Optimisation Différentiable*

J. Charles Gilbert†

19 février 2008

*Paru dans *Techniques de l'Ingénieur*.

†INRIA-Rocquencourt, BP 105, F-78153 Le Chesnay Cedex (France), `Jean-Charles.Gilbert@inria.fr`.

Cette synthèse raisonnée décrit les principaux algorithmes de résolution des *problèmes d'optimisation différentiable* et en donne leur motivation. Ces problèmes se posent lorsque l'on cherche à déterminer la valeur optimale d'un nombre fini de paramètres. L'optimalité signifie ici la minimalité d'un critère donné. La différentiabilité supposée des fonctions qui définissent le problème écarte d'emblée de notre propos l'*optimisation combinatoire* (les paramètres à optimiser ne prennent que des valeurs entières ou discrètes) et l'*optimisation non lisse* (les fonctions ont des irrégularités).

Les problèmes d'optimisation se présentent dans de nombreux domaines de l'ingénieur, ainsi qu'en science et en économie, souvent après avoir conduit à leur terme les étapes de simulation. Il arrive souvent que ces problèmes se posent en dimension infinie, c'est-à-dire que l'on cherche une fonction optimale plutôt qu'un nombre fini de paramètres optimaux. Il faut alors passer par une phase de discrétisation (en espace, en temps) pour retrouver le cadre qui est le nôtre et se ramener ainsi à un problème qui peut être résolu sur ordinateur. La *transcription directe* des problèmes de *commande optimale* suit une telle procédure de discrétisation.

Les méthodes numériques de l'optimisation ont principalement été développées après la seconde guerre mondiale, en parallèle avec l'amélioration des ordinateurs, et n'ont cessé depuis de s'enrichir. En optimisation non linéaire, on peut ainsi distinguer plusieurs vagues : méthodes de pénalisation, méthode du lagrangien augmenté (1958), méthodes de quasi-Newton (1959), méthodes newtoniennes ou SQP (1976), algorithmes de points intérieurs (1984). Une vague n'efface pas la précédente mais permet d'apporter de meilleures réponses à certaines classes de problèmes, comme ce fut le cas pour les méthodes de points intérieurs en optimisation semi-définie positive (SDP). Une attention particulière sera portée aux algorithmes pouvant traiter les problèmes de grande taille, ceux qui se présentent dans les applications.

Notations. La norme euclidienne (ou ℓ_2) est notée $\|\cdot\|_2$. L'inégalité $v \leq w$ (resp. $u < v$) entre deux vecteurs v et w signifie que $v_i \leq w_i$ (resp. $v_i < w_i$) pour tout indice i. On note $\mathcal{R}(M)$ et $\mathcal{N}(M)$ l'image et le noyau d'une matrice M. Pour indiquer qu'une matrice carrée M est symétrique semi-définie positive (resp. définie positive), on notera $M \succcurlyeq 0$ (resp. $M \succ 0$). L'ensemble des matrices symétriques d'ordre n est noté $\mathbb{S}^n$, $\mathbb{S}^n_+ := \{M \in \mathbb{S}^n : M \succcurlyeq 0\}$ et $\mathbb{S}^n_{++} := \{M \in \mathbb{S}^n : M \succ 0\}$. Une fonction f est dite de classe $C^{m,\alpha}$ si elle est m fois différentiable et si sa dérivée m-ième vérifie pour une constante C et pour tout x et y : $\|f^{(m)}(y) - f^{(m)}(x)\| \leq C\|y - x\|^\alpha$.

1 Outils théoriques, concepts algorithmiques

1.1 Le problème à résoudre

De manière assez formelle, un problème d'optimisation se pose lorsque l'on cherche un point d'un ensemble X en lequel une fonction f définie sur cet ensemble prend une valeur minimale. Nous l'écrirons de la manière suivante

$$(P_X) \quad \begin{cases} \min f(x) \\ x \in X. \end{cases} \tag{1.1}$$

La fonction f est appelée *critère* ou *fonction-coût* du problème. L'ensemble X est appelé l'*ensemble admissible* du problème (surtout s'il fait partie d'un ensemble plus grand) et un point de X est dit *admissible*. Une *solution* de (P_X) est un point $x_* \in X$ tel que $f(x_*) \leq f(x)$ pour tout $x \in X$. On parle aussi de *minimum global*, par opposition à un *minimum local* $x_* \in X$ qui ne vérifie $f(x_*) \leq f(x)$ que pour des $x \in X$ *voisins* de x_* (pour que cette notion de voisinage ait un sens, il faut que X soit un espace topologique). On dit que ces minima sont *stricts* si on a l'inégalité stricte $f(x_*) < f(x)$ pour des $x \in X$ (éventuellement voisins de x_*) et différents de x_*.

La formulation de (P_X) est très générale. Dans cette revue nous nous restreindrons au cas où X est une partie de $\mathbb{R}^n$ décrite par des contraintes fonctionnelles d'égalité et d'inégalité :

$$(P_{EI}) \quad \begin{cases} \min f(x) \\ c_i(x) = 0, \quad i \in E \\ c_i(x) \leq 0, \quad i \in I. \end{cases}$$

Les deux ensembles d'indices E et I sont supposés former une partition de $\{1, \ldots, m\}$, c'est-à-dire que $E \cup I = \{1, \ldots, m\}$ et $E \cap I = \varnothing$, tandis que $f : \mathbb{R}^n \to \mathbb{R}$ et les $c_i : \mathbb{R}^n \to \mathbb{R}$ sont des fonctions différentiables, éventuellement non convexes. On note $m_E = |E|$ et $m_I = |I|$; donc $m = m_E + m_I$. Dans (P_{EI}), l'ensemble admissible s'écrit

$$X := \{x \in \mathbb{R}^n : \ c_E(x) = 0, \ c_I(x) \leq 0\}.$$

Si $v \in \mathbb{R}^m$, on note v_E (resp. v_I) le vecteur de $\mathbb{R}^{m_E}$ (resp. $\mathbb{R}^{m_I}$) formé des composantes v_i de v avec $i \in E$ (resp. $i \in I$). Le problème (P_{EI}) est dit *convexe*, si f est convexe, si les composantes de c_I sont convexes et si c_E est affine.

En face d'un problème d'optimisation comme (P_X), plusieurs questions se posent. La première a trait à l'existence d'une solution et à l'unicité de celle-ci. Rien ne sert en effet d'essayer de résoudre numériquement un problème qui n'a pas de solution ! L'unicité est une propriété appréciée par beaucoup d'algorithmes, mais est moins essentielle. Si le problème de l'existence est souvent difficile, il ne faut pas manquer de vérifier si le résultat standard suivant ne s'applique pas.

Théorème 1.1 (Weierstrass) *Si X est un compact non vide et si $f : X \to \mathbb{R}$ est continue, alors le problème (P_X) a au moins une solution.*

Ce résultat a diverses extensions intéressantes. D'une part, on peut remplacer la continuité de f par sa semi-continuité inférieure. D'autre part, en dimension finie, on peut aussi remplacer X *compact* (un fermé borné en dimension finie) par X fermé et une hypothèse de croissance à l'infini de f :

$$\lim_{\substack{x \in X \\ \|x\| \to \infty}} f(x) = +\infty.$$

En ce qui concerne l'unicité d'une solution, le résultat le plus simple, mais bien utile, est le suivant.

Théorème 1.2 (unicité de solution) *Si X est une partie convexe d'un espace vectoriel $\mathbb{E}$ et si f est strictement convexe sur X, alors (P_X) a au plus une solution.*

1.2 Conditions d'optimalité

Les deux résultats ci-dessus ne sont d'aucune aide pour trouver une solution de (P_{EI}). Ce qu'il nous faut c'est une version analytique de l'optimalité, un ensemble d'équations et d'inéquations qui pourront être résolues par les algorithmes. On sait qu'en l'absence de contraintes, une fonction f a sa dérivée qui s'annule en un minimum x_* : $f'(x_*) = 0$, ce que l'on peut aussi écrire $\nabla f(x_*) = 0$, si

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_i}(x)\right)_{1 \leq i \leq n}$$

désigne le *gradient* de f en x, c'est-à-dire le vecteur de ses dérivées partielles (en fait le gradient dépend du produit scalaire que l'on se donne sur $\mathbb{R}^n$ et la définition ci-dessus est celle qui correspond au *produit scalaire euclidien* $(u, v) \mapsto u^\top v = \sum_{i=1}^n u_i v_i$). C'est cette relation que l'on cherche à généraliser au problème (P_{EI}). Deux observations préliminaires permettront de mieux comprendre ces conditions d'optimalité.

- On dit qu'une contrainte (d'inégalité, $i \in I$) est *active* en x si $c_i(x) = 0$. Seules les contraintes actives en une solution interviennent dans les conditions d'optimalité. Il sera donc utile de les désigner, ce qui se fera en introduisant

$$I^0(x) := \{i \in I : c_i(x) = 0\} \qquad \text{et} \qquad I^0_* := I^0(x_*).$$

- Si on peut représenter l'ensemble admissible X par divers choix de fonctions c_i (il ne change pas, par exemple, si on multiplie ces fonctions par un facteur strictement positif), toutes les représentations ne sont pas agréables pour exprimer l'optimalité. Par exemple, remplacer les $m_E > 1$ contraintes $c_E(x) = 0$ par l'unique contrainte $\|c_E(x)\|_2^2 = 0$ n'est pas une bonne idée. Il existe en réalité une notion de *qualification des contraintes en un point*, qui permet de sélectionner les représentations acceptables (s'il y en a). Elle est toutefois un peu longue à exprimer et difficile à vérifier. En pratique, on utilise plutôt l'*une* des conditions *suffisantes* de qualification suivantes en x.

  (QC-A) $c_{E \cup I^0(x)}$ est affine dans un voisinage de x.

  (QC-S) *Slater*: c_E est *affine* avec c'_E *surjective*, les composantes de $c_{I^0(x)}$ sont *convexes* et on peut trouver un point $\hat{x} \in X$ tel que $c_{I^0(x)}(\hat{x}) < 0$.

  (QC-IL) Les gradients des contraintes actives en x, $\{\nabla c_i(x) : i \in E \cup I^0(x)\}$, sont linéairement indépendants.

  (QC-MF) *Mangasarian-Fromovitz*: si $\sum_{i \in E \cup I^0(x)} \alpha_i \nabla c_i(x) = 0$ avec $\alpha_i \geq 0$ pour $i \in I^0(x)$, alors $\alpha_i = 0$ pour tout $i \in E \cup I^0(x)$.

  Si l'une de ces conditions est vérifiée en x, alors les contraintes sont qualifiées en ce point.

Les conditions nécessaires d'optimalité du *premier ordre* (ainsi dénommées car seules les dérivées premières y interviennent) énoncées ci-dessous ont été attribuées à Karush, Kuhn et Tucker (KKT), bien qu'ils ne soient pas les seuls à y avoir contribué.

Théorème 1.3 (CN1 – Karush, Kuhn et Tucker) *Soit x_* un minimum local de (P_{EI}). Supposons que f et $c_{E\cup I_*^0}$ soient dérivables en x_* et que les contraintes soient qualifiées en x_*. Alors, il existe $\lambda_* \in \mathbb{R}^m$ tel que l'on ait*

$$(KKT) \quad \left\{ \begin{array}{ll} (a) & \nabla f(x_*) + c'(x_*)^\top \lambda_* = 0 \\ (b) & c_E(x_*) = 0 \\ (c) & c_I(x_*) \leq 0 \\ (d) & (\lambda_*)_I \geq 0 \\ (e) & (\lambda_*)_I^\top c_I(x_*) = 0. \end{array} \right. \tag{1.2}$$

L'identité (a) s'écrit aussi $\nabla_x \ell(x_*, \lambda_*) = 0$, où ℓ est le *lagrangien* du problème (P_{EI}), c'est-à-dire la fonction $\ell : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, définie en (x, λ) par

$$\ell(x, \lambda) = f(x) + \lambda^\top c(x) = f(x) + \sum_{i=1}^{m} \lambda_i c_i(x). \tag{1.3}$$

Le vecteur λ_* est appelé *multiplicateur*, car il multiplie la contrainte dans le lagrangien. On notera qu'il y a un multiplicateur (une composante de λ) par contrainte. Un point x_* pour lequel il existe un multiplicateur λ_* tel que (1.2) ait lieu est appelé *stationnaire*.

Que signifie ces conditions (1.2), qui paraissent bien compliquées ? Il s'agit en fait d'une expression analytique d'une condition géométrique de l'optimalité, qui est relativement aisée de retrouver à partir de (1.2).

- Observons d'abord que l'on retrouve la condition $\nabla f(x_*) = 0$ s'il n'y a pas de contrainte.
- S'il n'y a que des contraintes d'égalité ($I = \varnothing$), elles expriment l'admissibilité de x_* (condition (b)) et le fait que $\nabla f(x_*) \in \mathcal{R}(c'(x_*)^\top) = \mathcal{N}(c'(x_*))^\perp$ (condition (a)), c'est-à-dire que $\nabla f(x_*)$ est orthogonal à l'espace tangent aux contraintes (le noyau $\mathcal{N}(c'(x_*))$ est l'ensemble des directions suivant lesquelles c ne varie pas au premier ordre en x_*; il s'agit donc bien de l'espace tangent aux contraintes en x_*). Géométriquement (voir la figure 1.1), cette condition exprime que le plan tangent

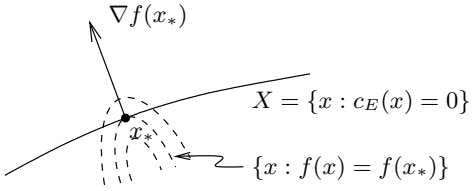


Figure 1.1: Conditions d'optimalité de Lagrange

à la variété $\{x : f(x) = f(x_*)\}$ contient le plan tangent à X en x_*.

- Supposons à présent qu'il n'y ait que des contraintes d'inégalité ($E = \varnothing$), pour simplifier. On remarque qu'alors les multiplicateurs ont un signe (condition (d)) et que la condition (e) s'écrit aussi $(\lambda_*)_i c_i(x_*) = 0$ pour tout $i \in I$ (on utilise (c) et (d)), c'est-à-dire que soit $(\lambda_*)_i = 0$ soit $c_i(x_*) = 0$, ou encore pour $i \in I$:

$$c_i(x_*) < 0 \quad \Longrightarrow \quad (\lambda_*)_i = 0.$$

Ceci montre que les contraintes inactives en x_* n'interviennent pas dans les conditions de KKT, ce que l'on avait déjà signalé. On comprend pourquoi (e) porte le nom de *conditions de complémentarité*. Si, pour $i \in I$, on a

$$c_i(x_*) < 0 \quad \Longleftrightarrow \quad (\lambda_*)_i = 0$$

on dit que l'on a *complémentarité stricte*. À présent la condition (a) s'écrit

$$\nabla f(x_*) = \sum_{i \in I_*^0} \underbrace{(\lambda_*)_i}_{\geq 0} (-\nabla c_i(x_*)).$$

Géométriquement (voir la figure 1.2), cette identité exprime que le gradient $\nabla f(x_*)$

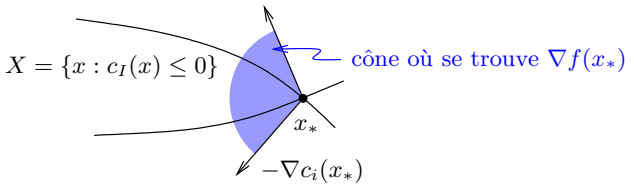


Figure 1.2: Conditions d'optimalité de KKT

est dans le *cône* engendré par l'opposé des gradients des contraintes actives en x_*. La figure 1.2 montre à l'évidence que l'optimisation différentiable repose sur l'analyse convexe, pas seulement sur l'algèbre linéaire (un cône n'est pas un objet de cette dernière).

Pour les problèmes d'optimisation convexe, les conditions de KKT sont suffisantes pour entraîner l'optimalité *globale*.

