وزارة التعليم العالي والبحث العلمي


**BADJI MOKHTAR –ANNABA UNIVERSITY**
**UNIVERSITE BADJI MOKHTAR ANNABA**

**جامعة باجي مختار - عنابة -**

**Faculté des Sciences**
**Département de Mathématiques**


**THESE**

Présentée en vue de l'obtention du diplôme de
**DOCTORAT EN MATHEMATIQUES**
**Option** : Analyse Numérique

# Méthode du Gradient Conjugué et Convergence Uniforme des Méthodes Multigrilles


Par
**MELLAL Romaissa**

Sous la direction de

**Pr. HAIOUR Mohamed**


Devant le jury

| | | | |
|---|---|---|---|
| **PRESIDENT** | **A. BENCHETTAH** | **Pr** | **U.B.M. ANNABA** |
| **EXAMINATEUR** | **M. Z. AISSAOUI** | **M.C.A** | **U. GUELMA** |
| **EXAMINATEUR** | **M. C. BOURAS** | **M.C.A** | **U.B.M. ANNABA** |
| **EXAMINATEUR** | **F. ELLAGOUNE** | **M.C.A** | **U. GUELMA** |
| **EXAMINATEUR** | **S. BOULARES** | **M.C.A** | **U. QASSIM A.S** |

**Année Universitaire 2014 / 2015**

# Table des matières

**1 Notions Préliminaires** **6**

1.1 Quelques généralités sur les problèmes de minimisation sans contraintes et leurs algorithmes . . . . . 6

1.1.1 Aspect général des algorithmes . . . . . 7

1.1.2 La notion de convergence globale . . . . . 8

1.2 Conditions d'optimalité des problèmes d'optimisation sans contraintes . . . 8

1.2.1 Conditions nécessaires d'optimalité . . . . . 9

1.2.2 Conditions suffisantes d'optimalité . . . . . 9

1.3 Modes de convergence . . . . . 9

1.4 Recherche linéaire . . . . . 10

1.4.1 Principe des recherches linéaires . . . . . 10

1.4.2 Objectifs à atteindre . . . . . 11

1.4.3 Type de recherche linéaire . . . . . 12

1.5 Recherches linéaires inexactes . . . . . 14

1.5.1 Schéma des recherches linéaires inexactes . . . . . 14

1.5.2 La règle d'Armijo . . . . . 15

1.5.3 La règle de Goldstein & Price. . . . . 16

1.5.4 La règle de Wolfe . . . . . 16

1.5.5 Algorithmes d'Armijo, Goldstein-Price et de Wolfe . . . . . 18

**2 Méthodes de gradient conjugué et multigrille** **19**

2.1 Méthode du gradient conjugué linéaire . . . . . 19

2.1.1 Principe de la méthode du gradient conjugué linéaire: . . . . . 19
2.1.2 Algorithme du gradient conjugué linéaire . . . . . 21
2.1.3 Convergence de la méthode du gradient conjugué linéaire . . . . . 22
2.2 Méthode du gradient conjugué non linéaire . . . . . 22
2.2.1 Différentes versions de la méthode du gradient conjugué non linéaire 22
2.2.2 Les directions des méthodes du gradient conjugué non linéaires sont-elles des directions de descente de $f$ ? . . . . . 24
2.2.3 Algorithmes des méthodes HS, FR, PRP, CD, LS, DY et HZ . . . . 25
2.2.4 Résultats de convergence des méthodes du gradient conjugué non linéaire avec une recherche linéaire inexacte . . . . . 25
2.3 Méthode multigrille . . . . . 29
2.3.1 Principe de la méthode multigrille . . . . . 30
2.3.2 Algorithme de la méthode multigrille . . . . . 31
2.3.3 Analyse de la convergence . . . . . 33

**3 Combinaisons des méthodes multigrille et gradient conjugué 35**
3.1 Problématique . . . . . 36
3.2 Méthode multigrille-gradient conjugué MG-CG . . . . . 37
3.2.1 Principe de la méthode MG-CG . . . . . 37
3.2.2 Algorithme du MG-CG . . . . . 38
3.3 Méthode multigrille-gradient MGgM . . . . . 38
3.3.1 Principe de la méthode MGgM . . . . . 38
3.3.2 L'algorithme de MGgM . . . . . 40
3.3.3 Analyse de la convergence . . . . . 41
3.4 Méthode multigrille gradient conjugué MGcgM . . . . . 41
3.4.1 Principe de la méthode MGcgM . . . . . 41
3.4.2 L'algorithme de MGcgM . . . . . 42
3.4.3 Analyse de la convergence . . . . . 44
3.5 Simulations Numériques . . . . . 44
3.6 Conclusion . . . . . 49

**4 Etude comparative de quelques méthodes du gradient conjugué non linéaire avec recherches linéaires inexactes** **50**

4.1 Simulations numériques . . . . . 51

4.1.1 La fonction BALF (Brown almost-linear ) . . . . . 53

4.1.2 La fonction BRYBND (Broyden banded ) . . . . . 55

4.1.3 La fonction BTF (Broyden tridiagonal ) . . . . . 59

4.1.4 La fonction DBVF (Discrete boundary value ) . . . . . 61

4.1.5 La fonction GENROSE (Generalized Rosenbrock) . . . . . 65

4.1.6 La fonction TRIDIA (cute ) . . . . . 67

4.2 Conclusion . . . . . 70

**Conclusion et perspectives** **72**

**Bibliographie** **75**

# *Remerciements*

En préambule à cette thèse, je tiens à remercier en premier lieu ALLAH qui m'a donné le pouvoir d'effectuer ce modeste travail.

Mes profonds remerciements, ma reconnaissance et ma gratitude vont à mon directeur de thèse monsieur le Professeur Mohamed Haiour. Je le remercie sincèrement pour la confiance qu'il m'a accordé, ses encouragements, ses conseils précieux ainsi que pour le temps qu'il m'a accordé malgrés ses obligations et ses responsabilités. Je le remercie également pour sa rigueur, sa bonne humeur et sa modestie.

Mes remerciements les plus respectueux vont au Professeur A. Benchettah d'avoir accepté de présider le Jury.

Je remercie vivement les Professeurs : M. C. Bouras, F. Ellaggoune, S. Boulares et M. Z. Aissaoui qui ont accepté la lourde tâche de lire, commenter et juger cette thèse.

Je tiens à exprimer ma gratitude et mes remerciements aux Professeurs Y. Saad, M. Al-Baali, F. Olivier, M. Lemarechal, B. T. Polyak, Y. Yuan, P. Spiteri, M. Goursat et S. Steer.

Je termine avec un grand remerciement bien particulier à toute personne qui de près ou de loin a contribué à ma réussite.

# Méthode du Gradient Conjugué et Convergence Uniforme des Méthodes Multigrille

## Résumé

**Résumé :** L'objectif de cette thèse est d'analyser les performances des méthodes hybrides multigrille (MG) et gradient conjugué (CG) pour résoudre plus efficacement certains problèmes. Pour cela nous avons fait plus de 200 tests numériques. D'abord nous avons effectué une étude comparative de deux approches de combinaison des méthodes MG et CG (MG-CG et MGcgM) pour la résolution du problème du Laplacien par sous domaine. Ensuite nous avons analysé les avantages de ces méthodes en comparaison avec les méthodes MG, CG, gradient conjugué préconditionné et GMRES. Nos simulations numériques (SCILAB) montrent que la méthode MG-CG converge plus rapidement que la méthode MGcgM pour notre problème. Cette étude nous a conduit à accélérer la convergence de la méthode MG-CG d'avantage et ceci en accélérant celle du CG toute en calculant le pas par des recherches linéaires inexactes. Notre choix est crucial, d'une part pour sa rapidité, mais aussi pour sa robustesse et la place mémoire qu'il requiert. Nous illustrons ceci en comparant sept différentes variantes de la méthode du gradient conjugué non linéaire avec recherche linéaire inexacte (Wolfe et Armijo) pour quelques fonctions testes bien connues. Nos résultats numérique prouvent que ce choix permet d'allier les avantages de la méthode CG et les règles inexacte étudiées. Nous avons implémenté les algorithmes étudiés dans cette thèse sous forme de générateur de programmes, ce qui nous permet de résoudre une large famille de problèmes types.

**Mots-clés :** Méthode multigrille, Méthode du gradient conjugué, Recherche linéaire inexacte, Règle d'Armijo, Règle de Wolfe, Méthode de Hestenes-Stiefel, Méthode de Fletcher-Reeves, Méthode de Polak-Ribière-Polyak, Méthode de la descente conjuguée, Méthode de Dai-Yuan, Méthode de Harger Zhang.

# Conjugate Gradient Method and Uniform Convergence of Multigrid Method

## Abstract

**Abstract :** This thesis aims to analyze the prefermances of the hybrid multigrid-conjugate gradient method to solve more effectively some problems and find tools to optimize. For this, we have made more than 200 numerical tests.

First we compare two approaches to combining the multigrid method and the conjugate gradient ( CG and MG- MGcgM ) for solving the problem of the Laplacian by subdomain with boundary conditions of Dirichlet homogeneous discretized by the classical scheme of finite difference of five points. Then we analyze the benefits of these methods in comparison with multigrid methods ( MG) , conjugate gradient (CG ), preconditioned conjugate gradient ( PCG ) and GMRES.

Our Numerical simulations ( SCILAB ) illustrate that the MG- GC method converges more quickly than the method MGcgM for our problem .

Given this result, we accelerate the convergence MG -CG by accelerating the convergence of the conjugate gradient by computing the step length using inexacte line search. We illustrate this result by comparing seven different variants of the method of nonlinear conjugate gradient with inexact line search ( Armijo and strong Wolfe ) for some test functions well knowns.

**Keywords** : Multigrid method, Conjugate gradient method, Inexact line search, Armijo line search, Wolfe line search, Hestenes-Stiefel method, Fletcher-Reeves method, Polak-Ribière-Polyak method, Conjugate descent method, Dai-Yuan method, Harger Zhang method.

# Introduction

Il existe plusieurs algorithmes pour la résolution des problèmes d'analyse numérique, cependant, des inconvénients majeurs tels que la convergence et la précision sont rencontrés lors de l'utilisation des algorithmes classiques.
Parmi les méthodes utilisées pour accélérer la convergence on cite deux algorithmes :

1. La méthode du gradient conjugué qui constitue une famille d'algorithmes très performante.

2. Les méthodes multigrilles qui sont des algorithmes à coût linéaire en fonction du nombre de points de discrétisation et ceci quelle que soit la dimension, elles permettent de résoudre des systèmes linéaire et non linéaire.

La *méthode du Gradient Conjugué* est une méthode itérative très puissante et très utilisée pour résoudre des systèmes linéaire et non linéaire de grande taille car elle nécessite peu de stockage. Elle repose sur le concept des directions conjuguées parce que les gradients successifs sont orthogonaux entre eux et aux directions précédentes.
L'idée initiale était de trouver une suite de directions de descente permettant de minimiser une fonction quadratique ou un système quadratique de $\mathbb{R}^n$ dans $\mathbb{R}$.
Cette méthode a été découverte en 1952 par Hestenes et Steifel ([34]) *(méthode de Hestenes et Steifel)* pour la minimisation de fonctions quadratiques strictement convexes (cas linéaire).
Plusieurs mathématiciens ont étendu cette méthode pour le cas non linéaire et pour la première fois par Fletcher et Reeves en 1964 ([24]) (*méthode de Fletcher-Reeves*) ainsi que Polak-Ribière ([50]) et Ployak ([51]) (*méthode de Polak-Ribière-Ployak*) en 1969, Fletcher en 1987 ([22]) ( *méthode de la descente conjuguée*), Liu and Storey en 1991 [42] (méthode

de Liu et Storey), Dai et Yuan en 1999([18]) (*méthode de Dai-Yuan*) et Harger et Zhangen en 2005 [32] (méthode de Harger-Zhang).
L'algorithme CG sélectionne les vecteurs de direction successifs de tels sorte qu'ils soient conjugués par rapport à la matrice $A$, ce qui signifie que $(d_i, d_j)_A = 0$, pour $i \neq j$. A l'étape k on évalue le vecteur de gradient négatif actuel et y ajoute une combinaison linéaire des vecteurs de directions précédentes pour obtenir un nouveau vecteur de direction conjugué le long de laquelle se déplacer.
Il y a trois principaux avantages de cette méthode qui provient de la sélection des directions. Tout d'abord, à moins que la solution soit atteinte en moins de n étapes, le gradient est toujours différent de zéro et linéairement indépendant de tous les vecteurs de direction précédentes. D'autre part, un avantage plus important de la méthode de gradient conjugué est la formule particulièrement simple utilisée pour déterminer le nouveau vecteur de direction. Troisièmement, puisque les directions sont basées sur les gradients, le processus fait de bons progrès vers la solution à chaque étape.

Les *méthodes multigrilles* sont au cœur des solveurs d'algèbre linéaire les plus performants. Ces méthodes sont connues depuis un peu plus de 70 ans (Southwell [60, 1935], [61, 1946] et Fedorenko [20,1962]) et n'utilisaient alors seulement que deux niveaux de grilles. Les premiers algorithmes et tests numériques ont été présentés par Brandt, pour la première fois en 1972 ([10]), et pour une première publication en 1977 ([11]). La littérature au sujet des multigrilles est trop conséquente pour autoriser une présentation exhaustive. Historiquement les premiers développements ont été faits pour la résolution de l'équation de Poisson, leur emploi se démocratise dans le cadre de la mécanique des fluides, où leur utilisation se conjugue assez facilement avec des discrétisations volumes finis ou différences finis (Wesseling [64,1992]). Mais leur utilisation dans le cadre d'une discrétisation éléments finis est plus récente (Mocellin [46,1999]). Cette complexité théorique a été démontrée dans divers cas de discrétisation d'équations elliptiques (Wesseling[63,1982]), (Hackbusch[30,1978]).
La caractéristique la plus intéressante de ces méthodes itératives est une convergence asymptotique en $\circ$(N) où N est le nombre d'inconnues du système résolu.

Comme leur nom l'indique, la particularité des méthodes multi-grilles est de considérer un problème non pas sur un seul maillage, mais sur différents maillages successivement raffinés, dans le but de fournir une méthode performante pour approcher le résultat à l'échelle la plus fine. En fonction de la méthode d'obtention de ces différents niveaux de calcul, on divise les multigrilles en deux familles, les approches géométrique (Janka et Guillard [38, 2003]) et algébrique (Alcouffe, A. Brandt, J.E. Dendy et J. W. Painter[2, 1981]), (Brandt, McCormick et Ruge [12, 1982]).

Le principe général de ces méthodes est, partant de la grille la plus fine :

– par des étapes de lissage de l'erreur, déterminer successivement sur les grilles de plus en plus grossières des résidus d'approximation de la solution,

– en déduire la correction sur la solution associée à la grille la plus grossière par une résolution complète,

– appliquer cette correction sur les grilles de plus en plus fines, en faisant suivre chaque correction de la solution par des étapes de lissage.

En résumé, les algorithmes multi-grilles font intervenir trois ingrédients principaux :

– des solveurs basiques, peu performants (Gauss-Seidel, Jacobi, ...), mais qui sont de bons lisseurs, utilisés à toutes les échelles sauf la plus fine,

– une méthode de résolution complète utilisée uniquement sur l'échelle la plus grossière,

– des opérateurs de prolongation et de restriction permettant de transférer les champs inconnus respectivement d'une grille grossière à une grille plus fine et d'une grille fine à une grille plus grossière.

Dans cette thèse on s'intéresse aux méthodes du gradient conjugué et multigrille. Nous avons fait plus de 200 tests numériques afin d'analyser la convergence de ces deux méthodes ainsi que de deux approches de combinaison entre eux.

Nous commençons par une comparaison de deux approches de combinaison de la méthode multigrille et la méthode du gradient conjugué pour la résolution du problème du Laplacien par sous domaine avec les conditions aux limites de Dirichlet homogènes (Laouar [39, 1988]), discrétisé par le schéma classique des différences finies à cinq points. La première combinaison a été proposé par Braess [8,1986] (MG-CG). L'idée consiste à utiliser la

méthode du gradient conjugué dans les étapes de pré-lissage et post-lissage de la méthode multigrille. Cet usage ne procure pas seulement le lissage, mais en même temps il améliore le calcul de la correction sur la grille grossière ce qui a accéléré la vitesse de convergence de l'algorithme du multigrille. Une nouvelle approche a été proposée par Pflaum [49, 2008] (MGcgM), qui construit des directions de correction qui sont conjuguées et leurs calculs se basent sur la restriction des gradients.

Notre étude est la comparaison et la consolidation par des simulations numérique (SCILAB) montrant que la méthode MG-CG converge plus rapidement que les méthodes MGgM et MGcgM pour ce problème.

D'autre part nous avons accéléré la convergence des méthodes du gradient conjugué non linéaire et nous avons illustré ceci en comparant sept différentes variantes du gradient conjugué (méthodes de Hestenes–Stiefel, Fletcher–Reeves, Polak–Ribière–Polyak, Descente conjugué, Liu–Storey, Dai–Yuan et Hager–Zhang) avec les recherches linéaire inexactes (Armijo et Wolfe forte). L'idée est d'utiliser des recherches linéaires inexactes pour le calcul de pas choisi dans la méthode du gradient conjugué au lieu d'appliquer la formule du pas qui nécessite que les directions soient orthogonales entre elles et qui ne se rencontrent pas dans la pratique. Ce nouveau choix a accéléré la convergence de la méthode du gradient conjugué.

Nos simulations numériques (Scilab) montrent que la recherche linéaire de Wolfe forte a accéléré la convergence des méthodes du gradient conjugué non linéaire étudiées mieux que la règle d'Armijo, et que les méthodes de Hager Zhang et de Dai-Yuan sont considérées plus performantes que les autres méthodes étudiées. Les exemples numériques illustrent ce résultat. On expliquera pourquoi on a abouti à ce résultat.

Notre thèse est divisée en quatre chapitres :

Dans le premier chapitre on introduit les outils de base. Tout d'abord nous rappellons quelques généralité sur les problèmes de minimisation sans contraintes. Ensuite nous décrivons les recherches linéaires inexactes d'Armijo, Goldstein-Price et de Wolfe, on donne leurs algorithmes et on cite les théorèmes qui assurent l'existence du pas de ces recherches. Dans le deuxième chapitre nous décrivons brièvement les méthodes du

gradient conjugué et multigrille, on présente leurs algorithmes et convergences. Dans le troisième chapitre on commence par présenter le problème sur lequel on va travailler. Ensuite nous abordons le cœur des approches de combinaison de méthodes multigrille et gradient conjugué (MG, MGgM et MGcgM) en décrivant l'idée sous-jacente à l'origine de ces méthodes ainsi que leurs algorithmes et la convergence de celles-ci et pour finir nous présentons nos simulations numériques. Et on termine ce chapitre par une conclusion. Dans le dernier chapitre nous rappelons les algorithmes des différentes versions de la méthode CG et les règles de Wolfes fortes et Armijo. Nous présentons après les six fonctions testes choisies et au-dessous de chacune de ces fonctions nous donnons les résultats obtenues ainsi qu'une conclusion. Finalement on présente une conclusion et des perspectives.

Notons que le quatrième chapitre a été présenté à la conférence ICMSA 2012 et fut l'objet d'une publication [44, 2013] dont l'ensemble des résultats est détaillé dans ce chapitre.

[1] R. Mellal and M. Haiour, Numerical simulations of some nonlinear conjugate gradient methods with inexact line searches, Proceedings of International Conference : Mathematical Science and Applications, Abu Dhabi, UAE. Universal Journal of Mathematics and Mathematical Sciences, Volume 4, Number 1, Pages 63-84, (2013).

CHAPITRE 1

# Notions Préliminaires

Ce chapitre est un rappel des notions préliminaires. On commence notre chapitre par un petit rappel de l'algorithme d'optimisation sans contraintes ensuite on décrit les principales règles de recherche linéaire inexacte (Armijo, Goldstein, Wolfe...) et on termine par l'étude des principales règles de recherche linéaire inexacte, où on va présenter leurs algorithmes et des théorèmes garantissants l'existence du pas calculé par ces recherches ainsi que leur contribution à la convergence des algorithmes à directions de descente.

## 1.1 Quelques généralités sur les problèmes de minimisation sans contraintes et leurs algorithmes

Soit $f : \mathbb{R}^n \to \mathbb{R}$. On appelle problème de minimisation sans contraintes le problème (1.1.1) suivant:

$$\min \{ f(x) : x \in \mathbb{R}^n \}. \qquad (P) \tag{1.1.1}$$

L'étude de ces problèmes est importante pour des raisons diverses.

La construction d'algorithmes consacrés à la recherche efficace de minimum d'une fonction sans contraintes est un problème complexe, car il ne suffit pas d'élaborer un algorithme il faut montrer en plus qu'il l'emporte sur ceux connus.

On compare des algorithmes en se basant sur plusieurs critères, par exemple, la précision du résultat, le nombre d'évaluations fonctionnelles et celui d'évaluations du gradient,

la vitesse de la convergence, le temps de calcul, l'occupation de la mémoire nécessaire, ....etc...

Même en se fixant des critères de comparaison, on ne peut pas classer les algorithmes ni en indiquer lequel est le meilleur. Donc un algorithme mauvais pour une vaste classe peut s'avérer efficace pour une autre, plus restreinte. L'utilisateur doit posséder tout un arsenal d'algorithmes pour être en mesure de faire face à chaque problème posé.

### 1.1.1 Aspect général des algorithmes

Pour construire des algorithmes de minimisation sans contraintes on fait appel à des processus itératifs du type

$$x_{k+1} = x_k + \alpha_k d_k \tag{1.1.2}$$

où $d_k$ détermine la direction de déplacement à partir du point $x_k$ et $\alpha_k$ est un facteur numérique dont la grandeur donne la longueur du pas dans la direction $d_k$.

