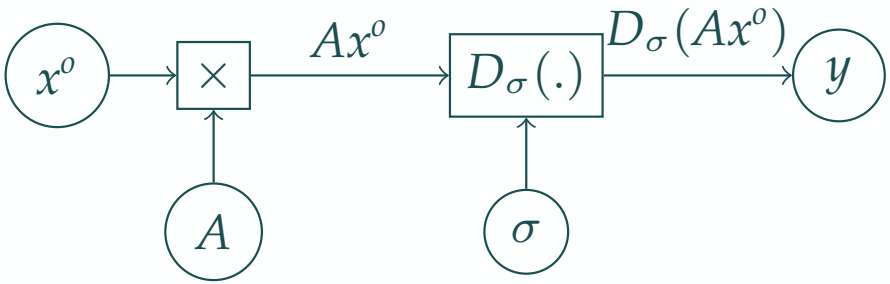
- Bases de traitement du signal (12h) :
  - analyse fréquentielle (Fourier)
  - échantillonnage
  - filtrage
  - représentations
- Calcul scientifique avec JAX (3h)
- **Hybridations avec les problèmes inverses et l'optimisation convexe** (8h)
  - Problèmes inverses et optimisation convexe : découverte

    problème de déconvolution
    algorithme ISTA pour le cas parcimonieux
  - **Problèmes inverses et optimisation convexe : approfondissement**

    Généralisations
  - Hybridation : méthodes Plug-and-Play
  - Hybridation : algorithmes déroulés
- Hybridations avec la physique (4h)

Notion et exemple de problème inverse

Modèle direct : $y = D_\sigma(Ax^o)$



- y : mesures/observations
- x^o : objet à estimer/signal original
- A : opérateur linéaire
- D_σ : perturbation/bruit

Problème inverse : estimer x^o à partir de y, de A et d'informations sur le bruit et sur x^o

Exemple : déconvolution 1D

$$\arg\min_x \underbrace{\|Ax - y\|_2^2}_{\text{attache aux données } \|.\|_2^2 \text{ car bruit gaussien}} + \underbrace{\lambda\|x\|_1}_{\text{régularisation selon hypothèse sur } x^o}$$

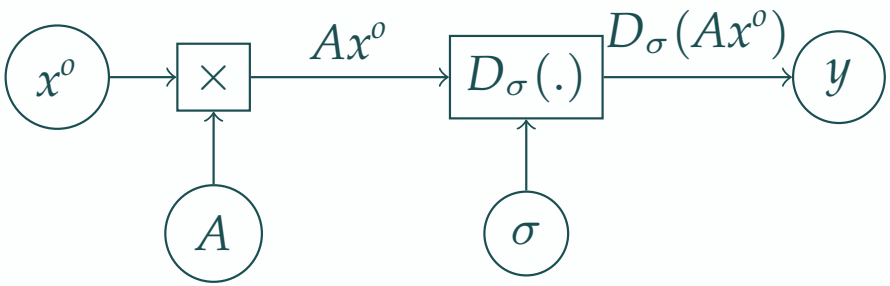
- hypothèse de parcimonie sur vecteur x^o
- A matrice associé à filtre h
- D_σ bruit additif gaussien de variance σ^2

$$D_\sigma(Ax) = Ax + b, b[i] \sim \mathcal{N}(0, \sigma^2)$$

$\rightarrow$ **Autres a priori sur x^o ? Autres bruits D_σ ? Autres opérateurs A ?**

Vers des problèmes inverses plus divers

Modèle direct : $y = D_\sigma(Ax^o)$



- y : mesures/observations
- x^o : objet à estimer/signal original
- A : opérateur linéaire
- D_σ : perturbation/bruit

Problème inverse : estimer x^o à partir de y, de A et d'informations sur le bruit et sur x^o

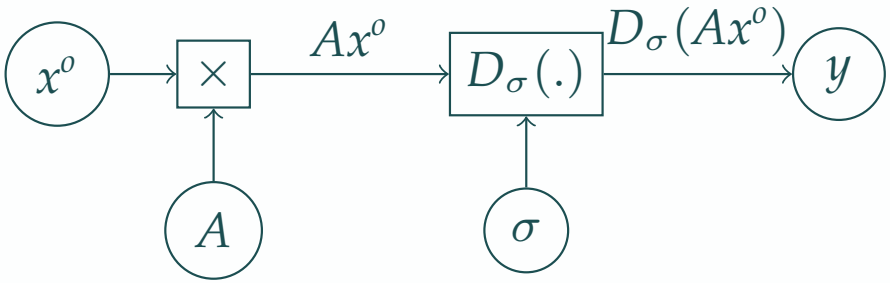
$$\arg\min_x \underbrace{E(Ax, y)}_{\text{attache aux données en fonction de } D_\sigma} + \underbrace{R(x)}_{\text{régularisation selon hypothèse sur } x^o}$$

- autres hypothèses sur x^o ?
- autres opérateurs A ?
- Autres bruits D_σ, attaches aux données E ?

Sommaire

Généralisation à d'autres a priori sur x^o

Modèle direct : $y = D_\sigma(Ax^o)$



Problème inverse

$$\arg\min_x \underbrace{E(Ax, y)}_{\text{attache aux données en fonction de } D_\sigma} + \underbrace{R(x)}_{\text{régularisation selon hypothèse sur } x^o}$$

- $R(x) = \lambda\|x\|_1$: parcimonie sur x
- $R(x) = \lambda\|x\|_2^2$: régularisation de Tikhonov
- $R(x) = \lambda\|\nabla x\|_1$: variation totale, parcimonie sur les variations de x

Très utile pour les images (parcimonie des contours)

- $R(x) = \lambda\|Bx\|_1$: parcimonie d'une représentation linéaire de x

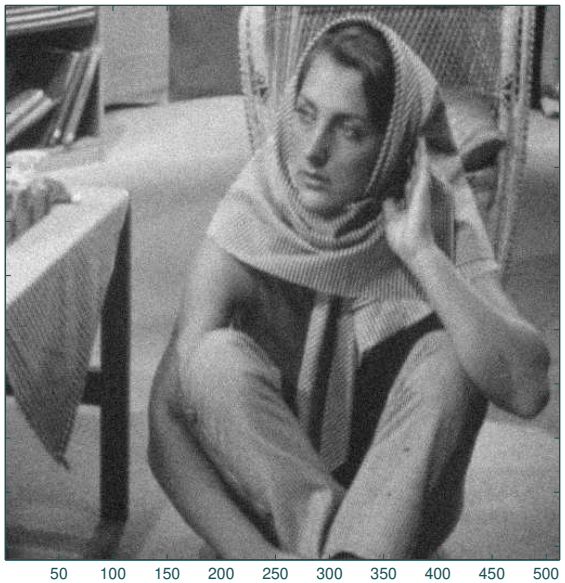
Exemples : ondelettes pour les images, temps-fréquence pour les sons

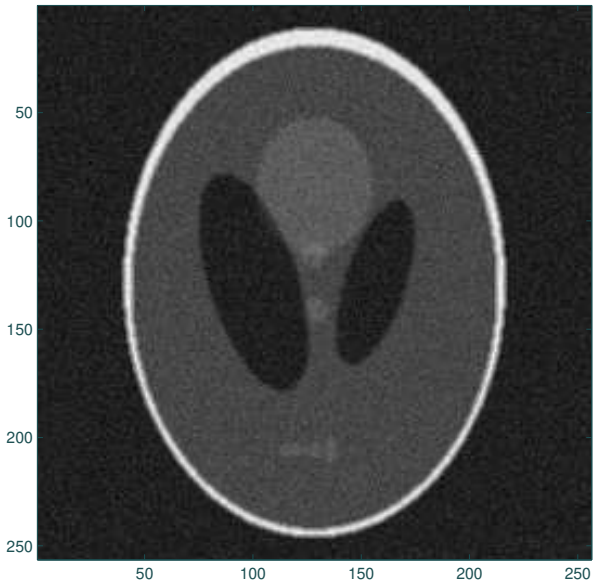
- Plug'n Play : a priori via un réseau de neurones préentrainé

