UFR DE MATHÉMATIQUES ET INFORMATIQUE

MASTER 1
MATHÉMATIQUES APPLIQUÉES

Optimisation et algorithmique

Georges Koepfler 2014-2016 - georges.koepfler@mi.parisdescartes.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.

Une référence très utile pour suivre ce cours est :

- P.G. Ciarlet, *Introduction à l'analyse matricielle et à l'optimisation*, Masson 1990.

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 :

- A. Auslender, *Optimisation. Méthodes Numériques*, Masson 1976 ;
- S. Boyd et L. Vandenberghe, *Convex Optimization*, Cambridge University Press 2004 ;
- R. Fletcher, *Practical Methods of Optimization*, J. Wiley & Sons 1987 ;
- J.B. Hiriart-Urruty et C. Lemaréchal, *Convex Analysis and Minimization Algorithms*, Vol. I et II, Springer 1996 ;
- J. Nocedal et S.J. Wright, *Numerical Optimization*, Springer Series in Operations Research 1999.

Chapitre 1

Introduction

Dans ce cours on va s'intéresser à la résolution numérique des problèmes suivants :

- Résolution de systèmes d'équations linéaires : $Ax = b$ où A est une matrice réelle. On présentera (resp. rappellera) des méthodes directes de factorisation de A et des méthodes itératives de résolution du système. Ces méthodes sont rappelées car utilisées dans la suite du cours.
- Optimisation continue : on considère une fonction coût réelle que l'on veut minimiser sans contraintes et l'on va présenter divers algorithmes classiques.

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.
   Les formules de quadrature pour calculer des intégrales et les schémas aux différences finies pour les équations aux dérivées partielles sont des exemples de formulations discrètes.
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 :
$f = (-1)^s \cdot 0, d_{-1}d_{-2}\dots d_{-r} \cdot b^j$, $m \le j \le M$ et $0 \le d_{-i} \le b-1$;
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.* $j \in \{-127, \dots, +128\}$, 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 $f = (-1)^s\, 1, d_{-1}d_{-2}\dots d_{-r}\, 2^j$.
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* $= \infty - \infty = \sqrt{-1} = 0/0 \dots$) 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 $O(N)$, $O(N \ln N)$, $O(N^2)$, $O(N!)\dots$

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

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

On peut aussi associer à p sa *matrice compagne*

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

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 $p(x)$:

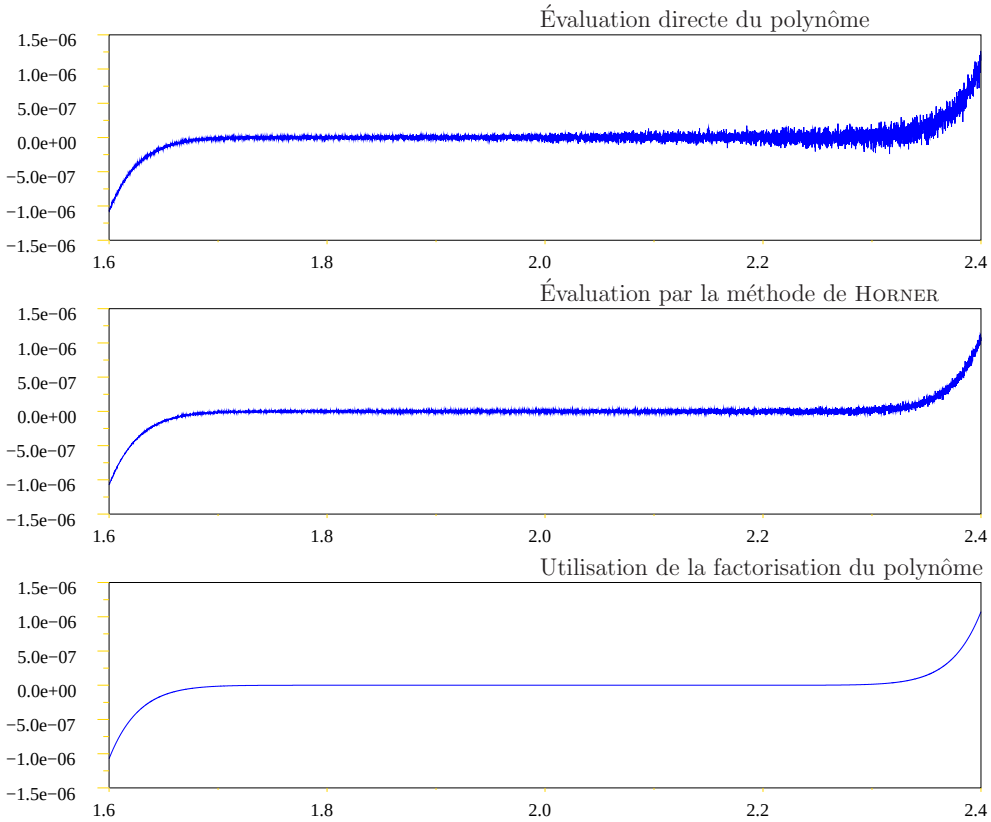
ALGORITHME 1.1 (HORNER)
Évaluation d'un polynôme en x à partir de ses coefficients $a_0, a_1, \ldots, a_d$.

$y = a_d$
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.

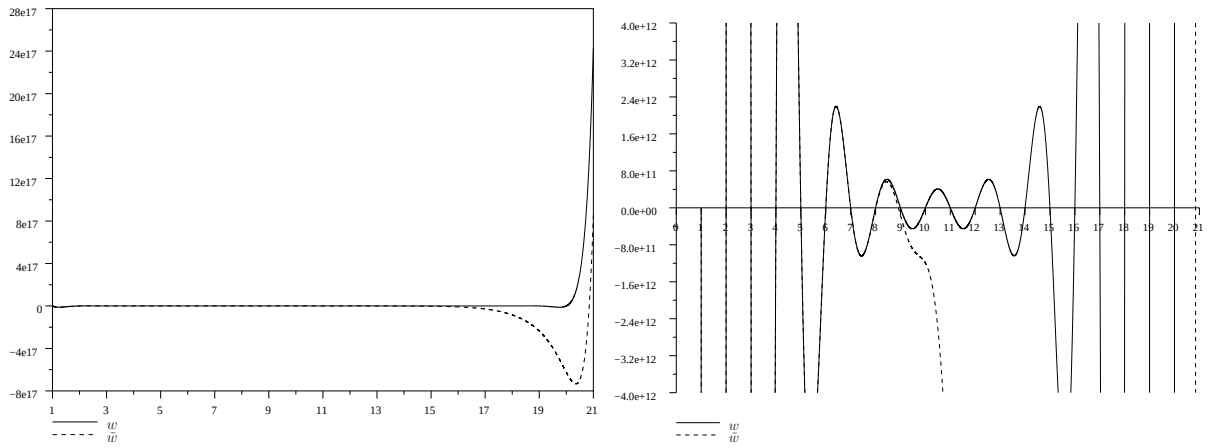
Exemple 2 : Détermination des racines d'un polynôme perturbé.
Pour illustrer l'instabilité de la résolution d'une équation polynomiale par rapport aux coefficients du polynôme on va utiliser l'exemple de proposé par Wilkinson (1963) :

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

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 $\tilde{w}$.



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.

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

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

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.
On peut noter aussi que des logiciels tels que `Scilab`[2] ou `Matlab`©, utilisent des opérations par blocs pour accroître leur performances (et il est fortement recommandé d'en faire usage en TP !).

2. © INRIA, logiciel disponible à `http//www.scilab.org/`

Chapitre 2

Algèbre linéaire

Dans la première partie de ce chapitre nous allons proposer des méthodes de résolution directes de l'équation $Ax = b$, où A est une matrice carrée d'ordre n inversible. Dans la seconde partie on s'intéresse au cas où A est rectangulaire, on cherchera x minimisant $\|Ax - b\|_2$. La dernière section présente des méthodes itératives pour le cas d'une grande matrice A carrée d'ordre n inversible.

2.1 Résolution de systèmes linéaires par des méthodes directes

On s'intéresse à l'équation $Ax = b$. En un premier temps nous allons étudier la stabilité de ce problème. Pour cela on considère le problème perturbé
$(A + \delta A)\tilde{x} = b + \delta b$, et on pose $\delta x = \tilde{x} - x$.
Par soustraction on trouve $\delta x = A^{-1}(\delta b - \delta A\tilde{x})$ et

$$\frac{\|\delta x\|}{\|\tilde{x}\|} \leq \|A^{-1}\| \, \|A\| \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|A\|\|\tilde{x}\|} \right) .$$

Le nombre $c(A) = \|A^{-1}\| \, \|A\|$ est le *conditionnement* de la matrice A pour le problème d'inversion (et pour la norme $\|.\|$). Si de plus $\|A^{-1}\| \, \|\delta A\| < 1$

$$\frac{\|\delta x\|}{\|x\|} \leq \frac{c(A)}{1 - c(A)\frac{\|\delta A\|}{\|A\|}} \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|} \right) .$$

Le théorème suivant donne une caractérisation intéressante du nombre $c(A)$.

Théorème 2.1.1
Soit A une matrice inversible, alors

$$\min \left\{ \frac{\|\delta A\|_2}{\|A\|_2} \Big/ A + \delta A \text{ singulière} \right\} = \frac{1}{\|A^{-1}\|_2 \|A\|_2} = \frac{1}{c_2(A)} .$$

Donc la distance relative de A à la matrice singulière la plus proche est l'inverse du conditionnement de A en $\|.\|_2$.

2.1.1 Factorisation LU de matrices

L'algorithme de résolution est basé sur le

Théorème 2.1.2 (Élimination de Gauss)
Soit A une matrice inversible, alors il existe des matrices de permutations P_1 et P_2, une matrice L triangulaire inférieure avec des 1 sur la diagonale et une matrice U triangulaire supérieure inversible telles que

$$P_1 A P_2 = L U .$$

En fait une seule matrice de permutation est nécessaire, de plus
(i) on peut choisir $P_2 = I$ et P_1 tel que a_{11} soit l'élément de valeur absolue maximale dans la première colonne, c'est l'algorithme d'élimination de Gauss avec pivot partiel.
(ii) on peut choisir P_1 et P_2 de façon à ce que a_{11} soit l'élément de valeur absolue maximale dans toute la matrice, c'est l'algorithme d'élimination de Gauss avec pivot total.

Remarque : Si $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$ est une permutation, alors la matrice associée P_σ est définie par $(P_\sigma)_{ij} = \delta_{i\sigma(j)}$, où δ_{ij} est le symbole de Kronecker. $P_\sigma A$ réordonne les lignes de A, AP_σ réordonne les colonnes de A et $P_\sigma A P_{\sigma'}$ réordonne tout.

Algorithme 2.1
Factorisation LU avec pivot.

pour $i = 1$ ***to*** $n - 1$
  Appliquer les permutations à A, L et U telles que $a_{ii} \neq 0$.
  pour $j = i + 1$ ***to*** n
    $l_{ji} = a_{ji}/a_{ii}$
  pour $j = i$ ***to*** n
    $u_{ij} = a_{ij}$
  pour $j = i + 1$ ***to*** n
    pour $k = i + 1$ ***to*** n
      $a_{jk} = a_{jk} - l_{ji} \cdot u_{ik}$

L'algorithme de factorisation LU est utilisé pour résoudre le système linéaire $Ax = b$.

Algorithme 2.2
Résoudre $Ax = b$ en utilisant l'élimination de Gauss avec pivot (cf. Scilab).

Factoriser A en $PA = LU$;
Par permutation : $LUx = Pb$;
Résoudre un système triangulaire inférieur : $Ly = Pb$;
Résoudre un système triangulaire supérieur : $Ux = y$;
Par permutation : $x = A^{-1}b = U^{-1}L^{-1}Pb$.

Coût : La résolution du système $Ax = b$ revient donc à la résolution de deux systèmes triangulaires qui demandent peu d'opérations, $O(n^2)$. La décomposition LU est d'ordre $O(\frac{2}{3}n^3)$ et le coût total de l'élimination de Gauss pour résoudre un système linéaire est donc $O(\frac{2}{3}n^3)$.
On n'a pas tenu compte des mouvements de données nécessaires pour le pivot par exemple, des implémentations effectives de l'algorithme de factorisation LU optimisent aussi cette partie de la méthode (*cf.* implémentations BLAS).

Pour pouvoir garantir la précision des résultats on aimerait avoir des bornes sur l'erreur relative, or pour cela il faut déterminer $\|\delta A\|/\|A\|$, $\|\delta b\|/\|b\|$ et $c(A) = \|A\|\|A^{-1}\|$.
Pour obtenir $\|\delta A\|$ on utilise le théorème suivant

THÉORÈME 2.1.3
Soit $\tilde{x}$ la solution calculée et $r = A\tilde{x} - b$ le résidu. Il existe une matrice δA tel que $\|\delta A\| = \frac{\|r\|}{\|\tilde{x}\|}$ et $(A + \delta A)\tilde{x} = b$. Il n'y pas de matrice de norme plus petite vérifiant $(A + \delta A)\tilde{x} = b$.

Pour déterminer $c(A)$ on doit calculer $\|A^{-1}\|$, c.-à-d. A^{-1} de façon explicite, ce qui est trop coûteux. On préfère utiliser un bon estimateur de $\|A^{-1}\|$. L'algorithme de HAGER, présenté plus loin, donne une borne inférieure de la norme $\|.\|_1$ d'une matrice B, il est basé sur une maximisation de la fonction $f(x) = \|Bx\|_1 = \sum_{i=1}^{n} \left| \sum_{j=1}^{n} b_{ij} x_j \right|$.

