People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

Setif 1 University Ferhat Abbas
Faculty of Sciences
Department of Mathematics

جامعـة سطيف 1 فرحـــات عباس
كليـــة العلـــــــوم
قســـــم الريـاضيـــــــات

Doctoral Thesis

submitted in fulfillment of the requirements for obtaining the degree of Doctor in Mathematics

Speciality: Optimization and Control

Theme:

New families of hybrid conjugate gradient descent directions for nonlinear optimization with practical applications

Presented by

Mr. Youcef El Hamam HEMICI

Supervisor: **Dr. Samia KHELLADI**
Co-supervisor: **Pr. Djamel BENTERKI**

Thesis defended on April 30th, 2026, in front of the jury composed of:

Mr.	Rachid ZITOUNI	Prof.	Setif 1 University Ferhat Abbas	President
Mme.	Samia KHELLADI	MCA	Setif 1 University Ferhat Abbas	Supervisor
Mr.	Djamel BENTERKI	Prof.	Setif 1 University Ferhat Abbas	Co- Supervisor
Mme.	Fatma Zohra NOURI	Prof.	University Badji Mokhtar, Annaba	Examiner
Mr.	Noureddine BENHAMIDOUCHE	Prof.	University Mohamed Boudiaf, Msila	Examiner
Mr.	Mohamed ACHACHE	Prof.	Setif 1 University Ferhat Abbas	Examiner
Mr.	Bachir MERIKHI	Prof.	Setif 1 University Ferhat Abbas	Guest

2026

ACKNOWLEDGMENTS

First of all, I would like to thank Allah (Exalted is He above all) who helped us fulfill this research work.

I extend my deep appreciation to my esteemed supervisor, Dr. Samia KHELLADI, for her unwavering support, understanding, and invaluable contributions that have greatly enriched my experience. I also wish to express sincere thanks to my co-supervisor, Professor Djamel BENTERKI, for his guidance and assistance.

I am grateful to Professor Rachid ZITOUNI, for chairing the committee, as well as to Professors Fatma Zohra NOURI, Noureddine BENHAMIDOUCHE and Mohamed ACHACHE, for examining my thesis. My gratitude also to Professor Bachir MERIKHI for his kind acceptance to take part in the jury as member invited.

My gratitude also to Professor Adnan YASSINE for his insightful comments and fruitful discussions during my visit to Le Havre University.

Special thanks are for Dr. Billel ZAOUI and my collegue Mohamed Lamine OUAOUA for their support and help.

I extend my appreciation to all my colleagues and fellow members of the Mathematics Department at Setif 1 University-Ferhat Abbas, especially those in the LMFN Laboratory.

Finally, I would like to thank all those who contributed directly or indirectly to the elaboration of this work.

HEMICI YOUCEF EL HAMAM

إهداء

الحمد لله تعالى أولًا وآخرًا، حمدًا يليق بجلاله وعظيم سلطانه،
الذي بنعمته تتمّ الصالحات، وبتوفيقه وصلتُ إلى هذه المرحلة.
أتقدّم بخالص الشكر وعظيم الامتنان إلى والديَّ الكريمين،
اللذين كانا سندي الدائم، وسببًا بعد الله في ما وصلتُ إليه،
عرفانًا بتضحياتهما وصبرهما ودعمهما اللامحدود.
كما أشكر أخواتي
على محبتهم الصادقة ومساندتهم الدائمة لي في مسيرتي العلمية.
وأتقدّم بشكر خاص وعميق إلى أستاذتي التي أعتبرها والدتي الثانية.
الأستاذة سامية خلادي،
على مساندتها وتوجيهاتها لي ودعمها المتواصل وتشجيعها الصادق.
كما لا يفوتني أن أعبّر عن شكري الخالص وامتناني العميق
إلى الأستاذ جمال بن تركي،
الذي ساندني ورافقني في كل كبيرة وصغيرة،
ولم يبخل عليّ بنصائحه القيمة وتوجيهاته العلمية.
وأتوجّه بالشكر والتقدير إلى جميع أساتذتي الأفاضل، دون استثناء،
الذين كان لهم الفضل في تكويني العلمي.
كما أتقدّم بشكري وامتناني إلى زملائي
على روح التعاون والمساندة المتبادلة طيلة سنوات التكوين.
كما أشكر أصدقائي الأعزاء
على وقوفهم إلى جانبي ودعمهم المعنوي خلال هذه المسيرة
إلى كل من ساندني ولو بكلمة طيبة،
أهدي هذا العمل المتواضع،
سائلًا الله تعالى أن يجعله خالصًا لوجهه الكريم.

Contents

Articles

Published

- Youcef Elhamam Hemici, Samia Khelladi and Djamel Benterki, *New hybrid conjugate gradient method for nonlinear optimization with application to image restoration problems*, **Kybernetika**, 60(4), (2024), 535–552.
  https://doi.org/10.14736/kyb-2024-4-0535

- Youcef Elhamam Hemici, Samia Khelladi and Djamel Benterki, *An Efficient Hybrid Three-Term Conjugate Gradient Method with Practical Applications*, **Iranian Journal of Numerical Analysis and Optimization**, 16(1), (2026), 264–282.
  https://ijnao.um.ac.ir/

- Samia Khelladi and Youcef Elhamam Hemici, *A New Hybrid Conjugate Gradient Method Based on RMIL, LS and CD Methods*, **Nonlinear Dynamics and Systems Theory**, 26(2) (2026), 184–194.
  http://www.e-ndst.kiev.ua/v26n2/7(104)a.pdf

- Youcef Elhamam Hemici, Samia Khelladi and Djamel Benterki, *An Efficient Hybrid Conjugate Gradient Method for Large-Scale Nonlinear Equations with Applications in Compressive Sensing*, **Journal of Nonlinear Modeling and Analysis**, 16(1), (2026).

- Youcef Elhamam Hemici, Samia Khelladi, Djamel Benterki and Bilel Zaoui, *New conjugate gradient method of Wei-Yao-Liu type applied to image restoration and compressive sensing problems*, **Algerian Journal of Science** , 2(2), (2026).

In revision

- Youcef Elhamam Hemici, Samia Khelladi and Djamel Benterki, *New Hybrid RMIL-FR Conjugate Gradient Method for Unconstrained Nonlinear Optimization*, **Filomat**, XX(X), XXX–XXX.

List of Communications

International communications

1. Hemici Youcef Elhamam, Khelladi Samia, Benterki Djamel, *Comparative numerical study on Fletcher–Reeves and Hestenes–Stiefel hybridization method*, International Conference on Contemporary Mathematics and its Applications (ICCMA 2023), 26–27 Novembre 2023, Centre Universitaire Abdelhafid Boussouf Mila.
   https://icma23mila.sciencesconf.org/

2. Hemici Youcef Elhamam, Khelladi Samia, Benterki Djamel, *Study of the numerical behavior of the EDL method towards other conjugate gradient methods of the type Dai–Liao*, International Conference on Nonlinear Mathematical Analysis and Its Applications (ICNMAA 2024), 14–15 Mai 2024, Université Bordj Bou Arreridj.
   https://icnmaa24mai2024bba.mystrikingly.com/

3. Hemici Youcef Elhamam, Khelladi Samia, Benterki Djamel, *A Comparative Study of Hybrid Three-Term Conjugate Gradient Methods*, International Symposium on Applied Mathematics and Mathematical Modeling (MAMM'2024), 12 Novembre 2024, Université Chadli Bendjedid El-Tarf.
   https://mamm.sciencesconf.org/

4. Hemici Youcef Elhamam, Khelladi Samia, Benterki Djamel, *Study of a modified PRP method using a diagonal Hessian approximation approach*, International Conference on Mathematics and its Applications in Science and Technology (ICMAST'2024), 15–16 Décembre 2024, Université Sétif 1 – Ferhat Abbas.
   https://events.univ-setif.dz/event/6/

5. Hemici Youcef Elhamam, Samia Khelladi and Djamel Benterki, *A Novel Hybrid Conjugate Gradient Method Based on a Convex Combination of RMIL and HS Parameters*,

Silver Jubilee International Pure Mathematics Conference (25th IPMC 2025), 29–31 August 2025, Islamabad, Pakistan. https://www.pmc.org.pk/

6. Hemici Youcef Elhamam, Khelladi Samia, Benterki Djamel, *New conjugate gradient method of Wei-Yao-Liu type based on RMIL approach for nonlinear optimization*, The First lnternational Conference on Algebraic Structures and Partial Differential Equations October (ICAPDE) 28-29, 2025, ENS Setif, El Eulma, Algeria. https://www.ens-setif.dz/index.php/les-manifestations-scientifiques/math-conf-25-26novembre2025

National communications

1. Hemici Youcef Elhamam, Samia Khelladi and Djamel Benterki, *Study of a Diagonal Conjugate Gradient-Like Method Using Different Line Searches*, National Conference on Mathematics and Applications (NCMA'2024), 27–28 Novembre 2024, University of Mohamed Boudiaf – M'sila. https://ncma2024.sciencesconf.org/

2. Hemici Youcef Elhamam, *Numerical comparison between a modified PRP and LS methods using a diagonal Hessian approximation approach*, National Conference of Mathematics and Application (CNMA2024), 7–8 Décembre 2024, Abdelhafid Boussouf University Center of Mila. https://sites.google.com/centre-univ-mila.dz/cnma2024/

3. Hemici Youcef Elhamam, *Study of a two-parameter conjugate gradient method with different formulas to calculate the stepsize*, National Symposium on Innovative Mathematics: Retrospective and Perspectives (SNMI), 11–12 Décembre 2024, Abdelhamid Ibn Badis University – Mostaganem. https://snmi.sciencesconf.org/

4. Hemici Youcef Elhamam, Samia Khelladi and Djamel Benterki, *New conjugate gradient direction of Wei-Yao-Liu type for nonlinear optimization problem*, National Conference on Mathematics And Applications, University of Adrar. (NCMA2025). November 4-5, 2025. https://fsmmi.univ-adrar.edu.dz/conferences/967/

List of Figures

List of Tables

Glossary of Abbreviations and Notations

Abbreviations

FR Fletcher–Reeves

PRP Polak–Ribière–Polyak

HS Hestenes–Stiefel

CD Conjugate Descent

LS Liu–Storey

DY Dai–Yuan

DL Dai–Liao

HZ Hager–Zhang

RMIL Rivaie–Mustafa–Ismail–Leong

WYLDL Wei–Yao–Liu type of Dai–Liao

MDLS Modified Dai–Liao–Storey

DLS Dai–Liao–Storey

PSNR Peak Signal-to-Noise Ratio

MSE Mean Squared Error

CPU Computing time

CUTE Constrained and Unconstrained Testing Environment (test library)

Mathematical Notations

- $\mathbb{R}^n$: n-dimensional Euclidean space
- $\mathbb{N}$: Set of natural numbers
- Ω: Level set $\{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$
- $V(x_0)$: Neighborhood of point x_0
- $\nabla f(x)$: Gradient of f at x
- $\nabla^2 f(x)$: Hessian matrix of f at x
- $\partial f(x)$: Subgradient of f at x
- $\langle x, y \rangle$: Inner product of x and y
- $\|x\|$: Euclidean norm
- x_k: Iterate at step k
- d_k: Search direction at step k
- g_k: Gradient of f at x_k, i.e., $\nabla f(x_k)$
- α_k: step-size at iteration k
- β_k: conjugate gradient parameter
- y_k= $g_{k+1} - g_k$
- s_k= $x_{k+1} - x_k$
- H_k Hessian of f at point x_k, $H_k = \nabla^2 f(x)$
- C1: Lipschitz continuity condition
- C2: Boundedness condition
- NC: Necessary Condition
- SC: Sufficient Condition

- NSC: Necessary and Sufficient Condition
- $PSNR$: Peak Signal-to-Noise Ratio, quality measure for images
- MSE: Mean Squared Error, defined as $\frac{1}{n}\|x - x^\star\|^2$
- $\arg\min$: Argument of the minimum
- $\liminf$: Inferior limit

Introduction

In applied mathematics and engineering, the optimization is considered as an important domain, due to its main objective of determining the best possible decision among a set of available alternatives. It is widely applied across various fields such as science, engineering, economics and industry, offering powerful tools and techniques to minimize losses, maximize profits and enhance system performance. Optimization contributes significantly to saving time, money, energy and materials, while also improving quality and maximizing user satisfaction. These techniques are employed in solving both simple analytical problems and managing complex systems.

Given its broad range of applications and growing importance, optimization plays a central role in both theoretical research and practical scientific applications.

Among the many branches of optimization, unconstrained optimization occupies a particularly important place, especially due to its theoretical simplicity and wide range of practical relevance. In unconstrained optimization, the goal is to find the minimum (or maximum) of a real-valued differentiable function without imposing any explicit constraints on the variables. Mathematically, the general form of such a problem is

$$\min_{x \in \mathbb{R}^n} f(x),$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function. Despite the apparent simplicity of this formulation, such problems can pose significant challenges, particularly with high dimensional problems.

To address these challenges, numerous numerical algorithms have been developed over the decades. Among the most prominent and efficient is the conjugate gradient method, it stands out as a powerful tool, particularly for this type of problems, due to its efficiency, low memory requirements and broad applicability in fields such as scientific computing, engineering, machine learning, data science, image processing,

and compressive sensing. It was firstly introduced by Hestenes and Stiefel (HS) in 1952 [30], to minimize convex quadratic functions. Later in 1964, Fletcher and Reeves (FR) [23] extended the method for solving nonlinear unconstrained optimization problems, then modified by Polak-Ribière-Polyak (PRP) [57, 58], since then several variants have been proposed until our days.

The general structure of conjugate gradient methods involves generating a sequence of iterates $\{x_k\}$ starting from intial point x_1, using a search direction d_k and a stepsize α_k as follows

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

where the search direction d_k is given by

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

and $g_k = \nabla f(x_k)$ is the gradient of f at x_k. The performance of conjugate gradient methods largely depends on the choice of the parameter β_k and numerous formulas have been proposed in the literature, each with distinct theoretical properties and practical behavior. Some of the most well-known choices of β_k include those proposed by Hestenes and Stiefel, Fletcher and Reeves (FR) [23], Polak-Ribière-Polyak (PRP) [57, 58], Conjugate Descent (CD) [22], Liu-Storey (LS) [44] and Dai-Yuan (DY) [13, 14]. The formulas of the parameters β_k mentioned above are

$$\beta_k^{HS} = \frac{g_{k+1}^T y_k}{d_k^T y_k}, \ \beta_k^{FR} = \frac{\|g_{k+1}\|^2}{\|g_k\|^2}, \ \beta_k^{PRP} = \frac{g_{k+1}^T y_k}{\|g_k\|^2},$$

$$\beta_k^{CD} = -\frac{\|g_{k+1}\|^2}{g_k^T d_k}, \beta_k^{LS} = -\frac{g_{k+1}^T y_k}{g_k^T d_k}, \ \beta_k^{DY} = \frac{\|g_{k+1}\|^2}{d_k^T y_k},$$

where $y_k = g_{k+1} - g_k$ and $\|.\|$ denotes the Euclidean norm.

While classical conjugate gradient methods FR, DY, CD, LS, HS and PRP have laid the theoretical foundation for solving unconstrained optimization problems, their behavior significantly differs depending on the nature of the objective function and the type of the used line search. These methods are identical when applied to strongly convex quadratic functions with exact line search. However, they exhibit varying performance when extended to general nonlinear functions under inexact line search.

Specifically, the FR, DY and CD methods are known for their strong theoretical convergence properties, yet they often suffer from poor practical performance. In contrast, LS, HS and PRP methods are widely recognized for their efficiency in practice, though they may fail to converge in certain situations [18].

To address these limitations, hybrid conjugate gradient methods have emerged as a promising alternative. These methods combine different conjugate gradient method formulas to leverage their respective strengths and mitigate weaknesses particularly to overcome the jamming phenomenon and the potential loss of sufficient descent. Recently, several search directions for conjugate gradient methods have been proposed by researchers worldwide. Building on the work of Dai and Liao [15], numerous formulas for the parameter β_k have been introduced to further modify and enhance the behavior of conjugate gradient algorithms. These developments aim to improve convergence properties, numerical stability, and overall computational performance. Meanwhile, additional approaches defining new families of search directions have been proposed by other authors, expanding both the theoretical framework and practical applicability of conjugate gradient methods presented in [4, 42, 55, 56, 68].

Hybridization can be either convex or non-convex in nature and aims to enhance both robustness and overall performance. This line of research remains active and continues to evolve. Notable hybrid approaches was given in [6, 16, 19, 18, 52, 61, 62, 63, 64, 70].

This thesis is focusing on the development and analysis of some new hybrid conjugate gradient methods for unconstrained nonlinear optimization problems. The primary objective is to construct efficient and globally convergent algorithms by proposing new formulations of the search direction and new conjugacy parameters.

The thesis contains introduction, five chapters and a general conclusion. The first chapter provides a comprehensive overview of convex analysis and unconstrained optimization. It covers key mathematical concepts, optimality conditions and foundational algorithms such as gradient, Newton and classical conjugate gradient methods. Additionally, it discusses line search techniques, both exact and inexact, which play a crucial role in the practical implementation of descent methods. The second chapter contains our first contributions, where we introduce three new hybrid conjugate gradient methods based on convex combinations of existing formulas. The proposed algorithms are applied to standard theoretical tests problems and to image restoration problems.

Besides, their convergence properties and numerical performance are analyzed and disused in detail. The third chapter presents our new method for solving large-scale nonlinear monotone equations subject to convex constraints applied to sparse signal problems. The algorithm's theoretical convergence is established and its effectiveness is shown through extensive numerical experiments. In the forth chapter we propose another new contribution based on hybridization of the Liu-Storey β_k^{LS} parameter with its modification β_k^{DLS}. The proposed algorithm incorporates a hybrid three-term technique applied to image restoration and sparse signal recovery. The chapter five contains our last contribution which represents a new efficient search direction based on a new parameter β_k^{ESDB}. The obtained algorithm globally converges and the numerical experiments are significant and confirm our purpose. Finally, a general conclusion concludes this manuscript.

CHAPTER 1

Overview of convex analysis and unconstrained optimization

In this chapter, we explore the fundamental principles of convex analysis and unconstrained optimization. We start by examining convex sets and convex functions, which are important in defining and solving optimization problems.

Then, we focus on understanding the conditions of existence and uniqueness of solutions, along with first and second-order optimalty conditions of an unconstrained optimization problem. To solve these problems, we introduce key methods such as gradient descent, Newton's method, and conjugate gradient methods, discussing their structure and convergence behavior.

Finally, we pass into line search techniques, both exact and inexact, which help to determine stepsize α_k in iterative methods. This chapter provides a comprehensive starting point for applying optimization methods effectively.

1.1 Convex sets and convex functions

Convex sets and convex functions are fundamental concepts in optimization theory. In this section, we introduce the key definitions and essential properties of convex sets and convex functions.

1.1.1 Convex sets

Definition 1.1. A set $C \subseteq \mathbb{R}^n$ is convex if for any two points $x, y \in C$ and any scalar $\lambda \in [0,1]$, the convex combination $\lambda x + (1-\lambda)y$ also lies within C, i.e.,

$$\lambda x + (1-\lambda)y \in C, \quad \forall x, y \in C,\ \forall \lambda \in [0,1].$$

Definition 1.2. A convex combination of m vectors $x_1, x_2, \ldots, x_m \in \mathbb{R}^n$ is defined as a linear combination of the form

$$\sum_{i=1}^{m} \lambda_i x_i$$

where $\lambda_i \geq 0$ for all $i = 1, \ldots, m$ and

$$\sum_{i=1}^{m} \lambda_i = 1.$$

Proposition 1.3. *[41] Let $C \subseteq \mathbb{R}^n$. Then*

$$C \text{ is convex} \iff C \text{ contains all convex combinations of its elements}.$$

Definition 1.4. Given a set $C \subseteq \mathbb{R}^n$, the convex hull of C, denoted by $\mathrm{conv}(C)$, is the set of all convex combinations of points in C, i.e.,

$$\mathrm{conv}(C) = \left\{ \sum_{i=1}^{k} \lambda_i x_i \ / \ x_i \in S,\ \lambda_i \geq 0,\ \sum_{i=1}^{k} \lambda_i = 1,\ k \in \mathbb{N} \right\},$$

and which is the smallest convex set that contains C.

Corollary 1.5. *Let $C \subseteq \mathbb{R}^n$. Then*

$$C \text{ is convex} \iff C = \mathrm{conv}(C).$$

1.1.2 Convex functions

Definition 1.6. Let $f : C \subseteq \mathbb{R}^n \to \mathbb{R}$, where C is a non-empty convex set. We define convexity of the function f as follows

- f is convex on C if for all $x, y \in C$ and $\lambda \in [0,1]$, we have

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

- f is strictly convex on C if for all $x \neq y \in C$ and $\lambda \in]0,1[$, we have

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

- f is strongly convex on C if there exists a constant $\alpha > 0$ such that for all $x, y \in C$,

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) - \frac{1}{2}\alpha\lambda(1-\lambda)\|x-y\|^2.$$

- If $-f$ is convex (strictly convex, strongly convex), then f is concave (strictly concave, strongly concave).

In general, it is often challenging to verify the convexity of a function using only its definition. The following propositions provide an easier way to prove the convexity of function.

Proposition 1.7. *[65] Let C be a convex subset of $\mathbb{R}^n$ and let $f : C \to \mathbb{R}$ be a differentiable function. The following statements hold:*

- *The function f is convex on C if and only if:*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle, \quad \forall x, y \in C.$$

  This is known as the first-order condition for convexity.

- *The function f is convex on C if and only if its gradient ∇f is monotone on C, i.e.,*

$$\langle \nabla f(y) - \nabla f(x), y - x \rangle \geq 0, \quad \forall x, y \in C.$$

- *The function f is strictly convex on C if and only if:*

$$f(y) > f(x) + \langle \nabla f(x), y - x \rangle, \quad \forall x, y \in C \text{ with } x \neq y.$$

- *The function f is strictly convex on C if and only if its gradient ∇f is strictly monotone on C, i.e.,*

$$\langle \nabla f(y) - \nabla f(x), y - x \rangle > 0, \quad \forall x, y \in C \text{ with } x \neq y.$$

Proposition 1.8. *[41] Let C be a convex subset of $\mathbb{R}^n$ and let $f : C \to \mathbb{R}$ be a twice continuously differentiable (C^2) function. The following statements are equivalent*

- *The function f is convex on C if and only if the Hessian matrix $\nabla^2 f(x)$ is positive semi-definite for all $x \in C$, i.e.,*

$$\langle \nabla^2 f(x)(y - x), y - x \rangle \geq 0, \quad \forall x, y \in C.$$

- *The function f is strictly convex on C if and only if the Hessian matrix $\nabla^2 f(x)$ is positive definite for all $x \in C$, i.e.,*

$$\langle \nabla^2 f(x)(y - x), y - x \rangle > 0, \quad \forall x, y \in C, \text{ with } x \neq y.$$

1.2 Unconstrained optimization: existence and optimalty conditions

Unconstrained optimization is a fundamental concept in mathematical optimization, where the goal consists of minimizing or maximizing a function which depends on a number of real variables without any constraint on the values of these variables.

Definition 1.9. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function. The unconstrained optimization problem is the minimization or maximization of f without any constraints on the variables. Mathematically, this can be represented as

$$\begin{cases} \min \ f(x) \\ x \in \mathbb{R}^n. \end{cases} \tag{1.1}$$

In general, we have two types of minimizers: local minimizer and global minimizer. In the following, we give their exact definitions.

Definition 1.10. A point $x^* \in \mathbb{R}^n$ is a global minimum of f over $\mathbb{R}^n$ if and only if

$$f(x^\star) \leq f(x) \quad \forall x \in \mathbb{R}^n.$$

Definition 1.11. A point $x^\star \in \mathbb{R}^n$ is a local minimum of f over $\mathbb{R}^n$ if and only if:

$$\exists r > 0 \text{ such that } f(x^\star) \leq f(x) \quad \forall x \in B(x^\star, r) = \{x \in \mathbb{R}^n / \, \|x - x^\star\| \leq r\},$$

where the condition $f(x^\star) \leq f(x)$ is satisfied only in the neighborhood of $x^\star$. In this case, $f(x^\star)$ is referred to as a local minimum value of f.

1.2.1 Results of existence and uniqueness of optimal solution

Before delving into the first and second-order optimality conditions for the problem (1.1), it is important to firstly verify the existence of a solution. Once this is established, we will proceed to discuss the results related to the uniqueness of the solution.

Theorem 1.12 (Weierstrass theorem [25]). *Let $f : D \to \mathbb{R}$ be a continuous function, where $D \subseteq \mathbb{R}^n$ is compact. Then f has a minimum on D, i.e., $\exists \, x^\star \in D$ such that*

$$f(x^\star) \leq f(x) \quad \textit{for all } x \in D.$$

Theorem 1.13 (Existence). *[41] Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuous and coercive function ($\lim_{\|x\|\to\infty} f(x) = +\infty$). Then, f attains its minimum on $\mathbb{R}^n$, i.e, $\exists\, x^\star \in \mathbb{R}^n$ such that*

$$f(x^\star) \leq f(x) \quad \textit{for all } x \in \mathbb{R}^n.$$

Corollary 1.14. *If $f : D \to \mathbb{R}$ is a continuous and coercive function on a closed set $D \subseteq \mathbb{R}^n$. Then, f has a minimum on D, i.e, $\exists\, x^\star \in D$ such that*

$$f(x^\star) \leq f(x) \quad \textit{for all } x \in D.$$

Theorem 1.15 (Uniqueness). *[41] If $f : \mathbb{R}^n \to \mathbb{R}$ is strictly convex, then there exists at most one $x^\star \in \mathbb{R}^n$ such that*

$$f(x^\star) \leq f(x) \quad \forall x \in \mathbb{R}^n.$$

Theorem 1.16 (Existence and Uniqueness). *[41] Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function satisfying:*

- ***Coercivity**: $\lim_{\|x\|\to\infty} f(x) = +\infty$*
- ***Continuity**: f is continuous on $\mathbb{R}^n$*
- ***Strict Convexity**:*

$$f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda) f(y)$$

for all $x \neq y \in \mathbb{R}^n$ and $\lambda \in (0,1)$

*Then f attains a **unique global minimum** on $\mathbb{R}^n$.*

1.2.2 First-order optimalty conditions

First-order necessary optimalty condition

Theorem 1.17. *[65] Let $D \subseteq \mathbb{R}^n$ be an open set and let $f : D \to \mathbb{R}$ be a continuously differentiable function. Suppose that $x^\star \in D$ is a local or global minimum of f. Then*

$$\nabla f(x^\star) = 0.$$

Remark 1.18.

1. A point $x^\star \in \mathbb{R}^n$ that satisfies $\nabla f(x^\star) = 0$ is referred to as a stationary point or critical point of f in $\mathbb{R}^n$ (a candidate for being a minimum or maximum). However, $\nabla f(x^\star) = 0$ does not necessarily imply that $x^\star$ is a minimum of f.
2. If $\nabla f(x^\star) \neq 0$, then $x^\star$ cannot be a minimum or maximum of f.

First-order sufficient and necessary optimalty condition

Theorem 1.19. *Let $D \subseteq \mathbb{R}^n$ be an open set and let $f : D \to \mathbb{R}$ be a differentiable function. Assume that f is convex on D. Then*

$$(x^\star \in D \text{ is a global minimum of } f) \iff (\nabla f(x^\star) = 0).$$

Proof. The first implication is evident (result from the previous theorem).
Now, suppose that $\nabla f(x^\star) = 0$. Since f is convex, we have for all $x \in D$

$$f(x) \geq f(x^\star) + \langle \nabla f(x^\star), x - x^\star \rangle.$$

Thus,

$$f(x) \geq f(x^\star).$$

This implies that $x^\star$ is a global minimum of f on $\mathbb{R}^n$. □

Remark 1.20. If f is convex (not necessarily differentiable), then the necessary and sufficient condition for $x^\star$ to be a global minimum is

$$x^\star \text{ minimizes } f \iff 0 \in \partial f(x^\star),$$

where $\partial f(x^\star)$ denotes the subdifferential of f at $x^\star$.

Recall that the set of all subgradients of f at x is denoted by

$$\partial f(x) = \left\{ g \in \mathbb{R}^n \,\middle|\, f(y) \geq f(x) + g^\top (y - x),\ \forall y \in \mathbb{R}^n \right\}.$$

1.2.3 Second-order optimalty conditions

Second-order necessary conditions

Theorem 1.21. *Let $f : D \to \mathbb{R}$ be a twice continuously differentiable function (i.e., $f \in \mathcal{C}^2(D)$) on an open set $D \subset \mathbb{R}^n$. If $x^\star \in D$ is a minimum (local or global) of f, then*

1. $\nabla f(x^\star) = 0$.
2. *The Hessian matrix $\nabla^2 f(x^\star)$ is positive semidefinite.*

Proof. Suppose that $f : D \to \mathbb{R}$ is twice continuously differentiable and that $x^\star \in D$ is a local minimizer of f. Suppose on the contrary, that the Hessian $\nabla^2 f(x^\star)$ is not positive semidefinite.

Then, there exists a direction $d \in \mathbb{R}^n$ such that:

$$d^\top \nabla^2 f(x^\star) d < 0.$$

Since $f \in \mathcal{C}^2$, the Hessian is continuous. Thus, there exists $r > 0$ such that for all μ small enough (with $0 < \mu < r$), we have:

$$x^\star + \mu d \in B(\mathbf{x}^\star, r) \subset D \quad \text{and} \quad d^\top \nabla^2 f(x^\star + \mu d) d < 0.$$

Using the second-order Taylor expansion of f at $x^\star$ and the fact that $\nabla f(x^\star) = 0$, we obtain

$$f(x^\star + \mu d) = f(x^\star) + \frac{1}{2}\mu^2 d^\top \nabla^2 f(x^\star + \theta\mu d) d,$$

for some $\theta \in (0, 1)$.

Since $d^\top \nabla^2 f(x^\star + \theta\mu d) d < 0$, it follows that

$$f(x^\star + \mu d) < f(x^\star),$$

which contradicts the assumption that $x^\star$ is a local minimizer of f. Hence, the Hessian $\nabla^2 f(x^\star)$ must be positive semidefinite. □

Second-order sufficient conditions

Theorem 1.22. *[65] Let $f : D \to \mathbb{R}$ be a twice continuously differentiable function on an open set $D \subset \mathbb{R}^n$. If $\nabla f(x^\star) = 0$ and the Hessian matrix $\nabla^2 f(x^\star)$ is positive definite, then $x^\star$ is a local minimizer of f.*

1.3 Methods of resolution for unconstrained optimization

In this section, we present some of the most significant unconstrained optimization methods that rely on gradient computation, emphasizing their definitions, strengths and weaknesses, as well as their convergence properties.

1.3.1 Principle of the descent methods

Descent methods are a fundamental class of algorithms used in optimization, particularly for solving unconstrained optimization problems (1.1). The main idea of these

methods is to find the minimum of a function by iteratively moving towards descent directions.

In practice, we look for d_k and α_k such that $f(x_k + \alpha_k d_k) \leq f(x_k)$ for all $k \geq 1$. In this way, we construct a sequence (x_k) starting from x_1 that converges to $x^\star$ (where $x^\star$ is a stationary point of f, i.e., $\nabla f(x^\star) = 0$) generated by the general iterative scheme of a descent method as follows

$$\begin{cases} x_1 \in \mathbb{R}^n \quad \text{(starting point)} \\ x_{k+1} = x_k + \alpha_k d_k \quad \text{for } k \geq 1, \end{cases}$$

where $d_k \in \mathbb{R}^n \setminus \{0\}$ is called the descent direction and $\alpha_k \in \mathbb{R}_+$ is the stepsize determined by a line search, which are chosen in a such a way that satisfying $f(x_k + \alpha_k d_k) \leq f(x_k)$.

Let us recall what is meant by a descent direction of f near a point x.

Definition 1.23. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function. A vector $d \in \mathbb{R}^n \setminus \{0\}$ is called a descent direction at a point $x \in \mathbb{R}^n$ if there exists a scalar $\alpha > 0$ such that

$$f(x + \alpha d) \leq f(x).$$

Proposition 1.24. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function of class $\mathcal{C}^1$. Then, a vector $d \in \mathbb{R}^n \setminus \{0\}$ is a descent direction of f in the neighborhood of a point x if*

$$\langle \nabla f(x), d \rangle < 0.$$

The algorithm for descent direction methods is described as follows.

Algorithm 1: Descent direction method

Step 1: Set $k = 1$; choose $x_1 \in \mathbb{R}^n$, $\epsilon > 0$ (desired precision).

Step 2: If $\|\nabla f(x_k)\| \leq \epsilon$, **Stop**; otherwise, go to **Step 3**.

Step 3: Compute $d_k \in \mathbb{R}^n$ and $\alpha_k > 0$ step size such that $f(x_k + \alpha_k d_k) \leq f(x_k)$.

Step 4: Update $x_{k+1} = x_k + \alpha_k d_k$.

Step 5: Let $k = k + 1$ and go back to **Step 2**.

The descent method is of significant importance in the field of optimization from a theoretical perspective. Its convergence theory is valuable not only in its own right but also as a foundation for understanding and developing other optimization methods. For further details, see [65].

Among the most well-known and widely used descent direction methods are the gradient method, the Newton's method and the conjugate gradient method.

1.3.2 Gradient method

When it comes to performing optimization tasks, gradient descent stands out as one of the most common methods to use. The main idea of the gradient method was introduced for the first time by Cauchy in 1847 [8]. He tried to use the first-order Taylor expansion of a function f between two successive iterates x_k and x_{k+1}

$$f(x_{k+1}) = f(x_k + \alpha_k d_k) = f(x_k) + \alpha_k \langle \nabla f(x_k), d_k \rangle + o(\alpha_k d_k),$$

where α_k is a positive stepsize and d_k is the chosen search direction.

To ensure that the function value decreases, i.e., $f(x_{k+1}) < f(x_k)$, a natural initial choice for the search direction is to take it as the negative gradient $d_k = -\nabla f(x_k)$. This direction corresponds to the steepest descent of f at the point x_k. Substituting this into the expansion yields

$$f(x_k + \alpha_k d_k) - f(x_k) = \alpha_k \langle \nabla f(x_k), -\nabla f(x_k) \rangle + o(\alpha_k d_k) = -\alpha_k \|\nabla f(x_k)\|^2 + o(\alpha_k d_k) < 0,$$

showing that the function value decreases along this direction, as long as the stepsize α_k is chosen appropriately.

Thus, the descent direction selected at each iteration is $d_k = -\nabla f(x_k)$ and the sequence of iterate points generated by this method is given by

$$\begin{cases} x_1 \in \mathbb{R}^n & \text{(initial point)} \\ x_{k+1} = x_k + \alpha_k d_k, & k \geq 1 \end{cases}$$

The stepsize $\alpha_k > 0$ may be constant or adaptively chosen to maximize progress towards the minimum.

The gradient method is also known as the steepest descent method [7].

The general algorithm of the gradient method is given as follows

Algorithm 2: Gradient method

Step 1: Set $k = 1$; choose an initial point $x_1 \in \mathbb{R}^n$ and tolerance $\epsilon > 0$.

Step 2: If $\|\nabla f(x_k)\| < \epsilon$, **Stop**; otherwise, proceed to **Step 3**.

Step 3: Compute the gradient $\nabla f(x_k)$ and set the descent direction $d_k = -\nabla f(x_k)$.

Step 4: Choose $\alpha_k > 0$ such that $f(x_k + \alpha_k d_k) < f(x_k)$.

Step 5: Compute $x_{k+1} = x_k + \alpha_k d_k$.

Step 6: Let $k = k + 1$ and go back to **Step 2**.

The choice of the stepsize α_k can be chosen in several ways, resulting in different types of gradient methods. This include a fixed stepsize and optimal stepsize for the gradient method, which we will discuss below.

Gradient method with fixed stepsize

The general idea behind the fixed stepsize gradient method is to start with an initial value and use a fixed stepsize, meaning we set $\alpha_k = \alpha > 0$ for all iterations k. This stepsize should be carefully chosen, it should not be too large, as that might cause the divergence of the algorithm, and it should not be too small, as that would unnecessarily slow down the convergence. In other words, it should be "just right" to ensure the algorithm steadily approaches the optimal solution without diverging. Selecting this stepsize properly is crucial, as it significantly impacts both the convergence speed and overall stability of the algorithm. The resulting algorithm can be described as follows

Algorithm 3: Fixed step gradient method

Step 1: Set $k = 1$; choose an initial point $x_1 \in \mathbb{R}^n$, and a fixed stepsize $\alpha > 0$, $\epsilon > 0$

Step 2: If $\|\nabla f(x_k)\| \leq \epsilon$, **Stop**; otherwise, proceed to **Step 3**.

Step 3: Compute $\nabla f(x_k)$ and set $d_k = -\nabla f(x_k)$.

Step 4: compute $x_{k+1} = x_k + \alpha d_k$.

Step 5: Let $k = k + 1$ and go back to **Step 2**.

Gradient method with optimal stepsize

In this approach, the stepsize α_k is chosen in an optimal manner at each iteration by solving the following one-dimensional problem

$$\alpha_k^\star = \arg\min_{\alpha>0} h(\alpha) = \arg\min_{\alpha>0} f(x_k + \alpha d_k),$$

where x_k is the current point, d_k is the descent direction and f is the objective function. The goal is to find the stepsize α_k that minimizes the function along the direction d_k. This approach is also referred to as an exact line search method. The algorithm is given as follows

Algorithm 4: Optimal stepsize gradient method

Step 1: Set $k = 1$; choose an initial point $x_1 \in \mathbb{R}^n$, $\epsilon > 0$.

Step 2: If $\|\nabla f(x_k)\| \leq \epsilon$, **Stop**; otherwise, proceed to **Step 3**.

Step 3: Compute $\nabla f(x_k)$ and set $d_k = -\nabla f(x_k)$.

Step 4: Determine $\alpha_k^\star$ such that $\alpha_k^\star = \arg\min_{\alpha>0} f(x_k + \alpha d_k)$.

Step 5: Compute $x_{k+1} = x_k + \alpha_k^\star d_k$.

Step 6: Let $k = k + 1$ and back to **Step 2**.

Theorem 1.25. *[6] Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function ($f \in \mathbb{R}^n$) that is coercive and strictly convex. Suppose that there exists a constant $L > 0$ such that the gradient of f is Lipschitz continuous, i.e.,*

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \quad \forall x, y \in \mathbb{R}^n.$$

Then, if the stepsize α_k is chosen from a fixed interval $[\rho_1, \rho_2]$, with $0 < \rho_1 < \rho_2 < \frac{2}{L}$, the gradient method converges to the unique minimizer of f.

1.3.3 Newton's method

The Newton's method is a powerful technique in optimization, relying on the quadratic approximation of the objective function f and the exact minimization of this approximation. In the neighborhood a current point x_k, the function f can be approximated using Taylor development as follows

$$f(x) \approx f(x_k) + \nabla f(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T\nabla^2 f(x_k)(x - x_k), \tag{1.2}$$

which represents the local quadratic approximation function of f around x_k. To find the search direction for the Newton method, we minimize the right-hand of (1.2), resulting in the calculation of d_k as the solution of the system

$$H_k d_k = -g_k, \tag{1.3}$$

where $g_k = \nabla f(x_k)$ and $H_k = \nabla^2(f(x_k))$.

Consequently, the Newton's method can be expressed as

$$x_{k+1} = x_k + d_k, \quad k = 1, 2, \ldots \tag{1.4}$$

Here, the stepsize $\alpha_k = 1$ for all k . It is important to note that d_k serves as a descent direction if and only if the Hessian matrix $H_k = \nabla^2 f(x_k)$ is positive definite, ensuring that the quadratic model is convex in the vicinity of x_k.

Algorithm 5: Newton's method

Step 1: Set $k = 1$; choose an initial point x_1 in a neighborhood of $x^\star$, and set $\epsilon > 0$ (desired precision).

Step 2: Compute $g_1 = \nabla f(x_1)$

Step 3: If $\|\nabla f(x_k)\| < \epsilon$, **Stop**; otherwise, proceed to **Step 3**.

Step 4: Compute d_k as the solution of the system $H_k d_k = -\nabla f(x_k)$.

Step 5: Compute $x_{k+1} = x_k + d_k$.

Step 5: Let $k = k + 1$ and go back to **Step 3**.

For the convergence of Newton's method, we have the following theorem

Theorem 1.26. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a C^2 function and let $x^\star$ be a local minimum of f. Additionally, assume that the Hessian matrix $H(x^\star)$ is positive definite. Then*

1. *There exists a neighborhood $\mathcal{N}^\star$ of $x^\star$ such that if $x_1 \in \mathcal{N}^\star$, the sequence of iterates (x_k), generated from x_1 by Newton's algorithm, converges to $x^\star$.*

2. *The convergence is quadratic.*

Remark 1.27. The main advantage of Newton's method is its high speed and simplicity, making it easy to implement, in condition, that the starting point x_1 belongs to the neighborhood of $x^\star$.

1.4 Conjugate gradient method

The conjugate gradient method was originally introduced by Hestenes and Stiefel in 1952 [30] for solving linear systems, which is equivalent to minimizing a quadratic function through an iterative scheme. Since then, the method has been extended and generalized to minimize non-quadratic functions.

This method addresses the orthogonality limitations of traditional gradient methods by using conjugate (non-orthogonal) directions, making it a compromise between the gradient method and Newton's method. This contributes to reaching the solution more efficiently and in fewer iterations.

Definition 1.28. Let $A \in \mathbb{M}_n(\mathbb{R})$ be a symmetric, positive-definite matrix.

1. Two vectors d_1 and d_2 in $\mathbb{R}^n \setminus \{0\}$ are said to be **A**-conjugate (or conjugate with respect to A) if
$$\langle Ad_1, d_2\rangle = (d_2)^T Ad_1 = 0.$$
2. A set of vectors $\{d_1, d_2, \ldots, d_k\} \subset \mathbb{R}^n \setminus \{0\}$ is said to be A-conjugate if
$$(d_j)^T Ad_i = \langle Ad_i, d_j\rangle = 0 \quad \forall i \neq j, \quad 1 \leq i \leq k, \quad 1 \leq j \leq k.$$

This implies that two **A**-conjugate vectors are orthogonal with respect to the inner product associated with matrix A, defined as

$$\langle x, y\rangle_A = y^T Ax, \quad \forall x, y \in \mathbb{R}^n.$$

Proposition 1.29. *Let $A \in \mathbb{M}_n(\mathbb{R})$ be a symmetric, positive-definite matrix and let $\{d_1, d_2, \ldots, d_k\}$ be a set of vectors in $\mathbb{R}^n \setminus \{0\}$. Then*

1. *If the set $\{d_1, d_2, \ldots, d_k\}$ is **A**-conjugate, then it is linearly independent in $\mathbb{R}^n$.*
2. *If the set $\{d_1, d_2, \ldots, d_n\}$ (where $k = n$) is **A**-conjugate, then it forms a basis of $\mathbb{R}^n$.*

We consider the unconstrained optimization problem (1.1).

The iterative scheme of the conjugate gradient method is as follows

$$\begin{cases} x_1 \in \mathbb{R}^n \quad \text{(initial point)} \\ x_{k+1} = x_k + \alpha_k d_k, \quad k \geq 1, \end{cases}$$

where

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

and $g_k = \nabla f(x_k)$.

The stepsize α_k is calculated using either an exact or inexact line search and the parameter $\beta_k \in \mathbb{R}$ is called the conjugacy parameter and can take different values, which imparts different properties and names to the algorithm.

The conjugate gradient method was initially designed for unconstrained minimization of quadratic functions and later extended to the minimization of non-quadratic convex functions.

Linear case (quadratic)

The conjugate gradient method is a powerful iterative technique for solving linear systems that arise in quadratic optimization problems of the form

$$\min_x f(x) = \frac{1}{2} x^T A x - b^T x,$$

where $A \in \mathbb{R}^{n \times n}$ is a symmetric positive-definite matrix and $b \in \mathbb{R}^n$ is a given vector.

The algorithm begins with an initial point x_1 and generates a sequence of approximations x_k that converge to the exact solution $x^\star$. The iterative update is defined as

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

where α_k is the optimal stepsize calculated by exact line search, given by

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

where $g_k = Ax_k - b$. At each step k, the direction d_k is obtained as a linear combination of the gradient at x_k and the previous direction d_{k-1}

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

The coefficients β_k are chosen such that d_{k-1} is conjugate to all previous directions, which means

$$d_k^T A d_{k-1} = 0.$$

From this, we deduce

$$d_k^T A d_{k-1} = 0 \implies (-g_k + \beta_k d_{k-1})^T A d_{k-1} = 0$$

$$\implies -g_k^T A d_{k-1} + \beta_k d_{k-1}^T A d_{k-1} = 0$$

$$\implies \beta_k = \frac{g_k^T A d_{k-1}}{d_{k-1}^T A d_{k-1}}.$$

Theorem 1.30. *[54] At an arbitrary iteration k of the algorithm where the optimum of $f(x)$ has not yet been reached (i.e., $g_i \neq 0$ for $i = 1, \ldots, k$), we have*

$$\alpha_k = \frac{\|g_k\|^2}{d_k^T A d_k},$$

$$\beta_k = \frac{g_{k+1}^T (g_{k+1} - g_k)}{g_k^T g_k} = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k} = \frac{\|g_{k+1}\|^2}{\|g_k\|^2}.$$

Algorithm 6: Conjugate gradient algorithm for the quadratic case

Step 1: Set $k = 1$; choose an initial point $x_1 \in \mathbb{R}^n$ and set $\epsilon > 0$ (a given precision). Calculate $g_1 = Ax_1 - b$ and set $d_1 = -g_1$.

Step 2: If $\|g_k\| < \epsilon$, **Stop**; otherwise, proceed to **Step 3**.

Step 3: Calculate $\alpha_k = \frac{\|g_k\|^2}{d_k^T A d_k}$.

Step 4: Set $x_{k+1} = x_k + \alpha_k d_k$.

Step 5: Calculate $g_{k+1} = Ax_{k+1} - b$ and $\beta_k = \frac{\|g_{k+1}\|^2}{\|g_k\|^2}$.

Step 6: Calculate $d_{k+1} = -g_{k+1} + \beta_k d_k$.

Step 7: Let $k = k + 1$ and go back to **Step 2**.

Theorem 1.31. *[65] (Convergence of the conjugate gradient algorithm for a quadratic function) Let $f(x) = \frac{1}{2} x^T A x - b^T x$, where $x \in \mathbb{R}^n$, $A \in M_n(\mathbb{R})$ is a symmetric positive definite matrix and $b \in \mathbb{R}^n$. Then, the conjugate gradient method finds the minimum of the function f in at most n iterations, where n is the order of the matrix A, for all starting point (global convergence).*

Non linear case (non-quadratic)

Consider the unconstrained optimization problem (1.1).