Le type d'algorithme permettant de résoudre le problème (1.1.1) sera déterminé dès qu'on définit les procèdes de construction du vecteur $d_k$ et le calcul de $\alpha_k$ à chaque itération.

La façon avec laquelle on construit les vecteurs $d_k$ et les scalaires $\alpha_k$ détermine directement les propriétés du processus et spécialement en ce qui concerne la convergence de la suite $\{x_k\}$ , la vitesse de la convergence,....

Pour s'approcher de la solution optimale du problème (1.1.1) (dans le cas général, c'est un point qui vérifie avec une certaine précision les conditions nécessaires d'optimalité de $f$), on se déplace naturellement à partir du point $x_k$ dans la direction de la décroissance de la fonction $f$. Cette classe de méthodes est nommée méthodes à direction de descente.

**Définition 1.1.1** ***([25]).*** *Soit $f : \mathbb{R}^n \to \mathbb{R}$ telle que $f$ soit différentiable au point $\overline{x} \in \mathbb{R}^n$. Soit $d \in \mathbb{R}^n$ telle que $\nabla f(\overline{x})^t d < 0$. Alors il existe $\delta > 0$ tel que $f(\overline{x} + \alpha d) < f(\overline{x})$ pour tout $\alpha \in ]0, \delta[$. La direction $d$ s'appelle dans ce cas direction de descente.*

### 1.1.2 La notion de convergence globale

La question importante est donc celle de la convergence de la suite $\{x_k\}_k$ vers « ce qu'il faut » quelque soit l'itéré initiale (convergence globale).

**Définition 1.1.2** *([25]) On dit qu'un algorithme est globalement convergent (ou encore : possède la propriété de convergence globale) si, quel que soit le point de départ $x_0$ choisi, la suite $\{x_k\}_k$ générée par cet algorithme (ou une sous-suite) converge vers un point satisfaisant une condition nécessaire d'optimalité.*

La notion de convergence globale concerne le fait qu'on aura limite même si l'itéré initial est très éloigné de la limite $x^*$.
Il est très important de souligner qu'elle n'implique pas (contrairement à ce que pourrait suggérer le terme) la convergence vers un optimum global pour tout point de départ $x_0$. Il s'agirait là, du reste, d'une condition beaucoup trop sévère, qui ne serait remplie pratiquement par aucun des algorithmes connus.
Néanmoins, on peut noter, dès qu'un algorithme possède la propriété de convergence globale, il suffit d'imposer une condition de convexité pour obtenir précisément la convergence de l'algorithme vers un optimum globale du problème, quelque soit le point de départ choisi.

## 1.2 Conditions d'optimalité des problèmes d'optimisation sans contraintes

**Définition 1.2.1** *([6]) Considérons le problème de minimisation sans contraintes* (1.1.1).

*a) $x_* \in \mathbb{R}^n$ s'appelle minimum global du problème* (1.1.1) *si*

$$f(x_*) \leq f(x),\ \forall x \in \mathbb{R}^n.$$

*b) $x_*$ est un minimum local de* (1.1.1) *s'il existe un voisinage $V_\varepsilon(x_*)$ de $x_*$ tel que*

$$f(x_*) \leq f(x),\ \forall x \in V_\varepsilon(x_*).$$

*c) $x_*$ est minimum local strict s'il existe un voisinage $V_\varepsilon(x_*)$ de $x_*$ tel que*

$$f(x_*) < f(x),\ \forall x \in V_\varepsilon(x_*) \quad et \quad x \neq x_*.$$

### 1.2.1 Conditions nécessaires d'optimalité

**Théorème 1.2.1** ***([45]).*** *Soit $f : \mathbb{R}^n \to \mathbb{R}$ deux fois différentiable en $\overline{x}$. Supposons que $\overline{x}$ soit un minimum local. Alors $\nabla f(\overline{x}) = 0$ et $H(\overline{x})$ est semi définie positive.*

**Corollaire 1.2.1** *Soit $f : \mathbb{R}^n \to \mathbb{R}$ différentiable en $\overline{x}$, si $\overline{x}$ est un minimum local alors $\nabla f(\overline{x}) = 0$.*

### 1.2.2 Conditions suffisantes d'optimalité

**Théorème 1.2.2** ***([45]).*** *Soit $f : \mathbb{R}^n \to \mathbb{R}$.*
*(i) Supposons que $f$ est deux fois différentiable en $\overline{x}$. Si $\nabla f(\overline{x}) = 0$ et $H(\overline{x})$ est définie positive, alors $\overline{x}$ est minimum local strict.*
*(ii) Supposons que $f$ est convexe et différentiable. Alors $x_*$ est un minimum global de $f$ si et seulement si $\nabla f(x_*) = 0$.*
*(iii) Les résultats (i) et (ii) demeurent vrais si on remplace $\mathbb{R}^n$ par un ouvert $S$ de $\mathbb{R}^n$.*

**Corollaire 1.2.2** *([45]) Dans le cas où $f$ est convexe, alors tout minimum local est aussi global. De plus si $f$ est strictement convexe, alors tout minimum local devient non seulement global mais aussi unique.*

## 1.3 Modes de convergence

Les méthodes itératives utilisées pour résoudre un système linéaire approché conduisent à un résultat qui est toujours entaché d'erreur. Cette erreur doit être suffisamment petite pour que la solution approchée converge vers la solution exacte. Dans ce cas l'algorithme (ou la méthode) est dit convergent. La vitesse de convergence est un facteur important de la qualité des algorithmes. Si la vitesse de convergence est élevée, l'algorithme converge rapidement et le temps de calcul est moindre. Ces préoccupations de rapidité de

convergence ont conduit à diversifier les modes de convergence et à chercher des processus optimaux.

**Définition 1.3.1** *([7]) Soit $\{x_k\}$ une suite de vecteurs convergente vers $\overline{x}$ telle que $x_k \neq \overline{x}$ pour tout $k$. L'ordre de la convergence de la suite $\{x_k\}$ est le supremum des nombres $p$ non négative tel que:*

$$\overline{\lim_{k\to\infty}} \frac{\|x_{k+1} - \overline{x}\|}{\|x_k - \overline{x}\|^p} = \beta < \infty.$$

Si $p = 1$ et $\beta < 1$ alors la convergence est dite linéaire. Si $p > 1$ ou $p = 1$ et $\beta = 0$ alors la convergence est dite superlinéaire.

En particulier, si $p = 2$ et $\beta < \infty$ alors la convergence est dite quadratique.

## 1.4 Recherche linéaire

Notre problème consiste à minimiser une fonction $f : \mathbb{R}^n \to \mathbb{R}$ .

La recherche linéaire consiste à trouver $\alpha_k$ de façon à diminuer la fonction $f$ *suffisamment* le long de cette direction.

Ce "*suffisamment*" sera quantifié dans cette section dans la description des conditions dites de Wolfe, Armijo et Goldstein&Price (recherches linéaires inexactes) . Ensuite on procèdera à la mise à jour $x_{k+1} = x_k + \alpha_k d_k$.

### 1.4.1 Principe des recherches linéaires

Faire de la recherche linéaire veut dire déterminer un pas $\alpha_k$ le long d'une direction de descente $d_k$, autrement dit résoudre le problème unidimensionnel :

$$\min f(x_k + \alpha d_k), \quad \alpha \in \mathbb{R} \tag{1.4.1}$$

Notre intérêt pour la recherche linéaire ne vient pas seulement du fait que dans les applications on rencontre, naturellement, des problèmes unidimensionnels, mais plutôt du fait que la recherche linéaire est un composant fondamental de toutes les méthodes traditionnelles d'optimisation multidimensionnelle. D'habitude, nous avons le schéma suivant d'une méthode de minimisation sans contraintes multidimensionnelle :

en regardant le comportement local de l'objectif $f$ sur l'itération courante $x_k$, la méthode choisit la "direction du mouvement" $d_k$ (qui, normalement, est une direction de descente de l'objectif : $\nabla^T f(x).d < 0$) et exécute un pas dans cette direction :

$$x_k \longmapsto x_{k+1} = x_k + \alpha_k d_k$$

afin de réaliser un certain progrès en valeur de l'objective, c'est-à-dire, pour assurer que : $f(x_k + \alpha_k d_k) \leq f(x_k)$.
Et dans la majorité des méthodes le pas dans la direction $d_k$ est choisi par la minimisation unidimensionnelle de la fonction : $h_k(\alpha) = f(x_k + \alpha d_k)$.
Ainsi, la technique de recherche linéaire est un brick de base fondamentale de toute méthode multidimensionnelle.

### 1.4.2 Objectifs à atteindre

Il s'agit de réaliser deux objectifs

#### Le premier objectif

Consiste à *faire décroître $f$ suffisamment.* Cela se traduit le plus souvent par la réalisation d'une inégalité de la forme

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \text{"un terme négatif "} \tag{1.4.2}$$

Le terme négatif, disons $\nu_k$, joue un rôle-clé dans la convergence de l'algorithme utilisant cette recherche linéaire.
L'argument est le suivant.
Si $f(x_k)$ est minorée (il existe une constante $C$ telle que $f(x_k) \geq C$ pour tout $k$), alors ce terme négatif tend nécessairement vers zéro : $\nu_k \to 0$. C'est souvent à partir de la convergence vers zéro de cette suite que l'on parvient à montrer que le gradient lui-même doit tendre vers zéro. Le terme négatif devra prendre une forme bien particulière si on veut pouvoir en tirer de l'information.
En particulier, il ne suffit pas d'imposer $f(x_k + \alpha_k d_k) < f(x_k)$.

**Le second objectif**

Consiste d'empêcher le pas $\alpha_k > 0$ d'être trop petit, trop proche de zéro.
Le premier objectif n'est en effet pas suffisant car l'inégalité (1.4) est en général satisfaite par des pas $\alpha_k > 0$ arbitrairement petit.
Or ceci peut entraîner une "*fausse convergence*", c'est-à-dire la convergence des itérés vers un point non stationnaire, comme le montre l'observation suivante ([20]).
Si on prend

$$0 < \alpha_k \leq \frac{\varepsilon}{2^k \|d_k\|}$$

la suite $\{x_k\}$ générée par (1.1.2) est de Cauchy, puisque pour $1 \leq l < k$ on a:

$$\|x_k - x_l\| = \left\| \sum_{i=1}^{i=k-1} \alpha_i d_i \right\| \leq \sum_{i=1}^{i=k-1} \frac{\varepsilon}{2^i} \to 0, \text{ lorsque: } l \to \infty.$$

Donc $\{x_k\}$ converge, disons vers un point $\bar{x}$. En prenant $l = 1$ et $k \to \infty$ dans l'estimation ci-dessus, on voit que $\bar{x} \in \bar{B}(x_1, \varepsilon)$ et donc $\bar{x}$ ne saurait être solution s'il n'y a pas de solution dans $\bar{B}(x_1, \varepsilon)$.
On a donc arbitrairement forcer la convergence de $\{x_k\}$ en prenant des pas très petits.
Pour simplifier les notations, on définit la restriction de $f$ à la droite $\{x_k + \alpha d_k / \alpha \in \mathbb{R}\} \subset \mathbb{R}^n$ : comme la fonction :

$$h_k : \mathbb{R}_+ \to \mathbb{R}; \quad \alpha \longmapsto h_k(\alpha) = f(x_k + \alpha d_k) \tag{1.4.3}$$

### 1.4.3 Type de recherche linéaire

Il existe deux grandes classes de méthodes qui s'intéressent à l'optimisation unidimensionnelle :

a) Les recherches linéaires exactes.

b) Les recherches linéaires inexactes.

Les recherches linéaires exactes, malgré qu'elles n'aboutissent qu'à une solution optimale approchée, nécessitent beaucoup d'observations à chaque itération de l'algorithme principale.

Le mot exact prend sa signification dans le fait que si $f$ est quadratique la solution de la recherche linéaire s'obtient de façon exacte et dans un nombre fini d'itérations.
On aurait aimé restricter notre étude sur ce domaine sympathique, malheureusement, c'est très rare de rencontrer des problèmes quadratiques, et donc par exemple pour une fonction non linéaire arbitraire,
- la détermination de ces pas demande en général beaucoup de temps de calcul et ne peut de toute façon pas être faite avec une précision infinie,
- l'efficacité supplémentaire éventuellement apportée à un algorithme par une recherche linéaire exacte ne permet pas, en général, de compenser le temps perdu à déterminer un tel pas,
- les résultats de convergence autorisent d'autres types de règles (recherche linéaire inexacte), moins gourmandes en temps de calcul.
Au lieu de demander que $\alpha_k$ minimise , on préfère imposer des conditions moins restrictives, plus facilement vérifiées, qui permettent toutefois de contribuer à la convergence des algorithmes. En particulier, il n'y aura plus un unique pas (ou quelques pas) vérifiant ces conditions mais tout un intervalle de pas (ou plusieurs intervalles), ce qui rendra d'ailleurs leur recherche plus aisée. C'est ce que l'on fait avec les règles d'Armijo, de Goldstein et de Wolfe décrites dans la prochaine section.

**Intervalle de sécurité**

Dans la plupart des algorithmes d'optimisation modernes, on ne fait jamais de recherche linéaire exacte, car trouver $\alpha_k$ signifie qu'il va falloir calculer un grand nombre de fois la fonction $h_k$ et cela peut être dissuasif du point de vue du temps de calcul.
En pratique, on recherche plutôt une valeur de $\alpha^*$ qui assure une décroissance suffisante de $f$.
Cela conduit à la notion d'intervalle de sécurité.

**Définition 1.4.1** *([25]) On dit que $[\alpha_g, \alpha_d]$ est un intervalle de sécurité s'il permet de classer les valeurs de $\alpha$ de la façon suivante :*

- *Si $\alpha < \alpha_g$ alors $\alpha$ est considéré trop petit,*

– *Si* $\alpha_d \geq \alpha \geq \alpha_g$ *alors* $\alpha$ *est satisfaisant,*

– *Si* $\alpha > \alpha_d$ *alors est considéré trop grand.*

Le problème est de traduire de façon numérique sur $h_k$ les trois conditions précédentes, ainsi que de trouver un algorithme permettant de déterminer $\alpha_g$ et $\alpha_d$.

Il faut maintenant préciser quelles sont les relations sur $h_k$ qui vont nous permettre de caractériser les valeurs de $\alpha$ convenables, ainsi que les techniques utilisées pour réduire l'intervalle.

## 1.5 Recherches linéaires inexactes

On considère la situation qui est typique pour l'application de la technique de recherche linéaire à l'intérieur de la méthode principale multidimensionnelle.

Sur une itération $k$ de la dernière méthode nous avons l'itération courante $x_k \in \mathbb{R}^n$ et la direction de recherche $d_k \in \mathbb{R}^n$ qui est direction de descente pour notre objectif : $f : \mathbb{R}^n \to \mathbb{R}$ :

$$\nabla^T f(x_k).d_k < 0 \tag{1.5.1}$$

Le but est de réduire "de façon importante" la valeur de l'objectif par un pas $x_k \longmapsto x_{k+1} = x_k + \alpha_k d_k$ de $x_k$ dans la direction $d_k$.

Pour cela de nombreux mathématiciens (Armijo, Goldstein, Wolfe, Albaali, Lemaréchal, Fletcher...) ont élaboré plusieurs règles.

D'abord présentons le schéma d'une recherche linéaire inexacte.

### 1.5.1 Schéma des recherches linéaires inexactes

Elles reviennent à déterminer, par tâtonnement un intervalle $[\alpha_g, \alpha_d]$ , où $\alpha^* \in [\alpha_g, \alpha_d]$, dans lequel :

$$h_k(\alpha_k) < h_k(0) \quad (f(x_k + \alpha_k d_k) < f(x_k))$$

Le schéma de l'algorithme est donc :

**Algorithme 1.1 (Schéma général des recherches linéaires inexactes)**

**Etape 0: (initialisation)**

$\alpha_{g,1} = \alpha_{d,1} = 0$, choisir $\alpha_1 > 0$, poser $k = 1$ et aller à l'étape 1.

**Etape 1 :**

Si $\alpha_k$ est satisfaisant (suivant un certain critère) : STOP($\alpha^* = \alpha_k$).

Si $\alpha_k$ est trop petit (suivant un certain critère) :

nouvel intervalle : [ $\alpha_{g,k+1} = \alpha_k$ , $\alpha_{d,k+1} = \alpha_d$ ] et aller à l'étape 2.

Si $\alpha_k$ est trop grand (suivant un certain critère) :

nouvel intervalle : [ $\alpha_{g,k+1} = \alpha_g$ , $\alpha_{d,k+1} = \alpha_k$ ] et aller à l'étape 2.

**Etape 2 :**

Si $\alpha_{d,k+1} = 0$ déterminer $\alpha_{k+1} \in ]\ \alpha_{g,k+1}\ , +\infty\ [$

Si $\alpha_{d,k+1} \neq 0$ déterminer $\alpha_{k+1} \in ]\ \alpha_{g,k+1}\ , \alpha_{d,k+1}[$

remplacer $k$ par $k+1$ et aller à l'étape 1.

Il nous reste donc à décider selon quel(s) critère(s) $\alpha$ est trop petit ou trop grand ou satisfaisant.

### 1.5.2 La règle d'Armijo

La règle d'Armijo [4, 1966] impose une contrainte sur le choix de $\alpha_k$ suffisante pour minimiser localement $h$. Une condition naturelle est de demander que $f$ décroisse autant qu'une portion $\rho \in ]0, 1[$ de ce que ferait le modèle linéaire de $f$ en $x_k$. Cela conduit à l'inégalité suivante, parfois appelée *condition d'Armijo* ou *condition de décroissance linéaire* :

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \rho \alpha_k \nabla^T f(x_k) d_k \tag{1.5.2}$$

Elle est de la forme (1.4.2), car $\rho$ devra être choisi dans $]0, 1[$.

Le théorème suivant on assure l'existence du pas d'Armijo en posant quelques conditions sur la fonction $h_k$.

**Théorème 1.5.1** ***([25]).*** *Si $h_k : \mathbb{R}_+ \to \mathbb{R}$;définie par (1.4.3) $h_k(\alpha) = f(x_k + \alpha d_k)$ est continue et bornée inférieurement, si $d_k$ est une direction de descente en $x_k$ $(h'_k(0) < 0)$ et si $\rho \in ]0, 1[$ , alors l'ensemble des pas vérifiant la règle d'Armijo est non vide.*

Décrivons maintenant la règle de Goldstein&Price.

### 1.5.3 La règle de Goldstein & Price.

Etant données deux réels $\rho$ et $\sigma$ tels que $0 < \rho < \delta < 1$ ; les conditions de Goldstein & Price ([29, 1969]) sont :

$$f(x_k + \alpha d_k) \leq f(x_k) + \rho\alpha\nabla^T f(x_k)d_k \tag{1.5.3}$$

$$f(x_k + \alpha d_k) \geq f(x_k) + \delta\alpha\nabla^T f(x_k)d_k \tag{1.5.4}$$

autrement dit :

$$h_k(\alpha) \leq h_k(0) + \rho h'_k(0)\alpha \tag{1.5.5}$$

$$h_k(\alpha) \geq h_k(0) + \delta h'_k(0)\alpha \tag{1.5.6}$$

Le théorème suivant assure l'existence du pas de Goldstein&price en posant quelques conditions sur la fonction $h_k$.

**Théorème 1.5.2** ***([25]).*** *Si $h_k : \mathbb{R}_+ \to \mathbb{R}$; définie par (1.4.3) $h_k(\alpha) = f(x_k + \alpha d_k)$ est continue et bornée inférieurement, si $d_k$ est une direction de descente en $x_k$ et si $\rho \in ]0, 1[$ , $\delta \in ]\rho, 1[$ , alors l'ensemble des pas vérifiant la règle de Goldstein&Price (1.5.3)-(1.5.4) est non vide.*

### 1.5.4 La règle de Wolfe

Les conditions *(1.5.3)-(1.5.4)* "règle de Goldstein&Price" peuvent exclure un minimum ce qui est peut être un inconvénient. Les conditions de wolfe ([66, 1969]) remèdent à cet inconvénient.

Etant donnés deux réels $\rho$ et $\sigma$ tel que $0 < \rho < \sigma < 1$, ces conditions sont :

$$\begin{aligned} f(x_k + \alpha d_k) &\leq f(x_k) + \rho\alpha\nabla^T f(x_k)d_k & (1.5.7)\\ \nabla^T f(x_k + \alpha d_k)d_k &\geq \sigma\nabla^T f(x_k)d_k & (1.5.8) \end{aligned}$$

autrement dit :

$$h_k(\alpha) \leq h_k(0) + \rho h'_k(0)\alpha \quad (1.5.9)$$

$$h'_k(\alpha) \geq \sigma h'_k(0) \quad (1.5.10)$$

La règle de Wolfe fait appel au calcul de $h'_k$, elle est donc en théorie plus couteuse que la règle de Goldstein&Price.

Cependant dans de nombreuses applications, le calcul du gradient $\nabla f(x)$ représente un faible cout additionnel en comparaison du cout d'évaluation de $f(x)$, c'est pourquoi cette règle est très utilisée.

### Conditions de Wolfe fortes

Pour certains algorithmes (par exemple le gradient conjugué non linéaire) il est parfois nécessaire d'avoir une condition plus restrictive que (1.5.8).

