Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon
College of Education for Pure Science
Department of Mathematics

Applying Line Search Techniques to Solve Non Linear Systems of Equations

A Thesis

Submitted to the Council of the College of Education for Pure Sciences, University of Babylon in Partial Fulfillment of the Requirements for the Degree of Master in Education / Mathematics

By

Doaa Hamzah Allawi Msarbat

Supervised by

Prof. Dr. Mushtak A.K. Shiker Al-Jenabi

2023 A.D **1445 A.H**

Abstract

This thesis deals with some new algorithms in line search technique. One of the most significant iterative approaches used to solve nonlinear systems of equations is the line search technique. The line search method main aim is determine the step length in a given direction.

For the purpose of solving nonlinear systems of equations, three new algorithms of the line search technique are introduced. The first algorithm combine a monotone technique into a modified line search algorithm. The second algorithm combine line search technique, modified spectral gradient method and projection method. Eventually, the third and the last algorithm combine line search technique, conjugate gradient technique and projection technique.

The goal of these algorithms is to reduce the CPU time, N_i and N_f and make the approach more effective. Global convergence of these algorithms has been demonstrated under standardized conditions. The new algorithms are shown to be promising for solving nonlinear systems of equations by preliminary numerical results.

Republic of Iraq
Ministry of Higher Education
and Scientific Research
University of Babylon
College of Education for Pure Sciences
Department of Mathematics

Applying Line Search Techniques to Solve Non Linear Systems of Equations

A Thesis

Submitted to the Council of the College of Education for Pure Sciences, University of Babylon in Partial Fulfilment of the Requirements for the Degree of Master in Education / Mathematics.

By

Doaa Hamzah Allawi Msarbat

Supervised by

Prof.Dr.Mushtak A.K. Shiker Al-Jenabi

2023 A.D. **1444 A.H.**

بسم الله الرحمن الرحيم

(يرفع الله الذين آمنوا منكم والذين أوتوا العلم
درجات والله بما تعملون خبير)

صدق الله العلي العظيم

سورة المجادلة الآية ١١

Dedication

To the force, My Dearest father whose support paved my way to successes.

To the big heart and the source of kindness, My Mother whose prayers make me hopeful in life.

To those who offered me unconditional, unlimited love and support and walked with me towards the dream (my sisters, my brother and his family).

To my soul mate who accompanied me every step to complete this journey.

To those who have always supported me and with whom I had the happiest times, my friends.

For everyone who supported me even with a word, to all of you, I dedicate this humble work.

Acknowledgments

Praise is to Allah, Lord of the whole creation and peace is upon his messenger Prophet Mohammad and his relatives.

I would like to express my sincere appreciation and deep gratitude to my supervisor **Prof. Dr. Mushtak A.K. Shiker Al-Jenabi** for suggesting this project and for his professional guidance and unlimited support.

I would like express my sincere gratitude to the Deanship of the College of Education for Pure Sciences and the presidency and professors of the Department of Mathematics at the University of Babylon for supporting me to reach this stage.

My sincere gratitude, love, and thanks to my family for their generous advice, and support during the study.

At last but not least special gratitude to my friends and everyone who gave me a hand of support.

Contents

Dedication i

Acknowledgments ii

List of Figures **v**

List of Tables **vi**

Abstract ix

Introduction 1

1 Basic Definitions and Concepts **4**

1.1 Introduction 5

1.2 Basic Definitions and Concepts 5

2 Nonlinear Systems of Equations **16**

2.1 Introduction 17

2.2 Some Methods for Solving Nonlinear Systems 18

2.2.1 Newton's Method 18

2.2.2 Broyden's Method 26

3 Line Search Techniques **32**

3.1 Introduction 33

3.2 Step Length [9] 34

3.2.1 The Wolfe Conditions [9] 35

3.2.2 The Goldstein Conditions [9] 38

3.3 The Framework of Line Search Methods 39

3.3.1 General Algorithm of Line Search technique . . 40
3.4 Line Search Algorithm for Solving Non Linear Systems of Equations 41
3.4.1 The Proposed Algorithm 42
3.4.2 Convergence Property 44
3.4.3 Numerical Results 48

4 Modified Technique to Solve Non Linear Systems of Equations 56
4.1 Introduction 57
4.2 The Proposed Algorithm 58
4.2.1 Convergence Property 60
4.3 Numerical Resuls 62

5 A Modified Conjugate Gradient Technique to Solve Non Linear Systems of Equations 70
5.1 Introduction 71
5.2 The Proposed Algorithm 72
5.2.1 Convergence Property 74
5.3 Numerical Resuls 77

6 Conclusions and Future Works 84
6.1 Conclusions 85
6.2 Future Works 86

References 87

List of Figures

3.1 The Ideal Step Length 35

3.2 Sufficient Decreases Condition 36

3.3 The Curvature Condition 37

3.4 Step Lengths Satisfying Wolfe Condition 38

3.5 The Goldstein Conditions 39

3.6 Flowchart of Line Search Algorithm 41

3.7 Flowchart of Proposed Algorithm 43

4.1 Flowchart of Proposed Algorithm 59

5.1 Flowchart of Proposed Algorithm 73

List of Tables

1 List of Symbols and Abbreviations vii

2.1 Numerical Results of Example 2.2.1.2 24

3.1 Numerical Results (N_i, N_f) 50

3.2 Numerical Results (CPU time) 53

4.1 Numerical Results (N_i, N_f) 64

4.2 Numerical Results (CPU time) 67

5.1 Numerical Results (N_i, N_f) 78

5.2 Numerical Results (CPU time) 81

Table 1: List of Symbols and Abbreviations

Symbols and Abb.	**Description**
$f(x)$	Objective Function
$F(x)$	The Nonlinear System of Equations
f_i	The Nonlinear Real Functions
$Minf(x)$	Minimum of Objective Function
$Maxf(x)$	Maximum of Objective Function
F	Feasible Set
$g(x)$	The Inequality Constraint
$h(x)$	The Equality Constraint
$\{\mathrm{x}_k\}$	The Iterative Sequence
S	The Convex Set
$\nabla f(\mathrm{x})$	The Gradient of f
$H_f(\mathrm{x})$	The Hessian of f
J(x)	The Jacobean Matrix
$Q(x)$	Quadratic Function
R_n	Lagrange Reminder
X	Topological Space
G	Set of Topological Space
d	The Descent Direction of f
d_k	The Steepest Direction
L	Lipschitz Constant
$f'(x)$	The First Derivative of f for x

$f''(x)$	The Second Derivative of f for x
x_k	The Current Iterate
x_{k+1}	The Next Iterate
x_0	The Initial Vector
B	Approximation Matrix
B^{-1}	The Inverse of Approximation Matrix
α_k	The Step Length
$\phi(.)$	The Univariate Function
T	The Transformation
N_i	The Number of Iterations
N_f	The Number of Functions Evaluations
Ω	Closed Convex Set
P_Ω	The Projection Operator
$\theta, \lambda, \delta, \sigma, \rho, \beta, \gamma, \mu, \xi$	The Parameters

s.t.	Such That
Iff	If and Only If
LC	Lipschitz Continuous
BFGS	Broyden, Fletcher, Golfarb and Shanno
SGPM	Spectral Gradient Projection Method
CGM	Conjugate Gradient Method
CGDM	Conjugate Gradient Descent Method

Abstract

This thesis deals with some new algorithms in line search technique. One of the most significant iterative approaches used to solve nonlinear systems of equations is the line search technique. The line search method main aim is determine the step length in a given direction.

For the purpose of solving nonlinear systems of equations, three new algorithms of the line search technique are introduced. The first algorithm combine a monotone technique into a modified line search algorithm. The second algorithm combine line search technique, modified spectral gradient method and projection method. Eventually, the third and the last algorithm combine line search technique, conjugate gradient technique and projection technique.

The goal of these algorithms is to reduce the CPU time, N_i and N_f and make the approach more effective. Global convergence of these algorithms has been demonstrated under standardized conditions. The new algorithms are shown to be promising for solving nonlinear systems of equations by preliminary numerical results.

Introduction

Optimization is one of the most important and effective instruments in our everyday life. Finding the optimal value that offer the minimum or maximum for the objective function is the goal of optimization. Optimization algorithms are a basic and effective way to obtain the solution, typically with the aid of a computer. [25]

Additionally, linear and nonlinear optimization problems come in two types. There are several applications for the linear optimization problem, but there are also many problems that cannot be solved in a linear type. This is where nonlinear optimization problems play a crucial role. There are real problems that may be categorized as nonlinear optimization problems including economics, financial engineering, and computer sciences [29].

There will be a focus on thorough unconstrained optimization methods where the objective function that depends on unconstrained variables is decreased. The most crucial action here is using the optimization technique to discover the solution because if the employed model is simple, it cannot provide a clear perspective of the problem, and if it is too complicated, it may not provide the answer in an understandable manner. No method exists that can be used to solve every optimization problem. Here in lies the crucial role that researchers or users must play in order to select the best algorithm for each problem that may be solved fast or slowly [21].

Numerous fields that have incorporated various domains and

characteristics, including nonlinear systems, and their solutions, are of utmost importance. The fact that most physical systems are nonlinear makes it an important area of mathematics and physical research. It also has applications in engineering, , and economics, etc. [40].

Many researchers have concentrated on supplying suitable methods to solve nonlinear systems, hence certain well-known algorithms have been suggested as providing such ways and means. These include, but are not limited to, optimization problems. The Newton technique is considered a smooth approach to obtain approximate solutions of equations and is one of the best numerical methods that employ the iterative method to solve these systems.

Consider the following optimization unconstrained problem:

$$minf(x), \quad x \in R$$

where f is the objective function. To solve this problem, many researchers have suggested different methods for example; Toint introduced a line search in (1982) to locate a lower value of the objective function at each iteration [32]. In (2020) a new non-monotone adaptive trust region with fixed step length for unconstrained optimization is proposed [33]. In (2022) a modification of the CG method has been proposed under strong Wolfe line research and this method has better ability when solving unconstrained optimization, also a derivative-free H-S type method is proposed to solve large-scale nonlinear equations [12, 13].

In (2023), it was proposed an optimal value for the scaled Perry (CG) method, with aims to solve monotone nonlinear equations [28],

and others methods [10, 8, 31] for solving unconstrained optimization and nonlinear systems of equations. This work focuses on line search methods and how to utilize them to solve nonlinear systems of equations. This thesis is illustrated in six chapters.

First chapter explains several definitions related to constrained and unconstrained optimization.

Second chapter consists of two sections. Section one includes an introduction about nonlinear systems of equations. Section two includes some approaches that used to solve these systems, such as Newton's method and Broyden's method with examples. Also, some advantage and disadvantage of them.

Third chapter contains four sections:

- first section, the line search method is explained in detail.

- the second section, the step length, Wolfe and Goldstein conditions are discussed.

- third section, the framework of the line search method and its algorithm is introduced.

- fourth section, a new algorithm using a line search technique is proposed for solving nonlinear systems of equations.

Fourth chapter, a new line search algorithm modified of (SGPM) is introduced for solving nonlinear systems of equations.

Fifth chapter introduces a new algorithm combines three different techniques: line search, conjugate gradient, and projection for solving nonlinear systems of equations. While chapter six includes conclusions and future works.

Chapter 1

Basic Definitions and Concepts

1.1 Introduction

In this chapter, a list of all the required definitions and basic concepts of optimization, nonlinear equations and line search techniques needed in subsequent chapters for the thesis are introduced [2, 14, 19, 22, 25, 24].

1.2 Basic Definitions and Concepts

Definition 1.2.1 Optimization

Optimization is achieving the best outcome given specific circumstances. Any system requires several decisions, all of these decisions ultimately aim to either maximize the anticipated benefit or reduce the amount of work needed. Mathematically, the process of finding the solutions that give the maximum or minimum value of a function.

Definition 1.2.2 Objective function

An objective function $f(x)$ is the primary goal of the model, which is either to be minimized or maximized. For example, in a manufacturing process, the objective can be to maximize profit or reduce cost.

Definition 1.2.3 Constraints

The constraints are mathematical expression of the limitation on the fulfillment of the objectives.

Definition 1.2.4 Feasible Solution

A feasible solution is a solution that satisfies all of the constraints.

Definition 1.2.5 Feasible set

A feasible set is a set of all feasible solutions.

Definition 1.2.6 Optimal Solution

An optimal solution is a feasible solution that produces the best objective function value.

Definition 1.2.7 Global Maximizer

Let $x^* \in F$ s.t. F be a feasible set, then a point x^* is a global maximizer if:

$$f(x^*) \geq f(x), \quad \forall x \in F \tag{1.1}$$

x^* is a strict global maximizer if $x \neq x^*$ and

$$f(x^*) > f(x). \tag{1.2}$$

Definition 1.2.8 Local Maximizer

Let $x^* \in F$ s.t. F be a feasible set, then a point x^* is a local maximizer if $\exists N, N$ is an open neighborhood of x^* s.t.

$$f(x^*) \geq f(x), \forall x \in N \tag{1.3}$$

x^* is a strict local maximizer if $x \neq x^*$ and

$$f(x^*) > f(x). \tag{1.4}$$

Definition 1.2.9 Global Minimizer

Let $x^* \in F$ s.t. F be a feasible set, then a point x^* is a global minimizer if:

$$f(x^*) \leq f(x), \forall x \in F \tag{1.5}$$

x^* is a strict global minimizer if $x \neq x^*$ and

$$f(x^*) < f(x). \tag{1.6}$$

Definition 1.2.10 Local Minimizer

Let $x^* \in F$ s.t. F be a feasible set, then a point x^* is a local minimizer if $\exists N, N$ is an open neighborhood of x^* s.t.

$$f(x^*) \leq f(x), \forall x \in N \tag{1.7}$$

x^* is a strict local minimizer if $x \neq x^*$ and

$$f(x^*) < f(x). \tag{1.8}$$

Definition 1.2.11 Convergence

Let x_k be any sequence; if $\{x_k\}$ has a limit, then it converges. Suppose that $\exists w, i > 0$ and that $\{x_k\}$ converges to x^* where $x_k \neq x^*$:

$$\lim_{k \to \infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^w} = i. \tag{1.9}$$

After this happens, it is said that $\{x_k\}$ converges to x^* of order w with a constant i.

Now, If $w = 1$ and $i < 1$, then $\{x_k\}$ is called Linearly Convergent. If $w = 2$ and $i < 1$, then $\{x_k\}$ is called Quadratic Convergent.
A sequence $\{x_k\}$ is called Super-Linearly Convergent if:

$$\lim_{k \to \infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = 0. \tag{1.10}$$

Definition 1.2.12 Constrained Optimization Problem

A general constraint optimization problem is the following.

$$max f(x),$$

subject to:

$$g_i(x) \leq b_i, \qquad i = 1, 2, \ldots, p$$

$$h_j(x) = c_j, \qquad j = 1, 2, \ldots, m$$

The functions $g(x)$ and $h(x)$ is an inequality and equality constraints. The functions f, g and h are differentiable functions and their derivatives are continuous.

Definition 1.2.13 Smooth Function

A function f said to be a smooth function, if it's continuous and have continuous derivatives at every point in the domain.

Definition 1.2.14 Unconstrained Optimization Problem

Consider the problem of minimizing $f(x)$ that relies on real variables without limiting their values. Mathematically, let x be a real vector such that $x \in R^n$ with $n \geq 1$ components and let f be a smooth function, where $f : R^n \to R$. Then the problem of unconstrained optimization is indicated by:

$$minf(x), \quad x \in R$$

Definition 1.2.15 Convex Set

Let $S \in R^n$ be a set, then it's said to be a convex set if the line segment joining any two points belong to the set S lies entirely inside S. Mathematically formulation: let $x, y \in S$, then:

$$\theta x + (1 - \theta)y \in S, \ \forall \theta \in [0, 1]. \tag{1.11}$$

Definition 1.2.16 Convex Function

A function $f : S \to R$ is said to be a convex function if $S \in R^n$ is a convex set. Equivalent, let $x, y \in S$, then:

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y), \ \forall \theta \in [0,1]. \tag{1.12}$$

Definition 1.2.17 The Gradient Operator

The gradient operator is a vector that contains partial derivatives of the function f. The gradient operator of f is denote by ∇f and written as:

$$\nabla f(\mathrm{x}) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix} \tag{1.13}$$

Definition 1.2.18 The Hessian Matrix The second derivative of a function with several variables is represented by the Hessian matrix. The Hessian matrix of f is denote by $H_f(\mathrm{x})$ and written as:

$$H_f(\mathrm{x}) = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial^2 x_1} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial^2 x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial^2 x_n} \end{bmatrix} \tag{1.14}$$

Definition 1.2.19 The Jacobian Matrix

The Jacobian matrix is represented by the matrix of the first order partial derivatives of f, assuming that f is a vector function. The

Jacobian Matrix is denoted by $J(\mathrm{x})$, and written as:

$$J(\mathrm{x}) = \begin{bmatrix} \frac{\partial f_1(x)}{\partial x_1} & \frac{\partial f_1(x)}{\partial x_2} & \cdots & \frac{\partial f_1(x)}{\partial x_n} \\ \frac{\partial f_2(x)}{\partial x_1} & \frac{\partial f_2(x)}{\partial x_2} & \cdots & \frac{\partial f_2(x)}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n(x)}{\partial x_1} & \frac{\partial f_n(x)}{\partial x_2} & \cdots & \frac{\partial f_n(x)}{\partial x_n} \end{bmatrix} \tag{1.15}$$

Definition 1.2.20 Square Matrix

Let A be a matrix that has the same number of columns and rows, then it's said to be a square matrix. This matrix of order n is known as $n \times n$ matrix.

