Université de Bordeaux

Master 1 : EDP - Modélisation - Probabilités

TRAITEMENT D'IMAGES PAR PROCÉDÉ PERONA-MALIK

Tara Vanhatalo - Fanny Compaire - Thomas Philibert - Matthieu Nastorg

2018-2019

Équation de la chaleur et différences finies

0.1 Une petite introduction ...

De nos jours, de nombreux secteurs d'activités utilisent le traitement d'images ou, plus généralement, de signaux. Par exemple, les hôpitaux capturent régulièrement de nombreuses images échographiques, les photographes ont souvent besoin de modifier leur prises, les cinéastes leur vidéos ... Cependant, les images recueillies sont rarement de bonne qualité et le traitement d'images permet de pallier ce problème. Il existe ainsi plusieurs manières de modifier le signal de départ (i.e le rendre plus net, filtrer le bruit, ...) qui reposent sur des algorithmes mathématiques. En effet, un signal à traiter peut être modélisé par une fonction réelle :

$$u : \Omega \longrightarrow \mathbb{R}$$

où Ω est un ouvert de $\mathbb{R}^d$ (en l'occurrence, pour une image en deux dimensions, on aurait $d = 2$).
En ce qui concerne le filtrage du bruit, la manière la plus naturelle est de passer par un filtre gaussien qui consiste en une convolution de la fonction u avec la fonction gaussienne :

$$G_\sigma(x) = \frac{1}{\sigma(2\pi)^{d/2}} \mathrm{e}(\frac{-|x|^2}{2\sigma^2}).$$

De plus, cette fonction est une solution fondamentale de l'équation de la chaleur **linéaire** :

$$\begin{cases} \dfrac{\partial u}{\partial t} = \Delta u(t,x) & \text{sur } \Omega \\ u(0,x) = u_0(x) & \text{dans } \Omega \end{cases}$$

Ainsi, il est alors possible de remplacer cette convolution par la résolution de l'équation de la chaleur pour un temps $t = \sigma$ et pour une condition initiale donnée par $u_0(x)$ qui correspond à l'image que nous allons traiter. Tout au long de notre projet nous rajouterons des conditions de bord, dites de NEUMANN, à savoir :

$$\frac{\partial u}{\partial N} = 0 \qquad \text{sur } \partial\Omega \times (0,T)$$

où N est le vecteur normal au bord de Ω. Ces dernières sont très souvent imposées en traitement d'image et permettent d'éviter l'influence d'éléments extérieurs.

Notre projet, qui est détaillé tout au long de ce document, a pour but d'écrire le filtrage par le procédé PERONA-MALIK. Ce dernier consiste en la résolution de l'équation de la chaleur **non-linéaire** illustrée par le problème ci-contre :

$$\begin{cases} \dfrac{\partial u}{\partial t} = \operatorname{div}\left(c\left(|\nabla u|\right)\nabla u\right) & \text{sur } \Omega \\ \dfrac{\partial u}{\partial N} = 0 & \text{sur } \partial\Omega \times (0,T) \\ u(0,x) = u_0(x) & \text{dans } \Omega \end{cases}$$

où c est une fonction monotone de $\mathbb{R}_+$ dans $\mathbb{R}_+$

Dans notre cas, nous aurons $\Omega = (0,1)$ et $T = 1$. Afin de simplifier le problème, nous utiliserons systématiquement des images en noir et blanc.

La différence majeure entre le problème **linéaire** et **non-linéaire** est la suivante : dans l'équation de la chaleur les bords de l'image (zone de fort gradient : changement blanc-noir) sont aussi lissés. Pour éviter cela on utilise un filtre non-linéaire qui est, dans notre cas, celui de PERONA-MALIK. Ce processus de détection des bords provient de la fonction c introduite dans l'équation.

Pour approximer les problèmes précédents, nous utiliserons la méthode des **différences finies**.

Dans un premier temps, nous allons étudier ce problème dans un cas simplifié, à savoir en dimension 1, puis, dans un second temps, nous nous intéresserons au cas de la dimension 2 (pour les images). Finalement, nous essaierons de toujours comparer les filtres linéaires et non-linéaires afin de bien comprendre l'utilité de ces derniers.

0.2 Les différences finies

Un ordinateur ne pouvant résoudre une équation sur un domaine continu, il est nécessaire de discrétiser l'équation en temps et espace. Pour ce faire, nous allons utiliser la méthode des différences finies. Nous allons être en mesure de discrétiser les opérateurs différentiels grâce aux formules de Taylor pour un point x et un pas h :

$$u(x+h) = u(x) + hu'(x) + \frac{h^2}{2}u''(x) + \frac{h^3}{3!}u^{(3)}(x) + \mathrm{O}(h^4) \tag{1}$$

$$u(x-h) = u(x) - hu'(x) + \frac{h^2}{2}u''(x) - \frac{h^3}{3!}u^{(3)}(x) + \mathrm{O}(h^4) \tag{2}$$

En effectuant des combinaisons linéaires des équations ci-dessus, on obtient alors trois approximations de $u'(x)$:

$1-2$	$u'(x) = \frac{u(x+h)-u(x-h)}{2h} + \mathrm{O}(h)$	centrée
1	$u'(x) = \frac{u(x+h)-u(x)}{h} + \mathrm{O}(h)$	décentrée à droite
2	$u'(x) = \frac{u(x)-u(x-h)}{h} + \mathrm{O}(h^2)$	décentrée à gauche

