

Modèles Stochastiques pour l'Analyse des Images

Aicha Boujandar

Juin 2020

1 Partie théorique

Dans cette partie, on cherche à faire un développement et des preuves autour de la phrase (à propos du prox_g^λ) : "The smoothness parameter $\lambda > 0$ controls the regularity properties of the proximal operator." (haut de la page 4)

L'opérateur proximal sera appliqué dans l'algorithmr SAPG à la focnntion $\theta^T g : \mathbb{R}^d \rightarrow]-\infty, +\infty]$. On se place ainsi dans le cadre des fonctions $g : \mathbb{R}^d \rightarrow]-\infty, +\infty]$.

On définit l'opérateur proximal d'une fonction $g : \mathbb{R}^d \rightarrow]-\infty, +\infty]$ par :

$$\text{prox}_g^\lambda(x) = \underset{\tilde{x} \in \mathbb{R}^d}{\operatorname{argmin}} g(\tilde{x}) + \frac{1}{2\lambda} \|\tilde{x} - x\|_2^2$$

Et on définit la régularisée de Moreau-Yosida de la fonction g la fonction $M_g^\lambda : \mathbb{R}^d \rightarrow \mathbb{R}$:

$$M_g^\lambda(x) = \inf_{\tilde{x} \in \mathbb{R}^d} g(\tilde{x}) + \frac{1}{2\lambda} \|\tilde{x} - x\|_2^2$$

L'explication de l'expression "The smoothness parameter $\lambda > 0$ controls the regularity properties of the proximal operator" est donnée par le théorème de la régularité de l'enveloppe de Moreau (smoothness of the Moreau envelope, theorem 6.60, [2]) :

Théorème 1 (smoothness of the Moreau envelope). Soit $g : \mathbb{R}^d \rightarrow]-\infty, +\infty]$ une fonction propre, sci (semi continue inférieurement), et convexe. Ainsi, M_g^λ est $\frac{1}{\lambda}$ -régulière (smooth) sur \mathbb{R}^d et pour tout $x \in \mathbb{R}^d$, on a :

$$\nabla M_g^\lambda(x) = \frac{1}{\lambda}(x - \text{prox}_g^\lambda(x))$$

Démonstration

Pour montrer que M_g^λ est $\frac{1}{\lambda}$ -régulière (smooth) sur \mathbb{R}^d , il faut montrer qu'elle différentiable et $\frac{1}{\lambda}$ -lipschitzienne. On démontre le résultat suivant : Pour toute fonction f_1, f_2 propres, sci, et convexes, on définit l'opérateur d'inf-convolution comme suit :

$$f_1 \square f_2(x) = \inf_{\tilde{x} \in \mathbb{R}^d} f_1(\tilde{x}) + f_2(x - \tilde{x})$$

On a $(f_1 + f_2)^* = f_1^* \square f_2^*$ avec f_i^* la transformée de Fenchel de f_i , $i = 1, 2$.

Démonstration du résultat

$$\begin{aligned} (f_1 \square f_2)^*(p) &= \sup_{x \in \mathbb{R}^d} p^T x - \inf_{\tilde{x} \in \mathbb{R}^d} f_1(\tilde{x}) + f_2(x - \tilde{x}) \\ &= \sup_{x, \tilde{x} \in \mathbb{R}^d} p^T x - f_1(\tilde{x}) - f_2(x - \tilde{x}) \\ &= \sup_{x, \tilde{x} \in \mathbb{R}^d} p^T(x - \tilde{x}) - f_2(x - \tilde{x}) + p^T \tilde{x} - f_1(\tilde{x}) \\ &= \sup_{y, \tilde{x} \in \mathbb{R}^d} p^T y - f_2 y + p^T \tilde{x} - f_1(\tilde{x}) \\ &= f_1^*(p) + f_2^*(p) \end{aligned}$$

En prenant : $f'_i = f_i^*$, $i = 1, 2$, on obtient : $(f_1^* \square f_2^*)^* = f_1 + f_2$ (car f_1, f_2 sont propres, sci, et convexes donc $f_i^{**} = f_i$, $i = 1, 2$). Et en prenant le conjugué de cette expression, on a :

$$(f_1^* \square f_2^*)^{**} = (f_1 + f_2)^*$$

Et on a : $f_1^* \square f_2^*$ est convexe, propre et sci car f_1^* et f_2^* sont convexes, propres et sci, donc :

$$f_1^* \square f_2^* = (f_1 + f_2)^*$$

cqfd

On a :

$$M_g^\lambda(x) = g \square \left(\frac{1}{2\lambda} |||_2^2 \right)$$

Or, g est convexe, sci et propre, et de même pour $\frac{1}{2\lambda} |||_2^2$, donc :

$$\begin{aligned} M_g^\lambda(x) &= (g^*)^* \square \left(\left(\frac{1}{2\lambda} |||_2^2 \right)^* \right)^*(x) \\ &= (g^* + \left(\frac{1}{2\lambda} |||_2^2 \right)^*)^*(x) \end{aligned}$$

$$\left[\left(\frac{1}{2\lambda} |||_2^2 \right)^*(x) = \frac{\lambda}{2} \|x\|_2^2 \right]$$

Ainsi M_g^λ est convexe et sci en tant que transformée de Fenchel de $g^* + \left(\frac{1}{2\lambda} |||_2^2 \right)^*$. On définit le sous-différentiel de M_g^λ en un point $x \in \mathbb{R}^d$:

$$\begin{aligned} \tilde{x} \in \partial M_g^\lambda(x) &\iff x \in \partial M_g^{\lambda*}(\tilde{x}) \\ &\iff x \in \partial(g^* + \left(\frac{1}{2\lambda} |||_2^2 \right)^*)(\tilde{x}) \end{aligned}$$

