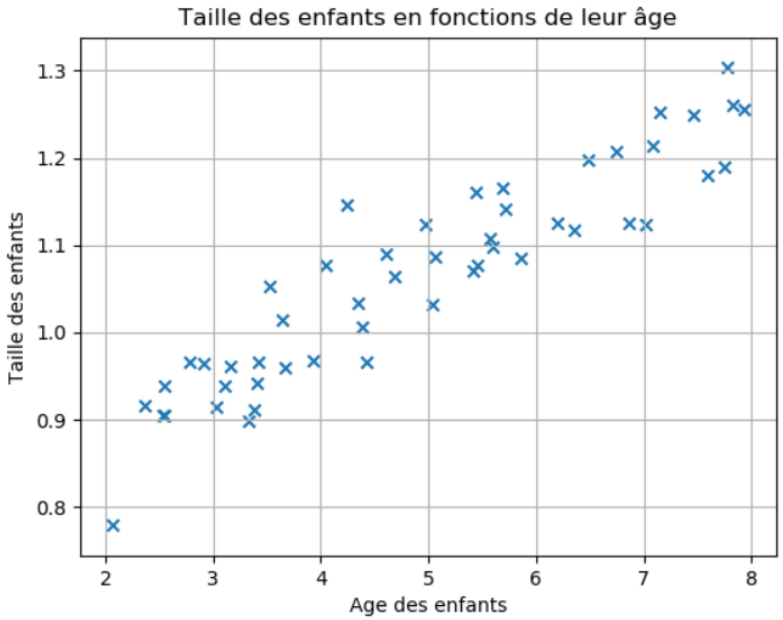
IPSA École d'ingénieurs

Optimisation Quadratique

Projet - Minimisation des fonctionnelles quadratiques par des méthodes à directions de descente



Steven Angélique
Louis Deschamps
Romain Dos Reis
Alessandra Paddeu

Professeur : M. BLETZACKER
M.CIRIL
Promo 2023

28 mars 2021

Table des matières

1 Introduction

Ce projet (accompagné d'une partie numérique traitée en Travaux Pratiques) poursuit deux objectifs :
Le premier consiste à étudier (au sens des mathématiques) un problème d'analyse de données (plus precisément un problème de calibrage ou identification de paramètres d'un modèle) posé au sens des moindres carrés. Cette étude fait l'objet de la section 2. Afin de résoudre numériquement le problème, nous allons considérer trois algorithmes à direction de descente : deux du type à direction de plus forte pente (l'un a pas fixe et l'autre à pas optimal), et le troisiéme basé sur les directions dites des gradients conjugués (algorithme des gradients conjugués).
Le second objectif consiste à présenter, étudier puis comparer théoriquement et numériquement ces trois algorithmes à directions de descentes. La présentation et l'étude numérique de ces trois méthodes feront l'objet de la section 3, tandis que la sections 4 sera consacrée à l'étude théorique (mathématiques) des méthodes. Nous pourrons ainsi mettre en relation les résultats numériques avec les développements théoriques.

2 Formulation et analyse mathématique

2.1 Identification aux sens des moindres carrés linéaires

La finalité de cette sous-section est d'expliciter le problème d'optimisation quadratique sans contrainte découlant de l'application de l'approche aux moindres carrés dans l'identification de paramètres d'un modèle (Linéaire vis-à-vis de ces paramètres *à déterminer*).

2.1.1 Principe général

Il s'agit d'un problème d'analyse de données. Supposons qu'au cours d'une expérience physique nous mesurons une grandeur ou quantité q qui dépend d'un paramètre p. Nous faisons m expériences et mesures différentes en faisant varier le paramètre m. Le problème consiste alors à savoir comment déterminer à partir de ses m mesures une loi exprimentale (plutôt simple) qui permettrait d'approcher *au mieux* (le *mieux* est à définir) la quantité étudier comme une fonction du paramètre p. La forme de cette loi expérimentale est imposée : dans le cas le plus courant, on considère une fonction $f(p;\,)$ d'une variable p, et dépendant de n paramètres inconnus $c_1, c_2, \dots c_n$ (constituant le vecteur). En général, le nombre de mesures m est très grand par rapport au nombre de n de paramètres. En d'autres termes, étant donné m valeurs $p_1, p_2, \dots, p_m$ du paramètre p (variable de la loi f) et m valeurs correspondantes de la grandeur mesuré $q_1, q_2, \dots, q_m$, on cherche à déterminer une fonction $f(p;^*\,)$ de la famille de fonction $f(p;\,)$ indéxée par les paramètres $c_1, c_2, \dots c_n$ qui minimise l'erreur commise entre la valeur expérimentale q_i et la valeur théorique $f(p_i;\,)$. On décide de mesurer l'erreur **au sens des moindres carrés**, *c.-à-d.* qu'on minimise la somme des carrés des erreurs individuelles, autrement dit la quantité

$$\forall (c_1, \dots c_n) \in^n, \quad E(c_1, \dots c_n) = \sum_{i=1}^{m} \left(q_i - f(p_i; c_1, \dots, c_n)\right)^2 . \tag{1}$$

Nous parlerons de **régression linéaire** dans le cas où la loi $f(p; c_1, \dots, c_n)$ dépend linéairement des n paramètres $c_1, c_2, \dots, c_n$, en d'autres termes on écrit $f(p; c_1, \dots, c_n)$ s'écrit dans une base $(\Phi_j)_{j=0}^{n}$ de fonctions (par exemple polynômes, fractions rationnelles, polynômes trigonométriques, exponentielle, etc.) par

$$f(p; c_1, \dots, c_n) = \sum_{j=1}^{n} c_j \Phi_j(p). \tag{2}$$

son extrême simplicité fait que cette méthode est très couramment utilisée de nos jours en sciences expérimentales. Une application courante est le lissage des données expérimentales par une fonction empirique (fonction linéaire, polynômes ou splines). Cependant, son usage le plus important est probablement la mesure de quantités physiques à partir de données expérimentales.

2.1.2 Ajustement linéraire

1. Grâce aux données fournies, nous pouvons construire un nuage de point qui illustre l'étude de ce projet :

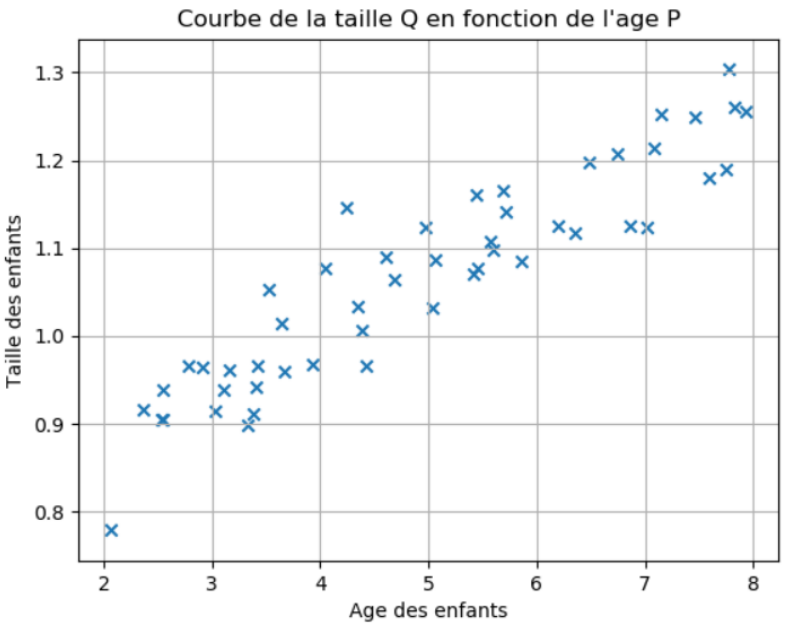


FIGURE 1 – Nuage de points

Nous observons que le problème peut se traduire par la modélisation d'une droite affine.

Nous obtenons alors le graphique suivant :

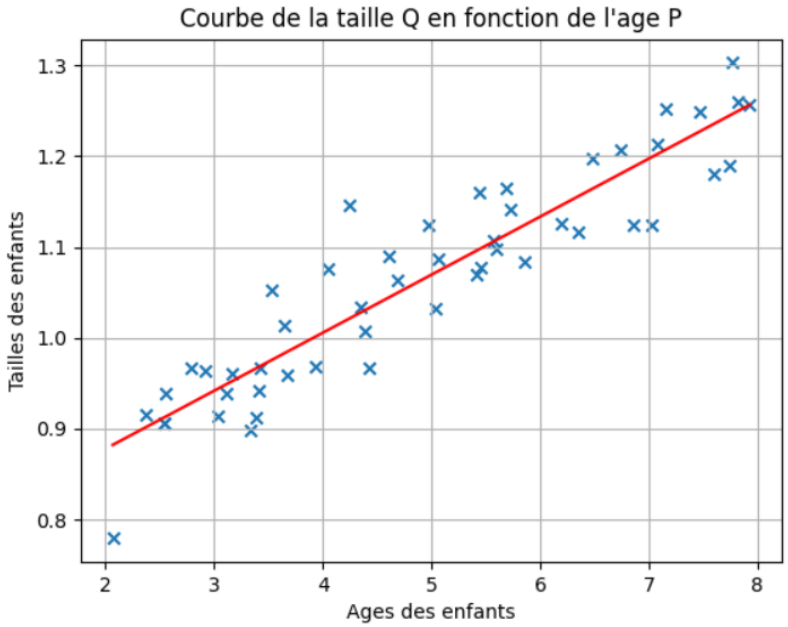


FIGURE 2 – Régression linéaire

A la vue de cette représentation graphique on peut tout à fait choisir un modèle de régression linéaire affine de la forme :

$$\forall p \in \mathbb{R}, f(p; c_1, c_2) = c_1 + c_2 p \tag{3}$$

Nous obtenons alors l'estimation suivante :

$$q_i = c_1 + c_2 p_i + r_i(c_1, c_2) \;\; 1 \leqslant i \leqslant m \tag{4}$$

Avec $r_i(c_1, c_2)$ les résidus du modèle dépendants des paramètres de régression.

2. D'après les équations (1) et (2) on peut donner une formulation du problème qui fait référence à la famille de fonctions affines (1) :

$$\begin{pmatrix} q_0 \\ \vdots \\ q_m \end{pmatrix} = \begin{pmatrix} f(p_0; c_1, c_2) \\ \vdots \\ f(p_n; c_1, c_2) \end{pmatrix} + \begin{pmatrix} r_0 \\ \vdots \\ r_m \end{pmatrix}$$

3. Réécrivons notre système d'équation (2) sous forme matricielle :

$$q = Xc + r \tag{5}$$

Avec :

$$q = \begin{pmatrix} q_1 \\ \vdots \\ q_m \end{pmatrix} \in \mathbb{R}^m \;,\; X = \begin{pmatrix} 1 & p_1 \\ \vdots & \vdots \\ 1 & p_m \end{pmatrix} \in \mathbb{R}^{m \times 2} \;,\; r = \begin{pmatrix} r_1 \\ \vdots \\ r_m \end{pmatrix} \in \mathbb{R}^m \;,\; c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \in \mathbb{R}^2$$

De cette facon :

$$q = Xc + r \iff \begin{pmatrix} q_1 \\ \vdots \\ q_m \end{pmatrix} = \begin{pmatrix} 1 & p_1 \\ \vdots & \vdots \\ 1 & p_m \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} + \begin{pmatrix} r_1 \\ \vdots \\ r_m \end{pmatrix} = \begin{pmatrix} c_1 + c_2 p_1 \\ \vdots \\ c_1 + c_2 p_m \end{pmatrix} + \begin{pmatrix} r_1 \\ \vdots \\ r_m \end{pmatrix}$$

On retrouve bien la forme (2) : $q_i = c_1 + c_2 p_i + r_i(c_1, c_2) \;\; (1 \leqslant i \leqslant m)$.

4. $\forall p \in \mathbb{R} \; f(p; c_1, c_2) = c_1 + c_2 p$ (3) $\iff f(p_i; c_1, c_2) = c_1 + c_2 p_i, i \in [0, m]$

Nous posons $r_i = 0$ car nous voulons résoudre le problème de minimisation sans contrainte :

Ainsi $q_i = f(p_i; c_1, c_2) = c_1 + c_2 p_i$

$$q_i = c_1 + c_2 p_i \iff \begin{pmatrix} q_1 \\ \vdots \\ q_m \end{pmatrix} = \begin{pmatrix} 1 & p_1 \\ \vdots & \vdots \\ 1 & p_m \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} pour \; i \in [0, m]$$

Donc résoudre (4) sans contrainte revient à résoudre $q = Xc \iff q - XC = 0$
Or résoudre $q - Xc = 0$ revient a trouver la solution de $F(c) = 0$ car $0.5 * \|q - Xc\|^2 = 0 \Longrightarrow q - XC = 0$.
Et trouver la solution de $F(c) = 0$ revient à minimiser $F(c)$.

Résoudre (4) sans contrainte revient à résoudre $q - XC = 0$, donc à trouver la solution de $F(c) = 0$ ce qui revient finalement à minimiser $F(c)$.

Soit

$$F(c) = 0.5 * \|q - Xc\|^2 = 0.5 * \|Xc - q\|^2$$
$$\Leftrightarrow F(c) = 0.5(Xc - q)^T * (Xc - q)$$

Car

$$\|A\|^2 = A^T A$$

Nous développons et nous obtenons

$$F(c) = 0.5 * (Xc)^T * Xc - 0.5(Xc)^T q - 0.5 q^T Xc + 0.5 q^T q$$

avec

$$0.5q^Tq = 0.5\left\|q\right\|^2$$

Comme c^TX^Tq est une matrice symétrique alors :

$$0.5 * (c^TX^Tq)$$
$$= 0.5 * (c^TX^Tq)^T$$
$$= 0.5 * q^TXc$$

Ainsi,

$$F(c) = 0.5 * c^TX^TXc - q^TXc + 0.5\left\|q\right\|^2$$

Avec $b = X^Tq$, $A = X^TX$ et $p = 0.5\left\|q\right\|^2$

Ainsi

$$\boxed{F(c) = 0.5c^TAc - b^Tc + p}$$

Donc nous avons bien $F(c)$ qui est une fonction quadratique sous forme matricielle.

5. Nous calculons :

$$X^TX = \begin{pmatrix} 1 & \cdots & 1 \\ p_1 & \cdots & p_m \end{pmatrix} \begin{pmatrix} 1 & p_1 \\ \vdots & \vdots \\ 1 & p_m \end{pmatrix} = \begin{pmatrix} m & \sum_{i=1}^m p_i \\ \sum_{i=1}^m p_i & ||p||^2 \end{pmatrix} = \begin{pmatrix} m & p^T1 \\ p^T1 & ||p||^2 \end{pmatrix}$$

Et

$$X^Tq = \begin{pmatrix} 1 & \cdots & 1 \\ p_1 & \cdots & p_m \end{pmatrix} \begin{pmatrix} q_1 \\ \vdots \\ q_m \end{pmatrix} = \begin{pmatrix} q^T1 \\ p^Tq \end{pmatrix}$$

6. Nous reprenons la définition d'une fonction scalaire quadratique.
Soit la fonction $f : \mathbb{R}^2 \to \mathbb{R}$. Elle s'écrit relativement à la base canonique de $\mathbb{R}^2$ sous la forme matricielle :

$$\forall x := \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathcal{M}_{N,1}(\mathbb{R},\ f(x) = \frac{1}{2}x^TAx - b^Tx + p$$

où :

- A est une matrice symétrique définie tel que :

$$\forall i \in [1, ..., N],\ \ a_ii := \alpha_{ii}\ et\ a_{ij} = a_{ji} := \frac{\alpha_{ij}}{2}$$

- b est une matrice uni-colonne
- p est une constante réelle

Donc la fonction F est bien quadratique, car par identification :

$$F(c) = \frac{1}{2}c^T(X^TX)c - (X^Tq)^Tc + \frac{1}{2}||q||^2$$

avec :

- $X^TX = A$: X^TX est bien une matrice symétrique tel que $a_{11} = m$, $a_{22} = ||p||^2$ et finalement $a_{12} = a_{21} = \frac{a_{12}}{2} = \frac{p^T1}{2}$.
- $(X^Tq)^T = b$: $(X^Tq)^T$ est bien une matrice uni-colonne car le produit matriciel X^Tq est une matrice uni-ligne donc sa transposée est bien une matrice uni-colonne.
- $\frac{1}{2}||q||^2 = p$: $\frac{1}{2}||q||^2$ est bien une constante réelle car $q \in \mathbb{R}^m$.

Donc la fonction F est bien quadratique.

2.2 Le problème (Q) est-il bien posé ?

La finalité de cette sous-section est de répondre à une question mathématique fondamentale que l'on se pose concernant tout problème : le problème de minimisation est-il bien posé au sens d'Hadamard ? En d'autres termes, nous allons tout d'abord chercher à savoir si le problème admet une solution et si elle est unique. Puis nous allons étudier la sensibilité de la solution (si il y a bien unicité) du problème vis-à-vis de la perturbation des données (ici les matrices X^TX et X^Tq).