In 1964, Fletcher and Reeves [23] extended the linear conjugate gradient method to general functions, followed by the method of Polak, Ribière, and Polyak (PRP) in

1969 [58, 57]. Since then, various variants have been proposed to the present day. These methods generate a sequence $\{x_k\}$ as follows

$$\begin{cases} x_1 \in \mathbb{R}^n & \text{(initial point)} \\ x_{k+1} = x_k + \alpha_k d_k, & k \geq 1 \end{cases}$$

where

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

with $g_k = \nabla f(x_k)$ and $\beta_k \in \mathbb{R}$ represents the conjugate gradient parameter that varies depending on the specific method used. The stepsize $\alpha_k \in \mathbb{R}$ is determined by an exact or inexact line search, such as those of Armijo, Goldstein or Wolfe, etc.

The different values attributed to β_k define the various nonlinear conjugate gradient methods. The formulas for β_k in different conjugate gradient method variants are as follows, where $y_k = g_{k+1} - g_k$, $s_k = x_{k+1} - x_k$ and θ_k, $\lambda \in [0, 1]$.

Table 1.1: Formulas for different conjugate gradient parameters β_k

Method	**Formula for β_k**	**Dates**
Hestenes-Stiefel (HS)	$\beta_k^{HS} = \frac{g_k^T y_{k-1}}{d_{k-1}^T y_{k-1}}$	1952
Fletcher-Reeves (FR)	$\beta_k^{FR} = \frac{\|g_k\|^2}{\|g_{k-1}\|^2}$	1964
Daniel	$\beta_k = \frac{g_k^T \nabla^2 f(x_k) d_k}{d_k^T \nabla^2 f(x_k) d_k}$	1967
Polak-Ribière-Polyak (PRP)	$\beta_k^{PRP} = \frac{g_k^T y_{k-1}}{\|g_{k-1}\|^2}$	1969
Conjugate Descent (CD)	$\beta_k^{CD} = -\frac{\|g_k\|^2}{d_{k-1}^T g_{k-1}}$	1987
Liu-Storey (LS)	$\beta_k^{LS} = -\frac{g_k^T y_{k-1}}{d_{k-1}^T g_{k-1}}$	1991
Dai-Yuan (DY)	$\beta_k^{DY} = \frac{\|g_k\|^2}{d_{k-1}^T y_{k-1}}$	1999
Dai and Liao (DL)	$\beta_k^{DL} = \frac{g_{k+1}^T y_k}{d_k^T y_k} - t\frac{g_{k+1}^T s_k}{d_k^T y_k}$	2001
Hager-Zhang (HZ)	$\beta_k^{HZ} = \frac{(y_{k-1} - 2d_{k-1}\|y_{k-1}\|^2 / d_{k-1}^T y_{k-1})^T g_k}{d_{k-1}^T y_{k-1}}$	2005
Wei	$\beta_k = \frac{g_k^T (g_k - \|g_k\|/\|g_{k-1}\| g_{k-1})}{\|g_{k-1}\|^2}$	2006
Andrei (PRPDY)	$\beta_k^{PRPDY} = (1-\theta_k)\beta_k^{PRP} + \theta_k \beta_k^{DY}$	2007
Fan et al. (MN)	$\beta_k^{MN} = \frac{\|g_k\|^2 - \|g_k\|\|g_{k-1}\| g_k^T g_{k-1}}{\|\langle g_k, d_{k-1}\rangle\| + \|g_{k-1}\|^2}$	2011
Rivaie et al. (RMIL)	$\beta_k^{RMIL} = \frac{g_k^T y_{k-1}}{\|d_{k-1}\|^2}$	2012
Sellami et al.	$\beta_k = \frac{(1-\tau_k)\|g_k\|^2 + \tau_k(-g_k^T d_{k-1})}{(1-\tau_k-\lambda_k)\|g_{k-1}\|^2 + (\tau_k+\lambda_k)(-g_{k-1}^T d_{k-1})}$	2013
Zhang and Zheng (DHSDL)	$\beta_k^{DHSDL} = \frac{\|g_k\|^2 - \|g_k\|\|g_{k-1}\| g_k^T g_{k-1}}{\lambda\|g_k^T d_{k-1}\| + d_{k-1}^T y_{k-1} - t g_k^T s_{k-1}}$	2017
Zhang and Zheng (DLSDL)	$\beta_k^{DLSDL} = \frac{\|g_k\|^2 - \|g_k\|\|g_{k-1}\| g_k^T g_{k-1}}{\lambda\|g_k^T d_{k-1}\| - d_{k-1}^T g_{k-1} - t g_k^T s_{k-1}}$	2017
Djordjevic (LSFR)	$\beta_k^{LSFR} = (1-\theta_k)\beta_k^{LS} + \theta_k \beta_k^{FR}$	2019
Ayidne et al. (DHSAYO)	$\beta_k^{\text{DHSAYO}} = \frac{\|g_k\|^2 - \frac{\|g_k\|}{\|g_{k-1}\|}\|d_{k-1}^T g_k\|}{\mu\|d_{k-1}^T g_k\| + y_{k-1}^T d_{k-1}} + \frac{t\, s_{k-1}^T g_k}{d_{k-1}^T g_{k-1}}$	2023
Amna et al. (WFR)	$\beta_k^{WFR} = \frac{\|g_k\|^2}{\|g_{k-1}\|\,\|d_{k-1}\|}$	2024
Nazzal et al. (SFA)	$\beta_k^{\text{SFA}} = -\frac{g_k^T\left(g_k - \frac{\|g_k\|}{\|g_{k-1}\|} g_{k-1}\right)}{d_{k-1}^T g_{k-1}}$	2024
Khelladi and Benterki (KB)	$\beta_k^{KB} = \frac{g_k^T y_{k-1}}{\lambda\|g_{k-1}\|^2 + (1-\lambda)\|d_{k-1}\|^2}$	2024
Ouaoua et al. (OKB)	$\beta_k^{OKB} = \frac{\|g_k\|^2 - \frac{\|g_k\|}{\|d_{k-1}\|}\left\|g_k^T d_{k-1}\right\|}{d_{k-1}^T y_{k-1}}$	2025

Algorithm 7: Conjugate gradient algorithm for the nonlinear case

Step 1: Set $k = 1$; choose an initial point $x_1 \in \mathbb{R}^n$ and set $\epsilon > 0$ (a given precision). Compute $g_1 = \nabla f(x_1)$.

Step 2: If $\|g_k\| < \epsilon$, **Stop**; otherwise, proceed to **Step 3**.

Step 3: Compute α_k using exact or inexact line search.

Step 4: Set $x_{k+1} = x_k + \alpha_k d_k$.

Step 5: Compute $g_{k+1} = \nabla f(x_{k+1})$ and β_k.

Step 6: Compute $d_{k+1} = -g_{k+1} + \beta_k d_k$.

Step 7: Let $k = k + 1$ and go back to **Step 2**.

1.5 Convergence results of the conjugate gradient method

1.5.1 Conditions C1 and C2 and Zoutendijk's theorem

Condition C1: Lipschitz continuity and boundedness

(a) The function f is lower bounded on the set $\mathcal{S} = \{x \in \mathbb{R}^n : f(x) \geq f(x_1)\}$, where x_1 is the starting point.

(b) The function f is differentiable in a neighborhood $\mathcal{N}$ of x_1 and its gradient is continuously Lipschitz continuous. That is, there exists a constant $L > 0$ such that

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \quad \text{for all} \quad x, y \in \mathcal{N}.$$

Condition C2: Bounded Level Set

The level set $\mathcal{S} = \{x \in \mathbb{R}^n : f(x) \leq f(x_1)\}$ is bounded. That is, there exists a constant $B < 1$ such that

$$\|x\| \leq B \quad \text{for all} \quad x \in \mathcal{S}.$$

Zoutendijk's theorem

Zoutendijk's theorem [73] is the main result used to prove the convergence of conjugate gradient methods for non-quadratic optimization problems.

This theorem is particularly effective when using inexact Wolfe and Goldstein line searches and requires certain assumptions on f (such as f being differentiable and its gradient ∇f being Lipschitz continuous).

Let θ_k denote the angle between the descent direction d_k and the gradient $-g_k$, i.e.,

$$\theta_k = \widehat{(d_k, -g_k)}.$$

Using the inner product $\langle x, y\rangle$ of two vectors x and y, we have

$$\langle x, y\rangle = \|x\|\|y\| \cos \widehat{(x, y)},$$

which leads to

$$\langle -g_k, d_k\rangle = -g_k^T d_k = \|g_k\|\|d_k\| \cos(\theta_k),$$

and consequently

$$\cos(\theta_k) = \frac{-g_k^T d_k}{\|g_k\|\|d_k\|}.$$

Theorem 1.32. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a lower-bounded and differentiable function whose gradient is Lipschitz continuous (i.e., it satisfies Condition C1). Consider an iterative algorithm that generates a sequence $\{x_k\}_{k\in\mathbb{N}}$ in $\mathbb{R}^n$, defined by*

$$x_1 \in \mathbb{R}^n, \quad x_{k+1} = x_k + \alpha_k d_k, \quad \alpha_k > 0,$$

where d_k is a descent direction (i.e., $\nabla f(x_k)^T d_k < 0$) and α_k satisfies the Wolfe conditions. Then, the following holds

$$\sum_{k=1}^{\infty} \|g_k\|^2 \cos^2(\theta_k) < \infty.$$

1.5.2 Zoutendijk's theorem and global convergence

The condition

$$\sum_{k=0}^{\infty} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty$$

is equivalent to

$$\sum_{k=1}^{\infty} \|g_k\|^2 \cos^2(\theta_k) < \infty,$$

where θ_k is the angle between the direction d_k and the gradient $-g_k$.

It is clear that if

$$\cos(\theta_k) \geq \epsilon > 0 \quad \text{for all } k \text{ and} \epsilon \text{ a constant},$$

then, we have

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

In the different conjugate gradient methods it is difficult to prove the previous result, but only the following

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{1.5}$$

Condition (1.5) implies that there exists a subsequence of $\{g_k\}$ that tends to zero.

Definition 1.33 (Sufficient descent direction)**.** We say that d_k is a sufficient descent direction if

$$g_k^T d_k \leq -C\|g_k\|^2, \quad \text{where } C > 0.$$

To prove (1.5), we associate the following two hypotheses with Zoutendijk's theorem

- **Hypothesis 1:** Sufficient descent is ensured, i.e., $g_k^T d_k \leq -C\|g_k\|^2$, where $C > 0$.
- **Hypothesis 2:** There exists a constant $\omega > 0$ such that

$$\|d_k\|^2 \leq \omega\|g_k\|^2.$$

Thus, we can give the sufficient conditions associated with Zoutendijk's theorem to prove global convergence.

Theorem 1.34. *[73] If the following conditions are satisfied*

$$\begin{cases} \textit{Zoutendijk's Condition:} & \sum_{k=0}^{\infty} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty, \\ \textit{Hypothesis 1:} & g_k^T d_k \leq -C\|g_k\|^2, \\ \textit{Hypothesis 2:} & \|d_k\|^2 \leq \omega\|g_k\|^2, \end{cases}$$

then

$$\liminf_{k\to\infty} \|g_k\| = 0.$$

1.6 Line search methods

Performing a line search consists of finding a step-size α_k along a descent direction d_k such that the objective function decreases, i.e., $f(x_k + \alpha_k d_k) \leq f(x_k)$.

Exact line search and inexact or economical line search are the two primary categories of line search techniques.

1.6.1 Exact line search

Let $\varphi(\alpha) = f(x_k + \alpha d_k)$.

Thus, the problem involves finding a stepsize in the direction d_k such that

$$\varphi(\alpha_k) = \min_{\alpha>0} \varphi(\alpha),$$

then this is called an exact line search or optimal line search and α_k is referred to as the optimal stepsize.

1.6.2 Inexact line search

In iteration k, we have the iterate $x_k \in \mathbb{R}^n$ as well as the corresponding search direction $d_k \in \mathbb{R}^n$ that is a descent direction for our objective $f : \mathbb{R}^n \to \mathbb{R}$ such that $\nabla f(x_k)^T d_k < 0$.

The aim is to substantially decrease the value of the objective by a displacement α_k in the direction of d_k. A number of rules have been created for this purpose by like the rules of **Armijo**, **Goldstein**, **Wolfe**, **backtracking**, etc. In the following, we will present the used main rules.

Armijo rule

The Armijo rule ensures that the objective function f decreases sufficiently along the search direction. It requires that f decreases by at least a fraction $\rho \in (0, 1)$ of the decrease predicted by the linear approximation of f at x_k. This condition is written as

$$f(x_k + \alpha d_k) \leq f(x_k) + \rho\alpha \nabla f(x_k)^\top d_k.$$

Here, ρ is a parameter chosen in the interval $(0, 1)$ that controls the strictness of the condition. This rule guarantees that the stepsize α produces a meaningful reduction in the function value.

In terms of the auxiliary function $\varphi_k(\alpha) = f(x_k + \alpha d_k)$, the Armijo condition can be expressed as

$$\varphi(\alpha) \leq \varphi(0) + \rho\varphi'(0)\alpha.$$

If this last inequality holds, α is accepted as a valid stepsize. Otherwise, α is considered too large and should be reduced.

Wolfe rule

The conditions of the Goldstein rule may exclude a minimum, which can be a drawback. The Wolfe conditions address this issue. The weak Wolfe conditions for $\alpha > 0$ are

$$f(x_k + \alpha d_k) \leq f(x_k) + \rho\alpha\nabla f(x_k)^T d_k, \tag{1.6}$$

$$\nabla f(x_k + \alpha d_k)^T d_k \geq \sigma\nabla f(x_k)^T d_k, \tag{1.7}$$

where ρ and σ are constants chosen such that $0 < \rho < \sigma < 1$.

The stepsize α satisfies the Wolfe conditions if

- $\varphi(\alpha) \leq \varphi_k(0) + \rho\varphi'(0)\alpha$ and $\varphi'(\alpha) \geq \sigma\varphi'(0)$, then α is acceptable.
- If $\varphi(\alpha) > \varphi(0) + \rho\varphi'(0)\alpha$, then α is too large.
- If $\varphi'(\alpha) < \sigma\varphi'(0)$, then α is too small.

The Wolfe rule requires the computation of $\varphi'(\alpha)$, making it theoretically more expensive than the Goldstein rule. However, in many applications, computing the gradient $\nabla f(x)$ adds little cost compared to evaluating $f(x)$. As a result, the Wolfe rule is widely used.

For certain algorithms, such as nonlinear conjugate gradient methods, a strong condition than (1.7) may be necessary. In this case, (1.7) is replaced with

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

The strong Wolfe conditions are therefore

$$f(x_k + \alpha d_k) \leq f(x_k) + \rho\alpha\nabla f(x_k)^T d_k, \tag{1.8}$$

$$\nabla f(x_k + \alpha d_k)^T d_k \leq -\sigma\alpha\nabla f(x_k)^T d_k, \tag{1.9}$$

where $0 < \rho < \sigma < 1$.

The strong Wolfe conditions imply the weak Wolfe conditions. Specifically, conditions (1.6) and (1.8) are equivalent, and (1.7) and (1.9) are related as

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

Thus, $\sigma\nabla f(x_k)^T d_k \leq \nabla f(x_k + \alpha d_k)^T d_k$, satisfying the weak Wolfe conditions.

Backtracking

In its most basic form, backtracking proceeds as follows

Algorithm 8: Backtracking algorithm

Step 0: Set the objective function $f(x)$; a search direction d_k at point x_k; choose parameters $0 < \rho < 1$ and $\mu \in (0, 1)$.

Step 1: Set $\alpha = 1$.

Step 2: While $f(x_k + \alpha d_k) > f(x_k) + \rho\alpha\nabla f(x_k)^T d_k$, update $\alpha = \mu\alpha$.

Step 3: Set $\alpha_k = \alpha$.

Note that the notion of strong Wolfe rule plays a crucial role in the proof of the global convergence of all algorithms proposed in this thesis.

CHAPTER 2

New hybrid conjugate gradient methods for nonlinear unconstrained optimization based on a convex combinations

In this chapter, we propose three new hybrid conjugate gradient methods for solving unconstrained optimization problems. The first method called RMILHS integrates two well-known conjugate gradient parameters, β_k^{RMIL} and β_k^{HS}, through a convex combination, using parameter θ_k which is calculated to align the search direction more closely with the Newton direction. The second variant named RMILFR combines β_k^{RMIL} and β_k^{FR}. The third method extends this idea further by merging three popular conjugate gradient parameters β_k^{RMIL}, β_k^{LS}, and β_k^{CD} through RMILCDLS . This multi-parameter strategy aims to construct a more robust and effective search direction for optimization. The sufficient descent direction along with the global convergence of the algorithms are ensured by employing the strong Wolfe conditions.

Numerical experiments show that all three versions perform better than existing methods in terms of the number of iterations, computation time, and the number of function and gradient evaluations. Additionally, we test our method RMILHS on image restoration problems. The results show that it performs well compared to current techniques, especially when the noise image is high. The results of the RMILHS method were published in the **Kybernetika Journal** [31]. Meanwhile, the results of the RMILFR have been submitted to **Filomat Journal**, currently in revision. As for the third method

RMILCDLS it is published in the journal of **Nonlinear Dynamics and Systems Theory** [43].

2.1 Introduction

The conjugate gradient method remains one of the most widely used iterative techniques for solving linear systems and nonlinear optimization problems due to its efficiency, numerical stability and versatility across different domains of science and engineering. Here, we focus on solving unconstrained optimization problems of the form

$$\begin{cases} \min f(x), \\ x \in \mathbb{R}^n, \end{cases} \tag{2.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously and differentiable function.

Conjugate gradient methods are particularly useful for solving large-scale problems like (2.1). The standard conjugate gradient method updates the solution iteratively as follows

$$x_{k+1} = x_k + \alpha_k d_k, \tag{2.2}$$

where x_k is the current point, $\alpha_k > 0$ is the stepsize found using a line search and d_k is the search direction defined by

$$d_0 = -g_0, \quad d_{k+1} = -g_{k+1} + \beta_k d_k, \quad \text{for } k \geq 0, \tag{2.3}$$

where $g_k = \nabla f(x_k)$ is the gradient of f at x_k, and β_k is a scalar parameter that determines the search direction. In recent years, researchers have continued to explore and refine the conjugate gradient method, proposing new β_k to address specific problem characteristics and computational challenges.

Some of the well known parameters β_k are those of Hesteness-Steifel (HS) [30], Fletcher and Reeves (FR) [23], Polak-Ribière-Polyak (PRP) [57, 58], Conjugate Descent (CD) [22], Liu-Storey (LS) [44], Dai-Yuan (DY) [13, 14] and Rivaie-Mustafa-Ismail-Leong (RMIL) [60]. Hager and Zhang [29] gave a good survey of nonlinear conjugate gradient methods in [29].

The formulas of the β_k mentioned above are

$$\beta_k^{HS} = \frac{g_{k+1}^T y_k}{d_k^T y_k}, \ \beta_k^{FR} = \frac{\|g_{k+1}\|^2}{\|g_k\|^2}, \ \beta_k^{PRP} = \frac{g_{k+1}^T y_k}{\|g_k\|^2}, \ \beta_k^{CD} = -\frac{\|g_{k+1}\|^2}{g_k^T d_k},$$

$$\beta_k^{LS} = -\frac{g_{k+1}^T y_k}{g_k^T d_k},\ \beta_k^{DY} = \frac{\|g_{k+1}\|^2}{d_k^T y_k},\ \beta_k^{RMIL} = \frac{g_{k+1}^T y_k}{\|d_k\|^2}.$$

Where $y_k = g_{k+1} - g_k$ and $\|.\|$ denotes the Euclidean norm. Many researchers proposed new families and combinations of the conjugate gradient methods, specifically the hybrid methods. The idea of the hybrid methods is to combine the standard conjugate gradient methods and exploit the attractive features of each of one and to avoid the jamming phenomenon. This combination can be convex or non-convex. Among these hybrid methods, we can cite those proposed by Dalladji et al. [16] and Mtagulwa and Kaelo [52], Djordjevic [19, 18], Ben Hanachi et al. [6], Yang et al. [70], Rivaie et al. [61] and the families of conjugate methods proposed by Sellami and Chaib [62, 63].

Recently, a great contribution in the area of conjugate gradient methods and its application has been done by Andrei in [2, 3].

2.2 New hybrid RMILHS conjugate gradient method applied to image restoration problems

Building on the above ideas, we propose a new hybrid parameter β_k^{RMILHS}, which combines β_k^{RMIL} and β_k^{HS} using a convex combination

$$\beta_k^{RMILHS} = (1-\theta_k)\beta_k^{RMIL} + \theta_k \beta_k^{HS}, \tag{2.4}$$

where θ_k is a scalar parameter between 0 and 1. The search direction is then updated as

$$d_{k+1} = -g_{k+1} + \beta_k^{RMILHS} d_k. \tag{2.5}$$

The stepsize α_k is computed using the strong Wolfe conditions

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \rho \alpha_k g_k^T d_k, \tag{2.6}$$

$$|g_{k+1}^T d_k| \leq -\sigma g_k^T d_k, \tag{2.7}$$

where $0 < \rho < \sigma < \frac{1}{2}$.

The parameter θ_k is chosen to align the search direction with the Newton direction. Assuming the Hessian $\nabla^2 f(x_{k+1})$ is invertible, we set θ_k such that

$$-\nabla^2 f(x_{k+1})^{-1} g_{k+1} = d_{k+1} = -g_{k+1} + \beta_k^{RMILHS} d_k. \tag{2.8}$$

After some algebraic manipulations, we define θ_k as

$$\theta_k = \begin{cases} \theta_k^{NT} & \text{if } 0 < \theta_k^{NT} < 1, \\ 0 & \text{if } \theta_k^{NT} \leq 0, \\ 1 & \text{if } \theta_k^{NT} \geq 1, \end{cases} \tag{2.9}$$

where

$$\theta_k^{NT} = \frac{-s_k^T g_{k+1} + y_k^T g_{k+1} - \beta_k^{RMIL} y_k^T d_k}{(\beta_k^{RMIL} - \beta_k^{HS}) y_k^T d_k}. \tag{2.10}$$

The algorithm for our hybrid conjugate gradient method is as follows

Algorithm 9: RMILHS

Begin algorithm

Step 0: Given a starting point x_0 and a parameter $\varepsilon > 0$.

Step 1: Set $k = 0$ and compute $d_0 = -g_0$.

Step 2: If $\|g_k\| \leq \varepsilon$, **Stop**; else go to **Step 3**.

Step 3: Find the stepsize $\alpha_k \in (0, 1]$ using strong Wolfe conditions (2.6) and (2.7).

Step 4: Compute $x_{k+1} = x_k + \alpha_k d_k$.

Step 5: Compute $g_{k+1} = \nabla f(x_{k+1})$, $y_k = g_{k+1} - g_k$, and $s_k = x_{k+1} - x_k$.

Step 6: Compute $\theta_k = \theta_k^{NT}$ using (2.10) and (2.9).

Step 7: Compute $\beta_k = \beta_k^{RMILHS} = (1 - \theta_k)\beta_k^{RMIL} + \theta_k \beta_k^{HS}$ using (2.4).

Step 8: If $| g_{k+1}^T g_k | \geq 0.2 \|g_{k+1}\|$ (restart criterion of Powell [59])

then set $d_{k+1} = -g_{k+1}$.

else compute $d_{k+1} = -g_{k+1} + \beta_k^{RMILHS} d_k$.

End if

Set the initial guess $\alpha_k = 1$.

Step 9: Let $k = k + 1$ and go to **Step 2**.

End algorithm

Remark 2.1. The Powell's restart criterion in Step 8 of the algorithm 9, serves as a safeguard to ensure that the search direction remains a true descent direction. Specifically, it evaluates the degree of alignment between the current and previous gradients. If the inner product between g_{k+1} and g_k isn't small (i.e., the two gradients are too closely aligned), this may indicate that the direction d_k has lost its conjugacy properties. In such cases, continuing along the current conjugate gradient direction may lead to poor progress or a divergence to the algorithm.

To address this, the algorithm resets the search direction to the steepest descent direction by setting $d_{k+1} = -g_{k+1}$ and $\alpha_{k+1} = 1$. This restart helps recover global convergence properties by restoring descent.

Importantly, the convergence analysis of the algorithm relies on this criterion to guarantee that the search directions maintain a sufficient descent condition. Without such a condition, our conjugate gradient method may fail to satisfy necessary conditions for convergence in general nonlinear cases.

2.2.1 Convergence analysis

Throughout this section, we make the following assumptions

(i) The level set $\mathcal{L} = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ is bounded, which means that there exists a constant $M < \infty$, such that

$$\|x\| \leq M, \text{ for all } x \in \mathcal{L}.$$

(ii) In a neighborhood $\mathcal{N}$ of $\mathcal{L}$, the function f is continuously differentiable and its gradient $\nabla f(x)$ is Lipschitz continuous, it means that exists $0 < L < \infty$ such that

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \text{ for all } x, y \in \mathcal{N}. \tag{2.11}$$

Under these assumptions, there exists a constant $\mu \geq 0$, where

$$\|\nabla f(x)\| \leq \mu, \text{ for all } x \in \mathcal{L}. \tag{2.12}$$

To establish the sufficient descent condition, we introduce the following theorem.

Theorem 2.2. *Let the sequences $\{g_k\}$ and $\{d_k\}$ be generated by the $RMILHS$ algorithm. Then, the search direction satisfies the sufficient descent condition*

$$g_k^T d_k \leq -c\|g_k\|^2, \textit{ for all } k. \tag{2.13}$$

Proof. From the $RMILHS$ algorithm, we know that if the restart criterion of Powell holds, then $d_k = -g_k$ and (2.13) holds.

So, we assume that Powell criterion doesn't hold. Then, we have

$$\mid g_{k+1}^T g_k \mid < 0.2\|g_{k+1}\|^2. \tag{2.14}$$

The following proof is by induction.

If $k = 0$, then $g_0^T d_0 = -\|g_0\|^2$, so (2.13) holds.

Next, for $k > 0$ we have

$$d_{k+1} = -g_{k+1} + \beta_k^{RMILHS} d_k,$$

which can be written as

$$\begin{aligned} d_{k+1} &= -(\theta_k g_{k+1} + (1-\theta_k) g_{k+1}) + ((1-\theta_k)\beta_k^{RMIL} + \theta_k \beta_k^{HS}) d_k \\ &= \theta_k(-g_{k+1} + \beta_k^{HS} d_k) + (1-\theta_k)(g_{k+1} + \beta_k^{RMIL} d_k) \end{aligned}$$

It follows that:

$$d_{k+1} = \theta_k d_{k+1}^{HS} + (1-\theta_k) d_{k+1}^{RMIL}. \tag{2.15}$$

By multiplying (2.15) by g_{k+1}^T from the left side, we get

$$g_{k+1}^T d_{k+1} = \theta_k g_{k+1}^T d_{k+1}^{HS} + (1-\theta_k) g_{k+1}^T d_{k+1}^{RMIL}. \tag{2.16}$$

Firstly, if $\theta_k = 0$, d_k coincides with descent direction of Rivaie et al., $d_{k+1}^{RMIL} = -g_{k+1} + \beta_k^{RMIL} d_k$, where they proved in [61] that

$$g_{k+1}^T d_{k+1}^{RMIL} \leq -c_1 \|g_{k+1}\|^2, \text{ for all } k. \tag{2.17}$$

Now, if $\theta_k = 1$, d_k coincides with descent direction of Hesteness-Steifel, $d_{k+1}^{HS} = -g_{k+1} + \beta_k^{HS} d_k$, where Djordjevic proved in [18] that

$$g_{k+1}^T d_{k+1}^{HS} \leq -c_2 \|g_{k+1}\|^2, \tag{2.18}$$

with

$$c_2 = \frac{1-(2.2)\sigma}{1-\sigma}, \text{ and } \sigma < \frac{5}{11}.$$

Finally, if $0 < \theta_k < 1$, then there exist scalars λ_1 and λ_2, such that

$$0 < \lambda_1 \leq \theta_k \leq \lambda_2 < 1. \tag{2.19}$$

From the formula (2.16) and using (2.19), we conclude

$$g_{k+1}^T d_{k+1} \leq \lambda_1 g_{k+1}^T d_{k+1}^{HS} + (1-\lambda_2) g_{k+1}^T d_{k+1}^{RMIL}. \tag{2.20}$$

Let $c = \lambda_1 c_2 + (1-\lambda_2) c_1$, then from (2.17), (2.18) and (2.20) we finally get

$$g_{k+1}^T d_{k+1} \leq -c\|g_{k+1}\|^2, \; c > 0.$$

□

Lemma 2.3. *Suppose that assumptions (i) and (ii) hold. Consider common iterate (2.2), where d_k is a descent direction (2.5) and α_k is determined by the strong Wolfe line search (2.6) and (2.7). Then, the Zoutendijk condition*

$$\sum_{k\geq 0} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty, \tag{2.21}$$

holds.

Proof. The proof follows directly from [73]. □

Theorem 2.4. *Consider the $RMILHS$ conjugate gradient method and suppose that assumptions (i), (ii) and (2.13) hold. Then either $g_k = 0$ for some k, or*

$$\lim_{k\to\infty} \inf \|g_k\| = 0. \tag{2.22}$$

Proof. Suppose that $g_k \neq 0$, for all k. Then we have to prove (2.22).

Suppose, on the contrary, that (2.22) doesn't hold. Then there exists a constant $t > 0$, such that

$$\|g_k\| \geq t, \text{ for all } k. \tag{2.23}$$

Let D be the diameter of the level set $\mathcal{L}$.

From the formula of β_k^{RMILHS} we get

$$\left|\beta_k^{RMILHS}\right| \leq \left|\beta_k^{RMIL}\right| + \left|\beta_k^{HS}\right| = \frac{\left|g_{k+1}^T y_k\right|}{\|d_k\|^2} + \left|\frac{g_{k+1}^T y_k}{d_k^T y_k}\right|. \tag{2.24}$$

Further, using the second strong Wolfe condition (2.7), we get

$$y_k^T d_k = g_{k+1}^T d_k - g_k^T d_k \geq (\sigma - 1) g_k^T d_k = -(1-\sigma) g_k^T d_k > 0, \tag{2.25}$$

which gives

$$\frac{1}{y_k^T d_k} \leq \frac{1}{-(1-\sigma) g_k^T d_k}. \tag{2.26}$$

The direction d_k satisfies (2.13), so it holds

$$-g_k^T d_k \geq c\|g_k\|^2,$$

which implies using (2.23) that

$$\frac{-1}{g_k^T d_k} \leq \frac{1}{c\|g_k\|^2} \leq \frac{1}{ct^2}. \tag{2.27}$$

So, from (2.27) the formula (2.26) satisfies

$$\frac{1}{y_k^T d_k} \leq \frac{1}{-(1-\sigma)g_k^T d_k} \leq \frac{1}{(1-\sigma)ct^2}. \tag{2.28}$$

Now, using (2.11) and (2.12), we get

$$\left|g_{k+1}^T y_k\right| \leq \|g_{k+1}\| \, \|g_{k+1} - g_k\| \leq \mu L \|x_{k+1} - x_k\| \leq \mu L \|s_k\| \leq \mu L D. \tag{2.29}$$

Then, from (2.28) and (2.29), we have

$$\beta_k^{HS} = \frac{g_{k+1}^T y_k}{y_k^T d_k} \leq \frac{\mu L D}{(1-\sigma)ct^2}. \tag{2.30}$$

On the one hand and using (2.11), we have

$$y_k^T d_k \leq \|y_k\| \, \|d_k\| \leq L \, \|s_k\| \, \|d_k\| = L\alpha_k \|d_k\|^2,$$

then

$$\|d_k\|^2 \geq \frac{1}{L\alpha_k} y_k^T d_k. \tag{2.31}$$

So, using (2.25) and (2.13) the formula (2.31) becomes

$$\|d_k\|^2 \geq -\frac{1}{L\alpha_k}(1-\sigma)g_k^T d_k \geq \frac{1}{L\alpha_k}(1-\sigma)c\|g_k\|^2.$$

Using (2.23), we get

$$\frac{1}{\|d_k\|^2} \leq \frac{L\alpha_k}{(1-\sigma)c\|g_k\|^2} \leq \frac{L\alpha_k}{(1-\sigma)ct^2}. \tag{2.32}$$

From (2.29), (2.32) and the formula of β_k^{RMIL}, we obtain

$$\left|\beta_k^{RMIL}\right| = \frac{\left|g_{k+1}^T y_k\right|}{\|d_k\|^2} \leq \frac{\mu L^2 D}{(1-\sigma)ct^2}\alpha_k. \tag{2.33}$$

Now, from (2.24), (2.30) and (2.33) we get

$$\mid \beta_k^{RMILHS} \mid \leq \frac{\mu L D}{(1-\sigma)ct^2}(L\alpha_k + 1). \tag{2.34}$$

Next, we are going to prove as in [19] that there exists $\alpha_* > 0$, such that

$$\alpha_k \geq \alpha_* > 0, \text{ for all } k.$$

Suppose, on the contrary, that there isn't an α_*, such that $\alpha_k \geq \alpha_* > 0$.

Then there exists an infinite subsequence $\alpha_k = \gamma^{j_k}$, $k \in K_1$ such that

$$\lim_{k \in K_1} \alpha_k = 0.$$

Then

$$\lim_{k\in K_1} \gamma^{j_k-1} = 0, \tag{2.35}$$

i.e.,

$$\lim_{k\in K_1} j_k - 1 = \infty.$$

Now, from (2.6) we get

$$f(x_k + \gamma^{j_k} d_k) - f(x_k) \leq \delta\gamma^{j_k} g_k^T d_k, \tag{2.36}$$

$$f(x_k + \gamma^{j_k-1} d_k) - f(x_k) > \delta\gamma^{j_k-1} g_k^T d_k, \tag{2.37}$$

where $\delta < 1$. From (2.37), we have

$$\frac{f(x_k + \gamma^{j_k-1} d_k) - f(x_k)}{\gamma^{j_k-1}} > \delta g_k^T d_k. \tag{2.38}$$

Using (2.35) and passing to the limit of (2.38), we conclude that

$$g_k^T d_k \geq \delta g_k^T d_k. \tag{2.39}$$

But, our method satisfies the sufficient descent, so $g_k^T d_k \leq 0$.

Also, $0 < \delta < 1$, so, the relation (2.39), is correct only if $g_k^T d_k = 0$. Then, from the second strong Wolfe condition (2.7), we get that $g_{k+1}^T d_k = 0$, which correspond to the exact line search. So, we have a contradiction.

Now,we can write

$$\|d_{k+1}\| \leq \|g_{k+1}\| + \mid \beta_k^{RMILHS} \mid \|d_k\|. \tag{2.40}$$

As $s_k = \alpha_k d_k$, we write $d_k = \dfrac{s_k}{\alpha_k}$. So, from (2.12) and (2.77), we have

$$\|d_{k+1}\| \leq \mu + \frac{\mu LD}{(1-\sigma)ct^2}(L\alpha_k + 1)\frac{\|s_k\|}{\alpha_k}.$$

Since $\alpha_k \geq \alpha_*$ and $\|s_k\| \leq D$, it follows that

$$\|d_{k+1}\| \leq \mu + \frac{\mu LD^2(L\alpha_* + 1)}{\alpha_*(1-\sigma)ct^2} = Cste, \tag{2.41}$$

which gives

$$\sum_{k\geq 1} \frac{1}{\|d_k\|^2} = \infty. \tag{2.42}$$

On the other hand, from (2.13), (2.23) and from the Zoutendijk condition (2.22), it results that

$$c^2 t^4 \sum_{k\geq 0} \frac{1}{\|d_k\|^2} \leq \sum_{k\geq 0} \frac{c^2\|g_k\|^4}{\|d_k\|^2} \leq \sum_{k\geq 0} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty,$$

which contradicts (2.42). Therefore, (2.23) doesn't hold.

Then, $\lim_{k\to\infty} \inf \|g_k\| = 0$. This completes the proof. □

2.2.2 Numerical experiments

In this section, we will study the effectiveness of our algorithm ($RMILHS$) on two parts. In this first part, we consider an unconstrained optimization problem of the form (1.1). To show the effectiveness and to measure the convergence behavior of our algorithm ($RMILHS$), we applied it on a set of test functions taken from [1] in comparing with some existing methods, namely $HSFR$ method [18], $LSCD$ method [19] and $LSDY$ method [70]. We take several dimensions, from $n = 10$ to $n = 10000$ and we considered the precision $\varepsilon = 10^{-6}$.

We used Matlab language on HP laptop with Intel(R) Core(TM) i5-6300U CPU @ 2.40 GHz processor and 8GB RAM memory and Windows 8.1.

We applied here the iterations number, CPU time and the gradient evaluations. The performance profile of Dolan and Moré [20] offers a systematic means to evaluate and compare the performance of a set of solvers S on a set P of problems. Assuming n_p problems and n_s solvers, for each problem p and solver s, they define $t_{p,s}$ the computing time required to solve problem p by solver s. A need of reference point for comparisons. They evaluate the performance of solver s on problem p against the optimal performance achieved by any solver on the same problem using the performance ratio

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in S\}}.$$

They suppose that a parameter $r_M \geq r_{p,s}$ for all chosen p, s, and $r_{p,s} = r_M$ if and only if solver s does not solve problem p. Define

$$\rho_s(\tau) = \frac{1}{n_p} size\{p \in P : \log_2 r_{p,s} \leq \tau\},$$

where $\rho_s(\tau)$ is the probability for solver $s \in S$ that the performance ratio $r_{p,s}$ is within a factor $\tau \in \mathbb{R}$ of the best possible ratio. The function ρ_s is the (cumulative) distribution function for the performance ratio. The value of $\rho_s(0)$ is the probability that the solver

will win over the rest of the solvers. Table 2.1 represents several test functions that are used in this experiments for different dimensions and different choices of the initial points.

Test function	Dimension	Initial point
Raydan 1	$10, 100, 500, 1000, 2000$	$(1, ..., 1)^T$
Raydan 2	$10, 100, 500, 1000, 3000$	$(4, ..., 4)^T$
Diagonal 4	$10, 100, 500, 1000, 10000$	$(1, ..., 1)^T$
Extended Woods	$10, 100, 500, 1000, 10000$	$(2, ..., 2)^T$
HIMMELBG	$10, 100, 500, 1000, 10000$	$(1.5, ..., 1.5)^T$
Extended Block-Diagonal 1	$10, 100, 500, 1000, 10000$	$(1, ..., 1)^T$
Prod1	$10, 100, 500, 1000, 10000$	$(1, ..., 1)^T$
Extended Maratos	$10, 100, 500, 1000, 10000$	$(0.1, ..., 0.1)^T$
Pertubed Quadratic	$10, 100, 500, 1000, 10000$	$(0.5, ..., 0.5)^T$
Extended White and Holst	$10, 100, 500, 1000, 10000$	$(1.5, ..., 1.5)^T$
Diagonal 2	$10, 100, 500, 1000, 10000$	$(1, \frac{1}{2}, ..., \frac{1}{n})^T$
POWER	$10, 100, 500, 1000, 10000$	$(1, ..., 1)^T$
Hager	$10, 100, 500, 1000, 5000$	$(1, ..., 1)^T$
DENSCHNA	$10, 100, 500, 1000, 10000$	$(1, ..., 1)^T$
HIMMELBC	$10, 100, 500, 1000, 10000$	$(1, ..., 1)^T$
Extended TET	$10, 100, 500, 1000, 2000$	$(0.1, ..., 0.1)^T$
Extended Cliff	$10, 100, 500, 1000, 5000$	$(1, ..., 1)^T$
Tridia	$10, 100, 500, 1000, 10000$	$(\frac{1}{n}, ..., \frac{1}{n})^T$
Diagonal 5	$10, 100, 500, 1000, 3000$	$(1.1, ..., 1.1)^T$
ARWHEAD	$10, 100, 500$	$(1, ..., 1)^T$
QUARTC	$10, 100, 500, 1000, 10000$	$(2, ..., 2)^T$

Table 2.1: The test functions and their dimensions with the initial points.

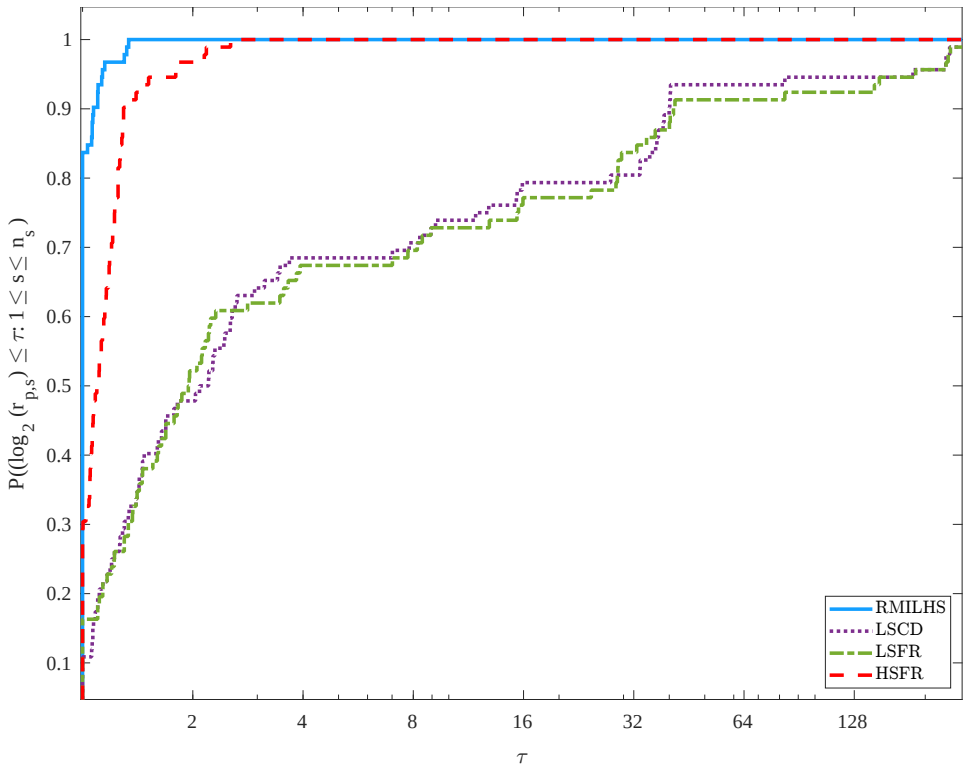


Figure 2.1: Performance profile based on the iteration number.

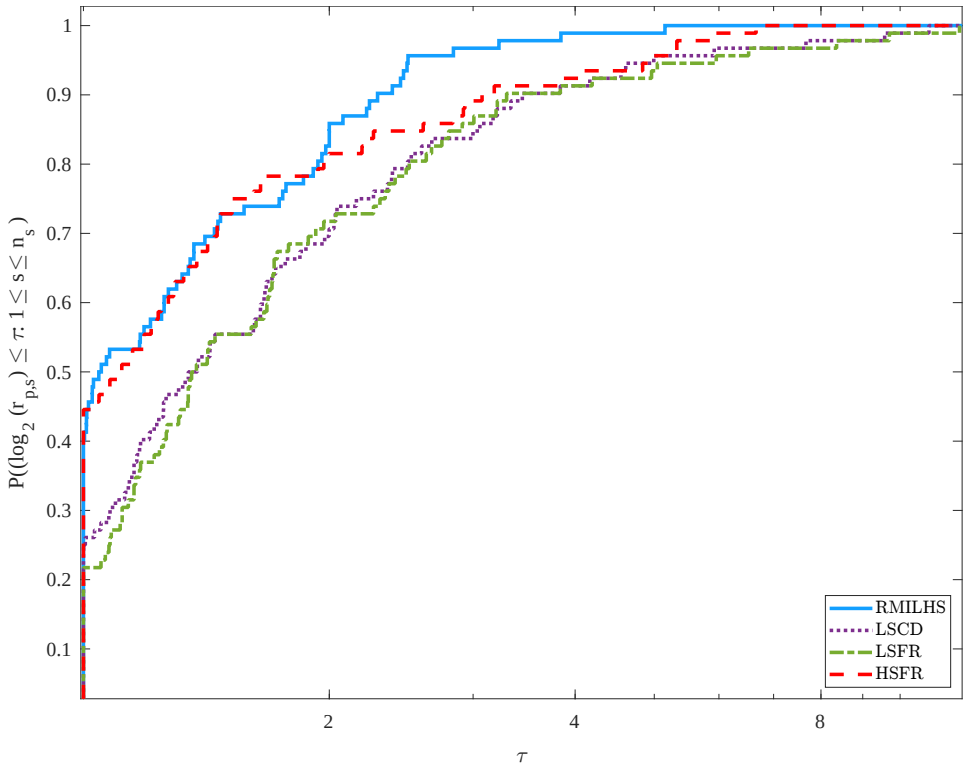


Figure 2.3: Performance profile based on gradient evaluations.

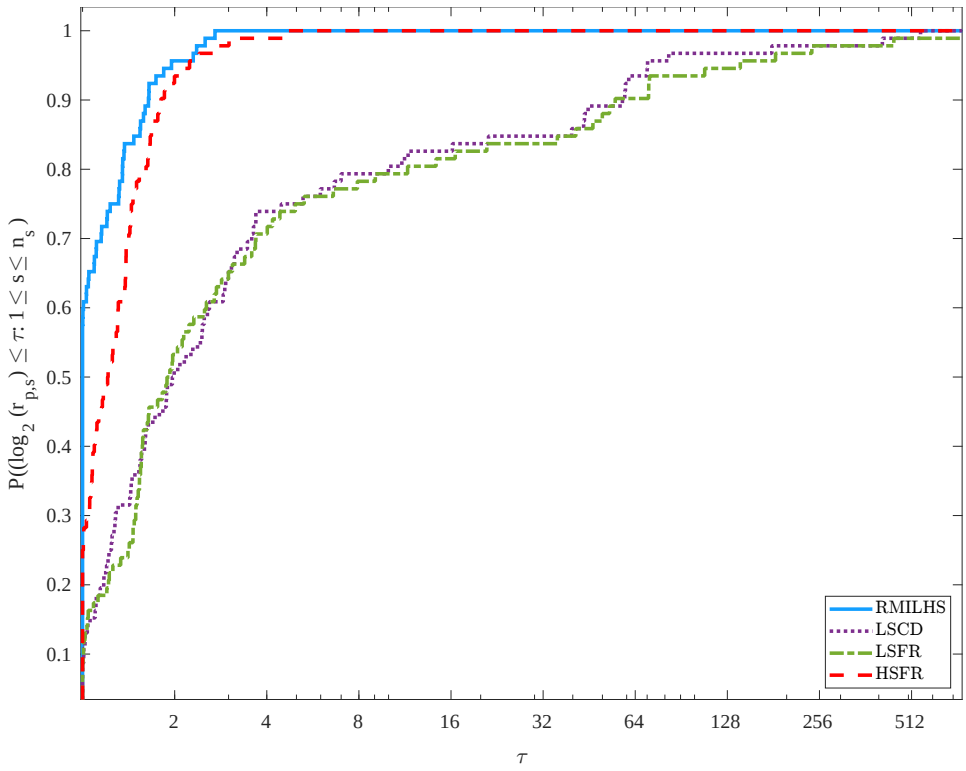


Figure 2.2: Performance profile based on the CPU time.