Definition 1.2.21 Positive Definite Matrix and Positive Semi Definite Matrix

The matrix A of $n \times n$ is called a positive definite if there exist a vector p, then:

$$p^T A\, p > 0,\ \forall p \neq 0, \tag{1.16}$$

and it is said to be a positive semi definite if:

$$p^T A\, p \geq 0,\ \forall p \neq 0. \tag{1.17}$$

Definition 1.2.22 Symmetric Matrix

The Square matrix A is called symmetric matrix if $A = A^T$.

Definition 1.2.23 Eigenvalues and Eigenvectors

Let A be a square matrix $n \times n$,there exist a nonzero vector p and a scalar β satisfying the equation:

$$Ap = \beta p, \tag{1.18}$$

then, a scalar β and a vector p is called an eigenvalue and an eigenvector of A respectively.

Definition 1.2.24 Symmetric Positive Definite Matrix

Let A be a symmetric matrix, then it's positive definite iff all eigenvalues of A are positive.

Definition 1.2.25 Iterative Methods

An iterative method is a process that is conducted repeatedly to find the root of an equation to solve a system of equations. As an example of iterative methods: Newton method, Quasi-Newton method, line search method, trust region method,... etc.

Definition 1.2.26 Linear Function

A function with one or two variables; it is also known as a first-degree function. In a graph, it is a function that results in a straight line. We may write it as follows:

$$f(x) = ax + b \tag{1.19}$$

Definition 1.2.27 Nonlinear Function

If the function $f : R^n \to R$ doesn't achieve the superposition principle that's:

$$f(x_1 + x_2 + \ldots + x_n) \neq f(x_1) + f(x_2) + \ldots + f(x_n) \tag{1.20}$$

then f is called nonlinear function.

Definition 1.2.28 Nonlinear System of Equations

A set of equations that contain at least one nonlinear equation is called system of nonlinear equations. The set of the following equations is said to be a system of nonlinear equations:

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0, \\ f_2(x_1, x_2, \ldots, x_n) &= 0, \\ &\vdots \\ f_n(x_1, x_2, \ldots, x_n) &= 0. \end{aligned}$$

The general form of this system is as follows:

$$F(\mathrm{x}) = 0, \tag{1.21}$$

where $F : R^n \to R^n$, $(x_1, x_2, \ldots, x_n) \in R^n$ and each of f_i is a nonlinear real function, $i = 1, 2, \ldots, n$.

Definition 1.2.29 Quadratic Function

One of the simplest functions is a quadratic function that has constant second derivatives. It is typical to write a quadratic function of n variables like follows:

$$Q(x) = \frac{1}{2} x^T A x + b^T x + c, \tag{1.22}$$

The gradient of this function is:

$$g(x) = Ax + b. \tag{1.23}$$

Definition 1.2.30 Merit Function

A modified cost function is called a merit function, and it's used in iteration regimes when constraints should be relaxed to avoid bad

convergence behavior. If you have a constrained optimization problem that simultaneously decreases the cost function and violates one of the constraints, you can use merit functions.

Definition 1.2.31 Taylor Series

If $f(x)$ has a real or complex values, then the power series is the Taylor series of the function f, which can be differentiated infinitely at a real or complex number x_0:

$$\begin{aligned} f(x) =& f(x_0) + (x - x_0) f'(x_0) + \frac{(x - x_0)^2}{2!} f''(x_0) \\ &+ \cdots + \frac{(x - x_0)^{\mathrm{n}}}{\mathrm{n!}} f^n(x_0) + R_n \end{aligned} \tag{1.24}$$

It can also be written as:

$$f(x) = \sum_{k=0}^{\mathrm{n}} \frac{(x - x_0)^k f^k(x_0)}{\mathrm{k!}} + R_n \tag{1.25}$$

R_n represent Lagrange reminder :

$$R_n = \int_{x_0}^{x} f^{(n+1)}(t) \frac{(x - t)^{\mathrm{n}}}{\mathrm{n!}} dt \tag{1.26}$$

Definition 1.2.32 Topological Space

A topological space is a geometrical space in which proximity is specified but cannot always be measured by a numeric distance.

Definition 1.2.33 The Neighborhood

Let X be a topological space and b is a point in X, a neighborhood of b is a subset N of X that includes an open set G containing b, ($b \in G \subseteq N$).

Definition 1.2.34 Accumulation Point

Let b be any point and G be a set of topological space X, then b is said to be an accumulation point if it can be approximated by points of G. every neighborhood of a point b also contains a point of G other than b.

Definition 1.2.35 Fixed Point

Let $F : G \to R^n$ is a real function s.t. $G \subset R^n$, if $f(a) = a$, for some $a \in G$, then a point a is said to be a fixed point of the function f.

Definition 1.2.36 Stationary Point

If f is a differentiable function of one variable, the stationary point of f is a point on the function graph where the derivative of the function f equals zero.

Definition 1.2.37 Descent Direction

If $f : R^n \to R$ is a differentiable at $x \in R^n$ and $d \in R^n$ is a vector, then if $\nabla f(x)^T d < 0$ is satisfied, then d is a descent direction of f at x.

Definition 1.2.38 Steepest – Descent Direction

The most obvious option for a line search approach is the steepest direction of descent $d_k = -\nabla f_k$. The direction that f decreases most quickly from x_k is the best one we could take because it only needs to calculate the gradient ∇f_k and not the second derivatives.

The steepest descent is an iterative optimization approach for finding a function's minimum that uses first-order derivatives. The line search strategy, where each step is taken along $d_k = -\nabla f_k$ (d_k is referred to as the steepest direction), is the steepest descent method.

Definition 1.2.39 Monotone Function

A function is called monotone iff non-increasing or non-decreasing s.t.:

Monotone non-decreasing (increasing) if :

$$x \leq y \Longrightarrow f(x) \leq f(y). \tag{1.27}$$

Monotone non-increasing (decreasing) if:

$$x \leq y \Longrightarrow f(x) \geq f(y). \tag{1.28}$$

Definition 1.2.40 Lipschitz Condition

Let f continuous at $[x, y]$ and differentiable at (x, y), $\exists L > 0$, then a function f satisfies a Lipschitz condition:

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \quad \forall x, y \in R^n. \tag{1.29}$$

Where L is Lipschitz constant.

Chapter 2

Nonlinear Systems of Equations

2.1 Introduction

Generally, the nonlinear systems of equations play a significant role in mathematics and are used widely in fields like chemistry, physics, economics and others fields. While certain nonlinear issues could be converted to linear ones, others could not. Consequently, developing techniques for solving nonlinear systems is essential [39]. Many of researchers have concentrated on supplying suitable methods to solve non-linear systems, hence certain well-known algorithms have been suggested as providing such ways and means. optimization problems are among such those problems.

The nonlinear system of equations is generally regarded as:

$$F(\mathrm{x}) = 0 \tag{2.1}$$

where $F : R^n \rightarrow R^n$ is continuous function. There exists a close connection between unconstrained optimization problems and nonlinear system of equations, making it appropriate to approach this problem using unconstrained optimization strategies [11]. Solutions to nonlinear system of equations are also quite difficult problems. There are various methods that have been presented, but they are expensive due to the high number of partial equations they must solve or they produce results that are inaccurate in terms of time or exact answer [9].

Some techniques to solve nonlinear system of equations is line search technique, the trust region technique and others [15]. Newton's technique had the forefront and located in the heart of many important algorithms that introduced to solve both of optimization problems and

the non-linear equations systems [7].

In section two, some different numerical approaches for solving nonlinear systems of equations, namely Newton's and Broyden's methods are discussed. The advantages and disadvantages of each methods are being explored.

2.2 Some Methods for Solving Nonlinear Systems

This section will present some approaches for solving the non-linear systems of equations.

2.2.1 Newton's Method

Although there are many important methods to solve nonlinear systems of equations, Newton's method is one of the most popular and efficient method to solve $F(\mathrm{x}) = 0$, but before talking about this system, we'll show how can Newton's method solve the nonlinear optimization problem $f(x) = 0$.

The Taylor series has been extended around the point x_1, then

$$f(x) = f(x_1) + (x - x_1) f'(x_1) + \frac{(x - x_1)^2}{2!} f''(x_1) + \cdots \tag{2.2}$$

where f, f' and f'' are computed at the point x_1.

Now, we take first and second terms of the series above:

$$f(x) \approx f(x_1) + (x - x_1) f'(x_1), \tag{2.3}$$

If $f(x) = 0$, then we get the equation's root and we receive the following results:

$$f(x_1) + (x - x_1) f'(x_1) = 0, \tag{2.4}$$

We can obtain the following approximation to the root by rearranging equation (2.4):

$$x = x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}, \tag{2.5}$$

In order to obtain Newton's iterative method, we can generalize equation (2.5) as follows:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k \in N. \tag{2.6}$$

where $x_k \to x^*$ (as $k \to \infty$), x^* is an approximation to a root of $f(x)$ [4]. Now, the steps of Newton's method for solving optimization problems with one variable will be mentioned.

2.2.1.1 Algorithm of Newton's Method [4]

Step 1: Let x_0 be an initial point.

Step 2: Find $f'(x)$, $f'(x_0)$ and calculate the iterations by:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad \text{where } f'(x_k) \neq 0.$$

Step 3: If the function's first derivative value is zero, we obtain the optimal values (maximum or minimum). If not, repeat step 2 with the new value.

Example 2.2.1.1 *[27] Find the optimal solution for the following function using Newton's method :* $f(x) = x^3 - 3x - 4, \qquad x_0 = 2$

Solution:- $f'(x) = 3x^2 - 3$

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

$$x_1 = x_0 - \frac{(x_0)^3 - 3(x_0) - 4}{3(x_0)^2 - 3} = 2.22222$$

$$x_2 = x_1 - \frac{(x_1)^3 - 3(x_1) - 4}{3(x_1)^2 - 3} = 2.19620$$

$$x_3 = x_2 - \frac{(x_2)^3 - 3(x_2) - 4}{3(x_2)^2 - 3} = 2.19582$$

$$x_4 = x_3 - \frac{(x_3)^3 - 3(x_3) - 4}{3(x_0)^2 - 3} = 2.19582$$

Since the final iteration resembles x_3, $k = 3$ provides a good approximation. It can be observed that this method is straightforward but efficient and that repetition will bring the outcome closer to the required root.

The fact that equation (2.6) only applied to optimization problems with a single variable. It has to be altered in order to be used to solve a set of multiple-variable nonlinear algebraic equations.

Linear algebra has shown us that we can take systems of equations and express those systems as matrices and vectors. In light of this and utilizing definition 1.2.28 nonlinear system of equations, we may represent a nonlinear system as a matrix with an Corresponding vector. So, it is possible to derive the following equation:

$$x_{k+1} = x_k - J(x_k)^{-1}F(x_k), \quad x \in R^n$$

where k indicates the iteration, F is a vector function, and $J(x_k)^{-1}$ is the inverse of the Jacobian matrix. We can solve the system $F(\mathrm{x}) = 0$ alternatively rather than the equation $f(x) = 0$. The steps of Newton's

method for solving system of nonlinear equations will be mentioned [26].

Step 1: Let x_0 be initial vector;

$$\mathrm{x}_0 = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Step 2: Find $F(\mathrm{x}_0)$, where

$$F(\mathrm{x}_0) = \begin{bmatrix} f_1(x_0) \\ f_2(x_0) \\ \vdots \\ f_n(x_0) \end{bmatrix}.$$

Step 3: Find $J(\mathrm{x}_0)$ from equation (1.15).

Step 4: Compute the vector y_0 from :

$$J(\mathrm{x}_0)\mathrm{y}_0 = -F(\mathrm{x}_0)$$

Step 5: Using **Step 4**, we finish the first iteration, and we get the following result:

$$\mathrm{x}_1 = \mathrm{x}_0 + \mathrm{y}_0$$

This method is repeated until x_k converges to x^* after we have compute x_1. This proves that the system has been solved when $F(\mathrm{x}) = 0$, where x^* is the system's solution [26].

A set of vectors converges when their norm, $\|x_k - x_{k-1}\| = 0$, this implies:

$$\|x_k - x_{k-1}\| = \sqrt{\left(x_1^{(k)} - x_1^{(k-1)}\right)^2 + \ldots + \left(x_n^{(k)} - x_n^{(k-1)}\right)^2} = 0.$$

Example 2.2.1.2 *[26] Solve the following nonlinear system using Newton's method :*

$$3x_1 - \cos(x_2 x_3) - \frac{1}{2} = 0,$$
$$x_1^2 - 81(x_2 + 0.1)^2 + \sin x_3 + 1.06 = 0,$$
$$e^{-x_1 x_2} + 20x_3 + \frac{10\pi - 3}{3} = 0,$$

when the initial vector is:

$$\mathrm{x}_0 = \begin{bmatrix} 0.1 \\ 0.1 \\ -0.1 \end{bmatrix}$$

Solution:-

We will define $F(\mathrm{x})$ and $J(\mathrm{x})$:

$$F(x) = \begin{bmatrix} 3x_1 - \cos(x_2 x_3) - \frac{1}{2} \\ x_1^2 - 81(x_2 + 0.1)^2 + \sin x_3 + 1.06 \\ e^{-x_1 x_2} + 20x_3 + \frac{10\pi - 3}{3} \end{bmatrix}$$

$$J(\mathrm{x}) = \begin{bmatrix} 3 & x_3 \sin(x_2 x_3) & x_2 \sin(x_2 x_3) \\ 2x_1 & -162(x_2 + 0.1) & \cos x_3 \\ -x_2 e^{-x_1 x_2} & -x_1 e^{-x_1 x_2} & 20 \end{bmatrix}$$

Now, we compute $F(\mathrm{x}_0)$ and $J(\mathrm{x}_0)$, where $\mathrm{x}_0 = (0.1, 0.1, -0.1)^T$

$$F(\mathrm{x}_0) = \begin{bmatrix} 0.3 - \cos(-0.01) - \frac{1}{2} \\ 0.01 - 3.24 + \sin(-0.1) + 1.06 \\ e^{(-0.01)} - 2 + \frac{10\pi - 3}{3} \end{bmatrix}$$
$$= \begin{bmatrix} -1.19995 \\ -2.269833417 \\ 8.462025346 \end{bmatrix}$$

$$J(\mathrm{x}_0) = \begin{bmatrix} 3 & (-0.1)\sin(-0.01) & 0.1\sin(-0.01) \\ 0.2 & -32.4 & \cos(-0.1) \\ -0.1e^{-0.01} & -0.1e^{-0.01} & 20 \end{bmatrix}$$
$$= \begin{bmatrix} 3 & 0.000999983 & -0.000999983 \\ 0.2 & -32.4 & 0.995004165 \\ -0.099004984 & 0.99004983 & 20 \end{bmatrix}$$

Find y_0 by:

$J(\mathrm{x}_0)\mathrm{y}_0 = -F(\mathrm{x}_0)$

$$\begin{bmatrix} 3 & 0.000999983 & -0.000999983 \\ 0.2 & -32.4 & 0.995004165 \\ -0.099004984 & 0.99004983 & 20 \end{bmatrix} \begin{bmatrix} y_1^0 \\ y_2^0 \\ y_3^0 \end{bmatrix}$$
$$= - \begin{bmatrix} -1.19995 \\ -2.269833417 \\ 8.462025346 \end{bmatrix}$$

Then,

$$J(\mathrm{y}_0) = \begin{bmatrix} 0.40003702 \\ -0.08053314 \\ -0.42152047 \end{bmatrix}$$

Using y_0 to find the first iteration $J(\mathrm{x}_1)$:

$$\mathrm{x}_1 = J(\mathrm{x}_0) + J(\mathrm{y}_0)$$

$$J(\mathrm{x}_1) = \begin{bmatrix} 0.1 \\ 0.1 \\ -0.1 \end{bmatrix} + \begin{bmatrix} 0.40003702 \\ -0.08053314 \\ -0.42152047 \end{bmatrix}$$

$$= \begin{bmatrix} 0.50003702 \\ 0.01946686 \\ -0.52152047 \end{bmatrix}.$$

We obtain the following iterations using the same iterating process, as shown in the table below:

Table 2.1: Numerical Results of Example 2.2.1.2

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$\left\|\mathrm{x}^{(\mathrm{k})} - \mathrm{x}^{(\mathrm{k}-1)}\right\|$
0	0.10000000	0.10000000	-0.10000000	-
1	0.50003702	0.01946686	-0.52152047	0.422
2	0.50004593	0.00158859	-0.52355711	0.0179
3	0.50000034	0.00001244	-0.52359845	0.00158
4	0.50000000	0.00000000	-0.52359877	0.0000124
5	0.50000000	0.00000000	-0.52359877	0

where, $\|x_k - x_{k-1}\| = \sqrt{\left(x_1^{(k)} - x_1^{(k-1)}\right)^2 + \ldots + \left(x_n^{(k)} - x_n^{(k-1)}\right)^2}$.

