Introduction à l'optimisation

Deuxième Partie : aspects numériques

Rozenn Texier-Picard *

novembre 2021

Table des matières

*rozenn.texier@ens-rennes.fr

1 Introduction

Dans toute cette partie, on se limitera à des problèmes de minimisation en dimension finie. Notons que des problèmes posés en dimension infinie peuvent souvent être approchés par des problèmes de dimension finie après discrétisation.

Cadre : $K \subset \mathbb{R}^N$ fermé non vide (ensemble des points admissibles)
$J : K \to \mathbb{R}$ fonction coût ou critère, différentiable.

But : trouver (ici, approcher) u vérifiant

$$\begin{cases} u \in K \\ J(u) = \inf_{v \in K} J(v) \end{cases} \tag{1}$$

Si $K = \mathbb{R}^N$, on parle de minimisation sans contrainte.

Exemples

Les problèmes de minimisation sont omniprésents dans les applications :

- en physique, chimie, biologie, rechercher un équilibre est souvent équivalent à minimiser une énergie ; → ex : public2009-B1.pdf (forme d'une molécule)
- en économie, de nombreux modèles conduisent à minimiser un risque, ou un coût (ou maximiser un profit) → ex : public2018-B5.pdf

Citons d'autres exemples liés à des problématiques mathématiques.

- Résoudre des systèmes linéaires symétriques définis positifs ; en effet, si $A \in S_N^{++}(\mathbb{R})$ et $b \in \mathbb{R}^N$, alors résoudre le système $Ax = b$ est équivalent à minimiser la fonction

$$J(y) = \frac{1}{2}\langle Ay, y\rangle - \langle b, y\rangle.$$

  (En effet, J est convexe et $\nabla J(y) = Ay - b, \forall y \in \mathbb{R}^N$.)
- Résolution d'un système d'équations au sens des moindres carrés, par exemple pour un problème d'identification de paramètres. En effet, les modèles physiques conduisent parfois à des systèmes d'équations de la forme $f_i(x_1, \cdots x_p) = 0$, $1 \leq i \leq n$. Les imprécisions sur les données peuvent conduire à ce que ces systèmes n'aient pas de solution exacte. On cherche alors à les résoudre au sens des moindres carrés, i.e. on cherche $\bar{x} \in \mathbb{R}^p$ qui minimise la fonction

$$J(x) = \sum_{i=1}^{n} (f_i(x))^2$$

- Calcul de valeurs propres d'une matrice symétrique (cf théorèmes de Rayleigh-Ritz et Courant-Fischer).

2 Algorithmes de minimisation sans contrainte

Dans cette partie, $K = \mathbb{R}^N$ et J est différentiable partout, ce qui, du point de vue des applications, est souvent vérifié. Pour garantir l'existence et l'unicité d'une solution, et obtenir facilement la convergence des algorithmes, on se place dans le cas (très restrictif!) où J est fortement convexe (on dit aussi α-convexe), au sens de la définition ci-dessous.

Définition 2.1. Forte convexité

Soit V un espace de Hilbert, $\langle \cdot, \cdot \rangle$ un produit scalaire sur V. Soit $K \subset V$ un convexe. Une fonction $f : K \to \mathbb{R}$ est dite fortement convexe ou uniformément convexe ou α-convexe ou α-elliptique s'il existe $\alpha > 0$ t.q. :

$$\forall (x, y) \in K^2, \forall t \in [0, 1],\ f(tx + (1-t)y) \leq tf(x) + (1-t)f(y) - \frac{\alpha}{2} t(1-t)\|x - y\|^2.$$

Remarques (admises provisoirement, voir prochain cours) :
- Ce cas inclut en particulier le cas des fonctions quadratiques à matrice symétrique définie positive.
- On a alors existence et unicité de la solution u au problème (1), et elle est caractérisée par

$$\nabla J(u) = 0.$$

On veut trouver un algorithme itératif pour approcher la solution u. Pour toute cette partie, on pourra consulter les livres [1, 2, 3]. Les références [4, 5, 6] sont proposées en complément ou pour les preuves de certains résultats plus pointus qui ne sont pas détaillés ici.

2.1 Méthode de Newton (ou Newton-Raphson)

Ce n'est pas à proprement parler une méthode d'optimisation, mais elle peut permettre de déterminer les points critiques du critère. Dans le cas convexe, on obtient exactement les points de minimum.

