Les Cahiers du GERAD ISSN: 0711–2440

Étude du comportement des méthodes BFGS et L-BFGS pour résoudre un sous-problème de région de confiance

J. Bourhis,
J.-P. Dussault, D. Orban

G–2019–64

September 2019

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2019
– Bibliothèque et Archives Canada, 2019

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2019
– Library and Archives Canada, 2019

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Étude du comportement des méthodes BFGS et L-BFGS pour résoudre un sous-problème de région de confiance

Johann Bourhis [a]
Jean-Pierre Dussault [b]
Dominique Orban [c]

[a] *Institut National des Sciences Appliquées, Rennes (Ille-et-Vilaine), France, 35708*

[b] *Université de Sherbrooke, Sherbrooke (Québec), Canada, J1H 5N4*

[c] *Département de Mathématiques et de Génie Industriel, Polytechnique Montréal (Québec) Canada, H3C 3A7*

johann.bourhis@insa-rennes.fr
jean-pierre.dussault@usherbrooke.ca
dominique.orban@gerad.ca

September 2019
Les Cahiers du GERAD
G–2019–64

Abstract: In this paper, we compare the BFGS and the conjugate gradient (CG) methods for solving unconstrained problems with a trust-region algorithm. The main result is a new relationship between CG and the Broyden class, the class of quasi-Newton methods that generalize the BFGS method. This new result allows to rediscover former results established by Broyden in 1970 [1]. In addition, we study the use of the limited-memory BFGS (L-BFGS) method in a trust-region algorithm by providing the same properties in comparison with CG. We present numerical results that show a difference of performance between these methods on ill-conditioned and large-scale problems. Some strategies are presented to improve performance by using various amounts of memory and a scaling factor in the L-BFGS method.

Keywords: Broyden class, BFGS method, conjugate gradient method, L-BFGS method, trust-region algorithm, ill-conditioned problems, large-scale optimization

Abstract: Dans ce papier, nous comparons la méthode BFGS à la méthode du gradient conjugué (CG) pour résoudre un problème d'optimisation sans contrainte avec un algorithme de régions de confiance. Le résultat clé est une nouvelle relation entre CG et la classe de Broyden, la classe de méthodes quasi-Newton qui généralise la méthode BFGS. Ce nouveau résultat permet de retrouver ceux établis par Broyden en 1970 [1]. Nous étudions ensuite l'utilisation de la méthode BFGS à mémoire limitée (L-BFGS) dans un algorithme de régions de confiance en donnant les mêmes relations qu'entre BFGS et CG. Nous présentons des résultats numériques pour mettre en lumière une différence de performance entre chacune de ces méthodes sur des problèmes mal conditionnés et en grande dimension. Certaines stratégies sont finalement présentées afin d'améliorer la performance de L-BFGS grâce à l'utilisation d'une mémoire variable et d'un facteur de mise à l'échelle.

Mots clés: Classe de Broyden, méthode BFGS, méthode du gradient conjugué, méthode L-BFGS, algorithme de régions de confiance, problèmes mal conditionné, optimisation en grande dimension

Acknowledgments: Nous tenons tout particulièrement à remercier Jean-Charles Gilbert (Université de Paris-Saclay) qui nous a fourni une preuve de la relation entre la méthode BFGS et la méthode du gradient conjugué [3]. Nous avons pu généraliser cette relation aux méthodes de la classe de Broyden, ce qui a joué un rôle déterminant dans le reste de nos travaux.

Notations et rappels

- Soient x, y deux vecteurs colonnes de $\mathbb{R}^n$, x^t désigne le vecteur transposé de x et $x^t y$ ainsi que $\|x\| = \sqrt{x^t x}$ désignent respectivement le produit scalaire usuel de x avec y et la norme euclidienne x.

- I désigne la matrice identité telle que pour tout vecteur x de $\mathbb{R}^n$, $Ix = x$.

- Soit une matrice A, A^t désigne la transposée de A. Si elle existe, A^{-1} désigne son inverse telle que $AA^{-1} = A^{-1}A = I$.

- On dit que la matrice A est symétrique si $A = A^t$ et qu'elle est symétrique définie positive si en plus de cela $x^t Ax > 0, \forall x \neq 0$. On notera alors $A \succ 0$. S'il existe $x \neq 0$ tel que $x^t Ax = 0$ et que $x^t Ax \geq 0, \forall x$, on dira que A est symétrique semi-définie positive et on notera alors $A \succeq 0$.

- Si $A \succ 0$, il existe une unique matrice $B \succ 0$ telle que $BB = A$. On notera alors $B = A^{1/2}$.

- Si $A \succ 0$ on peut alors définir un produit scalaire et une norme induits par A tels que, pour tous $x, y \in \mathbb{R}^n$, $< x, y >_A = x^t Ay$ et $\|x\|_A = \sqrt{x^t Ax}$. Par ailleurs, A^{-1} est symétrique définie positive également et l'inégalité de Cauchy-Schwarz nous donne

$$|x^t y| \leq \|x\|_A \|y\|_{A^{-1}} = \|y\|_A \|x\|_{A^{-1}}$$

- Soit un ensemble de vecteurs de $\mathbb{R}^n$ $\{x_1, x_2, ..., x_k\}$. On notera le sous-espace vectoriel engendré par ces vecteurs

$$\text{Vect}\{x_1, x_2, ..., x_k\} := \left\{ \sum_{i=1}^{k} \mu_i x_i \;\mid\; \mu_i \in \mathbb{R},\ i = 1, ..., k \right\}.$$

- Soit $E = \{x_1, x_2, ..., x_k\}$ une famille de vecteurs libres. On dira que E est une famille orthogonale si $x_i^t x_j = 0,\ i \neq j$. Soit $A \succ 0$, on dira que E est une famille conjuguée par rapport au produit scalaire induit par A si $x_i^t A x_j = 0,\ i \neq j$. On pourra dire plus simplement que E est une famille conjuguée.

- Si A est une matrice symétrique, alors elle possède n vecteurs propres orthogonaux deux à deux. On notera par $\{\lambda_i\}_{1 \leq i \leq n}$ les valeurs propres de A tels qu'il existe une suite de vecteurs propres $\{v_i\}_{1 \leq i \leq n}$ vérifiant

$$Av_i = \lambda_i v_i,\ i = 1, ..., n.$$

  De plus, $A \succ 0 \iff \lambda_i > 0$, pour tout $1 \leq i \leq n$.

- Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction deux fois différentiable en $x \in \mathbb{R}^n$,

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

  désignent respectivement le gradient et le hessien de f en x. Par ailleurs $\nabla^2 f$ est symétrique si $f \in \mathcal{C}^2$.

- Soit $\{x_k\}_{k \geq 0}$ une suite de vecteurs de $\mathbb{R}^n$. Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction différentiable. On notera

$$f_k = f(x_k) \quad et \quad g_k = \nabla f(x_k).$$

- Si f est une fonction quadratique on désignera par A sa matrice hessienne $\nabla^2 f$.

Introduction

L'optimisation non-linéaire sans contrainte a pour but de résoudre le problème

$$\min\{f(x) \mid x \in \mathbb{R}^n\}, \tag{$\mathcal{P}$}$$

où la fonction $f : \mathbb{R}^n \to \mathbb{R}$ est supposée être minorée et admettre une solution au problème ($\mathcal{P}$).

Il existe différentes approches numériques pour résoudre ce problème. Généralement, cette résolution ne distingue pas le minimum des autres points critiques et la solution calculée dépend notamment des paramètres de l'algorithme.

Il est bien connu qu'une quadratique strictement convexe, c'est à dire une fonction de la forme

$$q(x) = \tfrac{1}{2}x^t Ax + b^t x + c, \tag{1}$$

avec A une matrice symétrique définie positive, $b \in \mathbb{R}^n$ et $c \in \mathbb{R}$, possède un unique minimum. Il s'agit d'un cas simple bien utile en pratique pour tester la validité des algorithmes. En outre, de nombreux algorithmes, tels que la méthode du gradient conjugué, possèdent des propriétés intéressantes sur ces fonctions [5, 6, 9].

En supposant que $f \in \mathcal{C}^2$, nous pouvons écrire le développement de Taylor d'ordre 2 de f

$$f(x+p) = f(x) + \nabla f(x)^t p + \tfrac{1}{2}p^t \nabla^2 f(x) p + O(\|p\|^3), \tag{2}$$

qui est un modèle quadratique si nous négligeons la partie en $O(\|p\|^3)$. En utilisant ce résultat, l'algorithme de régions de confiance génère un modèle quadratique

$$q_x(p) = f(x) + \nabla f(x)^t p + \tfrac{1}{2}p^t \nabla^2 f(x) p, \tag{3}$$

à partir de la fonction f. On se donne ensuite un domaine Ω appelé la région de confiance à l'intérieur duquel on considère que cette approximation de f est bonne et nous minimisons un sous-problème quadratique avec les algorithmes qui possèdent les propriétés voulues.

L'objectif de cette étude est de comparer des méthodes quasi-Newton avec la méthode du gradient conjugué sur des quadratiques. Appliqué à une quadratique, l'algorithme du gradient conjugué converge en au plus n itérations, où n est la dimension du problème [1]. La méthode BFGS, du nom de ses inventeurs Broyden, Fletcher, Goldfarb et Shanno, est reconnue en pratique pour être très stable mais relativement coûteuse en calculs. Nous comparerons d'abord les deux méthodes sur des problèmes convexes sans contrainte. Le résultat le plus important de ce rapport sera de montrer qu'avec l'utilisation d'une recherche linéaire exacte, les deux algorithmes produisent les mêmes itérés à chaque itération. Nous verrons cependant qu'en grande dimension et avec des matrices A mal conditionnées, BFGS, qui effectue des calculs plus coûteux, continue à converger en au plus n itérations alors que le gradient conjugué souffre de problèmes numériques. Ce point est la principale motivation de nos recherches.

Lorsque $\nabla^2 f(x)$ n'est pas définie positive, le modèle quadratique généré n'est plus convexe et il faudra alors étudier ce qu'il advient des solutions données par BFGS et par le gradient conjugué. Il sera intéressant de distinguer deux approches. Pour certains problèmes indéfinis, les deux méthodes permettent de calculer le zéro du gradient (un point selle et non plus un minimum) et donc de résoudre le système linéaire $Ax = -b$, qui correspond à $\nabla q(p) = 0$. Enfin, si nous cherchons à minimiser la quadratique avec une contrainte de région de confiance, le problème de minimisation reste borné et la solution se trouve à la frontière de la région Ω.

La dernière partie de notre étude consistera à comparer le gradient conjugué ainsi que la méthode BFGS à la méthode L-BFGS qui est la version à mémoire limitée de BFGS. Celle-ci permet d'avoir des résultats intermédiaires à ceux des deux précédentes méthodes et donc de trouver un bon compromis en termes de performance de calculs et d'espace mémoire.

1 Étude de la méthode BFGS sur une quadratique strictement convexe

1.1 Construction de la méthode BFGS

Pour résoudre ($\mathcal{P}$), la méthode de Newton donne une direction de descente d_N telle que d_N minimise (3) :

$$\nabla^2 f(x_k) d_N = -\nabla f(x_k). \tag{4}$$

Lorsque f est une quadratique, la méthode de Newton trouve donc le point critique en une itération. Il s'agit donc d'une méthode très appréciée en optimisation qui continue à être efficace sur d'autres types de fonctions. L'inconvénient de la méthode de Newton est le calcul de $\nabla^2 f(x_k)$ qui devient très coûteux en grande dimension. Par ailleurs, la hessienne de f n'est souvent pas connue analytiquement ou du moins n'est pas implémentée numériquement. Les méthodes quasi-Newton visent à remplacer $\nabla^2 f$ par une approximation B_k de sorte à calculer une direction de descente qui vérifie

$$B_k d_k = -g_k. \tag{5}$$

C'est parfois même l'inverse $H_k = B_k^{-1}$ qui est directement recherchée de sorte à calculer

$$d_k = -H_k g_k. \tag{6}$$

Notons également que ces deux matrices ne sont pas recalculée entièrement à chaque itération car nous construisons B_{k+1} et H_{k+1} à partir d'une mise à jour de B_k et de H_k.

Nous effectuons ensuite une recherche linéaire de sorte à résoudre de façon approchée le sous-problème

$$\alpha_k = \min_{\alpha \geq 0} f(x_k + \alpha d_k). \tag{$\mathcal{SP}1$}$$

Avec une quadratique strictement convexe dont la hessienne est la matrice $A \succ 0$, il est possible d'exprimer explicitement la solution de ce sous-problème et nous calculons ainsi la longueur de pas correspondant à une recherche linéaire exacte

$$\alpha_k = -\frac{g_k^t d_k}{d_k^t A d_k}. \tag{7}$$

Itérativement nous nous rapprochons du minimum en mettant à jour le point courant

$$s_k = \alpha_k d_k, \tag{8.a}$$
$$x_{k+1} = x_k + s_k, \tag{8.b}$$

et plusieurs résultats de convergence ont été obtenus pour de nombreuses variantes de telles méthodes [1, 2].

Les méthodes de la sécante sont des méthodes quasi-Newton qui se ramènent à la méthode de la sécante en dimension 1. On construit itérativement une matrice H_{k+1} symétrique respectant l'équation de la sécante

$$H_{k+1} y_k = s_k, \tag{9}$$

où y_k est la différence de gradients

$$y_k = g_{k+1} - g_k. \tag{10}$$

à chaque itération, les directions précédemment explorées donnent à la nouvelle matrice un peu plus d'information sur la fonction et nous espérons nous rapprocher de mieux en mieux de $\nabla^2 f(x_k)$ et de son inverse.

Il existe en réalité une infinité de matrices symétriques satisfaisant (9). En imposant l'hérédité de la définie positivité entre H_k et H_{k+1} lorsque $s_k^t y_k > 0$ et en appliquant une approche variationnelle, Broyden définit la mise à jour de BFGS [1]

$$H_{k+1}^{BFGS} = (I - \rho_k s_k y_k^t) H_k (I - \rho_k y_k s_k^t) + \rho_k s_k s_k^t, \tag{11}$$

avec

$$\rho_k = 1/y_k^t s_k. \tag{12}$$

Il peut être parfois utile de pouvoir calculer B_k. Si $s_k^t B_k s_k \neq 0$, H_{k+1} est inversible et la formule directe de B_{k+1} nous est donnée par la formule de Sherman-Morrison-Woodbury [2]

$$B_{k+1}^{BFGS} = B_k - \frac{B_k s_k s_k^t B_k}{s_k^t B_k s_k} + \rho_k y_k y_k^t. \tag{13}$$

Si H_k est définie positive et $\rho_k > 0$, alors H_{k+1} est également définie positive (voir propriété 1 ci-dessous). C'est pour cela que l'on choisit généralement un multiple positif de la matrice identité pour H_0.

