# Rapport de Projet

Elaboré par

**KCHAOU Amira**

# Projet du cours "Méthodes Numériques en Optimisation"

Responsable :

**Julien Ah-Pine**

Réalisé au sein de

SIGMA Clermont

Mastère Spécialisé Expert en Science des Données


Année Universitaire 2024 - 2025

# Table des matières

# Table des figures

# Introduction générale

L'apprentissage supervisé repose sur le développement de modèles capables de prédire une variable cible à partir de données d'entraînement. Parmi les modèles de régression linéaire, la **régression Lasso** (*Least Absolute Shrinkage and Selection Operator*) s'est imposée comme une technique essentielle grâce à sa capacité à effectuer simultanément une sélection de variables et une régularisation. En contraignant les coefficients du modèle par une pénalisation $L_1$, le Lasso favorise la parcimonie et permet d'améliorer la généralisation des prédictions.

Cependant, l'estimation des paramètres du modèle Lasso est un problème d'optimisation non trivial en raison de la non-différentiabilité de la norme $L_1$. Pour surmonter ces défis, plusieurs algorithmes d'optimisation numérique ont été développés, notamment :

- **ISTA (Iterative Shrinkage-Thresholding Algorithm)** : Une approche simple basée sur la descente de gradient avec seuillage doux.
- **FISTA (Fast Iterative Shrinkage-Thresholding Algorithm)** : Une version améliorée d'ISTA, introduisant une accélération inspirée des méthodes de Nesterov.
- **CGDA (Coordinate Gradient Descent Algorithm)** : Une méthode qui met à jour les coefficients un par un, permettant une optimisation plus rapide dans certains cas.

Ce projet vise à explorer ces différentes méthodes d'optimisation en trois étapes :

- **Partie Théorique** : Présentation du modèle Lasso, des défis liés à l'estimation de ses paramètres et des concepts mathématiques sous-jacents.

— **Partie Algorithmique** : Étude approfondie des algorithmes ISTA, FISTA et CGDA, accompagnée de leur pseudo-code.

— **Partie Empirique** : Expérimentation sur le jeu de données *Communities and Crime*, en comparant les performances des algorithmes en termes de vitesse de convergence et de qualité de prédiction (mesurée par le RMSE).

L'objectif final est de mieux comprendre les avantages et limites de ces techniques et d'identifier l'algorithme le plus adapté à la résolution du problème Lasso dans un contexte réel.

*Chapitre 1*

# Partie théorique sur le modèle d'apprentissage

## 1.1 Introduction à la Régression Lasso

### 1.1.1 Définition et principe de la régression Lasso

La régression Lasso (*Least Absolute Shrinkage and Selection Operator*) est une méthode de régularisation introduite par **Robert Tibshirani en 1996**. Elle modifie la régression linéaire classique en ajoutant une **pénalité** $L_1$ sur la norme des coefficients, ce qui permet à la fois de **réduire la variance du modèle** et de **sélectionner automatiquement les variables les plus pertinentes**.

L'équation mathématique de Lasso est donnée par :

$$\hat{\beta} = \arg\min_{\beta} \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \tag{1.1}$$

où :

— $X \in R^{np}$ est la matrice des variables explicatives,

— $y \in R^n$ est le vecteur des réponses,

— $\beta \in R^p$ est le vecteur des coefficients à estimer,

— $\lambda$ est un **paramètre de régularisation** qui contrôle la pénalisation des coefficients.

### 1.1.2 Propriétés et effets de la régularisation Lasso

L'ajout de la pénalité $L_1$ dans Lasso entraîne deux effets principaux :

- **Réduction du surapprentissage (Overfitting)**

  La régression linéaire classique a tendance à **sur-apprendre** lorsque $p \gg n$, c'est-à-dire quand il y a plus de variables que d'observations. Lasso contraint les coefficients, empêchant ainsi le modèle de s'adapter trop fortement aux données d'entraînement.

- **Sélection automatique des variables (Feature Selection)**

  Contrairement à la *Ridge Regression* ($L_2$), Lasso force certains coefficients $\beta_j$ à **être exactement égaux à zéro**. Ainsi, certaines variables sont **totalement exclues** du modèle, ce qui simplifie l'interprétation et améliore la généralisation.

**Exemple :** Dans le domaine de la génomique, où l'on cherche à identifier des biomarqueurs parmi des milliers de gènes, Lasso est utilisé pour **sélectionner les gènes les plus pertinents** tout en évitant l'overfitting.

### 1.1.3 Utilité de Lasso en apprentissage supervisé

- **Modèles à haute dimensionnalité ($p \gg n$)**

  Lasso est particulièrement efficace lorsque le nombre de variables explicatives est **très grand** par rapport au nombre d'échantillons. Par exemple, en **finance**, il permet de sélectionner **les indicateurs macroéconomiques les plus influents** sur un prix d'actif.

- **Amélioration de l'interprétabilité des modèles**

  En supprimant les variables non pertinentes, le modèle devient **plus simple et lisible**. Cela permet d'**expliquer les décisions du modèle**, ce qui est

essentiel en **médecine, intelligence artificielle, finance, etc.**.

- **Gestion des données corrélées**

  Dans les bases de données où **les variables explicatives sont fortement corrélées**, Lasso a tendance à **sélectionner une seule variable représentative**, ce qui évite le problème de **multicolinéarité**.

**Exemple :** En **économie**, où plusieurs indicateurs sont souvent fortement liés, Lasso peut choisir la variable la plus significative.

Pour conclure, Lasso est une méthode de régularisation efficace pour la sélection de variables et l'amélioration des performances des modèles en apprentissage supervisé. Il est particulièrement utile dans les problèmes à haute dimensionnalité et dans les domaines nécessitant des modèles explicables et interprétables. Cependant, le choix du paramètre $\lambda$ joue un rôle crucial et doit être optimisé à l'aide de techniques comme la **validation croisée**.

## 1.2 Défis associés à l'inférence des paramètres du modèle Lasso

L'inférence des paramètres du modèle Lasso présente plusieurs défis en raison de la nature de sa fonction objective et des propriétés statistiques des estimateurs.

### 1.2.1 Problèmes liés à l'inférence des paramètres du Lasso

- **Non-différentiabilité de la pénalité $L_1$ :** La fonction objective du Lasso combine une perte quadratique avec une pénalité $L_1$ sur les coefficients. Cette pénalité $L_1$ n'est pas différentiable en zéro, ce qui complique l'application des méthodes d'optimisation classiques reposant sur la différentiabilité.

- **Sélection de variables et biais des estimateurs :** Bien que Lasso soit capable de sélectionner automatiquement les variables pertinentes en annulant certains coefficients, cette propriété introduit un biais dans les estimateurs. Les coefficients des variables sélectionnées peuvent être sous-estimés, affectant la performance prédictive du modèle.
- **Choix du paramètre de régularisation $\lambda$ :** La performance du modèle Lasso dépend fortement du choix du paramètre de régularisation $\lambda$.
  - Un $\lambda$ trop grand peut entraîner un modèle trop simple avec une sous-adaptation.
  - Un $\lambda$ trop petit peut conduire à un modèle trop complexe avec une sur-adaptation.