Théorème 1.4 (CS1) *Si le problème (P_{EI}) est convexe et si (x_*, λ_*) vérifie les conditions de Karush, Kuhn et Tucker* (1.2) *(f et $c_{E \cup I_*^0}$ sont supposées dérivables en x_*), alors x_* est un minimum global de (P_{EI}).*

Malgré l'importance et la complexité du sujet, nous serons plus concis sur les conditions du second ordre, celles qui font intervenir les dérivées secondes de f et c. On cherche ici à généraliser au problème (P_{EI}) la condition selon laquelle une fonction f

a son hessien semi-défini positif en un minimum local, ce que l'on peut aussi écrire $\nabla^2 f(x_*) \succcurlyeq 0$, si

$$\nabla^2 f(x) = \left(\frac{\partial^2 f}{\partial x_i \partial x_j}(x) \right)_{1 \leq i \leq n,\ 1 \leq j \leq n}$$

désigne le *hessien* de f en x, c'est-à-dire la matrice de ses dérivées partielles secondes (comme le gradient, le hessien dépend du produit scalaire que l'on se donne sur $\mathbb{R}^n$ et la définition ci-dessus est celle qui correspond au *produit scalaire euclidien*). Il y aura deux aspects nouveaux par rapport à cette condition simple. D'une part, nous aurons des informations sur le hessien du lagrangien en (x_*, λ_*), pas sur celui de f en x_*. D'autre part, la forme quadratique associée à ce hessien ne sera semi-définie positive que suivant des directions d appartenant au *cône critique*

$$C_* := \{d \in \mathbb{R}^n : c'_E(x_*) \cdot d = 0,\ c'_{I^0_*}(x_*) \cdot d \leq 0,\ f'(x_*) \cdot d \leq 0\} \tag{1.4}$$

et pour un multiplicateur optimal c'est-à-dire pour un élément particulier de

$$\Lambda(x_*) := \{\lambda_* : (x_*, \lambda_*) \text{ est solution de (1.2)}\}$$

dépendant de cette direction d. Les conditions nécessaires d'optimalité du second ordre (CN2) sont les suivantes.

Théorème 1.5 (CN2) *Soit x_* un minimum local de (P_{EI}). Supposons que f et c_E soient C^2 dans un voisinage de x_*, que $c_{I^0_*}$ soit deux fois dérivable en x_* et que $c_{I \setminus I^0_*}$ soit continue en x_*. Supposons également que les conditions de qualification de Mangasarian-Fromovitz (QC-MF) aient lieu en x_*. Alors*

$$\forall d \in C_*, \quad \exists \lambda_* \in \Lambda(x_*) : \quad d^\top \nabla^2_{xx} \ell(x_*, \lambda_*) d \geq 0. \tag{1.5}$$

Si (x_*, λ_*) vérifie les conditions d'optimalité du premier ordre (1.2), on peut récrire le cône critique comme suit :

$$C_* := \{d \in \mathbb{R}^n : c'_{E \cup I^{0+}_*}(x_*) \cdot d = 0,\ c'_{I^{00}_*}(x_*) \cdot d \leq 0\}, \tag{1.6}$$

où l'on a noté

$$I^{0+}_* := \{i \in I^0_* : (\lambda_*)_i > 0\} \quad \text{et} \quad I^{00}_* := \{i \in I^0_* : (\lambda_*)_i = 0\}.$$

Les contraintes d'indices $i \in I^{0+}_*$ sont dites *fortement actives* et celles d'indices $i \in I^{00}_*$ sont dites *faiblement actives*. Ces dernières, bien qu'actives ($c_i(x_*) = 0$), peuvent être ôtées du problème sans modifier la stationnarité de x_* ($(\lambda_*)_i = 0$). La forme (1.6) du cône critique montre qu'il se réduit au noyau de $c'_E(x_*)$, si le problème n'a que des contraintes d'égalité.

Nous concluons cette section par des conditions suffisantes d'optimalité du second ordre (CS2).

Théorème 1.6 (CS2) *Supposons que f et $c_{E\cup I_*^0}$ soient dérivables dans un voisinage d'un point $x_* \in \mathbb{R}^n$ et deux fois dérivables en x_*. Supposons également que $\Lambda(x_*) \neq \varnothing$. Supposons enfin que, pour une norme arbitraire $\|\cdot\|$, on ait*

$$\exists\, \bar\gamma > 0, \quad \forall d \in C_*, \quad \exists\, \lambda_* \in \Lambda(x_*) : \quad d^\top \nabla^2_{xx} \ell(x_*, \lambda_*) d \geq \bar\gamma \|d\|^2. \tag{1.7}$$

Alors, pour tout $\gamma \in [0, \bar\gamma[$, il existe un voisinage V de x_* tel que pour tout $x \in X \cap V$, différent de x_* :

$$f(x) > f(x_*) + \frac{\gamma}{2}\|x - x_*\|^2. \tag{1.8}$$

En particulier, x_* est un minimum local strict de (P_{EI}).

1.3 Prolégomènes à l'algorithmique

1.3.1 Quelques principes

En général, on ne peut pas trouver une solution d'un problème d'optimisation en un nombre fini d'étapes. Les exceptions sont peu nombreuses et on parle alors de *terminaison finie*; citons l'optimisation quadratique convexe non contrainte et l'optimisation linéaire. Le plus souvent, les algorithmes gérèrent des suites de points, que l'on appelle *itérés*, qu'ils s'efforcent de faire converger vers une solution. Même en optimisation linéaire, on préfère parfois éviter l'*algorithme du simplexe*, à terminaison finie mais qui peut requérir un nombre exponentiel d'itérations, pour les *algorithmes de points intérieurs* qui n'ont pas de terminaison finie mais trouvent une solution approchée très rapidement.

La qualité des algorithmes s'apprécie selon divers critères : leur capacité à générer des suites convergentes (on parle de *convergence globale*), la vitesse de convergence de celles-ci dans le pire des cas (section 1.3.2), le nombre d'opérations requises pour obtenir une solution à une précision donnée (on parle de *complexité*), etc.

Les méthodes numériques d'optimisation n'ont pas toutes le même rôle. Celui-ci est reflété par les résultats de convergence qu'elles permettent d'obtenir. On peut ainsi distinguer des méthodes *locales*, qui ont la propriété de générer des suites rapidement convergentes pourvu que les itérés ne soient pas trop éloignés d'une solution (sections 2.2, 3.1.1). Ces méthodes dérivent en général d'un procédé de linéarisation. Par ailleurs, il y a des *méthodes de globalisation*, qui sont conçues pour forcer la convergence des itérés, même si le premier d'entre eux est éloigné d'une solution (section 2.1). Un code d'optimisation combine toutes ces techniques.

1.3.2 Vitesse de convergence des suites

Si $\{x_k\}_{k\geq 1}$ est une suite qui converge vers x_*, avec $x_k \neq x_*$ pour tout k, on dit qu'elle converge

- *linéairement*, s'il existe une norme $\|\cdot\|$, un réel $\tau < 1$ et un indice k_0 tels que pour tout $k \geq k_0$: $\|x_{k+1} - x_*\|/\|x_k - x_*\| \leq \tau$,

- *super-linéairement*, si $\|x_{k+1} - x_*\|/\|x_k - x_*\| \to 0$,
- *quadratiquement*, si $\|x_{k+1} - x_*\|/\|x_k - x_*\|^2$ forme une suite bornée.

Il s'agit évidemment de propriétés asymptotiques (elles ne sont pas affectées par une modification d'un nombre fini de x_k), de plus en plus restrictives (linéaire $\Leftarrow$ super-linéaire $\Leftarrow$ quadratique) et qui rendent les suites qui les possèdent de plus en plus attrayantes. On peut montrer qu'une suite convergeant quadratiquement double asymptotiquement le nombre de chiffres significatifs corrects de ses itérés à chaque itération.

Il y a une *règle heuristique* selon laquelle plus les fonctions décrivant le problème ont un ordre élevé de différentiabilité et plus un algorithme sachant bien utiliser ces dérivées générera des suites rapidement convergentes. Ainsi les algorithmes newtoniens (sections 2.2.1 et 3.1.1), qui utilisent les dérivées secondes, génèrent des suites quadratiquement convergentes, alors que les algorithmes quasi-newtoniens bien conçus (sections 2.2.2, 2.3.3 et 3.1.1), qui n'utilisent que les dérivées premières, génèrent des suites super-linéairement convergentes. En pratique on se limite aux dérivées secondes, qui donnent déjà une vitesse de convergence appréciable (une exception : les *méthodes tensorielles* qui sont utilisées pour faire face à des hessiens singuliers). En optimisation différentiable, on considère qu'une convergence linéaire n'est pas satisfaisante.

1.3.3 Calcul des dérivées

Pour converger rapidement, les algorithmes utilisent les dérivées des fonctions définissant le problème (P_{EI}). Rappelons les deux types de dérivation d'une fonction $\varphi : \mathbb{E} \to \mathbb{F}$ ($\mathbb{E}$ et $\mathbb{F}$ sont deux espaces vectoriels de dimension finie).

- Dans la dérivation *directionnelle*, aussi appelée *en mode direct* ou *tangente*, on se donne une direction $d \in \mathbb{E}$ et on calcule la dérivée directionnelle, qui est l'élément de $\mathbb{F}$ suivant
$$\varphi'(x) \cdot d = \lim_{t \downarrow 0} \frac{\varphi(x + td) - \varphi(x)}{t}. \tag{1.9}$$
- Dans la dérivation *en mode inverse* ou *cotangente*, on se donne une direction $d \in \mathbb{F}$ et on calcule le gradient de la fonction $x \in \mathbb{E} \mapsto d^\top \varphi(x) \in \mathbb{R}$, qui est un élément de $\mathbb{E}$.

Le calcul de dérivées est une opération mécanique, mais aussi fastidieuse. Heureusement, elle peut parfois être automatisée. Il y a plusieurs techniques.

- La différentiation par *différences finies* calcule les dérivées en évaluant la fonction en des points voisins, disons x et $x+td$, et en approchant la dérivée par le quotient différentiel (1.9). Cette approche a deux inconvénients : elle est imprécise et non adaptée au calcul de gradient (requiert un temps proportionnel au nombre de variables). On l'utilise surtout lorsque l'on veut voir rapidement ce qu'un algorithme donnerait avec un calcul de dérivées ou lorsque les fonctions sont trop compliquées pour pouvoir mettre au point un calcul précis de dérivées.
- La différentiation *symbolique* travaille sur une représentation symbolique de la fonction à différentier. Cette approche n'est pas toujours possible (par exemple

si la fonction est définie par un processus itératif) et la dérivation cotangente est rarement implémentée.

- La différentiation *automatique* ou *par programme* travaille sur une représentation de la fonction par un programme informatique écrit dans un langage de haut niveau tel que Fortran, C, Matlab, Scilab, etc. Le différentiateur génère un autre programme qu'il faut exécuter pour calculer la dérivée en un point arbitraire. Les deux modes de différentiation sont possibles. Le mode cotangent est en fait une généralisation de la *méthode de l'état adjoint*, bien connue en *commande optimale*. Son implémentation dans les différentiateurs automatiques donne lieu à un problème d'encombrement mémoire, qui limite encore son utilisation (notons toutefois qu'elle est utilisée pour des problèmes de prévision météorologique dans lesquels il y a plusieurs millions de variables).

2 Optimisation sans contrainte

Dans cette section, nous nous intéressons à la minimisation d'une fonction $f : \mathbb{R}^n \to \mathbb{R}$ sur $\mathbb{R}^n$ tout entier (donc avec n variables non soumises à des contraintes), ce que nous écrivons

$$\min_{x \in \mathbb{R}^n} f(x). \tag{2.1}$$

La fonction est supposée régulière (i.e., différentiable). Son *gradient* et son *hessien* en un itéré x_k sont respectivement notés

$$g_k = g(x_k) = \nabla f(x_k) \quad \text{et} \quad H_k = \nabla^2 f(x_k).$$

Ceux-ci sont supposés calculés pour le *produit scalaire euclidien*, si bien que leurs composantes sont respectivement les dérivées partielles premières et secondes de f en x_k. Nous exposons les techniques de base de l'optimisation numérique, techniques qui sont aussi utilisées en optimisation avec contraintes.

2.1 Techniques de globalisation

Les deux techniques que nous décrivons ci-après sont utilisées pour forcer la convergence des itérés, même si le premier d'entre eux est éloigné d'une solution. La recherche linéaire (section 2.1.1) est facile à adapter à divers problèmes et contextes algorithmiques, en particulier aux problèmes de grande taille, mais elle est un peu moins robuste que la globalisation par régions de confiance (section 2.1.2).

2.1.1 Recherche linéaire

On appelle *direction de descente* de f en x, une direction $d \in \mathbb{R}^n$ telle que $f'(x) \cdot d < 0$. Par définition même de cette dérivée directionnelle, $f(x + \alpha d) < f(x)$ pour tout $\alpha > 0$ petit, si bien que f décroît en x dans la direction d. Les *méthodes à directions de descente* génèrent une suite $\{x_k\}$ par la récurrence

$$x_{k+1} = x_k + \alpha_k d_k, \quad \text{pour tout } k \geq 1,$$

où d_k est une direction de descente de f en x_k et $\alpha_k > 0$ est un *pas* le long de d_k. La convergence de cet algorithme (nous serons plus précis ci-dessous sur ce que l'on entend par là) est tributaire du choix des directions de descente et de la règle de détermination du pas α_k.

Choix de la direction de descente

La direction de descente choisie donne son nom à l'algorithme. En voici quelques unes (on suppose toujours que $g_k \neq 0$).

- Direction du *gradient* ou de la *plus profonde descente* : $d_k = -g_k$. Assure la convergence théorique, mais n'est pas bonne en pratique (n'utilise que les dérivées premières); à déconseiller donc.
- Direction du *gradient conjugué* :

  $$d_k = \begin{cases} -g_k & \text{si } k = 1 \\ -g_k + \beta_k d_{k-1} & \text{si } k \geq 2. \end{cases}$$

  C'est une direction de descente si la recherche linéaire est exacte (voir plus loin). Le scalaire β_k peut prendre diverses formes, toutes équivalentes pour une fonction quadratique strictement convexe et recherche linéaire exacte (la seule classe de problèmes pour lesquels l'algorithme est recommandé). Les formules de Fletcher-Reeves (FR) et de Polak-Ribière (PR) s'écrivent

  $$\beta_k^{\text{FR}} = \frac{\|g_k\|_2^2}{\|g_{k-1}\|_2^2} \quad \text{et} \quad \beta_k^{\text{PR}} = \frac{g_k^\top (g_k - g_{k-1})}{\|g_{k-1}\|_2^2}.$$

  Cette méthode n'est plus guère utilisée en optimisation non linéaire, car elle est en général moins rapide que l'algorithme ℓ-BFGS (section 2.2.2), pour un encombrement mémoire à peu près identique.
- Direction de *Newton* : $d_k = -H_k^{-1} g_k$. C'est une direction de descente si $H_k \succ 0$, donc dans le voisinage d'une solution vérifiant les conditions d'optimalité suffisantes du second ordre (théorème 1.6). Cet algorithme important sera décrit en détail à la section 2.2.1.
- Direction de *quasi-Newton* : $d_k = -M_k^{-1} g_k$, dans laquelle M_k est une approximation de H_k, déterminée par une technique bien particulière que nous verrons à la section 2.2.2. C'est une direction de descente si $M_k \succ 0$.
- Direction de *Gauss-Newton*. Elle est utilisée pour résoudre des *problèmes de moindres-carrés* non linéaires, dans lesquels f est définie par

  $$f(x) = \frac{1}{2} \|r(x)\|_2^2, \tag{2.2}$$

  où la fonction $r : \mathbb{R}^n \to \mathbb{R}^m$ est appelée *résidu*. Si on note $J_k = r'(x_k)$ la *jacobienne* de r en x_k, le gradient s'écrit $g_k = J_k^\top r(x_k)$ et la direction de Gauss-Newton d_k est calculée en résolvant le système linéaire (il a toujours une solution)

  $$(J_k^\top J_k) d_k = -g_k.$$

C'est toujours une direction de descente de f en x_k.

Si $g_k = 0$, les directions précédentes s'annulent. Dès lors, aucun des algorithmes associés ne sera capable de trouver mieux qu'un point stationnaire de f (comparez avec les méthodes à régions de confiance).

Détermination du pas

L'autre aspect des méthodes à direction de descente est la détermination du pas $\alpha_k > 0$; c'est ce que l'on appelle la *recherche linéaire*. Il y a deux écueils à éviter : prendre des petits pas constants et faire de la recherche linéaire exacte. Nous allons expliquer ce qui ne convient pas dans ces deux techniques (encore trop souvent rencontrées) et décrirons ensuite deux bonnes règles de recherche linéaire.

Prendre α_k constant et petit n'est pas conseillé : d'une part, il est difficile de déterminer la valeur de ces petits pas et, d'autre part, cette technique ralentit fortement la convergence. Dans la *recherche linéaire exacte*, on détermine α_k de manière à minimiser la fonction d'une variable réelle

$$\alpha \mapsto h_k(\alpha) := f(x_k + \alpha d_k)$$

sur $\mathbb{R}_+ := \{\alpha \in \mathbb{R} \colon \alpha \geq 0\}$. Cette technique est parfaite si h_k a un minimum qui se calcule rapidement (elle est utilisée par l'algorithme du gradient conjugué qui suppose que f est quadratique strictement convexe). Mais ces conditions ne sont en général pas remplies en optimisation non linéaire; elle est alors à éviter. De toute façon, trouver un pas minimisant exactement h_k ne peut être réalisé avec une précision infinie, si bien qu'un test d'arrêt devrait être introduit. Autant alors avoir un test d'acceptation du pas qui soit assez permissif pour que ce pas soit trouvé rapidement et qui contribue efficacement à la convergence de l'algorithme qui utilise le pas ainsi déterminé. Ces règles de recherche linéaire sont dites *inexactes* et sont recommandées, malgré leur dénomination peu engageante. Voici deux de ces règles, les plus souvent utilisées.