Rappel : supposons qu'on veuille résoudre numériquement un système d'équations non-linéaires de la forme $f(u) = 0$, où $f : \mathbb{R}^N \to \mathbb{R}^N$ est de classe C^2. On suppose que, dans un ouvert donné, la solution u existe, est unique, et vérifie que $Df(u)$ est inversible. On peut alors approcher u par l'algorithme de Newton, qui s'écrit sous forme condensée :

$$\begin{cases} x^0 \in \mathbb{R}^N, \\ x^{n+1} = x^n - (Df(x^n))^{-1} f(x^n). \end{cases}$$

En pratique, l'algorithme se présente ainsi :

Algorithme 2.2. Méthode de Newton

- *Démarrage.* Choisir $x^0 \in \mathbb{R}^N$, « suffisamment proche » de la solution ;
- *Étape k.*
  - Calculer $A^k = Df(x^k)$
  - Résoudre $A^k v^k = f(x^k)$
  - Poser $x^{k+1} = x^k - v^k$.

Si J est α-convexe et de classe C^3, on peut appliquer cet algorithme à $f = \nabla J$ pour déterminer le point critique. On a alors convergence vers la solution u, *si* u^0 *est assez proche de* u*...*

- **Avantages :** convergence quadratique, i.e. très rapide en pratique,
- **Inconvénients :** coût de calcul (déterminer et inverser la matrice hessienne !), résultat de convergence *local* seulement, hypothèses restrictives... Dans le cas non convexe, l'algorithme peut converger vers un point critique qui n'est pas un minimum !
- **En pratique :** cet algorithme est utilisable en petite dimension et pour des fonctions assez simples. Des variantes existent pour réduire le coût de calcul (méthodes de quasi-Newton) mais la convergence quadratique est perdue. Ces méthodes sont cependant très utilisées dans les applications car elles restent performantes.

2.2 Méthodes de descente : idée générale

Algorithme 2.3. Méthodes de descente : algorithme général

- *Démarrage.* On se donne u^0 arbitraire,
- *Étape* ($n \geq 0$). u^n étant supposé connu, on définit
  - une direction de descente $d^n \neq 0$ (le choix dépendra de la méthode),
  - le pas optimal associé μ^n défini par

$$J(u^n - \mu^n d^n) = \inf_{\mu \in \mathbb{R}} J(u^n - \mu d^n).$$

(i.e. on résout un problème de minimisation en une dimension, appelé "recherche linéaire").
- On pose alors $u^{n+1} = u^n - \mu^n d^n$.

Définition 2.4. Convergence d'une méthode de descente

On dira que la méthode *converge* si, pour tout choix de $u^0 \in \mathbb{R}^N$, la suite (u^n) ainsi construite converge vers u, solution du problème (1).

Dans toutes les méthodes de descente ci-après, les résultats de convergence seront montrés dans le cadre de l'hypothèse ci-dessous.

Hypothèse (H) : la fonction $J : \mathbb{R}^N \to \mathbb{R}$ sera supposée différentiable et fortement convexe ; on suppose de plus que le gradient de J est localement lipschitzien. Typiquement ces hypothèses sont vérifiées si J est une fonction quadratique associée à une matrice symétrique définie positive.

2.3 Méthode de relaxation

Elle consiste à choisir successivement les directions de descente parallèles aux différents axes, i.e. $d_n = e_{n \ mod \ N}$. C'est probablement l'algorithme le plus intuitif. En pratique, les premières étapes de l'algorithme se présentent sous la forme suivante :

Algorithme 2.5. Méthode de relaxation

$$
\begin{aligned}
u^0 &= (u_1^0, u_2^0, \cdots, u_N^0) \\
u^{1,1} &= (u_1^1, u_2^0, \cdots, u_N^0) \text{ tel que } J(u^{1,1}) = \min_{x\in\mathbb{R}} J(x, u_2^0, \cdots, u_N^0) \\
&\vdots \qquad\qquad \vdots \\
u^{1,N} &= (u_1^1, u_2^1, \cdots, u_N^1) \text{ tel que } J(u^{1,N}) = \min_{x\in\mathbb{R}} J(u_1^1, u_2^1, \cdots, x) \\
u^{2,1} &= (u_1^2, u_2^0, \cdots, u_N^0) \text{ tel que } J(u^{2,1}) = \min_{x\in\mathbb{R}} J(x, u_2^1, \cdots, u_N^1) \\
&\vdots \qquad\qquad \vdots
\end{aligned}
$$

Sous les hypothèses considérées, on peut montrer que l'algorithme converge (voir par exemple [1, 4]). Cependant, les directions des axes restent arbitraires et ne prennent pas en compte les spécificités de la fonction à minimiser. On peut donc s'attendre à ce que des algorithmes plus performants existent.

Dans le cas d'une fonction quadratique, on peut montrer que la méthode de relaxation est équivalente à la méthode de Gauss-Seidel pour résoudre le système linéaire associé.