  La sélection optimale de $\lambda$ est donc cruciale et nécessite souvent des techniques telles que la validation croisée.

## 1.3 Algorithmes d'optimisation pour le Lasso

Pour surmonter ces défis, plusieurs algorithmes d'optimisation ont été développés et étudiés. Dans leur article *"A Survey of Numerical Algorithms that can Solve the Lasso Problems"*, Zhao et Huo (2023) passent en revue cinq algorithmes représentatifs. On va présenter que les trois premiers dans notre cas.

### 1.3.1 Méthodes de résolution du problème du Lasso

- **Iterative Shrinkage-Thresholding Algorithm (ISTA) :**
  - ISTA est une méthode itérative qui met à jour les coefficients en appliquant une opération de seuillage doux (*soft-thresholding*) après chaque étape de descente de gradient.
  - Bien que simple à implémenter, ISTA converge lentement, avec un taux de convergence de l'ordre de $O(1/k)$, où $k$ est le nombre d'itérations.

- **Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) :**
  - FISTA est une version accélérée d'ISTA qui utilise une combinaison linéaire des itérations précédentes pour accélérer la convergence.
  - Il atteint un taux de convergence de $O(1/k^2)$, le rendant plus efficace que ISTA, surtout pour de grands ensembles de données.
- **Coordinate Gradient Descent Algorithm (CGDA) :**
  - CGDA met à jour les coefficients un par un en optimisant la fonction objective par rapport à une seule variable à la fois, tout en maintenant les autres constantes.
  - Cette approche est efficace pour les problèmes de grande dimension et permet une convergence stable.

Chaque algorithme présente des avantages et des inconvénients en termes de complexité computationnelle, de taux de convergence et de facilité d'implémentation. Le choix de l'algorithme approprié dépend souvent de la taille et de la nature des données, ainsi que des ressources computationnelles disponibles.

## 1.4 Définitions et Concepts Mathématiques

L'optimisation du problème de régression Lasso repose sur plusieurs concepts mathématiques fondamentaux, qui permettent de comprendre et d'analyser les algorithmes ISTA, FISTA et CGDA. Ces notions incluent la convexité, la régularité des gradients, les méthodes d'optimisation de premier ordre et la proximalité.

### 1.4.1 Fonction Convexe et Convexité Forte

— **Convexité :** Une fonction $f : R^n \to R$ est dite **convexe** si, pour tout $x_1, x_2 \in R^n$ et tout $\alpha \in [0, 1]$, on a :

$$f(\alpha x_1 + (1 - \alpha)x_2) \leq \alpha f(x_1) + (1 - \alpha)f(x_2). \tag{1.2}$$

Cela signifie que la courbe reliant deux points quelconques du graphe de $f$ ne descend jamais en dessous de $f$.

— **Convexité Forte :** Une fonction $f$ est dite **fortement convexe** avec un paramètre $\mu > 0$ si, pour tout $x_1, x_2 \in R^n$ :

$$f(x_2) \geq f(x_1) + \nabla f(x_1)^T(x_2 - x_1) + \frac{\mu}{2}\|x_2 - x_1\|_2^2. \tag{1.3}$$

Cette propriété assure une croissance quadratique autour du minimum de la fonction, garantissant une meilleure convergence des algorithmes.

### 1.4.2 Gradient Lipschitzien

— Une fonction différentiable $f$ possède un **gradient Lipschitzien** avec une constante $L > 0$ si, pour tout $x_1, x_2 \in R^n$, on a :

$$\|\nabla f(x_1) - \nabla f(x_2)\|_2 \leq L\|x_1 - x_2\|_2. \tag{1.4}$$

Cette condition garantit que le gradient ne varie pas trop rapidement, ce qui est crucial pour la convergence des méthodes d'optimisation basées sur le gradient.

### 1.4.3 Algorithmes de Premier et de Second Ordre

— **Algorithmes de Premier Ordre :** Ces algorithmes utilisent uniquement les informations du gradient (première dérivée) de la fonction objectif pour effectuer des mises à jour. Exemples : ISTA, FISTA et CGDA.

— **Algorithmes de Second Ordre :** Ces méthodes exploitent également la Hessienne (seconde dérivée) de la fonction objectif, permettant une meilleure convergence, mais avec un coût computationnel plus élevé.

### 1.4.4 Opérateur Proximal et Fonction de Seuil

— **Opérateur Proximal :** Dans les algorithmes ISTA et FISTA, la régularisation $L_1$ est gérée par l'opérateur proximal, défini comme suit pour une fonction $g$ :

$$\text{prox}_{\lambda g}(x) = \arg\min_z \left( \frac{1}{2}\|x - z\|_2^2 + \lambda g(z) \right). \tag{1.5}$$

— **Fonction de Seuil (Soft-Thresholding) :** Dans le cas où $g(x) = \|x\|_1$, l'opérateur proximal correspond à la fonction de seuillage doux :

$$S(x, \alpha) = \begin{cases} x - \alpha, & \text{si } x \geq \alpha, \\ x + \alpha, & \text{si } x \leq -\alpha, \\ 0, & \text{sinon.} \end{cases} \tag{1.6}$$

Cette fonction est essentielle pour induire la parcimonie dans les solutions des problèmes de régression Lasso.

### 1.4.5 Convergence et Taux de Convergence

— **Taux de Convergence :** Le taux de convergence mesure la rapidité avec laquelle un algorithme approche la solution optimale :

— **ISTA :** Convergence en $O(1/k)$.

— **FISTA :** Convergence accélérée en $O(1/k^2)$.

— **CGDA :** Convergence similaire à ISTA en $O(1/k)$, mais mise à jour par coordonnées.

Les algorithmes ISTA, FISTA et CGDA reposent sur des principes fondamentaux en optimisation convexe et en analyse numérique, tels que la convexité, la différentiabilité Lipschitzienne et la méthode de seuil proximal. Grâce à ces propriétés, ils permettent de résoudre efficacement le problème du Lasso tout en garantissant une certaine parcimonie des coefficients.