(voir page 21)

2.1.2 Matrices réelles définies positives

Dans beaucoup d'applications la matrice A est symétrique définie positive, dans ce cas on économise de l'espace mémoire en ne stockant que $n(n+1)/2$ termes. De plus il existe un algorithme de factorisation en $O(\frac{1}{3}n^3)$. Le théorème suivant regroupe quelques résultats utiles.

THÉORÈME 2.1.4

1. *Si $A = (a_{ij})_{1 \le i,j \le n}$ est une matrice symétrique définie positive, alors toute sous-matrice $M_m = (a_{ij})_{1 \le i,j \le m}$ est symétrique définie positive.*
2. *Soit B une matrice inversible, alors A est symétrique définie positive si et seulement si $B^t AB$ est symétrique définie positive.*
3. *Une matrice A est symétrique définie positive si et seulement si $A^t = A$ et toutes ses valeurs propres sont strictement positives.*
4. *Si A est symétrique définie positive, alors $\max_{1 \le i,j \le n} |a_{ij}| = \max_{1 \le i \le n} a_{ii} > 0$.*
5. *Une matrice A est symétrique définie positive si et seulement si il existe une matrice L triangulaire inférieure inversible telle que $A = L\,L^t$.*
   C'est la factorisation de CHOLESKY de la matrice A.
6. *Si A est une matrice symétrique définie positive, de valeurs propres $0 < \lambda_1 \le \cdots \le \lambda_n$, alors pour tout $x \in \mathbb{R}^n \setminus \{0\}$ on a l'inégalité de KANTOROVICH :*

$$\frac{(x^t x)^2}{(x^t Ax)(x^t A^{-1} x)} \ge \frac{4\lambda_1 \lambda_n}{(\lambda_1 + \lambda_n)^2} = 1 - \left(\frac{c_2(A) - 1}{c_2(A) + 1} \right)^2 ,$$

   avec $c_2(A) = \lambda_n / \lambda_1$.

Algorithme 2.3 (Cholesky)
Factorisation de Cholesky *d'une matrice symétrique définie positive.*

pour $j = 1$ ***to*** n

$$l_{jj} = \left(a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2 \right)^{\frac{1}{2}}$$

pour $i = j+1$ ***to*** n

$$l_{ij} = \frac{1}{l_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk} \right)$$

Pour résoudre $Ax = LL^t x = b$ on doit résoudre deux systèmes triangulaires.

2.1.3 Autres types de matrices

Dans les applications on rencontre souvent d'autres types de matrices : des matrices bande, tridiagonales, tridiagonales par blocs, creuses ...
Avec des méthodes adaptées on peut souvent améliorer les performances des algorithmes de factorisation, que ce soit au niveau de l'espace mémoire, du nombre d'opérations ou du transfert de données.
La plupart des librairies scientifiques (*e.g.* LAPACK, CLAPACK) ou logiciels (*e.g.* `Scilab`, `Matlab`) proposent des algorithmes et représentations pour les types de matrices les plus fréquents (*e.g.* matrices creuses).

2.2 Résolution de systèmes linéaires au sens des moindres carrées

Soit A une matrice de taille (m, n), un vecteur $x \in \mathbb{R}^n$ est solution du système $Ax = b$ au sens des moindres carrées si x minimise $\|Ax - b\|_2$.
Si $m = n$ et A inversible il existe une solution unique $x = A^{-1}b$; si $m > n$ on a un problème *surdéterminé*, en général il n'existe pas de solution de l'équation $Ax = b$; si $m < n$ on a un problème *sous-déterminé*, on peut avoir une infinité de solutions vérifiant le système linéaire.

Nous allons énoncer quelque résultats d'algèbre linéaire qui permettent de résoudre ce problème, les algorithmes de factorisation présentés font partie des librairies mathématiques standard.

2.2.1 Matrices de Householder

Définition 2.2.1
On appelle matrice de Householder *toute matrice de la forme*

$$H = I_m - 2v\,v^t$$

où $v \in \mathbb{R}^m$ *et* $\|v\|_2 = 1$.

On montre que H est une matrice symétrique et que $H\,H^t = I$; une matrice de Householder correspond à une symétrie par rapport à l'hyperplan perpendiculaire à v.

Soit $x \in \mathbb{R}^m$ tel que $\alpha^2 = x_k^2 + \ldots + x_m^2 > 0$, et soit $v \in \mathbb{R}^m$ tel que

$$v_i = \begin{cases} 0 & \text{pour } 1 \leq i \leq k-1 \\ x_k + \text{signe}(x_k)|\alpha| & \text{pour } i = k \\ x_i & \text{pour } k+1 \leq i \leq m \end{cases},$$

alors si $H = I - 2v\,v^t$, on a $Hx = x - v$ et $(Hx)_i = 0$ pour $k+1 \leq i \leq m$.

Soit A une matrice de taille (m,n), la *méthode de* Householder de résolution du système linéaire $Ax = b$ équivaut à construire des matrices de Householder H_1,..., H_n telles que la matrice $H_n H_{n-1} \cdots H_1 A$ soit triangulaire supérieure.
Il suffit ensuite de résoudre un système triangulaire supérieur par la méthode de la remontée.
Si $m = n$ il suffit d'appliquer $n-1$ transformation pour obtenir une matrice triangulaire supérieure, si $m > n$ on annule $m - n$ lignes de la matrice A.

Les matrices H étant orthogonales on a $\|Ax - b\|_2 = \|HAx - Hb\|_2$,
le résidu du système triangulaire obtenu par la méthode de Householder est identique en $\|.\|_2$ au résidu du problème initial.

2.2.2 Factorisation QR

Théorème 2.2.1 (Factorisation QR)
Soit A une matrice de taille (m,n) avec $m \geq n$ et telle que $\text{rang}(A) = n$. *Alors il existe une unique matrice Q de taille (m,n) telle que $Q^tQ = I_n$; une unique matrice R de taille (n,n) triangulaire supérieure dont les éléments diagonaux sont positifs, $r_{ii} > 0$, telles que $A = QR$.*

Grâce à la factorisation QR de A on a :

$$Ax - b = QRx - b = Q(Rx - Q^tb) - (I_m - QQ^t)b\,.$$

Les vecteurs $Q(Rx - Q^tb)$ et $(I_m - QQ^t)b$ étant orthogonaux on obtient :

$$\|Ax - b\|_2^2 = \left\|Q(Rx - Q^tb)\right\|_2^2 + \left\|(I_m - QQ^t)b\right\|_2^2\,,$$

cette somme de carrées est minimale pour $x = R^{-1}Q^tb$.

On peut utiliser les matrices de Householder pour obtenir une factorisation QR.
En effet $H_n \cdots H_1 A = A_n$ est une matrice (m,n) dont les $m-n$ dernières lignes sont nulles. Grâce aux propriétés des H_i on a $A = H_1 \cdots H_n A_n = QR$ où
$Q = (H_1 \cdots H_n)(1 \leq i \leq m; 1 \leq j \leq n)$, $Q^t\,Q = I_n$,
et $R = A_n(1 \leq i \leq n; 1 \leq j \leq n)$ triangulaire supérieure.

2.2.3 Décomposition en valeurs singulières

La décomposition en valeur singulières d'une matrice est souvent appelé décomposition SVD (angl. *Singular Values Decomposition*) .

THÉORÈME 2.2.2 (DÉCOMPOSITION SVD)
Soit A une matrice de taille (m,n) avec $m \geq n$.
a) On peut écrire $A = U\Sigma V^t$, où U est une matrice de taille (m,n) telle que $U^t U = I_n$; Σ est une matrice diagonale avec $\sigma_1 \geq \ldots \geq \sigma_n \geq 0$; V est de taille (n,n) et vérifie $V^t V = I_n$.
Les σ_i sont les valeurs singulières de A .
b) Les valeurs propres de la matrice $A^t A$ sont σ_i^2 , les vecteurs propres sont les vecteurs colonne $V_1, \ldots, V_n$.
c) Les valeur propres de la matrice AA^t sont σ_i^2 et $m-n$ zéros. Les vecteurs propres sont les $U_1, \ldots, U_n$ et $m-n$ vecteurs propres associés à 0 .
d) On pose $A_k = \sum_{i=1}^{k} \sigma_i U_i V_i^t$, alors $\|A - A_k\|_2 = \min_{B/rg(B)=k} \|A - B\|_2 = \sigma_{k+1}$.
e) Si $m = n$, alors $\|A\|_2 = \sigma_1$, si de plus et A inversible, alors $\|A^{-1}\|_2 = 1/\sigma_n$ et $c_2(A) = \dfrac{\sigma_1}{\sigma_n}$.

Si A est une matrice de taille (m,n) avec $m \geq n$ et $\text{rang}(A) = n$, on montre que la solution du problème des moindres carrées est $x = V\Sigma^{-1}U^t\, b$.

DÉFINITION 2.2.2
Soit A une matrice de taille (m,n) avec $m \geq n$ et $\text{rang}(A) = n$.
On appelle matrice pseudo-inverse de MOORE-PENROSE *la matrice*

$$A^+ = R^{-1}Q^t = V\Sigma^{-1}U^t \ .$$

On peut donc exprimer x minimisant $\|Ax - b\|_2$ grâce à A^+ .

On peut généraliser la définition de A^+ à des matrices de rang $r < n$ comme suit :
soit $A = U\Sigma V^t$, on écrit

$$A = (U_1\, U_2) \begin{pmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{pmatrix} (V_1\, V_2)^t = U_1\, \Sigma_1\, V_1^t \ ,$$

où Σ_1 est de taille (r,r) inversible et les matrices U_1 et V_1 ont r colonnes. On pose $A^+ = V_1\Sigma_1^{-1}U_1^t = V \begin{pmatrix} \Sigma_1^{-1} & 0 \\ 0 & 0 \end{pmatrix} U^t$.
Dans ce cas on a encore $x = A^+ b$ qui minimise $\|Ax - b\|_2$.

2.3 Résolution de systèmes linéaires par des méthodes itératives

Les méthodes directes présentées jusqu'ici permettent de calculer la solution d'un système linéaire en un nombre fini de pas (solution exacte en ne tenant pas compte des erreurs d'arrondi). Par contre pour de grands systèmes ces méthodes demandent souvent trop d'espace mémoire et trop de calculs. On utilise alors des méthodes itératives où l'on arrête l'itération dès que la précision désirée est atteinte.

Avant de brièvement rappeler les méthodes les plus courantes, nous allons donner quelques résultats généraux.

Soit A carrée d'ordre n inversible , on appelle *décomposition* de A tout couple de matrices (M, N) où M est inversible, et tel que $A = M - N$.

Le système $Ax = b$ devient alors $x = M^{-1}Nx + M^{-1}b = Rx + r$ et la méthode itérative s'écrit

$$x^{(k+1)} = R \cdot x^{(k)} + r \ .$$

Proposition 2.3.1
Soit (M, N) la décomposition associée à A et supposons que pour la norme matricielle subordonnée on a $\|R\| < 1$.
Alors pour tout vecteur $x^{(0)} \in \mathbb{R}^n$, la suite $x^{(k+1)} = R \cdot x^{(k)} + r$ est convergente.

Proposition 2.3.2
La suite $(x^{(k)})$ définie par $x^{(k+1)} = R \cdot x^{(k)} + r$ converge vers la solution de $Ax = b$, pour tout $x^{(0)} \in \mathbb{R}^n$, si et seulement si $\rho(R) < 1$.

On appelle *vitesse de convergence* de la méthode le nombre $-\log_{10}(\rho(R))$.

Les méthodes que nous allons présenter sont basées sur des choix qui permettent d'inverser facilement M et d'obtenir une matrice R pour laquelle $\rho(R)$ est petit.

2.3.1 Méthode de Jacobi

On décompose A en $A = D - K$ où D est la diagonale de A et $-K$ contient les éléments hors diagonale. On note $R_J = D^{-1}K$.

Algorithme 2.4 (Jacobi)
Après permutations éventuelles pour rendre D inversible,
une itération de la méthode de Jacobi s'écrit :

pour $i = 1$ ***to*** n

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij} x_j^{(k)} \right)$$

Proposition 2.3.3
Si la matrice A est à diagonale strictement dominante,

c.-à-d. pour tout $1 \leq i \leq n$: $\displaystyle\sum_{j=1, j\neq i}^{n} |a_{ij}| < |a_{ii}|$, alors la méthode de Jacobi converge.

2.3.2 Méthode de Gauss-Seidel

On écrit $A = (D - L) - U$ où D est la diagonale de A et $-L$, resp. $-U$, la matrice triangulaire strictement inférieure, resp. supérieure, de A. On note $R_{GS} = (D - L)^{-1}U$.

Algorithme 2.5 (Gauss-Seidel)
Après permutations éventuelles, une itération de la méthode de Gauss-Seidel *s'écrit :*

pour $i = 1$ ***to*** n

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right)$$

Proposition 2.3.4
Si la matrice A est à diagonale strictement dominante, alors la méthode de Gauss-Seidel *converge.*
Dans ce cas on a de plus $\|R_{GS}\|_\infty \leq \|R_J\|_\infty < 1$.

2.3.3 Méthode de relaxation

Le décomposition de A s'écrit $A = D - L - U$ et $M = \frac{1}{\omega}D - L$, $N = (\frac{1}{\omega} - 1)D + U$.
On note $R_{rel(\omega)} = (\frac{1}{\omega}D - L)^{-1}((\frac{1}{\omega} - 1)D + U)$.