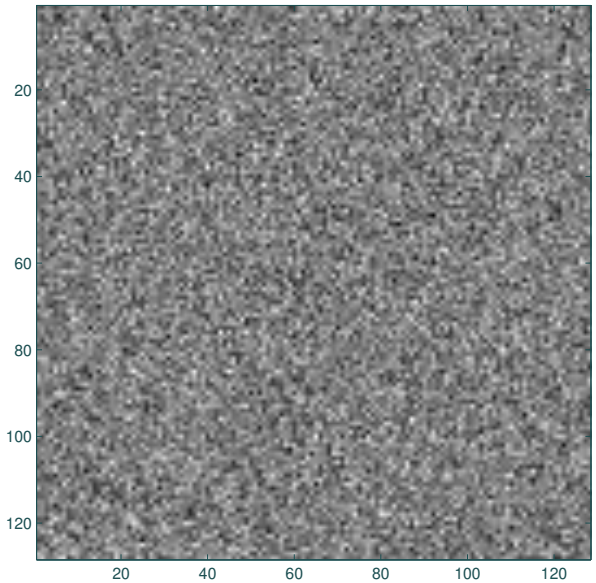
Voir prochaine séance

Exemple de débruitage : régularisation TV vs Tikhonov

Bruit faible



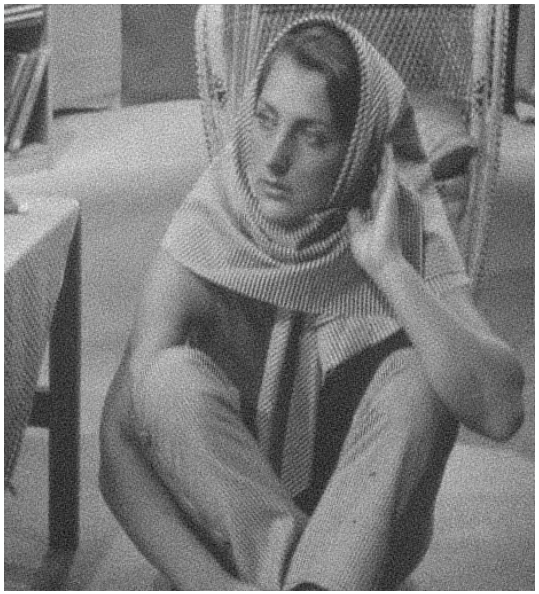




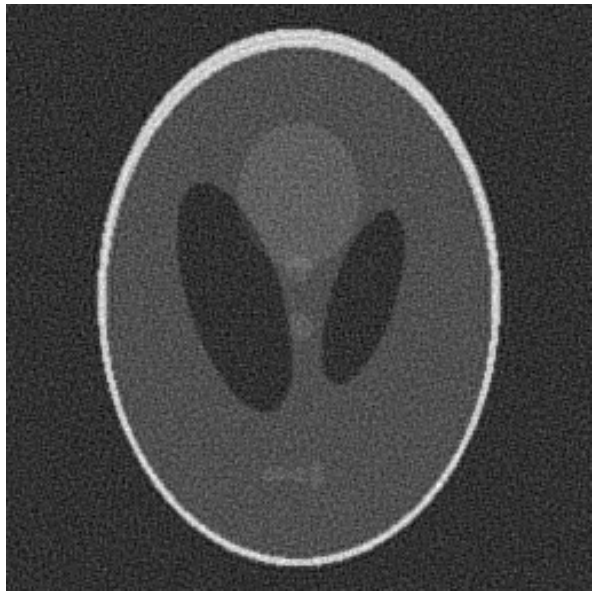
Images observées

Exemple de débruitage : régularisation TV vs Tikhonov

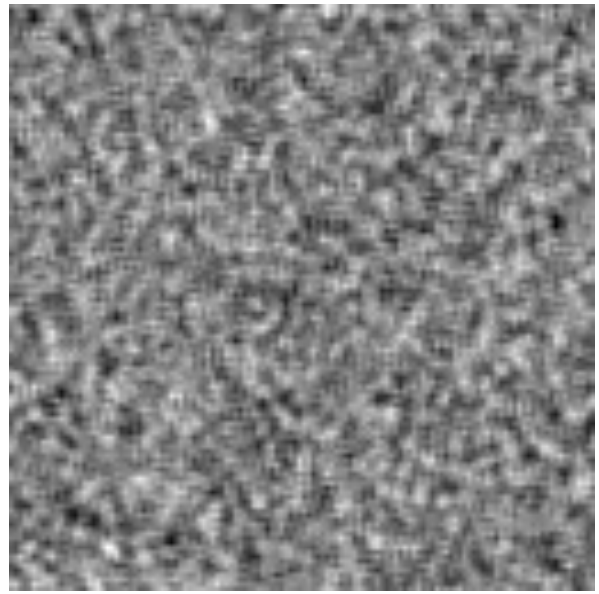
Bruit faible



SNR=15.72 dB



SNR=9.61 dB



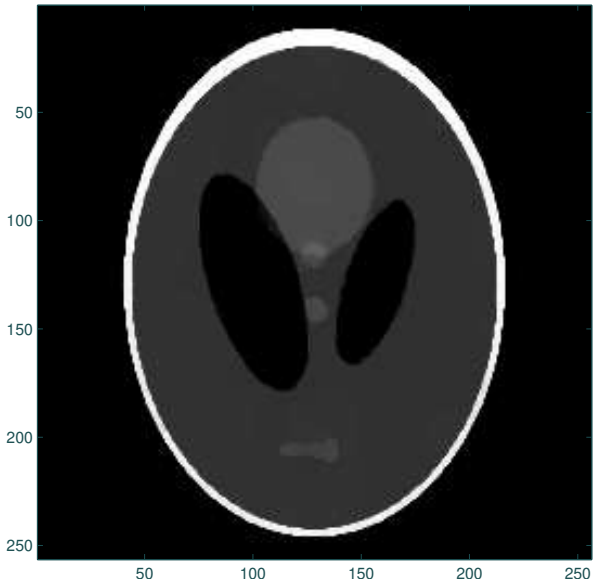
SNR=0.05 dB

Régularisation de Tikhonov

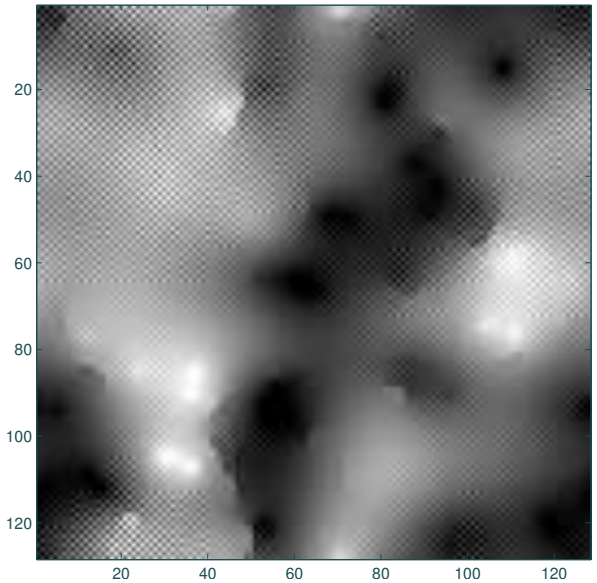
Exemple de débruitage : régularisation TV vs Tikhonov

