Déconvolution aveugle

(17 avril 2015)

Objectif On s'intéresse à un signal filtré et bruité, qu'on cherche à retrouver. Le modèle adopté est le suivant : si on note u le signal parfait, K le noyau de convolution (ou filtre) et n un bruit (blanc gaussien), alors le signal observé g est généré par

$$g = K \star u + n$$

où on supposera par ailleurs que le noyau K est de support compact de petite taille connue (typiquement contenu dans un carré de côté de longueur $2\,M$ avec $M \approx \{10, 20\}$). L'objectif final est d'estimer à la fois u et K connaissant g : on parle alors de *déconvolution aveugle*.

Quelques rappels sur la convolution discrète et la DFT

On rappelle dans ce paragraphe quelques notions théoriques utiles pour la suite. Pour des raisons numériques (utilisation de la FFT), on supposera que les images $u = (u_{i,j})_{\substack{i\in[\![\,0\,;\,N_x-1\,]\!]\\ j\in[\![\,0\,;\,N_y-1\,]\!]}}$ et $g = (g_{i,j})_{\substack{i\in[\![\,0\,;\,N_x-1\,]\!]\\ j\in[\![\,0\,;\,N_y-1\,]\!]}}$ sont de taille paire.

Produit de convolution discret On se place dans le domaine discret ici, ce qui conduit à considérer le produit de convolution discret

$$\forall\,(i,j) \in [\![\,0\,;\,N_x-1\,]\!] \times [\![\,0\,;\,N_y-1\,]\!]\,, \qquad (a\star b)_{i,j} = \sum_{m=0}^{N_x-1}\sum_{n=0}^{N_y-1} a_{m,n}\,\tilde{b}_{i-m,j-n}$$

où $\tilde{b}$ est la périodisée de b, c'est-à-dire où pour tout $(i,j) \in [\![\,0\,;\,N_x-1\,]\!] \times [\![\,0\,;\,N_y-1\,]\!]$, on a $b_{i,j} = \tilde{b}_{i+k\,N_x,j+\ell\,N_y}$ pour tout $(k,\ell)\in\mathbb{N}^2$. On rappelle aussi que le produit de convolution est symétrique :

$$\forall\,(i,j) \in [\![\,0\,;\,N_x-1\,]\!] \times [\![\,0\,;\,N_y-1\,]\!]\,, \qquad (a\star b)_{i,j} = (b\star a)_{i,j} = \sum_{m=0}^{N_x-1}\sum_{n=0}^{N_y-1} b_{m,n}\,\tilde{a}_{i-m,j-n}.$$

Transformée de Fourier discrète (DFT) Étant donné un signal 2D a, on note $\hat{a}$ sa DFT, définie par

$$\forall\,(k,\ell) \in \left[\!\!\left[-\frac{N_x}{2}\,;\,\frac{N_x}{2}-1 \right]\!\!\right] \times \left[\!\!\left[-\frac{N_y}{2}\,;\,\frac{N_x}{2}-1 \right]\!\!\right], \qquad \hat{a}_{k,\ell} = \frac{1}{N_x\,N_y}\sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1} a_{m,n}\,\exp\left(-\frac{2\,\mathrm{i}\,\pi k\,i}{N_x}\right)\exp\left(-\frac{2\,\mathrm{i}\,\pi \ell\,j}{N_y}\right)$$

La DFT est inversible ; la transformée de Fourier discrète inverse (iDFT) est donnée par

$$\forall\,(i,j) \in [\![\,0\,;\,N_x-1\,]\!] \times [\![\,0\,;\,N_y-1\,]\!]\,, \qquad a_{i,j} = \sum_{k=-N_x/2}^{N_x/2-1}\sum_{\ell=-N_y/2}^{N_y/2-1} \hat{a}_{k,\ell}\,\exp\left(\frac{2\,\mathrm{i}\,\pi k\,i}{N_x}\right)\exp\left(\frac{2\,\mathrm{i}\,\pi \ell\,j}{N_y}\right)$$

Remarque : Attention, Matlab adopte des conventions différentes ; les indices des signaux sont compris entre 1 et N_x, N_y, et les DFTs sont périodisées de sorte de ne considérer que des indices positifs (avec toujours un décalage d'indice pour avoir des indices strictement positifs).

Produit de convolution et DFT On rappelle que le produit de convolution dans le domaine spatial devient un produit simple dans le domaine de Fourier ; plus précisément,

$$c = a\star b \qquad \Longleftrightarrow \qquad \hat{c} = \hat{a}\,\hat{b}.$$

En effet, on a par définition

$$\forall\,(k,\ell) \in \left[\!\!\left[-\frac{N_x}{2}\,;\,\frac{N_x}{2}-1 \right]\!\!\right] \times \left[\!\!\left[-\frac{N_y}{2}\,;\,\frac{N_x}{2}-1 \right]\!\!\right], \qquad \hat{c}_{k,\ell} = \frac{1}{N_x\,N_y}\sum_{m=0}^{N_x-1}\sum_{m=0}^{N_y-1} c_{m,n}\,\exp\left(-\frac{2\,\mathrm{i}\,\pi k\,m}{N_x}\right)\exp\left(-\frac{2\,\mathrm{i}\,\pi \ell\,n}{N_y}\right).$$

Ainsi, en utilisant la définition du produit de convolution,

$$\begin{aligned}\hat{c}_{k,\ell} &= \frac{1}{N_x\,N_y}\sum_{m=0}^{N_x-1}\sum_{m=0}^{N_y-1}\left(\sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1} a_{i,j}\,\tilde{b}_{m-i,n-j}\right)\exp\left(-\frac{2\,\mathrm{i}\,\pi k\,m}{N_x}\right)\exp\left(-\frac{2\,\mathrm{i}\,\pi \ell\,n}{N_y}\right)\\ &= \frac{1}{N_x\,N_y}\sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1}\left(\sum_{m=0}^{N_x-1}\sum_{m=0}^{N_y-1} a_{i,j}\,\tilde{b}_{m-i,n-j}\,\exp\left(-\frac{2\,\mathrm{i}\,\pi k\,m}{N_x}\right)\exp\left(-\frac{2\,\mathrm{i}\,\pi \ell\,n}{N_y}\right)\right).\end{aligned}$$

Puisque $m = i + m - i$ et que $n = j + n - j$, on obtient grâce à la propriété de morphisme de l'exponentielle que

$$\hat{c}_{k,\ell} = \frac{1}{N_x\,N_y} \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} \left(\sum_{m=0}^{N_x-1} \sum_{m=0}^{N_y-1} a_{i,j}\, \tilde{b}_{m-i,n-j}\, \exp\left(-\frac{2\,\mathrm{i}\,\pi k\, i}{N_x}\right) \exp\left(-\frac{2\,\mathrm{i}\,\pi k\,(m-i)}{N_x}\right) \right.$$
$$\left. \exp\left(-\frac{2\,\mathrm{i}\,\pi \ell\, j}{N_y}\right) \exp\left(-\frac{2\,\mathrm{i}\,\pi \ell\,(n-j)}{N_y}\right) \right)$$
$$= \frac{1}{N_x\,N_y} \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} \left(a_{i,j}\, \exp\left(-\frac{2\,\mathrm{i}\,\pi k\, i}{N_x}\right) \exp\left(-\frac{2\,\mathrm{i}\,\pi \ell\, j}{N_y}\right) \right.$$
$$\left. \sum_{m=0}^{N_x-1} \sum_{m=0}^{N_y-1} \tilde{b}_{m-i,n-j}\, \exp\left(-\frac{2\,\mathrm{i}\,\pi k\,(m-i)}{N_x}\right) \exp\left(-\frac{2\,\mathrm{i}\,\pi \ell\,(n-j)}{N_y}\right) \right).$$

La périodicité assure alors que

$$\sum_{m=0}^{N_x-1} \sum_{m=0}^{N_y-1} \tilde{b}_{m-i,n-j}\, \exp\left(-\frac{2\,\mathrm{i}\,\pi k\,(m-i)}{N_x}\right) \exp\left(-\frac{2\,\mathrm{i}\,\pi \ell\,(n-j)}{N_y}\right) = \sum_{m=0}^{N_x-1} \sum_{m=0}^{N_y-1} \tilde{b}_{m,n}\, \exp\left(-\frac{2\,\mathrm{i}\,\pi k\, m}{N_x}\right) \exp\left(-\frac{2\,\mathrm{i}\,\pi \ell\, n}{N_y}\right)$$

ce qui permet de calculer indépendamment les deux doubles sommes, et obtenir le résultat désiré.

Égalité de Parseval L'égalité de Parseval assure une convervation de l'énergie entre le domaine spatial et le domaine de Fourier ; plus précisément, il montre que

$$\|\hat{a}\|^2 = \frac{1}{N_x\,N_y} \|a\|^2 \qquad \text{soit} \qquad \sum_{k=-N_x/2}^{N_x/2-1} \sum_{\ell=-N_y/2}^{N_y/2-1} \hat{a}_{k,\ell}{}^2 = \frac{1}{N_x\,N_y} \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} a_{i,j}{}^2.$$

Signalons également le résultat suivant

$$\langle \hat{a}, \hat{b} \rangle = \frac{1}{N_x\,N_y} \langle a, b \rangle \qquad \text{soit} \qquad \sum_{k=-N_x/2}^{N_x/2-1} \sum_{\ell=-N_y/2}^{N_y/2-1} \hat{a}_{k,\ell}\, \overline{\hat{b}}_{k,\ell} = \frac{1}{N_x\,N_y} \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} a_{i,j}\, b_{i,j}$$

où on note $\overline{\hat{a}}$ le conjugué complexe de $\hat{a}$, qui est complexe (et où par hypothèse a et b sont réels).

1 Estimation du noyau

1.1 Approche variationnelle

Dans cette partie, on suppose que l'on connaît à la fois u et g, et on cherche donc à estimer K. On adopte ici une approche variationnelle : on cherche à minimiser l'énergie

$$E(\rho) = \lambda\,\|\rho\|^2 + \frac{\mu}{2}\,\|\rho \star u - g\|^2$$

que l'on minimise sous la contrainte que le support $\mathrm{supp}(\rho)$ soit contenu dans un carré de côté $2\,M$. Cette énergie comporte un premier terme de régularisation du noyau, et un second terme d'attache aux données ; ces deux termes sont des termes quadratiques.

Remarque : On peut remarquer que, grâce au théorème de Parseval, on peut écrire cette énergie dans le domaine de Fourier, ce qui donne, en notant par un chapeau la DFT (transformée de Fourier discrète) des signaux,

$$E(\rho) = \bar{E}(\hat{\rho}) = \frac{\lambda}{N_x\,N_y}\,\|\hat{\rho}\|^2 + \frac{\mu}{2\,N_x\,N_y}\,\|\hat{\rho}\,\hat{u} - \hat{g}\|^2$$

ce qui nous conduit à considérer $\tilde{\lambda} = \lambda/N_x/N_y$ et $\tilde{\mu} = \mu/N_x/N_y$.

Support de ρ Bien que la contrainte sur ρ nous permet de travailler avec des ρ de taille $(2\,M)^2$, numériquement, il faut plonger ρ dans une image plus grande, de la taille des autres signaux u et g. Il existe un grand nombre de façon de procéder, mais il nous suffit de savoir comment le faire pour une masse de Dirac pour pouvoir le faire correctement pour tous les noyaux possibles.