2.4 Gradient à pas optimal

Elle consiste à choisir la direction de descente $d^n = \nabla J(u^n)$. Ce choix n'est pas anodin : localement, au voisinage de u^n, c'est la direction de plus forte pente, i.e. dans laquelle J croît le plus vite pour des pas μ infinitésimaux (ou décroît le plus vite si μ est négatif). L'algorithme prend la forme :

Algorithme 2.6. Gradient à pas optimal.

- *Démarrage.* On se donne u^0 arbitraire,
- *Étape n.* On calcule $r^n = \nabla J(u^n)$.
  - Si $r^n = 0$ alors $u^n = u$ et l'algorithme s'arrête.
  - Sinon, on calcule le pas optimal μ^n défini par

$$J(u^n - \mu^n r^n) = \inf_{\mu\in\mathbb{R}} J(u^n - \mu r^n).$$

  On pose alors $u^{n+1} = u^n - \mu^n r^n$.

Remarque 2.7 Caractérisation du pas optimal

Si on définit $f(\mu) = J(u^n - \mu r^n)$, alors f est fortement convexe, dérivable, donc admet un unique point de minimum μ^n qui est caractérisé par $f'(\mu^n) = 0$. Or on a

$$f'(\mu) = -\langle \nabla J(u^n - \mu r^n), r^n \rangle.$$

Donc $f'(\mu^n) = 0 = \langle r^{n+1}, r^n \rangle$. Ainsi, la condition d'optimalité donne l'orthogonalité de deux directions de descente consécutives.

Dans les cas simples, notamment pour une fonction quadratique, le calcul du pas optimal peut se faire explicitement. Voir page 7 pour le cas des fonctions quadratiques.

2.4.1 Convergence sous l'hypothèse (H)

Théorème 2.8. Convergence : gradient à pas optimal

Sous l'hypothèse (H), l'algorithme de gradient à pas optimal converge. De plus, on a l'estimation d'erreur

$$\|u^n - u\| \leq \frac{1}{\alpha}\|\nabla J(u^n)\|.$$

Preuve : (voir par exemple [1, 3]).

étape 1 : (u^n) est bornée.

Par construction, la suite $(J(u^n))_{n\in\mathbb{N}}$ est décroissante, et minorée, donc elle converge. Or J est coercive car fortement convexe. Ainsi, la convergence de $J(u^n)$ implique que la suite (u^n) est bornée : il existe donc $M > 0$ tel que

$$\forall n \in \mathbb{N}, \|u^n\| \leq M, \text{ et } \|u\| \leq M.$$

étape 2 : $u^{n+1} - u^n$ tend vers 0.

La fonction J étant fortement convexe, on peut écrire :

$$J(u^n) - J(u^{n+1}) \geq \langle \nabla J(u^{n+1}), u^n - u^{n+1}\rangle + \frac{\alpha}{2}\|u^{n+1} - u^n\|^2.$$

Or on a vu en remarque que $\langle \nabla J(u^{n+1}), u^n - u^{n+1}\rangle = \langle r^{n+1}, \mu^n r^n\rangle = 0$. D'où on déduit :

$$\|u^{n+1} - u^n\|^2 \leq \frac{2}{\alpha}(J(u^n) - J(u^{n+1})) \to 0.$$

étape 3 : $\nabla J(u^n)$ tend vers 0.

On a :

$$\begin{aligned}\|\nabla J(u^n)\|^2 &= \langle \nabla J(u^n), \nabla J(u^n) - \nabla J(u^{n+1})\rangle \\ &\leq \|\nabla J(u^n)\|\|\nabla J(u^n) - \nabla J(u^{n+1})\| \\ &\leq \|\nabla J(u^n)\| C_M \|u^n - u^{n+1}\|\end{aligned}$$

où C_M est la constante de Lipschitz de ∇J sur la boule de rayon M. On en déduit le résultat en divisant par $\|\nabla J(u^n)\|$ et en faisant tendre n vers $+\infty$.

étape 4 : u^n tend vers u.

On utilise encore une fois la forte convexité de J :

$$\alpha\|u^n - u\|^2 \leq \langle \nabla J(u^n) - \nabla J(u), u^n - u\rangle \leq \|\nabla J(u^n)\|\|u^n - u\|.$$

Ainsi, on a bien la convergence avec la majoration d'erreur annoncée.

2.4.2 Cas d'une fonction quadratique

Ici, le pas optimal peut être calculé explicitement. En effet, prenons

$$J(y) = \frac{1}{2}\langle Ay, y\rangle - \langle b, y\rangle$$

avec A symétrique définie positive. Alors, u^n étant supposé connu, on a : $r^n = Au^n - b$. Si on pose comme précédemment : $f(\mu) = J(u^n - \mu r^n)$, on a :

$$\begin{aligned} f'(\mu) &= -\langle A(u^n - \mu r^n) - b, r^n\rangle \\ &= -\langle r^n - \mu A r^n, r^n\rangle. \end{aligned}$$

Donc $f'(\mu^n) = 0$ équivaut à

$$\mu^n = \frac{\|r^n\|^2}{\langle Ar^n, r^n\rangle}. \tag{2}$$

(Notons que le quotient est bien défini si $r^n \neq 0$ puisque A est définie positive.)