1.2 La classe de Broyden

La méthode BFGS fait partie d'une classe de méthodes quasi-Newton appelée la classe de Broyden. Les itérés s_k sont construits à partir de la mise à jour d'une matrice H_k qui approche la hessienne d'itération en itération de manière à respecter l'équation de la sécante. On peut définir l'ensemble de ces méthodes à partir des méthodes BFGS et DFP, une autre méthode quasi-Newton. Cette dernière se construit de manière analogue à BFGS. En nous assurant que $y_k^t H_k y_k \neq 0$ et $s_k^t y_k \neq 0$, on obtient les formules de H_{k+1} et B_{k+1} en interchangeant s_k avec y_k et H_k avec B_k dans (11) et (13)

$$B_{k+1}^{DFP} = (I - \rho_k y_k s_k^t) B_k (I - \rho_k s_k y_k^t) + \rho_k y_k y_k^t, \tag{14.a}$$

$$H_{k+1}^{DFP} = H_k - \frac{H_k y_k y_k^t H_k}{y_k^t H_k y_k} + \rho_k s_k s_k^t. \tag{14.b}$$

Les matrices H_k formées à partir des méthodes de la classe de Broyden sont de la forme

$$H_{k+1} = (1 - \phi_k) H_{k+1}^{DFP} + \phi_k H_{k+1}^{BFGS}, \ \phi_k \in \mathbb{R}, \tag{15}$$

et respectent l'équation de la sécante (9). Nous pouvons également calculer la mise à jour inverse ([2], lemme 4.2)

$$B_{k+1} = \Phi_k B_{k+1}^{DFP} + (1 - \Phi_k) B_{k+1}^{BFGS}, \tag{16}$$

avec

$$\Phi_k \equiv \Phi_k(\phi_k) = \frac{(1 - \phi_k)\,(s_k^t y_k)^2}{(s_k^t y_k)^2 + \phi_k \left[(y_k^t H_k y_k)\,(s_k^t B_k s_k) - (s_k^t y_k)^2 \right]}. \tag{17}$$

En utilisant les expressions de H_{k+1}^{BFGS} et de H_{k+1}^{DFP}, nous pouvons enfin obtenir la relation équivalente à (15)

$$H_{k+1} = H_k + \frac{s_k s_k^t}{s_k^t y_k} - \frac{H_k y_k y_k^t H_k}{y_k^t H_k y_k} + \phi_k (y_k^t H_k y_k)(v_k v_k^t), \tag{18}$$

$$v_k = \frac{s_k}{s_k^t y_k} - \frac{H_k y_k}{y_k^t H_k y_k}. \tag{19}$$

Propriété 1 ([2], lemme 4.2) *Supposons que $\rho_k \neq 0$, et que H_{k+1}^{BFGS} et H_{k+1}^{DFP} soient inversibles. La matrice H_{k+1} construite en (15) est singulière si et seulement si $\phi_k = \phi_k^c$ où*

$$\phi_k^c = \frac{(y_k^t s_k)^2}{(y_k^t s_k)^2 - (y_k^t H_k y_k)(s_k^t B_k s_k)}. \tag{20}$$

Propriété 2 ([7], page 151) *Pour des valeurs de $\phi_k > \phi_k^c$, les matrices H_{k+1} et B_{k+1} demeurent définies positives pourvu que B_k et H_k soient définies positives et que $s_k^t y_k > 0$. Cela n'est plus vrai si $\phi_k < \phi_k^c$. La matrice H_{k+1} est toujours inversible mais peut être indéfinie.*

Corollaire 1 *Considérons les méthodes de la classe de Broyden correspondant à des valeurs de $\phi_k \geq 0$, dont font partie DFP et BFGS. Si nous appliquons ces méthodes à une quadratique strictement convexe en prenant H_0 (et B_0) symétrique définie positive, alors les itérations suivantes donneront des H_k (et B_k) symétriques définies positives.*

Démonstration. Avec une quadratique $f(x) = \frac{1}{2}x^t Ax + b^t x$,

$$y_k = g_{k+1} - g_k = A(x_k + s_k) + b - (Ax_k + b) = As_k. \tag{21}$$

Comme f est strictement convexe

$$y_k^t s_k = s_k^t A s_k > 0. \tag{22}$$

À présent, supposons par récurrence que H_k et B_k soient symétriques définies positives. Par application de l'inégalité de Cauchy-Schwarz, on voit que $\phi_k^c < 0$ et les propriétés 1 et 2 sont toujours vraies pour des valeurs de $\phi_k \geq 0$. □

Remarque 1 *Nous avons également étudié une troisième méthode, appelée SR1, qui appartient à la classe de Broyden mais qui ne vérifie pas $\phi_k \in [0, 1]$. En posant*

$$\phi_k = \frac{y_k^t s_k}{(s_k - H_k y_k)^t y_k}, \tag{23}$$

nous obtenons la mise à jour de H_{k+1} par la méthode SR1 qui nous est donnée par

$$H_{k+1} = H_k + \frac{(s_k - H_k y_k)(s_k - H_k y_k)^t}{(s_k - H_k y_k)^t y_k}. \tag{24}$$

Si $(s_k - H_k y_k)^t y_k = 0$, H_{k+1} n'est pas définie. SR1 fait partie de ces méthodes qui ne préservent pas forcément la définie positivité des mises à jour lorsque $s_k^t y_k > 0$. Cet inconvénient dans le cadre des quadratiques convexes a été une piste d'exploration pour la généralisation de notre étude aux quadratiques non-convexes.

1.3 Relations avec la méthode du gradient conjugué

La méthode du gradient conjugué (CG) est une méthode itérative de recherche de minimum applicable aux quadratiques strictement convexes. Pour ce faire, on construit une suite de vecteurs conjugués par rapport au produit scalaire induit par la hessienne A à partir de la famille de gradients calculés au fil des itérations. S'inspirant de la méthode d'orthogonalisation de Gram-Shmidt, on construit à chaque itération une nouvelle direction conjuguée aux précédentes. On choisit une direction de la forme

$$d_k = -g_k + \beta_k d_{k-1}, \tag{25}$$

et on détermine β_k de sorte à obtenir la propriété de conjugaison $d_k^t A d_{k-1} = 0$,

$$\beta_k = \frac{g_k^t A d_{k-1}}{d_{k-1}^t A d_{k-1}}. \tag{26}$$

L'algorithme 1 nous montre une des nombreuses façons d'implémenter la méthode du gradient conjugué.

Algorithm 1 Algorithme du gradient conjugué [7]

procedure CG($\nabla f,\ x_0,\ \epsilon > 0,\ N \in \mathbb{N}^*$)
  $k \leftarrow 0$
  $p_k \leftarrow 0$
  $g_k \leftarrow \nabla f(x_0)$
  $d_k \leftarrow -g_k$
  while $\|g_k\| > \epsilon$ **and** $k \leq N$ **do**
    $b_k \leftarrow \text{diff}(\nabla f,\ x_0,\ d_k)$ ▷ Calcul de Ad_k grâce au gradient
    $\alpha_k \leftarrow -g_k^t d_k / d_k^t b_k$
    $p_k \leftarrow p_k + \alpha_k d_k$
    $g_k \leftarrow g_k + \alpha_k b_k$
    $\beta_k \leftarrow g_k^t b_k / d_k^t b_k$
    $d_k \leftarrow -g_k + \beta_k d_k$
    $k \leftarrow k + 1$
  end while
  return p_k
end procedure

Propriété 3 ([5], théorème 5.1) *Soit k tel que $g_k \neq 0$. Les gradients calculés par l'algorithme du gradient conjugué sont orthogonaux deux à deux,*

$$g_i^t g_k = 0, \quad i < k. \tag{27}$$

Nous avons par ailleurs

$$d_i^t g_k = 0, \quad i < k. \tag{28}$$

Démonstration. Par construction, les directions $d_0, d_1, ..., d_k$ sont obtenues en orthogonalisant la famille de vecteurs $\{-g_0, -g_1, ..., -g_k\}$ pour le produit scalaire associé à A. Donc le sous-espace vectoriel engendré par les directions $d_0, d_1, ..., d_k$ est $E_k = \text{Vect}\{g_0, g_1, ..., g_k\}$. Comme x_k réalise un minimum de f sur $x_0 + E_{k-1}$, g_k est orthogonal à E_{k-1} et on a donc (27). Par (25), $d_i \in E_{k-1}$, pour $i < k$, et nous avons pour finir (28). □

Propriété 4 ([5], théorème 4.2) *Avec une recherche linéaire exacte, les directions engendrées par le gradient conjugué sont conjuguées deux à deux. En outre, si nous effectuons une recherche linéaire exacte sur une quadratique strictement convexe dans une base de directions conjuguées, alors nous trouvons le minimum de cette fonction en au plus n itérations. C'est la propriété dite de terminaison quadratique ou de terminaison finie.*

Preuve de la conjugaison. La fonction f est une quadratique strictement convexe donc par (21) et (8), nous avons $Ad_i = (g_{i+1} - g_i)/\alpha_i$. Il s'en suit d'après (28) que $d_k^t A d_i = 0$ pour $k = 0, ..., k-2$ et grâce à (26),

$$d_i^t A d_j = 0,\ i \neq j. \tag{29}$$

□

Les méthodes de la classe de Broyden possèdent quelques propriétés remarquables à mettre en lien direct avec la méthode du gradient conjugué.

Théorème 1 *Considérons les méthodes de la classe de Broyden avec $\phi_k \neq \phi_k^c$ de sorte à préserver l'inversibilité des matrices H_k et B_k. Nous choisissons x_0 un vecteur quelconque et H_0 la matrice identité. Appliquons l'algorithme quasi-Newton avec recherche linéaire exacte (7)–(8) sur une fonction*

quadratique $f(x) = \frac{1}{2}x^tAx + b^tx$ *et supposons que les deux conditions suivantes sont vérifiées pour tout* k *:*

$$s_k^tAs_k \neq 0, \tag{30}$$

$$y_k^tH_ky_k \neq 0. \tag{31}$$

Notons enfin d_k^{CG} *et* d_k^ϕ *les directions engendrées respectivement par la méthode du gradient conjugué et par une méthode de la classe de Broyden de paramètre* ϕ_k*. Pour tout* k*, il existe* $\gamma_k^\phi \neq 0$ *tel que* $d_k^\phi = \gamma_k^\phi d_k^{CG}$*. Nous avons par ailleurs l'identité*

$$d_{k+1}^\phi = \frac{\gamma_k^\phi g_k^tg_k + \phi_k g_{k+1}^tg_{k+1}}{\gamma_k^\phi g_k^tg_k + g_{k+1}^tg_{k+1}} d_{k+1}^{CG}. \tag{32}$$

Il s'en suit que les itérés s_k *générés par les méthodes de la classe de Broyden sont identiques à ceux du gradient conjugué et ne dépendent donc pas de la valeur de* ϕ_k*.*

Remarque 2 *Le théorème 1 s'applique à des quadratiques qui ne sont pas forcément convexes. Si les conditions du théorème sont réunies, alors les méthodes de la classe de Broyden peuvent trouver un point critique, et plus généralement la solution du système linéaire* $Ax = -b$ *correspondant à l'équation* $\nabla f(x) = 0$*. Tout comme le gradient conjugué, la solution est trouvée en au plus* n *itération, même sur une quadratique non-convexe. La longueur de pas optimale* α_k *associée à une recherche linéaire exacte pour la recherche d'un point critique correspond à la solution du problème*

$$\tfrac{d}{d\alpha} f(x_k + \alpha d_k)\big|_{\alpha_k} = d_k^t \nabla f(x_{k+1}) = 0, \tag{33}$$

avec α_k *et* x_{k+1} *qui sont toujours définis par (7) et (8). Dans le cas où* A *n'est pas définie positive, la longueur* α_k *peut alors être négative. Enfin, si* $s_k^tAs_k = 0$*,* α_k *n'est pas défini.*

Avant de démontrer le théorème 1, nous énonçons quelques propriétés importantes qui en découlent.

Corollaire 2 *Notons* A *la hessienne de* f *et* x^* *tel que* $\nabla f(x^*) = 0$*. Si les conditions du théorème 1 sont réunies, alors la méthode du gradient conjugué est bien définie et les méthodes de la classe de Broyden héritent entre autre des propriétés suivantes [5, 6] :*

(i) La propriété de conjugaison est vérifiée entre chacun des itérés

$$s_i^tAs_j = 0, \ i \neq j. \tag{34}$$

(ii) Les itérés convergent vers la solution en au plus n *itérations. Plus précisément, il existe* $k < n$ *tel que* $x_k = x^*$*.*

(iii) Si A *possède seulement* r *valeurs propres distinctes, alors les itérés convergent vers la solution en au plus* r *itérations. Plus précisément, il existe* $k < r$ *tel que* $x_k = x^*$*.*

(iv) Soient $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ *les valeurs propres de* A*,*

$$\|x_k - x^*\|_A \leq \left(\frac{\lambda_{n-k} - \lambda_1}{\lambda_{n-k} + \lambda_1}\right) \|x_0 - x^*\|_A. \tag{35}$$

(v) Le sous-espace vectoriel engendré par la suite des itérés correspond au sous-espace de Krylov

$$\begin{aligned} E_k = \text{Vect}\{s_0, s_1, ..., s_k\} &= \text{Vect}\{g_0, g_1, ..., g_k\} \\ &= \text{Vect}\{g_0, Ag_0, ..., A^kg_0\}. \end{aligned} \tag{36}$$

En outre, nous avons les propriétés suivantes sur la matrice H_k *qui découlent directement des points précédents [1] :*

(vi) *L'équation de la sécante est vérifiée entre H_k et tous les itérés précédents, c'est-à-dire*

$$H_{k+1}y_j = s_j, \quad j = 0, 1, ..., k. \tag{37}$$

(vii) *La matrice $H_{k+1}A$ possède au moins $k+1$ valeurs propres unitaires associées aux vecteurs propres $\{s_0, s_1, .., s_k\}$:*

$$H_{k+1}As_j = s_j, \quad j = 0, 1, ..., k. \tag{38}$$

(viii) *Si n itérations sont réalisées, alors nous avons*

$$H_n = A^{-1}. \tag{39}$$

Pour démontrer le théorème 1, il est plus pratique d'utiliser le résultat suivant.

Proposition 1 *Nous pouvons écrire les directions du gradient conjugué avec une nouvelle expression,*

$$d_k^{CG} = \begin{cases} -g_0, & si\ k = 0 \\ -g_k + \dfrac{g_k^t g_k}{g_{k-1}^t g_{k-1}} d_{k-1}^{CG}, & si\ k \geq 1. \end{cases} \tag{40}$$

Démonstration. Il suffit de remarquer que $As_{k-1} = g_k - g_{k-1}$. Nous pouvons ainsi utiliser les relations (27) et (28) dans la formule (26). □

Preuve du théorème 1. On montre par récurrence que les vecteurs d_k^ϕ générés par les méthodes de la classe de Broyden sont colinéaires aux vecteurs d_k^{CG} et que

$$H_i g_k = g_k, \ i = 0, ..., k-1. \tag{41}$$

Si les deux vecteurs d_k^ϕ et d_k^{CG} sont colinéaires, alors il s'en suit qu'avec une recherche linéaire exacte, les deux méthodes trouvent le même itéré $s_k = \alpha_k^{CG} d_k^{CG} = \alpha_k^\phi d_k^\phi$.

