Introduction générale : La science des données repose sur l'analyse, l'interprétation et l'exploitation de données massives pour extraire des connaissances exploitables et des insignes significatifs afin de prendre de bonnes décision. Dans cette quête, l'optimisation numérique est un élément essentiel de la science des données, fournissant des méthodes et des outils puissants pour résoudre une variété de problèmes tels que : modélisation et ajustement des données, sélection de modèle et régularisation des modèles pour une bonne sur-apprentissage, optimisation d'efficacité et de rentabilité des processus, analyse prédictive et prévisions des comportements et des tendances, ...
Une compréhension approfondie des techniques d'optimisation numérique est indispensable pour exploiter pleinement le potentiel des données et pour prendre des décisions éclairées dans un large éventail de domaines d'application : traitement et segmentation d'image,
Pour résoudre ces problèmes d'optimisation en science des données, une variété de techniques d'optimisation numérique sont utilisées notamment : Méthodes de Gradient à pas fixe (GPF), de Gradient à pas optimal (GPO), de Gradient Conjugué (CG), Descente de gradient et méthodes associées, Méthodes de Newton et Quasi-Newton, programmation linéaire et non linéaire pour des problèmes avec contraintes, ...
Dans ce premier travail pratique, nous allons nous pencher sur l'étude deux méthodes d'optimisation couramment utilisées : la méthode du gradient à pas optimal et la méthode du gradient conjugué. Nous examinerons leur convergence en utilisant une fonction quadratique spécifique définie comme suit :

$$J(x) = \frac{1}{2}x^T Ax - b^T x + c$$

où

- x est le vecteur des variables que nous cherchons à optimiser
- A est une matrice symétrique définie positive décrivant une situation en traitement d'image, telle que la matrice de convolution pour le débruitage d'une image.
- b est vecteur représentant le biais correcteur dans le processus de débruitage de l'image.
- c est une constante qui peut être interprétée comme un terme de régularisation ou un biais global correcteur de l'image.

Objectifs du TP : le but de ce travail est d'étudier comment les méthodes d'optimisation choisies convergent vers le minimum de cette fonction quadratique, en tenant compte du contexte spécifique du traitement d'image. Nous analyserons leur performance en termes de vitesse de convergence et de précision et discuter du préconditionnement.
Références utilisées :

- Jason Brownlee. Optimization for Machine Learning
- Stephen J. Wright, Benjamin Recht, Optimization for Data Analysis, 2022.
- Rachid Chelouah, Patrick Siarry. Optimization and Machine Learning : Optimization for Machine Learning and Machine Learning for Optimization, 2022.
- Laurent Rouvière, Pauline Tan. Cours de Master 2 en Machine Learning, 2019-2020.

Exercice n°1 Méthodes de Gradient (GPO et GC)

Pour répondre à notre problématique du débruitage d'image, par un bruit implusionnel, on utilise les deux méthodes GPO et GC pour minimiser le problème ci-dessous :

$$\min_{(x)\in\mathbb{R}^N} J(x) = \frac{1}{2}x^T Ax - b^T x$$

où

— A est une matrice creuse, décrivant le bruitage d'une image par le bruit impulsionnel, définie par $A_{i,i} = 2$ et $A_{i-1,i} = A_{i+1,i} = -1$. Le vecteur $b \in \mathbb{R}^N$ est donné par $b_i = i$, $i = 1, \cdots, N$.

1. Montrer que $\forall x \in R^N : \lambda_1 ||x||^2 \leqslant x^T A x \leqslant \lambda_N ||x||^2$ où λ_1 et λ_N désignent respectivement la plus petite valeur et la plus grande valeur propre de A.
2. Pour tout $x \in \mathbb{R}^N$, calculer le gradient de J noté par $\nabla J(x)$. Définir deux fonctions Python **def J(x)** et **def gradJ(x)** qui évaluent respectivement $J(x)$ et $\nabla J(x)$ en s'assurant que ces fonctions puissent être évaluées de manière générale sur des listes de N points $(x_1, x_2, \cdots, x_N)$ afin de l'utiliser pour le projet final.
3. Tracer les lignes de niveau et le graphe de la fonction J.
4. Implémenter la méthode de Gradient à pas Optimal et de Gradient Conjugué.
5. Représenter sur le même graphique la suite de points $(x^n)_{n\in\mathbb{N}}$ et les directions du gradient $\frac{\nabla J(x^n)}{||\nabla J(x^n)||}$.
6. Pour montrer la convergence de $J(x^n)$ vers $\min_{x\in\mathbb{R}^N} J(x)$, représenter les courbes $(J(x^n))_{n\in\mathbb{N}}$.
7. Pour montrer la convergence $(||x^n||)$ vers la solution du problème, représenter les courbes $(||x^n||)_{n\in\mathbb{N}}$.
8. Pour tout $x \in \mathbb{R}^N$, calculer la matrice Hessienne de J notée par $HessJ(x)$. Définir une fonction Python **def** $HessJ(x)$ en s'assurant que ces fonctions puissent être évaluées de manière générale sur des listes de N points $(x_1, x_2, \cdots, x_N)$ afin de l'utiliser pour le projet final.
9. Afin d'exploiter la matrice creuse A et diminuer le temps de calcul, faire tourner les deux méthodes pour $N \in \{10; 20; 30; 40; 50\}$, en utilisant le critère d'arrêt du type : $n \leqslant 2 \times 10^4$ ou $||Ax^n - b|| \leqslant 10^{-5}$.
10. Représenter graphiquement les points $\log(||Ax^n - b)||)$ en fonction de n pour les deux méthodes et pour des valeurs de $N = 50$ et $N = 100$. Commenter le résultat.
11. Reporter dans un tableau les temps de calcul des deux méthodes afin d'obtenir $||Ax^n - b|| \leqslant 10^{-5}$ avec $N = 50$ et $N = 100$, ainsi que les nombres d'itérations nécessaires. Commenter le tableau.

