BE : Restauration d'image

Consignes générales :

- *Par groupe de 4 étudiants, vous devez rédiger un rapport dans lequel vous répondrez aux questions du sujet et plus globalement au problème demandé. La qualité de rédaction ainsi que tous les compléments que vous apporterez (réflexions, analyses des réponses, tests sur d'autres données, etc...) seront fortement pris en compte. Ce rapport devra être rédigé sous un traitement de texte (les pages manuscrites ne sont pas acceptées) et rendu sous la forme d'un fichier .pdf.*
- *Toutes les fonctions programmées le seront en Python et seront abondamment commentées. Le rendu du projet se fera sous la forme d'un fichier .zip contenant le rapport ainsi que l'ensemble des programmes et données nécessaire à l'exécution des programmes.*
- *Les fonctions programmées doivent l'être dans un fichier librairie ayant pour nom :* ***Ma321_BE_lib.py****. Dans ce fichier les différentes fonctions doivent être regroupés dans des blocs (commande : #%%) correspondants aux parties du sujet.*
- *Les applications numériques des fonctions du sujet seront dans un fichier ayant pour nom :* ***Ma321_BE_main.py****. Dans ce fichier les différents cas d'études doivent être regroupés dans des blocs séparés.*
- *Les différentes images devront être enregistrés dans un dossier ayant pour nom* ***Images****. Ce dossier doit être mis dans le même répertoire que vos fichiers pythons* ***Ma321_BE_lib.py*** *et* ***Ma321_BE_main.py****.*
- *Les noms des différents membres du groupe doivent apparaître en commentaire en début de fichier.*
- *Le non respect des consignes précédentes entraînera des points de pénalités pouvant être conséquents...*
- *Les questions d'analyses de résultats sont généralement ouvertes. La qualité de vos réflexions sera fortement prise en compte pour la notation de ces questions.*
- **Il est impératif de lire le sujet en entier.**

Dans ce BE nous allons nous intéresser au problème de la restauration d'image (ou de l'agrandissement, cela fonctionne de la même manière). Pour simplifier nous présenterons ce problème pour une image en nuance de gris i.e. où une image est modélisée par une matrice $F = (f_{ij}) \in \mathcal{M}_{p,q}(\mathbb{R})$, où f_{ij} est la valeur de la luminosité au pixel de la ligne i et colonne j, et p (resp. q) est le nombre de lignes (resp. de colonnes) de pixels. On rappelle que les coefficients f_{ij} sont des entiers compris entre 0 et 255 (2^8 valeurs possibles). Le problème de la restauration d'image couleur se traite de la même façon en considérant chaque couche (RGB) indépendamment.

Supposons que l'on nous transmette une image « abîmée » F i.e. que certains pixels n'ont pas été transmis. Pour simplifier ici nous modéliserons ces pixels non transmis par des pixels valant exactement 0, ce qui correspond à des pixels noirs [1]. On aimerait « interpoler » entre ces pixels de façon à remplir ces zones indéterminées/noires. Par exemple prenons une veille image abîmée par le temps :

Figure 1 – Image abîmée

1. Il est facile d'adapter la suite pour des pixels valant d'autres valeurs.

Pour représenter les pixels non transmis ou abîmés on notera $M = (m_{ij})$ la matrice appelée **matrice du masque**, c'est à dire $m_{ij} \in \{0, 1\}$ avec m_{ij} qui vaut 1 si le pixel a été transmis et 0 si le pixel n'a pas été transmis (lorsque$f_{ij} = 0$). Nous noterons également $\overline{M} = (\overline{m}_{ij}) = 1 - m_{ij}$ le contraire du masque. Le but est de retrouver une image complète, qui corresponde le plus possible à l'image originale. Pour cela on cherche une matrice $U = (u_{ij})$ qui soit telle que $u_{ij} = f_{ij}$ si le pixel a été transmis et qui soit la plus « lisse » possible i.e. où les pixels « noirs » soient « coloriés » en utilisant à leurs « voisinages » les pixels correctement transmis. Sur l'exemple précédent avec un éditeur d'image (paint par exemple) nous avons repasser en noir (**attention il faut tracer en couleur noire pure et non avec des dégradés sur les bords de votre tracé**) les parties que l'on juge abîmées :

FIGURE 2 – Image abîmée dont les pixels abîmés sont recouverts de **noir pur**. La qualité de votre tracé en noir influencera fortement le résultat final.

