A MODIFIED SPECTRAL GRADIENT METHOD FOR SOLVING NON LINEAR SYSTEM

KOAY YEONG LIN

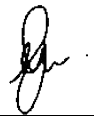
A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Applied Mathematics with Computing

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

September 2020

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : 

Name : Koay Yeong Lin

ID No. : 1701678

Date : 22/8/2020

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"A MODIFIED SPECTRAL GRADIENT METHOD FOR SOLVING NON LINEAR SYSTEM"** was prepared by **KOAY YEONG LIN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Applied Mathematics with Computing at Universiti Tunku Abdul Rahman.

Approved by,

Signature : 

Supervisor : Dr. Sim Hong Seng

Date : 24 August 2020

Signature :

Co-Supervisor :

Date :

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. Firstly, I would like to express my deepest appreciation of gratitude to my research supervisor, Dr. Sim Hong Seng, for his invaluable advice, guidance, recommendations and his immense patience throughout the development of the research. His tremendous knowledge and experience managed to help me to complete this project. He advised me the methodology to carry out the study and present the preliminary results obtained more clearly and professionally. I am grateful to him for his valuable time spent in guiding me, answering my questions, checking and correcting the mistake in my project report. Additionally, I wish to express my gratitude to my parents and friends. Their unconditional support, patience, advice and encouragement gives a lot of help to me throughout my study.

ABSTRACT

The objective of this project is to modify the spectral gradient method in solving the nonlinear systems. The multiple damping spectral gradient method with line search has been proposed for making improvements to the slow convergence issues. It operates separately on the gradient vector norm and the objective function at the same time and can be considered as an alternative for solving large-scale optimization problems. The results show that the spectral gradient method provides the best performance in solving the optimization problems, compared to the steepest descent method and conjugate gradient method, under the backtracking line search with Armijo condition (BTA). The main difference between these methods is the calculation of direction vector, d_k. Besides, there is a relationship between solving a series of nonlinear equations and finding the optimal solutions to the problems. Most of the methods used for solving nonlinear systems are optimization-based methods. Therefore, the spectral gradient method with the BTA line search technique has been modified in order to solve solving the nonlinear systems. The efficiency of the modified spectral gradient method is tested by comparing the number of iterations, the number of function call and the computational time, with the BFGS method, steepest descent method and conjugate gradient method. The step length of these methods is selected by using the modified BTA line search technique. Finally, the modified spectral gradient method shows a better performance compared to the steepest descent method and the conjugate gradient method. The modified method gives more stable results compared to the BFGS method because numerous papers from different researchers have suggested that the BFGS method is not an appropriate method in solving the large-scale problems. Furthermore, the SG method is popular due to the fact that less storage is needed for the calculation. The modified SG method can be used in solving some nonlinear application problems. Thus, the modified spectral gradient method can be considered as an alternative method for solving nonlinear systems. The improvements in the amount of tested problem, line search strategy and search direction are recommended, in order to increase the efficiency of the modified method.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

SG	Spectral Gradient method
SD	Steepest Descent method
CG	Conjugate Gradient method
BFGS	Broyden-Fletcher-Goldfarb-Shanno method
BTA	Backtracking Line Search with Armijo condition

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Mathematical optimization is a mathematical method to search out the solutions of a problem, towards achieving higher performance. It seeks to find the optimal solutions under constraints to the objective function. An optimal solution is a feasible solution that minimizes or maximizes the objective function. In the area of applied math, it focuses on choosing the best element in a given set, by using known mathematical techniques, principles, and methods. Concerning certain criteria, optimization aims to determine the best feasible solution. Furthermore, mathematical optimization involves the analysis of the mathematical structure of optimization problems, the study of the mathematical principles, the development of methods to solve optimization problems, and the application of these approaches on software.

Mathematical optimization is a type of applied mathematics that is applicable in numerous areas, from computing to industrial applications. In a real-world application, optimization is the mathematical process of finding the best decision within a specified set of constraints, such as the highest profit or lowest cost for a given task. However, most real-world and practical issues are dealing more on minimization problems.

To solve an optimization problem, the goal is to find the maxima or minima of a function. It works by consistently selecting input values in an allowed set and evaluating the function value. A comparison of various selections is allowed, for deciding which could be the “best”.

In mathematical optimization, constrained optimization is to examine situations involving constraints. To prevent from moving in certain directions forever, a constraint is a restriction on a variable's value. Usually, constrained optimization problems required mathematical programming. Finding points that meet all the constraints is always a challenging problem for constrained optimization. One of the solutions is to use an unconstrained optimization approach.

In contrast, the meaning of unconstrained optimization is the objective functions either have no boundaries, or the boundaries are soft. There is no limitations on the values of the parameters. Efficient unconstrained optimization algorithms use derivatives and partial derivatives to seek out the local optima. Techniques for unconstrained minimization are now becoming popular in recent years. The methods are useful for solving linear and nonlinear functions.

Solving nonlinear equation systems has always been a complex issue whereby a variety of different methods were carried out. While solving linear equations, or a particular nonlinear equation, the solutions have a well-developed concept of mathematics and computation. Nonlinear systems are resulting in a graph that is not a straight line. Alternatively, the graphs might be cubic functions, parabolas or radical functions. If the system of nonlinear equations does not show good linear or polynomial characteristics, the scenario becomes more challenging. Nonlinear equation systems exist in different practice areas, such as chemistry, engineering, and medicines. The systems also occur in other geometric computations including minimum distance, intersections, and in ordinary or partial differential equations, when solving preliminary or boundary function problems.

1.2 Importance of the Study

Many real-life applications involved the solving of the nonlinear systems. Abu-Arqub, Abo-Hammour and Momani (2014) have published a paper on the application of continuous genetic algorithms (CGA) for nonlinear systems of second-order boundary problems. Based on the genetic algorithm approach, a computational algorithm is launched to deal with problems of second-order boundary value (BVP) in a class of nonlinear systems. They used the CGA to solve the second-order BVP nonlinear system. The model is developed as an optimization problem in this approach. Besides, CGA has been defined as an effective way of solving optimization problems as it has been successfully applied in different areas. For example, to solve the fuzzy differential equations and optimal control problems. Furthermore, it is also applied in the motion planning of robot manipulators, which is extremely nonlinear.

Most of the actual physical systems are essentially nonlinear, therefore, for mathematicians, engineers, and physicists, nonlinear systems are very common (Krstic, 1995). Nonlinear equations are hard to solve by analytical technique and this causes curious phenomena like chaos. It can show an unpredictable behaviour, even if it is a simple dynamic nonlinear system. Chaos seeks its uses in a variety of fields, such as in biological systems, power converters and chemical reactors (Strogatz, 2014).

Nonlinear systems are typically found in natural phenomena with dynamic and sometimes unpredictable behaviours. Throughout various areas of daily life, nonlinear systems present, for instance, medicine, health science, engineering and manufacturing processes. Owing to the variety of nonlinear systems, it is challenging to provide general modelling methods. Nonlinear systems seem to be subjected to uncertainty (Ornelas-Tellez, Rico-Melgoza, Villafuerte, Zavala-Mendoza, 2019).

1.3 Problem Statement

In this project, we propose to develop a modified spectral gradient method for solving nonlinear system. The efficiency of the proposed method will be tested using some tested problems provided in the paper by Fang (2017). The general form of nonlinear equations is given as follows:

$$F(x) = 0, x \in R^n \tag{1.1}$$

For solving the nonlinear equation (1.1), there exist many iterative methods, such as Newton's method and quasi-Newton method. For Newton's method, the procedure is quite simple and straightforward. Besides that, its convergence is rapid for solving many problems. Therefore, Newton's method is commonly used for solving nonlinear equations. However, there are some disadvantages in Newton's method. The efforts needed to determine the solution might be excessive, due to the difficulty of direct evaluating the Jacobian matrix.

1.4 Aim and Objectives

The objective of this project is to modify the spectral gradient method in solving the nonlinear system of equations. Currently, the spectral gradient method is used in solving the optimization problems. Therefore, modification of this method is required, so that we can use the modified method to solve the nonlinear system of equations.

The second objective of this project is to develop Python code for the method to compare the efficiency of the method with the existing method. The graph will be plotted to make a comparison between those methods.

1.5 Scope and Limitation of the Study

The scope of this study is to determine the approximate solution of the nonlinear system. Instead of computing the exact solutions directly, we will approximate the solution of the nonlinear system. The study tends to generate better approximations in each iteration that may tend toward an exact solution.

The limitation of the study is the solution of the nonlinear system is an approximation and not the exact solution. Owing to the difficulty to determine the exact solution, we are determining the approximate solutions nonlinear systems such that the solutions are close to the exact solutions. The approximation is obtained through a series of iteration by setting the initial guess. The problems may fail to converge if the initial guess is not a good guess and may cause lots of iterations. The BTA algorithm is to ensure the function value decrease for next iteration, but not contributed to the convergence.

1.6 Contribution of the Study

In recent years, the role of numerical methods in solving engineering problems has significantly increased. Solving a nonlinear system of equations is essential to engineering problems. Most application problems found in engineering can be simplified to solving nonlinear systems of equations, which is also one of the common problems in mathematics. Numerous approaches have been developed to handle nonlinear systems. The modified spectral gradient method thus provides a good alternative for solving the nonlinear problems.

1.7 Outline of the Report

In the first chapter, a general introduction to mathematical optimization and nonlinear systems will be discussed. Nonlinear systems can be solved in many real-life applications and some examples are provided. This project consists of two objectives, which includes modifying the spectral gradient method in solving the nonlinear equations and develop the python code of the method for the comparison of the efficiency of the proposed method. The scope, limitation and the contribution of the study are also stated in this chapter.

Chapter 2 is a review of the literature. The literature review has involved different types of optimization methods in solving optimization problems and nonlinear problems. This chapter also briefly discussed that there is a close relationship in solving nonlinear problems and optimization problems.

Chapter 3 is the methodology section. This chapter involves the derivation of the SG method and the brief explanation of the methods used in this project for solving the optimization problems. The algorithms for the BTA line search strategy, SG method, SD method, and CG method are also listed in this chapter.

In chapter 4, the SG method has been applied to solve the optimization tested problems. The SG method is compared with the SD method and the CG method, under the same termination conditions and the same line search strategy (BTA). There are 19 tested problems used to compare the performance among these three methods. The preliminary results are presented using the comparison table and the line graph.

In chapter 5, the SG method and BTA line search strategy are incorporated and modified to solve the nonlinear systems. The general algorithms for the modified SG method, BFGS method, SD method, and CG method are listed. There are 31 tested problems used to evaluate the performance of these methods. By using the performance profile of Dolan and

Moré, the behaviours of the modified SG, BFGS, SD, and CG method have been illustrated. Some real-life application problems are solved using the proposed method to illustrate the performance of the modified SG method. Lastly, chapter 6 is the conclusions and recommendation chapter that conclude the entire report and the possible future recommendations.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In solving systems of nonlinear equations, there exist some iterative methods. Most of the methods used are optimization-based methods. There is a close relationship between solving a series of nonlinear equations and finding a local minimum. Seeking a solution is equivalent to minimizing the objective function based on a set of equations with several unknown variables. Such equations satisfied at the current point are considered as constraints at each stage, whereas others are considered as objective functions.

To evaluate the next feasible point to proceed, a quadratic function is minimized in a strategy, based on optimization approaches at each iteration. It stands to reason that many features of the algorithms are identical. For example, Newton's method for local minimization would be the same as Newton's method for nonlinear equations for a smooth function. Yet, there still exist significant differences.

2.2 Literature Review

There exist several optimization methods that can be used to solve nonlinear equations. One of these methods is Newton’s method, also known as the Newton-Raphson method. The Newton-Raphson method was first published by Wallis (1685). Simpson (1740) defined Newton's method as an iterative approach used to solve general nonlinear equations using calculus. Simpson (1740) also introduced the generalization of two equations systems in the same article. He claimed that Newton's method can also be used to overcome optimization problems by setting the slope to zero.

Broyden (1965) mentioned that Newton's method needs several linear equations to be solved at each stage, but the procedure is quite straightforward. Its convergence is rapid for many problems. However, there exist some weaknesses in this method. One of the weakness of Newton's approach is that it often fails to converge even without any changes. Even the convergence requirements of this approach are well understood, but still this approach depends on the initial assumption that the solution is reasonably good and a criterion that is not able to be implemented in practice. Thus, this method is not considered a successful practical procedure.

Furthermore, Broyden (1965) also noted that there exists another disadvantage, which is the difficulty to measure the Jacobian matrix. The efforts needed to determine the matrix might be excessive. Even though the functions are extremely straightforward to obtain their partial derivatives. In a certain situation, the function seems to be too complex, and a numerical approximation to the Jacobian matrix must be gained. In order to evaluate the Jacobian matrix, it is necessary to determine the vector function of the predictor variable. The predictor variable should be at least one set more than the original set of nonlinear equations. This might cause the procedures to become complicated.