We know that when a set of vectors converges the norm

$$\|x_k - x_{k-1}\| = 0.$$

The norm is equal to 0 at the fifth iteration, as shown by the table2.1. This shows that the solution represented by x^*, has been reached by the system $F(\mathrm{x})$. As a result, we see from table2.1 results that:

$$\mathrm{x}^* = \begin{bmatrix} 0.50000000 \\ 0.00000000 \\ -0.52359877 \end{bmatrix},$$

is an approximation solution of $F(\mathrm{x}) = 0$.

2.2.1.2 Advantages and Disadvantages of Newton's Method [26]

The following are some advantages and disadvantages of this method:

a. Advantages

Advantage of this method is that it has a simple structure and may be applied to a wide range of problems. Due to the presence of quadratic convergence, which occurs when they get close to the actual solution of the nonlinear system, the method requires less iterations than other methods.

b. Disadvantages

The main disadvantages of this method is requirement to compute $J(x)$ and its inversion for each iteration, computing Jacobian matrix

and its inverse may take a lot of time especially for large dimensions problems. Newton's method may fail to converge, which is another issue we can face when employing it. An oscillation between points will occur if Newton's method is unable to converge.

2.2.2 Broyden's Method

In the previous section, the Newton's methodology, a numerical method was explored. It is known that this technique's primary disadvantage is the requirement for computing both Jacobian matrix and its inverse for each repeat. Devoid of these problems is what we seek. Newton's technique is a series of methods that includes methods referred to as Broyden's method [26].

They employed it to solve the nonlinear equation system derived by the Burden and Faires techniques[4], replacing the Jacobin matrix with an approximation matrix that is updated in each iteration. By extension, this indicates that the Newton method and the Broyden method are virtually identical. The only exception is that $J(x)$ is replaced by an approximation matrix B_k. Consequently, in [4], the next equation is derived:

$$x_{k+1} = x_k - B_k^{-1} f(x_k), \tag{2.7}$$

This is known as Broyden's iterative process. B_k is defined as :

$$B_k = B_{k-1} + \frac{y_k - B_{k-1} S_k}{\| s_k \|_2^2} S_k^T, \tag{2.8}$$

where $y_k = f(x_k) - f(x_{k-1})$ and $s_k = x_k - x_{k-1}$. Hence, Broyden's method, uses that calculating B_k^{-1} instead of B_k, which leads us to the

theorem following :

Theorem 2.2.2.1 *Sherman-Morrison Forumula*

Let B be a non-singular matrix, x, y are vectors, then $B + xy^T$ is non-singular provided that $y^T B^{-1} x \neq -1$ and

$$(B + xy^T)^{-1} = B^{-1} - \frac{B^{-1} x y^T B^{-1}}{1 + y^T B^{-1} x}$$

See proof of this theorem in [4].

2.2.2.1 Algorithm of Broyden's Method

Now, the steps of Broyden's method for solving system of nonlinear equations will be mentioned.

Step 1: Given initial vector x_0.

Step 2: Compute $F(x_0)$ and B_0^{-1}.

Step 3: Calculate $x_1 = x_0 - B_0^{-1} F(x_0)$.

Step 4: Compute $F(x_1)$.

Step 5: Calculate $y_1 = F(x_1) - F(x_0)$ and $s_1 = x_1 - x_0$.

Step 6: Compute $y_1 B_0^{-1} s_1^T$.

Step 7: Calculate $B_1^{-1} = B_0^{-1} + \frac{(s_1 - B_0^{-1} y_1) s_1^T B_0^{-1}}{y_1 B_0^{-1} s_1^T}$.

Step 8: Calculate $x_2 = x_1 - B_1^{-1} F(x_1)$.

This step involves repeating the procedure until x^* is reached, i.e.

$$x_k = x_{k+1} = x^*.$$

Example 2.2.2.1 *[4] Solve the following nonlinear system using Broyden's method :*

$$3x_1 - \cos(x_2 x_3) - \frac{1}{2} = 0,$$
$$x_1^2 - 81(x_2 + 0.1)^2 + \sin x_3 + 1.06 = 0,$$
$$e^{-x_1 x_2} + 20x_3 + \frac{10\pi - 3}{3} = 0,$$

initial vector is:

$$\mathrm{x}_0 = \begin{bmatrix} 0.1 \\ 0.1 \\ -0.1 \end{bmatrix}$$

Solution:-

We start the system's solution in the same manner as Newton's first steps,

$$F(\mathrm{x}) = \begin{bmatrix} 3x_1 - \cos(x_2 x_3) - \frac{1}{2} \\ x_1^2 - 81(x_2 + 0.1)^2 + \sin x_3 + 1.06 \\ e^{-x_1 x_2} + 20x_3 + \frac{10\pi - 3}{3} \end{bmatrix}$$

$$J(\mathrm{x}) = \begin{bmatrix} 3 & x_3 \sin(x_2 x_3) & x_2 \sin(x_2 x_3) \\ 2x_1 & -162(x_2 + 0.1) & \cos x_3 \\ -x_2 e^{-x_1 x_2} & -x_1 e^{-x_1 x_2} & 20 \end{bmatrix}$$

where $\mathrm{x}_0 = (0.1, 0.1, -0.1)^T$ and $B_0 = J(\mathrm{x}_0)$

$$F(\mathrm{x}_0) = \begin{bmatrix} 0.3 - \cos(-0.01) - \frac{1}{2} \\ 0.01 - 3.24 + \sin(-0.1) + 1.06 \\ e^{(-0.01)} - 2 + \frac{10\pi - 3}{3} \end{bmatrix}$$

$$= \begin{bmatrix} -1.19995 \\ -2.269833417 \\ 8.462025346 \end{bmatrix}$$

$$J(\mathrm{x}_0) = \begin{bmatrix} 3 & (-0.1)\sin(-0.01) & 0.1\sin(-0.01) \\ 0.2 & -32.4 & \cos(-0.1) \\ -0.1e^{-0.01} & -0.1e^{-0.01} & 20 \end{bmatrix}$$

$$= \begin{bmatrix} 3 & 0.000999983 & -0.000999983 \\ 0.2 & -32.4 & 0.995004165 \\ -0.099004984 & 0.99004983 & 20 \end{bmatrix}$$

Now, we compute B_0^{-1} :

$$B_0^{-1} = \begin{bmatrix} 0.3333332 & 1.023852 \times 10^{-5} & 1.615701 \times 10^{-5} \\ 2.108607 \times 10^{-3} & -3.086883 \times 10^{-2} & 1.535836 \times 10^{-3} \\ 1.660520 \times 10^{-3} & -1.527577 \times 10^{-4} & 5.000768 \times 10^{-5} \end{bmatrix}$$

Then,

$$\mathrm{x}_1 = \mathrm{x}_0 - B_0^{-1} F(x_0)$$

$$= \begin{bmatrix} 0.4995697 \\ 1.946685 \times 10^{-2} \\ -0.5215205 \end{bmatrix}.$$

We calculate $F(\mathrm{x}_1)$

$$F(\mathrm{x}_1) = \begin{bmatrix} -3.394465 \times 10^{-4} \\ -0.3443879 \\ 3.188238 \times 10^{-2} \end{bmatrix}$$

$$\mathrm{y}_1 = \begin{bmatrix} 1.199611 \\ 1.925445 \\ -8.430143 \end{bmatrix}, \quad s_1 = \begin{bmatrix} 0.3998697 \\ -8.053315 \times 10^{-2} \\ -0.4215204 \end{bmatrix}$$

$$B_1^{-1} = B_0^{-1} + \frac{\left[\left(s_1 - B_0^{-1}\mathrm{y}_1\right) s_1^T B_0^{-1}\right]}{\mathrm{y}_1 B_0^{-1} s_1^T}$$

$$= \begin{bmatrix} 0.3333781 & 1.11050 \times 10^{-5} & 8.967344 \times 10^{-6} \\ -2.021270 \times 10^{-3} & -3.094849 \times 10^{-2} & 2.196906 \times 10^{-3} \\ 1.022214 \times 10^{-3} & -1.650709 \times 10^{-4} & 5.010986 \times 10^{-2} \end{bmatrix}$$

$$\mathrm{x}_2 = \mathrm{x}_1 - B_1^{-1} F\left(\mathrm{x}_1\right)$$

$$= \begin{bmatrix} 0.4999863 \\ 8.737833 \times 10^{-3} \\ -0.5231746 \end{bmatrix}$$

We continue in the same way by iterating until we reach x^*.

2.2.2.2 Advantages and Disadvantages of Broyden's Method

This method has some advantages and disadvantages as follows [26]:

a. Advantages

Reduce of calculations is the major advantage of Broyden's Method. Compared to Newton, this Method requires less computations because it is possible to construct the inverse of the approximation matrix B_k^{-1} directly from the previous repeat.

b. Disadvantages

The number of repetitions required to arrive the solution is greater than the number of iterations required by Newton's method since the problem does not converge quadratically. Unlike Newton's method, it doesn't use successive repetitions to fix faults that could cause minor inaccuracy. While Broyden's technique isn't as strictly self-correcting as Newton's is, the ultimate iteration will still be the same.

Chapter 3

Line Search Techniques

3.1 Introduction

The line search techniques typically start by generating the search direction and focuses on finding a suitable step along that direction. Any method used for optimization begins at a starting point called x_0 and iterates through a set of points until it reaches x^*, which is optimal point. The next point is determined at any k_{th} iteration by:

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

s.t. d_k is the search direction and α_k is the step length that must be a positive scalar, the step length is defined as the distance to be traveled in the direction of the search. The majority of line search algorithms demand that d_k be a descent direction Definition 1.2.37 drscent direction, where $\nabla f(x)^T d < 0$ guarantees that the function f in this direction can be minimized [22]. The function value at that new point should be less than or equal to the previous point.

$$f(x_{k+1}) \leq f(x_k)$$

$$f(x_k + \alpha_k d_k) \leq f(x_k)$$

One of the key factors of the effectiveness of the line search method is the accurately choice of step length α_k and direction d_k. There are also many optimization methods commonly used in applied mathematics, that using the line search techniques such as: trust region method, conjugate gradient method, and projection method [9, 15, 16, 20, 30]. Some were used in a new approaches in the following chapters.

The remaining of this chapter is organized as follow, in section two, we explains step length α_k, Wolfe and Goldstein conditions, in section three, we introduce the framework of line search method and algorithm. Finally, section four includes a new method that uses the line search technique to solve non-linear systems of equation .

3.2 Step Length [9]

The step length is represented by the horizontal distance along the progress plane that is covered in one step, and is shown by the symbol α. When determining the step length α_k, we must choose a choice to drastically lower f, but we also don't want to select for too long. The univariate function ϕ with the following definition would be the globally minimizer as the best choice.

$$\phi(\alpha) = f(x_k + \alpha_k d_k), \quad \alpha > 0 \tag{3.1}$$

However, determining this value is typically prohibitively expensive (see figure 3.1). To discover a local minimizer of ϕ with even modest precision, too many assessments of objective function f are generally required.

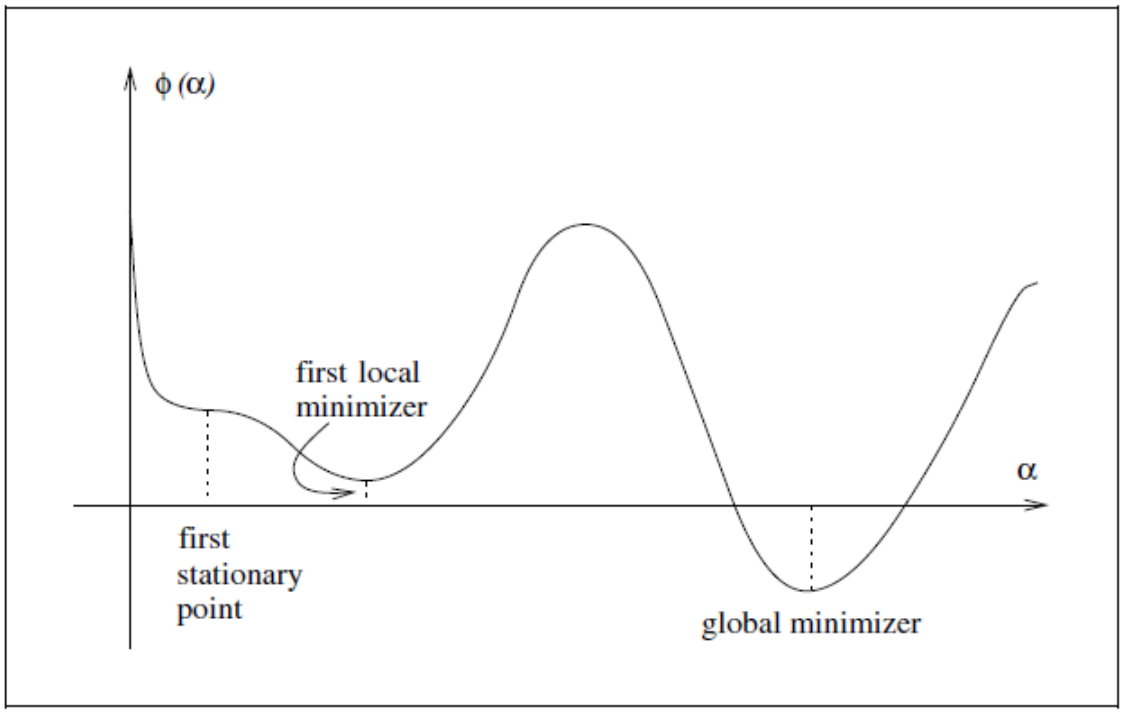

Figure 3.1: The Ideal Step Length

A straightforward conditions that might place on α_k is that f should be lowered, i.e.,

$$f(x_k + \alpha_k d_k) < f(x_k).$$

3.2.1 The Wolfe Conditions [9]

According the inequality, a frequent line search requirement is that α_k first provide a sufficient drop in the objective function f:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \xi_1 \alpha \nabla f_k^T d_k, \tag{3.2}$$

for some constant $\xi_1 \in (0, 1)$. To put it another way, the decrease in f should be proportional to both α_k and $\nabla f_k^T d_k$. Inequality (3.2) is also known as Armijo condition.

Figure 3.2 shows how to achieve the sufficient decreases condition. $\varphi(\alpha)$ is a linear function represents the right-hand side of inequality(3.2), $\varphi(.)$ has a negative slope of $\xi_1 \nabla f_k^T d_k$, but due to the fact that $\xi_1 \in (0, 1)$, for very small positive values of α, it is located above the graph of ϕ. If $\phi(\alpha) \leq \varphi(\alpha)$, then we say that α is acceptable, and this condition is said to be the sufficient decrease condition. figure 3.2 shows the intervals at which this condition is met. In reality, ξ_1 is chosen to be very small.

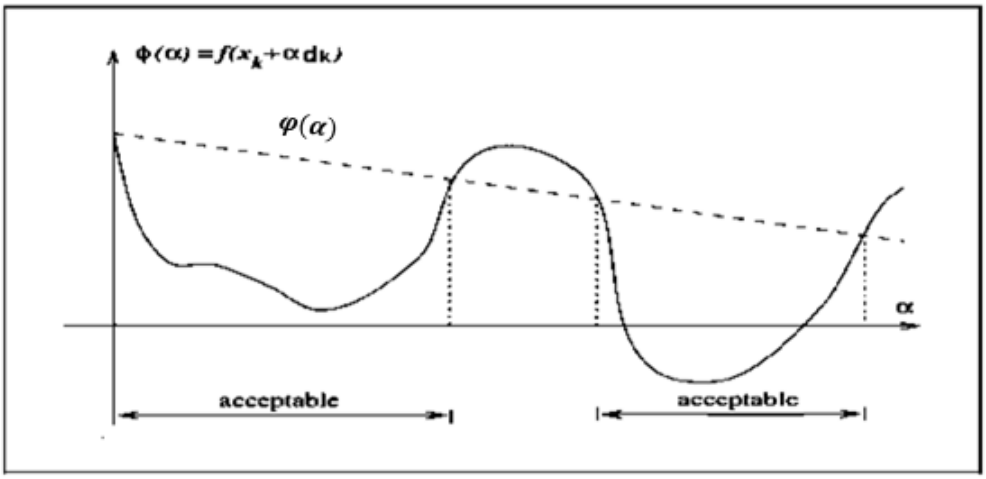

Figure 3.2: Sufficient Decreases Condition

The sufficient decrease condition is insufficient on its own to ensure that the algorithm advances adequately since, as shown in figure 3.2, it is fulfilled with all sufficiently small α values. We'll also include curvature condition, which requires α_k to achieve:

$$\nabla f(x_k + \alpha_k d_k)^T d_k \geq \xi_2 \nabla f_k^T d_k, \tag{3.3}$$

where ξ_1 and ξ_2 are constants such that $\xi_2 \in (\xi_1, 1)$, and ξ_1 is the constant from inequality (3.2). The goal of providing the aforementioned criteria is to to exclude the short steps that are unacceptable. figure 3.3 depicts

the condition of curvature.