Il est clair que pour $k = 0$, $d_0^\phi = d_0^{CG} = -g_0$ et donc $s_0^{CG} = s_0^\phi$. Nous savons également que $H_0 g_1 = g_1$. Supposons à présent que les hypothèses de récurrence soient vraies jusqu'à un indice $k \geq 0$ et montrons qu'elles sont toujours vérifiées à l'indice $k+1$ si $g_{k+1} \neq 0$. Sous l'hypothèse de récurrence, les itérés $s_1, s_2, ..., s_k$ des méthodes de la classe de Broyden sont identiques à ceux du gradient conjugué. Dès lors $s_i^t g_{k+1} = 0$ d'après (28) pour $i = 0, 1, ..., k$ et par (19) et (41),

$$v_i^t g_{k+1} = \left(\frac{s_i^t}{s_i^t y_i} - \frac{y_i^t H_i}{y_i^t H_i y_i} \right) g_{k+1} = -\frac{y_i^t g_{k+1}}{y_i^t H_i y_i} \tag{42}$$

Donc par (18) et (41),

$$\begin{aligned} H_{i+1}g_{k+1} &= H_i g_{k+1} - \frac{H_i y_i y_i^t H_i g_{k+1}}{y_i^t H_i y_i} + \phi_i (y_i^t H_i y_i) v_i v_i^t g_{k+1} \\ &= g_{k+1} - \frac{H_i y_i y_i^t g_{k+1}}{y_i^t H_i y_i} - \phi_i v_i y_i^t g_{k+1}. \end{aligned} \tag{43}$$

D'après (10) et (27), $y_i^t g_{k+1} = 0$ pour $i = 0, 1, ..., k-1$. Donc $H_i g_{k+1} = g_{k+1}$ pour $i = 0, 1, ..., k$. Ce qui démontre (41).

Toujours d'après (27),

$$y_k^t g_{k+1} = g_{k+1}^t g_{k+1}. \tag{44}$$

Par hypothèse de récurrence, il existe un réel $\gamma_k^\phi \neq 0$ tel que

$$d_k^\phi = \gamma_k^\phi d_k^{CG}. \tag{45}$$

D'après (28) et (40) nous avons les relations

$$y_k^t d_k^{CG} = \left(g_{k+1}^t - g_k^t\right)\left(-g_k + \beta_k d_{k-1}^{CG}\right) = g_k^t g_k, \tag{46}$$

$$H_k y_k = H_k\left(g_{k+1} - g_k\right) = g_{k+1} + d_k^{\phi} = g_{k+1} + \gamma_k^{\phi} d_k^{CG}. \tag{47}$$

Et donc

$$y_k^t H_k y_k = g_{k+1}^t g_{k+1} + \gamma_k^{\phi} g_k^t g_k. \tag{48}$$

En reprenant (43) avec $i = k$ et grâce à (19) et (44),

$$\begin{aligned}
H_{k+1}g_{k+1} &= g_{k+1} - \frac{H_k y_k y_k^t g_{k+1}}{y_k^t H_k y_k} - \phi_k g_{k+1}^t g_{k+1}\left(\frac{s_k}{s_k^t y_k} - \frac{H_k y_k}{y_k^t H_k y_k}\right) \\
&= g_{k+1} - \frac{g_{k+1}^t g_{k+1} H_k y_k}{y_k^t H_k y_k} + \phi_k \frac{g_{k+1}^t g_{k+1} H_k y_k}{y_k^t H_k y_k} - \phi_k g_{k+1}^t g_{k+1}\left(\frac{d_k^{CG}}{y_k^t d_k^{CG}}\right) \\
&= g_{k+1} - g_{k+1}^t g_{k+1}\left(1 - \phi_k\right)\left(\frac{g_{k+1} + \gamma_k^{\phi} d_k^{CG}}{g_{k+1}^t g_{k+1} + \gamma_k^{\phi} g_k^t g_k}\right) - \phi_k g_{k+1}^t g_{k+1}\left(\frac{d_k^{CG}}{g_k^t g_k}\right) \\
&= g_{k+1}\left(1 - \frac{g_{k+1}^t g_{k+1}\left(1 - \phi_k\right)}{g_{k+1}^t g_{k+1} + \gamma_k^{\phi} g_k^t g_k}\right) - d_k^{CG}\left(\frac{g_{k+1}^t g_{k+1}\gamma_k^{\phi}\left(1 - \phi_k\right)}{g_{k+1}^t g_{k+1} + \gamma_k^{\phi} g_k^t g_k} + \frac{\phi_k g_{k+1}^t g_{k+1}}{g_k^t g_k}\right) \\
&= g_{k+1}\left(\frac{g_{k+1}^t g_{k+1} + \gamma_k^{\phi} g_k^t g_k - g_{k+1}^t g_{k+1}(1 - \phi_k)}{g_{k+1}^t g_{k+1} + \gamma_k^{\phi} g_k^t g_k}\right) \\
&\qquad - d_k^{CG}\left(\frac{g_{k+1}^t g_{k+1}\left(g_k^t g_k \gamma_k^{\phi}(1 - \phi_k) + \phi_k g_{k+1}^t g_{k+1} + \gamma_k^{\phi}\phi_k g_k^t g_k\right)}{g_k^t g_k\left(g_{k+1}^t g_{k+1} + \gamma_k^{\phi} g_k^t g_k\right)}\right) \\
&= \left(\frac{\gamma_k^{\phi} g_k^t g_k + \phi_k g_{k+1}^t g_{k+1}}{\gamma_k^{\phi} g_k^t g_k + g_{k+1}^t g_{k+1}}\right)\left(g_{k+1} - \frac{g_{k+1}^t g_{k+1}}{g_k^t g_k} d_k^{CG}\right).
\end{aligned} \tag{49}$$

D'où, par (6) et (40),

$$d_{k+1}^{\phi} = \frac{\gamma_k^{\phi} g_k^t g_k + \phi_k g_{k+1}^t g_{k+1}}{\gamma_k^{\phi} g_k^t g_k + g_{k+1}^t g_{k+1}} d_{k+1}^{CG} = \gamma_{k+1}^{\phi} d_{k+1}^{CG}. \tag{50}$$

Notons que grâce aux hypothèses (30) et (31), d_{k+1}^{CG} et d_{k+1}^{ϕ} sont bien définies. Enfin $\phi_k \neq \phi_k^c$ entraîne d'après la propriété 1 que $H_{k+1}g_{k+1} \neq 0$ et les deux directions sont bien colinéaires. Avec une recherche linéaire exacte nous avons donc $s_{k+1} = \alpha_{k+1}^{\phi} d_{k+1}^{\phi} = \alpha_{k+1}^{CG} d_{k+1}^{CG}$ sous la condition que $s_{k+1}^t A s_{k+1} \neq 0$, d'après (7) et (8.a). Ceci termine donc la preuve du théorème 1. □

Preuve des assertions (37), (38) et (39) du corolaire 2. Les arguments de Broyden [1] sont les suivants. En utilisant le fait que $s_k^t A s_j = 0$ pour $j < k$ et que $y_k = A s_k$, on démontre par récurrence que $H_k y_j = s_j$ pour tout $j = 0, ..., k-1$. En effet, un rapide calcul nous montre que si $H_k y_j = s_j$ pour $j = 0, ..., k-1$, alors $H_{k+1} y_j = s_j$ pour $j = 0, ..., k-1$. Comme par construction $H_{k+1} y_k = s_k$ et $H_1 y_0 = s_0$, la récurrence est établie et nous avons donc (37). Pour obtenir (38), il suffit de remplacer y_j par $A s_j$ dans (37). Enfin, puisque les itérés du gradient conjugué sont indépendants, nous pouvons écrire à la n-ième itération que tout vecteur x est combinaison linéaire des $\{s_k\}_{k \geq 0}$. Il s'en suit d'après (38) que $H_n A x = x$ pour tout $x \in \mathbb{R}^n$, ce qui démontre (39). □

Théorème 2 *Si nous appliquons les méthodes de la classe de Broyden sur une quadratique strictement convexe avec une recherche linéaire exacte et $H_0 = I$, alors une condition suffisante pour que (31) soit vérifiée est que $\phi_k > \phi_k^c$. Les itérés s_k générés sont alors les mêmes que ceux du gradient conjugué et l'identité (32) est vérifiée. Nous avons en particulier la convergence de ces méthodes vers le minimum de la quadratique en au plus n itérations.*

Démonstration. Sur une quadratique strictement convexe, $s_k^t A s_k > 0$ et donc (30) est vérifiée pour tout k. D'après (48), une condition suffisante pour que $y_k^t H_k y_k \neq 0$ est que $\gamma_k^\phi \geq 0$. Par ailleurs d_{k+1}^ϕ et d_{k+1}^{CG} sont colinéaires à condition que $\gamma_{k+1}^\phi \neq 0$. Premièrement, $\gamma_0^\phi = 1$, puisque $d_0^\phi = d_0^{CG} = -g_0$. Supposons à présent par récurrence que $\gamma_k^\phi > 0$ jusqu'à un rang $k \geq 0$ et montrons que $\gamma_{k+1}^\phi > 0$ à condition que $\phi_k > \phi_k^c$. Étant donné que

$$\gamma_{k+1}^\phi = \frac{\gamma_k^\phi g_k^t g_k + \phi_k g_{k+1}^t g_{k+1}}{\gamma_k^\phi g_k^t g_k + g_{k+1}^t g_{k+1}}, \tag{51}$$

et que par hypothèse $\gamma_k^\phi g_k^t g_k + g_{k+1}^t g_{k+1} > 0$, il s'en suit que $\gamma_{k+1}^\phi > 0$ si et seulement si

$$\phi_k > -\frac{\gamma_k^\phi g_k^t g_k}{g_{k+1}^t g_{k+1}}. \tag{52}$$

D'après (20),

$$\begin{aligned} \phi_k^c &= \frac{(y_k^t s_k)^2}{(y_k^t s_k)^2 - (y_k^t H_k y_k)(s_k^t B_k s_k)} \\ &= \frac{(y_k^t d_k^\phi)^2}{(y_k^t d_k^\phi)^2 - (y_k^t H_k y_k)(d_k^{\phi\,t} B_k d_k^\phi)}. \end{aligned} \tag{53}$$

Par ailleurs, $d_k^\phi = -H_k g_k$ et donc $B_k d_k^\phi = -g_k$. D'où, par (45),

$$d_k^{\phi\,t} B_k d_k^\phi = -\gamma_k^\phi g_k^t d_k^{CG} = \gamma_k^\phi g_k^t g_k. \tag{54}$$

Il s'en suit avec (46) et (48) que

$$\begin{aligned} \phi_k^c &= \frac{\left(\gamma_k^\phi g_k^t g_k\right)^2}{\left(\gamma_k^\phi g_k^t g_k\right)^2 - \left(\gamma_k^\phi g_k^t g_k + g_{k+1}^t g_{k+1}\right)\gamma_k^\phi g_k^t g_k} \\ &= -\frac{\gamma_k^\phi g_k^t g_k}{g_{k+1}^t g_{k+1}}. \end{aligned} \tag{55}$$

Donc $\gamma_{k+1}^\phi > 0$ si et seulement si $\phi_k > \phi_k^c$. Nous obtenons alors par (48) que $y_{k+1}^t H_{k+1} y_{k+1} > 0$ et nous venons de montrer par récurrence que (31) est vérifiée pour tout k. □

Corollaire 3 *Sur une quadratique strictement convexe et avec $H_0 = I$, les méthodes de la classe de Broyden avec $\phi_k \geq 0$ sont toujours bien définies. Soient d_k^{BFGS} et d_k^{DFP} les directions engendrées par les algorithmes de BFGS et de DFP. Soient γ_k^{BFGS} et γ_k^{DFP} les coefficients tels que $d_k^{BFGS} = \gamma_k^{BFGS} d_k^{CG}$ et $d_k^{DFP} = \gamma_k^{DFP} d_k^{CG}$. Nous avons,*

$$d_{k+1}^{BFGS} = d_{k+1}^{CG}, \tag{56}$$

et donc $\gamma_k^{BFGS} = 1$ pour tout k. Nous avons également,

$$d_{k+1}^{DFP} = \frac{\gamma_k^{DFP} g_k^t g_k}{\gamma_k^{DFP} g_k^t g_k + g_{k+1}^t g_{k+1}} d_{k+1}^{CG}. \tag{57}$$

Démonstration. Pour (56) et (57) il suffit de remplacer ϕ_k respectivement par 1 et par 0 dans (32). En reprenant l'argument du corollaire 1, si f est une quadratique strictement convexe et H_0 est définie positive alors $\phi_k^c < 0$ pour tout k. Il s'agit ensuite d'une simple application du théorème 2 pour prouver que toutes les méthodes de la classe de Broyden avec $\phi_k \geq 0$ sont bien définies. □

Proposition 2 *Soit d_k^{SR1} la direction engendrée par l'algorithme de SR1 et soit γ_k^{SR1} le coefficient tel que $d_k^{SR1} = \gamma_k^{SR1} d_k^{CG}$. Sur une quadratique strictement convexe, si*

$$(s_k - H_k y_k)^t y_k \neq 0, \tag{58}$$

alors ϕ_k^{SR1} défini en (23) existe et

$$d_{k+1}^{SR1} = \frac{\left(\gamma_k^{SR1} - \alpha_k^{CG}\right) g_k^t g_k}{\left(\gamma_k^{SR1} - \alpha_k^{CG}\right) g_k^t g_k + g_{k+1}^t g_{k+1}} d_{k+1}^{CG}. \tag{59}$$

Démonstration. Nous pouvons obtenir (59) en remplaçant ϕ_k par (23). Avec (46) et (48) nous avons

$$(s_k - H_k y_k)^t y_k = \left(\alpha_k^{CG} - \gamma_k^{SR1}\right) g_k^t g_k - g_{k+1}^t g_{k+1}. \tag{60}$$

En utilisant (46) et (60) dans (23),

$$\phi_k^{SR1} = -\frac{\alpha_k^{CG} g_k^t g_k}{\left(\gamma_k^{SR1} - \alpha_k^{CG}\right) g_k^t g_k + g_{k+1}^t g_{k+1}} \tag{61}$$

Il s'en suit que

$$\begin{aligned} d_{k+1}^{SR1} &= \frac{\gamma_k^{SR1} g_k^t g_k + \phi_k^{SR1} g_{k+1}^t g_{k+1}}{\gamma_k^{SR1} g_k^t g_k + g_{k+1}^t g_{k+1}} d_{k+1}^{CG} \\ &= \frac{\gamma_k^{SR1} g_k^t g_k \left(\left(\gamma_k^{SR1} - \alpha_k^{CG}\right) g_k^t g_k + g_{k+1}^t g_{k+1}\right) - \alpha_k^{CG} g_{k+1}^t g_{k+1} g_k^t g_k}{\left(\gamma_k^{SR1} g_k^t g_k + g_{k+1}^t g_{k+1}\right)\left(\left(\gamma_k^{SR1} - \alpha_k^{CG}\right) g_k^t g_k + g_{k+1}^t g_{k+1}\right)} d_{k+1}^{CG} \\ &= \frac{\left(\gamma_k^{SR1} g_k^t g_k + g_{k+1}^t g_{k+1}\right)\left(\gamma_k^{SR1} - \alpha_k^{CG}\right) g_k^t g_k}{\left(\gamma_k^{SR1} g_k^t g_k + g_{k+1}^t g_{k+1}\right)\left(\left(\gamma_k^{SR1} - \alpha_k^{CG}\right) g_k^t g_k + g_{k+1}^t g_{k+1}\right)} d_{k+1}^{CG} \\ &= \frac{\left(\gamma_k^{SR1} - \alpha_k^{CG}\right) g_k^t g_k}{\left(\gamma_k^{SR1} - \alpha_k^{CG}\right) g_k^t g_k + g_{k+1}^t g_{k+1}} d_{k+1}^{CG}. \end{aligned} \tag{62}$$

□

Remarque 3 *Nous pouvons exprimer les coefficients γ_k^ϕ en fonction des longueurs de pas optimales (7) associées aux deux directions d_k^ϕ et d_k^{CG}, respectivement α_k^ϕ et α_k^{CG}. En effet, puisque les itérés s_k sont identiques aux deux méthodes, il s'en suit que $s_k = \alpha_k^{CG} d_k^{CG} = \alpha_k^\phi d_k^\phi$. D'où*

$$\gamma_k^\phi = \frac{\alpha_k^{CG}}{\alpha_k^\phi}. \tag{63}$$

Notons enfin que $\gamma_0^\phi = 1$ pour toutes les méthodes de la classe de Broyden pourvu que $H_0 = I$ et donc $d_0^\phi = d_0^{CG} = -g_0$.