Dans la *règle d'Armijo*, le pas α_k est pris comme le *premier* élément d'une suite $\{\alpha_k^i\}_{i\geq 1}$ qui vérifie la *condition de décroissance suffisante* suivante (ω_1 est une constante prise dans $]0,1[$, typiquement $\omega_1 = 10^{-4}$) :

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \omega_1 \alpha_k g_k^\top d_k. \tag{2.3}$$

On se donne le premier élément $\alpha_k^1 > 0$ de la suite et les éléments suivants sont en général déterminés par interpolation et projection de manière à vérifier

$$\alpha_k^{i+1} \in [\tau\alpha_k^i, (1-\tau)\alpha_k^i], \quad \text{pour } i \geq 1,$$

où τ est une constante fixée dans $]0, \frac{1}{2}]$. Ainsi, la suite $\{\alpha_k^i\}_{i\geq 1}$ est décroissante et on parle de *rebroussement*. Cette règle est souvent utilisée dans les algorithmes newtoniens, avec $\alpha_k^1 = 1$.

Dans la *règle de Wolfe*, le pas $\alpha_k > 0$ est choisi de manière à vérifier (2.3) et

$$g(x_k + \alpha_k d_k)^\top d_k \geq \omega_2 g_k^\top d_k. \tag{2.4}$$

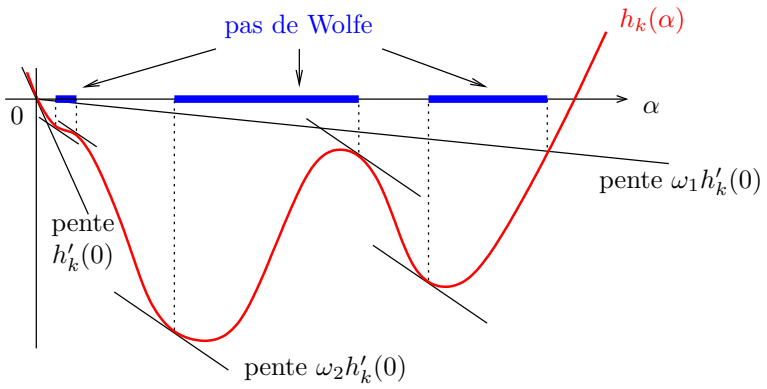


Figure 2.1: Règle de Wolfe.

La figure 2.1 donne les pas de Wolfe pour une fonction h_k légèrement mouvementée; on observera que $h_k'(0) = f'(x_k) \cdot d_k < 0$ et que l'inégalité (2.4) s'écrit aussi $h_k'(\alpha_k) \geq \omega_2 h_k'(0)$. On montre qu'un pas de Wolfe existe si f est bornée inférieurement et si $0 < \omega_1 < \omega_2 < 1$ (typiquement $\omega_2 = 0.99$). Cette règle est souvent utilisée dans les algorithmes quasi-newtoniens, car elle apporte la *propriété de monotonie*

$$(g_{k+1} - g_k)^\top (x_{k+1} - x_k) > 0, \tag{2.5}$$

qui est utile pour maintenir la définie positivité des matrices générées M_k (voir la section 2.2.2).

La contribution de la règle de Wolfe à la convergence s'exprime par le résultat suivant (il y a un résultat similaire pour la règle d'Armijo). On y a noté

$$\theta_k := \arccos \frac{-g_k^\top d_k}{\|g_k\| \, \|d_k\|} \tag{2.6}$$

l'angle entre la direction de descente d_k et $-g_k$ et

$$\mathcal{N}_1 := \{x \in \mathbb{R}^n : f(x) \leq f(x_1)\} \tag{2.7}$$

l'*ensemble de sous-niveau* déterminé par le premier itéré x_1.

Théorème 2.1 (condition de Zoutendijk) *Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction différentiable sur un voisinage de $\mathcal{N}_1$ et $C^{1,1}$ sur $\mathcal{N}_1$. On considère un algorithme à directions de descente d_k, qui génère une suite $\{x_k\}$ en utilisant la règle de Wolfe (2.3)-(2.4) et tel que $\{f(x_k)\}$ soit bornée inférieurement. Alors*

$$\sum_{k \geq 1} \|g_k\|^2 \cos^2 \theta_k < \infty. \tag{2.8}$$

La relation (2.8) porte le nom de *condition de Zoutendijk*. On peut déduire de ce résultat que la règle de recherche linéaire de Wolfe permet de faire converger l'algorithme du gradient ($\theta_k = 0$) ou tout algorithme qui maintient le $\cos\theta_k$ uniformément positif. On obtient en effet alors $g_k \to 0$ qui, par définition, est une expression de la *convergence de l'algorithme* (la convergence de la suite $\{x_k\}$ ne peut en général pas se démontrer sans hypothèse supplémentaire sur f).

2.1.2 Régions de confiance

Les méthodes à directions de descente peuvent ne pas converger lorsque l'angle θ_k défini en (2.6) tend vers $\pi/2$ (voir le théorème 2.1). On pourrait évidemment redresser la direction d_k lorsque le $\cos\theta_k$ passe en-dessous d'un seuil strictement positif, fixé de manière arbitraire, mais avec le risque de détériorer la performance apportée par la qualité supposée des directions d_k. Les méthodes à régions de confiance ont cette propriété de redressement de la direction, mais en suivant un principe général plus attrayant que l'adaptation à un résultat de convergence. Voici ce principe.

Le calcul des directions de descente décrit à la section 2.1.1 est fondé sur un modèle de la variation de f autour de x_k qui se limite aux termes quadratiques : $f(x_k + s) \simeq f(x_k) + \psi_k(s)$, où

$$\psi_k(s) = g_k^\top s + \frac{1}{2} s^\top M_k s \tag{2.9}$$

et la matrice symétrique M_k est une approximation du hessien H_k. Ce modèle est précis dans un voisinage de x_k, si bien qu'il semble raisonnable de le minimiser dans une boule de centre x_k et rayon $\Delta_k > 0$:

$$\left\{ \begin{array}{l} \min_s \ \psi_k(s) \\ \|s\|_2 \leq \Delta_k. \end{array} \right. \tag{2.10}$$

C'est ce que font les méthodes à régions de confiance : Δ_k est appelé le *rayon de confiance* et $\{s : \|s\|_2 \leq \Delta_k\}$ la *région de confiance*. Si la solution s_k de (2.10) est acceptable, on prend comme nouvel itéré $x_{k+1} = x_k + s_k$.

Il reste à préciser comment apprécier la qualité du déplacement s_k et comment on détermine le rayon de confiance. On considérera que s_k convient, si ce déplacement fait décroître f suffisamment, non pas en comparant cette décroissance à celle de l'approximation linéaire de f comme dans la règle d'Armijo (2.3), mais en la comparant à la décroissance du modèle ψ_k. Si $\psi_k(s_k) < \psi_k(0)$, on peut introduire la *concordance*

$$\rho_k = \frac{f(x_k) - f(x_{k+1})}{\psi_k(0) - \psi_k(s_k)},$$

qui est le ratio entre la décroissance de f et celle du modèle. Il mesure donc si, en $x_k + s_k$, modèle et fonction sont proches l'un de l'autre. Si c'est le cas ($\rho_k \simeq 1$), le modèle ψ_k est de qualité et comme s_k le minimise, il est raisonnable d'accepter ce déplacement. Il est aussi normal d'accepter s_k si f décroît davantage que ce qui est prédit par le modèle ($\rho_k \geq 1$). Par ailleurs, si la concordance n'est pas assez bonne, et a fortiori si f croît, on réduit Δ_k et on résout à nouveau le problème (2.10). On finit par trouver un rayon de confiance acceptable, car on montre que si $\Delta_k \to 0$, alors $\rho_k \to 1$. Le rayon testé en

premier à l'itération courante est celui obtenu à la fin de l'itération précédente; donc si la concordance est bonne on pourra accroître le rayon de confiance en vue de l'itération suivante. On peut à présent comprendre le mécanisme des régions de confiance : on se donne

- des seuils de concordance : $0 < \omega_1 < \omega_2 < \omega_3 < 1$,
- des facteurs de mise à jour de Δ_k : $0 < \tau_1 \leq \tau_2 < 1 < \tau_3$

(des valeurs typiques sont $\omega_1 = 10^{-2}$, $\omega_2 = 1/4$, $\omega_3 = 3/4$, $\tau_1 = 1/4$, $\tau_2 = 1/2$ et $\tau_3 = 2$), on réalise les étapes suivantes

1. **tant que** $\rho_k < \omega_1$, prendre $\Delta_k := \tau_1 \Delta_k$ et résoudre à nouveau (2.10)
2. **si** (2.10) n'a été résolu qu'une seule fois
   alors on calcule le rayon initial de l'itération suivante comme suit

   si $\rho_k < \omega_2$
   alors $\Delta_{k+1} := \tau_2 \Delta_k$
   sinon si $\rho_k > \omega_3$ **et** $\|s_k\|_2 = \Delta_k$
   alors $\Delta_{k+1} := \tau_3 \Delta_k$
   sinon $\Delta_{k+1} := \Delta_k$

   sinon $\Delta_{k+1} := \Delta_k$.

L'étape 1 est très proche de ce que fait la recherche linéaire d'Armijo, avec une différence essentielle cependant : comme annoncé, à chaque réduction du rayon de confiance, le déplacement s_k n'est pas colinéaire au précédent, mais s'aligne d'autant mieux sur $-g_k$ que Δ_k est petit (voir le théorème 2.2 ci-dessous). On peut en effet montrer que, hormis le *cas difficile* défini plus loin, la solution de (2.10) décrit une courbe paramétrée par le rayon Δ_k, qui est tangente en x_k à $-g_k$ (c'est la courbe $\lambda \mapsto s_\lambda$ à gauche dans la figure 2.2). L'étape 2 de prédiction du rayon de confiance suivant est nouvelle par rapport à la recherche linéaire (mais non essentielle) et trouve sa justification dans le fait qu'une région de confiance d'un modèle ne doit pas trop varier d'un itéré à l'autre.

Il y a autant de modèles ψ_k que de directions de descente. À la direction du gradient correspond le modèle $\psi_k(s) = g_k^\top s$, qui n'a d'ailleurs pas plus d'intérêt que l'algorithme du gradient (les déplacements calculés sont colinéaires au gradient, donc conduisent à une convergence lente). À la direction de Newton correspond le modèle $\psi_k(s) = g_k^\top s + \frac{1}{2} s^\top H_k s$ et les algorithmes de quasi-Newton s'obtiennent en prenant pour M_k dans (2.9) une approximation particulière du hessien H_k. Enfin, l'algorithme de Gauss-Newton s'obtient en prenant pour modèle $\psi_k(s) = \frac{1}{2}\|r(x_k) + r'(x_k)s\|_2^2$, où r est le résidu intervenant dans la définition des problèmes de moindres-carrés (2.2).

Le problème (2.10) n'est pas nécessairement convexe, car M_k peut avoir des valeurs propres négatives. Et pourtant, il a des conditions nécessaires et suffisantes d'optimalité.

Théorème 2.2 *Supposons que* $\Delta_k > 0$. *Alors,* s_k *est solution de* (2.10) *si et seule-*

ment si il existe $\lambda_k \in \mathbb{R}$ tel que l'on ait

$$\begin{cases} (a) & (M_k + \lambda_k I)s_k = -g_k \\ (b) & \|s_k\|_2 \leq \Delta_k \\ (c) & \lambda_k \geq 0 \\ (d) & \lambda_k(\Delta_k - \|s_k\|_2) = 0 \\ (e) & (M_k + \lambda_k I) \succcurlyeq 0. \end{cases} \tag{2.11}$$

On retrouve les conditions de KKT dans (a)-(d), tandis que (e) contient une partie des conditions nécessaires du second ordre (CN2).

La résolution de (2.10) n'est pas aisée, ce qui peut expliquer son introduction tardive. Il s'agit d'un problème non linéaire qui ne peut pas se résoudre exactement en un nombre fini d'étapes. Heureusement, si l'on se satisfait d'une convergence vers des points stationnaires de f, on peut le résoudre de manière très approximative. Dans ce cas, c'est le *point de Cauchy* qui joue le rôle-clé. Il s'agit du minimiseur $s_k^{\mathrm{C}} = -\alpha_k^{\mathrm{C}} g_k$ de ψ_k dans la région de confiance et le long de la direction $-g_k$. On aura convergence vers la stationnarité si s_k fait aussi bien que s_k^{C}, dans le sens où s_k vérifie

$$\psi_k(s_k) \leq \beta_1 \psi_k(s_k^{\mathrm{C}}) \quad \text{et} \quad \|s_k\|_2 \leq \beta_2 \Delta_k, \tag{2.12}$$

où β_1 et β_2 sont deux constantes strictement positives (en général, on prend $0 < \beta_1 \leq 1$ et $1 \leq \beta_2$). Ces conditions sont réalisées par diverses méthodes de résolution de (2.10). Citons l'*algorithme dogleg* de Powell (figure 2.2 à gauche), qui suppose que $M_k \succcurlyeq 0$ et

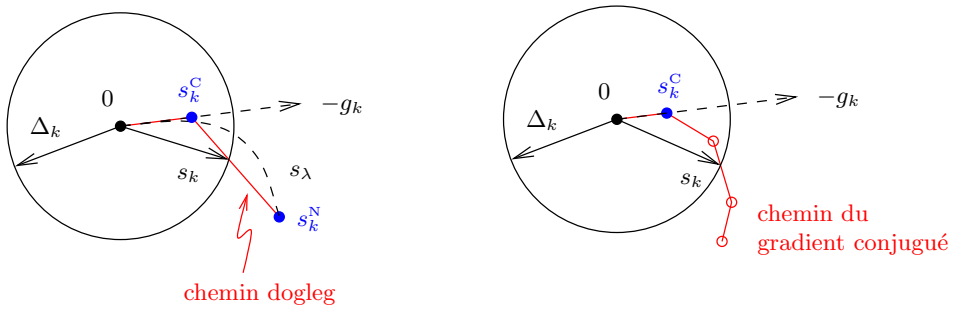


Figure 2.2: Résolutions approchées du sous-problème quadratique (2.10)

que ψ_k a un minimum, disons s_k^{N} qui vérifie donc $M_k s_k^{\mathrm{N}} = -g_k$ (on peut montrer que $\|s_k^{\mathrm{C}}\|_2 \leq \|s_k^{\mathrm{N}}\|_2$). Dans cet algorithme, s_k est la combinaison convexe entre s_k^{C} et s_k^{N} qui a sa norme la plus proche de Δ_k (en particulier, $s_k = s_k^{\mathrm{N}}$ si $\|s_k^{\mathrm{N}}\|_2 \leq \Delta_k$, et $\|s_k\|_2 = \Delta_k$ si $\|s_k^{\mathrm{N}}\|_2 > \Delta_k$). Citons également l'*algorithme du gradient conjugué tronqué* de Steihaug (très schématisé à droite dans la figure 2.2, car dans $\mathbb{R}^2$ il ne devrait pas y avoir plus de 2 itérations) dans lequel on minimise ψ_k par le gradient conjugué jusqu'à ce l'on traverse la frontière de la région de confiance (on montre que les itérés s_k^i croissent en norme au cours des itérations) ou que l'on rencontre une direction à courbure négative

(auquel cas on la suit jusqu'à la frontière de la région de confiance). Voici le résultat de convergence que l'on peut obtenir avec une résolution approchée de (2.10). On notera qu'il n'y a pas d'hypothèse sur l'inversibilité de M_k, ce qui est un atout des régions de confiance par rapport à la recherche linéaire, lequel contribue à leur robustesse.