Ce qui nous donne les formules de discrétisation en espace suivantes, pour $x = (x_1, x_2)$:

Différences finies centrées :

$$\frac{\partial u_{i,j}}{\partial x_1} = \frac{u_{i+1,j} - u_{i-1,j}}{2}$$
$$\frac{\partial u_{i,j}}{\partial x_2} = \frac{u_{i,j+1} - u_{i,j-1}}{2}$$

Différences finies décentrées à droite :

$$\frac{\partial u_{i,j}}{\partial x_1} = u_{i+1,j} - u_{i,j}$$

$$\frac{\partial u_{i,j}}{\partial x_2} = u_{i,j+1} - u_{i,j}$$

Différences finies décentrées à gauche :

$$\frac{\partial u_{i,j}}{\partial x_1} = u_{i,j} - u_{i-1,j}$$

$$\frac{\partial u_{i,j}}{\partial x_2} = u_{i,j} - u_{i,j-1}$$

Pour approximer la dérivée temporelle, on va utiliser un schéma d'**Euler explicite** c'est-à-dire que l'on fait une différence finie décentrée à droite. On obtient alors :

$$\frac{\partial u}{\partial t} = \frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\delta t}.$$

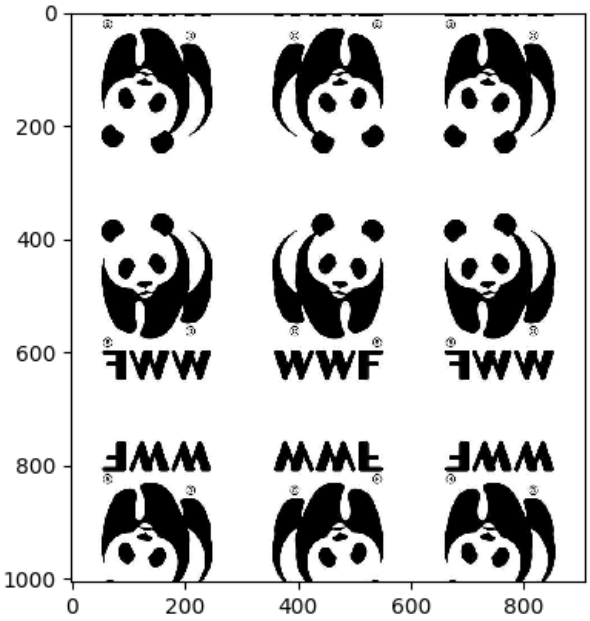
Si l'on se place en dimension 2, alors on notera $u_{i,j}^n$ la valeur de $u(n\delta t, hi, hj)$ où δt est le pas de temps et h le pas en espace. En dimension 1, on notera naturellement u_i^n la valeur $u(n\delta t, hi)$.

Grâce à ces préliminaires, nous serons en mesure de calculer les différents opérateurs différentiels et de développer les formules d'approximation des équations aux dérivées partielles dont nous avons besoin.

Finalement, pour les conditions de Neumann, nous appliquerons systématiquement la même démarche, à savoir prolonger l'image discrète par réflexion. Nous utiliserons alors notre code ***bord.py***.

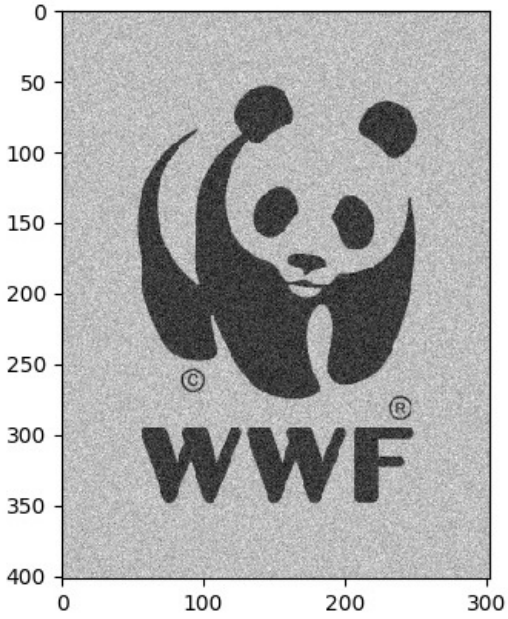
Pour les tests en 2 dimensions nous utiliserons notre mignon panda de WWF que l'on appellera TED :

Si l'on applique notre algorithme de bord pour une réflexion maximale on obtient 9 TED :



De plus, l'idée étant de rendre plus lisse des images bruitées, nous avons écrit un algorithme qui va bruiter nos images : ***bruit.py***

Ainsi, pour la suite, nous allons brouiller TED et travailler sur l'image suivante :



Pour effectuer des comparaisons, nous travaillerons également avec notre amie LENNA et notre photographe BRYAN.