Bruit faible

SNR=17.82 dB



SNR=19.74 dB

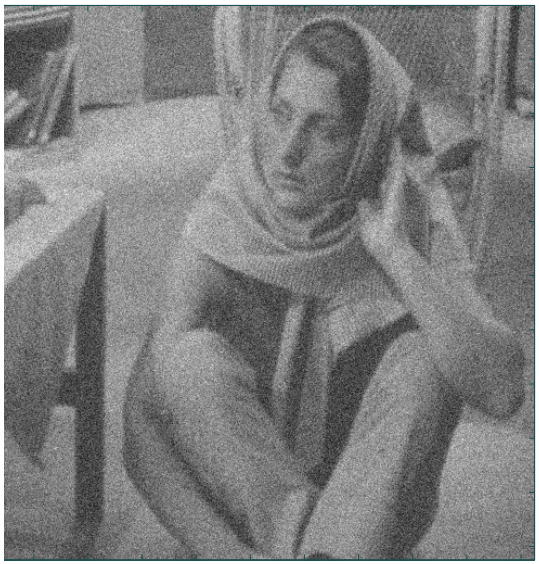


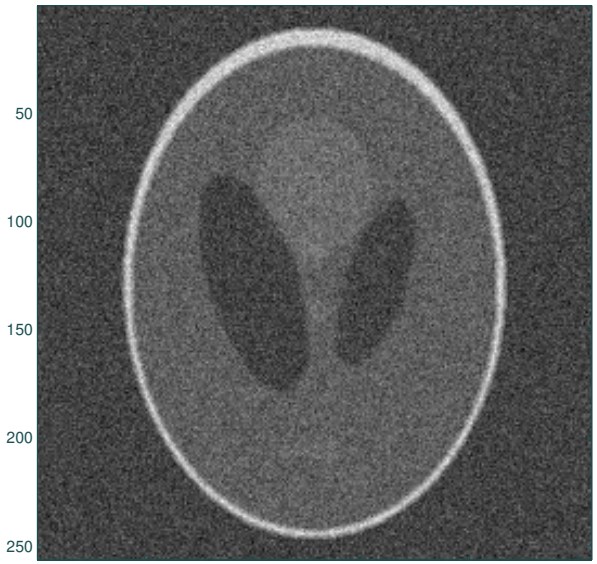
SNR=5.38 dB

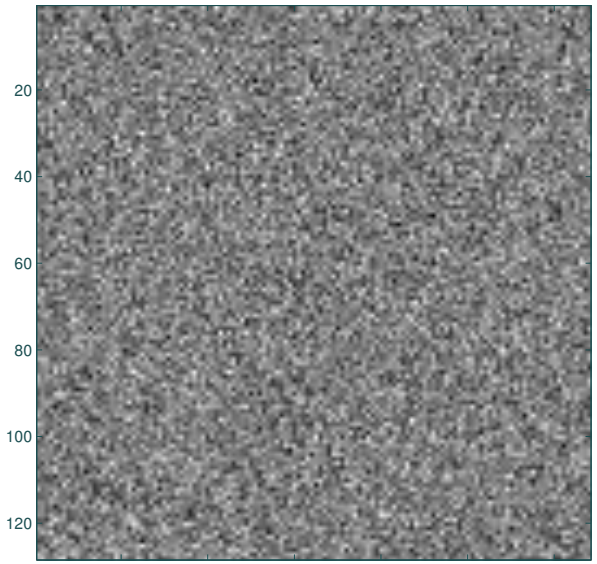
Régularisation par Variation Totale

Exemple de débruitage : régularisation TV vs Tikhonov

Bruit fort



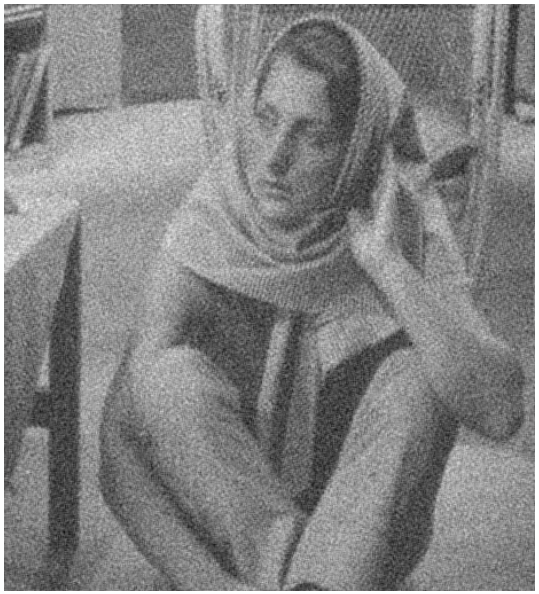




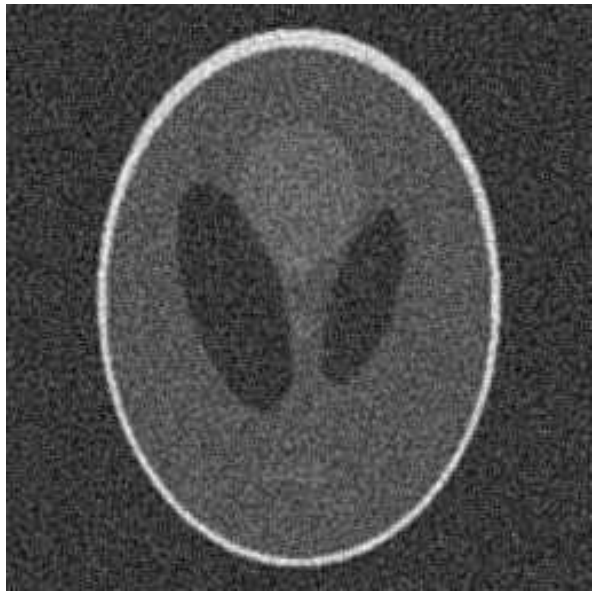
Images observées

Exemple de débruitage : régularisation TV vs Tikhonov

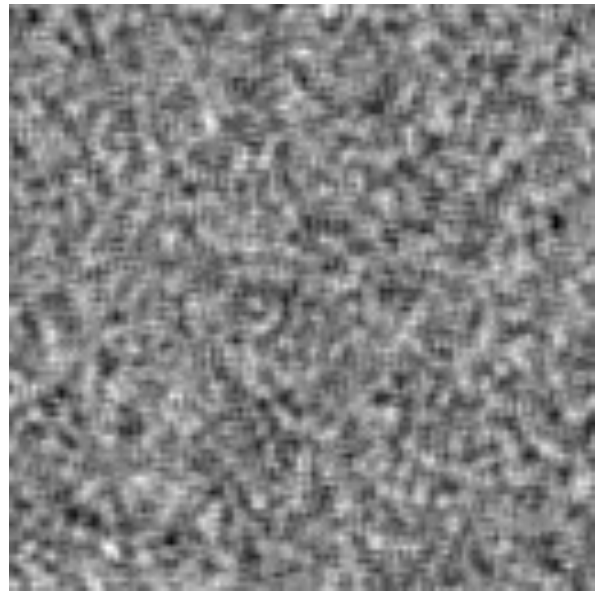
Bruit fort



SNR=10.62 dB



SNR=7.59 dB



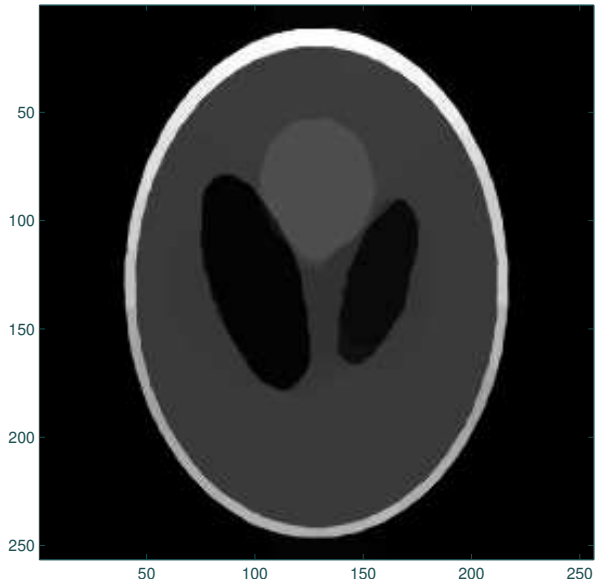
SNR=-0.13 dB

Régularisation de Tikhonov

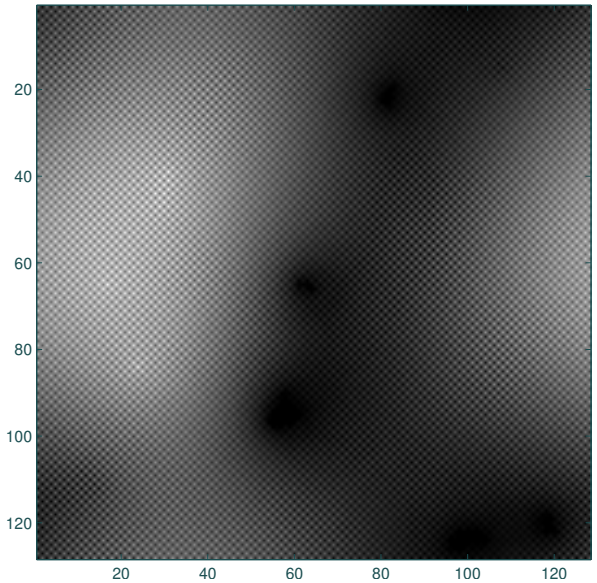
Exemple de débruitage : régularisation TV vs Tikhonov

Bruit fort

SNR=15.80 dB



SNR=9.93 dB



SNR=3.50 dB

Régularisation par Variation Totale

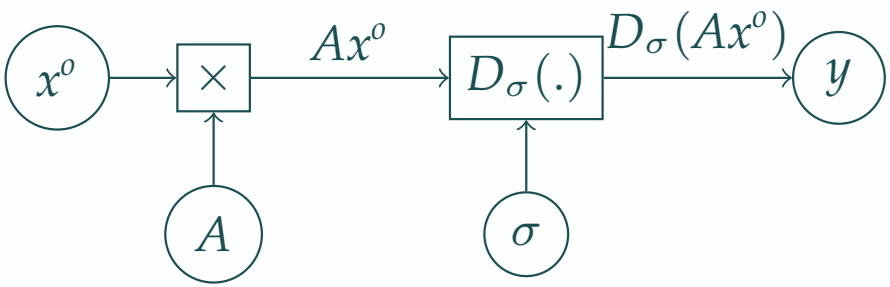
Informations a priori sur la solution x^o : à retenir

- Sans a priori sur la solution, on a souvent un problème mal posé : une infinité de solutions sans intérêt
- Les informations sur x^o sont utiles pour orienter la solution vers un ensemble approprié.
- Le terme de régularisation $R(x)$ encode ces informations.
- De nombreuses possibilités : parcimonie, norme l_2, dans l'espace original ou sur une représentation, etc.