Proposition 3 *Soient deux méthodes de la classe de Broyden convexe de paramètres $\phi_k^{(1)}$ et $\phi_k^{(2)}$ tels que $0 \leq \phi_k^{(1)} \leq \phi_k^{(2)} \leq 1$ pour tout k avec $H_0 = I$. Soient $\gamma_k^{(1)}$ et $\gamma_k^{(2)}$ définis par (63) pour les deux méthodes. Alors*

$$0 < \gamma_k^{(1)} \leq \gamma_k^{(2)} \leq 1 \tag{64}$$

Soient $\alpha_k^{(1)}$ et $\alpha_k^{(2)}$ les longueurs de pas optimales associées aux deux méthodes. Nous avons

$$\alpha_k^{(2)} \leq \alpha_k^{(1)} \quad si\ s_k^t A s_k > 0, \tag{65.a}$$

$$\alpha_k^{(1)} \leq \alpha_k^{(2)} \quad si\ s_k^t A s_k < 0. \tag{65.b}$$

Enfin, soient α_k^{CG} *la longueur de pas optimale associée à la méthode du gradient conjugué et* α_k^{ϕ} *la longueur de pas optimale associée à une méthode de la classe de Broyden convexe de paramètre* $0 \leq \phi_k \leq 1$ *avec* $H_0 = I$. *Nous avons*

$$0 < \alpha_k^{CG} \leq \alpha_k^{\phi} \quad si\ s_k^t A s_k > 0, \tag{66.a}$$

$$\alpha_k^{\phi} \leq \alpha_k^{CG} < 0 \quad si\ s_k^t A s_k < 0. \tag{66.b}$$

Démonstration. D'après la démonstration du théorème 2, $\gamma_{k+1}^{\phi} > 0$ si $\phi_k > \phi_k^c$ et $\gamma_k^{\phi} > 0$. Or d'après (55), $\phi_k^c < 0$ si $\gamma_k^{\phi} > 0$. Donc $\gamma_k^{\phi} > 0$ pour tout k si $\phi_k > 0$ et $\gamma_0^{\phi} > 0$. D'après (45), il vient que α_k^{ϕ} et α_k^{CG} sont alors du même signe et par transitivité nous avons également que $\alpha_k^{(1)}$ et $\alpha_k^{(2)}$ sont du même signe.

Soient $\gamma_{k+1}^{(1)}$ et $\gamma_{k+1}^{(2)}$ obtenus en remplaçant ϕ_k par $\phi_k^{(1)}$ et $\phi_k^{(2)}$ dans (51). Montrons par récurrence que pour tout k, $\gamma_k^{(1)} \leq \gamma_k^{(2)}$ si $0 \leq \phi_k^{(1)} \leq \phi_k^{(2)} \leq 1$. Tout d'abord nous avons $\gamma_0^{(1)} = \gamma_0^{(2)} = 1$ puisque $d_0 = -g_0$ est commun à toutes les méthodes de la classe de Broyden lorsque $H_0 = I$. Supposons que $\gamma_k^{(1)} \leq \gamma_k^{(2)}$ jusqu'à un rang $k \geq 0$ et montrons que c'est toujours vrai au rang $k+1$. Nous avons

$$\begin{aligned}
\gamma_{k+1}^{(1)} &= \frac{\gamma_k^{(1)} g_k^t g_k + \phi_k^{(1)} g_{k+1}^t g_{k+1}}{\gamma_k^{(1)} g_k^t g_k + g_{k+1}^t g_{k+1}} \\
&= 1 + \frac{g_{k+1}^t g_{k+1} \left(\phi_k^{(1)} - 1\right)}{\gamma_k^{(1)} g_k^t g_k + g_{k+1}^t g_{k+1}} \\
&= 1 - \frac{g_{k+1}^t g_{k+1} \left|\phi_k^{(1)} - 1\right|}{\gamma_k^{(1)} g_k^t g_k + g_{k+1}^t g_{k+1}}.
\end{aligned} \tag{67}$$

La dernière égalité nous vient du fait que $\phi_k^{(1)} \in [0,1]$. Comme $0 \leq \phi_k^{(1)} \leq \phi_k^{(2)} \leq 1$, nous avons

$$\left|\phi_k^{(1)} - 1\right| \geq \left|\phi_k^{(2)} - 1\right|. \tag{68}$$

Et comme par hypothèse de récurrence $0 < \gamma_k^{(1)} \leq \gamma_k^{(2)}$, nous avons

$$\gamma_k^{(1)} g_k^t g_k + g_{k+1}^t g_{k+1} \leq \gamma_k^{(2)} g_k^t g_k + g_{k+1}^t g_{k+1}. \tag{69}$$

Il s'en suit que

$$\gamma_{k+1}^{(1)} \leq 1 - \frac{g_{k+1}^t g_{k+1} \left|\phi_k^{(2)} - 1\right|}{\gamma_k^{(2)} g_k^t g_k + g_{k+1}^t g_{k+1}} = \gamma_{k+1}^{(2)}. \tag{70}$$

Par ailleurs, il est clair que $\gamma_k^{\phi} < 1$ pour tout $0 \leq \phi_k \leq 1$. Par récurrence, nous venons de montrer (64). Enfin, puisque

$$\gamma_k^{(1)} = \frac{\alpha_k^{CG}}{\alpha_k^{(1)}} \text{ et } \gamma_k^{(2)} = \frac{\alpha_k^{CG}}{\alpha_k^{(2)}}, \tag{71}$$

nous avons

$$\left|\alpha_k^{(2)}\right| \leq \left|\alpha_k^{(1)}\right|. \tag{72}$$

Nous obtenons (65) en notant que $\alpha_k^{(1)}$ et $\alpha_k^{(2)}$ sont du même signe que α_k^{CG} qui est du même signe que $s_k^t A s_k$. En effet, d'après (7) et (40),

$$\alpha_k^{CG} = \frac{g_k^t g_k}{d_k^t A d_k}. \tag{73}$$

On obtient enfin (66) en notant d'après (56) que $\alpha_k^{CG} = \alpha_k^{BFGS}$ et en remplaçant donc $\alpha_k^{(2)}$ par α_k^{CG} avec $\phi_k^{(2)} = 1$. □

La figure 1 illustre les différences de pas calculés par chacune des méthodes sur une quadratique non-convexe.

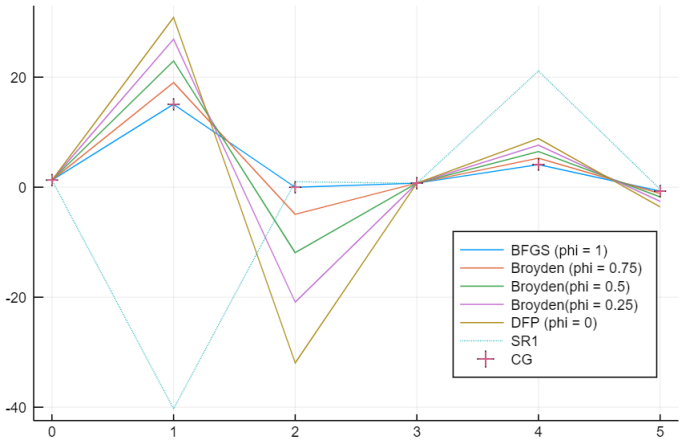

Figure 1: Évolution des longueurs de pas optimales α_k pour différentes méthodes de la classe de Broyden appliquées à une quadratique non-convexe en dimension 6. Les longueurs calculées par les méthodes de la classe de Broyden convexe sont ordonnées dans l'ordre décroissant par rapport au paramètre ϕ (en valeur absolue) tandis que celles générées par SR1 présentent un comportement erratique

1.4 Comportement des deux méthodes sur des problèmes mal conditionnés

En arithmétique exacte les méthodes de la classe de Broyden et le gradient conjugué génèrent les mêmes itérés. Cependant, la méthode du gradient conjugué souffre de difficultés numériques sur des problèmes mal conditionnés. En arithmétique inexacte, l'algorithme ne parvient plus à assurer la conjuguaison entre les itérés. Ceci est lié aux calculs réalisés dans la méthode d'orthogonalisation de Gram-Shmidt. C'est ce qu'on appelle la perte d'orthogonalité. Si le nouveau gradient g_{k+1} forme un angle trop faible avec les précédentes directions $d_0, ..., d_k$, la partie orthogonale de g_{k+1} par rapport à E_k va être très petite. Nous formons ainsi une base de vecteurs avec des éléments de plus en plus petits et nous perdons au fur et à mesure de l'information à cause de la précision numérique.

La méthode BFGS et les autres méthodes quasi-Newton étudiées sont bien plus robustes à la perte d'orthogonalité. Nous conservons en effet de l'information sur l'ensemble des directions d_k précédemment calculées et nous orthogonalisons donc d_{k+1} par rapport à chacune d'entre elles. Comme ces méthodes génèrent en théorie les mêmes itérés que ceux du gradient conjugué, nous gardons la bonne propriété de convergence en n itérations sur une quadratique strictement convexe. à l'inverse, la méthode du gradient conjugué va perdre la propriété de conjugaison entre les itérés générés et nous n'allons plus observer de terminaison quadratique.

Nous pouvons faire ces observations avec les figures 2 et 3. Nous comparons ici la méthode BFGS et la méthode du gradient conjugué sur une quadratique strictement convexe en dimension 30 avec une matrice A de conditionnement de l'ordre de 10^5.

L'algorithme de BFGS semble donc plus efficace et plus robuste au mauvais conditionnement de la matrice A. Cependant, le gros inconvénient de BFGS et des autres méthodes quasi-Newton est la lourdeur des calculs et l'utilisation importante de la mémoire lorsque l'on est en grande dimension. En effet, pour mettre à jour d_k, il faut tout d'abord réaliser un produit matrice-vecteur alors que le gradient conjugué effectue des additions entre deux vecteurs et des multiplications par des scalaires. Il faut en outre sauvegarder une matrice $n \times n$ avec BFGS alors que nous mettons directement à jour un vecteur de taille n avec le gradient conjugué. Nous devons donc considérer de manière relative la bonne performance de BFGS au regard de l'espace mémoire et du temps de calcul demandé. Doubler

le nombre d'itérations avec le gradient conjugué comparativement à BFGS ne veut absolument pas dire que l'on double le temps de calcul.

Nous verrons plus loin que la méthode L-BFGS présente une alternative intéressante à ces deux méthodes concernant l'espace mémoire occupé et le temps de calcul nécessaire. La méthode L-BFGS vise en effet à reconstruire le produit $H_k g_k$ de manière itérative à partir d'un nombre restreint d'informations sur les itérations précédentes.

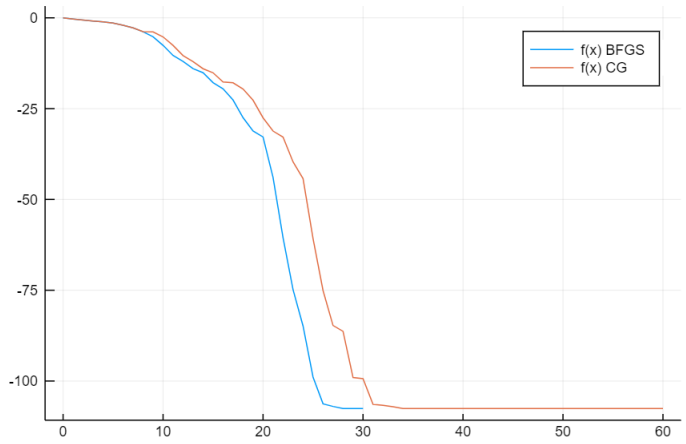

Figure 2: Évolution de $f(x_k)$ en fonction des itérations sur une quadratique convexe en dimension 30. L'algorithme BFGS parvient à retrouver le minimum de f en n itérations alors que le gradient conjugué peine à faire baisser la norme du gradient lorsqu'il se rapproche du minimum. De plus, les itérés calculés semblent être les mêmes mais le gradient conjugué montre des ≪décrochages≫ à plusieurs endroits : il n'améliore pas la solution d'un itéré sur l'autre et prend donc du retard sur la solution trouvée par BFGS

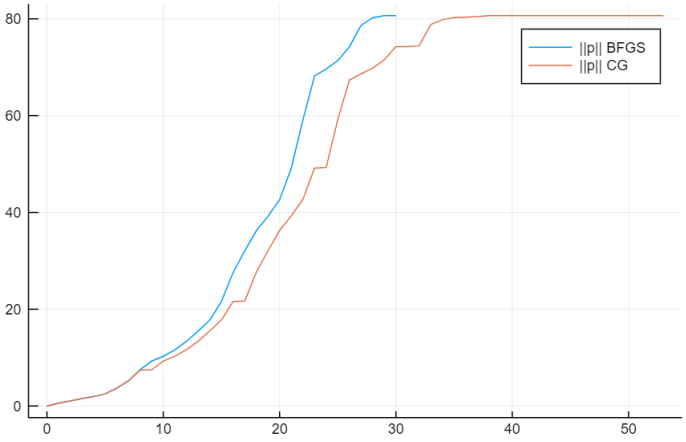

Figure 3: Évolution de la norme 2 de p_k en fonction des itérations (dimension 30). Avec les deux méthodes, $\|p_k\|$ augmente ce qui nous indique que nous pouvons les utiliser toutes les deux dans un algorithme de régions de confiance (voir section 2.2)

2 Utilisation de la méthode BFGS avec une contrainte de région de confiance

2.1 L'algorithme de régions de confiance

Dans ce chapitre nous allons étudier l'application d'une recherche linéaire exacte avec BFGS et CG conjointement à l'utilisation d'un algorithme de régions de confiance.