Considérons donc la masse de Dirac centrée en $(0,0)$:

$$(\delta_{(0,0)})_{i,j} = \begin{cases} 1 & \text{si } (i,j) = (0,0) \\ 0 & \text{sinon.} \end{cases}$$

Il est clair que $\delta_{(0,0)} \star u = u$. Ainsi, on en déduit que le support des noyaux doit être centré en $(0,0)$; on peut donc par exemple choisir $S_M = [\![-M\,;\, M-1]\!] \times [\![-M\,;\, M-1]\!]$, qui est centré en $(0,0)$ et de taille voulue.

Espace des noyaux On suppose dans notre modèle que les noyaux K recherchés sont de support contenu dans le carré S_M défini plus haut et tels que leur restriction sur S_M appartient au simplexe $\Sigma_{(2M)^2}$. On rappelle qu'il s'agit de l'ensemble (convexe) des signaux de taille $(2\,M)^2$ à coefficients positifs et dont la somme vaut 1 ; on sait par ailleurs calculer la projection sur ce convexe[1].

Problème étudié On cherche donc à résoudre le problème suivant

$$\min_{\substack{\rho=(\rho_{i,j})_{\substack{i\in[\![\,0\,;\,N_x-1\,]\!]\\ j\in[\![\,0\,;\,N_y-1\,]\!]}}\\ \mathrm{supp}(\rho)\subset S_M\\ \rho_{|S_M}\in\Sigma_{(2M)^2}}} \left\{\lambda\,\|\rho\|^2+\frac{\mu}{2}\,\|\rho\star u-g\|^2\right\} \tag{1}$$

que l'on peut encore écrire comme un problème sans contrainte

$$\min_{\substack{\rho=(\rho_{i,j})_{\substack{i\in[\![\,0\,;\,N_x-1\,]\!]\\ j\in[\![\,0\,;\,N_y-1\,]\!]}}}} \left\{\lambda\,\|\rho\|^2+\frac{\mu}{2}\,\|\rho\star u-g\|^2+\chi_{S_M\cap\Sigma_{(2M)^2}}(\rho)\right\}$$

l'on note de manière abusive $S_M\cap\Sigma_{(2M)^2}$ l'ensemble des ρ tels que $\mathrm{supp}(\rho)\subset S_M$ et tels que $\rho_{|S_M}\in\Sigma_{(2M)^2}$. Posons

$$f(\rho)=\frac{\mu}{2}\,\|\rho\star u-g\|^2 \qquad \text{et} \qquad g(\rho)=\chi_{S_M\cap\Sigma_{(2M)^2}}(\rho).$$

On reconnaît alors un problème de la forme de ceux résolubles par la méthode de projection de Dykstra[2]. En particulier, on sait que si on considère les conjuguées convexes des fonctions f et g, alors le problème (1) est équivalent au problème dual

$$\min_{P=(p_1,p_2)} \left\{ \underbrace{\frac{1}{4\,\lambda}\,\|p_1+p_2\|^2}_{=F(P)} + \underbrace{f^*(p_1)+g^*(p_2)}_{=G(P)} \right\}. \tag{2}$$

Démonstration : Puisque les fonctions f et g sont convexes, s.c.i. et propres, on peut écrire pour tout $\rho\in\mathbb{R}^{N_xN_y}$

$$\lambda\,\|\rho\|^2+f(\rho)+g(\rho)=\lambda\,\|\rho\|^2+\sup_{p_1\in\mathbb{R}^{N_xN_y}}\left\{\langle\rho,p_1\rangle-f^*(p_1)\right\}+\sup_{p_2\in\mathbb{R}^{N_xN_y}}\left\{\langle\rho,p_2\rangle-g^*(p_2)\right\}.$$

On en déduit que le problème (1) est équivalent au problème de point selle

$$\min_{\rho\in\mathbb{R}^{N_xN_y}}\ \sup_{\substack{p_1\in\mathbb{R}^{N_xN_y}\\ p_2\in\mathbb{R}^{N_xN_y}}} \left\{\lambda\,\|\rho\|^2+\langle\rho,p_1+p_2\rangle-f^*(p_1)-g^*(p_2)\right\}$$

qui s'écrit également de manière équivalente

$$\max_{\substack{p_1\in\mathbb{R}^{N_xN_y}\\ p_2\in\mathbb{R}^{N_xN_y}}}\ \inf_{\rho\in\mathbb{R}^{N_xN_y}} \left\{\lambda\,\|\rho\|^2+\langle\rho,p_1+p_2\rangle-f^*(p_1)-g^*(p_2)\right\}.$$

Or, $$\lambda\,\|\rho\|^2+\langle\rho,p_1+p_2\rangle=\lambda\left(\|\rho\|^2+2\left\langle\rho,\frac{p_1+p_2}{2\,\lambda}\right\rangle\right)=\lambda\left(\left\|\rho+\frac{p_1+p_2}{2\,\lambda}\right\|^2-\left\|\frac{p_1+p_2}{2\,\lambda}\right\|^2\right)$$

il en découle donc

$$\begin{aligned}\inf_{\rho\in\mathbb{R}^{N_xN_y}}\left\{\lambda\,\|\rho\|^2+\langle\rho,p_1+p_2\rangle-f^*(p_1)-g^*(p_2)\right\}&=\inf_{\rho\in\mathbb{R}^{N_xN_y}}\left\{\lambda\left(\left\|\rho+\frac{p_1+p_2}{2\,\lambda}\right\|^2-\left\|\frac{p_1+p_2}{2\,\lambda}\right\|^2\right)-f^*(p_1)-g^*(p_2)\right\}\\&=\inf_{\rho\in\mathbb{R}^{N_xN_y}}\left\{\lambda\left\|\rho+\frac{p_1+p_2}{2\,\lambda}\right\|^2\right\}-\frac{1}{4\,\lambda}\|p_1+p_2\|^2-f^*(p_1)-g^*(p_2)\\ \inf_{\rho\in\mathbb{R}^{N_xN_y}}\left\{\lambda\,\|\rho\|^2+\langle\rho,p_1+p_2\rangle-f^*(p_1)-g^*(p_2)\right\}&=-\frac{1}{4\,\lambda}\|p_1+p_2\|^2-f^*(p_1)-g^*(p_2)\end{aligned}$$

et le problème (1) est alors équivalent à

$$\max_{\substack{p_1\in\mathbb{R}^{N_xN_y}\\ p_2\in\mathbb{R}^{N_xN_y}}}\left\{-\frac{1}{4\,\lambda}\|p_1+p_2\|^2-f^*(p_1)-g^*(p_2)\right\} \iff \min_{\substack{p_1\in\mathbb{R}^{N_xN_y}\\ p_2\in\mathbb{R}^{N_xN_y}}}\left\{\frac{1}{4\,\lambda}\|p_1+p_2\|^2+f^*(p_1)+g^*(p_2)\right\}$$

avec $\rho^*=-\dfrac{{p_1}^*+{p_2}^*}{2\,\lambda}$. ■

[1] cf. *Projection sur le simplexe.*
[2] cf. *Méthodes proximales.*

Notons par ailleurs que la fonction F est de gradient $(1/\lambda)$-lipschitzien, avec

$$\nabla F(P) = \begin{pmatrix} (p_1+p_2)/(2\,\lambda) \\ (p_1+p_2)/(2\,\lambda) \end{pmatrix}.$$

En effet, ∇F est linéaire, et on a pour tout $P=(p_1,p_2)$

$$\|\nabla F(P)\|^2 = 2\,\frac{(p_1+p_2)^2}{(2\,\lambda)^2} \leq \frac{2\,({p_1}^2+{p_2}^2)}{2\,\lambda^2} \leq \frac{\|P\|^2}{\lambda^2}$$

où la majoration est optimale (il suffit de considérer $P=(1,1)$ par exemple).

Résolution numérique Pour résoudre le problème initial (1), il suffit donc de résoudre le problème dual (2), qui est résoluble par minimisation alternée par exemple[3] :

$$(p_1)_0, (p_2)_0 \qquad \text{et} \qquad \forall\, k \in \mathbb{N}, \qquad \left\{ \begin{array}{rcl} (p_1)_{k+1} & := & \displaystyle\operatorname*{argmin}_{p_1 \in \mathbb{R}^{N_x N_y}} \left\{ \frac{1}{4\,\lambda} \|p_1 + (p_2)_k\|^2 + f^*(p_1) + g^*((p_2)_k) \right\} \\ (p_2)_{k+1} & := & \displaystyle\operatorname*{argmin}_{p_2 \in \mathbb{R}^{N_x N_y}} \left\{ \frac{1}{4\,\lambda} \|(p_1)_{k+1} + p_2\|^2 + f^*((p_1)_{k+1}) + g^*(p_2) \right\} \end{array} \right.$$

où on reconnaît des opérateurs proximaux :

$$(p_1)_0, (p_2)_0 \qquad \text{et} \qquad \forall\, k \in \mathbb{N}, \qquad \left\{ \begin{array}{rcl} (p_1)_{k+1} & := & \operatorname{prox}_{2\lambda f^*}(-(p_2)_k) \\ (p_2)_{k+1} & := & \operatorname{prox}_{2\lambda g^*}(-(p_1)_{k+1}) \end{array} \right.$$

Pour assurer la convergence de cet algorithme, on a recours à une sur-relaxation de type FISTA :

$$(p_1)_0, (\bar{p}_2)_0 = (p_2)_0 \qquad \text{et} \qquad \forall\ \in \mathbb{N}, \qquad \left\{ \begin{array}{rcl} (p_1)_{k+1} & := & \operatorname{prox}_{2\lambda f^*}(-(\bar{p}_2)_k) \\ (p_2)_{k+1} & := & \operatorname{prox}_{2\lambda g^*}(-(p_1)_{k+1}) \\ (\bar{p}_2)_{k+1} & := & (1-\lambda_k)\,(p_2)_k + \lambda_k (p_2)_{k+1}. \end{array} \right.$$

Calcul des opérateurs proximaux On voit donc qu'il faut pouvoir calculer $\operatorname{prox}_{2\lambda g^*}$ et $\operatorname{prox}_{2\lambda f^*}$; or, l'identité de Moreau assure que

$$\operatorname{prox}_{2\lambda f^*}(p_1) = p_1 - 2\,\lambda\, \operatorname{prox}_{f/(2\lambda)}(p_1/(2\lambda)) \qquad \text{et} \qquad \operatorname{prox}_{2\lambda g^*}(p_2) = p_2 - 2\,\lambda\, \operatorname{prox}_{g/(2\lambda)}(p_2/(2\lambda)).$$

L'opérateur $\operatorname{prox}_{g/(2\lambda)}$ est facile à calculer : il s'agit de la projection sur $S_M \cap \Sigma_{(2M)^2}$, qui s'écrit

$$\operatorname{proj}_{S_M \cap \Sigma_{(2M)^2}}(p_2) = \operatorname*{argmin}_{p \in S_M \cap \Sigma_{(2M)^2}} \left\{ \frac{1}{2}\, \|p - p_2\|^2 \right\} = \operatorname*{argmin}_{p \in S_M \cap \Sigma_{(2M)^2}} \left\{ \frac{1}{2}\, \|p_{|S_M} - {p_2}_{|S_M}\|^2 + \frac{1}{2}\, \|p_{|{}^cS_M} - {p_2}_{|{}^cS_M}\|^2 \right\}$$