In addition, there is a recommendation for the Jacobian matrix. It is either the Jacobian matrix measured once and for all or once every several iterations. This method is easier and faster, rather than the computation of the entire Jacobian as necessarily needed throughout every iteration.

Moreover, the partial derivatives have to be determined and the linear model has to be resolved for each iteration. This will cause the execution of Newton's method expensive. This experience has encouraged the development of quasi-Newton approaches. Mario Martínez (2000) has mentioned that the quasi-Newton method is a stationary Newton method, and also a discrete Newton method. For the discrete Newton method, if the Jacobian matrix is large, it is not comparable with the inexpensive linear algebra models. Yet discrete Newton algorithms are successful in many large sparse issues. In such

situations, the limited difference method allows us to use a small number of functional calculations to measure the estimated Jacobian. The matrix form is not expensive to be factorized.

Quasi-Newton approaches are used for solving unconstrained optimization problems. Some quasi-Newton approaches are popular because many linear algebra iterations are avoided. For solving nonlinear systems, quasi-Newton techniques are not so common research into numerical analysis in the last several years. Mario Martínez (2000) stated that before 1990, there have been many published articles on numerical analysis research of quasi-Newton method for solving nonlinear systems. Occasionally, after the method inclusion into the usual practice of problem solvers in other fields, such as engineering, and manufacturing, the study might be out of practice. While the users are knowledgeable of these benefits and weaknesses, quasi-Newton methods can be used for solving nonlinear large-scale problems.

Additionally, the quasi-Newton method is modified by Fang (2017). The purpose of the modification is to make improvements based on the new quasi-Newton method. The modified quasi-Newton method can be used for solving nonlinear equations. This method is proven that the local superlinear convergence properties exist. According to the article published by Fang (2017), this method is improved by assuming the information from the last three iterates is related. A comparison is made for the modified quasi-Newton method with the other three similar Quasi-Newton methods. The initial point and stopping criteria are set to be tested on some problems. Finally, Fang (2017) proved that the modified method is the best method within these four similar methods.

BFGS method is an iterative approach and it is a part of the quasi-Newton methods. BFGS method can be used for solving unconstrained nonlinear optimization problems. With global and superlinear convergence, some modified BFGS approaches have been presented.

Yuan and Lu (2008) introduced a new backtracking inexact BFGS method for solving symmetric nonlinear equations. The modified BFGS method has a descent property norm, where under appropriate circumstances, the global and superlinear convergence will be guaranteed. Under the BFGS method, Yuan and Lu (2008) have compared two search techniques. The difference between these two search techniques is the presence of Jacobian matrix computation during the selection of the step length. At each iteration, one of the techniques will have to evaluate the Jacobian matrix. The evaluation might increase the difficulty of computation, especially for large-scale problems. Hence, the modified BFGS approach has been proposed by Yuan and Lu (2008) with the backtracking line search techniques that avoid the computing of the Jacobian matrix. Yuan and Lu (2008) have shown that the modified BFGS method with the new backtracking line search is more efficient than the technique that required the computation of the Jacobian matrix. The proposed method also showed global and superlinear convergence.

The trust-region method is one of the optimization methods that can be used to solve the nonlinear equation. From the article that published by Conn, Gloud and Toint (2000), trust-region approaches are iterative because it generates the solution of problem increasingly with better estimates. In this method, we need to develop a model in each iteration. It is used to approximate the objective function in a region and a centre point will then be defined. This region is known as the trust region. Generally, it represents a series of points. Thereafter, the testing point, as well as the value of the objective function at the point, should be determined. The comparison is made between the achieved and predicted reduction. A decision will be made, depending on the sign of the reduction ratio. It is either setting the test point as the next guess point or reducing the trust region.

The trust-region method has evolved over 50 years. It has been embedded well into the area of predicting nonlinear parameters. Levenberg (1944) has published the first paper in this field. In the scope of solving nonlinear least square problems, he suggested by inserting a multiple of identity to the Hessian matrix as a stabilization technique. Throughout this

way, Morrison (1960) has evolved further. He proved that the solutions of a linear system that includes the model's Hessian, are supplemented by a multiple of the identity matrix. This provides the answer to the corresponding sub-problem for suitable options of the damping parameter. The forecasted reduction of the model among this parameter is tedious. Both publications pointed out the fact that the Hessian can be estimated. It will further give the objective function of a quadratic model.

Thereafter, Powell (1970) proposed the trust-region methods for ensuring the convergence of an unconstrained optimization process. Besides, Powell also considered using Broyden quasi-Newton update, to solve nonlinear equations. However, Dennis (1978) is the one who initially mentioned the words "trust-region". Apart from this, such a method is also applicable in many areas, such as applied mathematics, physics, chemistry, computer science, engineering, and medicine, to solve various kinds of problems.

On the other hand, the earliest method that is used for minimizing non-linear functions is the steepest descent method, which was first raised by Cauchy (1847). Apart from very well-conditioned problems, the traditional steepest descent (SD) approach performs poorly. Raydan and Svaiter (2001) noted that the bad behaviour of the steepest descent approach is not related to the choice of search direction. The reason for poor behaviour is related to the optimal selection of step length by Cauchy. The SD approach has been known as extremely poor and inefficient due to the slow convergence speed and oscillatory behaviour, despite the small storage capacity and very low computational expense per execution.

The convergence of the Cauchy traditional steepest descent method has been deeply studied. It has been found that it is related to the Hessian matrix's spectral properties. Asmundis, Serafino, Riccio and Toraldo (2012) recommended a good way to improve the SD method. The purpose of the modification is to force the gradients as the iterations progress into a one-

dimensional subspace. This may avoid the key reason for the SD method's slow convergence, which is the classical zigzag pattern.

The modification of the SD method done by Asmundis, Serafino, Riccio and Toraldo (2012) is called Steepest Descent with Alignment (SDA). It managed to fit the search direction with the eigendirection concerning the smallest eigenvalue. Asmundis, Serafino, Riccio and Toraldo (2012) proved that several useful information regarding the current spectrum of the Hessian matrix is generated monotonically by the sequence of step lengths in the SD approach. The computational findings indicated that by reducing the value of epsilon, the SDA algorithm may be performed efficiently on the problems with the highest ill condition.

Other than that, spectral gradient methods for minimization originated in the Barzilai–Borwein paper. Barzilai and Borwein (1988) have proposed a method, called a two-point step size gradient method. This method is a non-monotone step length, which is associated with the gradient approach, to overcome the Cauchy method's weaknesses. This method is obtained by approximation of the secant equation for the steepest descent method. This method delivers better efficiency and cheaper calculations than the traditional steepest descent method. By making a comparison on the traditional steepest descent method with the new approach, the new approach provides a significant improvement.

Besides, if the objective function of the gradient methods does not have a good condition, it will lead to the inefficiency of the method. To reduce the function value, the gradient methods have a fixed condition in the selection of step length. This will tend to the slow convergence of a stable complex system. Dealing with the problem of inefficiency, Sim, Leong and Chen (2018) have modified the spectral gradient method. This method is proposed for making improvements to slow convergence issues. It operates separately on the gradient vector norm and the objective function at the same time. Furthermore, this method is combined with some line search strategy. The line search is used for reducing the function value, whereas the gradient vector is

damp by an individual adaptive parameter. Under the backtracking and nonmonotone line search, the proposed method is developed. The comparison is made between the proposed method and some well-known CG-based methods since the CG methods have extremely good convergence properties. According to the numerical results and discussion, Sim, Leong and Chen (2018) proved that the proposed spectral gradient method is an alternative for solving large-scale problems.

Hestenes and Stiefel (1952) published the first paper on the conjugate gradient (CG) method for solving a linear system of equations. The CG method is one of the commonly used methods to solve nonlinear problems of large-scale systems. The methodology has a small space requirement as well as good properties for global convergence. The CG approach is an iterative method. When the number of operations needed for the solution is infinite it will have speedy convergence. Each step of this approach provides information on the solution and it gives a better approximation than the preceding. It begins with a very simple technique at any stage, by taking the last approximation achieved as the initial approximation.

Hestenes and Stiefel (1952) have modified this method. The weakness would be the vector for error which is larger in each step than those in the original method. The process of the modified method is also complicated. However, in the modified approach, the approximations are better than those in the original method.

Besides, the conjugate gradient approach has been extended by Fletcher and Reeves (1964) to solve problems of general unconstrained optimization. Currently, conjugate gradient methods are often used as an iterative approach for solving unconstrained large-scale optimization issues, as the matrix memory is not needed. Furthermore, the general form of the three-term conjugate gradient method has been introduced by Narushima, Yabe and Ford (2008). The method usually produces an appropriate direction of descent. Also, depending on the Quasi-Newton multi-step approach, they implemented a new three-term conjugate gradient approach. Narushima, Yabe and Ford

(2008) have showed the good performance and high effectiveness of their method.

2.3 Summary

In recent years, there is a significant increase in the strength and application of optimization techniques. The problems containing millions of unknowns and constraints can be solved by recent methods for specialized issues such as linear programming and unconstrained optimization.

Due to the advantages and disadvantages of different classical optimization methods, many modifications have been made based on different methods. For instance, some of the researchers added useful techniques to continuously improve and thus propose a better method. The purpose is to maintain the strength of certain methods and overcome the weakness of the methods. Modification on those methods aims to improve the overall performance, such as efficiency and computational time, the complexity of computation and the rate of convergence. Additionally, more accurate approximation can be obtained by using the modified methods. In a nutshell, various kinds of optimization methods are proposed to solve the nonlinear system. In this project, the modified spectral gradient method will be proposed in for solving the nonlinear problems.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

In this project, we first make a comparison between the SD method, the CG method, and the SG method. The comparison is made based on their computational time, the number of function calls required, the number of iteration and the norm of the gradient function. The main difference between these three methods is the different in the calculation of direction vector, d_k.

In the development of optimization theory, the SD method has performed a significant role. SD method was first raised by Cauchy (1847). It is a classical method used for solving optimization problems. SD method is a well-known method and the simplest method. It has a slow performance in most real-world problems. Due to its slow convergence rate, the method is not often used in practice. The slow convergence rate may lead to long computational time and high costs. Therefore, some other gradient methods have been proposed.

On the other hand, the CG method is a simple and effective modification of the SD method. Both the SD method and the CG method are the classical method used for solving optimization problems. It has a simple update rule as well, and the directions are based on the gradients. The procedures hence make a good step towards a consistent solution at each step. Commonly, it produces a lower cost in each iteration than the SD method, while both begin at the same point.

The spectral gradient method (SG method) was proposed by Sim, Leong and Chen (2018). SG method is proposed for making improvements to slow convergence issues. It operates separately on the gradient vector norm and the objective function at the same time. It is an alternative method for solving large-scale problems. SG method performs well compared to the SD

method and the CG method. SG method is incorporated with the line search strategy, called BTA. The line search is used for reducing the function value, whereas the gradient vector is damp by an individual adaptive parameter.

BTA is a line search strategy that used to select the best step length. To evaluate the maximum amount to move along a given search direction, a line search method BTA can be used. For the movement along the search direction, it begins with a large approximation of the step length. Depending on the local gradient of the objective function, it will sequentially reduce the step size, known as "backtracking", until a reduction is detected in the objective function.

3.2 SG method

Based on the function

$$\emptyset(B) = tr(B) - \ln(\det(B)) \tag{3.1}$$

We aim to minimize the following problems:

$$\min tr(B_{k+1}) - \ln(\det(B_{k+1})) \tag{3.2}$$

$$s.t.\, s_k^T B_{k+1} s_k = s_k^T y_k \tag{3.3}$$

Let $B_{k+1} = diag(B_{k+1}^{(1)}, \dots, B_{k+1}^{(n)})$ and $s_k = (s_k^{(1)}, \dots, s_k^{(n)})$, the minimization problem (3.2) and (3.3) become

$$\min\left(\sum_{i=1}^{n} B_{k+1}^{(i)}\right) - \ln\left(\prod_{i=1}^{n} B_{k+1}^{(i)}\right) \tag{3.4}$$

$$s.t.\left(\sum_{i=1}^{n} (s_k^{(i)})^2 B_{k+1}^{(i)}\right) - s_k^T y_k = 0 \tag{3.5}$$

Next, we apply the Lagrange multiplier to the minimization problem (3.4) and (3.5), then we obtain

$$L(\alpha,\rho)=\left(\sum_{i=1}^{n}B_{k+1}^{(i)}\right)-\ln\left(\prod_{i=1}^{n}B_{k+1}^{(i)}\right)+\rho[(\sum_{i=1}^{n}(s_k^{(i)})^2B_{k+1}^{(i)})-s_k^Ty_k] \tag{3.6}$$

where

$$\rho\approx\frac{\mathrm{s_k}^T\mathrm{s_k}\ -\mathrm{s_k}^T\mathrm{y_k}}{\sum_{i=1}^{n}(s_k{}^{(i)})^4} \tag{3.7}$$

(3.7) is approximated by using Newton-Raphson method and

$$B_{k+1}{}^{(i)}=\frac{1}{1+\rho(s_k{}^{(i)})^2}\text{ , i = 1, 2, …, n} \tag{3.8}$$

$$s_k=x_{k+1}-x_k \tag{3.9}$$

$$y_k=g_{k+1}-g_k \tag{3.10}$$

Lastly, the updating formula for B_{k+1} is

$$B_{k+1}=\begin{cases}\mathrm{diag}\,(\mathrm{B}_{k+1}{}^{(1)},\dots,\mathrm{B}_{k+1}{}^{(n)}), & \text{if } \mathrm{s_k}^T\mathrm{s_k}\ >\ \mathrm{s_k}^T\mathrm{y_k}\\ \dfrac{s_k{}^Ty_k}{s_k{}^Ts_k}\ I, & \text{otherwise}\end{cases} \tag{3.11}$$

3.3 List of Algorithm

3.3.1 BTA Algorithm

Step 0: Given constants $\delta \in (0,1)$ and γ_1, γ_2 with $0 < \gamma_1 < \gamma_2 < 1$.