*Chapitre 2*

# Partie algorithmique sur les méthodes d'optimisation

Pour résoudre le problème de régression Lasso, plusieurs algorithmes d'optimisation numérique ont été développés. Parmi eux, l'Iterative Shrinkage-Thresholding Algorithm (ISTA), le Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) et le Coordinate Gradient Descent Algorithm (CGDA) sont particulièrement notables. Chacun de ces algorithmes présente des fondements théoriques spécifiques et des mécanismes de mise à jour distincts. Nous présenterons ci-dessous les principes fondamentaux de ces méthodes

## 2.1 Iterative Shrinkage-Thresholding Algorithm (ISTA)

L'Iterative Shrinkage-Thresholding Algorithm (ISTA) est une méthode de premier ordre largement utilisée pour résoudre le problème de régression Lasso, qui vise à minimiser la fonction objectif suivante :

$$F(\beta) = \frac{1}{2n}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1, \tag{2.1}$$

où $y \in R^n$ est le vecteur des réponses, $X \in R^{np}$ est la matrice des prédicteurs, $\beta \in R^p$ est le vecteur des coefficients à estimer, et $\lambda > 0$ est un paramètre de régularisation contrôlant la parcimonie du modèle.

### 2.1.1 Fondements Théoriques

ISTA est basé sur la méthode du gradient proximal, adaptée aux fonctions objectifs composées de deux termes :

— **Terme différentiable convexe** avec un gradient lipschitzien :

$$f(\beta) = \frac{1}{2n}\|y - X\beta\|_2^2$$

— **Terme convexe potentiellement non différentiable** :

$$g(\beta) = \lambda\|\beta\|_1$$

À chaque itération, ISTA effectue les étapes suivantes :

— **Calcul du gradient :** Évaluer le gradient de $f$ en $\beta^{(k)}$ :

$$\nabla f(\beta^{(k)}) = -\frac{1}{n}X^T(y - X\beta^{(k)}) \tag{2.2}$$

— **Mise à jour des coefficients :** Mettre à jour $\beta$ en appliquant l'opérateur proximal associé à $g$ :

$$\beta^{(k+1)} = \text{prox}_{\frac{\lambda}{L}}\left(\beta^{(k)} - \frac{1}{L}\nabla f(\beta^{(k)})\right) \tag{2.3}$$

où $L$ est la constante de Lipschitz du gradient de $f$, souvent choisie comme la plus grande valeur propre de $\frac{1}{n}X^TX$.

— **Opérateur proximal pour $g$ :** Dans le cas de la norme $L_1$, l'opérateur proximal correspond à la fonction de seuillage doux (*soft-thresholding*) :

$$S_\theta(z_i) = \text{sign}(z_i)\max(|z_i| - \theta, 0) \tag{2.4}$$

Cette approche permet de gérer efficacement la non-différentiabilité introduite par le terme de régularisation $L_1$ et favorise la parcimonie en annulant certains coefficients.

### 2.1.2 Pseudo code de l'algorithme ISTA

La figure 2.1 présente l'algorithme ISTA.

**ALGORITHM 1** **Iterative shrinkage-thresholding algorithms (ISTA)**

**Input:** $y_{n\times 1}, X_{n\times p}$, $L = \sigma_{\max}(X'X/n)$
**Output:** $\beta^{(K)}$: an estimator of $\beta$ after $K$ iterations
**initialization:** $\beta^{(0)}$
**for** $k = 0, 1, \ldots, K$ **do**
  $\beta^{(k+1)} = S(\beta^{(k)} - \frac{1}{nL}(X'X\beta^{(k)} - X'y), \lambda/L)$

**FIGURE 2.1 :** Présentation de l'algorithme ISTA

### 2.1.3 Implémentation python de l'algorithme ISTA

La figure 2.2 présente l'implémentation python de l'algorithme ISTA.

```python
#seuillage
def soft_thresholding(x, theta):
    return np.sign(x) * np.maximum(np.abs(x) - theta, 0)

def ista(X, y, lambda_, L, num_iterations):

    n, p = X.shape
    beta = np.zeros(p)

    for _ in range(num_iterations):
        gradient = - (1 / n) * X.T @ (y - X @ beta)
        beta = soft_thresholding(beta - (1 / L) * gradient, lambda_ / L)

    return beta
```

**FIGURE 2.2 :** Implémentation python de l'algorithme ISTA

## 2.2 Fast Iterative Shrinkage Threshold Algorithm FISTA

L'algorithme **FISTA (Fast Iterative Shrinkage-Thresholding Algorithm)**, introduit par **Beck et Teboulle (2009)**, est une amélioration de **ISTA**, visant

à accélérer la convergence pour la résolution du problème de régression Lasso.

### 2.2.1 Différence entre ISTA et FISTA

— **ISTA** est une méthode de *premier ordre* qui utilise uniquement le gradient de la solution précédente.

— **FISTA** introduit un **terme de momentum**, utilisant les gradients des deux solutions précédentes pour accélérer la convergence.

Grâce à cette amélioration, FISTA atteint une **convergence en $O(1/k^2)$**, contre **$O(1/k)$** pour ISTA.

### 2.2.2 Formulation Mathématique

L'objectif de FISTA est de résoudre le problème Lasso suivant :

$$F(\beta) = f(\beta) + g(\beta), \tag{2.5}$$

où :

— $f(\beta) = \frac{1}{2n}\|y - X\beta\|_2^2$ est une fonction convexe différentiable avec une constante de Lipschitz $L$.

— $g(\beta) = \lambda\|\beta\|_1$ est une fonction convexe non différentiable (pénalité $L_1$).

### 2.2.3 Mise à jour des coefficients dans FISTA

— **Définition d'une variable auxiliaire $\alpha^{(k)}$ :**

$$\alpha^{(k)} = \beta^{(k-1)} + \frac{t_{k-1} - 1}{t_k}(\beta^{(k-1)} - \beta^{(k-2)}) \tag{2.6}$$

— **Minimisation quadratique avec régularisation $L_1$ :**

$$\beta^{(k+1)} = \arg\min_{\beta} \left( f(\alpha^{(k)}) + \langle \beta - \alpha^{(k)}, \nabla f(\alpha^{(k)}) \rangle + \frac{L}{2} \|\beta - \alpha^{(k)}\|_2^2 + \lambda \|\beta\|_1 \right) \tag{2.7}$$

— **Application de l'opérateur proximal (Soft-Thresholding) :**

$$\beta^{(k+1)} = S_{\lambda/L} \left( \alpha^{(k)} - \frac{1}{L} \nabla f(\alpha^{(k)}) \right) \tag{2.8}$$

où $S_{\lambda/L}$ est l'opérateur de seuillage doux :

$$S_\theta(z_i) = \text{sign}(z_i) \max(|z_i| - \theta, 0) \tag{2.9}$$

— **Mise à jour du paramètre $t_k$ pour l'accélération :**

$$t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2} \tag{2.10}$$

### 2.2.4 Analyse de la Convergence

L'article de Zhao et Huo (2023) démontre que le taux de convergence de FISTA est donné par :

$$F(\beta^{(k)}) - F(\beta^*) \leq \frac{2\sigma_{\max}(X^T X/n) \|\beta^{(0)} - \beta^*\|_2^2}{(k+1)^2}, \tag{2.11}$$

où $\sigma_{\max}(X^T X/n)$ est la plus grande valeur propre de $X^T X/n$, et $\beta^*$ est la solution optimale.

