Mini-projet 4 – Optimisation

Mines Paris - PSL
https://moodle.psl.eu/ *2024-2025*

Ce mini-projet, à effectuer en binôme au sein du même groupe de PC (inscription sur Moodle), fera l'objet d'une soutenance orale, incluant la présentation des résultats obtenus et une séance de questions. La présentation comportera obligatoirement un Jupyter Notebook, comportant les réponses aux questions demandées dans le sujet, le code Python réalisé et les graphiques obtenus par des simulations sous Python. ***Ce Jupyter Notebook sera à déposer au plus tard le vendredi 18 avril à 9h*** *sur Moodle.*

Le code rendu devra être fonctionnel, et le notebook exécutable en direct durant la soutenance, dans l'ordre des cellules, avec un "Run All Cells". Tout code non-exécutable pourra donner lieu à une pénalisation sur la note obtenue. Il est également demandé à ce que les paramètres d'entrée du cas d'étude soient modifiables à la volée dans le code le jour de la soutenance. Il n'est pas nécessaire d'apporter un ordinateur le jour de la soutenance.

Restauration d'images

On cherche dans ce sujet à restaurer une image qui présente du bruit (pouvant provenir du capteur, des conditions d'éclairage...). Le bruit numérique est très répandue dans les images, encore plus avec la prolifération de capteurs photos de plus en plus petits (comme ceux présents dans les smartphones). Nous allons voir dans ce mini-projet comme résoudre le problème de restauration d'une image bruitée avec des méthodes de minimisation.

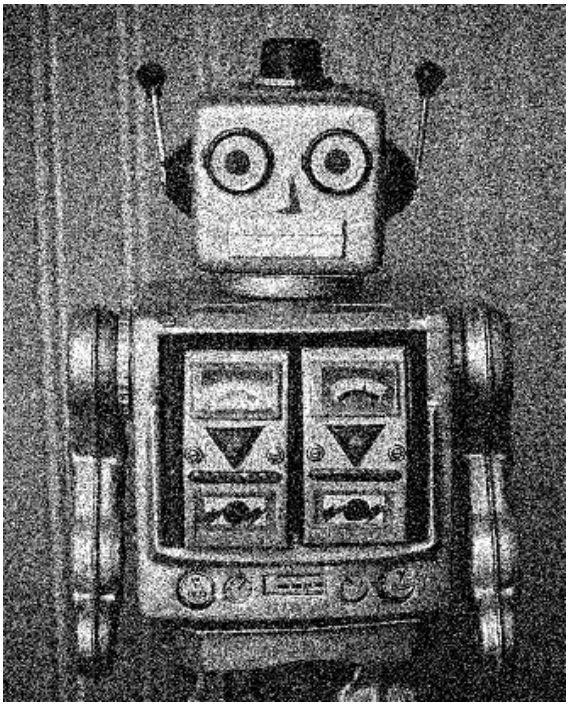
Fig.1 : Une image bruitée d'un robot jouet.

On note u une image de taille $n \times m$ avec $u(i,j)$ le pixel (i,j) en niveau de gris. Les niveaux de gris sont généralement encodés sur un entier de 8 bits avec des valeurs de 0 à 255. Pour la suite de ce projet, nous allons convertir toutes les images avec des pixels encodés sur 32 bits en virgule flottante entre 0 et 1.

On note ∇u le gradient discret de l'image u avec $\nabla u(i,j) = (\nabla_x u(i,j), \nabla_y u(i,j))$:

$$\nabla_x u(i,j) = \begin{cases} u(i+1,j) - u(i,j) & \text{si } i < n \\ 0 & \text{si } i = n \end{cases}$$

$$\nabla_y u(i,j) = \begin{cases} u(i,j+1) - u(i,j) & \text{si } j < m \\ 0 & \text{si } j = m \end{cases}$$

Le gradient discret d'une image u de taille $n \times m$ est donc une matrice de taille $n \times m \times 2$.

On rappelle que la divergence d'une fonction g de $\mathbb{R}^2$ dans $\mathbb{R}$ est définie par : $\text{div}(g) = \frac{\partial g_x}{\partial x} + \frac{\partial g_y}{\partial y}$
Dans le cas discret d'un vecteur v de taille $n \times m \times 2$, la divergence peut se définir comme :

$$\text{div}(v)(i,j) = \begin{cases} v_x(i,j) - v_x(i-1,j) & \text{si } 1 < i < n \\ v_x(i,j) & \text{si } i = 1 \\ -v_x(i-1,j) & \text{si } i = n \end{cases} + \begin{cases} v_y(i,j) - v_y(i,j-1) & \text{si } 1 < j < m \\ v_y(i,j) & \text{si } j = 1 \\ -v_y(i,j-1) & \text{si } j = m \end{cases}$$

On note enfin le Laplacien d'une image : $\Delta u = \text{div}(\nabla u)$.