Exercice n°2 Préconditionnement d'une matrice

Le préconditionnement d'une matrice est une technique utilisée dans l'optimisation numérique et d'autres domaines des mathématiques computationnelles pour améliorer la convergence des méthodes itératives, telles que les méthodes de résolution de systèmes linéaires ou les méthodes d'optimisation. L'idée est de modifier la matrice initiale du problème d'optimisation pour le rendre mieux conditionné ou plus favorable à la convergence des méthodes itératives. En d'autres termes, le préconditionnement vise à équilibrer les échelles de différentes variables ou à atténuer les effets indésirables de la géométrie sous-jacente du problème.

Pour avoir une idée sur le préconditionnement, on s'intéresse au problème de minimisation ci-dessous :

$$\min_{x\in\mathbb{R}^N} J(x) = \frac{1}{2} x^T A x - b^T x \tag{1}$$

où b est donné dans l'exercice 1, et $A \in \mathcal{M}(\mathbb{R})$ est une matrice tridiagonale symétrique définie par $A_{i,i} = 3i^2, i = 1, \ldots, N$ et $A_{i-1,i} = A_{i,i+1} = -1, i = 1, \ldots, N-1$.

1. Pour $N = 10^3$, reporter le nombre d'itérations nécessaires pour avoir $\|Ax^n - b\|_2 \leqslant 10^{-10}$ avec les méthodes GPO et GC pour $n \leqslant 2 \times 10^3$.

2. On souhaite diminuer le nombre d'itérations, et donc le coût des deux méthodes implémentées dans l'exercice 1. Pour se faire, on considère la matrice diagonale $D \in \mathcal{M}_N(\mathbb{R})$ telle que $D_{i,i} = \frac{1}{i}$ pour $i = 1, \cdots, N$ et on définit la nouvelle matrice $\widetilde{A} = DAD$. Vérifier que le problème de minimisation (1) de J sur $\mathbb{R}^N$ s'écrit sous la forme

$$\min_{x \in \mathbb{R}^N} \tilde{J}(x) = \frac{1}{2} x^T \widetilde{A} x - \widetilde{b}^T x$$

où $\widetilde{b}$ est un vecteur de $\mathbb{R}^N$ à identifier.

3. Reporter le nombre d'itérations nécessaires pour le nouveau problème associé à $\tilde{J}$ et le comparer à celui pour J.

4. Pour comprendre théoriquement et numériquement les résultats observés, on définit le conditionnement de A par

$$\kappa_A = \frac{\lambda_N(A)}{\lambda_1(A)}$$

où $\lambda_1(A)$ et $\lambda_N(A)$ sont respectivement les plus petite et plus grande valeurs propres de A.
Indication : On pourra voir en TD que lorsque κ_A est grand, la vitesse de convergence est proportionnelle à $\frac{1}{\kappa_A}$ pour les algorithmes GPF et GPO, et à $\frac{1}{\sqrt{\kappa_A}}$ pour l'algorithme GC.
   (a) Par des encadrements simples de λ_1 et λ_N, démontrer théoriquement et numériquement que $\kappa_A \geqslant N^2$.
   (b) Démontrer et vérifier numériquement $\kappa_{\widetilde{A}}$ reste borné lorsque N varie.
   Indication : on pourra calculer directement $\widetilde{A}$ puis remarquer que $\|\widetilde{A} - 3I_N\|_\infty \leqslant 1$.
   (c) Commenter le résultat.

1 Annexe : Algorithmes des méthode GPF, GPO et GC

— Méthode du Gradient à Pas Fixe (GPF)

(GPF)
On choisit x^0 un vecteur de $\mathbb{R}^N$ et $\eta > 0$ un pas fixe.
Itérer pour $n \geqslant 0$:
$x^{n+1} = x^n + \eta \varepsilon^n$
$\varepsilon^n = b - Ax^n$

— Méthode du Gradient À Pas Optimal

(GPO)
On choisit x^0 un vecteur de $\mathbb{R}^N$ et $\eta > 0$ un pas fixe.
Itérer pour $n \geqslant 0$:
$x^{n+1} = x^n + \eta_n \varepsilon^n$
avec $\eta_n = \frac{<\varepsilon^n, \varepsilon^n>}{<A\varepsilon^n, \varepsilon^n>}$ et $\varepsilon^n = b - Ax^n$

— Algorithme du Gradient Conjugué

(GC)
On choisit x^0 un vecteur de $\mathbb{R}^N$ et on pose $r^0 = Ax^0 - b > 0$, $\varepsilon^0 = -r^0$
Itérer pour $n \geqslant 0$:
$x^{n+1} = x^n + \eta_n \varepsilon^n, \quad \eta_n = \frac{\|r^n\|^2}{<A\varepsilon^n, \varepsilon^n>},$
$r^{n+1} = Ax^{n+1} - b,$
$\varepsilon^{n+1} = -r^{n+1} + \gamma_n \varepsilon^n, \quad \gamma_n = \frac{\left\|r^{n+1}\right\|^2}{\|r^n\|^2}$