(a) Lenna non-bruitée

(b) Bryan bruité

FIGURE 1

En Dimension 1

Problème Linéaire 1D

Dans cette section, nous allons discrétiser le problème de l'équation de la chaleur linéaire qui est le suivant :

$$\begin{cases} \dfrac{\partial u}{\partial t} = \Delta u(t,x) & \text{sur } \Omega \\ \dfrac{\partial u}{\partial N} = 0 & \text{sur } \partial\Omega \times (0,T) \\ u(0,x) = u_0(x) & \text{dans } \Omega \end{cases} \tag{3}$$

Pour cela, il nous faut, grâce à nos calculs préliminaires, discrétiser le **Laplacien**, qui correspond, dans le cas 1D, à l'approximation de la dérivée seconde. Pour cela on fait la combinaison linéaire suivante à partir des formules de Taylor : 1+2. On obtient

$$\Delta u_i = u_{i+1} - 2u_i + u_{i-1}.$$

On en conclut alors la forme discrétisée de notre équation de la chaleur en dimension 1

$$u_i^{n+1} = u_i^n + \delta t(u_{i+1}^n - 2u_i^n + u_{i-1}^n).$$

Grâce à cette formule, nous avons écrit le code ***problemelin1D.py*** pour le cas 1D dans lequel nous avons introduit notre condition de CFL, calculée en fonction du pas de temps δt.

Pour tester notre algorithme, nous avons utilisé la fonction sinus ainsi que le vecteur correspondant à la première ligne de la matrice de Lenna. On remarquera que la fonction sinus étant déjà lisse, nous avons besoin d'un nombre d'itérations important (ici de l'ordre de 80000) tandis que pour le vecteur "Lenna" seulement un nombre de 500 itérations temporelles est suffisant pour observer des résultats. On remarque que, dans les deux cas, nous perdons en précision et le graphique "s'aplatit" de plus en plus. Les images insérées étant de mauvaise qualité, vous pouvez les retrouver sur notre compte github.

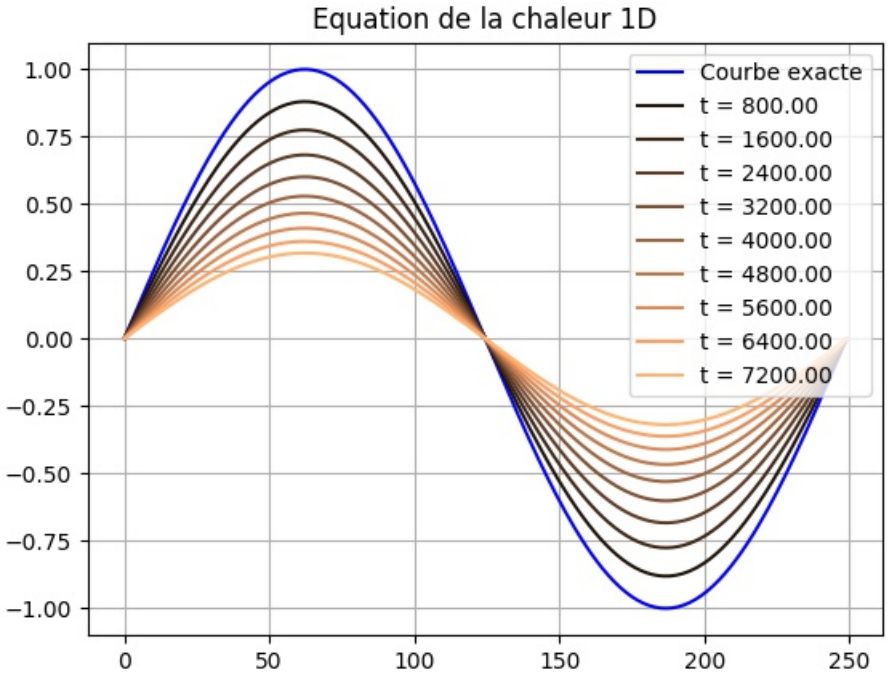


FIGURE 2 – Fonction sinus (0.1/80000)

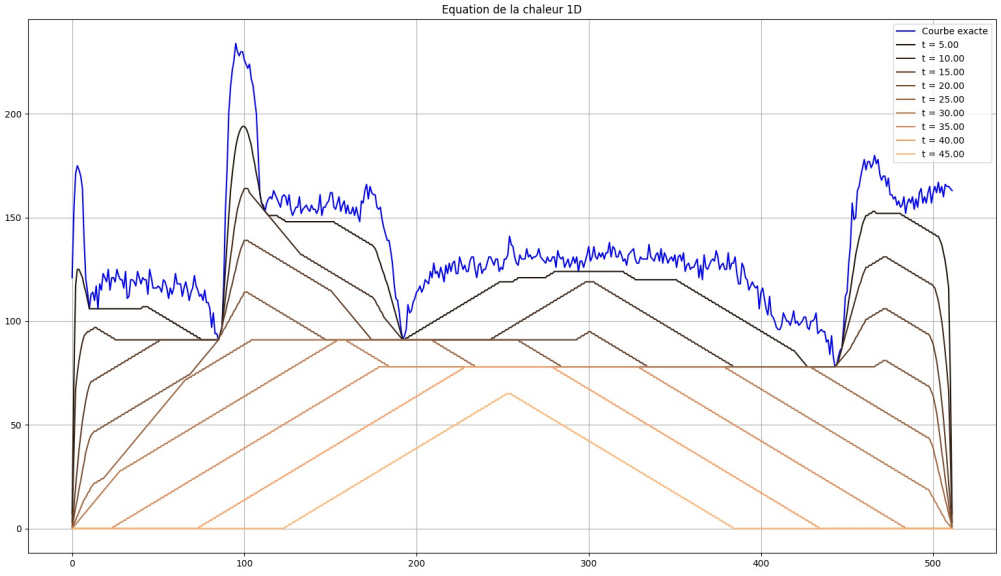


FIGURE 3 – Un bout de Lenna … (0.1/500)

Perona-Malik 1D

Pour améliorer les résultats obtenus par l'équation de la chaleur, Perona et Malik ont proposé de modifier l'équation en y intégrant le processus de détection des bords. Ce modèle permet de lisser les régions homogènes en préservant les bords. On rappelle l'équation de la chaleur **non-linéaire** :

$$\begin{cases} \dfrac{\partial u}{\partial t} = \operatorname{div}\left(c\left(|\nabla u|\right)\nabla u\right) & \text{sur } \Omega \\ \dfrac{\partial u}{\partial N} = 0 & \text{sur } \partial\Omega \times (0,T) \\ u(0,x) = u_0(x) & \text{dans } \Omega \end{cases}$$

où $|\nabla u|$ est la norme du gradient de u et c est une fonction décroissante de $\mathbb{R}^+$ dans $\mathbb{R}^+$.