Supposons que la fonction à minimiser soit une fonction f qui n'est pas forcément quadratique ni même convexe mais qui admet au moins des minimums globaux et/ou locaux que nous allons chercher à calculer. Supposons par ailleurs que f soit deux fois différentiable. L'algorithme de régions de confiance génère à partir du gradient et de la hessienne de f un développement de Taylor d'ordre 2 au point courant x_k,

$$q_k(p) = f(x_k) + \nabla f(x_k)^t p + \tfrac{1}{2} p^t \nabla^2 f(x_k) p. \tag{74}$$

Nous appelons cette fonction de p le modèle de f au point x_k.

Nous nous donnons ensuite une boule de rayon Δ appelée la région de confiance à l'intérieur de laquelle nous supposons que l'approximation de f par q est suffisamment acceptable

$$\Omega(\Delta) := \{p \in \mathbb{R}^n \ \mid \ \|p\| \leq \Delta\}. \tag{75}$$

Nous avons donc un modèle quadratique à minimiser avec une contrainte dite de région de confiance. Notons que cette quadratique n'est pas forcément convexe et qu'il faudra dans ce cas définir un protocole pour gérer la courbure négative. Le sous-problème

$$\min_{p \in \mathbb{R}^n} \{q_k(p) \ \mid \ p \in \Omega(\Delta)\} \tag{$\mathcal{SP}2$}$$

admettra toujours un minimum puisque le domaine considéré est borné.

Il est possible avec le théorème de Lagrange de trouver un minimum global de ce sous-problème mais des algorithmes de régions de confiance efficaces ne recherchent en réalité qu'une bonne minimisation de q compte tenu de la contrainte de région de confiance.

Une des idées est d'effectuer une recherche linéaire dans le domaine $\Omega(\Delta)$ en partant de son centre. Si le minimum sans contrainte se situe à l'intérieur du domaine, nous trouvons évidemment le minimum du sous-problème ($\mathcal{SP}2$). Si celui-ci se situe à l'extérieur, nous nous arrêtons à la frontière dans la dernière direction donnée par la recherche linéaire. Il ne s'agit certainement pas de la solution du problème sous contrainte mais nous allons voir dans la section 2.2 que les conditions sont réunies pour obtenir la convergence de l'algorithme. Le vecteur p_k résultant de ce sous-problème de minimisation permet de mettre à jour le point courant $x_{k+1} = x_k + p_k$ et nous calculons un nouveau modèle $q_{k+1}(p)$ à partir des nouvelles valeurs de f et ∇f au point x_{k+1}.

Enfin, le rayon Δ est variable. Selon un critère donné en paramètre, nous regardons si l'approximation de f par q_k au nouveau point est suffisamment bonne. Nous calculons pour se faire le ratio

$$\rho_k = \frac{f(x_{k+1}) - f(x_k)}{q_k(p_k) - q_k(0)}. \tag{76}$$

Si le ratio ρ_k est considéré comme très bon nous augmentons Δ dans une limite convenable. S'il est mauvais nous diminuons Δ et nous nous donnons un deuxième critère pour juger s'il faut également mettre à jour x_k ou non.

L'algorithme 2 résume les précédentes notions énoncées.

Algorithm 2 Algorithme de régions de confiance

procedure TR($f(\cdot)$, $\nabla f(\cdot)$, x_0, $\hat{\Delta} > 0$, $\varepsilon > 0$, $N \in \mathbb{N}^*$, $\mu \in [1/4, 1/2]$, $\eta \in [0, \mu[$)
  $\Delta \leftarrow \hat{\Delta}/2$
  $k \leftarrow 0$
  $x_k \leftarrow x_0$
  $f_k \leftarrow f(x_k)$
  $g_k \leftarrow \nabla f(x_k)$
  while $k \leq N$ **and** $\|g_k\| > \varepsilon$ **do**
    $p_k \leftarrow \text{LS}(\nabla f,\ x_k,\ \Delta)$
    $b_k \leftarrow \text{DIFF}(\nabla f,\ x_k,\ p_k)$ ▷ Calcul de $\nabla^2 f_k p_k$
    $\rho \leftarrow (f(x_k + p_k) - f_k)/(p_k^t g_k + 0.5 p_k^t b_k)$
    if $\rho \leq \mu$ **then** ▷ Mauvaise approximation : on réduit le rayon
      $\Delta \leftarrow \Delta/4$
    else if $\rho \geq 1 - \mu$ **then** ▷ Très bonne approximation : on agrandit le rayon
      $\Delta \leftarrow \min(2\Delta,\ \hat{\Delta})$
    end if
    if $\rho \geq \eta$ **then** ▷ Mise à jour du point courant si bonne approximation
      $x_k \leftarrow x_k + p_k$
      $f_k \leftarrow f(x_k)$
      $g_k \leftarrow \nabla f(x_k)$
    end if
  end while
  return x_k
end procedure

Remarque 4 *Il n'est à aucun moment nécessaire de calculer le hessien. Il existe des outils de différentiations automatique qui permettent d'obtenir le produit $\nabla^2 f(x_k)p$ à partir du gradient. En effet, posons ϵ proche de 0. Un développement de Taylor sur le gradient donne*

$$\nabla f(x_k + \epsilon p) = \nabla f(x_k) + \epsilon \nabla^2 f(x_k)p + O(\epsilon^2 \|p\|^2). \tag{77}$$

Il suffit alors de récupérer la partie en ϵ pour pouvoir évaluer q_k en p.

Pour réaliser une recherche linéaire exacte avec BFGS et avec le gradient conjugué, nous n'avons besoin que de ces produits hessien-vecteurs. Étant donné le bon comportement de BFGS et de CG sur des quadratiques convexes, il va être intéressant d'étudier l'utilisation ces deux méthodes pour la résolution d'un sous-problème de région de confiance.

2.2 Résolution d'un sous-problème de région de confiance avec la méthode BFGS

Dans cette sous-section, nous cherchons à adapter la méthode BFGS à une contrainte de région de confiance. Notons s_k les itérés générés par BFGS dans le sous-problème ($\mathcal{SP}2$). L'objectif ici est d'appliquer la recherche linéaire avec la mise à jour de BFGS tant que l'on est à l'intérieur de l'ensemble $\Omega(\Delta)$ et de trouver une stratégie pour calculer un itéré s_k dans le cas où l'on sortirait de cette région de confiance. Considérons tout d'abord que le modèle $q(p)$ est une quadratique convexe. Avec $A = \nabla^2 q = \nabla^2 f(x)$ et $g_k = \nabla q(p_k) = \nabla f(x) + \nabla^2 f(x) p_k$

Théorème 3 *Soient $d_k = -H_k g_k$ la direction de descente définie par la méthode BFGS et $\alpha_k = -\dfrac{g_k^t d_k}{d_k^t A d_k}$ la longueur de pas optimale du sous-problème ($\mathcal{SP}1$). Nous appliquons la méthode BFGS avec $H_0 = I$ sur une quadratique strictement convexe q. On définit*

$$p_k = \sum_{i=0}^{k-1} s_i, \tag{78}$$

$$\text{où } s_k = \alpha_k d_k.$$

Alors la norme $\|p_k\|$ est croissante et la fonction q décroît le long du chemin formé par les itérés s_k.

Preuve de la croissance de $\|p_k\|$. D'après le théorème 2, $d_k^{BFGS} = d_k^{CG} = -g_k + \beta_k d_{k-1}$, avec $\beta_k = \dfrac{g_k^t g_k}{g_{k-1}^t g_{k-1}}$. Comme nous effectuons une recherche linéaire exacte, $s_i^t g_k = 0$ pour $i = 0, ..., k-1$. Nous avons donc

$$\begin{aligned} p_k^t s_k &= \sum_{i=0}^{k-1} s_i^t s_k \\ &= \sum_{i=0}^{k-1} s_i^t \alpha_k \beta_k \beta_{k-1} ... \beta_{i+1} d_i \\ &= \sum_{i=0}^{k-1} \alpha_i \alpha_k \beta_k \beta_{k-1} ... \beta_{i+1} \|d_i\|^2 \\ &= \sum_{i=0}^{k-1} \alpha_i \alpha_k \frac{g_k^t g_k}{g_i^t g_i} \|d_i\|^2. \end{aligned} \tag{79}$$

Par ailleurs q est supposée strictement convexe donc $\alpha_i > 0$ pour $i = 0, ..., k$. D'où $p_k^t s_k > 0$. Il s'en suit que

$$\|p_{k+1}\|^2 = \|p_k + s_k\|^2 = \|p_k\|^2 + 2p_k^t s_k + \|s_k\|^2 > \|p_k\|^2. \tag{80}$$

□

Nous venons ainsi de montrer que la suite $\{\|p_k\|\}_{k\geq 1}$ est croissante et donc que les itérés s_k s'éloignent à chaque étape du centre de la région de confiance $\Omega(\Delta)$. Comme nous savons d'après le théorème 2 que la méthode BFGS avec recherche linéaire exacte converge en au plus n itérations sur une quadratique convexe, il y a alors deux possibilités d'arrêt de l'algorithme en tenant compte de la contrainte de région de confiance. Soit $q(p_k)$ converge vers le minimum, dans le cas où celui-ci se trouve à l'intérieur de $\Omega(\Delta)$, soit il existe k tel que p_{k+1} est en dehors de la région de confiance et il faut alors trouver le vecteur p^* sur la frontière entre p_k et p_{k+1}.

Décroissance de q. Pour un $k \in \mathbb{N}^*$ donné, posons la fonction $h : \mathbb{R} \to \mathbb{R}$

$$h(\alpha) = q(p_k + \alpha d_k) = \frac{1}{2}(p_k + \alpha d_k)^t A(p_k + \alpha d_k) + b^t(p_k + \alpha d_k). \tag{81}$$

Il s'en suit que

$$h'(\alpha) = d_k^t A(p_k + \alpha d_k) + b^t d_k, \tag{82.a}$$
$$h''(\alpha) = d_k^t A d_k. \tag{82.b}$$

Étant donné que $A \succ 0$, h"$(\alpha) > 0$ et donc $h'(\alpha)$ est croissante. Nous savons donc que $h'(\alpha)$ est négative pour tout $\alpha < \alpha_k = -\dfrac{g_k^t d_k}{d_k^t A d_k}$.

Or il se trouve que α_k est la longueur de pas optimale du sous-problème ($\mathcal{SP}1$). Ainsi le chemin constitué des itérés $\{d_k\}_{k\in\mathbb{N}}$ est de la forme

$$\mathcal{C}(k, \alpha) = p_k + \alpha d_k, \ k \in \mathbb{N}, \ \alpha \in [0, \alpha_k], \tag{83}$$

avec p_k défini en (78) et en considérant $p_0 = 0$.

Puisque $\alpha < \alpha_k$ pour tout k, $h'(\alpha) < 0$, et la quadratique q décroît le long du chemin $\mathcal{C}(k, \alpha)$. □

Le résultat précédent nous indique que nous pouvons choisir, s'il existe, le point au croisement entre $\mathcal{C}(k,\alpha)$ et $\Omega(\Delta)$ comme solution approchée du sous-problème ($\mathcal{SP}2$). Nous sommes alors assurés d'avoir fait décroître au mieux q dans la dernière direction d_k donnée, tout en respectant la contrainte de région de confiance.

Résultat 1 *Supposons que le minimum atteint par q ne se situe pas dans la région de confiance $\Omega(\Delta)$. Nous reprenons la définition (78) de p_k et de s_k. Soit r le premier entier tel que $\|p_{r+1}\| > \Delta$. Nous savons d'après le théorème 3 qu'il existe $\bar{\tau} \in]0,1[$ tel que*

$$\|p_r + \bar{\tau} s_r\| = \Delta. \tag{84}$$

Tout d'abord élevons cette norme au carré. L'équation (84) se résout alors en trouvant la racine positive de

$$\tau^2 \|s_r\|^2 + 2\tau p_r^t s_r + (\|p_r\|^2 - \Delta^2). \tag{85}$$

Cette équation possède deux solutions réelles distinctes puisque $(\|p_r\|^2 - \Delta^2) < 0$ par hypothèse. Cela est cohérent avec le fait que suivant la direction donnée par s_r il est possible d'atteindre la frontière de $\Omega(\Delta)$ dans le sens positif et dans le sens négatif. Les deux racines nous sont données par

$$\tau_{1,2} = \frac{-p_r^t s_r \pm \sqrt{(p_r^t s_r)^2 + \|s_r\|^2(\Delta^2 - \|p_r\|^2)}}{\|s_r\|^2}. \tag{86}$$

Il est clair que $\sqrt{(p_r^t s_r)^2 + \|s_r\|^2(\Delta^2 - \|p_r\|^2)} > p_r^t s_r$. Il existe donc une unique racine positive qui nous est donnée par

$$\bar{\tau} = \frac{-p_r^t s_r + \sqrt{(p_r^t s_r)^2 + \|s_r\|^2(\Delta^2 - \|p_r\|^2)}}{\|s_r\|^2}. \tag{87}$$

Comme voulu, $\bar{\tau} \in\ ]0,1[$. En effet, posons

$$h(\tau) = \tau^2 \|s_r\|^2 + 2\tau p_r^t s_r + \|p_r\|^2. \tag{88}$$

On a alors

$$h'(\tau) = 2\tau \|s_r\|^2 + 2p_r^t s_r, \tag{89.a}$$
$$h''(\tau) = 2\|s_r\|^2. \tag{89.b}$$

Comme $h''(\tau) > 0$, nous savons que $h'(\tau)$ est croissante et donc positive à partir de $\hat{\tau} = -\dfrac{p_r^t s_r}{\|s_r\|^2} < 0$. Enfin, puisque $h(1) = \|p_r + s_r\|^2 > \Delta^2 = h(\bar{\tau})$, et comme h est croissante à partir de $\hat{\tau}$, nous avons la relation $\hat{\tau} < 0 < \bar{\tau} < 1$.

Nous pouvons donc établir un protocole à suivre pour appliquer la méthode BFGS à un sous-problème de région de confiance. Tant que le minimum n'est pas atteint, nous choisissons s_k et p_{k+1} tels qu'indiqués par (78). Si $\|p_{k+1}\| > \Delta$, nous remplaçons p_{k+1} par $p_k + \bar{\tau} s_k$ et arrêtons la recherche linéaire.

Remarque 5 *Rien ne nous assure ici que nous avons bien trouvé le minimum du sous problème ($\mathcal{SP}2$), mais cela n'a que très peu d'importance. En effet, nous cherchons en réalité à trouver la valeur minimale de q suivant les directions d_k données par la méthode BFGS. Puisque nous sommes assurés que p_k s'éloigne du centre de $\Omega(\Delta)$, nous savons que nous nous sommes rapprochés de son minimum, comme désiré. La convergence des méthodes de régions de confiance repose sur la condition de décroissance suffisante, que le processus ci-dessus nous assure.*

2.3 Méthode tronquée pour des quadratiques non-convexes

Le but de notre étude est ici de généraliser la précédente recherche linéaire à la résolution de problèmes non-convexes. Lorsque le hessien de la fonction objectif n'est plus défini positif, nous pouvons obtenir dans certaines directions des courbures négatives. Une quadratique non-convexe, c'est à dire une quadratique possédant des courbures négatives, n'est pas minorée et la solution du problème sans contrainte ne permet donc pas de se rapprocher du minimum de f lorsque nous revenons à l'algorithme de régions de confiance. Si nous cherchons à minimiser une fonction non-convexe f via l'algorithme de régions de confiance, nous allons alors générer des modèles quadratiques q non-convexes en certains points. L'utilisation d'une contrainte de région de confiance permet donc de borner le problème. L'idée est de réaliser la recherche linéaire tant que les directions générées produisent des courbures positives puis de plonger à la frontière de $\Omega(\Delta)$ dans la direction de descente donnant une courbure négative.