Théorème 2.3 (convergence globale avec résolution approchée des sous-problèmes quadratiques) *Supposons que f soit bornée inférieurement avec un gradient uniformément continu dans un voisinage de $\mathcal{N}_1$, que $\{M_k\}$ soit bornée et que la solution approchée s_k du sous-problème (2.10) vérifie (2.12). Alors soit l'algorithme s'arrête en un itéré x_{k_0} tel que $g_{k_0} = 0$, soit $g_k \to 0$.*

Si l'on veut éviter les points stationnaires qui ne sont pas des minima locaux, il faudra en payer le prix en prenant $M_k = H_k$ (calcul des dérivées secondes) et en résolvant (2.10) avec davantage de précision. Le système (2.11), qui correspond à une résolution exacte de (2.10), révèle cette propriété qu'ont les itérés de s'échapper d'un point stationnaire x_k ($g_k = 0$) qui n'est pas un minimum local ($H_k \not\succeq 0$) : (e) implique alors que $\lambda_k > 0$, donc $s_k \neq 0$ par (d), si bien que $x_{k+1} = x_k + s_k \neq x_k$. Une résolution exacte n'est cependant pas indispensable, il suffit que la solution s_k vérifie

$$\psi_k(s_k) \leq \beta_1 \min\{\psi_k(s) : \|s\|_2 \leq \Delta_k\} \quad \text{et} \quad \|s_k\|_2 \leq \beta_2 \Delta_k, \tag{2.13}$$

où β_1 et β_2 sont des constantes vérifiant $0 < \beta_1 < 1$ et $0 < \beta_2$. Les algorithmes permettant de réaliser ces conditions ont une complexité qui va au-delà de ce que permet cette synthèse. Signalons l'*algorithme de Moré-Sorensen*, l'un des premiers, qui consiste dans le cas le plus simple à itérer sur $\lambda > \max(0, -\lambda_{\min}(M_k))$ de telle sorte que la solution $s = -(M_k + \lambda I)^{-1} g_k$ soit en norme égal à Δ_k. On a noté $\lambda_{\min}(M_k)$ la plus petite valeur propre de M_k. L'algorithme de Moré-Sorensen peut aussi traiter le *cas difficile*, qui se produit lorsque $\lambda_{\min}(M_k) < 0$ et que la projection de g_k sur l'espace propre associé à $\lambda_{\min}(M_k)$ est nulle (dans ce cas, il y a un problème de valeur propre à résoudre). On obtient alors le résultat de convergence suivant.

Théorème 2.4 (convergence globale avec résolution fine des sous-problèmes quadratiques) *Supposons que f soit bornée inférieurement, de classe C^2 dans un voisinage de $\mathcal{N}_1$ et que son hessien $\nabla^2 f$ soit borné sur $\mathcal{N}_1$. On suppose que l'on prend $M_k = \nabla^2 f(x_k)$ dans la définition du modèle ψ_k et que la solution approchée s_k de (2.10) vérifie (2.13). Alors soit l'algorithme s'arrête en un itéré x_{k_0} vérifiant $g_{k_0} = 0$ et $\nabla^2 f(x_{k_0})$ est semi-définie positive, soit une suite $\{x_k\}$ est générée, $g_k \to 0$, et, si $\{x_k\}$ est bornée, $\limsup \lambda_{\min}(\nabla^2 f(x_k)) \geq 0$.*

2.2 Méthodes rapidement convergentes

Les algorithmes détiennent leur efficacité locale d'une utilisation convenable des dérivées de la fonction à minimiser. Dans l'algorithme de Newton, ce sont les dérivées secondes

qui sont utilisées. Les algorithmes de quasi-Newton construisent une approximation du hessien à partir de la variation des dérivées premières d'une itération à l'autre. Les deux algorithmes ont des versions (Newton inexact, Newton tronqué et ℓ-BFGS) adaptées aux problèmes de grande taille.

2.2.1 Newton

Résolution d'équations non linéaires

On donne aujourd'hui le nom de *méthode de Newton* à toute approche algorithmique procédant par linéarisation des fonctions définissant le système dont on cherche une solution. Même si l'on se cantonne à l'optimisation différentiable sans contrainte, il est intéressant de commencer par présenter l'algorithme lorsque l'on cherche un *zéro* d'une fonction non linéaire $F : \mathbb{R}^n \to \mathbb{R}^n$, c'est-à-dire un point $x \in \mathbb{R}^n$ tel que

$$F(x) = 0. \tag{2.14}$$

En effet, l'équation d'optimalité de (2.1), $\nabla f(x) = 0$, montre que trouver un point stationnaire est un cas particulier de ce problème (la jacobienne de $\nabla f(x)$ est une matrice symétrique, ce qui n'est pas nécessairement le cas de $F'(x)$). Par ailleurs, le problème (2.14) est un cas particulier de l'optimisation sous contrainte d'égalité, puisqu'on peut le voir comme celui de minimiser une fonction constante sous la contrainte (2.14).

Pour résoudre (2.14), l'algorithme de Newton construit une suite $\{x_k\}$ par la récurrence

$$x_{k+1} := x_k + d_k, \quad \text{où } d_k \text{ est solution de } F'(x_k)d_k = -F(x_k). \tag{2.15}$$

Il s'agit donc de résoudre à chaque itération le système linéaire obtenu par linéarisation de F en x_k, que l'on appelle l'*équation de Newton*. On dira que l'algorithme est *bien défini*, si ce système linéaire a une solution à chaque itération. Cet algorithme converge localement quadratiquement.

Théorème 2.5 (convergence locale de l'algorithme de Newton) *On suppose que F a un zéro x_*, que F est $C^{1,1}$ dans un voisinage x_* et que $F'(x_*)$ est inversible. Alors l'algorithme de Newton* (2.15)*, démarrant en un point x_1 proche de x_*, est bien défini et génère une suite $\{x_k\}$ qui converge quadratiquement vers x_*.*

Si ce résultat montre l'atout principal de l'algorithme de Newton, sa convergence quadratique locale, celui-ci n'est pas sans défaut. Mentionnons : l'absence de convergence si le premier itéré n'est pas suffisamment proche d'un zéro, la nécessité de calculer la jacobienne $F'(x_k)$, le coût de la résolution du système linéaire et l'échec de l'algorithme lorsque la jacobienne est singulière. De nombreux travaux ont apporté des remèdes à ces défauts. Quelques uns sont décrits ci-dessous.

La *globalisation* d'un algorithme, c'est-à-dire le forçage de sa convergence à partir d'un premier itéré arbitraire, se fait souvent par la minimisation d'une fonction auxiliaire, dite *de mérite*, qui a pour minima les solutions du problème. Pour (2.14), une

fonction de mérite naturelle est la *fonction de moindres-carrés* $f : \mathbb{R}^n \to \mathbb{R}$, définie par

$$f(x) = \frac{1}{2} \|F(x)\|_2^2.$$

Si cette approche passant par l'optimisation est souvent fructueuse, elle peut aussi échouer, parce que f peut avoir des minima locaux qui ne sont pas des zéros de F. On pourra s'en consoler en sachant qu'on ne connaît pas aujourd'hui d'algorithme qui garantisse de trouver un zéro d'une fonction quel que soit le point initial.

Dans la globalisation de l'algorithme de Newton (2.15) par *recherche linéaire*, on calcule une direction d_k telle que

$$\|F(x_k) + F'(x_k)d_k\|_2 \leq \eta_k \|F(x_k)\|_2, \tag{2.16}$$

où $0 \leq \eta_k \leq \eta < 1$ (η est une constante). Cette condition sera vérifiée par la solution du système linéaire dans (2.15), mais elle permet aussi de résoudre celui-ci de manière approchée, par exemple en utilisant un solveur itératif tel que GMRES ou QMR. On parle alors de *méthode de Newton inexacte*. Il est aisé de voir que, si $F(x_k) \neq 0$, d_k est une *direction de descente* de f en x_k, ce qui semble providentiel. On a en effet $\nabla f(x_k) = F'(x_k)^\top F(x_k)$, puis en utilisant l'inégalité de Cauchy-Schwarz :

$$\nabla f(x_k)^\top d_k = F(x_k)^\top \big(F(x_k) + F'(x_k)d_k\big) - \|F(x_k)\|_2^2 \leq -(1-\eta_k)\|F(x_k)\|_2^2 < 0.$$

Dès lors, on peut faire de la recherche linéaire le long de cette direction : on prend $x_{k+1} = x_k + \alpha_k d_k$ avec un pas $\alpha_k > 0$ déterminé par la règle d'Armijo (2.3) avec pas initial $\alpha_k^1 = 1$. La convergence globale de l'approche que l'on vient de décrire est assurée dans les conditions données par le résultat suivant.

Théorème 2.6 (convergence globale de l'algorithme de Newton avec recherche linéaire) *Considérons l'algorithme de Newton inexact décrit ci-dessus, avec une recherche linéaire vérifiant la condition de Zoutendijk* (2.8)*, et supposons qu'il génère une suite* $\{x_k\}$ *telle que le conditionnement* $\kappa_2(F'(x_k))$ *soit borné. Alors*
1) $\nabla f(x_k) \to 0$,
2) *si, de plus, la suite* $\{F'(x_k)^{-1}\}$ *est bornée, alors* $F(x_k) \to 0$.

Par ailleurs on peut montrer un résultat de convergence locale de cet algorithme avec pas unité, qui est linéaire si η est assez petit, superlinéaire si $\eta_k \to 0$ et quadratique si $\eta_k = O(\|F(x_k)\|)$. Le théorème 2.6 nous apprend que cet algorithme simple peut souvent donner satisfaction. Les hypothèses qui assurent sa convergence révèlent aussi sa faiblesse : les itérés peuvent être attirés par les points x en lesquels la jacobienne $F'(x)$ est singulière. On connaît des contre-exemples très simples en dimension 2. Cet éventuel défaut d'inversibilité est mieux pris en compte par la technique des régions de confiance, que nous allons maintenant adapter à la résolution de (2.14).

Dans la globalisation de l'algorithme de Newton (2.15) par *régions de confiance*, on calcule le déplacement s_k en résolvant le problème de moindres-carrés linéaire

$$\begin{cases} \min_s \ \frac{1}{2}\|F(x_k) + F'(x_k)s\|_2^2 \\ \|s\|_2 \leq \Delta_k, \end{cases} \tag{2.17}$$

dont le critère est un modèle quadratique de la fonction de moindres-carrés non linéaire f, ne faisant intervenir que la dérivée première de F. Après adaptation du rayon de confiance Δ_k de manière à faire décroître f (voir la section 2.1.2), on prend $x_{k+1} = x_k + s_k$. La résolution approchée du problème quadratique (2.17) par dogleg choisit comme déplacement la combinaison convexe entre le point de Cauchy (qui est ici de la forme $-\alpha_k^{\mathrm{C}} F'(x_k)^\top F(x_k)$) et le pas de Newton $-F'(x_k)^{-1}F(x_k)$ (si celui-ci existe), qui est en norme la plus proche de Δ_k. Par ailleurs, la résolution approchée itérative de (2.17) par gradient conjugué tronqué est pénalisée par le mauvais conditionnement potentiel du hessien du critère quadratique, si bien que l'on préfère en général utiliser un solveur itératif du système de Newton dans (2.15) tel que GMRES ou QMR. Nous donnons ci-dessous un résultat de convergence globale typique que l'on peut obtenir avec la globalisation de l'algorithme de Newton par régions de confiance. Ce résultat peut varier légèrement d'une implémentation à l'autre, mais le point important, qui fait qu'il diffère du théorème 2.6, est le suivant : on n'a plus besoin de l'hypothèse sur l'inversibilité de $F'(x_k)$. C'est ce qui contribue à la robustesse de cette approche. Dans ce cas, si l'on ne peut pas garantir la convergence vers un zéro de F, on peut quand même assurer celle vers un point stationnaire de la fonction de mérite f.

Théorème 2.7 (convergence globale de l'algorithme de Newton avec régions de confiance) *Supposons que, dans un voisinage de $\mathcal{N}_1$, F soit $C^{1,1}$ et F' soit bornée. Alors les algorithmes de Newton avec régions de confiance décrits ci-dessus, démarrant en x_1, génèrent des suites $\{x_k\}$ telles que $\nabla f(x_k) \to 0$.*

Résolution de problèmes d'optimisation

Considérons à présent le problème d'optimisation sans contrainte (2.1). Ses points stationnaires sont les solutions de l'équation non linéaire $\nabla f(x) = 0$, que l'on peut résoudre par l'algorithme de Newton (2.15) dans lequel $F = \nabla f$. Cette observation conduit à l'algorithme de Newton suivant (rappelons que l'on note $g_k := \nabla f(x_k)$ et $H_k := \nabla^2 f(x_k)$ le gradient et le hessien de f en x_k) :

$$x_{k+1} := x_k + d_k, \quad \text{où } d_k \text{ est solution de } H_k d_k = -g_k. \tag{2.18}$$

Le système linéaire dans (2.18) s'appelle l'*équation de Newton*. Le théorème 2.5 s'adapte directement à cette situation.

Théorème 2.8 (convergence locale de l'algorithme de Newton en optimisation) *On suppose que f a un point stationnaire x_*, que f est $C^{2,1}$ dans un voisinage x_* et que $\nabla^2 f(x_*)$ est inversible. Alors l'algorithme de Newton* (2.15)*, démarrant en un point x_1 proche de x_*, est bien défini et génère une suite $\{x_k\}$ qui converge quadratiquement vers x_*.*

L'algorithme (2.18) a le même intérêt (sa convergence quadratique) et les mêmes inconvénients (convergence locale uniquement, dérivées secondes à calculer, coût de résolution du système linéaire et échec de l'algorithme lorsque le hessien est singulier) que son parent pour la résolution de systèmes non linéaires. Du point de vue de l'optimisation, il a un défaut supplémentaire, celui de ne pas faire de distinction entre minima, maxima ou points stationnaires quelconques. Ainsi, si le premier itéré est proche d'un maximum local avec un hessien défini négatif, l'algorithme (2.18) génère une suite convergeant vers ce maximum, ce qui est fâcheux lorsque l'on cherche à minimiser une fonction.

Les remèdes ressemblent à ceux introduits pour la résolution de systèmes non linéaires, avec une différence cependant : nul besoin cette fois d'introduire une fonction de mérite, puisque le critère f lui-même peut jouer ce rôle. Il y a une difficulté supplémentaire : la direction de Newton d_k n'est plus nécessairement une direction de descente de f, alors qu'elle est toujours une direction de descente de $x \mapsto \|\nabla f(x)\|_2^2$. Ceci n'implique nullement qu'il faut utiliser cette dernière comme fonction de mérite, pour au moins trois raisons : elle admet n'importe quel point stationnaire comme minimum global (ce qui exclurait toute chance de trouver un minimum local de f), sa dérivée fait intervenir le hessien de f (elle est donc coûteuse) et elle est moins bien conditionnée que f (son hessien contient le carré du hessien de f).

Lorsque $H_k \succ 0$, le problème quadratique obtenu en prenant un développement limité de f (on a écarté le terme constant $f(x_k)$),

$$\min_{d \in \mathbb{R}^n} \; g_k^\top d + \frac{1}{2} d^\top H_k d, \tag{2.19}$$

a une solution unique qui n'est autre que la direction de Newton définie en (2.18). Ce problème porte le nom de *problème quadratique osculateur* (PQO) et joue un rôle important dans la globalisation de l'algorithme de Newton.

Dans la globalisation de l'algorithme de Newton (2.18) par *recherche linéaire*, on doit déterminer une direction de descente de f en x_k, sachant que la direction de Newton, si elle existe, peut ne pas vérifier cette propriété. Il y a plusieurs manières de procéder.

- Une première possibilité consiste à déterminer d_k en résolvant le système linéaire $M_k d_k = -g_k$, dans lequel $M_k \succ 0$ est obtenue en modifiant H_k. C'est ce l'on appelle l'*algorithme de Newton modifié*. L'approche classique consiste à modifier H_k au cours de sa factorisation de Cholesky (celle-ci échoue si $H_k \not\succ 0$). Cette *factorisation de Cholesky modifiée* conduit ainsi à $M_k = H_k + D_k$, où $D_k \succcurlyeq 0$ est diagonale.
- Dans l'*algorithme de Newton tronqué*, on résout le PQO (2.19) par gradient conjugué (GC), tant que celui-ci est bien défini, et tant qu'une condition d'arrêt n'est pas satisfaite (il faut cependant faire au moins une itération pour que l'algorithme de Newton tronqué converge). L'algorithme du GC ne sera plus bien défini s'il rencontre une *direction à courbure négative*, c'est-à-dire une direction interne $d_{k,i}$ telle que $d_{k,i}^\top H_k d_{k,i} \leq 0$. Un test d'arrêt souvent utilisé porte sur le gradient du

critère de (2.19) :

$$\|g_k + H_k d_k\|_2 \leq \eta_k \|g_k\|_2,$$

avec des conditions sur η_k identiques à celles utilisées dans (2.16). Un autre test d'arrêt peut porter sur l'angle entre $-g_k$ et la direction d_k, qui croît au cours des itérations du GC, avec un seuil qui peut être mis à jour au cours des itérations externes k.