Et on a : g^* et $\left(\frac{1}{2\lambda} |||_2^2 \right)^*$ sont convexes et sci donc :

$$\begin{aligned} \tilde{x} \in \partial M_g^\lambda(x) &\iff x \in \partial(g^*)(\tilde{x}) + \partial\left(\left(\frac{1}{2\lambda} |||_2^2 \right)^*\right)(\tilde{x}) \\ &\iff \exists x_1 \in \partial(g^*)(\tilde{x}), x_2 \in \partial\left(\left(\frac{1}{2\lambda} |||_2^2 \right)^*\right)(\tilde{x}) \text{ tq } x = x_1 + x_2 \\ &\iff \exists x_1 \in \partial(g^*)(\tilde{x}) \text{ tq } x = x_1 + \lambda\tilde{x} \text{ (passage1 détaillé après)} \\ &\iff x - \lambda\tilde{x} \in \partial(g^*)(\tilde{x}) \\ &\iff x - \lambda\tilde{x} \in \partial(\lambda g)^*(\lambda\tilde{x}) \text{ (passage2 détaillé après)} \\ &\iff \lambda\tilde{x} = \text{prox}_{(\lambda g)^*}^1(x) \\ &\iff \tilde{x} = \frac{x - \text{prox}_{\lambda g}^1(x)}{\lambda} \\ &\iff \tilde{x} = \frac{x - \text{prox}_g^\lambda(x)}{\lambda} \end{aligned}$$

Détails du passage1 : On a :

$$\begin{aligned} x_2 \in \partial\left(\left(\frac{1}{2\lambda} |||_2^2 \right)^*\right)(\tilde{x}) &\iff x_2 \in \partial\left(\frac{\lambda}{2} |||_2^2\right)(\tilde{x}) \\ &\iff x_2 = \nabla\left(\frac{\lambda}{2} |||_2^2\right)(\tilde{x}) \\ &\iff x_2 = \lambda\tilde{x} \end{aligned}$$

Détails du passage2 : On a :

$$\partial(\lambda g)^*(\lambda x) = \{u \in \mathbb{R}^d, \lambda g(u) + (\lambda g)^*(\lambda x) = \langle u, \lambda x \rangle\}$$

Et on a :

$$\begin{aligned} (\lambda g)^*(\lambda x) &= \sup_{r \in \mathbb{R}^d} \langle r, \lambda x \rangle - \lambda g(r) \\ &= \lambda \left[\sup_{r \in \mathbb{R}^d} \langle r, x \rangle - g(r) \right] \\ &= \lambda g^*(x) \end{aligned}$$

Ainsi :

$$\partial(\lambda g)^*(\lambda x) = \{u \in \mathbb{R}^d, \lambda g(u) + \lambda g^*(x) = \langle u, \lambda x \rangle\}$$

Donc :

$$\partial(\lambda g)^*(\lambda x) = \{u \in \mathbb{R}^d, g(u) + g^*(x) = \langle u, x \rangle\} = \partial(g^*)(x)$$

Fin détails des passages.

Avec les hypothèses considérées sur la fonction g , on a :

$$\tilde{x} \rightarrow g(\tilde{x}) + \frac{1}{2} \|x' - x\|_2^2$$

est une fonction sci, propre et fortement convexe donc $\text{prox}_g^\lambda(x)$ admet une unique solution. Ainsi, $\partial M_g^\lambda(x) \forall x$ est un singleton, M_g^λ est différentiable et on a :

$$\nabla M_g^\lambda(x) = \frac{x - \text{prox}_g^\lambda(x)}{\lambda}$$

. Soit $x, x' \in \mathbb{R}^d$, g est une fonction convexe, sci et propre, donc : $x - \text{prox}_g^\lambda(x) \in \partial(\lambda g)(\text{prox}_g^\lambda(x))$ et $x' - \text{prox}_g^\lambda(x') \in \partial(\lambda g)(\text{prox}_g^\lambda(x'))$.

Par la monotonie du sous-différentiel, on a :

$$\begin{aligned} &\langle \frac{1}{\lambda}(x' - \text{prox}_g^\lambda(x')) - \frac{1}{\lambda}(x - \text{prox}_g^\lambda(x)), \text{prox}_g^\lambda(x') - \text{prox}_g^\lambda(x) \rangle \geq 0 \\ \iff &\lambda \langle \frac{1}{\lambda}(x' - \text{prox}_g^\lambda(x')) - \frac{1}{\lambda}(x - \text{prox}_g^\lambda(x)), \frac{1}{\lambda}(x - \text{prox}_g^\lambda(x)) - \frac{1}{\lambda}(x' - \text{prox}_g^\lambda(x')) \rangle \\ &+ \langle \frac{1}{\lambda}(x' - \text{prox}_g^\lambda(x')) - \frac{1}{\lambda}(x - \text{prox}_g^\lambda(x)), x' - x \rangle \geq 0 \\ \iff &-\lambda \|\nabla M_g^\lambda(x') - \nabla M_g^\lambda(x)\|_2^2 + \langle \nabla M_g^\lambda(x') - \nabla M_g^\lambda(x), x' - x \rangle \geq 0 \end{aligned}$$

Or, par Cauchy-Schwartz :

$$\langle \frac{1}{\lambda}(x' - \text{prox}_g^\lambda(x')) - \frac{1}{\lambda}(x - \text{prox}_g^\lambda(x)), x' - x \rangle \leq \|x' - x\|_2 \times \|\nabla M_g^\lambda(x') - \nabla M_g^\lambda(x)\|_2$$

On conclut que :

$$\|\nabla M_g^\lambda(x') - \nabla M_g^\lambda(x)\|_2 \leq \frac{1}{\lambda} \|x' - x\|_2$$

Ainsi M_g^λ est $\frac{1}{\lambda}$ -lipchitzienne.

M_g^λ est différentiable et $\frac{1}{\lambda}$ -lipchitzienne donc elle est $\frac{1}{\lambda}$ -régulière. Ce qui explique la phrase : *the smoothness parameter λ controls the regularity properties of the proximal operator*.