Étant donné que ni A ni B_k ne sont assurés d'être définies positives, quatre cas de figures sont à considérer en fonction de si la courbure est négative ou positive ou de si d_k est une direction ascendante ou descendante. Lorsque la courbure est positive, la longueur de pas optimale (7) est positive ou négative si d_k est respectivement une direction descendante ou ascendante. Ainsi, nous sommes assurés que $s_k = \alpha_k d_k$ est une direction de descente et nous n'avons rien à changer. Enfin si la courbure est négative, nous mettons à jour α_k de sorte à forcer le pas p_k à sortir de $\Omega(\Delta)$. Le signe de α_k est alors fonction du signe de $g_k^t d_k$ puisque nous cherchons à obtenir une direction de descente s_k qui vérifie $g_k^t s_k < 0$. Nous avons donc

$$\alpha_k := \begin{cases} -\dfrac{g_k^t d_k}{d_k^t A d_k} & \text{si } d_k^t A d_k > 0 \\ -\dfrac{\text{sign}(g_k^t d_k) 2\Delta}{\|d_k\|} & \text{si } d_k^t A d_k \leq 0 \end{cases} \tag{90}$$

Remarque 6 *Avec BFGS nous aurons toujours $g_k^t d_k^{BFGS} < 0$ puisque, par application du corollaire 3, $g_k^t d_k^{BFGS} = -\|g_k\|^2$. C'est également vrai avec DFP puisque, d'après la démonstration du théorème 2, $\gamma_k^{DFP} > 0$ si $\gamma_{k-1}^{DFP} > 0$, or $\gamma_0^{DFP} = 1$. Il s'en suit que $g_k^t d_k^{DFP} = -\gamma_k^{DFP}\|g_k\|^2 < 0$. Cela n'est pas forcément vrai avec SR1, étant donné que d'après (59), le signe de γ_k^{SR1} peut être négatif. Nous avons remarqué numériquement que cette dernière méthode a tendance à trouver des directions ascendantes, telle que $g_k^t d_k > 0$, avant même de rencontrer une courbure négative. Et ceci est vrai d'autant plus lorsque A est indéfinie. Notons tout de même que α_k étant choisie du signe opposé à celui de $g_k^t d_k$, s_k reste toujours une direction de descente.*

Remarque 7 *Il peut être intéressant de voir α_k comme étant le ratio entre la courbure calculée avec les matrices B_k et la vraie courbure de la quadratique q. En effet, nous avons le résultat suivant*

$$d_k^t B_k d_k = -d_k^t B_k H_k g_k = -d_k^t g_k. \tag{91}$$

Donc

$$\alpha_k = -\frac{g_k^t d_k}{d_k^t A d_k} = \frac{d_k^t B_k d_k}{d_k^t A d_k}. \tag{92}$$

Tout d'abord, cela nous montre que la longueur de pas optimale α_k se rapproche de 1 lorsque l'approximation de A^{-1} par H_k est bonne. En fait, le pas α_k fait figure de correcteur pour adapter le déplacement calculé avec BFGS au pas de Newton (4). Il est ainsi concevable d'avoir $\alpha_k < 0$ lorsque la courbure prédite et celle de la fonction objectif ne sont pas du même signe. Si $|\alpha_k| < 1$, cela signifie que la vraie courbure est plus grande que celle calculée et donc qu'il faut faire un plus petit déplacement pour atteindre le minimum. Dans le cas contraire, si $|\alpha_k| > 1$, il faut réaliser un plus gros déplacement pour atteindre le minimum étant donné que la vraie courbure est plus faible.

2.4 Implémentation de l'algorithme

Nous avons donc implémenté l'algorithme de régions de confiance avec la recherche linéaire tronquée décrite dans les deux sections plus haut. Il est possible d'appeler les différentes méthodes quasi-Newton ainsi que la méthode du gradient conjugué. L'algorithme 3 fait figure de récapitulatif des différents points abordés dans les deux précédentes sous-sections :

- Sans détection de courbure négative, la recherche linéaire s'arrête lorsque $\|\nabla q(p_k)\| < \epsilon$ ou lorsque $\|p_k\| > \Delta$. Dans ce dernier cas, on calcule τ tel que $\|p_k + \tau s_k\| = \Delta$. L'algorithme de régions de confiance s'arrête quant à lui lorsque $\|\nabla f(x_k)\| < \epsilon$.

- La valeur de $\alpha_k = -\text{sign}(g_k^t d_k) 2\Delta / \|d_k\|$ nous assure d'avoir une norme plus grande que le diamètre de $\Omega(\Delta)$ et c'est en sortant de la région de confiance que l'on calcule finalement un pas sur la frontière.

- Nous utilisons une fonction de différentiation automatique pour calculer les produits $\nabla^2 f(x_k)p$ que nous notons $\text{diff}(\nabla f, x_k, p)$. Grâce aux relations $y_k = As_k = \alpha_k A d_k$ et $g_{k+1} = y_k + g_k$, nous n'appelons la fonction diff qu'une seule fois par itération de la fonction LineSearch, au moment de calculer la courbure $d_k^t A d_k$.

- Concernant l'algorithme de régions de confiance, nous définissons deux paramètres $\mu \in [1/4, 1/2]$ et $\eta \in [0, \mu[$ qui nous permettent de juger de la bonne approximation de f par q. Le paramètre μ gère le rayon de $\Omega(\Delta)$ et η nous indique si nous pouvons mettre à jour x_k. Comme $\eta < \mu$, nous pouvons juger que l'approximation n'est pas bonne et décider de réduire le rayon Δ mais en conservant tout de même la mise à jour de x_k.

Algorithm 3 Recherche linéaire sous contrainte de région de confiance avec courbure négative

procedure LS($\nabla f,\ x_0,\ \epsilon > 0,\ N \in \mathbb{N}^*$)
  $k \leftarrow 0$
  $H_k \leftarrow I$
  $p_k \leftarrow 0$
  $g_k \leftarrow \nabla f(x_0)$
  while $\|g_k\| > \epsilon$ **and** $k \leq N$ **do**
    $H_k \leftarrow \text{update}(s_k,\ y_k,\ H_k)$ ▷ Laisser $H_k = I$ à la première itération
    $d_k \leftarrow -H_k g_k$
    $b_k \leftarrow \text{DIFF}(\nabla f,\ x_0,\ d_k)$ ▷ Calcul de Ad_k grâce au gradient
    if $d_k^t b_k \leq 0$ **then** ▷ Courbure négative
      $\alpha_k \leftarrow -\text{sign}(g_k^t d_k) 2\Delta / \|d_k\|$
    else
      $\alpha_k \leftarrow -g_k^t d_k / d_k^t b_k$
    end if
    $s_k \leftarrow \alpha_k d_k$
    $p_k \leftarrow p_k + s_k$
    if $\|p_k\| \geq \Delta$ **then** ▷ Sortie de la région de confiance
      $p_k \leftarrow p_k - s_k$
      Calculer (87) tel que $\|p_k + \tau s_k\| = \Delta$
      return $p_k + \tau s_k$
    end if
    $y_k \leftarrow \alpha_k b_k$ ▷ $y_k = g_{k+1} - g_k = As_k = \alpha_k A d_k$
    $g_k \leftarrow g_k + y_k$ ▷ $y_k = g_{k+1} - g_k$
    $k \leftarrow k + 1$
  end while
  return p_k
end procedure

3 Extension de notre étude à la méthode L-BFGS

3.1 Une méthode quasi-Newton à mémoire limitée

L'algorithme L-BFGS est une méthode quasi-Newton à mémoire limitée. Au lieu de sauvegarder une matrice H_k de taille $n \times n$, l'idée est de ne conserver qu'une quantité limitée m de couples (s_i, y_i) afin de reformer une approximation du produit $H_k g_k$ à chaque itération. Notons

$$V_k = I - \rho_k y_k s_k^t, \quad \text{et} \quad \rho_k = 1/y_k^t s_k. \tag{93}$$

Dans [7], il est montré pour $m = k$ et $H_k^0 = H_0$, que la matrice

$$\begin{aligned} H_k = &\left(V_{k-1}^t ... V_{k-m}^t\right) H_k^0 \left(V_{k-m} ... V_{k-1}\right) \\ &+ \rho_{k-m} \left(V_{k-1}^t ... V_{k-m+1}^t\right) s_{k-m} s_{k-m}^t \left(V_{k-m+1} ... V_{k-1}\right) \\ &+ \rho_{k-m+1} \left(V_{k-1}^t ... V_{k-m+2}^t\right) s_{k-m+1} s_{k-m+1}^t \left(V_{k-m+2} ... V_{k-1}\right) \\ &+ \rho_{k-1} s_{k-1} s_{k-1}^t \end{aligned} \tag{94}$$

est exactement celle calculée par la méthode BFGS. Si nous ne conservons qu'une quantité plus faible de couples (s_i, y_i), nous allons perdre de l'information nécessaire à la reconstruction de H_k mais nous espérons avoir gardé les informations les plus conséquentes pour déterminer une direction de descente $d_k = -H_k g_k$.

En réalité, la méthode L-BFGS appliquée à une quadratique possède les mêmes propriétés que les méthodes de BFGS et du gradient conjugué.

Théorème 4 *Soit une quadratique $f(x) = \frac{1}{2} x^t A x + b^t x$. Si nous effectuons une recherche linéaire exacte avec la méthode L-BFGS pour n'importe quel $m \geq 1$ et avec $H_k^0 = I$. Si de plus $s_k^t y_k \neq 0$ pour tout k. Alors les itérés $d_k = -H_k g_k$ sont les mêmes que ceux générés par les méthodes de BFGS et du gradient conjugué.*

Démonstration. La démonstration se fait par récurrence. Comme pour le théorème 1, nous cherchons à montrer que les itérés d_k générés par l'algorithme L-BFGS sont identiques à ceux générés par le gradient conjugué (40). Nous remarquons que pour $k = 0$, $d_0 = d_0^{CG} = -g_0$. Supposons que l'égalité ait lieu jusqu'à un rang $k \geq 1$ et montrons qu'elle a également lieu pour l'indice $k+1$ si $g_{k+1} \neq 0$. Sous l'hypothèse de récurence, les itérés $x_1, .., x_{k+1}$ de L-BFGS sont identiques à ceux générés par le gradient conjugué. Dès lors $s_i^t g_{k+1} = 0$ pour $i = 0, ..., k$. Nous avons donc

$$V_i g_{k+1} = \left(I - \rho_i y_i s_i^t\right) g_{k+1} = g_{k+1}, \ i = 0, ..., k. \tag{95}$$

Nous avons également la relation $g_i^t g_k = 0$ pour $i = 0, ..., k-1$, ce qui nous donne

$$V_i^t g_{k+1} = \left(I - \rho_i s_i y_i^t\right) g_{k+1} = g_{k+1}, \ i = 0, ..., k-1. \tag{96}$$

En appliquant ces deux relations à (94), nous avons,

$$H_{k+1} g_{k+1} = \left(V_k^t ... V_{k+1-m}^t\right) H_{k+1}^0 g_{k+1}. \tag{97}$$

Avec $H_{k+1}^0 = I$, il s'en suit que

$$\begin{aligned} -H_{k+1} g_{k+1} &= -V_k^t g_{k+1} \\ &= -g_{k+1} + \rho_k s_k \left(g_{k+1}^t - g_k^t\right) g_{k+1} \\ &= -g_{k+1} + \frac{g_{k+1}^t g_{k+1}}{y_k^t s_k} s_k \\ &= -g_{k+1} + \frac{g_{k+1}^t g_{k+1}}{y_k^t d_k} d_k \end{aligned} \tag{98}$$

Or par hypothèse de récurrence $d_k = d_k^{CG}$. Donc d'après (46) et (40),

$$d_{k+1} = -g_{k+1} + \frac{g_{k+1}^t g_{k+1}}{g_k^t g_k} d_k^{CG} = d_{k+1}^{CG}. \tag{99}$$

Ceci termine la preuve par récurrence. □

Remarque 8 *Nous pouvons interpréter ce résultat de la manière suivante. Dans la méthode BFGS, seule le dernier couple (s_k, y_k) est utile au calcul d'une direction de descente car grâce à l'utilisation de la longueur de pas optimale α_k, les éléments liés aux itérations d'avant sont orthogonaux au gradient. C'est ainsi que la méthode BFGS, qui à première vue effectue beaucoup de calculs, retrouve les mêmes résultats que le gradient conjugué qui effectue des additions et des multiplications sur le gradient à partir du dernier d_{k-1} sauvegardé. Seulement, nous avons vu qu'en grande dimension le gradient conjugué souffre de pertes d'orthogonalité et que les informations sur les itérations précédentes sont en réalité bien utiles pour corriger les problèmes d'arrondis.*

L'algorithme 4 nous montre une façon d'implémenter la méthode L-BFGS. Nous pouvons voir qu'il est possible de calculer de manière itérative le vecteur $H_k g_k$ sans faire de multiplications matricielles lourdes. La matrice H_k^0 est souvent choisie comme étant un multiple de l'identité mais il peut s'agir d'une autre matrice symétrique définie positive, souvent diagonale, qui peut aider dans sa construction à sauvegarder de l'information sur les précédentes itérations. Nous perdons alors la propriété de conjugaison mais son utilisation peut permettre des améliorations si nous n'avons pas accès au calcul de la longueur de pas optimale.

Algorithm 4 Implémentation de L-BFGS en deux boucles [7]

procedure LBFGS(g_k, $H_0, \{s_i, y_i\}_{i=k-m,\dots,k-1}$)
  $r \leftarrow g_k$
  for $i = k-1, k-2, \dots, k-m$ **do**
    $\alpha_i \leftarrow \rho_i s_i^t q$
    $q \leftarrow q - \alpha_i y_i$
  end for
  $r \leftarrow H_k^0 q$
  for $i = k-m, k-m+1, \dots, k-1$ **do**
    $\beta \leftarrow \rho_i y_i^t r$
    $r \leftarrow r + s_i(\alpha_i - \beta)$
  end for
  return r
end procedure

3.2 Perte d'orthogonalité des différentes méthodes

Pour comparer les différentes méthodes, nous générons aléatoirement des quadratiques convexes mal-condition-nées sur lesquelles nous effectuons une recherche linéaire exacte.

Sur un problème en dimension $n = 400$ avec un conditionnement de A de l'ordre de 10^8, nous comparons les algorithmes du gradient conjugué (la version définie par (26)) et de BFGS avec l'algorithme L-BFGS pour différentes valeurs de m exprimées en proportion de n. Pour $m = n$, L-BFGS retrouve la terminaison quadratique tout comme BFGS. Pour $m = 1$, L-BFGS obtient des résultats semblables à ceux du gradient conjugué. La figure 4 nous montre l'évolution de la norme du gradient en fonction des itérations.