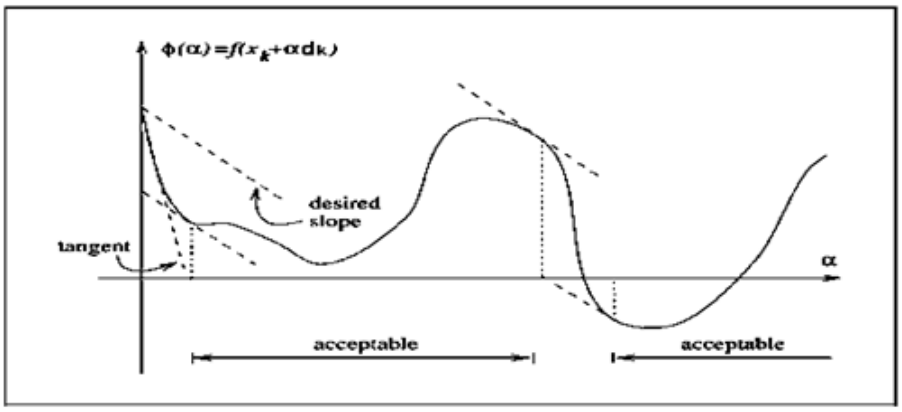

Figure 3.3: The Curvature Condition

The Wolfe conditions are a combination of the curvature and sufficient decrease conditions. They are displayed in figure 3.4 and repeated them below for future use:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \xi_1 \alpha_k \nabla f_k^T d_k$$

$$\nabla f(x_k + \alpha_k d_k)^T d_k \geq \xi_2 \nabla f_k^T d_k$$

with $0 < \xi_1 < \xi_2 < 1$. A step length, as seen in figure 3.4, can achieve the Wolfe conditions. without being particularly near to a minimizer of ϕ.

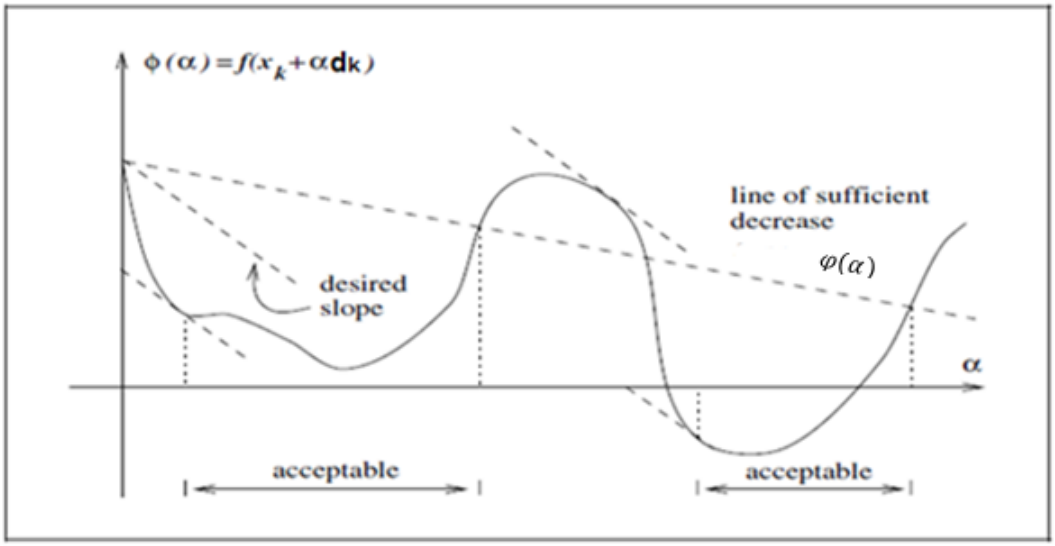

Figure 3.4: Step Lengths Satisfying Wolfe Condition

The curvature condition can be altered, though, to require that α_k be situated in a local minimizer of ϕ or at least a large neighborhood around a stationary point

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \xi_1 \alpha_k \nabla f_k^T d_k \tag{3.4}$$

$$\mid \nabla f(x_k + \alpha_k d_k)^T d_k \mid \geq \xi_2 \mid \nabla f_k^T d_k \mid$$

are the strong Wolfe conditions that α_k must meet, with $0 < \xi_1 < \xi_2 < 1$.

3.2.2 The Goldstein Conditions [9]

The Goldstein conditions, like the Wolfe conditions, guarantee that the step length sufficiently decrease but doesn't become excessively short. Also, can be expressed as a follows:

$$f(x_k) + (1-c)\alpha_k \nabla f_k^T d_k \leq f(x_k + \alpha_k d_k) \leq f(x_k) + c\alpha_k \nabla f_k^T d_k \tag{3.5}$$

with $0 < \xi < \frac{1}{2}$. The second inequality is the sufficient condition of decrease inequality (3.2), but the first inequality is introduced below to limit the step length figure 3.5. One drawbacks of the Goldstein conditions over the Wolfe conditions is that the first inequality in (3.5) can remove all minimizers of ϕ. Both Goldstein and Wolfe's conditions are quite similar, as are their theories of convergence.

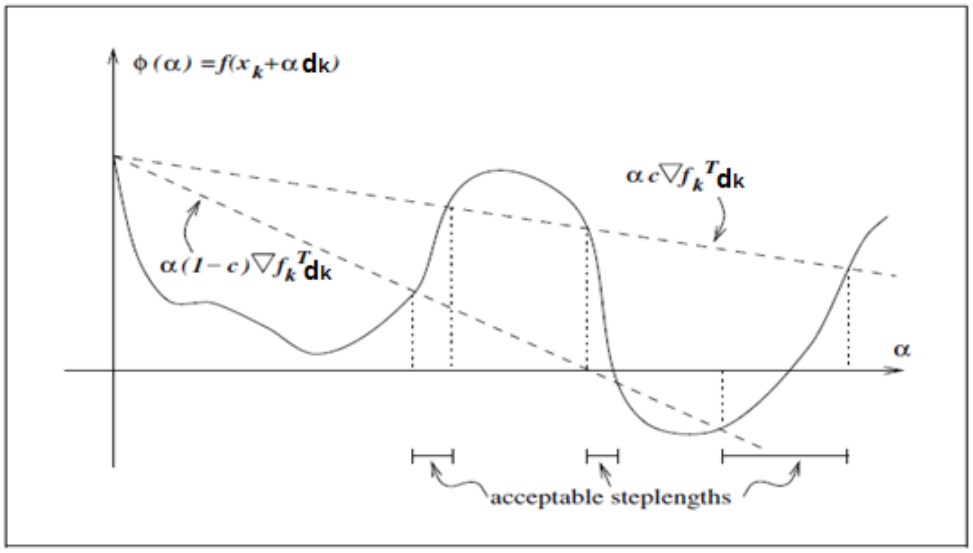

Figure 3.5: The Goldstein Conditions

3.3 The Framework of Line Search Methods

Every technique or strategy has its own framework or method. The framework of the line search technique begins when the algorithm selects a certain direction d_k and attempts to discover the length of that direction during the current iteration x_k. The goal is to get a new iteration with a minimized function value.

by solving the following problem to determine α_k, we may roughly

estimate the distance to be traveled in d_k

$$minf(x_k + \alpha_k d_k) \tag{3.6}$$

The problem above is a one-dimensional minimization problem. By solving (3.6), we will gain the most from the direction. A limited number of experimental step lengths are created using a line search algorithm until the suitable step length is determined around the minimum of(3.6). At each new location, a new step length and search direction d_k are determined, and the procedure is iterated.

The success and efficacy of line search algorithms are down to a proper choice of step length α_k and search directiond_k. These techniques must also select the decreasing, that $d_k^T \nabla f(x_k) < 0$.

3.3.1 General Algorithm of Line Search technique

Step 0: Given initial point $x_0 \in R^n$, set $k := 0$ and $\epsilon > 0$.

Step 1: Determine the search direction d_k.

Step 2: select the step length α_k, s.t. the objective function value decreases, i.e.,

$$f(x_k + \alpha_k d_k) < f(x_k).$$

Step 3: If $\| \nabla f(x_k) \| < \epsilon$, then stop and obtain $minf(x_k)$.

Step 4: Let $x_{k+1} = x_k + \alpha_k d_k$, back to step 1.

End

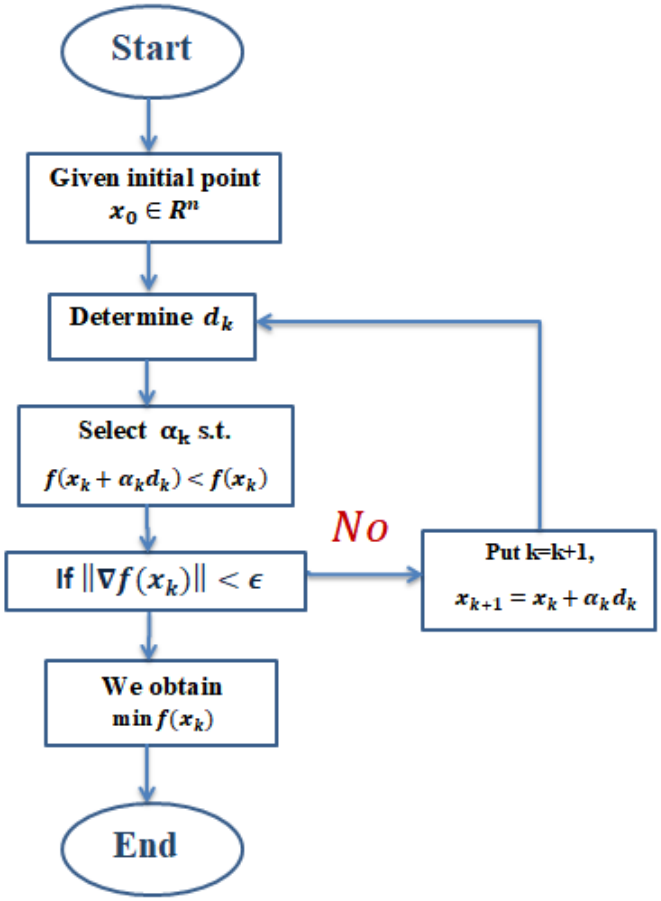

Figure 3.6: Flowchart of Line Search Algorithm

3.4 Line Search Algorithm for Solving Non Linear Systems of Equations

In this section, a new line search algorithm has been proposed such that combine a monotone technique with a modified line search algorithm. This algorithm can be reduce CPU time, number of iterations, and number of function evaluations while solving non-linear equation systems. Additionally, the proposed approach has been proven to converge globally, When the equation is monotone and Lipschitz continuous (LC). The numerical results proved the effectiveness of this method and promise for solving non-linear systems of monotone

equations.

In general regarded the non-linear equations system:

$$F(x) = 0, \tag{3.7}$$

$F : R^n \to R^n$ is continuous. The following requirement for monotonicity in $F(x)$ must be met for non-linear equations to be considered monotone.

$$\langle F(x) - F(y), x - y \rangle \geq 0, \forall x, y \in R^n$$

We shall use monotonicity to solve these problems because their convergence and performance are slow [11]. In order to show that the new technique has a global convergence, we solved a system of non-linear monotone equations. The proposed algorithm will be more competent when compared to the BFGS technique [40].

3.4.1 The Proposed Algorithm

The steps of the proposed Algorithm for solving system of nonlinear equations will be mentioned.

Step 0: Select any initial point $x_0 \in R^n$ and $\beta, \lambda \in (0, 1)$ and $\delta \in (0, \frac{1}{4}]$. Set $k = 0$.

Step 1: Determine d_k using:

$$d_k = -F(x_k), \tag{3.8}$$

If $d_k = 0$, Stop.

Step 2: Select step length $\alpha_k = \beta^{h_k}$ s.t. h_k is the smallest non-negative integer h satisfying :

$$-\langle F\left(x_k + \alpha_k d_k\right), d_k \rangle \geq \lambda \delta_k \beta^{h_k} \left\| F\left(x_k + \alpha_k d_k\right) \right\| \left\| d_k \right\|^2, \tag{3.9}$$

Where $\delta_k = \frac{1}{\delta + \|F(x_k + \alpha_k d_k)\|}$

If $\|F(x_k + \alpha_k d_k)\| = 0$, Stop.

Set trail point $z_k = x_k + \alpha_k d_k$

Step 3: Calculate

$$x_{k+1} = x_k - \frac{\langle F(z_k), x_k - z_k \rangle}{\| F(z_k) \|^2} F(z_k) \tag{3.10}$$

Let $k = k + 1$ and Go to **Step 1**.

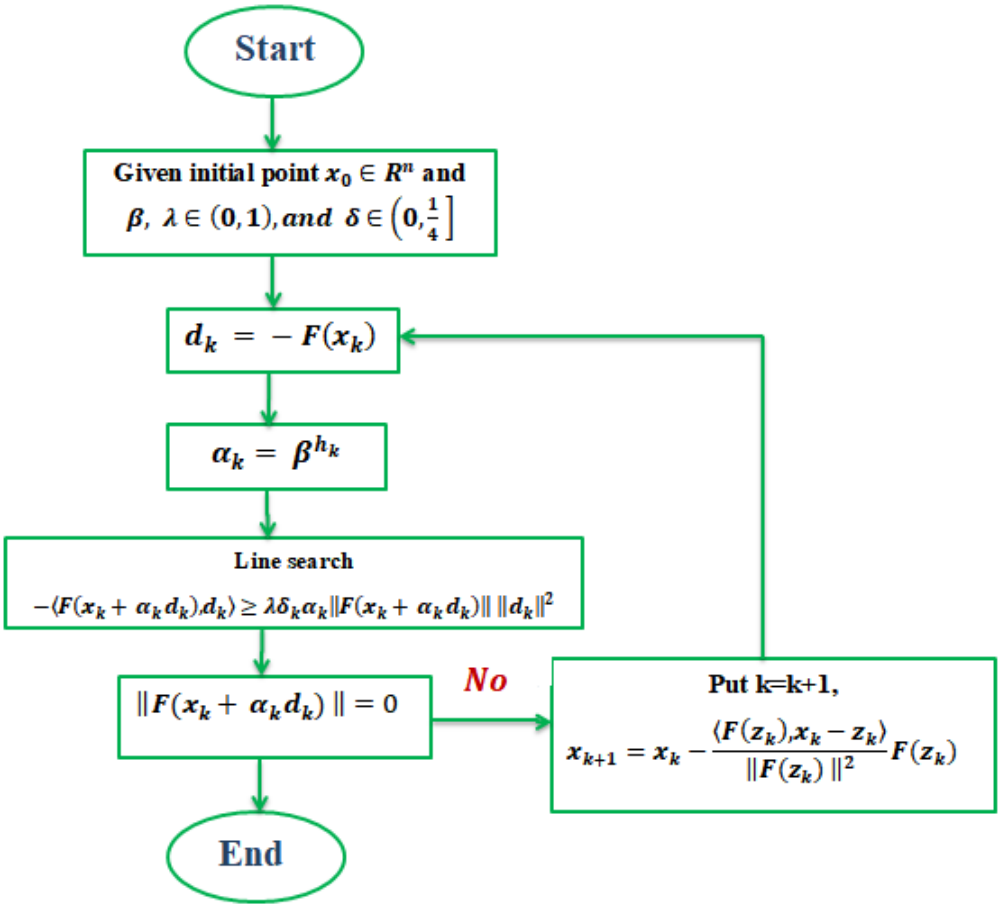

Figure 3.7: Flowchart of Proposed Algorithm

Remark 3.4.1.1 *(i) Suppose that F is (LC) and $\exists L > 0$ and $m > 0$.*

s.t.

$$\| F(x) - F(y) \| \geq L \| x - y \|, \forall x, y \in R^n \tag{3.11}$$

$$\frac{\| F(x_k) \|}{L + m} \leq \| d_k \| \leq \frac{\| F(x_k) \|}{m} \tag{3.12}$$

Now, we'll show how inequality (3.9) is well-defined manner analogous to that of [30].