2 Partie expérimentale : Denoising with a total generalized variation prior

Le modèle est présenté dans le jupyter notebook. Dans cette partie du rapport, on présente les détails des calculs qui n'ont pas été inclus dans le jupyter notebook par souci de clarté. On présentera également les résultats obtenus les plus importants.

Organisation du code

Dans [1], l'expérience a été réalisée à l'aide de Matlab R2018a, et les calculs ont été effectués avec un processseur Intel i9-8950HK@2.90GHz.

Dans notre cas, on a réalisé deux versions de code :

- la première a été implémentée dans le jupyter notebook accompagnant ce rapport. Le jupyter notebook a été exécuté sur la plateforme Google Colab avec les caractéristiques suivantes : Python3,instance n1-highmem-2, 2vCPU @ 2.2GHz, 13Go RAM,100Go d'espace libre, maximum 12 heures d'exécution.
- La deuxième implémentation est sur Matlab R2019b (version académique), et les calculs ont été effectués sur un processeur Intel(R) Core(TM) i5-7200U CPU@ 2.50 GHz 2.71 GHz. Cette version a été utilisée pour reproduire une seule expérience, et les résultats sont présentés en annexe.

Le solveur utilisé pour les problèmes de minimisation TGVdenoising et TGVprox est celui de [3].

Détails du calcul des opérateurs proximaux utilisés dans la fonction TGVdenoising :

On a :

$$F_x(x) = \frac{1}{2\sigma^2} \|y - x\|_2^2 + \epsilon\|x\|_2^2$$

ainsi :

$$\text{prox}_{\tau F_x}(x) = \underset{x' \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2} \|x - x'\|_2^2 + \tau F_x(x')$$

$x' \rightarrow \frac{1}{2}\|x - x'\|_2^2 + \tau F_x(x')$ convexe et différentiable donc :

$$\begin{aligned} \text{prox}_{\tau F_x}(x) = x' &\iff \nabla_{x'} \left(\frac{1}{2} \|x - x'\|_2^2 + \tau F_x(x') \right) = 0 \\ &\iff x' - x + \frac{\tau}{\sigma^2} (x' - y) + 2\epsilon x' = 0 \\ &\iff x' = \frac{\sigma^2 x + \tau y}{\sigma^2 + \tau + 2\sigma^2\epsilon} \end{aligned}$$

On a $F_r(r) = \theta^1 \|r\|_{1,2}$, donc :

$$\begin{aligned} \text{prox}_{\tau F_r}(r) &= \text{prox}_{\tau \theta^1 \| \cdot \|_{1,2}}(r) \\ &= r - \tau \theta^1 \text{proj}_B(\| \cdot \|_{1,2}^*) \left(\frac{r}{\tau \theta^1} \right) \\ &= r - \tau \theta^1 \frac{\frac{r}{\tau \theta^1}}{\max(\| \frac{r}{\tau \theta^1} \|_{1,2}^*, 1)} \\ &= r - \frac{r}{\max(\| \frac{r}{\tau \theta^1} \|_{1,2}^*, 1)} \end{aligned}$$

Et on a : $G(u) = \theta^2 |||_{1,Frob}$, Et on a :

$$\begin{aligned}
prox_{\tau'(\theta^2|||_{1,Frob})^*}(u) &= u - \tau' prox_{\frac{\theta^2}{\tau'}|||_{1,Frob}}\left(\frac{u}{\tau'}\right) \\
&= u - \tau'\left[\frac{u}{\tau'} - \frac{\theta^2}{\tau'} proj_B(|||_{1,Frob}^*)\left(\frac{u}{\tau'} \frac{\tau'}{\theta^2}\right)\right] \\
&= \theta^2 \frac{\frac{u}{\theta^2}}{\max\left(\left\|\frac{u}{\theta^2}\right\|_{1,Frob}^*, 1\right)} \\
&= \frac{u}{\max\left(\left\|\frac{u}{\theta^2}\right\|_{1,Frob}^*, 1\right)}
\end{aligned}$$

Débruitage avec différentes valeurs de θ^1 et θ^2

Afin de souligner l'impact du choix des paramètres θ^1 et θ^2 sur le résultat du débruitage avec une loi à priori variation totale généralisée (total generalized variation prior) (expérience figure 14 dans [1]), on teste l'algorithme de débruitage (fonction TGVdenoising) avec les valeurs : $\theta^1 \in \{1, 10, 100\}$ et $\theta^2 \in \{1.5, 15, 150\}$. Pour l'image, on choisit l'image parrotgray figure 1 (voir également dossier im) et pour le bruit (image figure 2), on choisit un bruit gaussien avec σ vérifiant SNR=5.6 db (rapport signal sur bruit). Les résultats sont présentés dans la figure 3 et



FIGURE 1: Image réelle FIGURE 2: Image
bruisée SNR = 5.6 db



commentés dans le jupyter notebook.

Quantitativement, on mesure l'impact du choix de θ^1 et de θ^2 en calculant le psnr entre l'image réelle et l'image débruitée pour plusieurs valeurs de paramètres comme montré dans la figure 4 et commenté dans le jupyter notebook. Pour le calcul du psnr, on utilise la fonction peak_to_signal_ratio du module python skimage.metrics.

