document

Minimisation d'une fonctionnelle quadratique

Kelvin LEFORT Selim LIMAM

Contents

1 Introduction

1.1 Description du problème

Soient A une matrice symétrique définie positive de taille n et b un vecteur de taille n. On considère la fonctionnelle quadratique

$$f(x) = \frac{1}{2}(Ax, x) - (b, x)$$

On rappelle que $\nabla f(x) = Ax - b$ et $(f''(x)h, h) = (Ah, h)$.
Le problème que nous nous posons est celui de la minimisation de cette fonctionnelle.

1.2 Existence et unicité du point de minimum

La première question qu'on se pose face à ce type de problème est celle de l'existence d'une solution et si possible de son unicité.
Pour ce faire, on peut montrer que f est α-convexe, pour un certain $\alpha > 0$.
En effet, en appliquant le théorème spectral (A est symétrique), on montre que

$$(Ah, h) \geqslant \alpha \|h\|_2^2$$

avec $\alpha = (\min_{j=1,\dots,n} \lambda_j)$, où les λ_j sont les valeurs propres de A (comptées avec leur multiplicité).
Mais puisque A est en plus définie positive, alors toutes ses valeurs propres sont strictement positives, i.e. $\alpha > 0$.
Remarquons aussi que f est continue en tant que fonctionnelle quadratique.
Ainsi, par le théorème d'existence et d'unicité (en dimension finie ou infinie), f admet un unique point de minimum, qu'on note x^*.

1.3 Caractérisation du point de minimum

Maintenant qu'on sait que x^* existe et est unique, la prochaine question qu'on se pose dans ce type de problème est comment caractériser cette solution.
La condition d'optimalité du premier ordre nous indique que $\nabla f(x^*) = 0$, autrement dit $Ax^* - b = 0$, c'est-à-dire $x^* = A^{-1}b$ (A est symétrique définie positive, donc inversible).

Remarque 1 *Le point de minimum de f est la solution du système linéaire*

$$Ax = b$$

Ainsi, minimiser f est équivalent à résoudre ce système linéaire.

1.4 Objectif des algorithmes

Dans ce cas bien précis (celui d'une fonctionnelle quadratique), l'utilisation des algorithmes n'est pas nécéssaire car on connait la solution. Néanmoins, dans le cas général, on peut savoir que la fonctionnelle admet un point de minimum (et si possible qu'il est unique) grâce à des théorèmes d'existence et d'unicité sans pouvoir le caractériser. C'est ici qu'intervient les algorithmes. Leur but est construire une suite de points qui converge vers un/le point de minimum (local ou global). Ainsi, en allant suffisamment loin dans le calcul des itérés de la suite, on pourra avoir à disposition une "bonne" approximation de la solution.
Nous allons programmer trois de ces algorithmes (ce sera des algorithmes de

descente du gradient) et nous pourrons les tester avec f comme fonctionnelle. L'avantage ici est qu'on connait la solution exacte donc on peut effectuer des tests sur ces algorithmes.

2 Algorithmes

On prendra $A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$, $x^* = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ et donc $b = Ax^* = \begin{pmatrix} 0 \\ 3 \end{pmatrix}$ pour tester les différents algorithmes.
De plus, on prendra $tolerance$ = eps et $itermax = 1000$ (voir les codes pour comprendre leurs significations).

2.1 Méthode du gradient à pas constant

2.1.1 Description de la méthode

Soient $\rho > 0$ et $x_0 \in \mathbb{R}^n$. La méthode du gradient à pas constant consiste à construire la suite $(x_k)_{k\in\mathbb{N}}$ de la manière suivante

$$x_{k+1} = x_k - \rho\nabla f(x_k), \forall k \in \mathbb{N}$$

2.1.2 Test de la méthode

Pour tester cette méthode, on a pris $\rho = 0,1$ et $x_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.
On constate alors que l'algorithme converge en 331 itérations.
Veuillez trouver ci-dessous une représentation graphique de f ainsi que la suite des itérés $(x_k, f(x_k))$.