Dans une première partie nous produirons un cadre formel au problème précédent. Ce cadre formel nous conduira à penser les images comme des fonctions à deux variables puis à discrétiser les équations obtenues. Enfin la seconde partie de ce BE consistera en la réalisation effective d'un algorithme résolvant ce problème.

On notera dans ce BE le **produit matriciel de Hadamard** de deux matrices A et B de même taille :

$$A \odot B$$

défini par :

$$(A \odot B)_{ij} = a_{ij} \times b_{ij}$$

où les a_{ij} sont les coefficients de A et b_{ij} les coefficients de B.

1 Formalisation théorique

Dans cette partie nous allons penser les images U et F comme des fonctions à deux variables réelles définies sur le rectangle $[0, p] \times [0, q]$ à valeurs dans $\mathbb{R}$. **Dans toute cette partie on supposera que les fonctions intervenant sont de classe $\mathcal{C}^\infty$.**

Soit $\Omega \subset [0, p] \times [0, q]$ un fermé connexe (i.e. d'un seul tenant) et notons $\partial\Omega$ la frontière (:= le bord) de Ω. Moralement on considérera l'intérieur de Ω noté $\mathring{\Omega} := \Omega \setminus \partial\Omega$ comme un ensemble de pixels non transmis (=abîmés) et le bord $\partial\Omega$ comme un ensemble de pixels correctement transmis. Le but est donc de trouver une fonction :

$$u : \quad \Omega \quad \to \quad \mathbb{R}$$

vérifiant certains critères, critères que nous développerons dans la suite, telle que la restriction de u au bord $\partial\Omega$ soit égale à f sur ce même bord. La fonction f sur $\partial\Omega$ représente les pixels correctement transmis. On note cela :

$$u_{|\partial\Omega} = f_{|\partial\Omega}$$

On peut représenter graphiquement cette situation par :