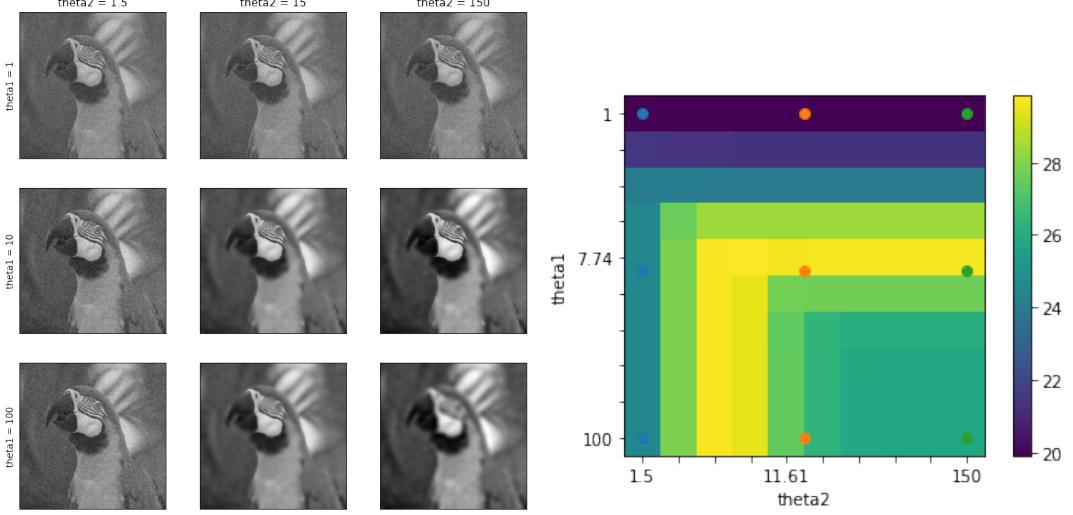


FIGURE 3: Résultats pour différentes valeurs de θ^1 et θ^2

FIGURE 4: PSNR pour différentes valeurs de θ^1 et θ^2

Débruitage des images boat et lake avec SNR = 8db

Comme décrit dans le jupyter notebook, on applique les fonctions SAPG et TGVdenoising aux deux images lake et boat bruitées (SNR = 8db), et on choisit comme critère d'arrêt le critère 1 avec Nbiter = 300 (la convergence pour ces images en appliquant les critères 2 et 3 est assurée en moins de 30 itérations). Les résultats de débruitage sont présentés dans la figure 5. Pour les paramètres, on choisit les mêmes paramètres que ceux utilisés dans l'article [1]. Pour les paramètres du solveur (qu'ils n'ont pas précisé), on a essayé de choisir les paramètres qui réalisent le meilleur rapport temps de calcul et résultat, ainsi, si on veut améliorer les résultats, on peut par exemple ajuster les paramètres τ et Nbiter dans le calcul du prox, et le Nbiter dans la fonction TGVdenoising. Le choix du critère nombre d'itérations = 300 au lieu de 2000 est dû à la contrainte temps de calcul. En effet, même en appliquant la fonction SAPG à un patch représentatif de 255×255 au lieu de l'appliquer à toute l'image, le temps de calcul nécessaire pour effectuer tous les calculs reste très conséquent dans la version python. Par exemple, pour un patch de l'image lake de 255×255 , la version python prend 4h30min pour 300 itérations, alors que la version Matlab prend 50 min pour 300 itérations pour le même patch de l'image lake. Cette différence dans le temps de calcul est due principalement à la performance de Matlab en terme de calcul matriciel par rapport au module numpy de python. Dans l'article [1], et vu que le processeur utilisé est beaucoup plus performant que ceux utilisés dans notre cas (voir la description des processeurs dans le paragraphe Organisation du code), les 2000 itérations ne prennent que 38 minutes pour un patch de la même taille.

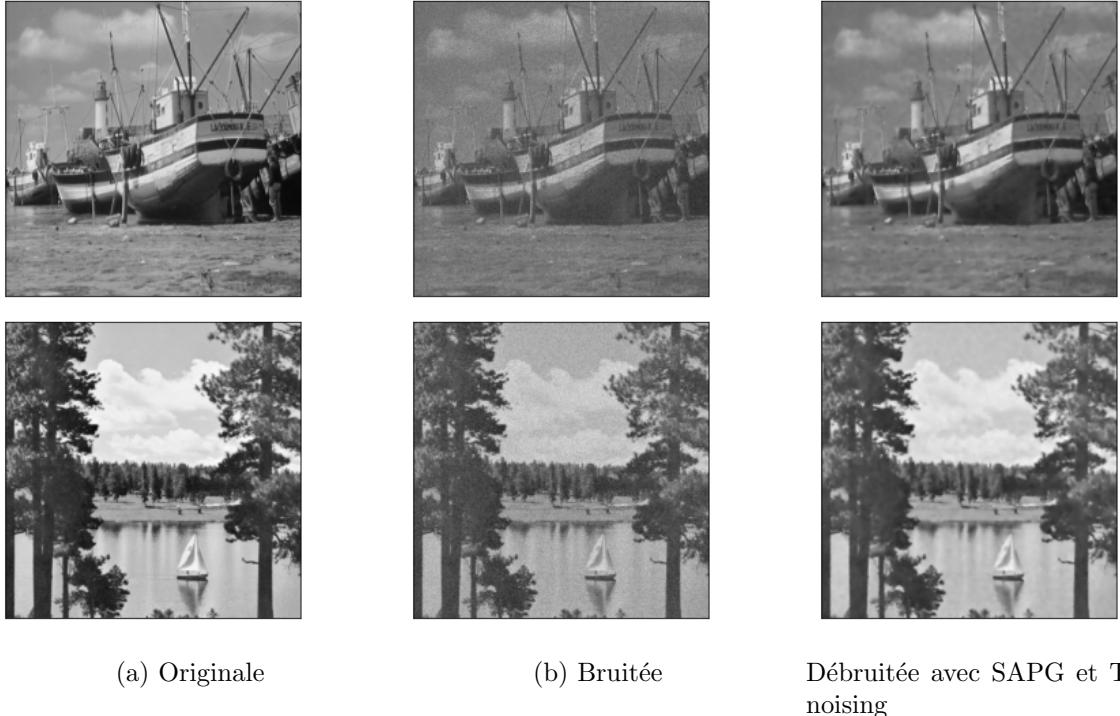


FIGURE 5: Débruitage des images boat et lake pour SNR=8 db avec SAPG et l'estimateur MAP

