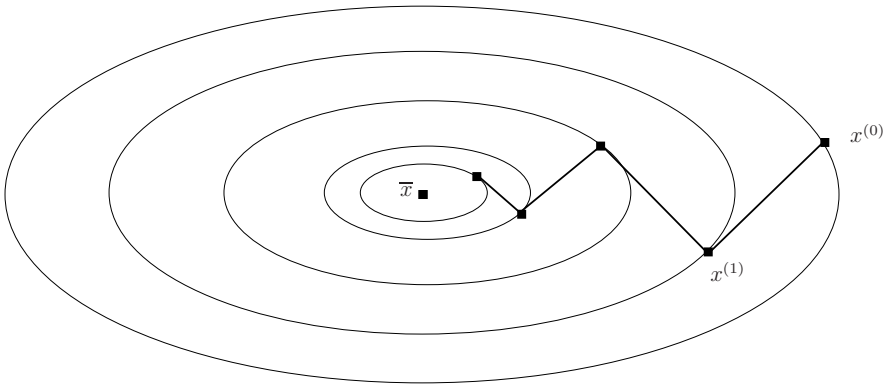
MATHÉMATIQUES ET INFORMATIQUE
Sciences
Université Paris Cité

MASTER 1 INFO

Optimisation et algorithmique



Georges Koepfler 2014-2024 - georges.koepfler@u-paris.fr

Table des matières

Avertissement : ces notes sont un support et complément du cours magistral, des travaux dirigés et pratiques. Leur contenu n'est pas équivalent au cours enseigné, en particulier les examens et contrôles se réfèrent au cours enseigné uniquement.

Bibliographie.

Pour trouver des algorithmes en `C` et des références utiles, on pourra consulter

- W.H. Press, B.P. Flannery, S.A. Teukolsky et W.T. Vetterling, *Numerical Recipes in C*, Cambridge University Press 1988.
  Site internet `http://www.nr.com/`

Quelques livres qui traitent de l'analyse numérique linéaire, et de ses applications, sont :

- J. Demmel, *Applied Numerical Linear Analysis*, SIAM 1997 ;
- P. Lascaux et J. Théodor, *Analyse numérique matricielle appliquée à l'art de l'ingénieur*, 2 tomes, Masson 1988.

Les livres suivants traitent des méthodes d'optimisation numérique :

- S. Boyd et L. Vandenberghe, *Convex Optimization*, Cambridge University Press 2004 ;
- R. Fletcher, *Practical Methods of Optimization*, J. Wiley & Sons 1987 ;
- J. Nocedal et S.J. Wright, *Numerical Optimization*, Springer Series in Operations Research 1999.

Chapitre 1

Introduction

Dans introduction on s'intéresse à la résolution numérique de divers problèmes afin d'illustrer les problèmes que l'on peut rencontrer, sans faire une étude approfondie.

Rappelons quelques points essentiels dont on doit tenir compte dans la résolution numérique d'un problème :

1. *Instabilité du problème* : Certains problèmes mathématiques sont instables : de petites perturbations des paramètres, dont dépend le problème, vont engendrer de grandes variations des solutions.
   Un exemple est la résolution de l'équation de la chaleur inversée $u_t = -\Delta u$; un autre la détermination des racines d'un polynôme.
   Comme les paramètres des modèles mathématiques viennent souvent d'expériences on utilisera autant que possible des modèles stables.
   Soit $s = \mathcal{A}[e]$ un algorithme ou problème qui, en fonction du paramètre e, produit le résultat s ; on s'intéresse à la stabilité de $\mathcal{A}$ et l'on veut contrôler l'erreur relative commise par une petite perturbation de e :

   $$\frac{|\tilde{s} - s|}{|s|} = \frac{|\mathcal{A}(e + \delta e) - \mathcal{A}(e)|}{|\mathcal{A}(e)|} \leq c(\mathcal{A}) \, \frac{|\delta e|}{|e|} \, .$$

   On appelle $c(\mathcal{A})$ le *conditionnement du problème* $\mathcal{A}$ (en choisissant la meilleure borne possible dans l'inégalité pour obtenir $c(\mathcal{A})$).

2. *Erreurs d'approximation* : Ce type d'erreur apparaît lorsque l'on remplace un problème «continue» par un problème «discret». Quand le pas de discrétisation tend vers zéro la formulation discrète doit tendre vers la formulation continue du problème.

3. *Erreurs d'arrondi* : Les calculs numériques sur ordinateur se font avec une précision finie : la mémoire disponible étant finie on ne peut pas représenter les nombres réels qui ont un développement décimal infini. On peut représenter $1/3$, mais pas son écriture décimale, on ne pourra représenter π que par un nombre fini de décimales.
   Un représentation fréquente est celle en *virgule flottante* des nombres réels :
   $f = (-1)^s \cdot 0, d_{-1} d_{-2} \ldots d_{-r} \cdot b^j$, $m \leq j \leq M$ et $0 \leq d_{-i} \leq b - 1$;
   où s est le signe, $d_{-1} d_{-2} \cdots d_{-r}$ la mantisse, b la base et r est le nombre de chiffres significatifs. La *précision machine* pour un flottant de mantisse r est b^{1-r} , c'est la distance entre 1 et le nombre flottant immédiatement plus grand $1 + b^{1-r}$.

Comme on ne dispose que d'un nombre fini de réels le résultat des opérations arithmétiques de base ($+$, $-$, $\times$, $/$) n'est pas nécessairement représentable : on est obligé d'arrondir.

L'arithmétique IEEE, utilisée sur les machines sous Unix et Linux, définit des nombres flottants en base 2 : en simple précision de 32 bits (type `float` en C) et double précision de 64 bits (type `double` en C). En simple précision on utilise un bit pour s, 1 octet pour l'exposant (signé) j, *i.e.* $j \in \{-127, \ldots, +128\}$, et 23 bits pour la mantisse. Si l'on suppose que la représentation est normalisée, *i.e.* $d_{-1} \neq 0$, on gagne un bit significatif et un flottant en simple précision s'écrit $f = (-1)^s\, 1, d_{-1}d_{-2}\ldots d_{-r}\, 2^j$.
dans ce cas la précision machine est de 2^{-23}, c.-à-d. approximativement 10^{-7}, et les flottants normalisés positifs vont de 10^{-38} à 10^{+38}.

En plus des résultats, les modules arithmétiques envoient des messages qui indiquent la nature du résultat de l'opération : si ce n'est pas un nombre (`NaN` = *NotANumber* $= \infty - \infty = \sqrt{-1} = 0/0\ldots$) ou un nombre trop grand, resp. petit, et qui ne peut être représenté.

4. *Complexité des calculs* : Pour la plupart des applications on veut obtenir un résultat en un temps raisonnable, il est donc important de pouvoir estimer le temps que l'algorithme va utiliser pour résoudre le problème. Pour cela, on compte le nombre d'opérations flottantes (angl. *flops*) en fonction de la taille du problème. Souvent il suffit d'avoir un ordre de grandeur : si N est la taille du problème (p.ex. nombre d'inconnues) on peut avoir des algorithmes en $O(N)$, $O(N \ln N)$, $O(N^2)$, $O(N!)\ldots$

Pour illustrer les problèmes cités en haut, on va présenter quatre exemples classiques où ces notions jouent un rôle important :

Exemple 1 : Évaluation d'un polynôme proche d'une racine multiple.

Soit $p(x) = \sum_{i=0}^{d} a_i\, x^i$ un polynôme à coefficients réels avec $a_d \in \mathbb{R}^*$.

Si l'on connaît les racines du polynôme on peut le factoriser $p(x) = a_d \prod_{i=1}^{d} (x - r_i)$.

On peut aussi associer à p sa *matrice compagne*

$$\begin{pmatrix} -\frac{a_{d-1}}{a_d} & -\frac{a_{d-2}}{a_d} & & -\frac{a_2}{a_d} & -\frac{a_1}{a_d} & -\frac{a_0}{a_d} \\ 1 & 0 & \ldots & 0 & 0 & 0 \\ 0 & 1 & & 0 & 0 & 0 \\ \vdots & & \ddots & & \vdots & \vdots \\ 0 & 0 & & 1 & 0 & 0 \\ 0 & 0 & \ldots & 0 & 1 & 0 \end{pmatrix},$$

dont les valeurs propres sont les racines de p.

Si l'on veut calculer $p(x)$ en utilisant la première représentation, on doit effectuer de l'ordre de $2d$ opérations élémentaires ($+$ et $\times$) et $d-1$ appels à la fonction puissance ; si l'on connaît les racines, il suffit de $2d$ opérations élémentaires. Mais comme on connaît

rarement une factorisation du polynôme on utilise l'*algorithme de* Horner pour calculer $p(x)$:

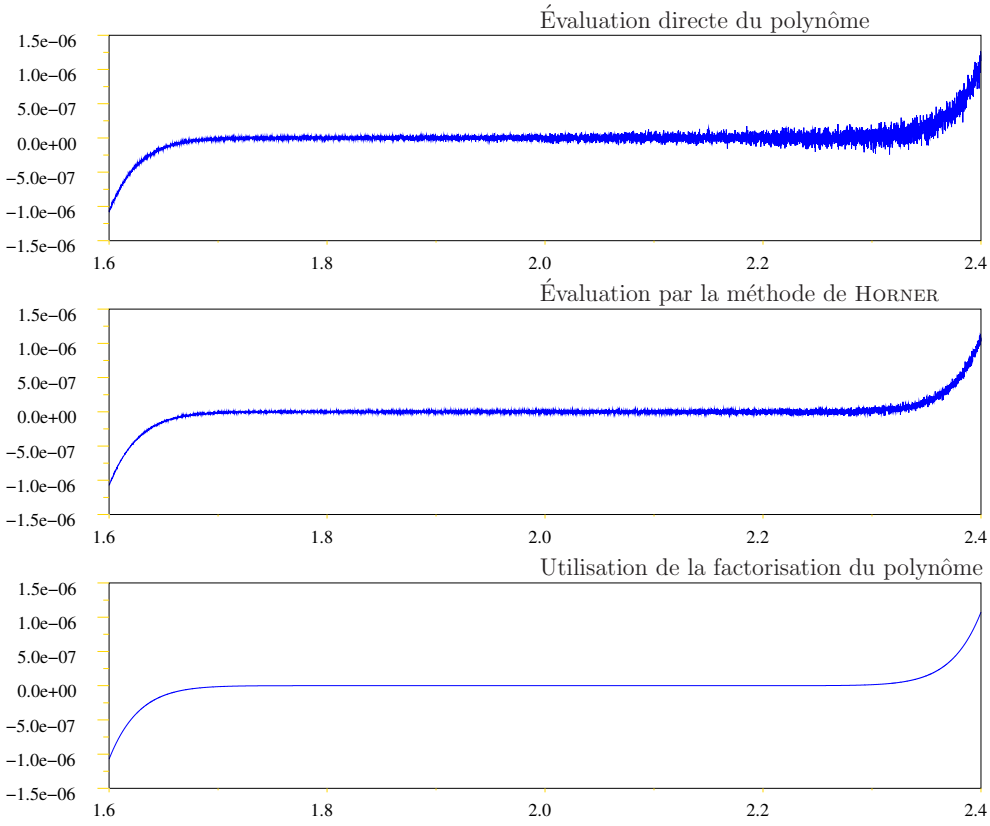
Algorithme 1.1 (Horner)
Évaluation d'un polynôme en x à partir de ses coefficients $a_0, a_1, \ldots, a_d$.

$y = a_d$
pour *i=d-1* ***to*** *0*
  $y = x \cdot y + a_i$

Cet algorithme permet d'évaluer $p(x)$ à partir des coefficients a_i en en $2d$ opérations élémentaires. On se propose de calculer et représenter le polynôme suivant :

$$\begin{aligned} p(x) &= x^{15} - 30x^{14} + 420x^{13} - 3640x^{12} + 21840x^{11} - 96096x^{10} \\ &\quad +320320x^9 - 823680x^8 + 1647360x^7 - 2562560x^6 + 3075072x^5 \\ &\quad -2795520x^4 + 1863680x^3 - 860160x^2 + 245760x - 32768 \\ &= (x-2)^{15} \ . \end{aligned}$$

Les résultats sont représentés à la figure suivante, on a évalué $p(x)$ pour $x \in [1.6, 2.4]$ et avec un pas de 10^{-4} .



Les oscillations qui apparaissent dans le graphe de p rendent difficiles la détection de la racine de p par une méthode comme celle de la dichotomie par exemple.
Pour calculer de façon rapide et précise les valeurs d'un polynôme il vaut donc mieux utiliser sa forme factorisée, or pour cela il faut trouver tous les zéros de l'équation polynomiale $p(x) = 0$, ce qui est instable comme l'on verra, ou il faut trouver les valeurs propres de la matrice compagne, ce qui est un autre problème, non moins difficile.

Exemple 2 : Détermination des racines d'un polynôme perturbé.

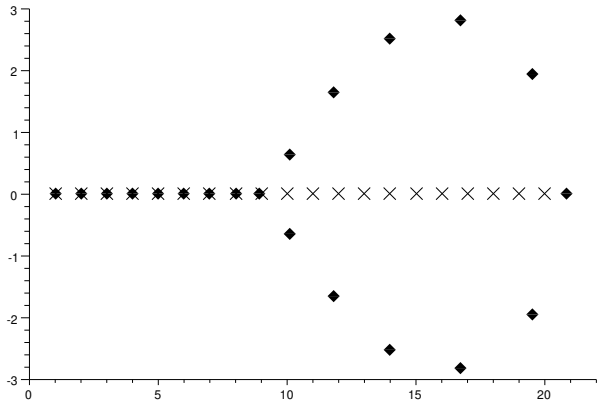
Pour illustrer l'instabilité de la résolution d'une équation polynomiale par rapport aux coefficients du polynôme on va utiliser l'exemple de proposé par Wilkinson (1963) :

$$\begin{aligned} w(x) &= \prod_{i=1}^{20}(x-i) \\ &= x^{20} - 210\,x^{19} + 20615\,x^{18} - 1256850\,x^{17} + 53327946\,x^{16} - 1.672\cdot 10^{9}\,x^{15} \\ &\quad +4.017\cdot 10^{10}\,x^{14} - 7.561\cdot 10^{11}\,x^{13} + 1.131\cdot 10^{13}\,x^{12} - 1.356\cdot 10^{14}\,x^{11} \\ &\quad +1.308\cdot 10^{15}\,x^{10} - 1.014\cdot 10^{16}\,x^{9} + 6.303\cdot 10^{16}\,x^{8} - 3.113\cdot 10^{17}\,x^{7} \\ &\quad +1.207\cdot 10^{18}\,x^{6} - 3.600\cdot 10^{18}\,x^{5} + 8.038\cdot 10^{18}\,x^{4} - 1.287\cdot 10^{19}\,x^{3} \\ &\quad +1.380\cdot 10^{19}\,x^{2} - 8.753\cdot 10^{18}\,x + 20! \end{aligned}$$

On considère ensuite le polynôme perturbé $\tilde{w}(x) = w(x) - 2^{-23}\,x^{19}$ et, en faisant des calculs numériques en haute précision, on obtient les racines suivantes pour $\tilde{w}$:

1, 00000 0000	10, 09526 6145 + 0, 64350 0904 i
2, 00000 0000	10, 09526 6145 − 0, 64350 0904 i
3, 00000 0000	11, 79363 3881 + 1, 65232 9728 i
4, 00000 0000	11, 79363 3881 − 1, 65232 9728 i
4, 99999 9928	13, 99235 8137 + 2, 51883 0070 i
6, 00000 6944	13, 99235 8137 − 2, 51883 0070 i
6, 99969 7234	16, 73073 7466 + 2, 81262 4894 i
8, 00726 7603	16, 73073 7466 − 2, 81262 4894 i
8, 91725 0249	19, 50243 9400 + 1, 94033 0347 i
20, 84690 8101	19, 50243 9400 − 1, 94033 0347 i

On obtient 5 racines complexes conjuguées de partie imaginaire non négligeable. Cet exemple célèbre montre comment une petite perturbation d'un coefficient change de façon profonde l'ensemble des solutions d'une équation polynomiale. Les figures suivantes donnent les graphes de w et $\tilde{w}$.



Les zéros de w ($\times$) et de $\tilde{w}$ ($\blacklozenge$) dans le plan complexe.

Exemple 3 : Calcul de la trajectoire de la fusée ARIANE 5

Le 4 juin 1996, le premier vol de la fusée ARIANE 5 s'est terminé par l'explosion de l'engin à peine 50s après le décollage. Dans le rapport ESA/CNES [1], établi suite à cet incident, on peut lire que les causes de l'échec sont dues à un signal d'*overflow* mal interprété par l'ordinateur de bord.

En effet, les deux systèmes de référence inertiels ont arrêté de fonctionner à cause d'une erreur lors d'une conversion d'un nombre flottants en 64 bit vers un entier signé en 16 bits. Le nombre à convertir ayant une valeur trop grande pour être représenté en 16 bits, l'opération s'est terminé par un signal d'*overflow*. Le premier système de référence inertiel a arrêté de fonctionner et le second à pris le relais, et la même erreur c'est produite. L'ordinateur de bord a cette fois pris en compte le signal d'*overflow*, mais en l'interprétant comme étant une donnée. Il y a eu un changement de trajectoire qui a mené la fusée vers une forte déviation, ceci a causé la désintégration du lanceur.

Le rapport constate que le programme en cause n'était plus utilisé après le décollage, de plus le code avait été transféré sans changement du système ARIANE 4 vers ARIANE 5, sans tenir compte du fait que sur le nouveau lanceur les paramètres prenaient d'autres valeurs.

Conclusion de ce feu d'artifice fort coûteux : il est important de connaître des fourchettes pour les valeurs d'entrée d'un programme et de protéger les logiciels contre les résultats erronés.

Exemple 4 : Multiplication de matrices

Soient A, B et C des matrices rectangulaires, de dimensions respectives (n, m), (m, p) et (p, q). Combien d'opérations sont nécessaires pour calculer ABC ? Comme $ABC = (AB)C = A(BC)$, on a deux possibilités d'évaluation.

Le calcul de $(AB)C$ nécessite de l'ordre de $2np(m + q)$ opérations, tandis que le calcul de $A(BC)$ se fait en $2mq(n + p)$ opérations. Si l'on prend par exemple $n = p = 10$ et $m = q = 100$ on trouve $4 \cdot 10^4$ et $4 \cdot 10^5$. La multiplication des matrices rectangulaires est bien distributive, mais le nombre d'opérations nécessaires peut varier de façon importante.

Soit $n = m = p$, alors A et B sont des matrices carrées et le coût de leur multiplication est $O(n^3)$, or il existe un algorithme qui permet de réduire ce coût à $O(n^{\log_2 7})$. La méthode est basé sur une réécriture de le multiplication de deux matrices $(2, 2)$, si les matrices sont carrées d'ordre n, avec $n = 2^r$, on applique l'algorithme de manière récursive.