La globalisation de l'algorithme de Newton (2.18) par *régions de confiance* suit la démarche qui a été exposée à la section 2.1.2, en utilisant le critère du PQO (2.19) comme modèle local de f.

2.2.2 Quasi-Newton

Comme leur nom l'indique, les algorithmes de quasi-Newton imitent les algorithmes de Newton, mais par une technique bien particulière (tous les algorithmes qui ressemblent à celui de Newton ne méritent donc pas cette appellation). Le but est d'éviter le calcul de la jacobienne $F'(x_k)$ dans la résolution d'équations non linéaires ou le hessien H_k de f en optimisation. Les algorithmes de quasi-Newton prennent donc en charge la génération de la suite des itérés $\{x_k\}$, mais aussi celle de la suite des matrices M_k approchant $F'(x_k)$ ou H_k. La première suite est générée par la récurrence

$$x_{k+1} = x_k + s_k,$$

où $s_k = \alpha_k d_k$ dans les algorithmes avec recherche linéaire ($\alpha_k > 0$ est le pas et d_k vérifie $M_k d_k = -F(x_k)$ ou $M_k d_k = -g_k$ selon le problème) et s_k est solution d'un problème d'optimisation quadratique avec une borne sur la norme du déplacement dans les algorithmes avec régions de confiance (voir (2.17) et (2.9)-(2.10)). Dans la suite, nous nous intéressons au calcul de la matrice M_{k+1} qui sera utilisée à l'itération suivante; on suppose donc que x_k, x_{k+1} et M_k sont connus.

Résolution d'équations non linéaires

Si l'on veut que M_{k+1} approche la jacobienne $F'(x_{k+1})$ sans calculer cette dernière, la variation y_k de F d'un itéré au suivant est une source d'information à ne pas négliger. Comme on a

$$y_k := F(x_{k+1}) - F(x_k) = \left(\int_0^1 F'(x_k + t s_k) \, \mathrm{d}t \right) s_k,$$

il semble naturel d'imposer à la matrice M_{k+1} de vérifier la même équation que la jacobienne "moyenne" ci-dessus, c'est-à-dire

$$y_k = M_{k+1} s_k. \tag{2.20}$$

Cette équation porte le nom d'*équation de quasi-Newton*. Les n égalités qu'elle contient ne peuvent déterminer les n^2 éléments de M_{k+1}, si bien que pour fixer ceux-ci on utilise en général un principe stabilisateur : prendre M_{k+1} aussi proche que possible de M_k tout

en requérant (2.20). Plusieurs approches sont possibles, conduisant à des formules de mise à jour de M_k différentes. La solution la plus simple est de résoudre $\min\{\|M - M_k\|: (2.20) \text{ a lieu}\}$, où $\|\cdot\|$ est la norme ℓ_2 ou la norme de Frobenius. Si $s_k = y_k = 0$, sa solution est $M_{k+1} = M_k$. Si $s_k \neq 0$, sa solution est donnée par la *formule de Broyden*,

$$M_{k+1} = M_k + \frac{(y_k - M_k s_k) s_k^T}{\|s_k\|_2^2},$$

qui apporte une correction de rang 1 à M_k.

Théorème 2.9 (convergence locale de Broyden) *Si x_1 est proche d'une solution x_* de (2.14) telle que $F'(x_*)$ est inversible et si M_1 est proche de $F'(x_*)$, alors la suite générée par $x_{k+1} = x_k - M_k^{-1} F(x_k)$, avec M_k mise à jour par la formule de Broyden, est bien définie et converge super-linéairement vers x_*.*

La vitesse de convergence super-linéaire est typique, nous l'avons dit, des algorithmes de quasi-Newton bien conçus.

Résolution de problèmes d'optimisation

La problématique est la même pour les problèmes d'optimisation : on cherche à ce que M_{k+1} soit proche du hessien $H_{k+1} = \nabla^2 f(x_{k+1})$, sans calculer ce dernier. Cette fois, c'est de la variation du gradient d'un itéré au suivant qu'il faut tirer de l'information. Puisque l'on a

$$y_k := g_{k+1} - g_k = \left(\int_0^1 \nabla^2 f(x_k + t s_k) \, \mathrm{d}t \right) s_k,$$

il est naturel d'imposer à la matrice M_{k+1} de vérifier l'*équation de quasi-Newton* :

$$y_k = M_{k+1} s_k. \tag{2.21}$$

Il est également normal d'imposer à M_{k+1} d'être symétrique (puisque H_{k+1} l'est). Comme (2.21) ne détermine pas les $n(n+1)/2$ éléments de M_{k+1}, on utilise à nouveau le principe stabilisateur qui consiste à choisir M_{k+1} aussi proche que possible de M_k tout en respectant (2.21) et la symétrie. Il y a également plusieurs approches possibles, conduisant à des formules de mise à jour de M_k différentes.

Si on impose à M_{k+1} de ne différer de M_k que par une matrice de rang 1 (comme dans la formule de Broyden), on obtient la *formule symétrique de rang 1* ou *SR1*,

$$M_{k+1} = M_k + \frac{(y_k - M_k s_k)(y_k - M_k s_k)^T}{(y_k - M_k s_k)^T s_k}.$$

Cette formule a d'excellentes propriétés algébriques mais ne préserve pas la définie positivité éventuelle de M_k. Elle n'est donc utilisée qu'avec des régions de confiance. La formule a un léger défaut (son dénominateur peut s'annuler) qui peut sembler créer une

instabilité numérique. On peut montrer que celle-ci est bien prise en compte si on passe la mise à jour chaque fois que $|(y_k - M_k s_k)^\top s_k| \leq \sigma \|y_k - M_k s_k\| \|s_k\|$, avec $\sigma \simeq 10^{-8}$.

Pour que $d_k = -M_k^{-1} g_k$ soit une direction de descente, on a besoin que $M_k \succ 0$. Cette condition définissant un ensemble ouvert ne peut être utilisée directement comme contrainte dans le problème définissant M_{k+1}. On s'en sort en faisant prendre en compte cette condition par le critère de ce problème qui contiendra une barrière du cône $\mathbb{S}^n_{++}$. On introduit la fonction $\psi : \mathbb{S}^n_{++} \to \mathbb{R}$ définie en $M \succ 0$ par

$$\psi(M) = \operatorname{tr} M - \log \det M.$$

Cette fonction a pour unique minimum $M = I$, si bien que l'on aura M_{k+1} proche de M_k en résolvant $\min \{\psi(M_k^{-1/2} M M_k^{-1/2})\colon M \in \mathbb{S}^n_{++},\ y_k = M s_k\}$. Si la *propriété de monotonie*

$$y_k^\top s_k > 0 \tag{2.22}$$

est vérifiée, ce problème a une solution unique qui est donnée par la *formule de BFGS* (initiales des auteurs) :

$$M_{k+1} = M_k + \frac{y_k y_k^\top}{y_k^\top s_k} - \frac{M_k s_k s_k^\top M_k}{s_k^\top M_k s_k}. \tag{2.23}$$

De plus $M_{k+1} \succ 0$ s'il en est ainsi de M_k. Comme (2.22) est assurée par la règle de recherche linéaire de Wolfe (comparez avec (2.5)), c'est toujours cette règle qui est utilisée avec la formule de BFGS. Une itération de l'*algorithme de BFGS*, passant de (x_k, M_k) à (x_{k+1}, M_{k+1}), s'écrit donc

1. Calcul de la direction de descente $d_k = -M_k^{-1} g_k$,
2. détermination d'un pas $\alpha_k > 0$ par la règle de Wolfe (2.3)-(2.4),
3. $x_{k+1} = x_k + \alpha_k d_k$,
4. M_{k+1} donné par la formule de BFGS (2.23).

Cet algorithme a un résultat de convergence locale similaire au théorème 2.9 et le résultat de convergence globale remarquable suivant. On rappelle que l'ensemble de sous-niveau $\mathcal{N}_1$ est défini en (2.7).

Théorème 2.10 (convergence globale de BFGS, Powell) *Supposons que f soit convexe et $C^{1,1}$ dans un voisinage convexe de $\mathcal{N}_1$. Si l'algorithme de BFGS démarre en x_1 avec une matrice $M_1 \in \mathbb{S}^n_{++}$ et génère une suite $\{x_k\}$ telle que $\{f(x_k)\}$ est bornée inférieurement, alors* $\liminf_{k\to\infty} \|g_k\| = 0$.

On pense que ce résultat ne peut pas être étendu aux fonctions non convexes, mais cela n'empêche pas d'utiliser l'algorithme de BFGS pour minimiser des fonctions non convexes (il reste bien défini), avec de très bons résultats numériques.

L'algorithme de BFGS a une version, dite *à mémoire limitée*, encore appelée ℓ-BFGS, qui est très utile pour les problèmes de grande taille. Si $n \simeq 10^6$, on ne peut pas

en effet stocker la matrice $M_k \in \mathbb{S}^n$ en mémoire. L'algorithme ℓ-BFGS est fondé sur l'observation non triviale du fait que, si on connaît M_1 et les couples $\{(y_i, s_i)\}_{i=1}^{k-1}$, on peut aisément calculer $d_k = -M_k^{-1} g_k$ sans calculer explicitement M_k ! On pourrait donc mémoriser M_1 (en général un multiple de l'identité, donc peu encombrant en mémoire) et les couples, plutôt que la matrice M_k elle-même. Mais, dès que k dépasse quelques dizaines, les couples deviennent eux-même encombrant en mémoire. Dans l'algorithme ℓ-BFGS, on ne garde en mémoire que quelques couples, disons m, et à chaque itération on se débarrasse du plus ancien couple pour le remplacer par un nouveau. De façon formelle, la matrice M_k utilisée à l'itération $k \geq m$ est donc définie par

1. choix de $M_{k,0} \succ 0$,
2. **pour** $i = 1, \ldots, m$ **faire** $M_{k,i} = \mathrm{BFGS}(M_{k,i-1}, y_{k-m+i-1}, s_{k-m+i-1})$,
3. $M_k = M_{k,m}$.

On a utilisé la notation $\mathrm{BFGS}(M_k, y_k, s_k)$ pour désigner la matrice M_{k+1} obtenue par la formule de BFGS (2.23). L'algorithme ℓ-BFGS a permis de résoudre des problèmes sans contrainte avec des millions de variables, notamment en prévision météorologique.

2.3 Problèmes de moindres-carrés

Un *problème de moindres-carrés* est un problème de minimisation dans lequel le critère $f : \mathbb{R}^n \to \mathbb{R}$ a une structure bien particulière, puisqu'il est supposé être défini en $x \in \mathbb{R}^n$ par

$$f(x) = \frac{1}{2} \|r(x)\|_2^2 = \frac{1}{2} \sum_{i=1}^{m} r_i(x)^2. \tag{2.24}$$

La fonction $r : \mathbb{R}^n \to \mathbb{R}^m$ est appelée *résidu*. En général $m \gg n$. On note $J_k = r'(x_k)$ la *jacobienne* de r en x_k. Alors, le gradient et le hessien de f en x_k s'écrivent

$$g_k = J_k^\top r(x_k) \quad \text{et} \quad H_k = J_k^\top J_k + \sum_{i=1}^{m} r_i(x_k) \nabla^2 r_i(x_k).$$

On retrouve donc le cadre de la globalisation de l'algorithme de Newton pour la résolution d'équations non linéaires (voir la section 2.2.1), avec deux différences importantes : la jacobienne J_k du résidu est une matrice rectangulaire de type $m \times n$, qui n'a donc aucune raison d'être inversible, et le résidu est en général non nul en une solution. L'algorithmique associée à ce problème rappelle donc ce que l'on a vu sur l'algorithme de Newton, mais avec des nuances liées à ces différences.

2.3.1 Gauss-Newton

Dans l'algorithme de Gauss-Newton pour minimiser le fonction de moindres-carrés (2.24), le déplacement d_k ressemble à celui de Newton, mais ne retient du hessien H_k que la partie $J_k^\top J_k$. Il est solution du système linéaire

$$J_k^\top J_k d_k = -J_k^\top r(x_k), \tag{2.25}$$

que l'on appelle *équation normale*. Comme $\mathcal{R}(J_k^\top) = \mathcal{R}(J_k^\top J_k)$, celle-ci a toujours une solution (non nécessairement unique), qui peut être calculée par de nombreuses méthodes (résolution de l'équation normale par factorisation de Cholesky ou GC, résolution directe du problème de moindres-carrés linéaire équivalent par factorisations QR ou en valeurs singulières). Dans cet algorithme, le nouvel itéré est obtenu par $x_{k+1} = x_k + d_k$. Si on la compare à la méthode de Newton sur f, cette approche à deux intérêts :

- il n'est pas nécessaire de calculer les dérivées secondes des résidus r_i et
- le système (2.25) a toujours une solution.

Mais elle a aussi un inconvénient majeur. La convergence locale de la suite $\{x_k\}$ vers un minimiseur x_* de f n'est plus assurée. Elle va, en fait, dépendre de l'erreur commise en approchant H_k par $J_k^\top J_k$. On peut montrer que si $r(x_*) = 0$ (erreur d'approximation nulle en la solution), il y a convergence locale et quadratique. Si $r(x_*)$ est petit ou si les dérivées secondes $\nabla^2 r_i(x_*)$ sont petites (erreur d'approximation petite), il y a convergence locale et linéaire. Mais, si aucune de ces deux conditions n'est satisfaite (erreur d'approximation importante), la suite $\{x_k\}$ peut ne pas converger.

2.3.2 Globalisation

La globalisation de l'algorithme de Gauss-Newton par *recherche linéaire* est fondée sur la propriété providentielle suivante :

- si $g_k \neq 0$, toute solution d_k de (2.25) est une direction de descente de f en x_k.

Dès lors, après avoir calculé d_k, on peut déterminer un pas $\alpha_k > 0$ le long de cette direction par recherche linéaire (souvent par la règle d'Armijo (2.3), avec un choix de pas initial α_k^1 non trivial) et prendre comme nouvel itéré $x_{k+1} = x_k + \alpha_k d_k$. La convergence globale de cet algorithme est liée au cosinus de l'angle θ_k entre $-g_k$ et d_k (théorème 2.1). On a

$$\cos\theta_k = \frac{-r(x_k)^\top J_k d_k}{\|J_k^\top r(x_k)\|_2 \, \|d_k\|_2} = \frac{\|J_k d_k\|_2^2}{\|J_k^\top J_k d_k\|_2 \, \|d_k\|_2} \geq \frac{\|J_k d_k\|_2 \, \sigma_{\min}(J_k)}{\|J_k^\top J_k d_k\|_2} \geq \frac{\sigma_{\min}(J_k)}{\sigma_{\max}(J_k)},$$

où $\sigma_{\min}$ et $\sigma_{\max}$ désignent les plus petite et plus grande valeurs singulières. On peut alors comprendre les hypothèses du résultat de convergence globale ci-dessous.

Théorème 2.11 (Gauss-Newton avec recherche linéaire) *Supposons que f donnée par (2.24) soit $C^{1,1}$ dans un voisinage de $\mathcal{N}_1$. Soit $\{x_k\}$ une suite générée par l'algorithme de Gauss-Newton avec recherche linéaire. Si les valeurs singulières de J_k sont dans un compact de $]0, \infty[$, alors $J_k^\top r(x_k) \to 0$.*

Dans la globalisation de l'algorithme de Gauss-Newton par *régions de confiance*, on prend comme modèle quadratique de f, la fonction ψ_k définie en (2.9) avec l'approximation $M_k = J_k^\top J_k$ de H_k. Ceci revient à linéariser le résidu r dans (2.24). Le

sous-problème quadratique à résoudre à l'itération k s'écrit

$$\begin{cases} \min_s \ \frac{1}{2}\|r(x_k) + J_k s\|_2^2 \\ \|s\|_2 \leq \Delta_k. \end{cases} \tag{2.26}$$

Le réglage de $\Delta_k > 0$ se fait comme dans l'algorithme cadre de la section 2.1.2. Ce problème a une particularité : il est convexe, ce qui n'est en général pas le cas du modèle du second ordre (2.9)-(2.10). Voyons les avantages que l'on peut en tirer.

La résolution approchée de (2.26) par dogleg choisit comme déplacement s_k la combinaison convexe entre le point de Cauchy $-\alpha_k^{\mathrm{C}} J_k^\top r(x_k)$ et un déplacement de Gauss-Newton s_k^{GN} (une solution de (2.25), c'est-à-dire un minimiseur du critère de (2.26), qui existe toujours), qui est en norme la plus proche de Δ_k. On peut aussi résoudre (2.26) de manière approchée par *gradient conjugué tronqué*, sachant qu'il faut faire face au mauvais conditionnement éventuel dû au hessien $J_k^\top J_k$ du critère quadratique.