Le psnr entre l'image boat réelle et l'image débruitée obtenue est égal à 28.97, le max des psnr calculés dans la figure 6 est 29.27 ce qui ne représente pas une grande différence, surtout en tenant compte des valeurs obtenues pour d'autres valeurs de paramètres (minimum : 22.14, moyenne : 24.15 , médiane : 23.11). (Ce résultat a été décrit dans l'article [1] par "Observe that the estimated solutions are extremely close to the optimal ones, which is remarkable given the difficulty of the problem and the fact that solutions are derived directly from statistical inference principles, without any form of ground truth"). Le point rouge représente l'emplacement des paramètres θ^1 et θ^2 obtenus. Ce point est représenté avec sa valeur exacte du psnr de ce dans la heatmap 3D de la figure 6. Le psnr entre l'image lake réelle et l'image débruitée obtenue est égal à 28.80, le max des psnr calculés dans la figure 7 est 29.15 ce qui ne représente pas une grande différence, surtout en tenant compte des valeurs obtenues pour d'autres valeurs de paramètres (minimum : 22.41, moyenne : 24.31 , médiane : 23.32). Le point rouge représente l'emplacement des paramètres θ^1 et θ^2 obtenus pour cette image. Ce point est représenté avec sa valeur exacte du psnr de ce dans la heatmap 3D de la figure 7.

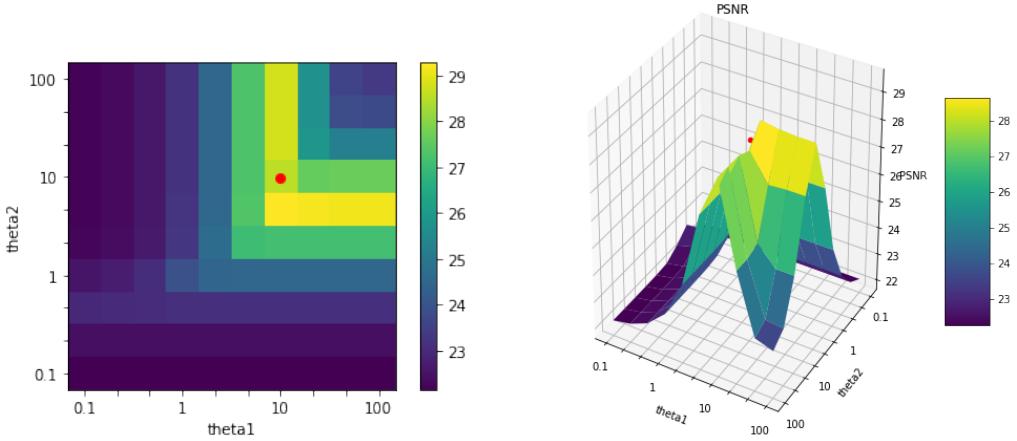


FIGURE 6: PSNR de l'image boat, SNR 8db pour différentes valeurs de θ^1 et θ^2

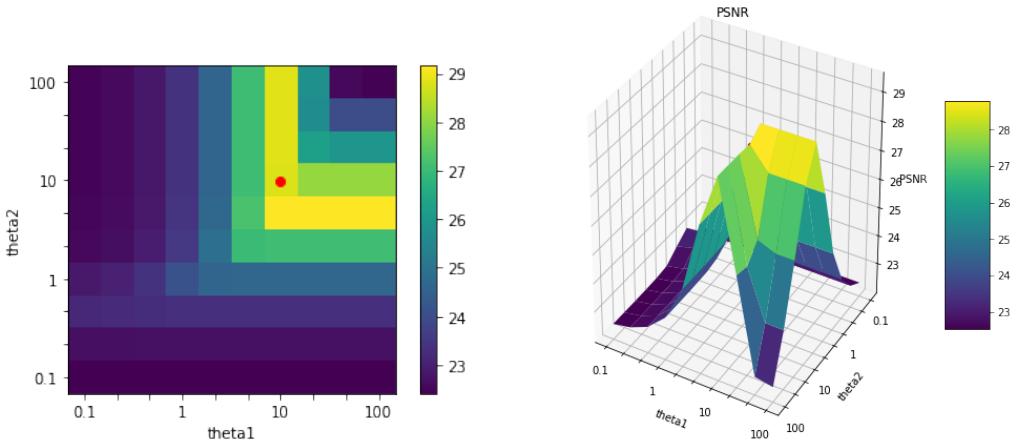


FIGURE 7: PSNR de l'image lake, SNR 8db pour différentes valeurs de θ^1 et θ^2

Débruitage de l'image lake pour différentes valeurs du SNR

Dans cette section, on applique l'algorithme SAPG sur l'image test lake pour les valeurs de SNR : 8db, 12 db et 20 db. Le but est de tracer l'évolution des paramètres θ^1 et θ^2 tout au long des itérations. On choisit pour cela N=300. Cette expérience est reproduite avec la version Matlab du code et les résultats sont présentés dans l'annexe.

Résultats pour SNR = 8db

Les résultats sont présentés dans les figures 8, 9, 10 et 11. Les commentaires sur ces figures sont présentés dans le jupyter notebook.



FIGURE 8: Image réelle

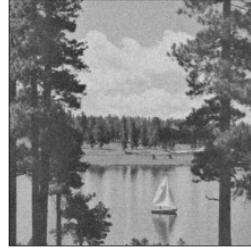


FIGURE 9: Image bruitée SNR = 8 db



FIGURE 10: Image débruitée avec SAPG et TGVdenoising

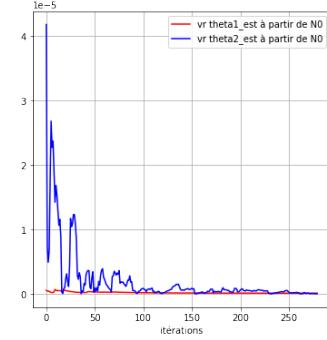
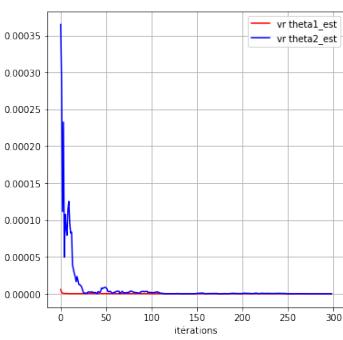
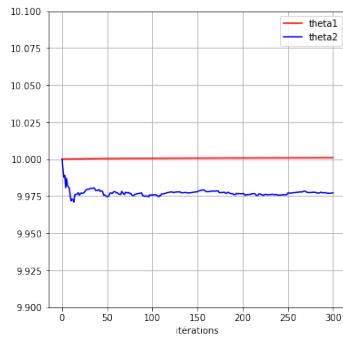


