INSA Département GM : TP Méthodes itératives et restauration d'image

A. Tonnoir (2023 - 2024)

L'objectif de ce TP est de présenter comment les méthodes itératives de résolution d'un système linéaire peuvent permettre de résoudre des problèmes de restauration d'images. Plus précisément, on s'intéressera à reconstruire (au mieux) une image dont un certain nombre de pixels ont été enlevés. C'est une problématique classique de l'inpainting. Ce TP est inspiré des Numerical Tours de Gabriel Peyré.

Une semaine après la séance de TP, vous m'enverrez par mail **vos codes et votre rapport au format pdf**. Dans vos rapports, n'incluez pas vos codes Python, ne recopiez pas le contenu du TP (les questions ou commentaires) et rédigez (si possible) vos rapports en latex (vous pouvez utiliser Lyx). Enfin, je vous invite à faire preuve d'esprit critique et de curiosité. N'hésitez pas à tester des nouvelles choses, même non demandées dans dans le TP !

Premiers tests sur un cas simple

Dans une première partie, nous nous intéresserons à la résolution itérative du système linéaire suivant :

$$(-u_{i-1} + 2u_i - u_{i+1}) + \alpha h^2 u_i = f_i, \quad \forall i \in \{1, \cdots, N\},$$

où $u_0 = u_{N+1} = 0$ et $h = \frac{1}{N+1}$. On prendra f_i t.q. la solution exacte soit le vecteur unitaire $u_i = \cos(2\pi i h)$.

(Théorie)

1. a) Écrire le système linéaire sous la forme $A\underline{u} = \underline{b}$ en précisant A et $\underline{b}$.
   b) Donner $\underline{b}$ en fonction de A et $\underline{u}$ pour que la solution du système linéaire soit $u_i = \cos(2\pi i h)$.
2. a) Dans le cas $\alpha \geq 0$, quelle(s) propriété(s) pouvaient vous donner sur A ?
   b) Quelle(s) méthode(s) itérative(s) peut-on appliquer ?
3. **(Pour aller plus loin)** Que peut-on dire si $\alpha < 0$?

(Code)

1. a) Comment calculer le produit $A\underline{x}$ efficacement (sans utiliser A).
   b) Compléter en conséquence dans le fichier *data.py* le produit matrice-vecteur dans la fonction *produit1*.
2. Dans le fichier *algo.py*, implémenter les méthodes de *descente de gradient* et *gradient conjugué* en exploitant la fonction *produit*.
3. Expliquer comment implémenter la *méthode de Jacobi* en exploitant la fonction *produit* et l'implémenter dans le fichier *algo.py*. Compléter également dans le fichier *data.py* la fonction *diag1* pour évaluer le produit $M^{-1}\underline{x}$ (efficacement).
4. Compléter dans le fichier *main.py* le calcul de $\underline{b}$. Pour différentes valeurs de N, α, tester les différentes méthodes. Commenter.

Restauration d'images

Considérons une image I en niveau de gris formée par $N_x \times N_y$ pixel. On représentera I à l'aide d'une matrice de $\mathcal{M}(N_x, N_y)$. Supposons que I soit détériorée par une fonction $M \in \mathcal{M}(N_x, N_y)$ avec $M_{i,j} = 0$ si le pixel (i,j) est enlevé et 1 sinon. Notre image détériorée est alors $I^{alt} = M * I$ où on considère ici le produit terme à terme (i.e. $I^{alt}_{ij} = M_{ij} I_{ij}$). Notre objectif va être de reconstruire une image I^r approchant "au mieux" l'image I de départ, voir image 1. Pour ce faire, on se propose de résoudre le problème de minimisation : Trouver I^r minimisant $J(I^r)$ où

$$J(I^r) = \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} \left(M_{i,j} I^r_{i,j} - M_{i,j} I^{alt}_{i,j} \right)^2 + \mathcal{R} \sum_{i=2}^{N_x - 1} \sum_{j=2}^{N_y - 1} \left(4I^r_{i,j} - I^r_{i-1,j} - I^r_{i,j-1} - I^r_{i+1,j} - I^r_{i,j+1} \right)^2 .$$

Le paramètre $\mathcal{R} > 0$ est un paramètre dit de régularisation. Dans la fonction ci-dessus, le premier terme est ce qu'on appelle l'**attache aux données** (on veut I^r proche de I sur les pixels connus), et le second un **terme de régularisation** imposant qu'au voisinage de chaque pixel (i,j), l'image ne "varie pas trop".