Sommaire

Généralisation à d'autres bruits

Modèle direct : $y = D_\sigma(Ax^o)$



Problème inverse

$$\arg\min_x \underbrace{E(Ax, y)}_{\text{attache aux données en fonction de } D_\sigma} + \underbrace{R(x)}_{\text{régularisation selon hypothèse sur } x^o}$$

Bruit blanc gaussien :

- Très courant
- $D_\sigma(Ax) = Ax + b, \forall i, b[i] \sim \mathcal{N}(0, \sigma^2)$ (i.i.d.)
- Lien avec l'attache aux données $E(Ax, y) = \|Ax - y\|_2^2$ (maximum de vraisemblance)

→ D'autres bruits sont parfois plus pertinents, selon la physique du modèle direct.

Bruit coloré gaussien $\mathcal{N}(0, \Sigma)$, bruit de grenaille/Poisson

→ Il faut choisir l'attache aux données en conséquence.

Autre exemple de bruit : bruit de grenaille/Poisson

→ Bruit adapté pour les images

Original

Grenaille/Poisson

Gaussien

Autre exemple de bruit : bruit de grenaille/Poisson

Original

Autre exemple de bruit : bruit de grenaille/Poisson

Grenaille/Poisson

Autre exemple de bruit : bruit de grenaille/Poisson

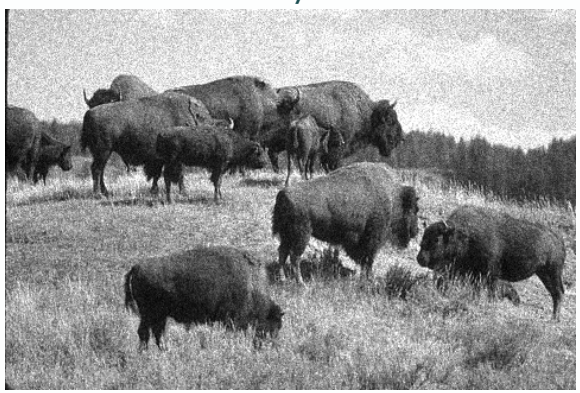
Gaussien

Autre exemple de bruit : bruit de grenaille/Poisson

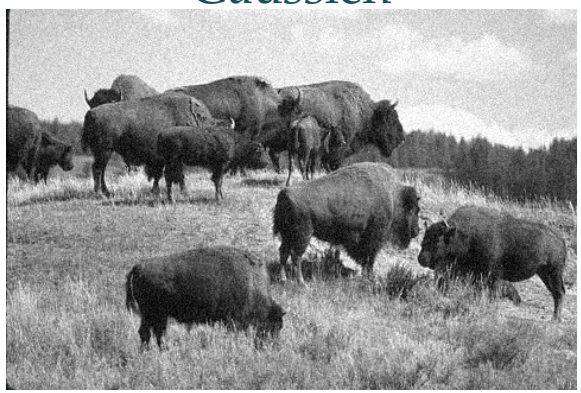
→ Bruit adapté pour les images

Original

Grenaille/Poisson



Gaussien



Autre exemple de bruit : bruit de grenaille/Poisson

Original

Autre exemple de bruit : bruit de grenaille/Poisson

Grenaille/Poisson

Autre exemple de bruit : bruit de grenaille/Poisson

Gaussien

Autre exemple de bruit : bruit de grenaille/Poisson

$\rightarrow$ Bruit adapté pour les images

Autre exemple de bruit : bruit de grenaille/Poisson

$\rightarrow$ Bruit adapté pour les images

Variable aléatoire réelle de Poisson

- $X \sim \mathcal{P}(\beta)$ est une variable de Poisson de paramètre $\beta > 0$.
- Sa loi est $P(X = n) = e^{-\beta} \frac{\beta^n}{n!}$ où $n = 0, 1, \ldots$.
- Interprétation : si β est le nombre d'occurrence moyenne qui se produisent sur une durée donnée, $P(X = n)$ est la probabilité que n occurrences surviennent sur cette durée.

La loi de Poisson modélise bien le bruit lié à l'émission des photons.

Bruit de Poisson sur une image Y

- $Z \sim \mathcal{P}(Y)$ est un vecteur aléatoire.
- $Z[m, n] \sim \mathcal{P}(y[m, n])$: les $Z[m, n]$ sont indépendants mais pas identiquement distribués, le bruit est plus important sur les pixels plus lumineux.

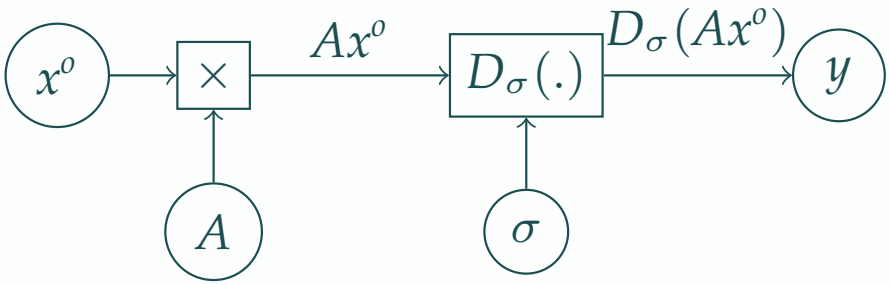
Bruit : à retenir

- Le type de bruit dépend de la physique du modèle direct.
- Le terme d'attache aux données est à définir en fonction du bruit

Sommaire

Les opérateurs : au-delà des matrices

Modèle direct : $y = D_\sigma(Ax^o)$



Problème inverse

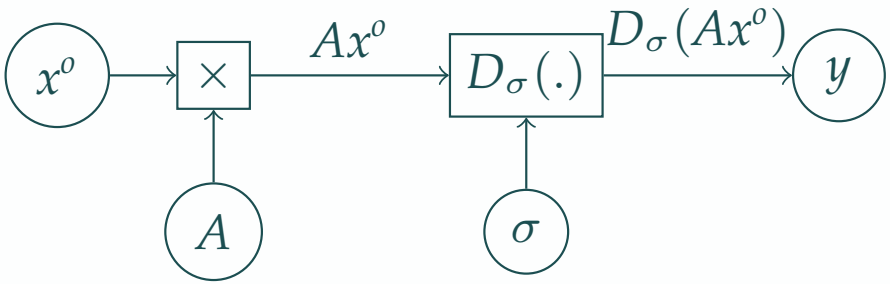
$$\arg\min_x \underbrace{E(Ax, y)}_{\text{attache aux données en fonction de } D_\sigma} + \underbrace{R(x)}_{\text{régularisation selon hypothèse sur } x^o}$$

A est un opérateur linéaire, par exemple

- Une matrice
- Autre ?