Puisque la restriction à ${}^cS_M$ de p est nulle, on en déduit que, si p^* est la projection de p_2 sur $S_M \cap \Sigma_{(2M)^2}$, alors

$$p_{|S_M} = \operatorname{proj}_{\Sigma_{(2M)^2}} \left\{ \frac{1}{2}\, \|p_{|S_M} - {p_2}_{|S_M}\|^2 \right\} \qquad \text{et} \qquad p_{|{}^cS_M} = 0.$$

En d'autres termes, on annule les valeurs de p_2 en dehors de S_M puis on projete ce qui reste sur Σ_M. Calculons à présent $\operatorname{prox}_{f/(2\lambda)}$. Par la caractérisation du point proximal, on a

$$p = \operatorname{prox}_{f/(2\lambda)}(p_1) \qquad \Longleftrightarrow \qquad p_1 - p = \frac{1}{2\,\lambda}\, \nabla f(p).$$

Cela nous amène à calculer $\nabla f(p)$. Soit $h = (h_{i,j})_{\substack{i \in [\![\,0\,;\,N_x-1\,]\!] \\ j \in [\![\,0\,;\,N_y-1\,]\!]}}$. On va passer dans le domaine de Fourier :

$$\begin{aligned} f(p+h) &= \frac{\mu}{2}\, \|(p+h) \star u - g\|^2 \\ &= \frac{\mu\, N_x\, N_y}{2}\, \|(\hat{p}+\hat{h})\, \hat{u} - \hat{g}\|^2 \\ &= \frac{\mu\, N_x\, N_y}{2}\, \|\hat{p}\,\hat{u} - \hat{g} + \hat{h}\,\hat{u}\|^2 \\ &= \frac{\mu\, N_x\, N_y}{2}\, \left(\|\hat{p}\,\hat{u} - \hat{g}\|^2 + 2\,\langle \hat{p}\,\hat{u} - \hat{g}, \hat{h}\,\hat{u} \rangle + \|\hat{h}\,\hat{u}\|^2 \right) \\ &= f(p) + \mu\, N_x\, N_y\, \langle \hat{p}\,\hat{u} - \hat{g}, \hat{h}\,\hat{u} \rangle + o(\|\hat{h}\|) \\ &= f(p) + \mu\, N_x\, N_y\, \langle \overline{\hat{u}}\,(\hat{p}\,\hat{u} - \hat{g}), \hat{h} \rangle + o(\|\hat{h}\|) \\ f(p+h) &= f(p) + \mu\, \langle u' \star (p \star u - g), h \rangle + o(\|\hat{h}\|) \end{aligned}$$

[3] cf. section Minimisation alternée dans *Méthodes proximales*

où u' est la transformée de Fourier inverse de $\overline{\hat{u}}$ (ce qui correspond en fait à la symétrisée de u). On en déduit ainsi que $\nabla f(p) = \mu\, u' \star (p \star u - g)$, ce qui assure que

$$p = \mathrm{prox}_{f/(2\lambda)}(p_1) \qquad \Longleftrightarrow \qquad p_1 - p = \frac{\mu}{2\,\lambda}\, u' \star (p \star u - g).$$

Passons à nouveau dans le domaine de Fourier :

$$p_1 - p = \frac{\mu}{2\,\lambda}\, u' \star (p \star u - g) \qquad \Longleftrightarrow \qquad \hat{p}_1 - \hat{p} = \frac{\mu}{2\,\lambda}\, \overline{\hat{u}}\, (\hat{p}\,\hat{u} - \hat{g}) \qquad \Longleftrightarrow \qquad \hat{p} = \frac{2\,\lambda\,\hat{p}_1 + \mu\,\overline{\hat{u}}\,\hat{g}}{2\,\lambda + \mu\,\|\hat{u}\|^2}$$

Ainsi, pour obtenir $p = \mathrm{prox}_{f/(2\lambda)}(p_1)$, il suffit de calculer sa DFT par la formule précédente, puis de repasser dans le domaine spatial par DFT inverse. Finalement, l'algorithme devient

$$(p_1)_0, (\bar{p}_2)_0 = (p_2)_0, \qquad \text{et} \qquad \forall\, k \in \mathbb{N}, \qquad \left\{ \begin{array}{rcl} (p_1)_{k+1} & := & -(\bar{p}_2)_k - 2\,\lambda\ \mathrm{prox}_{f/(2\lambda)}\big(-(\bar{p}_2)_k/(2\lambda)\big) \\ (p_2)_{k+1} & := & -(p_1)_k - 2\,\lambda\ \mathrm{proj}_{S_M \cap \Sigma_{(2M)^2}}\big(-(p_1)_k/(2\lambda)\big) \\ (\bar{p}_2)_{k+1} & := & (1-\lambda_k)\,(p_2)_k + \lambda_k (p_2)_{k+1}. \end{array} \right.$$

1.2 Code Matlab

On commence par proposer une fonction qui permet d'estimer K selon la méthode décrite plus haut dans ce paragraphe.

```
function rho = estimation_rho(g,u,M,lambda,mu)

% FFT des signaux
fft_u = fft2(u) ;
fft_g = fft2(g) ;

% descente de gradient
[Nx,Ny] = size(g) ;
niter = 1000 ; % nombre d'itérations maximal
support_x = Nx/2+1-M:Nx/2+M ;
support_y = Ny/2+1-M:Ny/2+M ;

% initialisation
p2_bar = randn(Nx,Ny) ; p2_old = p2_bar ;
tkold = 1 ;

for n=1:niter
    % MàJ de p1
    % calcul de prox_f dans le domaine de Fourier
    hat_p2_bar = fft2(-p2_bar/2/lambda) ;
    hat_prox = (2*lambda*hat_p2_bar + mu*conj(fft_u).*fft_g)...
                ./(2*lambda + mu*real(conj(fft_u).*fft_u)) ;
    % retour dans le domaine spatial
    prox = real(ifft2(hat_prox)) ;
    % mise à jour
    p1 = - p2_bar - 2*lambda*prox ;

    % MàJ de p2
    % annulation en dehors du support et projection sur le simplexe
    p1_shift = fftshift(p1) ;
    proj = zeros(Nx,Ny) ;
    proj(support_x,support_y) = projection_simplexe(-p1_shift(support_x,support_y)/2/lambda) ;
    proj = fftshift(proj) ;
    % mise à jour
    p2 = - p1 - 2*lambda*proj ;

    % Relaxation (FISTA)
    tk = (1+sqrt(4*tkold+1))/2 ;
    relax = 1+(tkold-1)/tk ;
    tkold = tk ;
    p2_bar = relax*p2 + (1-relax)*p2_old ;
    p2_old = p2 ;

end

% reconstruction du noyau
rho = fftshift(-(p1 + p2)/2/lambda) ; rho = rho(support_x,support_y) ;

end
```

La projection sur le simplexe est quant à elle donnée par la fonction suivante :

```
function Xproj = projection_simplexe(X)

% on ordonne les valeurs de X (et on supprime les doublons)
breakpoints = unique(X(:)) ;

% calcul de t* : on cherche t* tel que la somme de (X-t*)^+ vaille 1
% on commence par évaluer sum(X-t)^+ en chaque point de rupture
[nb_occurrence,unused] = hist(X(:),breakpoints) ; nb_occurrence = nb_occurrence' ;
slopes = cumsum( nb_occurrence(end:-1:1) ) ;
   slopes = -slopes(end:-1:1) ;
values = cumsum( nb_occurrence(end:-1:1).*breakpoints(end:-1:1)) ;
   values = values(end:-1:1)+slopes.*breakpoints ;

% on cherche l'intervalle dans lequel sum(X-t)^+ = 1
i = find(values<1,1) ; % t* dans [X_i-1,X_i]
tstar = (1-values(i))/slopes(i) + breakpoints(i) ;

% projection
Xproj = (X-tstar).*(X-tstar>0) ;

end
```

1.3 Résultats

Voici des résultats obtenus avec cet algorithme sur des images en niveaux de gris. Les résultats sont présentés de la manière suivante : en haut, on a l'image u originale, puis sa version filtrée (et éventuellement bruitée) g, puis le noyau K ; en bas, on présente la valeur absolue de la différence entre les noyaux exacts et estimés K et ρ, puis la valeur absolue de la différence entre l'image g et $u \star \rho$, et enfin l'estimée ρ. L'initialisation est à chaque complètement aléatoire. On signalera enfin l'estimation prend moins d'une seconde, pour 1000 itérations.

Influence des paramètres On se place dans le cas non bruité pour l'instant. Les figures 1–7 présentent les résultats obtenus pour différents noyaux, avec différentes valeurs de paramètres λ et μ. Lorsque le paramètre μ est grand, les résultats sont meilleurs car on force davantage la fidélité aux données. Cependant, dans le cadre d'une déconvolution aveugle, si l'image u n'est pas connue, mais seulement estimée, une forte attache aux données risque de dégrader les résultats ; c'est pourquoi on préfèrera pondérer plus légèrement ce terme.

Influence du bruit On ajoute à présent deux niveaux de bruit blanc gaussien différents. L'estimation de ρ est évidemment dégradée et ce, d'autant plus que le bruit ajouté est important, mais reste tout de même acceptable, ainsi qu'on peut le voir dans les figures 8–9.

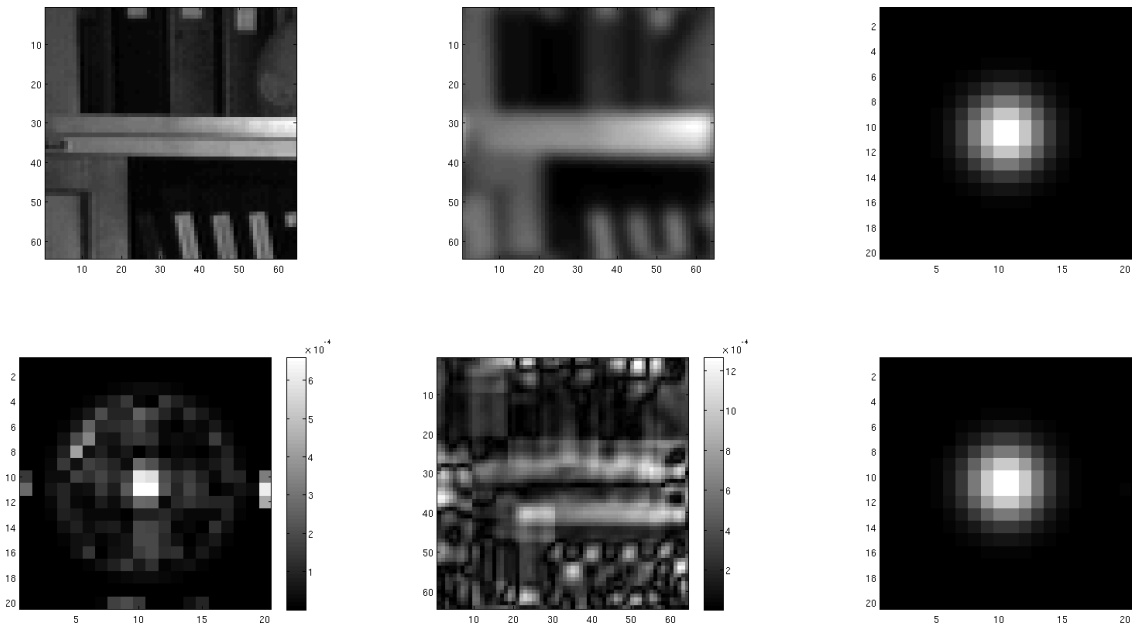

FIG. 1 – Noyau gaussien, pas de bruit, $\lambda = 1$, $\mu = 1$.