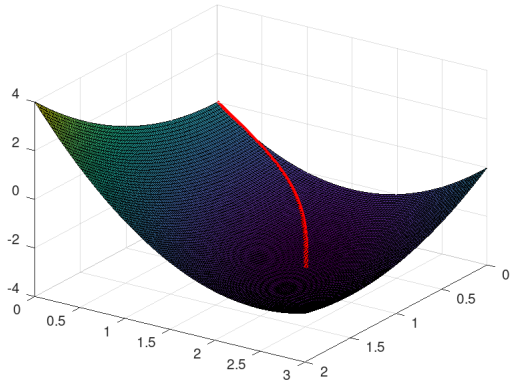

Figure 1: Représentation graphique de f et de la suite des itérés $(x_k, f(x_k))$

On observe bien que la suite "descend" vers le minimum de f, d'où le nom de ce type d'algorithme.

2.1.3 Etude de la convergence

On se demande alors pour quelles valeurs de ρ et x_0 cette méthode converge.

Théorique Pour tout $k \in \mathbb{N}$, on pose $e_k = x_k - x^*$ et pour tout $M \in \mathcal{M}_n(\mathbb{R})$, $r_\sigma(M)$ désigne le rayon spectral de M.
On montre alors que pour tout $k \in \mathbb{N}$

$$e_{k+1} = (I - \rho A)e_k$$

D'après un résultat d'analyse numérique, la suite $(e_k)_{k\in\mathbb{N}}$ converge vers 0 si et seulement si $r_\sigma(I-\rho A) < 1$. Il suffit donc d'étudier la fonction $\rho \longmapsto r_\sigma(I-\rho A)$.

Commençons par calculer les valeurs propres de $I - \rho A$.
La matrice A étant symétrique, il existe une matrice P orthogonale et une matrice D diagonale telles que

$$D = P^T AP$$

Donc

$$I - \rho D = P^T (I - \rho A) P$$

Ainsi, en notant λ_i les valeurs propres de A, les valeurs propres de $I - \rho A$ sont données par $1 - \rho\lambda_i$.
Pour les tests, on a pris $A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$. Ses valeurs propres sont 1 et 3 et donc les valeurs propres de $I - \rho A$ sont $1 - \rho$ et $1 - 3\rho$. D'où $r_\sigma(I - \rho A) = \max\{|1 - \rho|, |1 - 3\rho|\}$. La fonction $\rho \longmapsto r_\sigma(I - \rho A)$ est donc simple à étudier.

Voici sa représentation graphique.

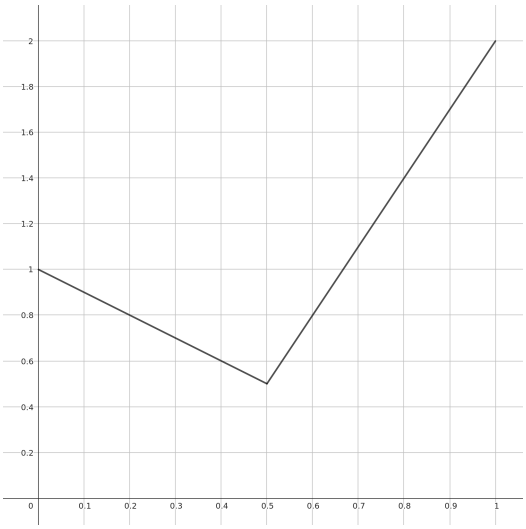

Figure 2: Courbe représentative de la fonction $\rho \longmapsto r_\sigma(I - \rho A)$ sur $[0, 1]$

On constate alors que $r_\sigma(I - \rho A) < 1$ si et seulement si $\rho \in]0, \frac{2}{3}[$. Ainsi, la méthode converge si et seulement si $\rho \in]0, \frac{2}{3}[$ et le ρ optimal est $\frac{1}{2}$ (ρ pour lequel $r_\sigma(I - \rho A)$ est minimal). On précisera la raison pour laquelle on dit que ce ρ est optimal dans la section Numérique. On remarque donc que la convergence de cette méthode ne dépend que de ρ et pas de x_0.