2.2.1 Autour de la fonction quadratique F

1. Couples propres et conditionnement de X^TX Ici nous désignons par λ_1 et λ_2 les deux valeurs propres, rangées dans l'ordre croissant ($\lambda_1 < \lambda_2$), de la matrice X^TX, et par v_1 et v_2 les deux vecteurs propres (unitaires) associés respectivement à λ_1 et λ_2.

   a. Afin d'obtenir les valeurs propres de la matrice X^TX, nous avons programmé le code suivant :

   ```
   lambda1,lambda2=np.linalg.eigvals(A)
   ```

   Avec la matrice A :

   ```
   La matrice A est :  [[  50.          246.1785986 ]
    [ 246.1785986  1358.30473017]]
   ```

   Nous obtenons ensuite les résulats suivants :

   ```
   Les valeurs propores de la matrice A=XtX sont :  5.210865991628907  et  1403.0938641799348
   Les vecteurs propres de la matrice A=XtX sont:  [-0.98384923  0.17899913]  et  [-0.17899913 -0.98384923]
   ```

   Finalement, nous obtenons les couples suivants :

   ```
   couple 1: ( 5.210865991628907 , [-0.98384923  0.17899913] )
   couple 2: ( 1403.0938641799348 , [-0.17899913 -0.98384923] )
   ```

   Les résultats numériques sont cohérents avec la définie positivité de la matrice X^TX : les valeurs propres $\lambda_1 \approx 5,211$ et $\lambda_2 \approx 1403$ sont positives avec $0 < \lambda_1 < \lambda_2$.

   b. Conditionnement Nous calculons dans un premier temps le conditionnement par la formule :

   $$cond(X^TX) = \frac{max(|\lambda_2|)}{max(|\lambda_1|)}$$

   Nous avons alors :

   $$cond(X^TX) = \frac{1403,0938641799348}{5,210865991628907}$$

   Nous obtenons le résulat suivant :

   $$cond(X^TX) = 269,2630872$$

   En utilisant Python pour calculer le conditionnement, nous avons en utilisant la commande :

   ```
   cond=np.linalg.cond(A)
   ```

   Nous obtenons ensuite :

   ```
   Le conditionnement de la matrice est : 269.2630872553599
   ```

Nous pouvons tout de suite remarquer que les deux méthodes sont extrêmement proches. Nous les comparons tout de même afin d'obtenir un aperçu du pourcentage d'erreur, en prenant comme valeur théorique de référence, le résulat rendu par Python :

$$\frac{|erreur_{thorique} - erreur_{relative}|}{|erreur_{thorique|}}$$

$$\frac{269,2630872553599 - 269,2603872}{269,2630872553599} = 0\%$$

Nous pouvons ainsi être sûr que le résulat est exact.

2. Quelques propriétés élémentaires de F

a. Pour démontrer que la fonction quadratique F est quadratique définie positive. Dans notre problème F est définie positive si elle suit la définition suivante :

$$F \text{ est définie positive } \iff \forall n \in \mathbb{R},\ \ 0 < \lambda_1 < \lambda_2 < ... < \lambda_n$$

Or grâce aux réponses précédentes, nous savons que $\lambda_1 < \lambda_2$. Ainsi, nous pouvons en conclure que la fonction F est définie positive.

b. Nous avons pu démontrer que la fonction F était définie positive, ainsi, ceci nous permet d'affirmer que F est strictement convexe et coercive.

c. Soit $F(c) = \frac{1}{2}\|q - Xc\|^2$, on cherche un point critique de cette fonction tel que :

$$F(c) = 0 \iff q - Xc = 0 \iff q = Xc \iff X^T q = X^T Xc$$

Or $X^T X$ est une matrice symétrique définie positive, ainsi d'après les propriétés d'une matrice définie postivie, la matrice $X^T X$ est inversible.
Donc

$$c^* = (X^T X)^{-1} X^T q$$

Avec c^* l'unique solution du système ainsi $c*$ est aussi l'unique point critique de F.

3. Fonctions partielles de F

a. L'expression analytique de la fonction partielle, notée $F_{a,d}$ de F en a suivant d est la suivante :

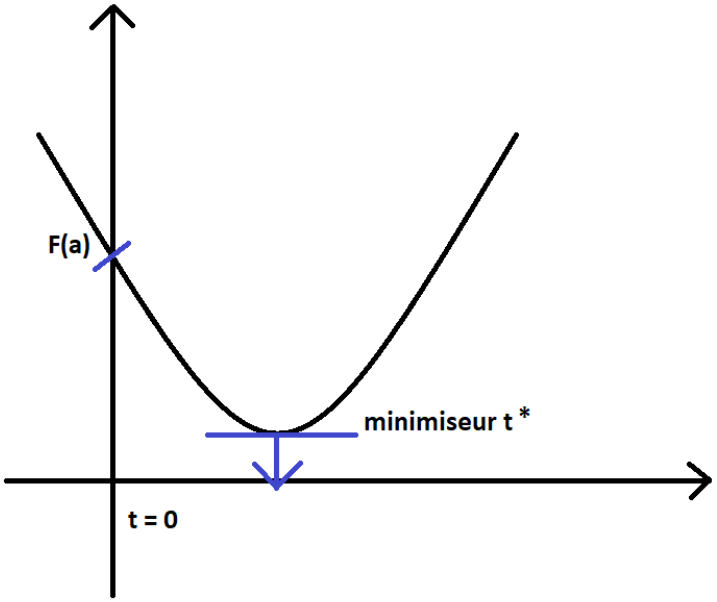
$$\forall t \in \mathbb{R}^N, F_{a,d}(t) = F(a + td) = \frac{1}{2} d^T A d t^2 - (b - Aa)^T dt + F(a)$$

b.

$$A = \begin{pmatrix} 50 & 246,1785986 \\ 246,1785986 & 1358,30473017 \end{pmatrix}$$

A étant une matrice symétrique définie positive, en prenant $d \neq 0$ alors $d^T A d > 0$. Ce qui implique que F est une fonction quadratique définie positive.

c. En ce qui concerne la coercivité de la fonction, nous savons que : $F_{a,d}$ est définie positive sur $\mathbb{R}$, d'après la question précédente. De plus, pour $|t| \to +\infty$, nous avons $F_{a,d} \sim d^T A d t^2$.
$A > 0$, $lim_{|t| \to +\infty} F_{a,d}(t) = +\infty$. Ainsi, F est bien coercive.
En ce qui concerne sa convexité, F étant une fontion partielle définie positive, nous pouvons affirmer qu'elle est strictement convexe, de sommet $(t^*_{a,d}, f_{a,d}(t^*_{a,d}))$, ce qui nous donne la figure suivante :

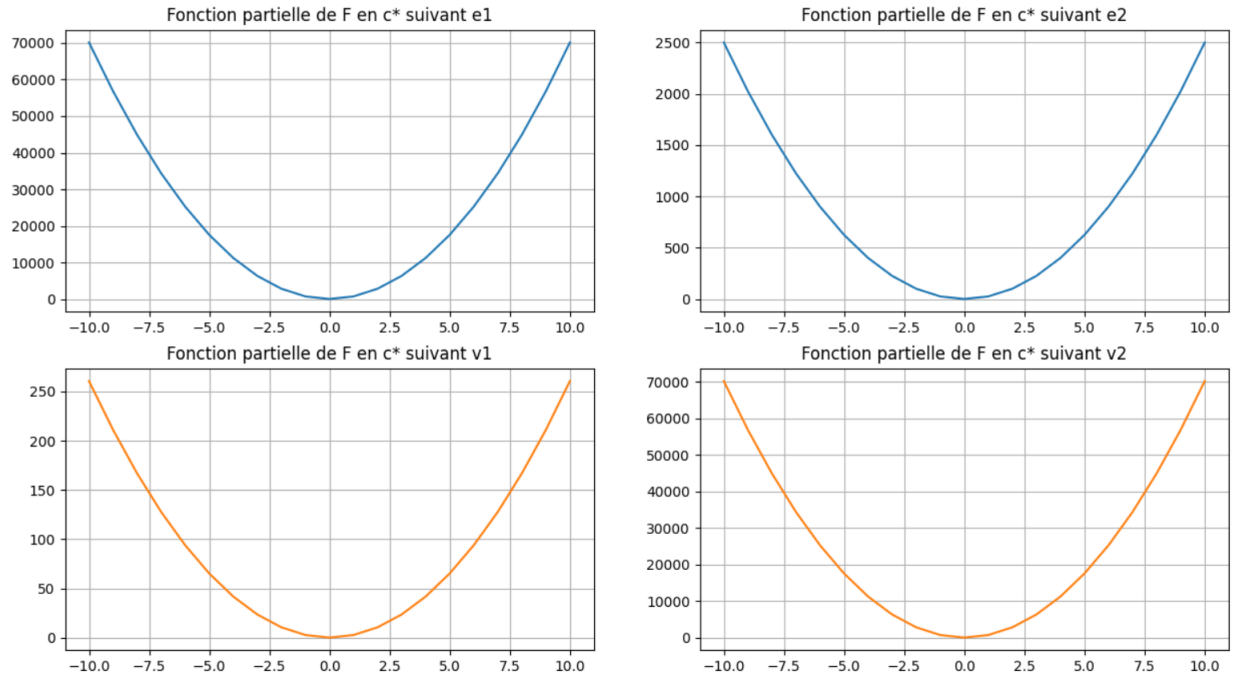


d. L'unique minimiseur global strict de F sur $\mathbb{R}$ est la solution de l'équation $d^T Adt^* = (b - Aa)^T d$. Finalement, nous obtenons que le minimiseur est le réel :

$$t^*_{a,d} = \frac{(b - Aa)^T d}{d^T Ad}$$

e. Sa courbe représentative est une parabole strictement convexe de sommet $(t^*_{a,d}, f_{a,d}(t^*_{a,d}))$ et d'axe de symétrie : la droite d'équation $t = t^*_{a,d}$.

f. Nous obtenons après programmation des fonctions partielles les courbes suivantes :



Représentation des fonctions partielles

Nous constatons que leur représentation correspond bien à la description énoncée à la question précédente. Les courbes ont bien un axe de symétrie d'équation $t = 0$, correspondant également au point critique. Leur sommet étant $(0, 0)$.

4. Carte de niveau

a. Nous construisons la matrice $Z = X^T X = (Z_{ij})_{1 \leqslant i,j \leqslant 2}$.
Nous avons alors :

$$Z = \begin{pmatrix} m & p^T 1 \\ p^T 1 & ||p||^2 \end{pmatrix} = \begin{pmatrix} Z_{1,1} & Z_{1,2} \\ Z_{2,1} & Z_{2,2} \end{pmatrix}$$

et

$$w = X^T q = \begin{pmatrix} q^T 1 \\ p^T q \end{pmatrix} = (w_1 \;\; w_2)^T$$

et

$$s = q^T q = (q_1, ..., q_m)^T q$$

Nous essayons maintenant de retrouver l'équation :

$$F(c) = F(c_1, c_2) = \frac{1}{2}\Big(Z_{1,1}c_1^2 + 2Z_{1,2}c_1c_2 + Z_{2,2}c_2^2 - 2(w_1c_1 + w_2c_2) + s\Big)$$

à partir de l'équation :

$$\forall c \in^2, \quad F(c) = \frac{1}{2}^T \left(X^T X\right) c - \left(X^T q\right)^T c + \frac{1}{2}\|q\|^2$$

Nous avons d'une part :

$$(X^T q)^T = (w_1 \;\; w_2)$$

et

$$(X^T q)^T c = (w_1 \;\; w_2) \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

d'où

$$(X^T q)^T c = c_1 w_1 + c_2 w_2$$

D'une autre part :

$$c^T(X^TX) = (c_1Z - 1, 1 + c_2Z_{2,1} \quad c_1Z_{1,2} + c_2Z_{2,2})$$

et

$$c^T(X^TX)c = (c_1^2Z_{1,1} + c_1c_2Z_{2,1} + c_2c_1Z_{1,2} + c_2^2Z_{2,2})$$

Comme Z est une matrice symétrique, nous avons : $Z_{1,2} = Z_{2,1}$, ainsi nous avons :

