A FAMILY OF CONJUGATE GRADIENT METHODS VIA EIGENVALUE ANALYSIS FOR SOLVING LARGE-SCALE SYSTEM OF NONLINEAR EQUATIONS

KABIRU AHMED

SPS/15/MMT/00035

B.Sc., Mathematics (BUK)

A Dissertation Submitted to the Department of Mathematical Sciences, Bayero University, Kano, in Partial fulfillment of the Requirements for the award of the degree of

MASTERS IN MATHEMATICS.

AUGUST, 2019

DECLARATION

I hereby declare that this work is the product of my research effort, undertaken under the supervision of Mohammad Waziri Yusuf, (Ph.D) and has not been presented anywhere else for the award of a degree or certificate. All sources have been duly acknowledged.

_____

(Signature/Date)

KABIRU AHMED

SPS/15/MMT/00035

CERTIFICATION

This is to certify that the research work for this dissertation and the subsequent write-up were carried out by Kabiru Ahmed (SPS/15/MMT/00035) under my supervision.

_____

(Signature/Date)

Dr. Mohammad Waziri Yusuf

(Supervisor)

_____

(Signature/Date)

Dr. Abbas Ja'afaru Badakaya

(Head of Department)

APPROVAL

This dissertation has been examined and approved for the award of MASTERS IN MATHEMATICS.

Dr. Hamisu Musa ———————————-

(External Examiner) Signature/Date

Dr. Sirajo Lawan Bichi ———————————-

(Internal Examiner) Signature/Date

Dr. Mohammad Waziri Yusuf ———————————-

(Supervisor) Signature/Date

Dr. Abbas Ja'afaru Badakaya ———————————-

(Head of Department) Signature/Date

———————————- ———————————-

(S.P.S Representative) Signature/Date

ACKNOWLEDGEMENTS

Praise be to Allah the creator of man; the most gracious most merciful, for guiding me throughout the exciting period of this research. May His peace and blessings be upon the holy prophet Muhammad (S.A.W).

I am especially grateful to my parents and grandparents for starting this journey and for their unconditional support and guidance. Next in line is my humble supervisor, Mohammed Yusuf Waziri (Ph.D), who was the motivating force and inspiration behind the research. He has been just outstanding, and I will forever be grateful for his immense contributions, guidance and encouragement in the course of this work. I would also like to extend my gratitude to my internal examiner, Dr. Sirajo Lawan Bichi for sparing his time to ensure I came up with an acceptable work. My sincere gratitude goes to Mal. Jamilu Sabiu of North-West University, Kano, for his immense assistance and being there for me each time I needed him. May Allah (SWT) reward him abundantly. I would also like to extend my profound gratitude to all my lecturers in the department of Mathematics for their support and encouragements. I would specifically, like to thank the following people for their encouragements: Prof. Bashir Ali, Dr. Abbas Ja'afaru Badakaya, Dr. Aliyu Ibrahim Kiri, Dr. Isa Baba, Mal. Mansur Zubairu, Mal. Ya'u Balarabe Musa, and Mal. Hassan Muhammad. Finally, I would like to thank my

classmates like, Mal. Basiru Abdullahi, Mal. Muntaka Bashir Jibril, Zulyadaini Dahiru, Bashir Danladi Garba, Adamu Y. Inuwa and my good friend, Muhammad Omuya for all the happy hours we shared during the class work.

DEDICATION

This work is dedicated to late Alhaji Wada Amadu of blessed memory.

Table of contents

Title page **i**

Declaration **ii**

Certification **iii**

Approval **iv**

Acknowledgment **v**

Dedication **vii**

Abstract **xvii**

CHAPTER ONE **1**

1.1 INTRODUCTION 1

1.2 SYSTEM OF NONLINEAR EQUATIONS 1

1.3 STATEMENT OF THE PROBLEM 3

1.4 MOTIVATION 3

1.5 AIM AND OBJECTIVES 3

1.6 SCOPE AND LIMITATIONS 4

1.7 BASIC DEFINITIONS 4

CHAPTER TWO 9

2.1 INTRODUCTION 9

2.2 CLASSICAL NEWTON'S METHOD AND ITS VARIANTS . . 9

2.3 CONJUGATE GRADIENT METHOD 11

CHAPTER THREE 18

3.1 INTRODUCTION 18

3.2 ENHANCED DAI-LIAO CONJUGATE GRADIENT METHOD 18

3.3 A DAI-LIAO CONJUGATE GRADIENT METHOD 27

3.4 MODIFIED HAGER-ZHANG CONJUGATE GRADIENT METHODS 33

3.4.1 Improved Hager-Zhang Conjugate Gradient Method . . . 33

3.4.2 Enhanced Hager-Zhang Conjugate Gradient Method . . . 38

3.5 CONVERGENCE ANALYSIS 43

3.5.1 Convergence Result of Enhanced Dai-Liao CG Method . . 44

3.5.2 Convergence Result of A Dai-Liao CG Method 49

3.5.3 Convergence Result of Improved Hager-Zhang CG method 52

3.5.4 Convergence Result of Enhanced Hager-Zhang CG method 54

CHAPTER FOUR 57

4.1 INTRODUCTION 57

4.2 INITIAL POINTS AND TEST FUNCTIONS 60

4.3 COMPUTATIONAL EXPERIMENTS 64

4.4 DISCUSSION OF RESULTS 87

CHAPTER FIVE 89

5.1 INTRODUCTION 89

5.2 CHANDRASEKHAR H-EQUATION 90

CHAPTER SIX 97

6.1 INTRODUCTION 97

6.2 SUMMARY 97

6.3 CONCLUSION 98

6.4 FUTURE RESEARCH 99

REFERENCES **99**

APPENDIX A **111**

APPENDIX B **112**

List of Tables

4.1 Initial points used for the test problems 60

4.2 Test results of EDLCG, FCGM, TTPRP, and MHSCG methods for problems 1-4 65

4.3 Test results of EDLCG, FCGM, TTPRP, and MHSCG methods for problems 5-8 66

4.4 Test results of EDLCG, FCGM, TTPRP, and MHSCG methods for problems 9-12 67

4.5 Test results of EDLCG, FCGM, TTPRP, and MHSCG methods for problems 13-15 68

4.6 Test results of ADLCG, FCGM, TTPRP, and MHSCG methods for problems 1-4 69

4.7 Test results of ADLCG, FCGM, TTPRP, and MHSCG methods for problems 5-8 70

4.8 Test results of ADLCG, FCGM, TTPRP, and MHSCG methods for problems 9-12 71

4.9 Test results of ADLCG, FCGM, TTPRP, and MHSCG methods for problems 13-15 72

4.10 Test results of IHZCG, FCGM, TTPRP, and MHSCG methods for problems 1-4 73
4.11 Test results of IHZCG, FCGM, TTPRP, and MHSCG methods for problems 5-8 74
4.12 Test results of IHZCG, FCGM, TTPRP, and MHSCG methods for problems 9-12 75
4.13 Test results of IHZCG, FCGM, TTPRP, and MHSCG methods for problems 13-15 76
4.14 Test results of EHZCG, FCGM, TTPRP, and MHSCG methods for problems 1-4 77
4.15 Test results of EHZCG, FCGM, TTPRP, and MHSCG methods for problems 5-8 78
4.16 Test results of EHZCG, FCGM, TTPRP, and MHSCG methods for problems 9-12 79
4.17 Test results of EHZCG, FCGM, TTPRP, and MHSCG methods for problems 13-15 80
4.18 Number of problems (with percentage) solved by EDLCG, FCGM, TTPRP, and MHSCG methods with least iterations and CPU time 81
4.19 Number of problems (with percentage) solved by ADLCG, FCGM, TTPRP, and MHSCG methods with least iterations and CPU time 81
4.20 Number of problems (with percentage) solved by IHZCG, FCGM, TTPRP, and MHSCG methods with least iterations and CPU time 82
4.21 Number of problems (with percentage) solved by EHZCG, FCGM, TTPRP, and MHSCG methods with least iterations and CPU time 82

5.22 Test results of Chandrasekhar H-equation with respect to number of iterations/CPU time 92
5.23 Test results of Chandrasekhar H-equation with respect to number of iterations/CPU time 93

List of Figures

4.1 Performance profile of EDLCG, FCGM, TTPRP, and MHSCG methods (for number of iterations) 83

4.2 Performance profile of EDLCG, FCGM, TTPRP, and MHSCG methods (for CPU time) 83

4.3 Performance profile of ADLCG, FCGM, TTPRP, and MHSCG methods (for number of iterations) 84

4.4 Performance profile of ADLCG, FCGM, TTPRP, and MHSCG methods (for CPU time) 84

4.5 Performance profile of IHZCG, FCGM, TTPRP, and MHSCG methods (for number of iterations) 85

4.6 Performance profile of IHZCG, FCGM, TTPRP, and MHSCG methods (for CPU time) 85

4.7 Performance profile of EHZCG, FCGM, TTPRP, and MHSCG methods (for number of iterations) 86

4.8 Performance profile of EHZCG, FCGM, TTPRP, and MHSCG methods (for CPU time) 86

5.9 Performance profile of EDLCG, ADLCG, FCGM, TTPRP, and MHSCG methods (for number of iterations) 94

5.10 Performance profile of EDLCG, ADLCG, FCGM, TTPRP, and MHSCG methods (for CPU time) 94

5.11 Performance profile of IHZCG, EHZCG, FCGM, TTPRP, and MHSCG methods (for number of iterations) 95

5.12 Performance profile of IHZCG, EHZCG, FCGM, TTPRP, and MHSCG methods (for CPU time) 95

ABSTRACT

In this research, we present a family of conjugate gradient (CG) methods for solving large-scale system of nonlinear equations that are both matrix and derivative-free. By carrying out eigenvalue analysis in the classical Dai-Liao (2001) and Hager-Zhang (2006) approach, we developed four enhanced CG update parameters β_k and applied them to develop our proposed methods. The first method incorporates an extension of the standard secant equation and the modified secant equations proposed by Li and Fukushima (2001), Zhang et al. (1999), and Wei et al (2006), while the second method employs an extension of the modified secant equations proposed by Zhang et al. (1999) and Wei et al. (2006). The third method is based on eigenvalue study in the Hager-Zhang (2006) approach, while the fourth method employs the modified secant equation proposed by Zhang et al. (1999) and Zhang and Xu (2001). Our anticipation is to suggest good CG parameters that will lead to a solution with less computational cost. Global convergence, numerical results, and comparison with some methods in the literature were established to show the efficiency of the methods. Preliminary results shows that the proposed methods are promising. To further demonstrate the efficiency of the proposed methods, they are applied to solve a discretized version of Chandrasekhar's

integral equation, which plays important role in radiactive transfer and transport theory.

CHAPTER ONE

INTRODUCTION

1.1 INTRODUCTION

In this chapter, some popular iterative methods for solving nonlinear system of equations, which includes the classical Newton, Quasi-Newton and conjugate gradient methods, are discussed. Also included in the chapter are statement of the problem, motivation of the study, aim and objectives, scope and limitations, and methodology of the research. Basic definitions of some terms are also included in the chapter.

1.2 SYSTEM OF NONLINEAR EQUATIONS

A typical system of nonlinear equations has the general form

$$F(x) = 0, \tag{1.2.1}$$

where $F : R^n \to R^n$ is a nonlinear mapping assumed to be continuously differentiable in a neighborhood of R^n. The function in (1.2.1) can also be represented as

$$F(x) = (f_1(x), f_2(x), ..., f_n(x))^T, \tag{1.2.2}$$

where each f_i $(i = 1, 2, ..., n)$ maps a vector $x = (x_1, x_2, ..., x_n)^T$ of the n-dimensional space R^n into the real line R. In addition, F is assumed to satisfy the following:

1. There exists $x^* \in R^n$ such that $F(x^*) = 0$.

2. The Jacobian F' is Lipschitz continuous near x^*.

3. $F'(x^*)$ is nonsingular.

Problems involving system of nonlinear equations often arise in sciences and engineering fields and for that, lots of computational methods are being developed by researchers to handle them. Newton and quasi-Newton strategies are the most famous methods for solving (1.2.1), see (Al-Baali, Spedicato & Maggioni, 2014 ; Barzilai & Borwein, 1998 ; Brown & Saad, 1994 ; Li & Fukushima, 2000 ; Martinez, 1990 ; Martinez, 2000 ; Natasa & Zorana, 2002 ; Shin, Dirvashi & Kim, 2010 ; Waziri, Hassan & Monsi, 2010 ; Waziri, Leong & Hassan, 2011) for details. Generally, these methods generate iterative sequence of points via

$$x_{k+1} = x_k - (F'(x_k))^{-1} F(x_k), \tag{1.2.3}$$

where $k = 0, 1, 2 \ldots$ and $F'(x_k)$ is the Jacobian matrix at x_k or its approximation in the case of quasi-Newton schemes.

Newton and quasi-Newton methods are attractive for their easy implementation and rapid convergence properties, (Waziri & Sabiu, 2015). However, the huge requirement for matrix storage at each iteration makes them not suitable for handling large-scale nonlinear systems. The nonlinear conjugate gradient (CG) methods are the ideal strategies for solving large-scale nonlinear systems because of their simple implementation, low memory requirement and global convergence properties (Babaie-Kafaki & Ghanbari, 2013). In this research, we present four conjugate gradient methods for solving large-scale system of nonlinear equations that are matrix and derivative-free. We achieve this by modifying the classical Dai-Liao (2001) method for unconstrained optimization and its adaptive version, the one-parameter Hager-Zhang (2006) method.

1.3 STATEMENT OF THE PROBLEM

The study of conjugate gradient method for system of nonlinear equations is rare as most of the methods developed are for unconstrained optimization problems. Also, not all the methods generate descent directions. In this research, we present a family of CG methods for system of nonlinear equations that generate descent search directions and are matrix and derivative-free.

1.4 MOTIVATION

Our motivation for this work originated from the following:

- The study of conjugate gradient method for system of nonlinear equations is rare as most of the methods developed are for unconstrained optimization.
- The DL method does not necessarily generate descent directions and it depends so much on the nonnegative parameter t for which there's no optimal value (Andrei, 2011 ; Babaie-Kafaki & Ghanbari, 2013).

The aforementioned and the remarkable work of (Babaie-Kafaki & Ghanbari, 2013 ; Babaie-Kafaki & Ghanbari, 2014 ; Babaie-Kafaki & Ghanbari, 2015 ; Babaie-Kafaki & Ghanbari, 2016 ; Babaie-Kafaki & Ghanbari, 2014) and (Arazm, Babaie-Kafaki & Ghanbari, 2017) in this area, inspired us to carry out this research.

1.5 AIM AND OBJECTIVES

The aim of this research work is to develop new methods for solving large-scale system of nonlinear equations, while the objectives are;

i. To derive a family of conjugate gradient methods **(AFCGM)** with sufficient descent directions via eigenvalue analysis for nonlinear systems.

ii. To present convergence results of the proposed methods.

iii. To design matlab codes for implementation of the algorithms developed.

iv To test the proposed methods on some tested problems and compare the numerical results with some other existing methods in the literature.

1.6 SCOPE AND LIMITATIONS

This research focuses only on CG methods for solving large scale system of nonlinear equation and their modifications.

1.7 BASIC DEFINITIONS

In this section we give some basic definitions that are important for better understanding of the concept of this write-up.

Definition 1.7.1 (Jacobian Matrix) *(Kelly, 1995) Let f_i be i^{th} component of F : $\mathbb{R}^n \to \mathbb{R}^n$. If the components of F are differentiable at $x \in \mathbb{R}^n$, then the Jacobian matrix $F'(x)$ can be define by*

$$F'(x)_{ij} = \frac{\partial f_i}{\partial x_j}(x). \tag{1.7.1}$$

Definition 1.7.2 (Norm) *(Sun & Yuan, 2006) A mapping $\|.\| : \mathbb{R}^n \to \mathbb{R}$ is called a Norm if and only if it satisfies the following:*

(N1): $\|x\| \geq 0$, $\forall x \in \mathbb{R}^n$; $\quad \|x\| = 0$ *if and only if* $x = 0$;

(N2): $\|\alpha x\| = |\alpha|\|x\|$, $\forall \alpha \in \mathbb{R}$ $\forall x \in \mathbb{R}^n$;

(N3): $\|x+y\| \leq \|x\| + \|y\|$, $\forall x, y \in \mathbb{R}^n$.

Definition 1.7.3 (l_1 Norm, Euclidean Norm and l_∞ Norm) *(Sun & Yuan, 2006) For a vector $x \in \mathbb{R}^n$, the well-known vector norms are defined as follows:*

- $\|x\|_1 = \sum_{i=1}^{n} (|x_i|)$ *(l_1-norm).*
- $\|x\|_2 = \left(\sum_{i=1}^{n} \left(x_i^2 \right) \right)^{\frac{1}{2}} = (x^T x)^{\frac{1}{2}}$. *The Euclidean norm (or l_2 norm).*
- $\|x\|_\infty = \max_{1 \leq i \leq n} (|x_i|)$. *The l_∞ norm.*

Definition 1.7.4 (Symmetric positive definite Matrix) *(Nocedal & Wright, 2006) Let $A \in \mathbb{R}^{n \times n}$, we say that A is symmetric if it is equal to its transpose. i.e.,*

$$A = A^T. \tag{1.7.2}$$

A symmetric matrix A is positive definite if there exists a positive constant α such that

$$x^T A x \geq \alpha \|x\|^2, \forall x \in \mathbb{R}^n. \tag{1.7.3}$$

Definition 1.7.5 (Frobenius Norm) *(Sun & Yuan, 2006) Let $A \in \mathbb{R}^{m \times n}$. The Frobenius norm of A is given by*

$$\|A\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2}. \tag{1.7.4}$$

Definition 1.7.6 (Eigenvalues and eigenvectors) *(Nocedal & Wright, 2006) A scalar value λ is an eigenvalue of the $n \times n$ matrix A if there is a nonzero vector x such that*

$$Ax = \lambda x. \tag{1.7.5}$$

The vector x is called the eigenvector of A.

Definition 1.7.7 (Trace and Determinant) *(Nocedal & Wright, 2006) The trace of an $n \times n$ matrix A is defined by*

$$trace(A) = \sum_{i=1}^{n} A_{ii}. \tag{1.7.6}$$

If the eigenvalues of A are denoted by $\lambda_1, \lambda_2, ..., \lambda_n$, then

$$trace(A) = \sum_{i=1}^{n} \lambda_i. \tag{1.7.7}$$

The determinant of an $n \times n$ matrix A is the product of its eigenvalues, that is,

$$\det(A) = \prod_{i=1}^{n} \lambda_i. \tag{1.7.8}$$

If the $\det(A) = 0$, then A is singular.

Definition 1.7.8 (Convex Set) *(Sun & Yuan, 2006) Let the set $S \subset \mathbb{R}^n$. If for any $x_1, x_2 \in S$, we have*

$$\alpha x_1 + (1-\alpha)x_2 \in S, \quad \forall \alpha \in [0,1], \tag{1.7.9}$$

then S is said to be a convex set.

Definition 1.7.9 (Convex Function) *(Sun & Yuan, 2006) Let $S \subset \mathbb{R}^n$ be a nonempty set.*
Let $f : S \subseteq \mathbb{R}^n \longrightarrow \mathbb{R}$. If for any $x_1, x_2 \in S$ and $\forall \alpha \in (0,1)$, we have

$$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha) f(x_2), \tag{1.7.10}$$

then the function f is said to be convex on S. If the above inequality is true as a strict inequality for all $x_1 \neq x_2$, i.e.,

$$f(\alpha x_1 + (1-\alpha)x_2) < \alpha f(x_1) + (1-\alpha) f(x_2), \tag{1.7.11}$$

then f is said to be strictly convex function on S. Also, if there is a constant $c > 0$ such that for any $x_1, x_2 \in S$,

$$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha) f(x_2) - \frac{1}{2} c\alpha(1-\alpha)\|x_1 - x_2\|^2, \quad (1.7.12)$$

then f is said to be uniformly (or strongly) convex function on S.

Definition 1.7.10 (Conjugate Vectors) *(Nocedal & Wright, 2006) Let G be an $n \times n$ symmetric and positive definite matrix, $d_1, d_2, ..., d_m \in \mathbb{R}^n$ be non-zero vectors, $m \geq n$. If*

$$d_i^T G d_j = 0, \quad \forall i \neq j, \quad (1.7.13)$$

the vectors $d_1, d_2, ..., d_m$ are called G conjugate or simply conjugate.

Definition 1.7.11 (Monotone Function) *(Sun & Yuan, 2006) A function $F : D \subset \mathbb{R}^n \Rightarrow \mathbb{R}^n$ is monotone on $D_0 \subset D$ if*

$$\langle F(x) - F(y), x - y \rangle \geq 0, \quad \forall x, y \in D_0. \quad (1.7.14)$$

Definition 1.7.12 (Limits and Continuity) *(Fletcher, 1987) Let $F : D \subset \mathbb{R}^n \to \mathbb{R}^n$. For some point $x_0 \in \overline{D}$ (closure of D), we write*

$$\lim_{x \to x_0} F(x) = F_0, \quad (1.7.15)$$

if for all $\varepsilon > 0$, there is $\delta_{(\varepsilon)} > 0$ such that

$$\|x - x_0\| < \delta \quad \text{and} \quad x \in D \Rightarrow \|F(x) - F_0\| < \varepsilon.$$

We say F is continuous at x_0 if $x_0 \in D$ and the equation (1.7.15) *holds with $F_0 = F(x_0)$. F is continuous on its domain D if F is continuous for all $x_0 \in D$.*

Definition 1.7.13 (Lipschitz Continuity) *(Kelly, 1995) Let $\Omega \subset \mathbb{R}^n$ and let $G : \Omega \to \mathbb{R}^n$. G is Lipschitz continuous on Ω with Lipschitz constant γ if*

$$\|G(x) - G(y)\| \leq \gamma \|x - y\|, \forall x, y \in \Omega. \quad (1.7.16)$$

Definition 1.7.14 (Convergence) *(Kelly, 1995) A sequence $\{x_k\}$ of n-vectors converges to x^* if*

$$\lim_{k\to\infty} \|x_k - x^*\| = 0. \tag{1.7.17}$$

Definition 1.7.15 (Linear Convergence) *(Kelly, 1995) Let $\{x_k\} \subset \mathbb{R}^n$ and $x^* \in \mathbb{R}^n$. Then $x_k \to x^*$* linearly *if*
i. $x_k \to x^*$
ii. and there exist α in $(0,1)$ such that

$$\|x_{k+1} - x^*\| \leq \alpha \|x_k - x^*\|. \tag{1.7.18}$$

Definition 1.7.16 (q-quadratically Convergence) *(Kelly, 1995) Let $\{x_k\} \subset \mathbb{R}^n$ and $x^* \in \mathbb{R}^n$. Then $x_k \to x^*$* q-quadratically *if $x_k \to x^*$ and there exist $\alpha > 0$ such that*

$$\|x_{k+1} - x^*\| \leq \alpha \|x_k - x^*\|^2. \tag{1.7.19}$$

Definition 1.7.17 (q-superlinearly Convergence) *(Kelly, 1995) Let $\{x_k\} \subset \mathbb{R}^n$ and $x^* \in \mathbb{R}^n$. Then*
$x_k \to x^*$ q-superlinearly *if*

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = 0. \tag{1.7.20}$$

CHAPTER TWO

LITERATURE REVIEW

2.1 INTRODUCTION

This chapter presents a review of the classical conjugate gradient (CG) methods as well as other schemes for solving systems of nonlinear equations.

2.2 CLASSICAL NEWTON'S METHOD AND ITS VARIANTS

The classical Newton's method is the most popular iterative method employed by many researchers for solving (1.2.1) by employing the iterative scheme in (1.2.3). From an initial guess, $x_0 \in \mathbb{R}^n$, the algorithm is as follows:

- Step1: compute $F(x_k)$ and check if $\|F(x_k)\| \leq \varepsilon$. if yes stop, else go to Step2
- Step2: compute $(F'(x_k))^{-1}$
- Step3: compute $x_{k+1} = x_k - (F'(x_k))^{-1}F(x_k)$
- Step4: Set $k = k+1$ and go to Step1 and repeat the process until convergence is attained.

Newton's method is easy to implement, especially for small dimension problems. Furthermore, provided the Jacobian is nonsingular and the initial starting point is

in the neighborhood of the solution, convergence of the method is guaranteed and its rate is quadratic (Waziri, Leong & Hassan, 2010), i.e

$$\|x_{k+1} - x^*\| \leq h\|x_k - x^*\|^2, \quad h > 0. \tag{2.2.1}$$

However, Newton's method is unattractive for large-scale nonlinear problems because it requires the computation and storage of the Jacobian matrix at each iteration, which is very costly most times. Moreover, computation of the Jacobian of some functions may be difficult or, in some cases, not available, which renders the method unsuitable to implement.

The quasi-Newton methods are a class of iterative methods closely related to the classical Newton's method, which are developed to remedy some of the shortcomings listed above. In the quasi-Newton scheme, the Jacobian or its inverse are avoided by computing an approximation of any of the two, which is updated using the following scheme:
For $k = 0, 1, 2, \ldots$

$$B_k s_k = -F(x_k), \tag{2.2.2}$$

where B_k is an $n \times n$ matrix, which approximates the Jacobian at x_k and s_k is the quasi-Newton search direction given by

$$s_k = -B_k^{-1} F(x_k). \tag{2.2.3}$$

The new iterate of the scheme is given by

$$x_{k+1} = x_k + s_k. \tag{2.2.4}$$

The most prominent quasi-Newton scheme is the BFGS method, named for the researchers that discovered it Broyden (1962), Fletcher (1970), Goldfarb (1970), and Shanno (1970). The update for the scheme is given by

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}, \tag{2.2.5}$$

where, $s_k = x_{k+1} - x_k$ and $y_k = F(x_{k+1}) - F(x_k)$. The method is computationally cheap compared to Newton's method and also has superlinear rate of convergence on practical problems. Other variants of the quasi-Newton's method includes Davidon, Fletcher and Powell (DFP), Symmetric Rank-One (SR1) and the Powell-Symmetric-Broyden (PSB) updates (Sun & Wright, 2006). However, these methods are for unconstrained optimization problems. The most successful quasi-Newton method for solving systems of nonlinear equations is the Broyden scheme with update parameter given by

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)s_k}{s_k^T s_k}. \tag{2.2.6}$$

The major advantage of Broyden's scheme is it has less computations, especially since the inverse of the approximation matrix, B_k^{-1} can be computed directly from previous iteration, B_k^{-1} reduces the number of computations needed for Broyden's method compared to Newton's scheme. However, the inability to converge quadratically, which means more iterations may be required in order to reach a solution, is one of the shortcomings of the method.

2.3 CONJUGATE GRADIENT METHOD

Conjugate gradient methods form an important class of algorithms used in solving large-scale unconstrained optimization problems. They represent an ideal choice for mathematicians and engineers engaged in large-scale problems because of their low memory requirement and strong global convergence properties (Babaie-Kafaki & Ghanbari, 2013). These methods began with the remarkable research of Magnus Hestenes (1906-1991) at the institute for Numerical Analysis and the independent work of Eduard Stiefel (1909-1978) at the Technische Hochschule Zurich. The two researchers presented a paper (Hestenes & Stiefel, 1952), where an alternative algorithm was given for solving a system of linear algebraic equations. Reid (1971) was the first to use the scheme as an iterative method for solving unconstrained optimization problems. Fletcher and Reeves (1964) introduced the first nonlinear conjugate gradient method for solving large-scale nonlinear optimization problems. Consequently, other variants of the method have

been developed by many researchers and are widely used in practice. Generally, the nonlinear conjugate gradient method is used to solve large-scale problems in the following form;

$$min\{f(x)|x \in \mathbf{R}^n\}, \tag{2.3.1}$$

where $f : \mathbf{R}^n \longrightarrow \mathbf{R}$ is a continuously differentiable function that is bounded from below and its gradient is available. The method generates a sequence of iterates x_k from an initial point $x_0 \in \mathbf{R}^n$ using the iterative formula

$$x_{k+1} = x_k + s_k, \quad s_k = \alpha_k d_k, \quad k = 0, 1, ... \tag{2.3.2}$$

where x_k is the current iterate, α_k is a step length computed via suitable line search procedure, and d_k is the CG search direction defined by

$$d_k = \begin{cases} -F_k, & if \quad k = 0, \\ -F_k + \beta_k d_{k-1}, & if \quad k \geq 1, \end{cases} \tag{2.3.3}$$

where β_k is a scalar known as the CG update parameter, and $F_k = \nabla f(x_k)$. It is also worth noting that a crucial element in any CG algorithm is the formula definition of the update parameter β_k (Andrei, 2011), which is why different CG algorithms corresponding to different choices of β_k in (2.3.3) have been proposed.

In a survey of nonlinear conjugate gradient methods, Hager and Zhang (2006) gave an extensive description of some of the essential update parameters, while Andrei (2008A), provided a list of 40 nonlinear conjugate gradient algorithms for unconstrained optimization. It can be observed that for the exception of the methods proposed by Daniel (1967), and Hager and Zhang (2005), the CG methods can be divided into two groups. The first group consists of CG methods with update parameter β_k having numerator $\|F_k\|^2$, which includes the ones proposed by Fletcher and Reeves (FR) (1964), Fletcher (conjugate descent (CD)) (1987), and Dai and Yuan (DY) (1999), with the following CG parameters:

$$\beta_k^{FR} = \frac{||F_k||^2}{||F_{k-1}||^2}, \quad \beta_k^{CD} = \frac{||F_k||^2}{-d_{k-1}^T F_{k-1}}, \quad \beta_k^{DY} = \frac{||F_k||^2}{d_{k-1}^T y_{k-1}} \tag{2.3.4}$$

The second group falls under CG methods with update parameter β_k having numerator $F_k^T y_{k-1}$, which includes the ones proposed by Hestenes and Stiefel (HS) (1952), Polak, Ribiére and Polyak (PRP) (1969), and Liu and Storey (LS) (1991), with the following CG parameters:

$$\beta_k^{HS} = \frac{F_k^T y_{k-1}}{d_{k-1}^T y_{k-1}}, \quad \beta_k^{PRP} = \frac{F_k^T y_{k-1}}{||F_{k-1}||^2}, \quad \beta_k^{LS} = \frac{F_k^T y_{k-1}}{-d_{k-1}^T F_{k-1}} \tag{2.3.5}$$

where, $||.||$ denote the Euclidean norm of vectors and $y_{k-1} = F_k - F_{k-1}$. Nazareth (1998), and Nocedal (1991), provided some reviews on nonlinear conjugate gradient methods. We state the following assumptions that are needed in the convergence of the CG methods and their properties:

- Assumption 1: The level set Ω defined by

$$\Omega = \{x \in R^n : f(x) \leq f(x_0)\}, \tag{2.3.6}$$

  is bounded, namely, there exists a positive constant M such that

$$||f(x)|| \leq M, \quad \forall x \in \Omega. \tag{2.3.7}$$

- Assumption 2: In some neighborhood N of Ω, f is continuously differentiable and its gradient is Lipschitz continuous, i.e., there exists a positive constant L such that

$$||\nabla f(x) - \nabla f(y)|| \leq L||x - y||, \quad \forall x, y \in N. \tag{2.3.8}$$

The FR method has nice theoretical convergence properties, and since it has the same numerator as the CD and DY methods , their numerical performances are very similar. Also, one theoretical difference between CG methods in the first group and other choices of the update parameter is that their global convergence theorem requires only the Lipschitz assumption and not the boundedness assumption (Hager & Zhang, 2006). Still, Powell (1977) noted a major disadvantage of these methods, which makes them perform poorly in practical computations; namely, if a small step is generated from the solution point, subsequent steps may

also be very short. This is known as jamming phenomenon. On the other hand, the methods in the second group can automatically recover once a small step is generated, and so they perform much better than methods in the first group. However, Powell (1984) provided a counter example to show that the PRP and HS methods, without restarts, can circle infinitely without approaching a solution. That shows that the methods have a drawback and are not globally convergent for general functions. Therefore, significant efforts have been directed over the years at finding new formulae for CG methods that are globally convergent, robust, and numerically efficient.

Hybrid CG methods and parametric families have been proposed to address drawbacks of the early methods. A hybrid CG method is developed by combining two different methods so as to exploit attractive features of each set. Hybridizations of CG methods based on the approach of discrete combinations of the CG parameters, include the ones proposed by, Touti-Ahmed and Storey (1990), Hu and Storey (1991), and Gilbert and Nocedal (1992), using PRP and FR methods. Others include the hybridization of HS and DY methods suggested by Dai and Yuan (2001), and the hybridization of LS and CD methods proposed by Touti-Ahmed and Storey (1990). Also, Andrei (2008B) and Andrei (2008C), in an effort to extend the idea of hybridization of CG methods, proposed and analyzed hybrid CG algorithms in which the parameter β_k is computed as a convex combination of parameters in the two groups. In (Dai & Yuan, 2001), a hybrid conjugate gradient method as a convex combination of PRP and DY conjugate gradient algorithms was proposed. The two methods were combined as a hybrid conjugate gradient algorithm by exploiting the good computational properties of the PRP method, and the strong convergence properties of DY method. Also, in (Andrei, 2008B) and (Andrei, 2008C) the parameter β_k is computed as a convex combination of β_k^{HS} (Hestenes-Stiefel) and β_k^{DY} (Dai-Yuan) algorithm, where the hybridization parameter is computed based on the standard secant equation. Similarly, Andrei (2010) developed an accelerated hybrid CG algorithm, where the update parameter β_k is obtained as a convex combination of β_k^{HS} and β_k^{DY}, with the hybridization parameter computed to satisfy the modified secant condition given by Li et al. (2007). Following Andrei's approach (2010), Babaie-Kafaki et al. (2011) proposed two

hybridizations of HS and DY methods in which the hybridization parameters are computed based on the modified secant equations proposed by Yuan (1991) and Li and Fukushima (2001A) and Li and Fukushima (2001B). Also, recently Babaie-Kafaki (2011) proposed another hybridization of HS and DY methods in which the hybridization parameter is computed by applying a conjugacy condition on a quadratic relaxation of a hybrid CG parameter proposed in (Dai & Yuan, 2001).

Also, CG methods can be combined together the same way that quasi-Newton methods have been combined together by introducing parameters, as in the Broyden (1970) family. These includes the one-parameter family of CG methods proposed by Dai and Yuan (1998) and Dai and Yuan (2003), their extension (Dai & Yuan, 1998), the two-parameter family proposed by Nazareth (1998), which employs convex combinations of the numerators and denominators of β_k^{FR} and β_k^{HS}, and the three-parameter family of CG methods proposed by Dai and Yuan (2001), which includes the six standard CG methods, the previous one-parameter and two-parameter families, and many hybrid methods as special cases.

Furthermore, CG methods for solving large-scale unconstrained optimization problems have been extended to solve large-scale nonlinear systems of equation by many researchers. Cheng (2009) proposed a PRP-type method for systems of monotone equations by using a combination of the PRP conjugate gradient method for unconstrained optimization (Polak, 1969) and the hyperplane projection method (Solodov & Svaiter, 1999). Yu (2010) and Yu (2011) extended the PRP method for unconstrained optimization (Polak, 1969) to solve large-scale nonlinear systems with monotone line search strategies, which are modifications of the Grippo et al. (1986) and Li-Fukushima (2000B) schemes. As a further research of the Perry's conjugate gradient method for unconstrained optimization, Dai et al (2015) combined the modified Perry conjugate gradient method for unconstrained optimization problems (Livieres & Pintelas, 2012) and the hyperplane projection technique of Solodov and Svaiter (1999) to propose a derivative-free method for solving large-scale nonlinear monotone equations.