Théorème 2.9. Estimation d'erreur : cas quadratique

Dans le cas d'une fonction quadratique, la convergence de l'algorithme de gradient à pas optimal est géométrique, et on a l'estimation

$$\frac{\langle Ae^n, e^n\rangle}{\langle Ae^0, e^0\rangle} \leq \left(\frac{c-1}{c+1}\right)^{2n},$$

où $c = \frac{\lambda_N}{\lambda_1}$ désigne le conditionnement de la matrice A pour la norme usuelle, et $e^n = u^n - u^0$ désigne l'erreur à l'étape n.

Remarque 2.10 Vitesse de convergence

- Pour justifier une convergence géométrique, on doit montrer une estimation du type $\|e^n\| \leq k^n\|e^0\|$. Ici, il se trouve que la norme euclidienne standard ne permet pas d'obtenir facilement une telle estimation, il est plus facile de travailler avec les normes induites par le produit scalaire associé à la matrice A. Notons que toutes les normes sont équivalentes, de sorte que l'estimation annoncée suffit à assurer la convergence géométrique.
- Le résultat montre que la vitesse de convergence dépend du conditionnement de la matrice A. La fonction $c \mapsto \frac{c-1}{c+1}$ est croissante sur $[0, +\infty[$, donc le résultat montre que la convergence sera a priori plus rapide pour une matrice bien conditionnée (i.e. si c est petit). En particulier, on peut se convaincre aisément que pour $c = 1$ (i.e. si A est scalaire), la méthode converge en une seule itération. Par contre, si c est très grand, l'algorithme aura probablement beaucoup de mal à converger et ne doit pas être envisagé sans un préconditionnement.
- Si on compare cette méthode à celle de Newton, on voit que la convergence est ici plus lente (géométrique au lieu de quadratique) ; par contre, elle est globale, i.e. indépendante du choix de u^0.

Ce résultat repose sur l'estimation de Kantorovitch, que l'on admettra ici (voir [5] pour une démonstration de ce lemme).

Lemme 2.11. Estimation de Kantorovitch

Soit A une matrice symétrique définie positive, de conditionnement $c = \frac{\lambda_N}{\lambda_1}$. Alors on a, pour tout $x \in \mathbb{R}^N$,

$$\|x\|^4 \leq \langle Ax, x\rangle\langle A^{-1}x, x\rangle \leq \frac{1}{4}\left(\sqrt{c} + \frac{1}{\sqrt{c}}\right)^2 \|x\|^4.$$

Preuve du théorème :
Remarquons que pour tout n, $Ae^n = Au^n - Au = Au^n - b = r^n$. On a donc

$$\begin{aligned}
\langle Ae^{n+1}, e^{n+1}\rangle &= \langle Ae^{n+1}, e^n - \mu^n r^n\rangle \\
&= \langle Ae^{n+1}, e^n\rangle - \mu^n\langle r^{n+1}, r^n\rangle \\
&= \langle Ae^{n+1}, e^n\rangle \\
&= \langle Ae^n, e^n\rangle - \mu^n\langle Ar^n, e^n\rangle \\
&= \langle Ae^n, e^n\rangle - \mu^n\|r^n\|^2.
\end{aligned}$$

En utilisant la relation (2) on obtient

$$\frac{\langle Ae^{n+1}, e^{n+1}\rangle}{\langle Ae^n, e^n\rangle} = 1 - \mu^n\frac{\|r^n\|^2}{\langle Ae^n, e^n\rangle} = 1 - \frac{\|r^n\|^4}{\langle Ar^n, r^n\rangle\langle Ae^n, e^n\rangle}.$$

On a de plus, d'après l'inégalité de Kantorovitch,

$$\langle Ar^n, r^n\rangle\langle Ae^n, e^n\rangle = \langle Ar^n, r^n\rangle\langle r^n, A^{-1}r^n\rangle \leq \frac{1}{4}\left(\sqrt{c} + \frac{1}{\sqrt{c}}\right)^2 \|r^n\|^4.$$

Finalement, on obtient

$$\frac{\langle Ae^{n+1}, e^{n+1}\rangle}{\langle Ae^n, e^n\rangle} = 1 - \frac{4\|r^n\|^4}{\left(\sqrt{c} + c^{-1/2}\right)^2\|r^n\|^4} = \frac{c + c^{-1} - 2}{c + c^{-1} + 2} = \left(\frac{c-1}{c+1}\right)^2.$$

L'inégalité annoncée s'en déduit immédiatement.