Les opérateurs : au-delà des matrices

Modèle direct : $y = D_\sigma(Ax^o)$



Problème inverse

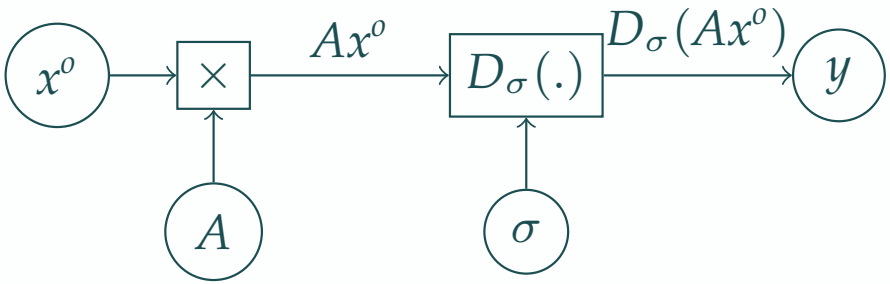
$$\arg\min_x \underbrace{E(Ax, y)}_{\text{attache aux données en fonction de } D_\sigma} + \underbrace{R(x)}_{\text{régularisation selon hypothèse sur } x^o}$$

A est un opérateur linéaire, par exemple

- Une matrice
- Une convolution : $Ax = h * x$

Les opérateurs : au-delà des matrices

Modèle direct : $y = D_\sigma(Ax^o)$



Problème inverse

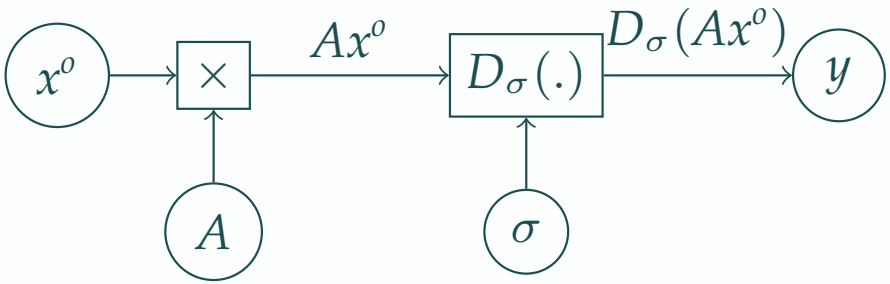
$$\arg\min_x \underbrace{E(Ax, y)}_{\text{attache aux données en fonction de } D_\sigma} + \underbrace{R(x)}_{\text{régularisation selon hypothèse sur } x^o}$$

A est un opérateur linéaire, par exemple

- Une matrice
- Une convolution : $Ax = h * x$
- Sous échantillonnage $Ax = x[\text{indices\_mask}]$

Les opérateurs : au-delà des matrices

Modèle direct : $y = D_\sigma(Ax^o)$



Problème inverse

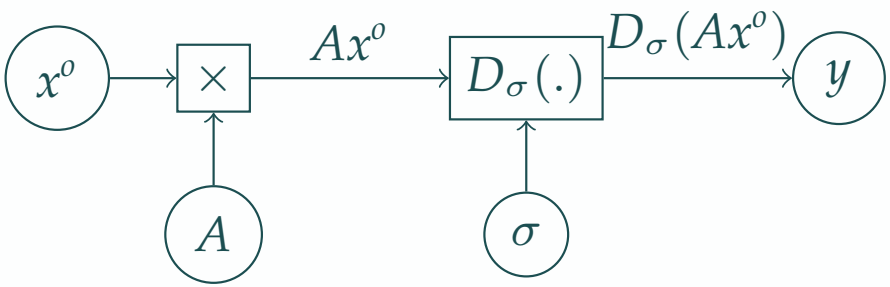
$$\arg\min_x \underbrace{E(Ax, y)}_{\text{attache aux données en fonction de } D_\sigma} + \underbrace{R(x)}_{\text{régularisation selon hypothèse sur } x^o}$$

A est un opérateur linéaire, par exemple

- Une matrice
- Une convolution : $Ax = h * x$
- Sous échantillonnage $Ax = x[\text{indices\_mask}]$
- Transformée de Fourier, ondelettes, etc.

Les opérateurs : au-delà des matrices

Modèle direct : $y = D_\sigma(Ax^o)$



Problème inverse

$$\arg\min_x \underbrace{E(Ax, y)}_{\text{attache aux données en fonction de } D_\sigma} + \underbrace{R(x)}_{\text{régularisation selon hypothèse sur } x^o}$$

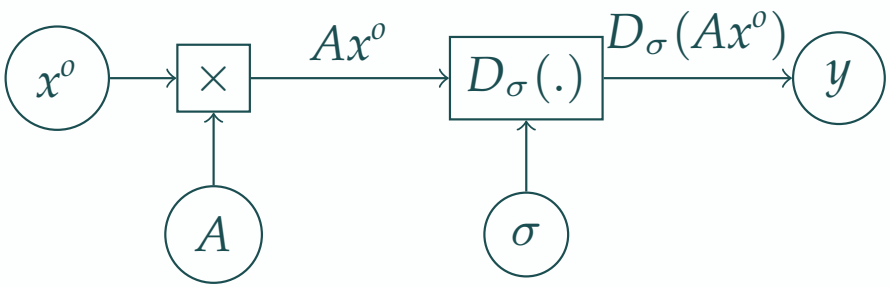
A est un opérateur linéaire, par exemple

- Une matrice
- Une convolution : $Ax = h * x$
- Sous échantillonnage $Ax = x[\text{indices\_mask}]$
- Transformée de Fourier, ondelettes, etc.

On peut parfois effectuer le calcul Ax de deux façons : produit matriciel ou calcul spécifique (filtrage, échantillonnage, masquage, etc.).
Pourquoi ?

Les opérateurs : au-delà des matrices

Modèle direct : $y = D_\sigma(Ax^o)$



Problème inverse

$$\arg\min_x \underbrace{E(Ax, y)}_{\text{attache aux données en fonction de } D_\sigma} + \underbrace{R(x)}_{\text{régularisation selon hypothèse sur } x^o}$$

A est un opérateur linéaire, par exemple

- Une matrice
- Une convolution : $Ax = h * x$
- Sous échantillonnage $Ax = x[\text{indices\_mask}]$
- Transformée de Fourier, ondelettes, etc.

On peut parfois effectuer le calcul Ax de deux façons : produit matriciel ou calcul spécifique (filtrage, échantillonnage, masquage, etc.).
Pourquoi ? Rapidité de calcul, codage plus facile

Appliquer un opérateur sans passer par une matrice

Exemple : résoudre $\arg\min_x \|Ax - y\|_2^2 + \lambda \|x\|_1$ avec $\lambda > 0$

Algorithme 1 ISTA avec matrice A

Entrées: $A \in \mathbb{R}^{N \times M}$, $y \in \mathbb{R}^N$, $\lambda > 0$, $\eta > 0$, N_{iter}

1: $x \leftarrow 0_M$
2: **pour** $i = 0 \dots N_{\text{iter}} - 1$ **faire**
3: $\quad u \leftarrow x - 2\eta A^T (Ax - y)$
4: $\quad x \leftarrow S_{\lambda\eta}(u)$
5: **fin pour**

Sorties: x

→ Le coût de chaque itération est dominé par les produits Ax et $A^T(\dots)$.

Appliquer un opérateur sans passer par une matrice

Exemple : résoudre $\arg\min_x \|Ax - y\|_2^2 + \lambda \|x\|_1$ avec $\lambda > 0$

Algorithme 2 ISTA avec matrice A

Entrées: $A \in \mathbb{R}^{N \times M}$, $y \in \mathbb{R}^N$, $\lambda > 0$, $\eta > 0$, N_{iter}

1: $x \leftarrow 0_M$
2: **pour** $i = 0 \dots N_{\text{iter}} - 1$ **faire**
3: $\quad u \leftarrow x - 2\eta A^T (Ax - y)$
4: $\quad x \leftarrow S_{\lambda\eta}(u)$
5: **fin pour**

Sorties: x

Algorithme 3 ISTA sans matrice A

Entrées: $A : \mathbb{R}^M \to \mathbb{R}^N$, $y \in \mathbb{R}^N$, $\lambda > 0$, $\eta > 0$, N_{iter}

1: $x \leftarrow 0_M$
2: **pour** $i = 0 \dots N_{\text{iter}} - 1$ **faire**
3: $\quad u \leftarrow x - 2\eta A^T (A(x) - y)$
4: $\quad x \leftarrow S_{\lambda\eta}(u)$
5: **fin pour**

Sorties: x

→ Le coût de chaque itération est dominé par les produits Ax et $A^T(\dots)$.

→ On remplace le produit Ax en $\mathcal{O}(MN)$ par l'appel à une fonction $A(x)$ de complexité parfois inférieure : $\mathcal{O}(N \log N)$ si transformée de Fourier rapide, $\mathcal{O}(N)$ si masquage, etc.