Also, some of the CG methods for unconstrained optimization are not globally convergent, so efforts were made by researchers to develop CG methods that are not only globally convergent but are numerically efficient. These new methods

are based on secant equations. For nonlinear conjugate gradient methods, the conjugacy condition is given by

$$d_k^T y_{k-1} = 0, \tag{2.3.9}$$

where, $y_{k-1} = F_k - F_{k-1}$.

Perry (1978) extended (2.3.9) by exploiting the following secant condition of quasi-Newton schemes:

$$B_k s_{k-1} = y_{k-1}, \tag{2.3.10}$$

where, $s_{k-1} = x_k - x_{k-1}$, and quasi-Newton search direction d_k given by

$$B_k d_k = -F_k, \tag{2.3.11}$$

where B_k is a square matrix, which approximates the Hessian $\nabla^2 f(x)$. By using (2.3.10) and (2.3.11), Perry gave an extension of (2.3.9) as:

$$d_k^T y_{k-1} = -F_k^T s_{k-1}, \tag{2.3.12}$$

and by using (2.3.3), the Perry search direction is given as

$$d_k = \begin{cases} -F_k, & if \quad k = 0, \\ -P_k F_k = -F_k + \beta_k^P d_{k-1}, & if \quad k \geq 1, \end{cases} \tag{2.3.13}$$

where

$$B_k^P = \frac{(y_{k-1} - s_{k-1})^T}{s_{k-1}^T y_{k-1}} F_k, \tag{2.3.14}$$

and

$$P_k = I - \frac{s_{k-1}(y_{k-1} - s_{k-1})^T}{s_{k-1}^T y_{k-1}}. \tag{2.3.15}$$

Following Perry's approach (1978), Dai and Liao (2001) incorporated a nonnegative parameter t to propose the following extension of (2.3.12):

$$d_k^T y_{k-1} = -t F_k^T s_{k-1}. \tag{2.3.16}$$

It is noted that for $t = 0$, (2.3.16) reduces to (2.3.9), and if $t = 1$, we obtain Perry's condition (2.3.12). Consequently, by employing (2.3.3) and (2.3.16), Dai and Liao (2001) proposed the following CG update parameter:

$$B_k^{DL} = \frac{(y_{k-1} - ts_{k-1})^T F_k}{d_{k-1}^T y_{k-1}}, \quad t \geq 0. \tag{2.3.17}$$

Based on the DL conjugacy condition (2.3.16), conjugate gradient methods have been proposed over the years using modified secant equations. For example, Babaie-Kafaki et al. (2010) and Yabe and Takano (2004) proposed CG methods by applying a revised form of the modified secant equation proposed by Zhang and Xu (2001) and Zhang et al. (1999) and the modified secant equation proposed by Li and Fukushima (2000A). Li et al. (2007) applied the modified secant equation proposed by Wei et al. (2006), while Ford et al. (2008) employed the multi-step quasi-Newton conditions proposed by Ford and Moghrabi (1994) and Ford and Moghrabi (1996). CG methods based on modified secant equations have also been studied by Narushima and Yabe (2012) and Arazm et al. (2017).

Numerical results have shown that the DL method (2.3.17) is effective, however, as one of the open problems in nonlinear conjugate gradient methods, Andrei (2011) noted that the nonnegative parameter t has no optimal value, and it may not necessarily generate descent directions (Babaie-Kafaki & Ghanbari, 2013). That is, the method may not satisfy the descent condition

$$F_k^T d_k < 0, \quad \forall k, \tag{2.3.18}$$

or the sufficient descent condition, namely there exists a constant $\lambda > 0$ such that

$$F_k^T d_k \leq -\lambda \|F_k\|^2, \quad \forall k. \tag{2.3.19}$$

In an attempt to develop DL-type CG methods for unconstrained optimization, reasonable and appropriate values for the DL parameter were provided in (Babaie-Kafaki & Ghanbari, 2013 ; Babaie-Kafaki & Ghanbari, 2014 ; Babaie-Kafaki & Ghanbari, 2015). Inspired by this development we propose to derive a family of CG methods for solving large-scale system of nonlinear equations.

CHAPTER THREE

METHODOLOGY

3.1 INTRODUCTION

In this chapter, four nonlinear CG methods are presented by employing extensions of some modified secant equations and carrying out eigenvalue analysis of the search direction matrices of the classical Dai-Liao (2001) and the one-parameter Hager-Zhang (2006) methods. Also throughout this work, the function f in (2.3.1) is specified to

$$f(x) := \frac{1}{2}\|F(x)\|^2, \tag{3.1.1}$$

where $\|.\|$ represents the Euclidean norm of vectors, $F_k = F(x_k)$, $F_{k-1} = F(x_{k-1})$, $y_{k-1} = F_k - F_{k-1}$ and $s_{k-1} = x_k - x_{k-1}$

3.2 ENHANCED DAI-LIAO CONJUGATE GRADIENT METHOD

This section presents a DL-type CG method, which is obtained by incorporating an extension of the standard secant equation (2.3.10) and modified secant equations proposed by Li and Fukushima (2001), Zhang et al. (1999), and Wei et al. (2006) in the DL (2001) method and carrying out eigenvalue analysis.

Recently, by applying the DL approach, Babaie-Kafaki and Ghanbari (2014) proposed the following extension of the PRP update parameter

$$\beta_k^{EPRP} = \beta_k^{PRP} - t\frac{F_k^T d_{k-1}}{\|F_{k-1}\|^2}, \tag{3.2.1}$$

where β_k^{PRP} is the classical PRP parameter and t is a nonnegative parameter, whose values were determined by carrying out eigenvalue analysis of the PRP search direction matrix. Motivated by this, and employing similar approach, we propose a modification of the classical DL update parameter. In what follows, we suggest an extension of some previously modified secant equations.

Following the Dai-Liao (2001) approach, Yabe and Takano (2004) presented nonlinear CG methods by applying the modified secant equation presented by Zhang and Xu (2001) and Zhang et al. (1999) with the following update parameters:

$$\beta_k^{new} = \frac{(\hat{y}_{k-1} - ts_{k-1})^T F_k}{d_{k-1}^T \hat{y}_{k-1}}, \quad t \geq 0, \tag{3.2.2}$$

where

$$\hat{y}_{k-1} = y_{k-1} + \left(\frac{\gamma_{k-1}}{s_{k-1}^T \mu_{k-1}}\right)\mu_{k-1}, \tag{3.2.3}$$

and

$$\gamma_{k-1} = 6(f_{k-1} - f_k) + 3(F_{k-1} + F_k)^T s_{k-1}, \tag{3.2.4}$$

where $\mu_{k-1} \in \mathbb{R}^n$ is a vector parameter such that $s_{k-1}^T \mu_{k-1} \neq 0$ (see (Zhang, Deng & Chen, 1999)) and $\mu_{k-1} = s_{k-1}$ (Yabe & Takano, 2004).

$$\beta_k^{new+} = \max\left\{\frac{F_k^T \hat{y}_{k-1}}{d_{k-1}^T \hat{y}_{k-1}}, 0\right\} - t\frac{F_k^T s_{k-1}}{d_{k-1}^T \hat{y}_{k-1}}, \tag{3.2.5}$$

Following a similar approach to Yabe and Takano (2004), Zhou and Zhang (2006) equally proposed a nonlinear CG method by applying the modified secant equation proposed by Li and Fukushima (2001) with the following update parameter:

$$\beta_k = \frac{(\hat{z}_{k-1} - ts_{k-1})^T F_k}{d_{k-1}^T \hat{z}_{k-1}}, \quad t \geq 0, \tag{3.2.6}$$

where

$$\hat{z}_{k-1} = y_{k-1} + C\|F_k\|^b s_{k-1}. \tag{3.2.7}$$

Similarly, Li et al. (2007) also applied the modified secant equation proposed by Wei et al. (2006) with the following update parameters:

$$\beta_k^{0*}(t) = \frac{(y_{k-1}^* - ts_{k-1})^T F_k}{d_{k-1}^T y_{k-1}^*}, \quad t \geq 0, \tag{3.2.8}$$

where

$$y_{k-1}^* = y_{k-1} + \left(\frac{\vartheta_{k-1}}{\|s_{k-1}\|^2}\right) s_{k-1}, \tag{3.2.9}$$

and

$$\vartheta_{k-1} = 2(f_{k-1} - f_k) + s_{k-1}^T (F_{k-1} + F_k). \tag{3.2.10}$$

$$\beta_k^{1*}(t) = \frac{(\hat{y}_{k-1}^* - ts_{k-1})^T F_k}{d_{k-1}^T \hat{y}_{k-1}^*}, \quad t \geq 0, \tag{3.2.11}$$

where

$$\hat{y}_{k-1}^* = \frac{\max\{\vartheta_{k-1}, 0\}}{\|s_{k-1}\|^2} s_{k-1}. \tag{3.2.12}$$

Here, in order to exploit the theoretical advantage of the modified secant conditions given in (3.2.3), (3.2.7), and (3.2.9), we propose the following extension

$$u_{k-1} = y_{k-1} + \phi \frac{\vartheta_{k-1}}{s_{k-1}^T \mu_{k-1}} \mu_{k-1} + C\|F_k\|^b s_{k-1}, \tag{3.2.13}$$

with ϕ a nonnegative parameter, ϑ_{k-1} as defined by (3.2.10) and $s_{k-1}^T \mu_{k-1} \neq 0$. We observe that for $\phi = 0$, (3.2.13) becomes (3.2.7), and for $\phi = C = 0$, it reduces to the standard secant equation defined by (2.3.10). Also, if $\phi = 1$, and $C = 0$, (3.2.13) becomes (3.2.10), while for $\phi = 3$ and $C = 0$, it reduces to (3.2.4). Substituting u_{k-1} in (3.2.13) for y_{k-1} in (2.3.17), we obtain the following version of the DL update parameter:

$$\bar{\beta}_k^{EDL} = \frac{(u_{k-1} - ts_{k-1})^T F_k}{d_{k-1}^T u_{k-1}}, \quad t \geq 0. \tag{3.2.14}$$

We observe that, in general, the denominator, $d_{k-1}^T u_{k-1}$ may not be nonzero since ϑ_{k-1} as defined in (3.2.10) may be non-positive. Therefore, we redefine u_{k-1} and obtain its revised form as

$$z_{k-1} = y_{k-1} + \phi \frac{\max\{\vartheta_{k-1}, 0\}}{s_{k-1}^T \mu_{k-1}} \mu_{k-1} + C\|F_k\|^b s_{k-1}. \tag{3.2.15}$$

Consequently, we get the revised form of (3.2.14) as

$$\hat{\beta}_k^{EDL} = \frac{(z_{k-1} - t s_{k-1})^T F_k}{d_{k-1}^T z_{k-1}}, \quad t \geq 0. \tag{3.2.16}$$

Andrei (2011) noted that the parameter t has no optimal choice and so, to obtain descent directions for our proposed method, we proceed to obtain appropriate values for t. By substituting (3.2.16) in (2.3.3) and setting $d_{k-1} = s_{k-1}$ for all $k \geq 1$, we obtain our search direction as

$$d_k = -F_k + \left(\frac{s_{k-1} z_{k-1}^T - t s_{k-1} s_{k-1}^T}{s_{k-1}^T z_{k-1}} \right) F_k. \tag{3.2.17}$$

Following Perry's approach (1978), we can write (3.2.17) as

$$d_k = -H_k F_k, \quad k \geq 1, \tag{3.2.18}$$

where H_k (called the search direction matrix) is given by

$$H_k = I - \frac{s_{k-1} z_{k-1}^T}{s_{k-1}^T z_{k-1}} + t \frac{s_{k-1} s_{k-1}^T}{s_{k-1}^T z_{k-1}}. \tag{3.2.19}$$

Clearly, H_k is not a symmetric matrix. To obtain a symmetrize form, we apply the approach adopted by Babaie-Kafaki and Ghanbari (2014). From (3.2.18) we can write

$$d_k^T F_k = -F_k^T H_k^T F_k = -F_k^T \frac{H_k^T + H_k}{2} F_k, \tag{3.2.20}$$

with

$$\bar{H}_k = \frac{H_k^T + H_k}{2} = I - \frac{1}{2}\left(\frac{s_{k-1}z_{k-1}^T + z_{k-1}s_{k-1}^T}{s_{k-1}^T z_{k-1}}\right) + t\frac{s_{k-1}s_{k-1}^T}{s_{k-1}^T z_{k-1}} \tag{3.2.21}$$

Thus, we have

$$d_k^T F_k = -F_k^T \bar{H}_k F_k, \quad k \geq 1. \tag{3.2.22}$$

Proposition 3.2.1 The matrix $\bar{H}_k$ defined by (3.2.21) is symmetric.

Proof. Direct computation shows that $\bar{H}_k = \bar{H}_k^T$. Hence, $\bar{H}_k$ is symmetric.

Theorem 3.2.2 Let the matrix $\bar{H}_k$ be defined by (3.2.21). Then, its eigenvalues consists of 1 with ($n-2$ multiplicity), λ_k^+ and λ_k^-, where

$$\lambda_k^+ = \frac{1}{2}\left[(1+v_k) + \sqrt{(v_k-1)^2 + w_k - 1}\right] \tag{3.2.23}$$

$$\lambda_k^- = \frac{1}{2}\left[(1+v_k) - \sqrt{(v_k-1)^2 + w_k - 1}\right] \tag{3.2.24}$$

with

$$v_k = t\frac{\|s_{k-1}\|^2}{s_{k-1}^T z_{k-1}} \tag{3.2.25}$$

$$w_k = \frac{\|s_{k-1}\|^2 \|z_{k-1}\|^2}{(s_{k-1}^T z_{k-1})^2}, \tag{3.2.26}$$

where $s_{k-1}^T z_{k-1} \neq 0$. Furthermore, all eigenvalues of $\bar{H}_k$ are positive real numbers.

Proof. Since $s_{k-1}^T z_{k-1} \neq 0$. So, the vectors s_{k-1} and z_{k-1} are nonzero vectors. Suppose span$\{s_{k-1}, z_{k-1}\} = V \subset R^n$, then $dim(V) \leq 2$ and $dim(V^\perp) \geq n-2$.

Therefore, there exists a set of mutually orthogonal vectors $\{\tau_{k-1}^i\}_{i=1}^{n-2} \subset V^\perp$ satisfying

$$s_{k-1}^T \tau_{k-1}^i = z_{k-1}^T \tau_{k-1}^i = 0. \tag{3.2.27}$$

Multiplying both sides of (3.2.21) by τ_{k-1}^i we obtain

$$\bar{H}_k \tau_{k-1}^i = \tau_{k-1}^i, \quad i = 1, ..., n-2, \tag{3.2.28}$$

which can be seen as an eigenvector equation. So, τ_{k-1}^{i}, for $i = 1,...,n-2$ are the eigenvectors of $\bar{H}_k$ with eigenvalue 1 each. Let λ_k^+ and λ_k^- be the remaining two eigenvalues respectively. Observe that (3.2.21) can be written as

$$\bar{H}_k = I - \frac{1}{2}\frac{s_{k-1}z_{k-1}^T}{s_{k-1}^T z_{k-1}} + \frac{(2ts_{k-1} - z_{k-1})s_{k-1}^T}{2s_{k-1}^T z_{k-1}}. \tag{3.2.29}$$

We can see that $\bar{H}_k$ represents a rank-two update, so from the fundamental algebra formula (see inequality $(1.2.70)$ of (Sun & Yuan, 2006)),

$$det(I + pq^T + uv^T) = (1 + q^T p)(1 + v^T u) - (p^T v)(q^T u). \tag{3.2.30}$$

From (3.2.29), Let $p = -s_{k-1}, \quad q = \frac{z_{k-1}}{2s_{k-1}^T z_{k-1}}, \quad u = \frac{2ts_{k-1} - z_{k-1}}{2s_{k-1}^T z_{k-1}}, \quad v = s_{k-1}$.
Therefore, since $s_{k-1}^T s_{k-1} = \|s_{k-1}\|^2$ and $z_{k-1}^T z_{k-1} = \|z_{k-1}\|^2$, we obtain

$$\begin{aligned}
det(\bar{H}_k) &= \left(1 - \frac{z_{k-1}^T s_{k-1}}{2s_{k-1}^T z_{k-1}}\right)\left(1 + s_{k-1}^T \frac{(2ts_{k-1} - z_{k-1})}{2s_{k-1}^T z_{k-1}}\right) + \left(\frac{t\|s_{k-1}\|^2}{2s_{k-1}^T z_{k-1}} - \frac{\|s_{k-1}\|^2\|z_{k-1}\|^2}{4(s_{k-1}^T z_{k-1})^2}\right) \\
&= \frac{1}{2}\left(1 + \frac{t\|s_{k-1}\|^2}{s_{k-1}^T z_{k-1}} - \frac{s_{k-1}^T z_{k-1}}{2s_{k-1}^T z_{k-1}}\right) + \left(\frac{t\|s_{k-1}\|^2}{2s_{k-1}^T z_{k-1}} - \frac{\|s_{k-1}\|^2\|z_{k-1}\|^2}{4(s_{k-1}^T z_{k-1})^2}\right) \\
&= \frac{1}{2} + \frac{t\|s_{k-1}\|^2}{2s_{k-1}^T z_{k-1}} - \frac{1}{4} + \frac{t\|s_{k-1}\|^2}{2s_{k-1}^T z_{k-1}} - \frac{\|s_{k-1}\|^2\|z_{k-1}\|^2}{4(s_{k-1}^T z_{k-1})^2} \\
&= t\frac{\|s_{k-1}\|^2}{s_{k-1}^T z_{k-1}} - \frac{1}{4}\frac{\|s_{k-1}\|^2\|z_{k-1}\|^2}{(s_{k-1}^T z_{k-1})^2} + \frac{1}{4}.
\end{aligned} \tag{3.2.31}$$

Also, since sum of the eigenvalues of a square symmetric matrix equals to its trace, from (3.2.21) we have

$$trace(\bar{H}_k) = n - 1 + t\frac{\|s_{k-1}\|^2}{s_{k-1}^T z_{k-1}} = \underbrace{1 + ... + 1}_{\text{(n-2)times}} + \lambda_k^+ + \lambda_k^-, \tag{3.2.32}$$

and we obtain

$$\lambda_k^+ + \lambda_k^- = 1 + t\frac{\|s_{k-1}\|^2}{s_{k-1}^T z_{k-1}}. \tag{3.2.33}$$

Using the relationship between trace and determinant of a matrix and its eigenvalues, we obtain λ_k^+ and λ_k^- as roots of the following quadratic polynomial:

$$\lambda^2 - \left(1 + t\frac{\|s_{k-1}\|^2}{s_{k-1}^T z_{k-1}}\right)\lambda + t\frac{\|s_{k-1}\|^2}{s_{k-1}^T z_{k-1}} - \frac{1}{4}\frac{\|s_{k-1}\|^2\|z_{k-1}\|^2}{(s_{k-1}^T z_{k-1})^2} + \frac{1}{4} = 0. \quad (3.2.34)$$

And applying the quadratic formula, we obtain

$$\lambda_k^{\pm} = \frac{1 + t\frac{\|s_{k-1}\|^2}{s_{k-1}^T z_{k-1}} \pm \sqrt{\left(1 + t\frac{\|s_{k-1}\|^2}{s_{k-1}^T z_{k-1}}\right)^2 + 4\left(\frac{1}{4} + t\frac{\|s_{k-1}\|^2}{s_{k-1}^T z_{k-1}} - \frac{1}{4}\frac{\|s_{k-1}\|^2\|z_{k-1}\|^2}{(s_{k-1}^T z_{k-1})^2}\right)}}{2}. \quad (3.2.35)$$

By expanding both terms of the discriminant in (3.2.35), we have

$$\lambda_k^{\pm} = \frac{1}{2}\left[1 + t\frac{\|s_{k-1}\|^2}{s_{k-1}^T z_{k-1}} \pm \sqrt{\left(t\frac{\|s_{k-1}\|^2}{s_{k-1}^T z_{k-1}} - 1\right)^2 + \frac{\|s_{k-1}\|^2\|z_{k-1}\|^2}{(s_{k-1}^T z_{k-1})^2} - 1}\right], \quad (3.2.36)$$

which proves (3.3.10) and (3.3.11). Furthermore, by Cauchy inequality, $\|s_{k-1}\|\|z_{k-1}\| \geq (s_{k-1}^T z_{k-1}), \quad \forall k \geq 1$. So, the discriminant of (3.2.36) is nonnegative. And since $1 + t\frac{\|s_{k-1}\|^2}{s_{k-1}^T z_{k-1}}$ is positive, we see that $\lambda_k^+ > 0$. In order to obtain λ_k^- as positive real number, we must have

$$\left[1 + t\frac{\|s_{k-1}\|^2}{s_{k-1}^T z_{k-1}} - \sqrt{\left(t\frac{\|s_{k-1}\|^2}{s_{k-1}^T z_{k-1}} - 1\right)^2 + \frac{\|s_{k-1}\|^2\|z_{k-1}\|^2}{(s_{k-1}^T z_{k-1})^2} - 1}\right] > 0. \quad (3.2.37)$$

So, we need to find values of t satisfying (3.2.37). From (3.2.37) we get

$$1 + t\frac{\|s_{k-1}\|^2}{s_{k-1}^T z_{k-1}} > \sqrt{\left(t\frac{\|s_{k-1}\|^2}{s_{k-1}^T z_{k-1}} - 1\right)^2 + \frac{\|s_{k-1}\|^2\|z_{k-1}\|^2}{(s_{k-1}^T z_{k-1})^2} - 1} \quad (3.2.38)$$

Squaring both sides of (3.2.38), we obtain

$$\left(1 + t\frac{\|s_{k-1}\|^2}{s_{k-1}^T z_{k-1}}\right)^2 > \left(t\frac{\|s_{k-1}\|^2}{s_{k-1}^T z_{k-1}} - 1\right)^2 + \frac{\|s_{k-1}\|^2\|z_{k-1}\|^2}{(s_{k-1}^T z_{k-1})^2} - 1. \quad (3.2.39)$$

Subtracting $\left(t\frac{\|s_{k-1}\|^2}{s_{k-1}^T z_{k-1}}-1\right)^2$ from both sides of (3.2.39), we get

$$\left(1+t\frac{\|s_{k-1}\|^2}{s_{k-1}^T z_{k-1}}\right)^2-\left(t\frac{\|s_{k-1}\|^2}{s_{k-1}^T z_{k-1}}-1\right)^2>\frac{\|s_{k-1}\|^2\|z_{k-1}\|^2}{(s_{k-1}^T z_{k-1})^2}-1 \tag{3.2.40}$$

And by expanding (3.2.40), we obtain

$$\frac{4t\|s_{k-1}\|^2}{s_{k-1}^T z_{k-1}}>\frac{\|s_{k-1}\|^2\|z_{k-1}\|^2}{(s_{k-1}^T z_{k-1})^2}-1. \tag{3.2.41}$$

Consequently, we obtain

$$t>\frac{1}{4}\left(\frac{\|z_{k-1}\|^2}{s_{k-1}^T z_{k-1}}-\frac{s_{k-1}z_{k-1}}{\|s_{k-1}\|^2}\right). \tag{3.2.42}$$

Hence, $\lambda_k^- > 0$ if (3.2.42) is satisfied. Finally, substituting the above estimation of t into (3.2.37) , we see that

$$\lambda_k^+ > \lambda_k^- > 0, \quad \forall k \geq 1. \tag{3.2.43}$$

Therefore, all the eigenvalues of the symmetric matrix $\bar{H}_k$ are positive real numbers, which ensures it is a positive-definite matrix. Hence, from (3.2.22), we have

$$d_k^T F_k = -F_k^T \bar{H}_k F_k \leq -\lambda_k^- \|F_k\|^2 < 0, \tag{3.2.44}$$

which clearly shows that the descent condition holds. Therefore, based on our eigenvalue analysis, we suggest the following two parameter choices for t in the proposed method:

$$t_k^{m,n} = m\frac{\|z_{k-1}\|^2}{s_{k-1}^T z_{k-1}} - n\frac{s_{k-1}^T z_{k-1}}{\|s_{k-1}\|^2}, \tag{3.2.45}$$

where $m > \frac{1}{4}$, and $n < \frac{1}{4}$.

Remark 3.2.3 Since the DL parameter t is nonnegative, we restrict the values of the parameter n in (3.2.45) to be negative so as to avoid a numerically unreasonable approximation.

Based on the above remark, we can write the modified Dai-Liao update parameter as

$$\beta_k^{EDL} = \left(\frac{(z_{k-1} - s_{k-1})^T F_k}{d_{k-1}^T z_{k-1}} \right) - t_k^{m,n} \frac{s_{k-1}^T F_k}{d_{k-1}^T z_{k-1}}. \tag{3.2.46}$$

with $m \geq \frac{1}{4}$ and $n \leq 0$ satisfying (3.2.45) and guaranteeing the descent condition. We also write the search direction for the proposed method as

$$d_k^{EDL} = -F_k + \left(\frac{(z_{k-1} - t_k^{m,n} s_{k-1})^T F_k}{d_{k-1}^T z_{k-1}} \right) d_{k-1}. \tag{3.2.47}$$

We apply the derivative-free line search proposed by Li and Fukushima (2000A) in order to compute our step length α_k. Let $\sigma_1 > 0$, $\sigma_2 > 0$ and $r \in (0,1)$ be constants and let $\{\eta_k\}$ be a given positive sequence such that

$$\sum_{k=0}^{\infty} \eta_k < \eta < \infty, \tag{3.2.48}$$

and

$$\|F_{k+1}\|^2 - \|F_k\|^2 \leq -\sigma_1 \|\alpha_k F_k\|^2 - \sigma_2 \|\alpha_k d_k\|^2 + \eta_k \|F_k\|^2. \tag{3.2.49}$$

Let i_k be the smallest non negative integer i such that (3.2.49) holds for $\alpha = r^i$. Let $\alpha_k = r^{i_k}$.

Next, we describe the algorithm of the proposed method as follows:

Algorithm 3.2.4 (**Enhanced Dai-Liao CG method** (**EDLCG**))
Step1: Given $\varepsilon > 0$, choose an initial point $x_0 \in \mathbf{R^n}$, a positive sequence $\{\eta_k\}$ satisfying (3.2.48), and constants $r \in (0,1), \sigma_1, \sigma_2 > 0$, $m \geq \frac{1}{4}$, $n < 0$. Compute $d_0 = -F_0$ and set $k = 0$.
Step2: Compute $F(x_k)$. If $\|F(x_k)\| \leq \varepsilon$, stop. Otherwise, compute the search direction d_k by (3.2.47).

Step3: Compute α_k via the line search in (3.2.49).
Step4: Set $x_{k+1} = x_k + \alpha_k d_k$.
Step5: Set $k := k+1$ and go to **Step** 2.

3.3 A DAI-LIAO CONJUGATE GRADIENT METHOD

In this section, we present a second DL-type method by applying the modified secant equations proposed by Zhang et al. (1999) and Wei et al. (2006) in the DL approach. Using (3.2.3) and (3.2.10), we propose the following extension:

$$B_k s_{k-1} = u^*_{k-1} = y_{k-1} + 2\zeta \frac{\vartheta_{k-1}}{s^T_{k-1}\mu_{k-1}}\mu_{k-1}, \tag{3.3.1}$$

where ζ is a nonnegative parameter, ϑ_{k-1} is defined by (3.2.10) and $s^T_{k-1}\mu_{k-1} \neq 0$. We observe that for $\zeta = 0$, (3.3.1) becomes the standard secant equation defined by (2.3.10), and if $\zeta = \frac{3}{2}$, (3.3.1) reduces to (3.2.4). Also, for $\zeta = \frac{1}{2}$, we see that (3.3.1) reduces to the modified secant equation proposed by Zhang et al. (1999). Substituting u^*_{k-1} in (3.3.1) for y_{k-1} in (2.3.17), we obtain the following version of the DL update parameter:

$$\bar{\beta}^{ADL}_k = \frac{(u^*_{k-1} - ts_{k-1})^T F_k}{d^T_{k-1}u^*_{k-1}}, \quad t \geq 0. \tag{3.3.2}$$

Observe that, in general, the denominator, $d^T_{k-1}u^*_{k-1}$ may not be nonzero since ϑ_{k-1} as defined in (3.2.10) may be non-positive. Therefore, we redefine u^*_{k-1} and obtain its revised form as

$$w_{k-1} = y_{k-1} + 2\zeta \frac{\max\{\vartheta_{k-1}, 0\}}{s^T_{k-1}\mu_{k-1}}\mu_{k-1}. \tag{3.3.3}$$

Consequently, we get the revised form of (3.3.2) as

$$\hat{\beta}^{ADL}_k = \frac{(w_{k-1} - ts_{k-1})^T F_k}{d^T_{k-1}w_{k-1}}, \quad t \geq 0. \tag{3.3.4}$$

By applying (3.3.4) in (2.3.3), as in the previous case, we obtain our search direction as

$$d_k = -F_k + \left(\frac{s_{k-1} w_{k-1}^T - t s_{k-1} s_{k-1}^T}{s_{k-1}^T w_{k-1}} \right) F_k. \tag{3.3.5}$$

Following Perry's approach (1978), we can write (3.3.5) as

$$d_k = -A_k F_k, \quad k \geq 1, \tag{3.3.6}$$

where A_k is known as the search direction matrix and is given by

$$A_k = I - \frac{s_{k-1} w_{k-1}^T}{s_{k-1}^T w_{k-1}} + t \frac{s_{k-1} s_{k-1}^T}{s_{k-1}^T w_{k-1}} \tag{3.3.7}$$

We observe that, A_k is not a symmetric matrix. To obtain a symmetrize form, we employ the approach adopted in (Babaie-Kafaki & Ghanbari, 2013 ; Babaie-Kafaki & Ghanbari, 2015). From (3.3.6) we can write

$$d_k^T F_k = -F_k^T A_k^T F_k = -F_k^T \bar{A}_k F_k, \quad k \geq 1, \tag{3.3.8}$$

with

$$\bar{A}_k = \frac{A_k^T + A_k}{2} = I - \frac{1}{2} \left(\frac{s_{k-1} w_{k-1}^T + w_{k-1} s_{k-1}^T}{s_{k-1}^T w_{k-1}} \right) + t \frac{s_{k-1} s_{k-1}^T}{s_{k-1}^T w_{k-1}}. \tag{3.3.9}$$

Proposition 3.3.1 The matrix $\bar{A}_k$ defined by (3.3.9) is symmetric.

Proof. By direct computation we see that $\bar{A}_k = \bar{A}_k^T$. So, $\bar{A}_k$ is symmetric.

Theorem 3.3.2 Let the matrix $\bar{A}_k$ be defined by (3.3.9). Then, its eigenvalues consists of 1 with ($n-2$ multiplicity), λ_k^+ and λ_k^-, where

$$\lambda_k^+ = \frac{1}{2} \left[(1 + a_k) + \sqrt{(a_k - 1)^2 + b_k - 1} \right] \tag{3.3.10}$$

$$\lambda_k^- = \frac{1}{2} \left[(1 + a_k) - \sqrt{(a_k - 1)^2 + b_k - 1} \right] \tag{3.3.11}$$

with

$$a_k = t\frac{\|s_{k-1}\|^2}{s_{k-1}^T w_{k-1}} \tag{3.3.12}$$

$$b_k = \frac{\|s_{k-1}\|^2\|w_{k-1}\|^2}{(s_{k-1}^T w_{k-1})^2}, \tag{3.3.13}$$

where $s_{k-1}^T w_{k-1} \neq 0$. Furthermore, all eigenvalues of $\bar{A}_k$ are positive real numbers.

Proof. Since $s_{k-1}^T w_{k-1} \neq 0$, then the vectors s_{k-1} and w_{k-1} are nonzero vectors. Suppose span$\{s_{k-1}, w_{k-1}\} = V \subset R^n$, then $dim(V) \leq 2$ and $dim(V^\perp) \geq n-2$.