2.4.3 Remarques pratiques

i **Test d'arrêt :** quand décide-t-on qu'on a convergé ?

- 1er choix : on stoppe les itérations si $\|u^{k+1} - u^k\| < \tau$, typiquement $\tau = 10^{-6}\|u^0\|$. Cela évite d'accumuler des itérations qui ne modifient pas la solution trouvée. Pb : on ne peut pas en général contrôler l'erreur. Si A est mal conditionnée, il se peut qu'on soit encore loin de la solution exacte.
- 2ème choix : on stoppe les itérations si $\|\nabla J(u^k)\| < \tau$, typiquement $\tau = 10^{-6}\|\nabla J(u^0)\|$. Avantage : si J est α-convexe, on a un contrôle de l'erreur car

$$\|u^k - u\| \leq \frac{1}{\alpha}\|\nabla J(u^k) - \nabla J(u)\| = \frac{1}{\alpha}\|\nabla J(u^k)\|.$$

ii **Recherche linéaire :** comment calculer le pas optimal en pratique ?
- Soit on peut le déterminer explicitement en résolvant exactement l'équation $f'(\mu^n) = 0$ (ex. cas quadratique)
- Soit on doit l'approcher par une méthode du type de celles proposées à la section 2 (ex : section dorée). En pratique, un petit nombre d'itérations de la méthode 1D (une dizaine) peut suffire.
- Des variantes existent, moins coûteuses : sélection de pas d'Armijo (voir [4]). Dans ce cas, on se contente de calculer un pas qui garantisse la convergence de l'algorithme, même s'il n'est pas optimal.

iii **Calcul du gradient :** Lorsqu'un calcul direct est trop difficile, on peut le calculer par différences finies. Préférer un gradient exact quand c'est simple.

iv **Cas où les hypothèses du théorème ne sont pas satisfaites :** on gère au mieux ! On utilisera souvent l'algo. du gradient même si on ne sait pas montrer qu'il converge vers la solution... En général, il marche bien malgré tout (méthode de descente).
Si J n'est pas convexe : on se ramène (si c'est possible) à un compact sur lequel il y a un seul point critique qui est bien un point de minimum.
Si J n'est pas différentiable (par exemple $x \mapsto |x|$) : des variantes de la méthode du gradient existent, largement hors programme ; voir [4].

2.5 Méthode de gradient à pas fixe

Il s'agit d'une variante de la méthode précédente, dans laquelle on garde un pas μ fixé pour toutes les itérations. Attention, ce n'est pas une méthode de descente ! En particulier, selon la taille du paramètre μ, la suite $J(u^n)$ n'est pas nécessairement décroissante...
L'algorithme prend la forme très simple :

Algorithme 2.12. Gradient à pas fixe

On fixe un paramètre $\mu > 0$.
- *Démarrage.* On choisit $u^0 \in \mathbb{R}^N$.
- *Étape n.* On pose $u^{n+1} = u^n - \mu \nabla J(u^n)$.

Notons que, le pas n'étant plus optimal, on perd ici l'orthogonalité des directions de descente successives !

2.5.1 Convergence : cas général

On a le résultat de convergence suivant :

Théorème 2.13. Convergence : gradient à pas fixe

Sous l'hypothèse (H), l'algorithme converge pour tout $\mu > 0$ assez petit vers la solution de (1). En particulier, la convergence a lieu pour tout

$$\mu \in \left]0, \frac{2\alpha}{C^2}\right[,$$

où C désigne la constante de Lipschitz de ∇J sur la boule fermée $B = B(u, \|u - u^0\|)$.

Remarque : dans la littérature, le théorème est en général démontré sous l'hypothèse que le gradient est globalement C-lipschitzien. L'avantage est que cela donne un intervalle explicite sur μ qui assure la convergence. Par contre, supposer que la fonction J est à la fois α-convexe et à gradient globalement lipschitzien est très restrictif. En particulier, on a :

$$\begin{aligned} J\,\alpha\text{-convexe} &\Rightarrow \forall v, w, \langle \nabla J(v) - \nabla J(w), v - w\rangle \geq \alpha \|v - w\|^2 \\ \nabla J\, C\text{-lipschitzien} &\Rightarrow \forall v, w, \langle \nabla J(v) - \nabla J(w), v - w\rangle \leq C \|v - w\|^2. \end{aligned}$$

Ces deux hypothèses réunies impliquent en particulier $C \geq \alpha$. Notons que ces deux hypothèses sont vérifiées pour une fonction quadratique associée à une matrice définie positive (on a alors $C = \lambda_N \geq \alpha = \lambda_1$).