In the second part, we applied our algorithm to image restoration problems with the use of the two-phase scheme.

As known, images can get corrupted due to various factors including noise during acquisition. In this part, we are going to deal with restoration of an image corrupted by impulse noise problems. Impulse noise is one of the most common noise models, where only a portion of the pixels is contaminated by the noise and the information on the true values of these pixels is completely lost. Chan et al. [11] have applied the two-phase scheme to restore a corrupted image. Several researchers have used these two phases to make conjugate gradient algorithms capable of restoring images corrupted by impulse noise, even though the noise ratio is high or even reaches 90% [39, 48, 49].

The two-phase scheme applied can be briefly described as follows. In the first phase, we use adaptive median filter to detect noisy pixels. In the second phase the noise from the corrupted pixels is removed by solving the following smooth problem (2.43) proposed by Cai et al.[9]. We used our proposed $RMILHS$ algorithm to solve the problem (2.43), in comparison with $HSFR$, $LSCD$ and $LSFR$ methods.

Let X be an image of size M-by-N and $\mathcal{A} = \{1, 2, ..., M\} \times \{1, 2, ..., N\}$ be the index

set of the image X. We minimize $F_\alpha(u)$ where

$$F_\alpha(u) = \sum_{(i,j)\in\mathcal{B}} \left(2 \sum_{(m,n)\in\mathcal{V}_{i,j}\setminus\mathcal{B}} \varphi_\alpha(u_{i,j} - y_{m,n}) + \sum_{(m,n)\in\mathcal{V}_{i,j}\cap\mathcal{B}} \varphi_\alpha(u_{i,j} - u_{m,n}) \right), \tag{2.43}$$

in which $\mathcal{B} \subset \mathcal{A}$, the set of indices of the noise pixels detected from the first phase and η its number of elements. $\mathcal{V}_{i,j} = \{(i, j-1), (i, j+1), (i-1, j), (i+1, j)\}$ is the set of the four closest neighbors for the pixel at pixel location, for all $(i, j) \in \mathcal{A}$, and $y_{i,j}$ be the observed pixel value of the image at pixel location (i, j). Also $\varphi_\alpha(t)$ is an edge-preserving functional which is chosen as $\varphi_\alpha(t) = \sqrt{t^2 + \alpha}$, in our tests we set $\alpha = 100$. We use the peak signal to noise ratio (PSNR), defined by

$$PSNR = 10 \log_{10} \frac{255^2}{\frac{1}{MN} \sum_{i,j} \left(x^r_{i,j} - x^*_{i,j} \right)^2},$$

where $x^r_{i,j}$ and $x^*_{i,j}$ denote the pixel values of the restored image and the original one, respectively. We tested Lena (512×512), Man (512×512) and Pepper (512×512). We also set the stopping criteria as

$$Itr > 300 \text{ or } \frac{|F_\alpha(u_k) - F_\alpha(u_{k-1})|}{F_\alpha(u_k)} \leq 10^{-4}.$$

The noise levels of the salt-and-pepper noise used are as follows: 30%, 50%, 70%, and 90%.

In the following figures (Fig. 4 and Fig. 5), we present the most significant results of the noisy images which correspond to 70% and 90%.

Figure 2.4: The original images, the noisy images with 70% salt-and-pepper noise and the restored images by RMILHS , HSFR, LSCD and LSFR.

Figure 2.5: The original images, the noisy images with 90% salt-and-pepper noise and the restored images by RMILHS , HSFR, LSCD and LSFR.

Images	Methods	RMILHS			HSFR			LSCD			LSFR		
	Ratio	ITER	CPU	PSNR	ITER	CPU	PSNR	ITER	CPU	PSNR	ITER	CPU	PSNR
Lena	30%	21	11.34	38.07	21	11.09	38.07	19	10.42	38.06	19	10.41	38.06
	50%	24	18.23	35.53	24	18.18	35.53	22	16.82	35.50	22	16.79	35.50
	70%	29	27.02	32.22	29	27.10	32.22	26	23.00	32.21	26	22.87	32.21
	90%	34	44.07	27.21	34	43.88	27.21	44	57.73	27.07	44	57.73	27.07
Man	30%	22	16.25	34.46	22	14.64	34.46	19	10.43	34.43	19	10.49	34.43
	50%	23	18.66	32.02	23	18.49	32.02	22	16.88	32.00	22	16.87	32.00
	70%	29	29.53	29.04	29	27.53	29.04	27	24.79	29.06	27	22.57	29.06
	90%	39	49.07	25.02	39	49.58	25.02	49	58.29	24.96	49	58.49	24.96
Peppers	30%	23	16.28	34.08	23	16.25	34.08	19	13.11	34.15	19	12.96	34.15
	50%	25	19.64	32.37	25	19.92	32.37	23	20.28	32.33	23	18.32	32.33
	70%	30	32.79	29.93	30	31.83	29.93	29	33.16	29.91	29	31.12	29.91
	90%	37	51.58	25.96	37	51.70	25.96	51	70.91	25.77	51	71.58	25.77

Table 2.2: Numerical results for image restoration problem.

The obtained results of the Table 2 are converted into a bar chart so that the subtle differences can be seen better.

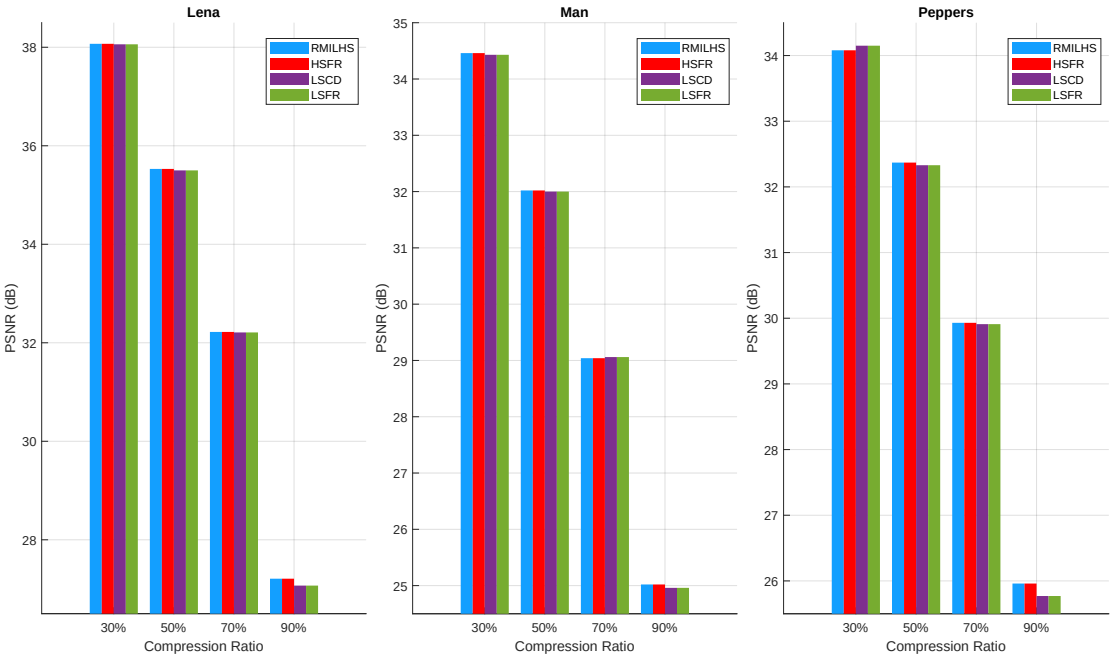


Figure 2.6: PSNR comparison for different methods and images.

Comments

Concerning the first part, based on the performance profiles depicted in figures 2.1, 2.2 and 2.3 for the number of iterations, CPU time and gradient evaluations, respectively,

we can say that our new approach employing β_k^{RMILHS} is more efficient than the other considered conjugate gradient methods.

For the second part, the numerical results cited in table 2.2, figures 2.4, 2.5 and 2.6 show the superiority of the LSCD and LSFR approaches compared to our RMILHS approach. We observe either an equality or a slight increase in the number of iterations or computation time in RMILHS compared to LSCD, LSFR, and HSFR. On the other hand, our approach RMILHS and also HSFR offer a better PSNR compared to LSCD and LSFR.

2.3 New hybrid RMILFR conjugate gradient method

In this section, based on the same idea that presented in the previous sections, we propose a new formula of β_k, which is a convex combination of β_k^{RMIL} and β_k^{FR}, defined by

$$\beta_k^{RMILFR} = (1-\theta_k)\beta_k^{RMIL} + \theta_k\beta_k^{FR}, \tag{2.44}$$

where $\theta_k \in [0,1]$ is a scalar parameter.

So, we have

$$d_{k+1} = -g_{k+1} + \beta_k^{RMILFR} d_k, \text{ for } k \geq 0. \tag{2.45}$$

In this study, to show the behavior of our algorithm, we use the strong Wolfe line search, i.e., we find the stepsize α_k using the following conditions

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \delta\alpha_k g_k^T d_k, \tag{2.46}$$

$$\left|g_{k+1}^T d_k\right| \leq -\sigma g_k^T d_k, \tag{2.47}$$

where $0 < \delta < \sigma < \frac{1}{2}$.

It is clear that, if $\theta_k = 0$, then $\beta_k^{RMILFR} = \beta_k^{RMIL}$, and if $\theta_k = 1$, then $\beta_k^{RMILFR} = \beta_k^{FR}$.

On the other side, if $0 < \theta_k < 1$, then β_k^{RMILFR} is proper convex combination of β_k^{RMIL} and β_k^{FR}.

Considering the formulas of $\beta_k^{RMIL}$and β_k^{FR}, the relation (2.44) becomes

$$\beta_k^{RMILFR} = (1-\theta_k)\frac{g_{k+1}^T y_k}{\|d_k\|^2} + \theta_k\frac{\|g_{k+1}\|^2}{\|g_k\|^2}. \tag{2.48}$$

Hence, the relation (2.45) becomes

$$d_{k+1} = -g_{k+1} + (1-\theta_k)\frac{g_{k+1}^T y_k}{\|d_k\|^2} d_k + \theta_k \frac{\|g_{k+1}\|^2}{\|g_k\|^2} d_k. \tag{2.49}$$

As known, if the point x_{k+1} is sufficiently close to a local minimizer x^*, then the best direction to follow is the Newton direction.

We choose the parameter θ_k to ensure that the search direction d_{k+1} aligns with the Newton direction.

So, assuming that the inverse of the hessian $\nabla^2 f(x)$ exists at each iterative point for the objective function f, we will choose the parameter θ_k in way that the search direction d_{k+1}, defined by (2.45), satisfies the Newton condition:

$$-\nabla^2 f(x_{k+1})^{-1} g_{k+1} = d_{k+1},$$

which means

$$-\nabla^2 f(x_{k+1})^{-1} g_{k+1} = -g_{k+1} + \beta_k^{RMILFR} d_k. \tag{2.50}$$

where $\nabla^2 f(x_{k+1})^{-1}$ is the inverse of the hessian matrix of f at the point x_{k+1} and $g_{k+1} = \nabla f(x_{k+1})$ is the gradient of f at x_{k+1}.

Let $s_k = x_{k+1} - x_k$.

Using the formula of β_k^{RMILFR} and multiplying (2.50) by $s_k^T \nabla^2 f(x_{k+1})$ from the left, we get :

$$\begin{aligned} -s_k^T g_{k+1} &= -s_k^T \nabla^2 f(x_{k+1}) g_{k+1} + (1-\theta_k)\beta_k^{RMIL} s_k^T \nabla^2 f(x_{k+1}) d_k \\ &\quad + \theta_k \beta_k^{FR} s_k^T \nabla^2 f(x_{k+1}) d_k. \end{aligned}$$

Further, we use the secant condition $\nabla^2 f(x_{k+1}) s_k = y_k$, we get:

$$-s_k^T g_{k+1} = -y_k^T g_{k+1} + (1-\theta_k)\beta_k^{RMIL} y_k^T d_k + \theta_k \beta_k^{FR} y_k^T d_k. \tag{2.51}$$

From (2.51) with some simple algebraic manipulations, we get a vlaue for θ_k denoted by:

$$\theta_k^{NT} = \frac{s_k^T g_{k+1} - y_k^T g_{k+1} + \beta_k^{RMIL} y_k^T d_k}{(\beta_k^{RMIL} - \beta_k^{FR}) y_k^T d_k}. \tag{2.52}$$

The parameter θ_k might be outside the interval [0, 1], a simple procedure is followed to ensure a convex combination:

$$\theta_k = \begin{cases} \theta_k^{NT} & \text{if } 0 < \theta_k^{NT} < 1, \\ 0 & \text{if } \theta_k^{NT} < 0, \\ 1 & \text{if } \theta_k^{NT} > 1. \end{cases} \tag{2.53}$$

The algorithm corresponding to our β_k is constructed as follows.

Algorithm 10: RMILFR

Begin algorithm

Step 0: Given a starting point x_0 and a parameter $\varepsilon > 0$.

Step 1: Set $k = 0$ and compute $d_0 = -g_0$.

Step 2: If $\|g_k\| \leq \varepsilon$, **Stop**; else go to **Step 3**.

Step 3: Find the stepsize $\alpha_k \in (0, 1]$ using strong Wolfe conditions (2.46) and (2.47).

Step 4: Compute $x_{k+1} = x_k + \alpha_k d_k$.

Step 5: Compute $g_{k+1} = \nabla f(x_{k+1})$, and $y_k = g_{k+1} - g_k$.

Step 6: Compute $\theta_k = \theta_k^{NT}$ using (2.52) and (2.53).

Step 7: Compute $\beta_k = \beta_k^{RMILFR} = (1 - \theta_k)\beta_k^{RMIL} + \theta_k \beta_k^{FR}$ using (2.48).

Step 8: If $\mid g_{k+1}^T g_k \mid \geq 0.2\|g_{k+1}\|$ (restart criterion of Powell [59])

then set $d_{k+1} = -g_{k+1}$.

else compute $d_{k+1} = -g_{k+1} + \beta_k^{RMILFR} d_k$.

End if

Set the initial guess $\alpha_k = 1$.

Step 9: Let $k = k + 1$ and go to **Step 2**.

End algorithm

2.3.1 Convergence analysis

Throughout this section, we make the assumption that:

- *(i)* The level set $\mathcal{L} = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ is bounded, which means that: $\exists M < \infty$ a constant, such that

$$\|x\| \leq M, \text{ for all } x \in \mathcal{L}.$$

- *(ii)* In a neighborhood $\mathcal{N}$ of $\mathcal{L}$ the function f is continuously differentiable and its gradient $\nabla f(x)$ is Lipschitz continuous, it means: $\exists 0 < L < \infty$ such that

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \text{ for all } x, y \in \mathcal{N}. \tag{2.54}$$

Under these assumptions, there exists a constant $\mu \geq 0$, where

$$\|\nabla f(x)\| \leq \mu, \text{ for all } x \in \mathcal{L}. \tag{2.55}$$

We introduce the following theorem to establish the sufficient descent condition.

Theorem 2.5. *Let $\{g_k\}$ and $\{d_k\}$ be the sequences generated by $RMILFR$ method. Then the search direction satisfies the sufficient descent condition*

$$g_k^T d_k \leq -c\|g_k\|^2, \text{ for all } k. \tag{2.56}$$

Where $c > 0$.

Proof. From the $RMILFR$ algorithm, we know that if the restart criterion of Powell holds, then $d_k = -g_k$ and (2.56) holds.

So, we assume that Powell criterion doesn't hold. Then we have

$$\mid g_{k+1}^T g_k \mid < 0.2\|g_{k+1}\|^2. \tag{2.57}$$

For $k = 0$, we have $g_0^T d_0 = -\|g_0\|^2$, so (2.56) holds.

Next it holds that

$$d_{k+1} = -g_{k+1} + \beta_k^{RMILFR} d_k.$$

We can write:

$$d_{k+1} = -(\theta_k g_{k+1} + (1 - \theta_k)g_{k+1}) + ((1 - \theta_k)\beta_k^{RMIL} + \theta\beta_k^{FR})d_k.$$

It follows that:

$$d_{k+1} = \theta_k(-g_{k+1} + \beta_k^{FR} d_k) + (1 - \theta_k)(-g_{k+1} + \beta_k^{RMIL} d_k),$$

i.e.,

$$d_{k+1} = \theta_k d_{k+1}^{FR} + (1 - \theta_k) d_{k+1}^{RMIL}. \tag{2.58}$$

Multiplying (2.58) by g_{k+1}^T from the left, we get

$$g_{k+1}^T d_{k+1} = \theta_k g_{k+1}^T d_{k+1}^{FR} + (1 - \theta_k) g_{k+1}^T d_{k+1}^{RMIL}. \tag{2.59}$$

Firstly, if $\theta_k = 0$, d_k coincides with descent direction of Rivaie et al., $d_{k+1}^{RMIL} = -g_{k+1} + \beta_k^{RMIL} d_k$, where they proved in [61] that

$$g_k^T d_k^{RMIL} \leq -c_1 \|g_k\|^2, \text{ for all } k. \tag{2.60}$$

Secondly, if $\theta_k = 1$, then $d_{k+1} = d_{k+1}^{FR}$. Let's emphasize that the sufficient descent condition is satisfied for the FR method when the strong Wolfe conditions are present and this was mentioned in [29]. So there exits a constant $c_2 > 0$, such that

$$g_{k+1}^T d_{k+1} \leq -c_2 \|g_{k+1}\|^2. \tag{2.61}$$

Now suppose that $0 < \theta_k < 1$, then there exist scalars λ_1 and λ_2, such that

$$0 < \lambda_1 \leq \theta_k \leq \lambda_2 < 1. \tag{2.62}$$

From the relation (2.59) and using (2.62), we conclude

$$g_{k+1}^T d_{k+1} \leq \lambda_1 g_{k+1}^T d_{k+1}^{FR} + (1 - \lambda_2) g_{k+1}^T d_{k+1}^{RMIL}.$$

Let $c = \lambda_1 c_2 + (1 - \lambda_2) c_1$, then from (2.60) and (2.61) we finally get

$$g_{k+1}^T d_{k+1} \leq -c \|g_{k+1}\|^2.$$

□

Lemma 2.6. *Suppose that assumptions (i) and (ii) hold. Consider common iterate (2.2), where d_k is a descent direction (2.45) and α_k is determined by the strong Wolfe line search (2.46) and (2.47). Then, the Zoutendijk condition*

$$\sum_{k \geq 0} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty, \tag{2.63}$$

holds.

Proof. The proof follows directly from [73]. □

Now, we give the global convergence theorem of our algorithm.

Theorem 2.7. *Consider the $RMILFR$ conjugate gradient method and suppose that assumptions (i), (ii) and (2.56) hold. Then either $g_k = 0$ for some k, or*

$$\lim_{k \to \infty} \inf \|g_k\| = 0. \tag{2.64}$$

Proof. We suppose that $g_k \neq 0$, for all k. Then we need to prove (2.64).

Suppose, on the contrary, that (2.64) doesn't hold. Then there exists a constant $t > 0$, such that

$$\|g_k\| \geq t, \text{ for all } k. \tag{2.65}$$

Let D be the diameter of the level set $\mathcal{L}$.

From the formula of β_k^{RMILFR} we get

$$\mid \beta_k^{RMILFR} \mid \leq \mid \beta_k^{RMIL} \mid + \mid \beta_k^{FR} \mid = \frac{\mid g_{k+1}^T y_k \mid}{\|d_k\|^2} + \frac{\|g_{k+1}\|^2}{\|g_k\|^2} \tag{2.66}$$

From (2.55) and (2.65), we have $\|g_{k+1}\| \leq \mu$ and $\|g_k\| \geq t$, then we get

$$\mid \beta_k^{FR} \mid = \frac{\|g_{k+1}\|^2}{\|g_k\|^2} \leq \frac{\mu^2}{t^2}. \tag{2.67}$$

Also using (2.54) and (2.55), we obtain

$$\mid g_{k+1}^T y_k \mid \leq \|g_{k+1}\| \, \|g_{k+1} - g_k\| \leq \mu L \|x_{k+1} - x_k\| \leq \mu L D. \tag{2.68}$$

On the one hand, we have

$$y_k^T d_k \leq \|y_k\| \, \|d_k\| \leq L \, \|s_k\| \, \|d_k\| = L\alpha_k \|d_k\|^2,$$

then

$$\|d_k\|^2 \geq \frac{1}{L\alpha_k} y_k^T d_k. \tag{2.69}$$

In the other hand, from (2.47) we have

$$y_k^T d_k = g_{k+1}^T d_k - g_k^T d_k \geq (\sigma - 1) g_k^T d_k = -(1 - \sigma) g_k^T d_k,$$

so (2.69) becomes

$$\|d_k\|^2 \geq -\frac{1}{L\alpha_k}(1 - \sigma) g_k^T d_k.$$

But, all conditions of Theorem 3.1 are satisfied, so it holds that:

$$g_k^T d_k \leq -c\|g_k\|^2, \text{ which implies } -g_k^T d_k \geq c\|g_k\|^2,$$

so

$$\|d_k\|^2 \geq \frac{1}{L\alpha_k}(1 - \sigma) c \|g_k\|^2,$$

then using (2.65), we get

$$\frac{1}{\|d_k\|^2} \leq \frac{L\alpha_k}{(1 - \sigma) c \|g_k\|^2} \leq \frac{L\alpha_k}{(1 - \sigma) c t^2}, \tag{2.70}$$

and from (2.68), (2.70) and the formula of β_k^{RMIL}, we obtain

$$\left|\beta_k^{RMIL}\right| = \frac{\left|g_{k+1}^T y_k\right|}{\|d_k\|^2} \leq \frac{\mu L^2 D}{(1-\sigma)ct^2}\alpha_k. \tag{2.71}$$

Now, using (2.66), (2.67) and (2.71), we get

$$\mid \beta_k^{RMILFR} \mid \leq \frac{\mu L^2 D}{(1-\sigma)ct^2}\alpha_k + \frac{\mu^2}{t^2}. \tag{2.72}$$

Next, we prove that there exists a constant $\alpha_* > 0$ such that

$$\alpha_k \geq \alpha_* > 0, \quad \text{for all } k.$$

Assume, the contrary. Then, for any $\alpha_* > 0$, there exists an infinite subsequence $\{\alpha_k\}_{k\in K_1}$ such that $\alpha_k = \gamma^{j_k}$ and

$$\lim_{k\in K_1} \alpha_k = 0.$$

Since $\gamma \in (0,1)$, this implies

$$\lim_{k\in K_1} \gamma^{j_k-1} = 0, \tag{2.73}$$

which is equivalent to

$$\lim_{k\in K_1} (j_k - 1) = \infty.$$

From the condition (2.46), we know that the accepted stepsize $\alpha_k = \gamma^{j_k}$ satisfies

$$f(x_k + \gamma^{j_k} d_k) - f(x_k) \leq \delta\gamma^{j_k} g_k^T d_k. \tag{2.74}$$

On the other hand, since γ^{j_k-1} is rejected, that means

$$f(x_k + \gamma^{j_k-1} d_k) - f(x_k) > \delta\gamma^{j_k-1} g_k^T d_k. \tag{2.75}$$

Dividing inequality (2.75) by γ^{j_k-1} gives

$$\frac{f(x_k + \gamma^{j_k-1} d_k) - f(x_k)}{\gamma^{j_k-1}} > \delta g_k^T d_k. \tag{2.76}$$

Taking the limit as $k \in K_1 \to \infty$, and using the fact that $\gamma^{j_k-1} \to 0$ from (2.73), the left-hand side of (2.76) tends to the directional derivative of f at x_k in the direction d_k, which is $g_k^T d_k$. Therefore, we obtain

$$g_k^T d_k \geq \delta g_k^T d_k. \tag{2.77}$$

Since $0 < \delta < 1$, we subtract $\delta g_k^T d_k$ from both sides of (2.77) to get

$$(1-\delta) g_k^T d_k \geq 0 \quad \Rightarrow \quad g_k^T d_k \geq 0,$$

which contradicts with the condition (2.56) where we have $g_k^T d_k < -c\|g_k\|^2 < 0$, because in the beginning of the proof we have supposed $g_k \neq 0$.

Therefore, our initial assumption must be false i.e., that there exists a constant $\alpha_* > 0$ such that $\alpha_k \geq \alpha_*$ for all k.

Now, we can write

$$\|d_{k+1}\| \leq \|g_{k+1}\| + \mid \beta_k^{RMILFR} \mid \|d_k\|. \tag{2.78}$$

Since $s_k = \alpha_k d_k$, we write $d_k = \dfrac{s_k}{\alpha_k}$ and

$$\|d_k\| = \frac{\|s_k\|}{\alpha_k} \leq \frac{D}{\alpha_k}. \tag{2.79}$$

So from (2.55), (2.72) and (2.79) the relation (2.78) becomes

$$\|d_{k+1}\| \leq \mu + \left(\frac{\mu L^2 D}{(1-\sigma)ct^2}\alpha_k + \frac{\mu^2}{t^2}\right)\frac{D}{\alpha_k},$$

and since $\alpha_k \geq \alpha_*$, wherefrom

$$\|d_{k+1}\| \leq \mu + \left(\frac{\mu L^2 D^2}{(1-\sigma)ct^2} + \frac{\mu^2 D}{t^2 \alpha_*}\right) = Cste, \tag{2.80}$$

which gives

$$\sum_{k\geq 0} \frac{1}{\|d_k\|^2} = \infty. \tag{2.81}$$

On the other hand, from (2.56), (2.101) and from the Zoutendijk condition (2.63), it results that

$$c^2 t^4 \sum_{k\geq 0} \frac{1}{\|d_k\|^2} \leq \sum_{k\geq 0} \frac{c^2\|g_k\|^4}{\|d_k\|^2} \leq \sum_{k\geq 0} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty,$$

which contradicts (2.81). Therefore, (2.101) does not hold.

Hence, $\lim\limits_{k\to\infty} \inf \|g_k\| = 0$. □

2.3.2 Numerical results

In this section, we present 100 numerical tests using different functions of unconstrained nonlinear optimization problems, taken from [1, 2, 26, 50].

The objective of this experiments is to show the performance of our new algorithm RMILFR in comparing with RMIL, FR , HMLSFR [27] and HA [38]. We considered $\varepsilon = 10^{-6}$ and the stopping criterion is $\|g_k\| \leq \varepsilon$ or the iteration number> 4000. The stepsize α_k is calculated using the strong Wolfe line search, where we set the parameters

$\sigma = 0.01$, $\delta = 0.1$. We have tested the problems for different dimensions, from $n = 10$ to $n = 800000$. This selection was made to ensure a representative mix of problem types and sizes, ranging from small-scale problems to very large-scale problems. We ensured that the problems tested cover both quadratic and highly nonlinear functions.

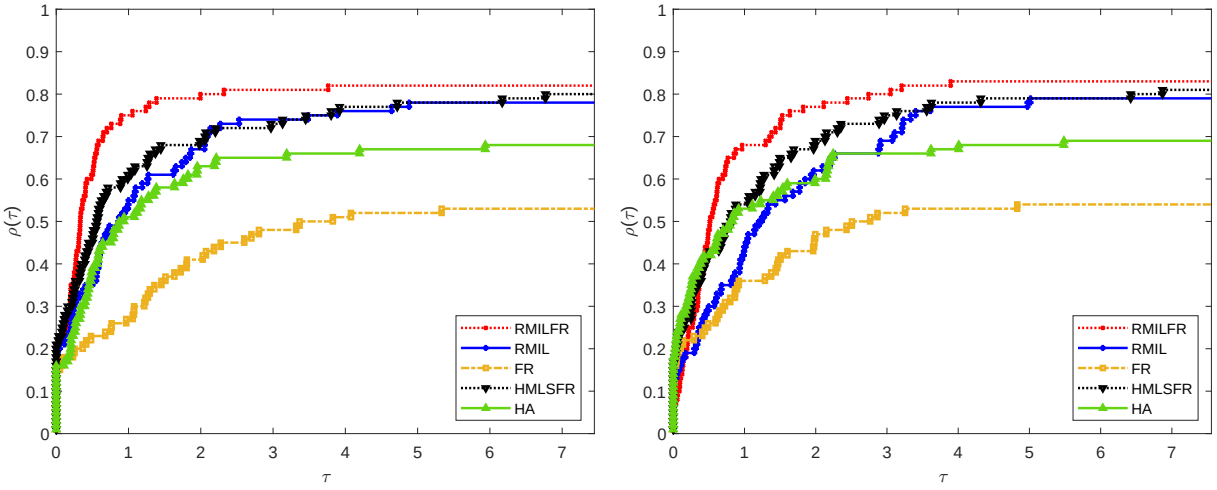
All experiments were conducted using MATLAB R2021a on a machine running Windows 8.1, without requiring any additional toolboxes.

The computations were executed on a computer equipped with an Intel Core i5-6300U CPU @ 2.4 GHz, 8 GB of RAM, and no dedicated GPU. This setup represents a typical personal computing environment.

The evaluation criterion used to determine the best method out of the three methods is, iterations number, CPU time, gradient evaluation and objective function evaluation. To describe the performance of our method, we used the performance profile of Dolan and Moré [20].

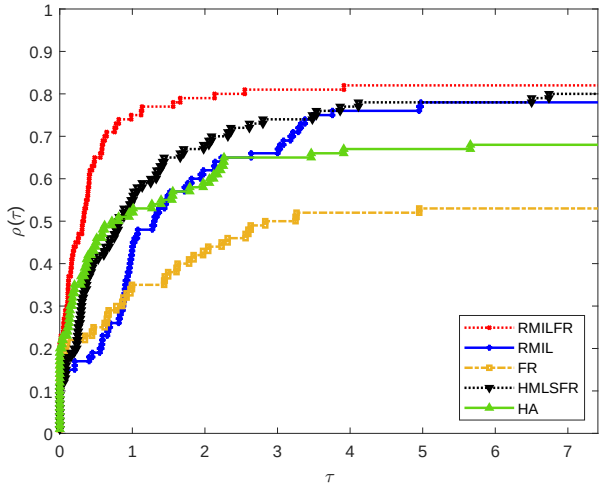
Table 2.3: List of test functions and their dimensions.

Tests functions	**Dimensions**	**Tests functions**	**Dimensions**
Cosine	6000, 100000, 800000	Tridia	300, 2000
Dixmaana	6000, 90000	Woods	150000, 200000
Dixmaanb	24000, 48000	Bdexp	5000, 50000, 500000
Dixmaanc	2700, 27000	Exdenschnf	90000, 280000, 600000
Dixmaand	12000, 90000	Exdenschnb	6000, 24000, 300000
Dixmaane	2400, 48000	Genquartic	9000, 90000, 500000
Dixmaanf	15000, 60000	Biggsb1	300
Dixmaang	12000, 90000	Sine	100000, 250000, 500000
Dixmaanh	6000, 150000	Fletcbv3	100
Dixmaani	360	Nonscomp	5000, 80000
Dixmaanj	3000, 15000	Power1	150
Dixmaank	12000, 12000	Raydan1	500, 5000
Dixmaanl	2400, 24000	Raydan2	2000, 20000, 500000
Dixon3dq	150	Diagonal1	800, 2000
Dqdrtic	9000, 90000	Diagonal2	8000, 50000
Dqrtic	5000, 150000	Diagonal3	500, 2000
Edensch	7000, 40000, 50000	Bv	2000, 20000
Eg2	100	Ie	500, 1500
Fletchcr	1000, 50000, 200000	Singx	1000, 2000
Freuroth	460	Lin	100, 1300
Genrose	10000	Osb2	10
Himmelbg	70000, 240000	Pen1	200, 1000
Liarwhd	6000, 30000	Pen2	160
Penalty1	4000, 10000	Rosex	500, 1000
Quartc	80000, 50000	Trid	500, 8000



(a) Iterations number

(b) CPU time



(c) Objective function evaluations

Figure 2.7: Performance profile of RMILFR.

Comments

From the performance profile results shown in figures (2.7a), (2.7b), and(2.7c), we observe that the RMILFR method delivers comparable results to the other conjugate gradient methods (RMIL, FR, HMLSFR, and HA) in the initial range $0 < \tau < \frac{1}{2}$, except in figure (b), we notice that our method RMILFR gives lower performance. However, when $\frac{1}{2} < \tau < 7$, RMILFR consistently outperforms the competing methods in all considered terms, namely iteration number, CPU time and objective function evaluations.

2.4 A new hybrid conjugate gradient method based on RMIL, CD and LS methods

In this section, we propose a new hybrid conjugate gradient method for solving (2.1). The method is based on a convex combination of the RMIL, CD and LS conjugate gradient formulas. The proposed parameter $\beta_k^{RMILCDLS}$ is defined by

$$\beta_k^{RMILCDLS} = \lambda_k \beta_k^{RMIL} + \theta_k \beta_k^{CD} + (1 - \lambda_k - \theta_k)\beta_k^{LS}, \quad \lambda_k, \theta_k \in [0, 1],\ \lambda_k + \theta_k \leq 1. \tag{2.82}$$

Substituting the expressions for the individual conjugate gradient parameters, we get

$$\beta_k^{RMILCDLS} = \lambda_k \frac{g_{k+1}^T y_k}{\|d_k\|^2} + \theta_k \frac{\|g_{k+1}\|^2}{-g_k^T d_k} + (1 - \lambda_k - \theta_k)\frac{g_{k+1}^T y_k}{-g_k^T d_k}, \tag{2.83}$$

where $y_k = g_{k+1} - g_k$.

The search direction is generated using the recurrence

$$d_0 = -g_0, \quad d_{k+1} = -g_{k+1} + \beta_k^{RMILCDLS} d_k. \tag{2.84}$$

To ensure global convergence and sufficient descent, we employ the strong Wolfe line search conditions

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \delta \alpha_k g_k^T d_k, \tag{2.85}$$

$$\sigma g_k^T d_k \leq g_{k+1}^T d_k \leq -\sigma g_k^T d_k, \tag{2.86}$$

where $0 < \delta \leq \sigma < \frac{4}{5}$. The parameters λ_k and θ_k are chosen to satisfy the conjugacy condition:

$$y_k^T d_{k+1} = 0. \tag{2.87}$$

Substituting (2.84) into (2.87) and using (2.82), we get

$$\begin{aligned} y_k^T d_{k+1} &= -y_k^T g_{k+1} + \beta_k^{RMILCDLS} y_k^T d_k = 0, \\ \Rightarrow y_k^T g_{k+1} &= \beta_k^{RMILCDLS} y_k^T d_k. \end{aligned} \tag{2.88}$$

Replacing $\beta_k^{RMILCDLS}$ by its expression from (2.82), we obtain

$$\begin{aligned} y_k^T g_{k+1} &= \left[\lambda_k \beta_k^{RMIL} + \theta_k \beta_k^{CD} + (1 - \lambda_k - \theta_k)\beta_k^{LS}\right] y_k^T d_k \\ &= \lambda_k (\beta_k^{RMIL} - \beta_k^{LS}) y_k^T d_k + \theta_k (\beta_k^{CD} - \beta_k^{LS}) y_k^T d_k + \beta_k^{LS} y_k^T d_k. \end{aligned} \tag{2.89}$$

From (2.89), we get

$$\lambda_k = \frac{y_k^T g_{k+1} - \theta_k(\beta_k^{CD} - \beta_k^{LS}) y_k^T d_k - \beta_k^{LS} y_k^T d_k}{(\beta_k^{RMIL} - \beta_k^{LS}) y_k^T d_k}. \tag{2.90}$$

To ensure feasibility of the parameters, we enforce the following bounds:

$$\lambda_k = \begin{cases} 0, & \text{if } \lambda_k < 0, \\ 1, & \text{if } \lambda_k > 1, \\ 1 - \theta_k, & \text{if } \lambda_k + \theta_k > 1. \end{cases}$$

Depending on the values of λ_k and θ_k, the parameter $\beta_k^{RMILCDLS}$ reduces to known methods:

$$\beta_k^{RMILCDLS} = \begin{cases} \beta_k^{RMIL}, & \lambda_k = 1,\ \theta_k = 0, \\ \beta_k^{CD}, & \lambda_k = 0,\ \theta_k = 1, \\ \beta_k^{LS}, & \lambda_k = 0,\ \theta_k = 0, \\ \lambda_k \beta_k^{RMIL} + (1 - \lambda_k)\beta_k^{LS}, & \theta_k = 0,\ \lambda_k \in (0,1), \\ \lambda_k \beta_k^{RMIL} + (1 - \lambda_k)\beta_k^{CD}, & \theta_k = 1 - \lambda_k,\ \lambda_k \in (0,1), \\ \theta_k \beta_k^{CD} + (1 - \theta_k)\beta_k^{LS}, & \lambda_k = 0,\ \theta_k \in (0,1), \\ \lambda_k \beta_k^{RMIL} + \theta_k \beta_k^{CD} + (1 - \lambda_k - \theta_k)\beta_k^{LS}, & \lambda_k, \theta_k \in (0,1),\ \lambda_k + \theta_k < 1. \end{cases} \tag{2.91}$$

Now, we give the corresponding algorithm to our $\beta_k^{RMILCDLS}$:

Algorithm 11: RMILCDLS

Begin algorithm

Step 0: Given a starting point $x_0 \in \mathbb{R}^n$ and a parameter $\varepsilon > 0$, compute $g_0 = \nabla f(x_0)$, then set $d_0 = -g_0$.

Step 1: If $\|g_k\| \leq \varepsilon$, **Stop**; else go to **Step 2**.

Step 2: Compute the stepsize α_k using (2.85) and (2.86).

Step 3: Update $x_{k+1} = x_k + \alpha_k d_k$.

Step 4: Compute $g_{k+1} = \nabla f(x_{k+1})$, $y_k = g_{k+1} - g_k$.

Step 5: If $g_{k+1}^T g_k = 0$, then set $\lambda_k = 0$

else compute λ_k as in (2.90) with $0 \leq \theta_k \leq 1$.

Step 6: Compute $\beta_k^{RMILCDLS}$ using formula (2.83).

Step 7: Set $d_{k+1} = -g_{k+1} + \beta_k^{RMILCDLS} d_k$.

Step 8: If $\mid g_{k+1}^T g_k \mid \geq 0.2\|g_{k+1}\|$ (restart criterion of Powell [59])

then set $d_{k+1} = -g_{k+1}$.

else compute $d_{k+1} = -g_{k+1} + \beta_k^{RMILCDLS} d_k$.

End if

Set the initial guess $\alpha_k = 1$.

Step 9: Let $k = k + 1$ and go to **Step 1**.

End algorithm

2.4.1 Convergence analysis

Throughout this section, we make the following assumptions

Assumption(i): The level set $S = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ is bounded, i.e. there exists a constant $B > 0$, such that

$$\|x\| \leq B, \text{ for all } x \in S \tag{2.92}$$

Assumption(ii): In a neighborhood N of S the function f is continuously differentiable and its gradient $\nabla f(x)$ is Lipschitz continuous, i.e., there exists a constant $0 < L < \infty$ such that

$$\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\|, \text{ for all } x, y \in N. \tag{2.93}$$

Under Assumptions **(i)** and **(ii)** on f, there exists a constant $\mu \geq 0$, such that

$$\|\nabla f(x)\| \leq \mu, \text{ for all } x, y \in N. \tag{2.94}$$

To establish the sufficient descent condition, we introduce the following theorem.

Theorem 2.8. *Let $\{d_k\}_{k\in\mathbb{N}}$ be given by (2.84), α_k satisfies (2.85) and (2.86), then*

$$g_{k+1}^T d_{k+1} \leq -c \|g_{k+1}\|^2, \ k = 0, 1, ... \tag{2.95}$$

where $c > 0$ and $\sigma < \frac{4}{5}$.

Proof. Induction is used to show (2.95).

Since $d_0 = -g_0$, we get $g_0^T d_0 = -\|g_0\|^2 < 0$. Consider that (2.95) holds for $k > 0$.

We have,

$$\left|g_{k+1}^T g_k\right| \geq 0.2 \|g_{k+1}\|^2. \tag{2.96}$$

If (2.96) holds, then $g_{k+1}^T d_{k+1} = -\|g_{k+1}\|^2 < 0$.

The search direction that meets the sufficient descent condition is achieved.

If (2.96) does not hold, then

$$\left|g_{k+1}^T g_k\right| < 0.2 \|g_{k+1}\|^2. \tag{2.97}$$

Using (2.86), we can get that

$$y_k^T d_k = (g_{k+1} - g_k)^T d_k \geq -(1-\sigma) g_k^T d_k. \tag{2.98}$$

And,

$$\left|\frac{g_{k+1}^T d_k}{y_k^T d_k}\right| \leq \frac{\sigma}{(1-\sigma)}.$$

Multiplying both sides of (2.84) by g_{k+1}^T, we get

$$\begin{aligned} g_{k+1}^T d_{k+1} &= -\|g_{k+1}\|^2 + \lambda_k \beta_k^{RMIL} g_{k+1}^T d_k + \theta_k \beta_k^{CD} g_{k+1}^T d_k \\ &\quad +(1-\lambda_k-\theta_k)\beta_k^{LS} g_{k+1}^T d_k \end{aligned}$$

We have seven cases.

Case 01: If $\lambda_k = 1$, $\theta_k = 0$, we have $g_{k+1}^T d_{k+1} = g_{k+1}^T d_{k+1}^{RMIL}$. For the RMIL direction $d_{k+1}^{RMIL} = -g_{k+1} + \beta_k^{RMIL} d_k$, it is proved in [61] that

$$g_{k+1}^T d_{k+1}^{RMIL} \leq -a_1 \|g_{k+1}\|^2, \text{ for all } k, \text{ where} a_1 > 0$$

where $a_1 > 0$. **Case 02:** If $\lambda_k = 0$, $\theta_k = 1$, we have $g_{k+1}^T d_{k+1}^{New} = g_{k+1}^T d_{k+1}^{CD}$. For the conjugate descent direction $d_{k+1}^{CD} = -g_{k+1} + \beta_k^{CD} d_k$, it is proved in [17] that

$$g_{k+1}^T d_{k+1}^{CD} \leq -a_2 \|g_{k+1}\|^2, \text{ for all } k, \text{ where} a_2 > 0$$

where $a_2 > 0$. **Case 03:** If $\lambda_k = 0$, $\theta_k = 0$, we have $g_{k+1}^T d_{k+1}^{New} = g_{k+1}^T d_{k+1}^{LS}$. For the Liu-Storey direction $d_{k+1}^{LS} = -g_{k+1} + \beta_k^{LS} d_k$, it is proved in [17] that

$$g_{k+1}^T d_{k+1}^{LS} \leq -a_3 \|g_{k+1}\|^2, \text{ for all } k, \text{ where} a_3 > 0$$

where $a_3 > 0$. **Case 04:** If $\lambda_k \in]0, 1[$, $\theta_k = 0$, we have $g_{k+1}^T d_{k+1} = g_{k+1}^T d_{k+1}^{RMILLS}$

We have

$$d_{k+1} = \lambda_k d_{k+1}^{RMIL} + \theta_k d_{k+1}^{CD} + (1 - \lambda_k - \theta_k) d_{k+1}^{LS}.$$

Hence,

$$g_{k+1}^T d_{k+1} = \lambda_k g_{k+1}^T d_{k+1}^{RMIL} + \theta_k g_{k+1}^T d_{k+1}^{CD} + (1 - \lambda_k - \theta_k) g_{k+1}^T d_{k+1}^{LS}. \tag{2.99}$$

With (2.99), we get

$$g_{k+1}^T d_{k+1}^{RMILLS} = \lambda_k g_{k+1}^T d_{k+1}^{RMIL} + (1 - \lambda_k) g_{k+1}^T d_{k+1}^{LS}.$$

$\exists w_1, w_2 \in \mathbb{R} : 0 < w_1 < \lambda_k < w_2 < 1$, then

$$g_{k+1}^T d_{k+1}^{RMILLS} \leq -(w_1 a_1 + w_2 a_3) \|g_{k+1}\|^2 = -a_4 \|g_{k+1}\|^2,$$

where $a_4 > 0$.

Case 05: If $\theta_k = 1 - \lambda_k$ and $\lambda_k, \theta_k \in]0, 1[$, we have $g_{k+1}^T d_{k+1} = g_{k+1}^T d_{k+1}^{RMILCD}$

With (2.99), we get

$$g_{k+1}^T d_{k+1}^{RMILCD} = \lambda_k g_{k+1}^T d_{k+1}^{RMIL} + (1 - \lambda_k) g_{k+1}^T d_{k+1}^{CD}.$$

Clearly, the sufficient descent condition is satisfied, which means that

$$g_{k+1}^T d_{k+1}^{RMILCD} \leq -a_5 \|g_{k+1}\|^2,$$

where $a_5 > 0$.

Case 06: If $\lambda_k = 0, \theta_k \in]0,1[$, we have $g_{k+1}^T d_{k+1} = g_{k+1}^T d_{k+1}^{CDLS}$

With (2.99), we get

$$g_{k+1}^T d_{k+1}^{CDLS} = \theta_k g_{k+1}^T d_{k+1}^{CD} + (1-\theta_k) g_{k+1}^T d_{k+1}^{LS}.$$

Case 06 is the same as **Case 04** and **Case 05**.

So, the sufficient descent condition is satisfied, which means

$$g_{k+1}^T d_{k+1}^{CDLS} \leq -a_6 \|g_{k+1}\|^2,$$

where $a_6 > 0$.

Case 07: If $\lambda_k, \theta_k \in]0,1[$ and $0 < \lambda_k + \theta_k < 1$, we have

$$g_{k+1}^T d_{k+1} = \lambda_k g_{k+1}^T d_{k+1}^{RMIL} + \theta_k g_{k+1}^T d_{k+1}^{CD} + (1-\lambda_k-\theta_k) g_{k+1}^T d_{k+1}^{LS}.$$

$\exists k_1, k_2, k_3, k_4 \in \mathbb{R} : 0 < k_1 < \lambda_k < k_2 < 1, 0 < k_3 < \theta_k < k_4 < 1$, then

$$\begin{aligned} g_{k+1}^T d_{k+1} &\leq -(k_1 a_1 + k_2 a_2 + (1-k_3-k_4) a_3) \|g_{k+1}\|^2 \\ &= -a_7 \|g_{k+1}\|^2. \end{aligned}$$

where $a_7 > 0$.

The proof is complete. □

Lemma 2.9. *Suppose that assumptions (i) and (ii) hold. Consider common iterate (2.2), where d_k is a descent direction that verifies (2.95) and α_k is determined by the strong Wolfe line search (2.85) and (2.86). Then, the Zoutendijk condition*

$$\sum_{k\geq 0} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty, \tag{2.100}$$

holds.