Step 1: Set $\mu = 1$

Step 2: Test the relation

$$f(x_k + \mu d_k) \leq f(x_k) + \delta \mu g_k^T d_k \tag{3.12}$$

Step 3: If relation (3.12) does not satisfy, choose a new $\mu \in [\gamma_1 \mu, \gamma_2 \mu]$ and go to Step 2. Else set $\mu_k = \mu$ and $x_{k+1} = x_k + \mu_k d_k$.

3.3.2 SG Algorithm

Step 0: Set $k = 0$. Given initial guessing point x_0, eps $\in (0,1)$ and B_0.

Step 1: Given $g_k = \nabla f(x_k)$. If $\| g_k \| \leq eps$, then stop.

Step 2: Compute

$$x_{k+1} = x_k + \mu_k d_k, for\ k \geq 0 \tag{3.13}$$

where

$$d_k = -{B_k}^{-1} g_k \tag{3.14}$$

and μ_k is obtained through BTA algorithm.

Next, compute B_{k+1} that defined by (3.11), where $B_{k+1}^{(i)}$ is given by (3.8), ρ is given by (3.7), s_k is given by (3.9) and y_k is given by (3.10).

Step 3: Set $k = k + 1$ and go to Step 1.

3.3.3 SD Algorithm

Step 0: Set k = 0. Given initial guessing point x_0 and eps ∈ (0,1).

Step 1: Given $g_k = \nabla f(x_k)$. If $\| g_k \| \leq eps$, then stop.

Step 2: Compute

$$x_{k+1} = x_k + \mu_k d_k, for\ k \geq 0 \tag{3.13}$$

where

$$d_k = -g_k \tag{3.15}$$

and μ_k is obtained through BTA algorithm.

Step 3: Set $k = k + 1$ and go to Step 1.

3.3.4 CG Algorithm

Step 0: Set k = 0. Given initial guessing point x_0 and eps ∈ (0,1).

Step 1: Given $g_k = \nabla f(x_k)$. If $\| g_k \| \leq eps$, then stop. Else, compute $d_0 = -g_0$ and $x_1 = x_0 + \mu_0 d_0$, where μ_0 is obtained through BTA algorithm.

Step 2: Compute

$$x_{k+1} = x_k + \mu_k d_k, for\ k \geq 1 \tag{3.16}$$

where

$$d_k = -g_k + \beta_{k-1} d_{k-1} \tag{3.17}$$

$$\beta_{k-1} = \frac{{g_k}^T g_k}{{g_{k-1}}^T g_{k-1}} \tag{3.18}$$

and μ_k is obtained through BTA algorithm.

Step 3: Set $k = k + 1$ and go to Step 1.

3.4 Summary

The comparison will be made between the SD method, the CG method, and the SG method. The results show that the SG method is an effective method for solving optimization problems. It provides good overall performance, with a shorter computational time, less function call and a smaller number of iterations. This method thus can be used for solving large scale problems. Therefore, the SG method will be modified for solving nonlinear systems.

CHAPTER 4

RESULTS AND DISCUSSION IN OPTIMIZATION

4.1 Introduction

In this project, the comparison is made between the SG method, SD method and CG method under the monotone line search strategy (BTA). The step lengths μ_k are generated by the BTA algorithm with the parameters $\gamma_1 = \gamma_2 = 0.5$ and $\delta = 0.1$. The step length $\mu = 1$ will be used as the initial step length and reduce if the certain relation does not satisfy. The minimum value for step length will be set as 2^{-7}.

Besides, the initial value of B_k is set to be $B_0 = I$ for the SG method. In addition, the termination criterion for SG method, SD method and CG method will be $\| g_k \| \leq 10^{-4}$. Another termination criterion is based on the number of iterations. The maximum number of iterations is set to be 10^4. If the number of iterations reached 10^4, the tested problem considered fail to converge. There are 19 tested problems given by Andrei (2008) that used to test the efficiency of the SG method, SD method and CG method. Furthermore, n = 10, 100, 1000, 2000 and 5000 are used as the dimensions for the tested problems. The codes are written in Python, by using the software Spyder 3.3.2.

4.2 Preliminary Results

The comparison tables of 19 tested problems based on the 3 different methods are constructed. Appendix A shows the details of the data in 19 tables. The symbol '-' represented the function fails to converge. According to the tables in Appendix A, the SG method can be successfully solving the different dimensions in most of the tested problems, while some of the large-scale problems are not able to be solved by using the SD method and CG method. Therefore, the SG method is a good alternative and appropriate method for solving large-scale problems.

The graph will be plotted by using the summation of the number of iterations, the number of function call and the computational time for 19 tested problems respectively. The "-" are replaced by some value in order to plot the graph. The value that used to replace the "-" is the multiplication of the maximum number in each factor and a constant value 60.

To clearly show the difference in numerical effects between the SD method, CG method and SG method, the performance is presented concerning the number of iterations (Figure 4.1), the number of function calls (Figure 4.2) and the computational time (Figure 4.3) respectively. Due to the large difference between the y-value, the natural logarithm scale is used as the scale of the y-axis. The tested problems are sorted in ascending order of y-value, instead of the order of tested problems. If the graph is plotted according to the order of tested problems, the performance of these 3 methods is not able shown clearly, since the line in the graph might be overlapping. Therefore, the tested problems are sorted according to the ascending order of y-value, to make the graph clearer and comparison can be made between these three methods.

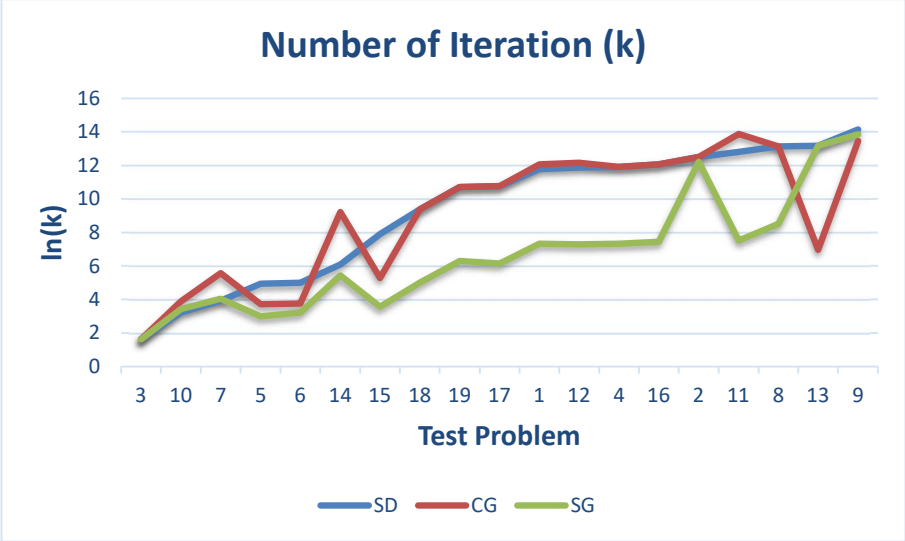


Figure 4.1: Number of Iteration for SD, CG and SG method.

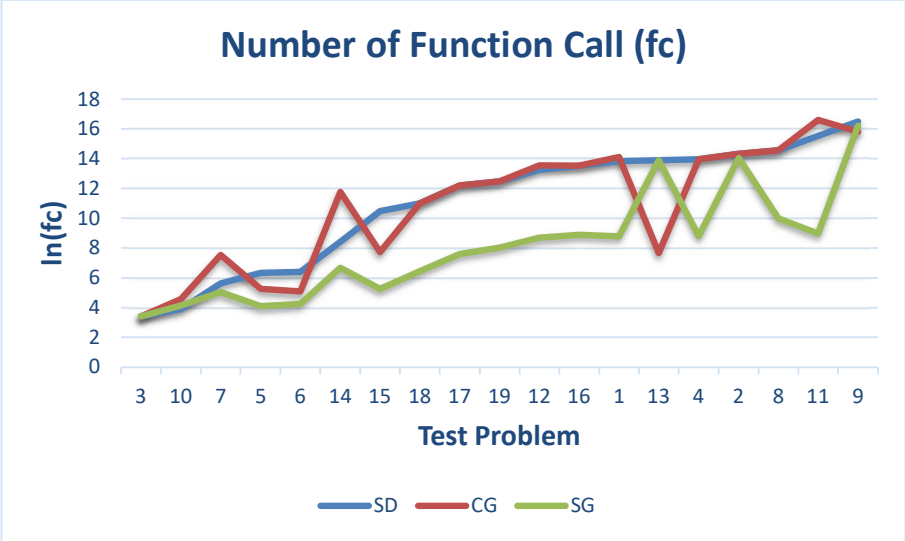


Figure 4.2: Number of Function Call for SD, CG and SG method.

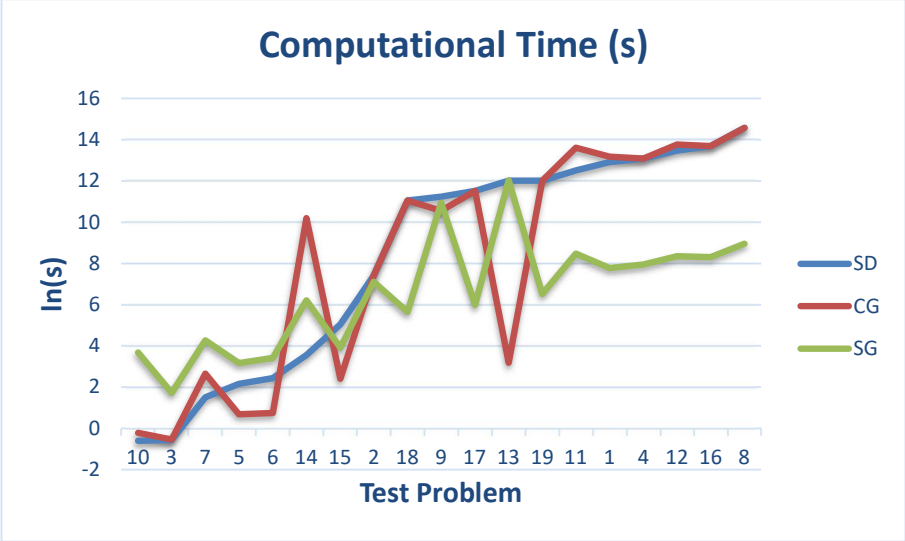


Figure 4.3: Computational Time for SD, CG and SG method.

Figure 4.1 illustrates the total number of iterations for 19 tested problems in these 3 methods. It is clearly to see that the SG method performs better than the other two methods for most tested problems with the smallest number of iterations. Comparing with the SG method, both of the CG and SD methods required a much greater number of iterations for solving the tested problems. Based on Figure 4.1, we can conclude that the SG method gives the best performance in the least number of iterations required to solve the tested problems. However, CG and SD method show weaker performance and both of them required almost the same number of iterations.

Figure 4.2 shows that the SG method required lesser number of functions calls for solving the tested problems, compare to the other two methods. Clearly, the SG method exhibits the best performance in terms of the number of function calls since it can solve most of the tested problems with the smallest number of function calls.

Figure 4.3 demonstrates that the SG method can solve about 50% of the tested problems with the shortest computational time. For the SD method and the CG method, both of them solve approximately 25% of the test problems with the shortest computational time.

Even though Figure 4.3 shows that the computational time of the SG method is not so significant, but the average performance of the SG method is still computable. Based on the number of iterations and the number of function call required for the SG method, it uses much lesser number of iterations and function calls than the CG method and SD method. It implies that the SG method possesses the best overall performance among these three methods.

4.3 Summary

In the real industry application, it is important in reducing the cost, since it helps to increase the profit or return of an industry. If more function calls are required, a higher cost budget and longer computational time will be required. Instead of using an approach requires longer computational time and more function calls, the procedures should be improved applying an alternative with shorter time and lower cost to solve the problems. Therefore, we can conclude that the SG method performs the highest efficiency and due to this reason, we propose to modify this method for solving nonlinear systems.