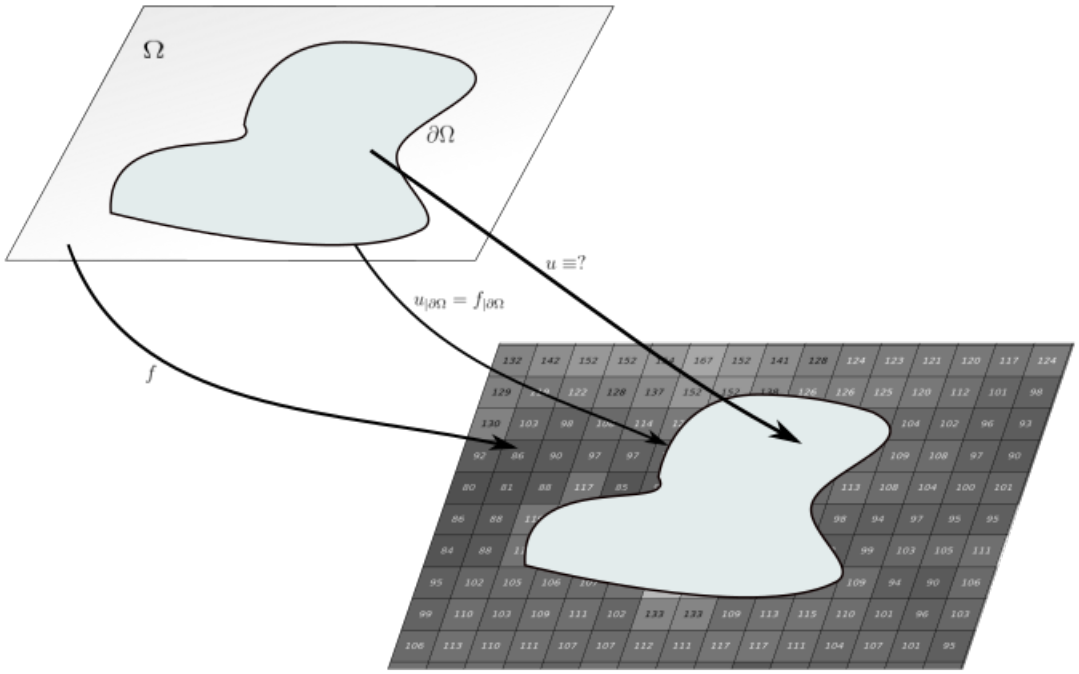

FIGURE 3 – La fonction f est bien définie sur l'extérieur de Ω et on cherche u tel que $u_{|\partial\Omega} = f_{|\partial\Omega}$ et u sur $\mathring{\Omega}$.

Afin de traiter ce problème nous allons commencer par une formulation très générale faisant intervenir une généralisation des équations d'Euler-Lagrange vues en TD.

On notera les variables de u et f : $(x, y) \in \Omega$ et :

$$\boxed{u_1 := \frac{\partial u}{\partial x}} \quad \boxed{u_2 := \frac{\partial u}{\partial y}}$$

De même pour les dérivées partielles de f.

1. Avec les notations précédentes posons une fonction d'action de la forme :

$$S(u) := \iint_\Omega L(x, y, u(x, y), u_1(x, y), u_2(x, y))\, dxdy$$

où L, le lagrangien, est une fonction de classe $\mathcal{C}^\infty$ définie sur $\mathbb{R}^5$ à valeurs dans $\mathbb{R}$. En vous inspirant de la démonstration des équations d'Euler-Lagrange vue dans le TD1, exercice 1, démontrer que les points critiques de S vérifient l'équation suivante :

$$\boxed{\frac{\partial L}{\partial u} = \frac{\partial}{\partial x}\left(\frac{\partial L}{\partial u_1}\right) + \frac{\partial}{\partial y}\left(\frac{\partial L}{\partial u_2}\right)}$$

2. Voyons à présent un exemple important.
   (a) Posons comme lagrangien :

$$L(x, y, u(x, y), u_1(x, y), u_2(x, y)) := \|\nabla u(x, y)\|_2^2$$

Montrer que dans ce cas une solution u au problème de cette partie vérifie les conditions suivantes :

$$\boxed{(\mathcal{P}) : \begin{cases} \Delta u = 0 \\ u_{|\partial\Omega} = f_{|\partial\Omega} \end{cases}}$$

où Δu note l'opérateur laplacien de u.

(b) Pouvez-vous donner une justification du lagrangien précédent pour la résolution de notre problème de restauration d'image ?

(c) Afin d'implémenter et résoudre numériquement le problème $(\mathcal{P})$ nous allons le discrétiser.

i. Soit $g : \mathbb{R} \to \mathbb{R}$ de classe $\mathcal{C}^\infty$. Montrer que pour $h > 0$ « petit » on a l'approximation suivante pour tout $x \in \mathbb{R}$:

$$g''(x) \simeq \frac{g(x+h) - 2g(x) + g(x-h)}{h^2}$$

ii. Pour une image on supposera que le point $(x, y) \in \Omega$ est la position (x, y) d'un pixel i.e x est le numéro de ligne de la matrice et y le numéro de la colonne. On considérera alors $h = 1$. En déduire que le problème $(\mathcal{P})$ se discrétise sous la forme :

$$(\mathcal{P})_d : \quad \begin{cases} (u_{x+1,y} - 2u_{x,y} + u_{x-1,y}) + (u_{x,y+1} - 2u_{x,y} + u_{x,y-1}) = 0 & \text{pour tout } (x,y) \in \mathring{\Omega} \\ u_{x,y} = f_{x,y} & \text{pour tout } (x,y) \in \partial\Omega \end{cases}$$

avec cette fois les u_{xy} et f_{xy} les coefficients des matrices U et F respectivement.

