Department of Electrical and Computer Engineering

Numerical Analysis in Nonlinear Least Squares Methods and Applications

Christina Nguk Ling Eu

This thesis is presented for the Degree of
Master of Philosophy (Electrical and Computer Engineering)
of
Curtin University

December 2017

DECLARATION

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgment has been made.

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Signature: [illegible]

Date: 21 / 12 / 2017

ACKNOWLEDGEMENTS

I would like to express my special thanks to my main supervisor, Professor Bean San Goh, for his quality guidance, continual encouragement and endless support throughout my MPhil study. During my research, he has provided me with invaluable academic advice and insightful suggestions and steered me into the right direction whenever I am heading towards the wrong path in my research. The door to his office was always open whenever I ran into a trouble spot or had a problem on my research or writing. He is not only a good supervisor but as a good friend as well.

I would also like to thank my co-supervisor, Dr. Hendra Gunawan Harno, for his continual patience, guidance and support throughout my research. He has always been helpful and supportive whenever I faced any academic and administrative issues. He always tried his best to help me by providing any possible assistance. My sincere thanks also goes to my second co-supervisor, Dr. Garenth King Hann Lim, for his support and cooperation throughout my years of study. Without their passionate input, this research project could not have been successfully conducted.

I would also like to express my appreciation to the chairperson of my thesis committee, Associate Professor Zhuquan Zang, for his encouragement and continual support throughout these years. He has really done a great job in ensuring that my studies were carried out well.

I would also like to acknowledge Professor Clem Kuek, the Dean of Research and Development, for his continual support in improving the quality of my research. He has always tried to organize research related seminars and talks that are beneficial to improve the quality of the research output.

It is also a pleasure to thank my colleagues for their friendly support and friendship that blossomed throughout the duration of my studies. They have enriched and broadened my knowledge and perspectives on various aspects of life.

I must express my very profound gratitude to my parents, especially my mother, for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

Last, but not least, I appreciate with gratitude the fee waiver award I received from Faculty of Engineering and Science for granting me an opportunity to conduct MPhil study in Electrical and Computer Engineering, Curtin University Malaysia. Thank you.

ABSTRACT

A nonlinear least squares (NLS) problem commonly arises in nonlinear data-fitting when a nonlinear mathematical model with n unknown parameters is used to fit a set of m observed data with $m > n$. The best fit to the m observed data is achieved when the residuals between the observed data and its corresponding fitted modeled data are minimized. This is made possible by minimizing an objective function formulated as the sum of squares residual functions of all the m observed data. This procedure is also known as parameter estimation in NLS data-fitting. As a result, the NLS problem is a special class of unconstrained minimization problem and the solution of this minimization problem yields the minimum point which gives the minimal value of the objective function of the NLS problem.

Various numerical methods have been developed to solve the NLS problem as unconstrained optimization. These methods can be classified into line search methods or trust region methods. In this thesis, four of the most well-known numerical methods in the NLS literature are considered. The line search methods considered are the steepest descent (SD) method, the Newton's method and the Gauss-Newton (GN) method while the only trust region method considered is the Levenberg-Marquardt (LM) method.

In order to avoid expensive computations of the Hessian matrix in each iteration, the GN and the LM methods use, without justification, a truncated Hessian matrix of the objective function of the NLS problem. However, this truncated Hessian matrix may not be valid especially when the iterations result in large residuals. In addition,

the computation of the derivatives of the objective function may be prone to analytical mistakes. This is especially true when computing derivatives for high-dimensional NLS problem. To address these issues, numerical differentiation, which uses finite difference approximations, is incorporated into numerical algorithms so that numerical derivatives can be performed by just providing the original objective function of the NLS problem. This saves time and effort while preventing analytical mistakes. Thus, the use of the truncated Hessian matrix can be avoided when developing new numerical methods for solving the NLS problem. With the incorporation of numerical differentiation, all the numerical methods can be implemented in practical problems.

The convergence analyses of the numerical methods follow from the Lyapunov function theorem where a sufficient decrease in the objective function is required at every iteration. The Lyapunov function theorem provides a feedback-type analysis which is robust against small numerical errors in the current iteration. If the level sets of the objective function are properly nested, all trajectories will converge to a minimum point x^* provided that the iterations stay within the properly nested region containing x^*. In order to implement the Lyapunov function theorem, all the line search numerical methods perform a backtracking line search so as to ensure a sufficient decrease of the objective function value of the NLS problem at every iteration. On the other hand, this sufficient decrease requirement of the Lyapunov function theorem is also ensured implicitly in the trust region LM method through the ratio test.

When using the MATLAB software to plot the level sets of a two-variable objective function, the level curves near stationary points may not appear in the plot. Hence, a stiff ordinary differential equation (ODE) method, which gives great control to the user, is used as a technique to plot a missing level curve around the stationary points of the objective function through a specific point. This is particularly useful when the objective function has multiple stationary points that are close to each other.

The approximate greatest descent (AGD) method has been developed to solve an unconstrained optimization problem. Unlike other methods, the AGD method uses the actual objective function to construct its iterations instead of an approximate linear or

quadratic model. Furthermore, the AGD method is constructed for long-term suboptimal outcomes to generate the next iterative point on the boundary of the current search region. However, it has not been applied to solve the NLS problem. In addition, a two-phase AGD method (abbreviated as AGDN) is also proposed as a new numerical approach to solve the NLS problem. It consists of two explicitly defined phases with AGD method in Phase-I where the current iterations are far away from the minimum point and then switches to the Newton's method in Phase-II when the gradient is sufficiently small (i.e. near the minimum point). This method is motivated by the fast quadratic convergence rate of the Newton's method near the minimum point.