Proof. The proof follows directly from [73]. □

Lemma 2.10. *Suppose that assumptions (i) and (ii) hold and $\alpha_k > 0$ is determined by the strong Wolfe line search (2.85) and (2.86). Then there exists $\alpha^* > 0$, such that*

$$\alpha_k \geq \alpha_* > 0, \text{ for all } k.$$

Proof. The proof follows directly from [17]. □

Theorem 2.11. *Consider the $RMILCDLS$ conjugate gradient method and suppose that assumptions (i), (ii) and (2.86) hold. Then either $g_k = 0$ for some k, or*

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{2.101}$$

Proof. Suppose that $g_k \neq 0$ for all k. Then, we are going to prove (2.101).

Suppose by contradiction that (2.101) doesn't hold. Then there exists $t > 0$ such that

$$\|g_k\| \geq t \text{ for all } k. \tag{2.102}$$

Let D denote the diameter of the level set S, and define the step $s_k = x_{k+1} - x_k = \alpha_k d_k$.

From equation (2.82), we have:

$$\left|\beta_k^{RMILCDLS}\right| \leq \left|\beta_k^{RMIL}\right| + \left|\beta_k^{CD}\right| + \left|\beta_k^{LS}\right|. \tag{2.103}$$

We begin by analyzing each of the components on the right-hand side.

By definition,

$$\left|\beta_k^{RMIL}\right| = \frac{\left|g_{k+1}^T y_k\right|}{\|d_k\|^2}.$$

From equations (2.102) and (2.95), we have

$$\|d_k\|^2 \geq \frac{-(1-\sigma)g_k^T d_k}{L\alpha_k} \geq \frac{(1-\sigma)c\|g_k\|^2}{L\alpha_k} \geq \frac{(1-\sigma)ct^2}{L\alpha_k}.$$

Thus, we obtain the bound

$$\frac{1}{\|d_k\|^2} \leq \frac{L\alpha_k}{(1-\sigma)ct^2}. \tag{2.104}$$

Next, for $\left|g_{k+1}^T y_k\right|$, we have

$$\begin{aligned}
\left|g_{k+1}^T y_k\right| &\leq \|g_{k+1}\| \, \|y_k\| \\
&\leq \mu \|g_{k+1} - g_k\| \\
&\leq \mu L \|x_{k+1} - x_k\| \\
&\leq \mu L \|s_k\| \\
\left|g_{k+1}^T y_k\right| &\leq \mu L D.
\end{aligned} \tag{2.105}$$

Combining equations (2.104) and (2.105), we get

$$\left|\beta_k^{RMIL}\right| \leq \frac{\mu L^2 D\alpha_k}{(1-\sigma)ct^2}. \tag{2.106}$$

For $\left|\beta_k^{CD}\right|$, we have

$$\left|\beta_k^{CD}\right| \leq \frac{\|g_{k+1}\|^2}{|-g_k^T d_k|} \leq \frac{\mu^2}{-c\|g_k\|^2} \leq \frac{\mu^2}{ct^2}. \tag{2.107}$$

Finally, for $\left|\beta_k^{LS}\right|$, we have

$$\left|\beta_k^{\text{LS}}\right| = \left|\frac{g_{k+1}^T y_k}{-g_k^T d_k}\right| \leq \frac{\|g_{k+1}\|\,\|y_k\|}{|-g_k^T d_k|} \leq \frac{\mu LD}{c_2 t^2}. \tag{2.108}$$

From (2.106), (2.107) and (2.108), we can write

$$\left|\beta_k^{RMILCDLS}\right| \leq \frac{\mu L^2 D\alpha_k}{(1-\sigma)ct^2} + \frac{\mu^2}{c_1 t^2} + \frac{\mu LD}{c_2 t^2}. \tag{2.109}$$

Now, we can write

$$\|d_{k+1}\| \leq \|g_{k+1}\| + \left|\beta_k^{RMILCDLS}\right| \|d_k\|. \tag{2.110}$$

We have $s_k = \alpha_k d_k$. So, we can write $d_k = \dfrac{s_k}{\alpha_k}$. Then, from (2.109) and (2.110), we get

$$\begin{aligned} \|d_{k+1}\| &\leq \mu + \left(\frac{\mu L^2 D\alpha_k}{(1-\sigma)ct^2} + \frac{\mu^2}{c_1 t^2} + \frac{\mu LD}{c_2 t^2}\right)\frac{\|s_k\|}{\alpha_k} \\ &\leq \mu + \left(\frac{\mu L^2 D\alpha_*}{(1-\sigma)ct^2} + \frac{\mu^2}{c_1 t^2} + \frac{\mu LD}{c_2 t^2}\right)\frac{D}{\alpha_*} = P. \end{aligned} \tag{2.111}$$

Which implies that

$$\sum_{k\geq 1} \frac{1}{\|d_{k+1}\|^2} = \infty. \tag{2.112}$$

On the other hand, from (2.96), (2.102) and from the Zoutendijk condition (2.100), it results that

$$C^2 t^4 \sum_{k\geq 0} \frac{1}{\|d_k\|^2} \leq \sum_{k\geq 0} \frac{C^2\|g_k\|^4}{\|d_k\|^2} \leq \sum_{k\geq 0} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty,$$

which contradicts (2.112). Therefore, (2.102) doesn't hold.

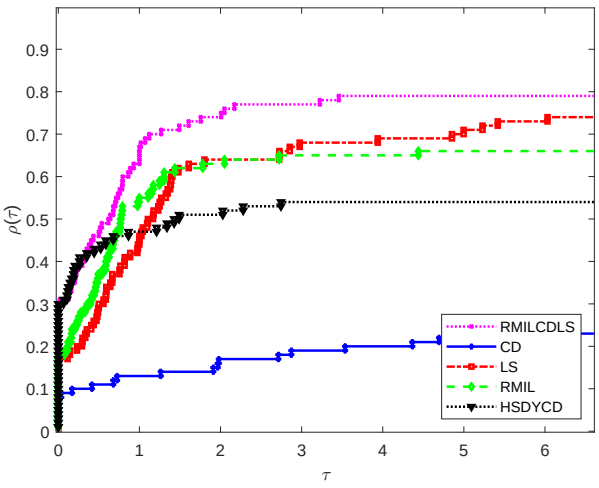
Then, $\lim\limits_{k\to\infty} \inf \|g_k\| = 0$. This completes the proof. □

2.4.2 Numerical experiments

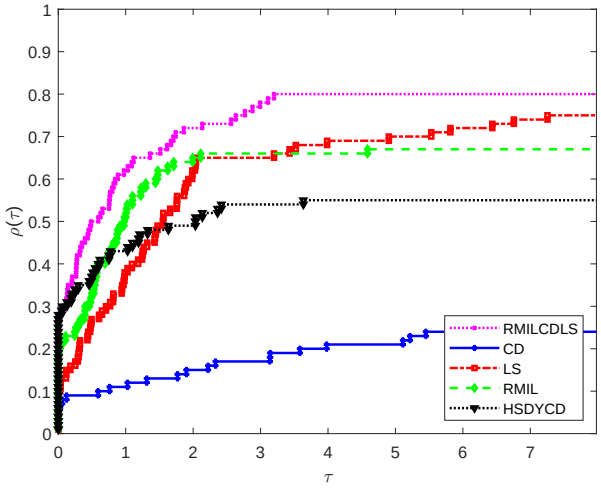
In this section, we present several numerical tests to evaluate the performance of RMILCDLS algorithm. We select various test functions given in Table 2.4, taken from the CUTE library [2, 26] for different dimensions, from $n = 10$ to $n = 800000$. At the same time, we present a numerical comparison with other conjugate gradient algorithm namely RMIL, LS, CD and HSDYCD [28]. The stopping criterion is $\|g_k\| \leq \varepsilon$ where $\varepsilon = 10^{-6}$, or the iteration number> 4000. The step-size α_k is computed by strong Wolfe line search with $\sigma = 0.01$ $\delta = 0.1$. In order to evaluate the data of iterations number, CPU time, gradient evaluation and objective function evaluation, we use the performance profile of Dolan and Moré [20] presented in figure 2.8.

Table 2.4: List of test functions and their dimensions.

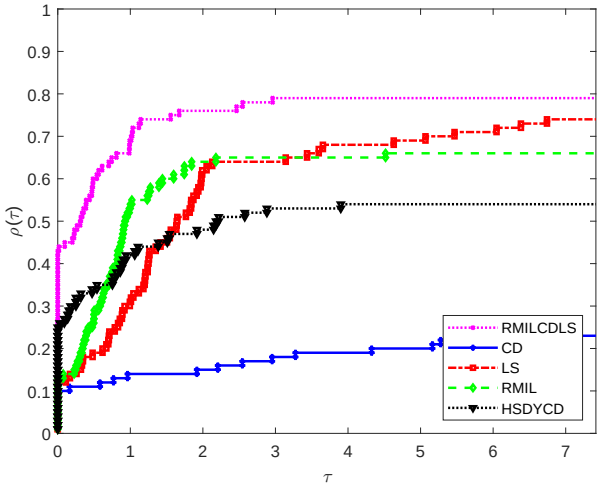
Tests functions	**Dimensions**	**Tests functions**	**Dimensions**
Cosine	6000, 100000, 800000	Tridia	300, 2000
Dixmaana	6000, 90000	Woods	150000, 200000
Dixmaanb	24000, 48000	Bdexp	5000, 50000, 500000
Dixmaanc	2700, 27000	Exdenschnf	90000, 280000, 600000
Dixmaand	12000, 90000	Exdenschnb	6000, 24000, 300000
Dixmaane	2400, 48000	Genquartic	9000, 90000, 500000
Dixmaanf	15000, 60000	Biggsb1	300
Dixmaang	12000, 90000	Sine	100000, 250000, 500000
Dixmaanh	6000, 150000	Fletcbv3	100
Dixmaani	360	Nonscomp	5000, 80000
Dixmaanj	3000, 15000	Power1	150
Dixmaank	12000, 12000	Raydan1	500, 5000
Dixmaanl	2400, 24000	Raydan2	2000, 20000, 500000
Dixon3dq	150	Diagonal1	800, 2000
Dqdrtic	9000, 90000	Diagonal2	8000, 50000
Dqrtic	5000, 150000	Diagonal3	500, 2000
Edensch	7000, 40000, 50000	Bv	2000, 20000
Eg2	100	Ie	500, 1500
Fletchcr	1000, 50000, 200000	Singx	1000, 2000
Freuroth	460	Lin	100, 1300
Genrose	10000	Osb2	10
Himmelbg	70000, 240000	Pen1	200, 1000
Liarwhd	6000, 30000	Pen2	160
Penalty1	4000, 10000	Rosex	500, 1000
Quartc	80000, 50000	Trid	500, 8000



(a) Iterations number



(b) CPU time



(c) Objective function evaluations

Figure 2.8: Performance profile of RMILCDLS.

Comments

The performance profile results, as depicted in figures (2.8a), (2.8b) and (2.8c), show that our proposed conjugate gradient method RMILCDLS, using $\beta_k^{RMILCDLS}$, consistently outperforms methods based on β_k^{RMIL}, β_k^{CD}, β_k^{LS} and β_k^{HSDYCD}.

2.5 Conclusion

In this chapter, we proposed several new hybrid conjugate gradient methods derived through convex combinations of classical conjugate gradient formulas. These hybrid approaches were developed with the aim of improving both convergence speed and ro-

bustness when solving nonlinear unconstrained optimization problems. The proposed algorithms were particularly applied to image restoration problems, where numerical experiments show their efficiency in terms of PSNR values, iteration number, and CPU time. The convergence analysis was provided for each method under suitable conditions, ensuring global convergence. Furthermore, through an extensive set of numerical experiments and comparative studies, it was shown that the proposed hybrid methods outperform several classical and existing hybrid conjugate gradient algorithms, especially in large-scale problems. These results confirm the effectiveness of combining the strengths of multiple conjugate gradient approaches, and they give a solid foundation for further development and application of hybrid methods in broader optimization contexts.

CHAPTER 3

An efficient hybrid conjugate gradient method for large-scale nonlinear equations with applications in compressive sensing

This chapter presents a new method for solving large-scale nonlinear monotone equations subject to convex constraints. To address this challenging problem, we develop an algorithm that effectively integrates the hybrid RMIL technique, known for its performance in unconstrained optimization, with the projection framework established by Solodov and Svaiter [67]. A key advantage of our proposed method is that it operates without the need to store large matrices, making it particularly efficient and well-suited for large-scale problems. We provide a rigorous theoretical foundation for the algorithm, proving its global convergence under standard conditions. The effectiveness of our approach is shown through comprehensive numerical experiments. These tests show favorable performance compared to existing methods across a variety of benchmark problems. Furthermore, we highlight the practical utility of the algorithm by applying it to a problem in compressive sensing. The results of this work was accepted for publication in **"Journal of Nonlinear Modeling and Analysis (JNMA)"** [32].

3.1 Introduction

Nonlinear equations have a wide range of applications in several fields. These equations appear in many practical scenarios, including chemical equilibrium systems [51], power flow equations [36], and contemporary applications like compressive sensing [47] and image restoration [37].

In our study, we focus on solving nonlinear constrained monotone equations of the form

$$F(x) = 0, \quad x \in C, \tag{3.1}$$

where $C \subset \mathbb{R}^n$ is a nonempty closed convex set and $F : \mathbb{R}^n \to \mathbb{R}^n$ is a continuous monotone nonlinear function. F monotone means that

$$\langle F(x) - F(y), x - y \rangle \geq 0, \ \forall x, y \in \mathbb{R}^n \tag{3.2}$$

Solving such equations is challenging, particularly in large-scale settings. Projection techniques, like those developed by Solodov and Svaiter [67], are useful for solving optimization problems with constraints. These methods are especially helpful when we can't use derivatives, allowing us to efficiently solve the problem without needing derivative information.

First order optimization methods are well known for their simplicity and low storage requirements, making them popular for solving large-scale unconstrained optimization problems. Examples include conjugate gradient methods, spectral gradient methods, and spectral conjugate gradient methods. Motivated by the projection schemes of Solodov and Svaiter, many researchers have extended these techniques to handle large-scale nonlinear equations.

Recently, Cheng [12] extended the Polak-Ribière-Polyak (PRP) [57] conjugate gradient method to solve unconstrained monotone equations. Similarly, Xiao et al.[69] adapted the algorithm of Hager and Zhang [29], and Liu and Li [45] extended the Dai and Yuan (DY) conjugate gradient method[13]. Liu and Feng [46] proposed a derivative-free iterative method to solve large-scale nonlinear monotone equations under convex constraints . Ibrahim et al. [37] proposed a hybrid approach combining a convex combination of Liu-Storey (LS) [44] and Fletcher-Reeves (FR) [23] conjugate gradient algorithm that proposed by Djordjevic [19] with projection techniques. Also Yin et al.

[71] made a hybrid three-term conjugate gradient projection method for constrained nonlinear monotone equations. Later, Guodong Ma et al. [53] proposed a modified inertial three-term conjugate gradient using projection method for constrained nonlinear equations. Additionally, Nermah et al. [55] integrated the projection schemes of Solodov and Svaiter with the Picard-Mann hybrid processes, achieving notable improvements.

Motivated by these advancements, we propose our hybrid conjugate gradient method based on the convex combination of the Rivaie–Mustafa–Ismail–Leong (RMIL) method [60] and the Hestenes–Stiefel (HS) method [30], as given in [31], together with projection-based techniques. The RMIL method has good theoretical convergence properties, while the HS method often performs well in practice. By combining them, we are able to take advantage of both approaches, achieving more stable convergence from RMIL and improved search directions from HS. This method is specifically designed to address the challenges posed by large-scale and non-smooth problems, ensuring global convergence under standard assumptions.

3.2 An efficient hybrid conjugate gradient method based on RMILHS parameter

In this section we propose a new algorithm to solve the problem (3.1), which generates the sequence $\{x_k\}$, starting from an initial point $x_1 \in \mathbb{R}^n$

$$x_{k+1} = x_k + \alpha_k d_k, \ k \geq 0, \tag{3.3}$$

First, let denotes $F_k = F(x_k)$.

In this algorithm, the step size $\alpha_k > 0$ is calculated using a line search method, and the search direction is given by

$$d_k = \begin{cases} -F_k, \text{ if } k = 0 \\ -F_k + \beta_k^{RMILHS} \left(I - \frac{F_k F_k^T}{\|F_k\|^2} \right) s_{k-1}, \text{ if } k \geq 1, \end{cases}$$

where

$$\beta_k^{RMILHS} = \begin{cases} (1-\theta_k)\frac{F_k^T y_{k-1}}{\|d_{k-1}\|^2} + \theta_k \frac{F_k^T y_{k-1}}{d_{k-1}^T y_{k-1}}, \text{ if } \theta_k \in (0,1) \\ \beta_k^{RMIL} = \frac{F_k^T y_{k-1}}{\|d_{k-1}\|^2}, \text{ if } \theta_k \geq 1, \\ \beta_k^{HS} = \frac{F_k^T y_{k-1}}{d_{k-1}^T y_{k-1}}, \text{ if } \theta_k \leq 0 \end{cases} \tag{3.4}$$

So,

$$d_k = -F_k + \left((1-\theta_k)\frac{F_k^T y_{k-1}}{\|d_{k-1}\|^2} + \theta_k \frac{F_k^T y_{k-1}}{d_{k-1}^T y_{k-1}}\right)\left(I - \frac{F_k F_k^T}{\|F_k\|^2}\right) s_{k-1}, \tag{3.5}$$

where

$$\theta_k = \frac{-s_{k-1}^T F_k + y_{k-1}^T F_{k-1} - \beta_k^{RMIL} y_{k-1}^T d_{k-1}}{(\beta_k^{RMIL} - \beta_k^{HS}) y_{k-1}^T d_{k-1}}, \tag{3.6}$$

$$t_{k-1} = 1 + \|F_{k-1}\|^{-1} \frac{\max\{0, -d_{k-1}^T \omega_{k-1}\}}{\|d_{k-1}\|^2}, \tag{3.7}$$

and

$$\begin{cases} \omega_{k-1} = F_k - F_{k-1}, \\ s_{k-1} = \alpha_k d_{k-1} \text{ and } y_{k-1} = \omega_{k-1} + t_{k-1}\|F_{k-1}\| d_{k-1}. \end{cases} \tag{3.8}$$

To describe the algorithm, we use the projection map P_C, which returns $\mathbb{R}^n$ onto the convex set C and it is defined as

$$P_C(x) = \arg\min\{\|x - y\| : y \in C, \ \forall x \in \mathbb{R}^n\}. \tag{3.9}$$

This projection map has the well-known nonexpansive property

$$\|P_C(x) - P_C(y)\| \leq \|x - y\|, \quad \forall x, y \in \mathbb{R}^n. \tag{3.10}$$

The new corresponding algorithm named YSD algorithm is given below, where YSD refers to the researchers names (Youcef, Samia and Djamel).

Algorithm 12: YSD

Input. Choose any random point $x_1 \in C$, the positive constants $\xi \in (0,1)$, $\rho \in (0,2)$, precision Tol > 0, $\sigma > 0$. Set $k = 1$.

Step 0. Compute F_k. If $\|F_k\| \leq$ Tol, stop. Otherwise, compute β_k using (3.4) and d_k using (3.5).

Step 1. Let $\alpha_k = \xi^i$ such that i is the first integer number for which the following inequality satisfied

$$-F(x_k + \xi^i d_k)^T d_k \;\geq\; \sigma\, \xi^i \, \|d_k\|^2. \tag{3.11}$$

Step 2. Compute

$$z_k = x_k + \alpha_k d_k. \tag{3.12}$$

Step 3. If $z_k \in C$ and $\|F(z_k)\| \leq$ Tol, stop. Otherwise, using (3.9) compute

$$x_{k+1} = P_C\left[x_k - \rho \phi_k F(z_k)\right], \tag{3.13}$$

where

$$\phi_k = \frac{F(z_k)^T (x_k - z_k)}{\|F(z_k)\|^2}.$$

Step 4. Set $k := k + 1$ and go back to **Step 0**.

3.3 Convergence analysis

To establish the global convergence, we make the assumption that

- *(i)* The function F is Lipschitz continuous on $\mathbb{R}^n$, i.e., there exists a constant $L > 0$ such that

$$\|F(x) - F(y)\| \leq L\|x - y\|, \quad \forall x, y \in \mathbb{R}^n.$$

- *(ii)* The solution set $C^\star$ is nonempty.

We introduce the following Lemma for the sufficient descent condition

Lemma 3.1. *Let d_k be the search direction generated by (3.5). Then, d_k satisfies the sufficient descent condition, that is,*

$$F_k^T d_k = -\|F_k\|^2, \text{ for all } k \geq 0. \tag{3.14}$$

Proof.

$$d_k = -F_k + \beta_k^{RMILHS}\left(I - \frac{F_k F_k^T}{\|F_k\|^2}\right) s_{k-1}. \tag{3.15}$$

Multiply both sides by F_k

$$\begin{aligned} F_k^T d_k &= -\|F_k\|^2 + \beta_k^{RMILHS}\left(F_k^T - \frac{F_k^T F_k F_k^T}{\|F_k\|^2}\right) s_{k-1} \\ &= -\|F_k\|^2 + \beta_k^{RMILHS}\left(F_k^T s_{k-1} - F_k^T s_{k-1}\right) \\ &= -\|F_k\|^2. \end{aligned} \tag{3.16}$$

This proves the sufficient descent condition. □

Lemma 3.2. *The line search is well-defined. That is, for all $k \geq 1$, there exists a nonnegative integer i satisfying* (3.11).

Proof. We proceed by contradiction. Suppose that there exists $k_0 \geq 1$ such that (3.11) is not satisfied for any nonnegative integer i, that is,

$$-F(x_{k_0} + \xi^i d_{k_0})^T d_{k_0} < \sigma \xi^i \|d_{k_0}\|^2, \quad \forall i \geq 1. \tag{3.17}$$

Using the continuity of F and letting $i \to \infty$, we obtain

$$-F_{k_0}^T d_{k_0} \leq 0, \tag{3.18}$$

which contradicts (3.16). This completes the proof. □

Lemma 3.3. *Suppose that F is monotone and Lipschitz continuous on $\mathbb{R}^n$, and the sequence $\{x_k\}$ is generated by the projection step in the YSD algorithm. Then, there exists $\mu > 0$ such that*

$$\|F_k\| \leq \mu. \tag{3.19}$$

Proof. Recall that, from the property (3.10) of the projection operator, it holds that for

any $x^\star \in C^\star$,

$$\begin{aligned}
\|x_{k+1} - x^\star\|^2 &= \|P_C[x_k - \rho\phi_k F(z_k)] - P_C(x^\star)\|^2 \\
&\leq \|x_k - \rho\phi_k F(z_k) - x^\star\|^2 \\
&= \|x_k - x^\star\|^2 - 2\rho\phi_k F(z_k)^T(x_k - x^\star) + \rho^2\phi_k^2\|F(z_k)\|^2 \\
&= \|x_k - x^\star\|^2 - 2\rho\frac{F(z_k)^T(x_k - z_k)}{\|F(z_k)\|^2}F(z_k)^T(x_k - x^\star) \\
&+ \rho^2\left(\frac{F(z_k)^T(x_k - z_k)}{\|F(z_k)\|}\right)^2 \\
&\leq \|x_k - x^\star\|^2 - 2\rho\frac{F(z_k)^T(x_k - z_k)}{\|F(z_k)\|^2}F(z_k)^T(x_k - z_k) \\
&+ \rho^2\left(\frac{F(z_k)^T(x_k - z_k)}{\|F(z_k)\|}\right)^2 \\
&= \|x_k - x^\star\|^2 - \rho(2-\rho)\left(\frac{F(z_k)^T(x_k - z_k)}{\|F(z_k)\|}\right)^2 \\
&\leq \|x_k - x^\star\|^2. \qquad (3.20)
\end{aligned}$$

The inequality (3.20) implies that the sequence $\{\|x_k - x^\star\|\}$ is decreasing. Therefore, the sequence $\{x_k\}$ is bounded, that is,

$$\|x_k\| \leq M, \quad M > 0. \qquad (3.21)$$

In addition, we obtain

$$\|x_{k+1} - x^\star\| \leq \|x_k - x^\star\| \leq \|x_{k-1} - x^\star\| \leq \cdots \leq \|x_0 - x^\star\|. \qquad (3.22)$$

Using the Lipschitz continuity of F, we have

$$\|F_k\| = \|F_k - F(x^\star)\| \leq L\|x_k - x^\star\| \leq L\|x_0 - x^\star\|. \qquad (3.23)$$

Setting $\mu = L\|x_0 - x^\star\|$, we obtain (3.19), which complete the proof. □

Lemma 3.4. *Let $\{z_k\}$ and $\{x_k\}$ be the sequences generated by YSD algorithm, using (3.12) and (3.13), respectively. Then, $-F(z_k)$ is a descent direction of the function $\frac{1}{2}\|x - x^\star\|^2$ at the point x_k, where $x^\star \in C^\star$ is an optimal solution.*

Proof. At x_k, the gradient of the function $\frac{1}{2}\|x - x^\star\|^2$ is given by $x_k - x^\star$. By the mono-

tonicity property (3.2), we have

$$\begin{aligned} F(z_k)^T(x_k - x^\star) &= F(z_k)^T(x_k + z_k - z_k - x^\star) \\ &= F(z_k)^T(z_k - x^\star) + F(z_k)^T(x_k - z_k) \\ &\geq F(x^\star)^T(z_k - x^\star) + F(z_k)^T(x_k - z_k) \\ &= F(z_k)^T(x_k - z_k) \\ &\geq \sigma\alpha_k^2\|d_k\|^2 \\ &= \sigma\|x_k - z_k\|^2 > 0. \end{aligned} \tag{3.24}$$

This inequality shows that

$$F(z_k)^T(x_k - x^\star) > 0.$$

Consequently, $-F(z_k)$ is a descent direction for the function $\frac{1}{2}\|x - x^\star\|^2$ at the iteration point x_k. □

Lemma 3.5. *Suppose that assumptions (i) and (ii) hold. Let be the sequences $\{z_k\}$ and $\{x_k\}$ generated by YSD algorithm, using (3.12) and (3.13), respectively. Then*

(1) $\{z_k\}$ *is bounded.*

(2) $\lim_{k\to\infty}\|x_k - z_k\| = 0$.

(3) $\lim_{k\to\infty}\|x_k - x_{k+1}\| = 0$.

Proof. **(1)** From (3.21), we know that the sequence $\{x_k\}$ is bounded. From (3.24), we have

$$F(z_k)^T(x_k - z_k) \geq \sigma\|x_k - z_k\|^2. \tag{3.25}$$

Using (3.19) and (3.2), we have

$$\begin{aligned} F(z_k)^T(x_k - z_k) &= \left(F(z_k) - F_k\right)^T(x_k - z_k) + F_k^T(x_k - z_k) \\ &\leq \|F_k\|\|x_k - z_k\| \\ &\leq \mu\|x_k - z_k\|. \end{aligned}$$

Combined with (3.25), it is easy to deduce that

$$\|x_k - z_k\| \leq \frac{\mu}{\sigma}.$$

Thus, we obtain

$$\|z_k\| \leq \frac{\mu}{\sigma} + \|x_k\|.$$

Hence, the sequence $\{z_k\}$ is bounded.

(2) From inequality (3.20), we get

$$\begin{aligned}\|x_{k+1} - x^\star\|^2 &\leq \|x_k - x^\star\|^2 - \rho(2-\rho)\frac{\left[F(z_k)^T(x_k - z_k)\right]^2}{\|F(z_k)\|^2} \\ &\leq \|x_k - x^\star\|^2 - \rho(2-\rho)\frac{\sigma^2\|x_k - z_k\|^4}{\|F(z_k)\|^2},\end{aligned}$$

which means

$$\rho(2-\rho)\|x_k - z_k\|^4 \leq \frac{\|F(z_k)\|^2}{\sigma^2}\left(\|x_k - x^\star\|^2 - \|x_{k+1} - x^\star\|^2\right).$$

From the continuity of F and the boundedness of $\{z_k\}$, we know that the sequence $\{\|F(z_k)\|\}$ is bounded. Hence, there exists a positive constant $t > 0$ such that $\|F(z_k)\| \leq t$. Furthermore,

$$\rho(2-\rho)\sum_{k=0}^{\infty}\|x_k - z_k\|^4 \leq \frac{t^2}{\sigma^2}\sum_{k=0}^{\infty}\left(\|x_k - x^\star\|^2 - \|x_{k+1} - x^\star\|^2\right) = \frac{t^2}{\sigma^2}\|x_0 - x^\star\|^2 < +\infty.$$

Hence,

$$\lim_{k\to\infty}\|x_k - z_k\| = 0. \tag{3.26}$$

(3) From the nonexpansive property (3.10) of the projection operator, we have

$$\begin{aligned}\|x_k - x_{k+1}\| &= \|x_k - P_C[x_k - \rho\phi_k F(z_k)]\| \\ &\leq \|x_k - (x_k - \rho\phi_k F(z_k))\| \\ &= \|\rho\phi_k F(z_k)\| \\ &\leq \rho\|x_k - z_k\|.\end{aligned}$$

Thus, combining this with (3.26), we conclude that $\lim_{k\to\infty}\|x_k - x_{k+1}\| = 0$. □

Theorem 3.6. *Suppose that assumptions (i) and (ii) hold, and the sequence $\{x_k\}$ is generated by the YSD algorithm . Then, we have*

$$\liminf_{k\to\infty}\|F_k\| = 0. \tag{3.27}$$

Proof. Assume, for the sake of contradiction, that (3.27) does not hold. Then, there exists $\epsilon > 0$ such that

$$\|F_k\| \geq \epsilon, \quad \forall k \geq 0. \tag{3.28}$$

By (3.14), we know that

$$\|F_k\|\|d_k\| \geq -F_k^T d_k \geq \|F_k\|^2,$$

which implies

$$\|d_k\| \geq \|F_k\| \geq \epsilon, \quad \forall k \geq 0. \tag{3.29}$$

Using the expression of t_{k-1} (3.7), it is clear that

$$t_{k-1} \geq 1 - \|F_k\|^{-1} \frac{\omega_{k-1}^T d_k}{\|d_k\|^2}. \tag{3.30}$$

Therefore, from (3.8) we have:

$$\begin{aligned} y_k^T d_k &= \omega_k^T d_k + t_k \|F_k\|\|d_k\|^2 \\ &\geq \omega_k^T d_k + \left(1 - \|F_k\|^{-1} \frac{\omega_k^T d_k}{\|d_k\|^2}\right) \|F_k\|\|d_k\|^2 \\ &= \|F_k\|\|d_k\|^2 \\ &\geq \epsilon \alpha_k \|d_k\|^2. \end{aligned}$$

This implies that

$$y_{k-1}^T d_{k-1} \geq \epsilon \alpha_{k-1} \|d_{k-1}\|^2. \tag{3.31}$$

Also, using (3.8), assumption (i), Lemma 3.3 and Lemma 3.5, we have

$$\begin{aligned} \|y_k\| &\leq \|\omega_k\| + t_k \|F_k\|\|d_k\| \\ &= \|\omega_k\| + \left(1 + \|F_k\|^{-1} \max\left\{0, -\frac{\omega_k^T d_k}{\|d_k\|^2}\right\}\right) \|F_k\|\|d_k\| \\ &\leq \|\omega_k\| + \left(1 + \|F_k\|^{-1} \frac{|\omega_k^T d_k|}{\|d_k\|^2}\right) \|F_k\|\|d_k\| \\ &\leq \|\omega_k\| + \left(1 + \|F_k\|^{-1} \frac{\|\omega_k\|\|d_k\|}{\|d_k\|^2}\right) \|F_k\|\|d_k\| \\ &\leq 2\|\omega_k\| + \|F_k\|\|d_k\| \\ &\leq 2L\|x_{k+1} - x_k\| + \mu\|d_k\| \\ &\leq 2L\rho\alpha_k\|d_k\| + \mu\|d_k\| \\ &= (2L\rho\alpha_k + \mu)\|d_k\| \\ &= (2L\rho\xi_k^i + \mu)\|d_k\| \\ &\leq (2L\rho\xi + \mu)\|d_k\|. \end{aligned}$$

From the formula of (3.4) and the above results we get

$$\begin{aligned}\left|\beta_k^{RMILHS}\right| \leq \left|\beta_k^{RMIL}\right| + \left|\beta_k^{HS}\right| &= \frac{\left|F_k^T y_{k-1}\right|}{\|d_{k-1}\|^2} + \left|\frac{F_k^T y_{k-1}}{d_{k-1}^T y_{k-1}}\right| \\ &\leq \frac{\|F_k\|\|y_{k-1}\|}{\|d_{k-1}\|^2} + \frac{\|F_k\|\|y_{k-1}\|}{d_{k-1}^T y_{k-1}} \\ &\leq \frac{\mu(2L\rho\xi+\mu)}{\|d_{k-1}\|} + \frac{\mu(2L\rho\xi+\mu)}{\epsilon\alpha_{k-1}\|d_{k-1}\|} \\ &\leq \frac{\mu(2L\rho\xi+\mu)}{\epsilon} + \frac{\mu(2L\rho\xi+\mu)}{\epsilon^2\lambda} \\ &\leq \frac{\mu(2L\rho\xi+\mu)}{\epsilon} + \frac{\mu(2L\rho\xi+\mu)}{\epsilon^2\lambda},\end{aligned}$$

where λ is small enough such that $\lambda \leq \alpha_{k-1} \leq 1$.

So, for all $k \in \mathbb{N}^\star$, we have

$$\begin{aligned}\|d_k\| &= \left\|-F_k + \beta_k^{RMILHS}\left(I - \frac{F_k F_k^T}{\|F_k\|^2}\right)s_{k-1}\right\| \\ &\leq \|-F_k\| + 2\left(\left|\beta_k^{RMIL}\right| + \left|\beta_k^{HS}\right|\right)\alpha_{k-1}\|d_{k-1}\| \\ &\leq \mu + 2\left(\frac{\mu(2L\rho\xi+\mu)}{\epsilon} + \frac{\mu(2L\rho\xi+\mu)}{\epsilon^2\lambda}\right)\alpha_{k-1}\|d_{k-1}\|.\end{aligned}$$

Since (3.26) holds, it follows that for every $c_1 > 0$, there exists k_0 such that

$$\alpha_{k-1}\|d_{k-1}\| < c_1 \quad \text{for every } k > k_0.$$

$$\|d_k\| \leq \mu + 2\left(\frac{\mu(2L\rho\xi+\mu)(\epsilon^2\lambda+1)}{\epsilon^2\lambda}\right)c_1$$

Set $C = \mu + 2\left(\frac{\mu(2L\rho\xi+\mu)(\epsilon^2\lambda+1)}{\epsilon^2\lambda}\right)c_1$, we have

$$\|d_k\| \leq C. \tag{3.32}$$

If $\alpha_k \neq \xi^i$, then $\bar{\alpha}_k = \xi^{-1}\alpha_k$ does not satisfy (3.11), that is,

$$-F(x_k + \xi^{-1}\alpha_k d_k)^T d_k < \sigma\xi^{-1}\alpha_k\|d_k\|^2. \tag{3.33}$$

Combining (3.33) with (3.14), we have

$$\begin{aligned}\|F_k\|^2 &= -F_k^T d_k \\ &= \left(F(x_k + \xi^{-1}\alpha_k) - F(x_k)\right)^T d_k - F(x_k + \xi^{-1}\alpha_k)^T d_k \\ &\leq \xi^{-1}\alpha_k L\|d_k\|^2 + \xi^{-1}\alpha_k\sigma\|d_k\|^2 \\ &= \xi^{-1}\alpha_k(L+\sigma)\|d_k\|^2.\end{aligned} \tag{3.34}$$

Since F is Lipschitz continuous, the above inequality holds. Thus, from (3.34),

$$\alpha_k \geq \frac{\xi}{L+\sigma}\frac{\|F_k\|^2}{\|d_k\|^2}.$$

$$\begin{aligned}\alpha_k\|d_k\| &\geq \frac{\xi}{L+\sigma}\frac{\|F_k\|^2}{\|d_k\|^2}\|d_k\| \\ &= \frac{\xi}{L+\sigma}\frac{\|F_k\|^2}{\|d_k\|} \\ &\geq \frac{\xi\epsilon^2}{(L+\sigma)C}.\end{aligned}$$

The last inequality yields a contradiction with (3.26). Consequently, (3.27) holds. The proof is complete.

□

3.4 Numerical experiments

In this section, we present the benchmark results of the YSD algorithm using the performance profile introduced by Dolan and Moré [20]. To evaluate the performance of the algorithm, we selected several test functions from [37, 46] with different dimensions and initial points.

The YSD algorithm is compared against four well-known methods, namely CGD, PCG, PDY and MCHGC as described in [45, 46, 69]. For our method, we set the parameters $\xi = 0.6$, $\sigma = 0.0001$ and $\rho = 1.8$. The parameters for the competing methods were chosen according to their respective recommendations. The tolerance for convergence was set to Tol $= 10^{-6}$ and the stopping criteria were defined as either $\|F_k\| \leq$ Tol or the iteration number exceeding 2000.

In the table of results, we designate by:

- Dim: Problem dimension.
- InitPoint: Initial point (x_1 to x_9).
- YSD, CGD, PCG, PDY, MCHGC: Algorithm names.
- IT: Number of iterations.
- FE: Number of function evaluations.

- CPU: Computing time (seconds).
- Norm: Norm of the gradient.
- F: Failed execution.

Let $F(x) = (f_1(x), f_2(x), \ldots, f_n(x))^T$, where the initial points for these problems are defined as follows

$$x_1 = (1, \ldots, 1)^T, \quad x_2 = (0.1, \ldots, 0.1)^T, \quad x_3 = \left(\frac{1}{2}, \ldots, \frac{1}{2n}\right)^T,$$

$$x_4 = \left(1 - \frac{1}{n}, \ldots, n-1\right)^T, \quad x_5 = \left(0, \frac{1}{n}, \ldots, \frac{n-1}{n}\right)^T,$$

$$x_6 = \left(1, \frac{1}{2}, \ldots, \frac{1}{n}\right)^T, \quad x_7 = \left(\frac{n-1}{n}, \frac{n-2}{n}, \ldots, 0\right)^T, \quad x_8 = \left(\frac{1}{n}, \frac{2}{n}, \ldots, 1\right)^T,$$

$$x_9 = \text{rand}(n, 1).$$

To evaluate the performance of the methods, we considered the following benchmark test problems

Function 1. Exponential Function

Constraint set: $C = \mathbb{R}^n_+$.

$$f_1(x) = e^{x_1} - 1,$$

$$f_i(x) = e^{x_i} + x_i - 1, \quad \text{for } i = 2, 3, \ldots, n.$$

Function 2. Modified Logarithmic Function

Constraint set: $C = \{x \in \mathbb{R}^n : \sum_{i=1}^n x_i \leq n,\ x_i > -1,\ i = 1, 2, \ldots, n\}$.

$$f_i(x) = \ln(x_i + 1) - \frac{x_i}{n}, \quad i = 1, 2, \ldots, n.$$

Function 3. Nonsmooth Function

Constraint set: $C = \mathbb{R}^n_+$.

$$f_i(x) = 2x_i - \sin|x_i|, \quad i = 1, 2, \ldots, n.$$

Function 4. Strictly Convex Function

Constraint set: $C = \mathbb{R}^n_+$.

$$f_i(x) = e^{x_i} - 1, \quad i = 1, 2, \ldots, n.$$

Function 5. Tridiagonal Exponential Function

Constraint set: $C = \mathbb{R}^n_+$.

$$f_1(x) = x_1 - e^{\cos(h(x_1+x_2))},$$

$$f_i(x) = x_i - e^{\cos(h(x_{i-1}+x_i+x_{i+1}))}, \quad \text{for } 2 \le i \le n-1,$$

$$f_n(x) = x_n - e^{\cos(h(x_{n-1}+x_n))}, \quad \text{where } h = \frac{1}{n+1}.$$

Function 6. Nonsmooth Function

Constraint set: $C = \{x \in \mathbb{R}^n : \sum_{i=1}^n x_i \le n,\ x_i \ge -1,\ 1 \le i \le n\}$.

$$f_i(x) = x_i - \sin|x_i - 1|, \quad i = 1, 2, \ldots, n.$$

Function 7. Discrete Boundary Value Problem

Constraint set: $C = \mathbb{R}^n_+$.

$$f_1(x) = 2x_1 + 0.5h^2(x_1 + h)^3 - x_2,$$

$$f_i(x) = 2x_i + 0.5h^2(x_i + ih)^3 - x_{i-1} + x_{i+1}, \quad \text{for } i = 2, 3, \ldots, n-1,$$

$$f_n(x) = 2x_n + 0.5h^2(x_n + nh)^3 - x_{n-1}, \text{ where } h = \frac{1}{n+1}.$$

Function 8. Penalty 1 Function

Constraint set: $C = \mathbb{R}^n_+$.

$$t_i = \sum_{i=1}^n x_i^2, \quad c = 10^{-5},$$

$$f_i(x) = 2c(x_i - 1) + 4(t_i - 0.25)x_i, \quad i = 1, 2, \ldots, n.$$

Function 9. Linear Function

Constraint set: $C = \mathbb{R}^n_+$.

$$f_i(x) = 8\left(\frac{1}{2}x_i - 1\right), \quad i = 1, 2, \ldots, n.$$

Function 10. Exponential-Trigonometric Function

Constraint set: $C = \mathbb{R}^n_+$.

$$f_i(x) = e^{x_i^2} + 3\sin x_i \cos x_i - 1, \quad \text{for } i = 1, 2, \ldots, n.$$

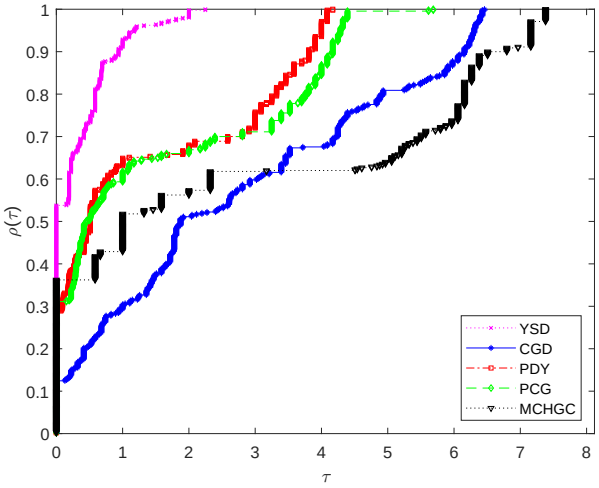


Figure 3.1: Performance profile based on the iterations number.

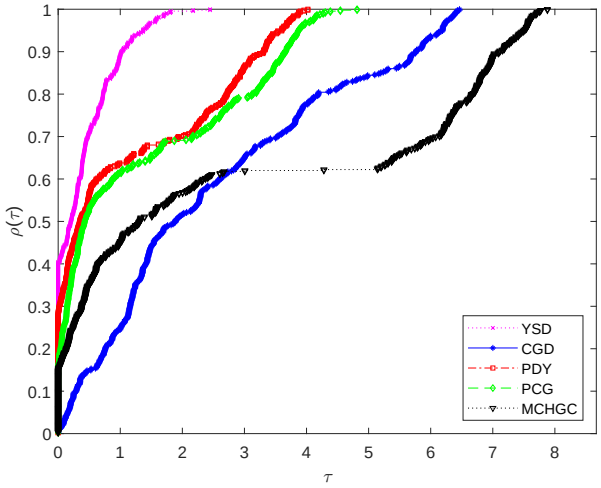


Figure 3.2: Performance profile based on CPU time.

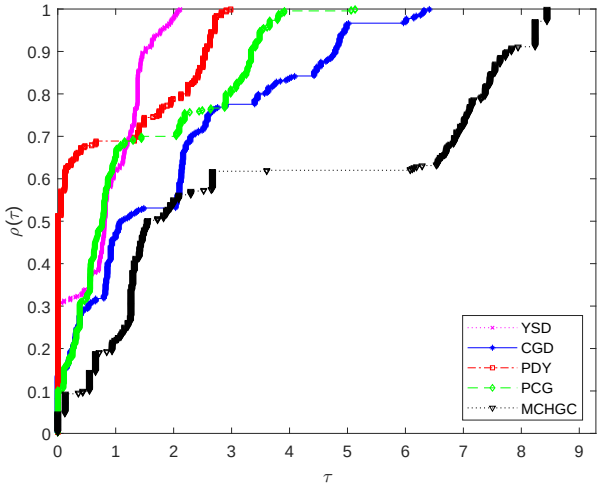


Figure 3.3: Performance profile based on the function evaluations.

Table 3.1: Numerical results of Function 1.