— **ISTA** converge en $O(1/k)$.

— **FISTA** converge en $O(1/k^2)$, ce qui signifie que FISTA atteint une meilleure solution en moins d'itérations.

### 2.2.5 Complexité Computationnelle

D'après Zhao et Huo, la complexité d'une itération de FISTA est :

$$O(p^2) \tag{2.12}$$

Bien que chaque itération de FISTA ait un coût similaire à ISTA, **le nombre d'itérations nécessaires est réduit**, ce qui rend FISTA plus efficace.

En conclusion, l'algorithme **FISTA** constitue une amélioration significative d'**ISTA**, en introduisant une accélération inspirée des méthodes de Nesterov. Grâce à son taux de convergence amélioré $O(1/k^2)$, FISTA permet d'obtenir des solutions précises en moins d'itérations qu'ISTA, tout en conservant la même complexité computationnelle par itération.

### 2.2.6 Pseudo code de l'algorithme FISTA

La figure 2.3 présente l'algorithme FISTA.

**ALGORITHM 2** **Fast iterative shrinkage-thresholding algorithms (FISTA)**

**Input:** $y_{n\times 1}, X_{n\times p}$, $L = \sigma_{\max}(X'X/n)$
**Output:** $\beta^{(K)}$: an estimator of $\beta$ after $K$ iterations

**initialization;**
$\beta^{(0)}$ $\alpha^{(1)} = \beta^{(0)}$, $t_1 = 1$
**for** $k = 1, \ldots, K$ **do**
  $\beta^{(k)} = S(\alpha^{(k)} - \frac{1}{nL}(X'X\alpha^{(k)} - X'y), \lambda/L)$
  $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
  $\alpha^{(k+1)} = \beta^{(k)} + \frac{t_k - 1}{t_{k+1}}(\beta^{(k)} - \beta^{(k-1)})$

**Figure 2.3 :** Présentation de l'algorithme FISTA

### 2.2.7 Implémentation python de l'algorithme FISTA

La figure 2.4 présente l'implémentation python de l'algorithme FISTA.

```python
#seuillage
def soft_thresholding(x, theta):
    return np.sign(x) * np.maximum(np.abs(x) - theta, 0)

def fista(X, y, lambda_, L, num_iterations):

    n, p = X.shape
    beta = np.zeros(p)
    alpha = np.copy(beta)
    t = 1

    for _ in range(num_iterations):

        gradient = - (1 / n) * X.T @ (y - X @ alpha)
        beta_new = soft_thresholding(alpha - (1 / L) * gradient, lambda_ / L) #mise a jour de beta
        t_new = (1 + np.sqrt(1 + 4 * t**2)) / 2 #mise a jour de t
        alpha = beta_new + ((t - 1) / t_new) * (beta_new - beta) #mise a jour de alpha
        beta, t = beta_new, t_new

    return beta
```

**Figure 2.4 :** Implémentation python de l'algorithme FISTA

## 2.3 Coordinate Gradient Descent Algorithm CGDA

L'algorithme **CGDA (Coordinate Gradient Descent Algorithm)** est une alternative aux méthodes globales comme **ISTA** et **FISTA** pour la résolution du problème de **régression Lasso**. Contrairement à ces méthodes, **CGDA met à jour un seul coefficient à la fois**, ce qui en fait une méthode de *descente de gradient par coordonnées*.

### 2.3.1 Principe de CGDA : mise à jour par coordonnées

ISTA et FISTA mettent à jour **tous les coefficients** $\beta_j$ simultanément à chaque itération. **CGDA adopte une approche différente** :

— À chaque itération, il **sélectionne un coefficient** $\beta_j$ et l'optimise individuellement.

— Il exploite l'**opérateur proximal** $L_1$ **(Soft-thresholding)** pour imposer la parcimonie.

— Il **répète cette procédure** pour **toutes les variables** avant de passer à l'itération suivante.

L'objectif est de minimiser la fonction de coût du **Lasso** :

$$F(\beta) = \frac{1}{2n}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1. \tag{2.13}$$

### 2.3.2 Mise à jour d'un seul coefficient dans CGDA

À chaque itération, CGDA sélectionne un index $j$ et met à jour $\beta_j$ en résolvant :

$$\frac{\partial}{\partial \beta_j} F(\beta) = 0. \tag{2.14}$$

En développant cette dérivée partielle, la mise à jour de $\beta_j^{(k+1)}$ devient :

$$\beta_j^{(k+1)} = S\left(y^T X e_j - \sum_{l \neq j} (X^T X)_{jl} \beta_l^{(k)}, n\lambda\right) / (X^T X)_{jj}, \tag{2.15}$$

où :

— $e_j$ est un vecteur de taille $p$, avec **1 en position** $j$ et 0 ailleurs.

— $S(\cdot)$ est l'**opérateur de seuillage doux (Soft-thresholding)**, défini par :

$$S(x, \theta) = \text{sign}(x) \max(|x| - \theta, 0). \tag{2.16}$$

— $(X^T X)_{jj}$ est la diagonale de la matrice de Gram $X^T X$.

### 2.3.3 Convergence et complexité de CGDA

L'article de **Zhao et Huo (2023)** démontre que CGDA possède un taux de convergence :

$$F(\beta^{(k)}) - F(\beta^*) \leq \frac{4\sigma_{\max}(X^T X/n)(1+p)\|\beta^{(0)} - \beta^*\|_2^2}{k + 8/p}. \tag{2.17}$$

### 2.3.4 Comparaison avec ISTA et FISTA

| Méthode | Type de mise à jour | Convergence | Complexité par itération |
|---|---|---|---|
| **ISTA** | Mise à jour globale de $\beta$ | $O(1/k)$ | $O(p^2)$ |
| **FISTA** | Mise à jour globale + Accélération | $O(1/k^2)$ (meilleure) | $O(p^2)$ |
| **CGDA** | Mise à jour par coordonnées | $O(1/k)$ | $O(p^2)$ |

**Tableau 2.1 :** Comparaison entre ISTA, FISTA et CGDA

**Observations** :

— ISTA et CGDA ont le même taux de convergence $O(1/k)$, mais CGDA est souvent plus rapide en pratique.

— FISTA reste plus efficace grâce à son taux de convergence $O(1/k^2)$.

— CGDA est particulièrement adapté aux problèmes de grande dimension où seule une fraction des coefficients a besoin d'être mise à jour.

En conclusion, l'algorithme **CGDA (Coordinate Gradient Descent Algorithm)** repose sur une mise à jour **séquentielle des coefficients**, contrairement à ISTA et FISTA qui les mettent à jour globalement.

### 2.3.5 Pseudo code de l'algorithme CGDA

La figure 2.5 présente l'algorithme CGDA.