(d) Le problème précédent $(\mathcal{P})_d$ est local. Globalement i.e. sur toute l'image, nous pouvons en déduire la forme généralisée suivante :

$$(\mathcal{P})_d^{\text{global}} : \quad \begin{cases} D_p U + U D_q^T = 0 \\ U \odot M = F \end{cases}$$

où la matrice D_r avec r un entier tel que $r > 2$ est une matrice $D_r \in \mathcal{M}_{r-2,r}(\mathbb{R})$ définie par :

$$(D_r)_{ij} := \begin{cases} 1 & \text{si } i = j \\ -2 & \text{si } i+1 = j \\ 1 & \text{si } i+2 = j \\ 0 & \text{sinon} \end{cases}$$

Justifier que le problème $(\mathcal{P})_d^{\text{global}}$ se généralise sous la forme du problème d'optimisation sous contrainte suivant :

$$\left(\widetilde{\mathcal{P}}\right)_d^{\text{global}} : \quad \arg\min_{U \in \mathcal{C}} J(U)$$

avec :

— $J(U) := \frac{1}{2}\left(\|D_p U\|_F^2 + \left\|U D_q^T\right\|_F^2\right)$, avec $\|\bullet\|_F$ la norme de Frobenius.

— et où la contrainte $\mathcal{C}$ est décrite par :

$$\mathcal{C} := \{U \in \mathcal{M}_{p,q}(\mathbb{R}) \ \backslash \ \forall (i,j) \in [\![1,p]\!] \times [\![1,q]\!] \text{ tel que si } m_{ij} = 1 \text{ , on a } u_{ij} = f_{ij}\}$$

C'est sous la forme de ce dernier problème $\left(\widetilde{\mathcal{P}}\right)_d^{\text{global}}$ que nous allons dans la partie suivante résoudre le problème de la restauration d'une image en utilisant une version matricielle de l'algorithme du gradient conjugué.

2 Résolution numérique

Cette partie se base sur les notations des parties précédentes. Dans cette partie nous allons résoudre numériquement le problème final de la partie précédente :

$$\left(\widetilde{\mathcal{P}}\right)_d^{\text{global}} : \begin{cases} \arg\min_{U\in\mathcal{C}} J(U) \\ \mathcal{C} := \{U \in \mathcal{M}_{p,q}(\mathbb{R}) \ \backslash\ \forall (i,j) \in [\![1,p]\!] \times [\![1,q]\!] \text{ tel que si } m_{ij} = 1 \text{ , on a } u_{ij} = f_{ij}\} \end{cases}$$

On notera $\langle \bullet, \bullet \rangle$ le produit scalaire canonique sur les matrices que l'on rappelle défini de la manière suivante :

$$\langle A, B \rangle := Tr\left(A^T B\right)$$

1. Quel est l'opérateur sous Python qui correspond au produit matriciel de Hadamard ?
2. Posons $X = U - F$. Montrer que le problème $\left(\widetilde{\mathcal{P}}\right)_d^{\text{global}}$ revient à minimiser la fonction :

   $$\Phi(X) := \frac{1}{2}\langle X, A(X)\rangle + \langle B, X\rangle$$

   où :

   $$\begin{cases} A(X) := \left(D_p^T D_P X + X D_q^T D_q\right) \odot \overline{M} \\ B := \left(D_p^T D_p F + F D_q^T D_q\right) \odot \overline{M} \end{cases}$$