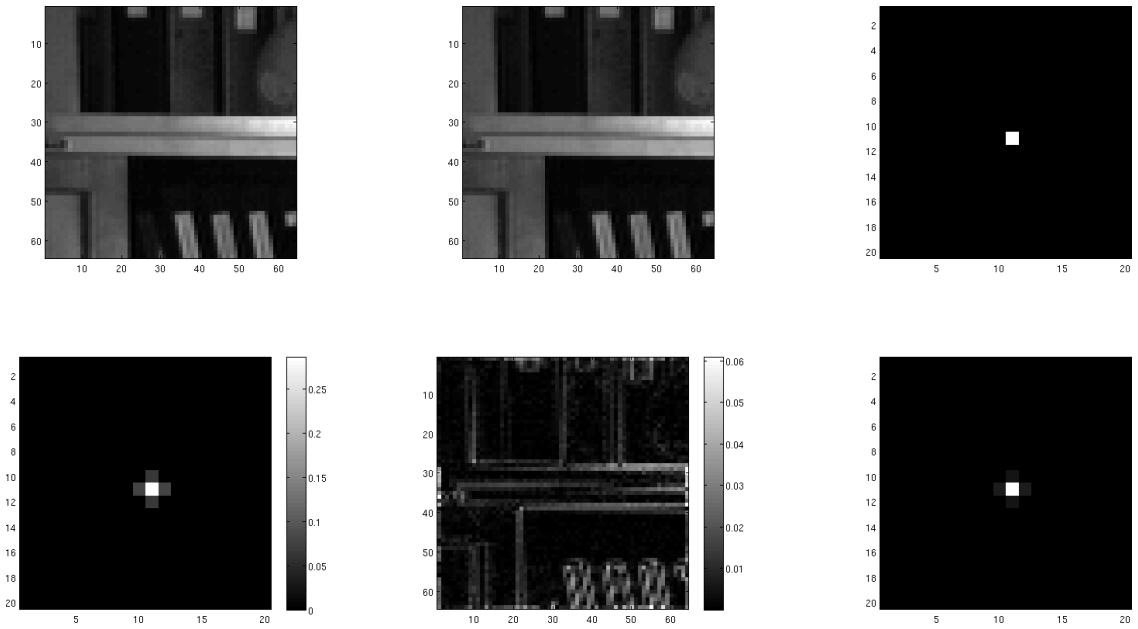

FIG. 2 – Noyau masse de Dirac, pas de bruit, $\lambda = 1$, $\mu = 1$.

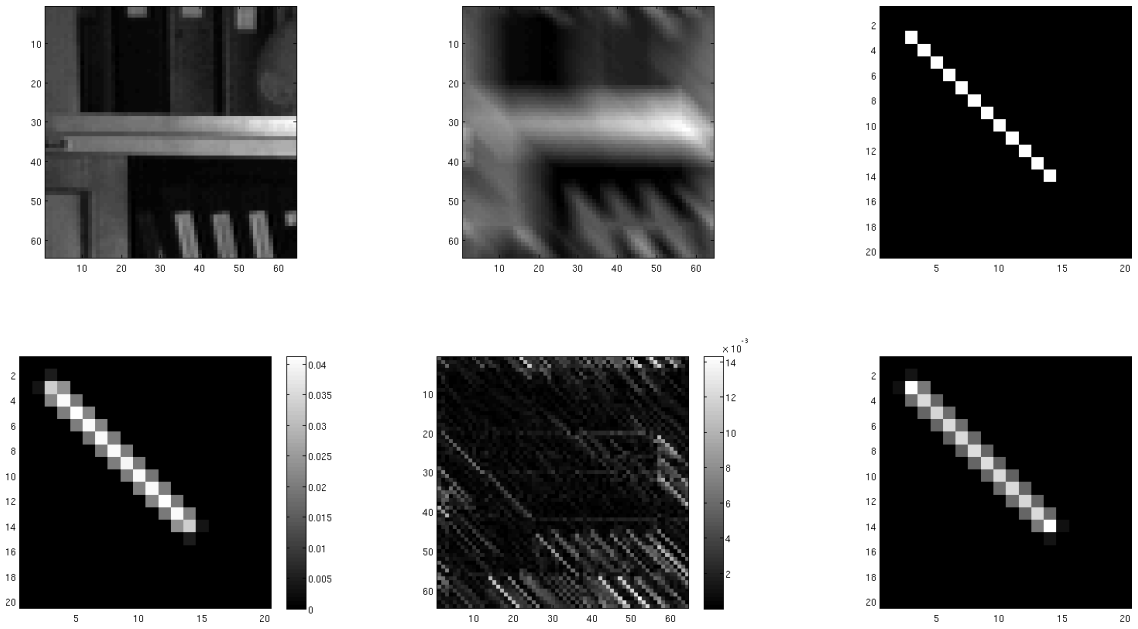

FIG. 3 – Noyau Trait, pas de bruit, $\lambda = 1$, $\mu = 1$.

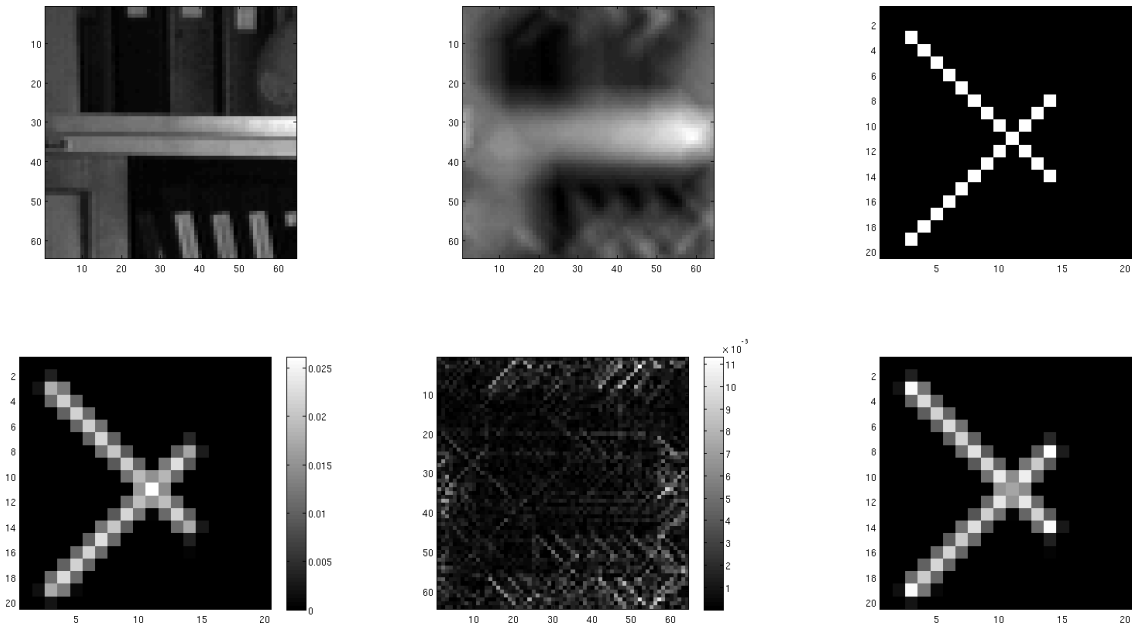

FIG. 4 – Noyau Croix, pas de bruit, $\lambda = 1$, $\mu = 1$.

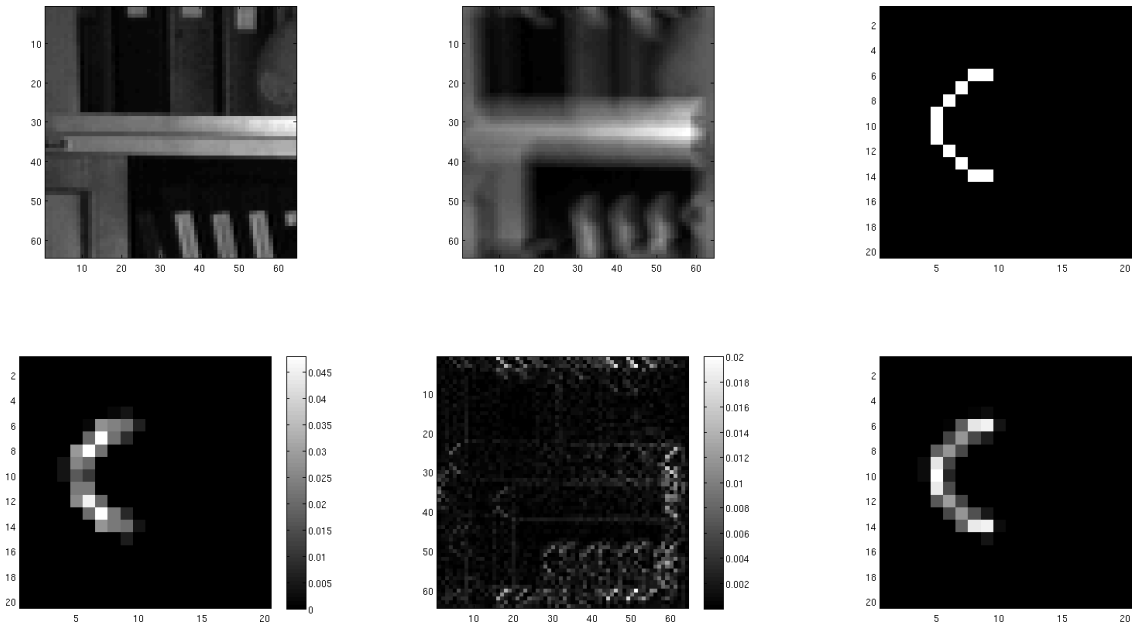

FIG. 5 – Noyau C, pas de bruit, $\lambda = 1$, $\mu = 1$.