Assume that inequality (3.9) is not achieved, for any non-negative integer h and any iteration index k, if we taking $\lim_{h\to\infty}$ for two sides:

$$-\lim_{h\to\infty} \left\langle F\left(x_k + \beta^{h_k} d_k\right), d_k \right\rangle < \lim_{h\to\infty} \lambda\delta_k\beta^{h_k} \left\| F\left(z_k\right) \right\| \left\| d_k \right\|^2$$

$$-\left\langle F\left(x_k\right), d_k \right\rangle < 0 \qquad \text{when } x_k = z_k - \alpha_k d_k$$

$$-\left\langle F\left(z_k - \alpha_k d_k\right), d_k \right\rangle < 0$$

$$\alpha_k \left\langle F\left(z_k + d_k\right), d_k \right\rangle < 0$$

$$\alpha_k \left\| F\left(z_k\right) \right\| \left\| d_k \right\|^2 < 0.$$

Therefore, a contradiction appears, it's impossible for every one of $\alpha_k, \| F(z_k) \|$ and $\| d_k \|^2 < 0$, therefore the inequality(3.9) is well-defined.

3.4.2 Convergence Property

The following lemma will be used by us to ensure that the new method globally converges.

Lemma 3.4.2.1 *[30] Let F be monotone, $x, y \in R^n$ achieve $\langle F(y), x - y\rangle > 0$. Let*

$$x^+ = x - \frac{\langle F(y), x - y\rangle}{\| F(y) \|^2} F(y)$$

for any $\hat{x} \in R^n$ s.t. $F(\hat{x}) = 0$, it holds that

$$\| x^+ - \hat{x} \|^2 \leq \| x - \hat{x} \|^2 - \| x^+ - x \|^2$$

The following theorem allows us to now establish the result of convergence for proposed algorithm [30].

Theorem 3.4.2.1 *Assume that* $\{x_k\}$ *is any sequence generated by proposed algorithm , F is monotone and (LC), as well as the set of solutions to equation (3.7) is non-empty. For any* $\hat{x}$ *achieving* $F(\hat{x}) = 0$*, we obtain:*

$$\| x_{k+1} - \hat{x} \|^2 \leq \| x_k - \hat{x} \|^2 - \| x_{k+1} - x_k \|^2$$

especially, $\{x_k\}$ *is bounded. Additionally, it must be achieved then,* $\{x_k\}$ *is either finite and the final iterate yields a solution, or* $\{x_k\}$ *is infinite and*

$$\lim_{k \to \infty} \| x_{k+1} - x_k \| = 0$$

As well, $\{x_k\}$ *converges to some* $\hat{x}$ *s.t.* $F(\hat{x}) = 0$.

Proof: To begin, assuming that there is an end to the algorithm at iteration k, hence

either $d_k = 0$, then obtain $F(x_k) = 0$

or $\| F(z_k) \| = 0$, in this state x_k or z_k is a solution of equation (3.7). Now, assume that $d_k \neq 0$, $F(x_k) \neq 0 \ \forall k$, then $\{x_k\}$ an infinite sequence, is generated. From inequality (3.9):

$$\begin{aligned}
\langle F(z_k), x_k - z_k \rangle &= \langle F(z_k), x_k - (x_k + \alpha_k d_k) \rangle \qquad \text{since } z_k = x_k + \alpha_k d_k \\
&= \langle F(z_k), x_k - x_k - \alpha_k d_k \rangle \\
&= \langle F(z_k), -\alpha_k d_k \rangle \\
&= -\alpha_k \langle F(z_k), d_k \rangle \\
&= -\alpha_k \langle F(x_k + \alpha_k d_k), d_k \rangle \\
&\geq \lambda \delta_k \| F(z_k) \| \alpha_k^2 \| d_k \|^2 > 0
\end{aligned}$$

Then,

$$\langle F(z_k), x_k - z_k \rangle \geq \lambda \delta_k \|F(z_k)\| {\alpha_k}^2 \|d_k\|^2 > 0 \tag{3.13}$$

Assume that $\hat{x}$ be any solution of $F(x) = 0$ and $F(\hat{x}) = 0$. From lemma (3.4.2.1), equation (3.10) and inequality (3.13) lead to the following:

$$\| x_{k+1} - \hat{x} \|^2 \leq \| x_k - \hat{x} \|^2 - \| x_{k+1} - x_k \|^2 \tag{3.14}$$

Especially, $\{\| x_k - \hat{x} \|\}$ is non-increasing and convergent. Consequently, $\{x_k\}$ will be bounded, and obtained:

$$\lim_{k \to \infty} \| x_{k+1} - x_k \| = 0 \tag{3.15}$$

By inequality (3.12), It's obvious that $\{d_k\}$ holds to be bounded and $\{z_k\}$ also bounded.

From equation (3.10) and inequality (3.13), we have:

$$\begin{aligned}
x_{k+1} - x_k &= -\frac{\langle F(z_k), x_k - z_k \rangle}{\|F(z_k)\|^2} F(z_k) \\
x_{k+1} - x_k &= \frac{\alpha_k \langle F(z_k), d_k \rangle}{\|F(z_k)\|^2} F(z_k) \\
&\geq \frac{\lambda \|F(z_k)\| \alpha_k^2 \|d_k\|^2}{\|F(z_k)\|^2} F(z_k) \\
&\geq \lambda \alpha_k^2 \|d_k\|^2
\end{aligned}$$

hence,

$$\|x_{k+1} - x_k\| \geq \alpha_k^2 \|d_k\|^2 \tag{3.16}$$

From equation (3.15) and inequality (3.16):

$$\lim_{k \to \infty} \alpha_k \|d_k\| = 0 \tag{3.17}$$

By inequality (3.12):

$$\lim_{k \to \infty} \inf \|F(x_k)\| = 0, \text{when} \lim_{k \to \infty} \inf \|d_k\| = 0.$$

Since $\{x_k\}$ is bounded and F is continuous, $\{x_k\}$ has an accumulation point at which x^* s.t. $F(x^*) = 0$. We get $\{\| x_k - x^* \|\}$ converges from equation (3.14). As a result, $\{x_k\}$ converges to x^*.

$$\lim_{k\to\infty} \inf\|F(x_k)\| > 0, \text{when} \lim_{k\to\infty} \inf\|d_k\| > 0.$$

By equation (3.17):

$$\lim_{k\to\infty} \alpha_k = 0 \tag{3.18}$$

Inequality (3.9) gives us:

$$-\left\langle F\left(x_k + \beta^{h_{k-1}} d_k\right), d_k\right\rangle < \lambda\delta_k\beta^{h_{k-1}} \left\|F\left(x_k + \beta^{h_{k-1}} d_k\right)\right\| \|d_k\|^2 \tag{3.19}$$

Since $\{x_k\}$, $\{d_k\}$ are bounded, in order to select a sub-sequence, let $k \to \infty$ in inequality (3.19), we get:

$$-\langle F(x^*), d^*\rangle \leq 0 \tag{3.20}$$

s.t. x^* and d^* are subsidiary sequences limiting that chosen. Moreover, by inequality (3.12) and already known argument,

$$-\langle F(x^*), d^*\rangle > 0 \tag{3.21}$$

Inequality (3.20) and inequality (3.21) are a contradiction. Then it's impossible to obtain:

$$\lim_{k\to\infty} \inf\|F(x_k)\| > 0$$

This proof is completed.

3.4.3 Numerical Results

All codes are written in MATLAB version $R2014a$ and all experiments have been run on a computer with $8GB$ of RAM and $CPU\,2.40\,GHz$. Running of the codes has the aim of comparing the outcomes of (D.H) with the aforementioned algorithm. The parameters are determined as follows for (D.H) algorithm: $\beta = 0.1, \lambda = 0.3, \delta = 0.25, \epsilon = 10^{-8}$.

The specific dimensions for these algorithms comparisons are set at $5000 - 50000$,for the following prime points:

$\mathrm{x}_0 = (10, 10, 10, \ldots, 10)^{\mathrm{T}}$,

$\mathrm{x}_1 = (-10, -10, -10, \ldots, -10)^{\mathrm{T}}$,

$\mathrm{x}_2 = (1, 1, 1, \ldots, 1)^{\mathrm{T}}$,

$\mathrm{x}_3 = (-1, -1, -1, \ldots, -1)^{\mathrm{T}}$,

$\mathrm{x}_4 = \left(1, \frac{1}{2}, \frac{2}{\mathrm{x}'}, \ldots, \frac{1}{\mathrm{n}}\right)^{\mathrm{T}}$,

$\mathrm{x}_5 = (0.1, 0.1, 0.1, \ldots, 0.1)^{\mathrm{T}}$,

$\mathrm{x}_6 = \left(\frac{1}{\mathrm{n}}, \frac{2}{\mathrm{n}}, \frac{3}{\mathrm{n}} \ldots, 1\right)^{\mathrm{T}}$,

$\mathrm{x}_7 = \left(1 - \frac{1}{\mathrm{n}}, 1 - \frac{2}{\mathrm{n}}, 1 - \frac{3}{\mathrm{n}} \ldots, 0\right)^{\mathrm{T}}$.

The problems that the code solved are:

Problem 1. [18]

$$F_i(x) = 2x_1 - \sin x_i, \quad i = 1, \ldots, n$$

Problem 2. [18]

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

Problem 3. [18]

$$F(x) = Ax + h^2 X - 10h^2 e$$

where $x = (x_1, \ldots, x_n)^T, X = \left(x_1^3, \ldots, x_n^3\right)^T, e = (1, \ldots, 1)^T \in R^n, n = r^2$,

$$h = \frac{1}{x+1} \text{ and}$$

$$A = \begin{pmatrix} B & -I & & & \\ -I & B & -I & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -I \\ & & & -I & B \end{pmatrix}$$

Where I is the $r \times r$ identity matrix and

$$B = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 4 \end{pmatrix}$$

Problem 4. [36]

$$F_i(x) = x_i - \frac{1}{n}x_1^2 + \frac{1}{n}\sum_{i=1}^{n} x_i + i \quad i = 1, \ldots, n$$

Problem 5. [13]

$$F_i(x) = x_i - e^{\cos\left(\frac{x_{i-1}+x_i+x_{i+1}}{n+1}\right)}, \quad i = 2, \ldots, n-1$$

Problem 6. [38] $F : R^n \to R^n$

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

where $F(x) = (F_1(x), \ldots, F_n(x))^T, x = (x_1, \ldots, x_n)^T$.

Numerical results are viewed in tables [3.1,3.2], the first table consist of N_i (number of iterations), N_f (number of functions evaluations) of these algorithms and the second table consist of CPU times of these algorithms.

Table 3.1: Numerical Results (N_i, N_f)

P.	**Dim.**	**S.P**	**D.H**		**BFGS**	
			N_i	N_f	N_i	N_f
P_1	50000	x_0	14	132	1012	7148
	50000	x_1	14	132	1012	7148
	50000	x_2	17	128	169	893
	50000	x_3	17	128	169	893
	50000	x_4	32	165	15	81
	50000	x_5	8	55	24	116
	50000	x_6	20	145	69	386
	50000	x_7	18	126	58	267
P_2	50000	x_0	20	139	1012	7148
	50000	x_1	18	133	1067	7534
	50000	x_2	17	94	169	893
	50000	x_3	15	105	227	1421
	50000	x_4	71	376	15	81
	50000	x_5	20	97	24	116
	50000	x_6	27	178	74	441
	50000	x_7	21	107	70	402

Numerical Results (N_i, N_f) - Continued

P.	Dim.	S.P	D.H		BFGS	
			N_i	N_f	N_i	N_f
P_3	10000	x_0	37658	272108	353152	2489793
	10000	x_1	14146	92524	131613	930106
	10000	x_2	69589	473180	446312	3081375
	10000	x_3	45121	266231	262062	1686274
	10000	x_4	72777	406549	114736	746208
	10000	x_5	23854	107939	31376	137154
	10000	x_6	17718	101985	51478	309810
	10000	x_7	17708	101670	51473	309765
P_4	5000	x_0	90	867	212753	2451872
	5000	x_1	77	661	209480	2410607
	5000	x_2	89	852	212479	2448413
	5000	x_3	95	936	211927	2441445
	5000	x_4	89	852	212204	2444941
	5000	x_5	89	852	212229	2445257
	5000	x_6	89	852	212289	2446007
	5000	x_7	88	837	212391	2447308
P_5	50000	x_0	20	87	624	3783
	50000	x_1	22	116	2062	14464
	50000	x_2	20	87	190	984
	50000	x_3	20	87	624	3783
	50000	x_4	20	87	191	998

Numerical Results (N_i, N_f) - Continued

P.	Dim.	S.P	D.H		BFGS	
			N_i	N_f	N_i	N_f
	50000	x_5	20	87	191	998
	50000	x_6	20	87	191	998
	50000	x_7	20	87	191	998
P_6	50000	x_0	117	735	1948	13643
	50000	x_1	36	233	2181	15298
	50000	x_2	161	525	103	312
	50000	x_3	40	239	108	579
	50000	x_4	99	200	49	100
	50000	x_5	5168	10338	2584	5170
	50000	x_6	195	515	93	244
	50000	x_7	195	515	93	244

The numerical results in table 3.1 showed that the proposed algorithm D.H is better in its performance than the well-known algorithm BFGS after comparing them.

The results in red and blue colours represent the points where the proposed algorithm achieved the best performance in terms of N_i and N_f, respectively.

Table 3.2: Numerical Results (CPU time)

P.	Dim.	S.P	CPU time	
			D.H	BFGS
P_1	50000	x_0	0.296875	13.85937
	50000	x_1	0.234375	13.57813
	50000	x_2	0.203125	1.21875
	50000	x_3	0.1875	1.15625
	50000	x_4	0.1875	0.109375
	50000	x_5	0.140625	0.203125
	50000	x_6	0.203125	0.453125
	50000	x_7	0.171875	0.390625
P_2	50000	x_0	0.375	14.35936
	50000	x_1	0.21875	14.65625
	50000	x_2	0.140625	1.328125
	50000	x_3	0.234375	2.09375
	50000	x_4	0.65625	0.09375
	50000	x_5	0.203125	0.15625
	50000	x_6	0.3125	0.59375
	50000	x_7	0.109375	0.578125
P_3	10000	x_0	0.491094	5.787359
	10000	x_1	0.223875	2.169641
	10000	x_2	3.31798	7.117422
	10000	x_3	1.036875	3.923203
	10000	x_4	1.430219	1.729875

Numerical Results (CPU time)- Continued

P.	Dim.	S.P	CPU time	
			D.H	BFGS
	10000	x_5	0.344469	0.317297
	10000	x_6	0.219719	0.718719
	10000	x_7	0.309266	0.719062
P_4	5000	x_0	0.140625	3.819063
	5000	x_1	0.09375	3.747188
	5000	x_2	0.140625	3.813906
	5000	x_3	0.109375	4.693125
	5000	x_4	0.140625	6.542188
	5000	x_5	0.125	4.028594
	5000	x_6	0.046875	4.030313
	5000	x_7	0.09375	3.795469
P_5	50000	x_0	0.28125	11.015625
	50000	x_1	0.25	40.4375
	50000	x_2	0.21875	2.84375
	50000	x_3	0.203125	10.28125
	50000	x_4	0.203125	2.78125
	50000	x_5	0.203125	2.578125
	50000	x_6	0.203125	2.875
	50000	x_7	0.203125	2.484375

Numerical results (CPU time) - Continued

P.	Dim.	S.P	CPU time	
			D.H	BFGS
P_6	50000	x_0	1.21875	23.390625
	50000	x_1	0.34375	26.0625
	50000	x_2	0.671875	0.421875
	50000	x_3	0.28125	0.6875
	50000	x_4	0.25	0.109375
	50000	x_5	6.484375	7.421875
	50000	x_6	0.53125	0.28125
	50000	x_7	0.59375	2.328125

The numerical results in table 3.2 showed that the proposed algorithm D.H is better in its performance than the algorithm BFGS after comparing them.

The results in red color represent the points where the proposed algorithm achieved the best performance in terms of CPU time.

Chapter 4

Modified Technique to Solve Non Linear Systems of Equations

4.1 Introduction

In this chapter, a new line search algorithm has been proposed is a modified of Spectral gradient projection method (SGPM), which combines the modified spectral gradient method and projection method, as a method for solving non-linear equations systems. This approach can be more efficient because it requires less CPU time, less function evaluation, and fewer iterations. Furthermore, if this equation is (LC) and monotone then we proved that this technique converges globally. The numerical results show how effective and beneficial this method is for solving non-linear systems of equations.

The non-linear equations system is generally regarded as:

$$F(x) = 0$$

$F : R^n \to R^n$ is continuous. The following requirement for monotonicity in $F(x)$ must be met for non-linear equations to be considered monotone.

$$\langle F(x) - F(y), x - y \rangle \geq 0, \forall x, y \in R^n$$

The new algorithm which combines modified spectral gradient method [3] and projection method [30] is used for solving the solution non-linear monotone equations systems and it will be proven to be globally convergent and well-defined. By comparing the new algorithm presented with the SGPM approach in [38] and the BFGS approach in [40], it will be more competent.