CHAPTER 5

RESULTS AND DISCUSSION IN NONLINEAR SYSTEM

5.1 Introduction

In chapter 4, we have shown that SG method is an efficient method for solving optimization problems. The comparison is made between SG, SD, and CG method, in terms of the number of iterations, the number of function call, and the computational time. The step length of these methods is selected by using the BTA line search strategy. For solving optimization problems, the testing condition used in the BTA line search has stated in (3.12).

Yuan and Lu (2008) have proposed the BFGS combined with a new backtracking line search approach. This method can be used for solving symmetric nonlinear systems. Yuan and Lu (2008) proved that the BFGS method holds global and superlinear convergence. The BFGS method has the property of descending the search direction for the norm function. The BFGS method shows an efficient performance and the new backtracking line search approach used in the BFGS method has stated in (5.1). Without evaluating the Jacobian matrix, the difficulty and complexity of the computations have reduced.

$$\| F(x_k + \mu_k d_k) \|^2 \leq \| F(x_k) \|^2 + \delta \alpha_k^2 F_k^T d_k \tag{5.1}$$

The SG method has been modified for solving the nonlinear system. The BTA line search technique is modified to have the inequality in (5.1). The performance of the modified SG method has been computed by comparing the number of iterations, the number of function call, and the computational time, with the BFGS method, SD method, and CG method.

5.2 General Algorithm for Modified SG, BFGS, SD and CG Method.

Step 0: Set $k = 0$. Given initial guessing point x_0, eps ∈ (0,1) and B_0.

Step 1: If $\| F_k \| = 0$, then stop.

Step 2: Compute

$$x_{k+1} = x_k + \mu_k d_k, for\ k \geq 0 \tag{3.13}$$

where d_k is obtained from Table 5.1 and μ_k is obtained from Table 5.2.

Step 3: (Modified SG and BFGS) Compute B_{k+1} that defined in Table 5.3, where s_k is given by (3.9) and

$$y_k = F_{k+1} - F_k \tag{5.2}$$

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

Table 5.1: Computation of d_k.

Modified SG BFGS	$d_k = -{B_k}^{-1} F_k$ (5.3)
SD	$d_k = -F_k$ (5.4)
CG	$d_k = -F_k + \beta_{k-1} d_{k-1}$ (5.5) where $\beta_{k-1} = \frac{{F_k}^T F_k}{{F_{k-1}}^T F_{k-1}}$ (5.6) and $\beta_{-1} d_{-1} = 0$.

Table 5.2: Computation of μ_k.

Modified SG	μ_k is obtained through BTA algorithm.
SD	
CG	
BFGS	If $\| F(x_k + d_k) \| \leq eps \| F(x_k) \|$, set $\mu_k = 1$. Otherwise, μ_k is obtained through BTA algorithm.

Table 5.3: Computation of B_{k+1}.

Modified SG	B_{k+1} is defined by (3.11), where $B_{k+1}^{(i)}$ is given by (3.8), ρ is given by (3.7)
BFGS	$B_{k+1} = \begin{cases} B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}, & \text{if } s_k{}^T y_k > 0 \\ B_k, & \text{otherwise} \end{cases}$ (5.7)

5.3 Results and Discussions

The comparison is made between the modified SG method, BFGS method, SD method, and the CG method under the modified BTA line search strategy. In the BTA step length selection, the value of parameters used, the initial step length, and the lower bound of the step length are the same as the values set for solving the optimization problems.

In addition, since the modified SG method and BFGS method required the computation of matrix B_k, the matrix B_0 is initialized to an identity matrix with dimension n. There exist two termination criteria for these methods, which include the norm of the nonlinear functions and the number of iterations. The first termination criterion will be $\| F_k \| \leq 10^{-4}$ and the maximum number of iterations is set to be 10^4. If the number of iterations exceeds 10^4, the tested problem will be considered as fail to converge.

There is a total of 31 tested problems have been used to test the performance of the modified SG method, the BFGS method, the SD method, and the CG method. Appendix D shows the 12 tested problems given by Fang (2017). The gradient functions of the optimization tested problems provided by Andrei (2008), as shown in Appendix B are used as the remaining 19 tested problems, since there is a similarity between the gradient function of the optimization problems and the nonlinear problems. The dimensions of the tested problems will be set as n = 10, 100, 200 and 500, if the dimensions are not provided in the tested problems.

The results of the tested problems are listed in Appendix C. The tables in Appendix C include the results of the number of iterations, the number of function call and the computational time (in seconds) for the four methods. The symbol '-' represents that the method failed to converge.

By using the performance profile of Dolan and Moré, the behaviour of the modified SG, BFGS, SD and CG method can be illustrated. Clearly, Figure 5.1, 5.2 and 5.3 are the performance profiling graphs for the four methods, based on the number of iterations, number of function call and the computational time.

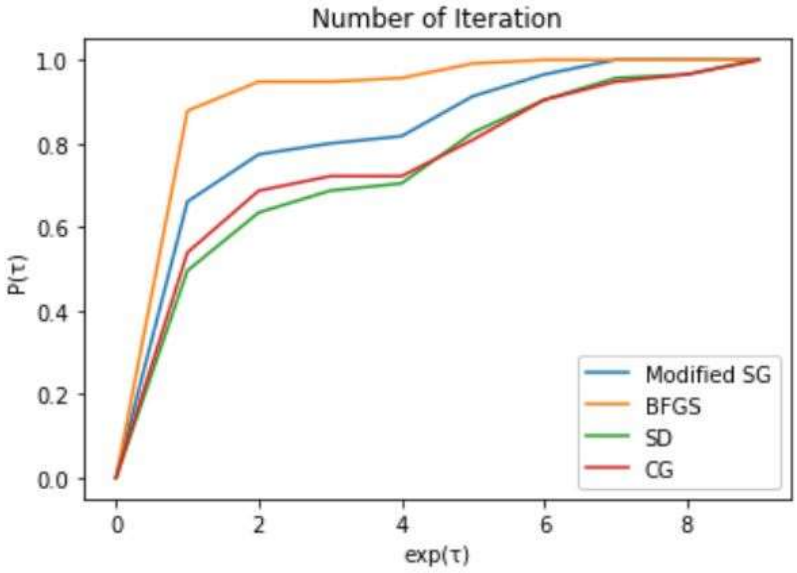


Figure 5.1: Number of Iteration for Modified SG, BFGS, SD and CG method.

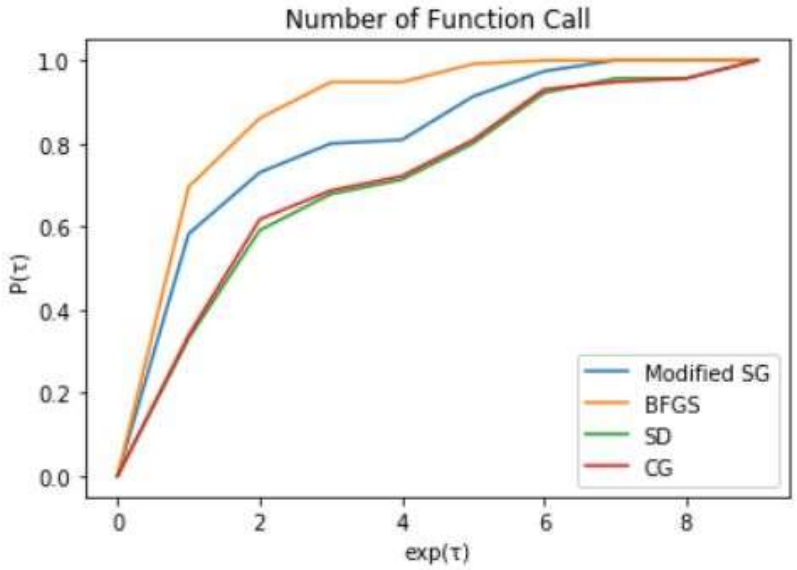


Figure 5.2: Number of Function Call for Modified SG, BFGS, SD and CG method.

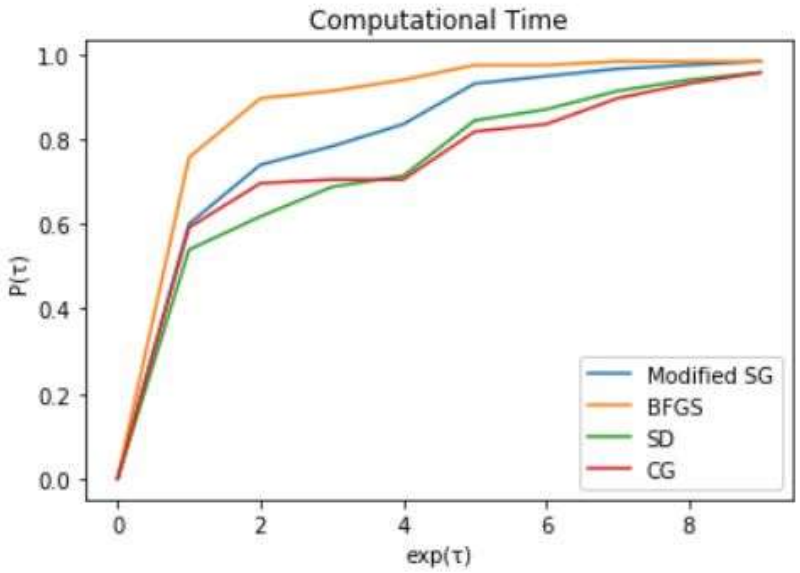


Figure 5.3: Computational Time for Modified SG, BFGS, SD and CG method.

Figures above show that the BFGS method performs the best among these methods in terms of the number of iterations, number of function call and computational time. The modified SG method indicates a better performance compared to SD and CG methods. On the other hand, the SD and CG methods exhibit a similar pattern, which show a poorer performance than the modified SG and BFGS method.

For the BFGS method, some of the tested problems require an extremely high number of iterations, number of function calls and computational time, compared to the modified SG method. Therefore, we conclude that the modified SG method gives a more stable result than the BFGS method. Although the modified SG method is not showing the best performance among these methods, it is still considered as an alternative to solve the nonlinear tested problems.

5.4 Non-Linear System in Real Life Application

Systems of nonlinear equations take place in many areas of practical importance such as engineering. In order to evaluate the performance of the modified SG method, 6 systems of non-linear equations are considered. The application problems are provided by Chen, Liu, Zhou and Peng (2017), Grosan and Abraham (2008), Buzzi-Ferraris and Manenti (2014) and Turgut and Coban (2014). These problems are applied in both the engineering and science fields.

5.4.1 Application 1: Kinematic Application

$$\begin{cases} x_i^2 + x_{i+1}^2 - 1 = 0 \\ a_{1i}x_1x_3 + a_{2i}x_1x_4 + a_{3i}x_2x_3 + a_{4i}x_2x_4 + \\ a_{5i}x_2x_7 + a_{6i}x_5x_8 + a_{7i}x_6x_7 + a_{8i}x_6x_8 + \\ a_{9i}x_1 + a_{10i}x_2 + a_{11i}x_3 + a_{12i}x_4 + a_{13i}x_5 + \\ a_{14i}x_6 + a_{15i}x_7 + a_{16i}x_8 + a_{17i} = 0 \\ 1 \le i \le 4 \end{cases} \tag{5.8}$$

The initial guessing point used is

$x_0 = [-0.06, 0.78, -0.05, 0.38, -0.56, -0.70, 0.40, 0.09]^T$

and the coefficients $a_{ki}, 1 \leq k \leq 17, 1 \leq i \leq 4$, are given in Table 5.4.