   Nous allons montrer que cette fonction est une forme quadratique sur l'espace complémentaire de l'espace des contraintes :

   $$\overline{\mathcal{C}} := \{X \in \mathcal{M}_{p,q}(\mathbb{R}) \ \backslash\ \forall (i,j) \in [\![1,p]\!] \times [\![1,q]\!] \text{ tel que si } m_{ij} = 1 \text{ , on a } x_{ij} = 0\}$$

3. Démontrer que l'opérateur $A(\bullet)$ est linéaire.
4. Montrer que pour A, B et C trois matrices de même taille $p \times q$ alors pour tout i les éléments diagonaux suivants sont égaux :

   $$\left(C^T(A \odot B)\right)_{ii} = \left(A^T(C \odot B)\right)_{ii}$$

   En déduire que l'opérateur $A(\bullet)$ est symétrique pour le produit scalaire canonique sur les matrices i.e :

   $$\forall (X, Y) \in \mathcal{M}_{p,q}(\mathbb{R})^2,\ \langle Y, A(X)\rangle = \langle A(Y), X\rangle$$

5. Soit H une matrice de taille $p \times q$, calculer la dérivée directionnelle de Φ en X dans la direction H. En déduire le « gradient » de Φ .
6. Réécrire l'algorithme du gradient conjugué dans ce cas, c'est-à-dire où le produit scalaire usuel de vecteur $\langle x, y\rangle = x^T y$ est remplacé par le produit scalaire matriciel $\langle A, B\rangle = Tr(A^T B)$. On appelle cet algorithme le **gradient conjugué matriciel**.
7. Programmation : dans le fichier ***Ma321_BE_lib.py*** implémenter les fonctions suivantes :
   (a) une fonction **masque(F,v)** qui prend en entrée une matrice F représentant une image avec des pixels « abîmés » (pixels noirs) et v un entier proche de 0 (on prendra en général $v = 0$) et qui renvoie en sortie la matrice du masque.
   (b) une fonction **D(r)** qui prend en entrée un entier et qui renvoie la matrice $D_r \in \mathcal{M}_{r-2,r}(\mathbb{R})$ de l'énoncé.
   (c) une fonction **A(X,M)** qui prend en entrée une matrice X et la matrice du masque M et qui renvoie l'opérateur $A(X)$ de l'énoncé.
   (d) une fonction implémentant l'algorithme du gradient conjugué matriciel **GCM(M,F,X0,epsilon)** où M est la matrice du masque, F la matrice de l'image « abîmée », X_0 une matrice d'initialisation et ϵ un réel représentant une précision souhaitée. On pourra prendre dans la pratique $\epsilon = 1$.
   (e) enfin une fonction **restauration(F,M,epsilon)** qui utilise les fonctions précédentes et qui renvoie une image restaurée.

   Quelques conseils pour le programme python :
   — on pourra utiliser la librairie opencv vu en Ma313 afin de charger les images. Par exemple pour charger une image en nuance de gris puis la visualiser :

```
F=cv2.imread('test1.png',0)
cv2.imshow("visualisation", F)
```

— il est nécessaire de convertir les images en uint8 avant de les afficher. Avec opencv on pourra utiliser par exemple :

```
cv2.convertScaleAbs(X)
```

Tester votre programme sur les images **test_1.png** et **test_2.png**.

8. En appliquant la démarche expliquée en introduction restaurer l'image **orig1.jpg.**
9. Programmation : dans le fichier ***Ma321_BE_lib.py*** implémenter une fonction **restauration_couleur(F,M,epsilon)** qui utilise la fonction précédente sur les trois couches d'une image couleur « abîmée » et qui renvoie une image couleur restaurée. Tester votre programme sur les images **test_1.png** et **test_2.png** puis sur **orig1.jpg.**
10. **Bonus** : réaliser des sorties vidéos des différentes étapes du gradient conjugué. Observer le processus de diffusion des couleurs correctes dans les pixels manquants.

FIGURE 4 – Exemple de restauration d'image.