Nous avons mis en place un test numérique pour étudier la perte d'orthogonalité. Soit P_k la matrice rectangulaire de taille $n \times k$ qui contient les k premiers itérés d_i normalisés. Si les $\{d_i\}_{i\geq 0}$ forment une base conjuguée, nous devrions avoir

$$P_k^t A P_k = I. \tag{100}$$

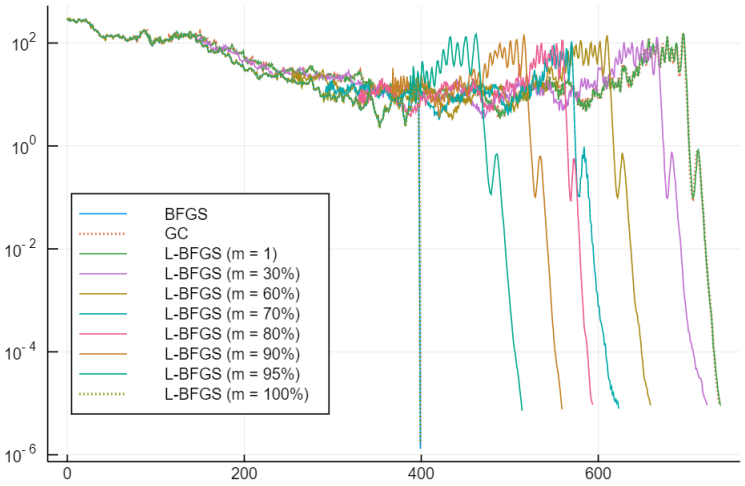

Figure 4: évolution de la norme 2 du gradient pour les différentes méthodes (dimension 400). L'algorithme de BFGS retrouve la solution en n itérations alors que les autres méthodes en font beaucoup plus (environ 300 de plus pour le gradient conjugué). Il est intéressant de noter qu'augmenter m permet de diminuer de manière quasi-systématique le nombre d'itérations effectuées par L-BFGS

Comme cette matrice est symétrique, notons U_k la matrice triangulaire strictement supérieure et D_k la matrice diagonale telles que

$$P_k^t A P_k = U_k^t + D_k + U_k. \tag{101}$$

L'identité (100) revient à dire que $U_k = 0$ et $D_k = I$. Une manière de calculer la conjugaison des itérés serait donc de vérifier que $\|U_k\| = 0$. Une autre méthode plus efficace pour calculer la perte d'orthogonalité [8] est de calculer la norme

$$\| (I + U_k)^{-1} U_k \|. \tag{102}$$

Cette quantité vaut 0 lorsque les itérés d_k sont conjugués deux à deux et croît lorsque l'on observe une perte d'orthogonalité entre les itérés. Nous pouvons retrouver les résultats de ces tests avec la figure 5.

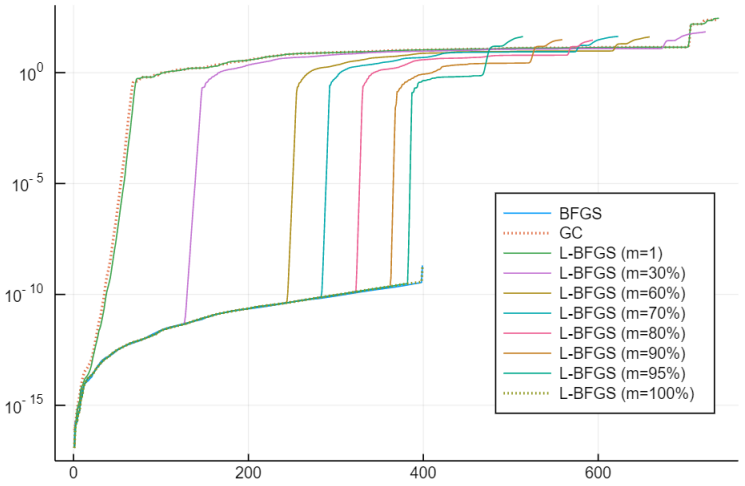

Figure 5: évolution de la perte d'orthogonalité mesurée avec $\|(U+I)^{-1}U\|$ (dimension 400). Hormis BFGS et L-BFGS avec $m = n$, chaque méthode perd tôt ou tard la conjuguaison entre les itérés. Notamment dès lors que $k > m$

Une remarque importante est qu'il faut augmenter m de manière conséquente pour obtenir de réelles améliorations concernant la perte d'orthogonalité. Cependant, il est intéressant de comparer le

nombre d'itérations au temps de calcul nécessaire et à l'espace mémoire utilisé. Il existe certainement un meilleur des mondes entre BFGS, qui converge en n itérations mais qui requière des calculs coûteux, et le gradient conjugué, qui effectue des calculs moins coûteux mais plus nombreux.

Enfin, il est utile de noter avec la figure 6 que les principaux efforts pour faire diminuer la valeur de f ont été réalisés dans les premières itérations. Cela semble nous indiquer qu'il peut être efficace dans un algorithme de régions de confiance d'utiliser peu de stockage ; étant donné que la frontière de la région de confiance se rencontre souvent en nettement moins de n itérations. Avec un rayon Δ suffisamment petit, il n'y aurait pas eu de grande différence à utiliser L-BFGS plutôt que L-BFGS.

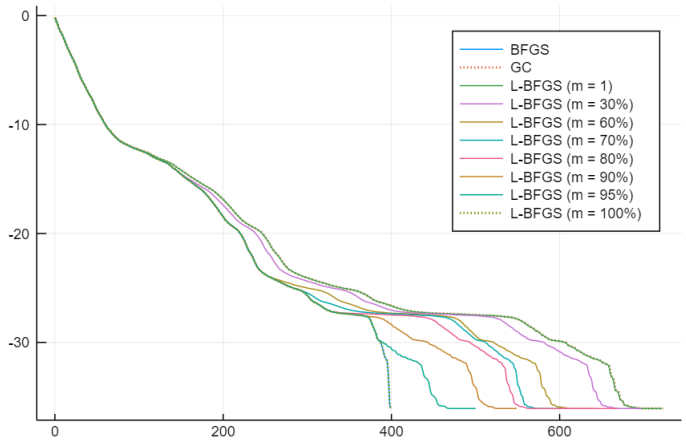

Figure 6: évolution de $f(x_k)$ pour les différentes méthodes (dimension 400)

3.3 Mise à l'échelle

Pour diminuer la perte d'orthogonalité à cause du mauvais conditionnement du problème, une idée présentée dans [4] est de modifier la matrice H_k^0 à chaque itération en tenant compte des informations données précédemment. La technique est appelée scaling ou encore mise à l'échelle et consiste à obtenir une meilleure approximation de la hessienne en comparaison de celle obtenue lorsque $H_k^0 = I$.

Théorème 5 *Soit une quadratique $f(x) = \frac{1}{2}x^tAx + b^tx$. Si nous effectuons une recherche linéaire exacte avec la méthode L-BFGS pour n'importe quel $m \geq 1$ et avec $H_k^0 = \delta_k I$, $\delta_k \neq 0$. Si de plus $y_k^t s_k \neq 0$ pour tout k. Alors les itérés s_k sont identiques à ceux générés par les méthodes de BFGS et du gradient conjugué.*

Démonstration. Nous faisons le même raisonnement par récurrence que pour démontrer le théorème 4. Avec les mêmes hypothèses de récurrence nous retrouvons (97). Puisque $H_k^0 = \delta_k I$,

$$\begin{aligned} d_k = -H_{k+1}g_{k+1} &= -\delta_k V_k^t g_{k+1} \\ &= \delta_k \left(-g_{k+1} + \frac{g_{k+1}^t g_{k+1}}{y_k^t d_k} d_k \right) \\ &= \delta_k d_{k+1}^{CG}. \end{aligned} \tag{103}$$

Étant donné que $\delta_k \neq 0$, d_k et d_k^{CG} sont colinéaires. Il s'en suit que la méthode $L-BFGS$ avec $H_k^0 = \delta_k I$ retrouve le même itéré s_{k+1} que la méthode du gradient conjugué, ce qui établie la récurrence. □

Nous savons donc que nous préservons la terminaison quadratique avec n'importe quel $\delta_k \neq 0$ et ce pour tout m. Toute la question est à présent de trouver pour quelles valeurs de δ_k la méthode

L-BFGS se comporte le mieux face à des problèmes mal-conditionnés. Dans la section suivante nous n'utiliserons qu'une méthode de mise à l'échelle qui nous est donnée par le résultat suivant

Théorème 6 ([4], propriété P_4') *La solution du problème*

$$\begin{aligned} &\underset{\delta>0}{\operatorname{argmin}} \ \left\{ \min_{H} \|\delta I - H\| \right\}, \\ &s.c. \ Hy_k = s_k, \end{aligned} \tag{104}$$

nous est donnée par

$$\delta_k = \frac{y_k^t s_k}{y_k^t y_k}. \tag{105}$$

Appliquée à une quadratique strictement convexe (ou tant que la courbure de la quadratique n'est pas négative ou nulle), $y_k^t s_k = s_k^t A s_k > 0$ et donc $\delta_k > 0$.

Ce facteur de mise à l'échelle est couramment utilisé et produit des résultats relativement bons. Il est entre autre implémenté dans le package *LinearOperators* de Julia que nous allons utiliser dans la section suivante.

3.4 Recherche d'un compromis entre le gradient conjugué et la méthode BFGS

La méthode L-BFGS permet à première vue d'obtenir un bon compromis entre la méthode du gradient conjugué et la méthode BFGS en termes de temps de calcul et en termes d'espace mémoire utilisé. Nous effectuons dans cette section des tests numériques pour évaluer certaines stratégies afin de trouver le meilleur compromis possible. La première idée est de regarder ce qu'il se passe pour différents m constants. Pour ce faire, nous testons notre algorithme avec $m \in \{1, 3, 5, 10, 100, 1000\}$.

En observant la figure 4, nous nous rendons compte que L-BFGS peine à faire diminuer le gradient aux toutes dernières itérations seulement. Dans un algorithme de régions de confiance, il peut donc être intéressant d'utiliser une petite mémoire tant que l'on va toucher la frontière de la région $\Omega(\Delta)$ rapidement. C'est ensuite que l'on va avoir besoin d'augmenter la mémoire m, lorsque le minimum du sous-problème ($\mathcal{SP}2$) est à l'intérieur de $\Omega(\Delta)$ et donc que l'on cherche le zéro du gradient. Nous proposons ainsi une deuxième stratégie. Nous rentrons en paramètre un critère de précision $\xi > 0$. Si $\|\nabla f(x_k)\| < \xi$, nous espérons être proches de la solution et nous appelons alors L-BFGS avec pleine mémoire (nous prenons $m = n$ en supposant que cela assure toujours la terminaison quadratique).

Enfin, nous testons le scaling comme présenté à la section précédente. Ceci nous donne une troisième stratégie faisant intervenir L-BFGS.

L'algorithme de régions de confiance avec L-BFGS à mémoire variable et avec scaling est implémenté sur la base du code *trunk.jl* du package *JSOSolvers* de Julia. Nous remplaçon l'appel de la fonction $cg(\cdot)$ par celui d'une nouvelle fonction $lbfgsTrunk(\cdot)$ faisant appel cette fois-ci aux fonctions de *lbfgs.jl* du package *LinearOperators*. La nouvelle fonction possède les mêmes paramètres que $cg(\cdot)$ en plus de deux options permettant d'obtenir les trois stratégies présentées : m constant, $m = n$ lorsque $\|\nabla f(x_k)\| < \xi$, et la mise à l'échelle. Nous comparons enfin LBFGS avec CG dont deux versions sont comparées. Le coefficient β_1 fait référence à (40) tandis que β_2 fait référence à (26). C'est la version de (40) qui est initialement implémentée avec *cg.jl*.

Nous réalisons nos tests sur cinq fonctions non-quadratiques en dimension 1000 (voir les fonctions en Annexes B. Les résultats des différents tests sont présentés en Annexes A.

Notons qu'il est difficile de comparer les différents paramètres grâce au temps de l'exécution. Ceux-ci sont relativement proches et le temps est trop variable pour pouvoir dire qu'en général une telle méthode est plus rapide qu'une autre. Lorsque les temps sont très proches (à moins de 4 secondes près), il va être intéressant de regarder le nombre d'itérations externes (*iter*) correspondant au nombre

d'itérations dans l'algorithme de régions de confiance. Les nombres d'évaluations de la fonction objectif et de son gradient (*neval_obj* et *neval_grad*) sont directement liés à ce nombre d'itérations. Le nombre d'évaluations des produits hessien-vecteur (*neval_hprod*) est quant à lui relié aux opérations internes effectuée dans la résolution des sous-problèmes de régions de confiance avec CG et LBFGS.

Nous pouvons observer dans un premier temps que les deux versions du gradient conjugué n'ont pas les mêmes performances. La version de CG avec β_1 se comporte généralement mieux, même si sur sparsine c'est la version avec β_2 qui s'en sort le mieux.

Concernant L-BFGS, augmenter m permet généralement de faire diminuer *neval_hprod* mais fait augmenter le temps de calcul. Avec $m = 3$, l'utilisaton de L-BFGS semble intéressante par rapport à celle du gradient conjugué. Pour des temps comparables à ceux du gradient conjugué, nous calculons moins de produits hessien-vecteur sur les problèmes *sparsine*, *broydn7d* et *woods* (une vingtaine de moins) mais nous en calculons environ 150 de plus sur le problème *nondquar*. Comme observé dans la précédente section, il faut augmenter m très largement pour avoir de nettes améliorations. Avec $m = 10$, les résultats sont relativement proches de ceux réalisés avec $m = 3$. L'utilisation de valeurs plus grande de m entraîne des temps d'exécution trop long. Enfin augmenter m ne fait pas systématiquement baisser *hprod*. Prenons l'exemple de *nondquar* où l'utilisation de $m = 1$ permet d'avoir 100 *hprod* de moins qu'avec $m = 3$. Cela vient sans doutes du fait que les régions de confiance générées n'ont pas été les mêmes à un moment donné et il est donc difficile de comparer le nombre de *hprod* en fonction de m.

Nous pouvons remarquer qu'en rajoutant le scaling, les résultats sont parfois meilleurs et parfois pires. Avec $m = 1$, on calcule 200 *hprod* de moins sur *nondquar* grâce au scaling alors qu'on en effectue 400 de plus sur *broydn7d*.

Enfin, l'utilisation de ξ choisi arbitrairement à 0.5 donne des résultats intéressants. Avec $m = 1$ on calcule moins de *hprod* sur *sparsine*, *broydn7d*, *nondquar* et $fletchcr$ que sans le paramètre ξ. Cependant le temps d'exécution peut être plus conséquent : deux fois plus sur $fletchcr$. La pleine mémoire de L-BFGS est peut-être enclenchée trop tôt. Une piste d'amélioration serait d'avoir une règle de décision moins arbitraire pour activer la pleine mémoire, en ayant une estimation du conditionnement des sous-problèmes quadratiques par exemple.