Dim	InitPoint	YSD				CGD				PCG				PDY				MCHGC			
		IT	FE	CPU	Norm	IT	FT	CPU	Norm	IT	FE	CPU	Norm	IT	FE	CPU	Norm	IT	FE	CPU	Norm
1000	x1	**1**	**12**	**0.01187**	**0**	50	204	0.02269	8.95585e-07	16	85	0.01655	3.81897e-07	15	109	0.01489	8.66166e-07	2	35	0.01652	0
	x2	**1**	**12**	**0.00043**	**0**	42	172	0.00977	9.37080e-07	14	75	0.00344	4.99165e-07	14	102	0.00366	6.18523e-07	2	33	0.00152	0
	x3	2	22	0.01323	0	57	232	0.01024	8.54099e-07	**14**	**74**	**0.00219**	**5.15863e-07**	23	164	0.00387	6.19382e-07	8	126	0.00219	2.55953e-08
	x4	**1**	**19**	**0.00167**	**0**	1	13	0.00181	0	1	13	0.00179	0	1	14	0.00174	0	1	19	0.00177	0
	x5	3	32	0.00261	0	34	140	0.00515	8.29111e-07	15	80	0.00224	8.14696e-07	19	132	0.00380	5.00132e-07	**2**	**35**	**0.00056**	**0**
	x6	**2**	**22**	**0.00048**	**0**	59	240	0.00868	8.70004e-07	16	85	0.00240	4.67157e-07	23	164	0.00358	7.54291e-07	3	49	0.00068	0
	x7	3	32	0.00066	0	52	212	0.00923	8.05625e-07	15	80	0.00337	8.41412e-07	19	132	0.00431	5.00817e-07	**2**	**35**	**0.00079**	**0**
	x8	3	32	0.00093	0	34	140	0.00794	8.30447e-07	15	80	0.00328	8.15774e-07	19	132	0.00282	5.02033e-07	**2**	**35**	**0.00059**	**0**
	x9	3	32	0.00066	0	44	180	0.00688	8.31286e-07	15	80	0.00238	8.10690e-07	19	132	0.00297	5.05890e-07	**2**	**35**	**0.00058**	**0**
5000	x1	**1**	**12**	**0.00148**	**0**	48	196	0.03560	9.16268e-07	16	85	0.01203	8.48826e-07	16	116	0.01388	6.44651e-07	2	35	0.00347	0
	x2	**1**	**12**	**0.00105**	**0**	40	164	0.03139	9.85809e-07	15	80	0.01330	3.67486e-07	15	109	0.01357	4.55468e-07	2	33	0.00237	0
	x3	2	22	0.00216	0	57	232	0.04352	8.53926e-07	**14**	**74**	**0.01082**	**5.16007e-07**	23	164	0.01833	6.20540e-07	5	78	0.00558	0
	x4	**1**	**19**	**0.01801**	**0**	1	13	0.01786	0	1	13	0.01671	0	1	14	0.01683	0	1	19	0.02410	0
	x5	3	32	0.00343	0	36	148	0.02794	6.67758e-07	16	85	0.01343	6.03649e-07	19	137	0.01484	3.89708e-07	**2**	**35**	**0.00273**	**0**
	x6	**2**	**22**	**0.00204**	**0**	59	240	0.04638	8.69854e-07	16	85	0.01258	4.67114e-07	23	164	0.01818	7.53851e-07	3	49	0.00353	0
	x7	3	32	0.00345	0	50	204	0.03995	8.19716e-07	16	85	0.01372	6.07647e-07	19	137	0.01535	3.89820e-07	**2**	**35**	**0.00275**	**0**
	x8	3	32	0.00318	0	36	148	0.02872	6.67922e-07	16	85	0.01258	6.03809e-07	19	137	0.01547	3.90004e-07	**2**	**35**	**0.00271**	**0**
	x9	3	32	0.00289	0	48	196	0.03669	9.23067e-07	16	85	0.01284	6.07628e-07	19	137	0.01671	3.90483e-07	**2**	**35**	**0.00305**	**0**
10000	x1	**1**	**12**	**0.00310**	**0**	47	192	0.07012	9.59688e-07	17	90	0.02816	4.01797e-07	16	116	0.02397	9.11038e-07	2	35	0.00583	0
	x2	**1**	**12**	**0.00188**	**0**	40	164	0.05910	8.36948e-07	15	80	0.02020	5.18236e-07	15	109	0.02500	6.42650e-07	2	33	0.00450	0
	x3	2	22	0.00415	0	57	232	0.07760	8.53905e-07	**14**	**74**	**0.02180**	**5.16025e-07**	23	164	0.03528	6.20685e-07	5	78	0.01079	0
	x4	**1**	**19**	**0.02132**	**0**	1	13	0.02230	0	1	13	0.02033	0	1	14	0.01915	0	1	19	0.02854	0
	x5	3	32	0.00575	0	36	148	0.05327	9.44414e-07	16	85	0.02235	8.53745e-07	19	137	0.02593	5.51170e-07	**2**	**35**	**0.00553**	**0**
	x6	**2**	**22**	**0.00370**	**0**	59	240	0.07930	8.69835e-07	16	85	0.02503	4.67108e-07	23	164	0.03584	7.53796e-07	3	49	0.00643	0
	x7	3	32	0.00581	0	49	200	0.06835	8.53808e-07	16	85	0.02441	8.56574e-07	19	137	0.02748	5.51249e-07	**2**	**35**	**0.00520**	**0**
	x8	3	32	0.00567	0	36	148	0.05661	9.44527e-07	16	85	0.02390	8.53859e-07	19	137	0.02848	5.51380e-07	**2**	**35**	**0.00473**	**0**
	x9	3	32	0.00603	0	49	200	0.07165	8.34848e-07	16	85	0.02489	8.53622e-07	19	137	0.03216	5.49154e-07	**2**	**35**	**0.00572**	**0**
50000	x1	**1**	**12**	**0.01147**	**0**	46	188	0.30281	8.09774e-07	17	90	0.12242	8.97905e-07	17	123	0.12030	6.81115e-07	2	35	0.02905	0
	x2	**1**	**12**	**0.00760**	**0**	39	160	0.24903	7.74756e-07	16	85	0.11010	3.82442e-07	16	116	0.12218	4.79914e-07	2	33	0.01919	0
	x3	2	22	0.01914	0	57	232	0.33898	8.53887e-07	**14**	**74**	**0.08960**	**5.16040e-07**	23	164	0.15225	6.20800e-07	5	78	0.04818	0
	x4	**1**	**19**	**0.09976**	**0**	1	13	0.09935	0	1	13	0.08828	0	1	14	0.09501	0	1	19	0.13829	0
	x5	3	32	0.02337	0	38	156	0.25947	7.60279e-07	17	90	0.11376	6.39440e-07	21	146	0.12877	5.85726e-07	**2**	**35**	**0.02120**	**0**
	x6	**2**	**22**	**0.01515**	**0**	59	240	0.35446	8.69820e-07	16	85	0.10132	4.67104e-07	23	164	0.14753	7.53753e-07	3	49	0.02868	0
	x7	3	32	0.02410	0	47	192	0.30959	8.84249e-07	17	90	0.10643	6.39862e-07	21	146	0.13107	5.85741e-07	**2**	**35**	**0.02521**	**0**
	x8	3	32	0.02448	0	38	156	0.25469	7.60297e-07	17	90	0.11392	6.39457e-07	21	146	0.13283	5.85771e-07	**2**	**35**	**0.02122**	**0**
	x9	3	32	0.02558	0	46	188	0.35079	8.19795e-07	17	90	0.13767	6.39901e-07	21	146	0.16075	5.83594e-07	**2**	**35**	**0.02418**	**0**
100000	x1	**1**	**12**	**0.02692**	**0**	45	184	0.62989	8.78571e-07	18	95	0.25642	4.21685e-07	17	123	0.26597	9.63123e-07	2	35	0.05658	0
	x2	**1**	**12**	**0.02093**	**0**	38	156	0.54890	9.34911e-07	16	85	0.24257	5.40700e-07	16	116	0.25429	6.78461e-07	2	33	0.04177	0
	x3	2	22	0.04093	0	57	232	0.79217	8.53885e-07	**14**	**74**	**0.19791**	**5.16041e-07**	23	164	0.33761	6.20815e-07	5	78	0.11322	0
	x4	**1**	**19**	**0.20829**	**0**	1	13	0.20034	0	1	13	0.18869	0	1	14	0.19130	0	1	19	0.27662	0
	x5	3	32	0.05304	0	39	160	0.55441	6.45123e-07	17	90	0.25741	9.04311e-07	21	146	0.30846	8.27860e-07	**2**	**35**	**0.04777**	**0**
	x6	**2**	**22**	**0.03843**	**0**	59	240	0.80435	8.69818e-07	16	85	0.23337	4.67103e-07	23	164	0.34395	7.53747e-07	3	49	0.06771	0
	x7	3	32	0.05834	0	46	188	0.67999	9.37718e-07	17	90	0.26483	9.04609e-07	21	146	0.31599	8.27871e-07	**2**	**35**	**0.04990**	**0**
	x8	3	32	0.05382	0	39	160	0.58319	6.45131e-07	17	90	0.25348	9.04323e-07	21	146	0.30843	8.27891e-07	**2**	**35**	**0.05762**	**0**
	x9	3	32	0.06200	0	39	160	0.58605	7.17911e-07	17	90	0.26692	9.04812e-07	21	146	0.35776	8.26759e-07	**2**	**35**	**0.05331**	**0**

Table 3.2: Numerical results of Function 2.

Dim	InitPoint	YSD				CGD				PCG				PDY				MCHGC			
		IT	FE	CPU	Norm	IT	FT	CPU	Norm	IT	FE	CPU	Norm	IT	FE	CPU	Norm	IT	FE	CPU	Norm
1000	x1	16	135	0.02247	9.79959e-07	65	262	0.02999	8.21116e-07	**14**	**58**	**0.01741**	**8.29181e-07**	17	103	0.01679	4.75098e-07	7	112	0.01439	8.78079e-07
	x2	17	142	0.00801	9.25575e-07	29	118	0.00852	9.97892e-07	11	47	0.00314	6.60058e-07	8	49	0.00219	7.76974e-07	**6**	**101**	**0.00281**	**2.47237e-07**
	x3	14	136	0.00737	9.72721e-07	53	214	0.01604	8.64409e-07	**12**	**50**	**0.00329**	**6.71268e-07**	15	90	0.00463	6.77180e-07	7	116	0.00363	5.82025e-07
	x4	**1**	**10**	**0.00099**	**0**	1	4	0.00054	0	1	4	0.00051	0	1	5	0.00051	0	1	10	0.00067	0
	x5	12	109	0.00534	3.05909e-07	63	254	0.01858	9.25776e-07	14	58	0.00391	5.24989e-07	17	102	0.00499	7.84273e-07	**10**	**154**	**0.00429**	**3.25457e-07**
	x6	**8**	**75**	**0.00331**	**5.66141e-07**	56	226	0.01777	8.01939e-07	13	54	0.00375	3.50856e-07	16	96	0.00541	4.25882e-07	8	131	0.00572	2.43338e-07
	x7	14	128	0.00699	4.04823e-07	63	254	0.02187	9.24406e-07	14	58	0.00452	5.24096e-07	17	102	0.00494	7.82371e-07	**11**	**161**	**0.00482**	**6.40296e-07**
	x8	14	129	0.00585	7.33720e-07	63	254	0.01924	9.27151e-07	14	58	0.00388	5.25884e-07	17	102	0.00484	7.86180e-07	**10**	**156**	**0.00479**	**4.01289e-07**
	x9	16	143	0.00687	9.60806e-07	63	254	0.01975	8.99326e-07	14	58	0.00392	5.09520e-07	17	102	0.00558	7.65807e-07	**10**	**159**	**0.00451**	**3.40145e-07**
5000	x1	**8**	**72**	**0.01755**	**2.86964e-07**	68	274	0.10653	9.18953e-07	15	62	0.02096	5.98489e-07	18	109	0.02691	4.07439e-07	8	128	0.01780	3.30136e-07
	x2	**6**	**55**	**0.01558**	**5.61624e-07**	32	130	0.04865	8.41291e-07	12	51	0.01913	4.66568e-07	9	55	0.01269	5.04006e-07	6	101	0.01716	6.52963e-07
	x3	**9**	**82**	**0.01958**	**6.27408e-07**	53	214	0.08241	8.99015e-07	12	50	0.01792	7.37581e-07	15	91	0.02303	4.62802e-07	7	116	0.01643	6.11957e-07
	x4	**1**	**10**	**0.00175**	**0**	1	4	0.00145	0	1	4	0.00135	0	1	5	0.00135	0	1	10	0.00138	0
	x5	15	138	0.03216	4.81950e-07	67	270	0.10172	8.49363e-07	15	62	0.02191	3.84461e-07	18	108	0.02695	6.81252e-07	**11**	**173**	**0.02539**	**3.19234e-07**
	x6	**10**	**93**	**0.01942**	**8.82981e-08**	56	226	0.08560	8.24803e-07	13	54	0.01938	3.69477e-07	16	96	0.02187	4.50255e-07	8	131	0.01752	8.74739e-07
	x7	15	138	0.03270	4.63507e-07	67	270	0.09931	8.49121e-07	15	62	0.02084	3.84333e-07	18	108	0.02691	6.80925e-07	**11**	**174**	**0.02455**	**2.18305e-07**
	x8	15	139	0.03291	6.94108e-07	67	270	0.09986	8.49606e-07	15	62	0.02150	3.84589e-07	18	108	0.02546	6.81579e-07	**11**	**175**	**0.02435**	**1.52211e-07**
	x9	15	138	0.03246	3.25434e-07	67	270	0.10036	8.42939e-07	15	62	0.02117	3.80996e-07	18	108	0.02696	6.72943e-07	**11**	**169**	**0.02479**	**6.57496e-07**
10000	x1	**8**	**72**	**0.03513**	**2.13805e-07**	70	282	0.20204	8.29342e-07	15	62	0.04107	8.43625e-07	18	109	0.04947	5.73857e-07	8	128	0.03435	4.61443e-07
	x2	**6**	**55**	**0.02568**	**4.32483e-07**	33	134	0.08281	9.10110e-07	12	51	0.03234	6.57403e-07	9	55	0.02277	6.81142e-07	6	101	0.02588	9.42093e-07
	x3	**9**	**82**	**0.03064**	**5.99709e-07**	53	214	0.13259	9.03399e-07	12	50	0.02973	7.46811e-07	15	91	0.03725	4.64995e-07	7	116	0.02752	6.15948e-07
	x4	**1**	**10**	**0.00303**	**0**	1	4	0.00279	0	1	4	0.00260	0	1	5	0.00256	0	1	10	0.00258	0
	x5	16	148	0.06670	7.95996e-07	68	274	0.20126	9.61486e-07	15	62	0.03971	5.43076e-07	18	108	0.04996	9.61193e-07	**11**	**170**	**0.04765**	**9.98876e-07**
	x6	**10**	**93**	**0.03598**	**6.41147e-08**	56	226	0.14398	8.27704e-07	13	54	0.03154	3.71951e-07	16	96	0.04014	4.53412e-07	8	131	0.03285	3.74833e-07
	x7	16	145	0.06467	9.28846e-07	68	274	0.19311	9.61350e-07	15	62	0.04033	5.42986e-07	18	108	0.04999	9.60963e-07	**11**	**175**	**0.04667**	**2.33588e-07**
	x8	16	148	0.06532	4.98664e-07	68	274	0.19500	9.61623e-07	15	62	0.03957	5.43166e-07	18	108	0.04945	9.61423e-07	**11**	**175**	**0.04707**	**2.06502e-07**
	x9	16	147	0.06395	2.06127e-07	68	274	0.19528	9.59221e-07	15	62	0.04120	5.41636e-07	18	108	0.04979	9.58278e-07	**11**	**175**	**0.04890**	**2.22396e-07**
50000	x1	**10**	**90**	**0.19879**	**2.36982e-08**	73	294	0.99386	9.47232e-07	16	66	0.20212	6.30647e-07	19	115	0.24291	5.05323e-07	9	144	0.17695	1.18250e-07
	x2	**8**	**73**	**0.16030**	**1.34041e-08**	37	150	0.42226	8.03134e-07	13	55	0.16180	4.97582e-07	10	61	0.11188	5.79493e-07	7	117	0.13947	2.47659e-07
	x3	**9**	**82**	**0.14039**	**6.03804e-07**	53	214	0.59859	9.06916e-07	12	50	0.12714	7.54366e-07	15	91	0.16092	4.66749e-07	7	116	0.11897	6.19180e-07
	x4	**1**	**10**	**0.01377**	**0**	1	4	0.01237	0	1	4	0.01101	0	1	5	0.01110	0	1	10	0.01109	0
	x5	16	157	0.33545	4.83790e-07	72	290	0.97119	8.80798e-07	16	66	0.19959	4.07281e-07	19	114	0.24777	8.47480e-07	**12**	**191**	**0.24174**	**1.83421e-07**
	x6	**10**	**93**	**0.16131**	**5.20558e-08**	56	226	0.65094	8.30031e-07	13	54	0.13986	3.73956e-07	16	96	0.17613	4.55955e-07	8	131	0.14254	6.73391e-07
	x7	16	157	0.33321	4.82306e-07	72	290	0.96546	8.80773e-07	16	66	0.20082	4.07268e-07	19	114	0.24792	8.47439e-07	**12**	**189**	**0.23836**	**7.09274e-07**
	x8	16	157	0.33288	4.85277e-07	72	290	0.96317	8.80823e-07	16	66	0.20010	4.07295e-07	19	114	0.24593	8.47521e-07	**11**	**175**	**0.22668**	**9.57224e-07**
	x9	17	166	0.34664	1.70649e-07	72	290	0.96981	8.80717e-07	16	66	0.20247	4.07399e-07	19	114	0.24609	8.49646e-07	**12**	**189**	**0.24286**	**1.17335e-07**
100000	x1	**10**	**90**	**0.42200**	**1.67537e-08**	75	302	2.13872	8.57078e-07	16	66	0.43696	8.91566e-07	19	115	0.52118	7.14257e-07	9	144	0.37997	1.67071e-07
	x2	**8**	**73**	**0.33178**	**9.47848e-09**	38	154	0.93346	9.04340e-07	13	55	0.34062	7.03419e-07	10	61	0.23816	8.15530e-07	7	117	0.30264	3.51028e-07
	x3	**9**	**82**	**0.29532**	**6.06147e-07**	53	214	1.26428	9.07356e-07	12	50	0.26845	7.55322e-07	15	91	0.34866	4.66968e-07	7	116	0.25779	6.19586e-07
	x4	**1**	**10**	**0.02765**	**0**	1	4	0.02770	0	1	4	0.02444	0	1	5	0.02296	0	1	10	0.02465	0
	x5	**18**	**178**	**0.76693**	**6.57685e-08**	73	294	2.05809	9.96321e-07	16	66	0.45171	5.75909e-07	20	120	0.53986	4.73399e-07	12	191	0.51570	1.88871e-07
	x6	**10**	**93**	**0.33684**	**5.09636e-08**	56	226	1.33964	8.30322e-07	13	54	0.29406	3.74208e-07	16	96	0.37419	4.56274e-07	8	131	0.30800	6.25419e-07
	x7	**18**	**178**	**0.77366**	**6.80305e-08**	73	294	2.14070	9.96306e-07	16	66	0.47795	5.75899e-07	20	120	0.59915	4.73388e-07	11	175	0.54069	8.85197e-07
	x8	**18**	**178**	**0.86267**	**6.37834e-08**	73	294	2.26698	9.96335e-07	16	66	0.46351	5.75918e-07	20	120	0.59950	4.73410e-07	12	191	0.56579	2.19893e-07
	x9	17	170	0.79645	8.23572e-07	73	294	2.25246	9.96515e-07	16	66	0.46748	5.76145e-07	20	120	0.59654	4.74084e-07	**12**	**191**	**0.57869**	**2.49558e-07**

Table 3.3: Numerical results of Function 3.

Dim	InitPoint	YSD				CGD				PCG				PDY				MCHGC			
		IT	FE	CPU	Norm	IT	FT	CPU	Norm	IT	FE	CPU	Norm	IT	FE	CPU	Norm	IT	FE	CPU	Norm
1000	x1	**1**	**11**	**0.00321**	**0**	78	316	0.01610	8.00867e-07	15	64	0.00576	7.56336e-07	18	111	0.00623	6.91921e-07	2	28	0.00384	0
	x2	**1**	**11**	**0.00032**	**0**	68	276	0.00930	8.22317e-07	13	56	0.00121	7.50537e-07	16	99	0.00149	5.68001e-07	2	27	0.00030	0
	x3	**1**	**11**	**0.00017**	**0**	60	244	0.00549	9.77259e-07	12	52	0.00130	4.45364e-07	14	87	0.00235	7.09954e-07	2	27	0.00042	0
	x4	8	76	0.00494	0	90	364	0.01477	9.65619e-07	23	103	0.00416	5.31215e-07	25	160	0.00297	8.45327e-07	**2**	**33**	**0.00065**	**0**
	x5	**1**	**11**	**0.00017**	**0**	75	304	0.00692	9.37292e-07	15	64	0.00133	4.63089e-07	18	111	0.00165	4.43694e-07	5	70	0.00082	0
	x6	**1**	**11**	**0.00016**	**0**	63	256	0.00776	9.53289e-07	12	52	0.00107	8.77305e-07	15	93	0.00138	5.04521e-07	3	44	0.00042	0
	x7	**1**	**11**	**0.00017**	**0**	75	304	0.00757	9.37292e-07	15	64	0.00221	4.63089e-07	18	111	0.00276	4.43694e-07	5	70	0.00100	0
	x8	**1**	**11**	**0.00024**	**0**	75	304	0.01140	9.38594e-07	15	64	0.00241	4.63748e-07	18	111	0.00372	4.44242e-07	5	70	0.00124	0
	x9	**1**	**11**	**0.00025**	**0**	75	304	0.01169	9.22827e-07	15	64	0.00189	4.56068e-07	18	111	0.00235	4.37850e-07	8	110	0.00145	0
5000	x1	**1**	**11**	**0.00074**	**0**	81	328	0.04230	9.16886e-07	16	68	0.00803	5.59715e-07	19	117	0.00940	6.11160e-07	2	28	0.00140	0
	x2	**1**	**11**	**0.00074**	**0**	71	288	0.03753	9.41444e-07	14	60	0.00650	5.51643e-07	17	105	0.00818	5.01712e-07	2	27	0.00130	0
	x3	**1**	**11**	**0.00085**	**0**	60	244	0.02976	9.77505e-07	12	52	0.00534	4.45475e-07	14	87	0.00694	7.10141e-07	2	27	0.00117	0
	x4	8	76	0.00481	0	97	392	0.06007	9.12944e-07	24	108	0.01552	9.72503e-07	28	179	0.01763	5.43987e-07	**2**	**33**	**0.00334**	**0**
	x5	**1**	**11**	**0.00070**	**0**	79	320	0.04699	8.58949e-07	16	68	0.00923	3.42841e-07	18	111	0.01349	9.92367e-07	5	70	0.00427	0
	x6	**1**	**11**	**0.00076**	**0**	63	256	0.04047	9.53551e-07	12	52	0.00636	8.77564e-07	15	93	0.00916	5.04686e-07	3	44	0.00219	0
	x7	**1**	**11**	**0.00075**	**0**	79	320	0.04214	8.58949e-07	16	68	0.00735	3.42841e-07	18	111	0.00803	9.92367e-07	5	70	0.00321	0
	x8	**1**	**11**	**0.00062**	**0**	79	320	0.04208	8.59188e-07	16	68	0.00722	3.42939e-07	18	111	0.00876	9.92612e-07	5	70	0.00318	0
	x9	**1**	**11**	**0.00064**	**0**	79	320	0.04070	8.61201e-07	16	68	0.00886	3.43639e-07	18	111	0.00868	9.95344e-07	5	70	0.00363	0
10000	x1	**1**	**11**	**0.00128**	**0**	83	336	0.07406	8.29870e-07	16	68	0.01264	7.91557e-07	19	117	0.01907	8.64238e-07	2	28	0.00560	0
	x2	**1**	**11**	**0.00177**	**0**	73	296	0.07502	8.52097e-07	14	60	0.01260	7.80141e-07	17	105	0.01543	7.09440e-07	2	27	0.00300	0
	x3	**1**	**11**	**0.00133**	**0**	60	244	0.06957	9.77536e-07	12	52	0.01281	4.45488e-07	14	87	0.01725	7.10164e-07	2	27	0.00270	0
	x4	8	76	0.01201	0	100	404	0.15846	8.97729e-07	26	117	0.04353	6.74311e-07	29	186	0.04995	4.17812e-07	**2**	**33**	**0.00872**	**0**
	x5	**1**	**11**	**0.00223**	**0**	80	324	0.09638	9.71861e-07	16	68	0.01823	4.84885e-07	19	117	0.02345	5.54463e-07	5	70	0.00709	0
	x6	**1**	**11**	**0.00132**	**0**	63	256	0.07174	9.53583e-07	12	52	0.01395	8.77597e-07	15	93	0.01667	5.04706e-07	3	44	0.00428	0
	x7	**1**	**11**	**0.00148**	**0**	80	324	0.10216	9.71861e-07	16	68	0.01795	4.84885e-07	19	117	0.02259	5.54463e-07	5	70	0.00770	0
	x8	**1**	**11**	**0.00144**	**0**	80	324	0.09659	9.71995e-07	16	68	0.02021	4.84954e-07	19	117	0.02430	5.54531e-07	5	70	0.00755	0
	x9	**1**	**11**	**0.00151**	**0**	80	324	0.09925	9.65182e-07	16	68	0.01864	4.81542e-07	19	117	0.02231	5.51436e-07	5	70	0.00856	0
50000	x1	**1**	**11**	**0.00487**	**0**	86	348	0.29692	9.50091e-07	17	72	0.06086	5.93139e-07	20	123	0.07337	7.63353e-07	2	28	0.01107	0
	x2	**1**	**11**	**0.00346**	**0**	76	308	0.29467	9.75538e-07	15	64	0.04789	5.87502e-07	18	111	0.06399	6.26633e-07	2	27	0.00821	0
	x3	**1**	**11**	**0.00341**	**0**	60	244	0.19313	9.77560e-07	12	52	0.03369	4.45500e-07	14	87	0.04060	7.10183e-07	2	27	0.00707	0
	x4	8	76	0.03379	0	107	432	0.53646	8.51703e-07	27	122	0.15545	8.11009e-07	32	205	0.19154	4.44254e-07	**2**	**33**	**0.03521**	**0**
	x5	**1**	**11**	**0.00393**	**0**	84	340	0.30020	8.90173e-07	17	72	0.05327	3.63405e-07	20	123	0.06995	4.89780e-07	5	70	0.01950	0
	x6	**1**	**11**	**0.00422**	**0**	63	256	0.21179	9.53609e-07	12	52	0.03659	8.77623e-07	15	93	0.04666	5.04723e-07	3	44	0.01139	0
	x7	**1**	**11**	**0.00371**	**0**	84	340	0.31809	8.90173e-07	17	72	0.06426	3.63405e-07	20	123	0.07263	4.89780e-07	5	70	0.02069	0
	x8	**1**	**11**	**0.00379**	**0**	84	340	0.30300	8.90198e-07	17	72	0.05450	3.63415e-07	20	123	0.06879	4.89792e-07	5	70	0.02281	0
	x9	**1**	**11**	**0.00403**	**0**	84	340	0.33654	8.93171e-07	17	72	0.05616	3.64587e-07	20	123	0.06710	4.91260e-07	5	70	0.02756	0
100000	x1	**1**	**11**	**0.01315**	**0**	88	356	1.02596	8.59924e-07	17	72	0.20890	8.38825e-07	21	129	0.23586	4.26522e-07	2	28	0.03253	0
	x2	**1**	**11**	**0.01212**	**0**	78	316	0.77772	8.82956e-07	15	64	0.14615	8.30853e-07	18	111	0.18466	8.86121e-07	2	27	0.02685	0
	x3	**1**	**11**	**0.00930**	**0**	60	244	0.53641	9.77563e-07	12	52	0.09869	4.45501e-07	14	87	0.12974	7.10186e-07	2	27	0.01917	0
	x4	8	76	0.08954	0	110	444	1.46877	8.31928e-07	29	131	0.43092	5.81217e-07	32	206	0.49973	6.31320e-07	**2**	**33**	**0.08002**	**0**
	x5	**1**	**11**	**0.01187**	**0**	86	348	0.86997	8.05699e-07	17	72	0.16359	5.13935e-07	20	123	0.20528	6.92567e-07	5	70	0.06253	0
	x6	**1**	**11**	**0.01194**	**0**	63	256	0.56493	9.53613e-07	12	52	0.09897	8.77626e-07	15	93	0.13783	5.04725e-07	3	44	0.03269	0
	x7	**1**	**11**	**0.01291**	**0**	86	348	0.85048	8.05699e-07	17	72	0.16336	5.13935e-07	20	123	0.20671	6.92567e-07	5	70	0.05943	0
	x8	**1**	**11**	**0.01045**	**0**	86	348	0.87794	8.05710e-07	17	72	0.15766	5.13943e-07	20	123	0.21136	6.92576e-07	5	70	0.05974	0
	x9	**1**	**11**	**0.01035**	**0**	86	348	0.88958	8.04551e-07	17	72	0.16727	5.13209e-07	20	123	0.21420	6.91641e-07	5	70	0.08487	0

Table 3.4: Numerical results of Function 4.

Dim	InitPoint	YSD				CGD				PCG				PDY				MCHGC			
		IT	FE	CPU	Norm	IT	FT	CPU	Norm	IT	FE	CPU	Norm	IT	FE	CPU	Norm	IT	FE	CPU	Norm
1000	x1	**1**	**12**	**0.00116**	**0**	75	304	0.01473	9.78107e-07	14	60	0.00288	7.33959e-07	16	99	0.00323	6.51108e-07	2	32	0.00072	0
	x2	**1**	**11**	**0.00057**	**0**	67	272	0.01735	9.67891e-07	13	56	0.00297	7.24009e-07	16	99	0.00363	5.07263e-07	2	27	0.00059	0
	x3	**1**	**11**	**0.00028**	**0**	58	236	0.00844	8.84763e-07	11	48	0.00191	8.39422e-07	13	81	0.00233	9.73918e-07	9	134	0.00240	0
	x4	2	36	0.00746	0	2	25	0.00564	0	2	25	0.00459	0	2	27	0.00385	0	**1**	**19**	**0.00159**	**0**
	x5	11	104	0.00214	8.98673e-07	74	300	0.00955	8.10792e-07	15	64	0.00181	4.04994e-07	18	111	0.00228	6.53390e-07	**8**	**115**	**0.00149**	**0**
	x6	**2**	**20**	**0.00037**	**0**	61	248	0.00779	8.27201e-07	12	52	0.00168	5.38324e-07	14	87	0.00180	6.40146e-07	6	85	0.00104	9.26970e-10
	x7	14	131	0.00217	8.05179e-07	74	300	0.00950	8.09900e-07	15	64	0.00173	4.04190e-07	18	111	0.00226	6.47098e-07	**9**	**130**	**0.00162**	**3.32335e-07**
	x8	14	128	0.00220	9.33325e-07	74	300	0.00997	8.11683e-07	15	64	0.00181	4.05799e-07	18	111	0.00218	6.59798e-07	**4**	**58**	**0.00082**	**0**
	x9	15	143	0.00336	6.82213e-07	74	300	0.00953	8.24469e-07	15	64	0.00198	4.06917e-07	18	111	0.00242	6.80363e-07	**6**	**89**	**0.00164**	**0**
5000	x1	**1**	**12**	**0.00109**	**0**	79	320	0.05761	8.96200e-07	15	64	0.01011	5.52243e-07	17	105	0.01157	5.75344e-07	2	32	0.00272	0
	x2	**1**	**11**	**0.00083**	**0**	71	288	0.04947	8.86840e-07	14	60	0.00946	5.32636e-07	17	105	0.01125	4.48249e-07	2	27	0.00180	0
	x3	**1**	**11**	**0.00083**	**0**	58	236	0.04119	8.85416e-07	11	48	0.00742	8.40102e-07	13	81	0.00896	9.74734e-07	**4**	**54**	**0.00374**	**0**
	x4	2	36	0.04147	0	2	25	0.03915	0	2	25	0.03328	0	2	27	0.03470	0	**1**	**19**	**0.02256**	**0**
	x5	15	140	0.01222	3.43571e-07	77	312	0.05426	9.28746e-07	15	64	0.00870	9.06314e-07	19	117	0.01176	5.79390e-07	**7**	**105**	**0.00688**	**7.71923e-07**
	x6	**2**	**20**	**0.00159**	**0**	61	248	0.04018	8.27908e-07	12	52	0.00739	5.38919e-07	14	87	0.00900	6.40819e-07	6	85	0.00497	2.54199e-09
	x7	15	142	0.01267	1.08124e-07	77	312	0.05713	9.28542e-07	15	64	0.00903	9.05954e-07	19	117	0.01149	5.78262e-07	**9**	**138**	**0.00863**	**0**
	x8	16	150	0.01255	1.74722e-07	77	312	0.05309	9.28950e-07	15	64	0.00984	9.06674e-07	19	117	0.01062	5.80522e-07	**4**	**58**	**0.00419**	**0**
	x9	21	201	0.02000	6.08620e-08	77	312	0.05682	9.30151e-07	15	64	0.01003	9.08734e-07	19	117	0.01399	5.78603e-07	**4**	**58**	**0.00439**	**0**
10000	x1	**1**	**12**	**0.00254**	**0**	81	328	0.14385	8.11188e-07	15	64	0.02383	7.81029e-07	17	105	0.02760	8.13622e-07	2	32	0.00569	0
	x2	**1**	**11**	**0.00191**	**0**	73	296	0.12228	8.02716e-07	14	60	0.02064	7.53298e-07	17	105	0.02659	6.33852e-07	2	27	0.00412	0
	x3	**1**	**11**	**0.00184**	**0**	58	236	0.09234	8.85497e-07	11	48	0.01517	8.40187e-07	13	81	0.01950	9.74836e-07	**4**	**54**	**0.00805**	**0**
	x4	2	36	0.05080	0	2	25	0.05154	0	2	25	0.04198	0	2	27	0.04276	0	**1**	**19**	**0.03009**	**0**
	x5	**4**	**43**	**0.00868**	**0**	79	320	0.13541	8.40669e-07	16	68	0.02354	4.25376e-07	19	117	0.02684	8.19704e-07	4	58	0.00942	0
	x6	**2**	**20**	**0.00349**	**0**	61	248	0.09821	8.27996e-07	12	52	0.01688	5.38994e-07	14	87	0.02160	6.40903e-07	6	85	0.01303	2.79689e-09
	x7	**4**	**43**	**0.01034**	**0**	79	320	0.12983	8.40577e-07	16	68	0.02005	4.25291e-07	19	117	0.01981	8.18905e-07	11	161	0.01845	2.22045e-16
	x8	**4**	**43**	**0.00701**	**0**	79	320	0.09994	8.40762e-07	16	68	0.01670	4.25460e-07	19	117	0.02265	8.20505e-07	4	58	0.00737	0
	x9	**4**	**43**	**0.00659**	**0**	79	320	0.10673	8.42462e-07	16	68	0.02145	4.25790e-07	19	117	0.02706	8.18842e-07	4	58	0.00857	0
50000	x1	**1**	**12**	**0.00809**	**0**	84	340	0.47484	9.28740e-07	16	68	0.08962	5.78190e-07	18	111	0.09532	7.18677e-07	2	32	0.01993	0
	x2	**1**	**11**	**0.00572**	**0**	76	308	0.42688	9.19039e-07	15	64	0.08423	5.67096e-07	18	111	0.09166	5.59895e-07	2	27	0.01279	0
	x3	**1**	**11**	**0.00557**	**0**	58	236	0.30607	8.85563e-07	11	48	0.04795	8.40256e-07	13	81	0.06835	9.74918e-07	**4**	**54**	**0.02462**	**0**
	x4	2	36	0.21556	0	2	25	0.20694	0	2	25	0.18078	0	2	27	0.18654	0	**1**	**19**	**0.13204**	**0**
	x5	16	156	0.09841	7.69338e-07	82	332	0.47437	9.62524e-07	16	68	0.08016	9.51245e-07	20	123	0.08673	7.24304e-07	**4**	**58**	**0.03688**	**0**
	x6	**2**	**20**	**0.01076**	**0**	61	248	0.33534	8.28067e-07	12	52	0.05449	5.39053e-07	14	87	0.06537	6.40970e-07	6	85	0.04115	3.00968e-09
	x7	**9**	**94**	**0.06328**	**0**	82	332	0.46661	9.62502e-07	16	68	0.07293	9.51207e-07	20	123	0.09081	7.24162e-07	4	58	0.03686	0
	x8	15	146	0.09196	6.21452e-07	82	332	0.44894	9.62545e-07	16	68	0.07107	9.51283e-07	20	123	0.08785	7.24445e-07	**4**	**58**	**0.03115**	**0**
	x9	21	201	0.15256	7.96018e-07	82	332	0.47125	9.62221e-07	16	68	0.09564	9.51895e-07	20	123	0.12036	7.21489e-07	**9**	**138**	**0.08917**	**0**
100000	x1	**1**	**12**	**0.02524**	**0**	86	348	1.40147	8.40603e-07	16	68	0.24258	8.17688e-07	19	117	0.30145	4.01573e-07	2	32	0.05136	0
	x2	**1**	**11**	**0.01597**	**0**	78	316	1.24289	8.31824e-07	15	64	0.22171	8.01999e-07	18	111	0.27857	7.91735e-07	2	27	0.04078	0
	x3	**1**	**11**	**0.01586**	**0**	58	236	0.83947	8.85571e-07	11	48	0.14639	8.40264e-07	13	81	0.18435	9.74928e-07	**4**	**54**	**0.07666**	**0**
	x4	2	36	0.50583	0	2	25	0.47791	0	2	25	0.41944	0	2	27	0.44505	0	**1**	**19**	**0.31010**	**0**
	x5	19	183	0.33246	6.79731e-07	84	340	1.32382	8.71187e-07	17	72	0.22017	4.49923e-07	21	129	0.28634	4.04733e-07	**4**	**58**	**0.09431**	**0**
	x6	**2**	**20**	**0.03122**	**0**	61	248	0.88667	8.28076e-07	12	52	0.15758	5.39061e-07	14	87	0.19925	6.40978e-07	6	85	0.11785	3.03684e-09
	x7	**16**	**158**	**0.28694**	**0**	84	340	1.33200	8.71178e-07	17	72	0.22341	4.49914e-07	21	129	0.28197	4.04694e-07	4	58	0.09678	0
	x8	**14**	**140**	**0.25322**	**3.25079e-09**	84	340	1.38592	8.71197e-07	17	72	0.27758	4.49932e-07	21	129	0.39109	4.04773e-07	4	58	0.13102	0
	x9	**16**	**160**	**0.46295**	**6.57102e-08**	84	340	1.78446	8.70480e-07	17	72	0.27075	4.48822e-07	21	129	0.33660	4.02721e-07	7	103	0.17644	0

Table 3.5: Numerical results of Function 5.

Dim	InitPoint	YSD				CGD				PCG				PDY				MCHGC			
		IT	FE	CPU	Norm	IT	FT	CPU	Norm	IT	FE	CPU	Norm	IT	FE	CPU	Norm	IT	FE	CPU	Norm
1000	x1	3	26	0.00175	0	4	13	0.00156	0	**3**	**10**	**0.00109**	**0**	4	17	0.00116	0	3	24	0.00107	0
	x2	14	114	0.00312	0	14	43	0.00267	0	**12**	**37**	**0.00196**	**0**	14	57	0.00213	0	19	136	0.00316	0
	x3	4	34	0.00097	0	5	16	0.00151	0	**4**	**13**	**0.00075**	**0**	5	21	0.00135	0	4	31	0.00122	0
	x4	**1**	**19**	**0.00173**	**0**	1	13	0.00182	0	1	13	0.00174	0	1	14	0.00168	0	1	19	0.00187	0
	x5	3	26	0.00083	0	3	10	0.00104	0	3	10	0.00111	0	**3**	**13**	**0.00118**	**0**	3	24	0.00122	0
	x6	3	26	0.00070	0	4	13	0.00161	0	**3**	**10**	**0.00059**	**0**	3	13	0.00073	0	3	24	0.00111	0
	x7	3	26	0.00078	0	3	10	0.00099	0	3	10	0.00087	0	**3**	**13**	**0.00109**	**0**	3	24	0.00121	0
	x8	3	26	0.00100	0	3	10	0.00096	0	3	10	0.00104	0	**3**	**13**	**0.00110**	**0**	3	24	0.00113	0
	x9	3	26	0.00077	0	3	10	0.00120	0	3	10	0.00124	0	**3**	**13**	**0.00114**	**0**	3	24	0.00113	0
5000	x1	3	26	0.00578	0	4	13	0.00742	0	**3**	**10**	**0.00471**	**0**	4	17	0.00551	0	3	24	0.00481	0
	x2	14	114	0.01686	0	14	43	0.01453	0	**12**	**37**	**0.01041**	**0**	14	57	0.01034	0	19	136	0.01223	0
	x3	4	34	0.00346	0	5	16	0.00533	0	**4**	**13**	**0.00227**	**0**	5	21	0.00505	0	4	31	0.00461	0
	x4	**1**	**19**	**0.01771**	**0**	1	13	0.01780	0	1	13	0.01680	0	1	14	0.01700	0	1	19	0.02409	0
	x5	3	26	0.00252	0	3	10	0.00397	0	3	10	0.00421	0	**3**	**13**	**0.00450**	**0**	3	24	0.00493	0
	x6	3	26	0.00217	0	4	13	0.00641	0	**3**	**10**	**0.00216**	**0**	3	13	0.00187	0	3	24	0.00420	0
	x7	3	26	0.00273	0	3	10	0.00521	0	3	10	0.00446	0	**3**	**13**	**0.00450**	**0**	3	24	0.00478	0
	x8	3	26	0.00248	0	3	10	0.00427	0	3	10	0.00429	0	**3**	**13**	**0.00453**	**0**	3	24	0.00463	0
	x9	3	26	0.00339	0	3	10	0.00535	0	3	10	0.00467	0	**3**	**13**	**0.00481**	**0**	3	24	0.00489	0
10000	x1	3	26	0.00838	0	4	13	0.01040	0	**3**	**10**	**0.00714**	**0**	4	17	0.00803	0	3	24	0.00677	0
	x2	14	114	0.02411	0	14	43	0.02200	0	**12**	**37**	**0.01600**	**0**	14	57	0.01731	0	19	136	0.02077	0
	x3	4	34	0.00560	0	5	16	0.00849	0	**4**	**13**	**0.00373**	**0**	5	21	0.00740	0	4	31	0.00624	0
	x4	**1**	**19**	**0.01898**	**0**	1	13	0.02033	0	1	13	0.01895	0	1	14	0.01839	0	1	19	0.02829	0
	x5	3	26	0.00459	0	3	10	0.00598	0	3	10	0.00630	0	**3**	**13**	**0.00590**	**0**	3	24	0.00626	0
	x6	3	26	0.00335	0	4	13	0.00862	0	**3**	**10**	**0.00313**	**0**	3	13	0.00281	0	3	24	0.00540	0
	x7	3	26	0.00550	0	3	10	0.00722	0	3	10	0.00624	0	**3**	**13**	**0.00598**	**0**	3	24	0.00616	0
	x8	3	26	0.00450	0	3	10	0.00566	0	3	10	0.00629	0	**3**	**13**	**0.00626**	**0**	3	24	0.00639	0
	x9	3	26	0.00580	0	3	10	0.00718	0	3	10	0.00683	0	**3**	**13**	**0.00636**	**0**	3	24	0.00695	0
50000	x1	3	26	0.03777	0	4	13	0.05015	0	**3**	**10**	**0.03365**	**0**	4	17	0.03512	0	3	24	0.02928	0
	x2	14	114	0.11761	0	14	43	0.09497	0	**12**	**37**	**0.07062**	**0**	14	57	0.08327	0	19	136	0.10123	0
	x3	4	34	0.02416	0	5	16	0.03660	0	**4**	**13**	**0.01866**	**0**	5	21	0.02928	0	4	31	0.02650	0
	x4	**1**	**19**	**0.09549**	**0**	1	13	0.09755	0	1	13	0.09067	0	1	14	0.09139	0	1	19	0.13559	0
	x5	3	26	0.02483	0	3	10	0.02620	0	3	10	0.02743	0	**3**	**13**	**0.02613**	**0**	3	24	0.02742	0
	x6	3	26	0.01530	0	4	13	0.03494	0	**3**	**10**	**0.01181**	**0**	3	13	0.01076	0	3	24	0.02416	0
	x7	3	26	0.02216	0	3	10	0.03002	0	3	10	0.03117	0	**3**	**13**	**0.02794**	**0**	3	24	0.02638	0
	x8	3	26	0.01882	0	3	10	0.02340	0	3	10	0.02603	0	**3**	**13**	**0.02835**	**0**	3	24	0.02703	0
	x9	3	26	0.02332	0	3	10	0.03183	0	3	10	0.03129	0	**3**	**13**	**0.02827**	**0**	3	33	0.12616	0
100000	x1	3	26	0.09386	0	4	13	0.11110	0	**3**	**10**	**0.05898**	**0**	4	17	0.07560	0	3	24	0.06724	0
	x2	14	114	0.23749	0	14	43	0.20037	0	**12**	**37**	**0.14689**	**0**	14	57	0.16532	0	19	136	0.21332	0
	x3	4	34	0.05122	0	5	16	0.07021	0	**4**	**13**	**0.04344**	**0**	5	21	0.07774	0	4	31	0.06749	0
	x4	**1**	**19**	**0.20616**	**0**	1	13	0.21042	0	1	13	0.20010	0	1	14	0.19784	0	1	19	0.29643	0
	x5	3	26	0.05592	0	3	10	0.06109	0	3	10	0.06568	0	**3**	**13**	**0.06623**	**0**	3	24	0.05681	0
	x6	3	26	0.03691	0	4	13	0.07170	0	**3**	**10**	**0.02337**	**0**	3	13	0.02829	0	3	24	0.04929	0
	x7	3	26	0.04705	0	3	10	0.06803	0	3	10	0.06242	0	**3**	**13**	**0.05854**	**0**	3	24	0.06241	0
	x8	3	26	0.04666	0	3	10	0.05435	0	3	10	0.05598	0	**3**	**13**	**0.06283**	**0**	3	24	0.05299	0
	x9	3	26	0.05613	0	3	10	0.07296	0	3	10	0.05836	0	**3**	**13**	**0.06348**	**0**	3	33	0.23962	0

Table 3.6: Numerical results of Function 6.