Pour cela la deuxième condition (1.5.8) est remplacée par

$$\left|\nabla^T f(x_k + \alpha d_k)d_k\right| \leq \sigma \left|\nabla^T f(x_k)d_k\right| = -\nabla^T f(x_k)d_k.$$

On aura donc les conditions de Wolfe fortes ([67,1971]) :

$$\begin{aligned} f(x_k + \alpha d_k) &\leq f(x_k) + \rho\alpha\nabla^T f(x_k)d_k & (1.5.11)\\ \left|\nabla^T f(x_k + \alpha d_k)d_k\right| &\leq -\sigma\nabla^T f(x_k)d_k & (1.5.12) \end{aligned}$$

autrement dit :

$$h_k(\alpha) \leq h_k(0) + \rho h'_k(0)\alpha \quad (1.5.13)$$

$$|h'_k(\alpha)| \leq -\sigma h'_k(0) \quad (1.5.14)$$

où $0 < \rho < \sigma < 1$.

**Remarque 1.5.1** *La seconde condition (1.5.8) ou condition de courbure interdit le choix de pas trop petit pouvant entrainer une convergence lente ou prématurée.*

**Remarque 1.5.2** *On voit bien que les conditions de Wolfe fortes impliquent les conditions de Wolfe faibles. Effectivement (1.5.11) est équivalente à (1.5.7), tandis que (1.5.12)*

$$\begin{aligned} \left|\nabla^T f(x_k+\alpha_k d_k)d_k\right| &\leq -\sigma\nabla^T f(x_k)d_k \\ &\Leftrightarrow \sigma\nabla^T f(x_k)d_k \leq \nabla^T f(x_k+\alpha_k d_k)d_k \leq -\sigma\nabla^T f(x_k)d_k \\ &\Rightarrow \sigma\nabla^T f(x_k)d_k \leq \nabla^T f(x_k+\alpha_k d_k)d_k \end{aligned}$$

*Le théorème suivant assure l'existence du pas de Wolfe en posant quelques conditions sur la fonction $h_k$.*

**Théorème 1.5.3** ***([68]).*** *Si $h_k : \mathbb{R}_+ \to \mathbb{R}$; définie par (1.4.3) $h_k(\alpha) = f(x_k + \alpha d_k)$ est dérivable et bornée inférieurement, si $d_k$ est une direction de descente en $x_k$ et si $\rho \in ]0,1[\,, \sigma \in ]\rho,1[\,$, alors l'ensemble des pas vérifiant la règle de Wolfe (faible) (2.10)-(2.11) est non vide.*

### 1.5.5 Algorithmes d'Armijo, Goldstein-Price et de Wolfe

**Algorithme 1.2 (Règle d'Armijo, Goldstein-Price et Wolfe)**

**Etape 0 : (initialisation)**

$\alpha_{g,1} = \alpha_{d,1} = 0$, choisir $\alpha_1 > 0$, $\rho \in ]0,1[$ poser $k = 1$ et aller à l'étape 1.

**Etape 1 :**

si $\alpha_k$ vérifie les critères d'Armijo ou de Goldstein ou de Wolfe: STOP ($\alpha^* = \alpha_k$).

si non $\alpha_{d,k+1} = \alpha_d$ ,$\alpha_{g,k+1} = \alpha_k$ et aller à l'étape 2.

**Etape 2 :**

si $\alpha_{d,k+1} = 0$ déterminer $\alpha_{k+1} \in ]\ \alpha_{g,k+1}\ ,\ +\infty\ [$

si $\alpha_{d,k+1} \neq 0$ déterminer $\alpha_{k+1} \in ]\ \alpha_{g,k+1}\ ,\ \alpha_{d,k+1}\ [$

remplacer $k$ par $k+1$ et aller à l'étape 1.

CHAPITRE 2

# Méthodes de gradient conjugué et multigrille

Ce chapitre est consacré à une étude brève des méthodes du gradient conjugué et multigrille. Dans la première partie on décrit la méthode du gradient conjugué. Tout d'abord on donne son principe, présente les algorithmes du gradient conjugué linéaire et non linéaires et ensuite on cite des théorèmes garantissant la convergence de ces algorithmes. Dans la deuxième partie on décrit la méthode multigrille. On commence par donner son principe, l'algorithme des différentes variantes de la méthode multigrille, et on termine ce chapitre par l'étude de la convergence de cette dernière.

## 2.1 Méthode du gradient conjugué linéaire

### 2.1.1 Principe de la méthode du gradient conjugué linéaire:

Il s'agit d'une méthode *itérative* qui, appliquée à une *fonction quadratique strictement convexe* de $n$ variables, conduisent à l'*optimum* en $n$ étapes au plus.
Considérons une fonction quadratique quelconque :

$$q(x) = \frac{1}{2}x^T Ax + b^T x + c \quad \text{avec } x \in \mathbb{R}^n \tag{2.1.1}$$

• où $A$ est une matrice $n \times n$ définie positive et symétrique;

- où $b$ est un n-vecteur $\begin{pmatrix} b_1 \\ . \\ . \\ b_n \end{pmatrix}$;
- et une $c$ constante.

Le principe de la méthode du gradient conjugué consiste à partir d'un point $x_0$, à minimiser $q(x)$ successivement suivant $n$ directions linéairement indépendantes $d_1, d_2, .., d_n$ possédant la propriété d'être *mutuellement conjuguées* par rapport à la matrice $A$ de la forme quadratique $q(x)$.

Donnons la définition de "*conjugaison*" :

**Définition 2.1.1** *([56]) Soit $A$ une matrice symétrique $n \times n$, définie positive. On dit que deux vecteurs $x$ et $y$ de $\mathbb{R}^n$ sont $A-$conjugués (ou conjugués par rapport à $A$) s'ils vérifient*

$$x^T A y = 0 \tag{2.1.2}$$

*Une telle famille est libre. Un exemple d'une telle famille est donné par une famille de vecteurs propres orthogonaux de $A$.*

**Remarque 2.1.1** *Dans l'algorithme du gradient conjugué linéaire, à chaque étape $k+1$ la direction $d_{k+1}$ est obtenue par combinaison linéaire du gradient $-\nabla q(x_{k+1})$ en $x_{k+1}$, et des directions précédentes $d_0, d_1, ..., d_k$, les coefficients de la combinaison linéaire étant choisis de telle sorte que $d_{k+1}$ soit conjuguée par rapport à toutes les directions précédentes, autrement dit :*

$$\begin{aligned} d_{k+1} &= -\nabla q(x_{k+1}) + \beta_{k+1} d_k \\ \text{avec } \beta_{k+1} \text{ tel que } d_{k+1}^T A d_k &= 0, \end{aligned}$$

*on en déduit :*

$$\begin{aligned} d_{k+1}^T A d_k &= 0 \Rightarrow \left(-\nabla q(x_{k+1}) + \beta_{k+1} d_k\right)^T A d_k = 0 \\ &\Rightarrow -\nabla^T q(x_{k+1}) A d_k + \beta_{k+1} d_k^T A d_k = 0 \\ &\Rightarrow \beta_{k+1} = \frac{\nabla^T q(x_{k+1}) A d_k}{d_k^T A d_k} \end{aligned}$$

*Comme*

$$\begin{aligned} d_k^T \nabla q(x_k) &= \left(-\nabla q(x_k) + \beta_{k+1} d_{k-1}\right)^T \nabla q(x_k) \\ &= -\nabla q(x_k)^T \nabla q(x_k) + \beta_{k+1} d_{k-1}^T \nabla q(x_k) \\ &= -\left\|\nabla q(x_k)\right\|^2 \quad (car\ d_{k-1}^T \nabla q(x_k) = 0\ ) \end{aligned}$$

$$d_k^T \nabla q(x_k) = -\left\|\nabla q(x_k)\right\|^2 < 0$$

*donc .$d_k$ est une direction de descente.*

### 2.1.2 Algorithme du gradient conjugué linéaire

Notons $g_k = \nabla q(x_k)$ le gradient de la fonction $q$ en $x_k$, l'algorithme du gradient conjugué est comme suit :

**Algorithme 2.1 (Algorithme du gradient conjugué linéaire**

**Etape 0: (initialisation)**

Soit $x_0$ le point de départ, $g_0 = \nabla q(x_0) = Ax_0 + b$, poser $d_0 = -g_0$

poser $k = 0$ et aller à l'étape 1.

**Etape 1 :**

si $\|g_k\| = 0$ : STOP ( $x^* = x_k$)."Test d'arrêt"

si non aller à l'étape 2.

**Etape 2 :**

Définir $x_{k+1} = x_k + \alpha_k d_k$ avec :

$$\alpha_k = \frac{-d_k^T g_k}{d_k^T A d_k} \tag{2.1.3}$$

$$d_{k+1} = -g_{k+1} + \beta_{k+1} d_k \tag{2.1.4}$$

$$\beta_{k+1} = \frac{g_{k+1}^T A d_k}{d_k^T A d_k} \tag{2.1.5}$$

Poser $k = k + 1$ et aller à l'étape 1.

**Remarque 2.1.2** *On peut remarquer que la consommation mémoire de l'algorithme est minime : on doit stocker les quatre vecteurs $x_k$, $g_k$, $d_k$, $Ad_k$ ( bien sur $x_{k+1}$ prend la place de $x_k$ au niveau de son calcul avec des remarques analogues pour $g_{k+1}$, $d_{k+1}$, $Ad_{k+1}$ et les scalaires $\alpha_k$, $\beta_{k+1}$).*

### 2.1.3 Convergence de la méthode du gradient conjugué linéaire

Le résultat suivant ([43, 1969]), donne une estimation de la décroissance de la norme de l'erreur $x_k - x^*$, en terme $l$ du conditionnement de la matrice $A$:

$$\kappa(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$$

où : $\lambda_{\max}(A)$, $\lambda_{\min}(A)$ sont respectivement, les plus grande, petite valeurs propres de la matrice $A$

La norme utilisée est la suivante :

$$\|x\|_A = \left(x^T A x\right)^{\frac{1}{2}}$$

**Théorème 2.1.1** *([43]) Soit $\kappa$ le conditionnement de $A$. La suite $(\{x_k\}_{k\in\mathbb{N}})$ générée par l'algorithme du gradient conjugué vérifie l'estimation*

$$\|x_{k+1} - x^*\|_A \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \|x_1 - x^*\|_A .$$

## 2.2 Méthode du gradient conjugué non linéaire

Ces méthodes sont des extensions de la méthode du gradient conjugué linéaire du cas quadratique.

### 2.2.1 Différentes versions de la méthode du gradient conjugué non linéaire

#### Méthode de Hestenes-Stiefel

Cette méthode fut découverte par Hestenes et Stiefel ([34, 1952]) dont la variante est :

$$\beta_k^{HS} = \frac{g_k^T(g_k - g_{k-1})}{d_{k-1}^T(g_k - g_{k-1})} \tag{2.2.1}$$

**Méthode de Fletcher-Reeves**

La méthode de Fletcher-Reeves ([24, 1964]) est une extension directe de la méthode du gradient conjugué linéaire au cas des fonctions quelconques. Appliquée à une fonction quadratique, elle est identique au gradient conjugué linéaire où la variante est :

$$\beta_k^{FR} = \frac{\|g_k\|^2}{\|g_{k-1}\|^2} \tag{2.2.2}$$

**Méthode de Polak-Ribière-Polyak**

Cette méthode fut découverte par Polak, Ribière ([50, 1969]) et Polyak ([51, 1969]) dont la variante est :

$$\beta_k^{PRP} = \frac{g_k^T(g_k - g_{k-1})}{\|g_{k-1}\|^2} \tag{2.2.3}$$

**Méthode de la descente conjuguée**

Cette méthode a été proposée en 1987 par Fletcher ([22]) dont la variante est :

$$\beta_{k+1}^{CD} = \frac{\|g_{k+1}\|^2}{-d_k^T g_k} \tag{2.2.4}$$

**Méthode de Liu-Storey**

Cette méthode fut découverte par Liu et Storey ([42]) dont la variante est :

$$\beta_k^{LS} = \frac{g_k^T(g_k - g_{k-1})}{d_{k-1}^T g_{k-1}} \tag{2.2.5}$$

**Méthode de Dai-Yuan**

Cette méthode fut découverte par Dai et Yuan ([18, 1999]), dont la variante est :

$$\beta_{k+1}^{DY} = \frac{\|g_{k+1}\|^2}{d_k^T\left[g_{k+1} - g_k\right]} = \frac{\|g_{k+1}\|^2}{d_k^T y_k} \tag{2.2.6}$$

**Méthode de Harger-Zhang**

Cette méthode était découverte par Harger et Zhang ([32, 2005]), dont la variante est :

$$\begin{aligned} \beta_{k+1}^{HZ} &= (y_k - 2d_{k-1}\frac{\|y_k\|^2}{d_{k-1}^T y_k})^T \frac{g_{k+1}}{d_{k-1}^T y_k} \\ \text{où } y_k &= g_{k+1} - g_k \end{aligned} \tag{2.2.7}$$

### 2.2.2 Les directions des méthodes du gradient conjugué non linéaires sont-elles des directions de descente de $f$ ?

Powell ([53, 1984]) a démontré la satisfaction de la propriété de descente de la direction de la méthode de Fletcher-Reeves avec recherche linéaire exacte.
Le premier théorème démontrant la satisfaction de la propriété de descente de la direction d'une méthode du gradient conjugué non linéaire avec une recherche linéaire inexacte était établi par Albaali ([1, 1985]) pour la méthode de Fletcher-Reeves avec la recherche de Wolfe forte où $\sigma \leq \frac{1}{2}$. Gilbert et Nocedal ([26, 1992]) ont généralisé ce résultat pour tout algorithme du gradient conjugué dont

$$|\beta_k| \leq \beta_k^{FR}$$

Aucun théorème n'a était établi afin de démontrer la satisfaction de la propriété de descente pour la méthode de Polak-Ribière-Polyak non linéaire.
Fletcher ([22, 1987]) a démontré que la méthode de la descente conjuguée est une méthode de descente si le pas $\alpha_k$ est déterminé par la règle de Wolfe forte (1.5.11)-(1.5.12) avec $\sigma \leq \frac{1}{2}$. Dai et Yuan ([16, 1996]) ont démontré que cette méthode avec la règle de Wolfe faible (1.5.7)-(1.5.8) où $0 < \sigma < 1$, génère des directions de descente suffisante à chaque itération $k \geq 1$.
Dai et Yuan ([17, 1998]) ont démontré qu'à chaque itération $k \geq 1$, la direction recherchée par la méthode de Dai-Yuan avec la recherche de Wolfe faible (1.5.7)-(1.5.8), est de descente si la fonction objectif $f$ est strictement convexe. Ils ([19, 2001]) ont généralisé ce résultat pour toute fonction régulière.

Harger-Zhang ([32, 2005]) ont démontré la satisfaction de la propriété de descente de la direction de la méthode de Harger-Zhang avec la règle de Wolfe forte (1.5.11)-(1.5.12).

### 2.2.3 Algorithmes des méthodes HS, FR, PRP, CD, LS, DY et HZ

**Algorithme 2.2 Méthode de HS, FR, PRP, CD, LS, DY et HZ**

**Etape0: (initialisation)**

Soit $x_0$ le point de départ, $g_0 = \nabla f(x_0)$, poser $d_0 = -g_0$

Poser $k = 0$ et aller à l'étape 1.

**Etape 1 :**

Si $g_k = 0$ : STOP ( $x^* = x_k$)."Test d'arrêt"

Sinon aller à l'étape 2.

**Etape 2 :**

Définir $x_{k+1} = x_k + \alpha_k d_k$ avec : $\alpha_k = \min_{\alpha>0} f(x_k + \alpha d_k)$

$$d_{k+1} = -g_{k+1} + \beta_{k+1} d_k$$

où $\beta_{k+1}$ est définie par l'une des formules (2.2.1)-(2.2.7)

Poser $k = k + 1$ et aller à l'étape 1.

### 2.2.4 Résultats de convergence des méthodes du gradient conjugué non linéaire avec une recherche linéaire inexacte

Dans cette section on va essayer de présenter une synthèse sur les différents résultats de convergence des méthodes du gradient conjugué pour la minimisation des fonctions sans contraintes. Ces méthodes seront utilisées avec une recherche linéaire inexacte (Wolfe forte ou faible). L'analyse couvre cinq classes de méthodes qui sont globalement convergentes pour des fonctions régulières non nécessairement convexes.

**Proposition 2.1.1 ([48])**

*(i) L'ensemble* $L := \{x \in \mathbb{R}^n; f(x) \leq f(x_1)\}$ *est borné; où* $x_0 \in \mathbb{R}^n$ *est le point initial.*

*(ii) Sur un voisinage* $N$ *de* $L$*, la fonction objectif* $f$ *est continûment différentiable et son gradient est lipschitzien i.e*

$$\exists L > 0 \text{ tel que } \quad \|g(x) - g(\tilde{x})\| \leq L \|x - \tilde{x}\|, \forall x, \tilde{x} \in \mathcal{N} \tag{2.2.8}$$

**Remarque 2.2.1** *Notons que ces suppositions impliquent qu'il existe $\lambda > 0$ tel que*

$$\|g(x)\| \leq \lambda, \forall x \in \mathcal{L} \tag{2.2.9}$$

**Définition 2.2.1** ***([48])*** *On dit que $d_k$ est une direction de descente suffisante si*

$$d_k^T g_k \leq -C \|g_k\|^2 ; \text{ où } C > 0 \tag{2.2.10}$$

Présentons maintenant un théorème fondamental qui assure la satisfaction de la condition de Zoutendijk, pour toute méthode du gradient conjugué non linéaire, dans laquelle le pas $\alpha_k$ est déterminé par la règle de Wolfe faible (1.5.7)-(1.5.8).
Ce théorème était démontré par Zoutendijk ([72, 1960]), ([71, 1970]) et Powell ([54, 1971]).

**Théorème 2.2.1** *([69]). Considérons une méthode du gradient conjugué dans laquelle $d_k$ est une direction de descente et le pas $\alpha_k$ est déterminé par la règle de Wolfe faible (1.5.7)-(1.5.8) avec $\sigma \in \left]0, \frac{1}{2}\right[$. Considérons aussi que la supposition 2.1.1 soit satisfaite. Alors pour une telle méthode la condition de Zoutendijk suivante :*

$$\sum_{k \geq 1} \cos^2 \theta \|g_k\|^2 < \infty \tag{2.2.11}$$

*est vérifiée.*

### Convergence de la méthode de Fletcher-Reeves

Le premier résultat de convergence de la méthode du gradient conjugué non linéaire ( version Flecher Reeves) avec des recherches linéaires inexactes (recherche linéaire inexacte de Wolfe forte (4.9)-(4.10) où $\sigma < \frac{1}{2}$) était démontré par Al-Baali ([1, 1985]). Touati Ahmed et Storey ([62,1990]) ont généralisé ce résultat pour $0 \leq \beta_k \leq \beta_k^{FR}$. Gilbert et Nocedal ([26, 1992]) ont généralisé ce résultat pour $|\beta_k| \leq \beta_k^{FR}$.

**Théorème 2.2.2** *([1]) Supposons que l'hypothèse 2.1.1 soit satisfaite. Considérons la méthode du gradient conjugué non linéaire de Fletcher-Reeves où le pas $\alpha_k$ satisfait aux conditions de Wolfe forte (1.5.9)-(1.5.10) où $0 < \sigma < \frac{1}{2}$. Alors cette méthode est globalement convergente, dans le sens suivant:*

$$\lim_{k \to \infty} \inf \|g_k\| = 0 \tag{2.2.12}$$

**Remarque 2.2.2** *Lui, Han et Yuan ([41, 1995]) ont prouvé le résultat (4.19) pour cette méthode si* $\rho = \sigma = \frac{1}{2}$. *Dai et Yuan ([15, 1996]) l'ont encore démontré mais par une approche plus simple.*

**Remarque 2.2.3** *La méthode de Fletcher-Reeves possède de bonne propriétés théoriquement, mais en pratique, elle converge parfois lentement, et même prématurément. En effet, si les pas sont trop petits, il se peut que ce comportement s'élargit pour un grand nombre d'itérations, et c'est ce qu'il nous oblige à réinitialiser en posant* $\beta_k^{FR} = 0$. *Powell ([55,1977]) est le premier qui a observé ce comportement, ainsi il a donné un contre exemple avec une recherche linéaire exacte. On évite cet inconvénient en réinitialisant chaque n itération par exemple. Nemirovsky et Yudin ([47, 1983]) ont démontré par un contre exemple que la méthode de Fletcher-Reeves converge plus lentement que la méthode de Steepest descent([14, 1847]). (cas d'une fonction quadratique avec recherche linéaire exacte).*

### Convergence de la méthode de PRP

La convergence de cette méthode est assurée pour une fonction fortement convexe avec recherche linéaire, mais si $f$ n'est pas convexe elle ne converge pas.
Pour remédier à cet inconvénient Powell ([52, 1986]) a modifié le choix de $\beta_k^{PRP}$ afin d'assurer la convergence avec une recherche linéaire exacte.
Gilbert et Nocedal ([26, 1992]) ont proposé une nouvelle méthode qui converge dans le sens (4.19) avec ou bien une recherche linéaire exacte ou inexacte.
Le résultat suivant est du à Polak et Ribière ([50, 1969]).

**Théorème 2.2.3** ***([50]).*** *Si* $f$ *est fortement convexe, continûment différentiable avec un gradient lipschitzien, alors la méthode de Polak Ribière avec recherche linéaire exacte génère une suite* $\{x_k\}$ *convergeant vers l'unique point* $x^*$ *réalisant le minimum de* $f$.