Algorithme 2.6
Une itération de la méthode de relaxation de paramètre ω s'écrit :

pour $i = 1$ ***to*** n

$$x_i^{(k+1)} = (1 - \omega)x_i^{(k)} + \frac{\omega}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right)$$

Théorème 2.3.1
On a $\rho(R_{rel(\omega)}) > |\omega - 1|$, *une condition nécessaire de convergence de la méthode de relaxation est donc que* $0 < \omega < 2$.

Théorème 2.3.2
Soit A une matrice symétrique définie positive, alors $\rho(R_{rel(\omega)}) < 1$ *pour* $0 < \omega < 2$. *Une condition suffisante de convergence de la méthode de relaxation est alors que* $0 < \omega < 2$.
Pour $\omega = 1$ la méthode de Gauss-Seidel *converge aussi.*

Chapitre 3

Optimisation continue sans contraintes

3.1 Définitions et rappels

3.1.1 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*.

La *valeur optimale* (ou *minimale*) du problème (3.2) est définie par $\overline{f} = \inf_{x \in \mathfrak{C}} f(x)$.

On a $\overline{f} \in \mathbb{R} \cup \{\pm\infty\}$.

On dira que $\overline{x}$ est un *point optimal* (ou *minimal*) si $\overline{x}$ est admissible et $f(\overline{x}) = \overline{f}$.
L'ensemble des points optimaux est $\{x \mid x \in \mathfrak{C}\,, f(x) = \overline{f}\}$.

Un point $\overline{x} \in \mathfrak{C}$ 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 $x \in \mathfrak{C} \cap V$
(resp. $f(\overline{x}) < f(x)$ pour tout $x \in \mathfrak{C} \cap V$, $x \neq \overline{x}$).

Pour $\varepsilon > 0$ donné, une solution à ε-près, ou ε-sous-optimale, de (3.2) est un élément $\overline{x}_\varepsilon \in \mathfrak{C}$ tel que $f(\overline{x}_\varepsilon) \leq \overline{f} + \varepsilon$.
La suite $(x_k) \in \mathfrak{C}$ est une *suite minimisante* si $\lim_{k \to +\infty} f(x_k) = \overline{f}$.

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.

Suivant la forme des fonctions f, g et h on obtient une classification des problèmes d'optimisation :

- *Programmation linéaire.* Dans ce cas la fonction coût est linéaire $f(x) =< c, x >$, avec $c \in \mathbb{R}^n$ donnée et g est affine. On a $g_i(x) =< a_i, x > -b_i$ pour $1 \leq i \leq m$ et $h = 0$. L'ensemble des contraintes est donné par $\mathfrak{C} = \{x \in \mathbb{R}^n \mid < a_i, x > \leq b_i, 1 \leq i \leq m\}$, c'est un polyèdre convexe fermé.

- *Optimisation quadratique.* La fonction-objectif est quadratique $f(x) = \frac{1}{2}\, x^t Q x + q^t\, x$ avec Q symétrique définie positive et les contraintes sont encore affines $\mathfrak{C} = \{x \in \mathbb{R}^n \mid < a_i, x > \leq b_i, 1 \leq i \leq m\}$.

- *Optimisation convexe.* La fonction-objectif f et les contraintes-inégalités g_i sont convexes, les h_i sont affines, de la forme $h_i(x) =< a_i, x > -b_i$ pour $1 \leq i \leq p$. Dans ce cas l'ensemble contrainte $\mathfrak{C}$ est un convexe.

- *Optimisation différentiable* ou *lisse.* Les fonctions f, g_i $(1 \leq i \leq m)$ et h_j $(1 \leq j \leq p)$ sont différentiables (p.ex. de classe $\mathcal{C}^1$, $\mathcal{C}^2$).

- *Optimisation non-différentiable.* C'est le cas si certaines fonctions parmi f, g_i et h_j ne sont pas régulières.

Les hypothèses sur f, g et h vont déterminer de façon essentielle l'existence et l'unicité d'une solution de (3.2). Ce n'est qu'en ayant des conditions nécessaires et suffisantes d'existence de points optimaux que l'on peut se poser la question d'appliquer un algorithme numérique pour déterminer ces solutions.

3.1.2 Ensembles et fonctions convexes

En optimisation on obtient des résultats forts d'existence et unicité de minima si on utilise la convexité, d'où l'importance des définitions et résultats de cette section.

Définition 3.1.1
Un ensemble $C \subset \mathbb{R}^n$ est un ensemble convexe si pour tout $(x, y) \in C^2$ le segment $[x, y] = \{t\, x + (1 - t)\, y\, ;\, 0 \leq t \leq 1\}$ 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. Le *simplexe unité* $S = \{x \in \mathbb{R}^n \,\Big|\, \sum_{i=1}^{n} x_i \leq 1\,,\, x_i \geq 0 \text{ pour } i = 1, \ldots, n\}$ est convexe.
5. Un ensemble C est un *cône* si pour tout $x \in C$ on : $\{t\, x \,|\, t \geq 0\} \subset C$. Un ensemble C est un cône convexe si pour tout $(x, y) \in C^2$, $(s, t) \in (\mathbb{R}_+)^2 \quad s\, x + t\, y \in C$.

L'*hyperoctant positif* $(\mathbb{R}_+)^n$ est un exemple simple de cône convexe.
L'ensemble des matrices carrées symétriques semidéfinies positives est aussi un cône convexe.

Opérations qui conservent la convexité des ensembles.

PROPOSITION 3.1.1
(a) Soit $\{C_i\}_{i\in I}$ une famille quelconque d'ensembles convexes,
alors $C = \bigcap_{i\in I} C_i$ est un ensemble convexe.
(b) Pour $i = 1, \ldots, k$, soient $C_i \subset \mathbb{R}^{n_i}$ des ensembles convexes, alors $C = C_1 \times \cdots \times C_k$ est un ensemble convexe de $\mathbb{R}^{n_1+\cdots+n_k}$.
(c) Soit $C \subset \mathbb{R}^n$ un ensemble convexe et $f : \mathbb{R}^n \to \mathbb{R}^m$ une application affine, alors $f(C) = \{f(x) \,|\, x \in C\}$ est convexe.
Inversement, si $g : \mathbb{R}^m \to \mathbb{R}^n$ est une application affine, alors $g^{-1}(C) = \{x \,|\, g(x) \in C\}$ est convexe.

DÉFINITION 3.1.2
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 $t \in]0, 1[$*

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

– *On dit que f est fortement convexe ou uniformément convexe de module $\nu > 0$ si*

$$f(t\,x + (1-t)\,y) \leq t\,f(x) + (1-t)\,f(y) - \frac{\nu}{2}\,t(1-t)\,\|x-y\|_2^2$$

pour tout $(x, y) \in C^2$ et pour tout $t \in]0, 1[$.

– *Une fonction $f : \mathbf{dom}\, f \to \mathbb{R}$ est quasi-convexe si tout ensemble de niveau inférieur $L_f(\alpha) = \{x \in \mathbf{dom}\, f \mid f(x) \leq \alpha\}$ est un ensemble convexe.*

PROPOSITION 3.1.2
(a) Une fonction f est convexe si et seulement si sa restriction à tout segment inclus dans $\mathbf{dom}\, f$ est une fonction convexe.
(b) Une fonction f est convexe si et seulement si son épigraphe
$\mathbf{epi} f = \{(x, \alpha) \in \mathbb{R}^n \times \mathbb{R} \,|\, f(x) \leq \alpha\}$ est un ensemble convexe.
(c) Une fonction f est fortement convexe de module $\nu > 0$ si et seulement si la fonction $x \longmapsto f(x) - \frac{\nu}{2}\,\|x\|_2^2$ est convexe.
(d) Une fonction f est quasi-convexe si et seulement si pour tout $(x, y) \in \mathbf{dom}\, f$ et pour tout $t \in]0, 1[$

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

On dit que f est strictement quasi-convexe si on a inégalité stricte ci-dessus.

Note : Rappelons qu'une fonction convexe f est localement Lipschitz.

Les propositions suivantes présentent des caractérisations de convexité pour des fonctions régulières.

PROPOSITION 3.1.3 (CRITÈRES DE CONVEXITÉ I)
Soit Ω un ouvert convexe non vide de $\mathbb{R}^n$ et $f \in \mathcal{C}^1(\Omega, \mathbb{R})$:

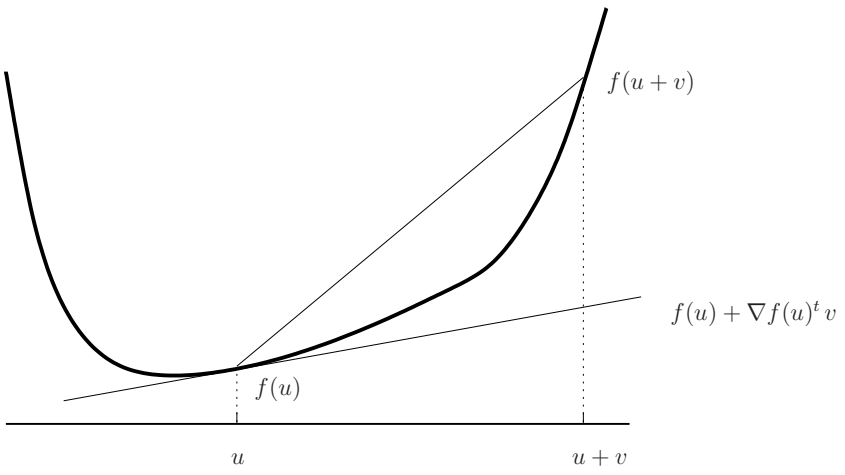
(a) f est convexe sur Ω si et seulement si

$$\forall u \in \Omega, \forall v \in \mathbb{R}^n \text{ tel que } u+v \in \Omega : D_v f(u) = \nabla f(u)^t\, v \leq f(u+v) - f(u) \; ;$$

(b) f est strictement convexe sur Ω si et seulement si

$$\forall u \in \Omega, \forall v \in \mathbb{R}^n \setminus \{0\} \text{ tel que } u+v \in \Omega : D_v f(u) = \nabla f(u)^t\, v < f(u+v) - f(u) \; .$$

Ces inégalités expriment que le graphe de f est en chaque point au-dessus du plan tangent. Pour $n = 1$, la fonction f est convexe, resp. strictement convexe, si et seulement si f' est croissante, resp. strictement croissante.



PROPOSITION 3.1.4 (CRITÈRES DE CONVEXITÉ II)
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.

(c) f est fortement convexe, de module $\nu > 0$, si et seulement si

$$\text{pour tout } x \in C \text{ et pour tout } w \in \mathbb{R}^n \; : \quad w^t \nabla^2 f(x) w \geq \nu \, \|w\|_2^2 \; .$$

i.e. *ν minore les valeurs propres de $\nabla^2 f(x)$ pour tout $x \in C$.*

Grâce à ces caractérisations on obtient les exemples suivants.

Exemples :

1. La *fonction indicatrice* I_C d'un ensemble convexe C est convexe :
$$I_C(x) = \begin{cases} 0 & x \in C \\ +\infty & x \notin C \,. \end{cases}$$
2. Les fonctions $x \mapsto e^{ax}$, $a \in \mathbb{R}$ et $x \mapsto |x|^p$, $p \geq 1$ sont convexes sur $\mathbb{R}$.
3. 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 $\mathbb{R}_+^*$.
4. Sur $\mathbb{R}^n$ toute norme, $x \mapsto \|x\|$, et la fonction $x \mapsto \max\{x_1, \ldots, x_n\}$ sont des fonctions convexes.
5. La moyenne géométrique $f(x) = \left(\prod_{i=1}^{n} x_i\right)^{1/n}$ est concave sur $\mathbf{dom}\, f = (\mathbb{R}_+^*)^n$.
6. La fonction $f : x \mapsto 1 - e^{-\|x\|_2^2}$ est une fonction strictement quasi-convexe sur $\mathbb{R}^n$ mais f n'est pas convexe.

Des opérations qui conservent la convexité sont données par la

Proposition 3.1.5

(a) Soient $f_1, \ldots, f_m$ des fonctions convexes, $\alpha_1, \ldots, \alpha_m$ des réels positifs, alors la fonction $f = \sum_{i=1}^{m} \alpha_i f_i$ est convexe sur $\mathbf{dom}\, f = \bigcap_{i=1}^{m} \mathbf{dom}\, f_i \neq \emptyset$.

(b) Soit $\{f_y\}_{y \in I}$ une famille quelconque de fonctions convexes, alors la fonction $f = \sup_{y \in I} f_y$ est convexe sur $\mathbf{dom}\, f = \{x \mid \sup_{y \in I} f_y(x) < +\infty\}$.

Exemple : Soit $f : \mathbb{R}^n \to \mathbb{R}$, la *fonction conjuguée* de f est définie par

$$f^*(x) = \sup_{y \in \mathbf{dom}\, f} (< x, y > - f(y)) \,,$$

où $\mathbf{dom}\, f^*$ est l'ensemble des x pour lesquels $< x, y > - f(y)$ est borné sur $\mathbf{dom}\, f$.
Comme $x \mapsto < x, y > - f(y)$ est une fonction convexe (affine) en x, la fonctionf^* est convexe.

- Soit $f(y) = e^y$, $y \in \mathbb{R}$, alors la fonction $y \mapsto xy - e^y$ n'est pas bornée pour $x < 0$; pour $x > 0$ on a un maximum en $y = \ln x$, donc $f^*(x) = x \ln x - x$; pour $x = 0$ on trouve $f^*(0) = 0$. Finalement $\mathbf{dom}\, f^* = \mathbb{R}_+$ et $f^*(x) = x \ln x - x$.
- Soit $f(y) = y \ln y$ définie sur $\mathbb{R}_+$, on montre que $\mathbf{dom}\, f^* = \mathbb{R}$ et $f^*(x) = e^{x-1}$.
- Si $f = I_S$ la fonction indicatrice d'un ensemble $S \subset \mathbb{R}^n$, alors $f^*(x) = I_S^*(x) = \sup_{y \in S} < x, y >$.