Dim	InitPoint	YSD				CGD				PCG				PDY				MCHGC			
		IT	FE	CPU	Norm	IT	FT	CPU	Norm	IT	FE	CPU	Norm	IT	FE	CPU	Norm	IT	FE	CPU	Norm
1000	x1	11	115	0.00443	6.94e-08	37	152	0.00817	8.54e-07	**12**	**63**	**0.00281**	**5.33e-07**	17	123	0.00418	8.39e-07	F	F	F	F
	x2	11	112	0.00359	4.68e-07	37	152	0.00878	7.67e-07	**13**	**68**	**0.00273**	**9.30e-07**	15	108	0.00322	3.99e-07	F	F	F	F
	x3	13	132	0.00263	8.34e-07	37	152	0.00527	9.83e-07	**14**	**73**	**0.00255**	**5.11e-07**	15	108	0.00227	9.20e-07	F	F	F	F
	x4	19	194	0.00459	3.95e-07	38	156	0.00540	7.66e-07	**17**	**88**	**0.00254**	**3.67e-07**	16	114	0.00276	6.92e-07	F	F	F	F
	x5	14	142	0.00278	9.83e-07	36	148	0.00500	8.39e-07	**16**	**83**	**0.00228**	**4.85e-07**	17	123	0.00249	5.38e-07	F	F	F	F
	x6	13	132	0.00419	8.49e-07	37	152	0.00879	9.77e-07	**14**	**73**	**0.00330**	**5.10e-07**	16	115	0.00406	3.87e-07	F	F	F	F
	x7	13	136	0.00267	8.48e-07	36	148	0.00504	8.39e-07	**16**	**83**	**0.00281**	**4.84e-07**	17	123	0.00264	5.38e-07	F	F	F	F
	x8	13	136	0.00263	8.69e-07	36	148	0.00635	8.39e-07	**16**	**83**	**0.00228**	**4.86e-07**	17	123	0.00262	5.38e-07	F	F	F	F
	x9	13	136	0.00264	7.86e-07	36	148	0.00505	8.67e-07	**16**	**83**	**0.00235**	**5.06e-07**	17	123	0.00269	5.57e-07	F	F	F	F
5000	x1	12	122	0.01336	5.18e-07	39	160	0.02936	7.47e-07	**13**	**68**	**0.01021**	**4.26e-07**	18	130	0.01252	7.07e-07	F	F	F	F
	x2	11	115	0.01221	6.91e-08	39	160	0.02936	6.72e-07	**14**	**73**	**0.01141**	**7.61e-07**	15	108	0.01193	8.91e-07	F	F	F	F
	x3	13	136	0.01526	5.13e-07	39	160	0.03270	8.65e-07	**15**	**78**	**0.01052**	**4.12e-07**	16	115	0.01321	6.94e-07	F	F	F	F
	x4	21	217	0.02708	4.45e-07	40	164	0.02980	6.70e-07	**17**	**88**	**0.01416**	**8.22e-07**	17	121	0.01410	5.83e-07	F	F	F	F
	x5	14	146	0.01977	4.88e-07	38	156	0.03618	7.35e-07	**17**	**88**	**0.02814**	**3.92e-07**	18	130	0.02188	4.54e-07	F	F	F	F
	x6	14	142	0.02022	8.13e-07	39	160	0.03646	8.64e-07	**15**	**78**	**0.01281**	**4.11e-07**	16	115	0.01439	7.16e-07	F	F	F	F
	x7	14	146	0.01724	4.61e-07	38	156	0.03637	7.35e-07	**17**	**88**	**0.01528**	**3.92e-07**	18	130	0.01838	4.54e-07	F	F	F	F
	x8	14	145	0.01983	2.42e-07	38	156	0.03544	7.35e-07	**17**	**88**	**0.01662**	**3.92e-07**	18	130	0.01818	4.54e-07	F	F	F	F
	x9	14	142	0.01693	9.52e-07	38	156	0.03466	7.30e-07	**17**	**88**	**0.01451**	**3.90e-07**	18	130	0.01618	4.51e-07	F	F	F	F
10000	x1	12	122	0.02668	8.42e-07	40	164	0.05472	6.61e-07	**13**	**68**	**0.01858**	**6.02e-07**	19	137	0.02735	3.77e-07	F	F	F	F
	x2	11	116	0.02296	7.58e-07	39	160	0.05530	9.50e-07	**15**	**78**	**0.02202**	**3.87e-07**	16	115	0.02336	4.75e-07	F	F	F	F
	x3	13	136	0.02775	8.17e-07	40	164	0.05781	7.66e-07	**15**	**78**	**0.01986**	**5.83e-07**	16	115	0.02108	9.65e-07	F	F	F	F
	x4	23	240	0.05347	5.68e-07	40	164	0.05921	9.48e-07	**18**	**93**	**0.02555**	**4.22e-07**	17	121	0.02400	8.25e-07	F	F	F	F
	x5	15	152	0.02808	9.57e-07	39	160	0.04992	6.50e-07	**17**	**88**	**0.02323**	**5.55e-07**	18	130	0.02733	6.42e-07	F	F	F	F
	x6	13	136	0.02416	8.53e-07	40	164	0.05552	7.66e-07	**15**	**78**	**0.02042**	**5.82e-07**	16	115	0.02234	9.82e-07	F	F	F	F
	x7	15	156	0.03827	3.93e-07	39	160	0.06791	6.50e-07	**17**	**88**	**0.02898**	**5.55e-07**	18	130	0.03151	6.42e-07	F	F	F	F
	x8	15	152	0.03612	8.06e-07	39	160	0.06814	6.50e-07	**17**	**88**	**0.02852**	**5.55e-07**	18	130	0.03154	6.42e-07	F	F	F	F
	x9	15	152	0.03509	7.99e-07	39	160	0.06886	6.51e-07	**17**	**88**	**0.03001**	**5.58e-07**	18	130	0.03029	6.43e-07	F	F	F	F
50000	x1	12	126	0.10776	7.97e-07	41	168	0.24952	9.25e-07	**14**	**73**	**0.08649**	**4.93e-07**	19	137	0.11899	8.43e-07	F	F	F	F
	x2	12	122	0.10787	8.43e-07	41	168	0.25074	8.31e-07	**15**	**78**	**0.09067**	**8.66e-07**	17	122	0.11220	4.01e-07	F	F	F	F
	x3	14	145	0.12661	2.28e-07	42	172	0.26604	6.71e-07	**16**	**83**	**0.09661**	**4.75e-07**	17	122	0.10647	8.02e-07	F	F	F	F
	x4	21	220	0.21532	9.92e-07	42	172	0.26520	8.30e-07	**18**	**93**	**0.11737**	**9.44e-07**	18	128	0.12089	6.96e-07	F	F	F	F
	x5	15	156	0.13802	4.11e-07	40	164	0.24562	9.10e-07	**18**	**93**	**0.10925**	**4.51e-07**	19	137	0.12816	5.42e-07	F	F	F	F
	x6	14	146	0.12107	6.55e-07	42	172	0.26472	6.71e-07	**16**	**83**	**0.09674**	**4.75e-07**	17	122	0.11087	8.05e-07	F	F	F	F
	x7	15	156	0.12849	4.13e-07	40	164	0.24942	9.10e-07	**18**	**93**	**0.10838**	**4.51e-07**	19	137	0.12222	5.42e-07	F	F	F	F
	x8	15	156	0.13676	4.10e-07	40	164	0.25026	9.10e-07	**18**	**93**	**0.11867**	**4.51e-07**	19	137	0.12808	5.42e-07	F	F	F	F
	x9	15	156	0.16418	4.40e-07	40	164	0.30073	9.10e-07	**18**	**93**	**0.13672**	**4.52e-07**	19	137	0.14595	5.42e-07	F	F	F	F
100000	x1	13	132	0.25819	4.81e-07	42	172	0.57148	8.18e-07	**14**	**73**	**0.20498**	**6.97e-07**	20	144	0.30227	4.50e-07	F	F	F	F
	x2	12	125	0.23946	8.10e-08	42	172	0.59747	7.35e-07	**16**	**83**	**0.22730**	**4.46e-07**	17	122	0.25667	5.67e-07	F	F	F	F
	x3	14	146	0.28594	6.21e-07	42	172	0.61450	9.49e-07	**16**	**83**	**0.22829**	**6.72e-07**	18	129	0.27247	4.27e-07	F	F	F	F
	x4	21	217	0.59621	8.88e-07	43	176	0.76607	7.34e-07	**19**	**98**	**0.33897**	**4.83e-07**	18	128	0.33776	9.84e-07	F	F	F	F
	x5	16	162	0.31465	9.35e-07	41	168	0.57225	8.05e-07	**18**	**93**	**0.25088**	**6.38e-07**	19	137	0.28419	7.66e-07	F	F	F	F
	x6	14	146	0.28764	9.29e-07	42	172	0.60567	9.49e-07	**16**	**83**	**0.23020**	**6.72e-07**	18	129	0.26057	4.28e-07	F	F	F	F
	x7	16	162	0.31743	9.36e-07	41	168	0.56260	8.05e-07	**18**	**93**	**0.25909**	**6.38e-07**	19	137	0.28092	7.66e-07	F	F	F	F
	x8	16	162	0.38489	9.33e-07	41	168	0.70652	8.05e-07	**18**	**93**	**0.31471**	**6.38e-07**	19	137	0.35392	7.66e-07	F	F	F	F
	x9	16	162	0.32223	8.93e-07	41	168	0.56573	8.06e-07	**18**	**93**	**0.25571**	**6.40e-07**	19	137	0.29473	7.67e-07	F	F	F	F

Table 3.7: Numerical results of Function 7.

Dim	InitPoint	YSD				CGD				PCG				PDY				MCHGC			
		IT	FE	CPU	Norm	IT	FT	CPU	Norm	IT	FE	CPU	Norm	IT	FE	CPU	Norm	IT	FE	CPU	Norm
1000	x1	57	572	0.04601	4.88e-07	40	164	0.06445	7.61e-07	**29**	**160**	**0.04920**	**4.34e-07**	35	244	0.05041	8.16e-07	F	F	F	F
	x2	45	456	0.02634	9.35e-07	33	136	0.04189	9.17e-07	**23**	**127**	**0.02874**	**7.29e-07**	29	202	0.03354	7.75e-07	F	F	F	F
	x3	32	322	0.05436	7.70e-07	45	184	0.05636	9.51e-07	**28**	**148**	**0.03350**	**8.94e-07**	33	229	0.03801	7.58e-07	F	F	F	F
	x4	68	683	0.03548	8.71e-07	48	196	0.05984	8.75e-07	**37**	**196**	**0.04386**	**7.16e-07**	49	342	0.05738	8.75e-07	F	F	F	F
	x5	35	356	0.01636	9.15e-07	35	144	0.04514	6.70e-07	**23**	**116**	**0.02739**	**9.78e-07**	33	230	0.03822	7.48e-07	F	F	F	F
	x6	34	342	0.05722	8.05e-07	48	196	0.06018	7.81e-07	**30**	**163**	**0.03583**	**4.03e-07**	35	243	0.04064	6.78e-07	F	F	F	F
	x7	62	622	0.03402	8.53e-07	41	168	0.05062	8.03e-07	**29**	**161**	**0.03621**	**5.14e-07**	35	244	0.04083	6.85e-07	F	F	F	F
	x8	37	376	0.02231	6.96e-07	35	144	0.04594	6.71e-07	**24**	**121**	**0.02799**	**8.29e-07**	33	230	0.04135	7.49e-07	F	F	F	F
	x9	63	633	0.09724	8.11e-07	38	156	0.04648	9.08e-07	**33**	**180**	**0.04125**	**4.12e-07**	41	286	0.04798	7.53e-07	F	F	F	F
5000	x1	57	572	0.13313	4.78e-07	39	160	0.27722	8.11e-07	**27**	**148**	**0.18877**	**5.68e-07**	34	237	0.22242	7.63e-07	F	F	F	F
	x2	45	456	0.11057	9.30e-07	34	140	0.22927	6.51e-07	**21**	**115**	**0.14455**	**9.92e-07**	28	195	0.17886	8.28e-07	F	F	F	F
	x3	32	322	0.30309	7.43e-07	45	184	0.30993	9.50e-07	**28**	**148**	**0.19877**	**8.91e-07**	33	229	0.21652	7.56e-07	F	F	F	F
	x4	69	693	0.14101	2.99e-07	53	216	0.34027	7.59e-07	**41**	**211**	**0.24535**	**9.13e-07**	53	370	0.31498	7.26e-07	F	F	F	F
	x5	29	296	0.05826	9.33e-07	36	148	0.23168	8.98e-07	**23**	**116**	**0.13934**	**6.90e-07**	33	230	0.19789	6.45e-07	F	F	F	F
	x6	34	342	0.29829	7.85e-07	48	196	0.30772	7.81e-07	**30**	**163**	**0.18602**	**4.02e-07**	35	243	0.20843	6.76e-07	F	F	F	F
	x7	61	612	0.11833	7.49e-07	39	160	0.24982	9.37e-07	**27**	**149**	**0.17494**	**7.50e-07**	33	230	0.19774	7.48e-07	F	F	F	F
	x8	31	316	0.06435	6.99e-07	36	148	0.23783	8.99e-07	**23**	**116**	**0.14197**	**6.90e-07**	33	230	0.23310	6.45e-07	F	F	F	F
	x9	65	656	0.53728	7.25e-07	39	160	0.24924	8.45e-07	**34**	**184**	**0.21620**	**4.57e-07**	42	293	0.25250	9.43e-07	F	F	F	F
10000	x1	57	572	0.25504	4.76e-07	39	160	0.54773	8.29e-07	**27**	**147**	**0.36514**	**5.54e-07**	34	237	0.42990	6.81e-07	F	F	F	F
	x2	45	456	0.20434	9.30e-07	34	140	0.46886	7.71e-07	**21**	**114**	**0.29196**	**9.86e-07**	28	195	0.35330	7.68e-07	F	F	F	F
	x3	32	322	0.56252	7.40e-07	45	184	0.57225	9.50e-07	**27**	**147**	**0.33545**	**4.84e-07**	33	229	0.38982	7.56e-07	F	F	F	F
	x4	65	653	0.26067	9.13e-07	55	224	0.70147	7.70e-07	**46**	**232**	**0.54003**	**7.04e-07**	54	377	0.63778	9.01e-07	F	F	F	F
	x5	27	276	0.10023	7.46e-07	37	152	0.46975	7.62e-07	**22**	**111**	**0.26025**	**9.83e-07**	32	223	0.37894	9.58e-07	F	F	F	F
	x6	34	342	0.59265	7.82e-07	48	196	0.61029	7.81e-07	**30**	**163**	**0.36983**	**4.02e-07**	35	243	0.41312	6.76e-07	F	F	F	F
	x7	60	602	0.21149	8.87e-07	39	160	0.50298	8.35e-07	**26**	**143**	**0.33339**	**9.49e-07**	32	223	0.37879	9.72e-07	F	F	F	F
	x8	28	286	0.10411	8.16e-07	37	152	0.48092	7.62e-07	**22**	**111**	**0.29156**	**9.83e-07**	32	223	0.41866	9.58e-07	F	F	F	F
	x9	64	648	1.12496	9.87e-07	38	156	0.51246	6.45e-07	**34**	**184**	**0.46489**	**5.63e-07**	43	300	0.55670	8.39e-07	F	F	F	F
50000	x1	57	572	1.09825	4.75e-07	40	164	2.63538	7.63e-07	**25**	**135**	**1.61541**	**8.62e-07**	33	230	2.10680	8.54e-07	F	F	F	F
	x2	45	456	0.76170	9.29e-07	35	144	2.15526	8.75e-07	**21**	**113**	**1.25475**	**5.96e-07**	28	195	1.61511	6.78e-07	F	F	F	F
	x3	32	322	2.96455	7.38e-07	45	184	3.23648	9.50e-07	**27**	**147**	**1.64852**	**4.84e-07**	33	229	1.94355	7.55e-07	F	F	F	F
	x4	**14**	**216**	**0.40047**	**2.00e-10**	59	240	3.66365	9.99e-07	43	216	2.49866	6.89e-07	59	412	3.39911	6.77e-07	F	F	F	F
	x5	21	216	0.36518	7.27e-07	39	160	2.42027	6.14e-07	**22**	**111**	**1.27706**	**8.68e-07**	32	223	1.85551	8.84e-07	F	F	F	F
	x6	34	342	2.91759	7.80e-07	48	196	2.98728	7.81e-07	**30**	**158**	**1.85420**	**9.99e-07**	35	243	2.01771	6.76e-07	F	F	F	F
	x7	59	592	0.98372	7.31e-07	39	160	2.42784	8.95e-07	**26**	**142**	**1.61655**	**4.87e-07**	32	223	1.85056	7.91e-07	F	F	F	F
	x8	22	226	0.40987	7.94e-07	39	160	2.42343	6.14e-07	**22**	**111**	**1.27153**	**8.68e-07**	32	223	1.90307	8.84e-07	F	F	F	F
	x9	51	528	4.68921	8.40e-07	39	160	2.41072	9.20e-07	**35**	**178**	**2.08251**	**7.57e-07**	45	314	2.63168	6.75e-07	F	F	F	F
100000	x1	57	572	2.07528	4.75e-07	40	164	5.08344	9.92e-07	**26**	**140**	**3.24379**	**4.54e-07**	33	230	3.90210	8.17e-07	F	F	F	F
	x2	45	456	1.67506	9.29e-07	36	148	4.58647	7.25e-07	**22**	**118**	**2.72213**	**3.23e-07**	27	188	3.22139	9.99e-07	F	F	F	F
	x3	32	322	5.63048	7.38e-07	45	184	5.72417	9.50e-07	**27**	**147**	**3.35727**	**4.84e-07**	33	229	3.99944	7.55e-07	F	F	F	F
	x4	**5**	**74**	**0.43942**	**2.00e-10**	61	248	7.72708	7.79e-07	60	302	4.55522	8.30e-07	61	427	7.29801	8.27e-07	F	F	F	F
	x5	18	186	0.70396	8.37e-07	39	160	4.98921	8.68e-07	**22**	**111**	**2.60650**	**8.28e-07**	32	223	3.79439	8.64e-07	F	F	F	F
	x6	34	342	6.43670	7.80e-07	48	196	6.53590	7.81e-07	**30**	**158**	**4.16545**	**1.00e-06**	35	243	4.15677	6.76e-07	F	F	F	F
	x7	58	582	2.00826	7.30e-07	40	164	5.09308	6.73e-07	**25**	**136**	**3.15722**	**7.01e-07**	32	223	3.79744	7.51e-07	F	F	F	F
	x8	19	196	0.73968	9.07e-07	39	160	4.96883	8.68e-07	**22**	**111**	**2.61349**	**8.28e-07**	32	223	3.81384	8.64e-07	F	F	F	F
	x9	64	661	10.82011	7.12e-07	40	164	5.03482	7.05e-07	**34**	**180**	**4.18787**	**7.30e-07**	45	314	5.32321	9.49e-07	F	F	F	F

Table 3.8: Numerical results of Function 8.

Dim	InitPoint	YSD				CGD				PCG				PDY				MCHGC			
		IT	FE	CPU	Norm	IT	FT	CPU	Norm	IT	FE	CPU	Norm	IT	FE	CPU	Norm	IT	FE	CPU	Norm
1000	x1	24	263	0.002466	6.38e-07	22	98	0.001115	2.98e-07	**16**	**98**	**0.000896**	**4.67e-07**	15	112	0.001443	7.84e-07	F	F	F	F
	x2	24	263	0.002139	6.38e-07	20	80	0.001453	2.98e-07	**17**	**102**	**0.001234**	**4.67e-07**	15	111	0.001046	5.57e-07	F	F	F	F
	x3	24	254	0.002075	6.38e-07	18	71	0.001304	2.98e-07	**16**	**89**	**0.001066**	**4.67e-07**	15	103	0.001013	7.84e-07	F	F	F	F
	x4	24	263	0.001327	6.38e-07	F	F	F	F	**16**	**98**	**0.000695**	**4.67e-07**	15	112	0.000647	7.84e-07	F	F	F	F
	x5	24	263	0.001244	5.92e-07	F	F	F	F	**16**	**98**	**0.000692**	**4.71e-07**	15	112	0.000638	7.96e-07	F	F	F	F
	x6	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	x7	24	263	0.002034	5.92e-07	F	F	F	F	**16**	**98**	**0.000667**	**4.71e-07**	15	112	0.000673	7.96e-07	F	F	F	F
	x8	24	263	0.001903	6.38e-07	F	F	F	F	**16**	**98**	**0.000636**	**4.67e-07**	15	112	0.000646	7.84e-07	F	F	F	F
	x9	24	263	0.001273	6.38e-07	F	F	F	F	**16**	**98**	**0.000619**	**4.67e-07**	15	112	0.000562	7.84e-07	F	F	F	F
5000	x1	21	232	0.007159	9.64e-07	23	120	0.006176	5.52e-07	**13**	**81**	**0.003573**	**7.94e-07**	13	98	0.003579	8.20e-07	F	F	F	F
	x2	21	232	0.007243	9.64e-07	28	130	0.007121	5.52e-07	**14**	**88**	**0.003773**	**7.94e-07**	13	98	0.003588	8.20e-07	F	F	F	F
	x3	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	x4	21	232	0.007561	9.64e-07	15	68	0.003745	5.52e-07	**13**	**81**	**0.003531**	**7.94e-07**	13	98	0.003530	8.20e-07	F	F	F	F
	x5	21	232	0.007675	9.08e-07	F	F	F	F	**13**	**81**	**0.003508**	**7.95e-07**	13	98	0.003534	8.28e-07	F	F	F	F
	x6	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	x7	21	232	0.007491	9.08e-07	F	F	F	F	**13**	**81**	**0.003611**	**7.95e-07**	13	98	0.003626	8.28e-07	F	F	F	F
	x8	21	232	0.007469	9.64e-07	F	F	F	F	**13**	**81**	**0.003568**	**7.94e-07**	13	98	0.003525	8.20e-07	F	F	F	F
	x9	21	232	0.006457	9.64e-07	F	F	F	F	**13**	**81**	**0.003616**	**7.94e-07**	13	98	0.003609	8.20e-07	F	F	F	F
10000	x1	22	243	0.019742	9.66e-07	31	152	0.018208	2.92e-07	**16**	**100**	**0.012192**	**9.44e-07**	14	107	0.008524	6.62e-07	F	F	F	F
	x2	22	243	0.016952	9.66e-07	17	72	0.009028	2.92e-07	**17**	**108**	**0.010410**	**9.44e-07**	14	107	0.008910	6.62e-07	F	F	F	F
	x3	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	x4	22	243	0.016744	9.66e-07	14	64	0.007466	2.92e-07	**16**	**100**	**0.010226**	**9.44e-07**	14	107	0.008770	6.62e-07	F	F	F	F
	x5	22	243	0.017233	9.15e-07	F	F	F	F	**16**	**100**	**0.008891**	**9.44e-07**	14	107	0.008088	6.72e-07	F	F	F	F
	x6	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	x7	22	243	0.015984	9.15e-07	F	F	F	F	**16**	**100**	**0.008746**	**9.44e-07**	14	107	0.007903	6.72e-07	F	F	F	F
	x8	22	243	0.016004	9.66e-07	F	F	F	F	**16**	**100**	**0.009018**	**9.44e-07**	14	107	0.007886	6.62e-07	F	F	F	F
	x9	22	243	0.015672	9.66e-07	F	F	F	F	**16**	**100**	**0.008821**	**9.44e-07**	14	107	0.007993	6.62e-07	F	F	F	F
50000	x1	23	256	0.081138	6.50e-07	26	133	0.070585	2.19e-07	**16**	**101**	**0.041742**	**6.09e-07**	12	93	0.037627	7.87e-07	F	F	F	F
	x2	23	256	0.089763	6.50e-07	23	111	0.059796	2.19e-07	**16**	**101**	**0.042731**	**6.09e-07**	12	93	0.038409	7.87e-07	F	F	F	F
	x3	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	x4	23	256	0.084228	6.50e-07	16	73	0.042900	2.19e-07	**16**	**101**	**0.041773**	**6.09e-07**	12	93	0.037660	7.87e-07	F	F	F	F
	x5	23	256	0.086990	6.05e-07	F	F	F	F	**16**	**101**	**0.048567**	**6.10e-07**	12	93	0.037951	7.93e-07	F	F	F	F
	x6	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	x7	23	256	0.079468	6.05e-07	F	F	F	F	**16**	**101**	**0.039340**	**6.10e-07**	12	93	0.025893	7.93e-07	F	F	F	F
	x8	23	256	0.075101	6.50e-07	55	240	0.112374	2.19e-07	**16**	**101**	**0.037840**	**6.09e-07**	12	93	0.032658	7.87e-07	F	F	F	F
	x9	23	256	0.086070	6.50e-07	F	F	F	F	**16**	**101**	**0.038221**	**6.09e-07**	12	93	0.028537	7.87e-07	F	F	F	F
100000	x1	20	224	0.184630	9.62e-07	16	99	0.115764	7.00e-07	**14**	**89**	**0.092163**	**7.88e-07**	14	110	0.108153	4.32e-07	F	F	F	F
	x2	20	224	0.218640	9.62e-07	18	83	0.136443	8.43e-07	**14**	**89**	**0.105534**	**7.88e-07**	14	110	0.118214	4.32e-07	F	F	F	F
	x3	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	x4	20	224	0.180234	9.62e-07	14	65	0.091114	8.43e-07	**14**	**89**	**0.092867**	**7.88e-07**	14	110	0.147266	4.32e-07	F	F	F	F
	x5	20	224	0.187211	9.27e-07	27	140	0.183606	8.43e-07	**14**	**89**	**0.095987**	**7.89e-07**	14	110	0.101754	4.45e-07	F	F	F	F
	x6	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	x7	20	224	0.189469	9.27e-07	27	140	0.176783	8.43e-07	**14**	**89**	**0.092387**	**7.89e-07**	14	110	0.106095	4.45e-07	F	F	F	F
	x8	20	224	0.218604	9.62e-07	27	140	0.217050	8.43e-07	**14**	**89**	**0.115369**	**7.88e-07**	14	110	0.122092	4.32e-07	F	F	F	F
	x9	20	224	0.293651	9.62e-07	26	135	0.267497	8.43e-07	**14**	**89**	**0.132654**	**7.88e-07**	14	110	0.143270	4.32e-07	F	F	F	F

Table 3.9: Numerical results of Function 9.

Dim	InitPoint	YSD				CGD				PCG				PDY				MCHGC			
		IT	FE	CPU	Norm	IT	FT	CPU	Norm	IT	FE	CPU	Norm	IT	FE	CPU	Norm	IT	FE	CPU	Norm
1000	x1	8	90	0.011018	5.19e-07	22	92	0.006835	7.37e-07	9	50	0.007027	8.44e-07	**6**	**46**	**0.006914**	**8.50e-07**	F	F	F	F
	x2	7	83	0.000770	5.07e-08	21	88	0.000900	6.65e-07	9	50	0.000265	3.31e-07	**6**	**46**	**0.000218**	**3.34e-07**	F	F	F	F
	x3	7	83	0.000287	1.57e-08	21	88	0.000531	9.19e-07	9	50	0.000433	4.57e-07	**6**	**46**	**0.000370**	**4.61e-07**	F	F	F	F
	x4	8	94	0.000304	1.48e-08	30	124	0.000731	8.32e-07	13	70	0.000334	5.88e-07	**10**	**74**	**0.000301**	**7.32e-08**	F	F	F	F
	x5	7	83	0.000266	2.47e-08	21	88	0.000574	8.49e-07	9	50	0.000249	4.22e-07	**6**	**46**	**0.000200**	**4.26e-07**	F	F	F	F
	x6	7	83	0.000264	1.63e-08	21	88	0.000498	9.14e-07	9	50	0.000244	4.55e-07	**6**	**46**	**0.000199**	**4.58e-07**	F	F	F	F
	x7	7	83	0.000306	2.47e-08	21	88	0.000493	8.49e-07	9	50	0.000242	4.22e-07	**6**	**46**	**0.000200**	**4.26e-07**	F	F	F	F
	x8	7	83	0.000479	2.45e-08	21	88	0.000505	8.50e-07	9	50	0.000269	4.23e-07	**6**	**46**	**0.000206**	**4.26e-07**	F	F	F	F
	x9	7	83	0.000276	2.76e-08	21	88	0.000513	8.27e-07	9	50	0.000251	4.11e-07	**6**	**46**	**0.000205**	**4.14e-07**	F	F	F	F
5000	x1	8	94	0.002235	9.56e-08	23	96	0.004075	7.15e-07	10	55	0.001989	2.63e-07	**7**	**53**	**0.001580**	**1.12e-07**	F	F	F	F
	x2	8	90	0.002045	4.44e-07	22	92	0.003860	6.46e-07	9	50	0.001694	7.40e-07	**6**	**46**	**0.001411**	**7.46e-07**	F	F	F	F
	x3	8	90	0.002029	6.51e-07	22	92	0.003869	8.99e-07	10	55	0.001950	1.43e-07	**7**	**53**	**0.001542**	**6.13e-08**	F	F	F	F
	x4	9	101	0.002231	6.52e-07	33	136	0.005844	7.63e-07	15	80	0.002718	1.89e-07	**10**	**74**	**0.002208**	**8.26e-07**	F	F	F	F
	x5	8	90	0.002025	5.91e-07	22	92	0.003916	8.25e-07	9	50	0.001819	9.44e-07	**6**	**46**	**0.001406**	**9.52e-07**	F	F	F	F
	x6	8	90	0.001987	6.50e-07	22	92	0.003178	8.97e-07	10	55	0.001886	1.43e-07	**7**	**53**	**0.001527**	**6.12e-08**	F	F	F	F
	x7	8	90	0.002002	5.91e-07	22	92	0.003918	8.25e-07	9	50	0.001701	9.44e-07	**6**	**46**	**0.001329**	**9.52e-07**	F	F	F	F
	x8	8	90	0.002027	5.91e-07	22	92	0.003847	8.25e-07	9	50	0.001669	9.45e-07	**6**	**46**	**0.001346**	**9.52e-07**	F	F	F	F
	x9	8	90	0.002095	5.93e-07	22	92	0.003856	8.27e-07	9	50	0.001670	9.47e-07	**6**	**46**	**0.001339**	**9.55e-07**	F	F	F	F
10000	x1	8	94	0.003623	7.13e-08	24	100	0.007216	4.39e-07	10	55	0.003460	3.71e-07	**7**	**53**	**0.002645**	**1.58e-07**	F	F	F	F
	x2	8	90	0.003504	6.63e-07	22	92	0.006734	9.14e-07	10	55	0.003365	1.46e-07	**7**	**53**	**0.002625**	**6.23e-08**	F	F	F	F
	x3	8	90	0.003439	9.48e-07	23	96	0.007195	5.53e-07	10	55	0.003271	2.03e-07	**7**	**53**	**0.002631**	**8.66e-08**	F	F	F	F
	x4	9	101	0.003865	9.49e-07	34	140	0.013849	9.37e-07	15	80	0.006345	5.34e-07	**12**	**88**	**0.005918**	**1.03e-07**	F	F	F	F
	x5	8	90	0.003476	8.64e-07	23	96	0.006946	5.06e-07	10	55	0.003271	1.86e-07	**7**	**53**	**0.002633**	**7.94e-08**	F	F	F	F
	x6	8	90	0.003406	9.47e-07	23	96	0.006815	5.52e-07	10	55	0.003326	2.03e-07	**7**	**53**	**0.002640**	**8.65e-08**	F	F	F	F
	x7	8	90	0.003444	8.64e-07	23	96	0.007301	5.06e-07	10	55	0.003344	1.86e-07	**7**	**53**	**0.002646**	**7.94e-08**	F	F	F	F
	x8	8	90	0.003483	8.64e-07	23	96	0.006828	5.07e-07	10	55	0.003383	1.86e-07	**7**	**53**	**0.002608**	**7.94e-08**	F	F	F	F
	x9	8	90	0.003467	8.61e-07	23	96	0.006622	5.05e-07	10	55	0.003419	1.85e-07	**7**	**53**	**0.002638**	**7.91e-08**	F	F	F	F
50000	x1	9	101	0.015806	3.35e-07	24	100	0.031995	9.82e-07	10	55	0.013292	8.31e-07	**7**	**53**	**0.010302**	**3.54e-07**	F	F	F	F
	x2	8	94	0.013881	8.20e-08	23	96	0.028034	8.87e-07	10	55	0.011449	3.26e-07	**7**	**53**	**0.010379**	**1.39e-07**	F	F	F	F
	x3	8	94	0.013524	5.24e-08	24	100	0.032417	5.37e-07	10	55	0.013250	4.54e-07	**7**	**53**	**0.011335**	**1.94e-07**	F	F	F	F
	x4	9	105	0.015095	5.24e-08	37	152	0.044790	8.58e-07	16	85	0.022685	9.51e-07	**22**	**158**	**0.031520**	**7.73e-07**	F	F	F	F
	x5	8	94	0.014072	6.10e-08	24	100	0.029497	4.92e-07	10	55	0.012059	4.16e-07	**7**	**53**	**0.013681**	**1.77e-07**	F	F	F	F
	x6	8	94	0.013779	5.24e-08	24	100	0.032055	5.37e-07	10	55	0.014364	4.54e-07	**7**	**53**	**0.018644**	**1.93e-07**	F	F	F	F
	x7	8	94	0.015422	6.10e-08	24	100	0.026929	4.92e-07	10	55	0.011586	4.16e-07	**7**	**53**	**0.010214**	**1.77e-07**	F	F	F	F
	x8	8	94	0.014711	6.10e-08	24	100	0.029293	4.92e-07	10	55	0.012782	4.16e-07	**7**	**53**	**0.013954**	**1.77e-07**	F	F	F	F
	x9	8	94	0.017933	6.11e-08	24	100	0.025929	4.92e-07	10	55	0.016246	4.16e-07	**7**	**53**	**0.015371**	**1.77e-07**	F	F	F	F
100000	x1	9	101	0.054705	5.17e-07	25	104	0.102340	6.03e-07	11	60	0.048262	2.33e-07	**7**	**53**	**0.035316**	**5.00e-07**	F	F	F	F
	x2	8	94	0.049037	5.10e-08	24	100	0.102626	5.45e-07	10	55	0.039321	4.61e-07	**7**	**53**	**0.033598**	**1.96e-07**	F	F	F	F
	x3	8	94	0.049253	1.50e-08	24	100	0.102121	7.60e-07	10	55	0.042580	6.42e-07	**7**	**53**	**0.035400**	**2.74e-07**	F	F	F	F
	x4	9	105	0.073958	1.50e-08	39	160	0.166158	4.58e-07	17	90	0.069408	4.89e-07	**26**	**186**	**0.130089**	**9.03e-07**	F	F	F	F
	x5	8	94	0.039519	2.49e-08	24	100	0.092098	6.96e-07	10	55	0.035102	5.88e-07	**7**	**53**	**0.028182**	**2.51e-07**	F	F	F	F
	x6	8	94	0.043162	1.50e-08	24	100	0.105626	7.60e-07	10	55	0.038025	6.42e-07	**7**	**53**	**0.027044**	**2.74e-07**	F	F	F	F
	x7	8	94	0.041473	2.49e-08	24	100	0.084998	6.96e-07	10	55	0.036813	5.88e-07	**7**	**53**	**0.027927**	**2.51e-07**	F	F	F	F
	x8	8	94	0.042030	2.49e-08	24	100	0.125724	6.96e-07	10	55	0.044029	5.88e-07	**7**	**53**	**0.028898**	**2.51e-07**	F	F	F	F
	x9	8	94	0.043425	2.50e-08	24	100	0.079260	6.95e-07	10	55	0.032999	5.88e-07	**7**	**53**	**0.033508**	**2.50e-07**	F	F	F	F

Table 3.10: Numerical results of Function 10.

Dim	InitPoint	YSD				CGD				PCG				PDY				MCHGC			
		IT	FE	CPU	Norm	IT	FT	CPU	Norm	IT	FE	CPU	Norm	IT	FE	CPU	Norm	IT	FE	CPU	Norm
1000	x1	**1**	**10**	**0.006177**	**0**	1	4	0.004390	0	1	4	0.004427	0	1	5	0.004595	0	1	10	0.004299	0
	x2	**1**	**13**	**0.001036**	**0**	18	76	0.009493	7.54e-07	8	45	0.005468	1.03e-07	3	25	0.004568	1.58e-07	1	19	0.000811	0
	x3	**1**	**13**	**0.000325**	**0**	19	78	0.003698	7.01e-07	14	85	0.002859	7.36e-07	21	164	0.004901	7.43e-07	2	35	0.004150	0
	x4	**1**	**19**	**0.005577**	**0**	1	13	0.005269	0	1	13	0.004862	0	1	14	0.004792	0	1	19	0.006223	0
	x5	**2**	**21**	**0.005318**	**0**	21	86	0.003369	5.42e-07	19	113	0.003081	6.34e-07	32	243	0.005840	8.25e-07	3	33	0.000835	0
	x6	**2**	**21**	**0.000501**	**0**	21	87	0.003349	5.23e-07	18	104	0.002871	6.89e-07	27	207	0.005540	1.99e-09	6	81	0.001311	0
	x7	**2**	**21**	**0.000551**	**0**	21	86	0.003399	5.42e-07	19	113	0.003362	6.34e-07	32	243	0.005129	8.25e-07	3	33	0.000698	0
	x8	**2**	**21**	**0.000553**	**0**	21	86	0.003374	5.41e-07	19	113	0.003534	6.39e-07	34	257	0.005228	8.87e-07	3	33	0.000738	0
	x9	**2**	**21**	**0.000613**	**0**	21	86	0.004368	5.43e-07	19	113	0.003702	6.95e-07	32	243	0.005715	8.92e-07	3	33	0.000779	0
5000	x1	**1**	**10**	**0.001562**	**0**	1	4	0.001463	0	1	4	0.001379	0	1	5	0.001406	0	1	10	0.001423	0
	x2	**1**	**13**	**0.001265**	**0**	19	80	0.021444	6.74e-07	8	45	0.009603	2.30e-07	3	25	0.003449	3.54e-07	1	19	0.001576	0
	x3	**1**	**13**	**0.001427**	**0**	19	78	0.017751	6.90e-07	14	85	0.012630	7.38e-07	21	164	0.015154	7.45e-07	2	35	0.003042	0
	x4	**1**	**19**	**0.022757**	**0**	1	13	0.022829	0	1	13	0.021208	0	1	14	0.021611	0	1	19	0.029850	0
	x5	**2**	**21**	**0.002782**	**0**	22	90	0.023003	4.96e-07	20	119	0.018531	6.47e-07	35	265	0.038116	8.80e-07	3	33	0.004821	0
	x6	**2**	**21**	**0.003295**	**0**	21	87	0.024386	5.19e-07	20	112	0.018343	9.73e-07	27	207	0.024601	1.97e-09	6	81	0.007249	0
	x7	**2**	**21**	**0.002946**	**0**	22	90	0.021826	4.96e-07	20	119	0.018374	6.47e-07	35	265	0.030168	8.80e-07	3	33	0.003705	0
	x8	**2**	**21**	**0.002748**	**0**	22	90	0.020782	4.96e-07	20	119	0.020804	6.47e-07	41	306	0.037015	9.71e-07	3	33	0.003774	0
	x9	**2**	**21**	**0.002989**	**0**	22	90	0.024403	4.93e-07	20	119	0.021040	6.37e-07	32	244	0.033070	8.64e-07	3	33	0.004014	0
10000	x1	**1**	**10**	**0.003066**	**0**	1	4	0.002896	0	1	4	0.002736	0	1	5	0.002742	0	1	10	0.002801	0
	x2	**1**	**13**	**0.002373**	**0**	19	80	0.035651	9.54e-07	8	45	0.015204	3.26e-07	3	25	0.006359	5.00e-07	1	19	0.002916	0
	x3	**1**	**13**	**0.002434**	**0**	19	78	0.035968	6.89e-07	14	85	0.024201	7.38e-07	21	164	0.029844	7.45e-07	2	35	0.005867	0
	x4	**1**	**19**	**0.025463**	**0**	1	13	0.025595	0	1	13	0.024873	0	1	14	0.025742	0	1	19	0.032985	0
	x5	**2**	**21**	**0.005250**	**0**	22	90	0.041852	7.07e-07	20	119	0.040137	9.15e-07	39	293	0.075043	4.22e-09	3	33	0.007621	0
	x6	**2**	**21**	**0.004489**	**0**	21	87	0.037424	5.18e-07	19	109	0.031207	5.44e-07	27	207	0.042809	1.97e-09	6	81	0.013483	0
	x7	**2**	**21**	**0.005090**	**0**	22	90	0.043129	7.07e-07	20	119	0.035024	9.15e-07	39	293	0.071239	4.22e-09	3	33	0.008197	0
	x8	**2**	**21**	**0.005413**	**0**	22	90	0.042535	7.07e-07	20	119	0.039289	9.15e-07	31	231	0.053189	2.53e-07	3	33	0.007104	0
	x9	**2**	**21**	**0.006347**	**0**	22	90	0.042429	7.17e-07	20	119	0.041867	9.16e-07	95	680	0.150198	9.98e-07	3	33	0.007591	0
50000	x1	**1**	**10**	**0.012797**	**0**	1	4	0.012802	0	1	4	0.011906	0	1	5	0.011330	0	1	10	0.016570	0
	x2	**1**	**13**	**0.009915**	**0**	20	84	0.177890	8.53e-07	8	45	0.070166	7.28e-07	4	32	0.043436	2.15e-09	1	19	0.012450	0
	x3	**1**	**13**	**0.012121**	**0**	19	78	0.143690	6.88e-07	14	85	0.103382	7.38e-07	21	164	0.128770	7.45e-07	2	35	0.029728	0
	x4	**1**	**19**	**0.130031**	**0**	1	13	0.128064	0	1	13	0.122594	0	1	14	0.119715	0	1	19	0.166274	0
	x5	**2**	**21**	**0.029236**	**0**	23	94	0.192296	6.40e-07	21	125	0.181857	9.30e-07	39	291	0.328785	2.43e-08	3	33	0.035883	0
	x6	**2**	**21**	**0.017691**	**0**	21	87	0.171733	5.17e-07	19	109	0.139116	5.17e-07	27	207	0.188409	1.98e-09	6	81	0.064829	0
	x7	**2**	**21**	**0.025982**	**0**	23	94	0.195568	6.40e-07	21	125	0.176307	9.30e-07	39	291	0.316929	2.43e-08	3	33	0.031570	0
	x8	**2**	**21**	**0.028365**	**0**	23	94	0.203319	6.40e-07	21	125	0.180336	9.30e-07	98	702	0.661070	9.99e-07	3	33	0.031860	0
	x9	**2**	**21**	**0.030731**	**0**	23	94	0.221829	6.38e-07	21	125	0.196961	9.30e-07	103	737	0.715879	9.99e-07	3	33	0.033156	0
100000	x1	**1**	**10**	**0.029186**	**0**	1	4	0.023440	0	1	4	0.025589	0	1	5	0.022210	0	1	10	0.028096	0
	x2	**1**	**13**	**0.019394**	**0**	21	88	0.365018	4.83e-07	9	50	0.169114	1.85e-07	4	32	0.072330	2.91e-09	1	19	0.027017	0
	x3	**1**	**13**	**0.017607**	**0**	19	78	0.279202	6.88e-07	14	85	0.199383	7.38e-07	21	164	0.276120	7.45e-07	2	35	0.061235	0
	x4	**1**	**19**	**0.288084**	**0**	1	13	0.261124	0	1	13	0.248296	0	1	14	0.245955	0	1	19	0.330298	0
	x5	**2**	**21**	**0.050210**	**0**	23	94	0.401655	9.07e-07	22	131	0.372767	5.98e-07	39	295	0.679531	7.52e-07	3	33	0.065710	0
	x6	**2**	**21**	**0.040422**	**0**	21	87	0.315651	5.17e-07	19	109	0.259315	5.11e-07	27	207	0.431498	1.98e-09	6	81	0.126862	0
	x7	**2**	**21**	**0.053434**	**0**	23	94	0.389122	9.07e-07	22	131	0.352039	5.98e-07	39	295	0.647403	7.52e-07	3	33	0.065320	0
	x8	**2**	**21**	**0.050193**	**0**	23	94	0.389484	9.07e-07	22	131	0.392706	5.98e-07	42	316	0.703851	8.15e-07	3	33	0.071855	0
	x9	**2**	**21**	**0.057414**	**0**	23	94	0.429859	9.06e-07	22	131	0.401173	5.97e-07	37	281	0.695595	8.68e-07	3	33	0.074987	0

3.4.1 Comments

Based on the results shown in tables 3.1 until 3.10 and figures 3.1, 3.2, and 3.3, it is clear that our YSD algorithm consistently outperforms the methods based on CGD, PDY, PCG and MCHGC. The YSD algorithm shows superior efficiency and robustness for solving nonlinear equations, with better performance on key indicators, particularly the

most significant ones including CPU time and iterations number required for optimality. Furthermore, it is observed that in some examples, the MCHGC and CGD fail whereas our YSD method provides optimal solutions in almost all examples in minimal time with a reasonable number of iterations. However, a slight superiority of methods PDY and PCG is recorded in a few examples regarding the evaluations number of the objective function.

3.5 Application in compressive sensing problem

In this section, we compare the performance of the three algorithms and analyze the effectiveness of our proposed YSD algorithm for the compressive sensing problem. Compressive sensing is a fundamental problem in signal processing, where the goal is to recover a sparse signal from noisy or incomplete measurements. The optimization problem can be formulated as

$$\min_{x\in\mathbb{R}^n} f(x) = \frac{1}{2}\|Ax-b\|^2 + \tau\|x\|_1, \tag{3.35}$$

where: $A \in \mathbb{R}^{m\times n}$ (with $m < n$) is the measurement matrix, $b \in \mathbb{R}^m$ represents the observed data, $\tau > 0$ is a regularization parameter that balances data fidelity and sparsity [10, 40].

To simplify the problem, we split the variables into their positive and negative parts using the following definitions

$$u = \max\{x, 0\}, \quad v = \max\{-x, 0\}, \tag{3.36}$$

where $x = u - v$ and $\|x\|_1 = \mathbf{e}_n^T u + \mathbf{e}_n^T v$, with $\mathbf{e}_n = (1, 1, \ldots, 1)^T \in \mathbb{R}^n$.

The quality of signal recovery is typically evaluated using the mean squared error (MSE), defined as

$$\text{MSE} = \frac{1}{n}\|x^\star - x\|^2, \tag{3.37}$$

where $x^\star$ is the recovered signal and x is the original sparse signal. A lower MSE indicates that the estimated signal is closer to the true signal, reflecting better performance.

In our experiments, the sparsity level N is set to 128, the problem size is $n = 2048$ and $m = 1024$ and the regularization parameter $\tau = 0.00528971$. The measurement matrix A is generated as a Gaussian matrix using the `randn(m,n)` command in MATLAB, and the observed data b is perturbed by Gaussian noise δ, i.e., $b = A\bar{x} + \delta$, where δ is distributed as $N(0, 10^{-4})$.

From [24], the optimization problem (3.35) can be reformulated as a system of nonlinear equations as

$$F(z) = \min\{z, Hz + c\} = 0, \tag{3.38}$$

where $z = \begin{pmatrix} u \\ v \end{pmatrix}$, $H = \begin{pmatrix} A^TA & -A^TA \\ -A^TA & A^TA \end{pmatrix}$, $c = \tau e_{2n} + \begin{pmatrix} -y \\ y \end{pmatrix}$, with $y = A^Tb$.

Here, H is a positive semi-definite matrix and the function $F(z)$ is Lipschitz continuous and monotone, as shown in [24]. This reformulation allows us to solve the problem using iterative methods for nonlinear equations, such as the proposed YSD algorithm.

The performance of the algorithms is evaluated based on the MSE and the number of iterations required to achieve a solution with a similar accuracy. The initial point for all methods is set to $x_0 = A^Tb$ and the algorithms terminate when the relative change in the objective function satisfies

$$\frac{\|f_k - f_{k-1}\|}{\|f_{k-1}\|} < 10^{-5}, \tag{3.39}$$

where f_k denotes the function value at x_k.

This setup allows us to compare the effectiveness of the proposed YSD algorithm with other methods in the context of compressive sensing, particularly in terms of convergence rate and the accuracy of signal recovery.

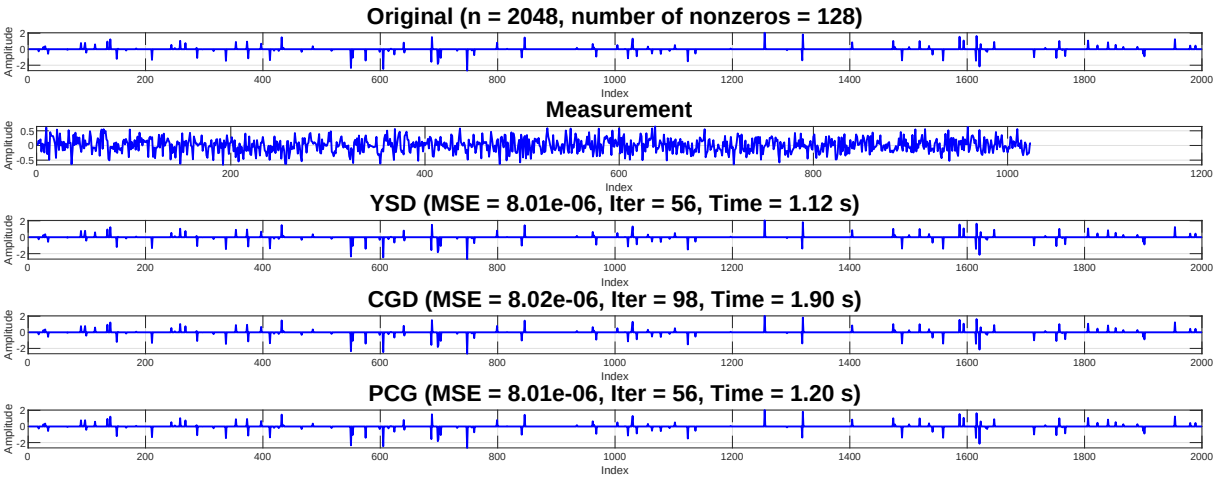


Figure 3.4: The original sparse signal compared to the restored signals.