**Remarque 2.2.4** *Si* $f$ *n'est pas convexe, la méthode de Polak-Ribière-Polyak peut ne pas converger. Powell ([53, 1984]) a donné un exemple de fonction (de 3 variables, deux fois continûment différentiable) pour laquelle l'algorithme génère une suite* $\{x_k\}$ *dont aucun*

*des points d'adhérence n'est stationnaire. En 1986 ([52]), il a modifié la variante* $\beta_k^{PRP}$ *en évitant les valeurs négatifs, autrement dit si à l'itération* $k$ *on a : si* $\beta_k^{PRP} < 0$, *on redémarre en posant* $\beta_k^{PRP} = 0$ *(prendre la direction de la plus profonde pente)*

$$\beta_k = \max\left\{0, \beta_k^{PRP}\right\}$$

*Ce choix assure la convergence si le pas* $\alpha_k$ *est déterminé par la règle de Wolfe forte.*

**Convergence de la méthode de la descente conjuguée**

Yuan ([69, 1993]) a démontré la convergence au sens (4.19) de cette méthode avec un pas satisfaisant aux conditions de Wolfe faible (1.5.7)-(1.5.8) pour $\sigma < \frac{1}{2}$. Dai et Yuan ([16,1996]) ont élargit ce résultat pour $\sigma < 1$.

**Théorème 2.2.4** *([16]). Supposons que l'hypothèse 2.1.1 est satisfaite. Toute méthode du type (4.2) et (4.3) dons laquelle* $\beta_k$ *vérifie (4.21) et le pas* $\alpha_k$ *est déterminé par la règle de Wolfe faible (1.5.7)-(1.5.8) où* $0 < \sigma < 1$; *est de descente convergente, dans le sens où*

$$\lim_{k\to\infty} \inf \|g_k\| = 0$$

**Convergence de la méthode de Dai-Yuan**

Dai-Yuan ([19, 2001]) ont démontré la convergence de la méthode de Dai-Yuan au sens (4.18) si le pas $\alpha_k$ est déterminé par la règle de wolfe faible.

**Théorème 2.2.5** *([19]). Supposons que la proposition 2.1.1 est satisfaite. La suite* $\{x_k\}$ *générée par l'algorithme de Dai-Yuan avec un pas satisfaisant aux conditions de Wolfe faible (1.5.7)-(1.5.8) converge dans le sens*

$$\lim_{k\to\infty} \inf \|g_k\| = 0$$

**Convergence de la méthode de Harger-Zhang**

Harger-Zhang ([33, 2006]) ont démontré la convergence de la méthode de Harger-Zhang au sens (4.18) si le pas $\alpha_k$ est déterminé par la règle de wolfe forte.

**Théorème 2.2.6** *([33]). Supposons que la proposition 2.1.1 est satisfaite. La suite $\{x_k\}$ générée par l'algorithme de Harger-Zhang avec un pas satisfaisant aux conditions de Wolfe forte (1.5.7)-(1.5.8) converge dans le sens*

$$\lim_{k\to\infty} \inf \|g_k\| = 0$$

## 2.3 Méthode multigrille

Soit $A$ une matrice $n \times n$ réelle symétrique définie positive et $b \in \mathbb{R}^n$. Et soit $\mathbb{R}^n$ l'espace de Hilbert munit du produit scalaire

$$(x, y) \to \langle x, y \rangle_A := x^T A y$$

et la norme

$$\|x\|_A := \sqrt{x^T A x}.$$

On bute à résoudre le problème suivant :

$$\text{Trouver } x \in \mathbb{R}^n \text{ tel que : } Ax = b. \tag{2.3.1}$$

Alors, la solution de (2.3.1) est équivalente au problème de minimisation suivant :

$$\text{Trouver } \hat{x} \in \mathbb{R}^n \text{ tel que : } \hat{x} = \min_{x\in\mathbb{R}^n} q(x), \text{ où } q(x) := \frac{1}{2}\langle x, x \rangle_A - b^T x. \tag{2.3.2}$$

**Définition 2.3.1** *([58]) Supposons qu'il existe une suite de grilles de dimension fini $\Omega_i$, $i = 1, ..., l$, de taille $n_i = |\Omega_i|$ tels que $n = n_l > n_{l-1} > ... > n_1$. On généralise les opérateurs de restrictions, prolongement et de lissage par :*
*$I_i^{i-1}$ L'opérateur de restriction*

$$I_i^{i-1} : \mathbb{R}^{n_i} \to \mathbb{R}^{n_{i-1}}, \quad \textit{pour } i = l, ..., n,$$

*opère le passage de la grille grossière (i) à la grille fine* $(i-1)$

$I_{i-1}^{i}$ *L'opérateur de prolongement*

$$I_{i-1}^{i} = (I_{i}^{i-1})^{T} : \mathbb{R}^{n_i} \rightarrow \mathbb{R}^{n_{i-1}}, \quad pour \quad i = l, ..., n,$$

*opère le passage de la grille fine à la grille grossière.*

*3) S L'opérateur de lissage qui permet de résoudre le système (2.3.1) sur une grille, généralement on prend la méthode de Gauss-Siedel, Jacobi ou la méthode du gradient conjugué.*

### 2.3.1 Principe de la méthode multigrille

La méthode itérative multigrille est un outil puissant pour la résolution numérique des grands systèmes linéaires creux résultant de la discrétisation des équations aux dérivées partielles elliptiques. Pratiquement, elles ont été introduites en 1972 par Brandt [ 2 ] . Ces méthodes peuvent résoudre les EDP elliptiques discrétisées en $\circ(N)$ opérations où N les le nombre des points de discrétisation.

Dans une itération multigrille (cycle multigrille ), l'équation est tout d'abord détendue sur la grille d'origine ( fine) par une méthode de relaxation pour lisser l'erreur et la rendre prête à résoudre sur une grille plus grossière. Puisque les calculs sur cette grille sont moins gourmands, on peut calculer un terme de correction, puis le transférer à la grille fine et l'ajouter à la solution approchée.

Enfin, l'équation est détendue à nouveau sur la grille fine (généralement par la même méthode de relaxation), pour lisser les modes d'erreur. L'ensemble de la procédure peut être décrit schématiquement sur un diagramme de la forme de la lettre latine V , d'où le

nom de "V cycle ".

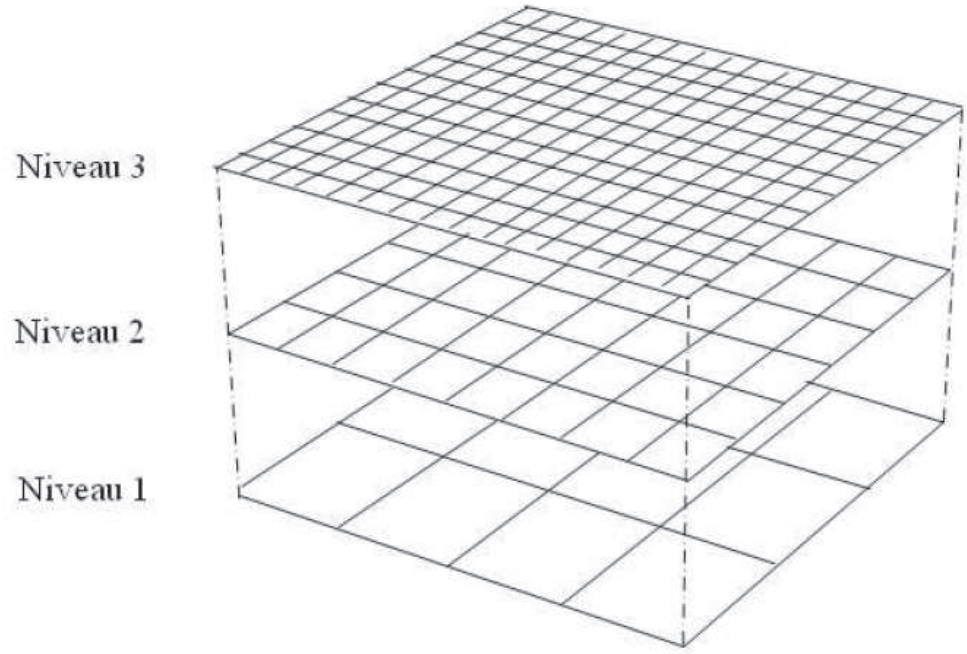


Figure 2.3.1 : Exemple de grilles

### 2.3.2 Algorithme de la méthode multigrille

L'algorithme de V-cycle est comme suit:

**Algorithme 2.3** ($u_h = V - Cycle(A_h, u_h, 0, f_h, \nu_1, \nu_2)$)

**Etape 1. Prélissage:** $u_h :=$ lisseur$\nu_1(A_h, u_h, 0, f_h)$

**Etape 2. Calcul du résidu:** $r_h = f_h - A_h u_h$

**Etape 3. :** $r_H = I_h^H r_h$

**Etape 4.** Si $(H == h_0)$ Résoudre: $A_H \delta H = r_H$

sinon : $\delta H = V - cycle(A_H, 0, r_H)$

**Etape 5. Correction:** $uh := uh + I_H^h \delta H$

**Etape 6. Postlissage:** $uh :=$lisseur$\nu_2(A_h, u_h, f_h)$

**Remarque 2.3.1** *En poussant le concept de l'algorithme de V-cycle jusqu'au bout, c'est-à-dire sans préciser les valeurs de $\nu_1$ et $\nu_2$, on peut voir que cet algorithme manipule des*

*niveaux de plus en plus petits en termes de nombre d'inconnus. Inévitablement, il arrive un moment où le système ne possède plus qu'un seul inconnu. Cette résolution se fait directement. Le test d'arrêt sur le niveau, non présenté dans l'algorithme, est alors indispensable. De plus, ce dernier permet à l'utilisateur de stopper l'algorithme à un niveau donné, là où il considère que le lisseur est suffisamment précis ou que la résolution directe est possible à moindre coût.*

**Algorithme 2.4** ($u_h = MG(A_h, u_h, 0, f_h, \nu_1, \nu_2, \gamma)$)

**Etape 1. Prélissage:** $u_h := \text{lisseur}\nu_1(A_h, u_h, 0, f_h)$

**Etape 2. Calcul du résidu:** $r_h = f_h - A_h u_{h.}$

**Etape 3. :** $r_H = I_h^H r_h$

**Etape 4.** Si $(H == h_0)$ résoudre: $A_H \delta H = r_H$

sinon $\delta H = MG\gamma(A_H, 0, r_H, \nu_1, \nu_2, \gamma)$

**Etape 5. Correction:** $uh := uh + I_H^h \delta H$

**Etape 6. Postlissage:** $uh :=$lisseur $\nu_2(A_h, u_h, f_h)$.

**Remarque 2.3.2** *Notons maintenant qu'il y a un nouveau paramètre $\gamma$, qui détermine combien de fois MG est réitéré dans la ligne 6 de l'algorithme MG. Ce qui veut dire que pour calculer $\delta H$ on réitérait $MG(A_H, 0, r_H, \nu_1, \nu_2)$, $\gamma$ fois. L'estimation initiale pour l'itération est $\delta H = 0$. Ainsi on distingue les cas suivant : Le cas $\gamma = 1$ donne le v-cycle du MG. Le cas $\gamma = 2$ est connu sous le W-cycle de MG. Le cas $\gamma = 3$ est rarement utilisé. (figure 2.3.2)*

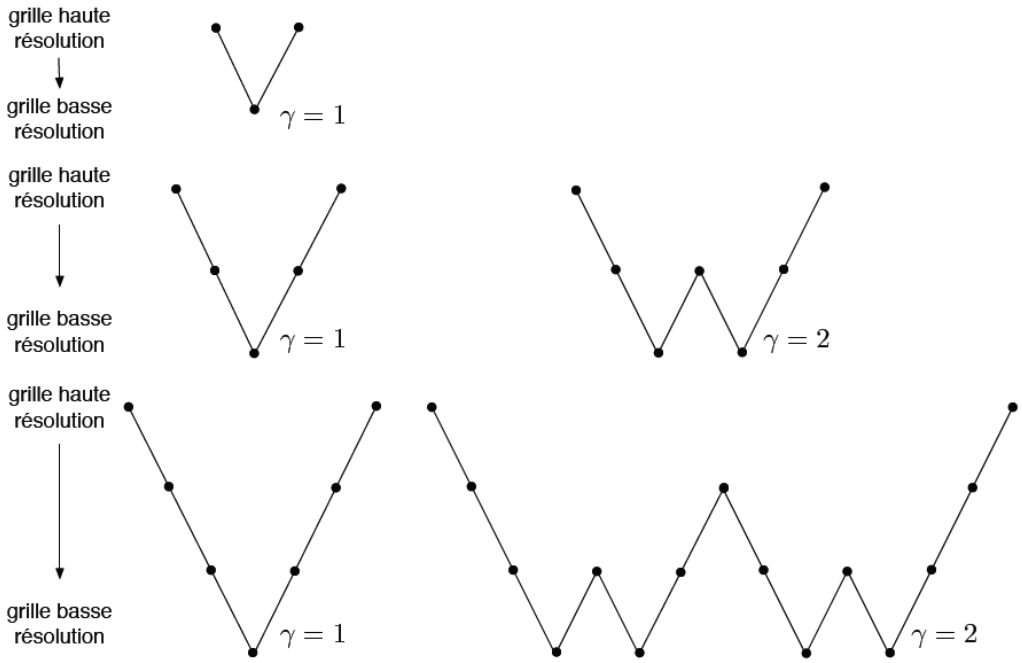


Figure 2.3.2 : Exemples de parcours des differentes grilles dans le cas de 2,3 ou 4 grilles

### 2.3.3 Analyse de la convergence

Soit $D$ la diagonale de la matrice $A$. Soient les notations :

$$\|x\|_D = \sqrt{(Dx, x)} = \left\| D^{\frac{1}{2}} x \right\|_2 . \tag{2.3.3}$$

et la norme suivante :

$$\|e\|_{A^h D^{-1} A_h} = \sqrt{(D^{-1} A_h e, A_h e)} = \|A_h e\|_{D^{-1}} . \tag{2.3.4}$$

Afin de simplifier les notations, $\|A_h e\|_{D^{-1}}$ dénote la norme de $e$.

**Définition 2.3.2** ***(Propriété de lissage) ([58])*** *On dit que le lisseur $(S^k)$ vérifie la propriété de lissage si*

$$\left\| S_h e^h \right\|_{A_h}^2 \leq \left\| e^h \right\|_{A_h}^2 - \alpha \left\| A e^h \right\|_{D^{-1}}^2 . \tag{2.3.5}$$

*indépendamment de $h$ avec $\alpha > 0$ et $e^h \in \Omega_h$.*

**Définition 2.3.3** ***(Propriété d'approximation) ([58])*** *On dit que la propriété d'approximation est vérifiée si :*

$$\min_{u_H \in \Omega_H} \left\| e^h - I_H^h e^H \right\|_D^2 \leq \beta \left\| e^h \right\|_{A_h}^2 . \tag{2.3.6}$$

*indépendamment de $h$ avec $\beta > 0$ et $e^h \in \Omega_h$.*

**Supposition 2.2.** **([58])** Supposons que $A$ est une matrice symétrique définie positive et que les opérateurs de prolongation et de restriction vérifient:

$$I_H^h = 2^d (I_h^H)^T. \tag{2.3.7}$$

où $d$ est la dimension de l'espace.

**Théorème 2.3.1** ***([58]) (Taux de convergence de la méthode multigrille)****Supposons que la supposition 2.2 et les propriétés de lissage et d'approximation (2.3.5)-(2.3.6) sont vérifiées pour un certain lisseur, où $\alpha > 0$ et $\beta > 0$. Alors $\alpha \leq \beta$, et la méthode multigrille converge et en plus on a*

$$\left\| S_h I_h^H \right\|_{A_h} \leq \sqrt{1 - \frac{\alpha}{\beta}}. \tag{2.3.8}$$

**Remarque 2.3.3** *Le premier résultat de convergence de la méthode multigrille (W-cycle) a été donné par Hackbusch [31, 1983] et pour V-cycle par Braess et Hackbusch [9,1983], mais leur démonstration est très compliquée et avec beaucoup de conditions. Par la suite Bank and Douglas [5, 1985] et H. Yserentant [70, 1992] ont démontré la convergence de la méthode multigrille avec moins de conditions mais la démonstration reste toujours compliquée.*

CHAPITRE 3

# Combinaisons des méthodes multigrille et gradient conjugué

Dans ce chapitre on va comparer deux approches de combinaison de la méthode multigrille et la méthode du gradient conjugué pour la résolution du problème du Laplacien par sous domaine avec les conditions aux limites de Dirichlet homogènes [39,1988], discrétisé par le schéma classique des différences finies à cinq points.

La première combinaison a été proposé par Braess [8, 1985] (MG-CG). L'idée consiste à utiliser la méthode du gradient conjugué dans les étapes de pré-lissage et post-lissage de la méthode multigrille. Cet usage ne procure pas seulement le lissage, mais en même temps il améliore le calcul de la correction sur la grille grossière ce qui accélére la vitesse de convergence de l'algorithme du multigrille..

Une nouvelle approche a été proposée par Pflaum [49,2008] (MGcgM). Elle consiste à construire de nouvelles corrections en se basant sur les directions du gradient conjugué.

Notre étude est la comparaison et la consolidation par des simulations numérique (SCILAB) illustrant que la méthode MG-CG converge plus rapidement que les méthodes MGgM et MGcgM pour ce problème.

## 3.1 Problématique

Soit $\Omega$ un ouvert borné de frontière $\partial\Omega$ assez régulière.

Considérons le problème du Laplacien avec les conditions aux limites de Dirichlet homogènes :

$$\begin{cases} -\triangle U = f \quad \text{sur } \Omega, \\ U = 0 \quad \text{sur } \partial\Omega; \end{cases} \tag{3.1.1}$$

où $f \in L^2(\Omega)$. On décompose le domaine $\Omega$ de la façon suivante :

$$\Omega = \Omega_1 \cup \Omega_2 \cup \Sigma \quad \text{tel que :} \begin{cases} \Omega_i \subset \Omega \ ; \quad \Omega_i \cap \Omega_j = \varnothing \quad i \neq j, \\ \Sigma = \partial\Omega_i \cap \partial\Omega_j \ , \\ \text{et } \Gamma_i = \partial\Omega_i \cap \Omega; \end{cases} \tag{3.1.2}$$

**Problème discret**

Prenons $\Omega = ]0,1[ \times ]0,1[ \subset \mathbb{R}^2$ et $\Omega_{k=} ]0,1[ \times \left]0, \frac{1}{2}\right[$, $k = 1, 2$; sur chaque sous domaine $\Omega_k$. On se donne un maillage régulier du pas $h = \frac{1}{n+1}$ et on discrétise le problème (3.1.1) par le schéma classique des différences finies à cinq points.

Pour chaque point intérieur de chacun de sous domaines $\Omega_k$ on a alors :

$$(-\triangle U^k)i, j = \frac{-U^k_{i,j-1} - U^k_{i-1,j} + 4U^k_{i,j} - U^k_{i,j+1} - U^k_{i+1,j}}{h^2} = f^k_{i,j}; \tag{3.1.3}$$

On obtient un système linéaire sous la forme suivante :

$$AX = B, \tag{3.1.4}$$

où $A = \begin{pmatrix} A_1 & 0 & -B_1^T \\ 0 & A_2 & -B_2^T \\ -B_1 & -B_2 & A' \end{pmatrix}$ est une matrice par block, tel que $A_i = \begin{pmatrix} a_{11} & -I & & \\ -I & \ddots & \ddots & \\ & \ddots & \ddots & -I \\ & & -I & a_{nn} \end{pmatrix}$,

$I$ étant l'identité,

$$a_{ii} = A' = \begin{pmatrix} 4 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 \end{pmatrix}, B_1^T = \begin{pmatrix} & & \\ & & \\ 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix}, B_2^T = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \\ & & \\ & & \end{pmatrix},$$

$X^T = (x_1, x_2, x)^T \in \mathbb{R}^n$, $B = (b_1, b_2, b)^T \in \mathbb{R}^n$ et $B = h^2 f(x_i\ , y_{i)}$.

Le problème (3.1.4) peut s'écrire sous la forme suivante :

$$\begin{cases} A_i.X_i - B_i^T.X_i = b_i\ , \\ -\sum_{i=1}^{i=2} B_i^T.X_i + \hat{A}.X = b, \quad \text{pour i=1,2;} \end{cases} \tag{3.1.5}$$

Après quelques calculs, on obtient le système (3.1.6), équivalent au système (3.1.4).

$$\begin{cases} \hat{A}X = \hat{B}, \\ \text{où} \quad \hat{A} = A' - \sum_{i=1}^{i=2} B_i A_i^{-1} B_i^T \quad \text{et} \quad \hat{B} = B - \sum_{i=1}^{i=2} B_i A_i^{-1} b_i; \end{cases} \tag{3.1.6}$$

**Théorème 3.1.1** *([39]) La matrice $\hat{A}$ définie en (3.1.6) est une matrice symétrique et définie positive.*

## 3.2 Méthode multigrille-gradient conjugué MG-CG

### 3.2.1 Principe de la méthode MG-CG

Cette méthode a été proposée par Braess [1, 1985] (MG-CG). L'idée consiste à utiliser la méthode du gradient conjugué dans les étapes de pré-lissage et post-lissage de la méthode multigrille.

On fait appel à l'algorithme de la méthode multigrille (section 2.2.2) et on utilise comme lisseur la méthode du gradient conjugué, ainsi on obtient l'algorithme du MG-CG.

Cet usage ne procure pas seulement le lissage, mais en même temps il améliore le calcul de la correction sur la grille grossière ce qui accélére la vitesse de convergence de l'algorithme du multigrille.

### 3.2.2 Algorithme du MG-CG

**Algorithme 3.1** ($u_h = MG - CG(A_h, u_h, 0, f_h, \nu_1, \nu_2, \gamma)$)

**Etape 1. Prélissage:** $u_h := \mathrm{CG}\nu_1(A_h, u_h, 0, f_h)$

**Etape 2. Calculer** $r_h = f_h - A_h u_{h.}$

**Etape 3.** $r_H = I_h^H r_h$

**Etape 4**. If $(H == h_0)$ Résoudre: $A_H \delta H = r_H$

sinon: $\delta H = MG - CG\gamma(A_H, 0, r_H, \nu_1, \nu_2, \gamma)$

**Etape 5. Correction:** $uh := uh + I_H^h \delta H$

**Etape 6. Postlissage:** $uh := CG\nu_2(Ah, uh, fh)$

## 3.3 Méthode multigrille-gradient MGgM

Avant de décrire la méthode MGcgM [49,2008] qui est une approche de la combinaison de la méthode multigrille et celle du gradient conjugué, on va d'abord décrire la méthode MGgM [49,2008] qui est une combinaison des méthodes multigrille et du gradient.