**ALGORITHM 3** **Coordinate gradient descent algorithm (CGDA)**

**Input:** $y_{n\times 1}, X_{n\times p}, \lambda$
**Output:** $\beta^{(K)}$: an estimator of $\beta$ after $K$ iterations
**initialization:** $\beta^{(0)}$
**for** $k = 0, 1, \ldots, K$ **do**
  **for** $j = 1 \cdots p$ **do**
    $\beta_j^{(k+1)} = S\left(y'Xe_j - \sum_{l\neq j}(X'X)_{jl}\beta_k^{(k)}, n\lambda\right) \Big/ (X'X)_{jj}$

**Figure 2.5 :** Présentation de l'algorithme CGDA

### 2.3.6 Implémentation python de l'algorithme CGDA

La figure 2.6 présente l'implémentation python de l'algorithme CGDA.

```python
#seuillage
def soft_thresholding(x, theta):
    return np.sign(x) * np.maximum(np.abs(x) - theta, 0)

def cgda(X, y, lambda_, num_iterations):

    n, p = X.shape
    beta = np.zeros(p)


    XTX = X.T @ X
    XTy = X.T @ y

    for _ in range(num_iterations):
        for j in range(p):
            # mise à jour  de la coordonnée j
            residual = XTy[j] - np.sum(XTX[j, :] * beta) + XTX[j, j] * beta[j]
            beta[j] = soft_thresholding(residual, n * lambda_) / XTX[j, j]

    return beta
```

**Figure 2.6 :** Implémentation python de l'algorithme CGDA

*Chapitre 3*

# Partie empirique, étude de cas

## 3.1 Description du Jeu de Données et Tâche à Résoudre

### 3.1.1 Présentation du Jeu de Données

Le jeu de données *Communities and Crime* regroupe des informations socio-économiqu démographiques et criminelles provenant de plusieurs sources, notamment :

— Le recensement américain de 1990 (*US Census*).

— Le rapport fédéral sur la criminalité de 1995 (*FBI Uniform Crime Report*).

— L'enquête sur la gestion des forces de l'ordre (*LEMAS*).

Ce jeu de données comprend un total de 1994 observations et 128 variables, dont :

— 122 variables prédictives couvrant des aspects variés tels que la démographie, le revenu, l'éducation, et la répartition des forces de police.

— Une variable cible : `ViolentCrimesPerPop`, qui représente le taux de crimes violents par habitant dans une communauté.

### 3.1.2 Objectif de la Tâche

L'objectif de cette étude est d'utiliser des modèles de régression régularisée, en particulier la régression Lasso, pour :

1. Prédire le taux de criminalité violente par habitant à partir des caractéristiques socio-économiques et policières.
2. Analyser les variables les plus influentes pour comprendre les facteurs prédictifs du crime.
3. Comparer différentes méthodes d'optimisation (ISTA, FISTA, CGDA) en termes de performance et convergence.

### 3.1.3 Gestion des Valeurs Manquantes

Dans le jeu de données *Communities and Crime*, plusieurs variables présentent des valeurs manquantes en quantité significative. Une gestion efficace de ces valeurs est essentielle pour garantir la robustesse et la fiabilité des modèles d'apprentissage supervisé.

#### 3.1.3.1 Stratégie Adoptée

Afin de minimiser l'impact des valeurs manquantes, nous avons suivi la méthodologie suivante :

— **Suppression des variables contenant plus de 80% de valeurs manquantes** : Certaines variables, notamment celles liées aux forces de police, avaient un nombre excessif de valeurs manquantes. Nous avons donc décidé de les supprimer afin d'éviter qu'elles n'introduisent du bruit dans l'analyse.

— **Imputation des valeurs manquantes restantes par la médiane** : Pour les variables numériques contenant encore quelques valeurs manquantes, nous avons remplacé ces valeurs par la médiane de chaque colonne. Cette approche permet de conserver la distribution originale des données tout en évitant l'impact des valeurs extrêmes.

En conclusion, cette stratégie garantit que les données utilisées pour la

prédiction du taux de criminalité sont fiables, tout en optimisant la qualité des entrées pour nos modèles.

### 3.1.4 Analyse Préliminaire des Données

Une première analyse exploratoire a été réalisée, notamment l'étude de la distribution de la variable cible `ViolentCrimesPerPop`. L'histogramme ci-dessous illustre cette distribution :

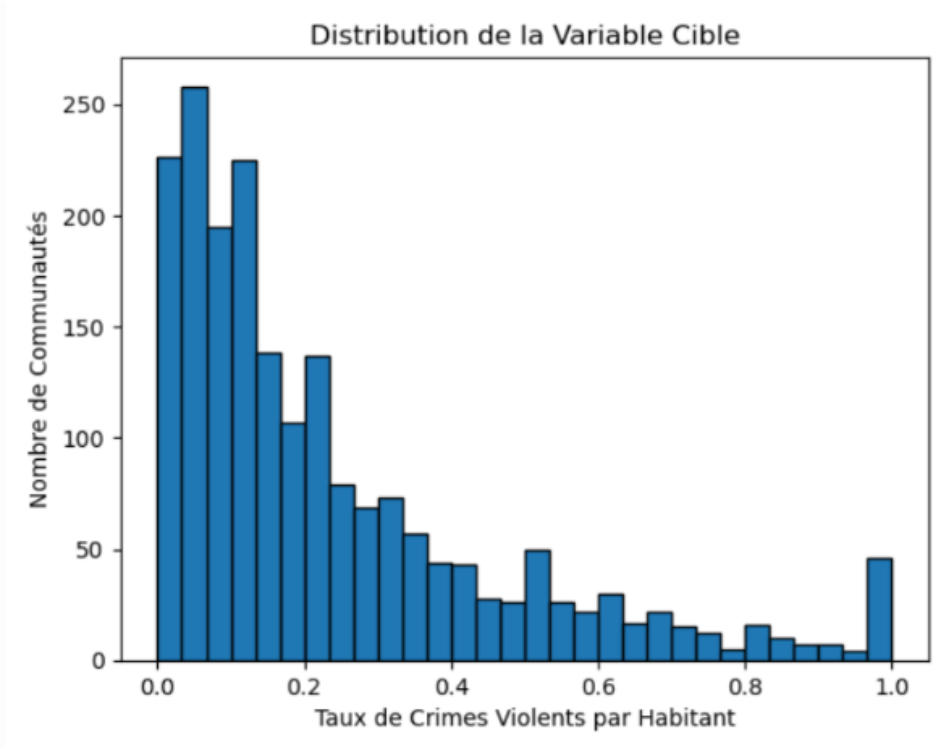


**Figure 3.1 :** Distribution de la Variable Cible `ViolentCrimesPerPop`

Observations :

— La distribution est fortement biaisée à droite, indiquant que la majorité des communautés ont un taux de criminalité relativement faible.

— Quelques communautés présentent des valeurs proches de 1, ce qui indique une criminalité violente très élevée.

