TP9 – Tomographie

Transformation de Radon et tomographie

La *transformation de Radon* d'une image $f(x, y)$ consiste à effectuer la projection orthogonale de cette image sur une droite $\mathcal{D}$ d'angle polaire θ, paramétrée par une abscisse t :

$$p_\theta(t) = \int_{L(\theta,t)} f(z)\, dz \tag{1}$$

Dans cette expression :

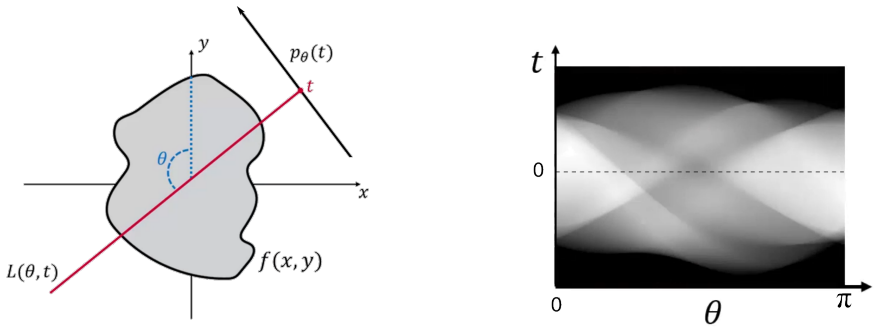
- La droite $L(\theta, t)$, qui a pour équation cartésienne $x \cos\theta + y \sin\theta = t$, est orthogonale à $\mathcal{D}$.
- La notation $f(z)$ est un raccourci pour désigner $f(x(z), y(z))$, z étant une abscisse le long de $L(\theta, t)$.

Les droites $\mathcal{D}$ et $L(\theta, 0)$, associées aux coordonnées respectives t et z, définissent donc un repère orthonormé dont la matrice de passage $\mathbf{M}$, relativement au repère image, est une rotation d'angle θ (cf. figure 1-a) :

$$\mathbf{M} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \tag{2}$$

Contrairement à la transformation de Fourier, qui est inversible, il est clair que la transformation de Radon, qui est une projection, n'est pas inversible : en d'autres termes, il est impossible de retrouver l'image à partir d'une unique transformée de Radon. Le but de la *tomographie* est de reconstruire l'image à partir d'un certain nombre de transformées de Radon obtenues pour différentes valeurs de θ.

En pratique, l'image $f(x, y)$ est discrète et les données sont regroupées dans une matrice 2D appelée *sinogramme* (cf. figure 1-b), dont le nombre de colonnes est égal au nombre n_θ d'orientations de $\mathcal{D}$, et le nombre de lignes est égal au nombre n_{rayons} de droites $L(\theta, t)$, qui est supposé être le même pour chaque valeur de θ.



(a) (b)

FIGURE 1 – (a) Définition des variables (θ, t) (cet exemple correspond au cas particulier $t = 0$). (b) Sinogramme : chaque colonne correspond à une valeur de θ.

Bien entendu, ce problème n'a d'intérêt que s'il est impossible d'obtenir directement l'image $f(x, y)$. En particulier, cela permet de « cartographier » le corps humain sans avoir besoin de le découper en tranches. En effet, les rayons X fournissent des transformées de Radon du corps humain, car ils sont suffisamment énergétiques pour traverser le corps humain en ligne droite (absence de réfraction), mais pas trop énergétiques quand même, ce qui fait que leur *atténuation* dépend directement de l'intégrale (1), dans laquelle la fonction $f(x, y)$, qui constitue l'inconnue du problème, désigne le *coefficient d'atténuation* local des rayons X (ce coefficient est plus élevé pour les os que pour les tissus).

Exercice 1 : résolution algébrique de la tomographie

Les équations discrètes à résoudre pour retrouver la fonction f en chaque pixel à partir de son sinogramme discret, peuvent être regroupées sous forme matricielle :

$$\mathbf{W}\,\mathbf{f} = \mathbf{p} \tag{3}$$

où :

- le vecteur $\mathbf{p}$, de taille $n_{\text{mesures}} = n_\theta\, n_{\text{rayons}}$, est une version vectorisée du sinogramme ;
- le vecteur $\mathbf{f}$, qui constitue l'inconnue, contient les $n_{\text{pixels}} = n_{\text{lignes}}\, n_{\text{colonnes}}$ valeurs de la fonction f ;
- la matrice $\mathbf{W}$, de taille $n_{\text{mesures}} \times n_{\text{pixels}}$, contient la longueur du trajet de chaque rayon à travers chaque pixel ; comme cette matrice est fastidieuse à calculer, elle vous est fournie (fonction `calcul_W`).