Les propriétés suivantes sont souvent utilisés pour obtenir des résultats d'existence de minima

Définition 3.1.3

- *On dit que la fonction $f : \mathbb{R}^n \to \mathbb{R}$ est <u>coercive</u> si $\lim_{\|x\| \to +\infty} f(x) = +\infty$.*

– *Soit C un ouvert convexe de $\mathbb{R}^n$, on dit que la fonction $F : C \to \mathbb{R}$ est monotone, respectivement fortement monotone de module $\nu > 0$, si pour tout $(x, y) \in C^2$ $< F(x) - F(y)\,,\, x - y >\geq 0$, respectivement si pour tout $(x, y) \in C^2$ $< F(x) - F(y)\,,\, x - y >\geq \nu\|x - y\|_2^2$.*

– *Soit C un ouvert convexe de $\mathbb{R}^n$, la fonction $f : C \to \mathbb{R}$ est elliptique si elle est de classe $\mathcal{C}^1$ sur C et si ∇f est fortement monotone de module $\nu > 0$ sur C.*

Proposition 3.1.6
Soit f une fonction de classe $\mathcal{C}^1$ sur l'ouvert convexe C, alors f est elliptique de module $\nu > 0$ si et seulement si f est fortement convexe de module ν.
Dans ce cas f est strictement convexe et coercive, en particulier les ensembles $L_f(\alpha) = \{x \in C \mid f(x) \leq \alpha\}$ sont bornés.

3.1.3 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 $\inf_{\mathcal{D}} f$.

Théorème 3.1.1
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.
Alors il existe au moins un $\overline{x} \in U$ avec $f(\overline{x}) = \inf_{x \in U} f(x)$.

Théorème 3.1.2
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 $\nabla^2 f(\overline{x})$ est semidéfinie positive.*
3. *Soit $f \in \mathcal{C}^2(\mathcal{D})$, si $\nabla f(\overline{x}) = 0$ et $\nabla^2 f(\overline{x})$ est définie positive, alors $\overline{x}$ est un minimum local strict de f.*

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

Théorème 3.1.3
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 3.1.4
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 3.1.5
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

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

THÉORÈME 3.1.6
Soit $f \in \mathcal{C}^2(\mathbb{R}^n, \mathbb{R})$, on suppose qu'il existe $\nu > 0$ tel que

$$\forall x \in \mathbb{R}^n, \forall u \in \mathbb{R}^n \; : \quad u^t \, H_f(x) \, u \geq \nu \, \|u\|_2^2 \, .$$

Alors f admet un minimum global unique sur $\mathbb{R}^n$

THÉORÈME 3.1.7
Soit $\mathcal{D}$ un ouvert convexe non vide de $\mathbb{R}^n$ et $f \in \mathcal{C}^2(\mathcal{D})$, si :
(i) soit il existe $\alpha \in \mathbb{R}$ tel que l'ensemble de niveau inférieur $L_f(\alpha) = \{x \in \mathcal{D} \mid f(x) \leq \alpha\}$ est fermé, soit $\mathcal{D} = \mathbb{R}^n$;
(ii) il existe $\nu > 0$ tel que $w^t \nabla^2 f(x) w \geq \nu \, \|w\|_2^2$ pour tout $x \in \mathcal{D}$ et tout $w \in \mathbb{R}^n$,
alors f admet un unique minimum global (strict) sur $\mathcal{D}$.

Exemple : On veut minimiser sur $\mathbb{R}^n$ la fonction quadratique $f(x) = x^t P x + q^t \, x + r$, où P est symétrique semidéfinie positive, $q \in \mathbb{R}^n$ et $r \in \mathbb{R}$. C'est un problème d'optimisation quadratique sans contrainte. L'hypothèse sur P entraîne que f est convexe et la CNS d'optimalité pour $\overline{x} \in \mathbb{R}^n$ s'écrit

$$\nabla f(\overline{x}) = 2P\overline{x} + q = 0 \, .$$

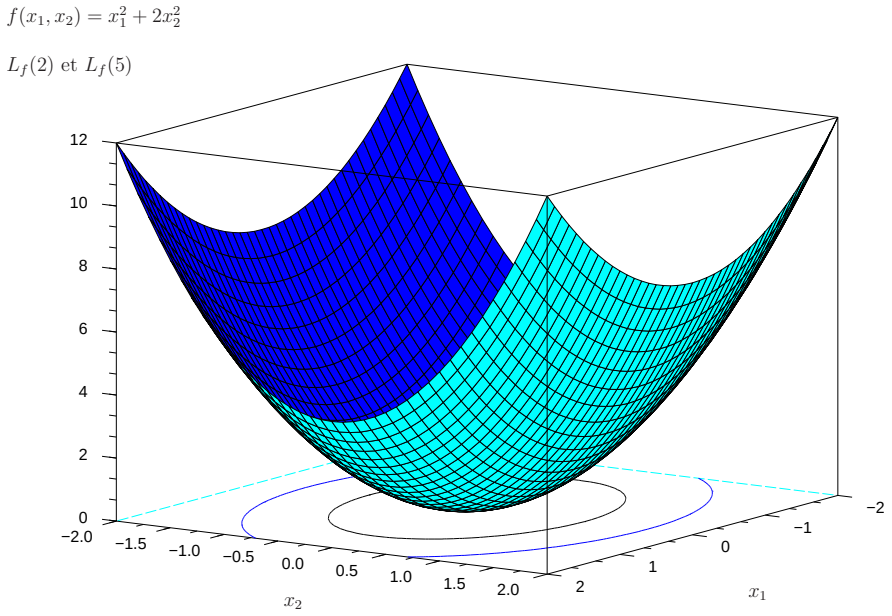
Si P est inversible, *i.e.* défini positive, f est strictement convexe et on a une solution unique $\overline{x} = -P^{-1}q$. Si $q \notin \mathbf{Im}(P)$, il n'y a pas de solution, f n'admet pas de borne inférieure ; si $q \in \mathbf{Im}(P)$, mais P non inversible, l'ensemble des points optimaux est un sous-espace affine de $\mathbb{R}^n$: $\overline{x} \in \mathbf{ker}(P) - P^+ q$, où P^+ est la matrice pseudo-inverse de MOORE-PENROSE .

THÉORÈME 3.1.8
Soit $\mathcal{D}$ un ouvert convexe non vide de $\mathbb{R}^n$ et $f : \mathcal{D} \to \mathbb{R}$ continue et quasi-convexe, alors tout minimum local strict est un minimum global.
S'il existe $\alpha \in \mathbb{R}$ tel que l'ensemble de niveau inférieur $L_f(\alpha) = \{x \in \mathcal{D} \mid f(x) \leq \alpha\}$ est compact, alors il existe $\overline{x} \in L_f(\alpha)$ minimum local de f .
Si f est de plus strictement quasi-convexe, alors il existe un unique minimum $\overline{x}$.

3.1.4 Exemple : l'algorithme de Hager

Soit A une matrice carrée réelle inversible de taille $n \in \mathbb{N}^*$ et dont on suppose connaître la décomposition LU d'un coût $O(2/3n^3)$, on s'intéresse au conditionnement $c(A) = \|A\|\|A^{-1}\|$ afin d'évaluer l'erreur relative commise en résolvant le système $Ax = b$ (voir théorème 2.1.3).

Dans toute la suite on va noter $B = A^{-1} = (b_1 \cdots b_n)$ et $I_n = (e_1 \cdots e_n)$, où b_j et e_j (base canonique) sont des vecteurs colonnes de $\mathbb{R}^n$.



Pour calculer $y = Bx$, on est obligé de résoudre le système $Ax = y$, or en utilisant la décomposition LU, ceci se fait en résolvant deux systèmes linéaires, c.-à-d. avec un coût en $O(n^2)$.

Comme $AA^{-1} = I_n$ on a, pour $1 \leq j \leq n$, $A\,b_j = e_j$, *i.e.* $b_j = U^{-1}(L^{-1}e_j)$, ainsi, pour obtenir les n colonnes de A^{-1}, on a besoin de $O(n^3)$ opérations. S'il faut résoudre le système linéaire une seule fois, déterminer le conditionnement du problème est donc aussi coûteux que résoudre le système linéaire.

En pratique, on préfère utiliser un estimateur de $\|A^{-1}\|$. Parmi les différentes méthodes existantes, on va présenter l'algorithme de Hager qui donne une borne inférieure de la norme $\|.\|_1$ d'une matrice. On applique l'algorithme à la matrice A^{-1} pour estimer $\|A^{-1}\|_1$. Noter qu'il suffit d'appliquer l'algorithme à la matrice $(A^{-1})^t$ pour estimer $\|A^{-1}\|_\infty$.

La méthode de Hager est basé sur une maximisation de la fonction définie,
pour tout $x \in \mathbb{R}^n$, par

$$f(x) = \|Bx\|_1 = \sum_{i=1}^{n} \left| \sum_{j=1}^{n} b_{ij} x_j \right| .$$

Comme $\|B\|_1 = \max\limits_{1\leq j\leq n} \sum\limits_{i=1}^{n} |b_{ij}|$, il existe j^* tel que $\|B\|_1 = \sum\limits_{i=1}^{n} |b_{ij^*}|$ et $f(e_{j^*}) = \|B\|_1$.
On cherchera le maximum de f sur les vecteurs de la base canonique, il suffit donc de tester tous les vecteurs e_j $(1 \leq j \leq n)$ pour déterminer le maximum exact, or pour cela il faut évaluer $y = Be_j$, *i.e.* résoudre deux systèmes triangulaires. Cette recherche exhaustive est en moyenne d'ordre $(n/2)\,O(n^2) = O(n^3)$, donc trop coûteuse.
En général l'algorithme proposé nécessite seulement quelques itérations pour avoir un résultat exploitable.

La fonction à maximiser f est convexe sur $\mathbb{R}^n$, mais pas différentiable sur
$\{y \in \mathbb{R}^n \,/\, \exists i \in \{1, \ldots, n\}, y_i = 0\} = \{\text{réunion des } n \text{ hyperplans définis par } y_i = 0\}$.

Soit x tel que les coordonnées de $y = Bx$, vérifient $\min_{1\leq i\leq n} |y_i| > 0$, par continuité il existe alors un voisinage $\mathcal{V}_x$ de x tel que pour tout $z \in \mathcal{V}_x$, on a $\min_{1\leq i\leq n} |(Bz)_i| > 0$.

On note $s_i = signe(y_i) = signe((Bz)_i)$ et alors, comme $|(Bz)_i| = signe((Bz)_i)\,(Bz)_i$, on a $f(z) = \sum_{i=1}^{n} |(Bz)_i| = \sum_{i=1}^{n} s_i\,(Bz)_i = \sum_{i=1}^{n}\sum_{j=1}^{n} s_i b_{ij} z_j$, f est donc une fonction linéaire en $z \in \mathcal{V}_x$,

On en déduit que dans un voisinage de x la fonction f est différentiable et

$$\frac{\partial f}{\partial x_k}(x) = \sum_{i=1}^{n} s_i b_{ik}\,,\ \text{d'où}\ \nabla f(x) = B^t s\,.$$

Soit $x \in \mathbb{R}^n$ tel que $\|x\|_1 = 1$ et $\min_{1\leq i\leq n} |y_i| > 0$, où $y = Bx$ et on note $g = \nabla f(x) = B^t s$.
Comme f est linéaire sur $\mathcal{V}_x$, on a $f(z) = f(x) + < \nabla f(x), z - x >$, pour tout $z \in \mathcal{V}_x$.
Alors si $\|g\|_\infty \leq g^t x$ et si $\|z\|_1 \leq 1$:

$$f(z) \leq f(x) + \sum_i |g_i||z_i| - g^t x \leq f(x) + \|g\|_\infty \|z\|_1 - g^t x \leq f(x)\,.$$

Supposons maintenant $\|g\|_\infty > g^t x$, il existe $j \in \{1, \ldots, n\}$ tel que $|g_j| = \|g\|_\infty$,
on pose $\tilde{x} = signe(g_j) e_j$.
Comme f est convexe sur $\mathbb{R}^n$: $f(\tilde{x}) \geq f(x) + < \nabla f(x), \tilde{x} - x >$, d'où

$$f(\tilde{x}) \geq f(x) + g^t \tilde{x} - g^t x = f(x) + \|g\|_\infty - g^t x > f(x)\,.$$

L'algorithme de Hager s'écrit comme suit

Algorithme 3.1 (Hager)

Choisir $x^{(0)}$ *tel que* $\left\|x^{(0)}\right\|_1 = 1$
$k = 0$
boucle
  $y = Bx^{(k)}$
  $s = \text{signe}(y)$
  $g = B^t\, s$
  si $\|g\|_\infty \leq g^t\, x^{(k)}$ ***alors***
    retourner $\|y\|_1$
  sinon
    $k = k + 1$
    $x^{(k)} = e_j$ *pour* j *tel que* $|g_j| = \|g\|_\infty$

D'après ce qui précède, si $\|y\|_1$ est retourné par l'algorithme, alors c'est un maximum local de f, sinon, le nouveau choix de x, en début de boucle, vérifie $f(x^{k+1}) > f(x^k)$.

L'algorithme de Hager est un exemple d'une méthode qui «remonte dans la direction du gradient» et, si l'on fait peu d'itérations, on arrive rapidement à une estimation de $\|B\|_1$.