$$c^T(X^TX)c = (c_1^2Z_{1,1} + 2c_1c_2Z_{2,1} + c_2^2Z_{2,2})$$

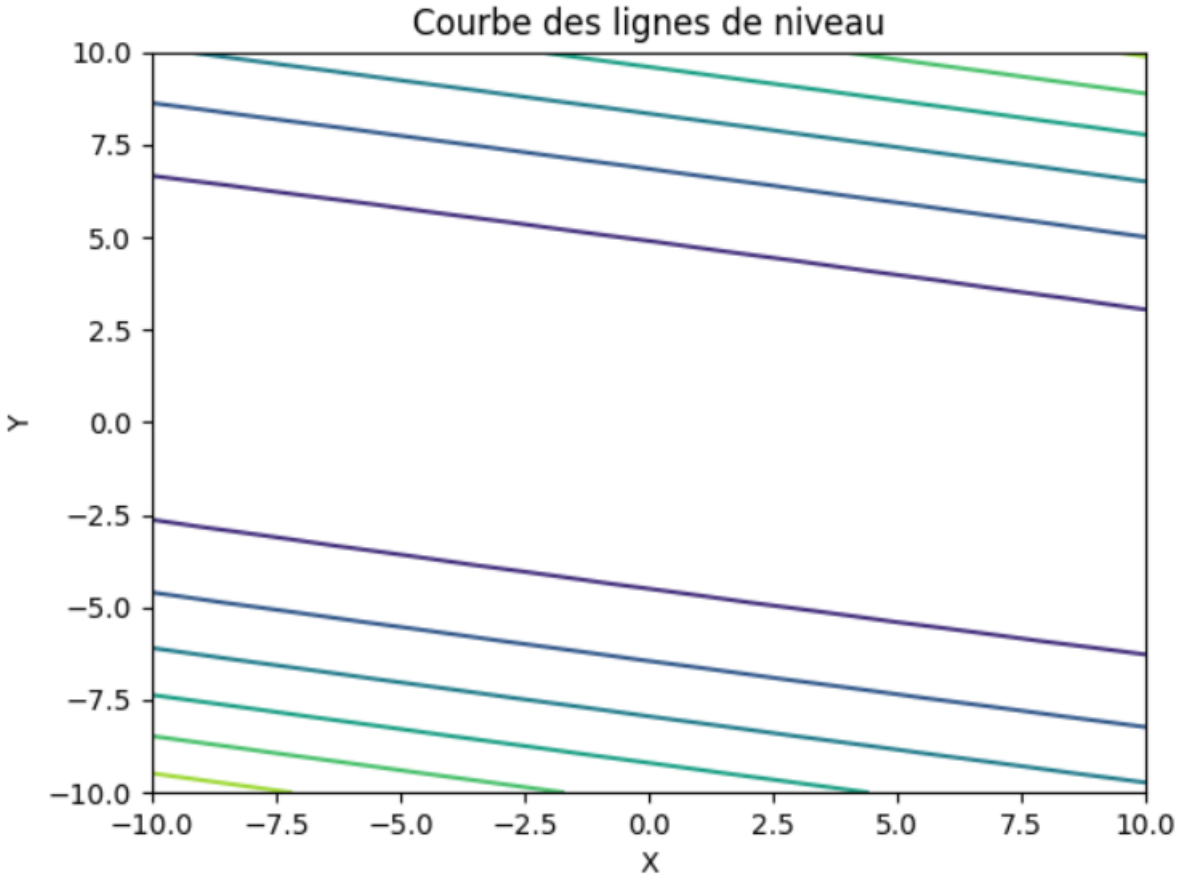
Enfin,

$$\|q\|^2 = q^Tq = s$$

Finalement, en additionnant et multipliant par $\frac{1}{2}$, nous obtenons

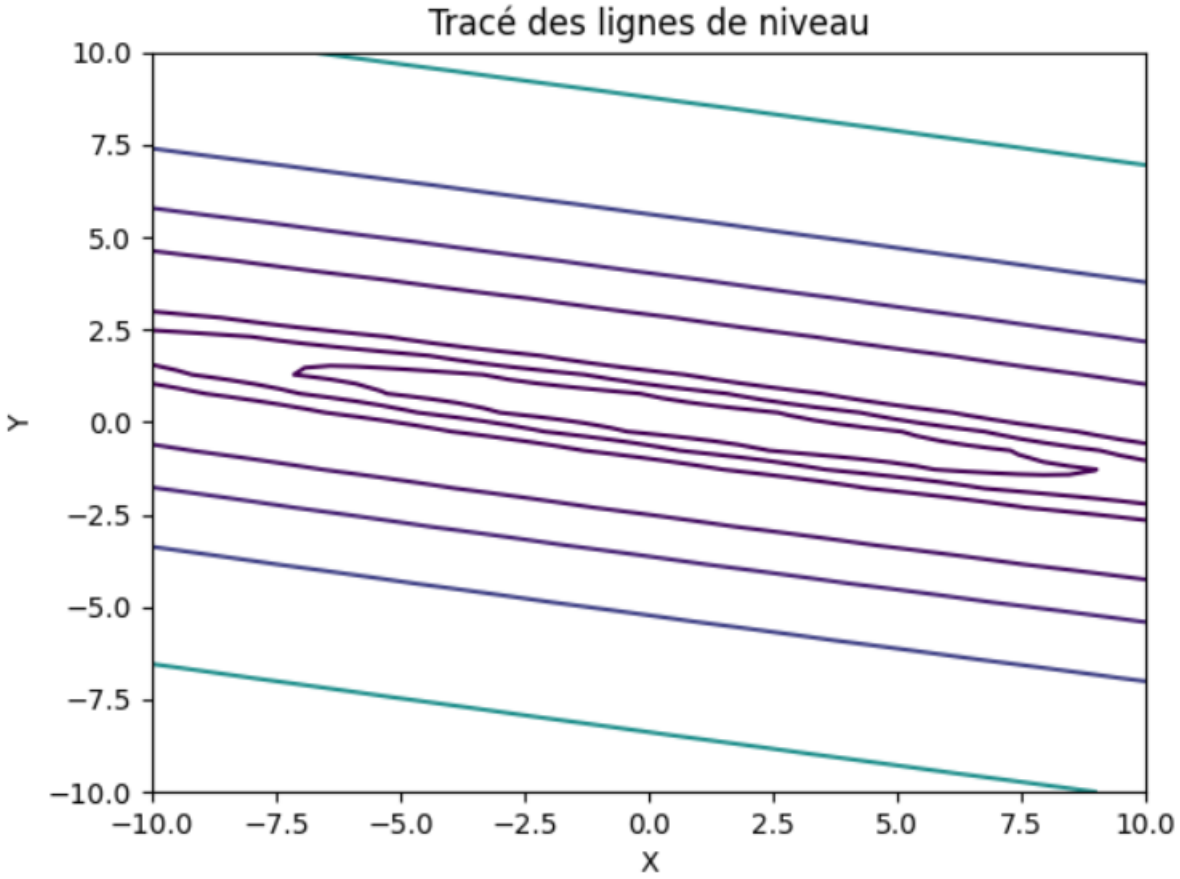
$$F(c) = F(c_1, c_2) = \frac{1}{2}\Big(Z_{1,1}c_1^2 + 2Z_{1,2}c_1c_2 + Z_{2,2}c_2^2 - 2(w_1c_1 + w_2c_2) + s\Big)$$

b. Nous affichons maintenant les courbes de niveaux de la fonction F :



Représentation de la carte de niveaux de la fonction F

A première vue, nous ne pouvons pas distinguer l'allure des lignes de niveau, mais en modifiant notre code, nous pouvons voir que la fonction F est constituée d'ellipses, comme nous pouvons le voir avec le graphique suivant :



Représentation plus précise de la carte de niveaux de la fonction F

c. La forme réduite de F est : $F_{\mathcal{V}_{c*}}(y) = \frac{1}{2}y^T Dy + F(c^*)$ et son repère de réduction est : $\mathcal{V}_{c*} = (\Omega, \mathcal{V})$.
Avec : Ω, le point de représentant matriciel $x^* = A^{-1}b$
et $\mathcal{V}_{c*}$, la base orthonormée des vecteurs propres de A, calculés prédécemment à la question $2.2.1.a$ soit : $\mathcal{V}_{c*} = (v_1, v_2)$.

d. La carte de niveaux de F est donnée par :
Si $\beta < F(c^*)$, alors $L_\beta(f) = \varnothing$
Si $\beta = F(c^*)$, alors $L_\beta(f) = c^*$
Si $\beta > F(c^*)$, alors $L_\beta(f)$ est la ligne de niveau d'équation cartésienne dans le repère $\mathcal{V}_c^*$:

$$\frac{y_1^2}{\left(\sqrt{\frac{2(\beta - F(c^*))}{\lambda_1}}\right)^2} + \frac{y_2^2}{\left(\sqrt{\frac{2(\beta - F(c^*))}{\lambda_2}}\right)^2} = 1$$

Avec les valeurs que nous avons calculées précédemment, nous obtenons :

$$\frac{y_1^2}{\left(\sqrt{\frac{2(\beta - 0)}{5{,}21086599}}\right)^2} + \frac{y_2^2}{\left(\sqrt{\frac{2(\beta - 0)}{1403.09386418}}\right)^2} = 1$$

Nous avons donc : un 2-elipsoïde de centre Ω de demi grand-axe de longueur : $\frac{y_1^2}{\left(\sqrt{\frac{2(\beta-0)}{5{,}21086599}}\right)^2}$ et de demi petit-axe de longueur : $\frac{y_2^2}{\left(\sqrt{\frac{2(\beta-0)}{1403.09386418}}\right)^2}$, portés respectivement par les vecteurs directeurs v_1 et v_2 de la matrice A.

e. L'équation donnée ci-dessus correspond à l'équation d'une ellipse. Ce est bien en accord avec le tracé des lignes de niveau que nous avons donné précédemment.
f. On a $\lambda_1 << \lambda_2$, donc le plus demi grand-axe de F est est supérieur au demi petit-axe. Ansi nous obtiendrons des ellipses beaucoup plus aplaties. Cette géometrie pourra nous renseigner sur le conditionnement de la matrice. En effet, une matrice avec un bon conditionnement aura des lignes de niveau plutôt circulaires. Nous pouvons ainsi déduire que la matrice $Z = X^T X$ n'a pas un bon conditionnement.

5. Surface représentative
a. Nous programmons maintenant un code afin d'obtenir le graphique de la surface F :

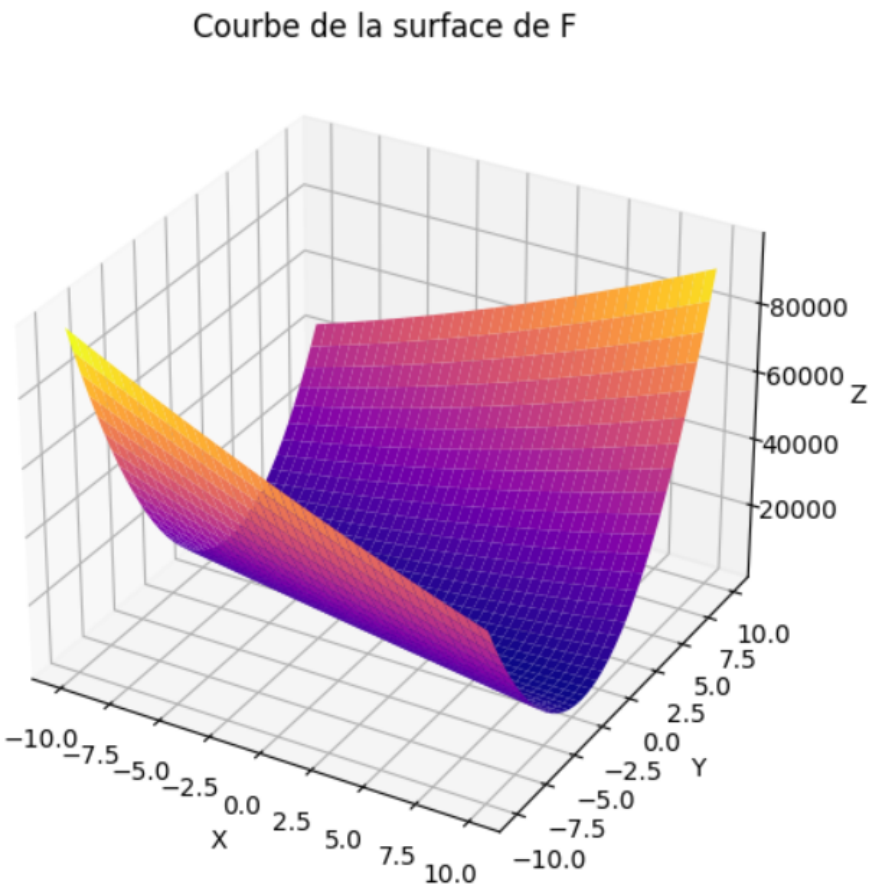
```
############ Tracé de la Surface de F############

fig = plt.figure(figsize=(8,6))
ax3d = plt.axes(projection='3d')

ax3d = plt.axes(projection='3d')
ax3d.plot_surface(X,Y,Z,cmap='plasma')
ax3d.set_title('Courbe de la surface de F')
ax3d.set_xlabel('X')
ax3d.set_ylabel('Y')
ax3d.set_zlabel('Z')

plt.show()
```

Nous obtenons alors :



Surface représentative de la fonction F

Nous pouvons constater que la surface est en forme de parabole, et que si nous coupons horizontalement cette surface, nous obtiendrons "un bol".

b. Les caractéristiques géométriques de la surface représentative de F, notée S_F, sont les suivantes :

- Des tranches horizontales correspondant à la carte de niveau formée d'ellipses centrées en $x*$.
- Des tranches verticales centrées au point Ω^+ : ce sont des paraboles strictements convexes d'axe de symétrie commun la droite $D_{\Omega^+,e3}$, passant par le point Ω^+ et dirigé par le vecteur $e3$, le troisième vecteur de la base canonique.
- Un centre : le point Ω^+.
- Un axe de symétrie : la droite D_{Ω^+}.

c. Nous pouvons dire que S_F est un paraboloïde elliptique d'après les caractéristiques géométriques de la surface représentative. En effet, les tranches horizontales sont des ellipses centrées au minimiseur global stricte de f et que les tranches verticales centrées en Ω^+ sont des paraboles.

2.2.2 Unicité de la solution de (Q)

Ici nous allons établir l'unicité de la solution du problème (Q). Le problème est de dimension 2 et nous donnerons une forme explicite de la solution.

6. On sait que la forme réduite de F est :

$$F(c) = \frac{1}{2}||q - Xc||^2$$

Donc $F(c) = 0 \iff (X^TX)c* = X^Tq$ avec c* l'unique solution de l'équation donc $c*$ est l'unique point critique de la fonction F.

7. Comme $c*$ est le minimiseur global stricte de la fonction mais aussi sont point critique, on a l'équivalence du problème suivante :

$$(X^Tx)c* = X^Tq$$

8. Nous allons chercher à expliciter $c*$, le représentant dans le repère canonique de l'unique solution de (Q).

a. Nous avons établie à la question précédente que $XX^Tc* = X^Tq$. On utilise l'expression (7) du sujet et on injecte dans cette équation tel que :

$$\begin{pmatrix} m & p^T1 \\ p^T1 & ||p||^2 \end{pmatrix} \begin{pmatrix} c_1^* \\ c_2^* \end{pmatrix} = \begin{pmatrix} q^T1 \\ p^Tq \end{pmatrix}$$

Or on sait que X^TX est symétrique définie positive donc on a :

$$\iff \begin{pmatrix} c_1^* \\ c_2^* \end{pmatrix} = \frac{1}{m||p||^2 - (p^T1)^2} \begin{pmatrix} ||p||^2 & -p^T1 \\ -p^T1 & m \end{pmatrix} \begin{pmatrix} q^T1 \\ p^Tq \end{pmatrix}$$

$$\iff \begin{pmatrix} c_1^* \\ c_2^* \end{pmatrix} = \frac{1}{m||p||^2 - (p^T1)^2} \begin{pmatrix} ||p||^2q^T - (P^T1)(P^Tq) \\ m(p^Tq) - (q^T1)(p^T1) \end{pmatrix}$$

Donc finalement on retrouve :

$$\boxed{\begin{cases} c_1^* = \frac{||p||^2q^T1 - (P^T1)(P^Tq)}{m||p||^2 - (p^T1)^2} \\ c_2^* = \frac{m(p^Tq) - (q^T1)(p^T1))}{m||p||^2 - (p^T1)^2} \end{cases}}$$

b. Afin de trouver les valeurs de $c*$, nous exécutons le code suivant :

```
###############Calcul de c* en fonction de p,q et 1#################################
U=np.ones((50,1))#creation du vecteur unitaire
c1star=((np.linalg.norm(p)**2)*np.transpose(q)@U-(np.transpose(p)@U)*(np.transpose(p)@q))/(50*(np.linalg.norm(p)**2)-(np.transpose(p)@U)**2)
c2star=(50*(np.transpose(p)@q)-(np.transpose(q)@U)*(np.transpose(p)@U))/(50*(np.linalg.norm(p)**2)-(np.transpose(p)@U)**2)
print("\ncalcul de c* a l'aide de p,q et 1 :")
print("c1*: ",c1star)
print("c2*: ",c2star)
```

Nous obtenons alors les résultats suivant :

```
calcul de c* a l'aide de p,q et 1 :
c1*:  [[0.75016254]]
c2*:  [[0.06388117]]
```

2.2.3 Conditionnement du problème

Dans ce paragraphe, nous allons cherché estimer les erreurs commises sur la solution * du problème d'optimsiation quadratique sans contrainte en fonction des erreurs commises sur les données du problème, *c.-à-d.* la matrice X^TX et le vecteur X^T définissant la fonction F. Comme nous avons établi à la sous-section que le problème de minimisation et le problème de résolution du système linéaire inversible sont équivalent (au sens fort d'avoir la même solution, qui est de plus unique), notre étude correspond à l'étude de la sensibilité aux données d'un système linéaire carré inversible Il faut noter que cette problématique a déjà été traité au premier semestre dans le cadre de l'enseignement Ma 313 *Algèbre linéaire numérique*.

Avant de poursuivre dans le vif du sujet, on peut se poser la question de l'utilité et de l'impact de cette étude. En effet, plus que la résolution théorique des problèmes équivalents et c'est leur résolution pratique sur ordinateur (résolution numérique) qui nous interresse ici en finalité (c'est notamment l'objectif principal d'un ingénieur !). Vous avez déjà vu précédement (autres enseignements d'analyse numérique comme Ma 123, Ma 223, Ma 313) que les algorithmes de résolution doivent être efficaces, *c.-à-d.* être rapide (en minimisant le nombre d'opération élémentaires, d'itération ou de temps CPU) et nécessiter peu de place mémoire. Par ailleur, il existe une autre exigence pratique pour les méthodes numériques : leur précision. En effet, limité à cause du nombre de bits utilisés pour représenter les nombres réels : d'habitude 32 ou 64 bits (ce qui fait à peu près 9 ou 16 chiffres significatifs). Il faut donc faire très attention aux inévitables erreurs d'arrondi et à leur propagation au cours d'un calcul. Dans le contexte de notre problème, la matrice X^TX et le vecteur X^T sont connue à une erreur près et les trois méthodes considérées sont suceptibles de propager ses erreurs. On aimerait que lerreur commise sur les données du problème (ici la matrice X^TX et le vecteur X^T) nait pas une conséquence trop grave sur le calcul de la solution du problème (ici * les coefficients du modèle linéaire). Si par exemple 1% derreur sur les coefficients de X^TX etX^T entraîne 100% derreur sur *, le modèle linéaire obtenu ne sera pas dune utilité redoutable. Dans ce contexte, Il faut s'assurer qu'une méthode de résolution (notamment les trois méthodes étudiées dans ce projet) ne favorise pas une telle amplification : c'est ce qu'on appelle la stabilité d'une méthode. Cette question est d'autant plus importante pour les méthodes itérative qui ne calcule pas comme les méthodes directes une solution exacte (au sens de l'arithmétique parfaite sans arrondi) mais une suite de solutions approchées qui converge vers la solution exacte. Les méthodes qui seront étudier par la suite sont itératives. Dans la suite de ce projet, au niveau du paragraphe (quastions 6 et 7) étudirons l'impacte des erreurs d'arrondi sur la précision des résultats numériques obtenus.
Pour quantifier le phénomène des erreurs d'arrondi, on considère la notion de conditionnement de la matrice symétrique définie positive X^TX (**Utiliser le lien :**Norme matricielle et conditionnement).

Au niveau de ce paragraphe, nous allons nous contenter d'estimer la sensibilité de la solution lorsque l'on perturbe (juste) le second membre X^T du problème. Plus précisément, nous allons établir une majoration qui estime la sensibilité relative de la solution du problème en fonction de la perturbation relative du vecteur X^T. Nous en déduirons que la sensibilité de l'erreur relative de la solution est d'estimée par le conditionnement (en norme euclidienne) de la matrice X^TX.