In order to demonstrate the efficiency, reliability and robustness of all the numerical methods, a standard set of two-variable and multi-variable test problems are selected from Moré et al. (1981) and Adorio (2005) and available in the constrained and unconstrained testing environment, revisited/safe threads (CUTEr/CUTEst) are used to perform numerical experiments. Furthermore, a performance profile is also used as a tool to provide an overall comparison of all the numerical methods in terms of number of iterations and the CPU time used to achieve convergence. When the numerical methods are applied to solve the two-variable and the multi-variable test problems, the numerical results indicate that both the AGD and the AGDN methods have shown encouraging results in terms of number of iterations and convergence rates as compared to the other methods. For the two-variable test problems, the AGD and the AGDN methods show similar results. However, the outcomes of these methods may differ when they are applied to solve the multi-variable test problems. The results prove that the AGDN method outperforms the AGD method since it has a faster convergence rate with less number of iterations. Nonetheless, the AGDN method may fail to converge if the Hessian matrix is singular near the minimum point. In this case, the AGD method should be used instead.

ABBREVIATIONS

Throughout the report, we use the following abbreviations:

NLS:	Nonlinear least squares
SD:	Steepest descent
GN:	Gauss-Newton
LM:	Levenberg-Marquardt
AGD:	Approximate greatest descent
AGDN:	Two-phase approximate greatest descent
TP:	Test Problem
w.r.t:	with respect to

NOTATIONS

Throughout the report, we use the following notations:

$\mathbb{R}$:	the set of all real numbers
$\|.\|$:	the Euclidean norm
m:	the number of equations in the NLS problem
n:	the number of unknown parameters in the NLS problem
x_0:	the initial/starting point of numerical iterations
x^*:	solution/minimum point of the NLS problem
x_{max}:	maximum point of the NLS objective function
k:	number of iterations
p_k:	search direction
α_k:	step length
$\nabla(.)$:	gradient of a function
$\Delta(.)$:	change of value of a function
μ_k:	Lagrange parameter

λ_k:	Lagrange multiplier
$F(.)$:	the objective function of the NLS problem
$r_i(.)$:	the residual functions with $i = 1, 2, \ldots, m$
$V(.)$:	Lyapunov function
$g(.)$:	gradient vector
$H(.)$:	Hessian matrix
$H_T(.)$:	truncated Hessian matrix
$J(.)$:	Jacobian matrix

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION TO NLS

This chapter gives an introduction to nonlinear least squares (NLS) problem through its important applications in data-fitting in various disciplines. This is followed by identifying some issues and drawbacks of the existing methods used to solve the NLS problem. A brief explanation of the solution to each issue and drawback, which is later implemented in numerical algorithms for NLS problem, is provided. The aims and objectives of this research and its significance are also stated. Then, a brief explanation for scope of the research is given. This is followed by an outline of the thesis.

1.1. Research background

A fundamental idea behind any NLS problem is to find n unknown parameters $x = [x_1, x_2, \dots, x_n]^T$ of a nonlinear mathematical fitting model $y = M(x, t)$ such that it provides the best fit to the m observed data points $(t_1, y_1), (t_2, y_2), \dots, (t_m, y_m)$ with $m > n$ (Boukamp, 1986; Wraith and Or, 1998; Schafer et al., 2002; Spalek et al., 2005; Waseda et al., 2008; Sapienza et al., 2015; El-Hayek et al., 2015; Gibson et al., 2016). This best fit is achieved when the residuals $r_i(x)$ between the observed data y_i

and its corresponding fitted modeled data $y = M(x, t_i)$ i.e.

$$r_i(x) = y_i - M(x, t_i) \quad \text{for} \quad i = 1, ..., m, \tag{1.1}$$

are made as small as possible. Consequently, this gives rise to a problem of solving an over-determined system of nonlinear equations.

Note that a nonlinear mathematical model $y = M(x, t)$ is an equation with its parameters $x = [x_1, x_2, \ldots, x_n]^T$ appearing nonlinearly in the equation, or a combination of linear and nonlinear formulation of these parameters. A parameter $x_j \in \mathbb{R}$ of $M(x, t)$ appears nonlinearly if the partial derivative $\frac{\partial M}{\partial x_j}$ is a function of x_j (Hansen et al., 2013). The following example shows how to determine the nonlinearity of a mathematical model.

Example 1.1. Consider the following non-normalized Gaussian function (Hansen et al., 2013)

$$M(x, t) = x_1 \mathrm{e}^{-\left[\frac{(t-x_2)^2}{2x_3^2}\right]}$$

where the parameters x_1, x_2 and x_3 denote the amplitude, the time shift and the width of the Gaussian function respectively. The partial derivatives of the function w.r.t its parameters are given by

- $\frac{\partial M}{\partial x_1} = \mathrm{e}^{-\left[\frac{(t-x_2)^2}{2x_3^2}\right]}$ which is independent of x_1;
- $\frac{\partial M}{\partial x_2} = \frac{x_1}{x_3^2}(t - x_2)\, \mathrm{e}^{-\left[\frac{(t-x_2)^2}{2x_3^2}\right]}$ which is dependent of x_2;
- $\frac{\partial M}{\partial x_3} = \frac{x_1}{x_3^3}(t - x_2)^2\, \mathrm{e}^{-\left[\frac{(t-x_2)^2}{2x_3^2}\right]}$ which is dependent of x_3;

and hence the Gaussian function is a nonlinear mathematical model since its parameters x_2 and x_3 appear nonlinearly in the model.

Other examples of nonlinear fitting models include a ratio of polynomials and power functions. The procedure of finding the n unknown parameters, which is also known as parameter estimations in NLS data-fitting, has various applications in areas such as physics, chemistry, biology, engineering, economics and finance (Wang et al., 2005; Weng et al., 2006; Chudamani et al., 2009; Koesler and Schymura, 2015).

As an example, Figure 1.1 below illustrates a nonlinear Osborne II fitting model, $M(x,t) = x_1 \mathrm{e}^{-x_5[\frac{t-1}{10}]} + x_2 \mathrm{e}^{-x_6[\frac{t-1}{10}-x_9]^2} + x_3 \mathrm{e}^{-x_7[\frac{t-1}{10}-x_{10}]^2} + x_4 \mathrm{e}^{-x_8[\frac{t-1}{10}-x_{11}]^2}$ with $n = 11$ unknown parameters and $m = 65$ data points.