La problème de restauration d'une image bruitée u_b peut se résoudre par la minimisation d'une fonction définie de $\mathbb{R}^{n \times m}$ dans $\mathbb{R}$ par :

$$\frac{1}{2}\|u - u_b\|_2^2 + \sum_{(i,j)} \|\nabla u(i,j)\|_2^2$$

où $\|\nabla u(i,j)\|_2^2 = \nabla_x u(i,j)^2 + \nabla_y u(i,j)^2$

1 Etude du problème d'optimisation

1. Formuler le problème d'optimisation à résoudre sous la forme :

$$\min_{c(z) \leq 0} f(z)$$

   On précisera les variables de décision z, leur nombre N, les contraintes c ainsi que la fonction objectif f à minimiser.

2. Montrer que la fonction f est convexe et différentiable.

3. Montrer $\nabla f(u) = u - u_b - 2\text{div}(\nabla u) = u - u_b - 2\Delta u$.

2 Résolution numérique

Pour la suite du mini-projet, récupérez les images bruitée et non bruitée disponibles à l'url suivant : `https://cloud.minesparis.psl.eu/index.php/s/P1zwrGIb08Q64NI`.
L'image bruitée a été produite en ajoutant un bruit gaussien d'écrat-type $\sigma = 0.2$ sur l'image normalisée. Utiliser la librairie matplotlib.pyplot pour lire une image avec la fonction imread.

4. Implémenter les fonctions gradient et divergence. Attention à faire des opérations vectorisées dans Python pour être suffisament rapide. Montrer l'image du Laplacien de l'image non bruitée du robot.

5. Implémenter une méthode de descente de gradient à pas fixe. Détailler les paramètres choisis ainsi que les conditions d'arrêt. Montrer l'image résultat. Commenter.

6. Pour avoir une notion plus quantitave de la qualité de la restauration, on peut calculer l'erreur RMSE (Root Mean Square Error) entre l'image d'origine et l'image trouvée par optimisation :

$$\text{RMSE}(u) = \sqrt{\frac{\|u_{VT} - u\|_2^2}{n \times m}}$$

   avec u_{VT} l'image Verité Terrain (c'est-à-dire l'image non bruitée d'origine).

Calculer la RMSE finale pour l'image trouvée u^* dans votre minimisation et comparer là à celle de l'image bruitée RMSE(u_b).

7. Comparer vos résultats de la méthode de gradient à pas fixe avec une méthode votre choix de scipy ou CasADi. Commenter.

3 Etude d'une nouvelle fonction TV-L2

On considère maintenant une nouvelle fonction TV-L2 à minimiser :

$$\text{TV-L2}(u) = \frac{1}{2}\|u - u_b\|_2^2 + \sum_{(i,j)} \|\nabla u(i,j)\|_1$$

où $\|\nabla u(i,j)\|_1 = |\nabla_x u(i,j)| + |\nabla_y u(i,j)|$

8. Montrer que le problème est toujours convexe mais non différentiable.

9. Montrer qu'un sous-gradient de TV-L2 est $g(u) = u - u_b - \text{div}(\text{sign}(\nabla u))$ avec :

$$\text{sign}(x) = \begin{cases} +1 & \text{si } x > 0 \\ -1 & \text{si } x < 0 \\ 0 & \text{si } x = 0 \end{cases}$$

10. Implémenter une méthode de sous-gradient pour résoudre le problème de minimisation de TV-L2(u). Détailler les paramètres choisis ainsi que les conditions d'arrêt. Montrer l'image résultat. Commenter. Utiliser la métrique RMSE pour comparer les minimums de f et de TV-L2. Faire une comparison visuelle des images débruitées en plus de la comparaison des RMSE.

11. Pour accélérer la méthode de sous-gradient, il est fréquent d'utiliser une direction avec momentum, c'est-à-dire à chaque itéré : $p^k = \beta p^{k-1} + (1-\beta)(-g(u^k))$ avec β le paramètre de momentum que nous prendrons dans la suite égal à 0.9. Comparer cette méthode avec la méthode de sous-gradient sans momentum.

12. Appliquer les deux méthodes de débruitage sur une autre image en niveaux de gris de votre choix (après avoir appliqué un bruit gaussien d'écart-type $\sigma = 0.2$). Commenter les résultats.