Table 5.4 Coefficients a_{ki} for the Kinematic Application

- 0.249150680	+ 0.125016350	- 0.635550070,	+ 1.48947730
+ 1.609135400	- 0.686607360	- 0.115719920	+ 0.23062341
+ 0.279423430	- 0.119228120	- 0.666404480	+ 1.32810730
+ 1.434801600	- 0.719940470	+ 0.110362110	- 0.25864503
+ 0.000000000	- 0.432419270	+ 0.290702030	+ 1.16517200
+ 0.400263840	+ 0.000000000	+ 1.258776700	- 0.26908494
- 0.800527680	+ 0.000000000	- 0.629388360	+ 0.53816987
+ 0.000000000	- 0.864838550	+ 0.581404060	+ 0.58258598
+ 0.074052388	- 0.037157270	+ 0.195946620	- 0.20816985
- 0.083050031	+ 0.035436896	- 1.228034200	+ 2.68683200
- 0.386159610	+ 0.085383482	+ 0.000000000	- 0.69910317
- 0.755266030	+ 0.000000000	- 0.079034221	+ 0.35744413
+ 0.504201680	- 0.039251967	+ 0.026387877	+ 1.24991170
- 1.091628700	+ 0.000000000	- 0.057131430	+ 1.46773600
+ 0.000000000	- 0.432419270	- 1.162808100	+ 1.16517200
+ 0.049207290	+ 0.000000000	+ 1.258776700	+ 1.07633970
+ 0.049207290	+ 0.013873010	+ 2.162575000	- 0.69686809

5.4.2 Application 2: Interval Arithmetic Benchmark Application

$$\begin{cases} x_1 - 0.25428722 - 0.18324757x_4x_3x_9 = 0 \\ x_2 - 0.37842197 - 0.16275449x_1x_{10}x_6 = 0 \\ x_3 - 0.27162577 - 0.16955071x_1x_2x_{10} = 0 \\ x_4 - 0.19807914 - 0.15585316x_7x_1x_6 = 0 \\ x_5 - 0.44166728 - 0.19950920x_7x_6x_3 = 0 \\ x_6 - 0.14654113 - 0.18922793x_8x_5x_{10} = 0 \\ x_7 - 0.42937161 - 0.21180486x_2x_5x_8 = 0 \\ x_8 - 0.07056438 - 0.17081208x_1x_7x_6 = 0 \\ x_9 - 0.34504906 - 0.19612740x_{10}x_6x_8 = 0 \\ x_{10} - 0.42651102 - 0.21466544x_4x_8x_1 = 0 \end{cases} \quad (5.9)$$

The initial guessing point used is $x_0 = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]^T$

5.4.3 Application 3: Chemical Equilibrium

$$
\begin{aligned}
x_1 + x_4 - 3 &= 0 \\
2x_1 + x_2 + x_4 + x_7 + x_8 + x_9 + 2x_{10} - R &= 0 \\
2x_2 + 2x_5 + x_6 + x_7 - 8 &= 0 \\
2x_3 + x_9 - 4R &= 0 \\
x_1 x_5 - 0.193 x_2 x_4 &= 0 \\
x_6\sqrt{x_1} - 0.002597\sqrt{x_2 x_4 \cdot TOT} &= 0 \\
x_7\sqrt{x_4} - 0.003448\sqrt{x_1 x_2 \cdot TOT} &= 0 \\
x_4 x_8 - 1.799 \times 10^{-5} x_1 \cdot TOT &= 0 \\
x_4 x_9 - 2.155 \times 10^{-4} x_1\sqrt{x_3 \cdot TOT} &= 0 \\
x_{10} x_4^2 - 3.846 \times 10^{-5} x_4^2 \cdot TOT &= 0
\end{aligned}
\tag{5.10}
$$

where R is 4.056734 and TOT is defined as $TOT = \sum_{i=1}^{10} x_i$.

The initial guessing point used is

$$x_0 = \begin{matrix}[0.15884, 0.89358, 8.11340, 2.84116, 3.08473,^T \\ 0.04039, 0.00300, 0.00002, 0.00013, 0.00058]\end{matrix}$$

5.4.4 Application 4: Neurophysiology application

$$
\begin{aligned}
f_1 &= x_1^2 + x_3^2 - 1 = 0 \\
f_2 &= x_2^2 + x_4^2 - 1 = 0 \\
f_3 &= x_5 x_3^3 + x_6 x_4^3 = 0 \\
f_4 &= x_5 x_1^3 + x_6 x_2^3 = 0 \\
f_5 &= x_5 x_1 x_3^2 + x_6 x_4^2 x_2 = 0 \\
f_6 &= x_5 x_1^2 x_3 + x_6 x_2^2 x_4 = 0
\end{aligned}
\tag{5.11}
$$

The initial guessing point used is

$x_0 = [0.446, -0.446, 0.895, -0.895, 0.367, 0.367]^T$

5.4.5 Application 5: Combustion application

$$
\begin{aligned}
f_1 &= x_2 + 2x_6 + x_9 + 2x_{10} - 10^{-5} = 0 \\
f_2 &= x_3 + x_8 - 3 \cdot 10^{-5} = 0 \\
f_3 &= x_1 + x_3 + 2x_5 + 2x_8 + x_9 + x_{10} - 5 \cdot 10^{-5} = 0 \\
f_4 &= x_4 + 2x_7 - 10^{-5} = 0 \\
f_5 &= 0.5140437 \cdot 10^{-7} x_5 - x_1^2 = 0 \\
f_6 &= 0.100632 \cdot 10^{-6} x_6 - 2x_2^2 = 0 \\
f_7 &= 0.7816278 \cdot 10^{-1}\ x_7 - x_4^2 = 0 \\
f_8 &= 0.1496236 \cdot 10^{-6} x_8 - x_1 x_3 = 0 \\
f_9 &= 0.6194411 \cdot 10^{-7} x_9 - x_1 x_2 = 0 \\
f_{10} &= 0.2089296 \cdot 10^{-14} x_{10} - x_1 x_2^2 = 0
\end{aligned}
\tag{5.12}
$$

The initial guessing point used is

$$
x_0 = \begin{matrix}[-5.9286 \cdot 10^{-8}, -6.9428\ \cdot 10^{-5}, -0.2980, -8.8526 \cdot 10^{-5}, -0.4127, \\ -0.0547, 4.9253\ \cdot 10^{-5}, 0.2981, 0.9453, -0.4179]^T\end{matrix}
$$

5.4.6 Application 6: Experimental Test

$$
\begin{aligned}
f_1(x_1, x_2) &= \cos(2x_1) - \cos(2x_2) - 0.4 = 0 \\
f_2(x_1, x_2) &= 2(x_2 - x_1) + \sin(2x_2) - \sin(2x_1) - 1.2 = 0
\end{aligned}
\tag{5.13}
$$

The initial guessing point used is $x_0 = [0.15, 0.49]^T$

5.4.7 Numerical Results for Application Problems

Table 5.5: Numerical Results for Application Problems

Number of Iteration					
Problem	Dim	Modified SG	BFGS	SD	CG
1	8	169	-	152	-
2	10	4	4	4	3
3	10	2	2	2	2
4	6	2	2	2	2
5	10	2	2	2	2
6	2	8171	-	881	-
Number of Function Call					
Problem	Dim	Modified SG	BFGS	SD	CG
1	8	3450	-	2644	-
2	10	18	26	18	14
3	10	19	41	25	37
4	6	52	56	52	52
5	10	10	14	10	13
6	2	193489	-	21241	-
Computational Time					
Problem	Dim	Modified SG	BFGS	SD	CG
1	8	1.8066	-	1.2886	-
2	10	0.0100	0.0139	0.0096	0.0070
3	10	0.0091	0.0107	0.0087	0.0069
4	6	0.0070	0.0091	0.0082	0.0072
5	10	0.0060	0.0110	0.0060	0.0060
6	2	16.3163	-	1.5309	-

According to Table 5.5, it can be concluded that the modified SG method serves as an another option to solve different systems of nonlinear equations in real-life applications. Generally, the modified SG method indicates better results than the other three existing methods. In the real industry application, computational time is an essential component to be considered. Less number of iterations and function calls might lead to a shorter computational time. A short computational time is allowed in reducing the cost and increasing the profit of an industry.

5.5 Summary

The numerical experiments show that the modified SG method can be an alternative in solving nonlinear system of equations not only in research tested problems but also in real-life applications. Therefore, it is worthwhile to modify the SG method. In the next chapter, some possible recommendations will be suggested to further improve the performance of the method.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

The SG method with BTA line search strategy has been proposed for solving the optimization problems. The performance of the SG method is tested against the SD method and CG method, through 19 optimization tested problems. SG method has shown the best overall performance among these three methods. It required the least number of iterations, number of function call and computational time. The SG method can be tested as an alternative for solving large scale problems.

The SG method and BTA line search have modified in order to solve the nonlinear systems. The efficiency of the modified SG method has been tested using 31 nonlinear tested problems and then compared with the BFGS method, SD method and CG method. From the graph constructed regarding the number of iterations, the number of function call and the computational time, BFGS method appears to have the best overall performance, followed by the modified SG method.

One of the reasons that the BFGS method gives better performance is it uses the full rank matrix for B_k, while the modified SG method uses the diagonal matrix in the updating formula of B_k. Numerous papers from different researchers have suggested that the BFGS method is not an appropriate method in solving the large-scale problems due to the fact that high storage is required. Thus, we can conclude that the modified SG method seems to be a better alternative when dealing with large-scale problems and in real-life applications.

The problem encountered in this project is some of the tested problems fail to converge to the approximate solution. Besides, the initial guessing points for the real-life application problems have not been provided in the papers published. Therefore, the values that close to the solution for the problems have been chosen as the initial guessing points. The selection of initial guessing points might lead to the failure of convergence for the method.

6.2 Recommendations for future work

It is recommended that the amount of the tested problems can be increased since there are only 31 nonlinear tested problems used to test the efficiency of the modified method. A higher amount of tested problems should be collected from other different papers, to generate more reliable, accurate and precise results.

Apart from this, in the selection of the step length, the line search strategy used in this project is the backtracking line search with Armijo condition. Instead of using the Armijo condition, there exist many different types of conditions that might produce a better step length selection result. For recommendation, the other techniques in selecting an appropriate step length are the Goldstein condition and the Wolfe condition, which is the combination of Armijo condition and the curvature condition. Armijo condition requires a reduction in the step length at the next iteration, while the Wolfe condition ensures an adequate reduction on both the step length and the slope. Goldstein condition is identical to the Wolfe condition, which guarantees that the sufficient reduction of the step length and prohibits the extremely small value of step length.

Other than applying the monotone line search strategy, the nonmonotone line search strategy has also suggested. Nonmonotone line search with some combined conditions provides the possibility to achieve better performance. Besides, the search direction is important the convergence of the method. Therefore, different search directions can be carried out to

obtain a better approximation. Some modified search directions proposed that give an improvement on the numerical performance are suggested, such as the combination of some classical search directions.

REFERENCES

Abu-Arqub, O., Abo-Hammour, Z. and Momani, S., 2014. Application of Continuous Genetic Algorithm for Nonlinear System of Second-Order Boundary Value Problems. *Applied Mathematics & Information Sciences*, [e-journal] 8(1), pp.235-248. http://dx.doi.org/10.12785/amis/080129.

Andrei, N., 2008. An Unconstrained Optimization Test Functions Collection. *Advanced Modeling and Optimization*, 10(1), pp.147-161.

Asmundis, R.D., Serafino, D.D., Riccio, F. and Toraldo, G., 2012. On spectral properties of steepest descent methods. *IMA Journal of Numerical Analysis*, [e-journal] 33, pp.1416-1435. doi:10.1093/imanum/drs056.

Barzilai, J. and Borwein, J. M., 1988. Two-Point Step Size Gradient Methods. *Journal of Numerical Analysis*, [e-journal] 8(1), pp.141-148. https://doi.org/10.1093/imanum/8.1.141.

Broyden, C. G., 1965. A Class of Methods for Solving Nonlinear Simultaneous Equations. *Mathematics of Computation*, [e-journal] 19(92), pp.557-593. https://doi.org/10.2307/2003941.

Buzzi-Ferraris, G. and Manenti, F., 2014. *Nonlinear System and Optimization for the Chemical Engineer*. [e-book] Nonlinear Systems. Germany: Wiley-VCH Verlag GmbH & Co. KGaA. Available through: Universiti Tunku Abdul Rahman Library website http://library.utar.edu.my/ [Accessed 24 July 2020], pp. 235-311.

Chen, X., Liu, Y. M., Zhou, W. and Peng, X. Y., 2017. *Simplex-Fruit Fly Optimization Algorithm for Solving Systems of Non-linear Equations*. In: Fuzzy Systems and Knowledge Discovery, 2017 13th International Conference on Natural Computation. China, 2017. New York: Institute of Electrical and Electronic Engineers.

Conn, A. R., Gloud, N. I. M. and Toint, P. L., 2000. *TRUST-REGION METHODS*. [e-book] Philadelphia: Society for Industrial and Applied Mathematics and Mathematical Programming Society. Available at: Google Books <https://books.google.com.my/books?hl=en&lr=&id=aJ-Hq71nVHMC&oi=fnd&pg=PR2&dq=info:Wq0gKULdvE8J:scholar.google.com/&ots=LylldAllP2&sig=I8pr6UvxEdYQGHWzu84eR4moyZk&redir_esc=y#v=onepage&q&f=false> [Accessed 10 February 2020]. pp.6-10.

Fang, X. W., 2017. A modified quasi-Newton method for nonlinear equations. *Journal of Computational and Applied Mathematics*, 328(2018), pp.44-58.

Grosan, C. and Abraham, A., 2008. A New Approach for Solving Nonlinear Equations Systems. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, [e-journal] 38(3), pp. 698-714. https://doi.org/10.1109/TSMCA.2008.918599.

Hestenes, M. R. and Stiefel, E., 1952. Methods of Conjugate Gradients for Solving Linear Systems. *Journal of Research of the National Bureau of Standards*, [e-journal] 49(6), pp. 409-411. doi:10.6028/jres.049.044.

MarioMartínez, J., 2000. Practical quasi-Newton methods for solving nonlinear system. *Journal of Computational and Applied Mathematics*, [e-journal] 124(1-2), pp.97-121. http://doi.org/10.1016/S0377-0427(00)00434-9.