On peut aussi calculer une solution exacte de (2.26), ce qui revient à résoudre ses conditions d'optimalité de KKT qui sont nécessaires et suffisantes (car le problème est convexe) : $\|s_k\|_2 \leq \Delta_k$ et il existe $\lambda_k \geq 0$ tel que

$$(J_k^\top J_k + \lambda_k I) s_k = -J_k^\top r(x_k) \quad \text{et} \quad \lambda_k(\Delta_k - \|s_k\|_2) = 0. \tag{2.27}$$

Comme $J_k^\top J_k \succcurlyeq 0$, le théorème 2.2 conduit aux mêmes conditions. Ces conditions sont plus faciles à résoudre que dans le cas général (2.10), car elles ne présentent pas le *cas difficile*. Il existe en fait une méthode spécifique de résolution de ces conditions qui est fondée sur le fait que la première condition de (2.27) décrit l'optimalité du problème de moindres-carrés linéaire

$$\min_s \ \frac{1}{2} \left\| \begin{pmatrix} J_k \\ \lambda_k^{1/2} I \end{pmatrix} s + \begin{pmatrix} r(x_k) \\ 0 \end{pmatrix} \right\|^2 .$$

La première condition de (2.27) nous montre que la résolution des problèmes de moindres-carrés par régions de confiance est une autre implémentation de l'*algorithme de Levenberg-Marquardt*. À chaque itération de celui-ci, on détermine un déplacement le long du chemin de descente $\lambda \mapsto s_\lambda$ (voir la figure 2.2 à gauche), où s_λ est solution du système linéaire $(J_k^\top J_k + \lambda I) s_\lambda = -J_k^\top r(x_k)$, de manière à faire décroître f suffisamment. C'est exactement ce que font les régions de confiance, mais au lieu de piloter l'algorithme par le multiplicateur λ_k associé à la région de confiance, elles le font par le rayon de confiance. Certains auteurs considèrent que cette dernière approche est préférable, car on connaît un bon candidat initial pour le rayon de confiance, celui de l'itération précédente, alors que le multiplicateur λ_k serait moins stable d'une itération à l'autre. Mais cette opinion est controversée.

2.3.3 Apports quasi-newtoniens

Dans la section précédente, nous avons exploré diverses techniques pour forcer la convergence de l'algorithme de Gauss-Newton, qui peut ne pas converger localement, en utilisant la recherche linéaire ou les régions de confiance. Un autre défaut de l'algorithme

de Gauss-Newton doit aussi parfois être corrigé. Il se peut que sa convergence locale soit lente. Lorsqu'il advient, ce comportement est essentiellement dû à la mauvaise approximation de H_k que réalise $H_k^1 := J_k^\top J_k$. Certains numériciens (Dennis, Gay et Welsch, 1981) ont proposé d'approcher le terme manquant $H_k - H_k^1$, à savoir

$$H_k^2 := \sum_{i=1}^{m} r_i(x_k)\nabla^2 r_i(x_k),$$

par une technique quasi-newtonienne (voir la section 2.2.2). On note $M_k = J_k^\top J_k + M_k^2$ l'approximation de H_k, dans laquelle M_k^2 est l'approximation quasi-newtonienne de H_k^2.

Dans les méthodes à *recherche linéaire*, on a besoin d'avoir $M_k \succ 0$, pour que la direction $d_k = -M_k^{-1} g_k$ soit une direction de descente de f en x_k.

- Dans une première approche (Yabe et al., 1988-1995), on génère M_k sous la forme factorisée $M_k = (J_k + L_k)^\top (J_k + L_k)$. L'approximation suivante est alors obtenue en mettant à jour L_k par une formule qui revient à prendre $M_{k+1} = \mathrm{BFGS}(M_k^\#, y_k, s_k)$, avec $M_k^\# = (J_{k+1} + L_k)^\top (J_{k+1} + L_k)$, $s_k = x_{k+1} - x_k$ et $y_k = g_{k+1} - g_k$.
- Une autre possibilité (Schittkowski, 1988) est de transformer le problème de moindres-carrés en un problème avec contrainte d'égalité $\min\{\frac{1}{2}\|z\|_2^2 \colon r(x) = z\}$. Le hessien du lagrangien de ce problème s'écrit $\mathrm{Diag}(\sum_i \lambda_i \nabla_{xx}^2 r_i(x), I_m)$, où λ est le multiplicateur associé à la contrainte d'égalité. On approche alors ce hessien par la formule de BFGS avec correction de Powell (voir la section 3.1.1).

Dans les méthodes à *régions de confiance*, il n'est pas nécessaire de requérir la définie positivité de M_k. La matrice M_k^2 est alors mise à jour en utilisant une formule ne conservant pas nécessairement la définie positivité (par exemple SR1), avec $s_k = x_{k+1} - x_k$ et $y_k = (J_{k+1} - J_k)^\top r(x_{k+1})$.

3 Optimisation avec contraintes d'égalité et d'inégalité

Nous présentons dans cette section deux classes d'algorithmes qui peuvent résoudre des problèmes avec contraintes, tels que

$$(P_{EI}) \quad \begin{cases} \min f(x) \\ c_i(x) = 0, & i \in E \\ c_i(x) \leq 0, & i \in I. \end{cases}$$

Les notations ont été précisées à la section 1.1. Il s'agit d'une part des algorithmes newtoniens, qui s'attaquent directement aux conditions d'optimalité, et d'autres part des méthodes de points intérieurs, qui peuvent être vues comme des méthodes de pénalisation.

La difficulté principale de (P_{EI}) ne réside pas tant dans la présence des contraintes d'égalité (on reste dans le domaine de l'analyse), mais dans celle des inégalités. Celles-ci introduisent un *aspect combinatoire* dans (P_{EI}), qui rend le calcul de sa solution très

difficile. En effet, on ne sait pas quelles sont les contraintes d'inégalité actives en la solution et il y a 2^{m_I} manières de réaliser cela ($m_I := |I|$). Il se fait que les algorithmes éprouvent des difficultés à gérer ce grand nombre de possibilités, surtout lorsque le problème n'est pas convexe. Par exemple, même si (P_{EI}) a un critère quadratique, avec un hessien diagonal mais non convexe, et des contraintes linéaires (avec $I \neq \varnothing$), il est *NP-ardu* (ou NP-hard), ce qui implique que l'on ne connaît pas aujourd'hui d'algorithmes polynomiaux pour le résoudre (il en est de même pour le calcul de ses minima locaux).

3.1 Méthodes newtoniennes

3.1.1 SQP local

Les algorithmes newtoniens sont fondés sur la linéarisation d'équations caractérisant les solutions que l'on cherche (section 2.2.1). Comme en optimisation sans contrainte, où on linéarisait l'équation d'optimalité $\nabla f(x) = 0$, c'est ici le système d'optimalité (1.2) qui est linéarisé. De ce fait, les algorithmes sont *primaux-duaux*, c'est-à-dire qu'ils génèrent à la fois une suite primale $\{x_k\}$ convergeant vers une solution x_* de (P_{EI}) et une suite duale $\{\lambda_k\}$ convergeant vers un multiplicateur optimal λ_* associé à x_*.

Linéarisons le système d'optimalité de KKT (1.2) en un itéré primal-dual (x_k, λ_k). On note

$$g_k = \nabla f(x_k) \quad \text{et} \quad L_k = \nabla^2_{xx} \ell(x_k, \lambda_k).$$

Si d_k (resp. μ_k) est l'accroissement à apporter à x_k (resp. λ_k), on trouve

$$\left\{ \begin{array}{l} L_k d_k + c'(x_k)^\top \mu_k = -\nabla_x \ell(x_k, \lambda_k) \\ c_E(x_k) + c'_E(x_k) d_k = 0 \\ c_I(x_k) + c'_I(x_k) d_k \leq 0 \\ (\lambda_k + \mu_k)_I \geq 0 \\ (\lambda_k + \mu_k)_I^\top (c(x_k) + c'(x_k) d_k)_I = 0. \end{array} \right.$$

On a ajouté à la dernière équation linéarisée le terme "négligeable" $(\mu_k)_I^\top c'_I(x_k) d_k$, pour la raison suivante. Par la présence d'inégalités, le système ci-dessus n'est pas simple à résoudre. On lui donne une interprétation agréable en constatant qu'il s'agit des conditions d'optimalité du problème quadratique suivant, généralisant ainsi un point de vue qui avait déjà été fécond en optimisation sans contrainte :

$$\left\{ \begin{array}{l} \min_d \; g_k^\top d + \frac{1}{2} d^\top L_k d \\ c_E(x_k) + c'_E(x_k) d = 0 \\ c_I(x_k) + c'_I(x_k) d \leq 0. \end{array} \right. \tag{3.1}$$

Celui-ci s'appelle le *problème quadratique osculateur* (PQO) associé à (P_{EI}). Dans cette récriture, le multiplicateur associé aux contraintes de (3.1) s'écrit $\lambda_k^{\text{PQ}} = \lambda_k + \mu_k$.

On peut à présent écrire une itération de l'algorithme de Newton associé à (P_{EI}), plus communément appelé *algorithme SQP* (pour Sequential Quadratic Programming), celle passant de (x_k, λ_k) à (x_{k+1}, λ_{k+1}) :

1. calculer un point stationnaire $(d_k, \lambda_k^{\mathrm{PQ}})$ de (3.1),
2. prendre $x_{k+1} := x_k + d_k$ et $\lambda_{k+1} := \lambda_k^{\mathrm{PQ}}$.

Dans cet algorithme, on reporte donc la *combinatoire* présente dans (P_{EI}) sur une *suite* de PQO, là où elle est plus simple à prendre en compte.

Comme méthode newtonienne, l'algorithme SQP converge localement quadratiquement, pourvu que l'on prenne certaines précautions sur le choix des points stationnaires de (3.1). La non convexité du hessien du lagrangien L_k et la présence de contraintes d'inégalité font que (3.1) peut avoir plusieurs points stationnaires, dont certains sont indésirables (on peut par exemple avoir $d_k \neq 0$, alors que (x_k, λ_k) est une solution). Ceux-ci sont éliminés en sélectionnant un point stationnaire de (3.1) tel que d_k soit de norme minimale parmi toutes les composantes primales des autres points stationnaires. Voici le résultat le plus simple.

Théorème 3.1 (convergence locale de SQP) *Supposons que f et c soient $C^{2,1}$ dans un voisinage d'un point stationnaire x_* de (P_{EI}), ayant λ_* comme multiplicateur associé. Supposons également que l'on ait complémentarité stricte et que*

$$\begin{pmatrix} \nabla^2_{xx}\ell(x_*, \lambda_*) & c'_{E\cup I^0_*}(x_*)^\top \\ c'_{E\cup I^0_*}(x_*) & 0 \end{pmatrix} \text{ soit inversible.} \tag{3.2}$$

Considérons l'algorithme SQP, dans lequel d_k est un point stationnaire de norme minimale de (3.1)*. Alors, il existe un voisinage V de (x_*, λ_*) tel que, si $(x_1, \lambda_1) \in V$, l'algorithme SQP génère une suite $\{(x_k, \lambda_k)\}$ qui converge quadratiquement vers (x_*, λ_*) et les contraintes actives dans* (3.1) *sont celles de (P_{EI}).*

La condition (3.2) est l'analogue de l'hypothèse d'inversibilité de $F'(x_*)$ requise par l'algorithme de Newton pour résoudre le système d'équations non linéaires $F(x) = 0$.

Le fait que, en présence de complémentarité stricte, les contraintes actives dans le PQO se stabilisent dans le voisinage d'une solution a deux conséquences importantes. D'une part, cela veut dire qu'asymptotiquement, il n'y a plus d'aspect combinatoire dans (3.1), puisque l'on connaît les contraintes actives (ce sont celles de l'itération précédente). D'autre part, si l'on veut tirer parti de cette propriété, il faut que le solveur de (3.1) soit à même d'être plus efficace s'il connaît cette information. Pour cette raison, ces solveurs intègrent souvent une technique d'*activation de contraintes*.

Nous l'avons déjà dit : si $I \neq \varnothing$ et si le PQO n'est pas convexe, celui-ci est NP-ardu. Dès lors, pour ne pas passer trop de temps dans la résolution des sous-problèmes quadratiques, la plupart des implémentations de SQP utilisent une approximation M_k de L_k qui est définie positive, en particulier celle fournie par les techniques quasi-newtoniennes.

Version quasi-newtonienne

On s'intéresse ici à la construction d'une approximation quasi-newtonienne M_k du hessien du lagrangien $L_k := \nabla^2_{xx}\ell(x_k, \lambda_k)$, qui sera utilisée à la place de ce dernier dans (3.1). Le but est double

- ne pas calculer de dérivées secondes,
- avoir $M_k \succ 0$ pour que le PQO soit plus facilement résoluble.

Devant approcher le hessien du lagrangien, il est naturel d'utiliser la variation du gradient du lagrangien (à multiplicateur fixé) d'une itération à l'autre. Si on définit

$$y_k^\ell := \nabla_x \ell(x_{k+1}, \lambda_{k+1}) - \nabla_x \ell(x_k, \lambda_{k+1}),$$

il semble normal de mettre à jour M_k par la formule de BFGS avec $s_k = x_{k+1} - x_k$ et $y_k = y_k^\ell$. On n'a malheureusement pas ici de moyen sérieux d'assurer la *propriété de monotonie* $(y_k^\ell)^\top s_k > 0$, laquelle est essentielle au maintien de la définie positivité de M_k. On utilise dès lors une heuristique et la plus communément implémentée est la *correction de Powell* dans laquelle on prend pour vecteur y_k une combinaison convexe entre y_k^ℓ et $M_k s_k$:

$$y_k^{\mathrm{P}} := \theta_k y_k^\ell + (1{-}\theta_k) M_k s_k.$$

Le paramètre θ_k est pris aussi grand que possible dans $[0, 1]$, de telle sorte que $(y_k^{\mathrm{P}})^\top s_k \geq \kappa\, s_k^\top M_k s_k$ pour une constant $\kappa \in]0, 1[$ (typiquement $\kappa = 0.2$). On trouve

$$\theta_k = \begin{cases} 1 & \text{si } (y_k^\ell)^\top s_k \geq \kappa\, s_k^\top M_k s_k \\ (1{-}\kappa)\, \dfrac{s_k^\top M_k s_k}{s_k^\top M_k s_k - (y_k^\ell)^\top s_k} & \text{sinon.} \end{cases} \tag{3.3}$$

Ensuite M_k est mise à jour par la formule de BFGS en utilisant les vecteurs s_k et $y_k = y_k^{\mathrm{P}}$.

3.1.2 Globalisation

On peut déceler deux objectifs à réaliser dans (P_{EI}), celui de trouver un point admissible et celui de trouver un point optimal (parmi les points admissibles). Dans la globalisation de l'algorithme SQP, on a besoin de décider si un nouvel itéré est meilleur que le précédent (on utilisait le critère à minimiser dans les problèmes sans contrainte). Clairement, on ne peut pas utiliser un seul des deux objectifs pour prendre une telle décision. Il se peut, par exemple, qu'un nouvel itéré soit très convenable parce qu'il améliore grandement l'admissibilité, malgré un accroissement non négligeable du critère. Face à ce constat, deux stratégies ont été développées.

- Dans les *méthodes de pénalisation*, on combine les deux objectifs dans une même *fonction de mérite*, que l'on cherche à faire décroître à chaque itération. On verra que le poids respectif des deux objectifs dans la fonction de mérite est assez bien contrôlé pour un type de pénalisation qualifiée d'exacte.

- Dans les *méthodes de filtre*, on utilise des techniques d'*optimisation multicritère*. Elles sont en cours d'exploration et nous n'en parlerons pas ici.

Une fonction de mérite fréquemment utilisée est la suivante

$$\Theta_\sigma(x) = f(x) + \sigma\|c(x)^\#\|_P,$$

où $\sigma > 0$, $\|\cdot\|_P$ est une norme sur $\mathbb{R}^m$ et, pour $v \in \mathbb{R}^m$, $v^\#$ est défini par $(v^\#)_E = v_E$ et $(v^\#)_I = \max(v_I, 0)$. On observera que $c(x)^\# = 0$ si et seulement si x est admissible pour (P_{EI}), ce qui signifie que le second terme de Θ_σ pénalise la violation des contraintes, comme souhaité. L'intérêt de Θ_σ est sa simplicité : elle ne contient pas de multiplicateur à mettre à jour et ne fait pas intervenir de dérivée. Sa non différentiabilité pose peu de difficulté, car on s'en servira essentiellement pour évaluer la qualité des déplacements, non pas pour construire ceux-ci.