Now, we will state formally the new algorithm :

4.2 The Proposed Algorithm

The steps of the proposed Algorithm for solving system of nonlinear equations will be mentioned.

Step 0: Select any initial point $x_0 \in R^n$ and $\mu, \sigma, \theta \in (0,1), \beta \in (0, \frac{1}{4}]$. Set $k = 0$.

Step 1: Determine d_k using:

$$d_k = \begin{cases} -F(x_k) & \text{if } k = 0, \\ -\sigma_k F(x_k) & \text{otherwise,} \end{cases} \tag{4.1}$$

Where, $\sigma_k = \frac{r_k^T r_k}{y_k^T r_k}, r_k = x_{k+1} - x_k$, $y_k = F(x_{k+1}) - F(x_k) + s r_k$ s $= \frac{1}{\|d_k\|}$

If $d_k = 0$, Stop.

Step 2: Select step length $\alpha_k = \mu \beta^{h_k}$ s.t. h_k is the smallest non-negative integer h satisfying :

$$-\langle F(x_k + \alpha_k d_k), d_k \rangle \geq \sigma \theta_k \alpha_k \|d_k\|^2 \tag{4.2}$$

$\theta_k = \frac{1}{\theta + \|d_k\|^2}$

Step 3: Calculate

$$x_{k+1} = x_k - \frac{\langle F(z_k), x_k - z_k \rangle}{\| F(z_k) \|^2} F(z_k) \tag{4.3}$$

Let $k = k + 1$, then move to **step 1**.

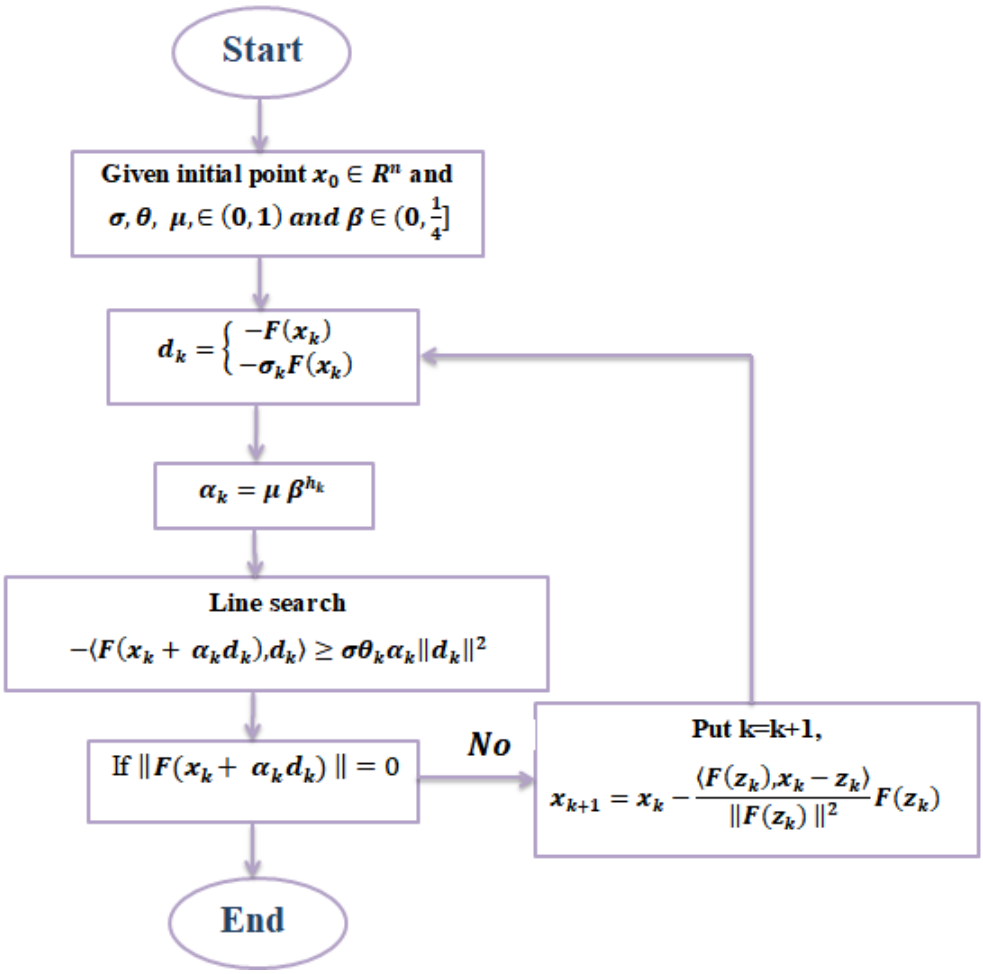

Figure 4.1: Flowchart of Proposed Algorithm

Now, we'll show how inequality (4.2) is well-defined. Assume that line search is not achieved, if we take $\lim_{h\to\infty}$ for two sides:

$$-\lim_{h\to\infty} \langle F\left(x_k + \mu\beta^{h_k} d_k\right), d_k\rangle < \lim_{h\to\infty} \sigma\theta_k \mu\beta^{h_k} |\, \|d_k\|^2$$

$$-\langle F\left(x_k\right), d_k\rangle < 0 \qquad \text{when } x_k = z_k - \alpha_k d_k$$

$$-\langle F\left(z_k - \alpha_k d_k\right), d_k\rangle < 0$$

$$\alpha_k \langle F\left(z_k + d_k\right), d_k\rangle < 0$$

$$\alpha_k \|F\left(z_k\right)\| \|d_k\|^2 < 0.$$

There is a contradiction; hence, the line search is well-defined.

4.2.1 Convergence Property

We can now determine the result of convergence for our proposed Algorithm by the next theorem analogous way [30].

Theorem 4.2.1.1 *Let* $\{x_k\}$ *be any sequence generated the proposed algorithm,* F *is monotone and (LC), as well as the set of solutions to* $F(x)$ *is non-empty. For any* $\hat{x}$ *achieving* $F(\hat{x}) = 0$ *:*

$$\| x_{k+1} - \hat{x} \|^2 \leq \| x_k - \hat{x} \|^2 - \| x_{k+1} - x_k \|^2$$

especially, $\{x_k\}$ *is bounded. Also, it must be achieved then, either* $\{x_k\}$ *is finite and the final iterate yields a solution, or* $\{x_k\}$ *is infinite and*

$$\lim_{k \to \infty} \| x_{k+1} - x_k \| = 0$$

$\{x_k\}$ *converges to some* $\hat{x}$ *s.t.* $F(\hat{x}) = 0$.

Proof: firstly, assuming that there is an end to the algorithm at iteration k, hence either $d_k = 0$, we get $F(x_k) = 0$, then x_k is a solution of inequality (4.2).

Now, let $d_k \neq 0$, $\forall k$, then $\{x_k\}$ an infinite sequence, is generated. From inequality (4.2):

$$\begin{aligned}
\langle F(z_k), x_k - z_k \rangle &= \langle F(z_k), x_k - (x_k + \alpha_k d_k) \rangle \qquad \text{since } z_k = x_k + \alpha_k d_k \\
&= \langle F(z_k), x_k - x_k - \alpha_k d_k \rangle \\
&= \langle F(z_k), -\alpha_k d_k \rangle \\
&= -\alpha_k \langle F(z_k), d_k \rangle \\
&= -\alpha_k \langle F(x_k + \alpha_k d_k), d_k \rangle \\
&\geq \sigma \theta_k \alpha_k^2 \|d_k\|^2 > 0
\end{aligned}$$

Then,

$$\langle F\left(z_k\right), x_k - z_k\rangle \geq \sigma\theta_k\alpha_k^2 \left\|d_k\right\|^2 > 0 \tag{4.4}$$

Assume that $\hat{x}$ be any solution of $F(x) = 0$ and $F(\hat{x}) = 0$. From lemma (3.4.2.1), equation (4.3) and inequality (4.4) lead to the following:

$$\| x_{k+1} - \hat{x} \|^2 \leq \| x_k - \hat{x} \|^2 - \| x_{k+1} - x_k \|^2 \tag{4.5}$$

Especially, $\{\| x_k - \hat{x} \|\}$ is non-increasing and convergent. Consequently, $\{x_k\}$ will be bounded and obtaining:

$$\lim_{k\to\infty} \| x_{k+1} - x_k \| = 0 \tag{4.6}$$

By inequality (3.12) in lemma (3.4.2.1), It is obvious that $\{d_k\}$ holds to be bounded and $\{z_k\}$ also bounded.

From equation (4.3) and inequality (4.4):

$$\begin{aligned} x_{k+1} - x_k &= -\frac{\langle F\left(z_k\right), x_k - z_k\rangle}{\left\|F\left(z_k\right)\right\|^2} F\left(z_k\right) \\ x_{k+1} - x_k &= \frac{\alpha_k \langle F\left(z_k\right), d_k\rangle}{\left\|F\left(z_k\right)\right\|^2} F\left(z_k\right) \\ &\geq \sigma\alpha_k^2 \left\|d_k\right\|^2 \end{aligned}$$

hence:

$$\|x_{k+1} - x_k\| \geq \alpha_k^2 \left\|d_k\right\|^2 \tag{4.7}$$

From equation (4.6) and inequality (4.7):

$$\lim_{k\to\infty} \alpha_k \|d_k\| = 0 \tag{4.8}$$

Also, by inequality (3.12):

$$\lim_{k\to\infty} \inf \|F(x_k)\| = 0, \text{when} \lim_{k\to\infty} \inf \|d_k\| = 0.$$

Since $\{x_k\}$ is bounded and F is continuous, $\{x_k\}$ has an accumulation point at which x^* s.t. $F(x^*) = 0$. We get $\{\| x_k - x^* \|\}$ converges from inequality (3.14). As a result, $\{x_k\}$ converges to x^*.

$$\lim_{k\to\infty} \inf \|F(x_k)\| > 0, \text{when} \lim_{k\to\infty} \inf \|d_k\| > 0.$$

By (4.8):

$$\lim_{k\to\infty} \alpha_k = 0 \tag{4.9}$$

Inequality (4.2) gives us:

$$-\left\langle F\left(x_k + \mu\beta^{h_{k-1}} d_k\right), d_k\right\rangle < \sigma\theta_k \mu\beta^{h_{k-1}} \|d_k\|^2 \tag{4.10}$$

Since $\{x_k\}$, $\{d_k\}$ are bounded, in order to select a subsidiary sequence, let $k \to \infty$ in inequality (4.10):

$$-\langle F(x^*), d^*\rangle \leq 0 \tag{4.11}$$

s.t. x^* and d^* are sub-sequences limiting that chosen. Moreover, by inequality (3.12) and already known argument:

$$-\langle F(x^*), d^*\rangle > 0 \tag{4.12}$$

Inequality (4.11) and inequality (4.12) are a contradiction. Then it's impossible to obtain

$$\lim_{k\to\infty} \inf \|F(x_k)\| > 0$$

This proof is finished.

4.3 Numerical Resuls

We will compare the effectiveness of the new approach (D.A) presented earlier in this paper with the following algorithms.

SGPM: From the source [38].

BFGS: From the source [40].

All codes are written in MATLAB version $R2014a$ and all experiments have been run on a computer with $8GB$ of RAM and $CPU\,2.40\,GHz$. The aim of running the codes is to compare the outcomes of the new algorithm (D.A) with the aforementioned algorithm. For (D.A) the parameters are determined as follows: $\mu = 0.2, \beta = 0.25, \theta = 0.3, \sigma = 0.2$. For SGPM $\beta = 0.4, \sigma = 0.01, r = 0.001$ and BFGS $\beta = 0.3, \sigma = 0.3$.

The specific dimensions for these algorithms comparisons are set at $10000 - 50000$,To the main points we mentioned in the numerical results in the previous chapter. Problems solved by the law are the same problems in the previous algorithm [1,2,3,5,6] except:

Problem 4. [6]

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

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

$$A_1 = \begin{pmatrix} \frac{5}{2} & 1 & & & \\ 1 & \frac{5}{2} & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \frac{1}{5} \\ & & & 1 & \frac{1}{2} \end{pmatrix}.$$

Numerical results are viewed in tables 4.1, 4.2, the first table consist of N_i (number of iterations) and N_f (number of functions evaluations) of these algorithms and the second table consist of CPU times of these algorithms.

Table 4.1: Numerical Results (N_i, N_f)

P.	Dim.	S.P	D.A		SGPM		BFGS	
			N_i	N_f	N_i	N_f	N_i	N_f
P_1	50000	x_0	48	249	108	1169	102	1099
	50000	x_1	48	249	108	1169	102	1099
	50000	x_2	46	202	108	1169	124	1321
	50000	x_3	46	202	108	1169	124	1321
	50000	x_4	132	133	110	1191	140	150
	50000	x_5	36	53	108	1169	152	1576
	50000	x_6	46	60	108	1169	162	881
	50000	x_7	40	53	108	1169	160	879
P_2	50000	x_0	48	249	108	1169	120	1297
	50000	x_1	72	664	108	1169	118	1275
	50000	x_2	46	202	108	1169	98	1038
	50000	x_3	44	212	108	1189	122	1310
	50000	x_4	32	133	110	1191	76	77
	50000	x_5	36	53	108	1169	32	244
	50000	x_6	150	182	108	1169	166	962
	50000	x_7	146	178	108	1169	164	960
P_3	10000	x_0	2660	2662	64378	64400	2678	2859
	10000	x_1	4950	4952	71498	71540	5100	6777
	10000	x_2	766	767	52476	52498	738	751
	10000	x_3	858	859	62064	62086	828	841

Numerical Results (N_i, N_f) - Continued

P.	Dim.	S.P	D.A		SGPM		BFGS	
			N_i	N_f	N_i	N_f	N_i	N_f
	10000	x_4	134	135	226	247	186	187
	10000	x_5	180	181	51326	51348	138	139
	10000	x_6	546	547	22544	22566	536	593
	10000	x_7	546	547	22540	22562	536	593
P_4	10000	x_0	46	867	96	1057	340	2724
	10000	x_1	68	661	96	1057	1162	10942
	10000	x_2	52	852	96	1057	114	714
	10000	x_3	170	936	96	1057	170	1102
	10000	x_4	48	152	96	1057	58	170
	10000	x_5	46	852	96	1057	46	101
	10000	x_6	56	852	96	1057	68	292
	10000	x_7	56	837	96	1057	72	299
P_5	50000	x_0	66	444	120	1301	122	1314
	50000	x_1	50	273	118	1279	114	1229
	50000	x_2	24	173	120	1301	390	4253
	50000	x_3	52	328	120	1301	120	1289
	50000	x_4	418	4351	120	1301	126	1352
	50000	x_5	52	309	120	1301	124	1330
	50000	x_6	70	559	120	1301	292	3177
	50000	x_7	48	259	120	1301	328	3573

Numerical Results (N_i, N_f) - Continued

P.	Dim.	S.P	D.A		SGPM		BFGS	
			N_i	N_f	N_i	N_f	N_i	N_f
P_6	50000	x_0	24	47	118	1279	114	1229
	50000	x_1	44	387	118	1279	108	1163
	50000	x_2	28	49	120	1301	122	1273
	50000	x_3	36	119	120	1301	110	1174
	50000	x_4	40	41	122	1323	40	41
	50000	x_5	22	23	64	706	22	23
	50000	x_6	112	113	120	1301	160	422
	50000	x_7	112	113	120	1301	160	422

The numerical results in table 4.1 showed that the proposed algorithm D.A is better in its performance than the algorithms BFGS and SGPM after comparing them.

The results in red and blue colours represent the points where the proposed algorithm achieved the best performance in terms of N_i and N_f, respectively.