FIGURE 11: Lake 8db : Evolution de $(\theta_k^1)_{0 \leq k \leq N}$ et de $(\theta_k^2)_{0 \leq k \leq N}$, Variation relative de $(\bar{\theta}_k^1)_{0 \leq k \leq N}$ et de $(\bar{\theta}_k^2)_{0 \leq k \leq N}$, Variation relative de $(\bar{\theta}_k^1)_{N_0 \leq k \leq N}$ et de $(\bar{\theta}_k^2)_{N_0 \leq k \leq N}$

Résultats pour SNR = 12db

Les résultats sont présentés dans les figures 12, 13, 14 et 15. Les commentaires sur ces figures sont présentés dans le jupyter notebook.

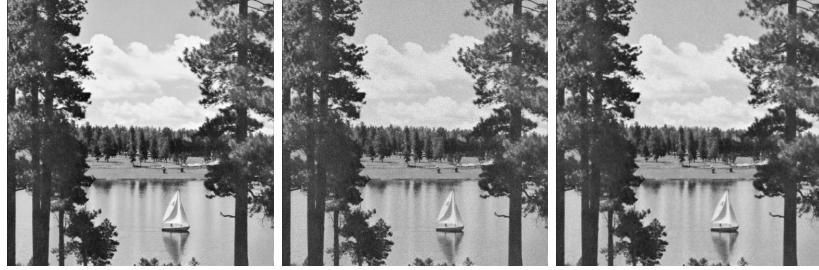


FIGURE 12: Image réelle

FIGURE 13: Image bruitée SNR = 12 db

FIGURE 14: Image débruitée avec SAPG et TGVdenoising

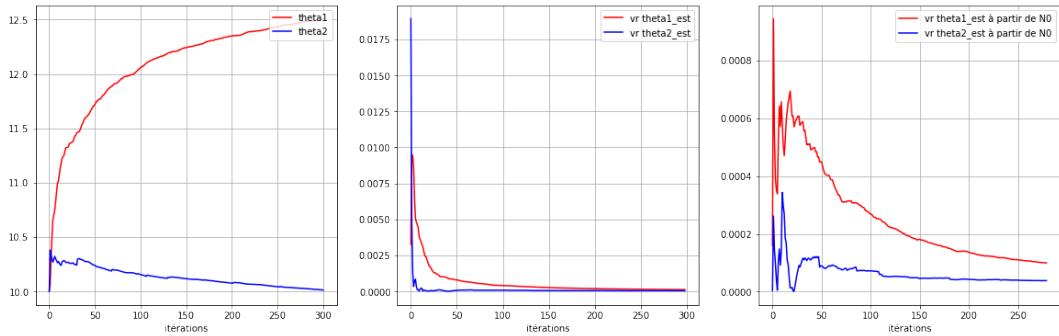


FIGURE 15: Lake 12db : Evolution de $(\theta_k^1)_{0 \leq k \leq N}$ et de $(\theta_k^2)_{0 \leq k \leq N}$, Variation relative de $(\bar{\theta}_k^1)_{0 \leq k \leq N}$ et de $(\bar{\theta}_k^2)_{0 \leq k \leq N}$, Variation relative de $(\bar{\theta}_k^1)_{N_0 \leq k \leq N}$ et de $(\bar{\theta}_k^2)_{N_0 \leq k \leq N}$

Résultats pour SNR = 20db

Les résultats sont présentés dans les figures 16, 17, 18 et 19. Les commentaires sur ces figures sont présentés dans le jupyter notebook.



FIGURE 16: Image réelle

FIGURE 17: Image bruitée SNR = 20 db

FIGURE 18: Image débruitée avec SAPG et TGVdenoising

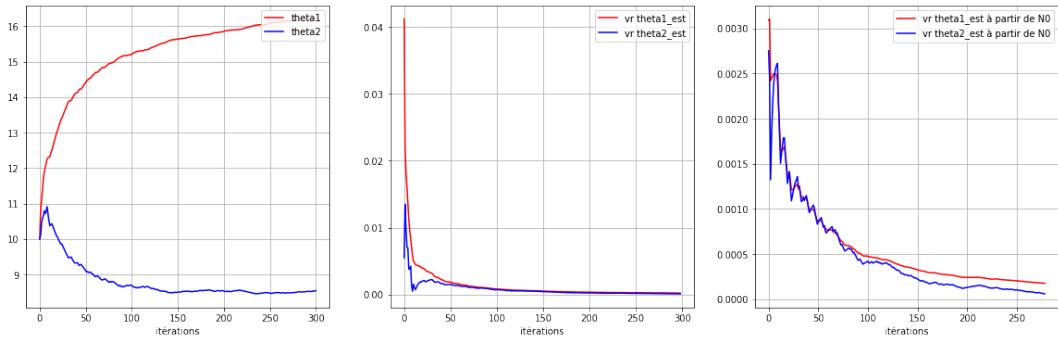


FIGURE 19: Lake 20db : Evolution de $(\theta_k^1)_{0 \leq k \leq N}$ et de $(\theta_k^2)_{0 \leq k \leq N}$, Variation relative de $(\bar{\theta}_1^k)_{0 \leq k \leq N}$ et de $(\bar{\theta}_2^k)_{0 \leq k \leq N}$, Variation relative de $(\bar{\theta}_1^k)_{N_0 \leq k \leq N}$ et de $(\bar{\theta}_2^k)_{N_0 \leq k \leq N}$

Débruitage de différentes images tests pour différentes valeurs du SNR

Le détail des images utilisées est présenté dans le jupyter notebook. Le tableau 1 présentent la moyenne des psnr obtenus dans chaque situation \pm l'écart type.