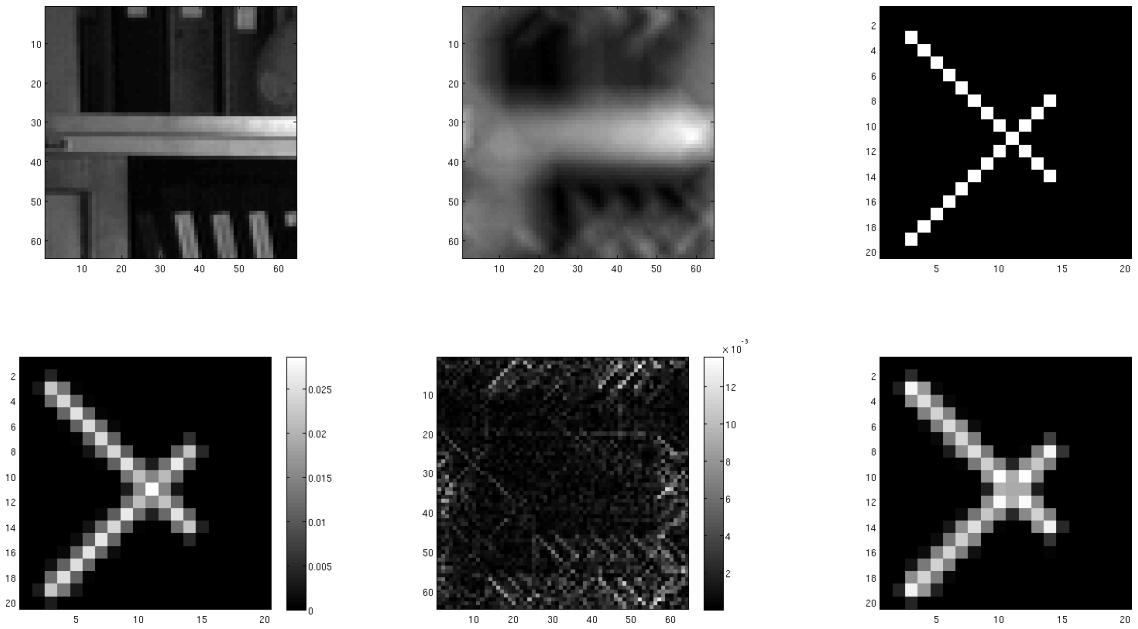

FIG. 6 – Noyau Croix, pas de bruit, $\lambda = 2$, $\mu = 1$.

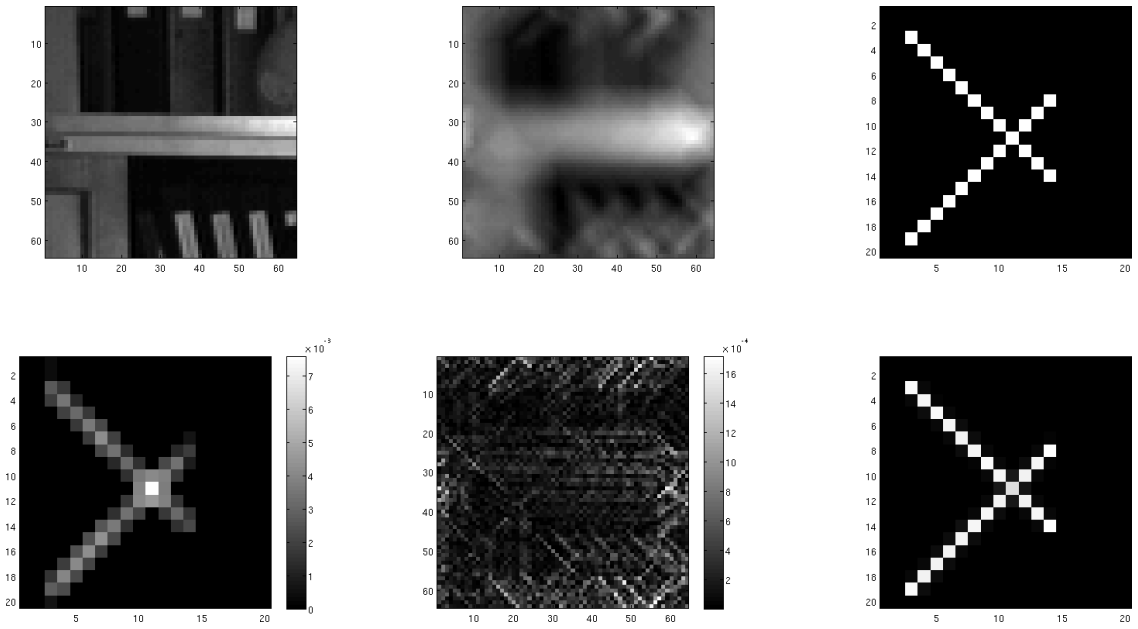

FIG. 7 – Noyau Croix, pas de bruit, $\lambda = 1$, $\mu = 20$.

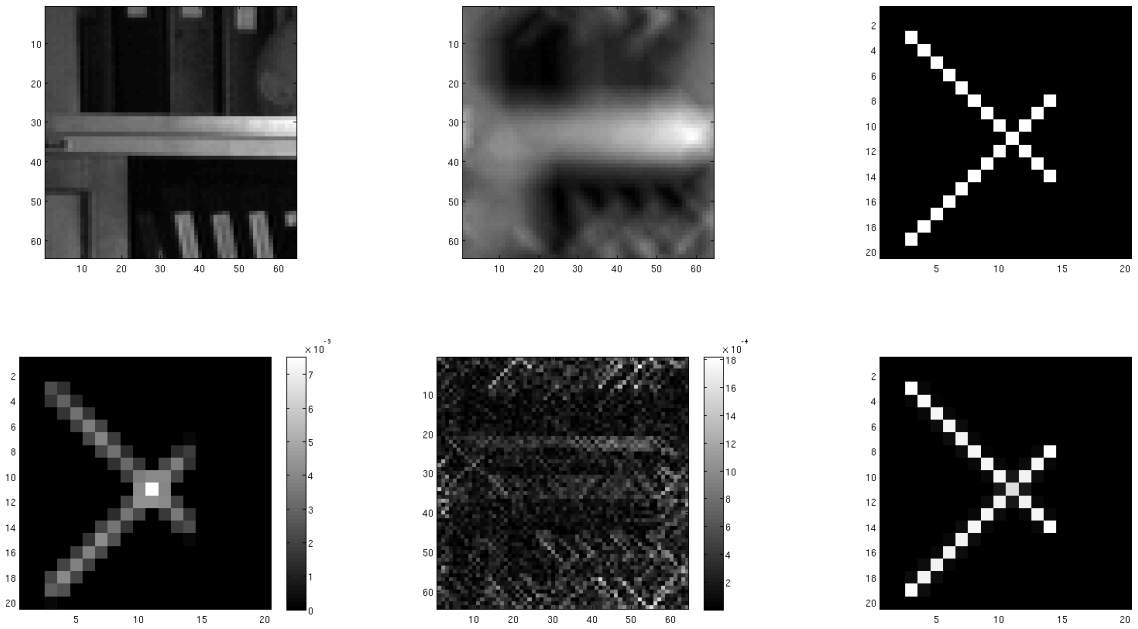

FIG. 8 – Noyau Croix, bruit de variance $\sigma = 10^{-4}$, $\lambda = 1$, $\mu = 20$.

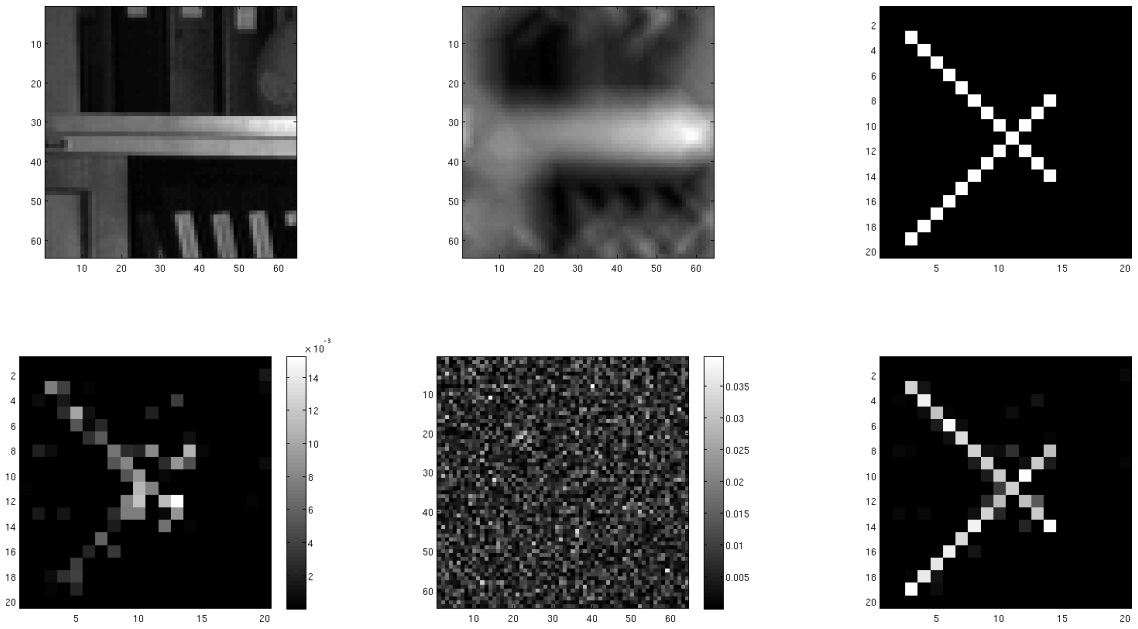

FIG. 9 – Noyau Croix, bruit de variance $\sigma = 10^{-2}$, $\lambda = 1$, $\mu = 20$.

Remarque : La méthode proposée pose un certain nombre de problèmes. Tout d'abord, le terme d'attache aux données est tel qu'on n'a aucun contrôle sur l'erreur (ce qui peut être une bonne chose si on n'a aucune idée sur la borne de cette erreur) ; ensuite, comment gérer intelligemment les images en couleur ? Au lieu de lancer cet algorithme indépendamment sur chaque canal couleur, on pourrait forcer le noyau à être le même sur chacun de ces canaux, ce qui devrait rendre la méthode plus robuste.

2 Estimation de l'image

On suppose dans cette section que K est connu, et on souhaite estimer l'image u à partir de g.

2.1 Régularisation TV avec contrainte sur v

Énergie et approche primal-duale Pour estimer l'image à partir de sa version convolée et bruitée et un noyau de convolution (exact ou estimé), on peut adopter une approche variationnelle. On souhaite résoudre le problème d'optimisation suivant :

$$\min_{\substack{v=(v_{i,j})_{i\in[\![0\,;N_x-1]\!]\\ j\in[\![0\,;N_y-1]\!]\\ v_{i,j}\in[0\,;1]}} \left\{\lambda\,\mathrm{TV}(v)+\frac{\mu}{2}\,\|K\star v-g\|^2\right\} \tag{3}$$

que l'on transforme en un problème primal-dual en écrivant la variation totale comme un problème dual :

$$\lambda\,\mathrm{TV}(v)=\lambda\sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1}\|(\nabla^h v)_{i,j}\|=\sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1}\sup_{\|p_{i,j}\|\le\lambda}\left\{\langle p_{i,j},(\nabla^h v)_{i,j}\rangle\right\}=\sup_{\substack{p=(p_{i,j})_{i\in[\![0\,;N_x-1]\!]\\ j\in[\![0\,;N_y-1]\!]\\ p_{i,j}=(p^x_{i,j},p^y_{i,j})\\ \|p_{i,j}\|\le\lambda}}\left\{\langle p,\nabla^h v\rangle\right\}$$

on note ∇^h l'opérateur (linéaire) gradient, défini par $\nabla^h v=(\partial_x v,\partial_y v)$, avec

$$(\partial_x v)_{i,j}=\begin{cases}v_{i+1,j}-v_{i,j} & \text{si } i<N_x-1\\ 0 & \text{sinon}\end{cases}\qquad\text{et}\qquad(\partial_y v)_{i,j}=\begin{cases}v_{i,j+1}-v_{i,j} & \text{si } j<N_y-1\\ 0 & \text{sinon.}\end{cases}$$

Ainsi, le problème (3) devient

$$\min_{\substack{v=(v_{i,j})_{i\in[\![0\,;N_x-1]\!]\\ j\in[\![0\,;N_y-1]\!]\\ v_{i,j}\in[0\,;1]}}\ \sup_{\substack{p=(p_{i,j})_{i\in[\![0\,;N_x-1]\!]\\ j\in[\![0\,;N_y-1]\!]\\ p_{i,j}=(p^x_{i,j},p^y_{i,j})\\ \|p_{i,j}\|\le\lambda}}\left\{\langle p,\nabla^h v\rangle+\frac{\mu}{2}\,\|K\star v-g\|^2\right\} \tag{4}$$

problème que l'on sait résoudre (par exemple en alternant descente de gradient projeté en v et montée de gradient projeté en p).