Dans notre cas, nous utiliserons

$$c(t) = \frac{1}{1+\left(\frac{t}{\alpha}\right)^2}.$$

En dimension 1, la norme du gradient n'est autre que la discrétisation du gradient (avec une racine carrée qui sera supprimée par le terme au carré dans la fonction c), nous n'avons donc pas de calculs particuliers à faire.

On utilise toujours un schéma d'**Euler explicite**

$$\frac{\partial u}{\partial t} = \frac{u_i^{n+1} - u_i^n}{\delta t}.$$

Et on obtient pour la discrétisation de l'espace en utilisant des différences finies décentrées à droite

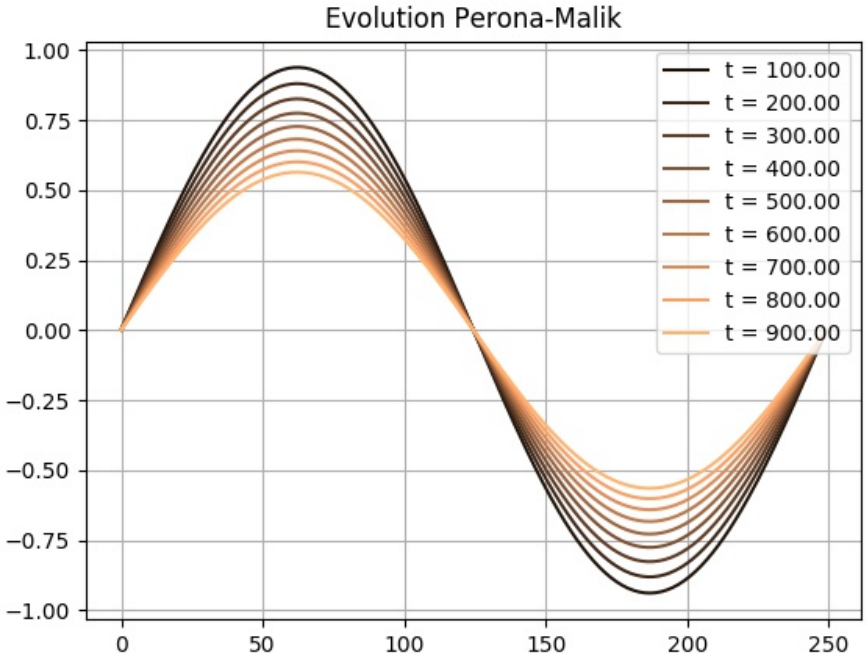
$$c_i \nabla_i = c_i\left(u_{i+1} - ui\right) - c_{i-1}\left(u_i - u_{i-1}\right).$$

On peut finalement écrire la discrétisation générale :

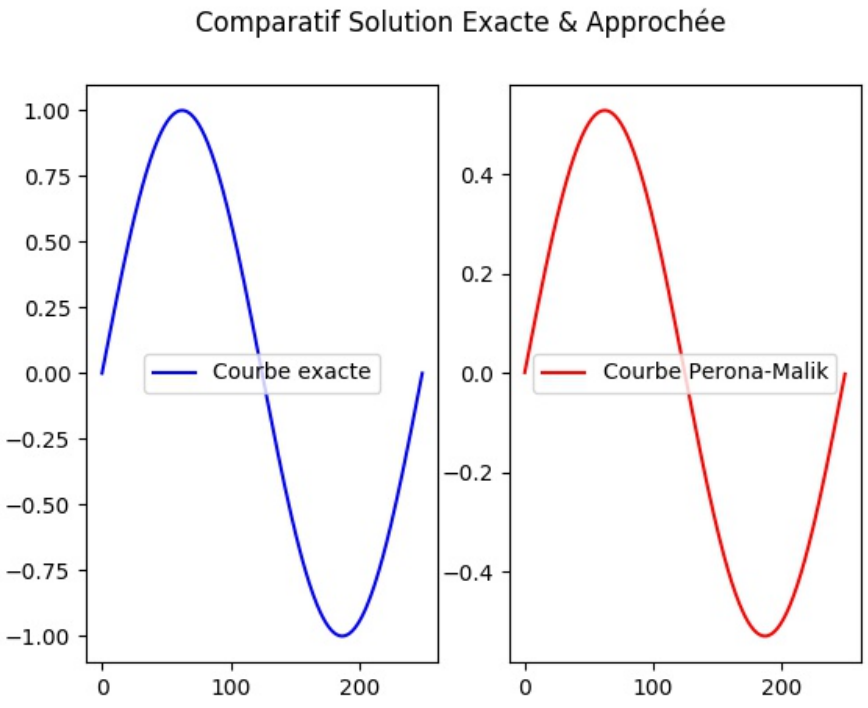
$$u_i^{n+1} = u_i^n + \delta t\left(c_i\left(u_{i+1}^n - u^n i\right) - c_{i-1}\left(u_i^n - u_{i-1}^n\right)\right).$$

Nous avons alors écrit un premier fichier Python contenant la discrétisation du gradient ainsi que la fonction c que nous utilisons : ***C1D.py***. Enfin, nous avons codé le procédé de Perona-Malik dans le cas de la dimension 1 dans le fichier ***Perona1D.py***.