### 3.3.1 Principe de la méthode MGgM

Soit $x_0 \in \mathbb{R}^n$ un vecteur initial. Supposons que $x_k \in \mathbb{R}^n$ est une approximation de la solution. Soit $r_k = Ax_k - b$ le résidu à l'itération $k$ et $\Re^k$ l'espace des résidus définie par :

$$\Re^k := \quad \mathrm{Vect}\,\{\omega_l, ..., \omega_1\} \text{ où } \omega_l = r_l = Ax - b \text{ et } \omega_i = I_l^i\, r_l \text{ pour } i = 1, ..., l-1. \quad (3.3.1)$$

Soit maintenant l'approximation $x_{k+1} \in \mathbb{R}^n$ définie par :
Trouver $d \in \Re^k$tel que :

$$x_{k+1} = x_k + d \text{ où } h(x_{k+1}) = \min_{d \in \Re^k} h(x_k + d). \quad (3.3.2)$$

Afin de résoudre ce système, on construit une base A-orthonormale de $\Re^k$ en utilisant le procédé de Gram-Scmidt sur les vecteurs $\omega_1, ..., \omega_l$.
Autrement dit :

pour $i = 1, ..., l$ on a :

$$\hat{d}_i = \omega_i - \sum_{j=1}^{i-1} d_j^T A \omega_i d_j \,, \tag{3.3.3}$$

$$d_i = \frac{\hat{d}_i}{\left\| \hat{d}_i \right\|_A}. \tag{3.3.4}$$

d'où on obtient

$$x_{k+1} = x_k + \sum_{i=1}^{l} \alpha_i d_i \text{ où } \alpha_i = -d_i^T r_k. \tag{3.3.5}$$

En utilisant ces notations, on prouve le lemme suivant

**Lemme 3.3.1** *([49])Soit le vecteur $d_i^{gro} \in \mathbb{R}^{n_i}$ vérifiant :*

$$d_i = I_i^l d_i^{gro}, \quad \text{pour } i = 1, ..., l. \tag{3.3.6}$$

$$d_1^{gro} = \frac{r_1}{\|r_1\|_A}, \tag{3.3.7}$$

$$\hat{d}_i^{gro} = r_i - \sum_{j=1}^{i-1} I_j^i \left\{ \left[ \left( d_j^{gro} \right)^T I_j^i A_j r_i \right] d_j^{gro} \right\}, \tag{3.3.8}$$

$$d_i^{gro} = \frac{\hat{d}_i^{gro}}{\left\| \hat{d}_i^{gro} \right\|_A} \quad \text{pour} \quad i = 2, ..., l.. \tag{3.3.9}$$

**Lemme 3.3.2** *([49]) La correction de la méthode MGgM est calculée comme suit:*

$$\alpha_i = -(d_i^{gro})^T r_i \ et \tag{3.3.10}$$

$$\begin{aligned} x_{k+1} - x_k &= \sum_{i=1}^{l} \alpha_i d_i \\ &= I_{l-1}^l (...I_2^3 (I_1^2 \alpha_1 d_1^{gro} + \alpha_2 d_2^{gro})... + \alpha_{l-1} d_{l-1}^{gro}) + \alpha_l d_l^{gro} \end{aligned} \tag{3.3.11}$$

En utilisant les lemmes 3.3.1 et 3.3.2, on obtient l'algorithme MGgM suivant:

### 3.3.2 L'algorithme de MGgM

**Algorithme 3.2 (Algorithme de MGgM )**

**Etape 1.: Calcul du gradient**

$g_l := b - Ax^k$

$d_l := r_l$

Pour $i = l : 2$ faire

$r_{i-1} = I_i^{i-1} r_i$

$d_{i-1} := r_{i-1}$

fin pour.

**Etape 2. Calcul des directions orthogonales:**

Pour $i = 1 : l$ faire

$k_i := A_i d_i$

pour $j = i : 2$ faire

$k_{j-1} := I_j^{j-1} k_j$

fin pour.

$s_1 = 0$

pour $j = 1 : (i-1)$ faire

$s_j := s_j + d_j(k_j^T d_{j)}$

$s_{j+1} := I_j^{j+1} s_j$

fin pour.

$d_i := d_i - s_i$

**Etape 3. Normalisation des directions:**

$k_i := A_i d_i$

$d_i := d_i (k_i^T d_i)^{\frac{-1}{2}}$

fin pour.

**Etape 4. Calculer la correction:**

$s_1 := (d_1^T r_1) d_1$

pour $i = 2 : l$ faire

$s_i := I_{i-1}^i s_{i-1}$

$s_i := s_i + (d_i^T r_i) d_i$

fin pour

$x^{k+1} := x^k + s_l$

End.

### 3.3.3 Analyse de la convergence

**Théorème 3.3.1** *([49]) Soit $\hat{x}$ la solution exacte de l'équation $Ax = b$. Soit $x_k$ et $x_{k+1}$ des itérations successive générer par l'algorithme de MGgM. Alors, il existe une constante $0 < \rho < 1$ indépendante de $x_k$ et du nombre du niveau de la grille $l$ tel que:*

$$\|\hat{x} - x_{k+1}\|_A \leq \rho \|\hat{x} - x_k\|_A . \tag{3.3.12}$$

## 3.4 Méthode multigrille gradient conjugué MGcgM

Cette méthode a été proposée par Pflaum [49, 2008] (MGcgM). Elle consiste à construire de nouvelles corrections en se basant sur les directions du gradient conjugué.

### 3.4.1 Principe de la méthode MGcgM

Les directions de corrections construites à partir de l'espace des résidus de la méthode MGgM ne sont pas optimales, en effet, dans les calculs des directions on peut trouver des itérations successives dans la même direction. L'idée est, comme dans la méthode du gradient conjugué, dans la méthode MGcgM on calcule la direction à l'itérée k (qui appartient à $\Re^k$ ) de tel sorte qu'elle soit A-orthogonale (ou A-orthonormale) à la direction précédente (qui appartient à $\Re^{k-1}$ ). Ainsi on obtient :

$$\Re^k \perp_A \Re^{k-1} \Rightarrow \text{accélérer la convergence.}$$

Le théorème suivant (Pflaum [49, 2008]), garantie que les espaces de correction sont A-orthogonales, sachant que les espaces de correction de la méthode MGgM ne le sont pas.

**Théorème 3.4.1** *([49]) Soit $\hat{x} \in \mathbb{R}^n$ la solution exacte de l'équation $Ax = b$. Soit $W, V \subset \mathbb{R}^n$ deux sous-espaces, soit $\tilde{x}$ une solution approximative de $\hat{x}$ vérifiant:*

$$(\tilde{x})^T Av = b^T v, \quad \forall v \in V. \tag{3.4.1}$$

*On construit le sous-espace $W^\perp$ tel que:*

$$vect(W, V) = vect(W^\perp, V) \text{ et } W^\perp \perp_A V. \tag{3.4.2}$$

*Soit $w_r \in W$ et $w_d \in W^\perp$ tels que:*

$$(\tilde{x} + w_r)^T Aw = b^T w, \quad \forall w \in W \tag{3.4.3}$$

$$(\tilde{x} + w_d)^T Aw = b^T w, \quad \forall w \in W^\perp. \tag{3.4.4}$$

*Alors $(\tilde{x} + w_r)$ et $(\tilde{x} + w_d)$ sont des approximations de la solution exacte $\hat{x}$, et on a de plus:*

$$\|\hat{x} - (\tilde{x} + w_d)\|_A \leq \|\hat{x} - (\tilde{x} + w_r)\|_A . \tag{3.4.5}$$

### 3.4.2 L'algorithme de MGcgM

**Algorithme 3.3 (Algorithme de MGcgM)**

**Etape 1. Initialisation**

Si $k = 0$ alors on fait une itération par l'algorithme du MGgM (Algorithme 5)

Sinon

**Etape 2.**

**Calculer les résidus:**

$r_l := b - Ax^k$

$d_l^{new} := r_l$

Pour $i = l : 2$ faire

$r_{i-1} = I_i^{i-1} r_i$

$d_{i-1}^{new} := r_{i-1}$

Fin pour.

**Calculer les nouvelles directions:**

Pour $i = 1 : l$ faire

$k_i := A_i d_i$

Pour $j = i : 2$ faire

$k_{j-1} := I_j^{j-1} k_j$

fin pour.

$s_1 = 0$

pour $j = 1 : (i-1)$ faire

$s_j := s_j + d_j^{new}(k_j^T d_j^{new})$

$s_{j+1} := I_j^{j+1} s_j$

fin pour.

$d_i := d_i - s_i$

$k_i := A_i d_i^{new}$

pour $j = i : 2$ faire

$k_{j-1} := I_j^{j-1} k_j$

fin pour.

$s_1 = 0$

pour $j = 1 : (i-1)$ faire

$s_j := s_j + d_j^{new}(k_j^T d_{j)}$

$s_{j+1} := I_j^{j+1} s_j$

fin pour.

$d_i^{new} := d_i^{new} - s_i$

$k_i := A_i d_i^{new}$

$s_i := d_i^{new} - d_j(k_j^T d_j)$

Si $\|s_i\|_\infty > \varepsilon \|d_i^{new}\|_\infty$ alors $d_i^{new} := s_i$

sinon

$d_i^{new} = d_i$

$d_i := 0$

fin pour

$k_i := A_i d_i^{new}$

$d_i = d_i^{new}(k_i^T d_i^{new})^{-\frac{1}{2}}$

End for.

**Calcul de la correction:**

$s_1 := (d_1^T r_1) d_1$

pour $i = 2 : l$ faire

$s_i := I_{i-1}^i s_{i-1}$

$s_i := s_i + (d_i^T r_i) d_i$

fin pour

$x^{k+1} := x^k + s_l$

FIN.

### 3.4.3 Analyse de la convergence

**Théorème 3.4.2** *([49])Soit $\hat{x}$ la solution exacte de l'équation $Ax = b$. Soit $(x_k)_{k\in\mathbb{N}}$ la suite générée par l'algorithme de MGcgM. Alors, elle converge en n itérations au plus.*

## 3.5 Simulations Numériques

Considérons le problème du Laplacien avec les conditions aux limites de Dirichlet homogènes :

$$\begin{cases} -\triangle U = f \quad \text{sur } \Omega \\ \quad U = 0 \quad \text{sur } \partial\Omega \end{cases}$$

La discrétisation de ce problème par le schéma classique des différences finies à cinq points nous a permis d'aboutir au système (3.1.4). Le système (3.1.4) est équivalent au système (3.1.6).

Dans nos tests numériques on a résolu le système (3.1.6) par les méthodes MG-CG, MGgM, MGcgM, MG, CG, PCG et GMRES.

Prenons $\Omega = ]0,1[ \times ]0,1[ \subset \mathbb{R}^2$ tel que $\Omega_{1=} ]0,1[ \times \left]\frac{1}{2}, 1\right[$et $\Omega_{2=} ]0,1[ \times \left]0, \frac{1}{2}\right[$ et $f$ est une fonction donnée de telle sorte que $U(x,y) = x.y.(1-x)(1-y)$ soit la solution exacte du problème (3.1.6). On se donne un maillage régulier du pas $h = \frac{1}{n+1}$.

On a comparé les méthodes MGcgM et MGgM à deux et trois cycles. Pour la méthode MGcgM3 la première itération est calculée par la méthode MGgM3.

La méthode MG (v-cycle ou w-cycle) utilisée possède comme opérateur de lissage la méthode de Jacobi et on a pris $\nu_1 = \nu_2 = 2$.

La méthode CG est utilisée avec redémarrage, c'est-à-dire dans le cas où le dénominateur de la variante du gradient conjugué $\beta_k$ est nulle on redémarre en prenant la direction de la plus profonde pente. Powell ([55, 1977]) a éclairé les bien effets du redémarrage dans l'algorithme.

La méthode MG-CG est utilisée à deux grilles dont l'opérateur de lissage est la méthode CG avec redémarrage et on a pris $\nu_1 = \nu_2 = 2$.

On a pris comme préconditionement de la méthode PCG ([58]) la matrice $M$ dont ces éléments de la diagonale sont égaux à ceux de la diagonale de la matrice A et nuls autrement.

Les programmes sont écrits sous scilab 5.4.0 et exécutés dans un ordinateur Intel PC i5 de vitesse 3.10 GHz et une RAM de 4 Go et un système Windows sept (64 bits).

On arrête l'exécution d'un programme si le nombre des itérations dépasse 5000 et on considère que l'algorithme a échoué.

Afin de minimiser le temps de calcul, on a diminué le nombre d'opérations effectuées dans notre problème, en utilisant des petits programmes en scilab dans le but d'exploiter la structure de la matrice de rigidité de notre problème.

Dans les tableaux suivants, nos résultats numériques sont écrits sous la forme NI/CPU où NI et CPU dénotent le nombre des itérations et le temps en secondes, respectivement. Dim dénote la dimension du problème. L'étoile * dénote que c'est le meilleur résultat obtenue dans l'exécution des cinq méthodes utilisée. Notre critère d'arrêt est $\|x_k - \hat{x}\| \leq \varepsilon$, ou $\hat{x}$ est la solution exacte du problème et $\varepsilon = 10^{-6}$.

Les résultats sont présentés dans le tableau suivant:

| Dim | MG-CG | MGcgM3 | CG | PCG | GMRES |
|---|---|---|---|---|---|
| 5 | 2/0.02* | Echec | 6/0.02 | 5/0.02 | 3/0.02 |
| 9 | 2/0.02* | Echec | 6/0.02 | 5/0.02 | 4/0.02 |
| 17 | 2/0.04 | Echec | 15/0.04 | 9/0.02* | 9/0.03 |
| 33 | 2/0.04* | Echec | 17/0.06 | 15/0.04 | 14/0.05 |
| 65 | 2/0.1 | Echec | 18/0.04* | 14/0.06 | 14/0.05 |

Tableau 1 : Comparaison des méthodes MGcgM, MG-CG, MG, CG, PCG et GMRES

D'après le tableau 3.5.1, on voit bien qu'en comparant les méthodes MG-CG, MGcgM3, CG, PCG et GMRES pour notre problème, la méthode MG-CG donne les meilleurs résultats. Notons que les méthodes de MG (V-cycle et W-cycle), MGcgM2, MGgM2 et MGgM3 divergent pour notre problème et ceci dans tout les cas étudiés.

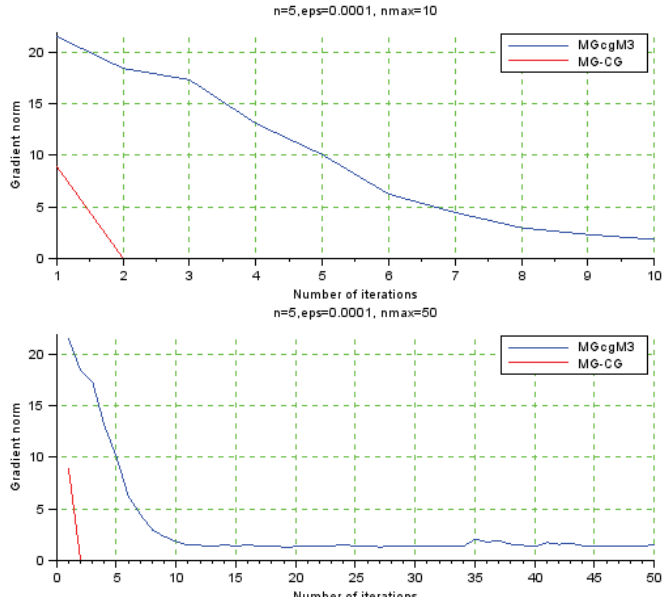


Fig 3.5.1 Comparaison des méthodes MG-CG et MGcgM3

D'après la figure 3.5.1 on observe bien que la méthode MG-CG est plus performante que MGcgM3 pour ce problème.

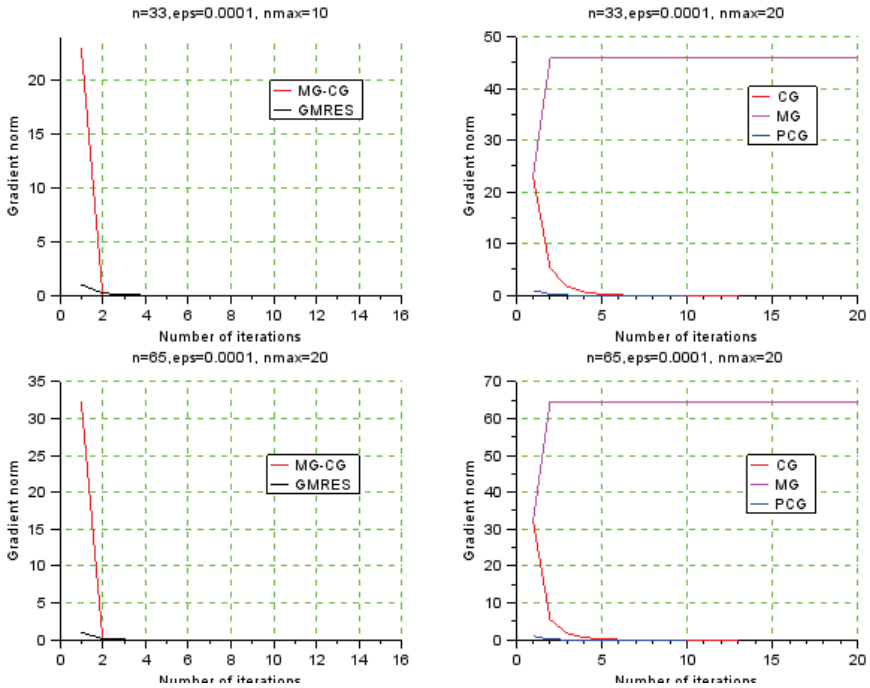


Fig 3.5.2 Comparaison des méthodes MG-CG, MG, GMRES, CG et PCG

D'après la figure 3.5.2 on remarque que la méthode MG-CG est plus performante que les méthodes PCG, GMRES, CG et MG, et ceci pour notre problème.

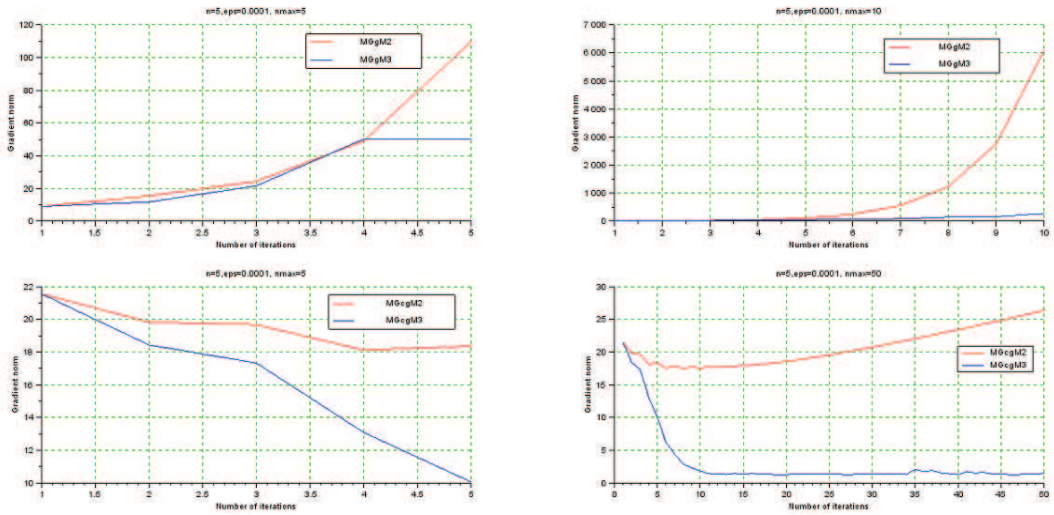


FIG 3.5.3 Comparaison des méthodes MGcgM2, MGcgM3, MGgM2 et MGgM3

D'après la figure 3.5.3 on observe bien que la méthode MGgM3 donne des résultats mieux que MGgM2, cette dernière explose après 15 itérations pour ce cas. D'autre part on voit bien que malgré que la méthode MGcgM3 diverge, elle réduit la norme du résidu et que la méthode MGcgM2 diverge pour ce problème.

Nos simulations numérique montrent que l'usage de la méthode du gradient conjugué avec redémarrage comme un lisseur de la méthode multigrille ne procure pas seulement le lissage, mais en même temps il améliore le calcul de la correction sur la grille grossière ce qui accélére la vitesse de convergence de l'algorithme du multigrille.

## 3.6 Conclusion

L'étude comparative de deux approches différentes de combinaison des méthodes multigrille et gradient conjugué pour la résolution du problème (3.1.6), où A est une matrice par bloc symétrique définie positive nous a permis de démontrer les avantages des schémas hybrides de combinaison des algorithmes du multigrille et gradient conjugué. Nos résultats numériques prouvent que ces algorithmes hybrides sont très efficaces pour la résolution des problèmes linéaires à petite et moyenne taille.

Dans nos simulations numériques six différentes méthodes ont été comparées : méthode multigrille-gradient conjugué (MG-CG), méthode multigrille-gradient (MGgM), méthode multigrille gradient conjugué (MGcgM), méthode multigrille (MG), méthode du gradient conjugué (CG) et méthode du gradient conjugué préconditionné (PCG). On a utilisé la méthode de Jacobi comme un lisseur dans la méthode multigrille (MG). Les méthodes MGgM et MGcgM sont utilisé au niveau l=2, l=3.