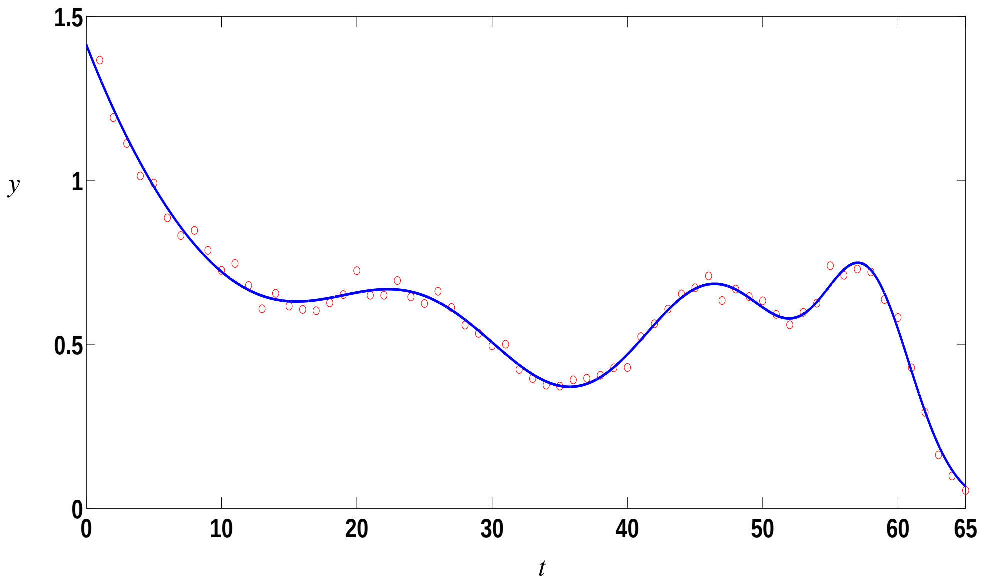


Figure 1.1. A nonlinear Osborne II fit. The red circled symbols denote the m data points and the blue curve represents the nonlinear Osborne II fitting model, $M(x,t) = x_1 \mathrm{e}^{-x_5[\frac{t-1}{10}]} + x_2 \mathrm{e}^{-x_6[\frac{t-1}{10}-x_9]^2} + x_3 \mathrm{e}^{-x_7[\frac{t-1}{10}-x_{10}]^2} + x_4 \mathrm{e}^{-x_8[\frac{t-1}{10}-x_{11}]^2}$ with $n = 11$ unknown parameters and $m = 65$ data points.

1.2. Research gaps and questions

Various numerical methods have been developed and modified to solve the NLS problem. Established methods include the steepest descent (SD) method, the Newton's method, the Gauss-Newton (GN) method and the Levenberg-Marquardt (LM) method (Dennis and Schnabel, 1983; Madsen et al., 2004; Hansen et al., 2013). Among all, the SD method, which was proposed by Cauchy in 1827, is considered as one of the oldest line search methods. Since then, it is usually assumed that the SD method has the best search direction. However, it uses an exact step length which is chosen such that it minimizes the next objective function value of an NLS problem in the given search direction. Such a strategy of using the exact step length is considered to be short-term optimal and is also used in the GN and the Newton's methods. This strategy is generally not ideal in practice since it may lead to numerical method failures (Goh and McDonald, 2015).

Both the GN and the LM methods use, without justification, a truncated Hessian matrix of the objective function of the NLS problem to avoid expensive computations of the nonlinear part (or the tensor terms) of the Hessian matrix in each iteration. However, the truncated Hessian may not be valid especially when the iterations are computed far away from the optimal solution (Nocedal and Wright, 2006). As a result, it may lead to failure of a numerical method. The use of truncated Hessian matrix can be avoided by the implementation of numerical differentiation in numerical methods.

The implementation of numerical differentiation, which uses the finite differencing, avoids the need to evaluate the derivatives of the objective function of the NLS problem analytically. This saves a lot of time and effort while preventing any evaluation mistakes done analytically. Moreover, numerical computations of the derivatives can be performed easily by just providing the original objective function of the NLS problem explicitly.

The convergence proof of most numerical methods follows from the Zoutendijk theorem (Nocedal and Wright, 2006). However, the Zoutendijk theorem only ensures the convergence of a trajectory from an initial point to a stationary point in an open loop manner. As a consequence, the trajectory may converge to a point which is either a maximum point, a minimum point or even a saddle point. This implies that it is possible to achieve an undesirable convergence towards a maximum point or a saddle point. Instead, one should consider the use of the Lyapunov function theorem as convergence analysis.

In this thesis, the convergence analyses of the numerical methods follow from the Lyapunov function theorem where a sufficient decrease of the objective function value is required at every iteration. The Lyapunov function theorem provides a feedback-type analysis which is robust against small numerical errors in the current iteration. It ensures the convergence of a numerical method towards a minimum point from any initial point provided that the objective function has properly nested level sets in a bounded region containing the minimum point (Goh et al., 2014). In order to implement the Lyapunov function theorem, all line search numerical methods performs a backtracking line search technique so as to ensure a sufficient decrease of the objective function value of the NLS problem at every iteration. On the other hand, this sufficient decrease requirement of the Lyapunov function theorem is also ensured implicitly in the trust region LM method through the ratio test.

When using the MATLAB software to plot the level sets of a two-variable objective function, the level curves near the stationary points may not appear in the plot. One solution to plot the missing level curves near different stationary points is to employ the stiff ordinary differential equation (ODE) method. With the help of the stiff ODE method, one can plot a level curve of the objective function through a specific point. This is particularly useful when the NLS objective function has multiple stationary points that are close to each other.

Recently, the approximate greatest descent (AGD) method has been developed by Goh (2009) to solve unconstrained optimization problems. Unlike the LM method, the AGD method uses the full Hessian matrix of the NLS objective function to construct its algorithm. Furthermore, the AGD method is constructed for long-term suboptimal outcomes to generate the next iterative point on the boundary of the current search region. Besides that, the convergence analysis of the AGD method follows from the Lyapunov function theorem. It was shown that the AGD method is capable of dealing with singular Hessian matrix of the Rosenbrock function with the initial point $\left(-1, \frac{201}{200}\right)$ and the indefinite Hessian matrix of the Powell's problem with the initial point $(0, 0)$. However, to date, the AGD method has not been applied to solve the NLS problem. This AGD method, which is the main focus of this research, will be developed on the MATLAB platform to solve the NLS problem.