Algorithme 1.2 (Strassen)
Calcul du produit des matrices carrées d'ordre $n = 2^r$ *:* $C = AB$
$C = STRASSEN(A, B, n)$
  si $n = 1$ ***alors***
    $C = A \cdot B$ *// cas scalaire //*
  sinon
    $A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$ *et* $B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$ *// où les* A_{ij} *et* B_{ij} *sont* $(n/2, n/2)$ *//*
    $Q_1 = STRASSEN(A_{11} + A_{22}, B_{11} + B_{22}, n/2)$
    $Q_2 = STRASSEN(A_{21} + A_{22}, B_{11}, n/2)$
    $Q_3 = STRASSEN(A_{11}, B_{12} - B_{22}, n/2)$
    $Q_4 = STRASSEN(A_{22}, -B_{11} + B_{21}, n/2)$

1. *ARIANE 5 : Flight 501 Failure*, rapport sous la direction de J.L. Lions, juillet 1996.

$$
\begin{aligned}
Q_5 &= STRASSEN(A_{11} + A_{12}, B_{22}, n/2)\\
Q_6 &= STRASSEN(-A_{11} + A_{21}, B_{11} + B_{12}, n/2)\\
Q_7 &= STRASSEN(A_{12} - A_{22}, B_{21} + B_{22}, n/2)\\
C_{11} &= Q_1 + Q_4 - Q_5 + Q_7\\
C_{12} &= Q_3 + Q_5\\
C_{21} &= Q_2 + Q_4\\
C_{22} &= Q_1 - Q_2 + Q_3 + Q_6\\
C &= \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}
\end{aligned}
$$

En pratique il faut n assez grand pour obtenir un gain satisfaisant avec la méthode de Strassen ($\log_2 7 = 2,8073$). Mais des méthodes simples de multiplication par blocs sont par exemple utilisées dans les librairies BLAS (angl. *Basic Linear Algebra Subroutines*). Nous allons en décrire le principe en quelques mots.

Il ne suffit pas d'utiliser la vitesse de calcul et la mémoire centrale des processeurs pour obtenir des programmes rapides : supposons que l'on veut multiplier deux grandes matrices dont la taille nécessite une sauvegarde dans une mémoire qui est d'accès lent. Il est clair que la plus grande partie du temps est utilisé pour transférer des données. Les librairies tels que BLAS implémentent des multiplications de matrices par blocs en tenant compte de l'architecture du processeur. Les blocs transférés de la mémoire lente sont de la taille des mémoires d'accès rapide, on fait moins de transferts lents et on passe proportionnellement plus de temps sur les calculs.

Les librairies LAPACK, CLAPACK et ScaLAPACK (`http//www.netlib.org`) utilisent cette approche.
Des logiciels tels que `Octave`[2] ou `Matlab`©, utilisent des opérations par blocs pour accroître leurs performances.

2. logiciel disponible à `https://www.gnu.org/software/octave/`

Chapitre 2

Calcul différentiel dans $\mathbb{R}^n$

Un point x de $\mathbb{R}^n$ considéré comme vecteur sera noté en matrice colonne $x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ dont la transposé, $x^t = (\, x_1 \quad \cdots \quad x_n \,)$, est une matrice $1 \times n$.
La norme euclidienne de x est noté $\|x\|_2 = \left(\sum_{i=1}^{n} x_i^2 \right)^{1/2}$ et le produit scalaire des vecteurs x et y s'écrit $< x, y >= x^t\, y = \sum_{i=1}^{n} x_i y_i$.

Définition 2.0.1 (Dérivée directionnelle)
Soit Ω un ouvert de $\mathbb{R}^n$ et $f : \Omega \to \mathbb{R}$. Soit $a \in \Omega$ et $v \in \mathbb{R}^n$, la dérivée de f au point a, dans la direction v, est définie par

$$D_v f(a) = \lim_{h \to 0} \frac{1}{h} \Big(f(a + h\, v) - f(a) \Big) \qquad (h \in \mathbb{R}) \,.$$

Si $v = e_i$, on obtient la dérivée partielle de f par rapport à la i^evariable, notée, $\dfrac{\partial f}{\partial x_i}(a)$.

Comme une dérivée partielle est encore une fonction de plusieurs variables, on peut de nouveau calculer ses dérivées partielles. Par exemple

$$\frac{\partial}{\partial x_2} \left(\frac{\partial f}{\partial x_1} \right) = \frac{\partial^2 f}{\partial x_2 \partial x_1} \,.$$

Proposition 2.0.1
Soit Ω un ouvert de $\mathbb{R}^n$ et $f : \Omega \to \mathbb{R}$ différentiable au point $a \in \Omega$, alors pour tout $v \in \mathbb{R}^n$:

$$D_v f(a) = \sum_{i=1}^{n} v_i \, \frac{\partial f}{\partial x_i}(a) = < \nabla f(a), v > = \nabla f(a)^t\, v = \|\nabla f(a)\|_2 \|v\|_2 \cos \Big(\text{angle}[\nabla f(a), v] \Big)$$

où $\nabla f(a) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(a) \\ \vdots \\ \frac{\partial f}{\partial x_n}(a) \end{pmatrix}$ *est le gradient de f au point a.*

Application : Soit $a \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$ différentiable en a, on a

$$\forall v \in \mathbb{R}^n, \|v\|_2 = 1 : \quad |D_v f(a)| \leq \|\nabla f(a)\|_2$$

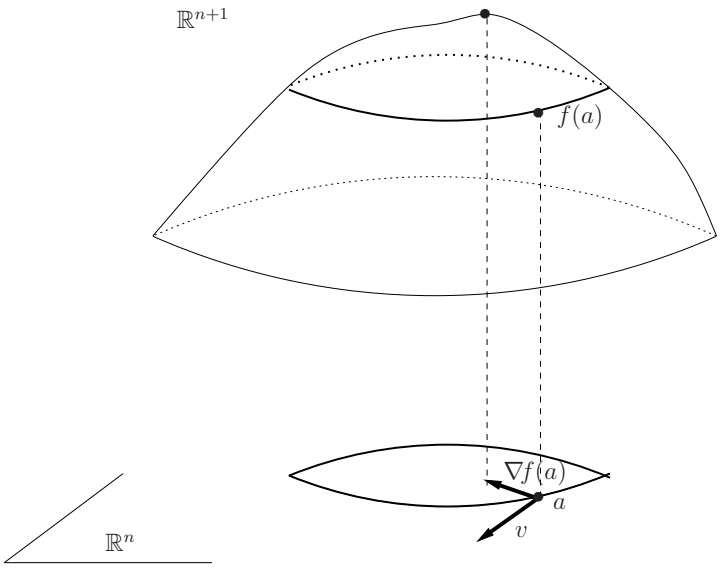
et

$$\min_{\|v\|_2=1} D_v f(a) = -\|\nabla f(a)\|_2 \text{ et est atteint en } v = -\nabla f(a)/\|\nabla f(a)\|_2 \, ;$$

$$\max_{\|v\|_2=1} D_v f(a) = +\|\nabla f(a)\|_2 \text{ et est atteint en } v = +\nabla f(a)/\|\nabla f(a)\|_2 \, .$$

Au point a, la direction de la plus forte croissance de f est donné par $+\nabla f(a)$ et la direction de la plus forte descente est donné par $-\nabla f(a)$.
D'où les algorithmes de minimisation dits de "*descente de gradient*".

Définition 2.0.2
Si $\nabla f(a) = O$, a est un point critique *et localement la fonction est plate.*



Dans la direction $\pm(\nabla f(a))^\perp$, $D_v f(a) = 0$, on reste à la même cote. On montre que le vecteur gradient est perpendiculaire aux lignes de niveau $L_f(\alpha) = \{x \in \mathbb{R}^n \,/\, f(x) = \alpha\}$.

Ceci permet de lire les cartes topographiques où certaines altitudes (*i.e.* niveaux) sont tracées sans avoir recours à une représentation en trois dimensions (voir figure 2.1).

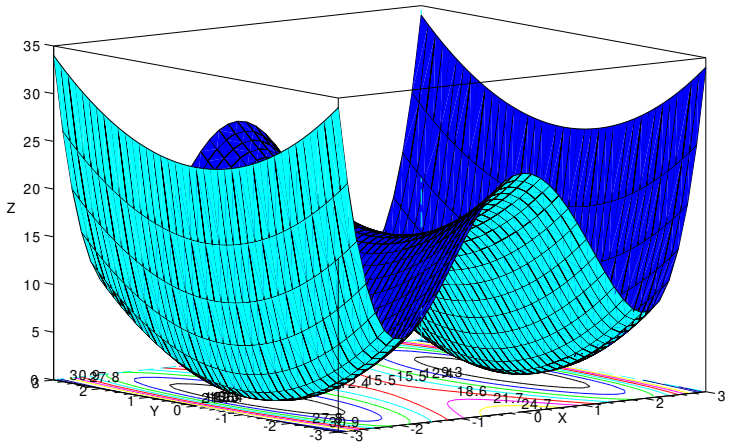


FIGURE 2.1 – Graphe et lignes de niveau de $f(x,y) = (x-2)^2(x+2)^2 + y^2$

DÉFINITION 2.0.3
Soit Ω un ouvert de $\mathbb{R}^n$, $p \in \mathbb{N}$. On dit que la fonction $f : \Omega \to \mathbb{R}$ est de classe p, si toutes les fonctions dérivées partielles, jusqu'à l'ordre p, existent et sont continues.

PROPOSITION 2.0.2
Soit Ω un ouvert de $\mathbb{R}^n$, $f : \Omega \to \mathbb{R}$ et $a \in \Omega$. On suppose que $f \in \mathcal{C}^2(\Omega, \mathbb{R})$, la différentielle d'ordre 2, $d^2 f_a$ est une forme bilinéaire symétrique, dont la matrice s'écrit

$$\nabla^2 f(a) = H_f(a) = \begin{pmatrix} D_{11}f(a) & D_{12}f(a) & \dots & D_{1n}f(a) \\ D_{21}f(a) & D_{22}f(a) & \dots & D_{2n}f(a) \\ \vdots & \vdots & & \vdots \\ D_{n1}f(a) & D_{n2}f(a) & \dots & D_{nn}f(a) \end{pmatrix} = \left(\frac{\partial^2 f}{\partial x_i \partial x_j}(a) \right)_{1 \leq, i, j \leq n} .$$

C'est la matrice hessienne de f en a, pour h, $k \in \mathbb{R}^n$: $d^2 f_a(h,k) = h^t \, \nabla^2 f(a) \, k$.