On remarquera très peu de changement pour le sinus dans le cas non-linéaire. En effet, il n'y a pas de grandes variations du gradient et donc pas de "bords" spécifiques à conserver. Cependant dans le graphique du vecteur de Lenna, on observe clairement un lissage. De plus, les points de fort gradient (pics du graphiques) sont beaucoup plus conservés que dans le cas linéaire. En effet, dans ce dernier, on remarque que pour 500 itérations, on arrive rapidement vers une fonction nulle, tandis que pour le cas non-linéaire, même pour 1500 itérations, on obtient un graphique qui ressemble encore à la solution exacte.

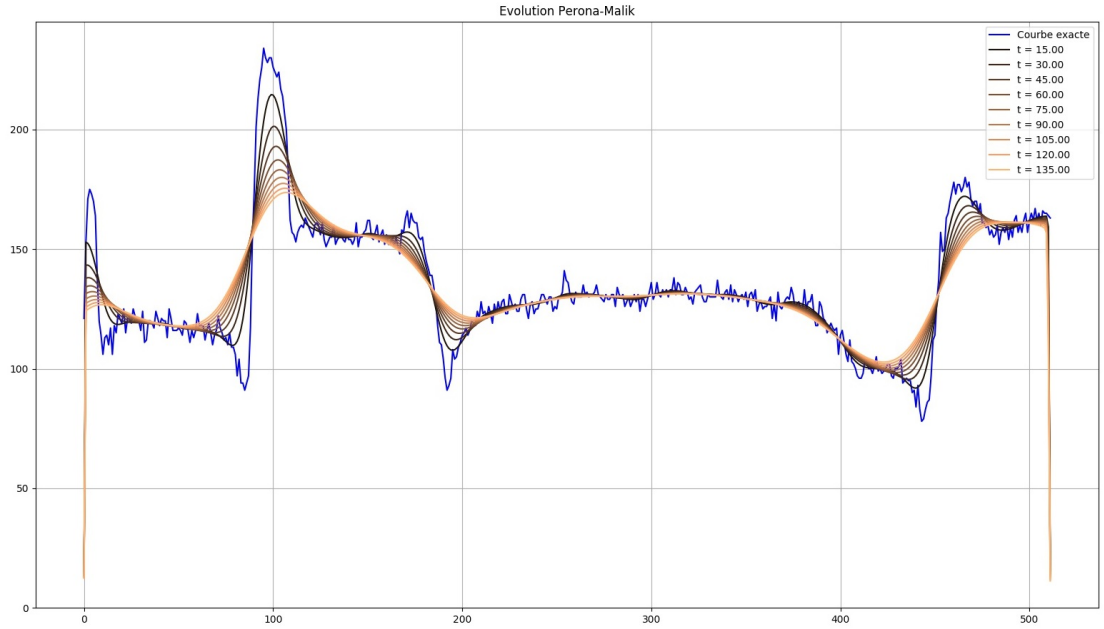


(a) Évolution au cours du temps du sinus Perona-Malik (0.1/10000/1000)