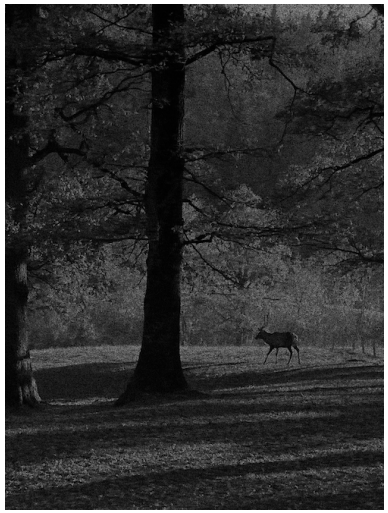
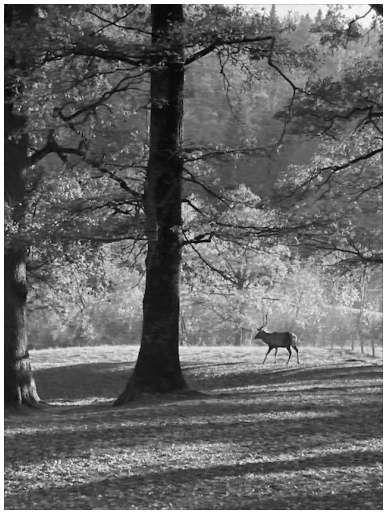

FIGURE 1 – À gauche, image détérioré, À droite, image restaurée.

(Théorie)

1. En ré-indicant sous la forme $l = (i-1)N_y + j$, montrer que le problème de minimisation revient à trouver $\underline{\widetilde{I}}^r \in \mathbb{R}^N$, où $N = N_x \times N_y$, minimisant $\|A\underline{\widetilde{I}}^r - \underline{b}\|_2^2$ en précisant A et $\underline{b}$. On a en particulier $\widetilde{I}^r_l = I^r_{i,j}$ où (i,j) sont les uniques indices t.q. $l = (i-1)N_x + j$.

2. a) Montrer que résoudre le problème de minimisation revient à résoudre l'équation normale :
$$A^t A\underline{\widetilde{I}}^r = A^t\underline{b}.$$
Indication : Par exemple, penser à calculer le gradient de $\|A\underline{\widetilde{I}}^r - \underline{b}\|_2^2$.

   b) Est-ce qu'une méthode directe vous semble envisageable pour résoudre ce système linéaire ?

3. **(Pour aller plus loin)** Justifier que
$$\underline{\widetilde{y}} = A^t A\underline{\widetilde{I}}^r \quad \Leftrightarrow \quad \widetilde{y}_l = M_{i,j} I^r_{i,j} + \mathcal{R}\left(4I^r_{i,j} - I^r_{i-1,j} - I^r_{i,j-1} - I^r_{i+1,j} - I^r_{i,j+1}\right) \tag{1}$$
Indication : Si vous n'arrrivez pas cette question, admettez le résultat pour l'implémentation.

(Code)

1. Dans le fichier *data.py*, compléter les fonctions *laplace* et *produitImg* correspondant respectivement au terme de régularisation et à au produit matrice-vecteur (1).

2. Tester votre code à l'aide du *main.py*. Tester notamment différentes valeurs de paramètres. Commenter.

Pour conclure le TP...

Dans la première partie de ce TP, nous avons revu quelques méthodes itératives classiques et notamment leur implémentation "optimisée". En effet, il n'est pas nécessaire de construire la matrice A pour évaluer le produit $A\underline{x}$. C'est précisément l'une des raisons qui rend les méthodes itératives efficaces. Néanmoins, la performance de ces méthodes itératives est grandement reliée au type de matrice considéré. Pour des problèmes ondulatoires par exemple, comme l'équation d'Helmholtz, les matrices obtenues conduisent à des méthodes itératives généralement peu performantes voir non convergentes.

Dans la deuxième partie, nous avons abordé un problème classique de traitement d'image, la restauration. Ce problème trouve des nombreuses applications dans divers domaines (archéologie, médecine, géophysique...). Il existe de nombreuses autres applications des mathématiques en traitement d'images comme la segmentation, qui vise à connaître les contours d'un objet (utile par exemple en médecine pour déterminer la taille d'une tumeur), ou le recalage d'images dont le but est de mettre en correspondance deux images (utile par exemple en photogrammétrie). Si vous êtes curieux d'en apprendre davantage dans ce domaine, sachez que plusieurs cours de la formation GM sont sur le traitement d'image et que c'est notamment une spécialité du Laboratoire de Mathématiques de l'INSA.