Remarquer qu'à chaque itération on doit calculer $y = Bx = A^{-1}x$ et $g = B^t s = (A^t)^{-1}s$. Pour cela il suffit de résoudre quatre systèmes triangulaires, coût $O(n^2)$, avec peu d'itérations on reste $O(n^2)$, ce qui est négligeable par rapport au coût de la décomposition LU.

Sur la page suivante on propose un programme SCILAB [1] qui compare notre implémentation aux fonctions `cond` et `rcond` incluses dans SCILAB.

1. © INRIA, logiciel disponible à `http://www.scilab.org/`

```
1  // Calcul de y=Bx en résolvant Ay=x, on <<simule>> ici la résolution
2  // des deux systèmes triangulaires.
3  // Pour de grandes matrices creuses utiliser ‘‘lufact’’ et ‘‘lusolve’’
4  function [y]= systemes_triangulaires(x,L,U,E)
5  y=U\(L\(E*x));
6  endfunction
7
8  // Méthode de HAGER pour déterminer ||B||_1, où B=A^{-1}
9  function [x_h,v_h]=hager(x0,A)
10
11 x_h=[ ]; // historique des points
12 v_h=[ ]; // historique des valeurs v
13
14 x=x0;           // valeur initiale de x
15 n=size(A,1);    // nb. de lignes de A
16 [L,U,E]=lu(A);  // decomposition LU d'une matrice pleine
17
18 for k=0:n
19   y=systemes_triangulaires(x,L,U,E);         // y=Bx
20   v=norm(y,1);
21   if min(abs(y))==0.0 then    // pour rester différentiable
22     mprintf('\n Probleme : min(|y_i|)=0 \n')
23     return
24   end
25   s=sign(y);
26   g=E'*systemes_triangulaires(E'*s,U',L',E); // g=B's
27   ng=norm(g,'inf');
28   x_h=[x_h x];
29   v_h=[v_h v];
30   if ( ng <= g'*x ) then
31     mprintf('\n  %d iterations \n',k+1);
32     return;
33   else
34     jj=find(abs(g)==ng);
35     x=zeros(n,1);
36     x(jj(1,1),1)=1;
37   end
38 end
39 mprintf('\n Probleme : %d iterations \n',n+1)
40 endfunction
41
42 // Comparer à la <<vraie>> valeur et à  rcond() de Scilab
43 n=50; A=rand(n,n);
44
45 c_1=norm(A,1)*norm(inv(A),1),       c_1_scilab=1/rcond(A)
46
47 x0=1/n*ones(n,1);   [x_h,v_h]=hager(x0,A);
48 c_1_hager=norm(A,1)*v_h(\$)
```

3.2 Algorithmes de minimisation sans contrainte

3.2.1 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

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

Si l'ensemble $L_f(f(x^{(0)}))$ 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 3.2 (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.
Dans ce cas il existe $\tilde{\sigma} > 0$ tel que pour tout $\sigma \in]0, \tilde{\sigma}[\,:\ f(x + \sigma d) < f(x)$.
On définit la *direction normalisée de plus grande descente* de f en x comme étant la solution de
$$\min\{< \nabla f(x), d > \ \mid\ \|d\| = 1\}\ .$$
Ce minimum existe, n'est pas nécessairement unique et dépend du choix de $\|.\|$.
Pour les algorithmes de descente il suffit en général d'avoir une *direction de plus grande descente*, *i.e.* $d \in \mathbb{R}^n \setminus \{0\}$ telle que $d\,/\,\|d\|$ soit une direction normalisée de plus grande descente.

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 $x^{(\mathrm{k})}$, ne pas mettre trop de temps à trouver σ, etc.
Si la fonction coût le permet, ou pour démontrer des résultats de convergence et vitesse de convergence, on peut supposer trouver un minimum exact de f sur $\{x + \sigma d \,|\, \sigma > 0\}$.

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 $x^{(\mathrm{k})}$.

Si $\lim\limits_{k\to+\infty} \dfrac{\left\|x^{(\mathrm{k}+1)} - \overline{x}\right\|}{\left\|x^{(\mathrm{k})} - \overline{x}\right\|} = 0$ on dit que l'on a *convergence superlinéaire*.

3.2.2 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. Parmi les nombreuses possibilités proposées dans la littérature nous allons présenter les *conditions de* Wolfe (faibles) sur $\sigma > 0$:

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

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

avec $0 < c_1 < c_2 < 1$.

Pour expliquer ces conditions posons $\phi(\sigma) = f(x^{(\mathrm{k})} + \sigma d^{(\mathrm{k})})$ et $l(\sigma) = \phi(0) + (c_2\phi'(0))\sigma$. On remarque que $\phi'(0) < 0$ car $d^{(\mathrm{k})}$ est une direction de descente.

La condition (W1) impose que la réduction de f est proportionnelle à σ et $\phi'(0)$. On 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.

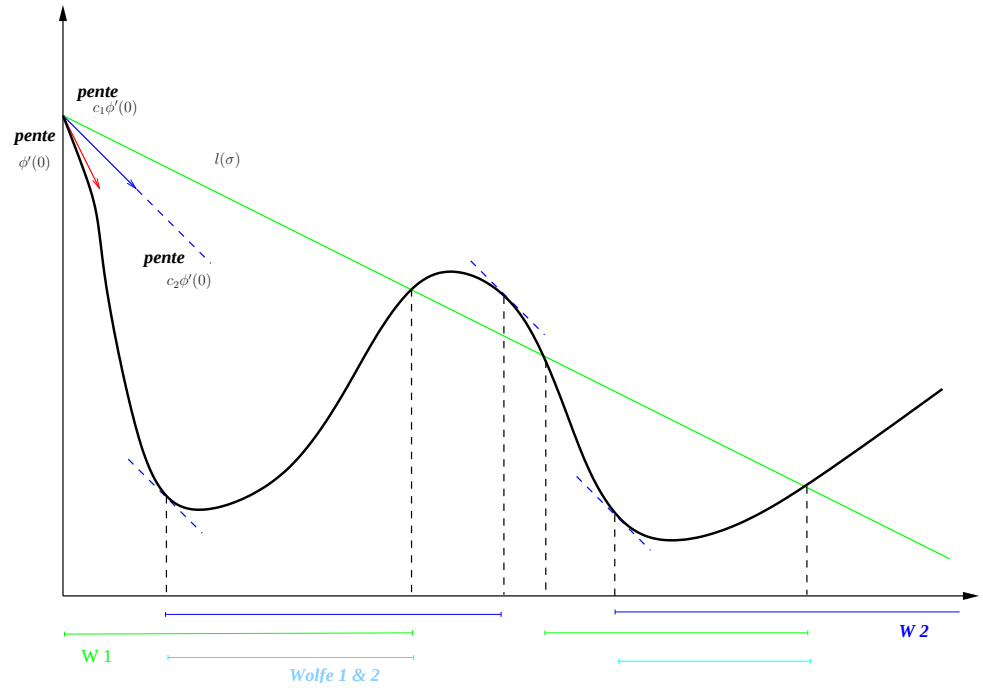
La condition (W2) implique que $\phi'(\sigma) \geq c_2\phi'(0) \geq \phi'(0)$: si $\phi'(\sigma)$ est «très» négatif (*i.e.* $< c_2\phi'(0)$), on va chercher plus loin, sinon on peut s'arrêter. De de cette façon le pas σ ne sera pas trop petit.

On obtient des contraintes plus fortes si l'on remplace (W2) par

$$\left|< \nabla f(x^{(\mathrm{k})} + \sigma d^{(\mathrm{k})})\,,\, d^{(\mathrm{k})} >\right| \leq c_2\ \left|< \nabla f(x^{(\mathrm{k})})\,,\, d^{(\mathrm{k})} >\right| \qquad \text{(W3)}$$

Les (W1) et (W3) sont les *conditions de* Wolfe *fortes*. La contrainte (W3) entraîne que $c_2\phi'(0) \leq \phi'(\sigma) \leq -c_2\phi'(0)$ c.-à-d. $\phi'(\sigma)$ n'est pas «trop» positif.

L'existence de valeurs de σ vérifiant (W1-2) et (W1-3) est donnée par la



PROPOSITION 3.2.1
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 *faibles et fortes.*

THÉORÈME 3.2.1 (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*, alors*

$$\sum_{k=0}^{+\infty} \cos^2\theta_k \, \left\|\nabla f(x^{(k)})\right\|_2^2 < +\infty \,,$$

où $\cos\theta_k = -\dfrac{\nabla f(x^{(k)}).d^{(k)}}{\left\|\nabla f(x^{(k)})\right\|_2 \, \left\|d^{(k)}\right\|_2}$.

COROLLAIRE 3.2.1
Si f et la méthode de descente $(x^{(k)})_k$ vérifient les hypothèses du théorème 3.2.1 et si de plus il existe $\delta > 0$ tel que pour k suffisamment grand on a $\cos\theta_k \geq \delta$, alors $\lim_k \left\|\nabla f(x^{(k)})\right\|_2 = 0$.

Ce corollaire est un résultat de *convergence globale* : pour des fonctions coût vérifiant les hypothèses du théorème 3.2.1 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 $x^{(0)}$ qui n'est pas nécessairement proche de $\overline{x}$.

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.

Algorithme 3.3 (Algorithme de backtracking)

Choisir σ_{init} , ρ, $c \in]0,1[$

$\sigma = \sigma_{init}$

tant que $f(x^{(k)} + \sigma\, d^{(k)}) > f(x^{(k)}) + c\,\sigma\, \nabla f(x^{(k)})^t\, d^{(k)}$

$\sigma = \rho\sigma$

$\sigma_k = \sigma$

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 $x^{(\mathrm{k})}$.

3.2.3 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 3.4 (Descente de gradient)

choisir $x^{(0)} \in \mathbf{dom}\, f$ *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 corollaire 3.2.1 la méthode de descente du gradient est globalement convergente. Pour étudier la vitesse de convergence on va étudier un cas particulier.

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

où f est une fonction fortement convexe 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 $\nabla^2 f(x) = Q$. La fonction f admet un minimum global unique $\overline{x}$ qui est solution de $Qx = b$.

Dans ce cas on peut déterminer le pas de descente optimal σ_k et l'itération s'écrit :

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

Posons $\|x\|_Q^2 = x^t Q x$, on montre alors que

$$\left\|x^{(\mathrm{k}+1)} - \overline{x}\right\|_Q^2 = \left(1 - \frac{\left\|\nabla f(x^{(\mathrm{k})})\right\|_2^4}{(\nabla f(x^{(\mathrm{k})})^t\, Q\, \nabla f(x^{(\mathrm{k})}))(\nabla f(x^{(\mathrm{k})})^t\, Q^{-1}\, \nabla f(x^{(\mathrm{k})}))}\right) \left\|x^{(\mathrm{k})} - \overline{x}\right\|_Q^2\,,$$

et, en utilisant l'inégalité de KANTOROVICH :

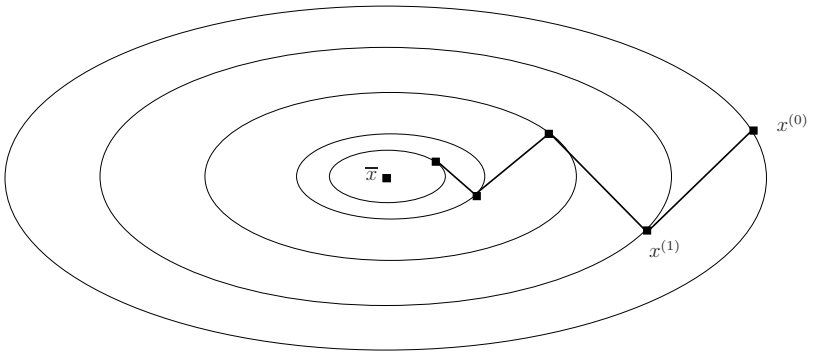
$$\left\|x^{(\mathrm{k}+1)}-\overline{x}\right\|_Q^2 \leq \left(\frac{c_2(Q)-1}{c_2(Q)+1}\right)^2 \left\|x^{(\mathrm{k})}-\overline{x}\right\|_Q^2 ,$$

où $c_2(Q)=\lambda_n/\lambda_1$.
Ainsi, dans le cas d'une fonction coût quadratique, la méthode du gradient est d'ordre un et la vitesse de convergence dépend de $c_2(Q)$: si $c_2(Q)=1$, c.-à-d. $Q=\lambda I_n$ on a convergence en une itération ; si Q est mal conditionnée la convergence devient très lente.