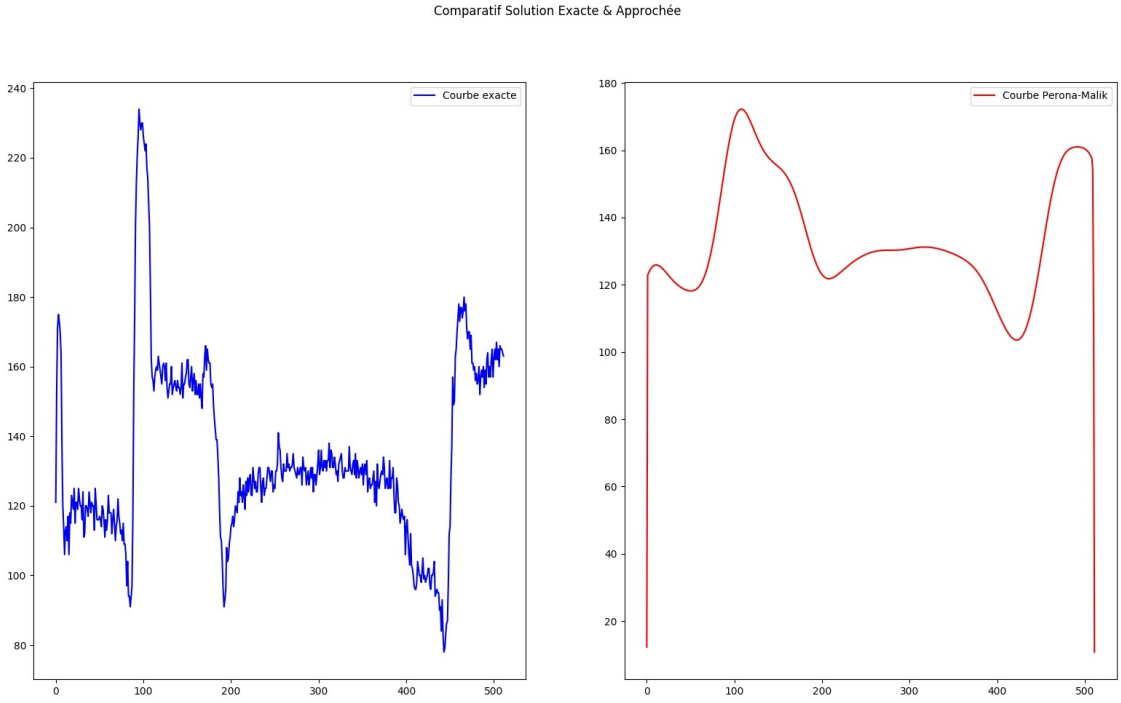


(b) Comparatif solution exacte sinus et dernière itération Perona-Malik

FIGURE 4



(a) Évolution au cours du temps Lenna1D Perona-Malik (0.1/1500/1000)



(b) Comparatif solution exacte Lenna1D et dernière itération Perona-Malik

FIGURE 5

En Dimension 2

Problème Linéaire 2D

On rappelle le problème de l'équation de la chaleur linéaire

$$\begin{cases} \dfrac{\partial u}{\partial t} = \Delta u(t,x) & \text{sur } \Omega \\ \dfrac{\partial u}{\partial N} = 0 & \text{sur } \partial\Omega \times (0,T) \\ u(0,x) = u_0(x) & \text{dans } \Omega \end{cases} \tag{4}$$

Dans cette section, nous allons discrétiser le problème de l'équation de la chaleur dans le cas 2D. Nous allons commencer, comme dans l'exemple précédent, par discrétiser le **Laplacien**. On obtient alors

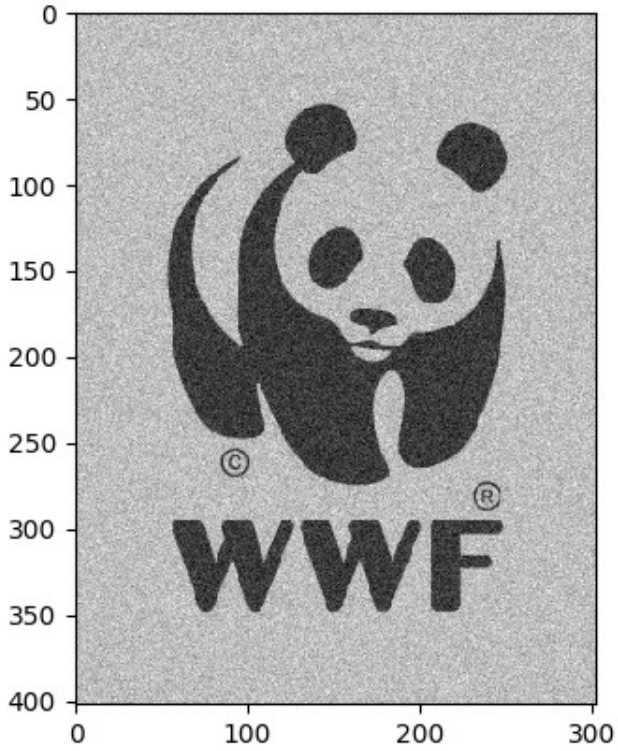
$$\begin{aligned} \Delta u_{i,j} &= \frac{\partial}{\partial x_1}(u_{i+1,j} - u_{i,j}) + \frac{\partial}{\partial x_2}(u_{i,j+1} - u_{i,j}) \\ &= u_{i+1,j} - u_{i,j} - (u_{i,j} - u_{i-1,j}) + u_{i,j+1} - u_{i,j} - (u_{i,j} - u_{i,j-1}) \\ &= u_{i+1,j} - 2u_{i,j} + u_{i-1,j} + u_{i,j+1} - 2u_{i,j} + u_{i,j-1} \\ &= u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}. \end{aligned}$$

On en déduit alors la discrétisation du problème

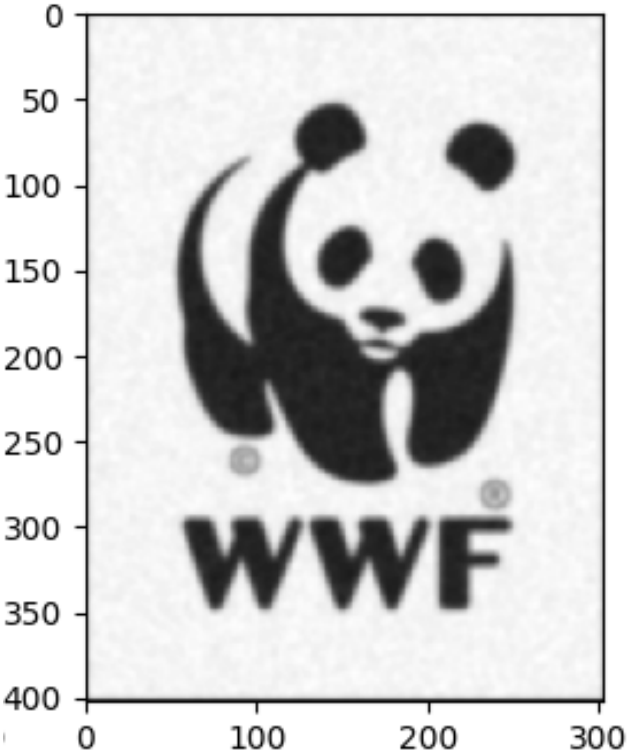
$$u_{i,j}^{n+1} = u_{i,j}^n + \delta t(u_{i+1,j}^n + u_{i-1,j}^n + u_{i,j+1}^n + u_{i,j-1}^n - 4u_{i,j}^n).$$

De cette formule, on écrit l'algorithme ***problemelin2D.py***

Voici le résultat obtenu pour une valeur de $\delta t = 0.1$, et 35 itérations. On observe clairement que l'image est lissée (floutée) par l'équation de diffusion. Cependant, les bords (contours de TED) sont eux-aussi lissés. D'où l'intérêt d'utiliser un filtre non-linéaire (dans notre cas PERONA-MALIK).