Therefore, there exists a set of mutually orthogonal vectors $\{\tau_{k-1}^i\}_{i=1}^{n-2} \subset V^\perp$ satisfying

$$s_{k-1}^T \tau_{k-1}^i = w_{k-1}^T \tau_{k-1}^i = 0. \tag{3.3.14}$$

By multiplying both sides of (3.3.9) by τ_{k-1}^i we obtain

$$\bar{A}_k \tau_{k-1}^i = \tau_{k-1}^i, \quad i = 1, ..., n-2, \tag{3.3.15}$$

which can be viewed as an eigenvector equation. So, τ_{k-1}^i, for $i = 1, ..., n-2$ are the eigenvectors of $\bar{A}_k$ with eigenvalue 1 each. Let λ_k^+ and λ_k^- be the remaining two eigenvalues respectively. Also, since sum of the eigenvalues of a square matrix equals to its trace, we have

$$trace(\bar{A}_k) = n - 1 + t\frac{\|s_{k-1}\|^2}{s_{k-1}^T w_{k-1}} = \underbrace{1 + ... + 1}_{\text{(n-2)times}} + \lambda_k^+ + \lambda_k^-, \tag{3.3.16}$$

and we obtain

$$\lambda_k^+ + \lambda_k^- = 1 + t\frac{\|s_{k-1}\|^2}{s_{k-1}^T w_{k-1}}. \tag{3.3.17}$$

Also, applying the properties of Frobenius norm, we have

$$\|A_k\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} = \sqrt{trace(A_k^T A_k)} = \sqrt{trace(A_k^2)}. \tag{3.3.18}$$

By squaring both sides of (3.3.18), we get

$$\begin{aligned}\|A_k\|_F^2 &= trace\left(I-\frac{1}{2}\left(\frac{s_{k-1}w_{k-1}^T+w_{k-1}s_{k-1}^T}{s_{k-1}^Tw_{k-1}}\right)+t\frac{s_{k-1}s_{k-1}^T}{s_{k-1}^Tw_{k-1}}\right)^2\\ &= trace\left(I-\frac{3}{4}\left(\frac{s_{k-1}w_{k-1}^T}{s_{k-1}^Tw_{k-1}}+\frac{w_{k-1}s_{k-1}^T}{s_{k-1}^Tw_{k-1}}\right)+\frac{ts_{k-1}s_{k-1}^T}{s_{k-1}^Tw_{k-1}}+\frac{\|w_{k-1}\|^2s_{k-1}s_{k-1}^T}{4(s_{k-1}^Tw_{k-1})^2}\right)\\ &+\frac{\|s_{k-1}\|^2\|w_{k-1}\|^2}{4(s_{k-1}^Tw_{k-1})^2}-\frac{t\|s_{k-1}\|^2w_{k-1}s_{k-1}^T}{2(s_{k-1}^Tw_{k-1})^2}-\frac{t\|s_{k-1}\|^2s_{k-1}w_{k-1}^T}{2(s_{k-1}^Tw_{k-1})^2}+\frac{t^2\|s_{k-1}\|^2s_{k-1}s_{k-1}^T}{(s_{k-1}^Tw_{k-1})^2}\end{aligned} \tag{3.3.19}$$

Using the property that the trace of sum of n matrices equals to sum of trace of each individual matrix, we obtain

$$trace(A_k^2) = n-\frac{3}{2}+\frac{1}{2}\frac{\|s_{k-1}\|^2\|w_{k-1}\|^2}{(s_{k-1}^Tw_{k-1})^2}+\frac{t^2\|s_{k-1}\|^4}{(s_{k-1}^Tw_{k-1})^2} \tag{3.3.20}$$

Also, trace of the square of a normal matrix equals to sum of the squares of its eigenvalues. Since a symmetric matrix is normal, we can write

$$\underbrace{1+...+1}_{\text{(n-2) times}}+\lambda_k^{+2}+\lambda_k^{-2} = n-\frac{3}{2}+\frac{1}{2}\frac{\|s_{k-1}\|^2\|w_{k-1}\|^2}{(s_{k-1}^Tw_{k-1})^2}+\frac{t^2\|s_{k-1}\|^4}{(s_{k-1}^Tw_{k-1})^2} \tag{3.3.21}$$

for which we get

$$\lambda_k^{+2}+\lambda_k^{-2} = \frac{1}{2}+\frac{1}{2}\frac{\|s_{k-1}\|^2\|w_{k-1}\|^2}{(s_{k-1}^Tw_{k-1})^2}+\frac{t^2\|s_{k-1}\|^4}{(s_{k-1}^Tw_{k-1})^2}. \tag{3.3.22}$$

From (3.3.17) and (3.3.22), using the identity

$$(\lambda_k^++\lambda_k^-)^2 = \lambda_k^{+2}+2\lambda_k^+\lambda_k^-+\lambda_k^{-2}, \tag{3.3.23}$$

we get

$$\lambda_k^-\lambda_k^+ = \frac{1}{4}+t\frac{\|s_{k-1}\|^2}{s_{k-1}^Tw_{k-1}}-\frac{1}{4}\frac{\|s_{k-1}\|^2\|w_{k-1}\|^2}{(s_{k-1}^Tw_{k-1})^2}. \tag{3.3.24}$$

Also from (3.3.17) and (3.3.24), we can obtain λ_k^+ and λ_k^- as roots of the following quadratic polynomial

$$\lambda^2 - \left(1 + t\frac{\|s_{k-1}\|^2}{s_{k-1}^T w_{k-1}}\right)\lambda + \frac{1}{4} + t\frac{\|s_{k-1}\|^2}{s_{k-1}^T w_{k-1}} - \frac{1}{4}\frac{\|s_{k-1}\|^2\|w_{k-1}\|^2}{(s_{k-1}^T w_{k-1})^2} = 0. \quad (3.3.25)$$

Applying the quadratic formula, we obtain

$$\lambda_k^{\pm} = \frac{1 + t\frac{\|s_{k-1}\|^2}{s_{k-1}^T w_{k-1}} \pm \sqrt{\left(1 + t\frac{\|s_{k-1}\|^2}{s_{k-1}^T w_{k-1}}\right)^2 + 4\left(\frac{1}{4} + t\frac{\|s_{k-1}\|^2}{s_{k-1}^T w_{k-1}} - \frac{1}{4}\frac{\|s_{k-1}\|^2\|w_{k-1}\|^2}{(s_{k-1}^T w_{k-1})^2}\right)}}{2}, \quad (3.3.26)$$

which, following similar approach as in the previous case, can be re-written as

$$\lambda_k^{\pm} = \frac{1}{2}\left[1 + t\frac{\|s_{k-1}\|^2}{s_{k-1}^T w_{k-1}} \pm \sqrt{\left(t\frac{\|s_{k-1}\|^2}{s_{k-1}^T w_{k-1}} - 1\right)^2 + \frac{\|s_{k-1}\|^2\|w_{k-1}\|^2}{(s_{k-1}^T w_{k-1})^2} - 1}\right]. \quad (3.3.27)$$

Furthermore, by Cauchy inequality, $\|s_{k-1}\|\|w_{k-1}\| \geq (s_{k-1}^T w_{k-1}), \quad \forall k \geq 1.$
So, the discriminant of (3.3.27) is nonnegative. And since $1 + t\frac{\|s_{k-1}\|^2}{s_{k-1}^T w_{k-1}}$ is positive, we see that $\lambda_k^+ > 0$. In order to obtain λ_k^- as positive real number, we must have

$$\left[1 + t\frac{\|s_{k-1}\|^2}{s_{k-1}^T w_{k-1}} - \sqrt{\left(t\frac{\|s_{k-1}\|^2}{s_{k-1}^T w_{k-1}} - 1\right)^2 + \frac{\|s_{k-1}\|^2\|w_{k-1}\|^2}{(s_{k-1}^T w_{k-1})^2} - 1}\right] > 0. \quad (3.3.28)$$

Following similar approach as in the EDLCG method, we obtain the following estimate for the parameter t

$$t > \frac{1}{4}\left(\frac{\|w_{k-1}\|^2}{s_{k-1}^T w_{k-1}} - \frac{s_{k-1} w_{k-1}}{\|s_{k-1}\|^2}\right). \quad (3.3.29)$$

Hence, $\lambda_k^- > 0$ if (3.3.29) is satisfied. Finally, substituting the above estimation of t into (3.3.27) , we see that

$$\lambda_k^+ > \lambda_k^- > 0, \quad \forall k \geq 1. \quad (3.3.30)$$

Therefore, all the eigenvalues of the symmetric matrix $\bar{A}_k$ are positive real numbers, which ensures it is a positive-definite matrix. Hence, from (3.3.8), we get

$$d_k^T F_k = -F_k^T \bar{A}_k F_k \leq -\lambda_k^- \|F_k\|^2 < 0, \tag{3.3.31}$$

which ensures that the descent condition holds. Therefore, based on the above discussion, we suggest the following two parameter choices for t in the proposed method:

$$t_k^{ADL} = m^* \frac{\|s_{k-1}\|^2}{w_{k-1}^T s_{k-1}} - n^* \frac{s_{k-1}^T w_{k-1}}{\|s_{k-1}\|^2}, \tag{3.3.32}$$

where $m^* > \frac{1}{4}$, and $n^* < \frac{1}{4}$.

Remark 3.3.3 Since the DL parameter t is nonnegative, we restrict the values of the parameter n^* in (3.3.32) to be negative so as to avoid a numerically unreasonable approximation.

Based on the above remark, we can write the modified Dai-Liao update parameter as

$$\beta_k^{ADL} = \frac{(w_{k-1} - t^{ADL} s_{k-1})^T F_k}{d_{k-1}^T w_{k-1}}, \quad t \geq 0. \tag{3.3.33}$$

with $m^* \geq \frac{1}{4}$ and $n^* < 0$ satisfying (3.3.32) and guaranteeing the descent condition. We also write the search direction for the proposed method as

$$d_k^{ADL} = -F_k + \left(\frac{(w_{k-1} - t_k^{ADL} s_{k-1})^T F_k}{d_{k-1}^T w_{k-1}} \right) d_{k-1}. \tag{3.3.34}$$

We now describe the algorithm for our method.

Algorithm 3.3.4 (**A Dai-Liao CG method** (**ADLCG**))
Step1: Given $\varepsilon > 0$, choose an initial point $x_0 \in \mathbf{R^n}$, a positive sequence $\{\eta_k\}$ satisfying (3.2.48), and constants $r \in (0,1), \sigma_1, \sigma_2 > 0$, $m^* \geq \frac{1}{4}$, $n^* < 0$. Compute $d_0 = -F_0$ and set $k = 0$.
Step2: Compute $F(x_k)$. If $\|F(x_k)\| \leq \varepsilon$, stop. Otherwise, compute the search direction d_k by (3.3.34).
Step3: Compute α_k via the line search in (3.2.49).
Step4: Set $x_{k+1} = x_k + \alpha_k d_k$.
Step5: Set $k := k+1$ and go to **Step** 2.

3.4 MODIFIED HAGER-ZHANG CONJUGATE GRADIENT METHODS

In this section, based on eigenvalue analysis and the modified secant equation of Zhang et al. (1999), we modified the one-parameter Hager-Zhang method (2006) to present two new CG update parameters β_k, and consequently propose a family of Hager-Zhang CG methods for solving system of nonlinear equations.

3.4.1 Improved Hager-Zhang Conjugate Gradient Method

By modifying the Hestenes-Stiefel (1952) method, Hager and Zhang (2006) obtained the following one-parameter CG update parameter:

$$\beta_k^{HZ}(\theta) = \frac{F_k^T y_{k-1}}{d_{k-1}^T y_{k-1}} - \theta_k \left(\frac{\|y_{k-1}\|^2 F_k^T d_{k-1}}{(d_{k-1}^T y_{k-1})^2} \right), \tag{3.4.1}$$

where, $\theta_k \geq 0$, $d_{k-1}^T y_{k-1} \neq 0$ and $y_{k-1} = F_k - F_{k-1}, \quad \forall k \geq 1$.

Applying similar approach as in the previous cases, we obtain the search directions of the Hager-Zhang method as

$$d_k = -Q_k F_k, \quad k \geq 1, \tag{3.4.2}$$

where Q_k (known as the search direction matrix) is given by

$$Q_k = I - \frac{s_{k-1}y_{k-1}^T}{y_{k-1}^T s_{k-1}} + \theta_k \frac{\|y_{k-1}\|^2}{y_{k-1}^T s_{k-1}} \frac{s_{k-1}s_{k-1}^T}{y_{k-1}^T s_{k-1}}. \tag{3.4.3}$$

where $s_{k-1}^T y_{k-1} \neq 0, \forall k \geq 1$. By taking transpose of both sides of (3.4.2) and multiplying through by F_k, we get

$$d_k^T F_k = -F_k^T Q_k^T F_k. \tag{3.4.4}$$

And by converting Q_k to a symmetric matrix, we can write

$$d_k^T F_k = -F_k^T \frac{Q_k^T + Q_k}{2} F_k \tag{3.4.5}$$

where,

$$\bar{Q}_k = \frac{Q_k^T + Q_k}{2} = I - \frac{s_{k-1}y_{k-1}^T + y_{k-1}s_{k-1}^T}{2y_{k-1}^T s_{k-1}} + \theta_k \frac{\|y_{k-1}\|^2 s_{k-1}s_{k-1}^T}{(y_{k-1}^T s_{k-1})^2}. \tag{3.4.6}$$

And so, we need to find eigenvalues of $\bar{Q}_k$ in order to analyze the descent property of the proposed method. Since $y_{k-1}^T s_{k-1} \neq 0, \forall k \geq 1$, therefore, $s_{k-1} \neq 0$ and $y_{k-1} \neq 0$, which implies that the vectors y_{k-1} and s_{k-1} are nonzero vectors. Suppose span$\{s_{k-1}, y_{k-1}\} = V \subset R^n$, then $dim(V) \leq 2$ and $dim(V^\perp) \geq n-2$.
Therefore, there exists a set of mutually orthogonal vectors $\{u_{k-1}^i\}_{i=1}^{n-2} \subset V^\perp$ satisfying

$$s_{k-1}^T u_{k-1}^i = y_{k-1}^T u_{k-1}^i = 0, \tag{3.4.7}$$

which from (3.4.6) leads to

$$\bar{Q}_k u_{k-1}^i = u_{k-1}^i, \quad i = 1, ..., n-2. \tag{3.4.8}$$

So, u_{k-1}^i, for $i = 1, ..., n-2$ are the eigenvectors of $\bar{Q}_k$ with eigenvalue 1 each. Let λ_k^+ and λ_k^- be the remaining two eigenvalues respectively. Since sum of the eigenvalues of a square symmetric matrix equals to its trace, we have

$$trace(\bar{Q}_k) = n - 1 + \theta_k \frac{\|y_{k-1}\|^2 \|s_{k-1}\|^2}{(y_{k-1}^T s_{k-1})^2} = \underbrace{1 + ... + 1}_{\text{(n-2) times}} + \lambda_k^+ + \lambda_k^-, \tag{3.4.9}$$

for which we have

$$\lambda_k^+ + \lambda_k^- = 1 + \theta_k \frac{\|y_{k-1}\|^2 \|s_{k-1}\|^2}{(y_{k-1}^T s_{k-1})^2}. \tag{3.4.10}$$

Also, applying the properties of Frobenius norm, as in the previous case, we have

$$\begin{aligned} \|\bar{Q}_k\|_F^2 &= trace(\bar{Q}_k^2) \\ &= n - \frac{3}{2} + \frac{1}{2} \frac{\|s_{k-1}\|^2 \|y_{k-1}\|^2}{(y_{k-1}^T s_{k-1})^2} + \theta_k^2 \frac{\|y_{k-1}\|^4 \|s_{k-1}\|^4}{(y_{k-1}^T s_k)^4} \\ &= \underbrace{1 + ... + 1}_{\text{(n-2) times}} + \lambda_k^{+2} + \lambda_k^{-2}, \end{aligned} \tag{3.4.11}$$

for which we get

$$\lambda_k^{+2} + \lambda_k^{-2} = \frac{1}{2} + \frac{1}{2} \frac{\|s_{k-1}\|^2 \|y_{k-1}\|^2}{(y_{k-1}^T s_{k-1})^2} + \theta_k^2 \frac{\|y_{k-1}\|^4 \|s_{k-1}\|^4}{(y_{k-1}^T s_{k-1})^4}. \tag{3.4.12}$$

From equation (3.4.10) and (3.4.12), using the identity

$$(\lambda_k^+ + \lambda_k^-)^2 = \lambda_k^{+2} + 2\lambda_k^+ \lambda_k^- + \lambda_k^{-2}, \tag{3.4.13}$$

we get

$$\lambda_k^- \lambda_k^+ = \frac{1}{4} + \theta_k \frac{\|y_{k-1}\|^2 \|s_{k-1}\|^2}{(y_{k-1}^T s_{k-1})^2} - \frac{1}{4} \frac{\|s_{k-1}\|^2 \|y_{k-1}\|^2}{(y_{k-1}^T s_{k-1})^2}. \tag{3.4.14}$$

Also from (3.4.10) and (3.4.14), we can obtain λ_k^+ and λ_k^- as roots of the following quadratic equation

$$\lambda^2 - \left(1 + \theta_k \frac{\|y_{k-1}\|^2 \|s_{k-1}\|^2}{(y_{k-1}^T s_{k-1})^2}\right)\lambda + \frac{1}{4} + \theta_k \frac{\|y_{k-1}\|^2 \|s_{k-1}\|^2}{(y_{k-1}^T s_{k-1})^2} - \frac{1}{4}\frac{\|y_{k-1}\|^2 \|s_{k-1}\|^2}{(y_{k-1}^T s_{k-1})^2} = 0. \tag{3.4.15}$$

Applying the quadratic formula we obtain

$$\lambda_k^{\pm} = \frac{1 + \theta_k \frac{\|y_{k-1}\|^2 \|s_{k-1}\|^2}{(y_{k-1}^T s_{k-1})^2} \pm \sqrt{\left(\theta_k \frac{\|y_{k-1}\|^2 \|s_{k-1}\|^2}{(y_{k-1}^T s_{k-1})^2} - 1\right)^2 + \frac{\|s_{k-1}\|^2 \|y_{k-1}\|^2}{(s_{k-1}^T z_{k-1})^2} - 1}}{2}. \tag{3.4.16}$$

By Cauchy inequality $(y_{k-1}^T s_{k-1}) \leq \|y_{k-1}\| \|s_{k-1}\|$, which implies that the discriminant in (3.4.16) nonnegative. Hence, $\lambda_k^+ > 0$. And to obtain $\lambda_k^- > 0$, we must have

$$\frac{1 + \theta_k \frac{\|y_{k-1}\|^2 \|s_{k-1}\|^2}{(y_{k-1}^T s_{k-1})^2} - \sqrt{\left(\theta_k \frac{\|y_{k-1}\|^2 \|s_{k-1}\|^2}{(y_{k-1}^T s_{k-1})^2} - 1\right)^2 + \frac{\|s_{k-1}\|^2 \|y_{k-1}\|^2}{(s_{k-1}^T z_{k-1})^2} - 1}.}{2} > 0. \tag{3.4.17}$$

Squaring both sides of (3.4.17) and solving for θ_k as in the previous cases, we obtain the following approximation

$$\theta_k > \frac{1}{4}\left(1 - \frac{(y_{k-1}^T s_{k-1})^2}{\|y_{k-1}\|^2 \|s_{k-1}\|^2}\right), \tag{3.4.18}$$

for which $\lambda_k^- > 0$. Using (3.4.18) in (3.4.16), we see that

$$\lambda_k^+ > \lambda_k^- > 0, \quad \forall k \geq 1. \tag{3.4.19}$$

Therefore, all the eigenvalues of the symmetric matrix $\bar{Q}_k$ are positive and that ensures that it is a positive-definite matrix. Hence, from (3.4.5), we have

$$d_k^T F_k = -F_k^T \bar{Q}_k F_k \leq -\lambda_k^- \|F_k\|^2 < 0, \tag{3.4.20}$$

which shows that the descent condition is satisfied.

Therefore, based on our eigenvalue analysis, we choose the following as our first choice for the parameter θ_k.

$$\theta_{k1} = \xi - \psi \frac{(y_{k-1}^T s_{k-1})^2}{\|y_{k-1}\|^2 \|s_{k-1}\|^2}, \tag{3.4.21}$$

where $\xi \geq \frac{1}{4}$ and $\psi < \frac{1}{4}$.

Remark 3.4.1 In order to maintain the condition that the parameter $\theta_k \geq 0$, we restrict the values of the parameter ψ in (3.4.21) to be negative so as to avoid a numerically unreasonable approximation.

Based on the above remark, we can write the modified Hager-Zhang update parameter as

$$\beta_k^{IHZ} = \frac{F_k^T y_{k-1}}{d_{k-1}^T y_{k-1}} - \theta_{k1} \left(\frac{\|y_{k-1}\|^2 F_k^T d_{k-1}}{(d_{k-1}^T y_{k-1})^2} \right). \tag{3.4.22}$$

with $\xi \geq \frac{1}{4}$ and $\psi < 0$ satisfying (3.4.21) and guaranteeing the descent condition. We also write the search direction for the proposed method as

$$d_k = -F_k + \left(\frac{F_k^T y_{k-1}}{d_{k-1}^T y_{k-1}} - \theta_{k1} \left(\frac{\|y_{k-1}\|^2 F_k^T d_{k-1}}{(d_{k-1}^T y_{k-1})^2} \right) \right) d_{k-1}. \tag{3.4.23}$$

Algorithm 3.4.2 (**Improved Hager-Zhang CG method** (**IHZCG**))
Step1: Given $\varepsilon > 0$, choose an initial point $x_0 \in \mathbf{R^n}$, a positive sequence $\{\eta_k\}$ satisfying (3.2.48), and constants $r \in (0,1), \sigma_1, \sigma_2 > 0$, $\xi \geq \frac{1}{4}$, $\gamma \leq 0$. Compute $d_0 = -F_0$ and set $k = 0$.
Step2: Compute $F(x_k)$. If $\|F(x_k)\| \leq \varepsilon$, stop. Otherwise, compute the search direction d_k using (3.4.23).
Step3: Compute α_k via the line search in (3.2.49).
Step4: Set $x_{k+1} = x_k + \alpha_k d_k$.
Step5: Set $k := k+1$ and go to **Step** 2.

3.4.2 Enhanced Hager-Zhang Conjugate Gradient Method

Going by (Babaie-Kafaki & Ghanbari, 2013 ; Babaie-Kafaki & Ghanbari, 2014), we can see (3.4.1) as an adaptive version of the famous Dai-Liao (DL) update parameter,

$$\beta_k^{DL} = \frac{F_k^T y_{k-1}}{d_{k-1}^T y_{k-1}} - t\frac{F_k^T s_{k-1}}{d_{k-1}^T y_{k-1}}, \tag{3.4.24}$$

with $t = \theta_k \frac{\|y_{k-1}\|^2}{d_{k-1}^T y_{k-1}}$. Applying the idea presented in (Arazm, 2017), we use the modified secant equation proposed by Zhang et al. (1999) and Zhang and Xu (2001),

$$B_k s_{k-1} = \bar{z}_{k-1} = y_{k-1} + \rho\left(\frac{\gamma_{k-1}}{s_{k-1}^T \mu_{k-1}}\right)\mu_{k-1}, \quad k \geq 1 \tag{3.4.25}$$

where,

$$\gamma_{k-1} = 6(f_k - f_{k+1}) + 3(F_k + F_{k+1})^T s_{k-1}, \tag{3.4.26}$$

where, $\rho \geq 0$ is a constant and $\mu_{k-1} \in R^n$ satisfying $s_{k-1}^T \mu_{k-1} \neq 0$.
Substituting $\bar{z}_{k-1}$ for y_{k-1} in (3.4.1), we obtain the following modified Hager-Zhang update parameter,

$$\beta_k^{EHZ}(\theta) = \frac{F_k^T \bar{z}_{k-1}}{d_{k-1}^T \bar{z}_{k-1}} - \theta_k\left(\frac{\|\bar{z}_{k-1}\|^2 F_k^T d_{k-1}}{(d_{k-1}^T \bar{z}_{k-1})^2}\right), \tag{3.4.27}$$

where $\bar{z}_{k-1}$ is as defined in (3.4.25). We Observe that, in general, the denominator, $d_{k-1}^T \bar{z}_{k-1}$ may not be nonzero since γ_{k-1} as defined in (3.4.26) may be non-positive. We therefore redefine $\bar{z}_{k-1}$ and obtain the following revised form

$$\hat{z}_{k-1} = y_{k-1} + \rho\frac{\max\{\gamma_{k-1}, 0\}}{s_{k-1}^T \mu_{k-1}}\mu_{k-1}. \tag{3.4.28}$$

Consequently, the revised form of (3.4.27) becomes

$$\hat{\beta}_k^{EHZ}(\theta) = \frac{F_k^T \hat{z}_{k-1}}{d_{k-1}^T \hat{z}_{k-1}} - \theta_k\left(\frac{\|\hat{z}_{k-1}\|^2 F_k^T d_{k-1}}{(d_{k-1}^T \hat{z}_{k-1})^2}\right), \tag{3.4.29}$$

From (2.3.3) and (3.4.27) we can write search directions of our method as

$$d_k = -E_k F_k, \quad k = 1,2,..., \tag{3.4.30}$$

where E_k is the search direction matrix, which is given by

$$E_k = I - \frac{s_{k-1}\hat{z}_{k-1}^T}{\hat{z}_{k-1}^T s_{k-1}} + \theta_k \frac{\|\hat{z}_{k-1}\|^2}{\hat{z}_{k-1}^T s_{k-1}} \frac{s_{k-1}s_{k-1}^T}{\hat{z}_{k-1}^T s_{k-1}}. \tag{3.4.31}$$

Since E_k is not a symmetric matrix, to obtain a symmetric form we apply the approach adopted in (Babaie-Kafaki & Ghanbari, 2013 ; Babaie-Kafaki & Ghanbari, 2015). From (3.4.30) we can write

$$d_k^T F_k = -F_k^T E_k^T F_k = -F_k^T \frac{E_k^T + E_k}{2} F_k, \tag{3.4.32}$$

with

$$\bar{E_k} = \frac{E_k^T + E_k}{2} = I - \frac{1}{2}\frac{s_{k-1}\hat{z}_{k-1}^T + \hat{z}_{k-1}s_{k-1}^T}{\hat{z}_{k-1}^T s_{k-1}} + \theta_k \frac{\|\hat{z}_{k-1}\|^2 s_{k-1}s_{k-1}^T}{(\hat{z}_{k-1}^T s_{k-1})^2}. \tag{3.4.33}$$

Thus, we have

$$d_k^T F_k = -F_k^T \bar{E}_k F_k, \quad k \geq 1. \tag{3.4.34}$$

Proposition 3.4.3 The matrix $\bar{E}_k$ defined by (3.4.33) is symmetric.

Proof. Direct computation shows that $\bar{E}_k = \bar{E}_k^T$. Hence, $\bar{E}_k$ is symmetric.

Theorem 3.4.4 Let the matrix $\bar{E}_k$ be defined by (3.4.33). Then, its eigenvalues consists of 1 with ($n-2$ multiplicity), λ_k^+ and λ_k^-, where

$$\lambda_k^+ = \frac{1}{2}\left[(1+a_k) + \sqrt{(a_k-1)^2 + b_k - 1}\right] \tag{3.4.35}$$

$$\lambda_k^- = \frac{1}{2}\left[(1+a_k) - \sqrt{(a_k-1)^2 + b_k - 1}\right] \tag{3.4.36}$$

with

$$a_k- = \theta_k \frac{\|\hat{z}_{k-1}\|^2 \|s_{k-1}\|^2}{(s_{k-1}^T \hat{z}_{k-1})^2} \tag{3.4.37}$$

$$b_k = \frac{\|s_{k-1}\|^2 \|\hat{z}_{k-1}\|^2}{(s_{k-1}^T \hat{z}_{k-1})^2}, \tag{3.4.38}$$

where $s_{k-1}^T \hat{z}_{k-1} \neq 0$. Furthermore, all eigenvalues of $\bar{E}_k$ are positive real numbers.

Proof. Since $s_{k-1}^T \hat{z}_{k-1} \neq 0$. So, the vectors s_{k-1} and $\hat{z}_{k-1}$ are nonzero vectors. Suppose span$\{s_{k-1}, \hat{z}_{k-1}\} = V \subset R^n$, then $dim(V) \leq 2$ and $dim(V^{\perp}) \geq n-2$.

Therefore, there exists a set of mutually orthogonal vectors $\{v_{k-1}^i\}_{i=1}^{n-2} \subset V^{\perp}$ satisfying

$$s_{k-1}^T v_{k-1}^i = \hat{z}_{k-1}^T v_{k-1}^i = 0, \tag{3.4.39}$$

which from (3.4.33) leads to

$$\bar{E}_k v_{k-1}^i = v_{k-1}^i, \quad i = 1, ..., n-2. \tag{3.4.40}$$

So, v_{k-1}^i, for $i = 1, ..., n-2$ are the eigenvectors of $\bar{E}_k$ with eigenvalue 1 each. Let λ_k^+ and λ_k^- be the remaining two eigenvalues respectively. Observe that (3.4.33) can be written as

$$\bar{E}_k = I - \frac{s_{k-1}\hat{z}_{k-1}^T}{2 s_{k-1}^T \hat{z}_{k-1}} - \frac{\hat{z}_{k-1}^T s_{k-1} (\hat{z}_{k-1} - 2\theta_k \|\hat{z}_{k-1}\|^2 s_{k-1}) s_{k-1}^T}{2(s_{k-1}^T \hat{z}_{k-1})^2}. \tag{3.4.41}$$

We can see that $\bar{E}_k$ represents a rank-two update, so from the fundamental algebra formula (see inequality (1.2.70) of (Sun & Yuan, 2006)), we have

$$det(I + pq^T + uv^T) = (1 + q^T p)(1 + v^T u) - (p^T v)(q^T u). \tag{3.4.42}$$

From (3.4.41), Let $p = -s_{k-1}$, $\quad q = \frac{z_{k-1}}{2 s_{k-1}^T z_{k-1}}$, $\quad u = \frac{\hat{z}_{k-1}^T s_{k-1} (\hat{z}_{k-1} - 2\theta_k \|\hat{z}_{k-1}\|^2 s_{k-1})}{2(s_{k-1}^T \hat{z}_{k-1})^2}$, $v = s_{k-1}$.

Therefore, proceeding as in the EDLCG method, and since $s_{k-1}^T s_{k-1} = \|s_{k-1}\|^2$ and $\hat{z}_{k-1}^T \hat{z}_{k-1} = \|\hat{z}_{k-1}\|^2$, we obtain

$$det(\bar{E}_k) = \frac{1}{4} + \theta_k \frac{\|\hat{z}_{k-1}\|^2 \|s_{k-1}\|^2}{(\hat{z}_{k-1}^T s_{k-1})^2} - \frac{1}{4} \frac{\|s_{k-1}\|^2 \|\hat{z}_{k-1}\|^2}{(\hat{z}_{k-1}^T s_{k-1})^2}. \tag{3.4.43}$$

Also, since sum of the eigenvalues of a square matrix equals to its trace, from (3.4.33) we obtain

$$trace(\bar{E}_k) = n - 1 + \theta_k \frac{\|\hat{z}_{k-1}\|^2 \|s_{k-1}\|^2}{(\hat{z}_{k-1}^T s_{k-1})^2} = \underbrace{1 + \ldots + 1}_{\text{(n-2) times}} + \lambda_k^+ + \lambda_k^-, \tag{3.4.44}$$

for which we have

$$\lambda_k^+ + \lambda_k^- = 1 + \theta_k \frac{\|\hat{z}_{k-1}\|^2 \|s_{k-1}\|^2}{(\hat{z}_{k-1}^T s_{k-1})^2}. \tag{3.4.45}$$

Using the relationship between trace and determinant of a matrix and its eigenvalues, we obtain λ_k^+ and λ_k^- as roots of the following quadratic polynomial:

$$\lambda^2 - \left(1 + \theta_k \frac{\|\hat{z}_{k-1}\|^2 \|s_{k-1}\|^2}{(\hat{z}_{k-1}^T s_{k-1})^2}\right)\lambda + \frac{1}{4} + \theta_k \frac{\|\hat{z}_{k-1}\|^2 \|s_{k-1}\|^2}{(\hat{z}_{k-1}^T s_{k-1})^2} - \frac{1}{4} \frac{\|\hat{z}_{k-1}\|^2 \|s_{k-1}\|^2}{(\hat{z}_{k-1}^T s_{k-1})^2} = 0. \tag{3.4.46}$$

And applying the quadratic formula we obtain

$$\lambda_k^{\pm} = \frac{1 + \theta_k \frac{\|\hat{z}_{k-1}\|^2 \|s_{k-1}\|^2}{(\hat{z}_{k-1}^T s_{k-1})^2} \pm \sqrt{\left(\theta_k \frac{\|\hat{z}_{k-1}\|^2 \|s_{k-1}\|^2}{(\hat{z}_{k-1}^T s_{k-1})^2} - 1\right)^2 + \frac{\|s_{k-1}\|^2 \|\hat{z}_{k-1}\|^2}{(s_{k-1}^T \hat{z}_{k-1})^2} - 1}}{2}. \tag{3.4.47}$$

By Cauchy inequality $(\hat{z}_{k-1}^T s_{k-1}) \leq \|\hat{z}_{k-1}\| \|s_{k-1}\|$, which implies that the discriminant in (3.4.47) is nonnegative. Hence, $\lambda_k^+ > 0$. And to obtain $\lambda_k^- > 0$, we must have

$$\frac{1 + \theta_k \frac{\|\hat{z}_{k-1}\|^2 \|s_{k-1}\|^2}{(\hat{z}_{k-1}^T s_{k-1})^2} - \sqrt{\left(\theta_k \frac{\|\hat{z}_{k-1}\|^2 \|s_{k-1}\|^2}{(\hat{z}_{k-1}^T s_{k-1})^2} - 1\right)^2 + \frac{\|s_{k-1}\|^2 \|\hat{z}_{k-1}\|^2}{(s_{k-1}^T \hat{z}_{k-1})^2} - 1}.}{2} > 0. \tag{3.4.48}$$

Squaring both sides of (3.4.48) and solving for θ_k as in the previous cases, we obtain the following approximation

$$\theta_k > \frac{1}{4}\left(1 - \frac{(\hat{z}_{k-1}^T s_{k-1})^2}{\|\hat{z}_{k-1}\|^2 \|s_{k-1}\|^2}\right), \tag{3.4.49}$$

for which $\lambda_k^- > 0$. Using (3.4.49) in (3.4.47), we see that

$$\lambda_k^+ > \lambda_k^- > 0, \quad \forall k \geq 1. \tag{3.4.50}$$

Therefore, all the eigenvalues of the symmetric matrix $\bar{E}_k$ are positive and that ensures that it is a positive-definite matrix. Hence, from (3.4.34), we have

$$d_k^T F_k = -F_k^T \bar{E}_k F_k \leq -\lambda_k^- \|F_k\|^2 < 0, \tag{3.4.51}$$

which shows that the descent condition is satisfied.

Therefore, based on the eigenvalue analysis, we choose the following as our second choice of the parameter θ_k.

$$\theta_{k2} = p^* - q^* \frac{(\hat{z}_{k-1}^T s_{k-1})^2}{\|\hat{z}_{k-1}\|^2 \|s_{k-1}\|^2}, \tag{3.4.52}$$

where $p^* \geq \frac{1}{4}$ and $q^* < \frac{1}{4}$.

Remark 3.4.5 In order to maintain the condition that the parameter $\theta_k \geq 0$, we restrict the values of the parameter q^* in (3.4.52) to be negative so as to avoid a numerically unreasonable approximation.

Based on the above remark, we can write our modified update parameter as

$$\beta_k^{EHZ} = \frac{F_k^T \hat{z}_{k-1}}{d_{k-1}^T \hat{z}_{k-1}} - \theta_{k2} \left(\frac{\|\hat{z}_{k-1}\|^2 F_k^T d_{k-1}}{(d_{k-1}^T \hat{z}_{k-1})^2} \right). \tag{3.4.53}$$

with $p^* \geq \frac{1}{4}$ and $q^* < 0$ satisfying (3.4.52) and guaranteeing the descent condition. We also write the search direction for the proposed method as

$$d_k = -F_k + \left(\frac{F_k^T \hat{z}_{k-1}}{d_{k-1}^T \hat{z}_{k-1}} - \theta_{k2} \left(\frac{\|\hat{z}_{k-1}\|^2 F_k^T d_{k-1}}{(d_{k-1}^T \hat{z}_{k-1})^2} \right) \right) d_{k-1}. \tag{3.4.54}$$

Algorithm 3.4.6 (Enhanced Hager-Zhang CG method (EHZCG))
Step1: Given $\varepsilon > 0$, choose an initial point $x_0 \in \mathbf{R^n}$, a positive sequence $\{\eta_k\}$ satisfying (3.2.48), and constants $r \in (0,1), \sigma_1, \sigma_2 > 0$, $\xi \geq \frac{1}{4}$, $\gamma < 0$. Compute

$d_0 = -F_0$ and set $k = 0$.

Step2: Compute $F(x_k)$. If $\|F(x_k)\| \leq \varepsilon$, stop. Otherwise, compute the search direction d_k using(3.4.54).

Step3: Compute α_k via the line search in (3.2.49).

Step4: Set $x_{k+1} = x_k + \alpha_k d_k$.

Step5: Set $k := k+1$ and go to **Step** 2.

3.5 CONVERGENCE ANALYSIS

In this section, we present convergence results of our proposed methods. We require the following Assumptions

Assumption 3.5.1 The level set

$$\Omega = \{x | \|F(x)\| \leq \|F(x_0)\|\} \tag{3.5.1}$$

is bounded, namely, there exists a constant $B > 0$ such that $\|F(x)\| \leq B$, $\forall x \in \Omega$.

Assumption 3.5.2 .

(1) The solution set of problem (1.2.1) is not empty.

(2) F is continuously differentiable on an open convex set Φ_1 containing Ω.

(3) F is Lipschitz continuous in some neighborhood $\mathbf{N}$ of Ω; namely, there exists a positive constant $L > 0$ such that,

$$\|F(x) - F(y)\| \leq L\|x - y\|, \tag{3.5.2}$$

for all $x, y \in N$.

Assumption 3.5.1 and condition (3) of Assumption 3.5.2 imply (see Proposition 1.3 of (Babaie-Kafaki, Fatemi, & Mahdavi-Amiri, 2011)) that there exists a positive constant ω such that

$$\|F(x_k)\| \leq \omega, \tag{3.5.3}$$

for all $x \in \Omega$.

(4) The Jacobian of F is bounded, symmetric and positive-definite on Φ_1, which implies that there exists constants $m_2 \geq m_1 > 0$ such that

$$\|F'(x)\| \leq m_2, \quad \forall x \in \Phi_1, \tag{3.5.4}$$

and

$$m_1\|d\|^2 \leq d^T F'(x)d, \quad \forall x \in \Phi_1, d \in \mathbf{R}^n. \tag{3.5.5}$$

3.5.1 Convergence Result of Enhanced Dai-Liao CG Method

Lemma 3.5.3 Let $\{x_k\}$ be generated by the Algorithm 3.2.4, then d_k is a descent direction for $F(x_k)$ at x_k. i.e

$$F(x)^T d_k < 0. \tag{3.5.6}$$

Proof. By (3.2.44), the Lemma is true and we can deduce that the norm function $f(x_k)$ is a descent along the direction d_k. i.e $\|F(x_{k+1})\| \leq \|F(x_k)\|$ is true $\forall k$.

Lemma 3.5.4 Suppose Assumptions 3.5.1 and 3.5.2 holds. Let $\{x_k\}$ be generated by the Algorithm 3.2.4, then $\{x_k\} \subset \Omega$. Moreover, $\{\|F_k\|\}$ converges.