Narushima, Y., Yabe, H. and Ford, J. A., 2011. A three-term conjugate gradient method with sufficient descent property for unconstrained optimization. *SIAM Journal on Optimization*, [e-journal] 21(1), pp.212-230. https://doi.org/10.1137/080743573.

Ornelas-Tellez, F., Rico-Melgoza, J. J., Villafuerte, A. E. and Zavala-Mendoza, F. J.,2019. *Artificial Neural Networks for Engineering Applications*. [e-book] A Methodology for Modeling and Control Design of Dynamical Systems. United States: Mara Conner. Available at: Google Books < https://books.google.com.my/books?hl=en&lr=&id=OtuGDwAAQBAJ&oi=fnd&pg=PP1&dq=artificial+neural+networks+for+engineering+applications&ots=WqSRFbPgQj&sig=KdcAjqzYH47HAgrH4fjofjKF91s#v=onepage&q=artificial%20neural%20networks%20for%20engineering%20applications&f=false> [Accessed 13 March 2020]. pp.21-38.

Raydan, M. and Svaiter, B. F., 2001. Relaxed Steepest Descent and Cauchy-Barzilai-Borwein Method. *Computational Optimization and Applications*, [e-journal], 21, pp.155-167. https://doi.org/10.1023/A:1013708715892.

Shukla, M. K., Sharma, B. B. and Azar, A. T.,2018. *Mathematical Techniques of Fractional Order Systems*. [e-book] Control and Synchronization of a Fractional Order Hyperchaotic System via Backstepping and Active Backstepping Approach. United States: Mara Conner. Available at: Google Books
<https://books.google.com.my/books?hl=en&lr=&id=1aRBDwAAQBAJ&oi=fnd&pg=PP1&dq=Mathematical+Techniques+of+Fractional+Order+Systems&ots=khkGaM_uOe&sig=9njEMtOff9_6KvkgGSNsgLKUemg&redir_esc=y#v=onepage&q=Mathematical%20Techniques%20of%20Fractional%20Order%20Systems&f=false> [Accessed 13 March 2020]. pp.559-595.

Sim, H. S., Leong, W. J. and Chen, C. Y., 2018. Gradient method with multiple damping for large-scale unconstrained optimization. *Optimization Letters*, 13(3), pp.617-632.

Turgut, O. E., Turgut, M. S. and Coban, M. T., 2014. Chaotic quantum behaved particle swarm optimization algorithm for solving nonlinear system

of equations. *Computers & Mathematics with Applications*, [e-journal] 68(4), pp. 508-530. https://doi.org/10.1016/j.camwa.2014.06.013.

Yuan, G. L. and Lu, X. W., 2008. A new backtracking inexact BFGS method for symmetric nonlinear equations. *Computers & Mathematics with Applications*, [e-journal] 55(1), pp. 116-129. https://doi.org/10.1016/j.camwa.2006.12.081.

APPENDICES

APPENDIX A: Tables of Optimization Test Problems

Table A-1: Test Problem 1 - Quadratic QF1.

TP 1	Steepest Descent Method				Conjugate Gradient Method				Spectral Gradient Method			
Dimension (n)	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**
10	35	234	7.558E-05	0.060837	27	190	8.339E-05	0.048257	20	58	5.8549E-05	0.031242
100	427	5630	9.212E-05	3.956597	-	-	-	-	106	408	9.8687E-05	0.577956
1000	-	-	-	-	-	-	-	-	325	1392	9.1098E-05	34.881029
2000	-	-	-	-	-	-	-	-	348	1546	7.8044E-05	115.203001
5000	-	-	-	-	-	-	-	-	724	3134	9.6348E-05	2222.221291

Table A-2: Test Problem 2 - POWER (CUTE).

TP 2	Steepest Descent Method				Conjugate Gradient Method				Spectral Gradient Method			
Dimension (n)	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**
10	456	6924	9.386E-05	1.405910	185	2860	8.822E-05	0.553721	105	406	8.2678E-05	0.237365
100	-	-	-	-	-	-	-	-	1118	4856	8.1892E-05	6.772840
1000	-	-	-	-	-	-	-	-	-	-	-	-
2000	-	-	-	-	-	-	-	-	-	-	-	-
5000	-	-	-	-	-	-	-	-	-	-	-	-

Table A-3: Test Problem 3 - QUARTC (CUTE).

TP 3	Steepest Descent Method				Conjugate Gradient Method				Spectral Gradient Method			
Dimension (n)	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**
10	1	6	0	0	1	6	0	0.003990	1	6	0	0.005983
100	1	6	0	0.015619	1	6	0	0.019811	1	6	0	0.184210
1000	1	6	0	0.109351	1	6	0	0.107208	1	6	0	0.518560
2000	1	6	0	0.109351	1	6	0	0.113483	1	6	0	1.672853
5000	1	6	0	0.343667	1	6	0	0.332819	1	6	0	3.266842

Table A-4: Test Problem 4 - Almost Perturbed Quadratic.

TP 4	Steepest Descent Method				Conjugate Gradient Method				Spectral Gradient Method			
Dimension (n)	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**
10	35	304	7.873E-05	0.078071	33	304	8.99E-05	0.070204	21	60	4.9286E-05	0.047519
100	425	6454	8.761E-05	4.170171	174	2776	8.77E-05	1.973649	82	326	8.1317E-05	0.437367
1000	-	-	-	-	-	-	-	-	228	992	9.4705E-05	26.641538
2000	-	-	-	-	-	-	-	-	379	1698	7.3804E-05	135.949297
5000	-	-	-	-	-	-	-	-	826	3592	9.9989E-05	2682.422505

Table A-5: Test Problem 5 - Diagonal 7.

TP 5	**Steepest Descent Method**				**Conjugate Gradient Method**				**Spectral Gradient Method**			
Dimension (n)	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**
10	23	98	7.405E-05	0.030916	7	34	1.082E-05	0.011943	4	12	3.6254E-06	0.015589
100	26	104	7.309E-05	0.144861	7	34	3.42E-05	0.040890	4	12	1.1464E-05	0.062484
1000	29	116	7.215E-05	1.583430	8	40	9.506E-06	0.354394	4	12	3.6254E-05	1.031004
2000	30	120	6.921E-05	1.579953	8	40	1.344E-05	0.383751	4	12	5.127E-05	2.151960
5000	31	124	7.423E-05	5.282017	11	46	4.95E-05	1.191037	4	12	8.1065E-05	20.781065

Table A-6: Test Problem 6 - Diagonal 8.

TP 6	Steepest Descent Method				Conjugate Gradient Method				Spectral Gradient Method			
Dimension (n)	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**
10	25	102	7.605E-05	0.031239	8	30	9.169E-06	0.012968	5	14	1.4994E-07	0.015612
100	28	114	8.009E-05	0.174508	8	30	2.899E-05	0.054827	5	14	4.7415E-07	0.046865
1000	31	126	8.435E-05	1.858927	8	30	9.169E-05	0.369179	5	14	1.4994E-06	1.396047
2000	32	130	8.268E-05	2.363073	9	34	9.137E-05	0.423841	5	14	2.1205E-06	2.988238
5000	33	134	9.061E-05	7.057156	10	36	2.317E-06	1.246357	5	14	3.3528E-06	26.409306

Table A-7: Test Problem 7 - Generalized Quartic.

TP 7	Steepest Descent Method				Conjugate Gradient Method				Spectral Gradient Method			
Dimension (n)	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**
10	8	46	2.09E-06	0.015619	21	128	9.748E-05	0.030932	10	28	2.1262E-06	0.046862
100	10	58	5.469E-09	0.093727	74	556	8.293E-05	0.493348	11	30	4.7947E-07	0.267987
1000	11	58	1.722E-08	0.765442	105	828	9.143E-05	6.674479	12	32	3.023E-05	3.155496
2000	11	58	1.722E-08	0.843554	33	180	8.44E-05	2.083458	12	32	4.8997E-05	6.869982
5000	11	58	1.722E-08	2.828467	28	158	7.682E-05	4.857070	12	32	6.2949E-05	60.728033

Table A-8: Test Problem 8 – Diagonal 9.

TP 8	Steepest Descent Method				Conjugate Gradient Method				Spectral Gradient Method			
Dimension (n)	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**
10	-	-	-	-	-	-	-	-	107	566	7.5758E-05	0.281159
100	-	-	-	-	-	-	-	-	405	2056	8.5691E-05	2.600285
1000	-	-	-	-	-	-	-	-	1357	6036	9.2222E-05	153.557265
2000	-	-	-	-	-	-	-	-	1354	5766	4.7957E-05	480.984253
5000	-	-	-	-	-	-	-	-	1692	7008	8.3679E-05	7112.805255

Table A-9: Test Problem 9 – BIGGSB1 (CUTE).

TP 9	Steepest Descent Method				Conjugate Gradient Method				Spectral Gradient Method			
Dimension (n)	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**
10	196	1134	8.91E-05	0.328046	164	1552	9.473E-05	0.290665	106	290	9.9803E-05	0.312426
100	-	-	-	-	1063	10596	8.517E-05	6.238520	5823	17982	9.9546E-05	63.462501
1000	-	-	-	-	5484	61036	9.73E-05	315.030891	-	-	-	-
2000	-	-	-	-	-	-	-	-	-	-	-	-
5000	-	-	-	-	-	-	-	-	-	-	-	-

Table A-10: Test Problem 10 – Raydan 2.

TP 10	Steepest Descent Method				Conjugate Gradient Method				Spectral Gradient Method			
Dimension (n)	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**
10	5	10	6.006E-08	0.015610	9	18	2.518E-05	0.055851	6	12	4.7819E-06	0.031214
100	5	10	1.899E-07	0.031240	9	18	7.964E-05	0.046882	6	12	1.5122E-05	0.070452
1000	5	10	6.006E-07	0.201459	10	20	5.145E-05	0.289199	6	12	4.7819E-05	1.484025
2000	5	10	8.494E-07	0.078108	10	20	7.276E-05	0.128236	6	12	6.7626E-05	2.944845
5000	5	10	1.343E-06	0.219627	11	22	1.87E-05	0.288230	7	14	1.6589E-08	35.217048

Table A-11: Test Problem 11 – Raydan 1.

TP 11	Steepest Descent Method				Conjugate Gradient Method				Spectral Gradient Method			
Dimension (n)	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**
10	61	122	9.665E-05	0.062486	22	44	9.142E-05	0.065114	33	66	8.4207E-05	0.093727
100	191	1164	9.891E-05	1.286014	100	782	9.875E-05	0.485901	131	416	9.5415E-05	0.687335
1000	2977	39122	9.984E-05	223.320333	-	-	-	-	361	1528	8.7275E-05	41.135214
2000	5952	90100	9.88E-05	638.218482	-	-	-	-	466	2074	9.5739E-05	274.855284
5000	-	-	-	-	-	-	-	-	836	3820	9.7701E-05	4516.750336

Table A-12: Test Problem 12 – Diagonal 1.

TP 12	Steepest Descent Method				Conjugate Gradient Method				Spectral Gradient Method			
Dimension (n)	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**
10	27	180	9.651E-05	0.050436	42	338	9.184E-05	0.046902	19	52	9.6945E-05	0.048870
100	238	3146	8.787E-05	3.476604	-	-	-	-	81	288	9.6226E-05	0.798925
1000	-	-	-	-	-	-	-	-	259	1082	9.2007E-05	49.777635
2000	-	-	-	-	-	-	-	-	335	1386	9.6718E-05	184.014975
5000	-	-	-	-	-	-	-	-	788	3072	7.8264E-05	3975.992915

Table A-13: Test Problem 13 – Diagonal 2.

TP 13	Steepest Descent Method				Conjugate Gradient Method				Spectral Gradient Method			
Dimension (n)	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**
10	65	130	9.233E-05	0.093729	23	46	9.188E-05	0.015595	33	66	8.5851E-05	0.093728
100	566	1132	9.924E-05	4.329444	76	152	9.485E-05	0.238334	312	628	9.9712E-05	3.532682
1000	4643	9286	9.999E-05	134.802648	219	438	9.932E-05	5.352855	4663	9524	9.9602E-05	814.979268
2000	8672	17344	9.995E-05	120.167802	298	596	9.836E-05	3.831710	6418	12844	9.9961E-05	2738.229675
5000	-	-	-	-	443	886	9.925E-05	14.858461	-	-	-	-

Table A-14: Test Problem 14 – Hager.

TP 14	Steepest Descent Method				Conjugate Gradient Method				Spectral Gradient Method			
Dimension (n)	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**
10	16	64	6.878E-05	0.015619	8	32	9.762E-05	0.015597	10	24	1.1567E-05	0.102723
100	32	214	9.5E-05	0.140615	37	284	7.844E-05	0.178350	20	58	8.3481E-05	0.285236
1000	102	1010	1.598E-05	5.415768	97	1158	9.743E-05	4.581524	56	190	2.4363E-05	14.186338
2000	163	1758	9.06E-05	9.273931	154	2088	9.42E-05	10.547082	62	226	8.2183E-05	37.687442
5000	129	1466	8.36E-05	20.112497	9780	125280	-	26838.83472	80	304	9.7398E-05	447.313912

Table A-15: Test Problem 15 – Diagonal 4.