Résolution du problème (4) Pour résoudre le problème (4), on peut utiliser un algorithme primal-dual du type :

$$\bar v_0=v_0,p_0\qquad\text{et}\qquad\forall\,k\in\mathbb{N},\qquad\begin{cases}p_{k+1} &:=& \mathrm{proj}_{B(0,\lambda)}\left(p_k+\sigma\,\dfrac{\partial F}{\partial p}(\bar v_k,p_k)\right)\\ v_{k+1} &:=& \mathrm{proj}_{[0\,;1]^{N_xN_y}}\left(v_k-\tau\,\dfrac{\partial F}{\partial v}(v_k,p_{k+1})\right)\\ \bar v_{k+1} &:=& 2v_{k+1}-v_k\end{cases}$$

où on note $B(0,\lambda)$ la boule centrée en 0 et de rayon λ, et

$$F(p,v)=\langle p,\nabla^h v\rangle+\frac{\mu}{2}\,\|K\star v-g\|^2.$$

On a alors $$\frac{\partial F}{\partial p}(v,p)=\nabla^h v\qquad\text{et}\qquad\frac{\partial F}{\partial v}(v,p)=\mathrm{div}^h p+\mu\,\tilde K\star(K\star v-g)$$

où div^h désigne l'opérateur adjoint à ∇^h. Pour le calculer, on rappelle que $\nabla^h v=(\partial_x v,\partial_y v)$, donnés par les différences finies :

$$(\partial_x v)_{i,j}=\begin{cases}v_{i+1,j}-v_{i,j} & \text{si } i<N_x-1\\ 0 & \text{sinon}\end{cases}\qquad\text{et}\qquad(\partial_y v)_{i,j}=\begin{cases}v_{i,j+1}-v_{i,j} & \text{si } j<N_y-1\\ 0 & \text{sinon}\end{cases}$$

ce qui implique que $(\mathrm{div}^h)^*=(\partial_x)^*+(\partial_y)^*$. Calculons par exemple l'opérateur adjoint à ∂_x. Soient v et w deux matrices de taille $N_x\times N_y$ et considérons le produit scalaire $\langle\partial_x v,w\rangle$:

$$\langle \partial_x v, w\rangle = \sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1} (\partial_x v)_{i,j}\, w_{i,j} = \sum_{i=0}^{N_x-2}\sum_{j=0}^{N_y-1} (v_{i+1,j}-v_{i,j})\, w_{i,j} = \sum_{i=0}^{N_x-2}\sum_{j=0}^{N_y-1} v_{i+1,j}\, w_{i,j} - \sum_{i=0}^{N_x-2}\sum_{j=0}^{N_y-1} v_{i,j}\, w_{i,j}.$$

Un changement d'indice assure alors que

$$\langle \partial_x v, w\rangle = \sum_{i=1}^{N_x-1}\sum_{j=0}^{N_y-1} v_{i,j}\, w_{i-1,j} - \sum_{i=0}^{N_x-2}\sum_{j=0}^{N_y-1} v_{i,j}\, w_{i,j} = \sum_{j=0}^{N_y-1}\left(-v_{0,j}\, w_{0,j} + \sum_{i=1}^{N_x-2} v_{i,j}\,(w_{i-1,j}-w_{i,j}) + v_{N_x-1,j}\, w_{N_x-2,j}\right).$$

On en déduit que
$$((\partial_x)^* w)_{i,j} = \begin{cases} -w_{0,j} & \text{si } i=0 \\ w_{i-1,j}-w_{i,j} & \text{si } 0<i<N_x-1 \\ w_{N_x-2,j} & \text{si } i=N_x-1 \end{cases}$$

puis, de même, que
$$((\partial_y)^* w)_{i,j} = \begin{cases} -w_{i,0} & \text{si } j=0 \\ w_{i,j-1}-w_{i,j} & \text{si } 0<j<N_y-1 \\ w_{i,N_y-2} & \text{si } j=N_y-1. \end{cases}$$

L'algorithme choisi devient alors

$$\bar v_0 = v_0, p_0 \qquad \text{et} \qquad \forall\, k\in\mathbb{N}, \quad \begin{cases} p_{k+1} & := & \mathrm{proj}_{B(0,\lambda)}\left(p_k + \sigma\,\nabla^h \bar v_k\right) \\ v_{k+1} & := & \mathrm{proj}_{[0\,;1]^{N_x N_y}}\left(v_k - \tau\left(\mathrm{div}^h\, p_{k+1} + \mu\,\tilde K \star (K\star v_k - g)\right)\right) \\ \bar v_{k+1} & := & 2v_{k+1} - v_k \end{cases}$$

où les pas de temps τ et σ doivent être choisis correctement pour assurer la convergence[4].

Code Matlab Voici le code Matlab proposé pour la méthode présentée dans ce paragraphe.

```
function v = estimation_u_TV_contrainte01(g,rho,lambda,mu,v_init)

% FFT
% noyau
[M,unused] = size(rho) ; M = M/2 ;
[Nx,Ny,unused] = size(g) ;
rhoext = zeros(Nx,Ny) ;
     rhoext(Nx/2+1-M:Nx/2+M,Ny/2+1-M:Ny/2+M) = rho ;
     rhoext = fftshift(rhoext) ;
fft_rho = fft2(rhoext) ;

% signal
fft_g = fft2(g) ;

% algorithme primal-dual
% paramètres
mu = mu/20 ;
lambda = lambda/20 ;
tau = 1/(sqrt(8)+mu) ; % pas de temps
niter = 5000 ; % nombre d'itérations

% initialisation
if nargin<5
    v_init = zeros(Nx,Ny) ;
end
v_bar = v_init ;
v_old = v_bar ; v = v_old ;
px = zeros(Nx,Ny) ; py = zeros(Nx,Ny) ;
norm_p = sqrt(px.^2+py.^2) ;
px(norm_p>lambda) = lambda*px(norm_p>lambda)./norm_p(norm_p>lambda) ;
py(norm_p>lambda) = lambda*py(norm_p>lambda)./norm_p(norm_p>lambda) ;

for n=1:niter
    % MàJ de p
    % montée de gradient
    [dxu,dyu] = nablah(v_bar) ;
    px = px + tau*dxu ; py = py + tau*dyu ;
```

[4] Mais comment ? à priori $\sigma\,\tau \le 1/(\|\mathrm{div}^h\| + \mu)^2$

```
    % projection sur B(0,lambda)
    norm_p = sqrt(px.^2+py.^2) ;
    px(norm_p>lambda) = lambda*px(norm_p>lambda)./norm_p(norm_p>lambda) ;
    py(norm_p>lambda) = lambda*py(norm_p>lambda)./norm_p(norm_p>lambda) ;

    % MàJ de u
    % descente de gradient
    nablaf = divh(px,py) ;
    fft_v = fft2(v) ;
    fft_nablag = mu*conj(fft_rho).*(fft_rho.*fft_v-fft_g) ;
    nablag = real(ifft2(fft_nablag)) ;
    nablaF = nablaf + nablag ;
    % projection sur [0,1]
    v = min(max(v - tau*nablaF,0),1) ;

    % relaxation
    v_bar = 2*v - v_old;
    v_old = v;

end

end
```

Résultats Voici des résultats obtenus avec cet algorithme sur des images en niveaux de gris. Les résultats sont présentés de la manière suivante : en haut, on a l'image u originale, puis sa version filtrée (et éventuellement bruitée) g, puis le noyau K ; en bas, on présente l'estimée v, puis la valeur absolue de la différence entre l'image g et $v \star K$, et enfin la valeur absolue de la différence entre les images exactes et estimées u et v. L'initialisation est l'image nulle. On signalera enfin l'estimation prend moins de cinq secondes, pour 5000 itérations.

On se place ici dans le cas non bruité. Les figures **??**–**??** présentent les résultats obtenus pour différents noyaux, avec différentes valeurs de paramètres λ et μ. On voit en particulier que v est plus régularisé si μ est petit, et bien meilleur si μ est particulièrement grand.

2.2 Régularisation TV sans contrainte sur v

Énergie et approche primal-duale On ignore dans ce paragraphe la contrainte sur v, ce qui revient à considérer le problème suivant :

$$\min_{\substack{v=(v_{i,j})\\ i\in[\![0\,;N_x-1]\!]\\ j\in[\![0\,;N_y-1]\!]}} \left\{\lambda\,\mathrm{TV}(v) + \frac{\mu}{2}\left\|K\star v - g\right\|^2\right\} \tag{5}$$

qui s'écrit à nouveau comme un problème primal-dual, sans contrainte cette fois-ci :

$$\min_{\substack{v=(v_{i,j})\\ i\in[\![0\,;N_x-1]\!]\\ j\in[\![0\,;N_y-1]\!]}} \sup_{\substack{p=(p_{i,j})\\ i\in[\![0\,;N_x-1]\!]\\ j\in[\![0\,;N_y-1]\!]\\ p_{i,j}=(p^x_{i,j},p^y_{i,j})}} \left\{\langle p, \nabla^h v\rangle + \frac{\mu}{2}\left\|K\star v - g\right\|^2 + \chi_{B(0,\lambda)^{N_xN_y}}(p)\right\}. \tag{6}$$

Résolution numérique On résout le problème (6) par montée et descente de gradient (projeté) alternées. Pour mettre à jour p à partir de $\bar{v}_k$, on effectue une montée de gradient projeté correspondant au problème :

$$\min_{\substack{p=(p_{i,j})\\ i\in[\![0\,;N_x-1]\!]\\ j\in[\![0\,;N_y-1]\!]}} \left\{\langle p, \nabla^h v\rangle + \frac{\mu}{2}\left\|K\star v - g\right\|^2 + \chi_{B(0,\lambda)^{N_xN_y}}(p)\right\}.$$

ce qui donne

$$p_{k+1} = \mathrm{proj}_{B(0,\lambda)^{N_xN_y}}\left(p_k + \tau\nabla^h\bar{v}_k\right).$$