In addition, a two-phase AGD method (abbreviated as AGDN) is also introduced to solve the NLS problem. It consists of two explicitly defined phases with the AGD method in Phase-I to compute iterations that are far away from the minimum point and then switches to the Newton's method in Phase-II when the gradient of the objective function is sufficiently small (i.e. near the minimum point). This idea is motivated by the fast quadratic convergence rate of the Newton's method near the minimum point.

From the above discussion on the research gaps in the NLS literature, this thesis concentrates on providing solutions to the following questions:

1. Can the incorporation of numerical differentiation using the finite differencing provides useful approximations to the derivatives of the objective function of the NLS problem?

2. How effective are the newly developed AGD and AGDN methods when they are applied to solve the NLS problem in terms of performances (i.e. efficiency, reliability and robustness) of a numerical method?

1.3. Aims and objectives

The aim of this research is to develop new numerical algorithms for solving the NLS problem. The objectives of this research are:

(1) To modify the existing methods, which are the SD, the Newton's, the GN and the LM methods and the newly developed AGD and the AGDN methods for solving NLS problem with the implementation of numerical differentiation using the finite differencing for computing derivatives and Lyapunov function theorem as convergence analysis.

(2) To employ the stiff ODE method to plot the missing level curves near different stationary points of the objective function of the NLS problem.

(3) To investigate the efficiency, reliability and robustness of the newly developed AGD and the AGDN methods for solving the NLS problem on the MATLAB platform by applying them to solve a well-known set of NLS test problems selected from Moré et al. (1981) and Adorio (2005) and available in CUTEst whose results are later compared critically with those of the SD, the Newton's, the GN and the LM methods.

1.4. Significance of research

As pointed out in Section (1.2), the strategy of the SD, the GN and the Newton's methods is only short-term optimal whereby exact step lengths are applied throughout numerical iterations. It is then not surprising that these methods can and may fail to solve the NLS problem. Hence, new numerical methods presented in this thesis are aimed to overcome this problem and their significant features are outlined as follows:

(1) The AGD method, which uses the actual objective function to construct its iterations, will be applied to solve the NLS problem. Unlike other existing methods, it is constructed in a logical and systematic manner for long-term suboptimal outcomes.

(2) An explicitly defined two-phase AGD method (abbreviated as AGDN), which has a faster convergence rate compared to the AGD method, is constructed and applied to solve the NLS problem.

(3) The stiff ODE method is employed to plot the missing level curve near a stationary point of an objective function by choosing a specific point through which the curve passes through.

(4) The Lyapunov function theorem, which provides feedback-type analysis, acts as a key tool for the convergence of a numerical method towards a minimum point in the NLS problem.

(5) The implementation of numerical differentiation (i.e. using the finite differencing) into numerical algorithms avoids the need of a truncated Hessian matrix. The truncated Hessian matrix may be an invalid approximation of the original Hessian matrix especially when the iterations are computed at a point with large residuals.

(6) MATLAB programs for the the new AGD and the AGDN methods are developed, tested, critically analyzed and compared with the SD, the Newton's, the

GN and the LM methods using a standard set of NLS test problems given in Moré et al. (1981) and Adorio (2005) and available in CUTEst.

1.5. Research scope

This thesis focuses on developing Lyapunov-based numerical methods for solving the NLS problem. In this regard, the Lyapunov function theorem is used in the convergence analysis of the numerical methods where a sufficient decrease of the objective function value is required at every iteration. This is achieved by ensuring a monotonic decrease of the objective function of the NLS problem so that convergence towards a minimum point is guaranteed. For line search numerical methods, a backtracking line search is used to ensure this monotonic decrease of the objective function. On the other hand, for trust region numerical method, this sufficient decrease requirement of the Lyapunov function theorem can be ensured implicitly through the ratio test. Four well-known numerical methods in the NLS literature are considered in this research – the SD method, the Newton's method, the GN method and the LM method. In addition, the AGD method, which is a new numerical method for the NLS problem, is also applied to solve the NLS problem. Furthermore, a two-phase version of the AGD method, abbreviated as AGDN, is constructed and compared with the AGD method and the other numerical methods.

1.6. Outline of the thesis

In this thesis, a brief description of an NLS problem is first presented with a simple example. Following that, some issues and drawbacks of the existing methods used to solve the NLS problem are identified and explained. The aims and objectives of this research and its significance are then listed. Finally, the scope of this thesis is explained in brief details.

Chapter 2 provides a review of some of the most well-known numerical methods use to solve the NLS problem. Furthermore, some important issues which are neglected in the NLS literature are identified. For instance, the major difference between the short-term and long-term optimal iterations in NLS are explained using a multi-stage network optimization path. Moreover, the significance of the tensor terms of the Hessian matrix of the objective function, which is often neglected, is emphasized with an NLS example.

In Chapter 3, some new approaches used to solve the NLS problem are proposed. These include the use of Lyapunov function theorem for convergence analysis of a numerical method, the practical importance of numerical differentiation to compute derivatives numerically and also the use of the stiff ODE method to plot the level curves of an objective function through a specific point. All the numerical methods discussed in this thesis are modified to incorporate the Lyapunov function theorem and numerical differentiation into their algorithms. Furthermore, the newly developed AGD and the AGDN methods are also applied to solve the NLS problem.

In Chapter 4, some numerical experiments are carried out to test and compare the efficiency, reliability and robustness of the SD method, the Newton's method, the GN method, the LM method, the AGD method and the AGDN method based on a set of two-variable and multi-variable NLS test problems selected from Moré et al. (1981) and Adorio (2005) and available in CUTEst. The numerical experiments are conducted using the MATLAB programming language where the codes and syntaxes are constructed based on the algorithms defined in Chapter 3.