TP 15	Steepest Descent Method				Conjugate Gradient Method				Spectral Gradient Method			
Dimension (n)	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**
10	463	6104	9.329E-05	0.827929	31	352	5.743E-05	0.061628	7	38	2.061E-12	0.017271
100	516	6802	8.794E-05	3.046180	36	412	5.576E-05	0.175193	7	38	6.5176E-12	0.111699
1000	564	7434	9.62E-05	26.234876	41	480	9.214E-05	1.987349	7	38	2.061E-11	1.874465
2000	581	7658	9.006E-05	33.035085	43	494	4.388E-05	2.861251	7	38	2.9147E-11	4.612329
5000	600	7908	9.707E-05	91.343327	46	546	7.001E-05	6.024417	7	38	4.6086E-11	43.448089

Table A-16: Test Problem 16 – Quadratic QF 2.

TP 16	Steepest Descent Method				Conjugate Gradient Method				Spectral Gradient Method			
Dimension (n)	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**
10	45	484	8.49E-05	0.062477	94	1308	8.11E-05	0.141028	32	102	2.6673E-05	0.109346
100	-	-	-	-	-	-	-	-	117	456	9.0096E-06	1.582681
1000	-	-	-	-	-	-	-	-	347	1414	9.5194E-05	69.219293
2000	-	-	-	-	-	-	-	-	485	2078	6.9896E-05	286.251893
5000	-	-	-	-	-	-	-	-	730	3128	9.8032E-05	3707.624165

Table A-17: Test Problem 17 – DQDRTIC (CUTE).

TP 17	Steepest Descent Method				Conjugate Gradient Method				Spectral Gradient Method			
Dimension (n)	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**
10	-	-	-	-	-	-	-	-	158	670	7.9618E-05	0.593603
100	-	-	-	-	-	-	-	-	82	328	3.0504E-07	1.117453
1000	-	-	-	-	-	-	-	-	92	410	9.8106E-05	19.517328
2000	-	-	-	-	-	-	-	-	74	302	7.1429E-05	44.279710
5000	-	-	-	-	-	-	-	-	65	284	4.4114E-06	334.735215

Table A-18: Test Problem 18 – ARWHEAD (CUTE).

TP 18	Steepest Descent Method				Conjugate Gradient Method				Spectral Gradient Method			
Dimension (n)	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**
10	13	146	6.717E-05	0.031243	23	238	2.214E-05	0.039565	13	42	1.4344E-05	0.046861
100	-	-	-	-	-	-	-	-	43	146	2.6329E-10	0.532612
1000	-	-	-	-	-	-	-	-	19	80	8.2288E-06	4.257379
2000	-	-	-	-	-	-	-	-	25	106	5.0406E-05	16.335002
5000	-	-	-	-	-	-	-	-	49	248	9.9657E-05	260.881529

Table A-19: Test Problem 19 – Extended Rosenbrock.

TP 19	Steepest Descent Method				Conjugate Gradient Method				Spectral Gradient Method			
Dimension (n)	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**	**No. of Iteration (k)**	**No. of Function Call**	**Norm of g**	**Computational Time (s)**
10	-	-	-	-	-	-	-	-	95	540	6.4118E-05	0.349580
100	-	-	-	-	-	-	-	-	96	540	7.3136E-05	1.453197
1000	-	-	-	-	-	-	-	-	92	508	9.222E-05	22.384946
2000	-	-	-	-	-	-	-	-	151	860	5.7023E-05	98.066637
5000	-	-	-	-	-	-	-	-	106	590	8.7849E-05	553.935570

APPENDIX B: List of Optimization Test Problems

TP1: Quadratic QF1 function

$$f(x) = \frac{1}{2}\sum_{i=1}^{n} ix_i^2 - x_n\,, x_0 = [1,1,\dots,1].$$

TP2: POWER function (CUTE)

$$f(x) = \sum_{i=1}^{n} (ix_i)^2\,, x_0 = [1,1,\dots,1].$$

TP3: QUARTC function (CUTE)

$$f(x) = \sum_{i=1}^{n} (x_i - 1)^4\,, x_0 = [2,2,\dots,2].$$

TP4: Almost Perturbed Quadratic function

$$f(x) = \sum_{i=1}^{n} ix_i^2 + \frac{1}{100}(x_1 + x_n)^2, x_0 = [0.5,0.5,\dots,0.5].$$

TP5: Diagonal 7 function

$$f(x) = \sum_{i=1}^{n} e^{x_i} - 2x_i - x_i^2, x_0 = [1,1,\dots,1].$$

TP6: Diagonal 8 function

$$f(x) = \sum_{i=1}^{n} x_i e^{x_i} - 2x_i - x_i^2, x_0 = [1,1,\dots,1].$$

TP7: Generalized Quartic function

$$f(x) = \sum_{i=1}^{n-1} x_i^2 + (x_{i+1} + x_i^2)^2, x_0 = [1,1,\dots,1].$$

TP8: Diagonal 9 function

$$f(x) = \sum_{i=1}^{n-1} (e^{x_i} - ix_i) + 10000x_n^2\,, x_0 = [1,1,\dots,1].$$

TP9: BIGGSB1 function (CUTE)

$$f(x) = (x_1 - 1)^2 + \sum_{i=1}^{n-1} (x_{i+1} - x_i)^2 + (1 - x_n)^2, x_0 = [0,0,\dots,0].$$

TP10: Raydan 2 function

$$f(x) = \sum_{i=1}^{n} e^{x_i} - x_i, x_0 = [1,1, \dots 1].$$

TP11: Raydan 1 function

$$f(x) = \sum_{i=1}^{n} \frac{i}{10}(e^{x_i} - x_i), x_0 = [1,1, \dots 1].$$

TP12: Diagonal 1 function

$$f(x) = \sum_{i=1}^{n} e^{x_i} - ix_i, x_0 = \left[\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n}\right].$$

TP13: Diagonal 2 function

$$f(x) = \sum_{i=1}^{n} e^{x_i} - \frac{x_i}{i}, x_0 = \left[\frac{1}{1}, \frac{1}{2}, \dots, \frac{1}{n}\right].$$

TP14: Hager function

$$f(x) = \sum_{i=1}^{n} e^{x_i} - \sqrt{i}x_i, x_0 = [1,1, \dots 1].$$

TP15: Diagonal 4 function

$$f(x) = \sum_{i=1}^{n/2} \frac{1}{2}(x_{2i-1}^2 + cx_{2i}^2), x_0 = [1,1, \dots ,1], c = 100.$$

TP16: Quadratic QF2 function

$$f(x) = \frac{1}{2}\sum_{i=1}^{n} i(x_i^2 - 1)^2 - x_n, x_0 = [0.5, 0.5, \dots ,0.5].$$

TP17: DQDRTIC function (CUTE)

$$f(x) = \sum_{i=1}^{n-2} (x_i^2 + cx_{i+1}^2 + dx_{i+2}^2), c = 100, d = 100, x_0 = [3,3, \dots ,3].$$

TP18: ARWHEAD function (CUTE)

$$f(x) = \sum_{i=1}^{n-1} (-4x_i + 3) + \sum_{i=1}^{n-1} (x_i^2 + x_n^2)^2, x_0 = [1,1, \dots ,1].$$

TP19: Extended Rosenbrock function

$$f(x) = \sum_{i=1}^{\frac{n}{2}} c(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2, x_0 = [-1.2,1, \dots , -1.2,1].$$

$$c = 100.$$

APPENDIX C: Tables of Nonlinear Test Problems

Table C-1: Number of Iteration.

Problem	Dim	Modified SG	BFGS	SD	CG
TP1	2	37	9	113	28
TP2	2	185	-	-	-
TP3	4	5281	335	-	-
TP4	10	23	506	-	-
	100	-	-	-	-
	200	-	-	-	-
	500	-	-	-	-
TP5	10	120	11	245	-
	100	-	28	-	-
	200	-	-	-	-
	500	-	-	-	-
TP6	10	111	17	3144	38
	100	-	197	-	-
	200	-	456	-	-
	500	-	1824	-	-
TP7	10	5	5	6	5
	100	8	14	7	18
	200	9	19	7	15
	500	9	16	7	10
TP8	10	17	20	15	-
	100	16	43	16	-
	200	18	46	17	-
	500	17	51	17	-
TP9	10	6	7	7	6
	100	6	7	6	8
	200	6	7	5	9
	500	6	7	5	9
TP10	10	7	7	6	9

	100	7	7	6	9
	200	7	7	6	9
	500	7	7	6	10
TP11	10	46	9	124	28
	100	57	12	140	31
	200	57	11	144	31
	500	61	11	149	36
TP12	10	10	20	13	30
	100	11	42	14	33
	200	11	37	14	34
	500	11	38	14	27
TP13	10	36	14	39	27
	100	395	92	417	94
	200	777	122	828	135
	500	-	206	-	-
TP14	10	419	18	448	76
	100	-	146	-	-
	200	-	317	-	-
	500	-	-	-	-
TP15	10	1	1	1	1
	100	1	1	1	1
	200	1	1	1	1
	500	1	1	1	1
TP16	10	43	16	38	32
	100	389	109	413	93
	200	752	177	-	-
	500	-	559	-	-
TP17	10	6	6	24	21
	100	6	6	27	25
	200	6	6	28	25
	500	6	6	29	27
TP18	10	7	43	34	6

	100	7	43	37	7
	200	7	43	38	7
	500	7	43	39	7
TP19	10	11	21	12	23
	100	12	23	11	27
	200	12	26	11	38
	500	12	25	12	52
TP20	10	1000	-	-	-
	100	-	-	-	-
	200	-	-	-	-
	500	-	-	-	-
TP21	10	108	11	188	35
	100	-	142	-	847
	200	-	256	-	-
	500	-	-	-	-
TP22	10	7	7	6	9
	100	7	8	6	9
	200	7	8	6	10
	500	7	8	6	10
TP23	10	34	17	62	22
	100	156	94	243	88
	200	304	-	484	546
	500	886	-	-	-
TP24	10	30	18	30	42
	100	215	111	229	-
	200	405	174	447	-
	500	-	-	-	-
TP25	10	34	20	66	23
	100	-	87	-	181
	200	-	152	-	343
	500	-	306	-	773
TP26	10	12	16	19	24

	100	40	38	36	26
	200	56	41	63	47
	500	89	69	86	96
TP27	10	185	4	453	55
	100	231	4	504	60
	200	234	4	520	62
	500	246	4	540	65
TP28	10	56	18	54	100
	100	466	118	-	-
	200	905	-	-	-
	500	-	-	-	-
TP29	10	903	13	-	-
	100	880	13	-	-
	200	909	14	-	-
	500	872	14	-	-
TP30	10	14	8	15	18
	100	153	56	-	-
	200	210	187	-	-
	500	263	216	-	-
TP31	10	-	354	-	-
	100	-	-	-	-
	200	-	-	-	-
	500	-	-	-	-

Table C-2: Number of Function Call.

Problem	Dim	Modified SG	BFGS	SD	CG
TP1	2	357	95	1996	477
TP2	2	4087	-	-	-
TP3	4	100047	6437	-	-
TP4	10	115	12263	-	-
	100	-	-	-	-
	200	-	-	-	-
	500	-	-	-	-
TP5	10	1307	80	2713	-
	100	-	491	-	-
	200	-	-	-	-
	500	-	-	-	-
TP6	10	746	143	2200	250
	100	-	2816	-	-
	200	-	7403	-	-
	500	-	39326	-	-
TP7	10	22	32	26	22
	100	37	89	39	86
	200	44	128	45	83
	500	50	113	45	57
TP8	10	79	176	161	-
	100	78	437	174	-
	200	83	458	181	-
	500	82	497	181	-
TP9	10	26	41	30	26
	100	26	41	26	34
	200	26	41	22	38
	500	26	41	22	38
TP10	10	30	44	26	38
	100	30	44	26	38
	200	30	44	26	38

	500	30	44	26	42
TP11	10	468	98	2184	477
	100	593	137	2467	537
	200	593	131	2537	537
	500	648	131	2629	638
TP12	10	51	182	171	350
	100	55	416	184	386
	200	55	368	184	408
	500	55	386	184	305
TP13	10	230	146	434	272
	100	5608	1853	8669	2061
	200	13130	2924	19664	3368
	500	-	5507	-	-
TP14	10	5998	260	10656	1821
	100	-	3098	-	-
	200	-	6476	-	-
	500	-	-	-	-
TP15	10	12	14	12	12
	100	12	14	12	12
	200	12	14	12	12
	500	12	14	12	12
TP16	10	267	179	538	457
	100	5443	2522	9826	2318
	200	12745	4619	-	-
	500	-	14717	-	-
TP17	10	29	38	170	119
	100	29	38	191	141
	200	29	38	198	141
	500	29	38	205	152
TP18	10	60	1007	324	53
	100	60	1007	345	57
	200	60	1007	352	57

	500	60	1007	359	57
TP19	10	58	173	110	229
	100	62	194	103	215
	200	62	221	103	355
	500	62	212	110	543
TP20	10	15105	-	-	-
	100	-	-	-	-
	200	-	-	-	-
	500	-	-	-	-
TP21	10	923	95	1825	331
	100	-	2558	-	16212
	200	-	4751	-	-
	500	-	-	-	-
TP22	10	30	44	26	38
	100	30	47	26	38
	200	30	47	26	42
	500	30	47	26	42
TP23	10	138	104	250	90
	100	1571	1382	2540	876
	200	4035	-	6804	11774
	500	15396	-	-	-
TP24	10	176	155	335	548
	100	2437	2099	4767	-
	200	5513	4100	10622	-
	500	-	-	-	-
TP25	10	138	122	266	94
	100	-	821	-	2067
	200	-	1784	-	4869
	500	-	4385	-	13648
TP26	10	53	101	132	167
	100	255	380	401	322
	200	418	476	809	745

	500	733	896	1243	1253
TP27	10	2764	41	9416	1095
	100	3389	50	10475	1193
	200	3392	56	10809	1228
	500	3629	59	11222	1294
TP28	10	346	224	935	2193
	100	6699	2834	-	-
	200	15724	-	-	-
	500	-	-	-	-
TP29	10	15935	179	-	-
	100	15468	176	-	-
	200	16067	212	-	-
	500	15307	230	-	-
TP30	10	82	89	260	281
	100	1574	638	-	-
	200	2657	2579	-	-
	500	4288	2888	-	-
TP31	10	-	7958	-	-
	100	-	-	-	-
	200	-	-	-	-
	500	-	-	-	-

Table C-3: Computational Time.