(a) TED bruité



(b) TED filtré par l'EDP de la chaleur

FIGURE 6

Perona-Malik 2D

On rappelle l'équation de la chaleur **non-linéaire** dans le cas de PERONA-MALIK

$$\begin{cases} \dfrac{\partial u}{\partial t} = \operatorname{div}\left(c\left(|\nabla u|\right)\nabla u\right) & \text{sur } \Omega \\ \dfrac{\partial u}{\partial N} = 0 & \text{sur } \partial\Omega \times (0,T) \\ u(0,x) = u_0(x) & \text{dans } \Omega \end{cases}$$

où $|\nabla u|$ est la norme du gradient de u et c est une fonction décroissante de $\mathbb{R}^+$ dans $\mathbb{R}^+$.
Dans notre cas, nous utiliserons

$$c(t) = \frac{1}{1+\left(\frac{t}{\alpha}\right)^2}.$$

Comme précédemment, nous obtenons par **Euler explicite**

$$\frac{\partial u}{\partial t} = \frac{u_{i,j}^{n+1} - u_{i,j}^n}{\delta t}.$$

On commence par discrétiser la norme du gradient en utilisant des différences finies décentrées à droite

$$c_{i,j} = c\left(\left|\nabla u_{i,j}\right|\right) = c\left(\sqrt{(u_{i+1,j}-u_{i,j})^2 + (u_{i,j+1}-u_{i,j})^2}\right).$$

On discrétise ensuite

$$c_{i,j}\nabla u_{i,j} = c_{i,j}\left(u_{i+1,j} - u_{i,j} \quad ; \quad u_{i,j+1} - u_{i,j}\right).$$

On sait que, si l'on utilise une discrétisation décentrée à droite du gradient, on peut exprimer la divergence de la façon suivante :

$$\left(\operatorname{div}\left(p\right)\right)_{i,j} = p_{i,j}^1 - p_{i-1,j}^1 + p_{i,j}^2 - p_{i,j-1}^2.$$

En reprenant nos calculs on obtient alors pour la divergence

$$\operatorname{div}\left(c_{i,j}\nabla u_{i,j}\right) = c_{i,j}\left(u_{i+1,j}-u_{i,j}\right) - c_{i-1,j}\left(u_{i,j}-u_{i-1,j}\right) + c_{i,j}\left(u_{i,j+1}-u_{i,j}\right) - c_{i,j-1}\left(u_{i,j}-u_{i,j-1}\right).$$

Sachant que

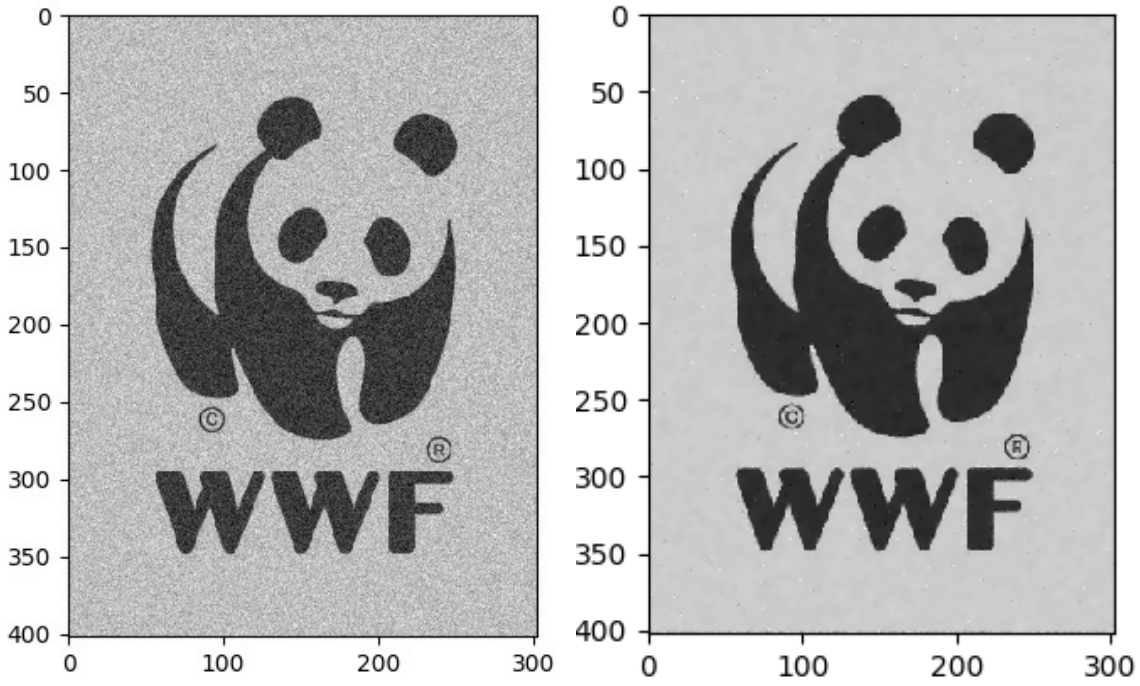
$$u_{i,j}^{n+1} = u_{i,j}^n + \delta t \operatorname{div}\left(c_{i,j}^n \nabla u_{i,j}^n\right),$$