Appliquer un opérateur sans passer par une matrice

Exemple : résoudre $\arg\min_x \|Ax - y\|_2^2 + \lambda \|x\|_1$ avec $\lambda > 0$

Algorithme 4 ISTA avec matrice A

Entrées: $A \in \mathbb{R}^{N \times M}$, $y \in \mathbb{R}^N$, $\lambda > 0$, $\eta > 0$, N_{iter}

1: $x \leftarrow 0_M$
2: **pour** $i = 0 \dots N_{\text{iter}} - 1$ **faire**
3: $\quad u \leftarrow x - 2\eta A^T (Ax - y)$
4: $\quad x \leftarrow S_{\lambda\eta}(u)$
5: **fin pour**

Sorties: x

Algorithme 5 ISTA sans matrice A

Entrées: $A : \mathbb{R}^M \to \mathbb{R}^N$, $y \in \mathbb{R}^N$, $\lambda > 0$, $\eta > 0$, N_{iter}

1: $x \leftarrow 0_M$
2: **pour** $i = 0 \dots N_{\text{iter}} - 1$ **faire**
3: $\quad u \leftarrow x - 2\eta (A(x) - y)$
4: $\quad x \leftarrow S_{\lambda\eta}(u)$
5: **fin pour**

Sorties: x

→ Le coût de chaque itération est dominé par les produits Ax et $A^T(\dots)$.

→ On remplace le produit Ax en $\mathcal{O}(MN)$ par l'appel à une fonction $A(x)$ de complexité parfois inférieure : $\mathcal{O}(N \log N)$ si transformée de Fourier rapide, $\mathcal{O}(N)$ si masquage, etc.

→ Problème ?

Appliquer un opérateur sans passer par une matrice

Exemple : résoudre $\arg\min_x \|Ax - y\|_2^2 + \lambda \|x\|_1$ avec $\lambda > 0$

Algorithme 6 ISTA avec matrice A

Entrées: $A \in \mathbb{R}^{N \times M}$, $y \in \mathbb{R}^N$, $\lambda > 0$, $\eta > 0$, N_{iter}

1: $x \leftarrow 0_M$
2: **pour** $i = 0 \dots N_{\text{iter}} - 1$ **faire**
3: $\quad u \leftarrow x - 2\eta A^T (Ax - y)$
4: $\quad x \leftarrow S_{\lambda\eta}(u)$
5: **fin pour**

Sorties: x

Algorithme 7 ISTA sans matrice A

Entrées: $A : \mathbb{R}^M \to \mathbb{R}^N$, $y \in \mathbb{R}^N$, $\lambda > 0$, $\eta > 0$, N_{iter}

1: $x \leftarrow 0_M$
2: **pour** $i = 0 \dots N_{\text{iter}} - 1$ **faire**
3: $\quad u \leftarrow x - 2\eta A^T (A(x) - y)$
4: $\quad x \leftarrow S_{\lambda\eta}(u)$
5: **fin pour**

Sorties: x

→ Le coût de chaque itération est dominé par les produits Ax et $A^T(\dots)$.

→ On remplace le produit Ax en $\mathcal{O}(MN)$ par l'appel à une fonction $A(x)$ de complexité parfois inférieure : $\mathcal{O}(N \log N)$ si transformée de Fourier rapide, $\mathcal{O}(N)$ si masquage, etc.

→ Problème ? Comment faire la multiplication par A^T ?

Appliquer un opérateur sans passer par une matrice

Exemple : résoudre $\arg\min_x \|Ax - y\|_2^2 + \lambda \|x\|_1$ avec $\lambda > 0$

Algorithme 8 ISTA avec matrice A

Entrées: $A \in \mathbb{R}^{N \times M}$, $y \in \mathbb{R}^N$, $\lambda > 0$, $\eta > 0$, N_{iter}

1: $x \leftarrow 0_M$
2: **pour** $i = 0 \dots N_{\text{iter}} - 1$ **faire**
3: $\quad u \leftarrow x - 2\eta A^T (Ax - y)$
4: $\quad x \leftarrow S_{\lambda\eta}(u)$
5: **fin pour**

Sorties: x

Algorithme 9 ISTA sans matrice A

Entrées: $A : \mathbb{R}^M \to \mathbb{R}^N$, $y \in \mathbb{R}^N$, $\lambda > 0$, $\eta > 0$, N_{iter}

1: $x \leftarrow 0_M$
2: **pour** $i = 0 \dots N_{\text{iter}} - 1$ **faire**
3: $\quad u \leftarrow x - 2\eta A^T (A(x) - y)$
4: $\quad x \leftarrow S_{\lambda\eta}(u)$
5: **fin pour**

Sorties: x

→ Le coût de chaque itération est dominé par les produits Ax et $A^T(\dots)$.

→ On remplace le produit Ax en $\mathcal{O}(MN)$ par l'appel à une fonction $A(x)$ de complexité parfois inférieure : $\mathcal{O}(N \log N)$ si transformée de Fourier rapide, $\mathcal{O}(N)$ si masquage, etc.

→ Problème ? Comment faire la multiplication par A^T ?

→ A^T s'appelle l'adjoint de A dans la théorie des opérateurs.

Notion d'opérateur adjoint

Définition : opérateur adjoint

Si $F : \mathbb{R}^M \to \mathbb{R}^N$ est un opérateur linéaire, alors il existe un unique opérateur $G : \mathbb{R}^N \to \mathbb{R}^M$ appelé adjoint de F tel que

$$\forall x \in \mathbb{R}^M, \forall y \in \mathbb{R}^N, \quad \underbrace{\langle F(x), y \rangle}_{\text{Produit scalaire dans } R^N} = \underbrace{\langle x, G(y) \rangle}_{\text{Produit scalaire dans } R^M}$$

Notion d'opérateur adjoint

Définition : opérateur adjoint

Si $F : \mathbb{R}^M \to \mathbb{R}^N$ est un opérateur linéaire, alors il existe un unique opérateur $G : \mathbb{R}^N \to \mathbb{R}^M$ appelé adjoint de F tel que

$$\forall x \in \mathbb{R}^M, \forall y \in \mathbb{R}^N, \quad \underbrace{\langle F(x), y \rangle}_{\text{Produit scalaire dans } R^N} = \underbrace{\langle x, G(y) \rangle}_{\text{Produit scalaire dans } R^M}$$

Opérateur adjoint dans le cas d'une matrice

Si $\forall x \in \mathbb{R}^M, F(x) = Ax$ où $A \in \mathbb{R}^{M \times N}$, alors $\forall y \in \mathbb{R}^N, G(y) = A^T y$.

Preuve ?

Notion d'opérateur adjoint

Définition : opérateur adjoint

Si $F : \mathbb{R}^M \to \mathbb{R}^N$ est un opérateur linéaire, alors il existe un unique opérateur $G : \mathbb{R}^N \to \mathbb{R}^M$ appelé adjoint de F tel que

$$\forall x \in \mathbb{R}^M, \forall y \in \mathbb{R}^N, \quad \underbrace{\langle F(x), y \rangle}_{\text{Produit scalaire dans } R^N} = \underbrace{\langle x, G(y) \rangle}_{\text{Produit scalaire dans } R^M}$$

Opérateur adjoint dans le cas d'une matrice

Si $\forall x \in \mathbb{R}^M, F(x) = Ax$ où $A \in \mathbb{R}^{M \times N}$, alors $\forall y \in \mathbb{R}^N, G(y) = A^T y$.

Preuve ? Pour tout $x \in \mathbb{R}^M$ et $y \in \mathbb{R}^N$, on a

$$\langle F(x), y \rangle = F(x)^T y = (Ax)^T y = x^T A^T y = x^T (A^T y) = \left\langle x, \underbrace{A^T y}_{G(y)} \right\rangle$$

Notion d'opérateur adjoint

Définition : opérateur adjoint

Si $F : \mathbb{R}^M \to \mathbb{R}^N$ est un opérateur linéaire, alors il existe un unique opérateur $G : \mathbb{R}^N \to \mathbb{R}^M$ appelé adjoint de F tel que

$$\forall x \in \mathbb{R}^M, \forall y \in \mathbb{R}^N, \quad \underbrace{\langle F(x), y \rangle}_{\text{Produit scalaire dans } R^N} = \underbrace{\langle x, G(y) \rangle}_{\text{Produit scalaire dans } R^M}$$

Opérateur adjoint dans le cas d'une matrice

Si $\forall x \in \mathbb{R}^M, F(x) = Ax$ où $A \in \mathbb{R}^{M \times N}$, alors $\forall y \in \mathbb{R}^N, G(y) = A^T y$.

Remarque : si A est orthogonale, $A^T = A^{-1}$.
Exemple : transformée de Fourier et son inverse.

Appliquer un opérateur sans passer par une matrice

Exemple : résoudre $\arg\min_x \|Ax - y\|_2^2 + \lambda \|x\|_1$ avec $\lambda > 0$

Algorithme 10 ISTA sans matrice A

Entrées: $A : \mathbb{R}^M \to \mathbb{R}^N$, son adjoint $A^T : \mathbb{R}^N \to \mathbb{R}^M$, $y \in \mathbb{R}^N$, $\lambda > 0$, $\eta > 0$, N_{iter}

1: $x \leftarrow 0_M$
2: **pour** $i = 0 \dots N_{\text{iter}} - 1$ **faire**
3: $\quad u \leftarrow x - 2\eta A^T (A(x) - y)$
4: $\quad x \leftarrow S_{\lambda\eta}(u)$
5: **fin pour**

Sorties: x

Il suffit de connaître l'opérateur A et son adjoint, noté A^T.