Note : grâce à régularité $\mathcal{C}^2$ de f on a $\dfrac{\partial^2 f}{\partial x_i \partial x_j} = \dfrac{\partial^2 f}{\partial x_j \partial x_i}$, $1 \leq, i, j \leq n$.

PROPOSITION 2.0.3 (FORMULE DE TAYLOR À L'ORDRE 2)
Soit Ω un ouvert de $\mathbb{R}^n$, $f : \Omega \to \mathbb{R}$ et $a \in \Omega$. On suppose que $f \in \mathcal{C}^2(\Omega, \mathbb{R})$, alors

$$\begin{aligned} f(a+h) &= f(a) + df_a(h) + \tfrac{1}{2} d^2 f_a(h,h) + o(\|h\|^2) \\ &= f(a) + (\nabla f(a))^t \, h + \tfrac{1}{2} h^t \, H_f(a) \, h + o(\|h\|^2) \qquad (h \in \mathbb{R}^n) . \end{aligned}$$

Exemple : $f : \mathbb{R}^2 \to \mathbb{R}$, $a = (a_1, a_2)$ et $h = (h_1, h_2)$:

$$f(a_1 + h_1, a_2 + h_2) = f(a_1, a_2) + \frac{\partial f}{\partial x_1} f(a) h_1 + \frac{\partial f}{\partial x_2} f(a) h_2 + \frac{1}{2} \frac{\partial^2 f}{\partial {x_1}^2}(a) h_1^2 + \frac{1}{2} \frac{\partial^2 f}{\partial {x_2}^2}(a) h_2^2 + \frac{\partial^2 f}{\partial x_1 \partial x_2}(a) h_1 h_2 + o(h_1^2 + h_2^2) \ .$$

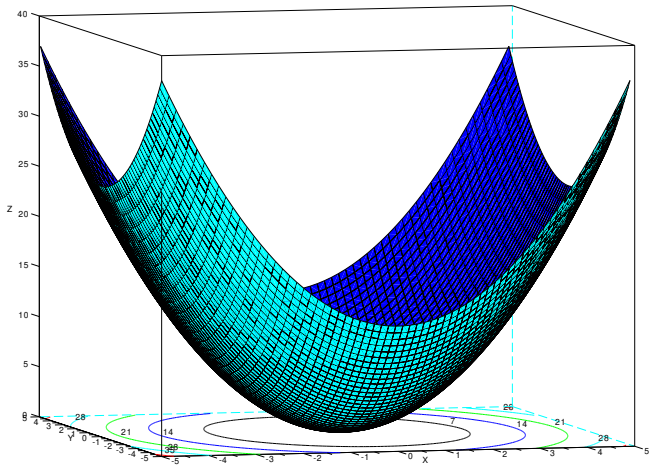


FIGURE 2.2 – Graphe et lignes de niveau d'une fonction quadratique.

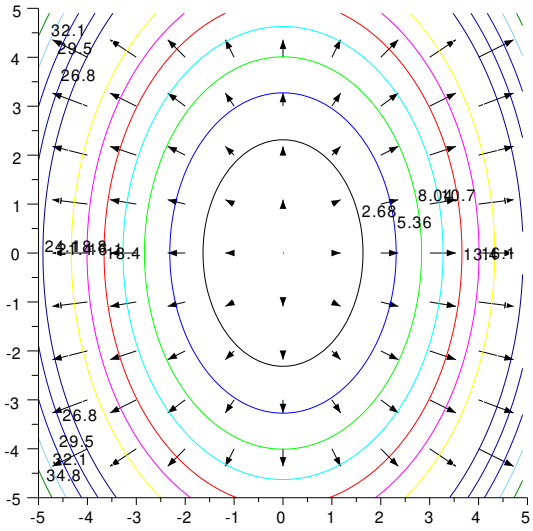


FIGURE 2.3 – Gradient et lignes de niveau d'une fonction quadratique.

Chapitre 3

Optimisation continue sans contraintes

3.1 Problème général d'optimisation continue

Soient f, g et h définies sur $\mathcal{D} \subset \mathbb{R}^n$ avec $\mathcal{D} = \mathbf{dom}\, f \cap \mathbf{dom}\, g \cap \mathbf{dom}\, h$
et $f : \mathcal{D} \to \mathbb{R}$, $g : \mathcal{D} \to \mathbb{R}^m$, $h : \mathcal{D} \to \mathbb{R}^p$.
Un *problème d'optimisation non linéaire* sur $\mathbb{R}^n$ se présente sous la *forme standard* :

$$\begin{cases} \text{Minimiser} & f(x) \\ \text{sous la condition} & x \in \mathfrak{C} = \{x \in \mathcal{D} \mid g(x) \leq 0\,, h(x) = 0\}. \end{cases} \tag{3.2}$$

La fonction (ou fonctionnelle) f est appelée *critère*, *coût*, *fonction-objectif* ou *fonction économique*;
l'ensemble $\mathcal{D} \subset \mathbb{R}^n$ est l'ensemble des *paramètres* ou *variables d'état*;
l'ensemble $\mathfrak{C} \subset \mathcal{D}$ est l'ensemble des *contraintes*, lorsque x se trouve dans $\mathfrak{C}$ on dit que x est *admissible* ou *réalisable*.

Lorsque $\mathfrak{C} = \mathcal{D}$ on a un problème *sans contraintes*, pour $m = 0 < p$ on a des *contraintes-égalités* et pour $p = 0 < m$ on a des *contraintes-inégalités*.
Pour la suite, on ne considère que des problèmes sans contrainte, c'est-à-dire $\mathfrak{C} = \mathcal{D}$, avec $\mathcal{D} = \mathbb{R}^n$ en général. Le problème (3.2) devient

$$\text{Minimiser } f(x) \quad \text{sous la condition } x \in \mathcal{D} \tag{3.4}$$

La *valeur optimale* (ou *minimale*) du problème (3.4) est définie par $\overline{f} = \inf_{x \in \mathcal{D}} f(x)$.
On a $\overline{f} \in \mathbb{R} \cup \{\pm\infty\}$.
On dira que $\overline{x}$ est un *point optimal* (ou *minimal*) si $\overline{x} \in \mathcal{D}$ et $f(\overline{x}) = \overline{f}$.
L'ensemble des points optimaux est $\{x \in \mathcal{D}\,, f(x) = \overline{f}\}$ et vérifie $x = \arg\min_{x \in \mathcal{D}} f(x)$.
Un point $\overline{x} \in \mathcal{D}$ est un *minimum local* (ou *relatif*) (resp. *minimum local strict*)
s'il existe un voisinage V de $\overline{x}$ tel que $f(\overline{x}) \leq f(x)$ pour tout $x \in \mathcal{D} \cap V$
(resp. $f(\overline{x}) < f(x)$ pour tout $x \in \mathcal{D} \cap V$, $x \neq \overline{x}$).

On se restreint ici aux problèmes où les paramètres x appartiennent à un espace vectoriel de dimension finie et sont continues, *i.e.* $x \in \mathbb{R}^n$. On ne considère pas les problèmes

d'*optimisation discrète* ou *combinatoire* où $x \in \mathbb{Z}^n$, ni les problèmes en dimension infinie où x appartient à un espace fonctionnel.

3.2 Caractérisation de points optimaux

Dans cette section on va présenter quelques résultats qui permettent, grâce aux hypothèses sur f (*e.g.* régularité, convexité, coercivité, ...) d'assurer l'existence et/ou l'unicité d'un minimum (local, global, strict, ...) du problème de minimisation $\inf_{x\in\mathcal{D}} f(x)$.

Théorème 3.2.1
Soit U une partie non vide fermée de $\mathbb{R}^n$ et $f : \mathbb{R}^n \to \mathbb{R}$ continue.
Si U est non borné, on suppose que f est coercive, c'est-à-dire que $\lim_{\|x\|\to+\infty} f(x) = +\infty$.
Alors il existe au moins un $\overline{x} \in U$ avec $f(\overline{x}) = \inf_{x\in U} f(x)$.

Théorème 3.2.2
Soit $\mathcal{D}$ un ouvert de $\mathbb{R}^n$ et $f : \mathcal{D} \to \mathbb{R}$ et $\overline{x} \in \mathcal{D}$.

1. *Si $f \in \mathcal{C}^1(\mathcal{D})$ et si $\overline{x}$ est un minimum local de f, alors nécessairement $\nabla f(\overline{x}) = 0$.*
2. *Si $f \in \mathcal{C}^2(\mathcal{D})$ et si $\overline{x}$ est un minimum local de f, alors nécessairement $H_f(\overline{x})$ est semidéfinie positive.*
3. *Soit $f \in \mathcal{C}^2(\mathcal{D})$, si $\nabla f(\overline{x}) = 0$ et $H_f(\overline{x})$ est définie positive, alors $\overline{x}$ est un minimum local strict de f .*

Exemple : Considèrer la fonction $f(x) = x^3$ pour montrer que la réciproque des points 1. et 2. est fausse.

Définition 3.2.1
Un ensemble $C \subset \mathbb{R}^n$ est un **ensemble convexe** *si pour tout $(x, y) \in C^2$ le segment $[x, y] = \{t\,x + (1-t)\,y\,;\, 0 \le t \le 1\}$ est inclus dans C .*

Exemples :

1. L'ensemble vide, l'espace entier $\mathbb{R}^n$ et tout singleton $\{x\}$, $x \in \mathbb{R}^n$, sont des ensembles convexes.
2. Les sous-espaces vectoriels et affines de $\mathbb{R}^n$ sont des ensembles convexes : l'*hyperplan affine* $\{x \in \mathbb{R}^n \,|\, a^t\,x = b\}$, $a \in \mathbb{R}^n$, $b \in \mathbb{R}$ est convexe et partage $\mathbb{R}^n$ en deux *demi-espaces* convexes $\{x \,|\, a^t\,x < b\}$ et $\{x \,|\, a^t\,x > b\}$ (fermés pour l'inégalité large).
3. Les boules $B(x, r)_{\|.\|} \subset \mathbb{R}^n$ sont des ensembles convexes.
4. L'*hyperoctant positif* $(\mathbb{R}_+)^n$ est un exemple simple de cône convexe.