In Chapter 5, the applications of NLS in data-fitting are presented using some of the test problems selected from Chapter 4. Based on the numerical solution (or minimum point) of the test functions, a least squares fitting curve is plotted for each test problem. From these plots, one can conclude that the solutions obtained from the numerical methods have provide good fitting curves for the given data points.

Chapter 6 gives an overall conclusion of the research project on the numerical methods for solving NLS problem. This is followed by some suggestions on the future work that can be carried out for further research.

Finally, a list of NLS test problems used in the numerical experiments in Chapter 4 are provided in Appendix A for the convenience of the reader.

CHAPTER 2

NUMERICAL METHODS FOR SOLVING NLS PROBLEM

This chapter provides a literature review of the existing numerical methods used to solve NLS problem. The mathematical formulation of an NLS problem and a general iterative equation involved in its numerical processes are first presented. This is followed by an analysis which distinguish a major difference between the short-term and the long-term optimal iterations via a multi-stage network optimization path. Furthermore, the mathematical expressions for the linear and quadratic models used to approximate an objective function of the NLS problem is given. Following that, the importance of the tensor terms of the Hessian matrix is emphasized with a simple NLS example. The optimality conditions which govern the properties of the optimal solution of the NLS problem are then stated. Moreover, a description of the different types of convergence rates of the numerical methods for NLS problem are also provided. Finally, the existing numerical methods used to solve the NLS problem are reviewed by identifying any shortcomings and difficulties associated with each method.

2.1. Mathematical formulation of NLS

As discussed in Section (1.1) of Chapter 1, an NLS problem commonly appears in data-fitting where a nonlinear mathematical model $y = M(x,t)$ with n unknown parameters $x = [x_1, x_2, \dots, x_n]^T$ is used to fit a set of m observed data points (t_1, y_1), $(t_2, y_2), \dots, (t_m, y_m)$ with $m > n$. This is achieved by finding the n unknown parameters of the fitting model $y = M(x,t)$ such that it provides the best fit to the m observed data points. Mathematically, the best approximation to obtain the n unknown parameters can be achieved by minimizing an objective function formulated as the sum of squares residual functions $r_i(x)$ of all the m observed data (Björck, 1996; Pav, 2005; Gander et al., 2014). As a result, the NLS problem is considered as a special class of unconstrained optimization problem defined as follows (Dennis and Schnabel, 1983):

Definition 2.1. Find a minimum point x^* of a nonlinear objective function F, i.e.

$$x^* = \underset{x}{\operatorname{argmin}}\, F(x)$$

where

$$F(x) = \frac{1}{2}\sum_{i=1}^{m} r_i(x)^2 = \frac{1}{2} \parallel r(x) \parallel_2^2, \qquad x \in \mathbb{R}^n, \tag{2.1}$$

with $r(x) = \begin{bmatrix} r_1(x) \\ \vdots \\ r_m(x) \end{bmatrix} \in \mathbb{R}^m$ and $F : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable for $m > n$.

Remark 2.1. In the NLS literature, an NLS problem is normally defined for $m \geqslant n$. However, there is a significant difference between solving a system of equations for $m > n$ and $m = n$. When $m > n$, it involves solving a system of over-determined system of nonlinear equations while the latter only involves solving a system of simultaneous equations.

All numerical methods use to solve NLS problem are iterative in nature which means that iterations start from an initial point x_0, then for each iteration a search direction p_k and a step length α_k are computed to give the iterative equation

$$x_{k+1} = x_k + \alpha_k p_k, \quad k = 0, 1, \dots. \tag{2.2}$$

As a result, the iterations produce a sequence of vectors x_1, x_2,...which are required to converge to the minimum point x^*. Hence, a descending condition given by

$$F(x_{k+1}) < F(x_k) \tag{2.3}$$

must be satisfied in order to ensure convergence towards a minimum point x^*. If the condition (2.3) is not satisfied, it is possible to lead to an undesirable convergence towards a maximum point or even a saddle point (i.e. it is neither a minimum point nor a maximum point). In cases where an objective function has multiple minimum points, the convergence of the numerical iterations depend on the initial point x_0. Moreover, convergence towards the nearest minimum point is not guaranteed (Eriksson, 1996; Madsen et al., 2004).

2.2. Long-term versus short-term optimal iterations in NLS

The critical issue in numerical methods is that numerical method in optimization is a dynamic process where long-term optimality rather than short-term optimality is important. In general, there is a major difference between short-term and the long-term optimal iterations in NLS. The short-term iterations are computed by using the so-called exact line search method where a search direction p_k is constructed and a step length α_k is chosen such that the next objective function value is minimized for each iteration (single-stage) in the given direction. In addition, there is a tacit assumption that the sum of single-stage optimal iterations may provide a long-term optimal iteration (Goh and McDonald, 2015).

On the other hand, the long-term iterations are computed such that the net value of the objective function for all iterations of the NLS problem is minimized. According to Goh (2009), in the computation of a numerical solution, we are interested in finding an optimal trajectory, which is obtained by joining an initial guessed point to the minimum point (i.e. optimal solution), in a finite time. However, in practice, the minimum point is normally unavailable and hence there is no practical information available on how this long-term optimal iteration can be constructed. Nevertheless, one can consider reformulating the NLS problem as a sequence of optimization problems.

The difference between the long-term and short-term iterations can be illustrated through a multi-stage network optimization path as shown in Figure 2.1. A similar figure can be found in Goh and McDonald (2015).

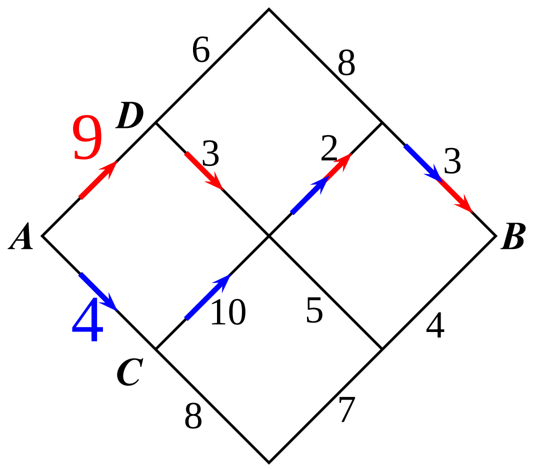


Figure 2.1. Short-term versus long-term iterations in NLS as indicated by the blue and red arrows respectively.