Table 4.2: Numerical Results (CPU time)

P.	Dim.	S.P	CPU time		
			D.A	SGPM	BFGS
P_1	50000	x_0	1.1875	15.375	15.65625
	50000	x_1	1	14.546875	13.96875
	50000	x_2	0.71875	15.53125	22.671875
	50000	x_3	0.65625	15.65625	23.234375
	50000	x_4	0.390625	16.671875	0.25
	50000	x_5	0.109375	15.40625	27.171875
	50000	x_6	0.1875	16.234375	2.46875
	50000	x_7	0.171875	16.546875	2.53125
P_2	50000	x_0	1.03125	11.5	18.734375
	50000	x_1	3.609375	15.265625	20.96875
	50000	x_2	0.625	22.171875	24.8125
	50000	x_3	0.609375	19.796875	17
	50000	x_4	0.390625	11.359375	0.25
	50000	x_5	0.09375	9.3125	0.59375
	50000	x_6	0.390625	10.078125	2.265625
	50000	x_7	0.5	10.125	2.21875
P_3	10000	x_0	6.90625	2.101875	7.5
	10000	x_1	12.890625	2.3415625	17.5
	10000	x_2	1.89062	1.6668750	1.890625
	10000	x_3	2.171875	1.975	2.328125
	10000	x_4	0.40625	0.65625	0.71875

Numerical Results (CPU time) - Continued

P.	Dim.	S.P	CPU time		
			D.A	SGPM	BFGS
	10000	x_5	0.46875	1.51171875	0.296875
	10000	x_6	1.4375	0.71953125	1.5
	10000	x_7	1.34375	0.71421875	1.359375
P_4	10000	x_0	0.03125	0.546875	1.078125
	10000	x_1	0.0625	0.484375	4.078125
	10000	x_2	0.03125	0.5	0.25
	10000	x_3	0.109375	0.78125	0.34375
	10000	x_4	0.03125	0.6875	0.0625
	10000	x_5	0.03125	0.6875	0.03125
	10000	x_6	0.0625	0.671875	0.09375
	10000	x_7	0.0625	0.59375	0.09375
P_5	50000	x_0	1.515625	28.296875	18.515625
	50000	x_1	0.71875	24.84375	15.5
	50000	x_2	0.40625	13.921875	56.25
	50000	x_3	0.8125	10.328125	16.8125
	50000	x_4	18.359375	11.53125	18.265625
	50000	x_5	0.90625	10.703125	17.34375
	50000	x_6	1.78125	12.421875	30.328125
	50000	x_7	0.671875	11.90625	31.25

Numerical Results (CPU time) - Continued

P.	Dim.	S.P	CPU time		
			D.A	SGPM	BFGS
P_5	50000	x_0	0.140625	10.5625	15.859375
	50000	x_1	1.875	9.4375	13.3125
	50000	x_2	0.109375	10.296875	14.21875
	50000	x_3	0.234375	10.6875	13.562
	50000	x_4	0.140625	2	0.140625
	50000	x_5	0.078125	1.109375	0.078125
	50000	x_6	0.28125	10.265625	0.703125
	50000	x_7	0.296875	12.0625	0.671875

The numerical results in table 4.2 showed that the proposed algorithm D.A is better in its performance than the algorithms BFGS and SGPM after comparing them.

The results in red color represent the points where the proposed algorithm achieved the best performance in terms of CPU time.

Chapter 5

A Modified Conjugate Gradient Technique to Solve Non Linear Systems of Equations

5.1 Introduction

In this chapter, a new three-term conjugate gradient method, was suggested to solve non linear system of equations. This technique combines three different techniques: line search technique, conjugate gradient technique, and projection technique. We prove that, under plausible assumptions, the suggested technique is globally convergent. This approach may be more successful because it requires less CPU time, less function evaluation, and fewer iterations. As a result of its simplicity and memory restrictions, Additionally, it works well with large-scale equations. The numerical results show how effective and beneficial this strategy is for solving non-linear equations systems.

The system of non-linear equations is as previously stated:

$$F(x) = 0$$

s.t. $F : \Omega \subset R^n \rightarrow R^n$ be non-linear and continuous function, Ω is non-empty closed convex. Non-linear equations must satisfy the following condition in order to be called monotone in $F(x)$.

$$\langle F(x) - F(y), x - y \rangle \geq 0, \forall x, y \in R^n$$

The projection method is a suitable method for solving large-scale equations. In particular, in [30] presented a Newton approach that combines Newton and projection methods. It is interesting that most recently developed approaches for non-linear monotone equations systems use the projection method suggested in [30], for example see [1, 23, 6, 18]. Using the current point x_k, it's possible to calculate the

following point x_{k+1} by:

$$x_{k+1} = P_\Omega[x_k - sF(x_k)]$$

when s is a positive scalar that satisfies the equation $0 < S_L \leq s \leq S_U < \frac{2}{L}$. The projection method is considered as one of the simplest numerical methods to solve optimization problems. The projection operator:

$$P_\Omega : R^n \to \Omega, \forall x \in R^n.$$

The definition of $P_\Omega(x)$, which represents the orthogonal projection of x, is

$$P_\Omega(x) = argmin\| \ y - x \ \|, \forall y \in R^n \tag{5.1}$$

where $\Omega \subseteq R^n$ is a convex set [5].

Now, we will state formally the new algorithm :

5.2 The Proposed Algorithm

The steps of the proposed Algorithm for solving system of nonlinear equations will be mentioned.

Step 0: Select any initial point $x_0 \in R^n$ and $\lambda, \gamma, \rho, \text{and} \mu \in (0, 1)$, Set $k = 0$.

Step 1: Determine d_k using:

$$d_k = \begin{cases} -F_k & \text{if } k = 0 \\ -F_k + \beta_k s_{k+} \theta_k y_k & \text{if } k \geq 1 \end{cases} \tag{5.2}$$

where $\beta_k = \frac{\gamma F_k^T y_k}{s_k y_k}, \theta_k = \frac{\gamma F_k^T s_k}{s_k y_k}, \gamma$ is a positive constant,
$s_k = F_{k+1} - F_k + r$, $y_k = x_{k+1} - x_k$ and $r = \frac{\rho \|F(z_k)\|}{\|d_k\|}$

If $d_k = 0$, stop.

Step 2: Select $\alpha_k = \lambda^{w_k}$ s.t. w_k is the smallest non-negative integer w satisfying :

$$-\langle F\left(x_k + \alpha_k d_k\right), d_k\rangle \geq \mu\alpha_k \left\|F(z_k)\right\| \left\|d_k\right\|^2 \tag{5.3}$$

Find trail point $z_k = x_k + \alpha_k d_k$

If $\| F(z_k) \| = 0$, stop.

Step 3: Calculate

$$x_{k+1} = x_k - \frac{\langle F(z_k), x_k - z_k\rangle}{\| F(z_k) \|^2} F(z_k) \tag{5.4}$$

Let $k = k + 1$ and back to **Step1**.

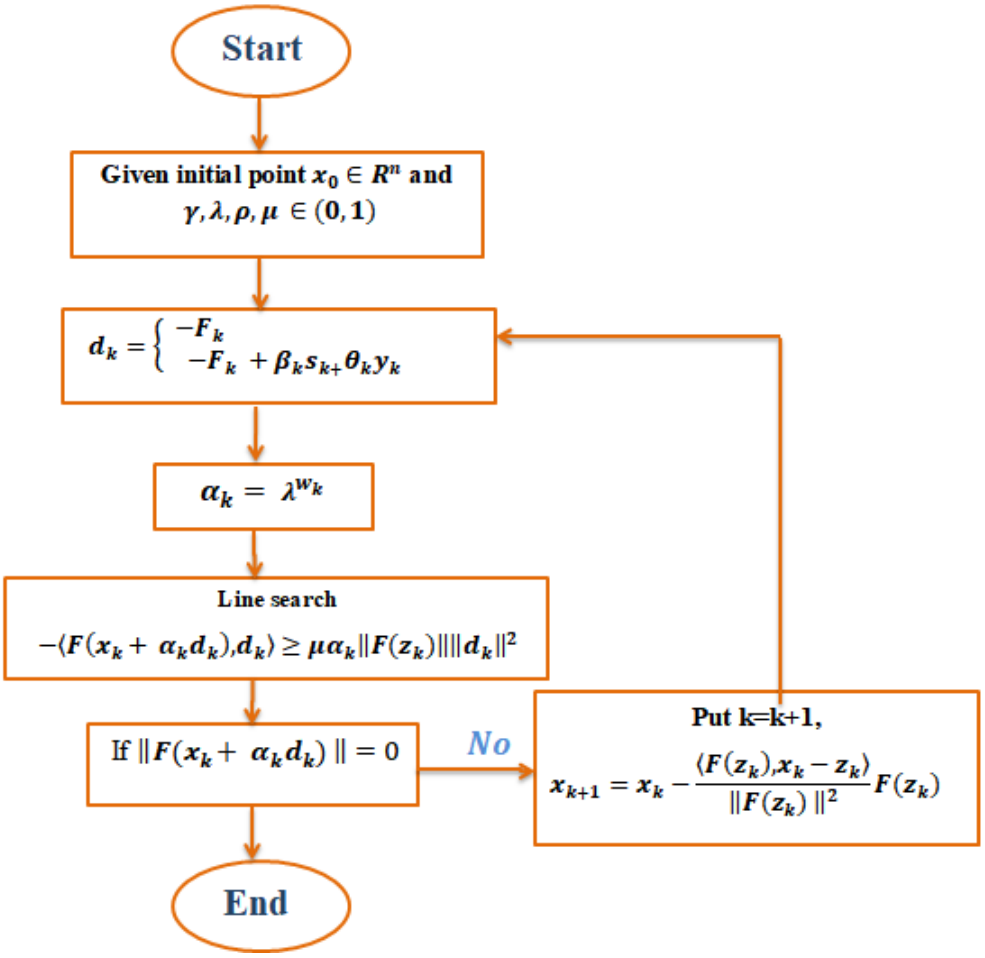

Figure 5.1: Flowchart of Proposed Algorithm

We can show how inequality (5.3) is well-defined in the same way as algorithms (D.H) and (D.A).

The following lemma states some well-known properties of the projection operator:

Lemma 5.2.1 *[17] The following conditions are true if $\Omega \subset R^n$ is a convex and closed set and $P_\Omega(x)$ is a projection of x onto Ω, $\forall x, y \in R^n$.*

1. $\langle \mathrm{P}_\Omega(\mathrm{x}) - \mathrm{x}, \mathrm{z} - \mathrm{P}_\Omega(\mathrm{x}) \rangle \geq 0, \forall \mathrm{x} \in \mathrm{R}^\mathrm{n}$ *and* $\forall \mathrm{z} \in \Omega$.
2. $\| \mathrm{P}_\Omega(\mathrm{x}) - \mathrm{P}_\Omega(\mathrm{y}) \| \leq \| \mathrm{x} - \mathrm{y} \|, \forall \mathrm{x}, \mathrm{y} \in \mathrm{R}^\mathrm{n}$.
3. $\langle \mathrm{P}_\Omega(\mathrm{x}) - \mathrm{P}_\Omega(\mathrm{y}), \mathrm{x} - \mathrm{y} \rangle \geq 0, \forall \mathrm{x} \in \mathrm{R}^\mathrm{n}$ *and* $\forall \mathrm{z} \in \Omega$.

5.2.1 Convergence Property

We can now determine the result of convergence for our proposed Algorithm by the next theorem analogous way [30].

Theorem 5.2.1.1 *Let $\{x_k\}$ be any sequence generated the proposed algorithm, F is monotone and (LC), as well as the set of solutions to $F(x)$ is non-empty. For any $\hat{x}$ achieving $F(\hat{x}) = 0$:*

$$\| x_{k+1} - \hat{x} \|^2 \leq \| x_k - \hat{x} \|^2 - \| x_{k+1} - x_k \|^2$$

especially, $\{x_k\}$ is bounded. Also, it must be achieved then, either $\{x_k\}$ is finite and the final iterate yields a solution, or $\{x_k\}$ is infinite and

$$\lim_{k \to \infty} \| x_{k+1} - x_k \| = 0$$

$\{x_k\}$ converges to some $\hat{x}$ s.t. $F(\hat{x}) = 0$.

Proof: firstly, assuming that there is an end to the algorithm at iteration

k, hence either $d_k = 0$, we get $F(x_k) = 0$, then x_k is a solution of inequality (5.3).

Now, let $d_k \neq 0$, $\forall k$, then $\{x_k\}$ an infinite sequence, is generated. From inequality (5.3):

$$\begin{aligned}\langle F(z_k), x_k - z_k\rangle &= \langle F(z_k), x_k - (x_k + \alpha_k d_k)\rangle \qquad \text{since } z_k = x_k + \alpha_k d_k \\ &= \langle F(z_k), -\alpha_k d_k\rangle \\ &\geq \mu\alpha_k^2 \|F(z_k)\| \|d_k\|^2 > 0\end{aligned}$$

Then,

$$\langle F(z_k), x_k - z_k\rangle \geq \mu\alpha_k^2 \|F(z_k)\| \|d_k\|^2 > 0 \tag{5.5}$$

Assume that $\hat{x}$ be any solution of $F(x) = 0$ and $F(\hat{x}) = 0$. From lemma (3.4.2.1), equation (5.4) and inequality (5.5) lead to the following:

$$\| x_{k+1} - \hat{x} \|^2 \leq \| x_k - \hat{x} \|^2 - \| x_{k+1} - x_k \|^2 \tag{5.6}$$

Especially, $\{\| x_k - \hat{x} \|\}$ is non-increasing and convergent. Consequently, $\{x_k\}$ will be bounded and obtaining:

$$\lim_{k\to\infty} \| x_{k+1} - x_k \| = 0 \tag{5.7}$$

By inequality (3.12) in lemma (3.4.2.1), It is obvious that $\{d_k\}$ holds to be bounded and $\{z_k\}$ also bounded.

From equation (5.4) and inequality (5.5):

$$\begin{aligned}x_{k+1} - x_k &= -\frac{\langle F(z_k), x_k - z_k\rangle}{\|F(z_k)\|^2} F(z_k) \\ x_{k+1} - x_k &= \frac{\alpha_k \langle F(z_k), d_k\rangle}{\|F(z_k)\|^2} F(z_k) \\ &\geq \mu\alpha_k^2 \|d_k\|^2\end{aligned}$$

hence:

$$\|x_{k+1} - x_k\| \geq \alpha_k^2 \|d_k\|^2 \geq 0 \tag{5.8}$$

From equation (5.7) and inequality (5.8):

$$\lim_{k\to\infty} \alpha_k \|d_k\| = 0 \tag{5.9}$$

Also, by inequality (3.12):

$$\lim_{k\to\infty} \inf \|F(x_k)\| = 0, \text{when} \lim_{k\to\infty} \inf \|d_k\| = 0.$$

Since $\{x_k\}$ is bounded and F is continuous, $\{x_k\}$ has an accumulation point at which x^* s.t. $F(x^*) = 0$. We get $\{\| x_k - x^* \|\}$ converges from equation (3.14). As a result, $\{x_k\}$ converges to x^*.

$$\lim_{k\to\infty} \inf \|F(x_k)\| > 0, \text{when} \lim_{k\to\infty} \inf \|d_k\| > 0.$$

By equation (5.9):

$$\lim_{k\to\infty} \alpha_k = 0 \tag{5.10}$$

Inequality (5.3) gives us:

$$-\left\langle F\left(x_k + \lambda^{w_{k-1}} d_k\right), d_k \right\rangle < \mu \lambda^{w_{k-1}} \|F(z_k)\| \|d_k\|^2 \tag{5.11}$$

The sequence $\{d_k\}$ is bounded if a requirement of lemma 5.2.1 is satisfied. If there exist $\gamma > 0$ [34],

$$\begin{aligned}
\|d_k\| &\le \|-F_k\| + \|\beta_k s_{k-1}\| + \|\theta_k y_{k-1}\| \\
&= \|F_k\| + \frac{\gamma \left\|F_k^T\right\| \|y_{k-1}\|}{\|s_{k-1}\| \|y_{k-1}\|} \|s_{k-1}\| + \frac{\gamma \left\|F_k^T\right\| \|s_{k-1}\|}{\|s_{k-1}\| \|y_{k-1}\|} \|y_{k-1}\| \\
&= \|F_k\| + 2\gamma \|F_k\| \\
&\le 3\gamma \|F_k\| .
\end{aligned}$$

Since $\{x_k\}$, $\{d_k\}$ are bounded, in order to choose a subsidiary sequence, let $k \to \infty$ in inequality (5.11):

$$-\langle F(x^*), d^* \rangle \le 0 \tag{5.12}$$

s.t. x^* and d^* are sub-sequences limiting that chosen. Moreover, by (3.12) and already known argument,

$$-\langle F(x^*), d^* \rangle > 0 \tag{5.13}$$

Inequality (5.12) and inequality (5.13) are a contradiction. Then it's impossible to obtain:

$$\lim_{k\to\infty} \inf \|F(x_k)\| > 0$$

This proof is finished.

5.3 Numerical Resuls

The effectiveness of the new approach (DHA) previously presented in this paper will be compared to the following algorithms:

CGM: From the source [35].

CGDM: From the source [37].

All codes are written in MATLAB version $R2014a$ and all experiments have been run on a computer with $8GB$ of RAM and CPU $2.40\,GHz$.

The aim of running the codes is to compare the outcomes of the new algorithm (DHA) with the aforementioned algorithm. All algorithms are completed when $\|F(x_k)\| \leq 10^{-8}$ or $\|F(z_k)\| \leq 10^{-8}$.For our method, the parameters are determined as follows: $\lambda = 0.7, \mu = 0.001, \rho = 0.2, \gamma = 0.04$. Other methods use a parameters that comes from [35, 37].

The specific dimensions for these algorithms comparisons are set at $10000 - 50000$. Problems solved by the law and the points are the same it in the algorithm (D.A).