TABLE 1: Moyenne des psnr \pm l'écart type dans différents cas

	SNR = 8 db	SNR = 12 db	SNR = 20 db
Critère 2	28.51 ± 2.74	33.29 ± 1.96	46.96 ± 0.95
Critère 3	28.48 ± 2.85	33.22 ± 2.00	46.96 ± 0.95

Débruitage de l'image flintstones pour différentes valeurs initiales de paramètres

On applique l'algorithme SAPG sur l'image test flintstones avec SNR = 12 db (figures 20 et 21) pour différentes valeurs initiales $\theta_0^1 = \theta_0^2 = 0.1$, $\theta_0^1 = \theta_0^2 = 10$, et $\theta_0^1 = \theta_0^2 = 40$. Le but de l'expérience est de débruiter l'image 21 avec les valeurs θ^1 et θ^2 de chaque itérations de l'algorithme SAPG, et de calculer ensuite le psnr entre l'image réelle 20 et l'image débruitée et de représenter l'évolution du psnr dans chaque cas (figure25).

Les figures 22, 23 et 24 représentent les images débruitées avec les valeurs finales obtenues par SAPG dans chaque cas.

Les commentaires sur la figure 25 sont présentés dans le jupyter notebook.



FIGURE 20: Image réelle



FIGURE 21: Image bruitée SNR = 12 db



FIGURE 22: Image débruitée avec SAPG et $\theta_0^1 = \theta_0^2 = 0.1$



FIGURE 23: Image débruitée avec SAPG et $\theta_0^1 = \theta_0^2 = 10$



FIGURE 24: Image débruitée avec SAPG et TGVdenoising et $\theta_0^1 = \theta_0^2 = 40$

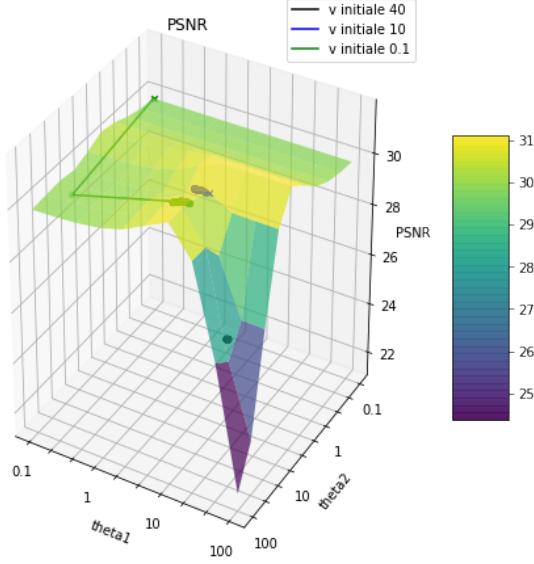


FIGURE 25: flint 12db : Evolution de $(\theta_k^1)_{0 \leq k \leq N}$ et de $(\theta_k^2)_{0 \leq k \leq N}$ et de leur psnr pour différentes valeurs initiales

Références

- [1] Marcelo Pereyra Ana F. Vidal, Valentin De Bortoli and Alain Durmus. Maximum likelihood estimation of regularisation parameters in high-dimensional inverse problems : an empirical bayesian approach. November 27, 2019.
- [2] Amir Beck. *First-Order Methods in Optimization, Chapter6*. Technion-Israel Institute for Technology, Technion, Haifa, Israel, 2017.
- [3] Laurent Condat. Matlab code for total generalized variation denoising. 2016.

3 Annexe : SAPG et TGVdenoising avec Matlab

Dans cette partie, on utilise la version du code Matlab pour reproduire l'expérience de l'application de SAPG+TGVdenoising sur l'image lake pour les valeurs du SNR : 8db, 12db et 20 db. De même que dans la version Python, on choisit les mêmes paramètres que l'article [1], et pour les paramètres du solveur (qu'ils n'ont pas précisé), on choisit les paramètres qui réalisent le meilleur rapport temps de calcul et résultat, ainsi, si on veut améliorer les résultats, on peut par exemple ajuster les paramètres τ et Nbiter dans le calcul du prox, et le Nbiter dans la fonction TGV-denoising. Tous les paramètres choisis sont les mêmes que la version Python, sauf le nombre d'itérations qu'on a augmenté (600 itérations). Pour les trois valeurs de SNR, on remarque que la variation relative des paramètres diminue le long des itérations et converge vers 0, et de même pour la variation relative de leurs moyennes (calculées à partir du seuil N_0), elle diminue et converge vers 0.

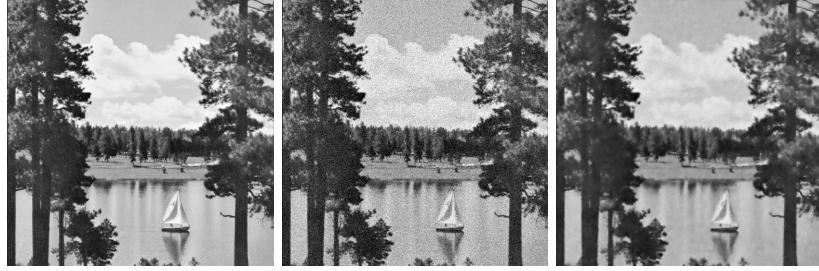


FIGURE 26: Image réelle

FIGURE 27: Image bruitée SNR = 8 db (matlab)

FIGURE 28: Image débruitée avec SAPG et TGVdenoising (matlab)

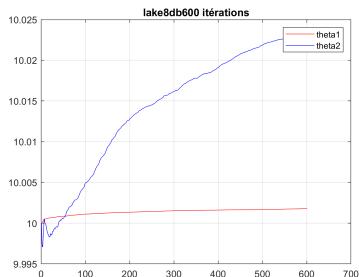


FIGURE 29: Lake 8db : Evolution de $(\theta_k^1)_{0 \leq k \leq N}$ et de $(\theta_k^2)_{0 \leq k \leq N}$ (matlab)