Les programmes sont écrits sous scilab 5.4.0 et exécuté dans un ordinateur Intel PC i5 de vitesse 3.10 GHz et une RAM de 4 Go et un système Windows sept (64 bits).

En comparant les méthodes MG-CG, MGgM, MGcgM, MG, CG et PCG on remarque que la méthode MG-CG est plus efficace pour notre problème.

D'après les résultats obtenus on note :

- La méthode MG-CG converge en un nombre fini d'itérations.

- Pour notre problème la méthode MG-CG converge plus rapidement que les autres méthodes étudiées.

- Les méthodes MGcgM et MGgM à deux et à trois cycles divergent dans tous les cas étudiés. Notons que même si MGcgM diverge elle réduit la norme du résidu.

CHAPITRE 4

# Etude comparative de quelques méthodes du gradient conjugué non linéaire avec recherches linéaires inexactes

Le but de ce chapitre est une comparaison de différentes variantes du gradient conjugué non linéaire(méthodes de Hestenes–Stiefel[34,1952], Fletcher–Reeves[24,1964], Polak–Ribière–Polyak[50,51, 1969], Descente conjuguée[22,1987], Liu–Storey([42, 1991]), Dai–Yuan[18,1999] et Hager–Zhang[32,2005]) avec les recherches linéaire inexactes (Armijo [4,1966] et Wolfe forte[67,1971]).

L'idée est d'utiliser des recherches linéaires inexactes pour le calcul du pas choisi dans la méthode du gradient conjugué non linéaire au lieu d'appliquer la formule du pas qui nécessite que les directions soient orthogonales entre elles et qui ne se rencontrent pas dans la pratique. Ce nouveau choix a accéléré la convergence de la méthode du gradient conjugué.

Nos simulations numériques (Scilab) montrent que la recherche linéaire de Wolfe forte a accéléré la convergence des méthodes citées ci-dessus mieux que la règle d'Armijo, et que

les méthodes de Hager Zhang et de Dai-Yuan sont plus efficaces en général que les autres méthodes étudiées.

## 4.1 Simulations numériques

Dans cette section, on va étudier les performances des méthodes HS, FR, PRP, PRP +, CD, LS, DY et de HZ avec la recherche linéaire de Wolfe forte, ainsi que la règle d'Armijo dans le but de les comparer pour des fonctions testes connues.
L'algorithme de la recherche linéaire assure que la direction est de descente. On va présenter deux choix différents du pas qui assurent la décroissance de la fonction objective. Les algorithmes mis en œuvre pour la résolution d'un problème d'optimisation sans contrainte sont :

**Algorithme 4.1(méthodes du GC non linéaire de HS, FR, PRP, CD, LS, DY et HZ**

**Etape 0: (initialisation)**

Soit $x_0$ le point de départ, $g_0 = \nabla f(x_0)$, poser $d_0 = -g_0$

Poser $k = 0$ et aller à l'étape 1.

**Etape 1 :**

Si $g_k = 0$ : STOP ( $x^* = x_k$)."Test d'arrêt"

Si non aller à l'étape 2.

**Etape 2 :**

Définir $x_{k+1} = x_k + \alpha_k d_k$ avec :

$\alpha_k$ est définie par une recherche linéaire inexacte (Algorithme)

$d_{k+1} = -g_{k+1} + \beta_{k+1} d_k$

où $\beta_{k+1}$ est définie par l'une des formule

Poser $k = k + 1$ et aller à l'étape 1. $\square$

**Algorithme 4.2 (Règles d'Armijo et de Wolfe forte avec rebroussement)**

**Etape 0 : (initialisation)**

A l'itération k, soit $\alpha_0$ le pas initial, $x_k$, $f_k$, $g_k$, $d_k$ et $0 < \beta < 1$.

Poser $\alpha_k = \alpha_0$ et $x_{k+1} = x_k + \alpha_k d_k$ aller à l'étape 1.

**Etape 1 :**

si $\alpha_k$ vérifie les critères de la règle d'Armijo (ou la règle de Wolfe forte) : STOP ($\alpha^* = \alpha_k$).

si non aller à l'étape 2.

**Etape 2 :**

$\alpha_{k+1} = \beta\alpha_k$

calculer $x_{k+1} = x_k + \alpha_k d_k$

poser $k = k + 1$ et aller à l'étape 1. □

Dans nos simulations numériques on a limité le nombre des itérations nécessaires dans l'algorithme de la recherche linéaire, et le pas initial est déterminé par la règle de Shanno et Phua ([59, 1980]).

Les paramètres sont donnés : $\rho = 0.1$, $\delta = 0.01$ et $\beta = 0.95$.

Dans le cas où le dénominateur de la variante du gradient conjugué $\beta_k$ est nulle on redémarre en prenant la direction de la plus profonde pente. Powell ([55, 1977]) a éclairé les bien effets du redémarrage dans l'algorithme.

Les programmes sont écrits sous scilab 5.4.0 et exécutés dans un ordinateur Intel PC i5 de vitesse 3.10 GHz et une RAM de 4Go et un système Windows sept.

On a arrêté l'exécution du programme du gradient conjugué non linéaire si le nombre des itérations dépasse 5000 ou si le programme de la recherche linéaire n'arrive pas à calculer un pas positif, dans ce cas on considère que l'algorithme a échoué.

Dans les tableaux suivants, nos résultats numériques sont écrits sous la forme NI/CPU où NI et CPU dénotent le nombre des itérations et le temps en secondes, respectivement. Dim dénote la dimension du problème. L'étoile * dénote que c'est le meilleur résultat obtenue dans l'exécution des huit variantes du gradient conjugué.

Les problèmes tests ainsi que les vecteurs de départ sont pris de deux articles bien connues (Moré and Hillstrom, 1981) et (Andrei, 2008). Pour chaque problème le critère d'arrêt est $\|g_k\| \leq \varepsilon$, ou $\varepsilon$ est déterminé à chaque test.
Maintenant, on va décrire chaque fonction teste utilisée puis au-dessous de chacune on présente nos résultats corresponds à cette fonction.

### 4.1.1 La fonction BALF (Brown almost-linear )

Cette fonction est numérotée 27 dans l'ensemble des fonctions testes proposées par (Moré and Hillstrom [47, 1981])
$f_i(x) = x_i + \sum_{i=1}^{n} x_i - (n+1)$, $1 \leq i < n$ et $f_n(x) = \left(\prod_{i=1}^{n} x_i\right) - 1$ où $x_0 = (\frac{1}{2}, ..., \frac{1}{2})$.

| Dim | HS | FR | PRP | PRP+ | CD | LS | DY | HZ |
|---|---|---|---|---|---|---|---|---|
| 2 | Echec | 60/0.125 | 68/0.141 | 68/0.094* | 141/0.234 | Echec | 72/0.124 | 58/0.109 |
| 3 | Echec | Echec | 61/0.125 | 61/0.094 | 24/0.093 | Echec | 59/0.141 | 19/0.078* |
| 4 | Echec | 57/0.141 | 26/0.078* | Echec | Echec | 11/0.078* | 68/0.156 | 37/0.094 |
| 5 | 15/0.094 | 83/0.219 | 95/0.171 | 466/0.46 | Echec | 13/0.078* | 82/0.203 | 51/0.124 |
| 9 | Echec | 80/0.187 | Echec | 126/0.187 | Echec | 18/0.093* | Echec | 63/0.156 |
| 10 | Echec | 102/0.25 | Echec | 96/0.172* | Echec | 129/0.312 | Echec | 108/0.218 |
| 11 | Echec | 78/0.218 | Echec | 107/0.187 | Echec | 22/0.109* | Echec | Echec |
| 12 | Echec | Echec | Echec | 135/0.218* | Echec | Echec | 151/0.328 | 202/0.39 |
| 15 | Echec | 108/0.375 | Echec | 69/0.171 | Echec | 30/0.14* | Echec | 205/0.39 |
| 20 | Echec | Echec | Echec | Echec | Echec | Echec | Echec | 512/1.248* |
| 25 | Echec | 83/0.312 | Echec | Echec | Echec | Echec | 51/0.282* | 811/2.002 |
| 30 | Echec | Echec | Echec | 123/0.53 | Echec | Echec | 136/0.452* | Echec |

Tableau 2: Différentes CG avec la règle d'Armijo pour BALF, $\varepsilon = 10^{-1}$

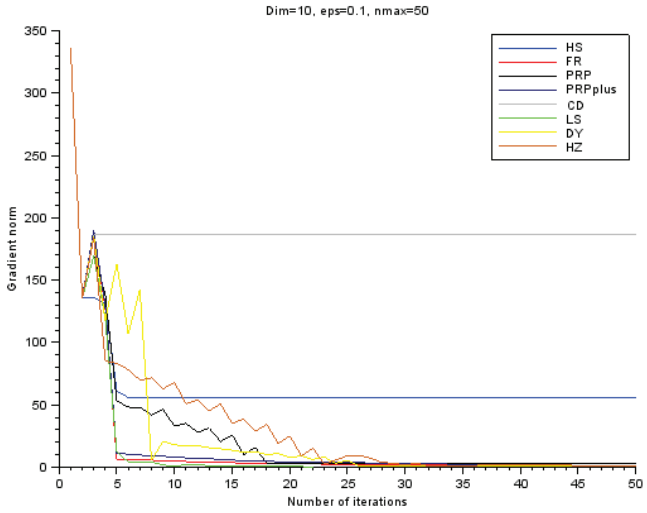


Figure 4.1.1 : Comparaison des methodes CG avec la regle d'Armijo pour BALF

| Dim | HS | FR | PRP | PRP+ | CD | LS | DY | HZ |
|---|---|---|---|---|---|---|---|---|
| 2 | Echec | 55/0.078* | 54/0.094 | 54/0.094 | Echec | Echec | 52/0.078* | 40/0.078* |
| 3 | Echec | Echec | 47/0.093 | 47/0.078 | 34/0.078 | Echec | 54/0.093 | 28/0.063* |
| 4 | Echec | 48/0.094 | 32/0.078 | Echec | Echec | 12/0.063* | 57/0.109 | 31/0.078 |
| 5 | Echec | Echec | 68/0.109 | 179/0.219 | Echec | 9/0.063* | 70/0.125 | 38/0.093 |
| 6 | Echec | 69/0.125 | 19/0.062* | 65/0.125 | Echec | Echec | 37/0.093 | 78/0.125 |
| 7 | Echec | 60/0.109 | 59/0.125 | 85/0.141 | Echec | 18/0.062* | Echec | 80/0.125 |
| 8 | Echec | Echec | Echec | 87/0.188 | Echec | 20/0.062* | Echec | 82/0.203 |
| 9 | Echec | 43/0.125* | Echec | 92/0.25 | Echec | Echec | Echec | 66/0.172 |
| 10 | Echec | 86/0.265 | Echec | 82/0.234* | Echec | Echec | 97/0.265 | 68/0.234* |
| 15 | Echec | 83/0.25 | Echec | 58/0.188* | Echec | Echec | Echec | Echec |
| 30 | Echec | 94/0.561* | Echec | 154/0.78 | Echec | Echec | 142/0.765 | Echec |

Tableau 3: Différentes CG avec la règle de Wolfe forte pour BALF, $\varepsilon = 10^{-1}$

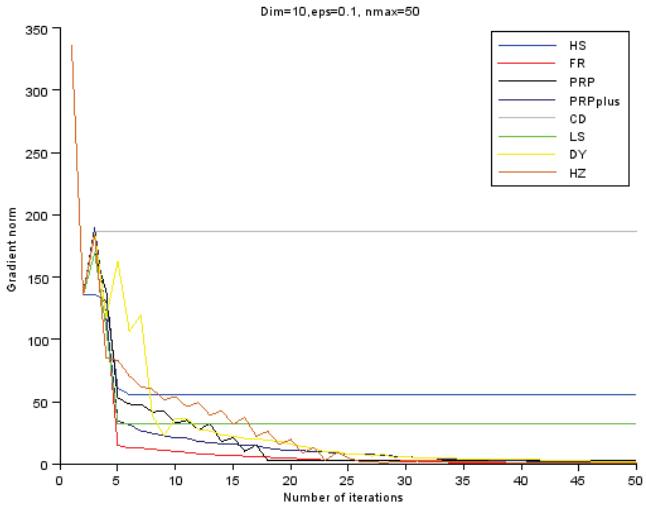


Figure 4.1.2 : Comparaison des CG avec la regle de Wolfe forte pour BALF

D'après les tableaux 2 et 3, et les figures 4.1.1 et 4.1.2, on observe que la règle de Wolfe forte accélère la convergence de toutes les méthodes CG mieux que la règle d'Armijo pour la fonction BALF.

Notons que pour 2<Dim<30, les meilleurs résultats sont donnés par les méthodes HZ, PRP+ et DY, et que les plus mauvais résultats sont donnés par les méthodes HS et CD. On remarque aussi que les méthodes FR, PRP et LS donnent de bons résultats.

Notons que pour Dim>30 toutes les méthodes étudiées divergent.

### 4.1.2 La fonction BRYBND (Broyden banded )

Cette fonction est numérotée 31 dans l'ensemble des fonctions testes proposées par (Moré and Hillstrom [47, 1981])

$$\begin{aligned} f_i(x) &= x_i(2+5x_i^2)+1-\sum_{j\in J_i} x_i(1+x_i),\ 1\le i<n, \\ où\ J_i &= \{j: j\neq i,\ max(1,i-m_l)\le j\le \min(n,m_u)\}, \\ m_l &= 5,\ m_u=1,\ \text{et } x_0=(-1,...,-1). \end{aligned}$$

| Dim | HS | FR | PRP | PRP+ | CD | LS | DY | HZ |
|---|---|---|---|---|---|---|---|---|
| 50 | Echec | 424/2.384 | 23/0.651* | 36/0.666 | Echec | 26/0.761 | Echec | 132/1.123 |
| 100 | Echec | Echec | Echec | Echec | Echec | Echec | Echec | 144/2.511* |
| 150 | Echec | 105/3.448 | Echec | 33/1.934* | Echec | 95/3.152 | Echec | 140/3.822 |
| 200 | Echec | Echec | Echec | 33/2.511* | Echec | 29/3.853 | Echec | 114/4.759 |
| 300 | Echec | Echec | Echec | 32/4.259* | Echec | 56/5.96 | Echec | 149/8.553 |
| 400 | Echec | Echec | Echec | 38/6.068* | Echec | 142/11.488 | Echec | 141/11.534 |
| 500 | Echec | Echec | Echec | 42/8.424* | Echec | 260/21.76 | Echec | Echec |
| 600 | Echec | Echec | Echec | 655/45.569 | Echec | Echec | Echec | 161/22.638* |
| 700 | Echec | Echec | Echec | 27/15.21* | Echec | 45/16.637 | Echec | 155/23.238 |
| 800 | Echec | Echec | Echec | 27/17.02* | Echec | Echec | Echec | 131/27.077 |
| 900 | Echec | Echec | Echec | 62/26.489* | Echec | 63/26.758 | Echec | 171/37.698 |
| 1000 | Echec | Echec | Echec | 55/29.698* | Echec | 120/46.243 | Echec | 174/52.142 |
| 2000 | Echec | Echec | Echec | 1885/874.765 | Echec | 107/144.723* | Echec | 162/167.88 |

Tableau 4: Différentes CG avec la règle d'Armijo pour BRYBND, $\varepsilon = 10^{-1}$

| Dim | HS | FR | PRP | PRP+ | CD | LS | DY | HZ |
|---|---|---|---|---|---|---|---|---|
| 50 | Echec | 58/1.045 | 23/0.749* | 58/1.076 | Echec | 44/0.921 | Echec | 102/1.247 |
| 100 | Echec | Echec | Echec | Echec | Echec | Echec | Echec | 102/2.135* |
| 150 | Echec | Echec | Echec | 76/3.042 | Echec | 47/2.465* | Echec | 118/3.573 |
| 200 | Echec | Echec | 31/2.87* | 27/3.104 | 433/12.574 | 107/5.257 | Echec | 82/3.955 |
| 300 | Echec | Echec | Echec | 29/4.633* | Echec | 121/7.691 | Echec | 129/7.754 |
| 400 | Echec | Echec | Echec | 159/11.263 | Echec | 142/11.232 | Echec | 114/9.953* |
| 500 | Echec | Echec | Echec | 34/7.971* | Echec | 36/8.143 | Echec | Echec |
| 600 | Echec | Echec | Echec | 47/12.589* | Echec | 47/13.915 | Echec | 125/19.378 |
| 700 | Echec | Echec | Echec | Echec | Echec | 60/19.11* | Echec | 116/24.773 |
| 800 | Echec | Echec | 125/29.172 | 26/18.066* | Echec | Echec | Echec | 98/24.726 |
| 900 | Echec | Echec | Echec | 62/26.894* | Echec | 114/33.774 | Echec | 130/35.318 |
| 1000 | Echec | Echec | Echec | Echec | Echec | Echec | Echec | 136/42.291* |
| 2000 | Echec | Echec | Echec | 99/91.587* | Echec | 237/147.558 | Echec | 106/494 |

Tableau 5: Différentes CG avec la règle de Wolfe forte pour BRYBND, $\varepsilon = 10^{-1}$

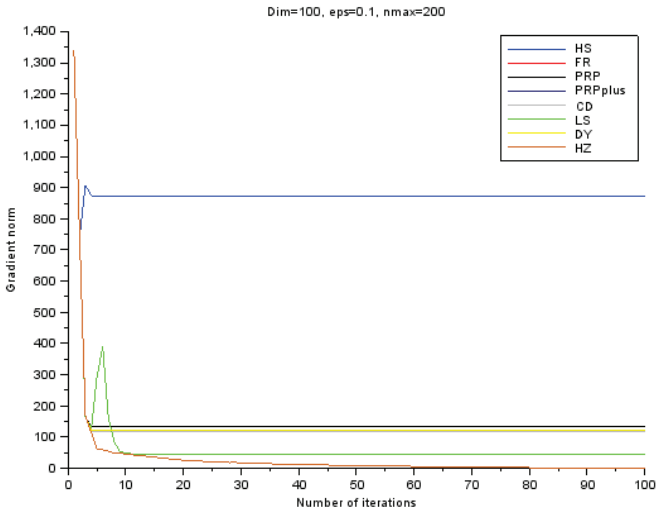


Figure 4.1.3 : Comparaison des CG avec la regle d'Armijo pour BRYBND

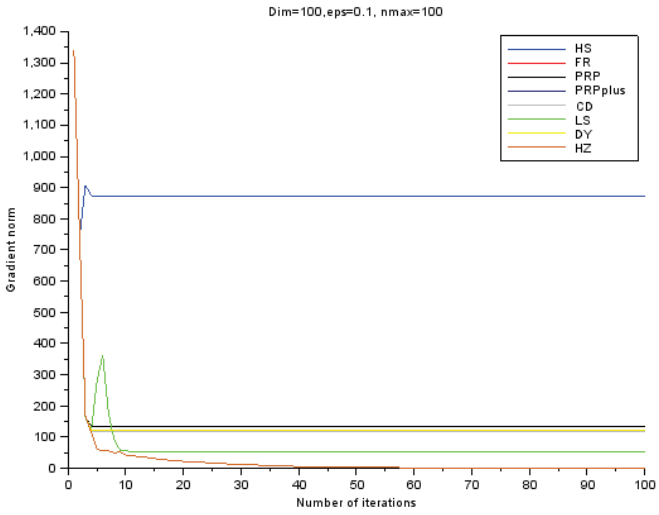


Figure 4.1.4 : Comparaison des CG avec la regle de Wolfe forte pour BRYBND

D'après les tableaux 4 et 5, et les figures 4.1.3 et 4.1.4, on observe que pour la fonction BRYBND où 50<Dim<2000, les meilleurs résultats sont donnés par les méthodes PRP+ et HZ, et que les plus mauvais résultats sont donnés par les méthodes PRP, FR et DY, et que la méthode HS diverge.
Notons que la règle de Wolfe forte accélère la convergence mieux que celle d'Armijo.