Conclusion

Dans ce cahier, nous avons présenté les méthodes de la classe de Broyden et leur application pour la résolution des sous-problèmes quadratiques dans un algorithme de régions de confiance. Nous avons en particulier concentré notre étude sur la méthode BFGS, ce qui nous a mené à étudier sa version à mémoire limitée, la méthode L-BFGS. Le fil conducteur de ce projet a été de démontrer qu'avec une recherche linéaire exacte, et sous certaines conditions additionnelles, toutes les méthodes étudiées produisent les mêmes itérés que la méthode du gradient conjugué. À ce propos, les démonstrations réalisées sont à notre connaissance originales et l'identité (32) obtenue dans le théorème 1 semble inédite.

Le second point de notre étude aura été de montrer numériquement que les résultats des différentes méthodes diffèrent lorsque les sous-problèmes sont mal-conditionnés. La méthode BFGS fait preuve d'une bonne résistance au mauvais conditionnement mais réalise des calculs plus longs que ceux réalisés par le gradient conjugué ou la méthode L-BFGS à faible mémoire. Nous avons réalisé des tests numériques pour trouver un juste milieu entre la méthode du gradient conjugué et la méthode BFGS. Pour ce faire, nous avons présenté trois stratégies à adapter sur la méthode L-BFGS avec recherche linéaire exacte :

- Trouver une mémoire constante permettant d'obtenir les performances désirées.
- Augmenter la mémoire lorsque que le gradient est relativement proche de zéro.
- Calculer un facteur de mise à l'échelle à chaque itération.

Nous aurions souhaité pouvoir étudier d'autres stratégies comme celle d'augmenter la mémoire par étape. Nous aurions également désiré trouver d'autres critères pour augmenter cette mémoire comme l'estimation du conditionnement du sous-problème.

Globalement, il est difficile d'estimer quelle stratégie est la meilleure. Nous pouvons toutefois conclure que l'utilisation d'une mémoire assez faible (m entre 5 et 10) permet de diminuer le nombre d'évaluations des produits hessien-vecteur pour des temps comparables à ceux du gradient conjugué. Utiliser des mémoires plus importantes n'est pas envisageable en très grande dimension.

Annexes

A - Tableaux

Table 1: CG avec β_1

name	status	objective	elapsed_time	iter	neval_obj	neval_grad	neval_hprod
fletchcr	first_order	3.9e−16	1.6e+01	470	471	467	6450
nondquar	first_order	1.7e−05	1.1e+00	54	55	49	644
woods	first_order	1.0e+00	9.6e−01	48	49	42	265
broydn7d	first_order	1.0e+02	9.7e+00	81	82	79	1976
sparsine	first_order	2.6e−11	3.2e+01	53	54	46	7419

Table 2: CG avec β_2

name	status	objective	elapsed_time	iter	neval_obj	neval_grad	neval_hprod
fletchcr	first_order	3.9e−16	1.7e+01	470	471	467	6455
nondquar	first_order	1.6e−05	1.3e+00	63	64	57	712
woods	first_order	1.0e+00	9.5e−01	52	53	45	287
broydn7d	first_order	8.4e+01	1.2e+01	95	96	93	2142
sparsine	first_order	4.8e−11	3.3e+01	53	54	46	7407

Table 3: $m = 1$

name	status	objective	elapsed_time	iter	neval_obj	neval_grad	neval_hprod
fletchcr	first_order	3.9e−16	2.6e+01	470	471	467	6453
nondquar	first_order	1.0e−05	1.9e+00	64	65	58	795
woods	first_order	1.0e+00	1.2e+00	52	53	44	288
broydn7d	first_order	1.0e+02	1.1e+01	80	81	79	1626
sparsine	first_order	3.5e−11	3.4e+01	53	54	46	7440

Table 4: $m = 3$

name	status	objective	elapsed_time	iter	neval_obj	neval_grad	neval_hprod
fletchcr	first_order	3.9e−16	2.3e+01	470	471	467	6454
nondquar	first_order	8.9e−06	2.0e+00	74	75	67	899
woods	first_order	1.0e+00	1.0e+00	46	47	40	257
broydn7d	first_order	1.0e+02	1.2e+01	84	85	81	1935
sparsine	first_order	3.4e−11	3.2e+01	53	54	46	7407

Table 5: $m = 5$

name	status	objective	elapsed_time	iter	neval_obj	neval_grad	neval_hprod
fletchcr	first_order	3.9e−16	2.1e+01	470	471	467	6456
nondquar	first_order	1.1e−05	2.0e+00	59	60	54	727
woods	first_order	1.0e+00	1.1e+00	46	47	40	257
broydn7d	first_order	8.4e+01	1.5e+01	82	83	80	1883
sparsine	first_order	4.1e−11	3.4e+01	53	54	46	7390

Table 6: $m = 10$

name	status	objective	elapsed_time	iter	neval_obj	neval_grad	neval_hprod
fletchcr	first_order	3.9e−16	2.1e+01	470	471	467	6457
nondquar	first_order	8.4e−06	2.2e+00	68	69	62	892
woods	first_order	1.0e+00	1.6e+00	46	47	40	257
broydn7d	first_order	8.4e+01	1.1e+01	89	90	85	1784
sparsine	first_order	5.3e−11	3.5e+01	53	54	46	7308

Table 7: $m = 100$

name	status	objective	elapsed_time	iter	neval_obj	neval_grad	neval_hprod
fletchcr	first_order	3.9e−16	2.8e+01	470	471	467	6459
nondquar	first_order	1.3e−05	1.3e+00	53	54	49	625
woods	first_order	1.0e+00	1.0e+00	46	47	40	257
broydn7d	first_order	8.4e+01	1.1e+01	89	90	85	1742
sparsine	first_order	4.8e−10	4.5e+01	53	54	46	6730

Table 8: $m = 1000$

name	status	objective	elapsed_time	iter	neval_obj	neval_grad	neval_hprod
fletchcr	first_order	3.9e−16	7.0e+01	470	471	467	5915
nondquar	first_order	1.3e−05	2.1e+00	53	54	49	625
woods	first_order	1.0e+00	1.8e+00	46	47	40	257
broydn7d	first_order	8.4e+01	1.6e+01	89	90	85	1714
sparsine	first_order	3.1e−09	5.0e+01	53	54	46	4705

Table 9: $m = 1$ **et** $\xi = 0.5$

name	status	objective	elapsed_time	iter	neval_obj	neval_grad	neval_hprod
fletchcr	first_order	3.9e−16	3.5e+01	470	471	467	6181
nondquar	first_order	1.1e−05	2.4e+00	58	59	53	690
woods	first_order	1.0e+00	2.1e+00	50	51	43	289
broydn7d	first_order	1.0e+02	1.1e+01	80	81	79	1596
sparsine	first_order	5.4e−11	4.0e+01	53	54	46	6809

Table 10: $m = 3$ **et** $\xi = 0.5$

name	status	objective	elapsed_time	iter	neval_obj	neval_grad	neval_hprod
fletchcr	first_order	3.9e−16	3.5e+01	470	471	467	6181
nondquar	first_order	1.2e−05	2.4e+00	55	56	51	653
woods	first_order	1.0e+00	2.1e+00	46	47	40	257
broydn7d	first_order	1.0e+02	1.3e+01	84	85	81	1882
sparsine	first_order	5.8e−11	3.8e+01	53	54	46	6786

Table 11: $m = 5$ **et** $\xi = 0.5$

name	status	objective	elapsed_time	iter	neval_obj	neval_grad	neval_hprod
fletchcr	first_order	3.9e−16	4.4e+01	470	471	467	6182
nondquar	first_order	1.4e−05	2.0e+00	48	49	44	576
woods	first_order	1.0e+00	2.2e+00	46	47	40	257
broydn7d	first_order	8.4e+01	1.6e+01	82	83	80	1809
sparsine	first_order	5.3e−11	4.1e+01	53	54	46	6782

Table 12: $m = 10$ **et** $\xi = 0.5$

name	status	objective	elapsed_time	iter	neval_obj	neval_grad	neval_hprod
fletchcr	first_order	3.9e−16	4.0e+01	470	471	467	6183
nondquar	first_order	1.3e−05	1.9e+00	53	54	49	625
woods	first_order	1.0e+00	2.2e+00	46	47	40	257
broydn7d	first_order	8.4e+01	1.3e+01	89	90	85	1734
sparsine	first_order	6.6e−11	4.6e+01	53	54	46	6704

Table 13: $m = 100$ **et** $\xi = 0.5$

name	status	objective	elapsed_time	iter	neval_obj	neval_grad	neval_hprod
fletchcr	first_order	3.9e−16	4.6e+01	470	471	467	6184
nondquar	first_order	1.3e−05	2.0e+00	53	54	49	625
woods	first_order	1.0e+00	1.4e+00	46	47	40	257
broydn7d	first_order	8.4e+01	1.3e+01	89	90	85	1733
sparsine	first_order	4.7e−10	4.9e+01	53	54	46	6134

Table 14: $m = 1$ **avec scaling**

name	status	objective	elapsed_time	iter	neval_obj	neval_grad	neval_hprod
fletchcr	first_order	3.9e−16	2.4e+01	470	471	467	6453
nondquar	first_order	2.5e−05	1.5e+00	52	53	47	575
woods	first_order	1.0e+00	1.4e+00	53	54	45	288
broydn7d	first_order	1.2e+02	1.3e+01	87	88	83	2029
sparsine	first_order	3.1e−11	4.1e+01	53	54	46	7422

Table 15: $m = 3$ **et scaling**

name	status	objective	elapsed_time	iter	neval_obj	neval_grad	neval_hprod
fletchcr	first_order	3.9e−16	2.7e+01	470	471	467	6451
nondquar	first_order	1.1e−05	2.2e+00	60	61	55	776
woods	first_order	1.0e+00	1.3e+00	54	55	47	296
broydn7d	first_order	8.4e+01	1.3e+01	82	83	81	1959
sparsine	first_order	4.6e−11	4.2e+01	53	54	46	7384

Table 16: $m = 5$ **et scaling**

name	status	objective	elapsed_time	iter	neval_obj	neval_grad	neval_hprod
fletchcr	first_order	3.9e−16	3.0e+01	470	471	467	6456
nondquar	first_order	1.8e−05	2.4e+00	59	60	53	655
woods	first_order	1.0e+00	1.7e+00	54	55	47	296
broydn7d	first_order	8.4e+01	9.5e+00	80	81	78	1358
sparsine	first_order	4.7e−11	4.2e+01	53	54	46	7387

Table 17: $m = 10$ **et scaling**

name	status	objective	elapsed_time	iter	neval_obj	neval_grad	neval_hprod
fletchcr	first_order	3.9e−16	2.6e+01	470	471	467	6451
nondquar	first_order	1.2e−05	1.7e+00	51	52	48	598
woods	first_order	1.0e+00	1.3e+00	54	55	47	296
broydn7d	first_order	1.0e+02	1.2e+01	87	88	83	1793
sparsine	first_order	6.3e−10	4.3e+01	53	54	46	7040

Table 18: $m = 100$ **et scaling**

name	status	objective	elapsed_time	iter	neval_obj	neval_grad	neval_hprod
fletchcr	first_order	3.9e−16	3.9e+01	470	471	467	6566
nondquar	first_order	2.0e−05	1.6e+00	49	50	45	535
woods	first_order	1.0e+00	1.5e+00	54	55	47	296
broydn7d	first_order	1.0e+02	1.5e+01	87	88	83	1762
sparsine	first_order	8.1e−11	5.3e+01	53	54	46	6698

Table 19: $m = 1000$ **et scaling**

name	status	objective	elapsed_time	iter	neval_obj	neval_grad	neval_hprod
fletchcr	first_order	3.9e−16	6.3e+01	470	471	467	5915
nondquar	first_order	2.0e−05	2.0e+00	49	50	45	535
woods	first_order	1.0e+00	1.8e+00	54	55	47	296
broydn7d	first_order	1.0e+02	1.4e+01	87	88	83	1746
sparsine	first_order	3.1e−09	4.9e+01	53	54	46	4705

B - Fonctions tests

fletchcr :

$$f(x) := 100 \sum_{i=1}^{n-1} \left(x_{i+1} - x_i + 1 - x_i^2\right)^2$$

nondquar :

$$n(x) := (x_1 - x_2)^2 + (x_{n-1} - x_n)^2 + \sum_{i=1}^{n-2} (x_i + x_{i+1} + x_n)^4$$

woods :

$$w(x) := 1 + \sum_{i=1}^{\lfloor \frac{1}{4} \rfloor} \Big(100 \left(x_{4i-2} - x_{4i-3}^2\right)^2 + (1 - x_{4i-3})^2 + 90 \left(x_{4i} - x_{4i-1}^2\right)^2 + (1 - x_{4i-1})^2 + 10 \left(x_{4i-2} + x_{4i} - 2\right)^2 + 0.1 \left(x_{4i-2} - x_{4i}^2\right)^2 \Big)$$

broydn7d :

$$b(x) := |1 - 2x_2 + (3 - x_1/2)x_1|^{7/3} + \sum_{i=2}^{n-1} |1 - x_{i-1} - 2x_{i+1} + (3 - x_i/2)x_i|^{7/3} + |1 - x_{n-1} + (3 - x_n/2)x_n|^{7/3} + \sum_{i=1}^{n/2} \left|x_i + x_{i+n/2}\right|^{7/3}$$

sparsine :

$$s(x) := \frac{1}{2} \sum_{i=1}^{n} \Big(i \sin(x_i) + \sin(x_{(2i-1) \bmod n+1}) + \sin(x_{(3i-1) \bmod n+1}) + \sin(x_{(5i-1) \bmod n+1}) + \sin(x_{(7i-1) \bmod n+1}) + \sin^2(x_{(11i-1) \bmod n+1}) \Big)$$

References

[1] Charles G. Broyden. The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations. IMA Journal of Applied Mathematics, 6:76–90, March 1970.

[2] John E. Dennis and Jorge J. Moré. Quasi-Newton Methods, Motivation and Theory. SIAM Review, 19:46–89, January 1977.

[3] Jean-Charles Gilbert. Fragments d'optimisation différentiable: Théories et algorithmes. Syllabus de cours ENSTA, Paris, pages 427–436, 2018. https://who.rocq.inria.fr/Jean-Charles.Gilbert/ensta/optim.html.

[4] Jean-Charles Gilbert and Claude Lemaréchal. Some numerical experiments with variable-storage quasi-Newton algorithms. Mathematical Programming, 45:40–435, 1989.

[5] Magnus R. Hestenes and Eduard Stiefel. Methods of Conjugate Gradients for solving linear systems. Journal of Research of the National Bureau of Standards, 49(6), December 1952.

[6] David G. Luenberger. Introduction to Linear and Nonlinear Programming, second ed. Addison Wesley, 1984.

[7] Jorge Nocedal and Stephen J. Wright. Numerical Optimization. Springer-Verlag, New York, 2nd edition, 2006.

[8] Christopher C. Paige. A useful form of unitary matrix obtained from any sequence of unit 2-norm n-vectors. SIAM Journal on Matrix Analysis and Applications, 45(2):565–583, May 2009.

[9] Trond Steihaug. The conjugate gradient method and trust regions in large scale optimization. SIAM Journal on Numerical Analysis, 20:626–637, June 1983.