Sous certaines conditions, en particulier si σ est assez grand, Θ_σ est une fonction de *pénalisation exacte*, c'est-à-dire que les solutions de (P_{EI}) sont des minima de Θ_σ.

Théorème 3.2 (conditions suffisantes d'exactitude) *Supposons que f et $c_{E\cup I^0_*}$ soient deux fois différentiables en un minimum local x_* de (P_{EI}) en lequel* (1.2) *et les conditions suffisantes d'optimalité du théorème* 1.6 *ont lieu, et que*

$$\sigma > \sup_{\lambda_* \in \Lambda(x_*)} \|\lambda_*\|_D. \tag{3.4}$$

Alors Θ_σ a un minimum local strict en x_*.

La norme $\|v\|_D = \sup\{v^\top u : \|u\|_P = 1\}$ est la *norme duale* de la norme $\|\cdot\|_P$ qui est utilisée dans Θ_σ. Le seuil dans (3.4) qui assure l'exactitude de Θ_σ n'est pas connu, mais pourra être estimé avec de plus en plus de précision au cours des itérations, à partir des multiplicateurs λ_k générés par l'algorithme SQP.

On peut à présent résumer la stratégie suivie pour faire converger globalement l'algorithme SQP. On calcule une solution primale-duale du PQO (avec une région de confiance additionnelle éventuelle). On estime sur Θ_σ (avec un paramètre de pénalisation σ bien choisi) la qualité du déplacement d_k calculé. Si celui-ci ne fait pas décroître Θ_σ suffisamment, soit on fait de la recherche linéaire le long de d_k, soit on résout à nouveau le PQO avec une région de confiance plus petite. Précisons ces deux stratégies.

Globalisation par recherche linéaire

On suppose ici que le hessien du lagrangien L_k utilisé dans le PQO est approché par une matrice M_k. L'observation fondamentale est la suivante.

Lemme 3.3 (propriété de descente) *Si $(d_k, \lambda_k^{\mathrm{PQ}})$ est une solution primale-duale du PQO, si $v \mapsto \|v^\#\|_P$ est convexe et si $\sigma \geq \|\lambda_k^{\mathrm{PQ}}\|_D$, alors $\Theta'_\sigma(x_k; d_k) \leq -d_k^\top M_k d_k$.*

On en déduit que si $M_k \succ 0$ et si (x_k, λ_k) n'est pas un point stationnaire de (P_{EI}), alors d_k est une direction de descente de Θ_σ en x_k.

Schématisons une itération de l'algorithme SQP avec recherche linéaire, celle qui passe de (x_k, λ_k, M_k) à $(x_{k+1}, \lambda_{k+1}, M_{k+1})$:

1. calculer une solution primale-duale $(d_k, \lambda_k^{\mathrm{PQ}})$ du PQO (3.1), avec la matrice M_k plutôt que L_k,
2. choisir $\sigma_k \geq \|\lambda_k^{\mathrm{PQ}}\|_D + \bar{\sigma}$ ($\bar{\sigma} > 0$ est une constante),
3. trouver un pas $\alpha_k > 0$ le long de d_k de manière à faire décroître Θ_{σ_k} suffisamment,
4. prendre $x_{k+1} := x_k + \alpha_k d_k$ et $\lambda_{k+1} := \lambda_k + \alpha_k(\lambda_k^{\mathrm{PQ}} - \lambda_k)$,
5. mettre à jour M_k par la méthode de BFGS corrigée de la section 3.1.1.

Globalisation par régions de confiance

Comme en optimisation sans contrainte, l'intérêt des régions de confiance est de ne pas exiger la stricte convexité du critère du PQO. Elles permettent ainsi de pouvoir utiliser les dérivées secondes des fonctions définissant le problème. Si le carcan qu'elles imposent conduit à des algorithmes plus robustes, celui-ci est moins bien adapté aux problèmes de grande taille. L'introduction de régions de confiance est ici moins aisée qu'en optimisation sans contrainte. Pour cette raison, nous considérerons le cas, plus simple, où il n'y a que des contraintes d'égalité ($I = \varnothing$ dans (P_{EI}), on note $c \equiv c_E$).

Une approche naturelle est de résoudre le PQO avec une région de confiance additionnelle : $\|d_k\| \leq \Delta_k$. Cependant, cette contrainte supplémentaire peut rendre vide l'ensemble admissible du PQO, parce qu'il n'y aurait pas de direction d_k en norme plus petite que Δ_k, qui vérifie les contraintes d'égalité $c(x_k) + c'(x_k)d_k = 0$ (comme dans la figure 3.1). D'ailleurs, ces contraintes linéarisées peuvent déjà être elles-mêmes incompatibles. La stratégie que nous allons suivre consiste à décomposer le calcul du déplacement en deux étapes : on s'occupe d'abord de l'admissibilité et ensuite de l'optimalité (voir la figure 3.1).

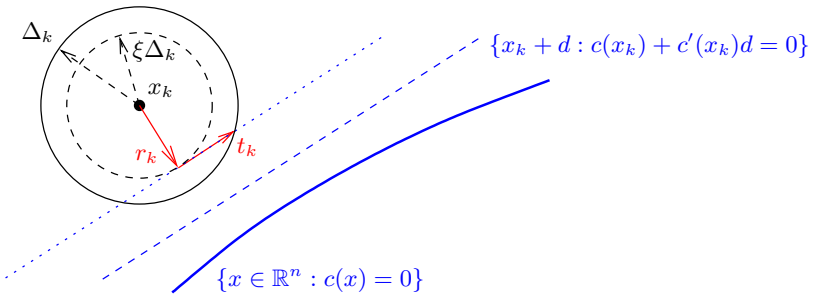


Figure 3.1: Région de confiance pour problèmes avec contrainte d'égalité

- Comme toute équation linéarisée, la contrainte $c(x_k) + c'(x_k)d_k = 0$ doit elle-même être soumise à une région de confiance. On adopte la technique des moindres-carrés employée dans la globalisation de l'algorithme de Newton par régions de confiance.

On calcule un déplacement r_k de restauration de la contrainte, qui est solution de

$$\begin{cases} \min_r \ \frac{1}{2}\|c(x_k) + c'(x_k)r\|_2^2 \\ \|r\|_2 \leq \xi\Delta_k, \end{cases} \tag{3.5}$$

où $\xi \in]0,1[$ est un scalaire (typiquement $\xi \simeq 0.7$) laissant une partie de la région de confiance à la disposition du déplacement tangent défini ci-dessous.

- On fait ensuite un déplacement tangent t_k minimisant le critère du PQO, tel que le déplacement total $d_k = r_k + t_k$ laisse inchangée la valeur des contraintes linárisées obtenue en fin de restauration : $c(x_k) + c'(x_k)d_k = c(x_k) + c'(x_k)r_k$ ou encore $c'(x_k)t_k = 0$, ce qui montre que t_k est tangent à la variété $\{x : c(x) = c(x_k)\}$. Le déplacement total d_k est donc solution de

$$\begin{cases} \min_d \ g_k^\top d + \frac{1}{2}d^\top M_k d \\ c'(x_k)d = c'(x_k)r_k \\ \|d\|_2 \leq \Delta_k. \end{cases} \tag{3.6}$$

  Ce problème est toujours réalisable (par $d = r_k$).

On règle ensuite le rayon de confiance Δ_k en mesurant l'adéquation du déplacement $d_k = r_k + t_k$ au moyen de la concordance entre le modèle donné par le PQO et la fonction de mérite (il est judicieux d'utiliser la norme ℓ_2 dans Θ_σ).

3.1.3 Commande optimale

Du point de vue de l'optimisation, un problème de commande optimale discrétisé se présente sous la forme

$$(P_{SI}) \quad \begin{cases} \min \ f(x) \\ c_S(x) = 0 \\ c_I(x) \leq 0. \end{cases}$$

Il ressemble très fort au problème (P_{EI}), si ce n'est que la fonction $c_S : \mathbb{R} \to \mathbb{R}^{m_S}$, qui remplace la fonction c_E, a des propriétés particulières. Il peut d'ailleurs y avoir en plus des contraintes d'égalité additionnelles $c_E(x) = 0$ sans ces propriétés particulières; nous les avons omises pour alléger l'exposé. Voici la structure apportée par l'équation $c_S(x) = 0$, dite *équation d'état*.

La variable $x = (y, u)$ est partitionnée en *variable d'état* $y \in \mathbb{R}^{m_S}$ et en *variable de commande* $u \in \mathbb{R}^{n-m_S}$. On partitionne de la même manière la jacobienne de c_S :

$$c_S'(x) = \begin{pmatrix} B(x) & N(x) \end{pmatrix},$$

avec une matrice $B(x)$ carrée d'ordre m_S. L'hypothèse-clé est de supposer que $B(x)$ est inversible en tout point rencontré, en particulier en la solution (et donc dans son voisinage). Cette hypothèse permet, dans certaines approches algorithmiques, de représenter l'état comme fonction implicite de la commande et donc d'éliminer l'état du problème. Ce n'est cependant pas ce point de vue que nous allons présenter ici. Notre but est de montrer que l'algorithme SQP, dans une version dite *réduite*, permet d'avoir le même

gain en espace mémoire et nombre d'opérations, tout en évitant la nécessité souvent coûteuse d'avoir des itérés qui satisfont l'équation d'état. L'approche est surtout avantageuse lorsqu'il n'y a que des contraintes d'inégalité sur la commande, autrement dit lorsque $\partial c_I/\partial y \equiv 0$.

Le problème quadratique osculateur (PQO) associé à (P_{SI}) en (x_k, λ_k) s'écrit

$$\left\{ \begin{array}{l} \min_d \; g_k^\top d + \frac{1}{2} d^\top L_k d \\ c_S(x_k) + c_S'(x_k) d = 0 \\ c_I(x_k) + c_I'(x_k) d \leq 0. \end{array} \right. \tag{3.7}$$

L'hypothèse d'inversibilité de $B_k = B(x_k)$ permet d'introduire un inverse à droite A_k^- de $A_k \equiv c_S'(x_k)$ et une matrice Z_k^- dont les colonnes forment une base du noyau de A_k :

$$A_k^- = \begin{pmatrix} B_k^{-1} \\ 0 \end{pmatrix} \quad \text{et} \quad Z_k^- = \begin{pmatrix} -B_k^{-1} N_k \\ I_{n-m} \end{pmatrix}.$$

Ces matrices ne peuvent en général pas être calculées explicitement, mais on s'autorise à les appliquer à un vecteur, ce qui requiert à chaque fois la résolution d'un système linéaire. Alors toute solution de l'équation d'état linéarisée est de la forme $d_k = r_k + Z_k^- h_k$, où $r_k = -A_k^- c_S(x_k) \in \mathbb{R}^n$ est un pas de restauration de l'équation d'état à commande fixée et $h_k \in \mathbb{R}^{n-m_S}$ est à déterminer. Si on reporte cette structure de d_k dans (3.7), le PQO devient

$$\left\{ \begin{array}{l} \min_h \; (g_k + L_k r_k)^\top Z_k^- h + \frac{1}{2} h^\top Z_k^{-\top} L_k Z_k^- h \\ c_I(x_k) + c_I'(x_k) r_k + c_I'(x_k) Z_k^- h \leq 0. \end{array} \right.$$

Ce problème se simplifie considérablement si l'on peut faire disparaître la matrice Z_k^-. Dans ce but, on suit les étapes suivantes :

- on approche $Z_k^{-\top} L_k Z_k^-$ par une matrice M_k générée par une technique quasi-newtonienne,
- ne pouvant plus calculer le terme $L_k r_k$ dans la partie linéaire du critère (car L_k n'est ni calculé ni approché), on le néglige,
- il faut par ailleurs supposer que la matrice $c_I'(x_k) Z_k^-$ est simple à calculer.

Les deux premières étapes définissent ce que l'on appelle les *méthodes de quasi-Newton réduites*. L'abandon du terme $L_k r_k$ fait perdre la convergence super-linéaire, mais on garde toutefois la *convergence super-linéaire en* 2 *pas*, c'est-à-dire $\|x_{k+2} - x_*\| / \|x_k - x_*\| \to 0$. Le troisième point ci-dessus sera certainement satisfait s'il y a peu de contraintes d'inégalité ou si celles-ci portent uniquement sur les variables de commande (alors $c_I'(x_k) Z_k^- = (\partial c_I/\partial u)(x_k)$). Dans ce dernier cas, le PQO devient particulièrement simple

$$\left\{ \begin{array}{l} \min_h \; g_k^\top Z_k^- h + \frac{1}{2} h^\top M_k h \\ c_I(x_k) + c_I'(x_k) r + (\partial c_I/\partial u)(x_k) h \leq 0. \end{array} \right.$$

Le vecteur $Z_k^{-\top} g_k = -N_k^\top B_k^{-\top} \nabla_y f(x_k) + \nabla_u f(x_k)$ s'appelle le *gradient réduit*. On y reconnaît l'*état adjoint* $B_k^{-\top} \nabla_y f(x_k)$.

3.2 Méthodes de points intérieurs

Le premier intérêt des méthodes de points intérieurs (PI) est leur propension à mieux gérer l'*aspect combinatoire* des problèmes d'optimisation avec contraintes d'inégalité, c'est-à-dire de trouver la bonne manière d'activer celles-ci parmi les 2^{m_I} possibilités (m_I est le nombre de contraintes d'inégalité). Plus exactement, elles remplacent la combinatoire par du *mauvais conditionnement*, celui des systèmes linéaires à résoudre. La gestion de ce mauvais conditionnement relève de l'algèbre linéaire et est en général plus simple à prendre en compte.

On n'obtient pas en optimisation non linéaire les résultats de complexité polynomiale qu'apportent les méthodes de PI en optimisation linéaire, quadratique convexe, semi-définie positive ou encore en optimisation convexe générale. La démarche suivie en optimisation non linéaire imite simplement celle qui a été mise au point sur ces problèmes plus structurés. Il y a deux points de vue complémentaires qui conduisent au même résultat : la *pénalisation logarithmique* et la *perturbation* des conditions d'optimalité.

Si on prend le point de vue de la pénalisation, on comprend que l'approche par PI requiert plus d'itérations que l'algorithme SQP (il faut résoudre une suite de problèmes d'optimisation non linéaires, chacun d'eux requérant la résolution d'une suite de systèmes linéaires), mais chaque itération est moins coûteuse (il suffit de résoudre un système linéaire plutôt qu'un problème quadratique). On appelle ici itération l'ensemble des opérations réalisées entre chaque évaluation des fonctions définissant le problème. Dès lors, en général, les méthodes de PI seront d'autant mieux adaptées que l'évaluation des fonctions n'est pas trop coûteuse et qu'il y a beaucoup de contraintes d'inégalité.

3.2.1 Conception d'un algorithme

Pour simplifier l'exposé, nous supposerons que le problème n'a que des contraintes d'inégalité non linéaires ($E = \varnothing$, on note $c \equiv c_I$, $m = m_I$) :

$$(P_I) \quad \begin{cases} \min_x f(x) \\ c(x) \leq 0. \end{cases}$$

Il y a plusieurs manières de définir un algorithme de points intérieurs pour ce problème et plusieurs type d'algorithmes. Nous présentons ci-dessous une approche assez communément suivie pour aboutir aux méthodes dites *primales-duales*, qui sont les plus performantes.

Dans les méthodes de points intérieurs les contraintes d'inégalité sont réalisées strictement (c'est ce que signifie le qualificatif "intérieur"). Cependant, un algorithme généraliste préfère ne pas imposer la réalisation stricte de contraintes *non linéaires*, car cela peut être coûteux (difficulté de trouver un point initial admissible et de maintenir l'admissibilité des itérés suivants). Dans ce but, le problème (P_I) est récrit sous une forme équivalente en introduisant des *variables d'écart* s :

$$(P'_I) \quad \begin{cases} \min_{(x,s)} f(x) \\ c(x) + s = 0 \\ s \geq 0. \end{cases}$$

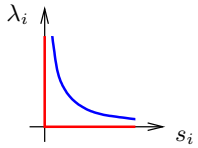
Ce sont les contraintes de positivité *linéaires* sur s qui seront réalisées strictement à chaque itération, alors que les contraintes d'égalité *non linéaires* ne seront satisfaites qu'asymptotiquement.