Le conditionnement de Q indique la déformation de l'ellipsoïde $x^tQx=\alpha>0$:
pour $c_2(Q)=1$ on trouve des sphères, sinon les lignes de niveau sont des ellipsoïdes allongés le plus dans la direction associé à λ_n.
Comme σ_k est le minimum exact de $\phi(\sigma)=f(x^{(\mathrm{k})}-\sigma\nabla f(x^{(\mathrm{k})}))$ on a :

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

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



3.2.4 Méthode de la plus forte descente

Pour une norme $\|.\|$ donnée on choisit

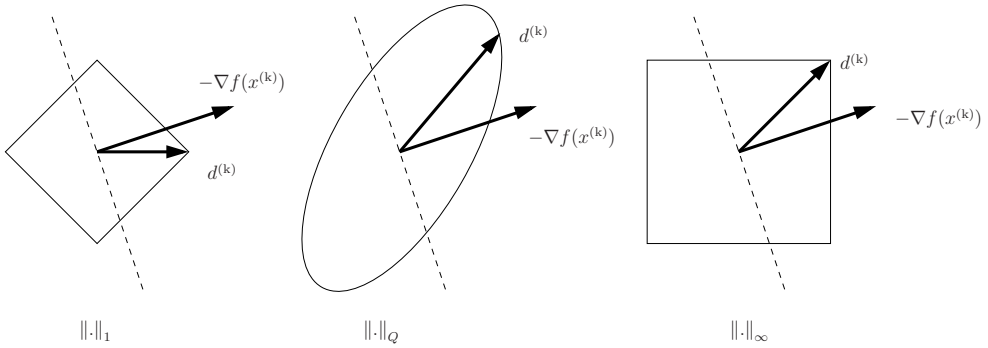
$$d^{(\mathrm{k})}=\arg\min\{\nabla f(x^{(\mathrm{k})})^t\, d \mid \|d\|=1\}\,,$$

c.-à-d. que l'on cherche une direction dans la sphère unité de $\|.\|$ qui s'étend le plus dans le demi-plan de $-\nabla f(x^{(\mathrm{k})})$:

– si $\|.\|=\|.\|_2$, on obtient la méthode de descente du gradient ;

– si $\|.\|=\|.\|_1$, on a $d^{(\mathrm{k})}=-signe\left(\left(\nabla f(x^{(\mathrm{k})})\right)_i\right)\, e_i$,
où $i\in\{1,\ldots,n\}$ est tel que $\left\|\nabla f(x^{(\mathrm{k})})\right\|_\infty=\left|\left(\nabla f(x^{(\mathrm{k})})\right)_i\right|$.

Sur la figure suivante on a représenté en dimension deux la sphère unité pour les normes $\|.\|_1$, $\|.\|_Q$ (Q symétrique définie positive) et $\|.\|_\infty$.



3.2.5 Méthode de relaxation ou des directions alternées

Dans cette méthode on prend successivement les vecteurs de la base de $\mathbb{R}^n$ comme direction de recherche : à chaque itération on minimise par rapport à une seule variable, *i.e.* à l'étape k, on minimise la fonction $\sigma \longmapsto f(x^{(k)} + \sigma\, e_k)$, pour $\sigma \in \mathbb{R}$.

Cette méthode est voisine de celle de la plus forte descente en $\|.\|_1$, mais $\pm e_k$ n'est pas nécessairement une direction de descente en $x^{(k)}$. Cet algorithme est très simple mais on ne peut pas garantir la convergence vers un point stationnaire et même si on on a convergence celle-ci peut être très lente.

Note : Si $f(x) = \frac{1}{2}\, x^t A x - b^t x$, avec A définie positive, alors cette méthode correspond à la méthode itérative de Gauss-Seidel pour résoudre $Ax = b$.

En effet, à l'itération $k \,(\mathrm{mod}\, n)$ on ne change que la k^{e} coordonnée de $x^{(k)}$, il faut n itérations avant d'avoir mis à jour toutes les coordonnées de $x^{(k)}$.

3.2.6 Méthode de Newton

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

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

Si $\nabla^2 f(x^{(k)})$ est définie positive, la direction $d^{(k)}$ sera le minimum de $\tilde{f}_k$:

$$d^{(k)} = -[\nabla^2 f(x^{(k)})]^{-1}\, \nabla f(x^{(k)})\ .$$

Dans ce cas $\nabla f(x^{(k)})^t\, d^{(k)} = -d^{(k)^t} \nabla^2 f(x^{(k)})\, d^{(k)} \leq 0$, on obtient bien une direction de descente.

Théorème 3.2.2
Soit f de classe $\mathcal{C}^2$, $\overline{x} \in \mathbb{R}^n$ tel que $\nabla f(\overline{x}) = 0$ et $\nabla^2 f(\overline{x})$ définie positive et $\nabla^2 f$ une fonction Lipschitz au voisinage de $\overline{x}$.
On considère la suite $(x^{(k)})$ définie par $x^{(0)}$ et

$$x^{(k+1)} = x^{(k)} - [\nabla^2 f(x^{(k)})]^{-1}\, \nabla f(x^{(k)})\ .$$

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 $\nabla^2 f(x^{(\mathrm{k})})\, d = \nabla f(x^{(\mathrm{k})})$, en utilisant l'algorithme de Cholesky cela fait $O(n^2)$ opérations.
3. Les *methodes quasi*-Newton remplacent la matrice hessienne par une approximation B_k qui vérifie la relation de la sécante

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

3.2.7 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 $f(x) = \dfrac{1}{2}\, x^t A x - b^t x$, 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 3.2.1
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 3.2.1
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

$$\sigma = -\frac{r(x)^t\, d}{d^t A\, d}\,. \tag{3.3}$$

Théorème 3.2.3
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 $\quad r(x^{(k)})^t\, d_i = 0 \quad$ *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

$$\beta_k = \frac{r(x^{(\mathrm{k})})^t A \, d^{(\mathrm{k\text{-}1})}}{d^{(\mathrm{k\text{-}1})^t} A \, d^{(\mathrm{k\text{-}1})}} \, . \tag{3.4}$$

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 3.5 (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 3.2.4
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* (3.3), *resp.* (3.4). *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 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 $x^{(\mathrm{k+1})} = x^{(0)} + P_k^*(A) r^{(0)}$ 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 :

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

- 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 $x^{(\mathrm{r})} = \overline{x}$: 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 $Q(X) = C \prod_{i=n-m+1}^{n} (X - \lambda_i) \left(X - \dfrac{\lambda_1 + \lambda_{n-m}}{2} \right)$,
  où la constante C telle que $Q(X) = 1 + X P_m(X)$, avec $P_m \in \mathbb{R}_m[X]$. On obtient alors

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

  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

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

  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 $x_{nouv} = C x_{anc}$ avec C telle que le conditionnement de la matrice $C^{-t} A C^{-1}$ 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 3.6 (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.

3.3 Méthode des moindres carrées

On généralise le problème linéaire de la section 2.2 en introduisant la fonction coût suivante :

$$f(x) = \frac{1}{2}\sum_{k=1}^{m} r_k(x)^2 = \frac{1}{2}\|r(x)\|_2^2 \tag{3.5}$$

où f est définie pour tout $x \in \mathbb{R}^n$ et le *vecteur résidu* $r(x) = (r_1(x) \cdots r_m(x))^t$ est dans $\mathcal{C}^2(\mathbb{R}^n, \mathbb{R}^m)$, on suppose de plus que $m \geq n$.

On note $J(x) = \begin{pmatrix} \frac{\partial r_1}{\partial x_1}(x) & \cdots & \frac{\partial r_1}{\partial x_n}(x) \\ \vdots & & \vdots \\ \frac{\partial r_m}{\partial x_1}(x) & \cdots & \frac{\partial r_m}{\partial x_n}(x) \end{pmatrix} \in \mathcal{M}(m,n)$ et l'on montre que

$$\nabla f(x) = \sum_{k=1}^{m} r_k(x)\nabla r_k(x) = J(x)^t\, r(x) \tag{3.6}$$

$$\nabla^2 f(x) = J(x)^t\, J(x) + \sum_{k=1}^{m} r_k(x)\nabla^2 r_k(x) \tag{3.7}$$

L'équation (3.6) montre que le gradient de la fonction coût définie en (3.5) est fonction de la matrice jacobienne de r et l'équation (3.7) que la matrice hessienne de f est la somme d'un terme d'ordre un, $J(x)^t\, J(x)$, et d'un terme d'ordre deux qui dépend des matrices hessiennes des r_k.
Noter que si tous les r_k sont linéaires $J(x)$ est indépendante de x, donc $\nabla f(x) = J^t r(x)$ et $\nabla^2 f(x) = J^t\, J$. On a alors, pour tout $d \in \mathbb{R}^d$

$$f(x+d) = f(x) + \nabla f(x)^t\, d + \tfrac{1}{2} d^t \nabla^2 f(x) d = f(x) + r(x)^t J d + \tfrac{1}{2} d^t J J^t d\,.$$

Application

Avant de proposer des méthodes de minimisation de 3.5 on va présenter un exemple classique où ce problème apparaît.
Dans un dispositif expérimental on récupère aux instants t_j les données y_j $(1 \leq j \leq m)$, or le phénomène étudié est modélisé grâce à une fonction ϕ qui dépend du temps t et de paramètres $x \in \mathbb{R}^n$.
On note $\epsilon_j = y_j - \phi(x, t_j)$, la différence entre le modèle et les observations et l'on suppose que ces erreurs sont indépendantes et identiquement distribués (i.i.d.) de loi $\mathcal{N}(0, \sigma^2)$ et de densité g_σ, alors la vraisemblance des observations y_j, $j = 1,\ldots,m$, est donnée par

$$L(y; x, \sigma) = \prod_{j=1}^{m} g_\sigma(\epsilon_j) = \frac{1}{(2\pi\sigma^2)^{-\frac{m}{2}}} \exp\left(-\frac{1}{2}\sum_{j=1}^{m} \frac{(y_j - \phi(x, t_j))^2}{\sigma^2}\right).$$

Pour obtenir le *maximum de vraisemblance*, à σ fixé, il faut déterminer

$$\min_x \frac{1}{2}\sum_{j=1}^{m} (y_j - \phi(x, t_j))^2\,,$$

c.-à-d. résoudre le problème des moindres carrées avec $r_j = y_j - \phi(x, t_j)$.

3.3.1 Méthode de Gauss-Newton

Cette méthode est une modification de la méthode de Newton qui utilise comme direction de descente la solution de l'équation $\nabla^2 f(x)d = -\nabla f(x)$;
Pour la méthode de Gauss-Newton on va déterminer la direction de descente grâce à

$$J(x)^t J(x)\, d = -J(x)^t\, r(x) \tag{3.8}$$

La matrice $J(x)^t\, J(x)$ est une bonne approximation de $\nabla^2 f(x)$ si, dans (3.7), les termes d'ordre deux sont dominés par les termes d'ordre un.
De plus, $J(x)^t\, J(x)$ est inversible dès que le rang de $J(x)$ est n . En effet, $J(x)^t\, J(x)$ a alors n valeurs propres $\sigma_1^2(x) \geq \ldots \sigma_n^2(x) > 0$, elle est donc symétrique, définie strict. positive, où $J(x) = U(x)\Sigma(\sigma_1(x), \ldots, \sigma_n(x))V(x)$ est la SVD de $J(x)$.
Si $J(x)$ est de rang n et $\nabla f(x)$ non nul, la solution du système (3.8) est une direction de descente :

$$d^t\, \nabla f(x) = d^t\, \nabla f(x) = -d^t J(x)^t J(x) d = -\left\| J(x)d \right\|_2^2 < 0\,,$$

car $J(x)d \neq 0$ sinon, par (3.8),$\nabla f(x) = 0$, d'où

Algorithme 3.7 (Méthode de Gauss-Newton)

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

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

déterminer $d^{(k)}$ *solution de* $J(x^{(k)})^t J(x^{(k)}) d^{(k)} = -J(x^{(k)})^t r(x^{(k)})$

déterminer $\sigma_k > 0$

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

$k = k + 1$

Un résultat de convergence globale est donné par le théorème suivant (voir page 28)

Théorème 3.3.1
On suppose que les fonctions résidus r_k *sont telles que* ∇f *est Lipschitz dans un voisinage* $\mathcal{V}$ *de l'ensemble de niveau* $L_f(f(x^{(0)})) = \{x \in \mathbb{R}^d \mid f(x) \leq f(x^{(0)})\}$ *et telles que que* $J(x)$ *vérifie :*

$$\exists \gamma \in \mathbb{R}_+^*,\, \forall x \in \mathcal{V},\, \forall z \in \mathbb{R}^d \,:\, \|J(x)z\|_2 \geq \gamma \|z\|_2\,.$$

Alors si la suite $(x^{(k)})_k$ *est construite grâce à l'algorithme de* Gauss-Newton *avec des pas* σ_k *vérifiant les conditions de* Wolfe *(page 27) alors*

$$\lim_{k \to +\infty} J(x^{(k)})^t r(x^{(k)}) = 0\,.$$

Remarques :

1. La constante $\gamma > 0$ est un minimiseur uniforme des valeurs singulières de $J(x)$: $0 < \gamma \leq \inf_{x \in L_f(x^{(0)})} \sigma_n(x)$.
2. Pour résoudre le système $J(x)^t J(x)\, d = -\nabla f(x)$ on peut utiliser l'algorithme de Cholesky.
3. Un avantage par rapport à la méthode de Newton est que l'on n'a pas besoin de calculer $\nabla^2 r_k(x)$ à chaque fois.

3.3.2 Méthode de Levenberg-Marquardt

Dans cette méthode on détermine en même temps la direction et le pas de descente à l'intérieur d'une «région de confiance».

Au voisinage d'un point x on va utiliser un modèle quadratique pour f :

$$\tilde{f}_x(d) = f(x) + \nabla f(x)^t\, d + \frac{1}{2} d^t J(x)^t J(x) d = \frac{1}{2}\|r(x)\|^2 + d^t J(x)^t r(x) + \frac{1}{2} d^t J(x)^t J(x) d\,.$$

On cherche à minimiser à chaque itération la fonction $\tilde{f}$ qui dépend de x, pour d appartenant à un voisinage de 0, par exemple une boule de rayon Δ, ce qui peut encore s'écrire

$$\min_d \frac{1}{2}\, \|J(x)d + r(x)\|_2^2\,, \qquad \text{avec } \|d\| \leq \Delta\,. \tag{3.9}$$

Supposons que l'on sache déterminer la solution de (3.9), il faut en plus pouvoir déterminer, à chaque itération la valeur de Δ . Pour celà on va considérer le rapport de la réduction vraie et de la réduction due au modèle :

$$\rho = \frac{f(x) - f(x+d)}{\tilde{f}_x(0) - \tilde{f}_x(d)} \tag{3.10}$$

Comme d minimise (3.9) le dénominateur est positif ou nul, et si l'on ne peut pas accepter d, on recommence à déterminer d avec une valeur plus petite pour Δ ; si la valeur de ρ est proche de 1 on augmente Δ seulement si $\|d\| = \Delta$, sinon on garde la même région.