9. La finalité de cette question est d'évaluer l'impact du conditionnement de la matrice X^TX sur la sensibilité de la solution de (Q). Nous ne perturberons que le second membre du système (8) et nous désignons par :
   - $\delta(X^Tq)$ une perturbation du second membre X^Tq du problème (8)
   - $c^* + \delta c^*$ l'unique solution du système linéaire perturbé inversible $X^TXc = X^Tq + \delta(X^Tq)$

a. Soit $c^* + \delta c^*$ l'unique solution de $X^TXc = X^Tq + \delta(X^Tq$. On a donc :

$$X^TX(c^* + \delta c^*) = X^Tq + \delta(X^Tq) \iff c^* + \delta c^* = (X^TX)^{-1}(X^Tq + \delta(X^Tq))$$

$$\iff \delta c^* = (X^TX)^{-1}(X^Tq + \delta(X^Tq)) - c^*$$

Or on sait que $X^TXc = X^Tq \iff c^* = (X^TX)^{-1}X^Tq$ donc $(X^TX)^{-1}Q^Tq - c^* = 0$ d'où :

$$\delta c^* = (X^TX)^{-1}\delta(X^Tq) \Rightarrow ||\delta c^*|| = ||(X^TX)^{-1}\delta(X^Tq)||$$

Nous avons établit que (X^TX) était symétrique définie positive donc son inverse l'est aussi. On a donc :

$$||\delta c^*|| = ||(X^TX)^{-1}\delta(X^Tq)|| \leqslant ||(X^TX)^{-1}||.||\delta(X^Tq)||$$

Finalement :

$$\boxed{||\delta c^*|| \leqslant ||(X^TX)^{-1}||.||\delta(X^Tq)||}$$

b. On sait que $||c^*|| > 0$ donc :

$$\frac{||\delta c^*||}{||c^*||} \leqslant \frac{||(X^TX)^{-1}||}{||c^*||}||\delta(X^Tq)||$$

Or :

$$||(X^TX)c^*|| = ||X^Tq||$$

et comme X^TX est définie positive on a alors :

$$||X^TXc^* \leqslant ||X^TX||.||c^*|| \iff ||X^Tq|| \leqslant ||X^TX||.||c^*||$$

$$\iff ||c^*|| \leqslant \frac{||X^Tq||}{||X^TX||}$$

$$\iff \frac{1}{||c^*||} \leqslant \frac{||X^TX||}{||X^Tq||}$$

Ainsi en reprenant l'inégalité en début de question et on injectant ce que nous venons de trouver :

$$\frac{||\delta c^*||}{||c^*||} \leqslant ||(X^TX)^{-1}||\frac{||X^TX||}{||X^Tq||}||\delta(X^Tq)||$$

Or :

$$cond_2(X^TX) = ||(X^TX)^{-1}||.||X^TX||$$

Donc finalement :

$$\boxed{\frac{||\delta c^*||}{||c^*||} \leqslant cond_2(X^TX)\frac{||\delta(X^Tq)||}{||X^Tq||}}$$

c. En reprenant le résultat de la question précédente nous avons :

$$cond_2(X^TX) \geqslant \frac{||\delta c^*||}{||c^*||}\frac{||X^Tq||}{||\delta(X^Tq)||}$$

On peut donc travailler sur le raisonnement suivant tel que :

- Plus $||\delta(X^Tq)||$ augmente plus le conditionnement donc la sensibilité diminue.
- Plus $||c^*||$ augmente plus le conditionnement donc la sensibilité diminue.
- Plus $||\delta c^*||$ diminue plus le conditionnement donc la sensibilité augmente.
- Plus $||X^Tq||$ diminue plus le conditionnement donc la sensibilité augmente.

Dans le cas idéal on voudrait que $||\delta(X^Tq)||$ et $||c^*||$ très grand ainsi que $||\delta c^*||$ et $||X^Tq||$ très petit pour avoir une sensibilité optimal du problème.

3 Méthodes à directions de descente : une étude numérique

3.1 Principes généraux

Dans cette sous-section, nous allons construire le schéma général de l'itération de la classe importante d'algorithmes de résolution d'un problème d'optimisation sans contrainte : les méthodes à direction de descente. Ces méthode d'applique sur des fonctions suffisamment régulières. C'est l'objet de l'enseignement de Filière système Ma 324 *Une introduction à l'optimisation différentiable*.

Dans le contexte de ce problème, on considère le problème *général* d'optimisation quadratique définie positif sans contrainte de dimension 2 :

$$\begin{cases} \text{Minimiser} \quad f() := \frac{1}{2}^T A -^T + c \\ := \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in^2 \end{cases} , \tag{G}$$

où

- $A \in \mathcal{M}_2()$ symétrique définie positive dont les deux valeurs propres sont : $0 < \lambda_1(A) \leqslant \lambda_2(A)$;
- $:= \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \in^2$;
- $c \in$.

Nous allons maintenant établir le schéma général de la $k-$ième itération de la classe des méthodes à directions de descente associée à la résolution du problème *général* (G). Nous supposons au début de l'itération k que l'on dipose d'un itéré noté $_k$. Nous allons maintenant montrer comment on calcul l'itéré suivant, noté bien évidement $_{k+1}$.

3.1.1 Direction de descente

1. Definition : Soient $f : \mathbb{R} \to \mathbb{R}$ une fonction différentiable. Soient $x_k \in \mathbb{R}^n$ tel que $\nabla f(x_k)^T \neq 0$ et $d \in \mathbb{R}^n$. Si d est une direction de descente, alors il existe $\eta > 0$ tel que :

   $$f(x_k + \alpha) < f(x_k), \forall 0 < \alpha \leqslant \eta$$

   De plus, pour tout $\beta < 1$, il existe $\widehat{\eta} > 0$ tel que :

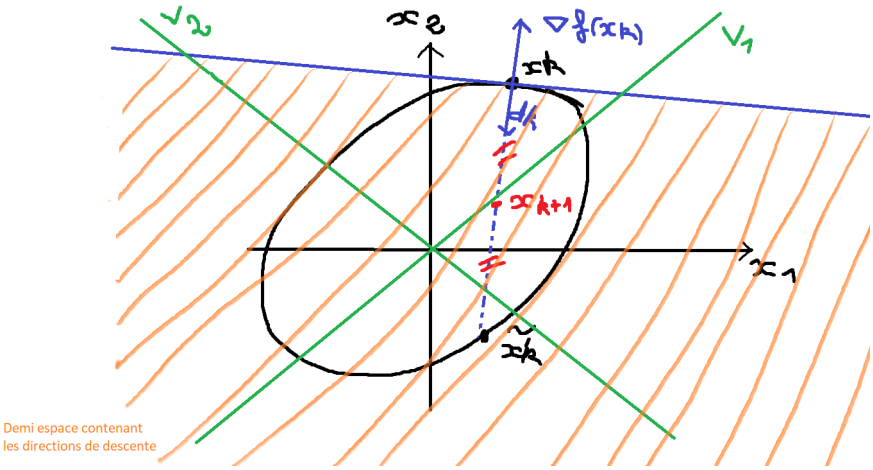
   $$f(x_k + \alpha_k d_k) \leqslant f(x_k) + \alpha_k \beta_1 \nabla f(x_k)^T d_k$$

   pour tout $0 < \alpha \leqslant \widehat{\eta}$,

   Soit d_k la direction de descente de la fonction f au point x_k tel que :

   $$f'_{x^k}, d_k(0) < 0 \Leftrightarrow D_{dk} f(x_k) < 0 \Leftrightarrow \nabla f(x_k)^T < 0 \tag{6}$$

2. Nous représentons les courbes de niveaux passant par le point x_k dans le plan x_1, x_2 :



Courbe de niveau passant par le point x_k dans le plan x_1, x_2

d_k est une direction de descente de f en x_k si et seulement si :

$$Dd_k f(x_k) = f'(0) < 0$$

On calcule $Dd_k f(x_k)$ dans le cas de $f(x) = \frac{1}{2}x^T Ax - b^T x + c$

$$f_{x_k, d_k}(\alpha) = \frac{1}{2}(d_k^T A d_k)\alpha^2 + (Ax_k - b)^T d_k \alpha + f(x_k)$$

$$\Leftrightarrow f'_{x_k, d_k}(\alpha) = d_k^T A d_k \alpha + (Ax_k - b)^T d_k$$

$$Dd_k f(x_k) = (Ax_k - b)^T d_k = \nabla f(x_k)^T d_k$$

En effet :

$$\nabla f(x_k) = \begin{pmatrix} \frac{df}{dx_1}(x_k) \\ \vdots \\ \frac{df}{dx_i}(x_k) \\ \vdots \\ \frac{df}{dx_n}(x_k) \end{pmatrix} = \begin{pmatrix} De_1 f(x_k) \\ \vdots \\ De_i f(x_k) \\ \vdots \\ De_n f(x_k) \end{pmatrix} = \begin{pmatrix} (Ax_k - b)^T e_1 \\ \vdots \\ (Ax_k - b)^T e_i \\ \vdots \\ (Ax_k - b)^T e_n \end{pmatrix} = Ax_k - b$$

Sur la figure, $\nabla f(x_k)$ est orthogonale à la ligne de niveau de f passant par le point x_k et indique le demi-espace contenant les directions de montées de f en x_k.

$Vd \neq 0$, appartient au demi-espace hachuré sur la figure, on a :

$$\nabla f(x_k)^T = \| \nabla f(x_k) \| \| d \| cos(\nabla f(x_k), d) > 0$$

Par définition, d est une direction de montée de f en x_k, par conséquent, le demi-espace ouvert de frontière la droite bleu ne contenant pas $\nabla f(x_k)$ continent les directions de descente de f en x_k notamment la direction d_k

Dans la suite de projet nous allons considérer deux choix de direction de descente : 1) descente du gradient ou de la plus profonde descente qui consiste à poser $d_k = -\nabla f(x_k)$; 2) descente du gradient conjugué qui sera une correction de la direction de plus profonde descente qui tiendra compte au travers de la matrice A de la géométrie du problème (courbure directionnelle au travers de la différentielle seconde).

3.1.2 Pas de decente

Nous déduisons du schéma établi à la question précédente, qu'il existe une infinité de choix de direction de descente. Dès lors, une fois choisie la façon de calculer la direction de descente $_k$ à l'itéré $_k$, il apparaît que nous pouvons *descendre* ou faire décroître la fonction f à partir de $_k$ suivant cette direction $_k$. En d'autres termes il existe α_k tel que :

$$f(x_k + \alpha_k d_k) < f(x_k). \tag{7}$$

Nous dirons alors que α_k est un **pas de descente** de f en $_k$ suivant $_k$. Nous allons montrer qu'il existe une infinité de choix de pas de descente de la fonction f à partir du point $_k$ suivant $_k$.

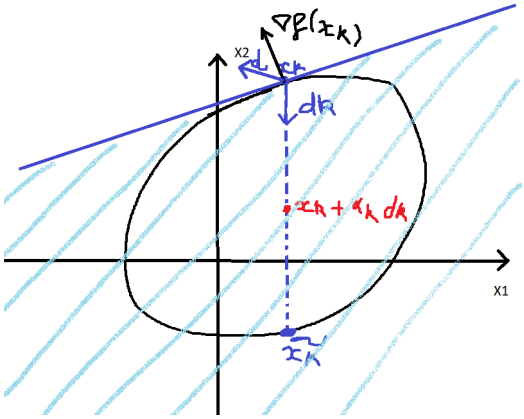
3. Dans cette question nous allons montrer le lemme ci-dessous qui justifie l'existence d'une infinité de pas de descentes pour une direction de descente calculée.

Proposition : (CNS d'existence d'un pas de descente) Soit $_k \neq \mathbf{0}$ une direction de descente de f au point x_k suivant d_k. Alors tout réel $\alpha \in]0, \tilde{\alpha}_k[$, où

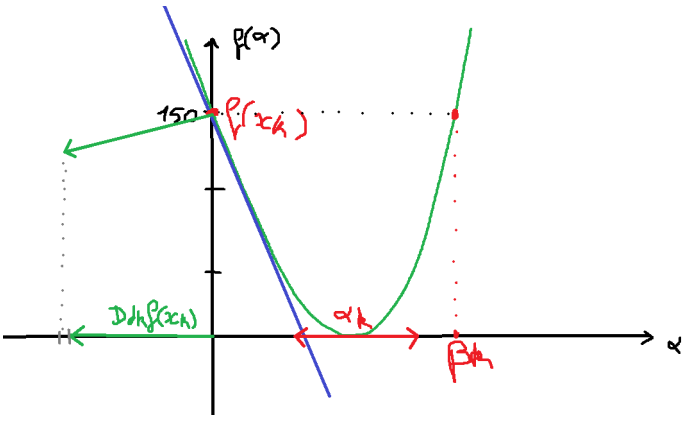
$$\tilde{\alpha}_k := -\frac{2\,(Ax_k - b)^T\,\mathbf{d}_k}{\mathbf{d}_k^T A \mathbf{d}_k}, \tag{8}$$

est un pas de descente de F en x_k suivant d_k.

a. Nous représentons maintenant les courbes de niveaux de la fonction f passant par le point x_k, ainsi que son approximation tangentielle $(figure\ 2a)$, ainsi que la représentation de la fonction partielle f_{x_k,d_k} $(figure\ 2b)$:



Courbe de niveau (2a) passant par le point x_k dans le plan x_1, x_2



Courbe (2b) représentative de la fonction partielle $f_{xk,d\,k}$

b.

$$f_{xk,d\,k}(t) = f(x_k + td_k) = \frac{1}{2}dk^T Ad_k t^2 + f(x_k)$$

c. Nous voulons choisir un $\alpha_k > 0$ tel que :

$$f(x_k + \alpha_k d_k) < f(x_k)$$

donc :

$$f_{xk,d\,k}(\alpha_k) < f_{xk,d\,k} \tag{9}$$

Dans le cas étudié, où f est quadratique définie positive :

$$f_{xk,d\,k}(\alpha) = \frac{1}{2}dk^T Ad_k t^2 + (AX_k - b)^T d_k + f(x_k)$$

Nous cherchons à vérifier analytiquement qu'il existe α_k vérifiant notre expression. Nous vérifions facilement analytiquement ce que nous observons sur notre courbe 2 :

Il existe $\beta_k > 0$ tel que pour tout α_k appartenant à $]0, \beta_k[$, on a bien notre expression (4)

Nous déterminons maintenant $\beta_k > 0$ et $f_{xk,d\,k}(\beta_k) = f(x_k)$

D'où :

$$\frac{1}{2}d_k Ad_k \alpha^2 + (Ax_k - b)^T dk\alpha + f(x_k) = f(x_k)$$

Ainsi, nous avons :

$$\alpha[\frac{1}{2}d_k^T Ad_k \alpha^2 + (Ax_k - b)^T dk] = 0$$

Cette équation a deux solutions :

$\alpha_1 = 0$, ce qui correspond à $[x_k]$ et $\tilde{\alpha_k} = \frac{-2(Ax_k - b)^T d_k}{d_k^T Ad_k} = \beta_k > 0$

En effet, nous avons $\tilde{\alpha_k} > 0$ car d_k est une direction de descente de f en x_k

Donc $Dd_k f(x_k) = \nabla f(x_k)^T d_k = (Ax_k - b)^T dk < 0$

Ainsi :

$$Dd_k f(x_k) = -2(Ax_k - b)^T d_k > 0$$

Sachant que f est définie positive, nous avons A définie positive, de plus, $d_k \neq 0$ par définition. Nous avons donc $\beta_k > 0$

Nous pouvons dire que comme $f_{x\,k,d\,k}$ est une parabole strictement convexe, nous a ons quelque soit $\alpha_k \in]0, \beta_k[$

$$f_{x\,k,d\,k}(\alpha_k) < f_{x\,k,d\,k}(0)$$
$$\Leftrightarrow f(x_k + \alpha_k d_k)$$

Nous avons donc un nouvel itéré :

$$x_k + 1 := x_k + \alpha_k d_k$$

Nous avons donc bien, montré que α_k est un pas descente de f en x_k suivant d_k.

d. Dans la question précédente, nous avons pu trouver que l'une des deux solutions de l'équation :

$$\alpha[\frac{1}{2} d_k^T A d_k \alpha^2 + (Ax_k - b)^T dk] = 0$$

était :

$$\tilde{\alpha_k} = \frac{-2(Ax_k - b)^T d_k}{d_k^T A d_k}$$

Ainsi il suffit de faire ce déplacement quantifié par le pas α_k le long d_k et obtenir le nouvelle itéré x_{k+1}. Les méthodes à directions de descente utilisent cette stratégie pour *descendre* vers le minimiseur de la fonction f. Elles construisent la suite des itérés $(x_k)_{k\in\mathbb{N}}$ approchant l'unique solution du problème d'optimisation (G), notée x*, par la récurrence

$$x_{k+1} = x_k + \alpha_k d_k. \tag{10}$$

Par conséquent pour définir une méthode à directions de descente il faut spécifier les deux choses :

- dire comment la direction de descente d_k est calculée ; la manière de procéder donne le nom à l'algorithme ;
- dire comment on détermine le pas de descente α_k ; c'est ce que nous appellons la **recherche linéaire**.