The problem is to find a path from A to B in the network such that the total sum of costs is minimized. Obviously, at the first stage, the single-stage optimal decision is AC with a cost of 4. Upon reaching B, the total path will incur a total cost of $4 + 10 + 2 + 3 = 19$. Conversely, if AD with a first-stage cost of 9 is chosen, the total cost will be $9 + 3 + 2 + 3 = 17$, which is considerably lower than the previous decision. This is because AC is only short-term optimal as compared to AD which is part of a long-term optimal path from A to B.

In brief, we conclude that iterations constructed far away from the optimal solution using an exact line search method may be counterproductive as exact step length may only be short-term optimal (Goh and McDonald, 2015). Nevertheless, exact step length is crucial and important and it has been proven to be so in only two cases:

(1) Newton's method when applied to quadratic function with exact step length equal to 1; and

(2) Conjugate gradient method when applied to quadratic function.

2.3. The approximate linear and quadratic models in NLS

Since numerical methods are used to solve the NLS problem, an approximation to the objective function $F(x)$ in (2.1) is normally used. This is done by applying the Taylor's theorem to expand $F(x)$ so that an approximate model of $F(x)$ can be obtained. The process of expanding the function using the Taylor's theorem is known as Taylor expansion. Since the Taylor's theorem is central to our analysis throughout the thesis, it is stated in the following theorem (Nocedal and Wright, 2006).

Theorem 2.1 (Taylor's theorem)**.** *Suppose $F : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and that $F \in \mathbb{R}^n$. Then*

$$F(x+p) = F(x) + \nabla F(x+tp)^T p$$

for some $t \in (0,1)$. Furthermore, if $F(x)$ is twice continuously differentiable, then

$$\nabla F(x+p) = \nabla F(x) + \int_0^1 \nabla^2 F(x+tp)p \, dt$$

and that

$$F(x+p) = F(x) + \nabla F(x)^T p + \frac{1}{2} p^T \nabla^2 F(x+tp)p$$

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

By applying the Taylor expansion about x, the objective function $F(x)$ can be approximated by either a linear or a quadratic model; i.e.

$$F(x+p) \approx \underbrace{\overbrace{F(x) + g(x)^T p}^{\text{linear model}} + \frac{1}{2} p^T H(x) p}_{\text{quadratic model}} \tag{2.4}$$

where $g(x) = \nabla F(x)$ is the gradient and $H(x) = \nabla^2 F(x)$ is the Hessian matrix of $F(x)$. Since the formulas of $g(x)$ and $H(x)$ are particularly important in the description and formulation of numerical methods for NLS problem, these formulas are derived as follows.

Consider the residual vector function $r : \mathbb{R}^n \to \mathbb{R}^m$ with $m > n$. If $r(x)$ is twice continuously differentiable, then its Taylor expansion can be written as

$$r(x + p) = r(x) + J(x)p + O(\|p\|^2), \tag{2.5}$$

where $J \in \mathbb{R}^{m \times n}$ is the Jacobian of $r(x)$ which consists of first partial derivatives of the function components

$$[J(x)]_{ij} = \frac{\partial r_i}{\partial x_j}(x); \quad \text{where} \quad i = 1, \dots, m, \quad j = 1, \dots, n.$$

Note that the ith row of $J(x)$ equals the transpose of the gradient of $r_i(x)$. Differentiating equation (2.1) yields

$$\begin{aligned} \frac{\partial F}{\partial x_j}(x) &= \sum_{i=1}^{m} r_i(x) \frac{\partial r_i}{\partial x_j}(x); \\ \frac{\partial^2 F}{\partial x_j \partial x_k}(x) &= \sum_{i=1}^{m} \left(\frac{\partial r_i}{\partial x_j}(x) \frac{\partial r_i}{\partial x_k}(x) + r_i(x) \frac{\partial^2 r_i}{\partial x_j \partial x_k}(x) \right); \end{aligned}$$

and it follows immediately that the gradient $g(x)$ and Hessian matrix $H(x)$ of $F(x)$ can be written in vector form as

$$\begin{aligned} g(x) &= \nabla F(x) = J(x)^T r(x) \\ H(x) &= \nabla^2 F(x) = J(x)^T J(x) + S(x) \end{aligned} \tag{2.6}$$

where $S(x) = \sum_{i=1}^{m} r_i(x) \nabla^2 r_i(x)$ denotes the tensor terms of the Hessian matrix $H(x)$ which consist of second order partial derivatives.

2.3.1. The truncated Hessian matrix in NLS

In the second equation of (2.6), notice that the Hessian matrix $H(x)$ is a symmetric $n \times n$ matrix which consists of the sum of product of Jacobian and its transpose (that is dependent on the first partial derivatives of $F(x)$; i.e. the linear part) and the term $S(x)$ which denotes the tensor terms (i.e. the nonlinear part). The tensor terms $S(x)$ are obtained from the sum of products between $r_i(x)$ and $\nabla^2 r_i(x)$, which have m-components and $n \times n$-components, respectively. As a result, the computation of $S(x)$

requires an expensive evaluation of mn^2 derivatives. For instance, if one would like to fit a mathematical model with $n = 3$ parameters to $m = 50$ data points, this implies that the computation of $S(x)$ requires the evaluation of $50 \times 3^2 = 450$ derivatives. Due to the expensive evaluation of $S(x)$, most numerical methods used to solve the NLS problem neglect the term $S(x)$ completely in their algorithms without any justification. Thus, the resulting Hessian matrix, which is obtained by setting $S(x) = 0$, is called the truncated Hessian matrix given by

$$H_T(x) = J(x)^T J(x). \tag{2.7}$$

Nonetheless, such a truncation of the Hessian matrix needs to be justified mathematically. Generally, there are two situations where the term $S(x)$ should not be neglected. The first situation occurs when the number of data points m or the number of parameters n or both are large. This is because the total number of derivatives mn^2 computed in $S(x)$ is large. Neglecting $S(x)$ completely implies that too many terms are thrown away. The other situation happens when the residuals $r_i(x)$ are large. This means that the term $S(x)$ is too significant to be ignored. As a consequence, when truncated Hessian matrix is used in numerical algorithms under these two situations, it is not surprising that the numerical method fails to work when solving the NLS problem. In situations where the numerical method works, the number of iterations require for convergence are relatively high and thus a longer amount of time is needed to compute the iterations (See Chapter 4). In short, the truncated Hessian matrix should not be used in numerical algorithms without justification.