Proof. By Lemma 3.5.3, we have $\|F(x_{k+1})\| \leq \|F(x_k)\|$. So, by Lemma 3.3 in (Dai, Han & Li, 1999), we conclude that $\{\|F_k\|\}$ converges. Moreover, for all k, we have

$$\|F(x_{k+1})\| \leq \|F(x_k)\| \leq \|F(x_{k-1}\| ... \leq \|F(x_0\|. \tag{3.5.7}$$

This implies that $\{x_k\} \subset \Omega$.

Lemma 3.5.5 Suppose Assumptions 3.5.1 and 3.5.2 holds. Let $\{x_k\}$ be generated by the Algorithm 3.2.4, then

$$\lim_{k\to\infty} \|\alpha_k d_k\| = \lim_{k\to\infty} \|s_k\| = 0, \tag{3.5.8}$$

and

$$\lim_{k\to\infty} \|\alpha_k F(x_k)\| = 0. \tag{3.5.9}$$

Proof. From the line search (3.2.49) and for all $k > 0$, we obtain

$$\begin{aligned}\sigma_2\|\alpha_k d_k\|^2 &\leq \sigma_1\|\alpha_k F_k\|^2 + \sigma_2\|\alpha_k d_k\|^2 \\ &\leq \|F_k\|^2 - \|F_{k+1}\|^2 + \eta_k\|F_k\|^2.\end{aligned} \tag{3.5.10}$$

And by summing up the above k inequality, we obtain

$$\begin{aligned}\sigma_2\sum_{i=0}^{k}\|\alpha_k d_k\|^2 &\leq \sum_{i=0}^{k}(\|F(x_i)\|^2 - \|F(x_{i+1})\|^2) + \sum_{i=0}^{k}\eta_i\|F(x_i)\|^2 \\ &= \|F(x_0)\|^2 - \|F(x_{k+1})\|^2 + \sum_{i=0}^{k}\eta_i\|F(x_i)\|^2 \\ &\leq \|F(x_0)\|^2 + \|F(x_0)\|^2\sum_{i=0}^{k}\eta_i \\ &\leq \|F(x_0)\|^2 + \|F(x_0)\|^2\sum_{i=0}^{\infty}\eta_i.\end{aligned} \tag{3.5.11}$$

Therefore, by (3.5.1) and since $\{\eta_i\}$ satisfies (3.2.48), then the series $\sum_{i=0}^{k}\|\alpha_k d_k\|^2$ is convergent, which implies that (3.5.8) holds. Using the same argument as above, with $\sigma_1\|\alpha_k F(x_k)\|^2$ on the left-hand sides, we obtain (3.5.9).

Lemma 3.5.6 (Yuan* & Lu, 2008). Suppose Assumptions 3.5.1 and 3.5.2 holds and $\{x_k\}$ be generated by Algorithm 3.2.4. Then there exists a constant $m > 0$ such that,

$$y_k^T s_k \geq m_2\|s_k\|^2 > 0, \quad \forall k \geq 1. \tag{3.5.12}$$

Proof. By mean-value theorem we have

$$y_k^T s_k = s_k^T(F(x_{k+1}) - F(x_k)) = s_k^T F'(\varphi)s_k \geq m_1\|s_k\|^2 > 0, \tag{3.5.13}$$

where $\varphi = \lambda x_k + (1-\lambda)x_{k+1}$, for some $\lambda \in (0,1)$. We obtain the last inequality from (3.5.5). Letting $m_1 = m_2$ the proof is established.

Lemma 3.5.7 Suppose Assumptions 3.5.1 and 3.5.2 holds. Let the sequence $\{x_k\}$ be generated by Algorithm 3.2.4 with update parameter β_k^{EDL}, then, there exists

$M > 0$ such that

$$\|d_k^{EDL}\| \leq M, \quad \forall k. \tag{3.5.14}$$

Proof. By (3.2.15) and Lemma 3.5.6, we have

$$\begin{aligned} s_{k-1}^T z_{k-1} &= s_{k-1}^T y_{k-1} + \phi \frac{\max\{\vartheta_{k-1}, 0\}}{s_{k-1}^T \mu_{k-1}} s_{k-1}^T \mu_{k-1} + C\|F_k\|^b \|s_{k-1}\|^2 \\ &\geq s_{k-1}^T y_{k-1} \geq m_2 \|s_{k-1}\|^2. \end{aligned} \tag{3.5.15}$$

Applying the mean-value theorem we have

$$\begin{aligned} |\vartheta_{k-1}| &= |2(f_k - f_{k+1}) + (F_{k-1} + F_k)^T s_{k-1}| \\ &= |(-2\nabla f(\zeta) + \nabla f(x_k) + \nabla f(x_{k+1}))^T s_{k-1}|, \end{aligned} \tag{3.5.16}$$

where $\zeta = \varsigma x_k + (1-\varsigma)x_{k+1}$, for some $\varsigma \in (0,1)$.

Hence from (3.5.2), we obtain

$$\begin{aligned} |\vartheta_{k-1}| &\leq (\|\nabla f(x_k) - \nabla f(\zeta)\| + \|\nabla f(x_{k+1}) - \nabla f(\zeta)\|)\|s_{k-1}\| \\ &\leq (L(1-\varsigma)\|s_{k-1}\| + L\varsigma\|s_{k-1}\|)\|s_{k-1}\| \\ &= L\|s_{k-1}\|^2. \end{aligned} \tag{3.5.17}$$

Also, from (3.2.15), (3.5.2), (3.5.35), and by setting $\mu_{k-1} = s_{k-1}$ and $\|F_k\| = \rho$, we obtain

$$\begin{aligned} \|z_{k-1}\| &\leq \|y_{k-1}\| + \phi \frac{|\vartheta_{k-1}|}{|s_{k-1}^T s_{k-1}|} \|s_{k-1}\| + C\rho^b \|s_{k-1}\| \\ &\leq L\|s_{k-1}\| + \phi L\|s_{k-1}\| + C\rho^b \|s_{k-1}\| \\ &= (L + \phi L + C\rho^b)\|s_{k-1}\|. \end{aligned} \tag{3.5.18}$$

And by using (3.2.45), (3.5.15) and (3.5.18), we get

$$\begin{aligned} |t^{EDL}| &= \left| m\frac{\|z_{k-1}\|^2}{s_{k-1}^T z_{k-1}} - n\frac{s_{k-1}^T z_{k-1}}{\|s_{k-1}\|^2} \right| \\ &\leq m\frac{((L+\phi L+C\rho^b)\|s_{k-1}\|)^2}{m_2\|s_{k-1}\|^2} + |n|\frac{m_2\|s_{k-1}\|^2}{\|s_{k-1}\|^2} \\ &= m\frac{(L+\phi L+C\rho^b)^2}{m_2} + |n|m_2. \end{aligned} \tag{3.5.19}$$

Also, from (2.3.3), (3.5.18), (3.5.19) and by setting $C\rho^b = \hat{m}$ we obtain,

$$\begin{aligned} \|d_k^{EDL}\| &= \| - F(x_k) + \beta_k^{EDL} d_{k-1}\| \\ &= \|F(x_k)\| + \frac{\|F(x_k)\|\|z_{k-1}\|}{s_{k-1}^T z_{k-1}}\|s_{k-1}\| + |t^{EDL}|\frac{\|F(x_k)\|\|s_{k-1}\|}{s_{k-1}^T z_{k-1}}\|s_{k-1}\| \\ &\leq \rho + \frac{\rho(L+\phi L+\hat{m})}{m_2} + \left(m\frac{(L+\phi L+\hat{m})^2}{m_2} + |n|m_2 \right)\frac{\rho}{m_2} \\ &= \left(1 + \frac{(L+\phi L+\hat{m})}{m_2} + m\frac{(L+\phi L+\hat{m})^2}{m_2^2} + |n| \right)\rho \\ &= \frac{(m_2^2 + m_2(L+\phi L+\hat{m}) + m(L+L\phi+\hat{m})^2 + |n|)\rho}{m_2^2} \\ &= \frac{c_1\rho}{m_2^2}, \end{aligned} \tag{3.5.20}$$

where $c_1 = (m_2^2 + m_2(L+\phi L+\hat{m}) + m(L+L\phi+\hat{m})^2 + |n|)$.

Setting $M := \frac{c_1\rho}{m^2}$ we obtain the required result.

We use the next theorem to establish global convergence of Algorithm 3.2.4.

Theorem 3.5.8 Suppose Assumptions 3.5.1 and 3.5.2 holds. Let the sequence $\{x_k\}$ be generated by Algorithm 3.2.4. Also, assume that for all $k > 0$

$$\alpha_k \geq c\frac{|F(x_k)^T d_k|}{\|d_k\|^2}, \tag{3.5.21}$$

where c is some positive constant. Then $\{x_k\}$ converges globally to a solution of problem (1.2.1); i.e,

$$\lim_{k\to\infty} \|F(x_k)\| = 0. \tag{3.5.22}$$

Proof. From (3.5.8) and the boundedness of $\{\|d_k\|\}$, we have

$$\lim_{k\to\infty} \alpha_k \|d_k\|^2 = 0. \tag{3.5.23}$$

From (3.5.21) and (3.5.23) we have

$$\lim_{k\to\infty} |F(x_k)^T d_k| = 0. \tag{3.5.24}$$

On the other hand, from (3.2.44) we have

$$\begin{aligned} F(x_k)^T d_k &= -\lambda_k^- \|F(x_k)\|^2 \\ \|F(x_k)\|^2 &= \left\| -\frac{1}{\lambda_k^-} F(x_k)^T d_k \right\| \\ &\leq |F(x_k)^T d_k| \left| \frac{1}{\lambda_k^-} \right|. \end{aligned} \tag{3.5.25}$$

But from (3.3.30), we have

$$\lambda_k^+ > \lambda_k^- > 0, \quad \forall k. \tag{3.5.26}$$

Thus, from (3.5.25) and applying the sandwich theorem, we obtain

$$0 \leq \|F(x_k)\|^2 \leq |F(x_k)^T d_k| \left(\frac{1}{\lambda_k^-} \right) \longrightarrow 0. \tag{3.5.27}$$

Therefore,

$$\lim_{k\to\infty} \|F(x_k)\| = 0. \tag{3.5.28}$$

And the proof is completed.

3.5.2 Convergence Result of A Dai-Liao CG Method

Lemma 3.5.9 Let $\{x_k\}$ be generated by Algorithm 3.3.4, then d_k is a descent direction for $F(x_k)$ at x_k. i.e

$$F(x)^T d_k < 0. \tag{3.5.29}$$

Proof. By (3.3.31), the lemma is true and we can deduce that the norm function $f(x_k)$ is a descent along the direction d_k. i.e $\|F(x_{k+1})\| \leq \|F(x_k)\|$ is true $\forall k$.

Lemma 3.5.10 Suppose Assumptions 3.5.1 and 3.5.2 holds. Let $\{x_k\}$ be generated by the Algorithm 3.3.4, then $\{x_k\} \subset \Omega$. Moreover, $\{\|F_k\|\}$ converges.

Proof. Similar to the one obtained for Algorithm 3.2.4.

Lemma 3.5.11 Suppose Assumptions 3.5.1 and 3.5.2 holds. Let $\{x_k\}$ be generated by the Algorithm 3.3.4, then

$$\lim_{k\to\infty} \|\alpha_k d_k\| = \lim_{k\to\infty} \|s_k\| = 0, \tag{3.5.30}$$

and

$$\lim_{k\to\infty} \|\alpha_k F(x_k)\| = 0. \tag{3.5.31}$$

Proof. The proof is similar to the one obtained for Algorithm 3.2.4.

Lemma 3.5.12 Suppose Assumptions 3.5.1 and 3.5.2 holds. Let $\{x_k\}$ be generated by the Algorithm 3.3.4, then, there exists $M > 0$ such that

$$\|d_k^{ADL}\| \leq M, \quad \forall k. \tag{3.5.32}$$

Proof. Using (3.3.3) and (3.5.12), we get

$$s_{k-1}^T w_{k-1} = s_{k-1}^T y_{k-1} + 2\phi \frac{\max\{\vartheta_{k-1}, 0\}}{s_{k-1}^T \mu_{k-1}} s_{k-1}^T \mu_{k-1} \geq s_{k-1}^T y_{k-1} \geq m_2 \|s_{k-1}\|^2. \tag{3.5.33}$$

Applying the mean-value theorem we have

$$\begin{aligned} |\vartheta_{k-1}| &= |2(f_k - f_{k+1}) + (F_{k-1} + F_k)^T s_{k-1}| \\ &= |(-2\nabla f(\varphi) + \nabla f(x_k) + \nabla f(x_{k+1}))^T s_{k-1}|, \end{aligned} \tag{3.5.34}$$

where $\varphi = \lambda x_k + (1-\lambda)x_{k+1}$, for some $\lambda \in (0,1)$.

Hence from (3.5.2), we have

$$\begin{aligned} |\vartheta_{k-1}| &\leq (\|\nabla f(x_k) - \nabla f(\varphi)\| + \|\nabla f(x_{k+1} - \nabla f(\varphi)\|)\|s_{k-1}\| \\ &\leq (L(1-\lambda)\|s_{k-1}\| + L\lambda\|s_{k-1}\|)\|s_{k-1}\| \\ &= L\|s_{k-1}\|^2. \end{aligned} \tag{3.5.35}$$

Utilizing (3.3.3), (3.5.2), (3.5.35), and setting $\mu_{k-1} = s_{k-1}$, we obtain

$$\begin{aligned} \|w_{k-1}\| &\leq \|y_{k-1}\| + 2\phi\frac{|\vartheta_{k-1}|}{|s_{k-1}^T s_{k-1}|}\|s_{k-1}\| \\ &\leq L\|s_{k-1}\| + 2\phi L\frac{\|s_{k-1}\|^2}{\|s_{k-1}\|^2}\|s_{k-1}\| \\ &= (L+2\phi L)\|s_{k-1}\|. \end{aligned} \tag{3.5.36}$$

And by using (3.3.32), (3.5.2) and (3.5.36), we get

$$\begin{aligned} |t^{ADL}| &= \left| m^* \frac{\|w_{k-1}\|^2}{s_{k-1}^T w_{k-1}} - n^* \frac{s_{k-1}^T w_{k-1}}{\|s_{k-1}\|^2} \right| \\ &\leq \left| m^* \frac{\|w_{k-1}\|^2}{s_{k-1}^T w_{k-1}} \right| + \left| n^* \frac{s_{k-1}^T w_{k-1}}{\|s_{k-1}\|^2} \right| \\ &\leq m^* \frac{((L+2\phi L)\|s_{k-1}\|)^2}{m_2\|s_{k-1}\|^2} + |n^* \frac{m_2\|s_{k-1}\|^2}{\|s_{k-1}\|^2} \\ &= m^* \frac{(L+2\phi L)^2}{m_2} + m_2|n^*|. \end{aligned} \tag{3.5.37}$$

By utilizing (2.3.3), (3.3.32), (3.3.33), (3.5.36) and (3.5.37) we obtain,

$$\begin{aligned}
\|d_k^{ADL}\| &= \| -F(x_k) + \beta_k^{ADL} d_{k-1}\| \\
&= \| -F(x_k) + \frac{F(x_k)^T w_{k-1}}{s_{k-1}^T w_{k-1}} s_{k-1} - t^{ADL} \frac{F(x_k)^T s_{k-1}}{s_{k-1}^T w_{k-1}} s_{k-1}\| \\
&\leq \|F(x_k)\| + \frac{\|F(x_k)\|\|w_{k-1}\|}{s_{k-1}^T w_{k-1}} \|s_{k-1}\| + |t^{ADL}| \frac{\|F(x_k)\|\|s_{k-1}\|}{s_{k-1}^T w_{k-1}} \|s_{k-1}\| \\
&\leq \|F(x_k)\| + \frac{\|F(x_k)\|(L+2\phi L)}{m_2} + \left(m^* \frac{(L+2\phi L)^2}{m_2} + m_2 |n^*| \right) \frac{\|F(x_k)\|}{m_2} \\
&= \left(1 + \frac{(L+2\phi L)}{m_2} + \left(m^* \frac{(L+2\phi L)^2}{m_2^2} + |n^*| \right) \right) \|F(x_k)\| \\
&= \frac{(m^2 + m_2(L+2\phi L) + ((L+2\phi L)^2 m^* + |n^*|))\omega}{m^2} \\
&= \frac{c_1 \omega}{m_2^2}
\end{aligned} \tag{3.5.38}$$

where $c_1 = (m_2^2 + m_2(L+2\phi L) + ((L+2\phi L)^2 m^* + |n^*|))$.

Setting $M := \frac{c_1 \omega}{m_2^2}$ we obtain the required result.

Next, we prove the global convergence of Algorithm 3.3.4.

Theorem 3.5.13 Suppose Assumptions 3.5.1 and 3.5.2 holds. Let $\{x_k\}$ be generated by the Algorithm 3.3.4. Also, assume that for all $k > 0$

$$\alpha_k \geq c \frac{|F(x_k)^T d_k|}{\|d_k\|^2}, \tag{3.5.39}$$

where c is some positive constant. Then $\{x_k\}$ converges globally to a solution of problem (1.2.1); i.e,

$$\lim_{k \to \infty} \|F(x_k)\| = 0. \tag{3.5.40}$$

Proof. Similar to the one obtained for Algorithm 3.2.4.

3.5.3 Convergence Result of Improved Hager-Zhang CG method

Lemma 3.5.14 Let $\{x_k\}$ be generated by Algorithm 3.4.2, then d_k is a descent direction for $F(x_k)$ at x_k. i.e

$$F(x)^T d_k < 0. \tag{3.5.41}$$

Proof. By (3.4.20), the Lemma is true and we can deduce that the norm function $f(x_k)$ is a descent along the direction d_k. i.e $\|F(x_{k+1})\| \leq \|F(x_k)\|$ is true $\forall k$.

Lemma 3.5.15 Suppose Assumptions 3.5.1 and 3.5.2 holds. Let $\{x_k\}$ be generated by Algorithm 3.4.2, then $\{x_k\} \subset \Omega$. Moreover, $\{\|F_k\|\}$ converges.

Proof. Similar to the one obtained for Algorithm 3.2.4

Lemma 3.5.16 Suppose Assumptions 3.5.1 and 3.5.2 holds. Let $\{x_k\}$ be generated by the Algorithm 3.4.2, then

$$\lim_{k\to\infty} \|\alpha_k d_k\| = \lim_{k\to\infty} \|s_k\| = 0, \tag{3.5.42}$$

and

$$\lim_{k\to\infty} \|\alpha_k F(x_k)\| = 0. \tag{3.5.43}$$

Proof. Similar to the one obtained for Algorithm 3.2.4.

Lemma 3.5.17 Suppose Assumptions 3.5.1 and 3.5.2 holds. Let $\{x_k\}$ be generated by the Algorithm 3.4.2, then there exists $M > 0$ such that

$$\|d_k^{IHZ}\| \leq M, \quad \forall k. \tag{3.5.44}$$

Proof By condition (3) of Assumption 3.2 we have

$$\begin{aligned} \|y_{k-1}\| &= \|F(x_k) - F(x_{k-1})\| \\ &\leq L\|x_k - x_{k-1}\| \\ &= L\|s_{k-1}\|. \end{aligned} \tag{3.5.45}$$

So,using (3.4.21), (3.5.2) and (3.5.12) we get

$$\begin{aligned}
|\theta_{k1}| &= \left| \xi - \psi \frac{(s_{k-1}^T y_{k-1})^2}{\|y_{k-1}\|^2 \|s_{k-1}\|^2} \right| \\
&\leq |\xi| + \left| \psi \frac{(s_{k-1}^T y_{k-1})^2}{\|y_{k-1}\|^2 \|s_{k-1}\|^2} \right| \\
&\leq \xi + |\psi| \frac{m_2^2 \|s_{k-1}\|^4}{L^2 \|s_{k-1}\|^4} \\
&= \left(\xi + |\psi| \frac{m_2^2}{L^2} \right).
\end{aligned} \tag{3.5.46}$$

Therefore, from (2.3.3), (3.5.12), (3.5.45) and (3.5.46) we obtain

$$\begin{aligned}
\|d_k^{IHZ}\| &= \| - F(x_k) + \beta_k^{IHZ} d_{k-1} \| \\
&= \| - F(x_k) + \frac{F(x_k)^T y_{k-1}}{s_{k-1}^T y_{k-1}} s_{k-1} - \theta_{k1} \frac{\|y_{k-1}\|^2 F(x_k)^T s_{k-1}}{(s_{k-1}^T y_{k-1})^2} s_{k-1} \| \\
&\leq \|F(x_k)\| + \frac{\|F(x_k)\| \|y_{k-1}\|}{s_{k-1}^T y_{k-1}} \|s_{k-1}\| + |\theta_{k1}| \frac{\|y_{k-1}\|^2 \|F(x_k)\| \|s_{k-1}\|}{(s_{k-1}^T y_{k-1})^2} \|s_{k-1}\| \\
&\leq \|F(x_k)\| + \frac{L\|F(x_k)\| \|s_{k-1}\|}{m_2 \|s_{k-1}\|^2} \|s_{k-1}\| + \left(\xi + |\psi| \frac{m_2^2}{L^2} \right) \frac{(L\|s_{k-1}\|)^2 \|F(x_k)\| \|s_{k-1}\|^2}{(m_2 \|s_{k-1}\|^2)^2} \\
&= \|F(x_k)\| + \frac{L\|F(x_k)\|}{m_2} + \left(\xi + |\psi| \frac{m_2^2}{L^2} \right) \frac{L^2 \|F(x_k)\|}{m_2^2} \\
&= \frac{(m_2^2 + m_2 L + \xi L^2 + |\psi| m_2^2) \|F(x_k)\|}{m_2^2} \\
&= \frac{c_3 \|F(x_k)\|}{m_2^2},
\end{aligned} \tag{3.5.47}$$

where $c_3 = (m_2^2 + m_2 L + \xi L^2 + |\psi| m_2^2)$.
Setting $M := \frac{c_3 \|F(x_k)\|}{m_2^2}$ we establish the result.

We use the next theorem to establish global convergence of Algorithm 3.4.2.

Theorem 3.5.18 Suppose Assumptions 3.5.1 and 3.5.2 holds. Let $\{x_k\}$ be generated by the Algorithm 3.4.2. Also, assume that for all $k > 0$

$$\alpha_k \geq c\frac{|F(x_k)^T d_k|}{\|d_k\|^2}, \tag{3.5.48}$$

where c is some positive constant. Then $\{x_k\}$ converges globally to a solution of problem (1.2.1); i.e,

$$\lim_{k\to\infty} \|F(x_k)\| = 0. \tag{3.5.49}$$

Proof. Similar to the one obtained for Algorithm 3.2.4

3.5.4 Convergence Result of Enhanced Hager-Zhang CG method

Lemma 3.5.19 Let $\{x_k\}$ be generated by Algorithm 3.4.6, then d_k is a descent direction for $F(x_k)$ at x_k. i.e

$$F(x)^T d_k < 0. \tag{3.5.50}$$

Proof. By (3.4.51), the Lemma is true and we can deduce that the norm function $f(x_k)$ is a descent along the direction d_k. i.e $\|F(x_{k+1})\| \leq \|F(x_k)\|$ is true $\forall k$.

Lemma 3.5.20 Suppose Assumptions 3.5.1 and 3.5.2 holds. Let $\{x_k\}$ be generated by the Algorithm 3.4.6, then $\{x_k\} \subset \Omega$. Moreover, $\{\|F_k\|\}$ converges.

Proof. Similar to the one obtained for Algorithm 3.2.4

Lemma 3.5.21 Suppose Assumptions 3.5.1 and 3.5.2 holds. Let $\{x_k\}$ be generated by the Algorithm 3.4.6, then

$$\lim_{k\to\infty} \|\alpha_k d_k\| = \lim_{k\to\infty} \|s_k\| = 0, \tag{3.5.51}$$

and

$$\lim_{k\to\infty} \|\alpha_k F(x_k)\| = 0. \tag{3.5.52}$$

Proof. Similar to the one obtained for Algorithm 3.2.4.

Lemma 3.5.22 Suppose Assumptions 3.5.1 and 3.5.2 holds. Let $\{x_k\}$ be generated by the Algorithm 3.4.6, then there exists $\bar{M} > 0$ such that

$$\|d_k^{EHZ}\| \leq \bar{M}, \quad \forall k. \tag{3.5.53}$$

Proof. By utilizing (3.4.28), (3.5.12) and setting $\mu_{k-1} = s_{k-1}$, we have

$$s_{k-1}^T \hat{z}_{k-1} = s_{k-1}^T y_{k-1} + \bar{\rho} \max\{\gamma_{k-1}, 0\} \geq s_{k-1}^T y_{k-1} \geq m_2 \|s_{k-1}\|^2. \tag{3.5.54}$$

And from the mean-value theorem we obtain

$$\begin{aligned} \gamma_{k-1} &= 6(f_k - f_{k+1}) + 3(F_{k-1} + F_k)^T s_{k-1} \\ &= 3(-2\nabla f(\xi) + \nabla f(x_k) + \nabla f(x_{k+1}))^T s_{k-1}, \end{aligned} \tag{3.5.55}$$

where $\xi = \lambda x_k + (1-\lambda) x_{k+1}$, for some $\lambda \in (0,1)$.
Hence from (3.5.2), we have

$$\begin{aligned} |\gamma_{k-1}| &\leq 3(\|\nabla f(x_k) - \nabla f(\xi)\| + \|\nabla f(x_{k+1}) - \nabla f(\xi)\|)\|s_{k-1}\| \\ &\leq 3(L(1-\lambda)\|s_{k-1}\| + L\lambda\|s_{k-1}\|)\|s_{k-1}\| \\ &= 3L\|s_{k-1}\|^2. \end{aligned} \tag{3.5.56}$$

Therefore, from (3.5.2), (3.5.54) and (3.5.56) with $\mu_{k-1} = s_{k-1}$, we obtain

$$\begin{aligned} \|\hat{z}_{k-1}\| &\leq \|y_{k-1}\| + \bar{\rho} \frac{|\gamma_{k-1}|}{s_{k-1}^T s_{k-1}} \|s_{k-1}\| \\ &\leq L\|s_{k-1}\| + \frac{3L\bar{\rho}\|s_{k-1}\|^2}{\|s_{k-1}\|^2} \|s_{k-1}\| \\ &= L\|s_{k-1}\| + 3L\bar{\rho}\|s_{k-1}\| = L\|s_{k-1}\|(1 + 3L\bar{\rho}) = L\|s_{k-1}\|\hat{M}. \end{aligned} \tag{3.5.57}$$

where, $\hat{M} = (1 + 3L\bar{\rho})$.

And by using (3.5.54) and (3.5.57), we get

$$\begin{aligned} |\theta_{k2}| &= \left| p^* - q^* \frac{(s_{k-1}^T \hat{z}_{k-1})^2}{\|\hat{z}_{k-1}\|^2 \|s_{k-1}\|^2} \right| \\ &\leq p^* + q^* \frac{(m_2\|s_{k-1}\|^2)^2}{(L\hat{M}\|s_{k-1}\|)^2 \|s_{k-1}\|^2} \\ &= p^* + q^* \frac{m_2^2}{L^2 \hat{M}^2}. \end{aligned} \tag{3.5.58}$$

Therefore, from (2.3.3), (3.5.12), (3.5.54) and (3.5.58) we obtain

$$\begin{aligned}
\|d_k^{EHZ}\| &= \| - F(x_k) + \beta_k^{EHZ} d_{k-1}\| \\
&= \| - F(x_k) + \frac{F(x_k)^T \hat{z}_{k-1}}{s_{k-1}^T \hat{z}_{k-1}} s_{k-1} - \theta_{k2} \frac{\|\hat{z}_{k-1}\|^2 F(x_k)^T s_{k-1}}{(s_{k-1}^T \hat{z}_{k-1})^2} s_{k-1}\| \\
&\leq \|F(x_k)\| + \frac{\|F(x_k)\|\|\hat{z}_{k-1}\|}{s_{k-1}^T \hat{z}_{k-1}} \|s_{k-1}\| + |\theta_{k2}| \frac{\|\hat{z}_{k-1}\|^2 \|F(x_k)\|\|s_{k-1}\|}{(s_{k-1}^T \hat{z}_{k-1})^2} \|s_{k-1}\| \\
&\leq \|F(x_k)\| + \frac{L\hat{M}\|F(x_k)\|\|s_{k-1}\|^2}{m_2 \|s_{k-1}\|^2} + \left(p^* + |q^*| \frac{m_2^2}{L^2 \hat{M}^2} \right) \frac{L^2 \hat{M}^2 \|s_{k-1}\|^4 \|F(x_k)\|}{(m_2 \|s_{k-1}\|^2)^2} \\
&= \|F(x_k)\| + \frac{L\hat{M}\|F(x_k)\|}{m_2} + \left(p^* + |q^*| \frac{m_2^2}{L^2 \hat{M}^2} \right) \frac{L^2 \hat{M}^2 \|F(x_k)\|}{m_2^2} \\
&= \frac{(m_2^2 + m_2 L\hat{M} + p^* L^2 \hat{M}^2 + |q^*| m_2^2)\|F(x_k)\|}{m_2^2} \\
&= \frac{c_4 \|F(x_k)\|}{m_2^2},
\end{aligned} \tag{3.5.59}$$

where $c_4 = (m_2^2 + m_2 L\hat{M} + p^* L^2 \hat{M}^2 + |q^*| m_2^2)$.
Setting $M := \frac{c_4 \|F(x_k)\|}{m_2^2}$ we establish the result.

We use the next theorem to establish global convergence of Algorithm 3.4.2.

Theorem 3.5.23 Suppose Assumptions 3.5.1 and 3.5.2 holds. Let $\{x_k\}$ be generated by the Algorithm 3.4.6. Also, assume that for all $k > 0$

$$\alpha_k \geq c \frac{|F(x_k)^T d_k|}{\|d_k\|^2}, \tag{3.5.60}$$

where c is some positive constant. Then $\{x_k\}$ converges globally to a solution of problem (1.2.1); i.e,

$$\lim_{k \to \infty} \|F(x_k)\| = 0. \tag{3.5.61}$$

Proof. Similar to the one obtained for Algorithm 3.2.4.

CHAPTER FOUR

NUMERICAL RESULTS

4.1 INTRODUCTION

In this section, we present numerical results to show the efficiency of the proposed methods by comparing their performance with the following existing methods in the literature.

- A family of conjugate gradient methods for large-scale nonlinear equations (**FCGM**) (Sun, Wang, & Feng, 2017) with search directions given by

$$d_k = \begin{cases} -F(x_k), & \text{if } k = 0, \\ -\left(1 - \beta_k \frac{F(x_k)^T d_{k-1}}{\|F(x_k)\|^2}\right) F(x_k) + \beta_k d_{k-1}, & \text{if } k \geq 1, \end{cases} \tag{4.1.1}$$

  where β_k is such that

$$|\beta_k| = t \frac{\|F(x_k)\|}{\|d_{k-1}\|}, \quad \forall k \geq 1, \quad t > 0. \tag{4.1.2}$$

- A three-terms Polak-Ribier*é*-Polyak conjugate gradient algorithm for large-scale nonlinear equations (**TTPRP**) (Yuan & Zhang, 2015) with search directions given by

$$d_k = \begin{cases} -g_k + \frac{g_k^T y_k d_{k-1} - g_k^T d_{k-1} y_k}{max\{\mu \|d_{k-1}\| \|y_k\|, \|g_{k-1}\|^2\}}, & \text{if } k \geq 1, \\ -g_k, & \text{if } k = 0, \end{cases} \tag{4.1.3}$$

where $\mu > 0$ is a constant.

- A modified Hestenes and Stiefel conjugate gradient algorithm for large-scale nonsmooth minimization and nonlinear equations (**MHSCG**) (Yuan, Meng, & Li, 2016) with search directions given by

$$d_k = \begin{cases} -g_k + \frac{g_k^T \delta_k d_{k-1} - d_{k-1}^T g_k \delta_k}{max\{2\gamma \|d_{k-1}\| \|\delta\|, d_{k-1}^T \delta_k\}}, & \text{if } k \geq 1, \\ -g_k, & \text{if } k = 0, \end{cases} \tag{4.1.4}$$

where $\delta_k = g_k - g_{k-1}$ and $\gamma > 0$ is a constant.

We used the same line search procedure for our proposed methods and set the parameters to $\sigma_1 = \sigma_2 = 10^{-4}$, $\alpha_0 = 0.01$, $r = 0.2$ and $\eta_k = \frac{1}{(k+1)^2}$. In addition, we set the other parameters for the methods as follows:

- EDLCG: we set $m = 0.9$, $n = -0.65$, $\phi = 1$ and $C = b = 0.01$.
- ADLCG: we set $m^* = 0.9$, $n^* = -0.6$, and $\zeta = 0.9$.
- IHZCG: we set $\xi = 1.5$ and $\psi = -0.5$.
- EHZCG: we set $p^* = 1.6$, $q^* = -0.65$ and $\rho = 0.01$.
- For the EDLCG, ADLCG and EHZCG, we set $\mu_{k-1} = s_{k-1}$.

For the FCGM, TTPRP, and MHSCG methods, we applied the monotone line search strategies presented in the respective papers and set the parameters as used by the authors. Also, the codes for all the methods were written in Matlab $R2014a$ environment and run on a **PC** (CPU $2.20GHZ$, $8GB$ memory) with windows operating system. Our stopping criteria is $\|F_k\| \leq 10^{-5}$.

Furthermore, using fifteen test problems with different initial points and dimensions, numerical results of experiment carried out with all the seven methods are reported in tables $(4.2-4.17)$. "Guess" and "Dim" indicates the starting point of the iteration and dimension of a problem. "iter" and "Time" stands for total number of iterations and CPU time in seconds respectively. Also, "$\|F_k\|$" stands for the residual at stopping point while "-" indicates failure of a method due to

one of the following:
(i) Insufficient memory
(ii) Number of iterations exceeds 1000 and no point x_k satisfies the stopping criteria.

Also, a summary of the results from tables $(4.2-4.17)$ are presented in tables $(4.18-4.21)$, where the number of problems for which each method outperforms the rest with respect to number of iterations and CPU time are reported. In addition, we use the performance profile of Dolan and Moré (2002) as an evaluation tool to approximately assess the performance and efficiency of each of the methods.

In order to analyze the efficiency of Algorithms, an important tool was presented by Dolan and Moré (2002). They introduced the notion of a performance profile as a means to evaluate and compare the performance of set of solvers S on a test set P. Assuming there exist n_s solvers and n_p problems, for each problem p and solvers s, they defined $t_{p,s}$ =computing time (also, the number of function evaluations or others) required to solve problem p by solvers s.

In order to obtain a baseline for comparisons, they compared the performance on problem p by solver s with the best performance by any solver on this problem; that is, using the performance ratio

$$r_{p,s} = \frac{t_{p,s}}{\min t_{p,s} : s\varepsilon S}. \tag{4.1.5}$$

Then they defined

$$\rho_s(\tau) = \frac{1}{n_p} size p \varepsilon P : r_{p,s} \leq \tau, \tag{4.1.6}$$

thus $\rho_s(t)$ is the probability for solver $s\varepsilon S$ that a performance ratio $r_{p,s}$ is within a factor $\tau \geq 1$ of the best possible ratio. Then function ρ_s is the distribution function for the performance ratio. The performance profile $\rho_s : R \to [0,1]$ for a solver is a nondecreasing, piecewise constant function, continuous from the right at each breakpoint. That is, for subset of the methods being analyzed, the fraction P of the problems for which any given method is within a factor τ of the best is plotted. The value of $\rho_s(1)$ is the probability that the solver win over the rest of the solvers.

According to the above rules, we know that one solver whose performance profile plot stays above the the others will win over the rest of the solvers (Steihuag & Sara, 2013).