Au vu du lemme, il existe de très nombreuses stratégies de calculer à l'itéré $_k$ un pas de descente α_k le long d'une direction de descente d_k. Dans ce projet, nous allons considérer deux stratégies. La première va consister de garder le même pas à toutes les itérations, nous parlerons alors de méthode à direction de descentes à pas constant. La seconde, est bien plus naturelle. En effet, comme nous cherchons à minimiser f, il semble naturel à l'itéré x_k de chercher à minimiser la fonction objectif f le long de la direction de descente d_k et donc de déterminer le pas α_k comme solution du problème de minisation d'une variable réelle :

$$\begin{cases} \text{Minimiser} \quad f_{\mathbf{x}_k,\mathbf{d}_k}(\alpha),\ \alpha \in \mathbb{R} \\ \text{Sous la contrainte :} \\ \alpha > 0. \end{cases} \tag{11}$$

Dans ce cas α_k sera appelé pas optimal. Si cette stratégie de recherche linéaire est choisie, nous parlerons alors de méthode à direction de descente à pas optimal.

4. Dans cette question nous allons montrer que le pas optimal, noté α_k^*, de f en $_k$ suivant la direction de descente (non nulle) $_k$ est donnée par :

$$\alpha_k^* = -\frac{(A\mathbf{x}_k - \mathbf{b})^T \mathbf{d}_k}{\mathbf{d}_k^T A \mathbf{d}_k}. \tag{12}$$

L'expression de la fonction partielle f_{x_k,d_k} de f au point x_k dans la direction $_k$. est :

$$\forall t \in^n, \quad f_{\mathbf{x}_k,\mathbf{d}_k}(t) = f(\mathbf{x}_k, \mathbf{d}_k t) = \frac{1}{2} dk^T A d_k t^2 - (b - Ax_k)^T . d_k t + f(x_k) \tag{13}$$

$f_{\mathbf{x}_k,\mathbf{d}_k}$ est bien quadratique définie positive.

a. Une fonction quadratique définie positive d'une variable x admet la solution de l'équation $ax = b$ notée x* comme minimiseur global stricte. Nous reprenons notre expression énoncée précédemment et nous procédons par identification pour avoir notre a et b :

$$\forall t \in^n, \quad f_{\mathbf{x}_k,\mathbf{d}_k}(t) = f(\mathbf{x}_k, \mathbf{d}_k t) = \frac{1}{2} dk^T A d_k t^2 - (b - Ax_k)^T . d_k t + f(x_k) \tag{14}$$

Nous identifions a comme étant : $d_k^T A d_k$, et b comme étant $(Ax_k - b)^T d_k$

Nous avons donc alors notre α_k^* qui est notre minimiseur global stricte qui est égal à :

$$\alpha_k^* = \frac{-(Ax_k - b)^T d_k}{d_k^T A d_k} \tag{15}$$

b. Nous voulons maintenant montrer que α_k^* est positif, et nous savons que :

$$-(Ax_k - b)^T d_k = -(x_k)^T d_k = -D d_k f(x_k) > 0 \tag{16}$$

Comme nous savons que la direction de descente est strictement décroissante de part sa nature, comme nous avons un signe - devant celle-ci, alors l'expression est croissante.

De plus, comme A est une matrice définie positive, $d_k^T A d_k$ est positif.

Comme nous avons un quotient de deux expressions positives, alors nous avons bien $\alpha_k^* > 0$.

c. La courbe représentative de f est Cf_{xk}, d_k, est une parabole strictement convexe de sommet α_k^* et d'axe de symétrie la droite d'équation $x1 = \alpha_k^*$

De plus, $0 < \alpha_k^* < \tilde{\alpha_k}$, donc :

$$f_{xk}, d_k(0) = f_{xk,d\,k}$$

Donc :

$$\alpha_k^* = \frac{\tilde{\alpha_k}}{2}$$

$$\Leftrightarrow \alpha_k^* d_k = \frac{\tilde{\alpha_k}}{2} d_k$$

$$\Leftrightarrow x_k + \alpha_k * d_k = x_k + \frac{\tilde{\alpha_k} d_k}{2}$$

$$\Leftrightarrow x_{k+1} = x_k + \frac{\tilde{\alpha_k} d_k}{2}$$

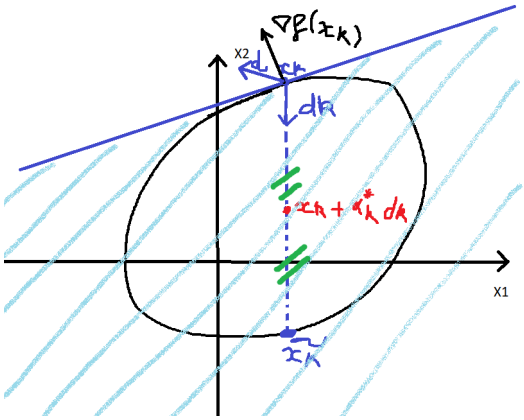
Nous remarquons que $x_k + \frac{\tilde{\alpha_k}}{2} d_k$ est le milieu du segment $[x_k, x_k + \alpha_k^* d_k]$

Nous avons donc M, le milieu du segment $[x_k, x_k + \alpha_k^* d_k]$, tel que :

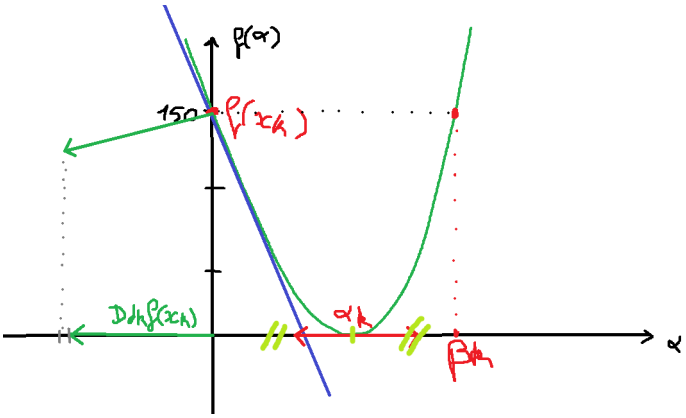
$$\frac{x_k + x_k + \alpha_k^* d_k}{2} = x_k + \frac{\tilde{\alpha_k} d_k}{2}$$

La position du $(k+1)$ ième itéré est donc le milieu du segment $[x_k, x_k + \alpha_k^* d_k]$

d. Nous réalisons les mêmes figures que précédemment illustrées, mais dans le cas où α_k désigne le pas optimal de f en x_k suivant d_k :



Courbe (2a) de la question 3. dans le cas ou α_k designe le pas optimal



Courbe (2b) de la question 3. dans le cas ou α_k designe le pas optimal

3.1.3 Critère d'arrêt

Au vu de la construction précédente (*c.f.* la formule (10) de calcul de $_{k+1}$ à partir de $_k$), les algorithmes à direction de descente sont des algorithmes de type itératif susceptible de *résoudre* le problème (G). Dans un monde idéal (*c.-à-d.* en supposant tous les calculs exacts et la capacité de calcul de la machine illimitée), soit l'algorithme produit une suite fini $_{0,1},\ldots,_k$ qui identifie l'unique solution * à la $k-$ième iteration (*c.-à-d.* $_k =$*), soit il construit (théoriquement) une suite infinie $_{0,1},\ldots,_k,\ldots$ de points de ^{2} qui converge vers *.

En pratique, il faut choisir un moyen d'arrêter l'algorithme. On parle alors de **test** ou **critère d'arrêt** de l'algorithme. Ce test d'arrêt devra être choisi pour garantir que l'algorithme s'arrête toujours après un nombre fini d'itérations et que le dernier point calculé soit *suffisamment* proche de *. En d'autre terme, pour une précision fixée notée $\varepsilon > 0$, l'algorithme doit s'arrêter à une certaine étape (itération) k* de manière que $\mathbf{x}_{k*}$ soit une $\varepsilon-$ approximation de la solution $\mathbf{x}$*, *c.-à-d.* $\|\mathbf{x}_{k*} -^* \| \leqslant \varepsilon$. Cependant, il est claire qu'en pratique, étant donné que la solution * n'est pas connue, on ne peut pas évaluer la quantité $\|\mathbf{x}_{k*} -^* \|$ et par conséquent considérer ce test d'arrêt. Il faut donc, choisir un critère d'arrêt que l'on peut calculer (évaluer) numériquement à chaque itération.

Pas conséquent, on modélise (mathématiquement) un critère d'arrêt par une fonction $\Delta :^N \rightarrow^+$

- pour laquelle on doit pouvoir l'évaluer à la $k-$ème itération, *c.-à-d.* calculer $\Delta(_k)$;
- qui garantit la précision de la solution, *c.-à-d.*

$$\Delta(_k) \leqslant \varepsilon \;\Rightarrow\; \|x_k - x^*\| \leqslant \varepsilon. \tag{17}$$

5. Dans cette question nous allons donner deux interprétations du critère dit du résidu.

a. Les "résidus" sont les différences entre les valeurs que l'on observe et les valeurs estimées par notre modèle de regression. Ils représentent donc la partie que l'on n'explique pas grâce à notre équation de

régression.

Le résidu d'un vecteur $u \in^N$ dans un système $Ax = b$ est noté : $r(u) = Au - b$
Avec x^* la solution, nous avons :

$$r(x^*) = Ax^* - b = 0$$
$$\Leftrightarrow r(x_k) = Ax_k - b = r_k$$

Le critère du résidu est noté :

$$\Delta(x_k) = \|r_k\|$$

b. Pour la résolution du problème de minimisation quadratique, le critère de résidus concerne l'erreur entre la valeur du minimiseur $c*$ et le minimiseur théorique. Pour le problème équivalent de résolution du système linéaire inversible $Ax = b$, le critère concerne l'écart entre $x*$ et le théorique.

Nous terminons ce paragraphe, par une étude de la **qualité** du critère d'arrêt du résidu. Un critère d'arrêt Δ sera dit de *bonne* qualité si et seulement si il permet de garantir la précision ε fixé de la solution approchée, *c.-à-d.* si l'implication (17) soit vérifiée. Cette étude s'applique à l'ensemble des méthodes de types itératives succeptibles de résoudre le problème (G)

6. Nous allons montrer que :

$$\forall k \in \mathbb{N}, \quad \frac{\|x_k - x^*\|}{\|x^*\|} \leqslant \mathrm{cond}_2(A) \ \frac{\|\mathbf{r}_k\|}{\|\mathbf{b}\|}, \tag{18}$$

où $\mathbf{r}_k$ désigne le résidu associé au système linéaire $A =$ de l'itéré $_k$.
Nous reprenons l'équation de la majoration de l'erreur relative :

$$\frac{\|x_k - x^*\|}{\|x^*\|} \leqslant \mathrm{cond}_2(A) \ \frac{\|\delta b\|}{\|\mathbf{b}\|}, \tag{19}$$

En prenant en compte le résidu r_k dans notre problème, nous avons d'abord l'égalité : $Ax_k = b + r_k$
Nous avons donc : $r_k = \|Ax_k - b\|$

Nous posons ensuite : $\|x_k - x^*\| = \|^*\|$

Nous pouvons donc réecrire l'équation de la majoration de l'erreur relative comme suit :

$$\forall k \in \mathbb{N}, \quad \frac{\|x_k - x^*\|}{\|x^*\|} \leqslant \mathrm{cond}_2(A) \ \frac{\|\mathbf{r}_k\|}{\|\mathbf{b}\|}, \tag{20}$$

7. Nous avons :

$$\|r_k\| \ \leqslant \epsilon$$

Donc

$$\frac{\|r_k\|}{\|b\|}$$

$$\Leftrightarrow \mathrm{cond}_2(A) \ \frac{\|\mathbf{r}_k\|}{\|\mathbf{b}\|} \leqslant \epsilon$$

$$\Leftrightarrow \frac{\|x_k - x^*\|}{\|x^*\|} \leqslant \epsilon$$

A partir de notre égalité, nous pouvons dire que le critère est de mauvaise qualité car, en effet, nous pouvons voir que notre écart relatif est majoré par une valeur de conditionnement établie auparavant, et qui est très élevée. Ainsi, nous pouons dire que nous avons une approximation très peu précise.
Par ailleurs, si nous conditionnons bien le problème et qu'on respecte le critère d'arrêt, alors nous aurons une approximation plus précise.

3.1.4 Schéma général

Nous pouvons maintenant décrire cette classe d'algortihme de manière présise.

Méthode à directions de descente

— **Objectif** Calculer une approximation $\tilde{tx}$ de l'unique solution optimale $\mathbf{x}^*$ du problème de minimisation (G).

— **Entrée**
− Données sur le problème (G) : la matrice symétrique définie positive $A \in \mathcal{M}_2(\mathbb{R})$, le vecteur $\in^2$.
− La précision $\varepsilon > 0$ associé au critère d'arrêt.

— **Sortie** Une $\epsilon-$approximation [1] $\tilde{\mathbf{x}}$ de $\mathbf{x}^*$.

— **Initialisation** $k = 0$.

— **Itération** Nous supposons qu'au début de l'itération k, on dispose d'un itéré $x_k \in^2$.

(a) Test d'arrêt : Si $\|\mathbf{r}_k\| = \|Ax_k - b\| \leqslant \varepsilon$, alors $\tilde{4x} := x_k$.

(b) Calcul de la direction d_k de descente de f en x_k :

$$\text{déterminer } d_k \text{ tel que } \ D_{x_k} f(x_k) = \nabla f(x_k)^T d_k = (Ax_k - b)^T d_k < 0. \tag{21}$$

(c) Calcul du pas α_k de descente de f au point x_k suivant d_k :

$$\text{déterminer } \alpha_k \text{ tel que } \ f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) < f(\mathbf{x}_k). \tag{22}$$

(d) Calcul du nouveau itéré $\mathbf{x}_{k+1}$:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k. \tag{23}$$

Dans la suite de ce projet nous allons étudier 3 algorithmes : L'algorithme du gradient à pas fixe et pas optimal, et l'algorithme du gradient conjugué.

3.2 Méthode de descente du gradient ou de plus forte descente

Dans cette sous-section, dans le contexte de résolution du problème d'ajustement linéaire nous allons expliciter, implémenter et tester numériquement les deux méthodes de descentes dite de plus forte descente suivante : 1) algorithme de descente du gradient à pas fixe ; 2) algorithme de descente du gradient à pas optimal.

3.2.1 Direction de plus forte descente

Comme son nom l'indique, c'est la direction de descente $_k$ (vérifie (21)) qui indique au point courant $_k$ de l'algorithme 3.1.4 la direction de plus forte descente (pente négative) :

$$\forall d \in^2 \ \text{ tel que } \ \|d\| = \|d_k\|, \quad d_k^T \nabla f(x_k) \leqslant d^T \nabla f(x_k). \tag{24}$$

1. **a.** d_k est une direction de descente de f au point x_d donc :

$$D_d k f(x_k) = \nabla f(x_k)^T d_k < 0$$

$$\Leftrightarrow \|\nabla f(x_k)\| \|d_k\| cos(\nabla f(x_k), d_k) < 0$$

Nous devons avoir $cos(\nabla f(x_k), d_k) = -1$ et donc comme le cosinus est compris dans [-1,1], nous devrons avoir un angle de 180°.
Ainsi, $d_k = \nabla f(x_k)$.

1. $\tilde{\mathbf{x}}$ est une $\epsilon-$approximation de $\mathbf{x}^*$ si et seulement si $\|\tilde{\mathbf{x}} - \mathbf{x}^*\| \leqslant \epsilon$.

b. Nous allons montrer que :

$$\forall d \in^2 \text{ tel que } \|d\| = \| - \nabla f(x_k)\|, \quad d^T \nabla f(x_k) \leqslant \|\nabla f(x_k)\|^2 \tag{25}$$

Nous savons que :

$$\|d\| = \| - \nabla f(x_k)\|$$

et,

$$\|d^T\| = \|d\|$$

Donc :

$$D_d f(x_k) = \nabla f(x_k)^T d$$

$$\Leftrightarrow \nabla f(x_k)^T d = -\nabla f(x_k)^T \nabla f(x_k)$$

$$\Leftrightarrow d^T \nabla f(x_k) \leqslant \| - \nabla f(x_k)^T \nabla f(x_k)\|$$

$$\Leftrightarrow d^T \nabla f(x_k) \leqslant \| - \nabla f(x_k)^T \| \|\nabla f(x_k)\|$$

$$\Leftrightarrow d^T \nabla f(x_k) \leqslant \|\nabla f(x_k)\|^2$$

Nous retrouvons donc bien notre expression de départ.

c. D'après la question précédente, nous savons que :

$$d^T \nabla f(x_k) \leqslant \|\nabla f(x_k)\|^2$$

Etant donné que d_k est une direction de plus forte descente :

$$d_k^T \nabla f(x_k) = \nabla f(x_k)^T d_k \neq 0$$

Ainsi :

$$d_k^T \nabla f(x_k) \neq d^T \nabla f(x_k)$$

3.2.2 L'algorithme de descente à gradient à pas fixe

Comme son nom l'indique, ses itérations sont définies par les choix suivants :

- Choix du calcul de la direction de descent : *méthode de gradient* signifie de choisir la direction de plus profonde descente à chaque itération ;
- Choix du calcul du pas de descente : *à pas fixe* signifie de choisir le même pas (constant) à chaque itération.