Numerical results are viewed in tables 5.1, 5.2, the first table consists of N_i (number of iterations) and N_f (number of functions evaluations) of these algorithms and the second table consist of CPU times of these algorithms.

Table 5.1: Numerical Results (N_i, N_f)

P.	**Dim.**	**S.P**	**DHA**		**SATCGM**		**CGDPM**	
			N_i	N_f	N_i	N_f	N_i	N_f
P_1	50000	x_0	52	240	728	1670	58	181
	50000	x_1	52	240	728	1670	58	181
	50000	x_2	30	92	626	1068	52	154
	50000	x_3	30	92	626	1068	52	154
	50000	x_4	16	19	366	368	28	31
	50000	x_5	24	66	482	484	40	82
	50000	x_6	30	92	580	822	48	130
	50000	x_7	30	92	580	822	48	130
P_2	50000	x_0	52	240	728	1670	58	181
	50000	x_1	96	393	308	1010	88	245
	50000	x_2	30	92	626	1068	52	154
	50000	x_3	72	244	228	530	106	351
	50000	x_4	82	110	366	368	118	148
	50000	x_5	24	66	482	484	40	82

Numerical Results (N_i, N_f) - Continued

P.	Dim.	S.P	DHA		SATCGM		CGDPM	
			N_i	N_f	N_i	N_f	N_i	N_f
	50000	x_6	112	204	580	822	144	255
	50000	x_7	112	204	580	822	144	255
P_3	10000	x_0	1616	8572	161152	161154	64383	757131
	10000	x_1	1594	8437	179096	179558	71505	763103
	10000	x_2	1616	8572	131362	131364	52481	755122
	10000	x_3	1612	8544	155354	155356	62069	755141
	10000	x_4	1616	8568	133000	133002	51323	756274
	10000	x_5	1616	8568	128468	128470	22547	756132
	10000	x_6	1612	8544	56440	56442	22543	655754
	10000	x_7	1608	8520	56430	56432	433818	655570
P_4	10000	x_0	122	248	400	642	182	367
	10000	x_1	136	280	560	902	220	417
	10000	x_2	100	200	350	472	146	279
	10000	x_3	144	227	446	628	164	323
	10000	x_4	100	145	366	369	156	218
	10000	x_5	132	204	396	398	186	270
	10000	x_6	124	212	388	470	176	301
	10000	x_7	120	208	388	470	176	301

Numerical Results (N_i, N_f) - Continued

P.	Dim.	S.P	DHA		SATCGM		CGDPM	
			N_i	N_f	N_i	N_f	N_i	N_f
P_5	50000	x_0	42	165	786	2028	66	228
	50000	x_1	58	265	830	2292	68	250
	50000	x_2	34	116	674	1336	56	178
	50000	x_3	36	138	734	1716	62	204
	50000	x_4	36	118	708	1530	60	202
	50000	x_5	36	118	706	1548	60	202
	50000	x_6	34	116	694	1416	58	180
	50000	x_7	34	116	694	1416	58	180
P_6	50000	x_0	432	4797	211	1887	62186	62548
	50000	x_1	136	508	10039	141738	196	2377
	50000	x_2	408	4650	2380	30586	62164	62406
	50000	x_3	108	353	6279	81537	801	238
	50000	x_4	2558	2560	3680	40369	3586	3588
	50000	x_5	4352	43528	3017	33134	60940	60942
	50000	x_6	43784	43866	908	911	61296	61358
	50000	x_7	43784	43866	908	911	61296	61358

The numerical results in table 5.1 showed that the proposed algorithm DHA is better in its performance than the algorithms **S**ATCGM and CGDPM after comparing them.

The results in red and blue colours represent the points where the proposed algorithm achieved the best performance in terms of N_i and N_f, respectively.

Table 5.2: Numerical Results (CPU time)

P.	Dim.	S.P	CPU time		
			DHA	SATCGM	CGDPM
P_1	50000	x_0	0.578125	6.640625	0.421875
	50000	x_1	0.484375	7.1875	0.453125
	50000	x_2	0.140625	4.890625	0.328125
	50000	x_3	0.265625	5.140625	0.328125
	50000	x_4	0.0625	2.40625	0.125
	50000	x_5	0.1875	3.59375	0.203125
	50000	x_6	0.203125	4.3125	0.328125
	50000	x_7	0.265625	3.234375	0.25
P_2	50000	x_0	0.515625	7.125	0.53125
	50000	x_1	0.84375	4.609375	0.609375
	50000	x_2	0.203125	4.703125	0.359375
	50000	x_3	0.546875	1.8125	0.625
	50000	x_4	0.375	1.8125	0.4375
	50000	x_5	0.21875	2.546875	0.296875
	50000	x_6	0.640625	3.59375	0.671875
	50000	x_7	0.484375	3.1875	0.75
P_3	10000	x_0	1.90625	5.568594	2.158844
	10000	x_1	1.75	6.231406	2.157938
	10000	x_2	1.65625	4.056406	2.149172
	10000	x_3	1.765625	4.88625	2.137657
	10000	x_4	2.09375	4.119531	2.163203

Numerical Results (CPU time) - Continued

P.	Dim.	S.P	CPU time		
			DHA	SATCGM	CGDPM
	10000	x_5	2.03125	3.992812	2.325188
	10000	x_6	2.015625	1.745	1.998766
	10000	x_7	1.78125	1.747813	2.0235
P_4	10000	x_0	0.21875	0.640625	0.234375
	10000	x_1	0.25	0.8125	0.265625
	10000	x_2	0.3125	0.375	0.1875
	10000	x_3	0.328125	0.390625	0.1875
	10000	x_4	0.140625	0.34375	0.1875
	10000	x_5	0.21875	0.421875	0.203125
	10000	x_6	0.21825	0.40625	0.21875
	10000	x_7	0.3125	0.5	0.15625
P_5	50000	x_0	0.734375	11.421875	0.8125
	50000	x_1	0.984375	12.65625	0.734375
	50000	x_2	0.34375	8.390625	0.578125
	50000	x_3	0.515625	8.296875	0.609375
	50000	x_4	0.328125	6.46875	0.578125
	50000	x_5	0.40625	6.6875	0.625
	50000	x_6	0.359375	6.609375	0.59375
	50000	x_7	0.265625	6.21875	0.640625

Numerical Results (CPU time) - Continued

P.	Dim.	S.P	CPU time		
			DHA	SATCGM	CGDPM
P_6	50000	x_0	2.387969	0.09375	2.277031
	50000	x_1	0.0175	0.078125	0.05
	50000	x_2	2.4025	0.125	2.36
	50000	x_3	0.011875	0.125	0.05313
	50000	x_4	0.107188	0.1625	0.131406
	50000	x_5	2.241407	0.125	2.21875
	50000	x_6	2.554688	0.109375	2.26375
	50000	x_7	2.422969	0.446875	2.424469

The numerical results in table 5.2 showed that the proposed algorithm DHA is better in its performance than the algorithms BFGS and SGPM after comparing them.

The results in red color represent the points where the proposed algorithm achieved the best performance in terms of CPU time.

Chapter 6

Conclusions and Future Works

6.1 Conclusions

In this thesis, three new algorithms of the line search technique for solving non-linear systems of equations are suggested. Matlab application was used in the application of these algorithms and many results were obtained by the comparison technique .

It is clear from the numerical results of many problems with various initial points and dimensions described in the tables that are mentioned in the last three chapters it's possible to conclude that Due to their low memory requirements, ability to be applied easily, the new proposed algorithms in this thesis are significant and suitable for solving systems of non-linear equations.

The performance of the proposed algorithms (D.H), (D.A) and (DHA) is the most efficient and effective because it reduces N_i, N_f and CPU time. These algorithms have also been proven to be a global convergent, meaning they are very promising to solve non-linear systems.

6.2 Future Works

In the future, it is expected that this study will inspire a number of works, such as:

1. Solving the non-linear systems of equations by using a new approach that combines projection, conjugate and trust region techniques.

2. Combining trust region technique and projection technique for Solving nonlinear systems of equations.

3. Solving the non-linear systems of equations by using a new approach that combines line search and trust region techniques.

References

[1] Ahookhosh, M., Amini, K., and Bahrami, S. Two derivative-free projection approaches for systems of large-scale nonlinear monotone equations. *Numerical Algorithms 64* (2013), 21–42.

[2] Arora, R. K. *Optimization: algorithms and applications*. CRC press, 2015.

[3] Barzilai, J., and Borwein, J. M. Two-point step size gradient methods. *IMA journal of numerical analysis 8*, 1 (1988), 141–148.

[4] Burden, R. L., Faires, J. D., and Burden, A. M. *Numerical analysis*. Cengage learning, 2015.

[5] Cai, G., Gibali, A., Iyiola, O. S., and Shehu, Y. A new double-projection method for solving variational inequalities in banach spaces. *Journal of Optimization Theory and Applications 178* (2018), 219–239.

[6] Cheng, W. A prp type method for systems of monotone equations. *Mathematical and Computer Modelling 50*, 1-2 (2009), 15–20.

[7] Chong, E. K., and Żak, S. H. *An introduction to optimization*, vol. 75. John Wiley & Sons, 2013.

[8] Cruz, W. L., and Raydan, M. Nonmonotone spectral methods for large-scale nonlinear systems. *Optimization Methods and software 18*, 5 (2003), 583–599.

[9] Dennis Jr, J. E., and Schnabel, R. B. *Numerical methods for unconstrained optimization and nonlinear equations*. SIAM, 1996.

[10] Griewank, A. The "global" convergence of broyden-like methods with suitable line search. *The ANZIAM Journal 28*, 1 (1986), 75–92.

[11] Hashim, K. H., Dreeb, N. K., Dwail, H. H., Mahdi, M. M., Wasi, H., Shiker, M. A., and Hussein, H. A. A new line search method to solve the nonlinear systems of monotone equations. *Journal of Engineering and Applied Sciences 14* (2019), 10080–10086.

[12] Hassan Ibrahim, A., Kumam, P., Hassan, B. A., Bala Abubakar, A., and Abubakar, J. A derivative-free three-term hestenes–stiefel type method for constrained nonlinear equations and image restoration. *International Journal of Computer Mathematics 99*, 5 (2022), 1041–1065.

[13] Ishak, M., Marjugi, S., and June, L. A new modified conjugate gradient method under the strong wolfe line search for solving unconstrained optimization problems. *Mathematical Modeling and Computing 9*, 1 (2022), 111–118.

[14] Jiang, H., and Ralph, D. Global and local superlinear convergence analysis of newton-type methods for semismooth equations with smooth least squares. *Reformulation: nonsmooth, piecewise smooth, semismooth and smoothing methods* (1999), 181–209.

[15] Kelley, C. T. *Iterative methods for linear and nonlinear equations*. SIAM, 1995.

[16] Kelley, C. T. *Iterative methods for optimization*. SIAM, 1999.

[17] Koorapetse, M., Kaelo, P., and Offen, E. A scaled derivative-free projection method for solving nonlinear monotone equations. *Bulletin of the Iranian Mathematical Society 45* (2019), 755–770.

[18] Li, Q., and Li, D.-H. A class of derivative-free methods for large-scale nonlinear monotone equations. *IMA Journal of Numerical Analysis 31*, 4 (2011), 1625–1635.

[19] Matoušek, J., and Gärtner, B. *Understanding and using linear programming*, vol. 33. Springer, 2007.

[20] Moré, J. J., and Sorensen, D. C. Computing a trust region step. *SIAM Journal on scientific and statistical computing 4*, 3 (1983), 553–572.

[21] Muhammed, A. A., Kumam, P., Abubakar, A. B., and Wakili, A. A projection hestenes-stiefel-like method for monotone nonlinear equations with convex constraints. *Thai Journal of Mathematics* (2018).

[22] Nocedal, J., and Wright, S. J. *Numerical optimization*. Springer, 1999.

[23] Papp, Z., and Rapajić, S. Fr type methods for systems of large-scale nonlinear monotone equations. *Applied Mathematics and Computation 269* (2015), 816–823.

[24] Rao, R. V., and Rao, R. V. *Teaching-learning-based optimization algorithm*. Springer, 2016.

[25] Rao, S. S. *Engineering optimization: theory and practice*. John Wiley & Sons, 2019.

[26] Remani, C. Numerical methods for solving systems of nonlinear equations. *Lakehead University Thunder Bay, Ontario, Canada 77* (2013).

[27] Richards, K. L. *Design engineer's reference guide: mathematics, mechanics, and thermodynamics*. CRC Press, 2014.

[28] Sabi'u, J., Althobaiti, A., Althobaiti, S., Sahoo, S. K., and Botmart, T. A scaled polak-ribi \grave {e} re-polyak conjugate gradient algorithm for constrained nonlinear systems and motion control. *AIMS Mathematics 8*, 2 (2023), 4843–4861.

[29] Singh, R. *Optimization Methods and Quadratic Programming*. PhD thesis, 2012.

[30] Solodov, M. V., and Svaiter, B. F. A globally convergent inexact newton method for systems of monotone equations. *Reformulation: Nonsmooth, piecewise smooth, semismooth and smoothing methods* (1999), 355–369.

[31] Tawhid, M. A., and Ibrahim, A. Solving nonlinear systems and unconstrained optimization problems by hybridizing whale optimization algorithm and flower pollination algorithm. *Mathematics and Computers in Simulation 190* (2021), 1342–1369.

[32] Toint, P. Towards an efficient sparsity exploiting newton method for minimization. In *Sparse matrices and their uses*. Academic press, 1981, pp. 57–88.

[33] Wang, X., Ding, X., and Qu, Q. A new filter nonmonotone adaptive trust region method for unconstrained optimization. *Symmetry 12*, 2 (2020), 208.

[34] Wang, X., Li, S., and Kou, X. A self-adaptive three-term conjugate gradient method for monotone nonlinear equations with convex constraints. *Calcolo 53* (2016), 133–145.

[35] Xiao, Y., and Zhu, H. A conjugate gradient method to solve convex constrained monotone equations with applications in compressive sensing. *Journal of Mathematical Analysis and Applications 405*, 1 (2013), 310–319.

[36] Yan, Q.-R., Peng, X.-Z., and Li, D.-H. A globally convergent derivative-free method for solving large-scale nonlinear monotone equations. *Journal of computational and applied mathematics 234*, 3 (2010), 649–657.

[37] Yuan, N. A derivative-free projection method for solving convex constrained monotone equations. *Science Asia 43*, 3 (2017), 195–200.

[38] Zhang, L., and Zhou, W. Spectral gradient projection method for solving nonlinear monotone equations. *Journal of Computational and Applied Mathematics 196*, 2 (2006), 478–484.

[39] Zhang, W. Methods for solving nonlinear systems of equations. *University of Washington* (2013).

[40] Zhou, W.-J., and Li, D.-H. A globally convergent bfgs method for nonlinear monotone equations without any merit functions. *Mathematics of Computation 77*, 264 (2008), 2231–2240.

الخلاصة :

تتناول الرسالة بعض الخوارزميات الجديدة في تقنية البحث الخطي. تعد تقنية البحث الخطي أحد أهم الطرق التكرارية المستخدمة لحل أنظمة المعادلات غير الخطية. الهدف الرئيسي لطريقة البحث الخطي هو تحديد طول الخطوة في اتجاه معين.

لغرض حل أنظمة المعادلات غير الخطية ، تم إدخال ثلاث خوارزميات جديدة لتقنية البحث الخطي ؛ تجمع الخوارزمية الأولى بين تقنية أحادية اللون وخوارزمية بحث خط معدلة ، وتجمع الخوارزمية الثانية بين تقنية البحث الخطي وطريقة التدرج الطيفي المعدل وطريقة الإسقاط ، بينما تجمع الخوارزمية الثالثة والأخيرة بين تقنية البحث الخطي وتقنية التدرج المترافق وتقنية الإسقاط.

الهدف من هذه الخوارزميات تقليل الوقت الذي تستغرقه الحاسبة في التنفيذ وعدد التكرارات وعدد الدوال المحلولة وجعل الطريق أكثر فعالية. تم إثبات التقارب العالمي لهذه الخوارزميات في ظل شروط قياسية. أظهرت الخوارزميات الجديدة أنها واعدة لحل أنظمة المعادلات غير الخطية من خلال النتائج العددية الأولية.

جمهورية العراق

وزارة التعليم العالي والبحث العلمي

جامعة بابل

كلية التربية للعلوم الصرفة

قسم الرياضيات

تطبيق تقنيات البحث الخطي في حل أنظمة المعادلات غير الخطية

رسالة مقدمة الى مجلس كلية التربية للعلوم الصرفة / جامعة بابل كجزء من

متطلبات نيل درجة الماجستير في التربية / الرياضيات

من قبل

دعاء حمزه علاوي مسربت

بأشراف

أ. د. مشتاق عبد الغني الجنابي

١٤٤٥هـ ٢٠٢٣م