Définition 3.2.2
Soit $C \subset \mathbb{R}^n$ un ensemble convexe non vide et $f : C \to \mathbb{R}$.

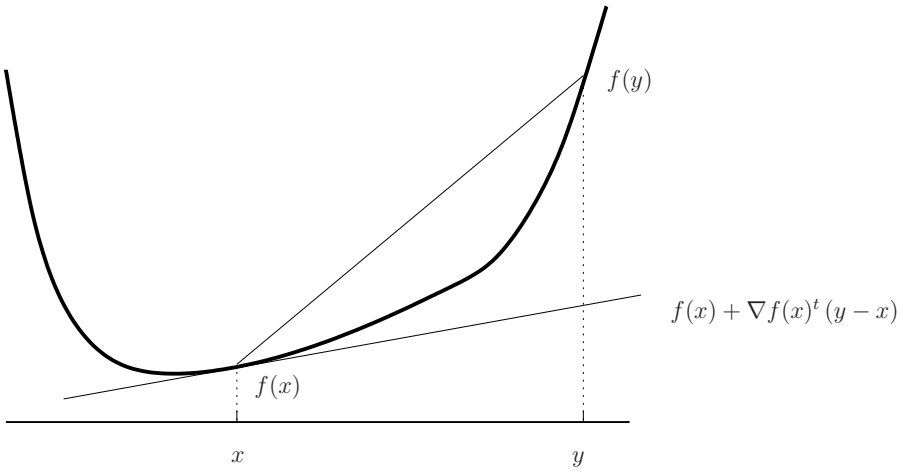
— *On dit que f est une* **fonction convexe** *si pour tout $(x, y) \in C^2$ et pour tout $t \in]0, 1[$*

$$f(t\,x + (1-t)\,y) \leq t\,f(x) + (1-t)\,f(y)\,.$$

f est concave si $-f$ est convexe.

— *La fonction f est strictement convexe si pour tout $(x, y) \in C^2$, $x \neq y$, et pour tout $t \in]0, 1[$*

$$f(t\,x + (1-t)\,y) < t\,f(x) + (1-t)\,f(y)\,.$$



Proposition 3.2.1 (Critères de convexité)
Soit Ω un ouvert convexe non vide de $\mathbb{R}^n$ et $f \in \mathcal{C}^2(\Omega, \mathbb{R})$:

(a) f est convexe sur Ω si et seulement si $H_f(x)$ est positive pour tout $x \in \Omega$;

(b) si $H_f(x)$ est définie positive pour tout $x \in \Omega$, alors f est strictement convexe sur Ω.

Exemples :

1. Les fonctions $x \mapsto e^{ax}$, $a \in \mathbb{R}$ et $x \mapsto |x|^p$, $p \geq 1$ sont convexes sur $\mathbb{R}$.
2. Sur $\mathbb{R}_+^*$ la fonction $x \mapsto x^a$ est convexe pour $a \geq 1$ ou $a \leq 0$ et concave si $0 \leq a \leq 1$. La fonction $x \mapsto \ln x$ est concave sur $\mathbb{R}_+^*$.
3. Sur $\mathbb{R}^n$ toute norme, $x \mapsto \|x\|$, et la fonction $x \mapsto \max\{x_1, \ldots, x_n\}$ sont des fonctions convexes.
4. La moyenne géométrique $f(x) = \left(\prod_{i=1}^{n} x_i\right)^{1/n}$ est concave sur $\mathbf{dom}\, f = (\mathbb{R}_+^*)^n$.

Les résultats suivants montrent que la **convexité** permet de passer de propriétés locales à des propriétés globales.

THÉORÈME 3.2.3
Soit C un convexe non vide de $\mathbb{R}^n$ et $f : C \to \mathbb{R}$ convexe, alors chaque minimum local de f est un minimum global sur C .

THÉORÈME 3.2.4
Soit C un convexe non vide de $\mathbb{R}^n$ et $f : C \to \mathbb{R}$ strictement convexe.
si f admet un minimum dans C, alors ce minimum est global et unique dans C.

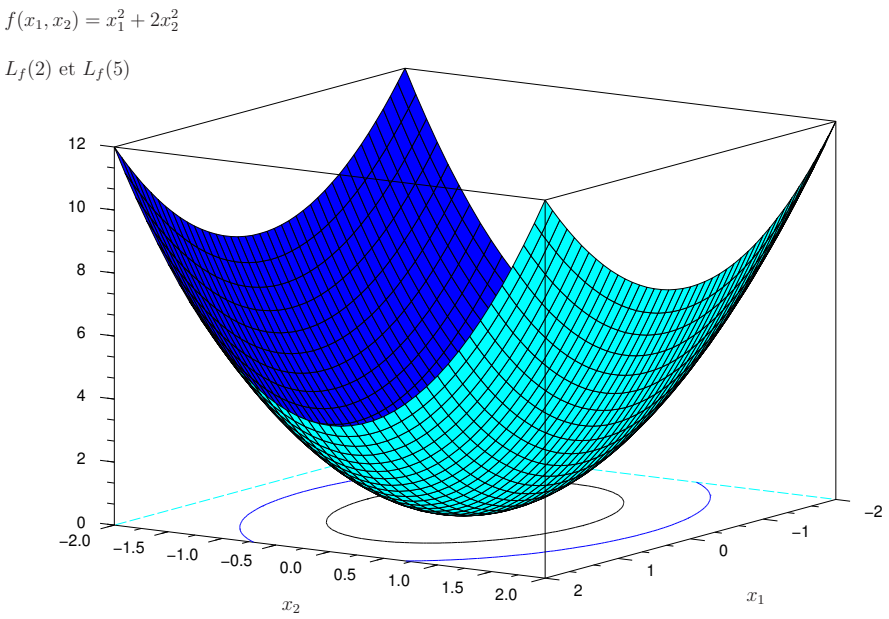
THÉORÈME 3.2.5
Soit Ω un ouvert convexe non vide de $\mathbb{R}^n$, $f \in \mathcal{C}^1(\Omega, \mathbb{R})$ et $f : C \to \mathbb{R}$ convexe sur Ω, alors

$$\nabla f(a) = 0 \quad \Leftrightarrow \quad a \text{ est un minimum global.}$$

Exemple : On veut minimiser sur $\mathbb{R}^n$ la fonction quadratique $f(x) = \frac{1}{2}x^t Ax - b^t x + c$, où A est symétrique semidéfinie positive, $b \in \mathbb{R}^n$ et $c \in \mathbb{R}$. C'est un problème d'optimisation quadratique sans contrainte. On a $\nabla f(x) = Ax - b$ et $H_f(x) = A$.
L'hypothèse sur A entraîne que f est convexe et la CNS d'optimalité pour $\overline{x} \in \mathbb{R}^n$ s'écrit

$$\nabla f(\overline{x}) = A\overline{x} - b = 0\,.$$

Si toutes les valeurs propres de A sont strictement positives (*i.e.* A est défini positive), alors A est inversible et on a une solution unique $\overline{x} = A^{-1}b$.
D'ailleurs comme $H_f(x) = A$, on a f strictement convexe.
Si $b \notin \mathbf{Im}(A)$, il n'y a pas de solution, f n'admet pas de borne inférieure ; si $b \in \mathbf{Im}(A)$, mais A non inversible, il existe une infinité de solutions.



3.3 Algorithmes de minimisation sans contrainte

3.3.1 Méthodes de descente. Vitesse de convergence

Dans ce qui suit on suppose en général que f est au moins de classe $\mathcal{C}^1$ sur l'ouvert $\mathcal{D}$ de $\mathbb{R}^n$. Pour un point $x^{(0)} \in \mathcal{D}$ donné on veut construire une suite $(x^{(\mathrm{k})})_k$ vérifiant

$$\begin{array}{ll} (i) & \left\|x^{(\mathrm{k}+1)} - x^{(\mathrm{k})}\right\| \to 0 \ \text{ pour } k \to +\infty \\ (ii) & \|\nabla f(x^{(\mathrm{k})})\| \to 0 \ \text{ pour } k \to +\infty \\ (iii) & f(x^{(\mathrm{k})}) \leq f(x^{(\mathrm{k}-1)}) \leq \cdots \leq f(x^{(0)})) \ \text{ pour tout } k \geq 1 \end{array}$$

Si l'ensemble $L_f(f(x^{(0)})) = \{x \in \mathbb{R}^n \,/\, f(x) \leq f(x^{(0)})\}$ est compact, on a existence d'une sous-suite qui converge vers un *point stationnaire* $\overline{x}$, *i.e.* $\nabla f(\overline{x}) = 0$.

Algorithme 3.1 (Algorithme de descente)
Un algorithme de descente général se présente sous la forme suivante :

choisir $x^{(0)}$, *poser* $k = 0$
tant que $\left\|\nabla f(x^{(k)})\right\| >$ *seuil de tolérance*
déterminer une direction $d^{(k)}$ *telle que* $\quad \exists \sigma > 0 \ : \ f(x^{(k)} + \sigma\, d^{(k)}) < f(x^{(k)})$
déterminer un pas convenable $\sigma_k > 0$
poser $x^{(k+1)} = x^{(k)} + \sigma\, d^{(k)}$ *et faire* $k = k + 1$

Remarques :

1. On dit que $d \in \mathbb{R}^n$ est une <u>*direction de descente*</u> de f en x si $< \nabla f(x), d >< 0$, c.-à-d. si la dérivée directionnelle de f dans la direction d est négative en x et donc f est décroissante dans la direction d.
   Dans ce cas il existe $\tilde{\sigma} > 0$ tel que pour tout $\sigma \in]0, \tilde{\sigma}[: \ f(x + \sigma d) < f(x)$.