4.2 INITIAL POINTS AND TEST FUNCTIONS

In this section, we present initial points and test functions used to test all the seven methods:

Table 4.1: Initial points used for the test problems

INITIAL POINTS (IP)	VALUES
x_1	$(1,1,...,1)^T$
x_2	$(\frac{1}{2},\frac{1}{2^2},\cdots,\frac{1}{2^n})^T$
x_3	$(1,\frac{1}{2},...,\frac{1}{n})^T$
x_4	$(-1,\frac{-3}{2},...,(\frac{1}{n})-2)^T$
x_5	$(\frac{1}{3},\frac{1}{3^2},\cdots,\frac{1}{3^n})^T$
x_6	$(1,\frac{1}{4},\cdots\frac{1}{n^2})^T$
x_7	$(0,-1,...,(\frac{2}{n})-2)^T$
x_8	$(\frac{1}{n},\frac{2}{n},...,1)^T$
x_9	$(1,\frac{2^2}{2^3},\frac{n^2}{n^3})^T$
x_{10}	$(1-\frac{1}{n},1-\frac{2}{n},...,0)^T$

The following are the benchmark problems used to test the proposed methods, where $F(x)=(f_1(x),f_2(x),...,f_n(x))^T$, and $x=(x_1,x_2,...,x_n)^T$.

Problem 1 (Zhou & Li, 2007). The elements of $F(x)$ are given by:

$$F_i(x) = 2x_i - sin|x_i|, \quad i = 1,\ldots,n.$$

Problem 2 (Yuan & Zhang, 2015). The elements of $F(x)$ are given by:

$$F_i(x) = \log(x_i + 1) - \frac{x_i}{n}, \quad i = 2,\ldots,n.$$

Problem 3 (Yuan & Zhang, 2015). The elements of $F(x)$ are given by:

$$F_i(x) = e^{x_i} - 1, \quad i = 1,2,\ldots,n.$$

Problem 4 (Zhou & Shen, 2014). The elements of $F(x)$ are given by:

$$\begin{aligned}
&F_1(x) = x_1(x_1^2 + x_2^2) - 1,\\
&F_i(x) = x_i(x_{i-1}^2 + 2x_i^2 + x_{i+1}^2) - 1, \quad i = 2,3,\ldots,n-1,\\
&F_n(x) = x_n(x_{n-1}^2 + x_n^2).
\end{aligned}$$

Problem 5 (Sun, Wang, & Feng, 2017). The elements of the $F(x)$ are given by:

$$\begin{aligned}
&F_1(x) = x_1 - e^{(\cos\frac{x_1+x_2}{n+1})},\\
&F_i(x) = x_i - e^{(\cos\frac{x_{1-1}+xi+x_{i+1}}{n+1})}, \quad i = 2,3,\ldots,n-1,\\
&F_n(x) = x_n - e^{(\cos\frac{x_{n-1}+x_n}{n+1})}.
\end{aligned}$$

Problem 6 (Liu & Feng, 2017). The elements of the $F(x)$ are given by:

$$\begin{aligned}
&F_1(x) = 2.5x_1 + x_2 - 1,\\
&F_i(x) = x_{i-1} + 2.5x_i + x_{i+1} - 1, \quad i = 2,\ldots,n-1,\\
&F_n(x) = x_{n-1} + 2.5x_n - 1.
\end{aligned}$$

Problem 7 (La Cruz, 2017). The elements of $F(x)$ are given by:

$$
\begin{aligned}
&F_1(x) = 2x_1 + \sin(x_1) - 1, \\
&F_i(x) = -2x_{i-1} + 2x_i + \sin(x_i) - 1, \quad i = 2, \ldots, n-1, \\
&F_n(x) = 2x_n + \sin(x_n) - 1.
\end{aligned}
$$

Problem 8 (Zhou & Shen, 2014). The function $F(x)$ is given by

$$F(x) = A_1 x + b_1,$$

where $b_1 = (e_1^x - 1, \ldots, e_n^x - 1)^T$, and

$$
A_1 = \begin{pmatrix}
2 & -1 & & & \\
-1 & 2 & -1 & & \\
 & \ddots & \ddots & \ddots & \\
 & & \ddots & \ddots & -1 \\
 & & & -1 & 2
\end{pmatrix}.
$$

Problem 9 (Zhou & Shen, 2014). The elements of $F(x)$ are given by:

$$F(x) = A_2 x + b_2,$$

where $b_2 = (sinx_1 - 1, \ldots, sinx_n - 1)^T$, and

$$
A_2 = \begin{pmatrix}
2 & -1 & & & \\
0 & 2 & -1 & & \\
 & \ddots & \ddots & \ddots & \\
 & & \ddots & \ddots & -1 \\
 & & & 0 & 2
\end{pmatrix}.
$$

Problem 10. (Yuan & Zhang, 2015) The elements of $F(x)$ are given by:

$$F_i(x) = 2\left(n + i(1-\cos x_i) - \sin x_i - \sum_{j=1}^{n} \cos x_j\right)(2\sin x_i - \cos x_i), \quad i = 1,2,\ldots,n.$$

Problem 11 (La Cruz, 2017). The function $F(x)$ is given by:

$$F(x) = A_3 x + b_3,$$

where $b_3 = (-1,-1,-1,\ldots,-1)^T$, and

$$A = \begin{pmatrix} 5/2 & 1 & & & \\ 1 & 5/2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & 5/2 \end{pmatrix}.$$

Problem 12 (Koorapetse, Kaelo, & Offen, 2018). The elements of $F(x)$ are given by:

$$F_i(x) = x_i - \tfrac{1}{n}x_i^2 + \tfrac{1}{n}\sum_{i=1}^{n} x_i + i, \quad i = 1,2,\ldots,n..$$

Problem 13 (Yuan, 2017). The elements of $F(x)$ are given by:

$$F_i(x) = \sqrt{10^{-5}}(x_i - 1),$$
$$F_n(x) = \tfrac{1}{4n}\sum_{j=1}^{n} x_j^2 - \tfrac{1}{4}, \quad i = 2,3,\ldots,n-1.$$

Problem 14 Artificial problem

$$F_1(x) = 2x_1 - x_2 + uh^2 \log(\cosh(x_1 - 1) - 1), \quad i = 2,\ldots,n-1,$$
$$F_i(x) = 2x_i - x_{i-1} + uh^2 \log(\cosh(x_i - 1) - 1),$$
$$F_n(x) = 2x_n - x_{n-1} + uh \log(\cosh(x_n - 1) - 1), \quad u = 10, h = \tfrac{1}{(n+1)}.$$

Problem 15 (Zhang & Zhou, 2006). The elements of $F(x)$ are given by:

$$F_i(x) = 2x_i - sin|x_i - 1|, \quad i = 1, \ldots, n.$$

4.3 COMPUTATIONAL EXPERIMENTS

In this section, numerical results are presented as follows:

Table 4.2: Test results of EDLCG, FCGM, TTPRP, and MHSCG methods for problems 1-4

			EDLCG			**FCGM**			**TTPRP**			**MHSCG**	
Problem	*Guess*	*Dim*	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$
1	x_1	10000	7	0.057296	5.70E-06	23	0.155001	9.65E-06	153	0.575608	9.14E-06	73	0.305228	1.23E-09
	$0.1x_1$	10000	5	0.054524	1.20E-06	20	0.085559	9.52E-06	132	0.505874	9.11E-06	52	0.236625	1.58E-09
	$0.01x_1$	10000	2	0.049391	3.50E-06	17	0.080258	7.63E-06	109	0.407292	0.00E+00	30	0.152127	2.28E-11
	$0.05x_1$	10000	4	0.094831	2.14E-06	19	0.128151	9.53E-06	125	0.591275	9.53E-06	51	0.279019	3.81E-09
	x_2	100000	12	1.102591	8.14E-06	16	1.224823	8.43E-06	104	5.637419	9.87E-06	_	_	_
	$0.1x_2$	100000	2	0.680844	8.80E-06	13	1.187925	7.04E-06	83	4.718356	9.19E-06	_	_	_
	$0.01x_2$	100000	1	0.738293	6.30E-08	10	0.980073	5.64E-06	61	3.573554	9.34E-06	_	_	_
	$0.05x_1$	100000	4	0.825859	6.78E-06	21	1.237148	7.54E-06	119	9.653821	3.93E-19	46	4.731775	2.57E-09
2	$0.1x_3$	10000	9	0.085746	9.32E-06	15	0.093266	8.61E-06	38	0.206777	6.75E-06	_	_	_
	$0.01x_3$	10000	2	0.059308	7.34E-06	10	0.082149	7.87E-06	12	0.104852	8.55E-06	_	_	_
	$0.05x_3$	10000	7	0.068875	7.88E-06	12	0.086349	9.97E-06	39	0.210418	8.36E-06	_	_	_
	x_3	10000	17	0.110759	5.93E-06	19	0.152132	7.96E-06	44	0.462105	3.01E-06	_	_	_
	$-0.1x_3$	100000	9	0.829688	9.73E-06	13	0.962553	9.50E-06	71	4.482847	3.03E-10	_	_	_
	$-0.01x_3$	100000	2	0.858736	6.89E-06	10	0.862709	7.82E-06	49	3.221027	3.24E-10	_	_	_
	x_3	100000	17	0.956671	5.95E-06	19	1.105058	7.95E-06	65	5.517453	3.60E-08	_	_	_
	$-0.05x_3$	100000	7	1.185058	7.95E-06	12	1.319637	9.65E-06	65	4.153584	2.77E-10	_	_	_
3	$0.1x_2$	10000	10	0.060109	5.60E-06	13	0.056946	6.75E-06	82	0.255208	9.98E-06	_	_	_
	$0.01x_2$	10000	2	0.048006	6.81E-06	10	0.070682	5.61E-06	61	0.201881	9.32E-06	_	_	_
	x_2	10000	17	0.063857	8.31E-06	16	0.116124	5.83E-06	102	0.396138	9.90E-06	_	_	_
	$0.05x_2$	10000	7	0.057868	8.36E-06	12	0.066723	6.90E-06	76	0.242659	9.50E-06	_	_	_
	x_4	100000	27	1.124108	7.80E-06	32	1.123662	5.56E-06	139	5.909545	9.86E-06	_	_	_
	$0.1x_4$	100000	24	1.201097	6.21E-06	21	0.925821	5.76E-06	109	4.501971	9.44E-06	_	_	_
	$0.01x_4$	100000	16	1.328544	8.01E-06	14	1.263989	8.06E-06	44	3.483048	8.31E-06	_	_	_
	$0.5x_4$	100000	28	1.047797	6.62E-06	33	1.264114	5.83E-06	125	5.219315	9.66E-06	_	_	_
4	0.1_1	10000	34	0.139711	8.18E-06	231	0.794097	9.93E-06	158	0.710892	9.45E-06	_	_	_
	$-0.1x_1$	10000	32	0.133743	8.22E-06	349	1.167591	9.89E-06	162	0.744559	8.62E-06	_	_	_
	$0.01x_1$	10000	30	0.129775	5.99E-06	155	0.559408	9.82E-06	113	0.522713	1.00E-05	_	_	_
	$-0.05x_1$	10000	35	0.143313	9.95E-06	168	0.597262	9.86E-06	145	0.666452	9.46E-06	_	_	_
	$0.5x_1$	100000	40	2.545517	6.75E-06	321	19.165534	9.73E-06	193	14.969885	7.48E-06	_	_	_
	$-0.01x_1$	100000	30	1.946841	8.94E-06	133	8.471215	9.91E-06	112	8.899216	9.90E-06	-	_	_
	$-0.5x_1$	100000	37	3.445715	8.94E-06	473	39.063297	9.70E-06	200	20.735541	9.79E-06	_	_	_
	$-0.05x_1$	100000	36	2.236615	8.08E-06	152	9.751605	9.89E-06	152	11.896874	9.16E-06	_	_	_

Table 4.3: Test results of EDLCG, FCGM, TTPRP, and MHSCG methods for problems 5-8

				EDLCG			**FCGM**			**TTPRP**			**MHSCG**	
Problem	*Guess*	*Dim*	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$
5	-x_5	10000	5	0.064471	8.27E-06	25	0.138438	8.10E-06	163	0.983816	9.46E-06	_	_	_
	0.1x_5	10000	5	0.066554	8.27E-06	25	0.130675	8.10E-06	163	0.987082	9.46E-06	_	_	_
	0.5x_5	10000	5	0.095697	8.27E-06	25	0.184446	8.10E-06	163	0.978568	9.46E-06	_	_	_
	x_6	10000	5	0.069838	8.20E-06	25	0.125153	8.85E-06	164	0.983153	9.30E-06	_	_	_
	-x_6	100000	1	0.518247	2.83E-06	27	1.822457	5.82E-06	173	15.748324	9.47E-06	_	_	_
	x_2	100000	1	0.507574	2.86E-06	27	1.963562	6.40E-06	174	15.826656	9.39E-06	_	_	_
	0.1x_2	100000	1	0.774194	2.86E-06	27	2.690999	6.40E-06	174	21.781523	9.39E-06	_	_	_
	-x_2	100000	1	0.544467	2.86E-06	27	1.806925	6.40E-06	174	15.689845	9.39E-06	_	_	_
6	x_2	10000	30	0.115383	8.51E-06	144	0.436209	9.33E-06	126	0.478785	9.17E-06	_	_	_
	-x_2	10000	32	0.121606	9.94E-06	154	0.460833	8.94E-06	142	0.538366	9.52E-06	_	_	_
	0.1x_7	10000	31	0.125082	7.52E-06	151	0.452602	9.72E-06	133	0.497542	8.50E-06	_	_	_
	-0.1x_7	10000	34	0.124744	6.82E-06	152	0.459534	9.50E-06	145	0.557342	9.90E-06	_	_	_
	x_2	100000	33	1.898667	7.06E-06	149	7.921229	9.35E-06	129	8.734475	9.64E-06	_	_	_
	-x_2	100000	30	1.751574	7.98E-06	151	8.147204	9.80E-06	141	9.351653	6.60E-06	_	_	_
	0.1x_7	100000	31	1.905686	7.35E-06	141	7.669758	9.49E-06	161	11.133489	7.53E-06	_	_	_
	-0.1x_7	100000	29	1.655015	6.96E-06	130	7.084815	8.06E-06	152	10.629789	9.88E-06	_	_	_
7	x_8	10000	6	0.001976	5.51E-06	10	0.002193	3.93E-06	36	0.004657	7.43E-06	36	0.005741	7.42E-06
	-1x_8	10000	8	0.001779	2.84E-06	10	0.001436	6.49E-06	37	0.005685	9.55E-06	37	0.005907	9.56E-06
	0.1x_8	10000	6	0.002803	1.34E-06	9	0.001893	4.14E-06	32	0.003777	9.71E-06	32	0.004922	9.71E-06
	-0.01x_8	10000	6	0.001789	1.92E-06	9	0.001923	5.99E-06	33	0.005483	9.99E-06	9	0.001637	5.99E-06
	-0.1x_8	100000	6	0.001339	2.37E-06	9	0.001695	7.50E-06	34	0.005071	8.85E-06	34	0.005368	8.85E-06
	0.5x_8	100000	6	0.001248	1.07E-06	9	0.003441	3.13E-06	31	0.005915	9.92E-06	31	0.005251	9.91E-06
	-0.5x_8	100000	6	0.001363	4.32E-06	10	0.001581	3.80E-06	36	0.004421	8.39E-06	36	0.008238	8.39E-06
	0.01x_8	100000	6	0.001213	1.82E-06	9	0.001738	5.65E-06	33	0.004494	9.42E-06	33	0.007239	9.42E-06
8	x_9	10000	2	0.008293	9.80E-06	3	0.010773	5.47E-06	10	0.037453	9.89E-06	10	0.040602	9.89E-06
	-x_9	10000	2	0.008253	9.80E-06	3	0.011277	5.47E-06	10	0.037975	9.89E-06	10	0.035228	9.88E-06
	x_8	10000	5	0.016973	6.91E-06	9	0.035571	3.07E-06	35	0.116387	8.68E-06	35	0.170489	8.68E-06
	0.1x_9	10000	2	0.006568	9.80E-07	1	0.006371	8.75E-06	4	0.016425	8.40E-06	4	0.018843	8.41E-06
	-0.1x_9	10000	2	0.007891	9.80E-07	1	0.006554	8.75E-06	4	0.018193	8.40E-06	4	0.016013	8.41E-06
	x_9	100000	2	0.005624	9.80E-07	1	0.003644	8.75E-06	4	0.015402	8.40E-06	4	0.014701	8.41E-06
	-0.1x_*	100000	5	0.017216	3.24E-06	8	0.024697	4.88E-06	29	0.097154	9.89E-06	29	0.127271	9.89E-06
	-x_9	100000	2	0.005204	9.80E-07	1	0.003854	8.75E-06	4	0.015057	8.40E-06	4	0.015639	8.40E-06

Table 4.4: Test results of EDLCG, FCGM, TTPRP, and MHSCG methods for problems 9-12

			EDLCG			**FCGM**			**TTPRP**			**MHSCG**	
Problem	*Guess*	*Dim*	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$
9	x_{10}	10000	6	0.018057	1.87E-06	9	0.025774	5.82E-06	33	0.104227	9.71E-06	33	0.107314	9.71E-06
	$0.1x_{10}$	10000	6	0.014962	1.87E-06	9	0.025674	5.82E-06	33	0.102471	9.71E-06	33	0.155634	9.71E-06
	$0.01x_{10}$	10000	6	0.015201	1.87E-06	9	0.025427	5.82E-06	33	0.102853	9.71E-06	33	0.141926	9.71E-06
	$0.05x_{10}$	10000	6	0.014906	1.87E-06	9	0.025229	5.82E-06	33	0.104138	9.71E-06	33	0.137566	9.71E-06
	x_8	100000	6	0.018558	5.51E-06	10	0.030186	3.93E-06	36	0.070646	7.43E-06	36	0.151436	7.42E-06
	$0.1x_8$	100000	6	0.015239	1.34E-06	9	0.025952	4.14E-06	32	0.102319	9.71E-06	32	0.133125	9.71E-06
	$0.01x_8$	100000	6	0.017519	1.82E-06	9	0.024548	5.65E-06	33	0.105205	9.42E-06	33	0.106442	9.42E-06
	$0.05x_8$	100000	6	0.015591	1.61E-06	9	0.026384	4.98E-06	33	0.546842	8.28E-06	33	0.147553	8.28E-06
10	x_8	10000	13	0.003208	4.60E-06	14	0.002979	5.83E-06	42	0.010205	9.62E-06	42	0.008833	9.61E-06
	$0.1x_8$	10000	11	0.002585	3.83E-06	5	0.0075	8.72E-06	47	0.011397	8.56E-06	47	0.010453	8.55E-06
	$0.01x_8$	10000	9	0.002485	7.59E-06	7	0.002193	9.35E-06	35	0.006223	8.43E-06	35	0.007927	8.42E-06
	$0.05x_8$	10000	10	0.002643	8.82E-06	8	0.001688	5.02E-06	43	0.013716	8.31E-06	43	0.008163	8.32E-06
	x_8	100000	13	0.003208	4.60E-06	14	0.003483	5.83E-06	42	0.006612	9.62E-06	42	0.009589	9.61E-06
	$0.1x_8$	100000	11	0.002561	3.83E-06	5	0.001439	8.72E-06	47	0.011248	8.56E-06	47	0.009827	8.55E-06
	$0.01x_8$	100000	9	0.002373	7.59E-06	7	0.001982	9.35E-06	35	0.009566	8.43E-06	35	0.011069	8.42E-06
	$0.05x_8$	100000	10	0.003502	8.82E-06	8	0.002381	5.02E-06	43	0.007416	8.31E-06	43	0.009422	8.32E-06
11	x_8	10000	10	0.020861	2.59E-06	14	0.038989	3.81E-06	45	0.145166	8.35E-06	45	0.137219	8.35E-06
	$0.1x_8$	10000	9	0.022582	4.11E-06	12	0.032704	9.67E-06	41	0.126207	9.43E-06	41	0.131898	9.43E-06
	$0.01x_8$	10000	9	0.024018	3.37E-06	12	0.033526	7.93E-06	41	0.126085	7.73E-06	41	0.127325	7.73E-06
	$0.05x_8$	10000	9	0.022225	3.69E-06	12	0.034162	8.70E-06	41	0.124181	8.49E-06	41	0.134344	8.48E-06
	x_8	100000	10	0.023377	2.59E-06	14	0.039831	3.81E-06	45	0.142512	8.35E-06	45	0.140629	8.35E-06
	$0.1x_8$	100000	9	0.023674	4.11E-06	12	0.032424	9.67E-06	41	0.127833	9.43E-06	41	0.131296	9.43E-06
	$0.01x_8$	100000	9	0.023244	3.37E-06	12	0.033165	7.93E-06	41	0.127934	7.73E-06	41	0.128443	7.73E-06
	$0.05x_8$	100000	9	0.022466	3.69E-06	12	0.032843	8.70E-06	41	0.122881	8.49E-06	41	0.153393	8.48E-06
12	x_{10}	10000	7	0.001378	3.65E-06	10	0.002139	8.97E-06	36	0.005094	9.25E-06	36	0.007093	9.25E-06
	$0.1x_{10}$	10000	7	0.002141	3.65E-06	10	0.001938	8.97E-06	36	0.005603	9.25E-06	36	0.009112	9.25E-06
	$0.01x_{10}$	10000	7	0.001253	3.65E-06	10	0.001977	8.97E-67	36	0.004819	9.25E-06	36	0.005888	9.25E-06
	$0.05x_{10}$	10000	7	0.001707	3.65E-06	10	0.001808	8.97E-06	36	0.004557	9.25E-06	36	0.008036	9.25E-06
	x_{10}	100000	7	0.001766	3.65E-06	10	0.002187	8.97E-06	36	0.007275	9.25E-06	36	0.005895	9.25E-06
	$0.1x_{10}$	100000	7	0.002107	3.65E-06	10	0.002243	8.97E-06	36	0.004927	9.25E-06	36	0.006378	9.25E-06
	$0.01x_{10}$	100000	7	0.001698	3.65E-06	10	0.002473	8.97E-06	36	0.005587	9.25E-06	36	0.006688	9.25E-06
	$0.05x_{10}$	100000	7	0.001818	3.65E-06	10	0.001901	8.97E-06	36	0.005117	9.25E-06	36	0.006275	9.25E-06

Table 4.5: Test results of EDLCG, FCGM, TTPRP, and MHSCG methods for problems 13-15

			EDLCG			**FCGM**			**TTPRP**			**MHSCG**		
Problem	*Guess*	*Dim*	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$
13	x_{10}	10000	22	0.003384	9.21E-06	36	0.003728	9.90E-06	36	0.004084	9.90E-06	36	0.004517	9.90E-06
	$0.1x_{10}$	10000	22	0.004185	9.21E-06	36	0.004395	9.90E-06	36	0.004228	9.90E-06	36	0.005335	9.90E-06
	x_8	10000	19	0.003724	7.29E-06	31	0.003449	9.68E-06	31	0.003684	9.68E-06	31	0.004298	9.68E-06
	$0.1x_8$	10000	21	0.004026	9.11E-06	35	0.004978	8.32E-06	35	0.003314	8.32E-06	35	0.006332	8.31E-06
	$0.01x_8$	100000	22	0.005239	8.76E-06	36	0.005642	9.46E-06	36	0.003575	9.46E-06	36	0.006559	9.46E-06
	$0.05x_8$	100000	22	0.003604	7.17E-06	36	0.003349	7.87E-06	36	0.004927	7.87E-06	36	0.004538	7.86E-06
	$-0.01x_8$	100000	22	0.003484	9.69E-06	37	0.004936	7.78E-06	37	0.004626	7.78E-06	37	0.005741	7.78E-06
	$-0.05x_8$	100000	23	0.003771	7.29E-06	37	0.003651	9.38E-06	37	0.003634	9.38E-06	37	0.004805	9.38E-06
14	$0.1x_1$	10000	7	0.067167	5.81E-06	26	0.132404	9.95E-06	152	0.898099	9.98E-06	_	_	_
	0.01_1	10000	7	0.064594	6.88E-06	27	0.132621	7.01E-06	153	0.920906	9.88E-06	_	_	_
	$0.5x_1$	10000	6	0.065032	5.03E-06	25	0.126765	7.62E-06	147	0.875104	9.39E-06	_	_	_
	$0.05x_1$	10000	7	0.066873	6.39E-06	27	0.134719	6.67E-06	153	0.912578	9.48E-06	_	_	_
	x_3	100000	59	3.518931	8.72E-06	54	3.003468	9.06E-06	164	13.754637	9.90E-06	_	_	_
	$0.1x_3$	100000	44	2.593925	8.71E-06	63	3.425866	9.34E-06	164	13.393925	9.90E-06	_	_	_
	$0.01x_3$	100000	29	1.894083	9.32E-06	47	2.705293	8.99E-06	164	13.376023	9.90E-06	_	_	_
	$0.05x_3$	100000	39	2.323923	9.49E-06	51	2.804204	8.51E-06	164	13.375545	9.90E-06	_	_	_
15	x_3	10000	19	0.090222	9.77E-06	33	0.121664	4.68E-06	78	0.329665	9.54E-06	_	_	_
	$0.1x_3$	10000	19	0.088137	9.78E-06	33	0.124292	7.98E-06	78	0.327049	9.55E-06	_	_	_
	$0.01x_3$	10000	19	0.087578	9.78E-06	40	0.144258	6.26E-06	78	0.326403	9.55E-06	_	_	_
	$0.05x_3$	10000	19	0.086951	9.78E-06	45	0.147036	8.25E-06	78	0.322093	9.55E-06	_	_	_
	x_3	100000	21	1.149274	5.45E-06	46	2.042894	4.49E-06	84	4.867944	8.71E-06	_	_	_
	$0.1x$	100000	21	1.251304	5.45E-06	43	1.886833	9.46E-06	84	4.875971	8.71E-06	_	_	_
	$0.01x_3$	100000	21	1.260426	5.45E-06	44	1.906549	5.45E-06	84	4.801281	8.71E-06	_	_	_
	$0.05x_3$	100000	21	1.234978	5.45E-06	45	1.984194	6.89E-06	84	4.839611	8.71E-06	_	_	_

Table 4.6: Test results of ADLCG, FCGM, TTPRP, and MHSCG methods for problems 1-4

			ADLCG			**FCGM**			**TTPRP**			**MHSCG**	
Problem	*Guess*	*Dim*	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$
1	x_1	10000	8	0.070879	4.82E-06	23	0.155001	9.65E-06	153	0.575608	9.14E-06	73	0.305228	1.23E-09
	$0.1x_1$	10000	5	0.062819	5.00E-06	20	0.085559	9.52E-06	132	0.505874	9.11E-06	52	0.236625	1.58E-09
	$0.01x_1$	10000	2	0.063231	5.00E-06	17	0.080258	7.63E-06	109	0.407292	0.00E+00	30	0.152127	2.28E-11
	$0.05x_1$	10000	4	0.084953	6.25E-06	19	0.128151	9.53E-06	125	0.591275	9.53E-06	51	0.279019	3.81E-09
	x_2	100000	27	1.821799	9.13E-06	16	1.224823	8.43E-06	104	5.637419	9.87E-06	_	_	_
	$0.1x_2$	100000	8	1.051748	8.16E-06	13	1.187925	7.04E-06	83	4.718356	9.19E-06	_	_	_
	$0.01x_2$	100000	1	0.743597	6.30E-08	10	0.980073	5.64E-06	61	3.573554	9.34E-06	_	_	_
	$0.05x_1$	100000	17	1.603096	5.83E-06	21	1.237148	7.54E-06	119	9.653821	3.93E-19	46	4.731775	2.57E-09
2	$0.1x_3$	10000	8	0.073494	8.61E-06	15	0.093266	8.61E-06	38	0.206777	6.75E-06	_	_	_
	$0.01x_3$	10000	2	0.067885	6.68E-06	10	0.082149	7.87E-06	12	0.104852	8.55E-06	_	_	_
	$0.05x_3$	10000	6	0.081394	8.82E-06	12	0.086349	9.97E-06	39	0.210418	8.36E-06	_	_	_
	x_3	10000	15	0.117528	5.08E-06	19	0.152132	7.96E-06	44	0.462105	3.01E-06	_	_	_
	$-0.1x_3$	100000	8	1.141013	8.96E-06	13	0.962553	9.50E-06	71	4.482847	3.03E-10	_	_	_
	$-0.01x_3$	100000	2	0.746516	6.26E-06	10	0.862709	7.82E-06	49	3.221027	3.24E-10	_	_	_
	x_3	100000	15	1.229913	5.09E-06	19	1.105058	7.95E-06	65	5.517453	3.60E-08	_	_	_
	$-0.05x_3$	100000	6	0.621588	8.89E-06	12	1.319637	9.65E-06	65	4.153584	2.77E-10	_	_	_
3	$0.1x_2$	10000	8	0.067164	9.46E-06	13	0.056946	6.75E-06	82	0.255208	9.98E-06	_	_	_
	$0.01x_2$	10000	2	0.053219	6.19E-06	10	0.070682	5.61E-06	61	0.201881	9.32E-06	_	_	_
	x_2	10000	15	0.076564	7.07E-06	16	0.116124	5.83E-06	102	0.396138	9.90E-06	_	_	_
	$0.05x_2$	10000	6	0.060167	9.41E-06	12	0.066723	6.90E-06	76	0.242659	9.50E-06	_	_	_
	x_4	100000	24	1.571869	7.68E-06	32	1.123662	5.56E-06	139	5.909545	9.86E-06	_	_	_
	$0.1x_4$	100000	21	1.336116	5.54E-06	21	0.925821	5.76E-06	109	4.501971	9.44E-06	_	_	_
	$0.01x_4$	100000	14	0.004756	7.67E-06	14	1.263989	8.06E-06	44	3.483048	8.31E-06	_	_	_
	$0.5x_4$	100000	24	1.189168	7.64E-06	33	1.264114	5.83E-06	125	5.219315	9.66E-06	_	_	_
4	$0.1x_1$	10000	31	0.205323	7.59E-06	231	0.794097	9.93E-06	158	0.710892	9.45E-06	_	_	_
	$-0.1x_1$	10000	30	0.148997	8.85E-06	349	1.167591	9.89E-06	162	0.744559	8.62E-06	_	_	_
	$0.01x_1$	10000	29	0.213344	9.45E-06	155	0.559408	9.82E-06	113	0.522713	1.00E-05	_	_	_
	$-0.05x_1$	10000	29	0.130482	6.40E-06	168	0.597262	9.86E-06	145	0.666452	9.46E-06	_	_	_
	$0.5x_1$	100000	40	4.195958	9.35E-06	321	19.165534	9.73E-06	193	14.969885	7.48E-06	_	_	_
	$-0.01x_1$	100000	28	3.256146	9.74E-06	133	8.471215	9.91E-06	112	8.899216	9.90E-06	-	_	_
	$-0.5x_1$	100000	40	3.965866	9.20E-06	473	39.063297	9.70E-06	200	20.735541	9.79E-06	_	_	_
	-0.05e	100000	30	2.505961	7.07E-06	152	9.751605	9.89E-06	152	11.896874	9.16E-06	_	_	_

Table 4.7: Test results of ADLCG, FCGM, TTPRP, and MHSCG methods for problems 5-8

				ADLCG			**FCGM**			**TTPRP**			**MHSCG**	
Problem	*Guess*	*Dim*	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$
5	-x_5	10000	5	0.067219	5.65E-06	25	0.138438	8.10E-06	163	0.983816	9.46E-06	_	_	_
	0.1x_5	10000	5	0.068908	5.65E-06	25	0.130675	8.10E-06	163	0.987082	9.46E-06	_	_	_
	0.5x_5	10000	5	0.076913	5.65E-06	25	0.184446	8.10E-06	163	0.978568	9.46E-06	_	_	_
	x_6	10000	5	0.057038	5.60E-06	25	0.125153	8.85E-06	164	0.983153	9.30E-06	_	_	_
	-x_6	100000	1	0.701405	2.83E-06	27	1.822457	5.82E-06	173	15.748324	9.47E-06	_	_	_
	x_2	100000	1	0.801959	2.86E-06	27	1.963562	6.40E-06	174	15.826656	9.39E-06	_	_	_
	0.1x_2	100000	1	0.863786	2.86E-06	27	2.690999	6.40E-06	174	21.781523	9.39E-06	_	_	_
	-x_2	100000	1	0.896437	2.86E-06	27	1.806925	6.40E-06	174	15.689845	9.39E-06	_	_	_
6	x_2	10000	28	0.158669	6.41E-06	144	0.436209	9.33E-06	126	0.478785	9.17E-06	_	_	_
	-x_2	10000	29	0.161041	7.24E-06	154	0.460833	8.94E-06	142	0.538366	9.52E-06	_	_	_
	0.1x_7	10000	31	0.133581	9.70E-06	151	0.452602	9.72E-06	133	0.497542	8.50E-06	_	_	_
	-0.1x_7	10000	29	0.168594	8.70E-06	152	0.459534	9.50E-06	145	0.557342	9.90E-06	_	_	_
	x_2	100000	32	2.444201	6.95E-06	149	7.921229	9.35E-06	129	8.734475	9.64E-06	_	_	_
	-x_2	100000	32	2.193178	9.86E-06	151	8.147204	9.80E-06	141	9.351653	6.60E-06	_	_	_
	0.1x_7	100000	30	2.240516	9.19E-06	141	7.669758	9.49E-06	161	11.133489	7.53E-06	_	_	_
	-0.1x_7	100000	31	2.978982	5.81E-06	130	7.084815	8.06E-06	152	10.629789	9.88E-06	_	_	_
7	x_8	10000	7	0.001228	2.61E-06	10	0.002193	3.93E-06	36	0.004657	7.43E-06	36	0.005741	7.42E-06
	-1x_8	10000	9	0.001792	1.76E-06	10	0.001436	6.49E-06	37	0.005685	9.55E-06	37	0.005907	9.56E-06
	0.1x_8	10000	6	0.001873	5.83E-06	9	0.001893	4.14E-06	32	0.003777	9.71E-06	32	0.004922	9.71E-06
	-0.01x_8	10000	6	0.001873	8.38E-06	9	0.001923	5.99E-06	33	0.005483	9.99E-06	9	0.001637	5.99E-06
	-0.1x_8	100000	7	0.001496	1.21E-06	9	0.001695	7.50E-06	34	0.005071	8.85E-06	34	0.005368	8.85E-06
	0.5x_8	100000	6	0.001037	4.54E-06	9	0.003441	3.13E-06	31	0.005915	9.92E-06	31	0.005251	9.91E-06
	-0.5x_8	100000	7	0.001365	2.25E-06	10	0.001581	3.80E-06	36	0.004421	8.39E-06	36	0.008238	8.39E-06
	0.01x_8	100000	6	0.001245	7.92E-06	9	0.001738	5.65E-06	33	0.004494	9.42E-06	33	0.007239	9.42E-06
8	x_9	10000	3	0.008414	1.40E-06	3	0.010773	5.47E-06	10	0.037453	9.89E-06	10	0.040602	9.89E-06
	-x_9	10000	3	0.010157	1.40E-06	3	0.011277	5.47E-06	10	0.037975	9.89E-06	10	0.035228	9.88E-06
	x_8	10000	6	0.016414	4.35E-06	9	0.035571	3.07E-06	35	0.116387	8.68E-06	35	0.170489	8.68E-06
	0.1x_9	10000	2	0.007711	1.40E-06	1	0.006371	8.75E-06	4	0.016425	8.40E-06	4	0.018843	8.41E-06
	-0.1x_9	10000	2	0.007041	1.40E-06	1	0.006554	8.75E-06	4	0.018193	8.40E-06	4	0.016013	8.41E-06
	x_9	100000	2	0.005427	1.40E-06	1	0.003644	8.75E-06	4	0.015402	8.40E-06	4	0.014701	8.41E-06
	-0.1x_8	100000	6	0.016555	1.31303-06	8	0.024697	4.88E-06	29	0.097154	9.89E-06	29	0.127271	9.89E-06
	-x_9	100000	2	0.005521	1.40E-06	1	0.003854	8.75E-06	4	0.015057	8.40E-06	4	0.015639	8.40E-06