on obtient finalement,

$$u_{i,j}^{n+1} = u_{i,j}^n + \delta t \left(c_{i,j} \left(u_{i+1,j}^n - u_{i,j}^n \right) - c_{i-1,j} \left(u_{i,j}^n - u_{i-1,j}^n \right) + c_{i,j} \left(u_{i,j+1}^n - u_{i,j}^n \right) - c_{i,j-1} \left(u_{i,j}^n - u_{i,j-1}^n \right) \right).$$

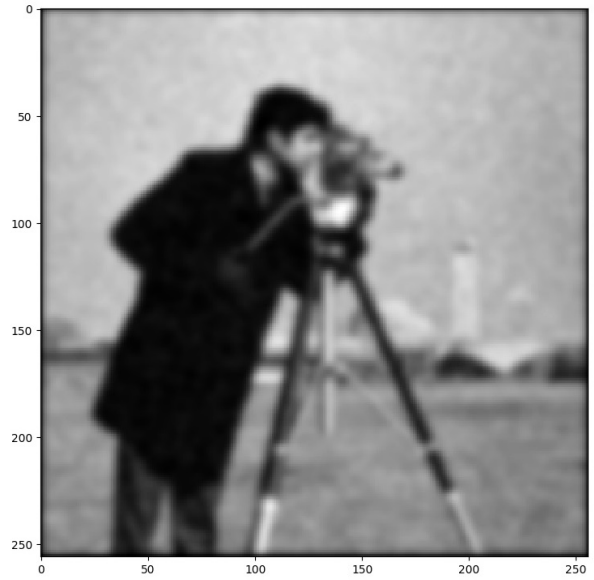
Pour l'écriture algorithmique, nous avons d'abord écrit un fichier comprenant la discrétisation du gradient ainsi que la fonction c que nous utilisons : ***C2D.py***. Puis nous avons finalement écrit l'algorithme pour Perona-Malik en reprenant les mêmes techniques que précédemment : ***Perona2D.py***

Voici le résultat obtenu pour un pas de temps $\delta t = 0.1$, 35 itérations et une valeur de $\alpha = 1000$. L'image est donc lissée et les contours de TED sont parfaitement conservés. Finalement nous faisons en dernière page, un comparatif entre les deux méthodes (en gardant les mêmes paramètres) grâce à LENNA et BRYAN, qui ont pris un sacré coup de jeune grâce à nous!



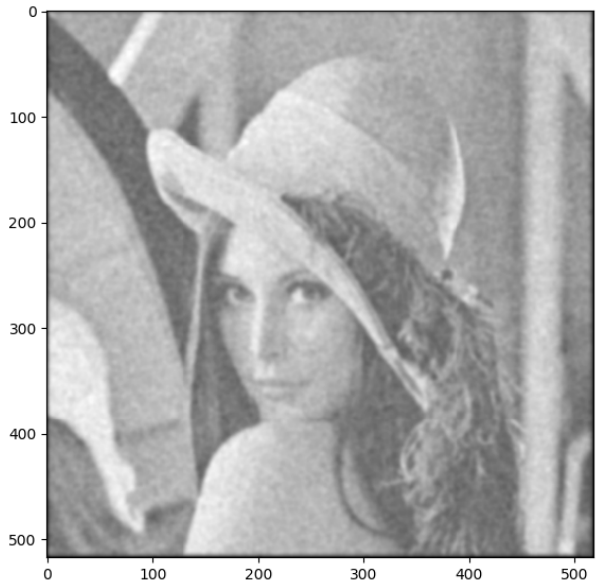
(a) TED bruité (b) TED filtré par Perona-Malik

FIGURE 7



(a) Chaleur : 0.1/50

(b) Perona-Malik : 0.1/50/1000



(c) Chaleur : 0.1/35

(d) Perona-Malik : 0.1/35/1000

FIGURE 8 – Différence entre l'EDP de la chaleur et Perona-Malik

Bibliographie

[1] Jean-François Aujol. Traitement d'images par approches variationnelles et équations aux dérivées partielles, 2005.

[2] Jean-Pierre Antoine et Laurent Jacques. Traitement de l'image : de l'équation de la chaleur aux ondelettes. *Université catholique de Louvain*.

[3] Eric Goncalvès. Résolution numérique, discrétisation des edp et edo, 2005.

[4] Satyanad Kichenassamy. The perona-malik paradox. *Society for Industrial and Applied Mathematics.*, 1997.

[5] Karol Mikula. Image processing with partial differential equations. *Slovak University of Technology*.

[6] Mathias Ortner. Algortihmes pour le traitement de l'image, 2004.

[7] Jitendra Malik Pietro Perona. Scale-space and edge detection using ansiotropic diffusion. *IEEE Transaction on Pattern Analysis and Machine Intelligence. Vol 12.*, 1990.

[8] S.Meignen. Traitement d'image avancé. *Ensimag*, 2016/2017.

[9] Maciej Wielgus. Perona-malik equation and its numerical properties, 2010.