De même, pour mettre à jour v à partir de p_{k+1}, on effectue une descente de gradient implicite-explicite correspondant au problème suivant

$$\min_{\substack{v=(v_{i,j})\\ i\in[\![0\,;N_x-1]\!]\\ j\in[\![0\,;N_y-1]\!]}} \left\{\langle p_{k+1}, \nabla^h v\rangle + \frac{\mu}{2}\left\|K\star v - g\right\|^2 + \chi_{B(0,\lambda)^{N_xN_y}}(p_{k+1})\right\}$$

ce qui donne cette fois, si on note $f(v) = \mu\left\|K\star v - g\right\|^2/2$,

$$v_{k+1} = \mathrm{prox}_{\tau f}\left(v_k - \tau\ \mathrm{div}^h p_{k+1}\right)$$

avec

$$p = \mathrm{prox}_{\tau f}(x) \qquad \Longleftrightarrow \qquad x - p = \tau\,\mu\,\tilde{K}\star\left(K\star p - g\right)$$

soit, en passant dans le domaine de Fourier,

$$\hat{x} - \hat{p} = \tau\,\mu\,\bar{\hat{K}}\,\left(\hat{K}\,\hat{p} - \hat{g}\right) \qquad \Longleftrightarrow \qquad \hat{p} = \frac{\hat{x} + \tau\,\mu\,\bar{\hat{K}}\,\hat{g}}{1 + \tau\,\mu\,\bar{\hat{K}}\,\hat{K}}.$$

On en déduit l'algorithme suivant :

$$\bar{v}_0 = v_0, p_0 \qquad \text{et} \qquad \forall\, k \in \mathbb{N}, \quad \left\{ \begin{array}{rcl} p_{k+1} & := & \mathrm{proj}_{B(0,\lambda)^{N_x N_y}}\left(p_k + \tau \nabla^h \bar{v}_k\right) \\ \hat{v}_{k+1} & := & \dfrac{\hat{v}_k - \tau\,\widehat{\mathrm{div}^h\, p_{k+1}} + \tau\,\mu\,\bar{\hat{K}}\,\hat{g}}{1 + \tau\,\mu\,\bar{\hat{K}}\,\hat{K}} \\ \bar{v}_{k+1} & := & 2v_{k+1} - v_k \end{array} \right.$$

avec $\tau \leq 1/\|\nabla^h\| = 1/(2\sqrt{2})$.

Code Matlab Voici le code Matlab proposé pour la méthode présentée dans ce paragraphe.

```
function v = estimation_u_TV_noncontrainte01(g,rho,lambda,mu,v_init)

% FFT
% noyau
[M,unused] = size(rho) ; M = M/2 ;
[Nx,Ny,unused] = size(g) ;
rhoext = zeros(Nx,Ny) ;
rhoext = zeros(Nx,Ny) ;
     rhoext(Nx/2+1-M:Nx/2+M,Ny/2+1-M:Ny/2+M) = rho ;
     rhoext = fftshift(rhoext) ;
fft_rho = fft2(rhoext) ;

% signal
fft_g = fft2(g) ;

% algorithme primal-dual
% paramètres
tau = 1/sqrt(8) ; % pas de temps
niter = 2500 ; % nombre d'itérations

% initialisation
if nargin<5
    v_init = zeros(Nx,Ny) ;
end
v_bar = v_init ;
v_old = v_bar ; v = v_old ;
px = randn(Nx,Ny) ; py = randn(Nx,Ny) ;
norm_p = sqrt(px.^2+py.^2) ;
px(norm_p>lambda) = px(norm_p>lambda)./norm_p(norm_p>lambda) ;
py(norm_p>lambda) = py(norm_p>lambda)./norm_p(norm_p>lambda) ;

for n=1:niter
    % MàJ de p
    % montée de gradient
    [dxu,dyu] = nablah(v_bar) ;
    px = px + tau*dxu ; py = py + tau*dyu ;
    % projection sur B(0,lambda)
    norm_p = sqrt(px.^2+py.^2) ;
    px(norm_p>lambda) = px(norm_p>lambda)./norm_p(norm_p>lambda) ;
    py(norm_p>lambda) = py(norm_p>lambda)./norm_p(norm_p>lambda) ;

    % MàJ de u
    % descente de gradient
    tmp = v - tau*divh(px,py) ;
    % proximal
    fft_tmp = fft2(tmp) ;
    fft_v = (fft_tmp + tau*mu*conj(fft_rho).*fft_g)./(1+tau*mu*conj(fft_rho).*fft_rho) ;
    v = real(ifft2(fft_v)) ;

    % relaxation
    v_bar = 2*v - v_old ;
    v_old = v ;
```

```
53
54  end
55
56  end
```

Résultats On montre dans les figures **??**–**??** certains résultats obtenus avec cette méthode. Ce qui est intéressant d'observer ici est que les résultats sont similaires à ceux obtenus avec la méthode précédente (régularisation TV avec contrainte sur v), mais que l'algorithme est deux à trois fois plus rapide.

On montre également quelques résultats obtenus sur des images bruitées (figures **??** et **??**).

2.3 Régularisation non convexe

Optimisation concave L'objectif maintenant est de résoudre un problème de la forme

$$\min_{\substack{v=(v_{i,j})_{i\in[\![0\,;N_x-1]\!]\\ j\in[\![0\,;N_y-1]\!]\\ v_{i,j}\in[\,0\,;1\,]}} \left\{\lambda\, h(\nabla u) + \frac{\mu}{2}\,\|K\star v - g\|^2\right\} \tag{7}$$

où
$$h(p^x,p^y) = \begin{cases} \displaystyle\sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1} f({p^x_{i,j}}^2 + {p^y_{i,j}}^2) & \text{(cas non séparable)}\\ \displaystyle\sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1} f({p^x_{i,j}}^2) + f({p^y_{i,j}}^2) & \text{(cas séparable)}\end{cases}$$

avec f une fonction concave : $f(x) = \begin{cases}\sqrt{x}\\ \ln(1+\delta\,x)\\ \dfrac{\delta\,x}{1+\delta\,x}\end{cases}$ et $f'(x) = \begin{cases}\dfrac{1}{\sqrt{x}}\\ \dfrac{\delta}{1+\delta\,x}\\ \dfrac{\delta}{(1+\delta\,x)^2}\end{cases}$ avec $\delta>0$ un paramètre d'échelle.

On remarquera que le cas où f est la racine carrée correspond respectivement à la régularisation TV et TV séparable (et on est alors dans un cas de minimisation convexe). Le problème n'est pas convexe dans les autres cas car $x\mapsto f(x^2)$ est convexe au voisinage de zéro, mais concave lorsqu'on s'en éloigne suffisamment.

Introduction d'une variable auxiliaire Plaçons-nous pour l'instant dans le cas non séparable. Puisque la fonction f est choisie concave, son opposée est convexe. En particulier, on peut considérer sa conjuguée convexe :

$$-f(x) = \sup_{w\in\mathbb{R}}\Big\{x\,w - (-f)^*(w)\Big\} \qquad \text{soit} \qquad f(x) = \inf_{w\in\mathbb{R}}\Big\{x\,w + (-f)^*(-w)\Big\}.$$

Soit $x\in\mathbb{R}$. Notons w^* un point où est atteint l'optimum, c'est-à-dire un point vérifiant

$$x\,w^* + (-f)^*(-w^*) = \inf_{w\in\mathbb{R}}\Big\{x\,w + (-f)^*(-w)\Big\}.$$

On a dans ce cas $\qquad f(x) = x\,w^* + (-f)^*(-w^*) \qquad\Longleftrightarrow\qquad (-f)^*(-w^*) = -x\,w^* + f(x).$

Or, $-f$ est convexe, s.c.i. et propre, donc

$$(-f)^*(-w^*) = \sup_{y\in\mathbb{R}}\Big\{-w^*\,y + f(y)\Big\} = -w^*\,x + f(x)$$

donc la borne supérieure est atteinte en x ; par ailleurs, l'équation d'Euler assure que

$$-w^* + f'(x) = 0 \qquad \text{soit} \qquad w^* = f'(x).$$

En d'autres termes, on vient de montrer que pour tout $x\in\mathbb{R}$,

$$f(x) = \inf_{w\in\mathbb{R}}\Big\{x\,w + (-f)^*(-w)\Big\} = x\,w^* + (-f)^*(-w^*) \qquad \text{avec} \qquad w^* = f'(x) > 0.$$

On en déduit que, pour tout $p=(p^x,p^y)$, on a

$$\begin{aligned} h(p^x,p^y) &= \sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1}\inf_{w_{i,j}\in\mathbb{R}}\Big\{({p^x_{i,j}}^2 + {p^y_{i,j}}^2)\,w_{i,j} - (-f)^*(-w_{i,j})\Big\}\\ &= \inf_{\substack{w=(w_{i,j})_{i\in[\![0\,;N_x-1]\!]\\ j\in[\![0\,;N_y-1]\!]}}\left\{\big\langle (p^x)^2+(p^y)^2, w\big\rangle - \sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1}(-f)^*(-w_{i,j})\right\}\end{aligned}$$

Le problème (7) peut donc s'écrire sous la forme d'un nouveau problème :

$$\min_{\substack{v\in[\,0\,;1\,]^{N_xN_y}\\ w\in\mathbb{R}^{N_xN_y}}}\left\{\lambda\,\big\langle(\partial_x v)^2 + (\partial_y v)^2, w\big\rangle - \lambda\sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1}(-f)^*(-w_{i,j}) + \frac{\mu}{2}\,\|K\star v - g\|^2\right\} \tag{8}$$

avec $\nabla v = (\partial_x v, \partial_y v)$, que l'on peut interpréter comme une énergie comportant un terme d'attache aux données pondéré par μ, avec un terme de régularisation à pondération variable en w, où les poids $w_{i,j}$ sont eux-mêmes optimisés suivant un critère $(-f)^*$.

Résolution du problème (8) On cherche à présent à résoudre le problème (8). On remarque d'abord que si v est fixé, alors le problème étudié devient

$$\min_{w\in\mathbb{R}^{N_x N_y}} \left\{ \lambda \left\langle (\partial_x v)^2 + (\partial_y v)^2, w \right\rangle - \lambda \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} (-f)^*(-w_{i,j}) \right\}$$

et que le minimum est atteint en w vérifiant

$$\forall\, (i,j) \in [\![0\,;\, N_x-1]\!] \times [\![0\,;\, N_y-1]\!], \qquad w_{i,j} = f'((\partial^x v)_{i,j}{}^2 + (\partial^y v)_{i,j}{}^2).$$

Par ailleurs, si w est fixé, le problème devient

$$\min_{v\in[\,0\,;1\,]^{N_x N_y}} \left\{ \lambda \left\langle (\partial_x v)^2 + (\partial_y v)^2, w \right\rangle + \frac{\mu}{2} \left\| K \star v - g \right\|^2 \right\}$$

que l'on peut résoudre avec un algorithme de type FISTA. Pour cela, il faut calculer le gradient de la fonction

$$F : v \mapsto \underbrace{\lambda \left\langle (\partial_x v)^2 + (\partial_y v)^2, w \right\rangle}_{=f(v)} + \underbrace{\frac{\mu}{2} \left\| K \star v - g \right\|^2}_{=g(v)}.$$

Pour calculer le gradient du premier terme f, on écrit

$$f(v+h) = \lambda \left\langle (\partial_x v + \partial_x h)^2 + (\partial_y v + \partial_y h)^2, w \right\rangle = f(v) + 2\,\lambda \left\langle \partial_x v\, \partial_x h + \partial_y v\, \partial_y h, w \right\rangle + o(\|h\|).$$