2. Après avoir fixé d, il faut déterminer un pas σ «convenable» :
   on souhaite avoir une descente significative de $f(x^{(\mathrm{k})})$ vers $f(x^{(\mathrm{k}+1)})$, ne pas rester trop près de $x^{(\mathrm{k})}$ afin de converger "vite", ne pas mettre trop de temps à trouver σ, etc.
   Si la fonction coût le permet on peut déterminer un minimum exact de f sur la demie-droite $\{x + \sigma d \,|\, \sigma > 0\}$.

Vitesse de convergence :

On considère une méthode itérative convergente $x^{(\mathrm{k})} \to \overline{x}$.
On dit que la méthode est *d'ordre* $r \geq 1$, s'il existe une constante $C \in \mathbb{R}_+^*$ telle que, pour k suffisamment grand

$$\frac{\left\|x^{(\mathrm{k}+1)} - \overline{x}\right\|}{\left\|x^{(\mathrm{k})} - \overline{x}\right\|^r} \leq C \ .$$

Si $r = 1$ il faut $C \in]0, 1[$ pour avoir convergence et on a alors une *convergence linéaire*.
Si $r = 2$ on a une *convergence quadratique*.
La quantité $-\log_{10} \left\|x^{(\mathrm{k})} - \overline{x}\right\|$ mesure le nombre de décimales exactes dans l'approximation $x^{(\mathrm{k})}$. Si pour une méthode d'ordre un on a asymptotiquement

$$-\log_{10} \left\|x^{(\mathrm{k}+1)} - \overline{x}\right\| \sim -\log_{10} \left\|x^{(\mathrm{k})} - \overline{x}\right\| - \log_{10}(C)$$

alors on gagne à chaque itération $\log_{10}(1/C)$ décimales.
Si pour une méthode d'ordre $r > 1$ on a asymptotiquement

$$-\log_{10}\left\|x^{(\mathrm{k}+1)} - \overline{x}\right\| \sim -r\,\log_{10}\left\|x^{(\mathrm{k})} - \overline{x}\right\| - \log_{10}(C)$$

alors $x^{(\mathrm{k}+1)}$ a r fois plus de décimales exactes que $x^{(\mathrm{k})}$.

3.3.2 Minimisation en une dimension

Dans cette section on suppose que l'on sait trouver à chaque itération une direction de descente $d^{(\mathrm{k})}$. On va s'intéresser au problème de la détermination d'un pas σ_k convenable. Les *conditions de* Wolfe (W1) et (W2) sont utiles pour obtenir des résultats théoriques :

$$f(x^{(\mathrm{k})} + \sigma d^{(\mathrm{k})}) \leq f(x^{(\mathrm{k})}) + c_1\,\sigma\, < \nabla f(x^{(\mathrm{k})}), d^{(\mathrm{k})} > \qquad \text{(W1)}$$

$$< \nabla f(x^{(\mathrm{k})} + \sigma d^{(\mathrm{k})}), d^{(\mathrm{k})} > \geq c_2\, < \nabla f(x^{(\mathrm{k})}), d^{(\mathrm{k})} > \qquad \text{(W2)}$$

avec $0 < c_1 < c_2 < 1$.

Pour expliquer ces conditions posons $\phi(\sigma) = f(x^{(\mathrm{k})} + \sigma d^{(\mathrm{k})})$ et $l(\sigma) = \phi(0) + (c_1\phi'(0))\sigma$.
On a $\phi'(\sigma) =< \nabla f(x^{(\mathrm{k})} + \sigma d^{(\mathrm{k})}), d^{(\mathrm{k})} >$ et $\phi'(0) =< \nabla f(x^{(\mathrm{k})}), d^{(\mathrm{k})} >$.
En particulier $\phi'(0) < 0$ car $d^{(\mathrm{k})}$ est une direction de descente.

La condition (W1) impose que $\phi(\sigma) \leq l(\sigma)$, ne retiendra que les valeurs de σ pour lesquelles le graphe de ϕ est en dessous de la droite l, comme $0 < c_1 < 1$ ceci est possible au moins pour σ petit. La réduction de f est proportionnelle à σ et $\phi'(0)$.

La condition (W2) implique que $\phi'(\sigma) \geq c_2\phi'(0) \geq \phi'(0)$: si $\phi'(\sigma)$ est «très» négatif (*i.e.* plus petit que $c_2\phi'(0)$), on peut chercher plus loin car f est ' «très» décroissante. Sinon on peut s'arrêter. De de cette façon le pas σ choisi ne sera pas trop petit.

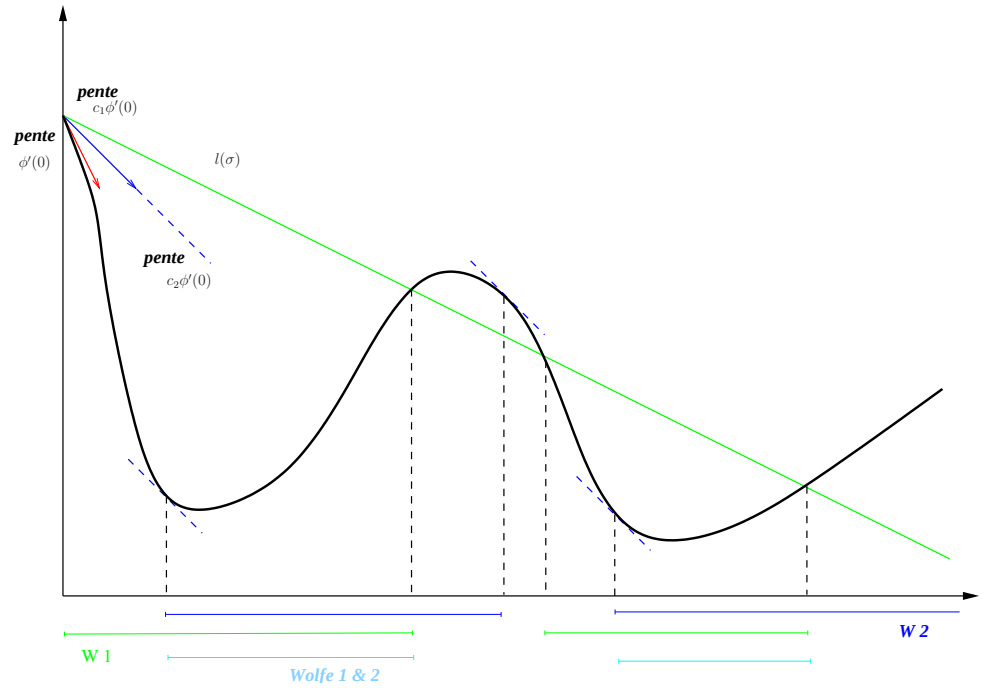
L'existence de valeurs de σ vérifiant (W1-2) est donnée par la

Proposition 3.3.1
Soit $f \in \mathcal{C}^1(\mathbb{R}^n)$ et d une direction de descente de f en x, on suppose que f est minorée sur $\{x + \sigma d \,|\, \sigma \geq 0\}$. Alors, si $0 < c_1 < c_2 < 1$, il existe des intervalles dans $\mathbb{R}_+$ qui vérifient les conditions de Wolfe.

Théorème 3.3.1 (Corollaire du théorème de Zoutendijk)
Soit f de classe $\mathcal{C}^1$ sur l'ouvert $\mathcal{D} \subset \mathbb{R}^n$ et $x^{(0)} \in \mathcal{D}$ tel que l'ensemble de niveau inférieur $L_f(f(x^{(0)})) = \{x \in \mathcal{D} \mid f(x) \leq f(x^{(0)})\}$ est fermé, de plus on suppose que f est minoré sur $L_f(f(x^{(0)}))$ et que ∇f est Lipschitz sur $\mathcal{D}$.
Considérons la suite $(x^{(k)})_k$, définie pour tout $k \geq 0$ par $x^{(k+1)} = x^{(k)} + \sigma_k d^{(k)}$,
où $d^{(k)}$ est une direction de descente et σ_k vérifie les conditions de Wolfe.
On note θ_k l'angle entre les vecteurs $-\nabla f(x^{(k)})$ et $d^{(k)}$.

Alors si il existe $\delta > 0$ tel que pour k suffisamment grand on a $\cos\theta_k \geq \delta$, alors $\lim_k \left\|\nabla f(x^{(k)})\right\|_2 = 0$.

Ceci est un résultat de *convergence globale* : pour des fonctions coût f vérifiant les hypothèses techniques du théorème et si la direction de descente ne devient pas perpendiculaire



au gradient, alors la méthode itérative converge vers un point stationnaire $\overline{x}$ de f, ceci à partir d'un point initial $x^{(0)}$ quelconque !

Il existe des algorithmes qui déterminent un σ_k vérifiant les conditions de Wolfe, nous allons seulement proposer un algorithme simplifié, le *backtracking* ou recherche rétrograde, qui donne souvent de bons résultats en pratique.

Algorithme 3.2 (Algorithme de backtracking)
Choisir σ_{init} , ρ, $c \in]0,1[$
$\sigma = \sigma_{init}$
tant que $f(x^{(k)} + \sigma\, d^{(k)}) > f(x^{(k)}) + c\,\sigma < \nabla f(x^{(k)}), d^{(k)} >$
$\sigma = \rho\sigma$
$\sigma_k = \sigma$

À partir d'une «grande» valeur σ_{init}, on détermine σ_k en diminuant à chaque itération la valeur de σ jusqu'à ce que σ vérifie (W1) et comme l'on part loin de 0 $x^{(\mathrm{k}+1)}$ ne sera pas trop proche de $x^{(\mathrm{k})}$.