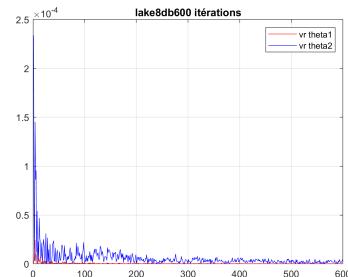


FIGURE 30: Lake 8db : Variation relative de $(\theta_k^1)_{0 \leq k \leq N}$ et de $(\theta_k^2)_{0 \leq k \leq N}$ (matlab)

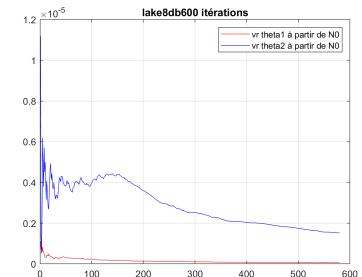


FIGURE 31: Lake 8db : Variation relative de $(\bar{\theta}_1^k)_{N_0 \leq k \leq N}$ et de $(\bar{\theta}_2^k)_{N_0 \leq k \leq N}$ (matlab)

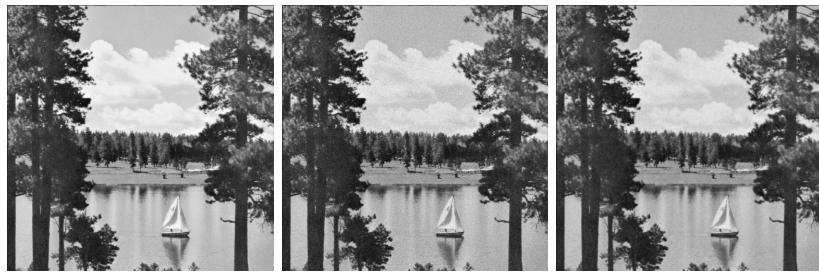


FIGURE 32: Image réelle

FIGURE 33: Image bruitée SNR = 12 db (matlab)

FIGURE 34: Image débruitée avec SAPG et TGVdenoising (matlab)

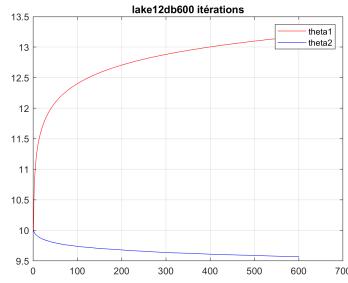


FIGURE 35: Lake 12db : Evolution de $(\theta_k^1)_{0 \leq k \leq N}$ et de $(\theta_k^2)_{0 \leq k \leq N}$ (matlab)

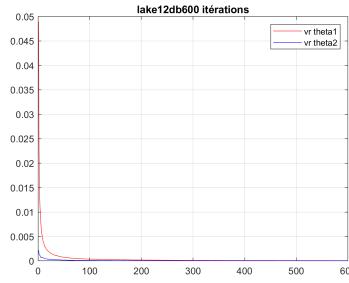


FIGURE 36: Lake 12db : Variation relative de $(\theta_k^1)_{0 \leq k \leq N}$ et de $(\theta_k^2)_{0 \leq k \leq N}$ (matlab)

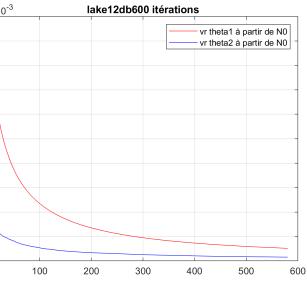


FIGURE 37: Lake 12db : Variation relative de $(\bar{\theta}_1^k)_{N_0 \leq k \leq N}$ et de $(\bar{\theta}_2^k)_{N_0 \leq k \leq N}$ (matlab)



FIGURE 38: Image réelle

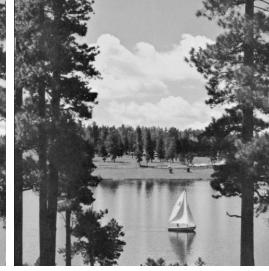


FIGURE 39: Image bruitée SNR = 20 db



FIGURE 40: Image débruitée avec SAPG et TGVdenoising

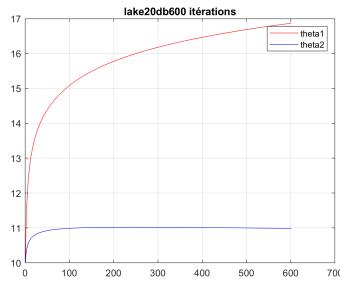


FIGURE 41: Lake 20db : Evolution de $(\theta_k^1)_{0 \leq k \leq N}$ et de $(\theta_k^2)_{0 \leq k \leq N}$ (matlab)

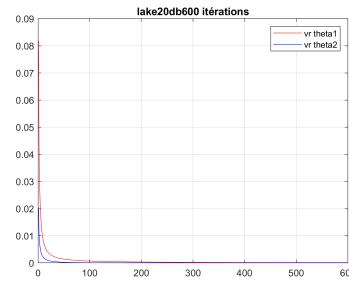


FIGURE 42: Lake 20db : Variation relative de $(\theta_k^1)_{0 \leq k \leq N}$ et de $(\theta_k^2)_{0 \leq k \leq N}$ (matlab)

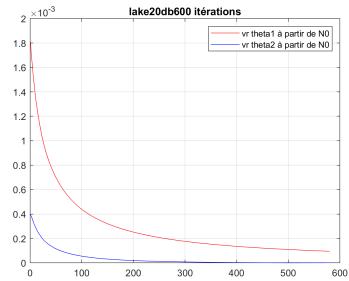


FIGURE 43: Lake 20db : Variation relative de $(\bar{\theta}_1^k)_{N_0 \leq k \leq N}$ et de $(\bar{\theta}_2^k)_{N_0 \leq k \leq N}$ (matlab)