Preuve du théorème :
On note toujours $e^n = u^n - u$ l'erreur à l'étape n. On va vérifier par récurrence que, pour tout $n \geq 0$, $u^n \in B$. C'est vrai pour $n = 0$. Soit $n \geq 0$ fixé et supposons que $u^n \in B$. Alors on a l'inégalité $\|\nabla J(u^n) - \nabla J(u)\| \leq C\|e^n\|$. Montrons que $u^{n+1} \in B$. On a :

$$\begin{aligned} e^{n+1} &= e^n - \mu \nabla J(u^n) = e^n - \mu(\nabla J(u^n) - \nabla J(u)) \\ \|e^{n+1}\|^2 &= \|e^n\|^2 - 2\mu\langle e^n, \nabla J(u^n) - \nabla J(u)\rangle + \mu^2 \|\nabla J(u^n) - \nabla J(u)\|^2 \\ &\leq (1 - 2\mu\alpha + \mu^2 C^2)\|e^n\|^2. \end{aligned}$$

Or le choix de

$$\mu \in \left]0, \frac{2\alpha}{C^2}\right[$$

assure à la fois que $u^{n+1} \in B$, ce qui achève la récurrence, et que la suite $(\|e^n\|)$ décroît vers 0 avec vitesse géométrique.

2.5.2 Cas des fonctions quadratiques

Soit J quadratique de hessienne A symétrique définie positive. Alors, quitte à changer l'origine, et à se placer dans la base de diagonalisation de A, J prend la forme

$$J(x) = \frac{1}{2}\sum_{i=1}^{N} \lambda_i x_i^2.$$

On suppose les valeurs propres rangées par ordre croissant : $0 < \lambda_1 \leq ... \leq \lambda_N$. L'algorithme se réécrit sous la forme

$$u_i^{n+1} = u_i^n(1 - \lambda_i \mu), \quad 1 \leq i \leq N, n \geq 0.$$

L'algorithme converge si et seulement si

$$\forall i, -1 < 1 - \lambda_i \mu < 1, \quad i.e.\ 0 < \mu < \frac{2}{\lambda_N}$$

Cet intervalle contient strictement l'intervalle donné par le théorème précédent, à savoir

$$0 < \mu < \frac{2\lambda_1}{\lambda_N^2}.$$

On peut chercher à choisir le meilleur pas μ possible, en minimisant

$$f(\mu) = \max(|1 - \mu\lambda_i|, 1 \leq i \leq N).$$

Un graphe permet de s'assurer que $f(\mu)$ est minimal pour la valeur

$$\mu = \frac{2}{\lambda_1 + \lambda_N}.$$

2.6 Méthode de gradient conjugué pour une fonction quadratique

Cette méthode a été proposée par Hestenes et Stiefel en 1952 pour les fonctions quadratiques. Des extensions ont été proposées ensuite pour des fonctions α-convexes plus générales, mais nous ne les présenterons pas ici.

2.6.1 Définition abstraite et convergence

Dans la méthode de gradient à pas optimal, le choix de la direction de descente est motivé par le fait que c'est la direction optimale pour des petits pas. Cependant, le pas peut parfois être grand, de sorte que la direction du gradient peut s'avérer mauvaise (surtout pour des matrices mal conditionnées) et conduire à une convergence très lente. L'idée de la méthode du gradient conjugué est de choisir une meilleure direction de descente, par un algorithme qui a *de la mémoire*. Ainsi, au lieu de considérer uniquement la direction du gradient au point u^n, on choisira la direction de descente en considérant également les gradients aux points précédents de l'algorithme $u^0, \cdots, u^n$.

L'algorithme abstrait s'écrit

$$\left\{\begin{array}{ll} \text{choisir} & u^0 \in \mathbb{R}^N, \\ \forall n \geq 0, & G^n = Vect(\nabla J(u^i), 0 \leq i \leq n) \\ & u^{n+1} = \underset{v \in u^n + G^n}{\text{Argmin}}\, J(v) \end{array}\right. \tag{3}$$

Remarque 2.14 Comparaison avec le gradient à pas optimal

En comparaison, l'algorithme de gradient à pas optimal s'écrirait :

$$\left\{\begin{array}{ll} \text{choisir} & u^0 \in \mathbb{R}^N, \\ \forall n \geq 0, & F^n = Vect(\nabla J(u^n)) \\ & u^{n+1} = \underset{v \in u^n + F^n}{\text{Argmin}}\, J(v) \end{array}\right.$$

On peut remarquer que

- dans les deux cas, l'argmin est bien défini grâce à la forte convexité de J.
- l'espace G^n étant plus gros que F^n, l'itéré u^{n+1} construit par la méthode du gradient conjugué est au moins aussi bon que celui construit par la méthode du gradient à pas optimal. Reste à savoir si on dispose d'un algorithme efficace pour le calculer ! Ce sera l'objet du paragraphe suivant.