3.3.3 Méthode de descente du gradient

Pour cet algorithme on choisit $d^{(\mathrm{k})} = -\nabla f(x^{(\mathrm{k})})$ comme direction de descente, l'algorithme complet s'écrit :

Algorithme 3.3 (Descente de gradient)

choisir $x^{(0)} \in$ **dom** f *et poser* $k = 0$

tant que $\left\|\nabla f(x^{(k)})\right\| > \varepsilon$

$d^{(k)} = -\nabla f(x^{(k)})$

déterminer $\sigma_k > 0$

$x^{(k+1)} = x^{(k)} + \sigma_k\, d^{(k)}$

$k = k + 1$

Grâce au théorème 3.3.1 la méthode de descente du gradient est globalement convergente. Pour étudier la vitesse de convergence on va étudier un cas particulier simple.

Application : On considère le problème d'optimisation quadratique $\min_{x\in\mathbb{R}^n} f(x)\,,$

où f est une fonction quadratique $f(x) = \dfrac{1}{2}\, x^t Q x - b^t x\,,$ avec $b \in \mathbb{R}^n$ et Q une matrice symétrique définie positive de valeurs propres $0 < \lambda_1 \leq \cdots \leq \lambda_n\,.$

On a $\nabla f(x) = Qx - b$ et $H_f(x) = Q\,.$ La fonction f admet un minimum global unique $\overline{x}$ qui est solution de $Qx = b\,.$

On a par ailleurs

$$\phi(\sigma) = f(x^{(\mathrm{k})} + \sigma\, d^{(\mathrm{k})}) = \frac{1}{2}\Big(\nabla f(x^{(\mathrm{k})})^t\, Q\, \nabla f(x^{(\mathrm{k})})\Big)\, \sigma^2 - \|\nabla f(x^{(\mathrm{k})})\|_2^2\, \sigma + f(x^{(\mathrm{k})}).$$

C'est un trinome en σ dont on détermine le minimum unique σ_k comme solution de $\phi'(\sigma) = 0.$

$$x^{(\mathrm{k}+1)} = x^{(\mathrm{k})} - \left[\frac{\left\|\nabla f(x^{(\mathrm{k})})\right\|_2^2}{\nabla f(x^{(\mathrm{k})})^t\, Q\, \nabla f(x^{(\mathrm{k})})}\right]\, \nabla f(x^{(\mathrm{k})})\ .$$

On montre alors que

$$\left\|x^{(\mathrm{k}+1)} - \overline{x}\right\|_Q \leq \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}\right)\, \left\|x^{(\mathrm{k})} - \overline{x}\right\|_Q\ ,$$

Ainsi, dans le cas d'une fonction coût quadratique, la méthode du gradient est d'ordre un et la vitesse de convergence dépend de λ_n/λ_1 : si ce rapport vaut 1, on a convergence en une itération ; si le rapport est grand la convergence devient très lente.

Géométriquement le rapport λ_n/λ_1 indique la déformation de l'ellipsoïde $x^t Q x = \alpha > 0.$

On montre aussi que

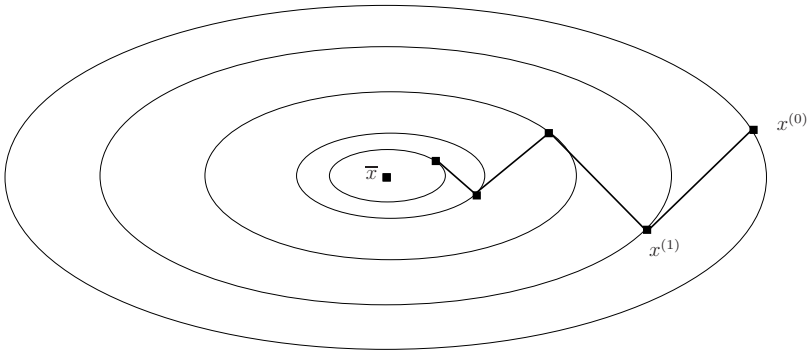
$$< d^{(\mathrm{k}+1)}, d^{(\mathrm{k})} > = < \nabla f(x^{(\mathrm{k}+1)}), \nabla f(x^{(\mathrm{k})}) > = -\phi'(\sigma_k) = 0\,,$$

le chemin de $x^{(0)}$ vers $\overline{x}$ est donc en zigzag, ceci est illustré en dimension deux dans la figure suivante.

3.3.4 Méthode de Newton

On suppose que f est de classe $\mathcal{C}^2$ et on utilise un modèle quadratique de f :

$$f(x^{(\mathrm{k})} + d) \sim \tilde{f}_k(d) = f(x^{(\mathrm{k})}) + \nabla f(x^{(\mathrm{k})})^t\, d + \frac{1}{2}\, d^t H_f(x^{(\mathrm{k})})\, d\,.$$



Si $H_f(x^{(\mathrm{k})})$ est définie positive, la direction $d^{(\mathrm{k})}$ sera le minimum de $\tilde{f}_k$:

$$d^{(\mathrm{k})} = -[H_f(x^{(\mathrm{k})})]^{-1}\,\nabla f(x^{(\mathrm{k})})\ .$$

Dans ce cas $< \nabla f(x^{(\mathrm{k})}),\, d^{(\mathrm{k})} >= -{d^{(\mathrm{k})}}^t H_f(x^{(\mathrm{k})})\, d^{(\mathrm{k})} \leq 0$, on obtient bien une direction de descente.

THÉORÈME 3.3.2
Soit f de classe $\mathcal{C}^2$, $\overline{x} \in \mathbb{R}^n$ tel que $\nabla f(\overline{x}) = 0$ et $H_f(\overline{x})$ définie positive et H_f une fonction Lipschitz au voisinage de $\overline{x}$.
On considère la suite $(x^{(k)})$ définie par $x^{(0)}$ et

$$x^{(k+1)} = x^{(k)} - [H_f(x^{(k)})]^{-1}\,\nabla f(x^{(k)})\ .$$

Alors si $x^{(0)}$ est suffisamment proche de $\overline{x}$
(i) la suite $(x^{(k)})$ converge vers $\overline{x}$;
(ii) la méthode de NEWTON *est d'ordre 2 ;*
(iii) la suite $\left(\left\|\nabla f(x^{(k)})\right\|\right)$ converge vers 0 de façon quadratique.

Remarques :

1. Si en un point stationnaire la matrice hessienne est définie positive et si $x^{(0)}$ est proche de $\overline{x}$, on a une convergence très rapide. Dans le cas contraire l'algorithme peut diverger.
2. Par contre le coût ce cette méthode est grand : à chaque itération on doit construire et garder en mémoire la matrice hessienne et résoudre $H_f(x^{(\mathrm{k})})\, d = \nabla f(x^{(\mathrm{k})})$, en utilisant l'algorithme de CHOLESKY cela fait $O(n^3/3)$ opérations.
3. Les *methodes quasi-*NEWTON remplacent la matrice hessienne par une approximation B_k qui vérifie la relation de la sécante

$$B_k(x^{(\mathrm{k})} - x^{(\mathrm{k\text{-}1})}) = \nabla f(x^{(\mathrm{k})}) - \nabla f(x^{(\mathrm{k\text{-}1})})\ .$$

3.3.5 Méthode du gradient conjugué

Dans cette méthode on utilise encore un modèle quadratique de f, nous allons donc d'abord présenter l'algorithme appliqué à une fonction fortement convexe quadratique.

Cas linéaire

Soit $f(x) = \frac{1}{2}\, x^t A x - b^t x$, avec A symétrique définie positive.
On sait alors qu'il existe un minimum unique sur $\mathbb{R}^n$ donné par $\overline{x} = A^{-1} b$.
Dans la suite on note $r(x) = Ax - b = \nabla f(x)$ le résidu.

Définition 3.3.1
Les vecteurs non nuls $\{d_1, \ldots, d_p\}$ sont dits conjugués par rapport à la matrice A si

$$\textit{pour tout } i \neq j \in \{1, \ldots, p\} \ : \ d_i^t A\, d_j = 0 \ .$$

Lemme 3.3.1
Un ensemble de vecteurs conjugués par rapport à A est un ensemble de vecteurs linéairement indépendants.

Posons $\phi(\sigma) = f(x + \sigma\, d)$, alors $\phi'(\sigma) = 0$ pour

$$\sigma = -\frac{r(x)^t\, d}{d^t A\, d} \ . \tag{3.5}$$

Théorème 3.3.3
Soit $x^{(0)} \in \mathbb{R}^n$ et $\{d_0, \ldots, d_{n-1}\}$ un ensemble de vecteurs conjugués par rapport à A .
On considère la suite définie par

$$x^{(k+1)} = x^{(k)} + \sigma_k\, d_k \ , \qquad \text{où } \sigma_k = -\frac{r(x^{(k)})^t\, d_k}{d_k^t A\, d_k} \ .$$

Alors $r(x^{(k)})^t\, d_i = 0 \quad$ *pour* $i = 0, \ldots, k-1$,
$x^{(k)}$ *minimise f sur* $x^{(0)} + \mathrm{Vect}\{d_0, \ldots, d_{k-1}\}$
et la suite $(x^{(k)})$ converge en au plus n itérations.

Pour pouvoir utiliser la méthode itérative du théorème précédent il faut construire un ensemble de vecteurs conjugués par rapport à A .
Or, afin de réduire le nombre d'opérations, on se propose de déduire $d^{(\mathrm{k})}$ à partir de $d^{(\mathrm{k\text{-}1})}$ uniquement. Dans l'algorithme du gradient conjugué linéaire on prend

$$d^{(\mathrm{k})} = -\nabla f(x^{(\mathrm{k})}) + \beta_k\, d^{(\mathrm{k\text{-}1})}$$

avec β_k tel que ${d^{(\mathrm{k})}}^t A\, d^{(\mathrm{k\text{-}1})} = 0$, on en déduit

$$\beta_k = \frac{r(x^{(\mathrm{k})})^t A\, d^{(\mathrm{k\text{-}1})}}{{d^{(\mathrm{k\text{-}1})}}^t A\, d^{(\mathrm{k\text{-}1})}} \ . \tag{3.6}$$

Comme $< d^{(\mathrm{k})}, \nabla f(x^{(\mathrm{k})}) >= -\left\| r(x^{(\mathrm{k})}) \right\|_2^2$, on obtient bien une direction de descente.