Opérateur et opérateur adjoint pour le masquage (inpainting)

Opérateur direct : `y = x[m]` avec `m = [1,0,1,0,0,1]`

Équivalent matriciel :

$$y = \begin{bmatrix} x[0] \\ x[2] \\ x[5] \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{=A} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \end{bmatrix} = Ax$$

Opérateur adjoint

$$w = A^T z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} z[0] \\ z[1] \\ z[2] \end{bmatrix} = \begin{bmatrix} z[0] \\ 0 \\ z[1] \\ 0 \\ 0 \\ z[2] \end{bmatrix}$$

```
w = zeros(6)
w[m] = z
```

Opérateur adjoint pour la convolution

$$h * x = Ax$$

avec

$$A = \begin{bmatrix} h[N/2] & h[N/2-1] & \dots & h[0] & h[N-1] & \dots & h[N/2+1] \\ h[N/2+1] & h[N/2] & h[N/2-1] & \dots & h[0] & \dots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & h[N-1] \\ h[N-1] & h[N-2] & \ddots & \ddots & \ddots & \ddots & h[0] \\ h[0] & h[N-1] & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & h[N/2-1] \\ h[N/2-1] & \dots & h[0] & h[N-1] & \dots & h[N/2+1] & h[N/2] \end{bmatrix}$$

L'adjoint est une convolution par une version retournée du filtre h :

$$A^T y = g * y$$

avec `g = h[::-1]` si h de longueur impaire,
`g = [h[0], h[-1:0:-1]]` si h de longueur paire.

Codage des opérateurs avec numpy/scipy

La classe `scipy.sparse.linalg.LinearOperator`[1] est faite pour vous :

- permet d'utiliser `A @ x` pour appliquer indistinctement une matrice ou un `LinearOperator`
- surcharger la méthode `matvec` pour coder le calcul de `A @ x`
- surcharger la méthode `rmatvec` pour coder le calcul de `A.T @ x` (adjoint)
- de nombreux opérateurs (convolution, etc.) sont déjà codés, voir paquet `pylops`[2].

DEMO !

1. `https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.linalg.LinearOperator.html`
2. `https://pylops.readthedocs.io/en/stable/`

Sommaire

Exemple illustrant la flexibilité et la portée de l'approche

Problème inverse

Reconstruire une image floue, bruitée et dont on n'observe pas tous les pixels !

On considère trois phénomènes affectant les images et une hypothèse :

1. flou/filtre gaussien
   $\rightarrow$ problème de défloutage
2. pixels manquants
   $\rightarrow$ problème d'inpainting
3. bruit additif
   $\rightarrow$ problème de débruitage
4. hypothèse de parcimonie dans le domaine des ondelettes

Original

Exemple illustrant la flexibilité et la portée de l'approche

Problème inverse

Reconstruire une image floue, bruitée et dont on n'observe pas tous les pixels !

On considère trois phénomènes affectant les images et une hypothèse :

1. flou/filtre gaussien
   $\rightarrow$ problème de défloutage
2. pixels manquants
   $\rightarrow$ problème d'inpainting
3. bruit additif
   $\rightarrow$ problème de débruitage
4. hypothèse de parcimonie dans le domaine des ondelettes

Flou

Exemple illustrant la flexibilité et la portée de l'approche

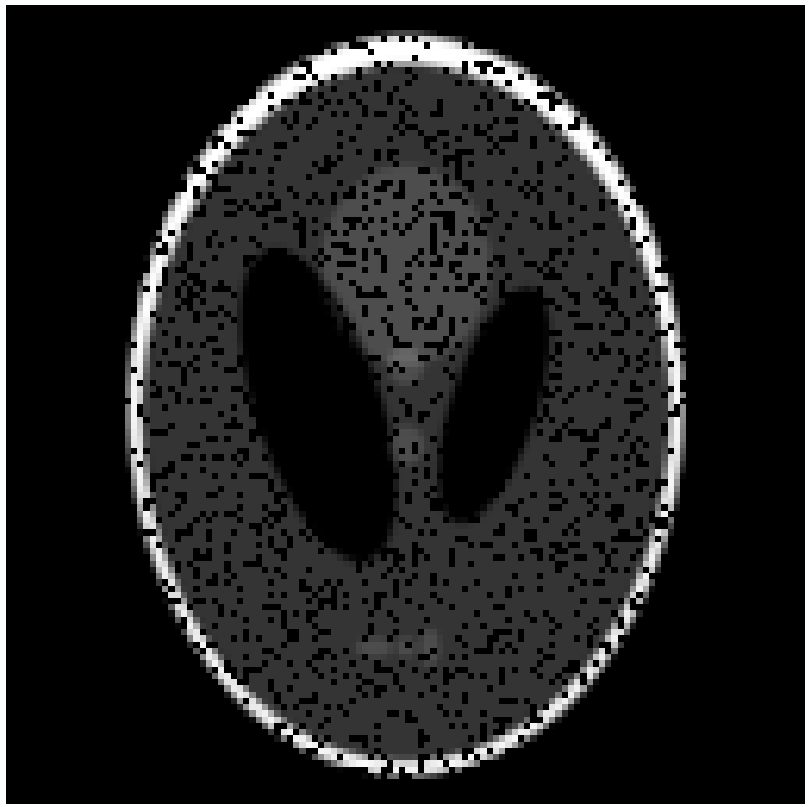
Problème inverse

Reconstruire une image floue, bruitée et dont on n'observe pas tous les pixels !

On considère trois phénomènes affectant les images et une hypothèse :

1. flou/filtre gaussien
   $\rightarrow$ problème de défloutage
2. pixels manquants
   $\rightarrow$ problème d'inpainting
3. bruit additif
   $\rightarrow$ problème de débruitage
4. hypothèse de parcimonie dans le domaine des ondelettes

Pixels manquants



Exemple illustrant la flexibilité et la portée de l'approche

Problème inverse

Reconstruire une image floue, bruitée et dont on n'observe pas tous les pixels !

On considère trois phénomènes affectant les images et une hypothèse :

1. flou/filtre gaussien
   $\rightarrow$ problème de défloutage
2. pixels manquants
   $\rightarrow$ problème d'inpainting
3. bruit additif
   $\rightarrow$ problème de débruitage
4. hypothèse de parcimonie dans le domaine des ondelettes

Bruit

Exemple illustrant la flexibilité et la portée de l'approche

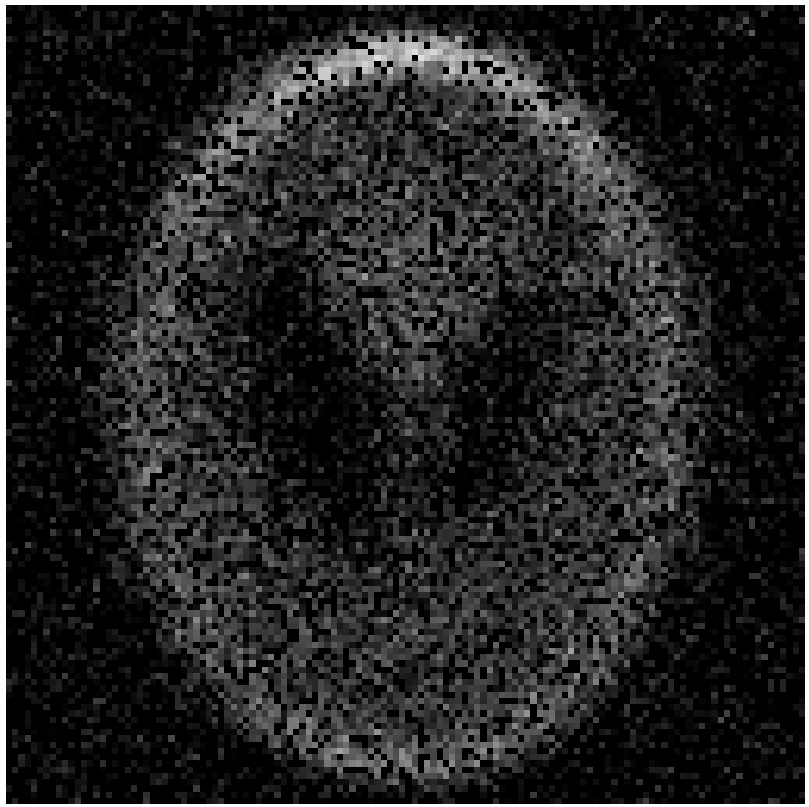
Problème inverse

Reconstruire une image floue, bruitée et dont on n'observe pas tous les pixels !