## 3.2 Implémentation et Évaluation de l'Algorithme ISTA

### 3.2.1 Méthodologie

— Implémentation de l'algorithme ISTA en mettant à jour les coefficients à chaque itération via une descente de gradient et un seuillage doux.

— Évaluation de la convergence en mesurant l'évolution de l'erreur quadratique moyenne (MSE).

— Validation croisée avec différentes valeurs du paramètre de régularisation $\lambda$ (0.1, 1, 10) afin d'optimiser la sélection du modèle.

### 3.2.2 Analyse et Résultats

La convergence de l'algorithme a été analysée à travers l'évolution du MSE en fonction du nombre d'itérations. La Figure 3.5 illustre cette évolution :

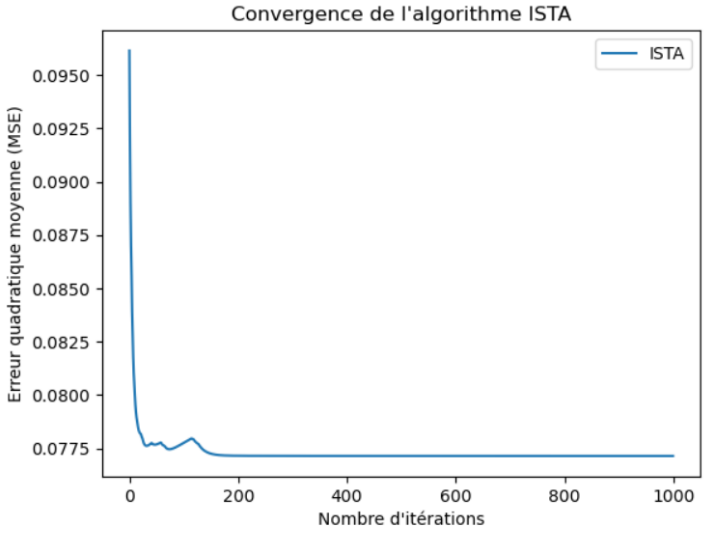


**Figure 3.2 :** Convergence de l'algorithme ISTA

Les performances de l'algorithme ont été évaluées en termes d'erreur quadratique moyenne (RMSE) pour différentes valeurs de $\lambda$, comme indiqué dans le tableau ci-dessous :

| **Valeur de $\lambda$** | **RMSE moyen (Validation croisée)** |
|---|---|
| 0.1 | 0.277 |
| 1 | 0.332 |
| 10 | 0.332 |

**Tableau 3.1 :** Résultats du RMSE moyen pour différentes valeurs de $\lambda$

### 3.2.3 Interprétation des résultats

L'analyse des résultats met en évidence plusieurs points :

— L'erreur RMSE décroît rapidement au début de l'apprentissage et se stabilise après environ 200 itérations, confirmant la convergence de l'algorithme.

— La valeur optimale du paramètre de régularisation est $\lambda = 0.1$, qui produit la meilleure généralisation avec un RMSE de 0.277.

— Lorsque $\lambda$ augmente (1 et 10), la régularisation devient trop forte, entraînant une sous-estimation des coefficients et une augmentation du RMSE.

## 3.3 Implémentation et Évaluation de l'Algorithme FISTA

### 3.3.1 Analyse des résultats de FISTA

#### 3.3.1.1 Convergence de l'algorithme

L'algorithme FISTA présente une convergence rapide comme illustré dans la Figure 3.6. On observe une stabilisation de l'erreur quadratique moyenne (MSE) après seulement quelques itérations. Contrairement à ISTA, FISTA utilise l'accélération de Nesterov, ce qui permet une descente plus rapide et réduit le nombre d'itérations nécessaires pour atteindre une convergence stable.

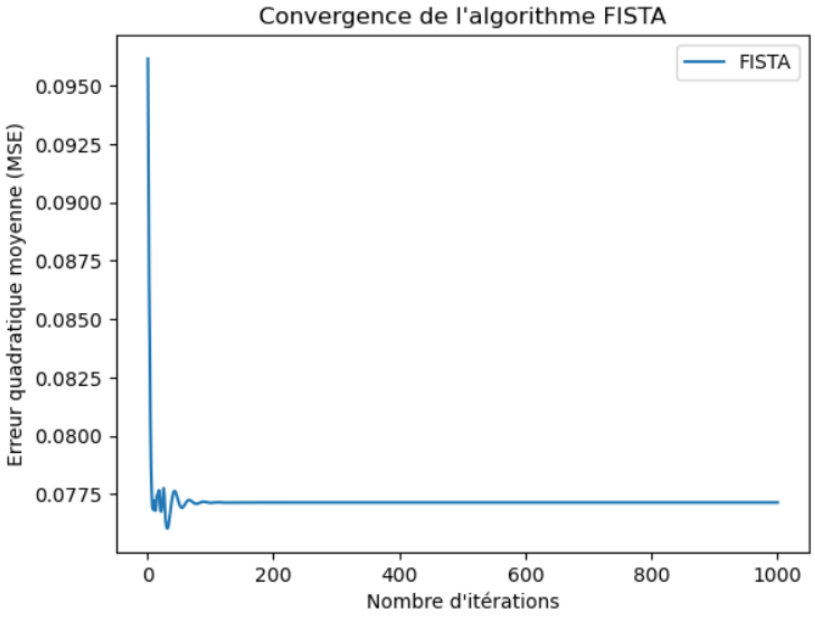


**Figure 3.3 :** Convergence de l'algorithme FISTA

#### 3.3.1.2 Analyse des performances pour différents $\lambda$

Nous avons testé trois valeurs de $\lambda$ pour analyser leur impact sur la performance du modèle. Les résultats en termes de Root Mean Squared Error (RMSE) moyen après validation croisée sont résumés dans le Tableau 3.2.

| **Valeur de $\lambda$** | **RMSE moyen (validation croisée)** |
|---|---|
| 0.1 | 0.277 |
| 1 | 0.332 |
| 10 | 0.332 |

**Tableau 3.2 :** Comparaison des performances de FISTA pour différents $\lambda$

### 3.3.2 Interprétation des résultats

— Pour $\lambda = 0.1$, le RMSE est le plus bas (0.277), ce qui montre une meilleure capacité de généralisation du modèle.

— Pour $\lambda = 1$ et $\lambda = 10$, le RMSE augmente à 0.332, indiquant une régularisation excessive qui réduit la performance du modèle.

### 3.3.3 Comparaison avec ISTA

— FISTA converge plus rapidement que ISTA, grâce à l'accélération de Nesterov.

— Les valeurs de RMSE sont similaires entre ISTA et FISTA, suggérant que FISTA améliore le temps de convergence sans impacter significativement la précision finale.