Numérique On rappelle que pour tout $k \in \mathbb{N}$, $e_{k+1} = (I - \rho A)e_k$. Donc

$$||e_{k+1}||_2 \leqslant |||(I - \rho A)|||_2 ||e_k||_2$$

Mais puisque A est symétrique, $I - \rho A$ est aussi symétrique, en particulier normale, donc $|||(I - \rho A)|||_2 = r_\sigma(I - \rho A)$. D'où

$$||e_{k+1}||_2 \leqslant r_\sigma(I - \rho A)||e_k||_2$$

La convergence est donc linéaire. Ceci nous amène à définir le taux de convergence numérique (avec N fixé suffisamment grand)

$$r_{num}(\rho) = \frac{||e_N||_2}{||e_{N-1}||_2}$$

Remarquons que plus ce taux est faible, meilleure est la vitesse de convergence et que ce taux est majoré par $r_\sigma(I - \rho A)$.
Voici une représentation graphique de ce taux pour ρ allant de 0 à 1 par pas de $0,01$ et $N = 30$.
On constate que cette représentation graphique coincide avec celle de $r_\sigma(I - \rho A)$. Donc la majoration du taux de convergence numérique par $r_\sigma(I - \rho A)$ est optimale et donc que l'algorithme converge le plus rapidement pour $\rho = \frac{1}{2}$ (d'où l'appellation du ρ optimal).

2.2 Méthode du gradient à pas optimal

2.2.1 Description de la méthode

Soit $x_0 \in \mathbb{R}^n$. Dans le cas quadratique, on montre aisément que la méthode du gradient à pas optimal consiste à construire la suite $(x_k)_{k\in\mathbb{N}}$ de la manière suivante

$$x_{k+1} = x_k - \rho_k \nabla J(x_k), \forall k \in \mathbb{N}$$

avec $\rho_k = \dfrac{\|\nabla J(x_k)\|_2^2}{(A\nabla J(x_k), \nabla J(x_k))}$.

On peut aussi montré que cette méthode converge pour toute donnée initiale x_0.

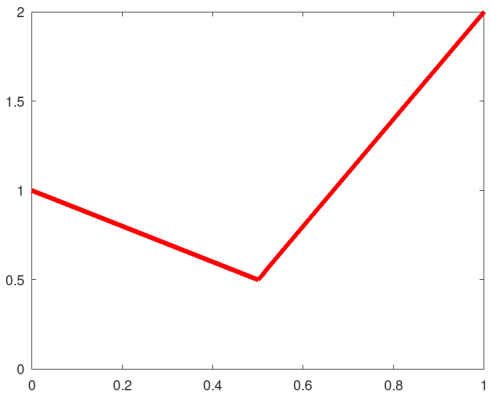

Figure 3: Taux de convergence numérique pour ρ allant de 0 à 1 par pas de $0,01$

2.2.2 Test de la méthode

Pour tester cette méthode, on a pris $x_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.
On constate alors que l'algorithme converge en 55 itérérations.
Voici une représentation graphique de f ainsi que la suite des itérés $(x_k, f(x_k))$.

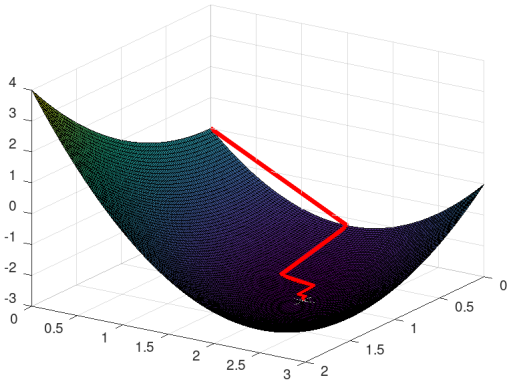

Figure 4: Représentation graphique de f et de la suite des itérés $(x_k, f(x_k))$

On observe bien que la suite "descend" vers le minimum de f tout comme la méthode du gradient à pas constant mais de façon orthogonale (on a montré ce résultat en TD).