2.3.2. The optimality conditions for NLS

Consider the change of function value $F(x)$ along the half line starting at x and with direction p. Then, by applying the the Taylor expansion, we have

$$F(x + \alpha p) = F(x) + \alpha g(x)^T p + O(\alpha^2)$$

and hence p is a descent direction for $F(x)$ if $g(x)^T p = p^T g(x) < 0$ since the linear term will dominate for sufficiently small α. In most numerical methods, this descent direction is computed for every iteration until an optimal solution is found. Specifically, if $F(x)$ is twice continuously differentiable, the nature of this optimal solution can be determined by examining just the gradient $g(x) = \nabla F(x)$ and the Hessian matrix $H(x) = \nabla^2 F(x)$ of $F(x)$ at the optimal solution. The optimal solution can be a minimum point, a maximum point or a saddle point. Therefore, it is important to state the optimality conditions for the NLS problem. These conditions are stated in the following theorems (Madsen et al., 2004; Nocedal and Wright, 2006).

Theorem 2.2 (First-order necessary condition)**.** *Suppose x^* is a local minimum point of $F(x)$ and $F(x)$ is continuously differentiable in an open neighbourhood of x^*, then $\nabla F(x^*)$=0.*

The point x^* is called a stationary point if $\nabla F(x^*) = 0$. According to Theorem (2.2), any local minimum point must be a stationary point.

Theorem 2.3 (Second-order necessary condition)**.** *Suppose x^* is a local minimum point of $F(x)$ and $\nabla^2 F(x)$ exists and is continuous in an open neighbourhood of x^*. Then, $\nabla F(x^*)$=0 and $\nabla^2 F(x^*)$ is positive semidefinite.*

Theorem 2.4 (Second-order sufficient condition)**.** *Suppose $\nabla^2 F(x)$ is continuous in an open neighbourhood of x^* and that $\nabla F(x^*) = 0$ and $\nabla^2 F(x^*)$ is positive definite. Then, x^* is a strict local minimum point of $F(x)$.*

From Theorem (2.2)–(2.4), the optimality conditions now take the special form
First order necessary condition: The gradient of $F(x^*)$ must be zero, i.e.

$$g(x^*) = \nabla F(x^*) = J(x^*)^T r(x^*) = 0;$$

Second order necessary condition: The Hessian matrix of $F(x^*)$; i.e.

$$H(x^*) = \nabla^2 F(x^*) = J(x^*)^T J(x^*) + S(x^*) \text{ is positive semidefinite.}$$

Second order sufficient condition: The Hessian matrix of $F(x^*)$; i.e.

$$H(x^*) = \nabla^2 F(x^*) = J(x^*)^T J(x^*) + S(x^*) \text{ is positive definite.}$$

These optimality conditions are used to check whether the optimal solution obtained from a numerical method is indeed the minimum point x^* of the NLS objective function $F(x)$. In other words, these conditions govern the properties of the optimal solution of $F(x)$. Nonetheless, in cases where these conditions are violated, they may provide some helpful information to improve the current estimate of the solution (Nocedal and Wright, 2006).

In addition, the second-order necessary condition stated in Theorem (2.3) is a weaker condition compared to the sufficient condition given in Theorem (2.4) since a strict local minimum point is guaranteed in the latter theorem. However, the second-order sufficient condition is not necessary since a point x^* can be a strict local minimum point while it fails to satisfy the sufficient condition. For instance, a function $F(x) = x^6$ has a strict local minimum point at $x^* = 0$ but its second derivative at x^* is zero (and so is not positive definite). In this case, higher order terms in the Taylor expansion of $F(x)$ are required to determine its nature.

2.4. Types of convergence rates

Before moving on to the discussion on numerical methods for NLS problem, it is worthwhile to state the different types of convergence rates that the numerical methods could take in the iterative process. When the initial iterate (or point) x_0 starts close to a local minimum point x^* at which the sufficient condition stated in Theorem (2.4) is satisfied, we said that a local convergence is achieved by the numerical method. Nonetheless, the convergence rate of a numerical method is a limiting concept which investigate how a trajectory generated by the numerical method converges near x^*.

The following definition distinguish between the different types of convergence rates (Dennis et al., 1981; Kelly, 1999; Madsen et al., 2004).

Definition 2.2 (Type of convergence rates)**.** Let $e_k = x_k - x^*$ be the current error of the iterative process. Then, the different types of convergence rates are:

Linear convergence: $\|e_{k+1}\| \leqslant \eta \|e_k\|$ when $\|e_k\|$ is small and $0 < \eta < 1$;

Superlinear convergence: $\lim_{k\to\infty} \frac{\|e_{k+1}\|}{\|e_k\|} = 0$;

Superlinear convergence with q-order $\alpha > 1$: $\|e_{k+1}\| \leqslant \eta\|e_k\|^\alpha$ and $\eta > 0$;

Quadratic convergence: $\|e_{k+1}\| \leqslant \eta\|e_k\|^2$ and $\eta > 0$.