### 4.1.3 La fonction BTF (Broyden tridiagonal )

Cette fonction est numérotée 30 dans l'ensemble des fonctions testes proposées par (Moré and Hillstrom [47, 1981])

$$\begin{aligned} f_i(x) &= (3-2x_i)x_i - x_i - 1 - 2x_{i+1} + 1,\ 1 \leq i < n, \\ où\ x_0 &= x_{n+1} = 0 \text{ et } x_0 = (-1, ..., -1). \end{aligned}$$

| Dim | HS | FR | PRP | PRP+ | CD | LS | DY | HZ |
|---|---|---|---|---|---|---|---|---|
| 2 | Echec | 22/0.093 | 63/0.125 | 32/0.078 | Echec | Echec | 14/0.062* | 44/0.093 |

Tableau 6: Différentes CG avec la règle d'Armijo pour BTF, $\varepsilon = 10^{-1}$

| Dim | HS | FR | PRP | PRP+ | CD | LS | DY | HZ |
|---|---|---|---|---|---|---|---|---|
| 2 | Echec | 18/0.078 | 25/0.094 | 25/0.078 | 50/0.14 | Echec | 24/0.063* | 56/0.14 |

Tableau 7: Différentes CG avec la règle de Wolfe forte pour BTF, $\varepsilon = 10^{-1}$

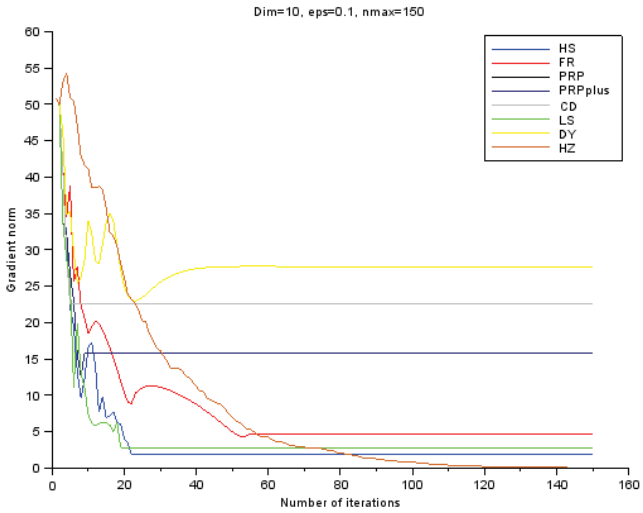


Figure 4.1.5 : Comparaison des CG avec la regle d'Armijo pour BTF

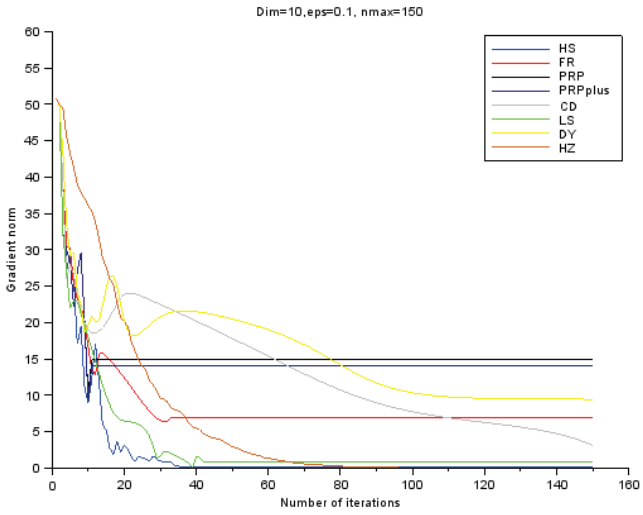


Figure 4.1.6 : Comparaison CG avec la regle de Wolfe forte pour BTF

En comparant les résultats des tableaux 6 et 7, et les figures 4.1.5 et 4.1.6, on observe que pour la fonction DTF où Dim=2, les meilleurs résultats sont donnés par les méthodes PRP+, HZ et DY, et que les plus mauvais résultats sont donnés par les méthodes HS et CD, et que la méthode HS diverge.
Notons que pour Dim>2 toutes les méthodes étudiées divergent.

### 4.1.4 La fonction DBVF (Discrete boundary value )

Cette fonction est numérotée 28 dans l'ensemble des fonctions testes proposées par (Moré and Hillstrom [47, 1981])

$$\begin{aligned} f_i(x) &= 2x_i - x_{i-1} - x_{i+1} + h^2(x_i + t_i + 1)^3/2,\ 1 \leq i < n, \\ où\ h &= 1/(n+1),\ t_i = ih,\ x_0 = x_{n+1} = 0 \\ \text{et } x_0 &= (\zeta_i) \text{ où } \zeta_i = t_i(t_i - 1). \end{aligned}$$

| Dim | HS | FR | PRP | PRP+ | CD | LS | DY | HZ |
|---|---|---|---|---|---|---|---|---|
| 4 | 31/0.312 | Echec | 131/0.749 | 34/0.109 | Echec | 17/0.094* | 24/0.094* | 76/0.156 |
| 5 | 8/0.062 | 994/1.451 | 4/0.062 | 4/0.047 | Echec | 5/0.063 | 3/0.046* | 3/0.047 |
| 6 | 16/0.078 | 1/0.031* | 1/0.031* | 1/0.047 | 1/0.032 | 1/0.047 | 1/0.031* | 1/0.031* |
| 7 | 1/0.032 | 1/0.031* | 1/0.047 | 1/0.047 | 1/0.032 | 1/0.031* | 1/0.047 | 1/0.046 |
| 8 | 1/0.078 | 1/0.047 | 1/0.047 | 1/0.031* | 1/0.047 | 1/0.047 | 1/0.031* | 1/0.031* |
| 9 | 1/0.047 | 1/0.063 | 1/0.032* | 1/0.047 | 1/0.047 | 1/0.047 | 1/0.063 | 1/0.046 |
| 10 | 1/0.063 | 1/0.063 | 1/0.078 | 1/0.063 | 1/0.063 | 1/0.046* | 1/0.062 | 1/0.047 |
| 15 | 1/0.078 | 1/0.063 | 1/0.078 | 1/0.062* | 1/0.078 | 1/0.078 | 1/0.093 | 1/0.065 |
| 20 | 1/0.124 | 1/0.125 | 1/0.124 | 1/0.078* | 1/0.11 | 1/0.093 | 1/0.093 | 1/0.078* |
| 25 | 1/0.125 | 1/0.109 | 1/0.125 | 1/0.093* | 1/0.125 | 1/0.188 | 1/0.126 | 1/0.125 |
| 30 | 1/0.172 | 1/0.109* | 1/0.172 | 1/0.109* | 1/0.125 | 1/0.141 | 1/0.125 | 1/0.141 |
| 40 | 1/0.156* | 1/0.172 | 1/0.172 | 1/0.172 | 1/0.156* | 1/0.203 | 1/0.156* | 1/0.187 |
| 50 | 1/0.203 | 1/0.219 | 1/0.249 | 1/0.234 | 1/0.218 | 1/0.203 | 1/0.249 | 1/0.187* |
| 70 | 1/0.265 | 1/0.327 | 1/0.25* | 1/0.312 | 1/0.375 | 1/0.296 | 1/0.265 | 1/0.265 |
| 100 | 1/0.483 | 1/0.396 | 1/0.516 | 1/0.374* | 1/0.515 | 1/0.421 | 1/0.499 | 1/0.453 |

Tableau 8: Différentes CG avec la règle de Wolfe forte pour DBVF, $\varepsilon = 10^{-1}$

| Dim | HS | FR | PRP | PRP+ | CD | LS | DY | HZ |
|---|---|---|---|---|---|---|---|---|
| 4 | 33/0.592* | Echec | 107/1.279 | 107/1.201 | Echec | 36/0.718 | 80/1.108 | 100/1.045 |
| 5 | 37/0.609* | Echec | 657/5.663 | 40/0.812 | Echec | 232/2.355 | 83/1.014 | 108/1.155 |
| 6 | 39/0.718 | Echec | 34/0.671 | 43/0.734 | Echec | 30/0.608* | 80/1.014 | 102/1.185 |
| 7 | 26/0.592 | Echec | 23/0.562 | 25/0.546* | Echec | 128/1.482 | 82/0.999 | 106/1.202 |
| 8 | 29/0.577* | Echec | 30/0.624 | 553/4.602 | Echec | 37/0.687 | 78/0.983 | 91/1.061 |
| 9 | 31/0.702* | Echec | 44/0.89 | 51/0.765 | Echec | 999/7.863 | 87/1.232 | 107/1.155 |
| 10 | 34/0.671 | Echec | Echec | 31/0.562 | Echec | 22/0.452* | 80/0.968 | 106/1.154 |
| 15 | 26/0.593 | Echec | 28/0.53* | 23/0.577 | Echec | 97/0.967 | 80/0.983 | 99/1.108 |
| 20 | 29/0.687 | Echec | 23/0.515* | 25/0.546 | Echec | 86/1.077 | 87/1.077 | 101/1.263 |
| 25 | 32/0.608 | Echec | 22/0.484* | 33/0.624 | Echec | 64/0.827 | 81/0.983 | 98/1.03 |
| 30 | 27/0.499* | Echec | 369/3.244 | 25/0.593 | Echec | 27/0.593 | 85/1.061 | 90/1.154 |
| 40 | Echec | Echec | 40/0.795* | 168/1.7 | Echec | 97/1.045 | 83/0.936 | 95/1.232 |
| 50 | 28/0.608 | Echec | 25/0.562* | 28/0.64 | Echec | 29/0.608 | 82/1.06 | 88/0.983 |
| 70 | 38/0.686 | Echec | 29/0.639* | 44/0.702 | Echec | 31/0.655 | 76/0.983 | 103/1.263 |
| 100 | 1/0.406 | 1/0.374 | 1/0.374 | 1/0.39 | 1/0.374 | 1/0.374 | 1/0.374 | 1/0.359* |

Tableau 9: Différentes CG avec la règle de Wolfe forte pour DBVF, $\varepsilon = 10^{-1}$

| Dim | HS | FR | PRP | PRP+ | CD | LS | DY | HZ |
|---|---|---|---|---|---|---|---|---|
| 100 | 25/0.749 | Echec | 19/0.688* | 27/0.749 | Echec | 28/0.765 | 1099/7.21 | 22/0.702 |
| 200 | 9/1.216* | 219/3.776 | 6/1.233 | 6/1.279 | Echec | 6/1.28 | 81/1.996 | 3/1.217 |
| 400 | 1/2.901 | 1/2.886 | 1/2.871 | 1/2.886 | 1/2.886 | 1/2.87* | 1/2.886 | 1/2.87* |
| 600 | 1/5.242 | 1/5.413 | 1/5.273 | 1/5.304 | 1/5.257 | 1/5.257 | 1/5.227* | 1/6.21 |
| 800 | 1/9.705 | 1/9.736 | 1/9.719 | 1/9.672 | 1/9.72* | 1/9.688 | 1/9.704 | 1/9.672 |
| 1000 | 1/15.367 | 1/15.304 | 1/15.304 | 1/15.227* | 1/15.303 | 1/15.273 | 1/15.339 | 1/15.241 |
| 1500 | 1/34.036 | 1/33.821 | 1/34.118 | 1/33.792* | 1/34.103 | 1/33.946 | 1/39.601 | 1/33.994 |
| 2000 | 1/61.637 | 1/61.342* | 1/61.872 | 1/61.574 | 1/61.606 | 1/61.731 | 1/61.747 | 1/61.419 |

Tableau 10: Différentes CG avec la règle de Wolfe forte pour DBVF, $\varepsilon = 10^{-4}$

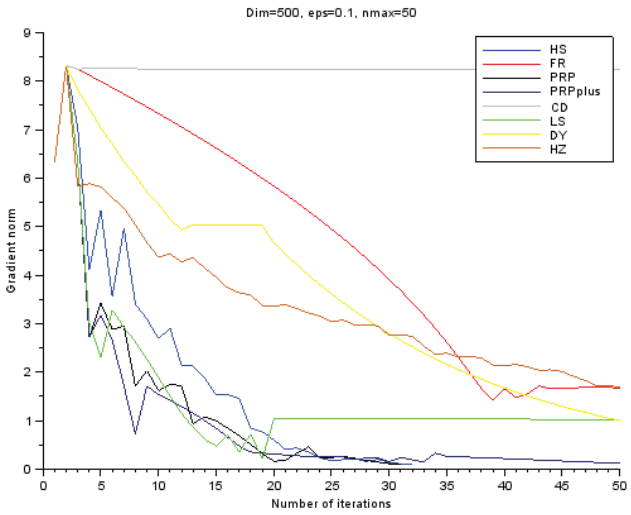


Figure 4.1.7 : Comparaison des CG avec la regle d'Armijo pour DBVF

| Dim | HS | FR | PRP | PRP+ | CD | LS | DY | HZ |
|---|---|---|---|---|---|---|---|---|
| 100 | 24/0.749* | 50/1.045 | 35/0.904 | 32/0.921 | 50/2.886 | 37/0.951 | 50/0.905 | 24/0.826 |
| 200 | 8/1.326 | 50/2.433 | 6/1.341 | 6/1.342 | 50/5.554 | 6/1.326 | 50/1.825 | 3/1.186* |
| 400 | 1/2.621 | 1/2.574* | 1/2.668 | 1/2.699 | 1/2.668 | 1/2.652 | 1/2.606 | 1/2.606 |
| 600 | 1/4.399 | 1/5.273 | 1/4.244 | 1/4.212 | 1/4.118* | 1/4.228 | 1/4.275 | 1/4.259 |
| 800 | 1/7.067 | Echec | 1/6.958* | 1/7.051 | 1/7.145 | 1/7.114 | 1/7.16 | 1/7.192 |
| 1000 | 1/12.168 | 1/11.996 | 1/11.606* | 1/11.871 | 1/11.84 | 1/11.888 | 1/11.903 | 1/11.919 |
| 1500 | Echec | Echec | Echec | Echec | Echec | Echec | 1/34.088 | Echec |
| 2000 | Echec | 1/39.406* | 1/40.294 | 1/40.155 | 1/39.858 | 1/39.936 | 1/39.843 | 1/39.986 |

Tableau 11: Différentes CG avec la règle de Wolfe forte pour DBVF, $\varepsilon = 10^{-4}$

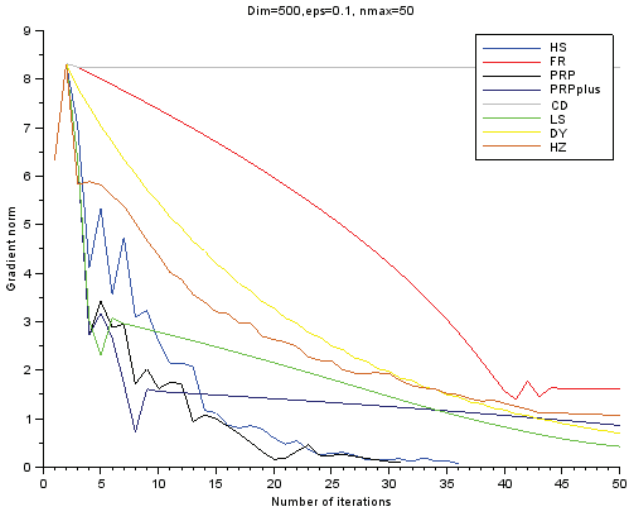


Figure 4.1.8 : Comparaison des CG avec la regle de Wolfe forte pour DBVF

En comparant les résultats des tableaux 8,9,10 et 11, et les figures 4.1.7 et 4.1.8, on observe que pour la fonction DBVF les méthodes CG donnent de bon résultats pour Dim>100. Dans le cas où Dim<100, les meilleurs résultats sont donnés par les méthodes DY, PRP+, HZ, LS et FR, tandis que les plus mauvais résultats sont donnés par la méthode CD.Et dans le cas où Dim>100 la méthode PRP est plus performante que les autres méthodes. Notons que la règle d'Armijo accélère la convergence des méthodes étudiées mieux que celle de Wolfe forte et que si $\epsilon = 10^{-5}$, toutes ces méthodes divergent.

### 4.1.5 La fonction GENROSE (Generalized Rosenbrock)

Cette fonction est numérotée dans l'ensemble des fonctions testes proposées par (Andrei [3, 2008])

$$\begin{aligned} f(x) &= \sum_{i=1}^{n-1} c(x_{2i} - x_{2i-1}^2)^2 - (1 - x_{2i-1})^2, \\ \text{où } c &= 100 \text{ et } x_0 = (-1.2, 1, ..., -1.2, 1). \end{aligned}$$

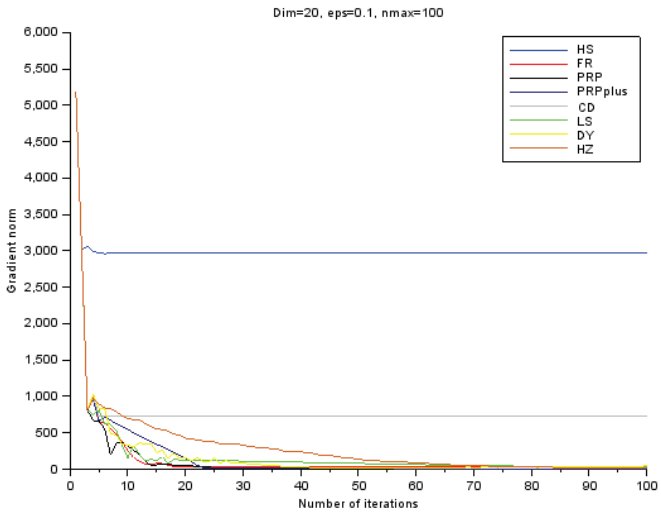


Figure 4.1.9 : Comparaison des CG avec la regle d'Armijo pour GENROSE

| Dim | HS | FR | PRP | PRP+ | CD | LS | DY | HZ |
|---|---|---|---|---|---|---|---|---|
| 10 | Echec | Echec | Echec | Echec | Echec | Echec | Echec | 165/48.612 |
| 50 | Echec | Echec | Echec | Echec | Echec | Echec | Echec | 165/51.59* |

Tableau 12: Différentes CG avec la règle d'Armijo pour GENROSE, $\varepsilon = 10^{-1}$

| Dim | HS | FR | PRP | PRP+ | CD | LS | DY | HZ |
|---|---|---|---|---|---|---|---|---|
| 10 | Echec | Echec | 2478/3.822 | Echec | Echec | Echec | 266/0.655 | 114/0.359* |
| 50 | Echec | Echec | Echec | Echec | Echec | Echec | Echec | 118/24.85* |

Tableau 13: Différentes CG avec la règle de Wolfe forte pour GENROSE, $\varepsilon = 10^{-1}$

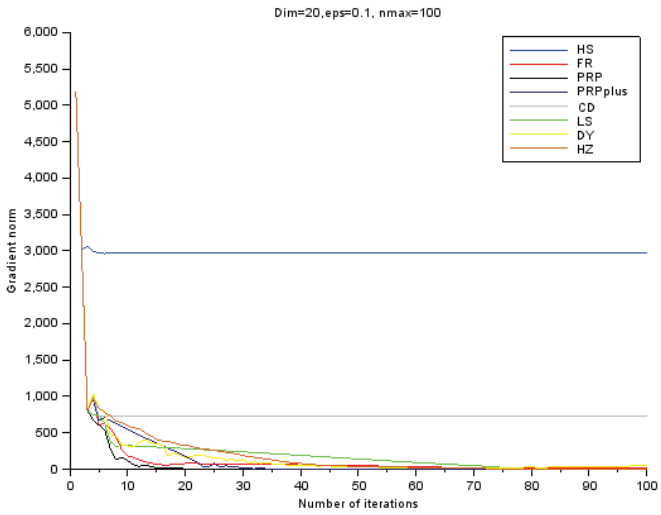


Figure 4.1.10 : Comparaison des CG avec la regle de Wolfe forte pour GENROSE

En comparant les résultats des tableaux 12 et 13, et les figures 4.1.9 et 4.1.10, on observe que pour la fonction GENROSE les meilleurs résultats sont donnés par les méthodes HZ, PRP et DY, et que les plus mauvais résultats sont donnés par les méthodes HS et CD, et malgrés que les méthodes FR, PRP+ et LS divergent, elles réduisent la norme du gradient. Remarquant que la règle de Wolfe forte accélère la convergence des méthodes étudiées mieux que celle d'Armijo.

### 4.1.6 La fonction TRIDIA (cute )

Cette fonction est numérotée dans l'ensemble des fonctions testes proposées par (Andrei [3, 2008])

$$f(x) = \gamma(\delta x_1 - 1)^2 + \sum_{i=2}^{n} (\alpha x_i - \beta x_{i-1})^2, \text{où } \alpha = 2,\ \beta = 1,\ \gamma = 1,\ \delta = 1 \text{ et } x_0 = (1, ..., 1).$$

Tableau 14: Différentes CG avec la règle d'Armijo pour TRIDIA, $\varepsilon = 10^{-1}$

| Dim | HS | FR | PRP | PRP+ | CD | LS | DY | HZ |
|---|---|---|---|---|---|---|---|---|
| 50 | Echec | Echec | 721/2.199 | 336/1.17 | Echec | 335/1.092 | 205/0.11* | 271/0.93 |
| 100 | Echec | Echec | 1199/6.271 | 303/1.91 | Echec | 579/2.948 | 296/2.152 | 136/1.014* |
| 200 | Echec | Echec | 496/4.54 | Echec | Echec | 865/7.41 | 508/4.914 | 288/3.307* |
| 500 | Echec | Echec | Echec | 1276/30.717 | Echec | 2355/52.447 | 889/23.291 | 591/17.05* |
| 700 | Echec | Echec | Echec | Echec | Echec | Echec | 1046/45.454 | 930/40.701* |
| 1000 | Echec | Echec | Echec | Echec | Echec | Echec | 1016/78.737 | 18/66.332* |
| 1500 | Echec | Echec | Echec | Echec | Echec | Echec | 1162/172.462 | 901/140.208 |
| 2000 | Echec | Echec | Echec | Echec | Echec | Echec | 1336/308.487 | 1097/248.843 |

Tableau 15: Différentes CG avec la règle de Wolfe forte pour TRIDIA, $\varepsilon = 10^{-1}$

| Dim | HS | FR | PRP | PRP+ | CD | LS | DY | HZ |
|---|---|---|---|---|---|---|---|---|
| 50 | 46/0.125* | Echec | 69/0.171 | Echec | Echec | 56/0.156 | 65/0.156 | 126/0.234 |
| 100 | Echec | Echec | 419/2.464 | 138/0.718* | Echec | 510/1.918 | 189/0.873 | 205/0.92 |
| 200 | Echec | Echec | Echec | Echec | Echec | Echec | 421/4.627 | 336/4.098* |
| 500 | Echec | Echec | Echec | Echec | Echec | Echec | 748/17.859 | 639/16.03* |
| 700 | Echec | Echec | Echec | Echec | Echec | Echec | 859/26.695 | 675/23.812* |
| 1000 | Echec | Echec | Echec | Echec | Echec | Echec | 817/39.877 | 828/38.173* |
| 1500 | Echec | Echec | Echec | Echec | Echec | Echec | 938/70.629* | 1158/86.203 |
| 2000 | Echec | Echec | Echec | Echec | Echec | Echec | 957/91.813* | 1069/104.944 |

En examinant les résultats des tableaux 15 et 16, et les figures 4.1.11 et 4.1.12, on observe que pour la fonction TRIDIA les meilleurs résultats sont donnés par les méthodes HZ et DY, et que les plus mauvais résultats sont donnés par les méthodes HS et CD, et malgrés que les méthodes PRP+, PRP, LS et FR divergent, elles réduisent la norme du gradient.