Algorithme 3.4 (Gradient conjugué linéaire)

choisir $x^{(0)}$ et poser $k = 0$

$r^{(0)} = Ax^{(0)} - b\,,\ d^{(0)} = -r^{(0)}\,,\ k = 0$

tant que $\left\|r(x^{(k)})\right\| > \varepsilon$

$$\sigma_k = \frac{r^{(k)^t}\, r^{(k)}}{d^{(k)^t} A\, d^{(k)}}$$

$$x^{(k+1)} = x^{(k)} + \sigma_k\, d^{(k)}$$

$$r^{(k+1)} = r^{(k)} + \sigma_k\, A\, d^{(k)}$$

$$\beta_{k+1} = \frac{r^{(k+1)^t}\, r^{(k+1)}}{r^{(k)^t}\, r^{(k)}}$$

$$d^{(k+1)} = -r^{(k+1)} + \beta_{k+1}\, d^{(k)}$$

$$k = k + 1$$

Noter que les formules pour σ_k et β_{k+1} ont changé, ceci est possible grâce au théorème suivant qui affirme en particulier que les $d^{(\mathrm{k})}$ sont conjugués.

THÉORÈME 3.3.4
Soit $x^{(k)} \neq \overline{x}$, obtenu après la k^e itération de l'algorithme du gradient conjugué linéaire avec σ_k, resp. β_{k+1}, défini par (3.5), resp. (3.6). Alors on a

$$(i) \qquad r^{(k)^t}\, r^{(i)} = 0 \qquad \textit{pour } i = 0, \ldots, k-1\ ;$$

$$(ii) \quad \mathrm{Vect}\{r^{(0)}, \ldots, r^{(k)}\} = \mathrm{Vect}\{r^{(0)}, Ar^{(0)}, \ldots, A^k r^{(0)}\}\ ;$$

$$(iii) \quad \mathrm{Vect}\{d^{(0)}, \ldots, d^{(k)}\} = \mathrm{Vect}\{r^{(0)}, Ar^{(0)}, \ldots, A^k r^{(0)}\}\ ;$$

$$(iv) \qquad d^{(k)^t} A\, d^{(i)} = 0 \qquad \textit{pour } i = 0, \ldots, k-1\ .$$

Le sous-espace vectoriel $\mathrm{Vect}\{r^{(0)}, Ar^{(0)}, \ldots, A^k r^{(0)}\}$ est l'*espace de* KRYLOV $K_{k+1}(A, r^{(0)})$.

Remarques :

1. Dans la démonstration du théorème il est essentiel de choisir $d^{(0)} = -\nabla f(x^{(0)})$, c'est à partir de $d^{(0)}$ que l'on construit des gradients orthogonaux et des directions conjugées.
2. L'algorithme du GC linéaire est surtout utile pour résoudre des grands systèmes creux, en effet il suffit de savoir appliquer la matrice A à un vecteur.

Vitesse de convergence.

On note $\|.\|_A$ la norme associé à A et définie par $\|x\|_A^2 = x^t\, A\, x$.
Soit $k \in \{0, \ldots, n-1\}$, il existe alors un polynôme de degré k , noté $P_k^* \in \mathbb{R}_k[X]$, tel que $x^{(k+1)} = x^{(0)} + P_k^*(A)r^{(0)}$ et qui est solution de

$$\min_{P_k \in \mathbb{R}_k[X]} \|x^{(0)} + P_k(A)r^{(0)} - \overline{x}\|_A^2 \ ,$$

où l'on note $\overline{x} = A^{-1}b$ le minimum de f.
Soit $x^{(0)} - \overline{x} = \sum_{i=1}^{n} \xi_i v_i$, où les v_i sont les vecteurs propres associés aux valeurs propres de $A : 0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$, les ξ étant les coordonnées de $x^{(0)} - \overline{x}$ dans la base de ces vecteurs propres.
On a le résultat général suivant :

$$\|x^{(k+1)} - \overline{x}\|_A^2 \leq \left(\min_{P_k \in \mathbb{R}[X]} \max_{1 \leq i \leq n} [1 + \lambda_i P_k(\lambda_i)]^2 \right) \|x^{(0)} - \overline{x}\|_A^2 \ .$$

- Si A n'admet que r valeurs propres distinctes $0 < \tau_1 < \ldots < \tau_r$, où $0 < r < n$, on pose

  $$Q(X) = \frac{(-1)^r}{\tau_1 \cdots \tau_r} \prod_{i=1}^{r} (X - \tau_i) \ \text{ et } \tilde{P}(X) = \frac{1}{X}(Q(X) - 1) \ .$$

  Alors $\tilde{P} \in \mathbb{R}_{r-1}[X]$ et on vérifie que $x^{(r)} = \overline{x}$: le minimum est atteint après r itérations.

- Si les valeurs propres de A sont telles que $0 < \lambda_1 \leq \lambda_{n-m} < \lambda_{n-m+1} \leq \lambda_n$, $1 < m < n$, on pose $Q(X) = C \prod_{i=n-m+1}^{n} (X - \lambda_i) \left(X - \frac{\lambda_1 + \lambda_{n-m}}{2} \right)$,
  où la constante C telle que $Q(X) = 1 + XP_m(X)$, avec $P_m \in \mathbb{R}_m[X]$. On obtient alors

  $$\|x^{(m+1)} - \overline{x}\|_A^2 \leq \left(\frac{\lambda_{n-m} - \lambda_1}{\lambda_{n-m} + \lambda_1} \right)^2 \|x^{(0)} - \overline{x}\|_A^2 \ .$$

  Donc, si $\lambda_1 \approx 1 \approx \lambda_{n-m} << \lambda_{n-m+1}$, on a $\lambda_{n-m} - \lambda_1 \approx 2\varepsilon$ et $\lambda_{n-m} + \lambda_1 \approx 2$, d'où

  $$\|x^{(m+1)} - \overline{x}\|_A^2 \leq \varepsilon^2 \|x^{(0)} - \overline{x}\|_A^2 \ ,$$

  c-à-d après $m+1$ itérations on obtient une bonne approximation de $\overline{x}$.

Pour accélérer la convergence on applique souvent des préconditionneurs, c.-à-d. on transforme le problème grâce à un changement de variables $x_{nouv} = Cx_{anc}$ avec C telle que le conditionnement de la matrice $C^{-t}AC^{-1}$ soit meilleur que celui de A.

Cas non linéaire

Pour pouvoir appliquer l'algorithme du GC linéaire au cas d'une fonction f quelconque il faut :

remplacer $r(x^{(k)})$ par $\nabla f(x^{(k)})$;
déterminer σ_k par une minimisation en dimension un.

L'algorithme s'écrit alors :

Algorithme 3.5 (Gradient conjugué [Fletcher-Reeves 1964])

choisir $x^{(0)}$

calculer $f^{(0)} = f(x^{(0)})$ *et* $\nabla f^{(0)} = \nabla f(x^{(0)})$

$d^{(0)} = -\nabla f^{(0)}$, $k = 0$

tant que $\left\|\nabla f^{(k)})\right\| > \varepsilon$

déterminer σ_k

$x^{(k+1)} = x^{(k)} + \sigma_k\, d^{(k)}$

calculer $\nabla f^{(k+1)} = \nabla f(x^{(k+1)})$

$$\beta_{k+1}^{FR} = \frac{\nabla f^{(k+1)^t}\, \nabla f^{(k+1)}}{\nabla f^{(k)^t}\, \nabla f^{(k)}}$$

$$d^{(k+1)} = -\nabla f^{(k+1)} + \beta_{k+1}^{FR}\, d^{(k)}$$

$k = k + 1$

Si f est une fonction fortement convexe quadratique et si σ_k est un minimum exact, alors cet algorithme est équivalent au GC linéaire. Dans le cas général $d^{(\mathrm{k})}$ peut ne pas être une direction de descente :

$$d^{(\mathrm{k})^t}\, \nabla f(x^{(\mathrm{k})}) = -\left\|\nabla f(x^{(\mathrm{k})})\right\|_2^2 + \beta_{k+1}^{FR}\, d^{(\mathrm{k\text{-}1})^t}\, \nabla f(x^{(\mathrm{k})})\ .$$

Si σ_{k-1} n'est pas un point stationnaire de $\sigma \mapsto f(x^{(\mathrm{k\text{-}1})} + \sigma\, d^{(\mathrm{k\text{-}1})})$ on peut avoir $\nabla f(x^{(\mathrm{k})})^t\, d^{(\mathrm{k})} > 0$. On montre que si σ_{k-1} vérifie les conditions de Wolfe fortes avec $0 < c_1 < c_2 < 1/2$, alors tout $d^{(\mathrm{k})}$ est une direction de descente.

Dans l'algorithme du GC on peut donner diverses expressions de β_{k+1} qui sont équivalentes dans le cas d'une fonction fortement convexe quadratique, c.-à-d. que l'on obtient à chaque fois l'algorithme du GC linéaire. L'expression suivante donne lieu à l'algorithme du GC de Polak-Ribière (1969) :

$$\beta_{k+1}^{PR} = \frac{\nabla f^{(\mathrm{k+1})^t}\, (\nabla f^{(\mathrm{k+1})} - \nabla f^{(\mathrm{k})})}{\nabla f^{(\mathrm{k})^t}\, \nabla f^{(\mathrm{k})}}\ .$$

Cet algorithme est en général plus performant que celui de Fletcher-Reeves et est utilisé le plus souvent en pratique. On peut noter qu'il existe un contre exemple qui montre que le GC de Polak-Ribière est divergent même si σ est un minimum exact.