Nous allons expliciter la méthode dite de gradient à pas fixe dans le contexte de résolution du problème d'optimisation quadratique (Q).

2. La construction de l'algorithme est la suivante :
(a) Fixer la condition d'arrêt avec le critère de résidu
(b) Expliciter la direction de descente d_k
(c) Expliciter le pas fixe α_k
(d) Calculer le k+1 itéré selon $x_{k+1} = x_k - \alpha r_k$

Il apparaît clairement dans l'algorithme du gradient à pas fixe, qu'il est nécessaire de spécifier le choix du pas de descente fixe $\alpha_k := \alpha$ afin qu'à chaque itération l'algorithme fasse décroître la valeur de la fonction, *c.-à-d.* le pas fixe α vérifie pour toutes les itérations la condition (22). C'est l'objet de la question ci-dessous.

3. **a.** Pour montrer que : si le réel α vérifie la condition

$$\forall k \in \mathbb{N}, \quad 0 < \alpha < \tilde{\alpha}_k := \frac{2\|\mathbf{r}_k\|^2}{\mathbf{r}_k^T X^T X \mathbf{r}_k}, \tag{26}$$

alors l'algorithme du gradient à pas fixe α est bien une méthode à direction de descente.

Nous voulons donc que $\alpha > 0$ et que : $f(x_k + \alpha d_k) < f(x_k)$

Comme $0 < \alpha_k < \tilde{\alpha_k}$, nous avons :

$$\alpha \in]0, \frac{-2(Ax_k - b)^T d_k}{d_k^T A d_k}[$$

De plus, nous avons $Ax_k - b = r_k$, donc :

$$\tilde{\alpha_k} = \frac{-2r_k^T d_k}{d_k^T A d_k}$$

Enfin, nous avons $A = X^T X$, $D_k = -r_k$ et $\|r_k\|^2 = r_k^T r_k$

Donc nous retrouvons bien

$$\forall k \in \mathbb{N}, \quad 0 < \alpha < \tilde{\alpha}_k := \frac{2\|\mathbf{r}_k\|^2}{\mathbf{r}_k^T X^T X \mathbf{r}_k}, \tag{27}$$

b. Nous cherchons à montrer que :

$$\forall k \in \mathbb{N}, \quad \tilde{\alpha}_k := \frac{2\|\mathbf{r}_k\|^2}{\mathbf{r}_k^T X^T X \mathbf{r}_k} \geqslant \frac{2}{\lambda_2}. \tag{28}$$

Comme $X^T X$ est une matrice symétrique définie positive , on a :

$$\lambda_1 \|r_k\|^2 \leqslant r_k^T X^T X r_k \leqslant \lambda_2 \|r_k\|^2$$
$$\Leftrightarrow r_k^T X^T X r_k \leqslant \lambda_2 \|r_k\|^2$$

En transformant notre équation, nous avons :

$$\frac{2\|r_k\|^2}{r_k^T X^T x r_k} \leqslant \frac{2\|r_k\|^2}{\lambda_2 \|r_k\|^2} \leqslant \frac{2}{\lambda_2}$$

Nous retrouvons bien ce que nous cherchons.

c. Nous avons bien démontré que $\forall k \in$, on a :

$$0 < \alpha_k < \tilde{\alpha_k} et < \tilde{\alpha_k} \geqslant \frac{2}{\lambda_2}$$

En conclusion :

$$0 < \alpha < \frac{2}{\lambda_2}$$

4. **a.** Afin de coder cette fonction nous avons utilisé les pseudo codes donnés en page 28 de l'énoncé. La première partie du code (en dehors de la boucle while) vise à initialiser et créer les différentes variables nécessaires au bon fonctionnement.
Dans le pseudo-code la valeur de i va de 1 à iMax cependant les lignes et colonnes des matrices et vecteurs Numpy commencent à 0 c'est pourquoi on initialise les premiers éléments au rang 0. Nous rentrons ensuite dans la boucle while de l'algorithme qui va se réaliser tant que la solution n'a pas atteint une certaine

```python
def gradientfix(A,b,X,tol,rho):
    """Fonction de la méthode du gradient à pas fixe"""

    i=1
    xit=[]
    xit.append(X)
    r=[]
    d=[]
    iMax=50000
    r.append(A@xit[0]-b)
    d.append(-r[0])
    xit.append(xit[0]+rho*d[0])
    while(np.linalg.norm((r[i-1]))>tol and i<iMax):
        r.append(A@xit[i]-b)
        d.append(-r[i])
        xit.append(xit[i]+rho*d[i])
        sol=xit[i]
        i=i+1
        nit=i
    return sol,xit,nit
```

FIGURE 3 – Code pour la méthode gradient à pas fixe

tolérance ou que le nombre d'itérations max ait été atteint. A chaque itération on incrémente i et on enregistre la dernière solution dans une liste. Enfin, nous retournons à l'utilisateur : la solution finale qui est le dernier élément de la liste des solutions, la liste des solutions et i le nombre d'itérations.
Nous obtenons alors le code suivant :

b. Nous fixons les différents paramètres donnés par l'énoncé et on créer le vecteur x0 ici appelé X. Puis on appelle la fonction créée plus haut, nous affichons la solution et le nombre d'itérations qui ont été nécessaires pour obtenir une précision de 10^{-6}.

```python
################test des methodes pour la resolution du probleme#####################

tol=10**-6
X=np.array([-9,-7])
rho=0.001
solGPF,xitGPF,nitGPF=gradientfix(A,b,X,tol,rho)
print("\nSolution gradient a pas fixe : ",solGPF,", nombre d'iterations:",nitGPF)
```

Nous obtenons alors :

```
Solution gradient a pas fixe :  [0.75016235 0.0638812 ] , nombre d'iterations: 3368
```

Ainsi, nous retrouvons bien la bonne solution au problème à 10^{-6} près et cela pour 3368 itérations.

c. Si nous paramètrons ρ à 10^{-1} nous obtenons une erreur de type overflow ce qui indique que la solution prend des valeurs de l'ordre de 10308 pour python ce qui se trouve être très loin de notre solution finale. Ainsi, nous en déduisons que le pas est trop grand pour l'algorithme. En prenant un pas égal à 10^{-5} la boucle s'arrête à la 50000ème itération puisque nous avons atteint le iMax défini dans la fonction du gradient à pas fixe. Ceci arrive car le pas est trop petit ainsi l'algorithme n'atteint jamais le point critique. En changeant le nombre iMax à 500 000 itérations nous trouvons qu'avec un pas de 10^{-5} il aurait fallu 337478 itérations.

```
Solution gradient a pas fixe :  [0.75016235 0.0638812 ] , nombre d'iterations: 337478
```

3.2.3 L'algorithme de descente du gradient à pas optimal

5. Les étapes de l'algorithme sont les suivantes :
   (a) Fixer le critère d'arrêt
   (b) Expliciter la direction de descente $d_k = -r_k$
   (c) Fixer le pas de descente α_k
   (d) Mettre en place l'itération $x_{k+1} = x_k - \alpha_k r_k$

6. **a.** Pour coder cette fonction nous utilisons la fonction GradientFixe écrite précédemment. Cependant cette fois-ci nous retirons le paramètre du pas de la fonction puisque celui-ci va varier tout au long de l'exécution. Dans la fonction elle-même nous définissons ρ qui va être recalculé à chaque itération afin de réduire considérablement le nombre d'itérations total. La fonction fonctionne de la même manière que la précédente, d'abord une partie hors de la boucle while afin d'initialiser les paramètres puis la boucle principale où se déroule le reste de la fonction.
   Nous obtenons alors le code suivant :

```
def gradientoptimal(A,b,X,tol):
    """Fonction de la méthode du gradient à pas optimal"""

    i=1
    xit=[]
    xit.append(X)
    r=[]
    d=[]
    rho=[]
    iMax=50000
    r.append(A@xit[0]-b)
    d.append(-r[0])
    rho.append((np.transpose(r[0])@r[0])/(np.transpose(r[0])@A@r[0]))
    xit.append(xit[0]+rho[0]*d[0])
    while(np.linalg.norm((r[i-1]))>tol and i<iMax):
        r.append(A@xit[i]-b)
        d.append(-r[i])
        rho.append((np.transpose(r[i])@r[i])/(np.transpose(r[i])@A@r[i]))
        xit.append(xit[i]+rho[i]*d[i])
        sol=xit[i]
        i=i+1
        nit=i
    return sol,xit,nit
```

FIGURE 4 – Code pour la méthode gradient à pas optimal

b. Nous exécutons la fonction et nous affichons le vecteur solution ainsi que le nombre d'itérations :

```
solGPO,xitGPO,nitGPO=gradientoptimal(A,b,X,tol)
print("\nSolution gradient a pas optimal: ",solGPO,", nombre d'iterations:",nitGPO)
```

Nous obtenons alors la réponse suivante :

```
Solution gradient a pas optimal:  [0.75016254 0.06388117] , nombre d'iterations: 10
```

c. Finalement, nous pouvons voir que nous obtenons une valeur plus précise que celle obtenue grâce à la méthode du gradient à pas fixe. De plus, le résultat est obtenu beaucoup plus rapidement puisqu'il n'a fallu que 10 itérations pour obtenir ce résultat. Ainsi nous pouvons en déduire que la vitesse de convergence de cette méthode est bien plus rapide que celle du gradient à pas fixe.

3.3 L'algorithme des gradients conjugués

Dans cette sous-section, dans le contexte de résolution du problème d'ajustement linéaire (**??**), nous allons explicité, implémenter et tester numériquement l'algorithme des gradients conjugués.

3.3.1 Direction du gradient conjugué

1. Pour $k = 0$, nous avons :

   $$d_0 = -\nabla f(x_0)$$

   Or, par définition, la direction du gradient d est, en réalité, l'opposé du gradient : $d = -\nabla f(x)$. Ce qui nous montre bien que d_0 est une direction de descente au point x_0.

2. **a.** Nous cherchons à montrer que le pas de descente optimal α_{k-1} de f au point courant $_{k-1}$ suivant direction $_{k-1}$ est donné par :

   $$\alpha_{k-1} = -\frac{\mathbf{r}_{k-1}^T k-1}{\overset{T}{k-1} A_{hk-1}}. \tag{29}$$

   Nous savons déjà que :

   $$\alpha_k^* = \frac{-(Ax_k - b)^T d_k}{d_k^T A d_k} \tag{30}$$

   Pae ailleurs, $r_k = Ax_k - b$, en remplaçant dans l'équation, nous avons :

   $$\alpha_k^* = \frac{-r_k^T d_k}{d_k^T A d_k} \tag{31}$$

   Ainsi, pour une direction de descente d_{k-1}, nous avons :

   $$\alpha_{k-1}^* = \frac{-r_{k-1}^T d_{k-1}}{d_{k-1}^T A d_{k-1}} \tag{32}$$

   b. Nous pouvons d'abord dire que $r_k = x_k - b$ car $d_k = -\nabla f(x_k) = -r_k$

   Nous pouvons aussi dire que :

   $$x_{k+1} = x_k + \alpha_k d_k$$

   $$\Leftrightarrow x_k = x_{k-1} + \alpha_{k-1} d_{k-1}$$

   Nous avons donc :

   $$r_k = Ax_k - b$$

   $$\Leftrightarrow r_k = A(x_{k-1} + \alpha_k -_1 d_{k-1}) - b$$

   $$\Leftrightarrow r_k = Ax_{k-1} + A\alpha_k -_1 d_{k-1} - b$$

   Or, nous savons que $A_{x-1} - b = r_{k-1}$

   Ainsi, nous avons :

   $$r_k = r_{k-1} + \alpha_{k-1} A d_{k-1}$$

c. Nous posons

$$d_k = -rk \Leftrightarrow d_k^T = -rk^T \Leftrightarrow r_k^T = -dk^T \Leftrightarrow r_k^T d_{k-1} = -dk^T d_k$$

Les directions de descente sont perpendiculaires les unes aux autres, nous avons ainsi :

$$r_k^T d_{k-1} = 0$$

d. Une direction de descente de f au point x_k est définie par $f(x_k + \lambda d_k) < f(x_k)$
Nous notons aussi que $x_{k+1} = x_k + _k$.
e. Nous avons $d_k = -r_k$ et donc aussi $d_{k-1} = -r_{k-1}$. Comme les résidus sont orthogonaux, nous pouvons en conclure que $r_k^T r_{k-1} = 0$
Ainsi l'orthogonalité des résidus nous permet de conclure que d_k est bien une direction de descente.

3. Nous avons montré que le gradient a un pas α_{k-1} qui suit une direction d_{k-1} et qui est strictement négatif. De plus nous avons aussi démontré que les directions de descente sont perpendiculaires les unes des autres et que les résidus sont orthogonaux entre eux. Enfin on a montré que :

$$r_k = r_{k-1} + \alpha_{k-1} d_{k-1}$$

Nous pouvons ainsi dire que le gradient conjugué est bien une méthode à directions de descentes.

3.3.2 Pseudo-code et test numérique

4. **a.** Nous cherchons à montrer que le coefficient de conjugaison β_{k-1} entre les directions d_{k-1} et d_k est défini par :

$$\beta_{k-1} = \frac{\mathbf{r}_k^T A_h d_{k-1}}{d_{k-1}^T A_h d_{k-1}}. \tag{33}$$

Les directions d_k sont des combinaisons linéraires entre la directtion de la plus forte pente $-r - k = -\nabla f(x_k)$ et la direction de d_{k-1} :

$$d_k = -r_k = \beta_k d_{k-1}$$

Où β_{k-1} est choisi tel que la conjugaison entre d_k et d_{k-1} soit vérifiée. Nous multiplions ensuite les deux côtés par $d_{k-1}A$, nous obtenons alors :

$$d_k d_{k-1}^T A = -r_k d_{k-1}^T A + \beta_{k-1} d_{k-1} d_{k-1}^T A$$

Or, $d_k A^T d_{k-1} = 0$ et donc $dd_{k-1} = d_k A^T d_{k-1}$.
Finalement, nous avons :

$$-r_k d_{k-1}^T A + \beta_{k-1} d_{k-1} d_{k-1}^T A = 0 \Longleftrightarrow \beta_{k-1} d_{k-1} d_{k-1}^T A = r_k d_{k-1}^T A$$

$$\Longleftrightarrow \beta_{k-1} d_{k-1} A^T d_{k-1} = r_k A^T d_{k-1}$$

$$\Longleftrightarrow \beta_{k-1} = \frac{r_k A^T d_{k-1}}{d_{k-1} A^T d_{k-1}}$$

$$\Longleftrightarrow \beta_{k-1} = \frac{r_k^T A d_{k-1}}{d_{k-1}^T A d_{k-1}}$$

Or $A = X^T X$, nous obtenons alors :

$$\beta_{k-1} = \frac{r_k^T (X^T X) d_{k-1}}{d_{k-1}^T (X^T X) d_{k-1}}$$

Nous obtenons bien le coefficient β_{k-1} recherché.

b. La construction de l'algorithme est la suivante :
(a) Fixer la condition d'arrêt avec le critère de résidu
(b) Expliciter le pas de descente d_{k-1}

(c) Calculer la direction de descente x_k
(d) Calculer le coefficient de conjugaison β_{k-1}
(e) Calculer la nouvelle direction de descente d_{k-1}

5. **a.** Pour cette fonction nous utilisons la base de la fonction GradientOptimal. Cependant nous créons aussi le paramètre β. La fonction fonctionne de la même manière que la précédente, d'abord une partie hors de la boucle while afin d'initialiser les paramètres puis la boucle principale où se déroule le reste de la fonction.

   Nous obtenons ainsi le code suivant :

```
def gradientconj(A,b,X,tol):
    """Fonction de la méthode du gradient à pas conjugué"""

    i=1
    xit=[]
    xit.append(X)
    r=[]
    d=[]
    rho=[]
    beta=[]
    iMax=50000
    r.append(A@xit[0]-b)
    d.append(-r[0])
    rho.append((np.transpose(r[0])@r[0])/(np.transpose(d[0])@A@d[0]))
    xit.append(xit[0]+rho[0]*d[0])

    while(np.linalg.norm((r[i-1]))>tol and i<iMax):
        r.append(A@xit[i]-b)
        beta.append((np.linalg.norm(r[i])**2)/(np.linalg.norm(r[i-1])**2))
        d.append(-r[i]+beta[i-1]*d[i-1])
        rho.append((np.transpose(r[i])@r[i])/(np.transpose(d[i])@A@d[i]))
        xit.append(xit[i]+rho[i]*d[i])
        sol=xit[i]
        i=i+1
        nit=i

    return sol,xit,nit
```

FIGURE 5 – Code pour la méthode gradient à pas conjugué

b. Nous exécutons la fonction et nous affichons le vecteur solution ainsi que le nombre d'itérations :

```
solGC,xitGC,nitGC=gradientconj(A,b,X,tol)
print("\nSolution gradient conjugué : ",solGC,", nombre d'iterations:",nitGC)
```

Nous obtenons alors la réponse suivante :

```
Solution gradient conjugué :  [0.75016254 0.06388117] , nombre d'iterations: 3
```

c. Ainsi nous pouvons voir que nous obtenons les mêmes valeurs que celles obtenues grâce à la méthode du gradient à pas optimal. Cependant le résultat est obtenu beaucoup plus rapidement puisqu'il n'a fallu que 3 itérations pour l'obtenir. Ainsi nous en déduisons que la vitesse de convergence de cette méthode est bien plus rapide que celles des deux autres méthodes.