Table 4.8: Test results of ADLCG, FCGM, TTPRP, and MHSCG methods for problems 9-12

			ADLCG			**FCGM**			**TTPRP**			**MHSCG**	
Problem	*Guess*	*Dim*	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$
9	x_{10}	10000	6	0.017571	8.15E-06	9	0.025774	5.82E-06	33	0.104227	9.71E-06	33	0.107314	9.71E-06
	$0.1x_{10}$	10000	6	0.018939	8.15E-06	9	0.025674	5.82E-06	33	0.102471	9.71E-06	33	0.155634	9.71E-06
	$0.01x_{10}$	10000	6	0.016854	8.15E-06	9	0.025427	5.82E-06	33	0.102853	9.71E-06	33	0.141926	9.71E-06
	$0.05x_{10}$	10000	6	0.018519	8.15E-06	9	0.025229	5.82E-06	33	0.104138	9.71E-06	33	0.137566	9.71E-06
	x_8	100000	7	0.020425	2.61E-06	10	0.030186	3.93E-06	36	0.070646	7.43E-06	36	0.151436	7.42E-06
	$0.1x_8$	100000	6	0.016018	5.83E-06	9	0.025952	4.14E-06	32	0.102319	9.71E-06	32	0.133125	9.71E-06
	$0.01x_8$	100000	6	0.016156	7.92E-06	9	0.024548	5.65E-06	33	0.105205	9.42E-06	33	0.106442	9.42E-06
	$0.05x_8$	100000	6	0.018836	7.00E-06	9	0.026384	4.98E-06	33	0.546842	8.28E-06	33	0.147553	8.28E-06
10	x_8	10000	13	0.004195	8.21E-06	14	0.002979	5.83E-06	42	0.010205	9.62E-06	42	0.008833	9.61E-06
	$0.1x_8$	10000	11	0.002902	6.52E-06	5	0.0075	8.72E-06	47	0.011397	8.56E-06	47	0.010453	8.55E-06
	$0.01x_8$	10000	10	0.002482	4.59E-06	7	0.002193	9.35E-06	35	0.006223	8.43E-06	35	0.007927	8.42E-06
	$0.05x_8$	10000	11	0.003098	5.69E-06	8	0.001688	5.02E-06	43	0.013716	8.31E-06	43	0.008163	8.32E-06
	x_8	100000	13	0.002953	8.21E-06	14	0.003483	5.83E-06	42	0.006612	9.62E-06	42	0.009589	9.61E-06
	$0.1x_8$	100000	11	0.002986	6.52E-06	5	0.001439	8.72E-06	47	0.011248	8.56E-06	47	0.009827	8.55E-06
	$0.01x_8$	100000	10	0.002784	4.59E-06	7	0.001982	9.35E-06	35	0.009566	8.43E-06	35	0.011069	8.42E-06
	$0.05x_8$	100000	11	0.002456	5.69E-06	8	0.002381	5.02E-06	43	0.007416	8.31E-06	43	0.009422	8.32E-06
11	x_4	10000	10	0.030219	6.68E-06	14	0.038989	3.81E-06	45	0.145166	8.35E-06	45	0.137219	8.35E-06
	$0.1x_8$	10000	9	0.022993	9.54E-06	12	0.032704	9.67E-06	41	0.126207	9.43E-06	41	0.131898	9.43E-06
	$0.01x_8$	10000	9	0.022797	7.82E-06	12	0.033526	7.93E-06	41	0.126085	7.73E-06	41	0.127325	7.73E-06
	$0.05x_8$	10000	9	0.023082	8.58E-06	12	0.034162	8.70E-06	41	0.124181	8.49E-06	41	0.134344	8.48E-06
	x_8	100000	10	0.025042	6.68E-06	14	0.039831	3.81E-06	45	0.142512	8.35E-06	45	0.140629	8.35E-06
	$0.1x_8$	100000	9	0.023247	9.54E-06	12	0.032424	9.67E-06	41	0.127833	9.43E-06	41	0.131296	9.43E-06
	$0.01x_8$	100000	9	0.023961	7.82E-06	12	0.033165	7.93E-06	41	0.127934	7.73E-06	41	0.128443	7.73E-06
	$0.05x_8$	100000	9	0.023919	8.58E-06	12	0.032843	8.70E-06	41	0.122881	8.49E-06	41	0.153393	8.48E-06
12	x_{10}	10000	8	0.006405	1.75E-06	10	0.002139	8.97E-06	36	0.005094	9.25E-06	36	0.007093	9.25E-06
	$0.1x_{10}$	10000	8	0.002913	1.75E-06	10	0.001938	8.97E-06	36	0.005603	9.25E-06	36	0.009112	9.25E-06
	$0.01x_{10}$	10000	8	0.001776	1.75E-06	10	0.001977	8.97E-67	36	0.004819	9.25E-06	36	0.005888	9.25E-06
	$0.05x_{10}$	10000	8	0.001668	1.75E-06	10	0.001808	8.97E-06	36	0.004557	9.25E-06	36	0.008036	9.25E-06
	x_{10}	100000	8	0.001561	1.75E-06	10	0.002187	8.97E-06	36	0.007275	9.25E-06	36	0.005895	9.25E-06
	$0.1x_{10}$	100000	8	0.001709	1.75E-06	10	0.002243	8.97E-06	36	0.004927	9.25E-06	36	0.006378	9.25E-06
	$0.01x_{10}$	100000	8	0.001672	1.75E-06	10	0.002473	8.97E-06	36	0.005587	9.25E-06	36	0.006688	9.25E-06
	$0.05x_{10}$	100000	8	0.002468	1.75E-06	10	0.001901	8.97E-06	36	0.005117	9.25E-06	36	0.006275	9.25E-06

Table 4.9: Test results of ADLCG, FCGM, TTPRP, and MHSCG methods for problems 13-15

				ADLCG			**FCGM**			**TTPRP**			**MHSCG**	
Problem	*Guess*	*Dim*	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$
13	x_{10}	10000	23	0.007484	8.66E-06	36	0.003728	9.90E-06	36	0.004084	9.90E-06	36	0.004517	9.90E-06
	$0.1x_{10}$	10000	23	0.003393	8.66E-06	36	0.004395	9.90E-06	36	0.004228	9.90E-06	36	0.005335	9.90E-06
	x_8	10000	20	0.003781	6.62E-06	31	0.003449	9.68E-06	31	0.003684	9.68E-06	31	0.004298	9.68E-06
	$0.1x_8$	10000	22	0.003368	8.42E-06	35	0.004978	8.32E-06	35	0.003314	8.32E-06	35	0.006332	8.31E-06
	$0.01x_8$	100000	23	0.003825	8.24E-06	36	0.005642	9.46E-06	36	0.003575	9.46E-06	36	0.006559	9.46E-06
	$0.05x_8$	100000	23	0.003504	6.75E-06	36	0.003349	7.87E-06	36	0.004927	7.87E-06	36	0.004538	7.86E-06
	$-0.01x_8$	100000	23	0.002787	9.10E-06	37	0.004936	7.78E-06	37	0.004626	7.78E-06	37	0.005741	7.78E-06
	$-0.05x_8$	100000	24	0.003577	6.97E-06	37	0.003651	9.38E-06	37	0.003634	9.38E-06	37	0.004805	9.38E-06
14	$0.1x_1$	10000	7	0.078658	7.44E-06	26	0.132404	9.95E-06	152	0.898099	9.98E-06	_	_	_
	$0.01x_1$	10000	7	0.072553	8.82E-06	27	0.132621	7.01E-06	153	0.920906	9.88E-06	_	_	_
	$0.5x_1$	10000	6	0.071375	6.17E-06	25	0.126765	7.62E-06	147	0.875104	9.39E-06	_	_	_
	$0.05x_1$	10000	7	0.074329	8.20E-06	27	0.134719	6.67E-06	153	0.912578	9.48E-06	_	_	_
	x_3	100000	54	3.032616	9.63E-06	54	3.003468	9.06E-06	164	13.754637	9.90E-06	_	_	_
	$0.1x_3$	100000	40	2.424672	9.11E-06	63	3.425866	9.34E-06	164	13.393925	9.90E-06	_	_	_
	$0.01x_3$	100000	26	1.974709	9.29E-06	47	2.705293	8.99E-06	164	13.376023	9.90E-06	_	_	_
	$0.05x_3$	100000	36	2.415458	8.74E-06	51	2.804204	8.51E-06	164	13.375545	9.90E-06	_	_	_
15	x_3	10000	20	0.102028	9.43E-06	33	0.121664	4.68E-06	78	0.329665	9.54E-06	_	_	_
	$0.1x_3$	10000	20	0.105891	9.44E-06	33	0.124292	7.98E-06	78	0.327049	9.55E-06	_	_	_
	$0.01x_3$	10000	20	0.104356	9.44E-06	40	0.144258	6.26E-06	78	0.326403	9.55E-06	_	_	_
	$0.05x_3$	10000	20	0.113504	9.44E-06	45	0.147036	8.25E-06	78	0.322093	9.55E-06	_	_	_
	x_3	100000	22	1.376324	5.74E-06	46	2.042894	4.49E-06	84	4.867944	8.71E-06	_	_	_
	$0.1x_3$	100000	22	1.155618	5.74E-06	43	1.886833	9.46E-06	84	4.875971	8.71E-06	_	_	_
	$0.01x_3$	100000	22	1.194392	5.74E-06	44	1.906549	5.45E-06	84	4.801281	8.71E-06	_	_	_
	$0.05x_3$	100000	22	1.263172	5.74E-06	45	1.984194	6.89E-06	84	4.839611	8.71E-06	_	_	_

Table 4.10: Test results of IHZCG, FCGM, TTPRP, and MHSCG methods for problems 1-4

			IHZCG			**FCGM**			**TTPRP**			**MHSCG**		
Problem	*Guess*	*Dim*	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$
1	x_1	10000	37	0.156296	8.92E-06	23	0.155001	9.65E-06	153	0.575608	9.14E-06	73	0.305228	1.23E-09
	$0.1x_1$	10000	6	0.104941	1.66E-06	20	0.085559	9.52E-06	132	0.505874	9.11E-06	52	0.236625	1.58E-09
	$0.01x_1$	10000	2	0.069399	3.33E-06	17	0.080258	7.63E-06	109	0.407292	0.00E+00	30	0.152127	2.28E-11
	$0.05x_1$	10000	17	0.128565	7.18E-06	19	0.128151	9.53E-06	125	0.591275	9.53E-06	51	0.279019	3.81E-09
	x_2	100000	14	1.008053	9.08E-07	16	1.224823	8.43E-06	104	5.637419	9.87E-06	_	_	_
	$0.1x_2$	100000	8	0.702957	8.16E-06	13	1.187925	7.04E-06	83	4.718356	9.19E-06	_	_	_
	$0.01x_2$	100000	1	0.544602	6.30E-08	10	0.980073	5.64E-06	61	3.573554	9.34E-06	_	_	_
	$0.05x_1$	100000	17	1.601567	9.03E-06	21	1.237148	7.54E-06	119	9.653821	3.93E-19	46	4.731775	2.57E-09
2	$0.1x_3$	10000	8	0.067195	8.61E-06	15	0.093266	8.61E-06	38	0.206777	6.75E-06	_	_	_
	$0.01x_3$	10000	2	0.053142	6.68E-06	10	0.082149	7.87E-06	12	0.104852	8.55E-06	_	_	_
	$0.05x_3$	10000	6	0.062956	8.82E-06	12	0.086349	9.97E-06	39	0.210418	8.36E-06	_	_	_
	x_3	10000	15	0.100594	6.53E-06	19	0.152132	7.96E-06	44	0.462105	3.01E-06	_	_	_
	$-0.1x_3$	100000	8	0.783425	8.96E-06	13	0.962553	9.50E-06	71	4.482847	3.03E-10	_	_	_
	$-0.01x_3$	100000	2	0.601417	6.26E-06	10	0.862709	7.82E-06	49	3.221027	3.24E-10	_	_	_
	x_3	100000	15	1.217513	6.55E-06	19	1.105058	7.95E-06	65	5.517453	3.60E-08	_	_	_
	$-0.05x_3$	100000	6	0.722499	8.89E-06	12	1.319637	9.65E-06	65	4.153584	2.77E-10	_	_	_
3	$0.1x_2$	10000	8	0.054248	9.46E-06	13	0.056946	6.75E-06	82	0.255208	9.98E-06	_	_	_
	$0.01x_2$	10000	2	0.060044	6.19E-06	10	0.070682	5.61E-06	61	0.201881	9.32E-06	_	_	_
	x_2	10000	15	0.086016	9.61E-06	16	0.116124	5.83E-06	102	0.396138	9.90E-06	_	_	_
	$0.05x_2$	10000	6	0.064924	9.41E-06	12	0.066723	6.90E-06	76	0.242659	9.50E-06	_	_	_
	x_4	100000	21	1.151238	9.64E-06	32	1.123662	5.56E-06	139	5.909545	9.86E-06	_	_	_
	$0.1x_4$	100000	7	0.811756	7.95E-06	21	0.925821	5.76E-06	109	4.501971	9.44E-06	_	_	_
	$0.01x_4$	100000	11	1.098621	5.32E-06	14	1.263989	8.06E-06	44	3.483048	8.31E-06	_	_	_
	$0.5x_4$	100000	28	1.182758	5.84E-06	33	1.264114	5.83E-06	125	5.219315	9.66E-06	_	_	_
4	$0.1x_1$	10000	32	0.135933	3.58E-06	231	0.794097	9.93E-06	158	0.710892	9.45E-06	_	_	_
	$-0.1x_1$	10000	28	0.126882	8.68E-06	349	1.167591	9.89E-06	162	0.744559	8.62E-06	_	_	_
	$0.01x_1$	10000	28	0.125078	7.56E-06	155	0.559408	9.82E-06	113	0.522713	1.00E-05	_	_	_
	$-0.05x_1$	10000	30	0.178465	7.54E-06	168	0.597262	9.86E-06	145	0.666452	9.46E-06	_	_	_
	$0.5x_1$	100000	26	1.789252	8.48E-06	321	19.165534	9.73E-06	193	14.969885	7.48E-06	_	_	_
	$-0.01x_1$	100000	29	2.242087	7.89E-06	133	8.471215	9.91E-06	112	8.899216	9.90E-06	-	_	_
	$-0.5x_1$	100000	42	3.771355	7.59E-06	473	39.063297	9.70E-06	200	20.735541	9.79E-06	_	_	_
	$-0.05x_1$	100000	31	2.182115	6.61E-06	152	9.751605	9.89E-06	152	11.896874	9.16E-06	_	_	_

Table 4.11: Test results of IHZCG, FCGM, TTPRP, and MHSCG methods for problems 5-8

			IHZCG			**FCGM**			**TTPRP**			**MHSCG**	
Problem	*Guess*	*Dim*	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$
5	-x_5	10000	4	0.083406	3.33E-06	25	0.138438	8.10E-06	163	0.983816	9.46E-06	_	_	_
	0.1x_5	10000	4	0.081204	3.32E-06	25	0.130675	8.10E-06	163	0.987082	9.46E-06	_	_	_
	0.5x_5	10000	4	0.075566	3.32E-11	25	0.184446	8.10E-06	163	0.978568	9.46E-06	_	_	_
	x_6	10000	4	0.084096	2.17E-11	25	0.125153	8.85E-06	164	0.983153	9.30E-06	_	_	_
	-x_6	100000	1	0.720705	2.83E-06	27	1.822457	5.82E-06	173	15.748324	9.47E-06	_	_	_
	x_2	100000	1	0.491211	2.86E-06	27	1.963562	6.40E-06	174	15.826656	9.39E-06	_	_	_
	0.1x_2	100000	1	0.796306	2.86E-06	27	2.690999	6.40E-06	174	21.781523	9.39E-06	_	_	_
	-x_2	100000	1	5.08E-07	2.86E-06	27	1.806925	6.40E-06	174	15.689845	9.39E-06	_	_	_
6	x_2	10000	27	0.106026	6.23E-06	144	0.436209	9.33E-06	126	0.478785	9.17E-06	_	_	_
	-x_2	10000	27	0.107211	9.85E-06	154	0.460833	8.94E-06	142	0.538366	9.52E-06	_	_	_
	0.1x_7	10000	28	0.117105	6.60E-06	151	0.452602	9.72E-06	133	0.497542	8.50E-06	_	_	_
	-0.1x_7	10000	25	0.098382	9.19E-06	152	0.459534	9.50E-06	145	0.557342	9.90E-06	_	_	_
	x_2	100000	28	1.696057	4.21E-06	149	7.921229	9.35E-06	129	8.734475	9.64E-06	_	_	_
	-x_2	100000	27	1.603018	5.54E-06	151	8.147204	9.80E-06	141	9.351653	6.60E-06	_	_	_
	0.1x_7	100000	33	1.881507	7.60E-06	141	7.669758	9.49E-06	161	11.133489	7.53E-06	_	_	_
	-0.1x_7	100000	22	1.320448	7.07E-06	130	7.084815	8.06E-06	152	10.629789	9.88E-06	_	_	_
7	x_8	10000	7	0.000937	2.48E-06	10	0.002193	3.93E-06	36	0.004657	7.43E-06	36	0.005741	7.42E-06
	-1x_8	10000	10	0.001931	1.18E-06	10	0.001436	6.49E-06	37	0.005685	9.55E-06	37	0.005907	9.56E-06
	0.1x_8	10000	6	0.000778	5.83E-06	9	0.001893	4.14E-06	32	0.003777	9.71E-06	32	0.004922	9.71E-06
	-0.01x_8	10000	6	0.001208	8.33E-06	9	0.001923	5.99E-06	33	0.005483	9.99E-06	9	0.001637	5.99E-06
	-0.1x_8	100000	7	0.000849	1.20E-06	9	0.001695	7.50E-06	34	0.005071	8.85E-06	34	0.005368	8.85E-06
	0.5x_8	100000	6	0.000925	4.54E-06	9	0.003441	3.13E-06	31	0.005915	9.92E-06	31	0.005251	9.91E-06
	-0.5x_8	100000	7	0.001029	1.82E-06	10	0.001581	3.80E-06	36	0.004421	8.39E-06	36	0.008238	8.39E-06
	0.01x_8	100000	6	0.001266	7.89E-06	9	0.001738	5.65E-06	33	0.004494	9.42E-06	33	0.007239	9.42E-06
8	x_9	10000	3	0.007979	1.40E-06	3	0.010773	5.47E-06	10	0.037453	9.89E-06	10	0.040602	9.89E-06
	$-x_9$	10000	3	0.010248	1.40E-06	3	0.011277	5.47E-06	10	0.037975	9.89E-06	10	0.035228	9.88E-06
	x_8	10000	6	0.016976	2.92E-06	9	0.035571	3.07E-06	35	0.116387	8.68E-06	35	0.170489	8.68E-06
	0.1x_9	10000	2	0.007213	1.40E-06	1	0.006371	8.75E-06	4	0.016425	8.40E-06	4	0.018843	8.41E-06
	-0.1x_9	10000	2	0.007874	1.40E-06	1	0.006554	8.75E-06	4	0.018193	8.40E-06	4	0.016013	8.41E-06
	x_9	100000	2	0.005685	1.40E-06	1	0.003644	8.75E-06	4	0.015402	8.40E-06	4	0.014701	8.41E-06
	-0.1x_8	100000	6	0.014551	1.31E-06	8	0.024697	4.88E-06	29	0.097154	9.89E-06	29	0.127271	9.89E-06
	-x_9	100000	2	0.004752	1.40E-06	1	0.003854	8.75E-06	4	0.015057	8.40E-06	4	0.015639	8.40E-06

Table 4.12: Test results of IHZCG, FCGM, TTPRP, and MHSCG methods for problems 9-12

			IHZCG			**FCGM**			**TTPRP**			**MHSCG**	
Problem	*Guess*	*Dim*	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$
9	x_{10}	10000	6	0.017192	8.11E-06	9	0.025774	5.82E-06	33	0.104227	9.71E-06	33	0.107314	9.71E-06
	$0.1x_{10}$	10000	6	0.013469	8.11E-06	9	0.025674	5.82E-06	33	0.102471	9.71E-06	33	0.155634	9.71E-06
	$0.01x_{10}$	10000	6	0.017649	8.11E-06	9	0.025427	5.82E-06	33	0.102853	9.71E-06	33	0.141926	9.71E-06
	$0.05x_{10}$	10000	6	0.013882	8.11E-06	9	0.025229	5.82E-06	33	0.104138	9.71E-06	33	0.137566	9.71E-06
	x_8	100000	7	0.018358	2.48E-06	10	0.030186	3.93E-06	36	0.070646	7.43E-06	36	0.151436	7.42E-06
	$0.1x_8$	100000	6	0.014997	5.82E-06	9	0.025952	4.14E-06	32	0.102319	9.71E-06	32	0.133125	9.71E-06
	$0.01x_8$	100000	6	0.017532	7.89E-06	9	0.024548	5.65E-06	33	0.105205	9.42E-06	33	0.106442	9.42E-06
	$0.05x_8$	100000	6	0.014559	6.98E-06	9	0.026384	4.98E-06	33	0.546842	8.28E-06	33	0.147553	8.28E-06
10	x_8	10000	13	0.002774	8.21E-06	14	0.002979	5.83E-06	42	0.010205	9.62E-06	42	0.008833	9.61E-06
	$0.1x_8$	10000	11	0.001667	6.52E-06	5	0.0075	8.72E-06	47	0.011397	8.56E-06	47	0.010453	8.55E-06
	$0.01x_8$	10000	10	0.001541	4.59E-06	7	0.002193	9.35E-06	35	0.006223	8.43E-06	35	0.007927	8.42E-06
	$0.05x_8$	10000	11	0.002153	5.69E-06	8	0.001688	5.02E-06	43	0.013716	8.31E-06	43	0.008163	8.32E-06
	x_8	100000	11	0.002227	5.69E-06	14	0.003483	5.83E-06	42	0.006612	9.62E-06	42	0.009589	9.61E-06
	$0.1x_8$	100000	11	0.001344	6.52E-06	5	0.001439	8.72E-06	47	0.011248	8.56E-06	47	0.009827	8.55E-06
	$0.01x_8$	100000	10	0.001832	4.59E-06	7	0.001982	9.35E-06	35	0.009566	8.43E-06	35	0.011069	8.42E-06
	$0.05x_8$	100000	11	0.001694	5.69E-06	8	0.002381	5.02E-06	43	0.007416	8.31E-06	43	0.009422	8.32E-06
11	x_8	10000	10	0.021773	4.55E-06	14	0.038989	3.81E-06	45	0.145166	8.35E-06	45	0.137219	8.35E-06
	$0.1x_8$	10000	9	0.020038	9.49E-06	12	0.032704	9.67E-06	41	0.126207	9.43E-06	41	0.131898	9.43E-06
	$0.01x_8$	10000	9	0.020544	7.80E-06	12	0.033526	7.93E-06	41	0.126085	7.73E-06	41	0.127325	7.73E-06
	$0.05x_8$	10000	9	0.020223	8.55E-06	12	0.034162	8.70E-06	41	0.124181	8.49E-06	41	0.134344	8.48E-06
	x_8	100000	10	0.020692	4.55E-06	14	0.039831	3.81E-06	45	0.142512	8.35E-06	45	0.140629	8.35E-06
	$0.1x_8$	100000	9	0.023247	9.54E-06	12	0.032424	9.67E-06	41	0.127833	9.43E-06	41	0.131296	9.43E-06
	$0.01x_8$	100000	9	0.018939	9.49E-06	12	0.033165	7.93E-06	41	0.127934	7.73E-06	41	0.128443	7.73E-06
	$0.05x_8$	100000	9	0.023087	8.55E-06	12	0.032843	8.70E-06	41	0.122881	8.49E-06	41	0.153393	8.48E-06
12	x_{10}	10000	8	0.001018	1.75E-06	10	0.002139	8.97E-06	36	0.005094	9.25E-06	36	0.007093	9.25E-06
	$0.1x_{10}$	10000	8	0.001134	1.75E-06	10	0.001938	8.97E-06	36	0.005603	9.25E-06	36	0.009112	9.25E-06
	$0.01x_{10}$	10000	8	0.000842	1.75E-06	10	0.001977	8.97E-67	36	0.004819	9.25E-06	36	0.005888	9.25E-06
	$0.05x_{10}$	10000	8	0.001438	1.75E-06	10	0.001808	8.97E-06	36	0.004557	9.25E-06	36	0.008036	9.25E-06
	x_{10}	100000	8	0.001967	1.75E-06	10	0.002187	8.97E-06	36	0.007275	9.25E-06	36	0.005895	9.25E-06
	$0.1x_{10}$	100000	8	0.001165	1.75E-06	10	0.002243	8.97E-06	36	0.004927	9.25E-06	36	0.006378	9.25E-06
	$0.01x_{10}$	100000	8	0.001422	1.75E-06	10	0.002473	8.97E-06	36	0.005587	9.25E-06	36	0.006688	9.25E-06
	$0.05x_{10}$	100000	8	0.000924	1.75E-06	10	0.001901	8.97E-06	36	0.005117	9.25E-06	36	0.006275	9.25E-06

Table 4.13: Test results of IHZCG, FCGM, TTPRP, and MHSCG methods for problems 13-15

			IHZCG			**FCGM**			**TTPRP**			**MHSCG**		
Problem	*Guess*	*Dim*	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$
13	x_{10}	10000	23	0.002321	8.66E-06	36	0.003728	9.90E-06	36	0.004084	9.90E-06	36	0.004517	9.90E-06
	$0.1x_{10}$	10000	23	0.002721	8.66E-06	36	0.004395	9.90E-06	36	0.004228	9.90E-06	36	0.005335	9.90E-06
	x_8	10000	20	0.002104	6.62E-06	31	0.003449	9.68E-06	31	0.003684	9.68E-06	31	0.004298	9.68E-06
	$0.1x_8$	10000	22	0.002432	8.42E-06	35	0.004978	8.32E-06	35	0.003314	8.32E-06	35	0.006332	8.31E-06
	$0.01x_8$	100000	23	0.002476	8.24E-06	36	0.005642	9.46E-06	36	0.003575	9.46E-06	36	0.006559	9.46E-06
	$0.05x_8$	100000	23	0.001746	6.75E-06	36	0.003349	7.87E-06	36	0.004927	7.87E-06	36	0.004538	7.86E-06
	$-0.01x_8$	100000	23	0.001726	9.10E-06	37	0.004936	7.78E-06	37	0.004626	7.78E-06	37	0.005741	7.78E-06
	$-0.05x_8$	100000	24	0.002752	6.97E-06	37	0.003651	9.38E-06	37	0.003634	9.38E-06	37	0.004805	9.38E-06
14	$0.1x_1$	10000	9	0.092251	6.70E-06	26	0.132404	9.95E-06	152	0.898099	9.98E-06	_	_	_
	$0.01x_1$	10000	10	0.097659	8.71E-06	27	0.132621	7.01E-06	153	0.920906	9.88E-06	_	_	_
	$0.5x_1$	10000	7	0.088474	7.12E-06	25	0.126765	7.62E-06	147	0.875104	9.39E-06	_	_	_
	$0.05x_1$	10000	9	0.091923	8.43E-06	27	0.134719	6.67E-06	153	0.912578	9.48E-06	_	_	_
	x_3	100000	23	1.684116	8.79E-06	54	3.003468	9.06E-06	164	13.754637	9.90E-06	_	_	_
	$0.1x_3$	100000	18	1.273259	8.87E-06	63	3.425866	9.34E-06	164	13.393925	9.90E-06	_	_	_
	$0.01x_3$	100000	14	1.145738	7.44E-06	47	2.705293	8.99E-06	164	13.376023	9.90E-06	_	_	_
	$0.05x_3$	100000	17	1.280964	8.39E-06	51	2.804204	8.51E-06	164	13.375545	9.90E-06	_	_	_
15	x_3	10000	21	0.101822	7.32E-06	33	0.121664	4.68E-06	78	0.329665	9.54E-06	_	_	_
	$0.1x_3$	10000	21	0.103916	5.56E-06	33	0.124292	7.98E-06	78	0.327049	9.55E-06	_	_	_
	$0.01x_3$	10000	19	0.103268	8.18E-06	40	0.144258	6.26E-06	78	0.326403	9.55E-06	_	_	_
	$0.05x_3$	10000	20	0.105858	4.69E-06	45	0.147036	8.25E-06	78	0.322093	9.55E-06	_	_	_
	x_3	100000	27	1.565245	6.59E-06	46	2.042894	4.49E-06	84	4.867944	8.71E-06	_	_	_
	$0.1x_3$	100000	25	1.543934	6.89E-06	43	1.886833	9.46E-06	84	4.875971	8.71E-06	_	_	_
	$0.01x_3$	100000	28	1.882472	6.63E-06	44	1.906549	5.45E-06	84	4.801281	8.71E-06	_	_	_
	$0.05x_3$	100000	27	1.665508	4.93E-06	45	1.984194	6.89E-06	84	4.839611	8.71E-06	_	_	_

Table 4.14: Test results of EHZCG, FCGM, TTPRP, and MHSCG methods for problems 1-4

			EHZCG			**FCGM**			**TTPRP**			**MHSCG**		
Problem	*Guess*	*Dim*	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$
1	x_1	10000	36	0.148597	6.87E-06	23	0.155001	9.65E-06	153	0.575608	9.14E-06	73	0.305228	1.23E-09
	$0.1x_1$	10000	18	0.084631	8.94E-06	20	0.085559	9.52E-06	132	0.505874	9.11E-06	52	0.236625	1.58E-09
	$0.01x_1$	10000	2	0.049529	5.83E-06	17	0.080258	7.63E-06	109	0.407292	0.00E+00	30	0.152127	2.28E-11
	$0.05x_1$	10000	16	0.113646	9.05E-06	19	0.128151	9.53E-06	125	0.591275	9.53E-06	51	0.279019	3.81E-09
	x_2	100000	20	1.042494	7.40E-06	16	1.224823	8.43E-06	104	5.637419	9.87E-06	_	_	_
	$0.1x_2$	100000	2	0.506884	7.45E-06	13	1.187925	7.04E-06	83	4.718356	9.19E-06	_	_	_
	$0.05x_1$	100000	18	1.893175	7.75E-06	21	1.237148	7.54E-06	119	9.653821	3.93E-19	46	4.731775	2.57E-09
	$0.01x_2$	100000	1	0.547453	6.30E-08	10	0.980073	5.64E-06	61	3.573554	9.34E-06	_	_	_
2	$0.1x_3$	10000	11	0.078177	6.79E-06	15	0.093266	8.61E-06	38	0.206777	6.75E-06	_	_	_
	$0.01x_3$	10000	2	0.059277	8.01E-06	10	0.082149	7.87E-06	12	0.104852	8.55E-06	_	_	_
	x_3	10000	20	0.134541	6.81E-06	19	0.152132	7.96E-06	44	0.462105	3.01E-06	_	_	_
	$0.05x_3$	10000	8	0.072361	8.03E-06	12	0.086349	9.97E-06	39	0.210418	8.36E-06	_	_	_
	$-0.1x_3$	100000	11	0.848263	7.10E-06	13	0.962553	9.50E-06	71	4.482847	3.03E-10	_	_	_
	$-0.01x_3$	100000	2	0.650914	7.52E-06	10	0.862709	7.82E-06	49	3.221027	3.24E-10	_	_	_
	x_3	100000	20	1.944418	6.83E-06	19	1.105058	7.95E-06	65	5.517453	3.60E-08	_	_	_
	$-0.05x_3$	100000	8	0.824267	8.12E-06	12	1.319637	9.65E-06	65	4.153584	2.77E-10	_	_	_
3	$0.1x_2$	10000	11	0.054849	7.38E-06	13	0.056946	6.75E-06	82	0.255208	9.98E-06	_	_	_
	$0.01x_2$	10000	2	0.040251	7.43E-06	10	0.070682	5.61E-06	61	0.201881	9.32E-06	_	_	_
	$0.05x_2$	10000	8	0.047977	8.48E-06	12	0.066723	6.90E-06	76	0.242659	9.50E-06	_	_	_
	x_2	10000	21	0.075819	6.07E-06	16	0.116124	5.83E-06	102	0.396138	9.90E-06	_	_	_
	x_4	100000	31	1.515763	6.06E-06	32	1.123662	5.56E-06	139	5.909545	9.86E-06	_	_	_
	$0.1x_4$	100000	25	1.087755	6.72E-06	21	0.925821	5.76E-06	109	4.501971	9.44E-06	_	_	_
	$0.01x_4$	100000	19	1.512887	6.39E-06	14	1.263989	8.06E-06	44	3.483048	8.31E-06	_	_	_
	$0.5x_4$	100000	20	1.235466	7.48E-06	33	1.264114	5.83E-06	125	5.219315	9.66E-06	_	_	_
4	$0.1x_1$	10000	35	0.168621	6.64E-06	231	0.794097	9.93E-06	158	0.710892	9.45E-06	_	_	_
	$-0.1x_1$	10000	35	0.175702	7.20E-06	349	1.167591	9.89E-06	162	0.744559	8.62E-06	_	_	_
	$0.01x_1$	10000	29	0.159991	9.83E-06	155	0.559408	9.82E-06	113	0.522713	1.00E-05	_	_	_
	$-0.05x_1$	10000	33	0.166416	9.00E-06	168	0.597262	9.86E-06	145	0.666452	9.46E-06	_	_	_
	$0.5x_1$	100000	36	2.659826	6.26E-06	321	19.165534	9.73E-06	193	14.969885	7.48E-06	_	_	_
	$-0.01x_1$	100000	34	2.764094	6.44E-06	133	8.471215	9.91E-06	112	8.899216	9.90E-06	-	_	_
	$-0.05x_1$	100000	33	4.956198	8.27E-06	168	0.597262	9.86E-06	145	0.666452	9.46E-06	_	_	_
	$-0.05x_1$	100000	33	3.018176	8.27E-06	152	9.751605	9.89E-06	152	11.896874	9.16E-06	_	_	_

Table 4.15: Test results of EHZCG, FCGM, TTPRP, and MHSCG methods for problems 5-8