Dans l'algorithme ci-dessous on distinguera trois parties : calcul de d et ρ, mise à jour de la taille de la région Δ et de la nouvelle valeur du point minimisant x .

L'algorithme général s'écrit :

Algorithme 3.8 (Région de confiance)

choisir $\Delta_0 \in]0, \widehat{\Delta}[$, $\eta \in [0, \frac{1}{4}[$ *et* $k = 0$

tant que $\left\|\nabla f(x^{(k)})\right\| = \left\|J(x^{(k)})^t r(x^{(k)})\right\| > \varepsilon$
  déterminer $d^{(k)}$ *solution (approchée) de* (3.9)
  déterminer ρ_k *grâce à* (3.10)
  si $\rho_k \leq \frac{1}{4}$ ***alors***
    $\Delta_{k+1} = \Delta_k/4$
  sinon
    si $\rho_k > \frac{3}{4}$ *et* $\|d^{(k)}\| = \Delta_k$ ***alors***

$\Delta_{k+1} = \min(2\Delta_k, \widehat{\Delta})$
sinon
$\Delta_{k+1} = \Delta_k$
si $\rho_k > \eta$ ***alors***
$x^{(k+1)} = x^{(k)} + d^{(k)}$
sinon
$x^{(k+1)} = x^{(k)}$
$k = k + 1$

La solution de (3.9) est caractérisée grâce au

Lemme 3.3.1
Le vecteur d^{LM} est solution du sous-problème

$$\min_d \frac{1}{2} \|J(x)d + r(x)\|_2^2 , \qquad \text{avec } \|d\| \leq \Delta ,$$

pour $\Delta > 0$ si et seulement si il existe $\lambda \in \mathbb{R}_+$ tel que

$$(J(x)^t J(x) + \lambda I)d^{LM} = -J(x)^t r(x)$$
$$\lambda(\Delta - \|d^{LM}\|) = 0$$

Enfin, On peut montrer le résultat de convergence

Théorème 3.3.2
On suppose que les fonctions r_i sont de classe $\mathcal{C}^2$ dans un voisinage de $L_f(f(x^{(0)})) = \{x \in \mathbb{R}^d \mid f(x) \leq f(x^{(0)})\}$ et que dans l'algorithme 3.8, à chaque itération k, la solution $d^{(k)}$ du sous-problème (3.9) *est exacte, caractérisé par le lemme précédent. On a alors*

$$\lim_{k \to +\infty} J(x^{(k)})^t r(x^{(k)}) = 0 .$$

Annexe A

Rappels

A.1 Algèbre linéaire et analyse matricielle

On note $A = (a_{ij})$ une matrice et $a_{ij} = (A)_{ij}$ est l'élément de la i-ième ligne , j-ième colonne. Les majuscules seront en général réservées aux matrices et les minuscules aux éléments des matrices.

Proposition A.1.1
Soit $(E, <.,.>)$ un espace euclidien, $u \in \mathcal{L}(E)$ un endomorphisme de E.
Il existe un élément $u^* \in \mathcal{L}(E)$ unique, vérifiant :

$$\forall (x, y) \in E^2 : < u(x), y > = < x, u^*(y) > \ .$$

On appelle u^* l'endomorphisme adjoint de u.

Proposition A.1.2
Soient u et v des éléments de $\mathcal{L}(E)$, α, β des réels, alors :

$$(\alpha\, u + \beta\, v)^* = \alpha\, u^* + \beta\, v^* \,; \quad (u^*)^* = u \,; \quad (u \circ v)^* = v^* \circ u^* \,;$$

$$\ker u^* = (\mathrm{I}m\, u)^{\perp} \,; \quad \mathrm{I}m\, u^* = (\ker u)^{\perp} \,.$$

Définition A.1.1
Soit $(E, <.,.>)$ un espace euclidien, $u \in \mathcal{L}(E)$:
- *u est dit autoadjoint ou symétrique si et seulement si $u^* = u$;*
- *u est dit orthogonal si et seulement si $u \circ u^* = Id_E$, resp. $u^* = u^{-1}$;*
- *u est dit antisymétrique si et seulement si $u^* = -u$;*
- *u est dit normal si et seulement si $u \circ u^* = u^* \circ u$.*

Note

Remarques :
Un endomorphisme *autoadjoint* vérifie : $\forall (x, y) \in E^2 : < u(x), y > = < x, u(y) >$.
Un endomorphisme *orthogonal* vérifie : $\forall (x, y) \in E^2 : < u(x), u(y) > = < x, y >$.

Proposition A.1.3 (Écriture matricielle)
Soit $(E, <.,.>)$ un espace euclidien et $\mathcal{B} = \{e_1, \ldots, e_n\}$ une base de E.
Au vecteur $x = \sum_{i=1}^{n} x_i e_i \in E$ on associe le vecteur de coordonnées $X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \in \mathcal{M}(n,1)$.
Soit ϕ une forme bilinéaire sur E^2, on note $\Phi = (\phi(e_i, e_j))_{1 \le i,j \le n} \in \mathcal{M}(n,n)$, la matrice associée à ϕ dans la base $\mathcal{B}$, alors

$$\forall (x,y) \in E^2 \ : \phi(x,y) = X^t \, \Phi \, Y \ .$$

En particulier, pour $S = (< e_i, e_j >)_{1 \le i,j \le n}$ on a : $< x, y >= X^t \, S \, Y$.

Soit $u \in \mathcal{L}(E)$, on note $U = Mat_{\mathcal{B}} u$ et $U^* = Mat_{\mathcal{B}} u^*$, alors $U^* = S^{-1} \, U^t \, S$.
Cas important : la base $\mathcal{B}$ est orthonormée, on a $S = I_n$, alors :

$$< x, y >= X^t \, Y \quad \text{et} \quad U^* = U^t \ .$$

Si u est autoadjoint, la matrice U est symétrique $U = U^t$.
Si u est orthogonal, la matrice U est orthogonale et vérifie $U^t = U^{-1}$.
Si u est antisymétrique, la matrice U est antisymétrique et vérifie $U^t = -U$.
Si u est normal, la matrice U est normale et vérifie $U \, U^t = U^t \, U$.
Les propriétés de la proposition A.1.2 se traduisent matriciellement de façon évidente :

$$(U^t)^t = U \ \text{ et } \ (UV)^t = V^t \, U^t .$$

Proposition A.1.4 (Réduction d'une forme quadratique)
Soit $(E, <.,.>)$ un espace euclidien et $\mathcal{B} = \{e_1, \ldots, e_n\}$ une base de E.
Soit ϕ une forme bilinéaire symétrique sur E^2 et q la forme quadratique associée définie, pour tout $x \in E$, par $q(x) = \phi(x,x)$.
La matrice de ϕ dans la base $\mathcal{B}$, $\Phi \in \mathcal{M}(n,n)$, est réelle et symétrique. Elle admet n valeurs propres réelles $\lambda_1 \le \ldots \le \lambda_n$ et les vecteurs propres associés $\{v_1, \ldots, v_n\}$ forment une base orthonormée de E. La matrice $P = (V_1 \cdots V_n)$, dont les colonnes sont composées des coordonnées des v_i est orthogonale, $P^{-1} = P^t$ et $P^t \, \Phi \, P = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$.

$$\text{On a, pour tout } x = \sum_{i=1}^{n} x_i e_i = \sum_{i=1}^{n} \tilde{x}_i v_i \text{ de } E : \qquad q(x) = \sum_{i,j=1}^{n} \Phi_{ij} x_i x_j = \sum_{i=1}^{n} \lambda_i \tilde{x}_i^2 \ ,$$

et

$$\lambda_1 \|x\|^2 \le X^t \, \Phi \, X \le \lambda_n \|x\|^2 \ .$$

Si toutes les valeurs propres sont strictement positives, Φ, resp. q, est définie positive :
Si toutes les valeurs propres sont strictement négatives, Φ, resp. q, est définie négative.
Dans tous les autres cas, Φ, resp. q, change de signe ou s'annule.
En particulier, si 0 est valeur propre, Φ, resp. q, est dégénérée.

Définition A.1.2
– *On dit qu'une matrice A réelle symétrique est définie positive* (cf. *ci-dessus*) *si pour tout $w \in \mathbb{R}^n$, $w \neq 0$ $w^t A w > 0$;*
la matrice A est semi-définie positive si pour tout $w \in \mathbb{R}^n$ $w^t A w \ge 0$;.

– *Le rayon spectral d'une matrice carrée A d'ordre n est défini par*

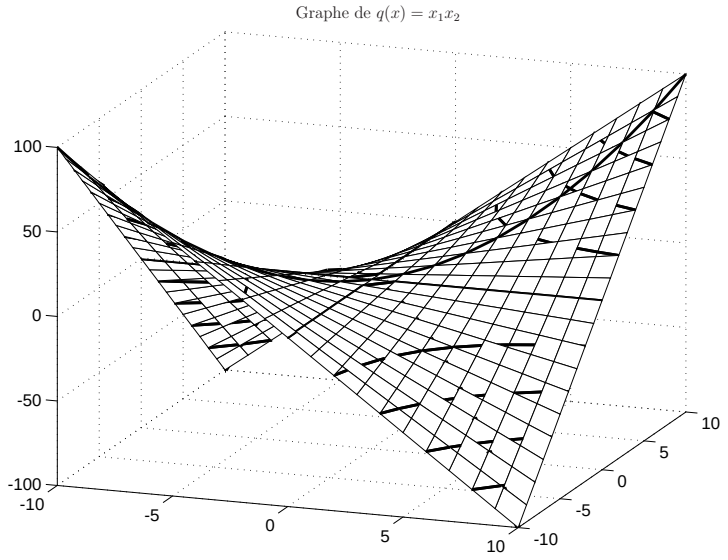
$$\rho(A) = \max_{1\leq i\leq n}\{|\lambda_i| \,|\, \lambda_i \textit{valeur propre de } A\} \ .$$

Exemple : Sur $\mathbb{R}^2$, la forme quadratique

$$q(x) = x_1x_2 = \frac{1}{2}\left(\frac{x_1+x_2}{\sqrt{2}}\right)^2 - \frac{1}{2}\left(\frac{x_1-x_2}{\sqrt{2}}\right)^2$$

change de signe et

$$\min_{\|x\|_2\leq 1} q(x) = -1/2, \ \text{ atteint en } \left(+1/\sqrt{2}, -1/\sqrt{2}\right) \text{ et } \left(-1/\sqrt{2}, +1/\sqrt{2}\right) ;$$
$$\max_{\|x\|_2\leq 1} q(x) = +1/2, \ \text{ atteint en } \left(+1/\sqrt{2}, +1/\sqrt{2}\right) \text{ et } \left(-1/\sqrt{2}, -1/\sqrt{2}\right) \ .$$



Pour simplifier un calcul, une démonstration ou présenter un algorithme, on considère souvent les décompositions par blocs de matrices.

Définition A.1.3 (Matrices blocs)
– *Soient A et B des matrices de même dimensions avec*

$$A = \begin{pmatrix} A_{11} & \dots & A_{1q} \\ \vdots & & \vdots \\ A_{p1} & \dots & A_{pq} \end{pmatrix} \quad et \quad B = \begin{pmatrix} B_{11} & \dots & B_{1q} \\ \vdots & & \vdots \\ B_{p1} & \dots & B_{pq} \end{pmatrix}$$

où les blocs respectifs A_{ij} et B_{ij} ont même dimensions. Alors on obtient l'addition par blocs de A et B :

$$A + B = \begin{pmatrix} A_{11}+B_{11} & \dots & A_{1q}+B_{1q} \\ \vdots & & \vdots \\ A_{p1}+B_{p1} & \dots & A_{pq}+B_{pq} \end{pmatrix} \ .$$

– *Soient A et B des matrices tel que AB est défini et*

$$A = \begin{pmatrix} A_{11} & \dots & A_{1q} \\ \vdots & & \vdots \\ A_{p1} & \dots & A_{pq} \end{pmatrix} \quad et \quad B = \begin{pmatrix} B_{11} & \dots & B_{1s} \\ \vdots & & \vdots \\ B_{r1} & \dots & B_{rs} \end{pmatrix} ,$$

si $q = r$ et si les produits $A_{ik}B_{kj}$ $(i = 1, \dots, i\,;\, j = 1, \dots, s\,;\, k = 1, \dots, q)$ sont définis, alors la multiplication par blocs de A et B s'écrit :

$$AB = \begin{pmatrix} \sum_{k=1}^{q} A_{1k}B_{k1} & \dots & \sum_{k=1}^{q} A_{1k}B_{ks} \\ \vdots & & \vdots \\ \sum_{k=1}^{q} A_{pk}B_{k1} & \dots & \sum_{k=1}^{q} A_{pk}B_{ks} \end{pmatrix} .$$

– *Si A est une matrice diagonale par blocs $A = (A_{ij})$ $(1 \le i, j \le r)$, c.-à-d. les blocs A_{ij} ont comme dimension (p_i, q_j) $(1 \le i, j \le r)$ avec $A_{ij} = 0$ si $i \neq j$ et $p_i = q_i$ pour A_{ii}. On peut calculer le déterminant par blocs :*

$$dét(A) = \prod_{i=1}^{r} dét(A_{ii}) .$$

Exemple : Cas particulier $r = q = 2$, $s = 1$, alors

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \in \mathcal{M}(d, d) \text{ et } x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^d \;:\; Ax = \begin{pmatrix} A_{11}x_1 + A_{12}x_2 \\ A_{21}x_1 + A_{22}x_2 \end{pmatrix} .$$

Définition A.1.4 (Norme matricielle)
Une norme matricielle sur l'e.v. des matrices carrées d'ordre n, $\mathcal{M}_n$, est une application de $\mathcal{M}_n$ vers $\mathbb{R}_+$ vérifiant, pour tous $A, B \in \mathcal{M}_n$, $\alpha \in \mathbb{R}$:

$$\begin{aligned} &\|A\| = 0 \Leftrightarrow A = 0 \\ &\|\alpha A\| = |\alpha| \, \|A\| \\ &\|A + B\| \le \|A\| + \|B\| \\ &\|AB\| \le \|A\| \, \|B\| \end{aligned}$$

Définition A.1.5 (Norme matricielle subordonnée)
Soit $\|.\|$ une norme vectorielle, on appelle norme matricielle subordonnée (à cette norme vectorielle) l'application de $\mathcal{M}_n$ vers $\mathbb{R}_+$ définie par

$$|||A||| = \sup_{x \in V, x \neq 0} \frac{\|Ax\|}{\|x\|} = \sup_{x \in V, \|x\| \le 1} \|Ax\| = \sup_{x \in V, \|x\| = 1} \|Ax\| .$$

Proposition A.1.5
Soit $|||.|||$ une norme matricielle subordonnée à la norme vectorielle $\|.\|$

1. *Les trois définitions sont équivalentes et il existe $u \in V$ avec $\|u\| = 1$ tel que $|||A||| = \|Au\|$.*
   Comme le sup *est atteint, on peut le remplacer par un* max.