3.4 Analyse des résultats numériques

L'objectif de cette sous-section est de commenter *numériquement* les comportements des trois méthodes : descente du gradient à pas fixe, à pas optimal et des gradients conjugués. Pour cela, pour chacune des méthodes : 1) on trace la (les) trajectoire(s) générée(s) (il y a autant de trajectoires que de points initiaux choisis) ; 2) on compte le nombre d'itérations (on peut aussi mesurer le temps CPU) ; 3) on donne une représentation du modèle *calculé* vis-à-vis des données. Bien évidement, si vous le désirez vous pouvez ajouter des critères de comparaison. Vous y êtes même encouragé !

3.4.1 Les résultats

1. Trajectoires générées par les méthodes
   b. Les courbes de niveaux sont les suivantes :

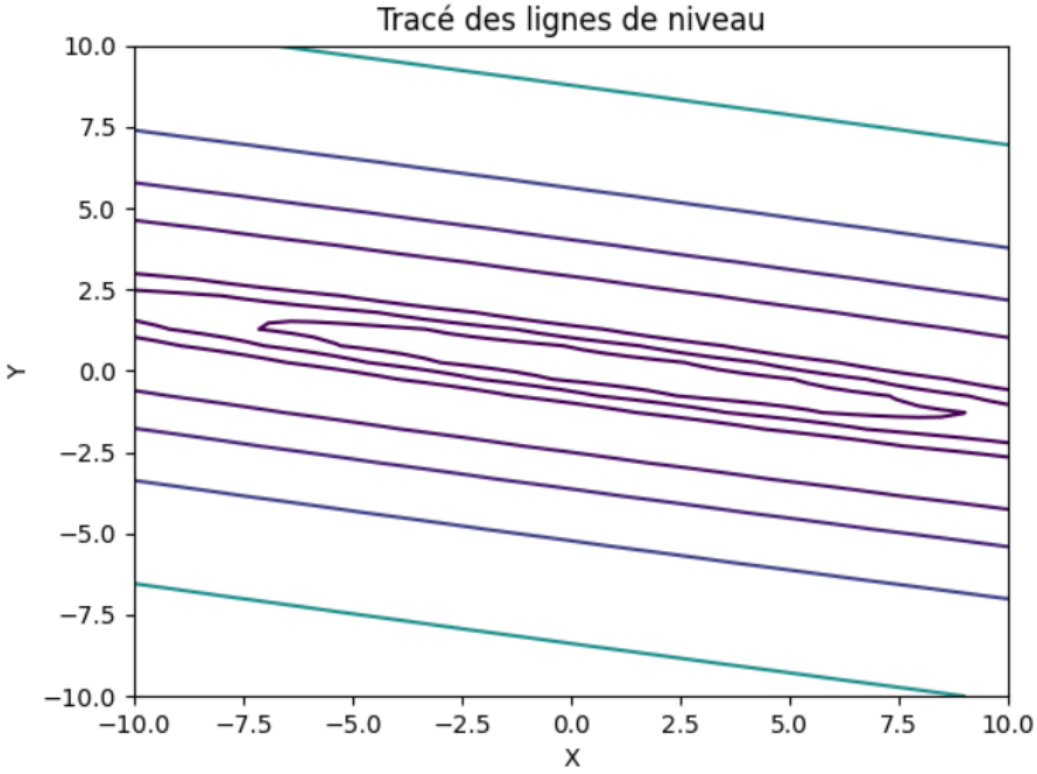


FIGURE 6 – Code pour la méthode gradient à pas conjugué

Bien que cela reste difficilement visible, nous pouvons voir de façon graphique que le point critique de la fonction se situe aux alentours de l'origine ce qui correspond aux résultats obtenus de manière numérique ($[0.750162540.06388117]$) :

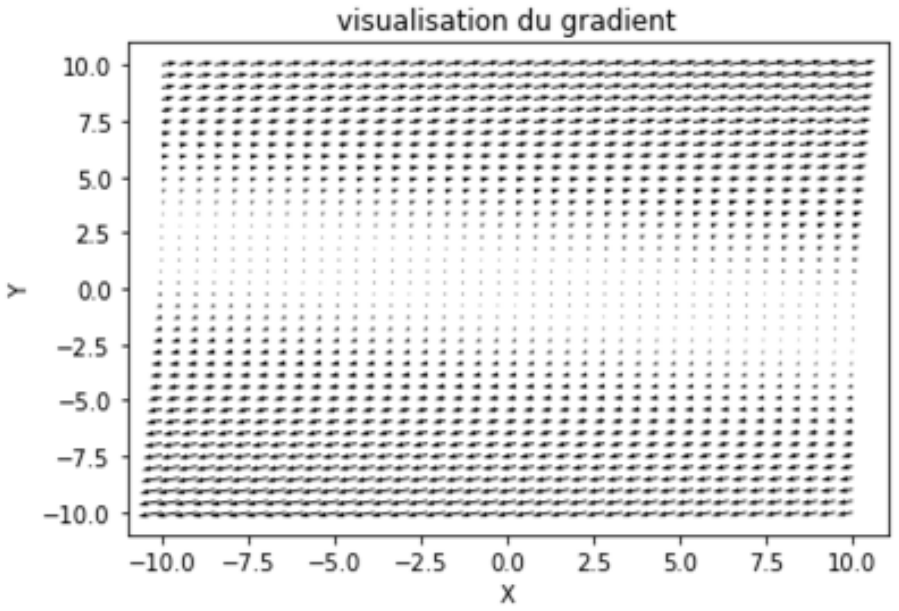


FIGURE 7 – Visualisation du gradient

c. Ici nous réalisons la même opération sur les trois algorithmes. Nous utilisons la liste des solutions obtenues puis nous traçons chaque point sur le graphique des lignes de niveau et enfin nous les relions afin d'avoir la trajectoire.
Nous avons le code suivant :

```
############ Tracé des lignes de niveau et de la trajectoire reliant les iterations a la solution#############
########pour le gradiant a pas fixe#####################
fig, ax = plt.subplots()
CS = ax.contour(X, Y, Z)
ax.set_title('Tracé des lignes de niveau et de la trajectoire des itérations')
ax.set_xlabel('X')
ax.set_ylabel('Y')
xs = [x[0] for x in xitGPF]
ys = [x[1] for x in xitGPF]
plt.plot(xs, ys,'--r')
plt.show()


############ Tracé des lignes de niveau et de la trajectoire reliant les iterations a la solution#############
##########pour le gradiant a pas optimal#############
fig, ax = plt.subplots()
CS = ax.contour(X, Y, Z)
ax.set_title('Tracé des lignes de niveau et de la trajectoire des itérations')
ax.set_xlabel('X')
ax.set_ylabel('Y')
xs = [x[0] for x in xitGPO]
ys = [x[1] for x in xitGPO]
plt.plot(xs, ys,'--r')
plt.show()

############ Tracé des lignes de niveau et de la trajectoire reliant les iterations a la solution#############
#########pour le gradiant conjugué#############
fig, ax = plt.subplots()
CS = ax.contour(X, Y, Z)
ax.set_title('Tracé des lignes de niveau et de la trajectoire des itérations')
ax.set_xlabel('X')
ax.set_ylabel('Y')
xs = [x[0] for x in xitGC]
ys = [x[1] for x in xitGC]
plt.plot(xs, ys,'--r')
plt.show()
```

FIGURE 8 – Code pour afficher les trajectoires

Nous obtenons alors ces graphiques :
Tout d'abord pour la méthode du gradient à pas fixe :

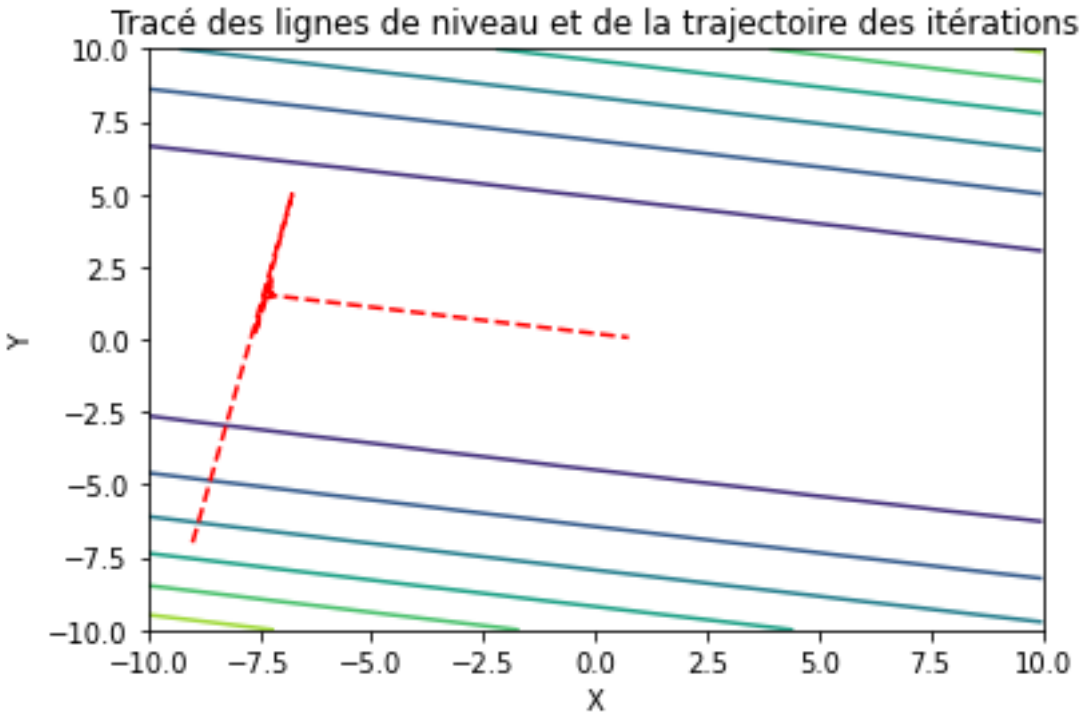


FIGURE 9 – Trajectoire pour la méthode du gradient à pas fixe

Via cette méthode nous pouvons voir graphiquement que celle-ci génère plus de points. En effet nous observons qu'un grand nombre de « rebonds » ont été fait avant que la trajectoire n'atteigne le point critique.

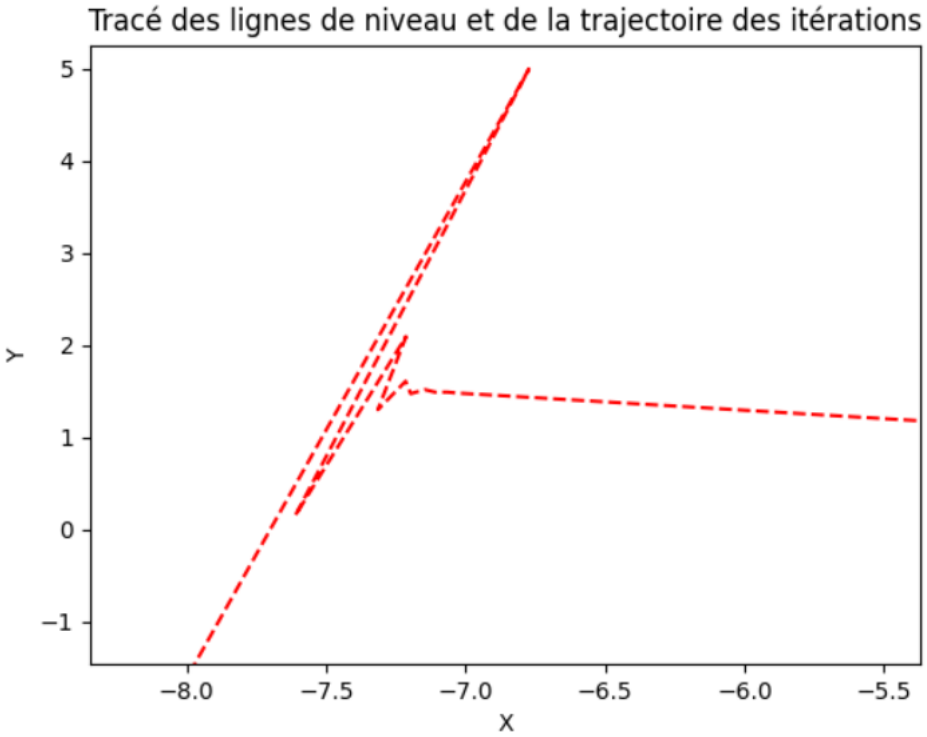


FIGURE 10 – Allers retours de la trajectoire

Nous essayons maintenant de visualiser la trajectoire pour la méthode du gradient à pas optimal :

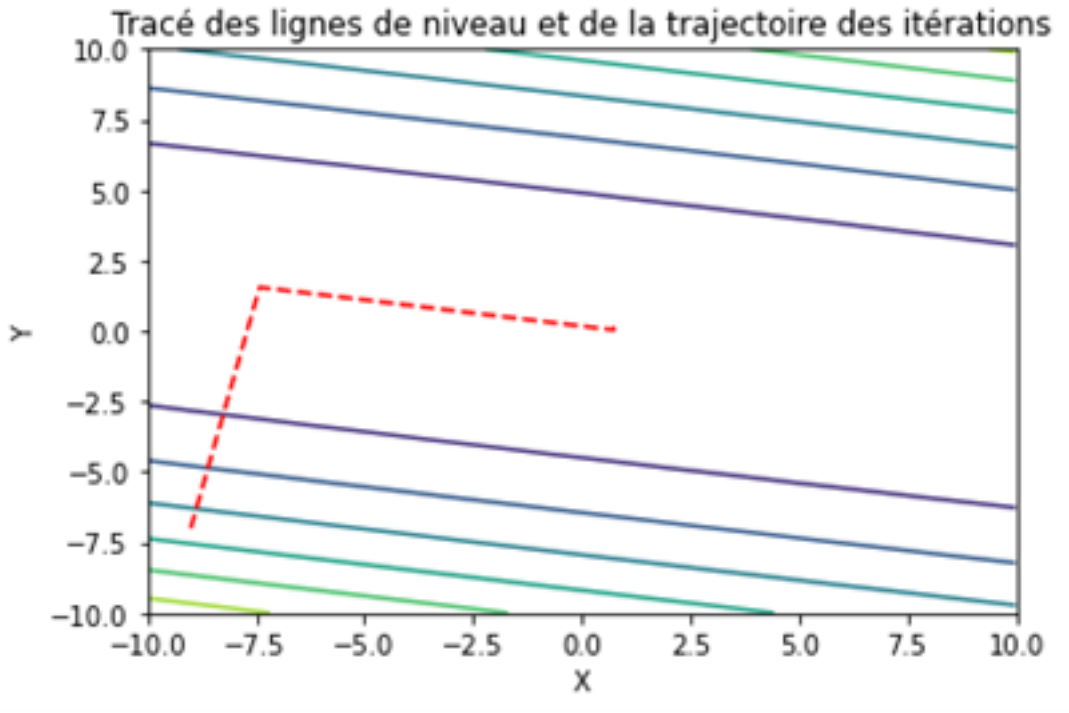


FIGURE 11 – Trajectoire pour la méthode du gradient à pas optimal

Pour cette méthode, nous pouvons voir que la trajectoire est beaucoup plus claire puisqu'elle nécessite moins d'itérations. De plus, si nous observons bien on remarque que les derniers changements de trajectoires sont très proches de la solution finale.
Pour finir, nous testons le code avec la méthode du pas conjugué, ce qui nous donne :

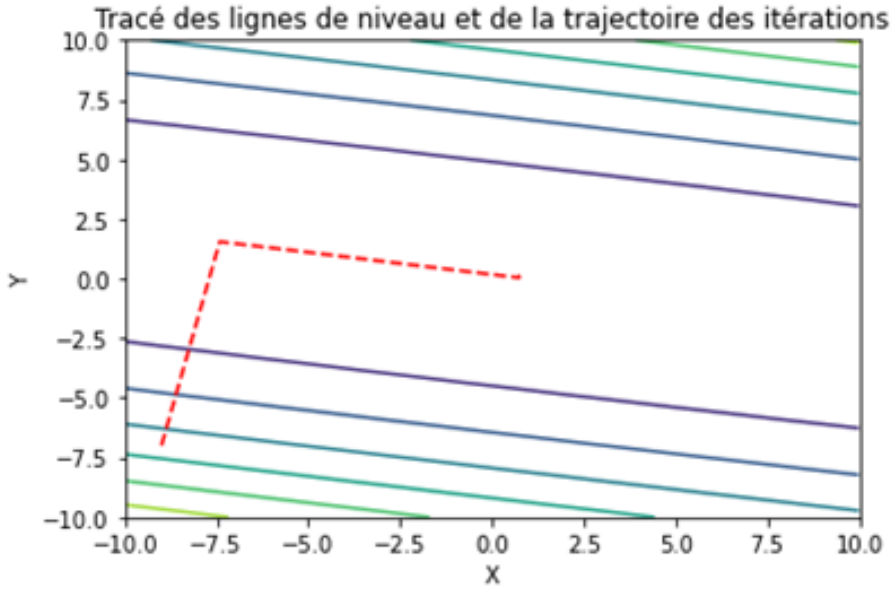


FIGURE 12 – Trajectoire pour la méthode du gradient à pas conjugué

Nous pouvons constater que le graphique est quasiment identique à celui décrivant la méthode du gradient à pas optimal. Ceci est dû à leur nombre d'itérations relativement faible pour arriver au point critique. Cependant, la trajectoire du gradient à pas conjugué est encore plus efficace puisqu'elle n'est constituée que de trois points : celui de départ, un point intermédiaire et le point critique.
Nous pouvons observer cette différence en zoomant sur les trajectoirs.
En effet, nous pouvons observer un léger rebond à la fin de la trajectoire pour la méthode du gradient à

pas optimal, tandis qu'il n'est pas présent sur la méthode à pas conjugué.