## 3.4 Implémentation et Évaluation de l'Algorithme CGDA

### 3.4.1 Convergence de l'algorithme

L'algorithme CGDA présente une convergence très rapide, comme illustré dans la Figure 3.7. On observe que l'erreur quadratique moyenne (MSE) diminue brusquement dès les premières itérations avant de se stabiliser. Cette rapidité s'explique par la mise à jour coordonnée par coordonnée des paramètres, permettant un ajustement plus fin dès le début de l'optimisation.

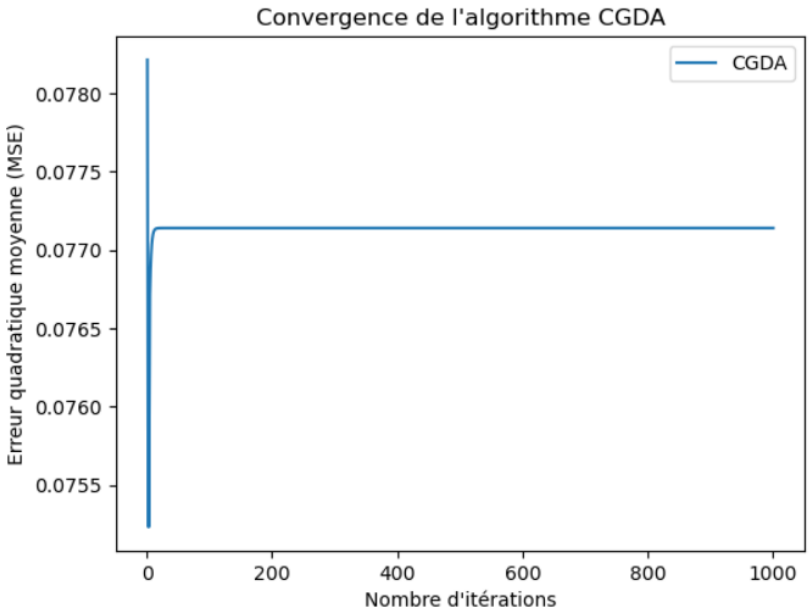


**Figure 3.4 :** Convergence de l'algorithme CGDA

### 3.4.2 Analyse des performances pour différents $\lambda$

Nous avons testé trois valeurs de $\lambda$ pour analyser leur impact sur la performance du modèle. Les résultats en termes de Root Mean Squared Error (RMSE) moyen après validation croisée sont résumés dans le Tableau 3.3.

| **Valeur de $\lambda$** | **RMSE moyen (validation croisée)** |
|---|---|
| 0.1 | 0.0773 |
| 1 | 0.1109 |
| 10 | 0.1109 |

**Tableau 3.3 :** Comparaison des performances de CGDA pour différents $\lambda$

### 3.4.3 Interprétation des résultats

— Pour $\lambda = 0.1$, le RMSE est le plus bas (0.0773), ce qui indique une meilleure généralisation du modèle.

— Pour $\lambda = 1$ et $\lambda = 10$, le RMSE augmente à 0.1109, suggérant une régularisation excessive réduisant la performance du modèle.

— Un $\lambda$ trop grand impose une forte pénalisation sur les coefficients du modèle, ce qui empêche le modèle d'apprendre efficacement des données.

## 3.5 Comparaison des Méthodes ISTA, FISTA et CGDA

### 3.5.1 Comparaison des Performances en Précision (RMSE)

Afin d'évaluer la capacité de généralisation de chaque méthode, nous avons comparé les performances des algorithmes ISTA, FISTA et CGDA à l'aide du Root Mean Squared Error (RMSE) moyen après validation croisée. Les résultats sont présentés dans le Tableau 3.4.

| **Valeur de $\lambda$** | **ISTA (RMSE)** | **FISTA (RMSE)** | **CGDA (RMSE)** |
|---|---|---|---|
| 0.1 | 0.2773 | 0.2773 | 0.0773 |
| 1 | 0.3324 | 0.3324 | 0.1109 |
| 10 | 0.3324 | 0.3324 | 0.1109 |

**Tableau 3.4 :** Comparaison des performances (RMSE) des méthodes ISTA, FISTA et CGDA

#### 3.5.1.1 Analyse des Résultats

— CGDA donne les meilleurs résultats en termes de RMSE pour toutes les valeurs de $\lambda$. Cela suggère que la mise à jour coordonnée des coefficients permet un meilleur ajustement aux données.

— ISTA et FISTA ont des performances similaires en termes de RMSE, avec une amélioration notable pour $\lambda = 0.1$.

— Un $\lambda$ plus élevé ($\lambda = 1$ **et** $\lambda = 10$) entraîne une régularisation excessive pour toutes les méthodes, augmentant ainsi l'erreur de prédiction.

### 3.5.2 Comparaison des Vitesses de Convergence

Nous avons analysé la convergence des algorithmes en mesurant la diminution de l'erreur quadratique moyenne (MSE) au fil des itérations. Les figures suivantes comparent la convergence des trois algorithmes.