La résolution en moindres carrés du système linéaire (3) s'écrit :

$$\min_{\mathbf{f}\in\mathbb{R}^{n_{\text{pixels}}}} \|\mathbf{W}\,\mathbf{f} - \mathbf{p}\|^2 \tag{4}$$

mais comme la matrice $\mathbf{W}$ est généralement énorme, il est impossible de calculer sa pseudo-inverse. Un algorithme plus adapté à la résolution de (3) est *l'algorithme de Kaczmarz*, qui est itératif :

$$\mathbf{f}^{k+1} = \mathbf{f}^k + \frac{p_i - \mathbf{w}_i\mathbf{f}^k}{\|\mathbf{w}_i\|^2}\,\mathbf{w}_i^\top \tag{5}$$

où $i = k\ [n_{\text{mesures}}]$ (modulo), et $\mathbf{w}_i$ et p_i désignent, respectivement, la $i^{\text{ème}}$ ligne de $\mathbf{W}$ et le $i^{\text{ème}}$ élément de $\mathbf{p}$. Remarque : si $\|\mathbf{w}_i\| = 0$, l'itération (5) doit être remplacée par $\mathbf{f}^{k+1} = \mathbf{f}^k$.

Lancez le script `calcul_sinogramme`, qui calcule le sinogramme d'une image carrée, puis écrivez la fonction `kaczmarz`, appelée par le script `exercice_1`, permettant de résoudre le système (3) par des itérations du type (5).

Le critère d'arrêt le plus simple consiste à fixer à l'avance un nombre $k_{\max} = n_{\text{boucles}}\, n_{\text{mesures}}$ itérations de type (5). Testez différentes valeurs de $n_{\text{boucles}} \in \{1, \ldots, 20\}$. Faites également varier le nombre de rayons, qui doit être impair, ainsi que le pas `delta_theta` entre deux angles θ successifs. Enfin, ne vous privez pas de tester d'autres images, à condition que celles-ci soient carrées, de taille paire.

Conseils de programmation :

- La fonction `kaczmarz` est très lente si l'on ne prend pas garde à calculer **en dehors de la double boucle** les normes au carré et la transposée de la matrice $\mathbf{W}$!
- Évitez d'utiliser la fonction `norm` pour calculer la norme euclidienne.

Exercice 2 : résolution de la tomographie par rétroprojection

Le principal problème de l'approche précédente est sa lenteur. Une autre manière de retrouver une image $f(x, y)$ à partir de son sinogramme consiste à « déprojeter » les données, ce qui revient à calculer, en chaque point (x, y) de l'image, la somme des contributions des différentes déprojections (voir cours) :

$$f(x, y) \approx \frac{1}{n_\theta} \sum_\theta p_\theta(x\cos\theta + y\sin\theta) \tag{6}$$

En pratique, pour chaque angle θ, et pour chaque pixel de l'image, il est nécessaire de chercher la valeur de l'abscisse t qui correspond au rayon $L(\theta, t)$ passant au plus près de ce pixel.

Écrivez la fonction `retroprojection`, appelée par `exercice_2`, permettant de résoudre le problème ainsi.

Conseils de programmation :

- L'abscisse x correspond au numéro de colonne, l'ordonnée y à l'opposé du numéro de ligne.
- Attention à bien traduire chaque angle en radians, avant de calculer son sinus et son cosinus !
- Veillez à bien positionner l'origine de la coordonnée t.

Ce script fournit des résultats flous, car les hautes fréquences sont bien moins représentées dans le sinogramme que les basses fréquences (voir cours). Une idée pour améliorer les résultats consiste à appliquer un filtrage aux différentes colonnes du sinogramme, de manière à rétablir l'équilibre entre basses et hautes fréquences. Un tel filtrage peut être effectué dans le domaine de Fourier (cf. TP8). C'est généralement le filtre de Ram-Lak qui est utilisé en tomographie. Ce filtre 1D est tout simplement égal à la valeur absolue de la fréquence.

Faites une copie du script `exercice_2`, de nom `exercice_2_bis`, où vous commencerez par appeler une fonction de nom `filtrage_sinogramme`, qu'il vous est demandé d'écrire, avant d'appliquer la rétroprojection. Il est conseillé d'utiliser la fonction `linspace` de Matlab.

Exercice 3 : utilisation du théorème du profil central (facultatif)

Il semble possible d'utiliser directement le *théorème du profil central*, qui fournit un échantillonnage de la transformée de Fourier de la fonction f recherchée. La difficulté vient de ce que cet échantillonnage n'est pas régulier. En effet, les échantillons fournis par une colonne $p_\theta(\cdot)$ du sinogramme se situent sur une droite passant par l'origine du plan de Fourier, d'angle polaire θ, ce qui donne à l'ensemble des échantillons l'allure d'une cible de fléchettes (voir cours).

L'idée la plus naturelle pour calculer la transformée de Fourier selon une grille régulière de mêmes dimensions que l'image, qui seule permettra de calculer f par transformation de Fourier inverse, est celle de l'*interpolation*. Les méthodes d'interpolation les plus connues sont l'interpolation « au plus proche voisin », l'interpolation bilinéaire et l'interpolation bicubique. Nous vous suggérons ici d'utiliser la méthode d'interpolation par « spline de type plaque mince » (*thin plate spline*), qui consiste à déformer le moins possible une plaque de métal passant par un ensemble de points 3D. La fonction `tpaps` de Matlab effectue une telle interpolation.

Remarques :

- La fonction à interpoler, en l'occurrence $\mathrm{TF}\{f\}$, est une fonction à valeurs complexes.
- Vos enseignants n'ayant pas eu le temps de tester cette approche, ils ne peuvent pas vous garantir qu'elle fonctionnera correctement...