2.3 Méthode du gradient conjugué

2.3.1 Description de la méthode

Soit $x_0 \in \mathbb{R}^n$. Dans le cas quadratique, on a montré en TD que la méthode du gradient conjugué consiste à construire la suite $(x_k)_{k \in [\![0, N_{\max}]\!]}$ (pour un certain $N_{\max} \in [\![1, n]\!]$) de la manière suivante

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

avec

$$\begin{cases} g_0 &= Ax_0 - b \\ g_{k+1} &= g_k + \rho_k A d_k \end{cases}$$

$$\begin{cases} d_0 &= -g_0 \\ d_{k+1} &= -g_{k+1} + \frac{(g_{k+1}, g_{k+1})}{(g_k, g_k)} d_k \end{cases}$$

$$\begin{cases} \rho_0 &= -\frac{(g_0,d_0)}{(d_0,Ad_0)} \\ \rho_{k+1} &= -\frac{(g_{k+1},d_{k+1})}{(d_{k+1},Ad_{k+1})} \end{cases}$$

Ainsi construit, $x_{N_{\max}}$ est le point de minimum de f. La suite converge donc en un nombre fini d'itérations, ce qui est un avantage majeur.

2.3.2 Test de la méthode

Pour tester cette méthode, on a pris $x_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.
L'algorithme converge ici en 2 itérations.
Voici une représentation graphique de f ainsi que la suite des itérés $(x_k, f(x_k))$.

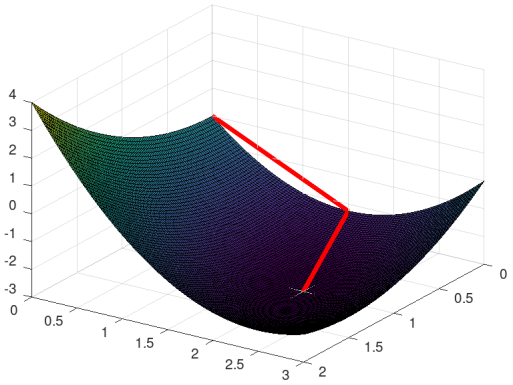

Figure 5: Représentation graphique de f et de la suite des itérés $(x_k, f(x_k))$

2.4 Comparaison des méthodes

Maintenant qu'on a implémenté et testé ces 3 algorithmes, on souhaite les classer pour savoir qui est "meilleur" que qui (en prenant $\rho = \frac{1}{2}$ pour la méthode du gradient à pas constant). Nous chercherons ici un moyen statistique pour classer ces algorithmes.

2.4.1 Idée pour comparer

Pour ce faire, on prendra N valeurs de x_0 choisis aléatoirement (avec N grand), on testera ces algorithmes pour chacun des x_0 et on stockera :

- le nombre d'itérations dans une matrice $nbIter$ de 3 lignes (correspondant aux 3 algorithmes) et de N colonnes (correspondant aux N valeurs de x_0)

On dira qu'un algorithme A_1 est meilleur qu'un algorithme A_2 au sens du nombre d'itérations si la moyenne du nombre d'itérations de A_1 est supérieur à A_2.

2.4.2 Conclusion

Grâce à ces deux relations transitives, on peut maintenant classer les algorithmes.
En prenant $N = 1000$, on a trouvé que la méthode du gradient conjugué est melleure que la méthode du gradient à pas optimal qui est meilleure que la méthode du gradient à pas constant, et ce au sens du nombre d'itérations.

Ainsi, de façon plutôt objective, on peut dire que la meilleure méthode est celle du gradient conjugué, ensuite il s'agit de celle du gradient à pas optimal, et enfin celle du gradient à pas constant.

D'une manière plus visuelle, voici une représentation graphique de l'erreur en fonction du nombre d'itérations pour chacun des algorithmes (pour $x_0 = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$).