				EHZCG			**FCGM**			**TTPRP**			**MHSCG**	
Problem	*Guess*	*Dim*	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$
5	-x_5	10000	5	0.072507	8.17E-06	25	0.138438	8.10E-06	163	0.983816	9.46E-06	_	_	_
	0.1x_5	10000	5	0.075983	8.17E-06	25	0.130675	8.10E-06	163	0.987082	9.46E-06	_	_	_
	x_6	10000	3	0.065268	9.32E-06	25	0.125153	8.85E-06	164	0.983153	9.30E-06	_	_	_
	0.5x_5	10000	5	0.094494	8.17E-06	25	0.184446	8.10E-06	163	0.978568	9.46E-06	_	_	_
	-x_6	100000	1	0.531193	2.83E-06	27	1.822457	5.82E-06	173	15.748324	9.47E-06	_	_	_
	x_2	100000	1	0.511996	2.86E-06	27	1.963562	6.40E-06	174	15.826656	9.39E-06	_	_	_
	0.1x_2	100000	1	0.600347	2.86E-06	27	2.690999	6.40E-06	174	21.781523	9.39E-06	_	_	_
	-x_2	100000	1	5.27E-01	2.86E-06	27	1.806925	6.40E-06	174	15.689845	9.39E-06	_	_	_
6	x_2	10000	30	0.129917	9.02E-06	144	0.436209	9.33E-06	126	0.478785	9.17E-06	_	_	_
	-x_2	10000	32	0.135301	6.68E-06	154	0.460833	8.94E-06	142	0.538366	9.52E-06	_	_	_
	0.1x_7	10000	33	0.139049	7.72E-06	151	0.452602	9.72E-06	133	0.497542	8.50E-06	_	_	_
	-0.1x_7	10000	29	0.117555	7.70E-06	152	0.459534	9.50E-06	145	0.557342	9.90E-06	_	_	_
	x_2	100000	30	2.099385	9.00E-06	149	7.921229	9.35E-06	129	8.734475	9.64E-06	_	_	_
	-x_2	100000	33	2.300234	6.95E-06	151	8.147204	9.80E-06	141	9.351653	6.60E-06	_	_	_
	0.1x_7	100000	35	2.437956	8.69E-06	141	7.669758	9.49E-06	161	11.133489	7.53E-06	_	_	_
	-0.1x_7	100000	32	2.061218	4.77E-06	130	7.084815	8.06E-06	152	10.629789	9.88E-06	_	_	_
7	x_8	10000	6	0.000781	6.79E-07	10	0.002193	3.93E-06	36	0.004657	7.43E-06	36	0.005741	7.42E-06
	-1x_8	10000	8	0.001409	6.30E-07	10	0.001436	6.49E-06	37	0.005685	9.55E-06	37	0.005907	9.56E-06
	0.1x_8	10000	5	0.000682	2.87E-06	9	0.001893	4.14E-06	32	0.003777	9.71E-06	32	0.004922	9.71E-06
	-0.01x_8	10000	5	0.001478	4.00E-06	9	0.001923	5.99E-06	33	0.005483	9.99E-06	9	0.001637	5.99E-06
	-0.1x_8	100000	5	0.000672	4.77E-06	9	0.001695	7.50E-06	34	0.005071	8.85E-06	34	0.005368	8.85E-06
	0.5x_8	100000	5	0.000785	2.42E-06	9	0.003441	3.13E-06	31	0.005915	9.92E-06	31	0.005251	9.91E-06
	-0.5x_8	100000	5	0.000794	3.96E-06	10	0.001581	3.80E-06	36	0.004421	8.39E-06	36	0.008238	8.39E-06
	0.01x_8	100000	5	0.000678	3.80E-06	9	0.001738	5.65E-06	33	0.004494	9.42E-06	33	0.007239	9.42E-06
8	x_9	10000	2	0.007707	5.60E-06	3	0.010773	5.47E-06	10	0.037453	9.89E-06	10	0.040602	9.89E-06
	-x_9	10000	2	0.009336	5.60E-06	3	0.011277	5.47E-06	10	0.037975	9.89E-06	10	0.035228	9.88E-06
	0.1x_9	10000	2	0.007131	5.60E-06	1	0.006371	8.75E-06	4	0.016425	8.40E-06	4	0.018843	8.41E-06
	x_8	10000	5	0.013349	6.82E-06	9	0.035571	3.07E-06	35	0.116387	8.68E-06	35	0.170489	8.68E-06
	-0.1x_9	10000	2	0.008896	5.60E-07	1	0.006554	8.75E-06	4	0.018193	8.40E-06	4	0.016013	8.41E-06
	x_9	100000	2	0.005166	5.60E-07	1	0.003644	8.75E-06	4	0.015402	8.40E-06	4	0.014701	8.41E-06
	-0.1x_8	100000	4	0.011239	9.21E-06	8	0.024697	4.88E-06	29	0.097154	9.89E-06	29	0.127271	9.89E-06
	-x_9	100000	2	0.005342	5.60E-07	1	0.003854	8.75E-06	4	0.015057	8.40E-06	4	0.015639	8.40E-06

Table 4.16: Test results of EHZCG, FCGM, TTPRP, and MHSCG methods for problems 9-12

				EHZCG			**FCGM**			**TTPRP**			**MHSCG**	
Problem	*Guess*	*Dim*	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$	*Iter*	*Time(s)*	$\|F_k\|$
9	x_{10}	10000	5	0.016272	3.90E-06	9	0.025774	5.82E-06	33	0.104227	9.71E-06	33	0.107314	9.71E-06
	$0.1x_{10}$	10000	5	0.016594	3.90E-06	9	0.025674	5.82E-06	33	0.102471	9.71E-06	33	0.155634	9.71E-06
	$0.01x_{10}$	10000	5	0.014063	3.90E-06	9	0.025427	5.82E-06	33	0.102853	9.71E-06	33	0.141926	9.71E-06
	$0.05x_{10}$	10000	5	0.014261	3.90E-06	9	0.025229	5.82E-06	33	0.104138	9.71E-06	33	0.137566	9.71E-06
	x_8	100000	6	0.016418	6.79E-07	10	0.030186	3.93E-06	36	0.070646	7.43E-06	36	0.151436	7.42E-06
	$0.1x_8$	100000	5	0.016952	2.87E-06	9	0.025952	4.14E-06	32	0.102319	9.71E-06	32	0.133125	9.71E-06
	$0.01x_8$	100000	5	0.014939	3.80E-06	9	0.024548	5.65E-06	33	0.105205	9.42E-06	33	0.106442	9.42E-06
	$0.05x_8$	100000	5	0.014264	3.40E-06	9	0.026384	4.98E-06	33	0.546842	8.28E-06	33	0.147553	8.28E-06
10	x_8	10000	12	0.007932	6.95E-06	14	0.002979	5.83E-06	42	0.010205	9.62E-06	42	0.008833	9.61E-06
	$0.1x_8$	10000	10	0.001599	6.05E-06	5	0.0075	8.72E-06	47	0.011397	8.56E-06	47	0.010453	8.55E-06
	$0.01x_8$	10000	9	0.001759	4.91E-06	7	0.002193	9.35E-06	35	0.006223	8.43E-06	35	0.007927	8.42E-06
	$0.05x_8$	10000	10	0.002015	5.33E-06	8	0.001688	5.02E-06	43	0.013716	8.31E-06	43	0.008163	8.32E-06
	x_8	100000	12	0.002332	6.95E-06	14	0.003483	5.83E-06	42	0.006612	9.62E-06	42	0.009589	9.61E-06
	$0.1x_8$	100000	10	0.003233	6.05E-06	5	0.001439	8.72E-06	47	0.011248	8.56E-06	47	0.009827	8.55E-06
	$0.01x_8$	100000	9	0.002396	4.91E-06	7	0.001982	9.35E-06	35	0.009566	8.43E-06	35	0.011069	8.42E-06
	$0.05x_8$	100000	10	0.002784	5.33E-06	8	0.002381	5.02E-06	43	0.007416	8.31E-06	43	0.009422	8.32E-06
11	x_8	10000	9	0.025667	2.10E-06	14	0.038989	3.81E-06	45	0.145166	8.35E-06	45	0.137219	8.35E-06
	$0.1x_8$	10000	8	0.020843	7.93E-06	12	0.032704	9.67E-06	41	0.126207	9.43E-06	41	0.131898	9.43E-06
	$0.01x_8$	10000	8	0.021499	6.53E-06	12	0.033526	7.93E-06	41	0.126085	7.73E-06	41	0.127325	7.73E-06
	$0.05x_8$	10000	8	0.022548	7.16E-06	12	0.034162	8.70E-06	41	0.124181	8.49E-06	41	0.134344	8.48E-06
	x_8	100000	9	0.026148	2.10E-06	14	0.039831	3.81E-06	45	0.142512	8.35E-06	45	0.140629	8.35E-06
	$0.1x_8$	100000	8	0.020521	7.93E-06	12	0.032424	9.67E-06	41	0.127833	9.43E-06	41	0.131296	9.43E-06
	$0.01x_8$	100000	8	0.019362	6.53E-06	12	0.033165	7.93E-06	41	0.127934	7.73E-06	41	0.128443	7.73E-06
	$0.05x_8$	100000	8	0.022237	7.16E-06	12	0.032843	8.70E-06	41	0.122881	8.49E-06	41	0.153393	8.48E-06
12	x_{10}	10000	6	0.006582	9.04E-06	10	0.002139	8.97E-06	36	0.005094	9.25E-06	36	0.007093	9.25E-06
	$0.1x_{10}$	10000	6	0.001516	9.04E-06	10	0.001938	8.97E-06	36	0.005603	9.25E-06	36	0.009112	9.25E-06
	$0.01x_{10}$	10000	6	0.001489	9.04E-06	10	0.001977	8.97E-67	36	0.004819	9.25E-06	36	0.005888	9.25E-06
	$0.05x_{10}$	10000	6	0.001523	9.04E-06	10	0.001808	8.97E-06	36	0.004557	9.25E-06	36	0.008036	9.25E-06
	x_{10}	100000	6	0.001274	9.04E-06	10	0.002187	8.97E-06	36	0.007275	9.25E-06	36	0.005895	9.25E-06
	$0.1x_{10}$	100000	6	0.001161	9.04E-06	10	0.002243	8.97E-06	36	0.004927	9.25E-06	36	0.006378	9.25E-06
	$0.01x_{10}$	100000	6	0.001583	9.04E-06	10	0.002473	8.97E-06	36	0.005587	9.25E-06	36	0.006688	9.25E-06
	$0.05x_{10}$	100000	6	0.001533	9.04E-06	10	0.001901	8.97E-06	36	0.005117	9.25E-06	36	0.006275	9.25E-06

Table 4.17: Test results of EHZCG, FCGM, TTPRP, and MHSCG methods for problems 13-15

			EHZCG			**FCGM**			**TTPRP**			**MHSCG**		
Problem	*Guess*	*Dim*	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$	*Iter*	*Time(s)*	$\|\|F_k\|\|$
13	x_{10}	10000	22	0.008039	6.07E-06	36	0.003728	9.90E-06	36	0.004084	9.90E-06	36	0.004517	9.90E-06
	$0.1x_{10}$	10000	22	0.002654	6.07E-06	36	0.004395	9.90E-06	36	0.004228	9.90E-06	36	0.005335	9.90E-06
	x_8	10000	18	0.001844	8.28E-06	31	0.003449	9.68E-06	31	0.003684	9.68E-06	31	0.004298	9.68E-06
	$0.1x_8$	10000	21	0.001783	6.10E-06	35	0.004978	8.32E-06	35	0.003314	8.32E-06	35	0.006332	8.31E-06
	$0.01x_8$	100000	21	0.002636	9.62E-06	36	0.005642	9.46E-06	36	0.003575	9.46E-06	36	0.006559	9.46E-06
	$0.05x_8$	100000	21	0.002842	7.86E-06	36	0.003349	7.87E-06	36	0.004927	7.87E-06	36	0.004538	7.86E-06
	$-0.01x_8$	100000	22	0.002152	6.39E-06	37	0.004936	7.78E-06	37	0.004626	7.78E-06	37	0.005741	7.78E-06
	$-0.05x_8$	100000	22	0.003226	7.86E-06	37	0.003651	9.38E-06	37	0.003634	9.38E-06	37	0.004805	9.38E-06
14	$0.1x_1$	10000	10	0.096011	7.85E-06	26	0.132404	9.95E-06	152	0.898099	9.98E-06	_	_	_
	$0.01x_1$	10000	12	0.090336	5.81E-06	27	0.132621	7.01E-06	153	0.920906	9.88E-06	_	_	_
	$0.5x_1$	10000	7	0.076867	5.05E-06	25	0.126765	7.62E-06	147	0.875104	9.39E-06	_	_	_
	$0.05x_1$	10000	12	0.092601	4.82E-06	27	0.134719	6.67E-06	153	0.912578	9.48E-06	_	_	_
	x_3	100000	73	4.257021	9.52E-06	54	3.003468	9.06E-06	164	13.754637	9.90E-06	_	_	_
	$0.1x_3$	100000	55	3.405001	9.92E-06	63	3.425866	9.34E-06	164	13.393925	9.90E-06	_	_	_
	$0.01x_3$	100000	37	2.647096	9.37E-06	47	2.705293	8.99E-06	164	13.376023	9.90E-06	_	_	_
	$0.05x_3$	100000	50	3.041086	9.90E-06	51	2.804204	8.51E-06	164	13.375545	9.90E-06	_	_	_
15	x_3	10000	18	0.118722	5.29E-06	33	0.121664	4.68E-06	78	0.329665	9.54E-06	_	_	_
	$0.1x_3$	10000	18	0.131118	7.44E-06	33	0.124292	7.98E-06	78	0.327049	9.55E-06	_	_	_
	$0.01x_3$	10000	18	0.126021	5.46E-06	40	0.144258	6.26E-06	78	0.326403	9.55E-06	_	_	_
	$0.05x_3$	10000	17	0.122486	7.19E-06	45	0.147036	8.25E-06	78	0.322093	9.55E-06	_	_	_
	x_3	100000	16	1.510277	9.59E-06	46	2.042894	4.49E-06	84	4.867944	8.71E-06	_	_	_
	$0.1x_3$	100000	16	1.538883	6.33E-06	43	1.886833	9.46E-06	84	4.875971	8.71E-06	_	_	_
	$0.01x_3$	100000	16	1.623486	6.37E-06	44	1.906549	5.45E-06	84	4.801281	8.71E-06	_	_	_
	$0.05x_3$	100000	16	1.459846	6.33E-06	45	1.984194	6.89E-06	84	4.839611	8.71E-06	_	_	_

Table 4.18: Number of problems (with percentage) solved by EDLCG, FCGM, TTPRP, and MHSCG methods with least iterations and CPU time

Method	Iter	Percentage	CPU time	Percentage
EDLCG	106	88.33%	95	79.17%
FCGM	14	11.67%	20	16.67%
TTPRP	0	0	4	3.33%
MHSCG	0	0	1	0.83%
Undecided	0	0	0	0

Table 4.19: Number of problems (with percentage) solved by ADLCG, FCGM, TTPRP, and MHSCG methods with least iterations and CPU time

Method	Iter	Percentage	CPU time	Percentage
ADLCG	104	86.67%	91	75.83%
FCGM	11	9.16%	25	20.83%
TTPRP	0	0	3	2.50%
MHSCG	0	0	1	0.84%
Undecided	5	4.17%	0	0

Table 4.20: Number of problems (with percentage) solved by IHZCG, FCGM, TTPRP, and MHSCG methods with least iterations and CPU time

Method	Iter	Percentage	CPU time	Percentage
IHZCG	106	88.33%	107	89.17%
FCGM	11	9.17%	12	10.00%
TTPRP	0	0	1	0.83%
MHSCG	0	0	0	0
Undecided	3	2.50%	0	0

Table 4.21: Number of problems (with percentage) solved by EHZCG, FCGM, TTPRP, and MHSCG methods with least iterations and CPU time

Method	Iter	Percentage	CPU time	Percentage
EHZCG	102	85.00%	99	82.50%
FCGM	8	15.00%	20	16.67%
TTPRP	0	0	1	0.83%
MHSCG	0	0	0	0
Undecided	3	0	0	0

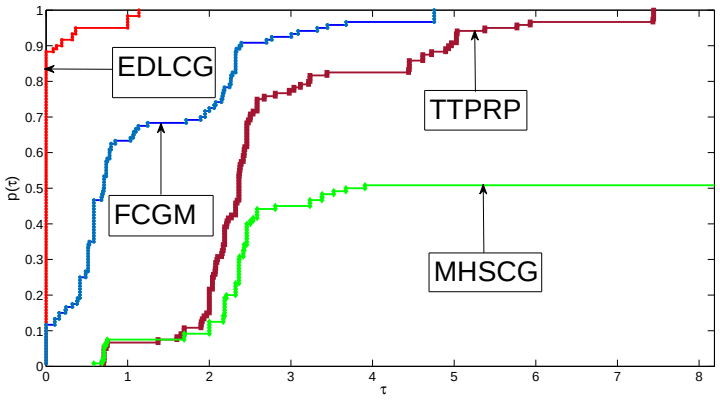

Figure 4.1: Performance profile of EDLCG, FCGM, TTPRP, and MHSCG methods (for number of iterations)

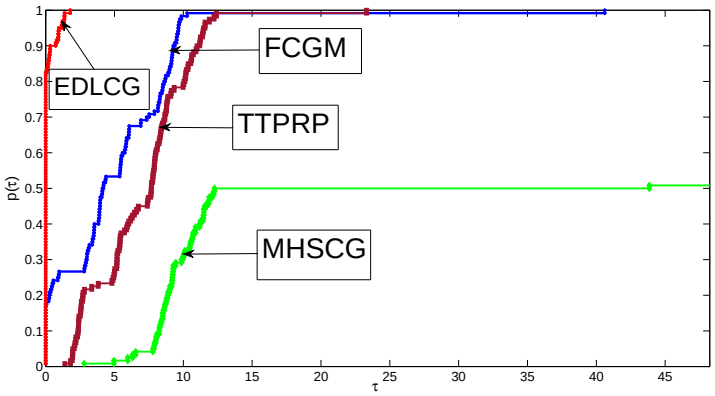

Figure 4.2: Performance profile of EDLCG, FCGM, TTPRP, and MHSCG methods (for CPU time)

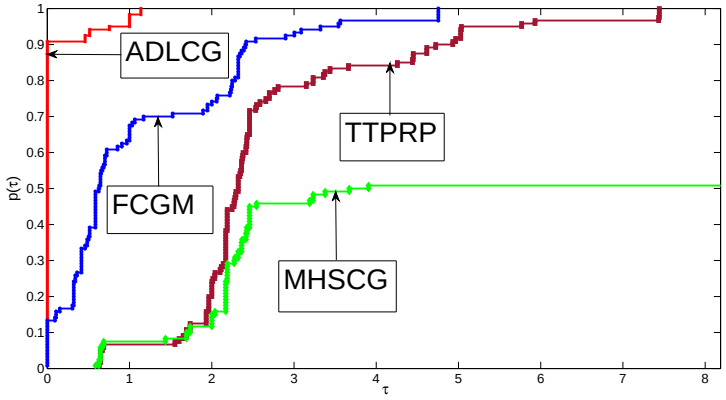

Figure 4.3: Performance profile of ADLCG, FCGM, TTPRP, and MHSCG methods (for number of iterations)

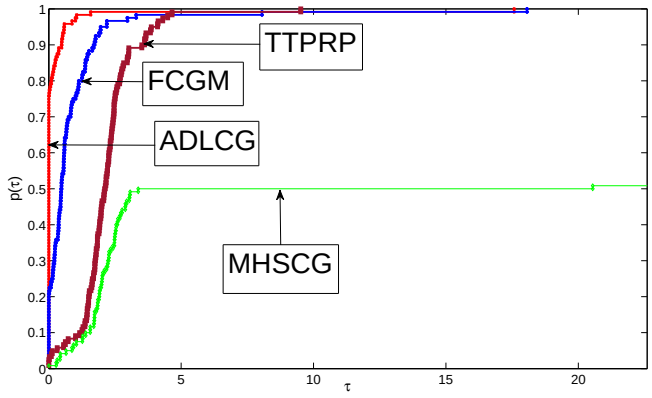

Figure 4.4: Performance profile of ADLCG, FCGM, TTPRP, and MHSCG methods (for CPU time)

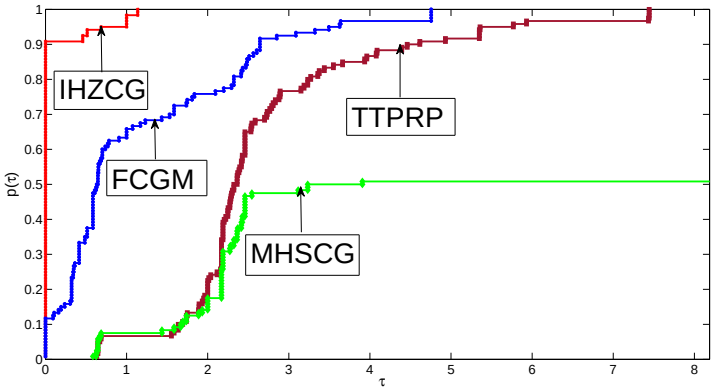

Figure 4.5: Performance profile of IHZCG, FCGM, TTPRP, and MHSCG methods (for number of iterations)

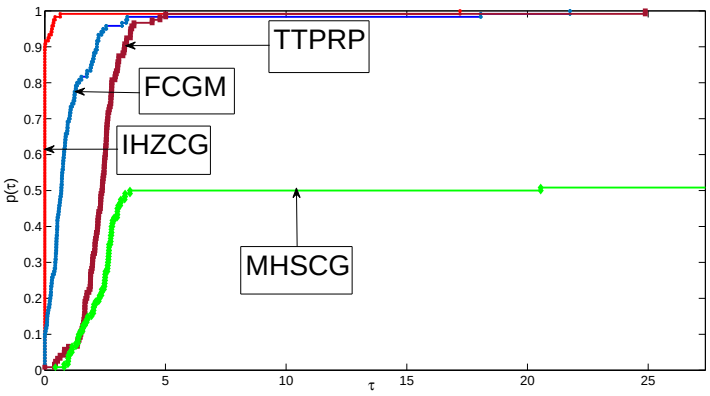

Figure 4.6: Performance profile of IHZCG, FCGM, TTPRP, and MHSCG methods (for CPU time)

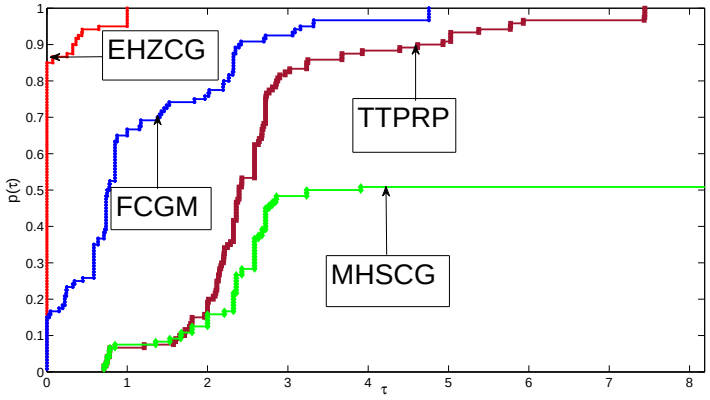

Figure 4.7: Performance profile of EHZCG, FCGM, TTPRP, and MHSCG methods (for number of iterations)

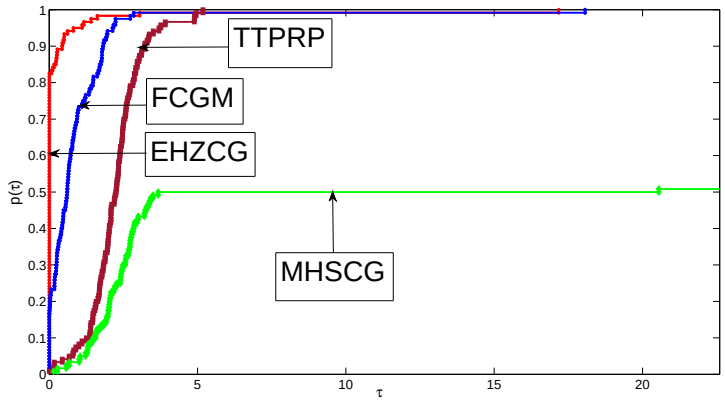

Figure 4.8: Performance profile of EHZCG, FCGM, TTPRP, and MHSCG methods (for CPU time)

4.4 DISCUSSION OF RESULTS

This section analyzes the test results presented in tables $(4.2-4.17)$, the summary that follows in tables $(4.18-4.21)$, as well as interpretations of Figures $(4.1-4.8)$. From tables $(4.2-4.17)$, it is observed that our proposed methods perform better than the three methods used to compare them with in terms of less iterations and CPU time. This is partly due to our choice of the nonnegative parameter t, the inclusion of secant conditions and also because our methods generate descent search directions.

As reported in tables $(4.2-4.5)$, the EDLCG method outperforms the FCGM, TTPRP, and the MHSCG methods in terms of number of iterations as well as CPU time respectively. The tables reveal that the EDLCG method solves more of the test problems with less number of iterations and minimum CPU time. To support that claim, a summary of the analysis is presented in table (4.18). The table indicates that the EDLCG method solves (88.33%) of the problems with less number of iterations compared to FCGM (11.67%). Both TTPRP and MHSCG methods did not record least number of iterations. The table also shows that (79.17%) of the problems were solved by the EDLCG scheme with least CPU time as compared with FCGM (16.67%), TTPRP (3.33%), and MHSCG (0.83%).

Numerical results in tables $(4.6-4.9)$ reveals that the ADLCG method is more efficient than the FCGM, TTPRP, and the MHSCG methods as it solves more problems with minimum number of iteration and CPU time than the three methods. The summary drawn in table (4.19) indicates that the ADLCG method solves (86.67%) of the problems with less number of iterations compared to FCGM, which solves (9.16%) with least number of iterations. In this case also, both TTPRP and MHSCG methods did not record least number of iterations than the others. We also observed that two of the methods scored the same number of iterations in 5 problems, which translates to (4.17%) and is marked as undecided. The table also shows that the ADLCG method solves, (75.83%) of the problems with minimum CPU time compared to FCGM (20.83%), TTPRP (2.50%), and MHSCG (0.84%).

Similarly, test results in tables $(4.10-4.13)$ indicate that the IHZCG method outperforms the FCGM, TTPRP, and MHSCG methods. The summary of the results in table (4.20) shows that IHZCG solves (88.33%) of all the test problems with the least number of iterations compared to NDFCG (9.17%). Two of the methods also solve 3 problems with the same number of iterations, which translates to (2.50%). The table shows that both TTPRP and MHSCG methods did not record least number of iterations. It equally shows that the IHZCG method solves (89.17%) of the problems with minimum CPU time compared to FCGM (10.0%), and TTPRP (0.83%).

It is also observed from tables $(4.14-4.17)$ that the EHZCG method is a winner compared to the FCGM, TTPRP, and MHSCG methods. As in the previous cases, summary of the test results in table (4.21) indicates that the EHZCG scheme solves (85.0%) of the problems with less number of iterations compared with FCGM (15.0%). Both TTPRP and MHSCG methods did not record least number of iterations. The summary also indicates that EHZCG method solves (82.5%) of the test problems with less CPU time compared to FCGM (16.67%) and TTPRP (0.83%).

In addition to tables $(4.2-4.17)$ and their summary in tables $(4.18-4.21)$, performance profile of the seven methods are displayed in Figures $(4.1-4.8)$, using the profile of Dolan and Moré (2002), which shows the performance of these methods relative to CPU time and number of iteration respectively. For each of the eight methods, we plot the fraction $P(\omega)$ of the problems for which the method is within a factor ω of the best time. The top curve in each figure corresponds to the method that solved the most problems in a time that is within a factor ω of the best time. And since our proposed methods correspond to top curves in all the figures, it shows they are more efficient with respect to number of iterations and CPU time.

CHAPTER FIVE

APPLICATION OF THE METHODS

5.1 INTRODUCTION

In this chapter, we highlight the application of our proposed methods in solving the Chandrasekhar H-equation, which is a system of nonlinear equations obtained by dicretizing the Chandrasekhar integral equation, which arises in problems of radiactive transfer. The Chandrasekhar integral equation plays an important role in the theory of radiactive transfer in semi-infinite atmospheres. It was first developed by S. Chandrasekhar (1950) and has since been a subject of much investigation. It has been used to model diverse forms of scattering via the nonlinear integral equation of Chandrasekhar, defined by

$$H(\mu) = 1 + H(\mu) \int_0^1 \frac{\mu}{\mu + t} \psi(t) H(t) dt \tag{5.1.1}$$

Different methods of solving (5.1.1) have been developed because of the important role it plays in radiactive transfer and transport theory (Hivey, 1978). The most common approach of finding approximate solution of (5.1.1) is discretizing it by a vector $\bar{x} \in R^n$, then replacing the integrals by quadrature sums and the derivatives by difference quotients involving only the component of $\bar{x} \in R^n$ (see(James & Warner, 1966)). And so, (5.1.1) becomes a problem of finding the solution of system of n nonlinear equations with n-unknowns as presented in (1.2.1).

5.2 CHANDRASEKHAR H-EQUATION

Here, we give detailed process of discretizing the Chandrasekhar-type integral equations in the radiactive transfer problem in (Chandrasekhar & Breen, 1947), Chandrasekhar and Breen compute H-equation as the solution of the nonlinear integral equation

$$H(\mu) - c\frac{\mu}{2}H(\mu)\int_0^1 \frac{H(y)}{\mu+y}dy = 1, \tag{5.2.1}$$

where c is bounded in the interval $0 \leq c \leq 1$ and $H : [0,1] \longrightarrow R$ is an unknown continuous function. From (5.2.1), we obtain

$$H(\mu)\left[1 - \frac{c}{2}\int_0^1 \frac{\mu H(y)}{\mu+y}dy\right] = 1. \tag{5.2.2}$$

Next, we partition the interval $[0,1]$ into n subintervals, $0 < \mu_1 < ... < \mu_j = j/n < ... < 1$. Denote H_k as $H(\mu_k)$, then the evaluation of (5.2.1) at every μ_i yields the equation

$$H_i\left[1 - \frac{c}{2}\int_0^1 \frac{\mu_i H(y)}{\mu_i+y}dy\right] = 1, \quad i = 1,2,...,n. \tag{5.2.3}$$

By multiplying both sides of (5.2.3) by $\left[1 - (\frac{c}{2}\int_0^1 (\mu H(y)/(\mu+y))dy\right]^{-1}$ and performing some algebra, we arrive at the following

$$F(H)(\mu) = H(\mu) - \left(1 - \frac{c}{2}\int_0^1 \frac{\mu H(y)dy}{\mu+y}\right)^{-1} = 0, \tag{5.2.4}$$

which is known as the Chandrasekhar H-equation (La Cruz & Raydan, 2003).

By discretizing (5.2.4) using the midpoint quadrature formula

$$\int_0^1 f(t)dt = \frac{1}{n}\sum_{j=0}^{n} f(t_j), \tag{5.2.5}$$

for $t_j = (j-0.5)h$, $0 \leq j \leq 1$, $\quad i = 2,...,n, h = \frac{1}{n}$, $c \in (0,1)$, then we have the following:

$$F_i = \mu_i - \left(1 - \frac{c}{2n}\sum_{j=1}^{n} \frac{t_i \mu_j}{t_i + t_j}\right)^{-1}. \tag{5.2.6}$$

Function (5.2.6) is called the discretized Chandrasekhar H-equation which can be solved by some iterative methods.

We apply our methods, and three other methods (i.e A family of conjugate gradient methods for large-scale nonlinear equations (FCGM) (Sun, Wang, & Feng, 2017), A three-terms Polak-Ribier*é*-Polyak conjugate gradient algorithm for large-scale nonlinear equations (TTPRP) (Yuan & Zhang, 2015) and modified Hestenes and Stiefel conjugate gradient algorithm for large-scale nonsmooth minimization and nonlinear equations (MHSCG) (Yuan, Meng, & Li, 2016)) to solve the discretized Chandrasekhar H-equation and present the results in the following tables:

Table 5.22: Test results of Chandrasekhar H-equation with respect to number of iterations/CPU time

	Guess	**Dim**	**EDLCG**	**ADLCG**	**FCGM**	**TTPRP**	**MHSCG**
$c = 0.9$	x_1	10000	26/0.138279	24/0.129067	27/0.140762	142/1.094605	143/1.065634
	$0.1x_1$	10000	22/0.123377	19/0.117541	18/0.122463	120/0.880392	120/0.881846
	$0.01x_1$	10000	16/0.101137	13/0.92295	15/0.112958	99/0.740176	99/0.756527
	0.05_1	10000	21/0.120721	18/0.108912	17/0.116297	114/0.853616	113/0.821018
	x_3	100000	12/1.203871	11/1.055571	17/1.621756	112/12.724008	_
	$0.1x_3$	100000	8/0.930625	8/0.823522	14/1.302955	90/9.175859	_
	$0.0x_3$	100000	5/1.185866	4/0.954697	11/1.145516	68/1.102902	_
	$0.05x_3$	100000	7/1.067205	7/0.975836	13/1.653383	84/11.929352	_
$c = 0.99$	x_8	10000	16/0.007971	17/0.003282	17/0.005133	54/0.014425	54/0.013744
	$0.1x_8$	10000	17/0.004639	20/0.005944	21/0.014131	55/0.011574	55/0.014055
	$0.01x_8$	10000	21/0.005776	20/0.004512	18/0.005386	54/0.011493	54/0.013357
	$0.05x_8$	10000	19/0.003483	21/0.005944	18/0.005457	55/0.010536	55/0.011911
	x_5	100000	9/1.305957	7/0.991077	16/1.335521	54/7.588736	_
	$0.1x_5$	100000	5/1.071616	7/1.307488	13/2.042463	36/4.402335	_
	$0.01x_5$	100000	17/1.744009	15/1.402135	13/2.239192	8/2.010338	_
	$0.05x_5$	100000	14/1.549904	12/1.668095	14/2.033192	30/3.972562	_
$c = 0.999$	x_2	10000	9/0.096039	8/0.082131	16/0.121233	90/0.766591	_
	$0.1x_2$	10000	9/0.105176	7/0.077574	13/0.107245	78/0.554128	_
	$0.01x_2$	10000	16/0.139109	14/0.126423	_	66/0.678516	_
	$0.05x_2$	10000	12/0.093105	15/0.138212	16/0.112861	65/0.645718	_
	$-x_2$	100000	7/1.192758	7/1,138663	16/2.167357	5/1.197072	_
	$-0.1x_2$	100000	3/1.003224	3/1.010213	13/1.812344	5/1.358422	_
	$-0.01x_2$	100000	15/1.862809	13/1.958411	13/1.912028	5/0.972271	_
	$-0.05x_2$	100000	6/1.137324	6/1.286585	14/2.329455	8/1.661712	_

Table 5.23: Test results of Chandrasekhar H-equation with respect to number of iterations/CPU time

	Guess	**Dim**	**IHZCG**	**EHZCG**	**FCGM**	**TTPRP**	**MHSCG**
$c = 0.9$	x_1	10000	25/0.176982	18/0.164092	27/0.140762	142/1.094605	143/1.065634
	$0.1x_1$	10000	22/0.179162	23/0.146827	18/0.122463	120/0.880392	120/0.881846
	$0.01x_1$	10000	15/0.142819	21/0.122963	15/0.112958	99/0.740176	99/0.756527
	$0.05x_1$	10000	20/0.124821	21/0.167982	17/0.116297	114/0.853616	113/0.821018
	x_3	100000	11/1.275887	14/1.524367	17/1.621756	112/12.724008	_
	$0.1x_3$	100000	8/1.438136	9/1.230411	14/1.302955	90/9.175859	_
	$0.0x_3$	100000	4/1.616318	5/0.960052	11/1.145516	68/1.102902	_
	$0.05x_3$	100000	7/1.101806	8/0.853667	13/1.653383	84/11.929352	_
$c = 0.99$	x_8	10000	17/0.003056	14/0.002426	17/0.005133	54/0.014425	54/0.013744
	$0.1x_8$	10000	20/0.003977	7/0.001816	21/0.014131	55/0.011574	55/0.014055
	$0.01x_8$	10000	19/0.004971	19/0.004372	18/0.005386	54/0.011493	54/0.013357
	$0.05x_8$	10000	22/0.003519	19/0.005686	18/0.005457	55/0.010536	55/0.011911
	x_5	100000	8/1.336957	8/1.187484	16/1.335521	54/7.588736	_
	$0.1x_5$	100000	6/0.988136	8/1.348793	13/2.042463	36/4.402335	_
	$0.01x_5$	100000	15/1.396006	20/1.792157	13/2.239192	8/2.010338	_
	$0.05x_5$	100000	12/1.718533	16/1.685939	14/2.033192	30/3.972562	_
$c = 0.999$	x_2	10000	8/0.095144	6/0.071114	16/0.121233	90/0.766591	_
	$0.1x_2$	10000	8/0.078559	8/0.078564	13/0.107245	78/0.554128	_
	$0.01x_2$	10000	14/0.097744	18/0.118579	_	66/0.678516	_
	$0.05x_2$	10000	11/0.091375	17/0.110784	16/0.112861	65/0.645718	_
	$-x_2$	100000	7/1.114345	8/1.043873	16/2.167357	5/1.197072	_
	$-0.1x_2$	100000	3/1.104879	3/1.019275	13/1.812344	5/1.358422	_
	$-0.01x_2$	100000	13/1.070642	17/1.665565	13/1.912028	5/0.972271	_
	$-0.05x_2$	100000	7/0.751059	5/1.047656	14/2.329455	8/1.661712	_

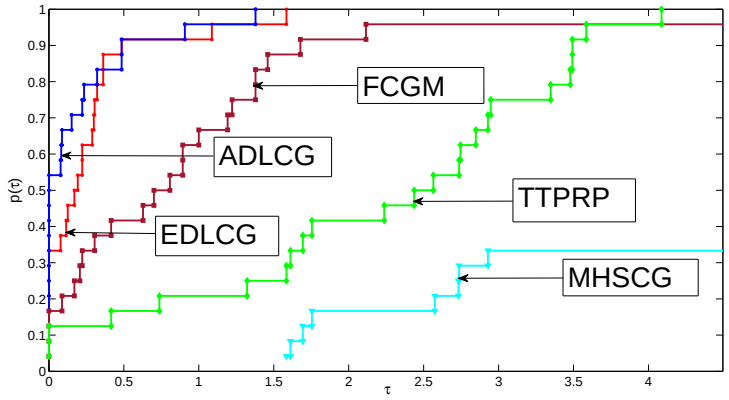

Figure 5.9: Performance profile of EDLCG, ADLCG, FCGM, TTPRP, and MH-SCG methods (for number of iterations)

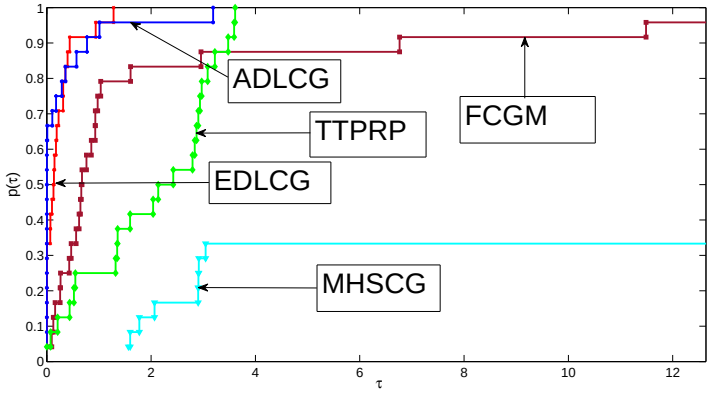

Figure 5.10: Performance profile of EDLCG, ADLCG, FCGM, TTPRP, and MH-SCG methods (for CPU time)

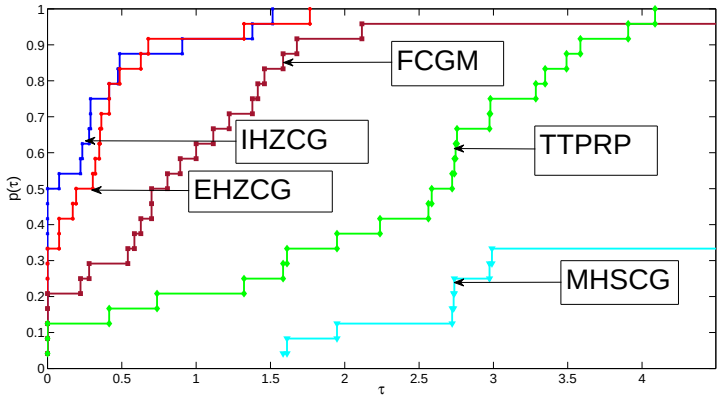

Figure 5.11: Performance profile of IHZCG, EHZCG, FCGM, TTPRP, and MHSCG methods (for number of iterations)

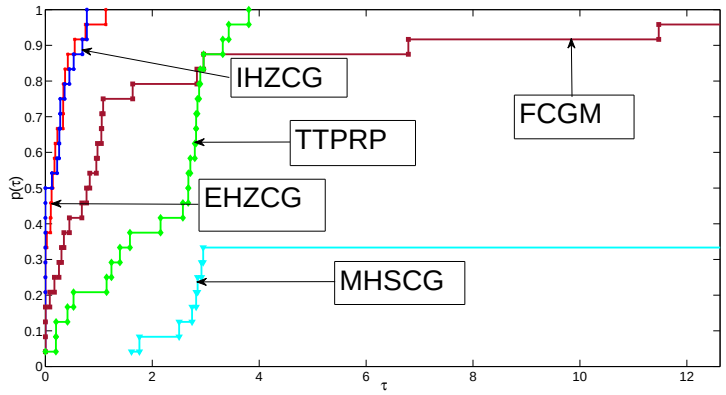

Figure 5.12: Performance profile of IHZCG, EHZCG, FCGM, TTPRP, and MHSCG methods (for CPU time)

Using different dimensions with different values of the parameter c, we observe that our methods were able to solve all of the problems used in the numerical experiment with better results than the other methods.