Problem	Dim	Modified SG	BFGS	SD	CG
TP1	2	0.0377	0.0156	0.1588	0.0327
TP2	2	0.3477	-	-	-
TP3	4	12.8596	0.7998	-	-
TP4	10	0.0659	1.9320	-	-
	100	-	-	-	-
	200	-	-	-	-
	500	-	-	-	-
TP5	10	0.2786	0.0156	0.3791	-
	100	-	0.2031	-	-
	200	-	-	-	-
	500	-	-	-	-
TP6	10	0.2364	0.0409	0.4999	0.0468
	100	-	1.7897	-	-
	200	-	12.8433	-	-
	500	-	141.8134	-	-
TP7	10	0.0183	0.0252	0.0156	0.0156
	100	1.3747	3.3273	1.2978	2.7794
	200	6.3383	17.6500	5.9025	11.1102
	500	39.5548	90.4540	35.7008	46.3121
TP8	10	0.0345	0.0469	0.0199	-
	100	0.0937	0.3302	0.0781	-
	200	0.2656	0.7811	0.1480	-
	500	0.6092	2.6087	0.3593	-
TP9	10	0.0159	0.0160	0.0156	0.0157
	100	0.0469	0.0468	0.0270	0.0299
	200	0.1187	0.1249	0.0312	0.0625
	500	0.2343	0.3280	0.0781	0.1406
TP10	10	0.0169	0.0190	0.0156	0.0100
	100	0.0468	0.0625	0.0312	0.0469
	200	0.1249	0.1406	0.0459	0.0468

	500	0.2499	0.3436	0.0938	0.1406
TP11	10	0.1037	0.0280	0.2500	0.0548
	100	0.588	0.1406	1.6251	0.3280
	200	1.6090	0.3437	2.9293	0.6249
	500	4.0928	0.9060	8.0522	1.9062
TP12	10	0.0355	0.0941	0.0469	0.1250
	100	0.0986	0.5075	0.1657	0.2968
	200	0.2031	0.6717	0.2022	0.4975
	500	0.4374	2.2963	0.5097	0.9060
TP13	10	0.3348	0.0496	0.0419	0.0312
	100	2.9450	0.7132	2.5780	0.5201
	200	16.5315	2.6681	9.0869	1.4572
	500	-	13.7111	-	-
TP14	10	1.5635	0.0457	0.8904	0.1406
	100	-	1.3542	-	-
	200	-	8.5907	-	-
	500	-	-	-	-
TP15	10	0.0063	0.0065	0	0
	100	0.0187	0.0194	0.0156	0.0130
	200	0.0451	0.0503	0.0249	0.0312
	500	0.1036	0.1114	0.0558	0.0625
TP16	10	0.1210	0.0400	0.0469	0.0298
	100	2.9416	0.9077	2.7168	0.5624
	200	16.1627	5.7721	-	-
	500	-	39.3981	-	-
TP17	10	0.0161	0.0153	0.0312	0.0312
	100	0.0395	0.0436	0.1106	0.109
	200	0.0800	0.0979	0.2031	0.1666
	500	0.2332	0.3102	0.5035	0.4062
TP18	10	0.0200	0.1439	0.0469	0.0156
	100	0.0710	0.7358	0.2595	0.0469
	200	0.1608	1.9194	0.4766	0.0937

	500	0.4561	4.9442	1.2028	0.2045
TP19	10	0.0270	0.0810	0.0210	0.0469
	100	0.1235	0.2830	0.1249	0.2343
	200	0.2702	0.8715	0.1963	0.7141
	500	0.7603	2.2219	0.5230	2.6054
TP20	10	4.6466	-	-	-
	100	-	-	-	-
	200	-	-	-	-
	500	-	-	-	-
TP21	10	0.3965	0.0402	0.3112	0.0469
	100	-	1.8850	-	9.3494
	200	-	9.3099	-	-
	500	-	-	-	-
TP22	10	0.0146	0.0196	0	0.0156
	100	0.0394	0.0471	0.0156	0.0469
	200	0.0762	0.0877	0.0389	0.0625
	500	0.2243	0.3465	0.0625	0.1147
TP23	10	0.0781	0.0312	0.0540	0.0312
	100	1.3068	0.8830	1.3822	0.4718
	200	6.7494	-	5.5519	8.3839
	500	59.4872	-	-	-
TP24	10	0.0468	0.0534	0.0389	0.0469
	100	1.6106	0.9886	1.8360	-
	200	7.5562	4.3202	5.8892	-
	500	-	-	-	-
TP25	10	0.0625	0.0469	0.0469	0.0230
	100	-	0.4414	-	0.6561
	200	-	2.0771	-	2.2784
	500	-	14.3440	-	12.2185
TP26	10	0.0378	0.0378	0.0312	0.0312
	100	0.2025	0.2090	0.1250	0.1093
	200	0.6721	0.5710	0.3905	0.3173

	500	3.1619	3.3452	1.3098	1.4216
TP27	10	0.4545	0.0156	0.7967	0.0625
	100	1.5540	0.0378	2.7672	0.2968
	200	3.8791	0.0846	4.7032	0.5186
	500	12.5077	0.2124	11.4012	1.2728
TP28	10	0.1381	0.0534	0.0958	0.2274
	100	5.9972	2.1725	-	-
	200	33.7302	-	-	-
	500	-	-	-	-
TP29	10	2.5528	0.0469	-	-
	100	6.6330	0.1023	-	-
	200	16.3265	0.2795	-	-
	500	43.3721	0.8184	-	-
TP30	10	0.0313	0.0313	0.0469	0.0519
	100	2.6854	1.0350	-	-
	200	10.5229	10.7394	-	-
	500	42.5368	31.2650	-	-
TP31	10	-	1.4573	-	-
	100	-	-	-	-
	200	-	-	-	-
	500	-	-	-	-

APPENDIX D: List of Nonlinear Test Problems

TP1: Freudenstein and Roth function: $F(x) = (f_1(x), f_2(x))^T$

$$f_1(x) = 10(x_2 - x_1^2)$$

$$f_2(x) = 1 - x_1$$

$$x_0 = [6, 3]^T$$

TP2: Beale function: $F(x) = (f_1(x), f_2(x))^T$

$$f_1(x) = 1.5 - x_1(1 - x_2)$$

$$f_2(x) = 2.25 - x_1(1 - x_2^2)$$

$$x_0 = [1, 1]^T$$

TP3: Wood function: $F(x) = (f_1(x), f_2(x), f_3(x), f_4(x))^T$

$$f_1(x) = 200x_1(x_1^2 - x_2) + x_1 - 1$$

$$f_2(x) = 100(x_2 - x_1^2) + 10(x_2 + x_4 - 2) + \frac{1}{10}(x_2 - x_4)$$

$$f_3(x) = 180x_3(x_3^2 - x_4) + x_3 - 1$$

$$f_4(x) = 90(x_4 - x_3^2) + 10(x_2 + x_4 - 2) - \frac{1}{10}(x_2 - x_4)$$

$$x_0 = [-3, -1, -3, -1]^T$$

TP4: Variably dimensioned function: $F(x) = (f_1(x), f_2(x), \dots, f_n(x))^T$

$$f_i(x) = x_i - 1 + i\sum_{j=1}^{n} j\left(x_j - 1\right) + 2i\left(\sum_{j=1}^{n} j(x_j - 1)\right)^3$$

$$x_0 = [\frac{n-1}{n}, \frac{n-2}{n}, \dots, \frac{1}{n}, 0]^T$$

TP5: Brown almost-linear function: $F(x) = (f_1(x), f_2(x), \dots, f_n(x))^T$

$$f_i(x) = x_i + \sum_{j=1}^{n} x_j - (n+1), for\ i = 1,2, \dots, n-1,$$

$$f_n(x) = \left(\prod_{j=1}^{n} x_j\right) - 1$$

$$x_0 = [\frac{1}{2}, \frac{1}{2}, \dots, \frac{1}{2}]^T$$

TP6: Discrete boundary function: $F(x) = (f_1(x), f_2(x), \dots, f_n(x))^T$

$$f_i(x) = 2x_i - x_{i-1} - x_{i+1} + 0.5h^2(x_i + t_i + 1)^3, i = 1,2, \dots n,$$

$$h = \frac{1}{n+1}, t_i = ih \; and \; x_0 = x_{n+1} = 0$$

$$x_0 = [t_1(t_1 - 1), t_2(t_2 - 1), \dots, t_n(t_n - n)]^T$$

TP7: Discrete integral equation function: $F(x) = (f_1(x), f_2(x), \dots, f_n(x))^T$

$$f_i(x) = x_i + 0.5h[(1 - t_i)\sum_{j=1}^{i} t_j(x_i + t_i + 1)^3 + t_i \sum_{j=i+1}^{n} (1 - t_j)(x_i + t_i + 1)^3], i = 1,2, \dots n,$$

$$h = \frac{1}{n+1}, t_i = ih \; and \; x_0 = x_{n+1} = 0$$

$$x_0 = [t_1(t_1 - 1), t_2(t_2 - 1), \dots, t_n(t_n - n)]^T$$

TP8: Broyden tridiagonal function: $F(x) = (f_1(x), f_2(x), \dots, f_n(x))^T$

$$f_i(x) = (3 - 2x_i)x_i - x_{i-1} - 2x_{i+1} + 1, for \; i = 1,2, \dots, n,$$

$$x_0 = x_{n+1} = 0$$

$$x_0 = [-1, -1, \dots, -1]^T$$

TP9: Logarithmic function: $F(x) = (f_1(x), f_2(x), \dots, f_n(x))^T$

$$f_i(x) = \ln(x_i + 1) - \frac{x_i}{n}, for \; i = 1,2, \dots, n,$$

$$x_0 = [1,1, \dots, 1]^T$$

TP10: Strictly convex function: $F(x) = (f_1(x), f_2(x), \dots, f_n(x))^T$

$$f_i(x) = e^{x_i} - 1, for \; i = 1,2, \dots, n,$$

$$x_0 = [\frac{1}{n}, \frac{2}{n}, \dots, 1]^T$$

TP11: Extended Freudenstein and Roth function: $F(x) = \big(f_1(x), f_2(x), \dots, f_n(x)\big)^T$

$$f_{2i-1}(x) = x_{2i-1} + \big((5 - x_{2i})x_{2i} - 2\big)x_{2i} - 13, for \; i = 1,2, \dots, \frac{n}{2},$$

$$f_{2i}(x) = x_{2i-1} + \big((1 + x_{2i})x_{2i} - 14\big)x_{2i} - 29, for \; i = 1,2, \dots, \frac{n}{2}$$

$$x_0 = [6,3,6,3, \dots, 6,3]^T$$

TP12: The discretized two-point boundary value function

$$F(x) = Ax + \frac{1}{(n+1)^2} G(x),$$

$$where\ A\ is\ the\ n \times n\ tridiagonal\ matrix\ given\ by$$

$$\begin{matrix} 8 & -1 & & & \\ -1 & 8 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 8 \end{matrix}$$

$$and\ G(x) = (sinx_1 - 1, sinx_2 - 1, \dots, sinx_n - 1)^T$$

$$x_0 = [50,0,50,0, \dots ,50,0]^T$$