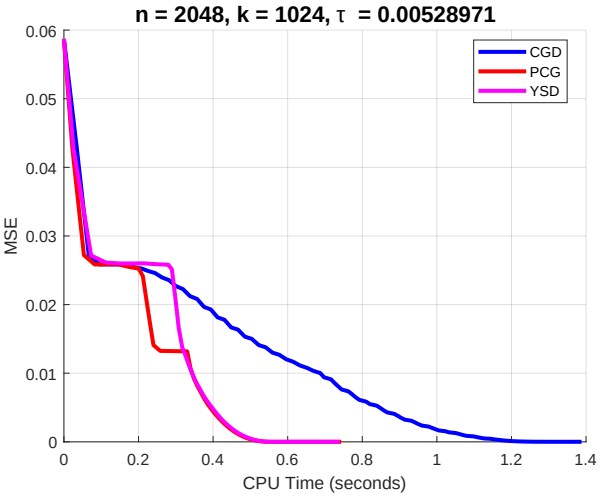


Figure 3.5: MSE in terms of iterations.

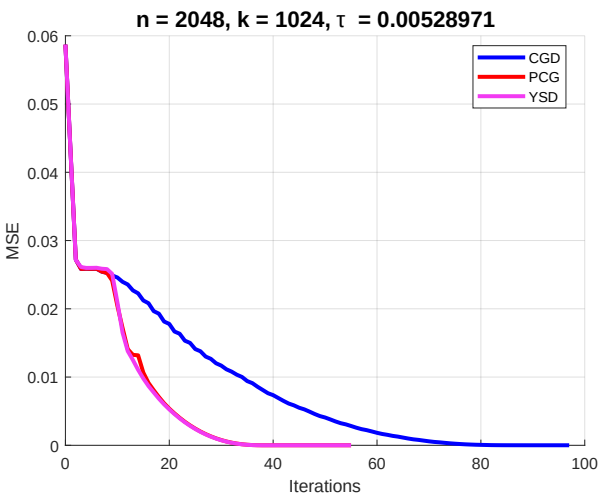


Figure 3.6: MSE in temrs of CPU time.

3.5.1 Commentaries

For the compressive sensing application, the results shown in figures 3.4, 3.5 and 3.6 indicate that the proposed YSD method performs very similarly to PCG in terms of accuracy, with both achieving nearly identical MSE values. However, the YSD algorithm shows a slight advantage in computational efficiency, requiring less time to complete the task, as seen in figure 3.6. While CGD also performs well, it requires more iterations and longer computational time compared to YSD and PCG, as illustrated in figure 3.5.

3.6 Conclusion

In this chapter, we proposed a new hybrid conjugate gradient projection method YSD for solving large-scale nonlinear monotone equations with convex constraints. The method combines the RMIL and HS conjugate gradient techniques with the projection-based approach of Solodov and Svaiter, making it efficient for large-scale and nonsmooth problems without requiring large matrix storage.

We established the global convergence under certain assumptions. Numerical experiments show that YSD outperforms existing methods such as CGD, PDY, PCG and MCHGC in terms of computational efficiency, accuracy, and robustness. In compressive sensing applications, the YSD algorithm compared with CGD and PCG achieves competitive signal recovery accuracy with low MSE values and superior computational efficiency.

CHAPTER 4

An efficient hybrid three-term conjugate gradient method with practical applications

In this chapter, we introduces a new hybrid conjugate gradient parameter β_k, designed for the efficient solution of unconstrained optimization problems. The new parameter is constructed by hybridizing the Liu-Storey β_k^{LS} parameter with its modification β_k^{DLS}. The proposed algorithm incorporates a hybrid three-term technique that guarantees sufficient descent for the search direction, a property established under the strong Wolfe line search conditions. The practical efficiency of the proposed approach is validated through comprehensive numerical experiments. These tests are conducted on a set of well-known benchmark problems and extended to real-world applications, including image restoration and sparse signal recovery, showing the robustness and effectiveness of the method. The results of this work are published online in the **Iranian Journal of Numerical Analysis and Optimization (IJNAO)** [35].

4.1 Introduction

Optimization domain is widely and increasingly used in science, engineering, economics, management, industry, and other areas. The most common optimization algorithms are the descent methods namely gradient, Newton's and conjugate gradient methods. Let be $f : \mathbb{R}^n \to \mathbb{R}$ a continuously differentiable function. Our goal is to solve the unconstrained optimization problem $\{\min f(x),\ x \in \mathbb{R}^n\}$. The principle of the

conjugate gradient method is to construct a sequence

$$x_1 \in \mathbb{R}^n,\ x_{k+1} = x_k + \alpha_k d_k,\ \text{for } k \geq 1, \tag{4.1}$$

where x_1 is an initial point, and denote by x_k the current iterate. The stepsize $\alpha_k > 0$ is computed using an exact or inexact line search, while the search direction d_k is defined as

$$d_1 = -g_1,\ d_{k+1} = -g_{k+1} + \beta_k d_k,\ \text{for } k \geq 1,$$

where $g_k = \nabla f(x_k)$ is the gradient of f at x_k and β_k is a scalar conjugacy coefficient.

Dai and Wen [21] proposed two modified conjugate gradient methods of PRP and HS respectively, denoted by DPRP and DHS given by

$$\beta_k^{DPRP} = \frac{\|g_{k+1}\|^2 - \frac{\|g_{k+1}\|}{\|g_k\|}\left|g_{k+1}^T g_k\right|}{\mu\left|g_{k+1}^T d_k\right| + \|g_{k+1}\|^2},\ \mu > 1, \tag{4.2}$$

$$\beta_k^{DHS} = \frac{\|g_{k+1}\|^2 - \frac{\|g_{k+1}\|}{\|g_k\|}\left|g_{k+1}^T g_k\right|}{\mu\left|g_{k+1}^T d_k\right| + d_k^T y_k},\ \mu > 1. \tag{4.3}$$

Additionally, inspired by (4.2) and (4.3) Zheng et al. proposed in [72] a modified LS method denoted as DLS given by

$$\beta_k^{DLS} = \frac{\|g_{k+1}\|^2 - \frac{\|g_{k+1}\|}{\|g_k\|}\left|g_{k+1}^T g_k\right|}{\mu\left|g_{k+1}^T d_k\right| - d_k^T y_k},\ \mu > 1. \tag{4.4}$$

In 2022 Jiang et al. [39], based on the hybrid conjugate gradient method and the convex combination technique, proposed a new family of hybrid three-term conjugate gradient methods

$$d_{k+1} = \begin{cases} -g_{k+1} \quad \text{if} \quad k = 0, \\ -g_{k+1} + (1-\lambda_k)\beta_k^\star d_k + \lambda_k \theta_k g_k,\ \text{if}\ \ k > 0, \end{cases} \tag{4.5}$$

where $\beta_k^\star = \max\{0, \min\{\beta_k, \beta_k^{DY}\}\}$, β_k is any conjugate parameter, $\lambda_k = \frac{\left|g_{k+1}^T d_k\right|}{\|g_{k+1}^T\|\|d_k\|}$, with $\lambda_k \in [0,1]$ and $\theta_k = -\eta\frac{g_{k+1}^T g_k}{\|g_k\|^2}$, $0 < \eta < 1$.

The objective of this work is to propose a new hybridization between LS and DLS using the formula (4.5) to calculate a new descent search direction.

4.2 New hybrid three-term conjugate gradient algorithm

Inspired by the works of Zheng et al.[72] and Jiang et al.[39], we propose a new hybrid parameter based on LS and DLS parameters (4.4), noted β_k^{MDLS}, defined by

$$\beta_k^{MDLS} = \begin{cases} \beta_k^{LS} = -\frac{g_{k+1}^T y_k}{g_k^T d_k}, & \text{if} \quad \|g_{k+1}\|^2 \geq \left|g_{k+1}^T g_k\right|, \\ \beta_k^{DLS} = \frac{\|g_{k+1}\|^2 - \frac{\|g_{k+1}\|}{\|g_k\|}\left|g_{k+1}^T g_k\right|}{\mu\left|g_{k+1}^T d_k\right| - d_k^T y_k}, & \text{otherwise}, \end{cases} \tag{4.6}$$

where $\mu > 1$.

The search direction is given as follows

$$d_{k+1} = \begin{cases} -g_{k+1} \quad \text{if} \quad k = 0 \text{ or } \left|g_{k+1}^T g_k\right| \geq 0.5\|g_{k+1}\|^2, \\ -g_{k+1} + (1-\lambda_k)\beta_k^{MDLS} d_k + \lambda_k \theta_k g_k, \text{ if } k > 0, \end{cases} \tag{4.7}$$

where $\lambda_k = \frac{\left|g_{k+1}^T s_k\right|}{\|g_{k+1}\|\|s_k\|}$, $s_k = x_{k+1} - x_k$ and $\theta_k = -\eta\frac{g_{k+1}^T g_k}{\|g_k\|^2}$. Note that $\lambda_k \in [0,1]$ and $0 < \eta < 1$.

In this study, to show the behavior of our algorithm, we use the strong Wolfe line search, i.e., we find the stepsize α_k using the following conditions:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \delta\alpha_k g_k^T d_k, \tag{4.8}$$

$$\left|g_{k+1}^T d_k\right| \leq -\sigma g_k^T d_k, \tag{4.9}$$

where $0 < \delta < \sigma < \frac{1}{2}$.

The corresponding algorithm is given bellow

Algorithm 13: MDLS

Step 0: Given a starting point x_0 and a parameter $\varepsilon > 0$.

Step 1: Set $k = 1$ and compute $d_1 = -g_1$.

Step 2: If $\|g_k\| \leq \varepsilon$, **Stop**; else go to **Step 3**.

Step 3: Find the stepsize $\alpha_k \in]0,1]$ (using strong Wolfe conditions (4.8), (4.9)).

Step 4: Compute $x_{k+1} = x_k + \alpha_k d_k$.

Step 5: Compute $g_{k+1} = \nabla f(x_{k+1})$, $y_k = g_{k+1} - g_k$.

Step 6: Compute $\beta_k = \beta_k^{MDLS}$ (4.6).

Step 7: Compute d_{k+1} using (4.7).

Step 8: Let $k = k + 1$ and go to **Step 2.**

4.3 Convergence analysis

We introduce the following theorem to establish the sufficient descent condition.

Theorem 4.1. *Let $\{g_k\}$ and $\{d_k\}$ be the sequences generated by $MDLS$ algorithm. Then the search direction satisfies the sufficient descent condition*

$$g_{k+1}^T d_{k+1} \leq -c\|g_{k+1}\|^2, \text{ for all } k \geq 0. \tag{4.10}$$

Proof. From $MDLS$ algorithm, we know that if $\left|g_{k+1}^T g_k\right| \geq 0.5\|g_{k+1}\|^2$ or $k = 0$, then $d_{k+1} = -g_{k+1}$ and (4.10) holds.

Now, we assume that

$$| g_{k+1}^T g_k | < 0.5\|g_{k+1}\|^2, \tag{4.11}$$

and $k > 0$.

By multiplying (4.7) by g_{k+1}^T from the left, we get

$$g_{k+1}^T d_{k+1}^{MDLS} = -\|g_{k+1}\|^2 + (1-\lambda_k)\beta_k^{MDLS} g_{k+1}^T d_k + \lambda_k \theta_k g_{k+1}^T g_k, \tag{4.12}$$

and we can analyze this into two cases as follows

Case 1: If $\|g_{k+1}\|^2 \geq \left|g_{k+1}^T g_k\right|$, then $\beta_k^{MDLS} = \beta_k^{LS}$, and (4.12) becomes

$$\begin{aligned} g_{k+1}^T d_{k+1}^{MDLS} &= -\|g_{k+1}\|^2 + (1-\lambda_k)\beta_k^{LS} g_{k+1}^T d_k - \eta\lambda_k \frac{g_{k+1}^T g_k}{\|g_k\|^2} g_{k+1}^T g_k \\ &= -(1+\eta\lambda_k \cos^2\omega)\|g_{k+1}\|^2 + (1-\lambda_k)\frac{g_{k+1}^T y_k}{-g_k^T d_k} g_{k+1}^T d_k, \end{aligned} \tag{4.13}$$

where ω is the angle between g_k and g_{k+1}.

Using the second condition of strong Wolfe line search, we get

$$\begin{aligned} g_{k+1}^T d_{k+1}^{MDLS} &\leq -(1+\eta\lambda_k \cos^2\omega)\|g_{k+1}\|^2 + (1-\lambda_k)\sigma g_{k+1}^T y_k \\ &= -(1+\eta\lambda_k \cos^2\omega)\|g_{k+1}\|^2 + (1-\lambda_k)\sigma\|g_{k+1}\|^2 - (1-\lambda_k)\sigma g_{k+1}^T g_k \\ &\leq -(1+\eta\lambda_k \cos^2\omega)\|g_{k+1}\|^2 + 2(1-\lambda_k)\sigma\|g_{k+1}\|^2 \\ &= -(1-2\sigma+(2\sigma+\cos^2\omega)\lambda_k)\|g_{k+1}\|^2. \end{aligned}$$

Hence,

$$g_{k+1}^T d_{k+1}^{MDLS} \leq -c\|g_{k+1}\|^2,$$

where $c = 1 - 2\sigma + (2\sigma + \cos^2\omega)\lambda_k > 0$. Therefore, (4.10) holds.

Case 2: If $\|g_{k+1}\|^2 < \left|g_{k+1}^T g_k\right|$, then $\beta_k^{MDLS} = \beta_k^{DLS}$. We also know that $\left|g_{k+1}^T g_k\right| \geq 0.5\|g_{k+1}\|^2$ and from [66] $\beta_k^{DLS} \leq -\frac{\|g_{k+1}\|^2}{g_k^T d_k}$. So, following the first case, we get

$$\begin{aligned} g_{k+1}^T d_{k+1}^{MDLS} &\leq -(1 + \eta\lambda_k \cos^2\omega)\|g_{k+1}\|^2 + (1 - \lambda_k)\sigma\|g_{k+1}\|^2 \\ &= -(1 - \sigma + (\sigma + \cos^2\omega)\lambda_k)\|g_{k+1}\|^2. \end{aligned}$$

Hence,

$$g_{k+1}^T d_{k+1}^{MDLS} \leq -c\|g_{k+1}\|^2,$$

where $c = 1 - \sigma + (\sigma + \cos^2\omega)\lambda_k > 0$. Therefore, (4.10) holds. The proof is complete. □

To establish the global convergence, we make the following assumptions

- *(i)* The level set $\mathcal{L} = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ is bounded, which means there exists a constant M, such that
$$\|x\| \leq M, \text{ for all } x \in \mathcal{L}.$$
- *(ii)* In a neighborhood $\mathcal{N}$ of $\mathcal{L}$ the function f is continuously differentiable and its gradient $\nabla f(x)$ is Lipschitz continuous, it means: $\exists 0 < L < \infty$ such that
$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \text{ for all } x, y \in \mathcal{N}. \tag{4.14}$$

Under these assumptions, there exists a constant $\mu \geq 0$, where

$$\|\nabla f(x)\| \leq \mu, \text{ for all } x \in \mathcal{L}. \tag{4.15}$$

Lemma 4.2. *Suppose that assumptions (i) and (ii) hold. Consider common iterate (4.1), where d_k is a descent direction (4.7) and α_k is determined by the strong Wolfe line search (4.8) and (4.9). Then, the Zoutendijk condition*

$$\sum_{k\geq 0} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty, \tag{4.16}$$

holds.

Proof. The proof follows directly from [73]. □

Lemma 4.3. *Suppose that assumptions (i) and (ii) hold and $\alpha_k > 0$ is determined by the strong Wolfe line search (4.8) and (4.9), then there exist $\alpha_* > 0$ such that*

$$\alpha_k \geq \alpha_*, \textit{ for all } k \geq 1.$$

Proof. The proof follows directly from [19]. □

Theorem 4.4. *Consider the $MDLS$ algorithm and suppose that assumptions (i), (ii) and (4.10) hold. Then either $g_k = 0$ for some k, or*

$$\lim_{k\to\infty} \inf \|g_k\| = 0. \tag{4.17}$$

Proof. We suppose that $g_k \neq 0$ for all k. Then we need to prove (4.17).

Suppose, on the contrary, that (4.17) doesn't hold. Then there exists a constant $t > 0$ such that

$$\|g_k\| \geq t, \text{ for all } k. \tag{4.18}$$

Let D be the diameter of the level set $\mathcal{L}$.

We have

$$|\beta_k^{DLS}| \leq -\frac{\|g_{k+1}\|^2}{g_k^T d_k} \leq \frac{\mu^2}{ct^2}$$

$$|\beta_k^{LS}| = \frac{g_{k+1}^T y_k}{-g_k^T d_k} \leq \frac{\mu LD}{ct^2}.$$

On the one hand, we have

$$\begin{aligned} \|d_{k+1}\| &= \|g_{k+1}\| + (1-\lambda_k)|\beta_k^{MDLS}|\|d_k\| + \lambda_k|\theta_k|\|g_k\| \\ &= \|g_{k+1}\| + (1-\lambda_k)|\beta_k^{MDLS}|\frac{\|s_k\|}{\alpha_k} + \lambda_k|\theta_k|\|g_k\| \end{aligned} \tag{4.19}$$

From lemma 4.3, we get two conditions

1. If $\|g_{k+1}\|^2 \geq \left|g_{k+1}^T g_k\right|$, then (4.19) becomes

$$\|d_{k+1}\| \leq \mu + \frac{\mu LD}{\alpha^\star ct^2} + \eta\mu = C, \tag{4.20}$$

   where C is a constant.

2. If $\|g_{k+1}\|^2 < \left|g_{k+1}^T g_k\right|$, but in this case we have $\left|g_{k+1}^T g_k\right| \geq 0.5\|g_{k+1}\|^2$, then (4.19) becomes

$$\|d_{k+1}\| \leq \mu + \frac{\mu^2}{\alpha^\star ct^2} + 2\eta\mu = C, \tag{4.21}$$

   where C is a constant.

Now, we obtain

$$\sum_{k\geq 1} \frac{1}{\|d_{k+1}\|^2} = \infty. \tag{4.22}$$

On the other hand, from (4.10), (4.18), and from the Zoutendijk condition (4.16), it results that

$$c^2 t^4 \sum_{k\geq 0} \frac{1}{\|d_k\|^2} \leq \sum_{k\geq 0} \frac{c^2\|g_k\|^4}{\|d_k\|^2} \leq \sum_{k\geq 0} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty,$$

which contradicts (4.22). Therefore, (4.18) doesn't hold. Then,

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

□

4.4 Numerical experiments

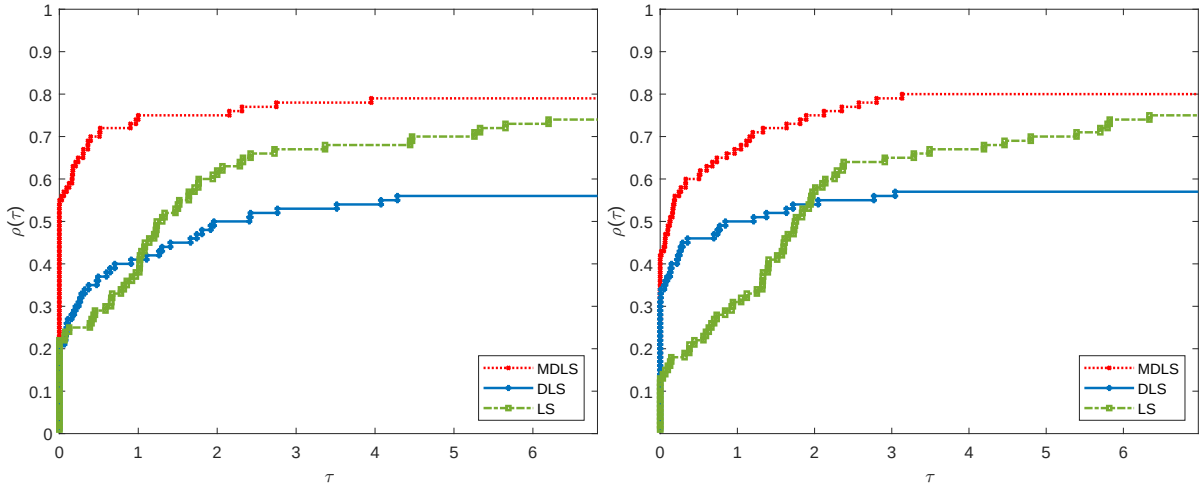
In this section, we present the performance of our new MDLS algorithm via some theoretical standard tests problems and practical applications.

4.4.1 Standard theoretical tests problems

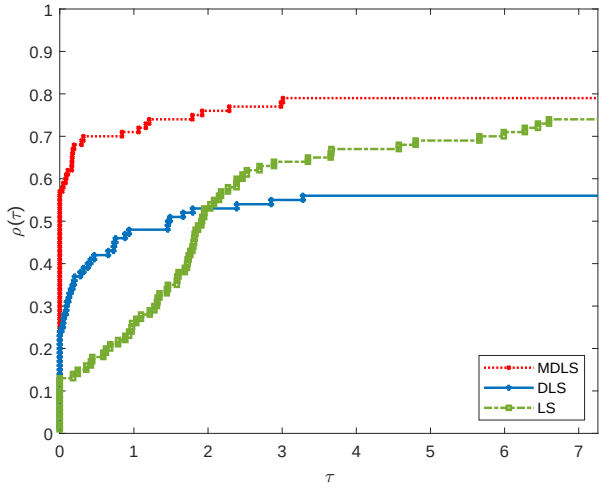
We have selected various test functions given in table 4.1, taken from the CUTE library of [1], [2], [26] and [50] for different dimensions, from $n = 10$ to $n = 800000$. At the same time, we present a numerical comparison with other conjugate gradient algorithm (LS, DLS). We set the parameter $\eta = 0.4$, $\varepsilon = 10^{-6}$ and the stopping criterion is $\|g_k\| \leq \varepsilon$ or the iteration number > 4000. The stepsize α_k is calculated using the strong Wolfe line search with $\sigma = 0.01$ $\delta = 0.1$. In order to evaluate the data of iterations number, CPU time, gradient evaluation and objective function evaluation, we use the performance profile of Dolan and Moré [20] presented in figure 4.1.

Table 4.1: List of test functions and their dimensions.

Tests functions	**Dimensions**	**Tests functions**	**Dimensions**
Cosine	6000, 100000, 800000	Tridia	300, 2000
Dixmaana	6000, 90000	Woods	150000, 200000
Dixmaanb	24000, 48000	Bdexp	5000, 50000, 500000
Dixmaanc	2700, 27000	Exdenschnf	90000, 280000, 600000
Dixmaand	12000, 90000	Exdenschnb	6000, 24000, 300000
Dixmaane	2400, 48000	Genquartic	9000, 90000, 500000
Dixmaanf	15000, 60000	Biggsb1	300
Dixmaang	12000, 90000	Sine	100000, 250000, 500000
Dixmaanh	6000, 150000	Fletcbv3	100
Dixmaani	360	Nonscomp	5000, 80000
Dixmaanj	3000, 15000	Power1	150
Dixmaank	12000, 12000	Raydan1	500, 5000
Dixmaanl	2400, 24000	Raydan2	2000, 20000, 500000
Dixon3dq	150	Diagonal1	800, 2000
Dqdrtic	9000, 90000	Diagonal2	8000, 50000
Dqrtic	5000, 150000	Diagonal3	500, 2000
Edensch	7000, 40000, 50000	Bv	2000, 20000
Eg2	100	Ie	500, 1500
Fletchcr	1000, 50000, 200000	Singx	1000, 2000
Freuroth	460	Lin	100, 1300
Genrose	10000	Osb2	10
Himmelbg	70000, 240000	Pen1	200, 1000
Liarwhd	6000, 30000	Pen2	160
Penalty1	4000, 10000	Rosex	500, 1000
Quartc	80000, 50000	Trid	500, 8000



(a) Iterations number (b) CPU time



(c) Objective function evaluation

Figure 4.1: Performance profile of MDLS, DLS and LS.

4.4.2 Practical applications

Image restoration

In this part, we evaluate the performance of the MDLS algorithm in comparison with the other conjugate gradient methods (LS and DLS) for image restoration problems effected by impulse noise using the same data given in chapter 2. In this processing, we use the two-phase restoration scheme proposed by Chan et al. [11], which has proven effective in handling high-density of salt-and-pepper noise. The experiments are conducted on three grayscale images of size 512×512 *Lady*, *Baboon*, and *Bridge*. We test the performance of the algorithms under various levels of salt-and-pepper noise: 30%, 50%, 70%, and 90%.

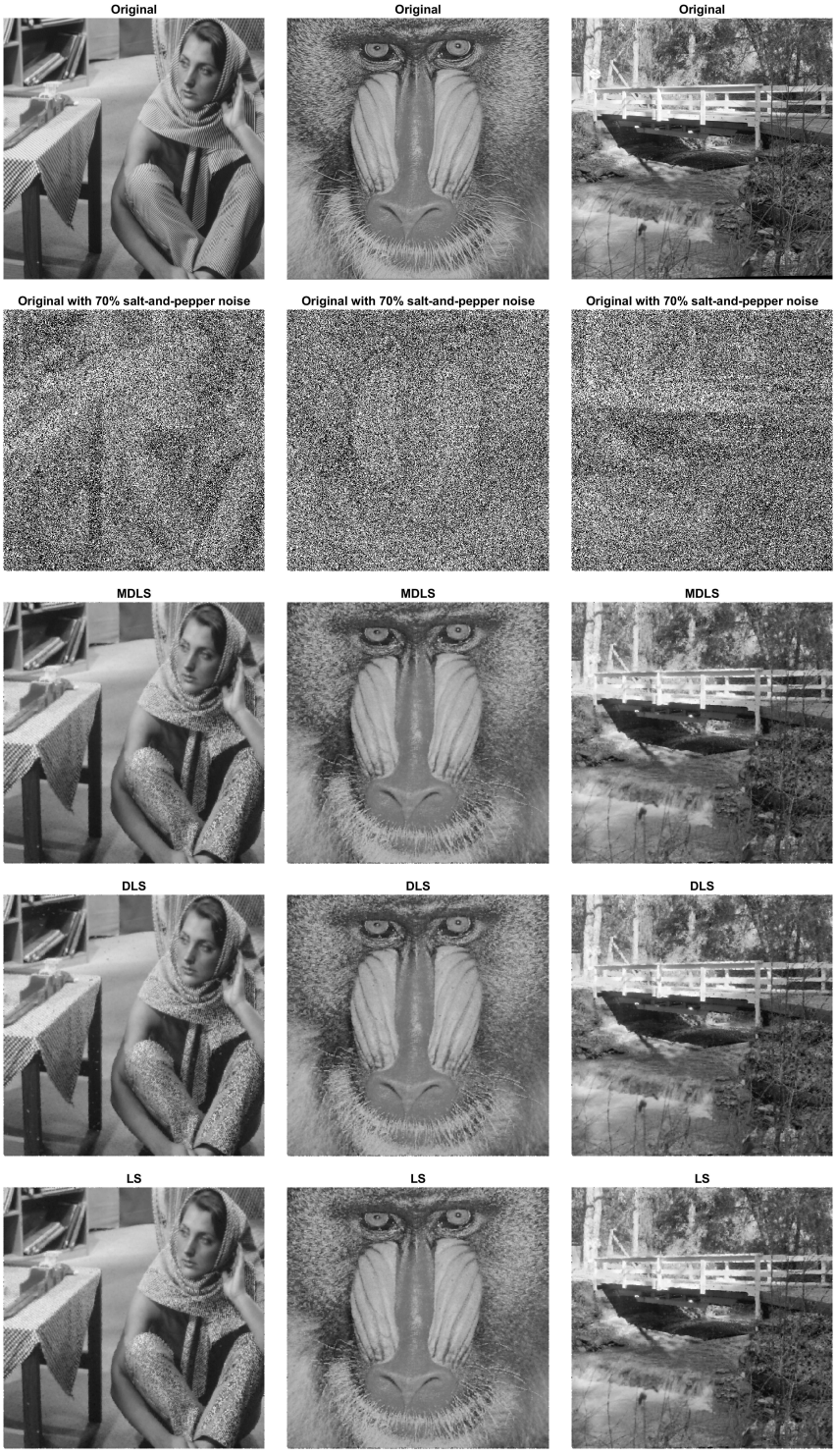


Figure 4.2: The noisy images with 70% salt-and-pepper noise and the restored images by MDLS, DLS and LS.

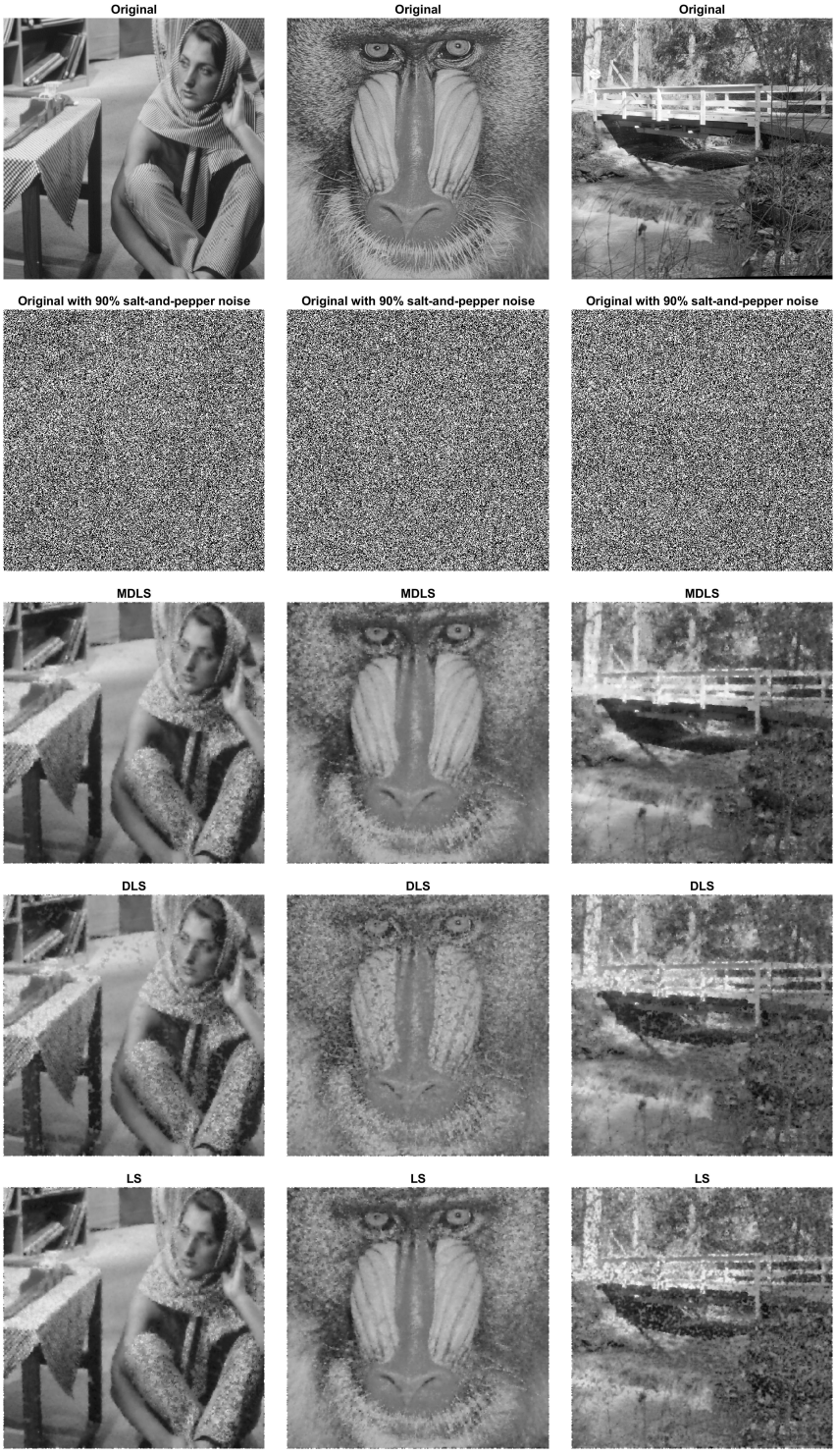


Figure 4.3: The noisy images with 90% salt-and-pepper noise and the restored images by MDLS, DLS and LS.

Table 4.2: Performance results for different images and compression Levels

Image/Noise	MDLS			DLS			LS		
	Iter	Time	PSNR	Iter	Time	PSNR	Iter	Time	PSNR
Barbara/0.30	16	8.73	28.39	55	13.71	28.40	15	14.40	28.27
Barbara/0.50	19	16.26	26.37	34	23.29	26.39	16	22.55	26.13
Barbara/0.70	16	21.43	24.36	56	37.22	24.34	24	43.75	24.37
Barbara/0.90	28	36.05	22.11	105	81.23	21.64	30	61.19	22.09
Baboon/0.30	18	7.52	27.91	39	11.95	27.89	15	9.11	27.84
Baboon/0.50	13	12.37	25.96	27	19.12	25.93	16	23.75	25.60
Baboon/0.70	18	28.81	23.80	75	38.74	23.75	16	25.47	23.75
Baboon/0.90	27	37.59	21.38	63	46.15	20.39	31	63.04	21.37
Bridge/0.30	15	8.54	28.54	31	11.35	28.39	14	8.23	28.46
Bridge/0.50	20	21.05	26.61	41	27.19	26.58	22	26.86	26.30
Bridge/0.70	24	23.80	24.45	88	55.35	24.36	26	46.82	24.42
Bridge/0.90	37	41.61	21.42	75	76.54	20.20	18	91.73	19.43

Sparse signal recovery

This part is dedicated to make a comparison over the three algorithms and analyze the effectiveness of our proposed MDLS algorithm for solving sparse signal recovery problems. Sparse signal recovery is a critical problem in signal processing, where the objective is to recover a sparse signal from noisy or incomplete measurements. The optimization problem can be formulated as

$$\min_{x\in\mathbb{R}^n} f(x) = \frac{1}{2}\|Ax-b\|^2 + \theta\|x\|_1,$$

where $A \in \mathbb{R}^{m\times n}$ (with $m < n$) is the measurement matrix, $b \in \mathbb{R}^m$ represents the observed data, $\|x\|_1 = \mathbf{e}_n^T x$ with $\mathbf{e}_n = (1, 1, \ldots, 1)^T \in \mathbb{R}^n$ and $\theta > 0$ is a regularization parameter controlling the trade-off between data fidelity and sparsity [10]. The effectiveness of signal recovery is typically evaluated using the mean squared error (MSE) given by

$$\text{MSE} = \frac{1}{n}\|x^\star - x\|^2,$$

where $x^\star$ is the recovered signal and x the original sparse signal. Note that a low MSE means that the estimated signal is very close to the true signal, indicating better

performance. In our tests, N represents the sparsity level which equals to 32. The size of the problems n=4096 and m=1024. The regularization parameter $\theta = 2.9 \times 10^{-3}$.

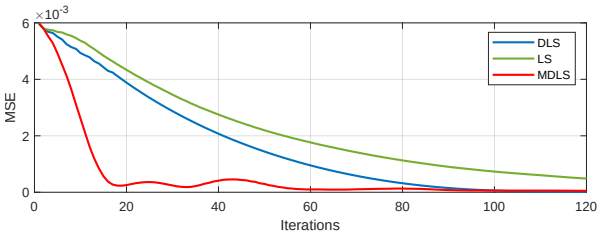


Figure 4.4: MSE in terms of iterations.

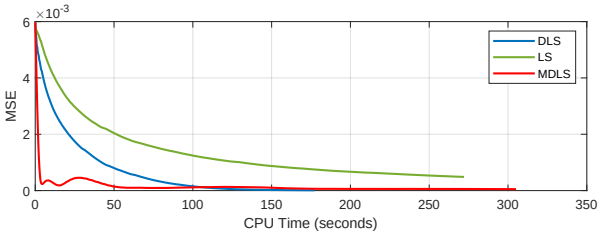


Figure 4.5: MSE in temrs of CPU time.

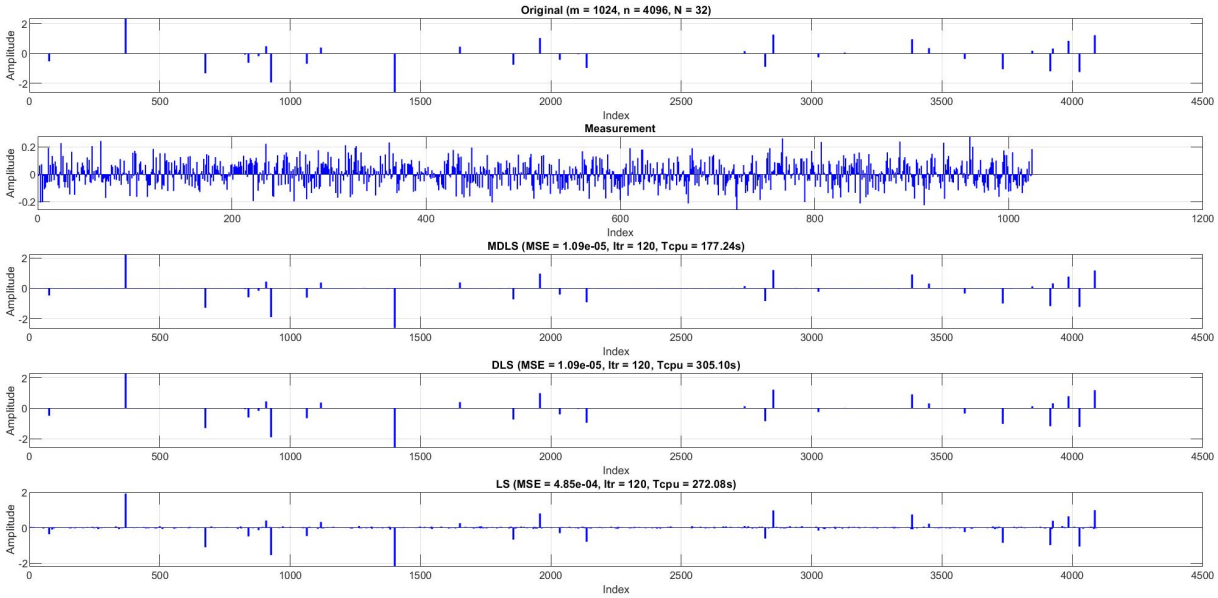


Figure 4.6: The original sparse signal compared to the restored signals.

4.4.3 Comments

The performance profile results, as depicted in Figures (4.1a), (4.1b) and (4.1c), show that our proposed conjugate gradient method MDLS, using β_k^{MDLS}, consistently outperforms methods based on β_k^{DLS} and β_k^{LS} for unconstrained optimization problems.

In the context of image restoration, the MDLS method strikes an optimal balance between computational speed and image quality, particularly in scenarios with high levels of noise. It achieves superior PSNR values compared to other methods while requiring fewer iterations than the DLS and LS approaches.

For sparse signal restoration, Figures 4.4, 4.5, and 4.6 highlight the robustness and efficiency of the MDLS algorithm in addressing sparse signal recovery problems.

It shows faster convergence and reduced computational costs, offering a significant improvement over the other considered algorithms.

4.5 Conclusion

We proposed a hybrid conjugate gradient method for solving unconstrained optimization problems, where the parameter β_k^{MDLS} combines β_k^{LS} and β_k^{DLS}. Additionally, we employed a three-term search direction to enhance algorithm performance. The sufficient descent condition was established, and global convergence was proven under the strong Wolfe line search. Numerical results validate the MDLS method's efficiency compared to LS and DLS. The proposed algorithm consistently reduced iterations and computational time while maintaining robust performance in image restoration and sparse signal recovery, demonstrating its potential for solving various optimization problems.

CHAPTER 5

New conjugate gradient method of Wei-Yao-Liu type applied to image restoration and compressive sensing problems

The conjugate gradient method is a commonly used iterative method for solving high-dimensional optimization problems because of its efficiency and minimal memory requirements. In this chapter, we propose a new conjugate gradient parameter of the type Wei-Yao-Liu, based on the Hestenes-Stiefel approach. Based on strong Wolfe conditions,the sufficient descent direction and the global convergence are proved. Numerical experiments of the proposed algorithm applied to unconstrained optimization problems, image restoration and compressive sensing problems reveal its accuracy and reliability. This work was accepted in **Algerian Journal of Science** [33].

5.1 Introduction

The conjugate gradient approach is the most commonly techniques which solve large-scale unconstrained optimization problems. Its efficiency and low memory requirements make it particularly suitable for problems where direct methods are computationally expensive. This method has been successfully applied in various fields such that image processing, compressive sensing, deep learning, medical imaging, robotic motion controls and portfolio optimization. Let be $f : \mathbb{R}^n \to \mathbb{R}$ a continuously differentiable

function. Our goal is to solve the unconstrained optimization problem: " $\min_{x\in\mathbb{R}^n} f(x)$".

Recall that the principle of the conjugate gradient method is to construct a sequence

$$x_0 \in \mathbb{R}^n,\ x_{k+1} = x_k + \alpha_k d_k,\ \text{for } k \geq 0, \tag{5.1}$$

where α_k is a stepsize need to be computed by a selected line search method at each iteration and d_k is a descent direction given by

$$d_0 = -g_0 = -\nabla f(x_0),\ d_{k+1} = -g_{k+1} + \beta_k d_k,\ \text{for } k \geq 0, \tag{5.2}$$

where $g_{k+1} = \nabla f(x_{k+1})$ is the gradient of f at x_{k+1} and β_k is a conjugacy parameter.

In 2006, Wei et al. [68] proposed a new parameter β_k by modifying the vector y_k used in the PRP method. The resulting expression of their β_k is given as follows

$$\beta_k^{WYL} = \frac{g_{k+1}^T \tilde{y}_k}{\|g_k\|^2}, \tag{5.3}$$

where $\tilde{y}_k = g_{k+1} - \frac{\|g_{k+1}\|}{\|g_k\|} g_k$.

Motivated by the above idea, Saleh Nazzal Alsuliman et al. [5] proposed a modification of the Wei-Yao-Liu (WYL) method based on the LS parameter, defined as follows

$$\beta_k^{SFA} = -\frac{g_{k+1}^T \tilde{y}_k}{g_k^T d_k}. \tag{5.4}$$

The objective of this chapter is to propose a new WYL parameter, resulting in a new conjugate gradient method with an improved descent direction.

5.2 New modified conjugate gradient algorithm

Starting from the idea introduced in WYL (5.3) and SFA (5.4), and incorporating the parameter β_k^{HS}, we give our new formula of the parameter β_k as follows

$$\beta_k^{ESDB} = \frac{g_{k+1}^T \tilde{y}_k}{d_k^T \tilde{y}_k}. \tag{5.5}$$

with $\tilde{y}_k = g_{k+1} - \frac{\|g_{k+1}\|}{\|g_k\|} g_k$.

We take search direction given as follows:

$$d_0 = -g_0 = -\nabla f(x_0),\ d_{k+1} = -\left(1 + \beta_k^{ESDB} \frac{g_{k+1}^T d_k}{\|g_{k+1}\|^2}\right) g_{k+1} + \beta_k^{ESDB} d_k,\ \text{for } k \geq 0. \tag{5.6}$$

The stepsize α_k is chosen to satisfy the following strong Wolfe conditions

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \delta \alpha_k g_k^T d_k, \tag{5.7}$$

$$\left| g_{k+1}^T d_k \right| \leq \sigma |g_k^T d_k|, \tag{5.8}$$

where $0 < \delta < \sigma < 1$.

The algorithm corresponding to β_k^{ESDB} is given bellow

Algorithm 14: ESDB Algorithm

Step 0: Start with an initial point x_0 and a parameter $\varepsilon > 0$.

Step 1: Set $k = 0$ and compute $d_0 = -g_0 = -\nabla f(x_0)$.

Step 2: If $\|g_k\| \leq \varepsilon$, **Stop**; else go to **Step 3**.

Step 3: Compute a stepsize $\alpha_k \in (0, 1]$ satisfying the strong Wolfe conditions (5.7) and (5.8).

Step 4: Compute $x_{k+1} = x_k + \alpha_k d_k$.

Step 5: Compute $g_{k+1} = \nabla f(x_{k+1})$ and $\tilde{y}_k = g_{k+1} - \dfrac{\|g_{k+1}\|}{\|g_k\|} g_k$.

Step 6: Compute $\beta_k = \beta_k^{ESDB}$ using equation (5.5).

Step 7: If $|g_{k+1}^T g_k| \geq 0.2\|g_{k+1}\|$ (Powell's restart criterion [59]), then

$$d_{k+1} = -g_{k+1};$$

else

$$d_{k+1} = -\left(1 + \beta_k^{ESDB} \frac{g_{k+1}^T d_k}{\|g_{k+1}\|^2}\right) g_{k+1} + \beta_k^{ESDB} d_k.$$

Step 8: Set $k = k + 1$ and go to **Step 2**.

5.3 Convergence analysis

We introduce the following theorem to establish the sufficient descent condition.

Theorem 5.1. *Let $\{g_k\}$ and $\{d_k\}$ be the sequences generated by $ESDB$ algorithm. Then, the search direction of (5.6) verifies*

$$g_{k+1}^T d_{k+1} \leq 0, \text{ for all } k \geq 0, \tag{5.9}$$

which means that d_k is a descent direction.

Proof. If $k = 0$, then $g_0^T d_0 = -\|g_0\|^2$, so (5.9) holds.