Les conditions d'optimalité de (P'_I) s'écrivent (les deux contraintes ont le même multiplicateur λ)

$$\left\{\begin{array}{l} \nabla f(x) + c'(x)^\top \lambda = 0 \\ c(x) + s = 0 \\ s^\top \lambda = 0, \quad s \geq 0, \quad \lambda \geq 0. \end{array}\right. \tag{3.8}$$

Toute la *combinatoire* du problème se trouve dans la dernière ligne de (3.8) : pour tout $i = 1, \ldots, m$, il faut choisir si c'est s_i ou λ_i qui est nul, soit 2^m possibilités en tout. On peut la faire disparaître en perturbant la condition $s_i\lambda_i = 0$ en $s_i\lambda_i = \mu$ (pour tout i) et en imposant la stricte positivité de s et λ :

$$\left\{\begin{array}{l} \nabla f(x) + c'(x)^\top \lambda = 0 \\ c(x) + s = 0 \\ S\lambda = \mu e, \end{array}\right. \tag{3.9}$$



où $\mu > 0$ est un paramètre, $S = \mathrm{Diag}(s_1, \ldots, s_m)$ et $e = (1 \ \cdots \ 1)^\top$. Les contraintes $s > 0$ et $\lambda > 0$ sont implicites (maintenues à chaque itération, soit par recherche linéaire, soit par régions de confiance). Le graphique à droite ci-dessus montre comment, dans l'espace (s_i, λ_i), la courbe *non différentiable* définie par $s_i\lambda_i = 0$ dans (3.8) est transformée en la courbe *différentiable* définie par $s_i\lambda_i = \mu$ dans (3.9). Au passage, cette observation montre le lien implicite qu'il peut y avoir entre combinatoire et non différentiabilité.

Nous venons d'exposer l'approche par perturbation. On peut aussi obtenir le système (3.9) par une technique de pénalisation. En effet, (3.9) sont les conditions d'optimalité du problème obtenu à partir de (P'_I) par pénalisation logarithmique de sa contrainte de positivité :

$$\left\{\begin{array}{l} \min_{(x,s)} \ f(x) - \mu \sum_{i=1}^m \log s_i \\ c(x) + s = 0. \end{array}\right. \tag{3.10}$$

On appelle (3.10) le *problème barrière* (les logarithmes forment une barrière empêchant s de quitter l'*orthant positif* $\mathbb{R}^m_+ := \{s \in \mathbb{R}^m : s \geq 0\}$). L'application (ou son image) qui à μ fait correspondre les solutions de (3.9) ou (3.10) (ce sont les mêmes en optimisation convexe, mais le concept est ambigu en optimisation non linéaire) est ce que l'on appelle le *chemin central*. Cependant, si f et c sont régulières, si la condition de qualification de contrainte (QC-IL) a lieu en une solution x_*, si l'on a complémentarité stricte en (x_*, λ_*) et si les conditions suffisantes du second ordre CS2 ont lieu, alors le chemin central est bien défini pour $\mu > 0$ petit et converge vers x_* lorsque $\mu \downarrow 0$.

Dans les algorithmes de points intérieurs, on résout approximativement le système d'optimalité perturbé (3.9) ou le problème barrière (3.10) pour une suite de paramètres μ tendant vers zéro. Il faut donc résoudre une suite de problèmes d'optimisation non linéaires, plus simples que le problème original. Le paramètre μ est le plus souvent multiplié par une constante entre 0 et 1 (typiquement 0.2), après chaque résolution

partielle de ces problèmes. Proche d'une solution, μ peut décroître plus rapidement. D'autre part, la précision requise sur (3.9) est prise proportionnelle à μ.

3.2.2 Résolution d'un problème barrière

L'intérêt de (3.9) ou (3.10) est de ne plus avoir d'inégalité, si bien qu'à chaque itération de résolution de ces problèmes, il suffit de résoudre un système linéaire. C'est une amélioration importante par rapport à l'algorithme SQP, dans lequel chaque itération requiert la résolution du problème quadratique osculateur (3.1).

Le système (3.9) est en général résolu par des itérations de Newton (ou de quasi-Newton). Le déplacement de Newton $d = (dx, ds, d\lambda)$ en $z = (x, s, \lambda)$ est solution du système linéaire (on l'a réordonné et transformé pour le rendre symétrique)

$$\begin{pmatrix} L & 0 & A^\top \\ 0 & S^{-1}\Lambda & I \\ A & I & 0 \end{pmatrix} \begin{pmatrix} dx \\ ds \\ d\lambda \end{pmatrix} = - \begin{pmatrix} \nabla_x \ell(x,\lambda) \\ \lambda - \mu S^{-1} e \\ c(x) + s \end{pmatrix}, \tag{3.11}$$

où on a noté $L = \nabla^2_{xx}\ell(x,\lambda)$, $A = c'(x)$ et $\Lambda = \mathrm{Diag}(\lambda_1, \ldots, \lambda_m)$. Si au lieu d'utiliser l'algorithme de Newton sur (3.9), nous l'avions fait sur (3.10), c'est-à-dire si nous avions opté pour l'algorithme SQP sur (3.10), nous aurions obtenu une équation identique à (3.11), si ce n'est que la matrice diagonale $S^{-1}\Lambda$ à gauche aurait été remplacée par μS^{-2}. On obtient alors ce que l'on appelle l'*algorithme primal*, qui donne de moins bons résultats que l'*algorithme primal-dual* (3.11). On peut donc voir la direction définie par (3.11) comme une direction SQP sur (3.10), mais modifiée. Ce point de vue n'est pas sans intérêt, car il suggère d'utiliser les techniques de globalisation de l'algorithme SQP sur (3.10) (section 3.1.2), pour globaliser l'algorithme primal-dual (3.11).

Avant de décrire la globalisation de l'algorithme, revenons un instant sur le système (3.11) pour en dénoncer le mauvais conditionnement, qui est la pierre d'achoppement des algorithmes de points intérieurs. Lorsque μ est petit (l'algorithme doit finalement prendre de tels μ pour que les solutions calculées de (3.10) soient proches de celles de (P'_I)), les solutions de (3.10) ont des s_i et des λ_i proches de zéro (car c'est le cas pour les solutions de (P'_I)). On comprend alors que le mauvais conditionnement du système (3.11) est en fait concentré dans la matrice $S^{-1}\Lambda$ qui a des éléments diagonaux qui tendent vers 0 ou vers $+\infty$ lorsque $\mu \downarrow 0$. Pour cette raison, on procède souvent par factorisation, non pas de la matrice de (3.11), qui est de très grande taille, mais de celle obtenue après élimination de ds et de $d\lambda$, c'est-à-dire

$$L + A^\top S^{-1}\Lambda A.$$

Par ailleurs, la résolution itérative de (3.11) requiert un préconditionneur. On peut par exemple multiplier la second ligne et la seconde colonne par $\mu^{-1/2}S$, pour trouver

$$\begin{pmatrix} L & 0 & A^\top \\ 0 & \mu^{-1}S\Lambda & \mu^{-1/2}S \\ A & \mu^{-1/2}S & 0 \end{pmatrix}.$$

Comme le long du chemin central, $\mu^{-1}S\Lambda = I$, on a amélioré le conditionnement.

La difficulté principale de la globalisation de l'algorithme de Newton (3.11) pour résoudre (3.10) tient au fait que le critère de ce problème n'est pas défini partout. Il est infini si $s \not> 0$ et il tend vers l'infini lorsque s se rapproche de la frontière de l'orthant positif. Ce fait peut paraître anodin, mais il a pour conséquence d'écarter la *recherche linéaire* comme technique de globalisation, en tout cas pour la direction définie par (3.11). On peut en effet trouver des contre-exemples dans lesquels le fait de réaliser la contrainte $c(x)+s$ linéarisée (troisième équation de (3.11)) et de maintenir la positivité des variables d'écart par recherche linéaire conduit à générer une suite $\{x_k\}$ qui reste emprisonnée dans un domaine fermé ne contenant aucun point admissible, ni aucun point stationnaire de la mesure d'admissibilité $x \mapsto \|c(x)^+\|_2^2$. Ce *piégeage d'itérés* ne se produit pas dans la globalisation par régions de confiance, que nous allons présenter brièvement.

Le problème (3.10) n'ayant que des contraintes d'égalité, sa globalisation par régions de confiance imite l'algorithme qui a été décrit à la section 3.1.2, avec une nuance cependant. Il faut en effet tenir compte de la contrainte implicite $s > 0$ donnée par la barrière logarithmique du critère. La région de confiance devient elliptique :

$$\{(dx, ds) : \|(dx, S^{-1}ds)\|_2 \leq \Delta\}.$$

Sans cette modification, lorsque $s_k > 0$ a de petites composantes, la phase de restauration de la contrainte $c(x) + s = 0$ aurait tendance à déterminer un déplacement semblable au déplacement r_k^E de la figure 3.2 qui conduirait à un point violant la con-

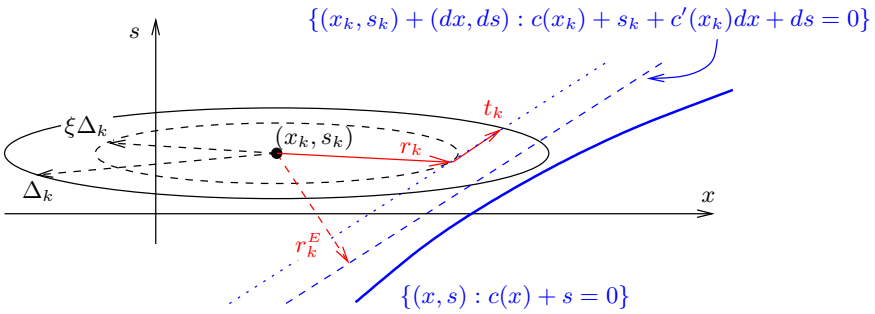


Figure 3.2: Région de confiance pour un algorithme de points intérieurs

trainte de positivité. Ceci aurait pour effet de devoir prendre un rayon de confiance inutilement très petit pour assurer la positivité des variables d'écart après la phase de restauration. La mise à l'échelle de la région de confiance corrige la direction du déplacement de restauration qui devient r_k à la figure 3.2, sans devoir prendre celui-ci très petit. Cette technique et quelques autres permettent d'obtenir un algorithme efficace, ainsi qu'un résultat de convergence des itérés vers une solution de (3.10).

4 Pour en savoir plus

De nombreux sujets n'ont pas été abordés dans cette courte synthèse. Citons les algorithmes sans calcul de dérivées, les variantes non-monotones des algorithmes, la pénalisation, les méthodes de dualité, la globalisation par filtre, les méthodes de projection et d'activation de contraintes, les méthodes de PI polynomiales en optimisation convexe, l'optimisation sous contraintes de complémentarité, l'optimisation SDP non linéaire, l'optimisation robuste, l'optimisation stochastique, les particularités liées aux problèmes séparables, l'optimisation multicritère, les langages de modélisation, etc. Ces sujets témoignent, s'il en était besoin, d'une activité importante des numériciens dans cette discipline.

La bibliographie sur l'algorithmique des problèmes d'optimisation différentiable est très vaste et il serait dommage de faire un tri trop sélectif. Nous avons regroupés par thèmes quelques ouvrages qui méritent d'être consultés.

- Conditions d'optimalité : Gauvin [8], Hiriart-Urruty [12], Bonnans et Shapiro [4], Gilbert [9].
- Prolégomènes à l'algorithmique : Ortega et Rheinboldt [18] pour la vitesse de convergence des suites, Griewank [11] pour la différentiation automatique, Vavasis [20] pour la complexité algorithmique en optimisation.
- Optimisation sans contrainte : Dennis et Schnabel [6], Kelley [13, 14, 15].
- Optimisation avec contraintes d'égalité et d'inégalité : Fletcher [7], Bertsekas [2], Conn, Gould et Toint [5], Gould, Orban et Toint [10], Nocedal et Wright [17], Bonnans, Gilbert, Lemaréchal et Sagastizábal [3], Gilbert [9].
- Points intérieurs : Nesterov et Nemirovskii [16], Wright [21], Renegar [19], Ben-Tal et Nemirovski [1], Nocedal et Wright [17], Bonnans, Gilbert, Lemaréchal et Sagastizábal [3].

On peut bien sûr développer des codes d'optimisation soi-même en s'inspirant de cette synthèse et des ouvrages cités ci-dessus mais, pour des applications standards de taille petite ou moyenne, on aura souvent intérêt à utiliser des codes développés par les numériciens ayant contribué à la mise au point des algorithmes sur lesquels ils reposent. Dans ce cas, les deux sites ci-dessous peuvent aider à faire un bon choix de code :

- `http://www-fp.mcs.anl.gov/otc/Guide/SoftwareGuide/`,
- `http://plato.asu.edu/guide.html`.

Pour conclure, nous listons sans ordre particulier et avec un minimum de commentaire, quelques codes de différentiation et d'optimisation, qui pourront être localisés facilement sur Internet :

- Différentiation symbolique : `AXIOM`, `MACSYMA`, `MAPLE`, `MATHEMATICA`, `MUPAD`.
- Différentiation automatique : `ADIFOR` (Fortran 77), `ADIC` (C), `ADOL-C` (C++), `TAPENADE` (Fortran 95, serveur Web), `MAD` (Matlab).
- Optimisation sans contrainte : `LBFGS` (ℓ-BFGS), `M1QN3` (ℓ-BFGS), `TN` (Newton tronqué), `NL2SOL` (moindres-carrés).

- Optimisation avec contraintes de borne : `LBFGSB` (ℓ-BFGS), `M2QN1` (BFGS), `TNBC` (Newton tronqué et recherche linéaire), `TRON` (Newton tronqué préconditionné et régions de confiance), `VE08AD` (Newton et quasi-Newton tronqué).
- Optimisation avec contraintes non linéaires : `LANCELOT` (lagrangien augmenté), `DONLP2` (SQP), `NLPQL` (SQP), `NPSOL` (SQP), `SNOPT` (SQP), `SPRNLP` (SQP), `SQPLAB` et `SQPPRO` (SQP), `VF02AD` (SQP), `FSQP` (SQP admissible), `FAIPA` (PI), `IPOPT` (PI), `KNITRO` (PI), `LOQO` (PI).

Bibliographie

[1] A. Ben-Tal, A. Nemirovski (2001). *Lectures on Modern Convex Optimization – Analysis, Algorithms, and Engineering Applications*. MPS/SIAM Series on Optimization 2. SIAM.

[2] D.P. Bertsekas (1999). *Nonlinear Programming* (seconde édition). Athena Scientific.

[3] J.F. Bonnans, J.Ch. Gilbert, C. Lemaréchal, C. Sagastizábal (2006). *Numerical Optimization – Theoretical and Practical Aspects* (seconde édition). Universitext. Springer Verlag, Berlin.

[4] J.F. Bonnans, A. Shapiro (2000). *Perturbation Analysis of Optimization Problems*. Springer Verlag, New York.

[5] A.R. Conn, N.I.M. Gould, Ph.L. Toint (2000). *Trust-Region Methods*. MPS/SIAM Series on Optimization 1. SIAM and MPS, Philadelphia.

[6] J.E. Dennis, R.B. Schnabel (1983). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs.

[7] R. Fletcher (1987). *Practical Methods of Optimization* (seconde édition). John Wiley & Sons, Chichester.

[8] J. Gauvin (1992). *Théorie de la programmation mathématique non convexe*. Les Publications CRM, Montréal.

[9] J.Ch. Gilbert (2007). *Optimisation Différentiable – Théorie et Algorithmes*. Syllabus de cours à l'ENSTA. `http://www-rocq.inria.fr/~gilbert/ensta/optim.html`.

[10] N. Gould, D. Orban, Ph.L. Toint (2005). Numerical methods for large-scale nonlinear optimization. In *Acta Numerica 2005*, pages 299–361. Cambridge University Press.

[11] A. Griewank (2000). *Evaluating Derivatives – Principles and Techniques of Algorithmic Differentiation*. SIAM Publication.

[12] J.-B. Hiriart-Urruty (1996). *L'Optimisation*. Que sais-je 3184. Presses Universitaires de France.

[13] C.T. Kelley (1995). *Iterative Methods for Linear and Nonlinear Equations*. SIAM Publication, Philadelphia.

[14] C.T. Kelley (1999). *Iterative Methods for Optimization*. SIAM Publication, Philadelphia.

[15] C.T. Kelley (2003). *Solving Nonlinear Equations with Newton's Method*. SIAM Publication, Philadelphia.

[16] Y.E. Nesterov, A.S. Nemirovskii (1994). *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM Studies in Applied Mathematics 13. SIAM, Philadelphia.

[17] J. Nocedal, S.J. Wright (2006). *Numerical Optimization* (seconde édition). Springer Series in Operations Research. Springer, New York.

[18] J.M. Ortega, W.C. Rheinboldt (1970). *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, New York.

[19] J. Renegar (2001). *A Mathematical View of Interior-Point Methods in Convex Optimization*. MPS/SIAM Series on Optimization 3. SIAM.

[20] S.A. Vavasis (1991). *Nonlinear Optimization – Complexity Issues*. Oxford University Press, New York.

[21] S.J. Wright (1997). *Primal-Dual Interior-Point Methods*. SIAM Publication, Philadelphia.

Index