Comme on l'a fait dans le cas de la méthode de gradient à pas optimal, on peut chercher à écrire la caractérisation du point de minimum u^{n+1}. L'ensemble $G^n + u^n$ étant convexe (c'est un espace affine), le point de minimum est caractérisé par

$$u^{n+1} \in u^n + G^n, \qquad \langle \nabla J(u^{n+1}) w \rangle = 0, \forall w \in G^n.$$

ce qui se réécrit

$$\langle \nabla J(u^{n+1}), \nabla J(u^k) \rangle = 0, \ 0 \leq k \leq n. \tag{4}$$

On obtient une propriété d'orthogonalité plus forte que pour le gradient à pas optimal : le gradient à chaque étape est orthogonal à tous les gradients précédents. En particulier :

- Si $\nabla J(u^n) = 0$, alors $u^n = u$ et l'algorihme s'arrête ;

- si $\forall k \in \{0, \cdots, n\}, \nabla J(u^k) \neq 0$ alors la famille $(\nabla J(u^0), \cdots, \nabla J(u^n))$ forme une base de G^n, donc $\dim G^n = n + 1$. Or $G^n \subset \mathbb{R}^N$.

On en déduit que le théorème suivant.

Théorème 2.15. Convergence : algorithme de gradient conjugué

Quel que soit le point u^0 choisi, l'algorithme converge en N itérations au plus vers la solution exacte de (1), i.e. $\exists k \leq N, u^k = u$.

Remarque 2.16 Convergence en un nombre fini d'itérations.

En théorie, c'est très séduisant : cet algorithme peut être vu comme une méthode directe pour résoudre les systèmes linéaires symétriques définis positifs. En pratique, il y a quand-même quelques bémols :

- d'abord, l'algorithme présenté ci-dessus reste abstrait ; il faut donc s'assurer qu'on est effectivement capable de calculer simplement les itérées successives ;
- comme pour toute méthode numérique, les erreurs d'arrondi sont inévitables dans le calcul pratique ;
- cette méthode est souvent utilisée pour des systèmes de très grosse dimension ($N = 10^5$) provenant par exemple de la discrétisation d'une EDP. Dans ce cas, on n'a pas forcément envie de faire N itérations pour avoir un résultat ;
- enfin, il faut comparer la méthode aux autres méthodes directes, en particulier en terme de coût de l'implémentation numérique et de propagation des erreurs d'arrondi. Voir page 14 pour quelques éléments de réponse.

2.6.2 Reformulation comme méthode de descente

On va voir que l'algorithme abstrait présenté ci-dessus se reformule en une méthode de descente où les directions et le pas sont assez faciles à calculer à chaque itération. Le point-clé est que les directions de descente de la méthode sont *conjuguées* deux à deux (d'où le nom de la méthode), au sens défini ci-dessous.

Définition 2.17. Directions conjuguées

Soit A une matrice symétrique définie positive. Deux vecteurs x et y sont dits conjugués par rapport à A s'ils sont orthogonaux pour le produit scalaire associé à A, i.e.

$$\langle Ax, y\rangle = \langle x, Ay\rangle = 0.$$

Proposition 2.18. Les directions de descente sont conjuguées 2 à 2.

On note $p^k = u^{k+1} - u^k$, $0 \leq k$ les directions de descente consécutives de la méthode du gradient conjugué. Alors on a, pour tous $0 \leq k < l$, $\langle p^k, Ap^l\rangle = 0$. Autrement dit, les directions de descente de cette méthode sont conjuguées 2 à 2.

Preuve : Soit $l > 0$ fixé et $0 \leq k < l$. Alors on a

$$\langle Ap^l, \nabla J(u^k)\rangle = \langle Au^{l+1} - Au^l, \nabla J(u^k)\rangle = \langle \nabla J(u^{l+1}) - \nabla J(u^l), \nabla J(u^k)\rangle = 0$$

d'après la propriété (4). On en déduit que $Ap^l \in (G^{l-1})^\perp$ donc $\langle p^k, Ap^l\rangle = 0$.

Cette propriété permet d'obtenir une stratégie pour calculer les directions de descente successives.