On remarque alors que

$$\left\langle \partial_x v\, \partial_x h + \partial_y v\, \partial_y h, w \right\rangle = \left\langle \partial_x v\, \partial_x h, w \right\rangle + \left\langle \partial_y v\, \partial_y h, w \right\rangle = \left\langle \partial_x h, \partial_x v\, w \right\rangle + \left\langle \partial_y h, \partial_y v\, w \right\rangle = \left\langle h, (\partial_x)^*(\partial_x v\, w) \right\rangle + \left\langle h, (\partial_y)^*(\partial_y v\, w) \right\rangle$$

ce qui implique que $$\nabla f(v) = 2\,\lambda \left((\partial_x)^*(\partial_x v\, w) + (\partial_y)^*(\partial_y v\, w) \right)$$

où on a noté respectivement $(\partial_x)^*$ et $(\partial_y)^*$ les opérateurs adjoints de ∂_x et ∂_y. Quant au terme g, le gradient est donné par

$$\nabla g(v) = \mu\, \tilde{K} \star (K \star v - g) \qquad \text{avec } \hat{\tilde{K}} = \bar{\hat{K}}.$$

Finalement, $$\nabla F(v) = 2\,\lambda \left((\partial_x)^*(\partial_x v\, w) + (\partial_y)^*(\partial_y v\, w) \right) + \mu\, \tilde{K} \star (K \star v - g)$$

qui est lipschitzien, de constante de Lipschitz β. L'algorithme FISTA devient alors

$$\bar{v}_0 = v_0 \qquad \text{et} \qquad \forall\, k \in \mathbb{N}, \quad \left\{ \begin{array}{rcl} v_{k+1} & := & \mathrm{proj}_{[\,0\,;1\,]^{N_x N_y}} \left(\bar{v}_k - \tau\, \nabla F(\bar{v}_k) \right) \\ \bar{v}_{k+1} & := & \lambda_k v_{k+1} + (1-\lambda_k)\, v_k \end{array} \right.$$

avec $\tau = 1/\beta$. L'idée est donc d'alterner minimisation exacte en w et M pas de descente de gradient en v, à savoir une initialisation $(w^0)_{i,j} = 1$, $\bar{v}_0 = v_0$ et pour tout $k \in \mathbb{N}$,

$$\left\{ \begin{array}{l} \forall\, m \in [\![0\,;\, M-1]\!] \quad \left\{ \begin{array}{rcl} v_{k+(m+1)/M} & := & \mathrm{proj}_{[\,0\,;1\,]^{N_x N_y}} \Big(\bar{v}_{k+m/M} \\ & & \qquad -2\,\tau\,\lambda \left((\partial_x)^*(\partial_x v\, w) + (\partial_y)^*(\partial_y v\, w) \right) - \tau\, \mu\, \tilde{K} \star (K \star v - g) \Big) \\ \bar{v}_{k+(m+1)/M} & := & \lambda_k v_{k+(m+1)/M} + (1-\lambda_k)\, v_{k+m/M} \end{array} \right. \\ \qquad (w_{k+1})_{i,j} \;:=\; f'((\partial_x v_{k+1})_{i,j}{}^2 + (\partial_y v_{k+1})_{i,j}{}^2). \end{array} \right.$$

Code Matlab Voici le code Matlab proposé pour la méthode présentée dans ce paragraphe.

```
function v = estimation_u_nonconvexe(g,rho,lambda,mu,delta,u_init)

addpath('functions') ;
[Nx,Ny] = size(g) ;
M = size(rho,1)/2 ;

% FFT
% noyau
rhoext = zeros(Nx,Ny) ;
     rhoext(Nx/2+1-M:Nx/2+M,Ny/2+1-M:Ny/2+M) = rho ;
     rhoext = fftshift(rhoext) ;
fft_rho = fft2(rhoext) ;

% signal
fft_g = fft2(g) ;

% Algorithme
```

```
% paramètres
mu = mu/20 ;
lambda = lambda/20 ;
tau = 1/10/(4*lambda*4+mu) ; % pas de temps

% régularisation
f = @(x) delta*x./(1+delta*x) ; fprime = @(x) delta./(1+delta*x).^2 ;
% f = @(x) log(1+delta*x); fprime = @(x) delta./(1+delta*x);
% f = @(x) sqrt(x + delta); fprime = @(x) 0.5./sqrt(x + delta);
niter = 2000 ; % nombre d'itérations total 2000 par défaut
M = 20 ; % nombre de pas de descente dans l'estimation de v

% initialisation
if nargin<6
    u_init = zeros(Nx,Ny) ;
end
v = u_init ; v_old = v ; v_bar = v ;
w = ones(Nx,Ny) ;
tkold = 1 ;

for n=1:niter

    % descente de gradient pour l'estimation de v
    for m=1:M
        % calcul du gradient
        fft_nablag = mu*conj(fft_rho).*(fft_rho.*fft2(v_bar)-fft_g) ;
        nablag = real(ifft2(fft_nablag)) ;

        [dxv,dyv] = nablah(v_bar) ;
        tmp = w.*dxv ; [tmpx,unused] = nablahstar(tmp) ;
        tmp = w.*dyv ; [unused,tmpy] = nablahstar(tmp) ;
        nablaf = 2*lambda*(tmpx+tmpy) ;

        nablaF = nablaf + nablag ;

        % gradient projeté
        v = min(max(v_bar - tau*nablaF,0),1) ;

        % relaxation (FISTA)
        lambda_k = 1+2*(tkold-1)/(1+sqrt(4*tkold^2+1)) ;
        tkold = (1+sqrt(4*tkold^2+1))/2 ;
        v_bar = (1-lambda_k)*v_old + lambda_k*v;
        v_old = v;
    end

    % MàJ de w
    [dxv,dyv] = nablah(v) ;
    w = fprime(dxv.^2+dyv.^2) ;

end

end
```

Résultats On montre dans les figures **??**–**??** différents résultats obtenus avec différentes fonctions f. Le temps d'exécution moyen est ici de 27 secondes.

2.4 Utilisation d'une base orthonormée

Un autre type de régularisation peut être proposé en utilisant une base orthonormée. On suppose qu'il existe une base orthonormée $(e_{i,j})_{\substack{i\in[\![0\,;\,N_x-1]\!]\\ j\in[\![0\,;\,N_x-1]\!]}}$, vérifiant

$$\forall\, i \in [\![0\,;\,N_x-1]\!], \quad \forall\, j \in [\![0\,;\,N_x-1]\!], \qquad \|e_i\| = 1 \qquad \text{et} \qquad \langle e_i, e_j\rangle = 0 \text{ si } i \neq j$$

et telle que, pour tout $u \in \mathbb{R}^{N_x N_y}$, on puisse écrire

$$u = \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} \langle u, e_{i,j}\rangle\, e_{i,j}$$

Lorsque $\langle u, e_{i,j}\rangle \in \mathbb{C}$, on dira que la base est complexe, et lorsque $\langle u, e_{i,j}\rangle \in \mathbb{R}$, on dira que la base est réelle.

Exemples : De telles bases existent, et sont fréquemment utilisées en traitement de l'image. On peut citer la base intervenant dans la transformée de Fourier discrète, qui est complexe, celle de la transformée en cosinus discrète, ou encore les bases d'ondelettes.

Si on choisit une base dans laquelle on peut avoir une représentation parcimonieuse de l'image u, c'est-à-dire dans laquelle u s'écrit avec un (petit) nombre de coefficients non nuls, il peut commencer à être intéressant de considérer le terme de régularisation suivant :

$$f(v) = \lambda \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} |\langle u, e_{i,j}\rangle|$$

à la place de la régularisation TV, ce qui conduit à considérer l'énergie suivante

$$E(v) = \lambda \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} |\langle u, e_{i,j}\rangle| + \mu \|K \star v - g\|^2$$

Théorème de Parseval L'intérêt des bases orthonormées réside dans le théorème de Parseval : on a pour tout $u \in \mathbb{R}^{N_x N_y}$

$$\|u\|^2 = \left\| \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} \langle u, e_{i,j}\rangle\, e_{i,j} \right\|^2 = \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} \|\langle u, e_{i,j}\rangle\|^2$$

grâce à l'orthogonalité de la base e.

Minimisation de l'énergie Pour minimiser l'énergie E, on adopte une approche explicite-implicite, ce qui conduit à calculer l'opérateur proximal de f. Par définition, on a

$$p = \operatorname{prox}_{\tau f}(x) \qquad \Longleftrightarrow \qquad p = \operatorname*{argmin}_{x\in\mathbb{R}^{N_x N_y}} \left\{ \tau\, f(x) + \frac{1}{2}\|x-p\|^2 \right\} = \operatorname*{argmin}_{x\in\mathbb{R}^{N_x N_y}} \left\{ \tau\,\lambda \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} |\langle x, e_{i,j}\rangle| + \frac{1}{2}\|x-p\|^2 \right\}$$

Le théorème de Parseval assure alors que

$$p^* = \operatorname{prox}_{\tau f}(x) \qquad \Longleftrightarrow \qquad p^* = \operatorname*{argmin}_{p\in\mathbb{R}^{N_x N_y}} \left\{ \tau\,\lambda \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} |\langle x, e_{i,j}\rangle| + \frac{1}{2} \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} |\langle x-p, e_{i,j}\rangle|^2 \right\}$$

soit

$$p^* = \operatorname{prox}_{\tau f}(x) \qquad \Longleftrightarrow \qquad p^* = \operatorname*{argmin}_{p\in\mathbb{R}^{N_x N_y}} \left\{ \tau\,\lambda \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} |\langle x, e_{i,j}\rangle| + \frac{1}{2} |\langle x, e_{i,j}\rangle - \langle p, e_{i,j}\rangle|^2 \right\}$$

avec $p^* = \displaystyle\sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} \langle p^*, e_{i,j}\rangle\, e_{i,j}$. Il s'agit donc de calculer chacun des $\langle p^*, e_{i,j}\rangle$. Ceux-ci vérifient en réalité

$$\forall\, i \in [\![0\,;\,N_x-1]\!], \quad \forall\, j \in [\![0\,;\,N_y-1]\!], \qquad \langle p^*, e_{i,j}\rangle = \operatorname*{argmin}_{p_{i,j}\in\mathbb{R}} \left\{ \tau\,\lambda\,|p_{i,j}| + \frac{1}{2}\,|\langle x, e_{i,j}\rangle - p_{i,j}|^2 \right\}$$

On en déduit (soit en utilisant une variable duale, soit en utilisant le sous-gradient) que si la base e est réelle, alors on a

$$\forall\, i \in [\![0\,;\,N_x-1]\!], \quad \forall\, j \in [\![0\,;\,N_y-1]\!], \qquad \langle p^*, e_{i,j}\rangle = \operatorname{sgn}(\langle x, e_{i,j}\rangle) \left(|\langle x, e_{i,j}\rangle| - \lambda \right)^+$$

ce qui revient à réaliser un seuillage doux sur les coefficients de x dans la base e.

Algorithme proposé On propose donc l'algorithme suivant, de type FISTA, avec $\tau = 1/2/\mu$.

$$\bar{v}_0 = v_0 \qquad \text{et} \qquad \forall\, k \in \mathbb{N}, \qquad \left\{ \begin{array}{rcl} v_{k+1} & := & \mathrm{SoftThresh}_\lambda\Big(\bar{v}_k - 2\,\tau\,\mu\,\tilde{K} \star (K \star \bar{v}_k - g)\Big) \\ \bar{v}_{k+1} & := & \lambda_k v_{k+1} + (1-\lambda_k)\, v_k \end{array} \right.$$