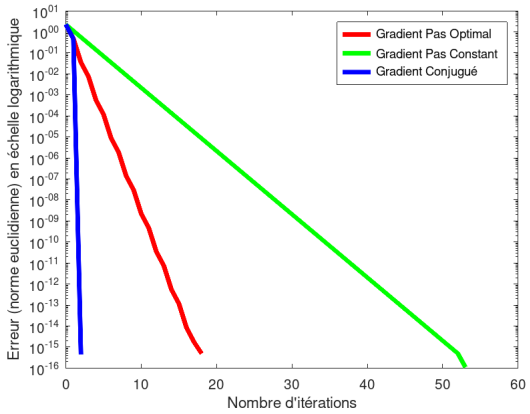

Figure 6: Erreur (norme euclidienne) en fonction du nombre d'itérations pour $x_0 = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$

On aboutit à la même conclusion (et même mieux, l'erreur pour la méthode du gradient conjugué est toujours inférieure à celle pour la méthode du gradient à pas optimal qui est toujours inférieure à celle pour la méthode du gradient à pas constant, et ce pour n'importe quelle itération).

2.4.3 Axes d'amélioration

Un problème se pose. On a mené une comparaison des algorithmes pour une matrice A et un vecteur b prédéfini. On peut alors se demander si le classement dépend de ces derniers. Si ça se trouve, pour certaines matrices A de très grande dimension, il est plus judicieux d'utiliser la méthode du gradient à pas optimal ou celle à pas constant plutôt que celle conjugué. Ce serait intéressant d'étudier ceci.
On élimine la méthode du gradient à pas constant. En effet, les pas pour lesquels il y a convergence dépendent de la matrice A, ce qui est embêtant en pratique (il faudrait déterminer les pas qui permettent d'avoir la convergence, ce qui est assez fastidieux). Il nous reste les deux autres méthodes.
Prenons par exemple la matrice de discrétisation du Laplacien en une dimension, qui est très utile en pratique, on l'utilise notamment lors de la résolution numérique d'EDP par des schémas aux différences finis (on peut montrer qu'elle est symétrique définie positive).
On remarque que, pour $n = 100$, pour $x_0 = (0, ..., 0) \in \mathbb{R}^n$, et pour $x^* = (1, ..., 1) \in \mathbb{R}^n$ par exemple, on a :

- l'erreur initiale vaut 10.
- l'erreur à la 20000^e itération vaut environ $0,000563648$ pour la méthode du gradient à pas optimal.
- l'erreur à la 51^e itération vaut environ $1,11 \times 10^{-14}$ pour la méthode du gradient conjugué.

On observe alors que la méthode du gradient à pas optimal prend "beaucoup de temps" à converger (j'ai essayé pour un nombre plus élevé d'itérations pour la méthode du gradient à pas optimal mais mon ordinateur n'a pas apprécié...), ce qui pose problème (on a pris $n = 100$ seulement).
On pourrait faire ce test pour différentes valeurs de n, de x_0, et de x^* et arriver au même problème pour la méthode du gradient à pas optimal.
Ainsi, il est bien plus commode d'utiliser la méthode du gradient conjugué.

3 Ouverture vers un autre problème

Une des applications de l'optimisation est **l'apprentissage**. On est souvent amené en apprentissage à trouver les "meilleurs" paramètres, c'est-à-dire ceux qui minimisent une certaine fonction. C'est le cas du problème (2) et le problème alternatif (3), décris sur la feuille **optimisation : algorithme Adaboost et optimisation**.
Pour résoudre le problème (3) (on peut facilement montrer que ce problème admet une unique solution), nous implémentons la méthode de relaxation et une variante (qu'on appellera relaxation modifiée). Nous remarquons alors que l'algorithme de relaxation modifiée nécessite au moins deux fois plus d'itérations que celui de relaxation non modifiée pour converger. Ce n'est pas très étonnant, c'est l'algorithme de relaxation non modifiée qu'on a étudié en cours.