If $|g_{k+1}^T g_k| \geq 0.2\|g_{k+1}\|^2$, then the search direction is chosen as $d_{k+1} = -g_{k+1}$, which directly yields

$$g_{k+1}^T d_{k+1} = -\|g_{k+1}\|^2.$$

Otherwise, the search direction d_{k+1} is defined by

$$d_{k+1} = -\left(1 + \beta_k^{ESDB} \frac{g_{k+1}^T d_k}{\|g_{k+1}\|^2}\right) g_{k+1} + \beta_k^{ESDB} d_k. \tag{5.10}$$

The multiplication of both sides of (5.10) with g_{k+1}^T gives

$$\begin{aligned} g_{k+1}^T d_{k+1} &= -\|g_{k+1}\|^2 \left(1 + \beta_k^{ESDB} \frac{g_{k+1}^T d_k}{\|g_{k+1}\|^2}\right) + \beta_k^{ESDB} g_{k+1}^T d_k \\ &= -\|g_{k+1}\|^2 - \beta_k^{ESDB} g_{k+1}^T d_k + \beta_k^{ESDB} g_{k+1}^T d_k \\ &= -\|g_{k+1}\|^2. \end{aligned}$$

Thus, condition (5.9) is always satisfied, which concludes the proof. □

To prove the global convergence of the algorithm, we make the following assumptions:

Assumption(i) The level set $S = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ is bounded; that is, there exists a constant $B > 0$ such that

$$\|x\| \leq B, \quad \forall x \in S.$$

Assumption(ii) In a neighborhood $\mathcal{N}$ of S, the gradient $\nabla f(x)$ is Lipschitz continuous; i.e., there exists a constant $L > 0$ such that

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \quad \forall x, y \in \mathcal{N}.$$

Remark 5.2. Under assumptions (i) and (ii), there exists a constant $\kappa \geq 0$ such that

$$\|\nabla f(x)\| \leq \kappa, \quad \forall x \in S. \tag{5.11}$$

Lemma 5.3. *Suppose that assumptions (i) and (ii) hold. Consider common iterate (5.1), where d_k is a descent direction (5.4) and α_k is determined by the strong Wolfe line search (5.5) and (5.6). Then, the Zoutendijk condition*

$$\sum_{k \geq 0} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty, \tag{5.12}$$

holds.

Proof. The proof follows directly from [73]. □

Lemma 5.4. *Assume that assumptions **(i)** and **(ii)** hold, and that $\alpha_k > 0$ is computed using the strong Wolfe line search conditions (5.6) and (5.7). So, there exists a constant $\alpha^\star > 0$ satisfies*

$$\alpha_k \geq \alpha^\star > 0, \quad \forall k \geq 0.$$

Proof. The proof follows directly from [19]. □

Theorem 5.5. *Under the assumptions (i), (ii), and (5.9), the ESDB algorithm converges and satisfies*

$$\lim_{k\to\infty} \inf \|g_k\| = 0. \tag{5.13}$$

Proof. Assume that, $\forall k,\ g_k \neq 0$. Our objective is to prove (5.13).

Suppose that (5.13) does not hold. Then, there exists a constant $\lambda > 0$ such that

$$\|g_k\| \geq \lambda, \quad \forall k \geq 0. \tag{5.14}$$

Consider $D > 0$, the diameter of the level set S and the ESDB conjugate gradient parameter:

$$\beta_k^{\text{ESDB}} = \frac{g_{k+1}^T \tilde{y}_k}{d_k^T \tilde{y}_k}, \quad \text{where } \tilde{y}_k = g_{k+1} - \frac{\|g_{k+1}\|}{\|g_k\|} g_k.$$

We aim to demonstrate that $|\beta_k^{\text{ESDB}}|$ remains bounded under the strong Wolfe conditions (5.7) and (5.8), the sufficient descent property (5.9).

Let us expand the denominator $d_k^T \tilde{y}_k$ of β_k^{ESDB}

$$d_k^T \tilde{y}_k = d_k^T g_{k+1} - \frac{\|g_{k+1}\|}{\|g_k\|} d_k^T g_k.$$

From (5.8) and (5.9), we obtain

$$d_k^T \tilde{y}_k \leq -\sigma g_k^T d_k - \frac{\|g_{k+1}\|}{\|g_k\|} g_k^T d_k = -\left(\sigma + \frac{\|g_{k+1}\|}{\|g_k\|}\right) g_k^T d_k.$$

Since $g_k^T d_k = -\|g_k\|^2$, it follows that

$$d_k^T \tilde{y}_k \leq -\left(\sigma + \frac{\|g_{k+1}\|}{\|g_k\|}\right) \|g_k\|^2.$$

For the numerator of β_k^{ESDB} and the Cauchy–Schwarz inequality we have:

$$g_{k+1}^T \tilde{y}_k = \|g_{k+1}\|^2 - \frac{\|g_{k+1}\|}{\|g_k\|} g_{k+1}^T g_k \geq \|g_{k+1}\|^2 - \|g_{k+1}\|^2 = 0.$$

Hence, we conclude that:

$$g_{k+1}^T \tilde{y}_k \geq 0.$$

Substitute bounds into β_k^{ESDB}:

$$|\beta_k^{\text{ESDB}}| = \frac{g_{k+1}^T \tilde{y}_k}{|d_k^T \tilde{y}_k|} \leq \frac{\|g_{k+1}\|^2 + \|g_{k+1}\|\|g_k\|}{\left(\sigma + \frac{\|g_{k+1}\|}{\|g_k\|}\right)\|g_k\|^2}.$$

Therefore

$$|\beta_k^{\text{ESDB}}| \leq \frac{2\kappa^2}{\lambda^2\sigma + \lambda} = P. \tag{5.15}$$

From (5.10), we have

$$d_{k+1} = -\left(1 + \beta_{k+1}^{\text{ESDB}} \frac{g_{k+1}^T d_k}{\|g_{k+1}\|^2}\right) g_{k+1} + \beta_{k+1}^{\text{ESDB}} d_k,$$

and hence,

$$\|d_{k+1}\| \leq \|g_{k+1}\| + \|g_{k+1}\| \left|\beta_{k+1}^{\text{ESDB}}\right| \frac{|g_{k+1}^T d_k|}{\|g_{k+1}\|^2} + \left|\beta_k^{\text{ESDB}}\right| \|d_k\|.$$

From Cauchy–Schwarz inequality with relation (5.15), we found

$$\begin{aligned}
\|d_{k+1}\| &\leq \|g_{k+1}\| + P\|g_{k+1}\| \frac{|g_{k+1}^T d_k|}{\|g_{k+1}\|^2} + P\|d_k\| \\
&= \|g_{k+1}\| + P\frac{|g_{k+1}^T d_k|}{\|g_{k+1}\|} + P\|d_k\| \\
&\leq \|g_{k+1}\| + P\|d_k\| + P\|d_k\| \\
&= \|g_{k+1}\| + 2P\|d_k\|.
\end{aligned}$$

As $x_{k+1} - x_k = \alpha_k d_k$, we write $d_k = \dfrac{x_{k+1} - x_k}{\alpha_k}$. So, from (5.11) and lemma 5.3, we have

$$\|d_{k+1}\| \leq \kappa + 2P \left\| \frac{x_{k+1} - x_k}{\alpha_k} \right\|,$$

we also have $x_k,\ x_{k+1} \in S$, so $\|x_{k+1} - x_k\| \leq D$

$$\|d_{k+1}\| \leq \kappa + 2P\frac{D}{\alpha_*} = C,$$

where C is a constant. So, we obtain

$$\sum_{k\geq 0} \frac{1}{\|d_{k+1}\|^2} = \infty. \tag{5.16}$$

Besides, combining inequalities (5.9) and (5.14) with the Zoutendijk condition (5.12), we obtain:

$$\lambda^4 \sum_{k\geq 0} \frac{1}{\|d_k\|^2} \leq \sum_{k\geq 0} \frac{\|g_k\|^4}{\|d_k\|^2} = \sum_{k\geq 0} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty.$$

This implies that $\sum_{k\geq 0} \frac{1}{\|d_k\|^2}$ is convergent, which contradicts condition (5.16). Therefore, assumption (5.14) must be not true.

Consequently, we conclude that

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

□

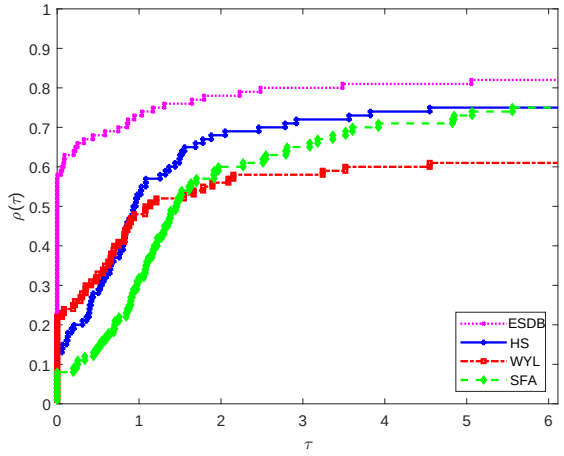
5.4 Numerical experiments

5.4.1 Standard theoretical tests problems

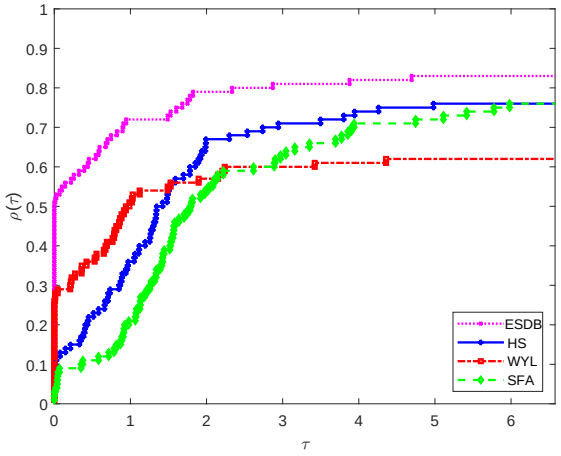
In this section, we present several numerical tests to evaluate the performance of ESDB algorithm. We select various test functions given in Table 5.1, taken from the CUTE library [1, 2, 26, 50] for different dimensions, from $n = 10$ to $n = 800000$. At the same time, we present a numerical comparison with other conjugate gradient algorithm namely HS, WYL and SFA. The stopping criterion is $\|g_k\| \leq \varepsilon$ where $\varepsilon = 10^{-6}$, or the iteration number> 4000. The stepsize α_k is computed by strong Wolfe line search with $\sigma = 0.01$ $\delta = 0.1$. All experiments were conducted using MATLAB R2021a on a machine running Windows 8.1, without requiring any additional toolboxes. The computations were executed on a computer equipped with an Intel Core i5-6300U CPU @ 2.4 GHz, 8 GB of RAM, and no dedicated GPU. This setup represents a typical personal computing environment. In order to evaluate the data of iterations number, CPU time, and objective function evaluation, we use the performance profile of Dolan and Moré [20] presented in figure 5.1.

Tests functions	**Dimensions**	**Tests functions**	**Dimensions**
Cosine	6000, 100000, 800000	Tridia	300, 2000
Dixmaana	6000, 90000	Woods	150000, 200000
Dixmaanb	24000, 48000	Bdexp	5000, 50000, 500000
Dixmaanc	2700, 27000	Exdenschnf	90000, 280000, 600000
Dixmaand	12000, 90000	Exdenschnb	6000, 24000, 300000
Dixmaane	2400, 48000	Genquartic	9000, 90000, 500000
Dixmaanf	15000, 60000	Biggsb1	300
Dixmaang	12000, 90000	Sine	100000, 250000, 500000
Dixmaanh	6000, 150000	Fletcbv3	100
Dixmaani	360	Nonscomp	5000, 80000
Dixmaanj	3000, 15000	Power1	150
Dixmaank	12000, 12000	Raydan1	500, 5000
Dixmaanl	2400, 24000	Raydan2	2000, 20000, 500000
Dixon3dq	150	Diagonal1	800, 2000
Dqdrtic	9000, 90000	Diagonal2	8000, 50000
Dqrtic	5000, 150000	Diagonal3	500, 2000
Edensch	7000, 40000, 50000	Bv	2000, 20000
Eg2	100	Ie	500, 1500
Fletchcr	1000, 50000, 200000	Singx	1000, 2000
Freuroth	460	Lin	100, 1300
Genrose	10000	Osb2	10
Himmelbg	70000, 240000	Pen1	200, 1000
Liarwhd	6000, 30000	Pen2	160
Penalty1	4000, 10000	Rosex	500, 1000
Quartc	80000, 50000	Trid	500, 8000

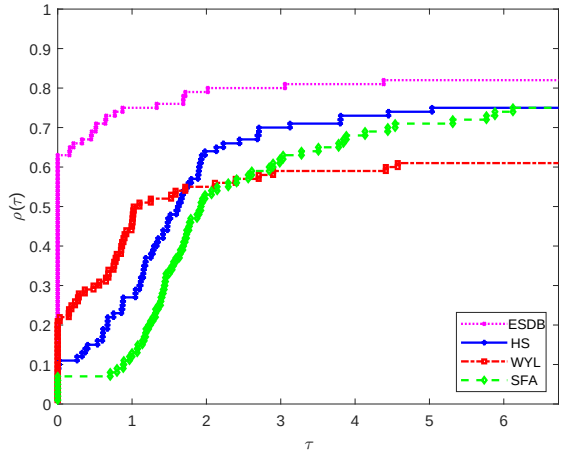
Table 5.1: List of test functions and their dimensions.



(a) Iteration number



(b) CPU time



(c) Objective function evaluation

Figure 5.1: Performance profile of ESDB, HS, WYL and SFA.

5.4.2 Practical applications

Image restoration

In this part, we evaluate the performance of the ESDB algorithm in comparison with the other conjugate gradient method (HS, WYL and SFA) for image restoration problems effected by impulse noise, using the same data as in chapter 2. In this processing, we use the two-phase restoration scheme proposed by Chan et al. [11], which has proven effective in handling high-density of salt-and-pepper noise. The experiments are conducted on three grayscale images of size 512×512: *Yasser*, *Cat*, and *Faculty of Sciences*. We test the performance of the algorithms under various levels of salt-and-pepper noise: 30%, 50%, 70%, and 90%.

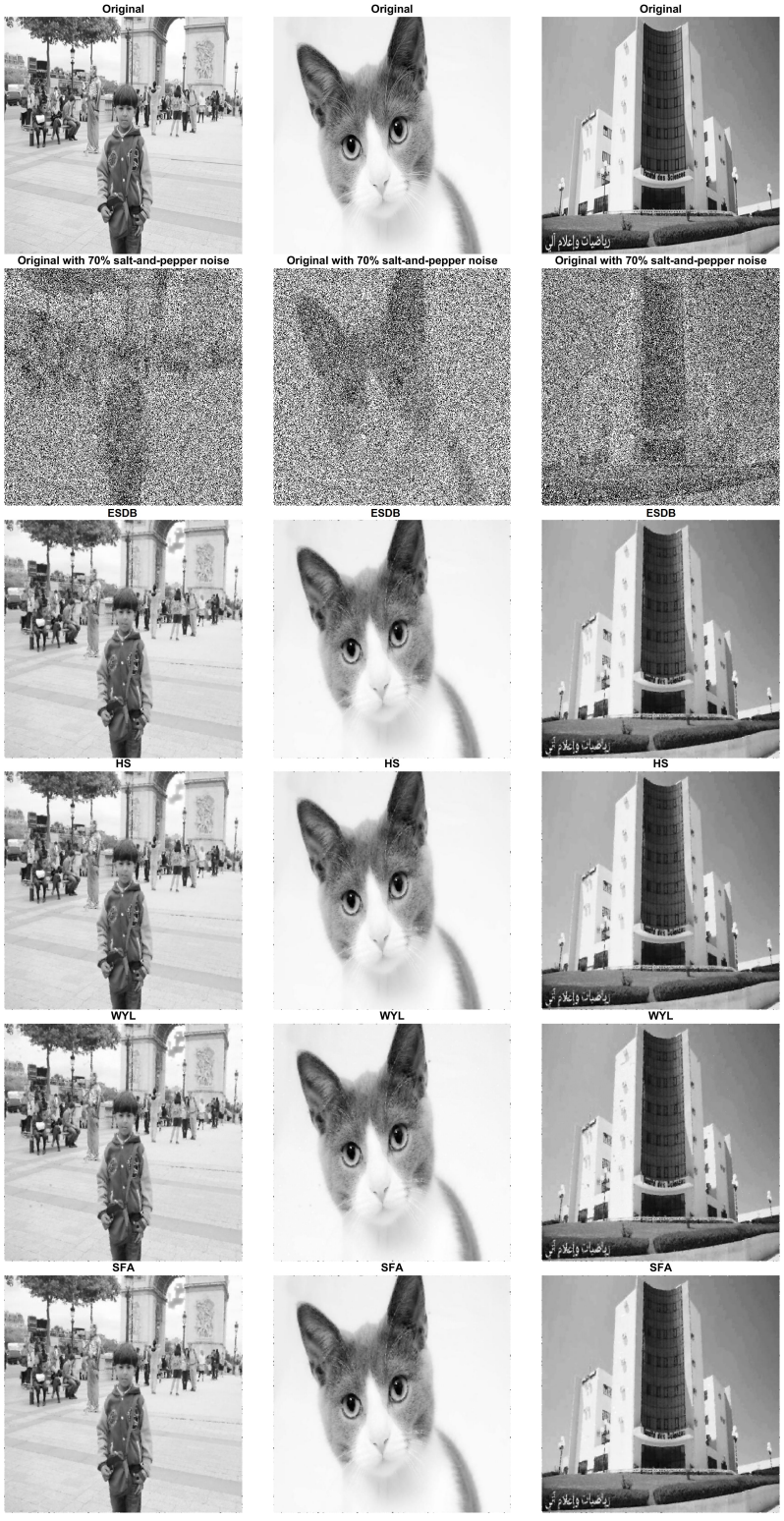


Figure 5.2: Restoration of 70% salt-and-pepper noisy images using ESDB, HS, WYL, and SFA.

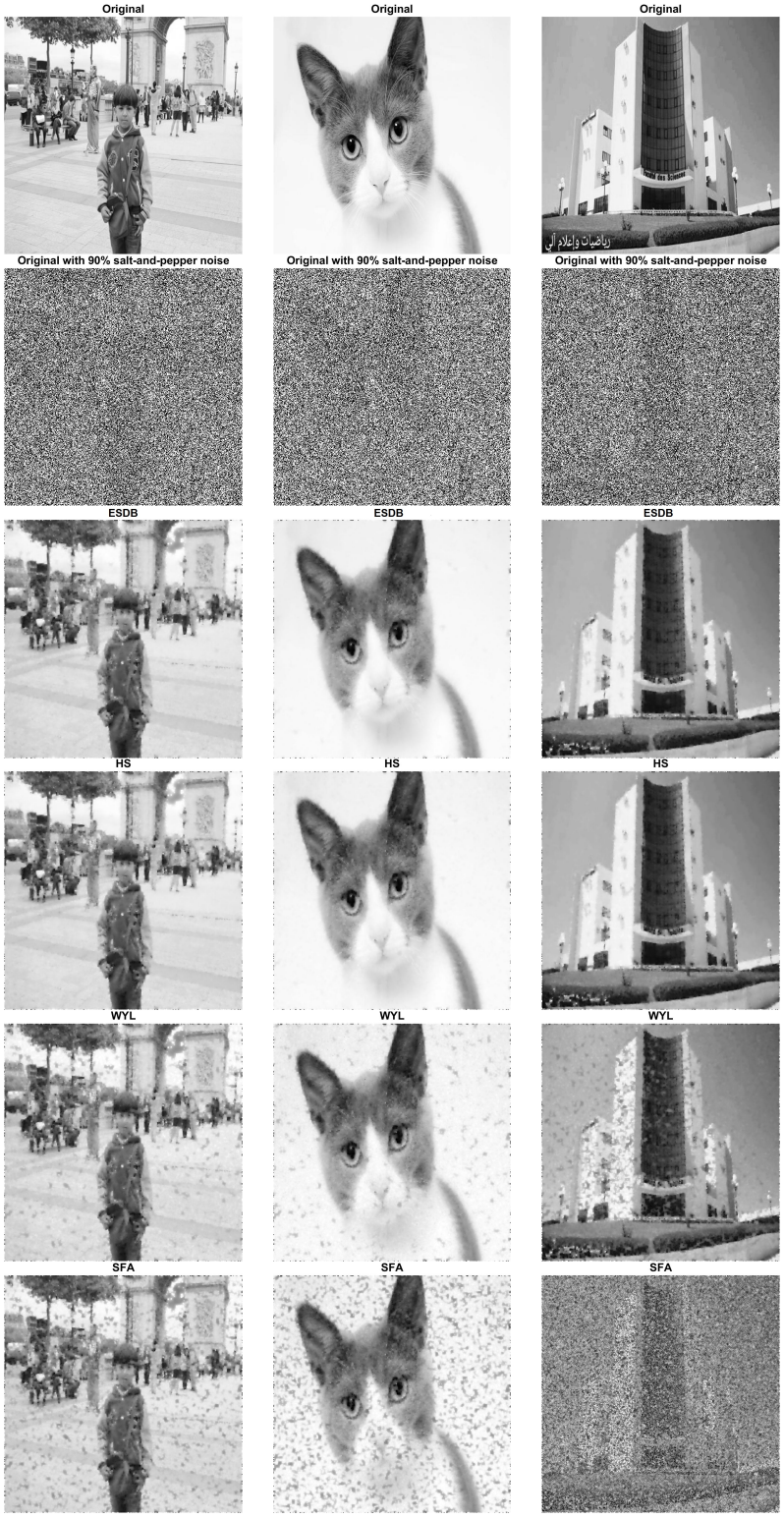


Figure 5.3: Restoration of 90% salt-and-pepper noisy images via ESDB, HS, WYL, and SFA.

Image/Noise	ESDB			HS			WYL			SFA		
	ITER	CPU	PSNR	ITER	CPU	PSNR	ITER	CPU	PSNR	ITER	CPU	PSNR
Yasser/0.30	16	10.73	32.70	22	22.48	32.85	25	26.64	32.91	19	19.98	32.66
Yasser/0.50	21	17.41	29.77	21	19.57	29.67	18	31.53	29.08	25	36.31	28.87
Yasser/0.70	19	27.95	26.42	31	42.00	26.67	18	41.92	26.20	25	49.67	26.35
Yasser/0.90	30	48.72	22.18	33	53.95	22.26	26	68.77	21.44	22	69.71	20.66
Cat/0.30	16	11.50	38.17	16	9.27	38.17	18	15.08	38.17	22	22.46	37.91
Cat/0.50	20	28.84	35.75	22	27.09	35.67	26	44.51	35.67	25	30.13	35.64
Cat/0.70	20	25.39	32.06	29	34.39	32.14	25	43.81	31.97	27	41.07	32.12
Cat/0.90	32	41.49	27.33	31	36.68	27.25	30	113.42	25.84	25	139.83	20.72
Faculty/0.30	15	8.10	34.77	20	10.31	34.77	15	10.58	34.31	15	9.62	34.29
Faculty/0.50	17	12.55	31.84	19	19.45	31.82	12	24.51	25.59	19	21.27	31.24
Faculty/0.70	26	22.38	28.45	25	27.28	28.45	22	44.05	28.25	26	41.22	28.45
Faculty/0.90	34	36.60	24.15	30	35.22	24.18	21	50.35	21.74	5	23.62	11.94

Table 5.2: Numerical results for image restoration on Yasser, Cat, and FacS images.

Sparse signal recovery

In the third part, we made a comparison over three algorithms (ESDB, WYL and SFA) and we analyze the effectiveness of our proposed ESDB algorithm for solving sparse signal recovery problems, using the same data as in chapter 4.

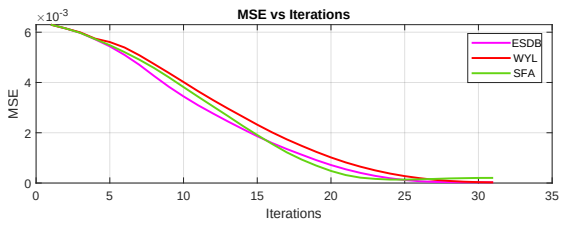


Figure 5.4: MSE in terms of iterations.

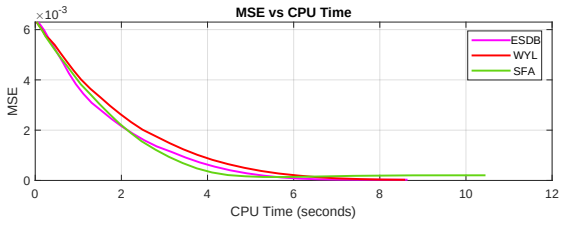


Figure 5.5: MSE in terms of CPU time.

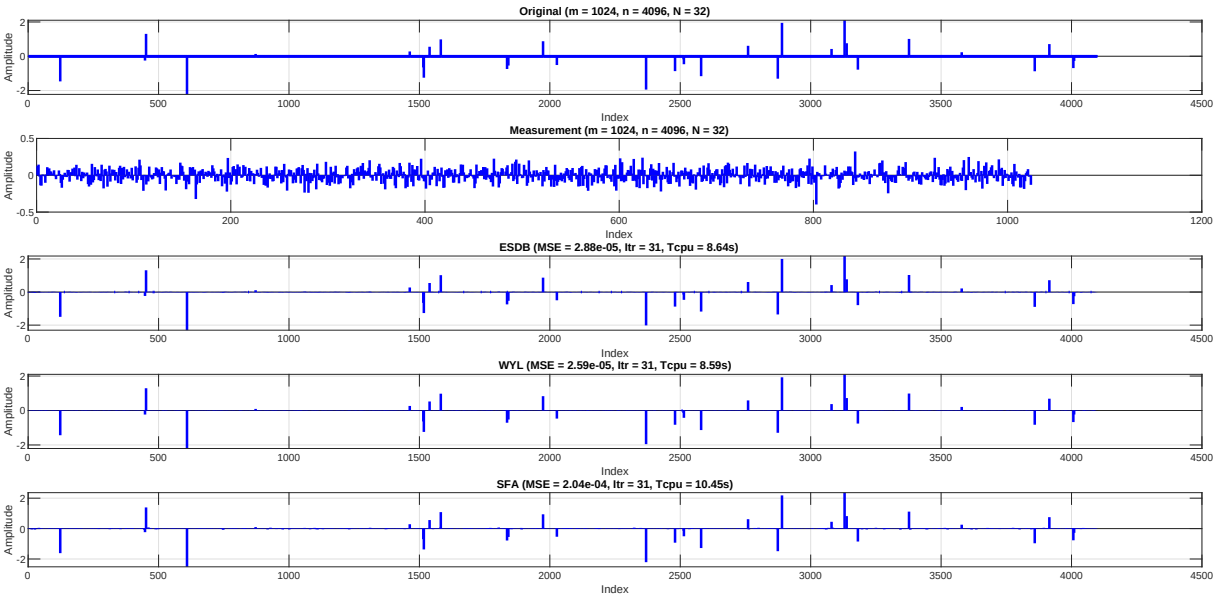


Figure 5.6: The original sparse signal compared to the restored signals.

5.4.3 Comments

The performance profile results shown in Figures 5.1a, 5.1b and 5.1c, which correspond to the number of iterations, CPU time and objective function evaluations, respectively, confirm that the proposed method using β_k^{ESDB} outperforms the conjugate gradient methods based on β_k^{HS}, β_k^{WYL} and β_k^{SFA} in solving unconstrained optimization problems. In the context of image restoration, from figures 5.2, 5.3 and from table 5.2, the ESDB method strikes an optimal balance between computational speed and image quality, particularly in scenarios with high levels of noise. It achieves superior PSNR values compared to other methods while requiring fewer iterations than the DLS and LS approaches. For sparse signal restoration, Figures 5.4, 5.5, and 5.6 show that ESDB, WYL, and SFA yield closely comparable results in addressing sparse signal recovery problems. Each method gives faster convergence and lower computational cost, highlighting a significant improvement in performance throughout all three approaches.

5.5 Conclusion

In this chapter, we have introduced a new variant of the WYL conjugate gradient method, constructed by incorporating the HS approach. We established the descent property of the proposed search direction and proved the global convergence of our

algorithm for solving nonlinear optimization problems under the strong Wolfe line search conditions. Numerical experiments have been conducted to show the efficiency and robustness of the proposed ESDB approach, in terms of number of iterations, computation time and objective function evaluation compared with other standard conjugate gradient method namely, HS, WYL and SFA algorithms for unconstrained optimization problems. The proposed algorithm consistently reduced iterations and computational time while maintaining robust performance in image restoration and compressive sensing, showing its potential for solving various optimization problems.

General conclusion and perspectives

This thesis has been dedicated to the advancement of conjugate gradient methods, with a central focus on the design, analysis, and application of novel hybrid algorithms. Throughout chapters 2 to 5 the successful development of a family of efficient and robust optimization techniques that address the limitations of classical conjugate gradient methods have been developed. By strategically blending existing formulas and incorporating innovative search directions, we have created algorithms that show superior performance on both standard benchmark problems and challenging real-world applications.

Starting with hybridization based on the principle of convex combination, we introduced a series of new conjugate gradient methods such as RMILHS, RMILFR, and RMILCDLS. The goal was to avoid relying on a single conjugate gradient parameter and instead dynamically combine the strengths of multiple parameters. For instance, the RMILHS method merges the robustness of the RMIL method with the efficient performance of the HS method, using a parameter θ_k, to align the search direction towards the Newton direction. Similarly, RMILFR combines the robust convergence of FR with RMIL methods, while the three combination of RMILCDLS method represents a more sophisticated strategy to build a highly resilient and effective search direction. For each algorithm, we provided a theoretical proofs, where we proved that the search directions satisfy the sufficient descent condition and establishing their global convergence under the strong Wolfe line search. The numerical experiments show that these hybrid methods consistently outperform their classical and existing hybrid counterparts in terms of iteration number, CPU time, and number of function evaluations across a vast set of large-scale test problems. Their practical utility was further confirmed through successful application to image restoration, where they effectively recovered images corrupted by high levels of impulse noise.

We have also extended the RMILHS hybrid method from basic unconstrained optimization to the more difficult problem of solving large-scale nonlinear monotone equations with convex constraints. To achieve this, we introduced the YSD algorithm, which combines the RMILHS strategy with the projection framework of Solodov and Svaiter. One of the main strengths of the YSD method is that it does not require storing or inverting matrices, making it very suitable for large problems. We proved that the method is globally convergent, and numerical results showed that it is both efficient and robust. Its successful application to a compressive sensing problem further shown its practical value.

In addition, we developed a new hybrid three-term conjugate gradient method. Unlike the traditional two-term conjugate gradient formula, this method adds an extra term to improve orthogonality and reduce the chance of stagnation. By combining the Liu–Storey (LS) and the modified Dai–Liao–Storey (DLS) parameters, we created a method that is globally convergent and performs well in practice. Numerical experiments, especially in image restoration and sparse signal recovery, shown that this three-term method is consistently reliable and effective.

Finally, we improved a specific conjugate gradient family by proposing a refined version of the Wei–Yao–Liu (WYL) method. The improved formula keeps the good properties of the original method while offering better performance in real computations. Tests on unconstrained optimization and image restoration problems confirmed that this new method is more efficient than the standard WYL, HS and SFA.

Overall, this thesis highlighted that carefully designed hybrid conjugate gradient methods can significantly improve performance, both in theory and in real applications.

The different works are important contributions that allow for improving the numerical behavior of the conjugate gradient methods. Of all the results, one is published and are accepted or in revision [31, 32, 33, 34, 35, 43].

Several interesting topics remain for further research:

- **Extension to constrained problems:** Adapt the proposed methods to inequality- and box-constrained optimization, for instance by integrating projection, barrier, or penalty techniques.

- **Application to broader domains:** Extend the applicability of the algorithms to areas such as machine learning, deep learning optimization, control theory, portfolio

of minimizing risks, and inverse problems in signal and image processing.

- **Stochastic and large-scale data optimization:** Generalize the proposed methods to stochastic settings suitable for large-scale optimization problems.

Bibliography

[1] N. Andrei: An unconstrained optimization test functions collection. Adv. Model. Optim., 10 (2008), 147–161.

[2] N. Andrei: Nonlinear conjugate gradient methods for unconstrained optimization. Springer, Springer Optimization and its Applications, 158 (2020), 455–466.

[3] N. Andrei: Modern numerical nonlinear optimization. Springer, Springer Optimization and its Applications, 195 (2022), 81–107.

[4] A. Alhawarat, Z. Salleh, H. Alolaiyan, H. El Hor, S. Ismail: A three-term conjugate gradient descent method with some applications. J. Inequal. Appl., 2024 (2024), 73.

[5] S. N. Alsuliman, N. F. Ibrahim, N. A. H. Aizam: A modified Wei–Yao–Liu conjugate gradient method for unconstrained optimization and motion control of robotic motion manipulators. Journal of Advanced Research in Applied Sciences and Engineering Technology, 34 (2024), 314–327.

[6] S. Ben Hanachi, B. Sellami, M. Belloufi: New iterative conjugate gradient method for nonlinear unconstrained optimization. RAIRO-Oper. Res., 56 (2022), 2315–2327.

[7] S. Boyd, L. Vandenberghe: Convex Optimization. Cambridge University Press, 2004.

[8] A. Cauchy: Méthode générale pour la résolution des systèmes d'équations simultanées. Compte Rendu de l'Académie des Sciences, Paris, 1847.

[9] J.F. Cai, R. Chan, B. Morini: Minimization of an edge-preserving regularization functional by conjugate gradient type methods. In: Image processing based on partial differential equations, Springer, Berlin, Heidelberg (2007), 109–122.

[10] E. Candes, J. Romberg, T. Tao: Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. IEEE Transactions on Information Theory, 52(2) (2006), 489–509.

[11] R.H. Chan, C.W. Ho, M. Nikolova: Salt-and-pepper noise removal by median-type noise detectors and detail-preserving regularization. IEEE Trans. Image Process., 14(10) (2005), 1479–1485.

[12] W. Y. Cheng: A PRP type method for systems of monotone equations. Math. Model. Comput., 50(1-2) (2009), 15–20.

[13] Y.H. Dai, Y. Yuan: A nonlinear conjugate gradient method with a strong global convergence property. SIAM J. Optim., 10(1) (1999), 177–182.

[14] Y.H. Dai, Y. Yuan: An efficient hybrid conjugate gradient method for unconstrained optimization. Ann. Oper. Res., 103 (2001), 33–47.

[15] Y.-H. Dai, L.-Z. Liao: New conjugacy conditions and related nonlinear conjugate gradient methods. Appl. Math. Optim., 43 (2001), 87–101.

[16] S. Delladji, M. Belloufi, B. Sellami: New hybrid conjugate gradient method as a convex combination of FR and BA methods. Journal of Information and Optimization Sciences, 42(3) (2021), 591–602.

[17] S. Djordjevic: New Hybrid Conjugate Gradient Method as a Convex Combination of LS and CD methods. Filomat, 31(6) (2017), 1813–1825.

[18] S. Djordjevic: New hybrid conjugate gradient method as a convex combination of HS and FR methods. Journal of Applied Mathematics and Computation, 2(9) (2018), 366–378.

[19] S. Djordjevic: New hybrid conjugate gradient method as a convex combination of LS and FR methods. Acta Mathematica Scientia, 39B(1) (2019), 214–228.

[20] E.D. Dolan, J.J. Moré: Benchmarking optimization software with performance profiles. Math. Program., 91 (2002), 201–213.

[21] Z. Dai, F. Wen: Another improved Wei–Yao–Liu nonlinear conjugate gradient method with sufficient descent property. Applied Mathematics and Computation, 218 (2012), 7421–7430.

[22] R. Fletcher: Practical methods of optimization. Unconstrained Optimization. vol. 1. Wiley, New York (1987).

[23] R. Fletcher, C. M. Reeves: Function minimization by conjugate gradients. Comput. J., 7(2) (1964), 149–154.

[24] M. A. T. Figueiredo, R. D. Nowak and S. J. Wright: Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. IEEE J. Sel. Topics Signal Process., 1(4) (2007), 586–597.

[25] J. C. Gilbert: Eléments d'optimisation différentiable : théorie et algorithmes. Notes de cours, École Nationale Supérieure de Techniques Avancées, Paris, 2007.

[26] N.I.M. Gould, D. Orban, Toint PL.: CUTEr and SifDec: A constrained and unconstrained testing environment, revisited. ACM Trans. Math. Softw. (TOMS), 29(4) (2003), 373–394.

[27] I. Guefassa, Y. Chaib, T. Bechouat: A Hybrid Conjugate Gradient Method Between MLS and FR in Nonparametric Statistics. Communications in Combinatorics and Optimization. (2023), 1–17.

[28] A. Hallal, M. Belloufi, B. Sellami: An efficient new hybrid CG-method as convex combination of DY and CD and HS algorithms. RAIRO Oper. Res., 56 (2022), 4047–4056.

[29] W.W. Hager, H. Zhang: A survey of nonlinear conjugate gradient methods. Pacific Journal of Optimization, 2 (2006), 35–58.

[30] M. R. Hestenes, E. Steifel: Methods of conjugate gradients for solving linear systems. J. Res. Natl. Bur. Stand., 49(6) (1952), 409–436.

[31] Y. E. Hemici, S. Khelladi, D. Benterki: New hybrid conjugate gradient method for nonlinear optimization with application to image restoration problems. Kybernetika, 60(4) (2024), 535–552.

[32] Y. E. Hemici, S. Khelladi, D. Benterki: An Efficient Hybrid Conjugate Gradient Method for Large-Scale Nonlinear Equations with Applications in Compressive Sensing. Journal of Nonlinear Modeling and Analysis, 8(3), (2026).

[33] Y. E. Hemici, S. Khelladi, D. Benterki: New conjugate gradient method of Wei-Yao-Liu type applied to image restoration and compressive sensing problems. Algerian Journal of Science, 2(2), (2026).

[34] Y. E. Hemici, S. Khelladi, D. Benterki: New hybrid RMIL-FR conjugate gradient method for unconstrained nonlinear optimization. Filomat, (2025), In revision.

[35] Y. E. Hemici, S. Khelladi and D. Benterki, An efficient hybrid three-term conjugate gradient method with practical applications, Iranian Journal of Numerical Analysis and Optimization, 16(1) (2026), 264–282.

[36] M. D. Ilić, F. D. Galiana and L. Fink: Power system restructuring: Engineering and economics. Springer, (1999).

[37] A. H. Ibrahim, P. Kumam, A. B. Abubakar, W. Jirakitpuwapat and J. Abubakar: A hybrid conjugate gradient algorithm for constrained monotone equations with application in compressive sensing. Heliyon, 6(3) (2020), e03466.

[38] F.N. Jardow, G.M. Al-Naemi: A new hybrid conjugate gradient algorithm for unconstrained optimization with inexact line search. Indonesian Journal of Electrical Engineering and Computer Science. 20(2) (2020), 939–947.

[39] X. Jiang, W. Liao, J. Yin, J. Jian: A new family of hybrid three-term conjugate gradient methods with applications in image restoration. J. Appl. Math. Comput., 91(1) (2022), 161–191.

[40] J. Jia, A. Prater-Bennette, L. Shen and E. E Tripp: Sparse recovery: The square of l_1/l_2 norms. J Sci Comput, 102(24) (2024)

[41] S. Khelladi: Optimisation sans contrainte, cours et exercices. Université Ferhat Abbas Sétif1, 2020.

[42] S. Khelladi, D. Benterki: Efficient one-parameter family of conjugate gradient methods. Journal of Information and Optimization Sciences, 45(3) (2024), 697–715.

[43] S. Khelladi, Y. E. Hemici: A New Hybrid Conjugate Gradient Method for Unconstrained Optimization Based on RMIL, LS, and CD Methods. Nonlinear Dynamic System Theory, 26(2) (2026), 184–194.

[44] Y. Liu, C. Storey: Efficient generalized conjugate gradient algorithm. Part 1: Theory. J. Optim. Theory Appl., 69(1) (1991), 129–137.

[45] J. K. Liu and S. J. Li: A projection method for convex constrained monotone nonlinear equations with applications. Comput. Math. Appl., 70(10) (2015), 2442–2453.

[46] J. Liu and Y. Feng: A derivative-free iterative method for nonlinear monotone equations with convex constraints. Numer. Algorithms, 82(1) (2019), 245–262.

[47] J. K. Liu, Z. L. Lu and J. L. Xu: An efficient projection-based algorithm without Lipschitz continuity for large-scale nonlinear pseudo-monotone equations. J. Comput. Appl. Math., 403 (2022), 113822.

[48] G. Ma, H. Lin, W. Jin, D. Han: Two modified conjugate gradient methods for unconstrained optimization with applications in image restoration problems. J. Appl. Math. Comput., 68 (2022), 4733–4758.

[49] M. Malik, I. M. Sulaiman, A. B. Abubakar, G. Ardaneswari, A. Sukono: A new family of hybrid three-term conjugate gradient method for unconstrained optimization with application to image restoration and portfolio selection. To be published (2023).

[50] J.J. Moré, B.S. Garbow, K.E. Hillstrom: Testing unconstrained optimization software. ACM Trans. Math. Softw. (TOMS), 7(1) (1981), 17–41.

[51] K. Meintjes and A. P. Morgan: Chemical equilibrium systems as numerical test problems. ACM Trans. Math. Software, 16(2) (1990), 143–151.

[52] P. Mtagulwa, P. Kaelo: A convergent modified HS-DY hybrid conjugate gradient method for unconstrained optimization problems. Journal of Information and Optimization Sciences, 40(1) (2019), 97–113.

[53] G. Ma, J. Jin, J, Jian , J. Yin and D. Han: A modified inertial three-term conjugate gradient projection method for constrained nonlinear equations with applications in compressed sensing. Numer. Algor, 92 (2023), 1621–1653.

[54] J. Nocedal, S. J. Wright: Numerical Optimization. Springer Science Business Media, 2006.

[55] E. Nermeh, M. Addullahi and A. S. Halilu: Modification of a conjugate gradient approach for convex constrained nonlinear monotone equations with applications in signal recovery. Commun. Nonlinear Sci. Numer. Simul., 136 (2024), 108079.

[56] M. L. Ouaoua, S. Khelladi, D. Benterki: New parameter of conjugate gradient method for unconstrained nonlinear optimization. Statistics, Optimization and Information Computing, 13(6) (2025), 2382–2390.

[57] E. Polak, G. Ribiere: Note sur la convergence des méthodes de directions conjuguées. Rev. Française Informatique Recherche Opérationnelle, 16 (1969), 35–43.

[58] B. T. Polyak: The conjugate gradient method in extreme problems. U.S.S.R. Comput. Math. Phys., 9 (1969), 94–112.

[59] M. J. D. Powell: Restart procedures of the conjugate gradient method. Math. Program., 2 (1977), 241–254.

[60] M. Rivaie, M. Mustafa, W.J. Leong, M. Ismail: A new class of nonlinear conjugate gradient coefficients with global convergence properties. Applied Mathematics and Computation, 218(22) (2012), 11323–11332.

[61] M. Rivaie, M. Mustafa, A. Abdelrhaman: A new class of non linear conjugate gradient coefficients with exact and inexact line searches. Applied Mathematics and Computation, 268 (2015), 1152–1163.

[62] B. Sellami, Y. Chaib: A new family of globally convergent conjugate gradient methods. Ann. Oper. Res., 241 (2016), 497–513.

[63] B. Sellami, Y. Chaib: New conjugate gradient method for unconstrained optimization. RAIRO Operations Research, 50 (2016), 1013–1026.

[64] Ibrahim M. Sulaiman, P. Kaelo, Ruzelan Khalid, Mohd Kamal M. Nawawi: A Descent Generalized RMIL Spectral Gradient Algorithm for Optimization Problems. International Journal of Applied Mathematics and Computer Science, 34(2) (2024), 225–233.

[65] W. Sun, Y. Yuan: Optimization Theory and Methods: Nonlinear Programming. Springer Science+Business Media, 2006.

[66] P. Stanimirovic, B. Ivanov, D. Mosic: A survey of gradient methods for solving nonlinear optimization. Electronic Research Archive, 28(4) (2020), 1573–1624.

[67] M. Solodov and B. Svaiter: A new projection method for variational inequality problems. SIAM J. Control Optim., 37(3) (1999), 765–776.

[68] Z. Wei, S. Yao, L. Liu: The convergence properties of some new conjugate gradient methods. Applied Mathematics and Computation, 183 (2006), 1341–1350.

[69] Y. Xiao and H. Zhu: A conjugate gradient method to solve convex constrained monotone equations with applications in compressive sensing. J. Math. Anal. Appl., 405(1) (2013), 310–319.

[70] X. Yang, Z. Luo, X. Dai: A global convergence of LS-CD hybrid conjugate gradient method. Adv. Numer. Anal., 2013 (2013), 5 pages.

[71] J. Yin, J. Jian, X. Jiang, M. Liu and L. Wang: A hybrid three-term conjugate gradient projection method for constrained nonlinear monotone equations with applications. Numer Algor, 88 (2021), 389–418.

[72] Y. Zhang, B. Zheng: Two new Dai-Liao-type conjugate gradient methods for unconstrained optimization problems. Journal of Optimization Theory and Applications, 175 (2017), 502–509.

[73] G. Zoutendijk: Nonlinear programming, computational methods. In: Integer and Nonlinear Programming, J. Abadie (Ed.), North-Holland, Amsterdam (1970), 37–86.

ملخص :
تتناول هذه الأطروحة حل مسائل الأمثلة غير الخطية وغير المقيدة باستخدام طرق التدرج المترافق، حيث نقترح مجموعة طرق جديدة من نوع التدرج المترافق، وقد قمنا في كل حالة بإثبات شرط التناقص بالنسبة لاتجاهات البحث الجديدة الناتجة وبرهنا التقارب الإجمالي للخوارزميات المتحصل عليها باستخدام الشروط القوية لوولف.
كما تم إبراز أهمية النتائج المتحصل عليها من خلال تجارب عددية على مسائل كلاسيكية مختلفة ذات أبعاد كبيرة وأخرى تطبيقية، أثبتت تفوق الطرق المقترحة مقارنة مع طرق أخرى.
كلمات مفتاحية :
الأمثلة غير الخطية غير المقيدة، طريقة التدرج المترافق، اتجاه التناقص، البحث الخطي غير الدقيق، التقارب الإجمالي.

Abstract:
This thesis focuses on solving unconstrained nonlinear optimization problems using conjugate gradient methods. We propose new families of search directions of the conjugate gradient method, based on new conjugacy parameters. For each variant, we establish the descent condition of the new search directions, and we proved the global convergence of the corresponding algorithms, under the strong Wolfe conditions.
The efficiency of the proposed algorithms is confirmed through numerical tests carried out on several large-scale classical test problems as well as several practical applications.
Keywords:
Unconstrained nonlinear optimization, Conjugate gradient method, Descent direction, Inexact line search, Global convergence.

Résumé :
Cette thèse concerne la résolution des problèmes d'optimisation non linéaire sans contraintes, en utilisant les méthodes de gradient conjugué. Nous avons proposé de nouvelles familles de directions de recherche de méthode de gradient conjugué basées sur nouveaux paramètres de conjugaison. Pour chaque variante nous avons démontré la condition de descente des directions obtenues, ainsi que la convergence globale des algorithmes correspondants sous les conditions de Wolfe fortes.
L'efficacité des algorithmes proposés est confirmée à travers des tests numériques établis sur plusieurs problèmes classiques de grandes dimensions ainsi que sur quelques problèmes pratiques.
Mots clés :
Optimisation non linéaire sans contraintes, Méthode du gradient conjugué, Direction de descente, Recherche linéaire inexacte, Convergence globale.