2. *Une norme matricielle subordonnée est une norme matricielle.*
3. *Pour tous* $A \in \mathcal{M}_n$, $v \in V$: $\|Av\| \leq |||A||| \, \|v\|$.
4. *On a* $|||I_n||| = 1$.

Remarque : en pratique on n'utilise pas la notation $|||.|||$ mais plutôt classique $\|.\|$ pour les normes matricielles subordonnées. Le contexte permet de savoir si on a affaire à une norme de vecteur ou de matrice.

Proposition A.1.6 (Exemples fondamentaux)
Soit $v \in \mathbb{R}^n$ *et* A *une matrice carrée réelle, pour les normes vectorielles usuelles on a les normes matricielles subordonnées suivantes :*

$$\|v\|_\infty = \max_{1\leq i\leq n} |v_i| \qquad et \quad \|A\|_\infty = \max_{1\leq i\leq n} \sum_{j=1}^{n} |a_{ij}| \quad \text{«maximum sur les lignes»}$$

$$\|v\|_1 = \sum_{1\leq i\leq n} |v_i| \qquad et \quad \|A\|_1 = \max_{1\leq j\leq n} \sum_{i=1}^{n} |a_{ij}| \quad \text{«maximum sur les colonnes»}$$

$$\|v\|_\infty = \left(\sum_{1\leq i\leq n} v_i^2 \right)^{1/2} \quad et \quad \|A\|_2 = \sqrt{\rho(A^t\, A)} \qquad \text{où } \rho(A^t\, A) \text{ est le rayon spectral de la matrice symétrique semi-définie positive } A^t A\,.$$

Proposition A.1.7 (Propriétés)
Soit A *une matrice carrée réelle, alors*

1. $\|A\|_1 = \|A^t\|_\infty$;
2. $\|A\|_2 = \|A^t\|_2$;
3. *Si* U *est orthogonale,* $\|UA\|_2 = \|AU\|_2\| = \|A^t\|_2$;
4. *Si* A *est normale,* $\|A\|_2 = \rho(A)$
   Cas particuliers : si A *est symétrique,* $\|\|A\|_2 = \rho(A)$,
   si A *est orthogonale,* $\|\|A\|_2 = 1$

Remarque : On peut avoir des normes sur les matrices qui ne sont pas des normes matricielles car non compatibles avec la multiplication matricielle. De même il n'existe des normes matricielles non subordonnées, comme le montre le résultat suivant.

Proposition A.1.8 (norme de Frobenius)
La norme de Frobenius *définie, pour toute matrice* $A \in \mathcal{M}_n$, *par*

$$\|A\|_F = \left(\sum_{1\leq i,j\leq n} |a_{ij}|^2 \right)^{\frac{1}{2}} = (trace(A^t\, A))^{\frac{1}{2}}\,,$$

est une norme matricielle, non subordonnée qui vérifie :

1. *Si* U *est une matrice orthogonale, alors* $\|UA\|_F = \|AU\|_F = \|A\|_F$.
2. *On a l'encadrement* $\|A\|_2 \leq \|A\|_F \leq \sqrt{n}\, \|A\|_2$.
3. *Pour tout* $x \in V$, $\|Ax\|_2 \leq \|A\|_F \|x\|_2$.

THÉORÈME A.1.1
Soit $A \in \mathcal{M}_n$, alors

1. *Pour toute norme matricielle $\|.\|$ on a $\rho(A) \leq \|A\|$.*
2. *Pour tout $\varepsilon \in \mathbb{R}_+^*$, il existe une norme matricielle subordonnée $\|.\|_\varepsilon$ telle que $\|A\|_\varepsilon \leq \rho(A) + \varepsilon$.*

PROPOSITION A.1.9 (CS)
Soit $A \in \mathcal{M}_n$ et $\|.\|$ une norme matricielle

1. *Si $\|A\| < 1$, alors $\lim_{k\to+\infty} A^k = O$.*
2. *Si $I_n - A$ est une matrice singulière, alors $\|A\| \geq 1$.*
3. *Si $\|A\| < 1$, alors $I_n - A$ est inversible et*

$$(I_n - A)^{-1} = \sum_{k=0}^{+\infty} A^k \quad \text{avec} \quad \|(I_n - A)^{-1}\| \leq \frac{1}{1 - \|A\|} .$$

THÉORÈME A.1.2 (CNS)
Soit $A \in \mathcal{M}_n$, alors

1. $\rho(A) < 1 \iff \lim_{k\to+\infty} A^k = O \iff \lim_{k\to+\infty} A^k v = 0$, *pour tout* $v \in V$
2. $\rho(A) < 1 \iff \sum_{k=0}^{+\infty} A^k = (I_n - A)^{-1}$.

A.2 Calcul différentiel dans $\mathbb{R}^n$

Un point $x = (x_1, \ldots, x_n)$ de $\mathbb{R}^n$ considéré comme vecteur sera noté $x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$
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 $(x, y)_2 =< x, y >= x^t y = \sum_{i=1}^n x_i y_i$.

DÉFINITION A.2.1
Soit Ω un ouvert de $\mathbb{R}^n$ et $f : \Omega \to \mathbb{R}^m$. Soit $a \in \Omega$ et $v \in \mathbb{R}^n$, la dérivée de f au point a, dans la direction v, est définie par

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

Si $v = e_i$, on obtient la dérivée partielle de f par rapport à la i^evariable, notée, $D_i f(a) \in \mathbb{R}^m$.
Si $m = 1$, $f : \Omega \to \mathbb{R}$, on note aussi $D_i f(a) = \dfrac{\partial f}{\partial x_i}(a)$.

DÉFINITION A.2.2
Soit Ω un ouvert de $\mathbb{R}^n$ et $f : \Omega \to \mathbb{R}^m$. On dit que f est différentiable en $a \in \Omega$ s'il existe une application linéaire $L \in \mathcal{L}(\mathbb{R}^n, \mathbb{R}^m)$ qui vérifie

$$\|f(a+u) - f(a) - L(u)\| = o(\|u\|) \qquad (u \in \mathbb{R}^n) \, .$$

On note $L = df_a$ la différentielle de f en a et $Df(a) \in \mathcal{M}(m,n)$ la matrice associée est appelée matrice jacobienne.

PROPOSITION A.2.1
Soit Ω un ouvert de $\mathbb{R}^n$ et $f : \Omega \to \mathbb{R}^m$ différentiable au point $a \in \Omega$, alors pour tout $v \in \mathbb{R}^n$:

$$D_v f(a) = df_a(v) \; = Df(a)\, v.$$

Si on note f_1,..., f_m, les fonctions coordonnées de f, alors la matrice jacobienne s'écrit

$$\mathrm{D}f(a) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(a) & \frac{\partial f_1}{\partial x_2}(a) & \dots & \frac{\partial f_1}{\partial x_n}(a) \\ \frac{\partial f_2}{\partial x_1}(a) & \frac{\partial f_2}{\partial x_2}(a) & \dots & \frac{\partial f_2}{\partial x_n}(a) \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1}(a) & \frac{\partial f_m}{\partial x_2}(a) & \dots & \frac{\partial f_m}{\partial x_n}(a) \end{pmatrix} .$$

Si $m = 1$, $f : \Omega \to \mathbb{R}$, on a : $Df(a) = \left(\frac{\partial f}{\partial x_1}(a) \quad \cdots \quad \frac{\partial f}{\partial x_n}(a) \right)$.

Note : la réciproque est fausse, il existe des fonctions pour lesquelles toutes les dérivées directionnelles existent et qui ne sont pas différentiables.

PROPOSITION A.2.2
Soit Ω un ouvert de $\mathbb{R}^n$ et $f : \Omega \to \mathbb{R}$ différentiable au point $a \in \Omega$, alors pour tout $v \in \mathbb{R}^n$:

$$D_v f(a) = \sum_{i=1}^{n} v_i \, \frac{\partial f}{\partial x_i}(a) = < \nabla f(a), v > = \nabla f(a)^t \, v$$

où $\nabla f(a) = (Df(a))^t = \begin{pmatrix} \frac{\partial f}{\partial x_1}(a) \\ \vdots \\ \frac{\partial f}{\partial x_n}(a) \end{pmatrix}$ *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

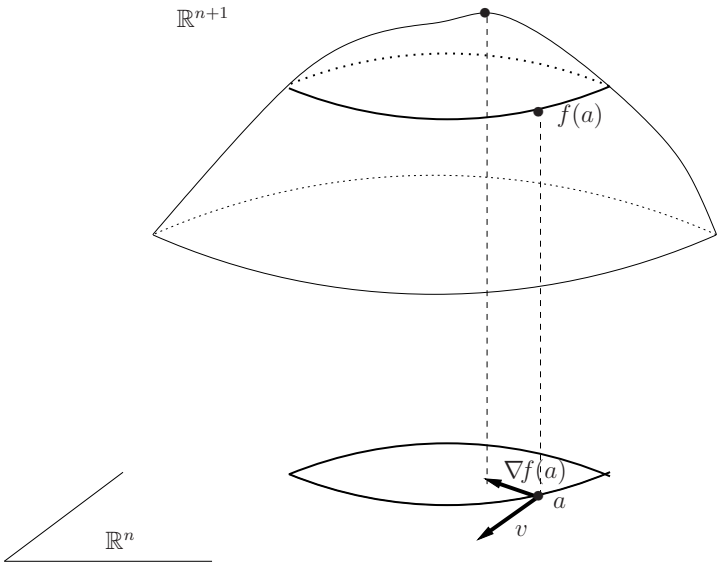
$$\min_{\|v\|_2=1} D_v f(a) = -\|\nabla f(a)\|_2 \text{ et est atteint en } v = -\nabla f(a)/\|\nabla f(a)\|_2 \, ;$$

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

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*".

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

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



Proposition A.2.3
Soit Ω un ouvert de $\mathbb{R}^n$, $f : \Omega \to \mathbb{R}^m$ et $a \in \Omega$. Si au point a toutes les dérivées partielles $D_i f(a)$, $1 \leq i \leq n$, existent et si les fonctions $x \mapsto D_i f(x)$ $1 \leq i \leq n$, sont continues dans un voisinage de a, alors f est différentiable en a.
On dit que f est continûment différentiable, $f \in \mathcal{C}^1(\Omega, \mathbb{R}^m)$, si on a continuité des d.p. en tout point de l'ouvert Ω.

Proposition A.2.4
Soient $f : \mathbb{R}^n \to \mathbb{R}^m$, $g : \mathbb{R}^m \to \mathbb{R}^p$ des fonctions différentiables. On pose $h = g \circ f$, alors $h : \mathbb{R}^n \to \mathbb{R}^p$ est différentiable et, pour tout $a \in \mathbb{R}^n$:

$$dh_a = dg_{f(a)} \circ df_a$$

et

$$\begin{pmatrix} D_1h_1(a) & \dots & D_nh_1(a) \\ \vdots & & \vdots \\ D_1h_p(a) & \dots & D_nh_p(a) \end{pmatrix} = \begin{pmatrix} D_1g_1(f(a)) & \dots & D_mg_1(f(a)) \\ \vdots & & \vdots \\ D_1g_p(f(a)) & \dots & D_mg_p(f(a)) \end{pmatrix} \begin{pmatrix} D_1f_1(a) & \dots & D_nf_1(a) \\ \vdots & & \vdots \\ D_1f_m(a) & \dots & D_nf_m(a) \end{pmatrix}.$$

Exemple : Si $n = p = 1$, $h(a) = g(f_1(a), \dots, f_m(a)) \in \mathbb{R}$, $a \in \mathbb{R}$ et

$$h'(a) = \sum_{i=1}^{m} D_i g(f(a))\, f_i'(a)\ .$$

PROPOSITION A.2.5
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 A.2.6 (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 $h = (h_1, h_2)$:

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