On considère trois phénomènes affectant les images et une hypothèse :

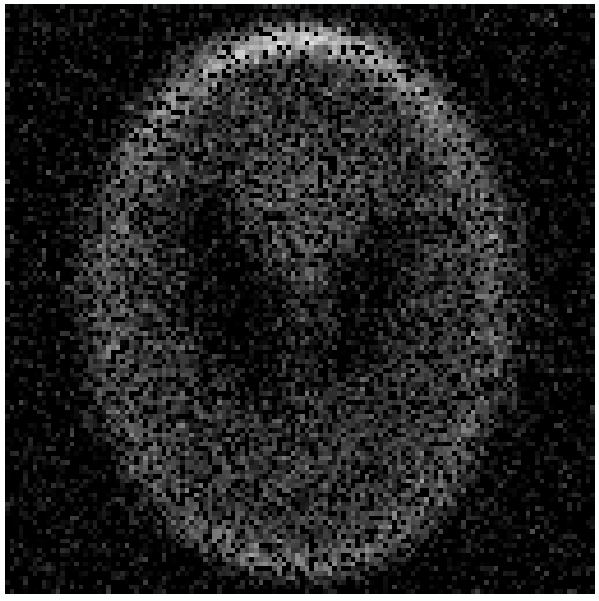
1. flou/filtre gaussien
   $\rightarrow$ problème de défloutage
2. pixels manquants
   $\rightarrow$ problème d'inpainting
3. bruit additif
   $\rightarrow$ problème de débruitage
4. hypothèse de parcimonie dans le domaine des ondelettes

Les trois ensemble



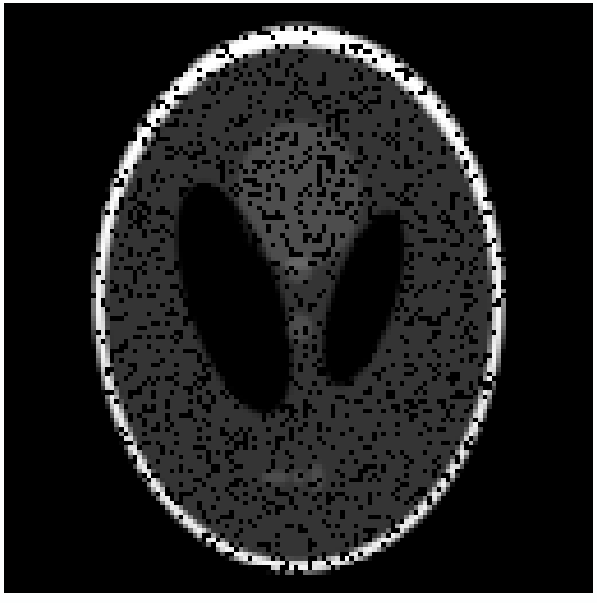
Original

Les trois ensemble



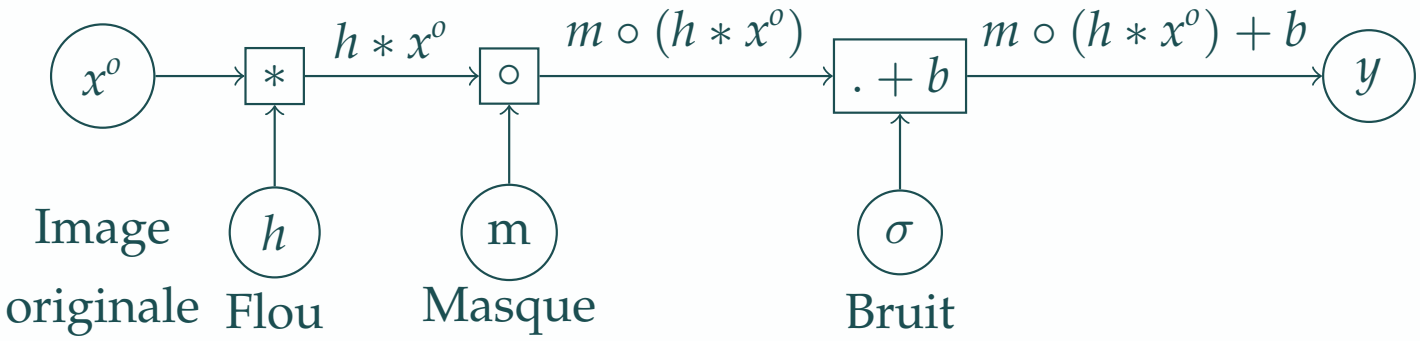
Flou

Pixels manquants



Bruit

Modèle direct



Problème inverse

Étant donné l'opérateur de transformée en ondelette W :

$$\arg\min_x \|m \circ (h * x) - y\|_2^2 + \lambda \|Wx\|_1$$

Changement de variable :
$z = Wx$ représentation en ondelettes de x

$$\arg\min_z \| \underbrace{m \circ (h * W^T z)}_{Az} - y\|_2^2 + \lambda \|z\|_1$$

$\rightarrow A : z \mapsto m \circ (h * W^T z)$ opérateur linéaire, composition de trois opérateurs linéaires.
$\rightarrow$ `A = A1 @ A2 @ A3` en Python.

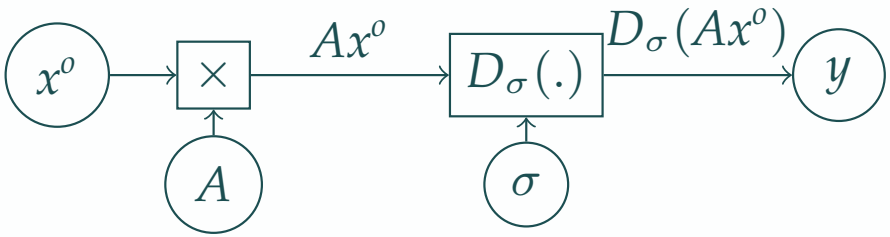
TP (avec code fourni + exemples de résultats)

- (TP précédent/déconv. 1D : coder opérateur convolution, comparer tps de calcul.)
- Débruitage d'image :
  - modèle direct : $y = x + b$ où x est une image (2D !) et b est un bruit blanc gaussien
  - hypothèse : la représentation en ondelette Wx de x est parcimonieuse
  - problème inverse :
$$\arg\min_x \|x - y\|_F^2 + \lambda \|Wx\|_1$$
  - problème équivalent : changement de variable $z = Wx$
$$\arg\min_z \left\|W^T z - y\right\|_F^2 + \lambda \|z\|_1$$
  - TODO : coder W^T comme opérateur (attention, W doit correspondre à une base orthonormée), simuler et résoudre le problème
    Aide : vectoriser l'image en entrée et sortie de l'opérateur ($\mathbb{R}^N \to \mathbb{R}^N$ où N est le nombre total de pixels de l'image).
- Déconvolution d'image ($y = x * h + b$) avec parcimonie sur Wx
- Inpainting d'image ($y = x[mask] + b$) avec parcimonie sur Wx
- Déconvolution + inpainting + débruitage d'image

Sommaire

Conclusion

Modèle direct : $y = D_\sigma(Ax^o)$



Problème inverse

$$\arg\min_x \underbrace{E(Ax, y)}_{\text{attache aux données en fonction de } D_\sigma} + \underbrace{R(x)}_{\text{régularisation selon hypothèse sur } x^o}$$

Grande variété de problèmes inverses, flexibilité de modélisation et de mise en œuvre :

- x^o : signal (1D), image (2D), etc.
- Régularisation : selon les hypothèses sur x^o (parcimonie, etc.)
- Bruit/perturbation D_σ :
  - en fonction de la physique du modèle direct
  - choisir une attache aux données $E(.,.)$ appropriée
- Opérateur linéaire A :
  - sous forme matricielle ou non (transformée rapide, convolution, échantillonnage, etc.)
  - combinaison possible d'opérateurs (inpainting, convolution, représentation, etc.)
- Algorithmes :
  - ISTA : vu pour le Lasso, existe dans un cadre plus général + version rapide (FISTA)
  - nombreux autres algorithmes d'optimisation convexe
- Toolboxes : `scilops`, `deepinv`, etc.
- *À venir : problème inverse et réseaux de neurones !*