Remarquant que la règle de Wolfe forte accélère la convergence des méthodes étudiées mieux que celle d'Armijo.

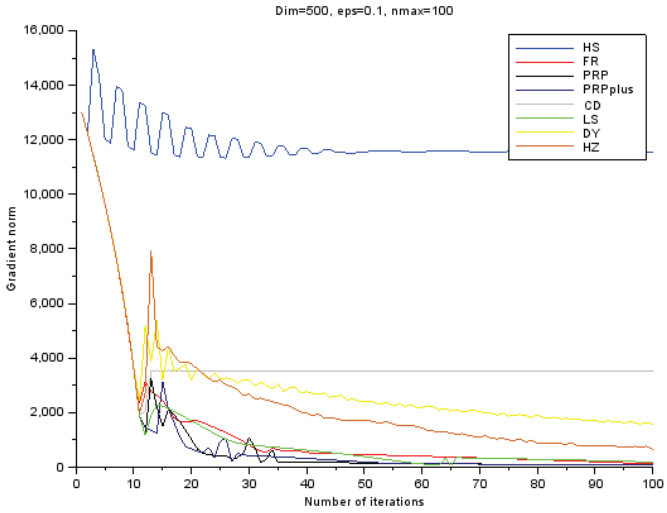


Figure 4.1.11 : Comparaison des CG avec la regle d'Armijo pour TRIDIA

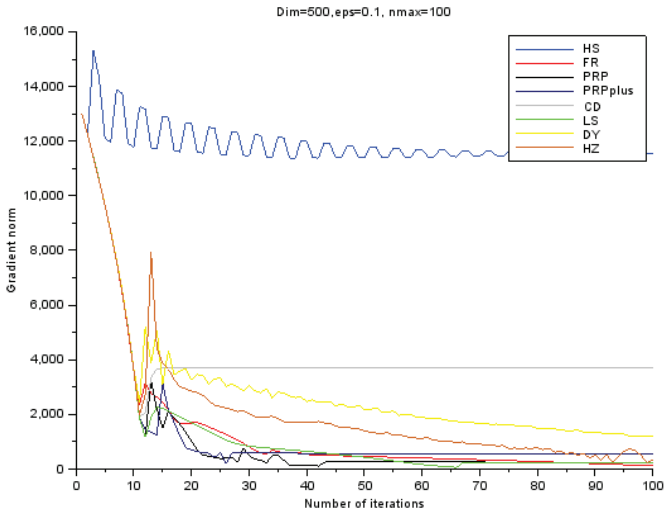


Figure 4.1.12 : Comparaison des methodes CG avec la regle de Wolfe forte pour la fonction TRIDIA

## 4.2 Conclusion

Dans ce chapitre on a fait une étude numérique comparative des performances des méthodes du gradient conjugué non linéaire de HS, FR, PRP, PRP+, CD, LS, DY et HZ, avec les recherches linéaires inexactes de Wolfe forte et d'Armijo avec rebroussement. Ces méthodes sont testées sur un ensemble de six fonctions testes choisies des articles de (Andrei, [3,2008]) et (Moré et Hillstrom, [47,1981]).

Notons que les différentes méthodes étudiées pour ces problèmes divergent si on n'utilise pas les recherches linéaires inexactes étudiées ou si le pas est calculé par la règle de Goldstein-Price ou celle de Wolfe (Wolfe faible). Cet échec de convergence se produit si on utilise les mêmes paramètres utilisés dans les règles de Wolfe forte et d'Armijo. Notons aussi que lors de nos simulations numériques et si parfois on limite le nombre des itérations dans le calcul du pas à 25 itérations, les méthodes étudiées divergent. Remarquons aussi que la détermination du pas initial de la recherche linéaire inexacte utilisée par la règle de Shanno et Phua [59, 1980], et surtout pour des problèmes de grandes tailles, nous a aidé à converger plus rapidement en se comparant au choix du pas initial fixe (généralement on le prend égale à 1), ce qui nous mènent en général à la divergence des méthodes CG. Notons aussi que le redémarrage de la méthode CG (si la variante $\beta_k$ est proche du zéro) accélère aussi la convergence de la méthode CG et évite l'échec du programme.

Durant nos simulations numériques, on a remarqué que malgré que la méthode FR diverge parfois pour des fonctions testes, elle réduit énormément la norme du gradient.

Vu nos résultats numériques on conclut qu'en général la règle de Wolfe forte accélère la convergence des méthodes CG étudiées mieux que celle d'Armijo et que les méthodes HZ, PRP+ et DY sont plus performantes que les autres méthode du CG étudiées.

Notons que l'algorithme du gradient conjugué est particulièrement économe en place mémoire. C'est l'un de ses principaux attraits. Il ne requière que le stockage de 4 vecteurs qui sont mémorisés (les trois premiers sont mis à jour à chaque itération en remplaçant le vecteur de l'itération précédente).

La méthode du gradient conjugué est une méthode dans laquelle les erreurs d'arrondi sont amplifiées au cours des itérations. Peu à peu les relations de conjugaison des gradients avec

les premières directions sont perdues et cela est dû à la présence d'erreurs d'arrondi dans les relations d'orthogonalité des gradients entre eux qui ne sont pas vérifiées exactement dans la pratique. En imposant moins de conditions dans le calcul de la variante de HZ (conjugaison et orthogonalité) que celles de HS, FR, PRP, PRP+, CD, LS et DY, la méthode du gradient conjugué de HZ est considérée plus performante qu'elles.

Dans le cas linéaire les différentes variantes sont égales théoriquement, mais dans la pratique les résultats montrent le contraire.

# Conclusion et Perspectives

Dans cette thèse on a fait une étude numérique comparative des performances de deux méthodes récentes, et des combinaisons entre elles. Il s'agit des méthodes multigrille(MG) et gradient conjugué(CG).
Notre travail se divise en deux parties. La première partie consacrée à une étude numérique comparative de deux approches différentes de combinaison des méthodes multigrille et gradient conjugué, à savoir les méthodes MG-CG et MGcgM, nous a permis de résoudre le problème du Laplacien par sous domaine avec les conditions aux limites de Dirichlet homogènes, discrétisé par le schéma classique des différences finies à cinq points.

Et l'objectif de démontrer les avantages des schémas hybrides de combinaison des algorithmes du multigrille et gradient conjugué a été atteint. Nos résultats numériques prouvent que ces algorithmes hybrides sont très efficaces pour la résolution des problèmes linéaires à petite et moyenne taille. Dans nos simulations numériques six différentes méthodes ont été comparées [méthode multigrille-gradient conjugué (MG-CG), méthodes multigrille-gradient (MGgM) et multigrille gradient conjugué (MGcgM) à deux et à trois niveaux, méthode multigrille (MG), méthode du gradient conjugué (CG), méthode du gradient conjugué préconditionné (PCG) et méthode GMRES. Notons qu'on a utilisé la méthode de Jacobi comme un lisseur dans la méthode multigrille (MG)]. Les méthodes MGgM et MGcgM ont été utilisé au niveaux l=2 et l=3.
En comparant les méthodes MG-CG, MGgM, MGcgM, MG, CG et PCG on remarque que la méthode MG-CG est plus efficace pour notre problème.

Dans la deuxième partie on a illustré numériquement la contribution des recherches linéaires d'Armijo et de Wolfe forte dans l'accélération de la convergence des méthodes CG en comparant huit différentes méthodes du gradient conjugué non linéaire de HS, FR, PRP, PRP+, CD, LS, DY et HZ où le pas est calculé par une recherche linéaire inexacte (Wolfe forte ou Armijo) avec rebroussement, pour six fonctions testes choisies de deux articles (Andrei 2008 et Moré and Hillstrom 1981). Notons aussi que le pas de Shanno a joué un role important dans la convergence des méthodes du gradient conjugué non linéaires.

Dans nos tests numérique on a remarqué que ces méthodes divergent si le pas est calculé par une recherche linéaire exacte ou par les règles de Goldstein-Price ou de Wolfe faible

Notons que l'algorithme du gradient conjugué est particulièrement économe en place mémoire. C'est l'un de ses principaux attraits. Il ne requière que le stockage de 4 vecteurs qui sont mémorisés (les trois premiers sont mis à jour à chaque itération en remplaçant le vecteur de l'itération précédente).

La méthode du gradient conjugué est une méthode dans laquelle les erreurs d'arrondi sont amplifiées au cours des itérations. Peu à peu les relations de conjugaison des gradients avec les premières directions sont perdues et cela est dû à la présence d'erreurs d'arrondi dans les relations d'orthogonalité des gradients entre eux qui ne sont pas vérifiées exactement dans la pratique. En imposant moins de conditions dans le calcul de la variante de Harger-Zhang (conjugaison et orthogonalité) que celles de Hestense-Stiefel, Fletcher-Reeves, Descente conjuguée, Polak-Polyak-Ribière, Liu-Storey et Dai-Yuan, la méthode du gradient conjugué de Harger-Zhang est considérée plus performante que les méthodes du gradient conjugué non linéaires citées au dessus.

Notons que dans le cas linéaire les différentes variantes sont égales théoriquement, mais dans la pratique, les résultats montrent que la méthode du Harger-Zhang est meilleure que les autres méthodes.

Les programmes sont écrits sous scilab 5.4.0 et exécuté dans un ordinateur Intel PC i5 de vitesse de 3.10 GHz et une RAM de 4 Go et un système Windows sept (64 bits).

**Perspectives**

•) L'utilisation de notre générateur de programmes pour la résolution de certains problèmes à frontières libre.

•) Etablir des programmes permettant de calculer le pas par la méthode à région de confiance liée aux options américaines.

•) On a pu faire des programmes des méthodes MGgM et MGcgM à deux et à trois niveaux. On a remarqué que les méthodes à trois niveaux sont plus performantes que ceux à deux niveaux. On opte à établir des programmes de ces méthodes à niveau superieure et essayer de généraliser ces programmes.

# Bibliographie


[1] M. Al-Baali, Descent property and global convergence of the Fletcher-Reeves method with inexacte line search. *IMA J. Num. Anal., Vol. 5, pp.121-124, (1985).*

[2] R. E. Alcouffe, A. Brandt, J.E. Dendy, J. W. Painter, The multi-grid method for diffusion equation with strongly discontinuous coefficients. SIAM J. Sci. Statis. Comput., vol.2,pp.430-454, (1981).

[3] N. Andrei, An unconstrained optimization test functions collection, Advenced Modeling and Optimization 10(1): 147–161, (2008).

[4] L. Armijo, Minimzation of function having lipschitz continous first partial derivatives, *Pacific Journal of Mathematics, Vol. 16(1), pp.1-3, (1966).*

[5] R. E. Bank and C. C. Douglas, Sharp estimates for multi-grid rates of convergence with general smoothing and acceleration, SIAM J. Numer. Anal. 22, pp. 617-633 (1985).

[6] M. S. Bazaraa, H. D. Sherali, et C. M. Shetty, Nonlinear Programming, Theory and Algorithms, *Wiley-Interscience, (1993).*

[7] M. Bergounioux, Optimisation et Contrôle des Systèmes Linéaires, *Dunod, (2001).*

[8] D. Braess, On the combination of the multigrid method and conujugate gradients, in Multigrid Methods II (W. Hackbusch and U. Trottenberg, eds.), vol. 1228 of Lecture Notes in Mathematics, pp.52-64, Springer-Verlag, (1986).

[9] D. Braess and W. Hackbusch, A new convergence proof for the multigrid method including the V-cycle, SIAM J. Numer. Anal. 20, 967-975, (1983).

[10] A. Brandt, Multilevel adaptive technique (MLAT) for fast numerical solution to boundary value problems, Proc. Third international Conference on Numerical Methods in Fluid Mechanics, Paris (1972).

[11] A. Brandt, Multi-level adaptive solutions to boundary-value problems. Mathematics of Computation 31, 138, 333–390, (1977).

[12] A. Brandt, S. F. McCormick, J. Ruge, Algebraic multigrid (AMG) for automatic multigrid solution with application to geodic computations. Institute for Computational Studies, POB 1852, Fort Collins, Colorado, (1982).

[13] W. L. Briggs, V. E. Henson and S. F. McCormick , A Multigrid Tutorial, 2nd edn, SIAM, Philadelphia, USA (2000).

[14] A. Cauchy, Analyse mathématique, Méthode générale pour la résolution des systèmes d'équations simultanées, *Comptes Rendus de l'Académie des Sciences de Paris, t-25, pp. 536-538, (1847).*

[15] Y. H. Dai and Y. Yuan, Convergence properties of the Fletche-Reeves method, *IMA J Numer. Anal., Vol.16(2), pp. 155-164, (1996).*

[16] Y. H. Dai and Y. Yuan, Convergence properties of the conjugate descent method, *Advances in Mathematics, 6, pp.552-562,* (1996).

[17] Y. H. Dai and Y. Yuan, Some properties of a new conjugate gradient method, in : *Advances in Nonlinear Programming, ed . Kluwer Academic, Boston, pp. 251-262, (1998).*

[18] Y.H. Dai and Y. Yuan, A non linear conjugate gradient with a strong global convergence property, *SIAM J. Optimization, Vol. 10(1),*

*pp.177-182, (1999).*

[19] H. Dai and Y. Yuan, New properties of a new conjugate gradient method, *Numer. Math., Vol. 89(2), pp. 83-98, (2001).*

[20] R. P. Fedorenko, A relaxation method for solving elliptic difference equation. U.S.S.R. Comput. Math. And Math. Phys., (1962).

[21] A. V. Fiacco and G. P. McCormick, Nonlinear Programming, *John Wiley, New York, (1968).*

[22] R. Fletcher, Practical methods of optimization, *John Wiley&Sons, Chichster, (1987).*

[23] R. Fletcher and M. J. D. Powell, A rapidly convergent descent method for minimization, *Computer. J., 6, pp. 163-168, (1963).*

[24] R. Fletcher and C. Reeves, Function minimization by conjugate gradients. *Comput. J., 7, pp.149-154, (1964).*

[25] J. C. Gilbert, Eléments d'Optimisation Différentiable : Théorie et Algorithmes, *Notes de cours, École Nationale Supérieure de Techniques Avancées, Paris, (2007).*

[26] J.C. Gilbert and J. Nocedal, Global convergence properties of conjugate gradient methods for optimization, *SIAM J. Optimization. Vol. 3, No.1, pp. 21-42, (1992).*

[27] A.A. Goldstein, Constructive Real Analysis*, A Harper International Edition,* (1967).

[28] A. A. Goldstein, On steepest descent, *SIAM J. on Control A, Vol. 3, No. 1, pp. 147-151, (1965).*

[29] A.A. Goldstein and J.F. Price, An effective algorithm for minimzation, *Num.Math., 10, pp. 184-189, (1969).*

[30] W. Hackbusch, On the multi-grid method applied to difference equations. Computing, vol.20, pp.291–306, (1978).

[31] W. Hackbusch, H. D. Mittelmann, On Multi-grid Methods for Variational Inequalities, Numer. Math.42, pp65-75, (1983).

[32] W.W. Hager et H. Zhang, A new conjugate gradient method with guaranteed descent and an efficient line search, *SIAM J. Optim., 16,* pp. *170-192, (2005).*

[33] W.W. Hager et H. Zhang, CG DESCENT, a Conjugate Gradient Method with Guaranteed Descent, ACM Transactionson MathematicalSoftware, Vol. 32, No. 1, March, Pages 113–137, (2006).

[34] M. R. Hestenes and E.L. Stiefel, Methods of conjugate gradients for solving linear systems, *J. Res. Nat. Bur. Standars Sect., 5(49),*

*pp. 409-436, (1952).*

[35] R. H. W. Hoppe, Multi-Grid Methods For Hamilton-Jacobi- Bellman Equations, Numer. Math. 49, 239-254, (1986).

[36] R. H. W. Hoppe, Multi-Grid Methods for Variational Inequalities, SIAM J. Numer. Anal., vol. 24,n°5, (1987)

[37] Y.F. Hu and C. Story, Global convergence result for conjugate gradient methods, *JOTA, 71(2), pp. 399-405, (1991).*

[38] A. Janka et H. Guillard, Développement et tests de méthodes multi-grille pour l'accélération de la convergence des méthodes de résolution des équations de Navier-Stokes sur maillages non structurés, Final report, Dassault contract no. 42G01441, ref. AW/MIT, (2003).

[39] A.Laouar, Aspects de l'analyse numérique de méthodes itératives de point fixe : Erreurs d'arrondi, Accélération de convergence, Sous domaines, Thèse de Doctorat de l'Université de Franche-comté, France, (1988).

[40] Z. Li, J. Chen and N. Deng, A New Conjugate GradientMethod and its Global Convergence Properties, *Systems Science and Mathematical Sciences, Vol. 11, pp. 53–60, (1998).*

[41] G.H. Liu and J.Y. Han and H.X. Yin, Global convergence of the Fletcher-Reeves algorithm with an inexact line search, *Appl. math. J. Chinese Univ. Ser. B, 10, pp. 75-82, (1995).*

[42] Y. Liu and C. Storey, Efficient generalized conjugate gradient algorithms: I. theory, J. Optim. Theory Appl.69: 129–137, (1991).

[43] D. G. Luenberger, Optimization by Vector Space Methods, *John Wiley and Sons, Inc. New York, (1969).*

[44] R. Mellal and M. Haiour, Numerical simulations of some nonlinear conjugate gradient methods with inexact line searches, Proceedings of International Conference : Mathematical Science and Applications, Abu Dhabi, UAE. Universal Journal of Mathematics and Mathematical Sciences, Volume 4, Number 1, Pages 63-84, (2013).

[45] M. Minoux, Programmation Mathématique, Théorie etAlgorithmes, tome 1 , *Dunod, (1983).*

[46] K. Mocellin, Contribution à la simulation numérique tridimensionnelle du forgeage à chaud. Etude du contact et calcul multigrille. Thèse de doctorat, Ecole Nationale supérieure des Mines de Paris, Sophia Antipolis, (1999).

[47] J.G. B. Moré and K. Hillstrom, Testing unconstrained optimization software, ACM-Transactions on Mathematical Software7: 17–41,( 1981).

[48] J. Nocedal, Theory of algorithm for unconstrained optimization, *Acta Numerica, pp. 199-242, (1991).*

[49] C. Pflaum, A multigrid conjugate gradient method, Applied Numerical Mathematics, Volume 58, Issue 12, Pages 1803-1817, (2008).

[50] E. Polak and G. Ribière, Note sur la convergence de directions conjuguées, *Rev. Française Informat. Recherche Operationelle, 3e année, 16, pp. 35-43, (1969).*

[51] B.T. Polyak, The conjugate gradient method in extremem problems, *Comput. Math. Math. Phys., 9, pp. 94-112, (1969).*

[52] M.J.D. Powell, Convergence properties of algorithms for nonlinear optimization, *SIAM rev., 28, pp. 487-500, (1986).*

[53] M.J.D. Powell, Non convex minimzation calculation and the conjugate gradient method, in : *Lecture Notes in Mathematics 1066 (Springer, Berlin), pp. 122-141, (1984).*

[54] M.J.D. Powell, On the convergence of the variable metric algorithm; *J. Inst. Math. Appl, 7, pp. 21-36, (1971).*

[55] M.J.D. Powell, Restart procedures for the conjugate gradient method, *Math Programming, 2 , pp 241-254, (1977).*

[56] A. Quarteroni and F. Saleri, Scientific Computing with Matlab and Octave, Second Edition, Springer, May (2006).

[57] A. Reusken, On Maximum Norm Convergence of Multigrids Methods for Elliptic Boundary Value Problems, SIAM J. Numer. Anal., vol. 29,n°6,pp1569-1578, December (1992).

[58] Y. Saad, Iterative methods for sparse linear systems, 2nd ed., SIAM, (2003).

[59] D. F. Shanno, and K. H. Phua, Remark on algorithm 500: Minimization of unconstrained multivariate functions, ACM Trans. Math. Software 6: 618–622, (1980).

[60] R. V. Southwell, Stress-calculation in frameworks by the method of "systematic relaxation of constraints. Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences 151, 872, 56–95, (1935).

[61] R. V. Southwell,. Relaxation methods in theoretical physics. Tech. rep., Oxford University Press, (1946).

[62] D. Touati-Ahmed and C. Storey, Efficient hybrid conjugate gradient techniques, *JOTA, 64, pp. 379-397, (1990).*

[63] P. Wesseling, Theoretical and practical aspects of a multigrid method. SIAM J. Sci. Comput., vol.3, pp.387–407, (1982).

[64] P. Wesseling, An introduction to multigrid method. Wiley, Chichester, (1992).

[65] P. Wolfe, A duality theorem for nonlinear programming, *Quart. Appl. Math.*, 19, pp. 239-244, *(1961).*

[66] P. Wolfe, Convergence conditions for ascent methods, *SIAM Review, 11, pp. 226-235, (1969).*

[67] P.Wolfe, Conditions for ascent methods some corrections 2, *SIAM Review, 13, pp.185-188, (1971).*

[68] P. Wolfe, Methods of Nonlinear Programming, *in Recent Advances in Mathematical Programming (Eds R. L. Graves and P. Wolfe), McGraw-Hill, New York, (1963).*

[69] Y.X.Yuan, Numerical Methods for Non linear Programming, *Shanghai scientific & Technical Publishers (in Chinese), (1993).*

[70] H. Yserentant, Old and new convergence proofs for multigrid methods, Acta Numeric, pp. 285-326, (1993).

[71] G. Zoutendijk, Nonlinear Programming Computational Methods, *Integer and Nonlinear Programming, North Holland, Amsterdam, (1970).*

[72] G. Zoutendijk, Methods of Feasible Directions, *Elsevier, Amesterdam, (1960).*