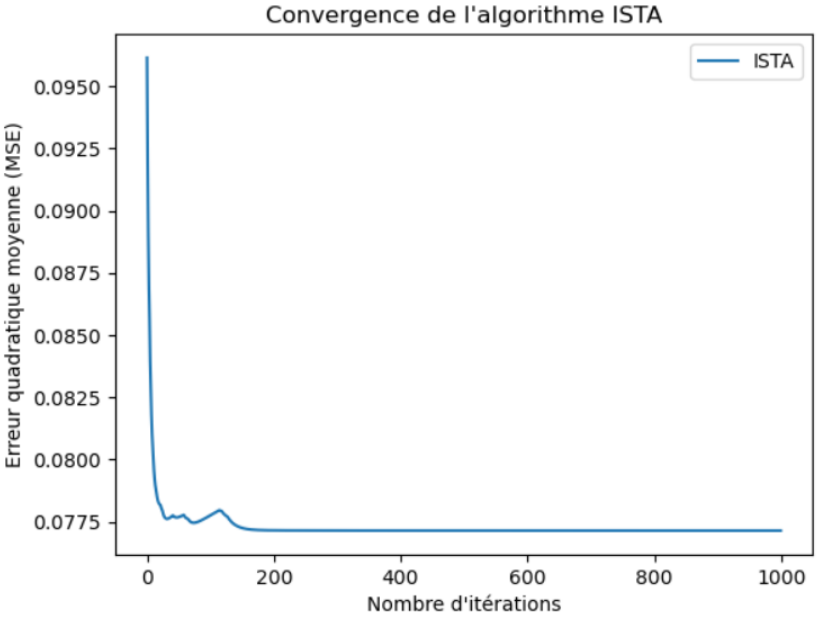


**Figure 3.5 :** Convergence de l'algorithme ISTA

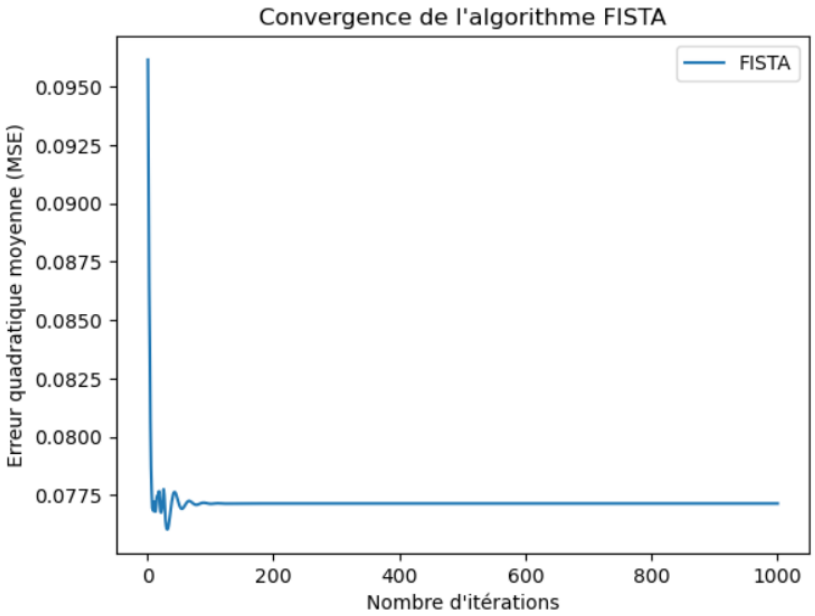


**Figure 3.6 :** Convergence de l'algorithme FISTA

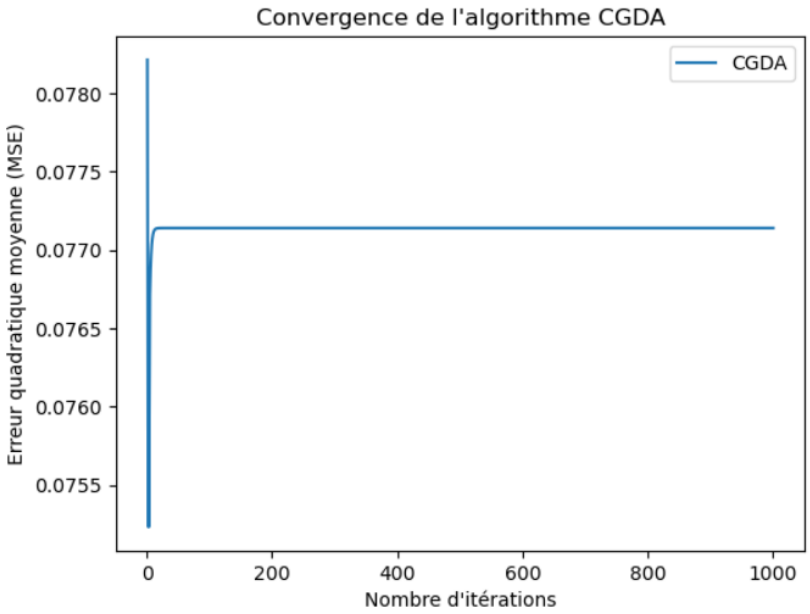


**Figure 3.7 :** Convergence de l'algorithme CGDA

#### 3.5.2.1 Analyse des Résultats

— ISTA converge plus lentement que les autres méthodes. L'absence d'accélération ralentit la mise à jour des coefficients, nécessitant plus d'itérations pour atteindre un plateau.

— FISTA améliore considérablement la vitesse de convergence grâce à l'optimisation de Nesterov. On observe une descente plus rapide dès les premières itérations.

— CGDA est l'algorithme le plus rapide à converger. Il atteint un plateau très rapidement, ce qui en fait une méthode efficace pour des problèmes de grande dimension.

### 3.5.3 Synthèse et Conclusion

— En termes de RMSE, CGDA est la méthode la plus performante, particulièrement pour $\lambda = 0.1$.

— En termes de convergence, FISTA et CGDA surpassent ISTA. CGDA converge le plus rapidement.

— ISTA reste une méthode simple et efficace, mais ses performances sont limitées par une convergence plus lente.

— FISTA est un compromis intéressant, combinant rapidité et précision, bien qu'il n'améliore pas significativement le RMSE final par rapport à ISTA.

— CGDA se distingue nettement en offrant la meilleure précision et la convergence la plus rapide, en raison de la mise à jour séquentielle des coefficients.

Ainsi, pour des applications où la rapidité de convergence est primordiale, CGDA est le choix optimal. Si l'on privilégie une méthode simple et bien comprise, ISTA reste une alternative viable. Enfin, FISTA est recommandé lorsque l'on souhaite un bon compromis entre rapidité et précision.

# Conclusion générale

Dans ce projet, nous avons comparé trois algorithmes d'optimisation pour la régression Lasso : **ISTA**, **FISTA** et **CGDA**. L'objectif était d'évaluer leur **vitesse de convergence** et leur **précision** sur le jeu de données *Communities and Crime*.

Les résultats montrent que :

— **ISTA** est simple mais converge lentement.

— **FISTA** accélère ISTA grâce à la méthode de Nesterov, réduisant le nombre d'itérations nécessaires.

— **CGDA** est le plus performant en termes de rapidité et de RMSE, grâce à sa mise à jour coordonnée des paramètres.

Ainsi, le choix de l'algorithme dépend des contraintes du problème : **FISTA** est un bon compromis rapidité/précision, tandis que **CGDA** est optimal pour des calculs plus efficaces.

Pour aller plus loin, une exploration d'autres valeurs de $\lambda$, l'application à des jeux de données plus vastes et la comparaison avec des variantes comme *Elastic Net* seraient des perspectives intéressantes.

## Bibliographie

— **Zhao, Y., & Huo, X. (2023).** *A Survey of Numerical Algorithms that can Solve the Lasso Problems.* arXiv preprint arXiv :2303.03576.

— **Beck, A., & Teboulle, M. (2009).** *A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems.* SIAM Journal on Imaging Sciences, 2(1), 183–202.

— **Friedman, J., Hastie, T., & Tibshirani, R. (2010).** *Regularization Paths for Generalized Linear Models via Coordinate Descent.* Journal of Statistical Software, 33(1), 1-22.

— **Robert Tibshirani (1996).** *Regression Shrinkage and Selection via the Lasso.* Journal of the Royal Statistical Society : Series B. Disponible sur : `https://web.stanford.edu/~hastie/Papers/LASSO_JRSSB.pdf`

— **Hastie, Tibshirani & Wainwright (2015).** *Statistical Learning with Sparsity : The Lasso and Generalizations.* Disponible sur : `https://web.stanford.edu/~hastie/StatLearnSparsity_files/SLS_corrected_1.4.16.pdf`

— **IBM (2023).** *Introduction to Lasso Regression.* Disponible sur : `https://www.ibm.com/fr-fr/topics/lasso-regression`