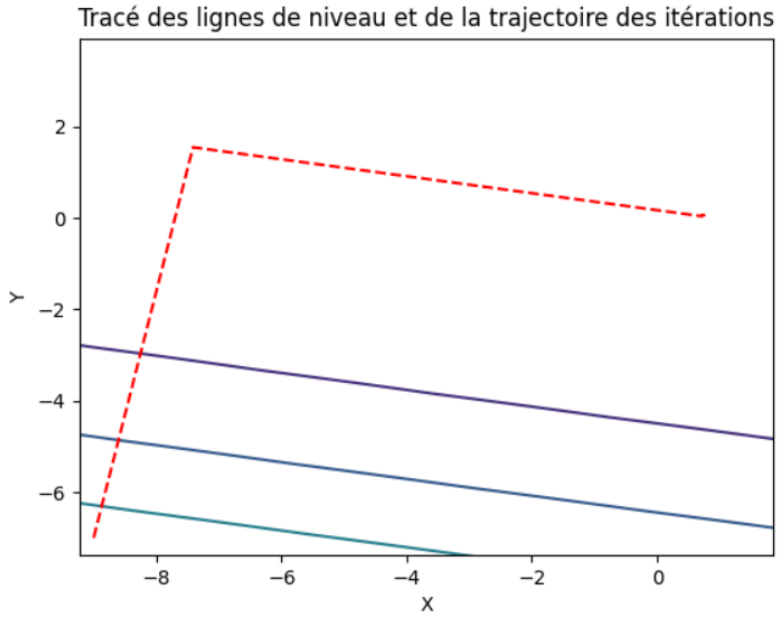


FIGURE 13 – Zoom de la trajectoire pour la méthode du gradient à pas optimal

Nous pouvons clairement constaté que la méthode du gradient à pas optimal ne suit pas une ligne droite dans les dernières itérations :

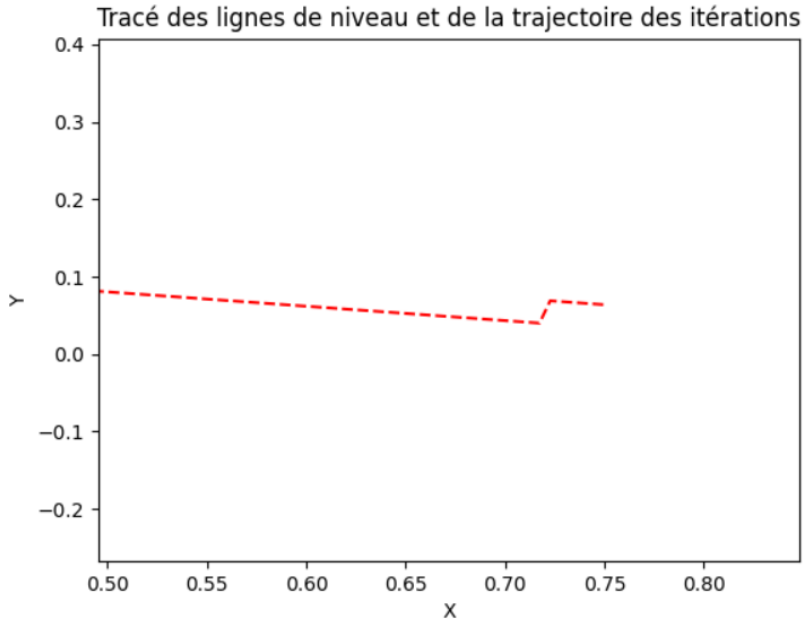


FIGURE 14 – Zoom plus grand de la trajectoire pour la méthode du gradient à pas optimal

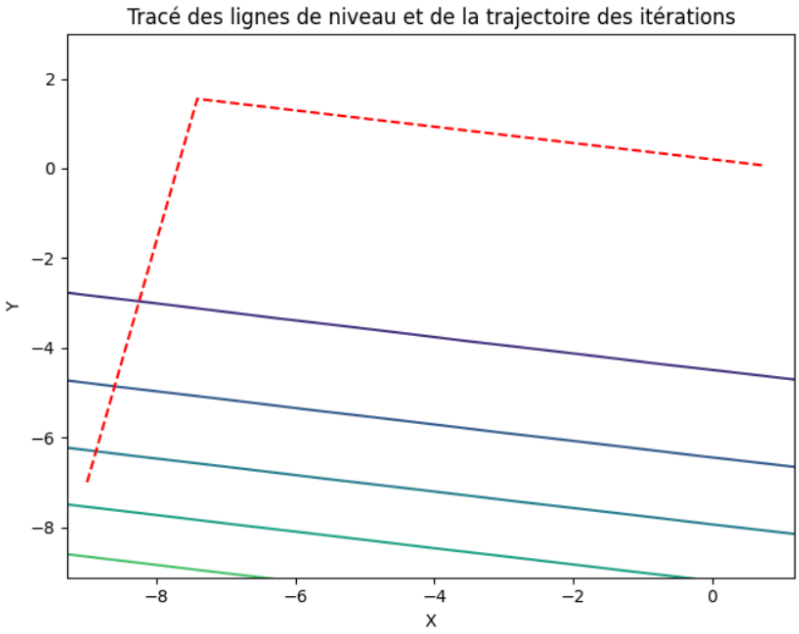


FIGURE 15 – Zoom de la trajectoire pour la méthode du gradient à pas conjugué

2. Visualisation des modèles linéaires calculés

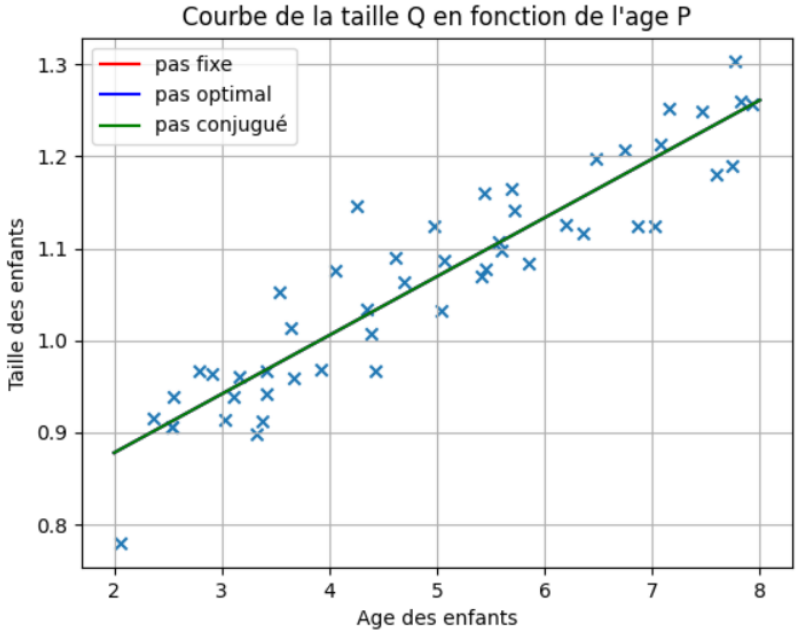


FIGURE 16 – Représentation linéaire des trois méthodes

En traçant les droites obtenues nous pouvons voir que celles-ci se superposent. Ceci est normal puisque les 3 algorithmes nous rendent un résultat avec la même tolérance d'erreur. Ainsi nous pouvons voir que les trois méthodes nous donnent le bon résultat et par conséquent amènent à la même droite de régression. Cependant, nous avons pu voir au long de ce projet que certaines méthodes sont bien plus rapides que d'autres en vitesse de convergence.

3. Convergence et *vitesse*
Nous avons pu ainsi voir que les trois méthodes convergeaient :

```
Solution gradient à pas fixe :  [0.75016235 0.0638812 ] , nombre d'itérations: 3368

Solution gradient à pas optimal:  [0.75016254 0.06388117] , nombre d'itérations: 10

Solution gradient conjugué :  [0.75016254 0.06388117] , nombre d'itérations: 3
```

FIGURE 17 – Point critique et nombre d'itérations de chaque méthode

3.4.2 Comportements des méthodes

Ainsi, les trois méthodes convergent, mais à des vitesse différentes. En effet, nous constatons que la méthode du gradient à pas fixe trouve le point critique au bout de 3368 itérations, la méthode du gradient à pas optimal au bout de 10 itérations tandis que la méthode du gradient conjugué nous amène au point critique au bout de seulement 3 itérations.

En effet, nous avons pu voir sur les graphiques précédent que la méthode du gradient à pas fixe a besoin de beaucoup d'itérations pour arriver au point fixe :

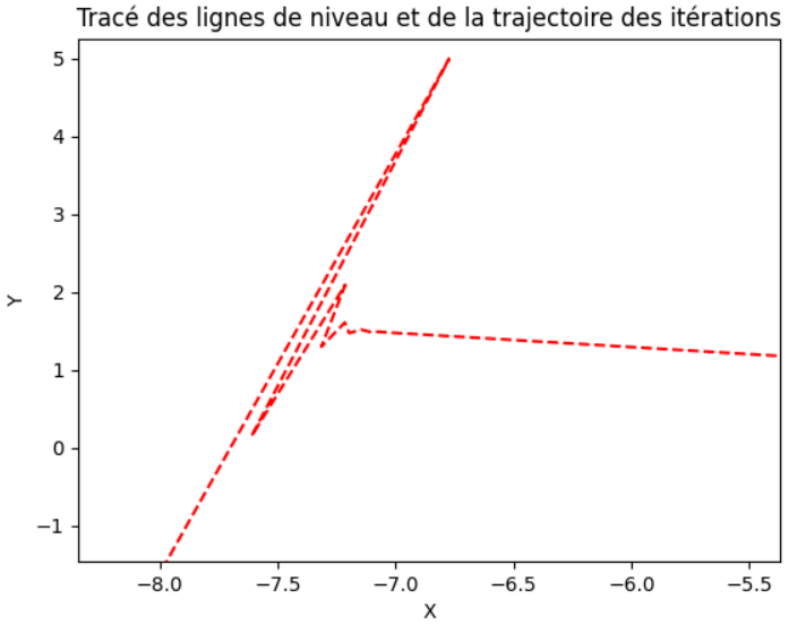


FIGURE 18 – Trajectoire de la méthode du gradient à pas fixe

Pour la méthode du gradient à pas optimal, nous constatons qu'elle est plutôt efficace car elle permet d'atteindre rapidement le point critique.

En ce qui concerne la méthode du gradient à pas conjugué, nous pouvons affirmer qu'il s'agit de la méthode la plus efficace, car elle permet d'atteindre le résultat avec la même précision que les deux autres méthodes et ceci, en un temps plus faible.

En conclusion, nous pouvons en déduire que la première méthode est beaucoup moins efficace que les deux autres. Pour la méthode du gradient à pas otpimal et du gradient conjugué, nous pouvons constater qu'en terme d'efficacité, elles sont relativement proches. Cependant, la dernière méthode reste la plus efficace avec seulement 3 itérations pour atteindre le résultat souhaité.

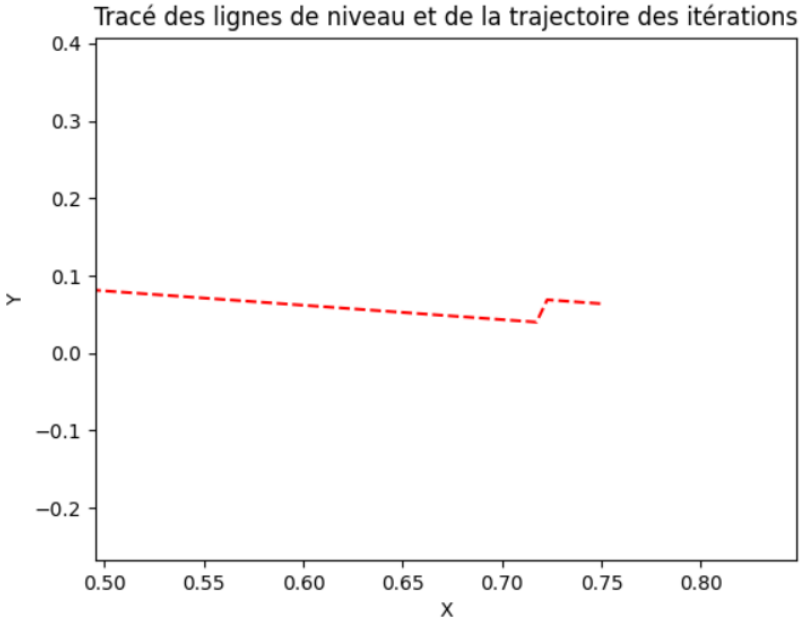


FIGURE 19 – Trajectoire de la méthode du gradient à pas optimal

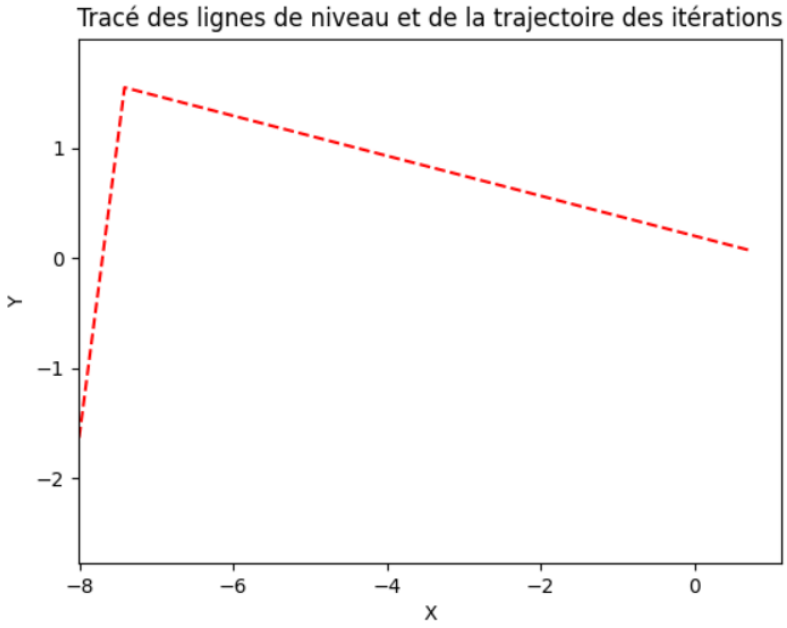


FIGURE 20 – Trajectoire de la méthode du gradient à pas conjugué

Par ailleurs, nous pouvons également calculer le temps d'exécution de chaque méthode afin de vérifier laquelle est la plus rapide :

```
Solution gradient à pas fixe :  [0.75016235 0.0638812 ] , nombre d'itérations: 3368
temps d'exécution de la méthode à pas fixe :  0.04231090000000037 sec

Solution gradient à pas optimal:  [0.75016254 0.06388117] , nombre d'itérations: 10
temps d'exécution de la méthode à pas optimal :  0.000244500000000869 sec

Solution gradient conjugué :  [0.75016254 0.06388117] , nombre d'itérations: 3
temps d'exécution de la méthode à pas conjugué :  0.00014640000000021303 sec
```

FIGURE 21 – Temps d'exécution de chaque méthode

Nous constatons que la méthode du gradient à pas conjugué est bien la plus rapide en terme d'exécution.