CHAPTER SIX

SUMMARY, CONCLUSION AND RECOMMENDATIONS

6.1 INTRODUCTION

This chapter gives the summary and conclusion of the entire research, together with suggestions and recommendations for further research.

6.2 SUMMARY

The conjugate gradient (CG) methods are an important class of iterative methods for unconstrained optimization problems. Their low memory requirement and global convergence properties makes them the ideal choice for engineers and mathematicians engaged in solving large-scale system of nonlinear equations.

In chapter one of this research, we introduced the concept of systems of nonlinear equations, some prominent iterative methods for solving them, as well as some areas, where they are applied. The classical conjugate gradient method for solving large-scale system of nonlinear equations was also introduced. In addition, the chapter included statement of the problem, motivation for the study, aim and objectives, scope and limitations of the study as well as some basic definitions of terms used in the write-up.

Chapter two deals with a review of related literature of methods for solving system of nonlinear equations. Large portion of this chapter is dedicated to conjugate gradient methods. In chapter three, details of all the four proposed methods

was presented. This includes derivation of each method, algorithms, and convergence analysis.

We reported numerical results of the experiment carried out for the proposed methods in chapter four. We employed the performance profile developed by Dolan and More (2002) to show the performance and efficiency of our proposed methods in terms of number of iterations and CPU time respectively. The results indicated that the proposed methods are better choice than FCGM, TTPRP and MHSCG methods for solving large-scale system of nonlinear equations.

Lastly, we used chapter five to illustrate application of the proposed methods in solving the popular integral equation of Chandrasekhar type in the radiative transfer problems.

6.3 CONCLUSION

In this research, we proposed four DL-type conjugate gradient methods for large-scale nonlinear systems. Two of the methods are modifications of the classical Dai-Liao (2001) method for unconstrained optimization, while the other two are modifications of the one-parameter Hager-Zhang (2006) method. The first two methods were obtained by employing extensions of some modified secant equations and carrying out eigenvalue analysis in the DL approach. The other two methods were developed by incorporating a modified secant equation in the one-parameter Hager-Zhang (2006) scheme and carrying out eigenvalue study to determine appropriate values for the parameter θ. Our anticipation is to suggest a good CG parameter that will lead to a solution with less computational cost.

By using basic assumptions, we proved the global convergence of the proposed schemes. Extensive numerical results drawn in tables as well as performance profile introduced by Dolan and Moré (2002) were used to demonstrate the performance and efficiency of the suggested methods. All the analysis show that the proposed methods are effective for large-scale nonlinear systems.

6.4 FUTURE RESEARCH

- As the values of the DL and HZ parameters obtained in this research are not the only possible choices, other approaches and techniques can be applied to obtain better numerical results.

- The derivative-free line search proposed by Li and Fukushima (2000A) was used in this work. It would be interesting in the future to study the possibility of developing CG algorithms using nonmonotonic line searches for solving systems of nonlinear equations.

REFERENCES

Al-Baali, M., Spedicato, E. and Maggioni, F. (2014). Broydens Quasi-Newton Methods for a Nonlinear System of Equations and Unconstrained Optimization: A Review and Open Problems. *Optim. Meth. Soft.*, 29(5): 937-954.

Andrei, N. (2008A). 40 Conjugate Gradient Algorithms for Unconstrained Optimization. A Survey on their Definition. *ICI Technical Report No.*13/08: 1-13.

Andrei, N. (2008B). Another Hybrid Conjugate Gradient Algorithm for Unconstrained Optimization. *Numer. Algor.*, 47: 143-156.

Andrei, N. (2008C). A Hybrid Conjugate Gradient Algorithm for Unconstrained Optimization as a Convex Combination of Hesteines-Stiefel and Dai-Yuan. *Stud. Inform. Control*, 17: 143-156.

Andrei, N. (2011). Open Problems in Conjugate Gradient Algorithms for Unconstrained Optimization. *Bull. Malays. Math. Sci. Soc.*, 34(2): 319-330.

Andrei, N. (2010). Accelerated Hybrid Conjugate Gradient Algorithm with Modified Secant Condition for Unconstrained Optimization. *Numer. Algor.* 54: 23-46.

Andrei, N. (2017). Accelerated Adaptive Perry Conjugate Gradient Algorithms Based on the Self-scaling BFGS Update. *Journal of computational and applied mathematics*, 325: 149-164.

Arazm, M.R., Babaie-Kafaki, S. and Ghanbari, R. (2017). An Extended Dai-Liao Conjugate Gradient Method with Global Convergence for Nonconvex Functions. *Glasnik matematicki*, 52(72): 361-375.

Babaie-Kafaki, S., Ghanbari, R. and Mahdavi-Amiri, N. (2010). Two New Conjugate Gradient Methods Based on Modified Secant Equations. *J. Comput. Appl. Math.*, 234(5): 1374-1386.

Babaie-Kafaki, S. (2011). A Hybrid Conjugate Gradient Method Based on a Quadratic Relaxation of the Dai-Yuan Hybrid Conjugate Gradient Parameter. *Optimization (2011), DOI: 10.1080/ 02331934.2011.611512.*

Babaie-Kafaki, S., Fatemi, M. and Mahdavi-Amiri, N. (2011). Two Effective Hybrid Conjugate Gradient Algorithms Based on Modified BFGS Updates. *Numer. Algor.*, 58: 315-331.

Babaie-Kafaki, S. and Ghanbari, R. (2013). A Descent Family of Dai-Liao Conjugate Gradient Methods. *Optimization Methods and Software*, 29(3): 583-591.

Babaie-Kafaki, S. and Ghanbari, R. (2014). The Dai-Liao Nonlinear Conjugate Gradient Method with Optimal Parameter Choices. *European Journal of Operational Research*, 234: 625-630.

BabaieKafaki, S. and Ghanbari, R. (2015). Two Optimal Dai-Liao Conjugate Gradient Methods. *Optimization, A Journal of Mathematical Programming and Operations Research*, 64: 2277-2287.

Babaie-Kafaki, S. and Ghanbari, R. (2016). An Adaptive Hager-Zhang Conjugate Gradient Method. *Filomat*, 30(14): 3715-3723.

Babaie-Kafaki, S. and Ghanbari, R. (2014). A Descent Extension of the Polak-Ribier-Polyak Conjugate Gradient Method. *Computers and Mathematics with Applications*, 68(2014): 2005-2011.

Barzilai, J. and Borwein, J.M. (1988). Two-point Step size Gradient Methods. *IMA J. Numer. Anal.*, 8: 141-148.

Brown, P.N. and Saad, Y. (1994). Convergence Theory of Nonlinear Newton-Krylov Algorithms. *SIAM J. Optim.* 4(2): 297-330.

Broyden, C.G. (1962). A Class of Methods for Solving Nonlinear Simultaneous Equations. *Math. Comput.*, 12: 577-593.

Broyden, C.G. (1970). The Convergence of a Class Double-rank Minimization Algorithms. *Journal of the Institute of Mathematics and its Applications*, 6: 76-90.

Chandrasekhar, S. (1950). Radiative Transfer. *Oxford University Press, London, UK*.

Chandrasekhar, S. and Breen, F.H. (1947). On the Radiative Equilibrium of a stellar Atmosphere. XIX. *The Astrophysical Journal*, 106: 143-144.

Cheng, W. (2009). A PRP type Method for Systems of Monotone Equations. *Mathematical and Computer Modelling*, 50: 15-20.

Dai, Y.H., Han, J.Y., Liu, G.H., Sun, D.F., Yin, H.X. and Yuan, Y.X. (1999). Convergence Properties of Nonlinear Conjugate Gradient Methods. *SIAM J. Optim.*, 10(2): 348-358.

Dai, Y.H. and Liao, L.Z. (2001). New Conjugacy Conditions and Related Nonlinear Conjugate Gradient Methods. *Appl. Math. Optim.*, 43(1): 87-101.

Dai, Y.H. and Yuan, Y. (1999). A Nonlinear Conjugate Gradient Method with a strong Global Convergence Property. *SIAM J. Optim.*, 10: 177-182.

Dai, Y.H. and Yuan, Y. (2001). An Efficient Hybrid Conjugate Gradient Method for Unconstrained Optimization. *Ann. Oper. Res.*, 103: 33-47.

Dai, Y.H. and Yuan, Y. (1998). A Class of Globally Convergent Conjugate Gradient Methods. *Research Report ICM-98-030, Institute of Computational Mathematics and Scientific/Engineering Computing, Chinese Academy of Sciences*.

Dai, Y.H. and Yuan, Y. (2003). A Class of Globally Convergent Conjugate Gradient Methods. *Sci. China Ser. A*, 46: 251-261.

Dai, Y.H. and Yuan, Y. (1998). Extension of a Class of Nonlinear Conjugate Gradient Methods. *Research Report ICM-98-049, Institute of Computational Mathematics and Scientific/Engineering Computing, Chinese Academy of Sciences*.

Dai, Y.H. and Yuan, Y. (2001). A Three-Parameter Family of Hybrid Conjugate Gradient Method. *Math. Comp.*, 70: 1155-1167.

Dai, Z., Chen, X. and Wen, F. (2015). A Modified Perrys Conjugate Gradient Method-Based Derivative-free Method for Solving Large-scale Nonlinear Monotone Equation. *Applied Mathematics and Computation*, 270: 378-386.

Daniel, J.W. (1967). The Conjugate Gradient Method for Linear and Nonlinear Operator Equations. *SIAM J. Numer. Anal.*, 4: 10-26.

Dolan, E.D. and Mor*é*, J.J. (2002). Benchmarking Optimization Software with Performance Profiles. *Math. Program,* 91(2): 201-2013.

Fletcher, R. and Reeves, C.M. (1964). Function Minimization by Conjugate Gradients. *Computer Journal*, 7: 149-154.

Fletcher, R. (1970). A New Approach to Variable Metric Algorithms. *Computer J*, 13: 317-322.

Fletcher, R. (1987). Practical Methods of Optimization vol. 1: Unconstrained Optimization. *John Wiley Sons, New York*.

Ford, J.A. Narushima, Y. and Yabe, H. (2008). Multi-Step Nonlinear Conjugate Gradient Methods for Unconstrained Minimization. *Comput. Optim. Appl.*, 40(2): 191-216.

Ford, J.A. and Moghrabi, I.A. (1994). Multi-Step Quasi-Newton Methods for Optimization. *J. Comput. Appl. Math.*, 50(13): 305-323.

Ford, J.A. and Moghrabi, I.A. (1996). Using Function-Values in Multi-Step Quasi-Newton Methods. *J. Comput. Appl. Math.*, 66(12): 201-211.

Gilbert, J.C. and Nocedal, J. (1992). Global Convergence Properties of Conjugate Gradient Methods for Optimization. *SIAM J. Optim.*, 2: 21-42.

Goldfarb, D. (1970). A Family of Variable Metric Methods Derived by Variation Mean. *Mathematics of Computation*, 23: 23-26.

Gomes-Ruggiero, M.A., Martnez, J.M. and Moretti, A.C. (1992). Comparing Algorithms for Solving Sparse Nonlinear Systems of Equations. *SIAM Journal on Scientific and Statistical Computing*, 13:(2) 459-483.

Griewank, A. (1986). The Global Convergence of Broyden-like Methods with Suitable Linesearch. *J. Austral. Math. Soc. Ser.*, 28: 75-92.

Grippo, L., Lampariello, F. and Lucidi, C. (1986). A Nonmonotone Linesearch Technique for Newtons Method. *SIAM J. Numer. Anal.*, 23: 707-716.

Hager, W.W. and Zhang, H. (2005). A New Conjugate Gradient Method with Guaranteed Descent and an Efficent Linesearch. *SIAM J. Optim.*, 16: 170-192.

Hager, W.W. and Zhang, H. (2006). A Survey of Nonlinear Conjugate Gradient Methods. *Pac. J. Optim.*, 2(1): 35-58.

Hestenes, M.R. and Stiefel, E.L. (1952). Methods of Conjugate Gradients for Solving Linear Systems. *J. Research Nat. Bur. Standards*, 49: 409-436.

Hively, G.A. (1978). On a Class of Nonlinear Integral Equations arising in Transport Theory. *SIAM Journal on Mathematical Analysis*, 9(5): 787-792.

Hu, Y.F. and Storey, C. (1991). Global Convergence Result for Conjugate Gradient Methods. *J. Optim. Theory Appl.*, 71: 399-405.

James, M.O. and Werner, C.R. (1966). On Discretization and Differentiation of Operators with Application to Newtons Method. *SIAM Journal on Numerical Analysis*, 3(1): 143-156.

Kelley, C.T. (1995). Iterative Methods for Linear and Nonlinear Equations. *Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics SIAM, Philadelphia, Pa, USA*, 16.

Koorapetse, M., Kaelo, P., and Offen, E.R. (2018). A Scaled Derivative-free Projection Method for Solving Nonlinear Monotone Equations. *Bulletin of the Iranian Mathematical Society*.

La Cruz, W. and Raydan, M. (2003). Nonmonotone Spectral Methods for Large-scale Nonlinear Systems. *Optim. Methods Softw.*, 18: 583-599.

La Cruz, W. (2017). A Spectral Algorithm for large-scale Systems of Nonlinear Monotone Equations. *Numer Algor DOI 10.1007/s1107s-017-0299-8*.

La Cruz, W., Martnez, J.M. and Raydan, M. (2006). Spectral Residual Method without Gradient Information for Solving Large-scale Nonlinear Systems of Equations. *Math. Comput.*, 75: 1429-1448.

Li, D.H. and Fukushima, M. (2000A). A Globally and Superlinearly Convergent Gauss-Newton-based BFGS Method for Symmetric Nonlinear Equations. *SIAM Journal on Numerical Analysis*, 37(1): 152-172.

Li, D.H. and Fukushima, M. (2000B). A Derivative-free Linesearch and Global Convergence of Broyden-like Method for Nonlinear Equations. *Optim. Methods Softw.*, 13: 583-599.

Li, D.H. and Fukushima, M. (2001A). A Modified BFGS Method and its Global Convergence in Nonconvex Minimization. *J. Comput. Appl. Math.*, 129(12): 15-35.

Li, D.H. and Cheng, W.Y. (2007). Recent Progress in the Global Convergence of Quasi-Newton Methods for Nonlinear Equations. *Hokkaido Math. J.*, 36: 729-743.

Li, D.H. and Fukushima, M. (2001B). On the Global Convergence of the BFGS Method for Nonconvex Unconstrained Optimization problems. *SIAM J. Optim.*, 11: 1054-1064.

Li, G., Tang, C. and Wei, Z. (2007). New Conjugacy Condition and Related New Conjugate Gradient Methods for Unconstrained Optimization. *J. Comput. Appl. Math.*, 202(2): 523-539.

Liu, Y. and Storey, C. (1991). Efficient Generalized Conjugate Gradient Algorithms, Part 1: *Theory, J. Optim. Theory Appl.*, 69: 129-137.

Liu, D.Y. and Shang, Y.F. (2010). A New Perry Conjugate Gradient Method with the Generalized Conjugacy Condition. *In: Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on Issue Date:* (2010): 10-12.

Liu, D.Y. and Xu, G.Q. (2011). A Perry Descent Conjugate Gradient Method with Restricted Spectrum. *Optimization Online, Nonlinear Optimization (Unconstrained Optimization)*. 1-19.

Liu, J.K. (2016). Derivative-free Spectral PRP Projection Method for Solving Nonlinear Monotone Equations with Convex Constraints. *Math Numer Sin*, 38: 113-124.

Liu, J.K. and Feng, Y.M. (2017). A Norm Descent Derivative-free Algorithm for Solving large-scale Nonlinear Symmetric Equations.*doi.org/10.1016/j.cam.2018.05.006*.

Livieris, I.E. and Pintelas, P. (2012). Globally Convergent Modified Perrys Conjugate Gradient Method. *Appl. Math. Comput.*, 218: 9197-9207.

Martnez, J.M. (1990). A Family of Quasi-Newton Methods for Nonlinear Equations with Direct Secant Updates of Matrix Factorizations. *SIAMJ. Numer. Anal.*, 27: 1034-1049.

Martnez, J.M. (2000). Practical Quasi-Newton Methods for Solving Nonlinear Systems. *J. Comput. Appl. Math.* 124: 97-121.

Natasa, K. and Zorana, L. (2002). Newton-like Method with Modification of the Right-hand-side vector. *Math. Comput.* 71: 237-250.

Nazareth, J.L. (1998). Conjugate Gradient Methods. *In Floudas C, Pardalos P, Editors. Encyclopedia of Optimization. Boston (MA), Dordrecht, The Netherlands: Kluwer Academic Publishers;* 319-323.

Nazareth, J.L. (1998). Conjugate Gradient Methods, Encyclopedia of Optimization. *C. Floudas and P. Pardalos, Eds., Kluwer Academic Publishers, Boston*.

Nocedal, J. (1991). Theory of Algorithms for Unconstrained Optimization. *Acta Numerica*, 1: 199-242.

Nocedal, J. and Wright, S.J. (2006). Numerical Optimization. *Springer, NewYork*.

Perry, A. (1978). A Modified Conjugate Gradient Algorithm. *Oper. Res. Tech. Notes*, 26(6): 1073-1078.

Polak, B.T. (1969). The Conjugate Gradient Method in Extreme Problems. *USSR Comput Math Math Phys*, 4: 94-112.

Polak, E. and Ribire, E. (1969). Notesurla Convergence de Directions Conjuges. *Rev. Francaise Imformat Recherche Opertionelle*, 16: 35-43.

Powell, M.J.D. (1977). Restart Procedures of the Conjugate Gradient Method. *Math. Prog.*, 2: 241-254.

Powell, M.J.D. (1984). Nonconvex Minimization Calculations and the Conjugate Gradient. *Numer. Anal.*, 1066: 122-141.

Reid, J.K. (1971). On the Method of Conjugate Gradients for the Solution of Large Systems of Linear Equations. *J.K. Reid (ed.) , Large Sparse Sets of Linear Equations , Acad. Press.*

Shanno, D.F. (1970). Conditioning of Quasi-Newton Methods for Function Minimization. *Math. Comput*, 24: 647-656.

Shin, B.C., Dirvashi, M.T. and Kim, C.H. (2010). A Comaprison of the Newton-Krylov Method with high order Newton-like Methods to Solve Nonlinear Systems. *Appl. Math. Comput.*, 217: 3190-3198.

Solodov, M.V. and Svaiter, B.F. (1999). A Globally Convergent Inexact Newton Method for Systems of Monotone Equations. *In: Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods, Springer*, 355-369.

Steihuag, T. and Sara, S. (2013). Rate of Convergence of Higher order Methods. *Applied Numerical Mathematics*, 67: 230-242.

Sun, W. and Yuan, Y.X. (2006). Optimization Theory and Methods: Nonlinear Programming. *Springer, New York*.

Sun, M., Wang, X. and Feng, D. (2017). A Family of Conjugate Gradient Methods for Large-scale Nonlinear equations. *Journal of Inequalities and Applications*, 236: 1-8.

Touati-Ahmed, D, and Storey, C. (1990). Efficient Hybrid Conjugate Gradient Techniques. *J. Optim. Theory Appl.*, 64: 379-397.

Waziri, M.Y., Leong, W.J., Hassan, M.A. and Monsi, M. (2010). Jacobian Computation-free Newton's Method for System of Nonlinear Equations. *Journal of Numerical Mathematics and Statistics*, 2(1): 54-63.

Waziri, M.Y., Leong, W.J. and Hassan, M.A. (2011). Jacobian free-Diagonal Newton's Method for Nonlinear Systems with Singular Jacobian. *Malaysian Journal of Mathematical Sciences*, 5(2): 241-255.

Waziri, M.Y. and Sabi'u, J. (2015). A Derivative-free Conjugate Gradient Method and its Global Convergence for Solving symmetric nonlinear equations. *Journal of Mathematics and Mathematical Sciences*, 1-8.

Wei, Z., Li, G. and Qi, L. (2006). New Quasi-Newton Methods for Unconstrained Optimization Problems. *Appl. Math. Comput.*, 175(2): 1156-1188.

Wenyu, S. and Yuan, Y.X. (2006). Optimization Theory and Method, Nonlinear Programming. *Library of Congress Control*. 1-688.

Xiao, Y. and Zhu, H. (2013). A Conjugate Gradient Method to Solve Convex Constrained Monotone Equations with Epplications in Compressive Sensing. *J. Math. Anal. Appl.*, 405: 310-319.

Yabe, H. and Takano, M. (2004). Global Convergence Properties of Nonlinear Conjugate Gradient Methods with Modified secant condition. *Comput. Optim. Appl.*, 28(2): 203-225.

Yao, S., Lu, X., and Qin, B. (2014). A Modified Conjugacy Condition and Related Nonlinear Conjugate Gradient Method. *Mathematical Problems in Engineering*, 2014: 1-9.

Yasushi, N. and Hiroshi, Y. (2012). Conjugate Gradient Methods Based on Secant Conditions that Generate Descent Directions for unconstrained optimization. *Journal of Computational and Applied Mathematics*, 236: 4303-4317.

Yu, G. (2010). A Derivative-free Method for Solving Large-scale Nonlinear Systems of Equations. *J. Ind. Manag. Optim.*, 6: 149-160.

Yu, G. (2011). Nonmonotone Spectral Gradient-type Methods for Large-scale Unconstrained Optimization and Nonlinear systems of equations. *Pac. J. Optim*, 7: 387-404.

Yuan*, G. and Lu, X. (2008). A New Backtracking Inexact BFGS Method for Symmetric Nonlinear Equations. *Journal of Computers and Mathematics with Applications*, 55: 116-129.

Yuan, G. and Zhang, M. (2015). A three-terms Polak-Ribier*é*-Polyak conjugate gradient algorithm for large-scale nonlinear equations. *Journal of Computational and Applied Mathematics*, 286(2015): 186-195.

Yuan, G. Meng, Z., and Li, Y. (2016). A Modified Hestenes and Stiefel Conjugate Gradient Algorithm for large-scale Nonsmooth Minimizations and Nonlinear Equations. *J Optim Theory Appl* 168: 129-152.

Yuan. N. (2017). A Derivative-free Projection Method for Solving convex constrained Monotone Equations.*ScienceAsia* 43: 195-200.

Yuan, Y.X. (1991). A modified BFGS Algorithm for Unconstrained Optimization. *IMA J. Numer. Anal.*, 11: 325-332.

Zhang, J.Z., Deng, N.Y. and Chen, L.H. (1999). New Quasi-Newton Equation and Related Methods for Unconstrained Optimization. *J. Optim. Theory Appl*, 102: 147-167.

Zhang, J.Z. and Xu, C.X. (2001). Properties and Numerical Performance of Quasi-Newton Methods with Modified Quasi-Newton equations. *J. Comput. Appl. Math.*, 137: 269-278.

Zhang, H. and Hager, W. (2004). A Nonmonotone Line Search Technique and its Application to Unconstrained Optimization. *SIAM J. Optim.*, 4: 1043-1056.

Zhang, L. and Zhou, W. (2006). Spectral Gradient Projection Method for Solving Nonlinear Monotone Equations. *J. Comput Appl. Math.*, 196: 478-484.

Zhou, W. and Zhang, L. (2006). A Nonlinear Conjugate Gradient Method Based on the MBFGS Secant Condition. *Optim. Methods Softw.*, 21(5): 707-714.

Zhou, W. and Shen, D. (2014). An Inexact PRP Conjugate Gradient Method for Symmetric Nonlinear Equations. *Numerical Functional Analysis and Optimization*, 35(3): 370-388.

Zhou, W. and Shen, D. (2015). Convergence Properties of an Iterative Method for Solving Symmetric Non-linear Equations. *Journal of Optimization Theory and Applications*, 164(1): 277-289.

Zhou, W. and Li, D. (2007). Limited Memory BFGS Method for Nonlinear Monotone Equations. *J. Comput. Math.*, 25(1): 89-96.

Appendix A

LIST OF PUBLICATIONS

i. M. Y. Waziri, K. Ahmed, and J. Sabi'u, A Family of Hager-Zhang Conjugate Gradient Methods for System of Nonlinear Equations, Applied Mathematics and Computations(ELSEVIER), **PUBLISHED** April 2019.

ii. M. Y. Waziri, K. Ahmed, and J. Sabi'u, A Dai-Liao Conjugate Gradient Method via Modified Secant Equations For System of Nonlinear Equations, Arabian Journal of Mathematics (SPRINGER), **PUBLISHED** June 2019.

iii. M. Y. Waziri, K. Ahmed, and J. Sabi'u, Enhanced Family of Dai-Liao Conjugate Gradient Methods for Nonlinear Systems of Equations, Manuscript Number: JOTA-D-19-00551, Journal of Optimization Theory and Applications (SPRINGER), Round 1 Review.

Appendix B

Matlab codes for EDLCG, ADLCG, IHZCG and EHZCG Methods

We implemented the codes of the iterative methods described in Chapter 3 using Matlab computer language.

i. **Matlab code for EDLCG Method.**

```
function EDLCG(G,Guess)
% solves the non-linear vector equation  F(x)=0
% solution = The solution to F(x)=0
tic
t = cputime;
tol=0.00001;
maxit=1000;
x0 = Guess;
p=0;
f =feval(G,x0);
mag_of_f=sqrt(sum(f.^2));
d0=-f;
while ( p < maxit and mag_of_f >tol  )
           f =feval(G,x0);
```

```
 sg1=0.0001;
        sg2=sg1;
        n1=1/(p+1)^2;
        m=0;
        r=0.2;
 while (0.5*(norm(feval(G,\text{x0}+(r)^m*d0)))^20.5*(norm(f))^2-s
    m=m+1;

 end
alp=(r)^m;

x1=x+alp*d0;
s=x1-x0;
f1=feval(G,x1);
y=f1-f;
q=0.001;
c=0.001;
g=(c*(norm(f1)^q)*s);
e=2*((0.5*(norm(f)^2))-(0.5*(norm(f1)^2)))+s'*(f+f1);
i=max(e,0);
z=(y+(i/(s'*s))*s)+g;
w=(z'*z)/(s'*z);
v=(s'*z)/(s'*s);
h=0.9;
k=-0.65;
t=(h*w)-(k*v);
hs=(f1'*z)/(d0'*z);
dl=(f1'*s)/(d0'*z);
b=hs-(t*dl);
d=-f1+b*d0;
x0=x1;
d0=d;
p=p+1;
```

```
          mag_of_f=sqrt(sum(f1.^2));
 end
 x0
  mag_of_f1
%e=(cputime-t)
toc
p
```

ii. **Matlab code for ADLCG Method.**

```
function ADLCG(G,Guess)
% solves the non-linear vector equation  F(x)=0
% solution = The solution to F(x)=0
tic
t = cputime;
tol=0.00001;
maxit=1000;
x0 = Guess;
p=0;
f =feval(G,x0);
mag_of_f=sqrt(sum(f.^2));
d0=-f;
while ( p < maxit and mag_of_f >tol  )
            f =feval(G,x0);
            sg1=0.0001;
                  sg2=sg1;
                  n1=1/(p+1)^2;
                  m=0;
                  r=0.2;
    while (0.5*(norm(feval(G,\text{x0}+(r)^m*d0)))^2 > 0.5*(norm(f))^2-sg1*
              m=m+1;
      end
     alp=(r)^m;
     x1=x+alp*d0;
     s=x1-x0;
     f1=feval(G,x1);
     y=f1-f;
     e=2*((0.5*(norm(f)^2))-(0.5*(norm(f1)^2)))+s'*(f+f1);
     i=max(e,0);
     z=(y+2*0.9*(i/(s'*))*s);
```

```
    w=(z'*z)/(s'*z);
    v=(s'*z)/(s'*s);
    h=0.9;
    k=-0.6;
    t=(h*w)-(k*v);
    hs=(f1'*z)/(d0'*z);
    dl=(f1'*s)/(d0'*z);
    b=hs-(t*dl);
    d=-f1+b*d0;
    x0=x1;
    d0=d;
    p=p+1;
    mag_of_f=sqrt(sum(f1.^2));
 end
 x0
   mag_of_f
%e=(cputime-t)
toc
```

iii. **Matlab code for IHZCG Method.**

```
function IHZCG(G,Guess)
% solves the non-linear vector equation  F(x)=0
% solution = The solution to F(x)=0
tic
t = cputime;
tol=0.00001;
maxit=1000;
x0 = Guess;
p=0;
f =feval(G,x0);
mag_of_f=sqrt(sum(f.^2));
d0=-f;
while ( p < maxit and mag_of_f >tol  )
         f =feval(G,x0);
         sg1=0.0001;
               sg2=sg1;
               n1=1/(p+1)^2;
               m=0;
               r=0.2;
     while (0.5*(norm(feval(G,\text{x0}+(r)^m*d0)))^2 > 0.5*(norm(f))^2-sg1*
               m=m+1;
          end
        alp=(r)^m;
        x1=x+alp*d0;

        s=x1-x0;
        f1=feval(G,x1);
        y=f1-f;
        v=1.5;
        w=-0.5;
```

```
        h=v-(w*((s'*y)^2/(y'*y)*(s'*s)));
        hs=(f1'*y)/(d0'*y);
        dl=(y'*y)*(f1'*s)/(d0'*y)^2;
        b=hs-(h*dl);
        d=-f1+b*d0;
        x0=x1;
        d0=d;
        p=p+1;
        mag_of_g=sqrt(sum(f1.^2));
 end
x0
  mag_of_f
%e=(cputime-t)
toc
p
```

iv. **Matlab code for EHZCG Method.**

```
function EHZCG(G,Guess)
% solves the non-linear vector equation  F(x)=0
% solution = The solution to F(x)=0
tic
t = cputime;
tol=0.00001;
maxit=1000;
x0 = Guess;
p=0;
f =feval(G,x0);
mag_of_f=sqrt(sum(f.^2));
d0=-f;
while ( p < maxit and mag_of_f >tol  )
         f =feval(G,x0);
         sg1=0.0001;
               sg2=sg1;
               n1=1/(p+1)^2;
               m=0;
               r=0.2;
      while (0.5*(norm(feval(G,\text{x0}+(r)^m*d0)))^2 > 0.5*(norm(f))^2-sg
               m=m+1;
          end
        alp=(r)^m;
        x1=x+alp*d0;

        s=x1-x0;
        f1=feval(G,x1);
        y=f1-f;
        v=1.6;
        w=-0.65;
```

```
        e=6*((0.5*(norm(f)^2)-(0.5*(norm(f1)^2)))+3(s'*(f+f1));
        i=max(e,0);
        z=(y+0.01*(i/(s'*d0))*d0);
        h=v-(w*((s'*z)^2/(z'*z)*(s'*s)));
        hs=(f1'*z)/(d0'*z);
        dl=(z'*z)*(f1'*d0)/(d0'*z)^2;
        b=hs-(h*dl);
        d=-f1+b*d0;
        x0=x1;
        d0=d;
        p=p+1;
        mag_of_g=sqrt(sum(f1.^2));
 end
  x0
  mag_of_f
%e=(cputime-t)
toc
p
```