- La première direction d^0 est naturellement celle du gradient : $d^0 = \nabla J(u^0)$.
- à l'étape $k \geq 1$, la direction d^k est telle que $(d^0, \cdots, d^k)$ est une base A-conjuguée de G^k. Or on connait une base de G^k qui est fournie par les $\nabla J(u^l)$, $l \leq k$. On peut donc déterminer d^k par un procédé de Gram-Schmidt pour le produit scalaire associé à A. Un calcul simple montre que cela conduit à la formule suivante :

$$d^k = \frac{\|\nabla J(u^k)\|^2}{\|\nabla J(u^{k-1})\|^2} d^{k-1} + \nabla J(u^k).$$

Sachant que $u^{k+1} - u^k$ doit être colinéaire à ce d^k, et que u^{k+1} est le point de minimum de J sur un ensemble plus gros que $u^k + \mathbb{R}d^k$, alors il vérifie nécessairement :

$$u^{k+1} = \underset{v \in u^k + \mathbb{R}d^k}{\text{Argmin}}\, J(v).$$

Ainsi, une fois connue la direction de "descente", u^{k+1} peut s'obtenir en calculant le pas optimal dans cette direction. On cherche donc μ^k tel que

$$J(u^k + \mu^k d^k) = \min_{\mu \in \mathbb{R}} J(u^k + \mu d^k),$$

ce qui donne

$$\langle \nabla J(u^k - \mu^k d^k), d^k\rangle = 0$$

et finalement

$$\mu^k = \frac{\langle \nabla J(u^k), d^k\rangle}{\langle Ad^k, d^k\rangle}.$$

On aboutit finalement à l'algorithme pratique suivant.

Algorithme 2.19. Gradient conjugué : algorithme pratique

- *Démarrage.* On choisit $u^0 \in \mathbb{R}^N$, et on calcule $r^0 = Au^0 - b$.
- *étape* $k \geq 0$. On suppose u^k, r^k connus.
  - Si $r^k = 0$, c'est fini.
  - Sinon, on pose

$$d^k = r^k + \frac{\|r^k\|^2}{\|r^{k-1}\|^2} d^{k-1}, \quad v^k = Ad^k, \quad \mu^k = \frac{\langle r^k, d^k\rangle}{\langle v^k, d^k\rangle}, \quad u^{k+1} = u^k - \mu^k d^k, \quad r^{k+1} = r^k - \mu^k v^k.$$

En pratique, à cause des erreurs d'arrondi, on s'arrêtera lorsque la norme de r^k sera inférieure à un seuil fixé. Grâce à la forte convexité de J, cela garantit que l'erreur est assez petite.

2.6.3 Avantages

On peut chercher à comparer le coût numérique de cette méthode avec celui des autres méthodes directes. Regardons le coût de l'étape k. On suppose u^k et r^k connus.

- calcul de $\|r^k\|^2$: N multiplications, $N-1$ additions ;
- calcul de d^k : N multiplications, N additions, 1 division ;
- calcul de $v^k = Ad^k$: N^2 multiplications, N^2 additions ;
- calcul de μ^k : $2N$ multiplications, $2N-2$ additions, 1 division ;
- calcul de u^{k+1} : N multiplications, N additions ;
- calcul de r^{k+1} : N multiplications, N additions.

Au total, si on considère N itérations avant la convergence, on a au maximum $2N^3 + O(N^2)$ opérations. La méthode semble donc moins intéressante que celle du pivot de Gauss (de l'ordre de $2N^3/3$ opérations) ou celle de Cholesky qui est possible ici car la matrice est symétrique définie positive.

Notons cependant que la matrice A n'intervient que par le calcul des vecteurs $v^k = Ad^k$, ce qui permet de réduire le coût de calcul dans le cas des matrices creuses, et également de limiter la place mémoire.

On a regardé ci-dessus la méthode du gradient conjugué comme une méthode directe. La plupart du temps, il est plus pertinent de la regarder comme une méthode itérative ; on ne cherchera donc pas à faire N itérations, mais on s'arrêtera lorsque l'écart à la solution exacte sera suffisamment petite. Pour la comparer à la méthode du gradient à pas optimal (par exemple), on a besoin d'une estimation de l'erreur. On peut montrer l'estimation suivante (voir par exemple [6]).

Théorème 2.20. Estimation d'erreur : algorithme de gradient conjugué

Pour tout n, on a l'estimation d'erreur suivante :

$$\frac{\langle Ae^n, e^n\rangle}{\langle Ae^0, e^0\rangle} \leq 4\left(\frac{\sqrt{c}-1}{\sqrt{c}+1}\right)^{2n},$$

où $c = \frac{\lambda_N}{\lambda_1}$ désigne le conditionnement de la matrice A, et $e^n = u^n - u^0$ désigne l'erreur à l'étape n.

Il est intéressant de comparer cette estimation avec celle obtenue dans le cas du gradient à pas optimal, voir théorème 2.9. Dans les deux cas, la convergence est géométrique, avec une raison qui croît avec le conditionnement de A. Cependant, pour des matrices mal conditionnées, la convergence est beaucoup plus rapide par la méthode du gradient conjugué.