2.5. Line search and trust region numerical methods for NLS

Over the decades, various numerical methods have been proposed and modified to solve NLS problem. Most numerical methods for NLS problem used the approximate models in (2.4) to construct the required iterative step (2.2) (Han et al., 2005). Basically, these methods are classified into the first order methods and the second order methods. First order methods utilize the first derivative or gradient of the objective function in its computations, e.g. the steepest descent (SD) method. On the other hand, any method which uses the Hessian matrix of the objective function or models or estimates of the Hessian matrix are classified as the second order methods, e.g. the Newton's method, the Gauss-Newton (GN) method and the Levenberg-Marquardt (LM) method. Furthermore, all these numerical methods can also be classified into line search methods or trust region methods (Yuan, 1999; Nocedal and Wright, 2006). Among these four methods, the LM method is the only trust region method. In this section, some issues and drawbacks of these methods are identified and discussed.

2.5.1. The steepest descent (SD) method for NLS

The steepest descent (SD) or the gradient method, which was proposed by Cauchy in 1827, represents one of the oldest line search methods used in optimization problems. It uses the approximate linear model in (2.4) to construct its iterations with the search direction p_k in the iterative step (2.2) evaluated as the negative gradient of the objective function $F(x)$ at the current point; i.e.

$$p_k^{SD} = -g(x_k) = -J(x_k)^T r(x_k) \tag{2.8}$$

in order to find the minimum point x^*. As a result, from (2.2) and (2.8), the SD iterative equation takes the form

$$x_{k+1} = x_k - \alpha_k J(x_k)^T r(x_k)$$

which results in search directions that are orthogonal to the level sets of $F(x)$ at the current iterate points (See Figure 2.2 below).

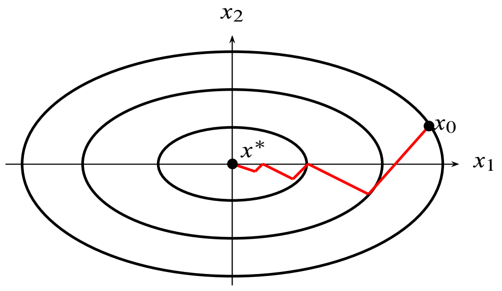


Figure 2.2. The zigzag behaviours of the trajectory shown in red from the initial point $x_0 = (x_{10}, x_{20})$ to the minimum point $x^* = (x_1^*, x_2^*)$ with the SD direction.

According to Madsen et al. (2004), among all the directions we could move from x_k, this search direction p_k^{SD} is considered to decrease the objective function most rapidly at a point for small displacements. In other words, the negative gradient direction is a local optimal search direction which provides a maximum descent search direction. Despite this advantage, Nocedal and Wright (2006) further stated that the decrease in the objective function is only guaranteed when the step length α_k is made sufficiently or arbitrarily small. This step length α_k, which is allowed to change at every iteration, can be found by a backtracking line search method (see Algorithm (1)).

The SD method may perform well in the initial stage of the iterative process for most problems; i.e. when x is far away from the solution x^*. In addition, the cost of computation for this method is relatively low as it only requires the evaluation of the first derivatives. Moreover, SD method always generate a descent direction and is globally convergent provided that all the level sets of the objective function are properly nested in a bounded region containing x^*.

However, due to the orthogonality of the search directions, SD method creates iterations that zigzag towards the minimum point x^* (See Figure 2.2). Obviously, this zigzag behaviour is not the optimal and fastest path to reach x^*. Therefore, the convergence rate which is generally linear, is excruciatingly slow. For instance, Simionescu and Mehrubeoglu (2012) shown that the Rosenbrock function converges only after more than 1000 iterations. Furthermore, Goh et al. (2008) illustrated that the convergence of this function depends on the choice of the step length α_k. This shows that the SD is not a robust numerical method and can be problematic if the chosen step length is inappropriate.

2.5.2. The Newton's method for NLS

Newton's method, which is also known as the Newton-Raphson's method (named after Isaac Newton and Joseph Raphson), is a line search method derived from the condition $\nabla F(x^*) = 0$ where x^* is the minimum point of $F(x)$. By differentiating the quadratic model in (2.4), one obtains

$$\nabla F(x + p) \approx \nabla F(x) + H(x)p. \tag{2.9}$$

Since $\nabla F(x + p) = 0$ and from (2.6), the well-known Newton's step is given by

$$H(x_k)p_k^N = -g(x_k) \tag{2.10}$$

$$\Rightarrow \quad p_k^N = -[J(x_k)^T J(x_k) + S(x_k)]^{-1} g(x_k). \tag{2.11}$$

It follows from (2.2) and (2.11) that the Newton's iterative equation takes the form

$$x_{k+1} = x_k - \alpha_k [J(x_k)^T J(x_k) + S(x_k)]^{-1} g(x_k) \tag{2.12}$$

where α_k is found by backtracking line search method. Equation (2.12) is sometimes referred to as the damped Newton's method where the presence of a damping parameter helps to control the step length so that a sufficient decrease in the objective function can be ensured. In this case, the step length parameter α acts as the damping parameter on its own (Hansen et al., 2013). When $\alpha_k = 1$ for all k, equation (2.12) gives the Newton's method with exact step length for a quadratic objective function. It is important to note that the use of Newton's method with exact step length is only valid for iterations that are close to x^* (for sufficiently small gradients) since the objective function is approximately quadratic. However, in the NLS literature, it is seen that $\alpha = 1$ for all iterations regardless of the initial point (Madsen et al., 2004; Hansen et al., 2013).

Suppose that $H(x)$ is positive definite, then it is nonsingular so that equation (2.10) has a unique solution. It is obvious that p_k^N is a descent direction of $F(x)$ by multiplying ${p_k^N}^T$ on both sides of equation (2.10).

Newton's method exhibits quadratic convergence rate and hence it converges more rapidly especially at the final stage of the iterations, where x is close to x^* for sufficiently small gradient. However, its cost per iteration is usually high since it requires the expensive evaluation of $S(x_k)$ of the Hessian matrix $H(x_k)$ where the computation of second derivatives are needed. Furthermore, the computation of the inverse Hessian matrix is also required at every iteration. This is extremely expensive when solving large-scale NLS problem. Nevertheless, the Newton's method is a unique method for quadratic function since it is able to converge to the minimum point in just one step. Therefore, all numerical methods should merge with the Newton's method near the minimum point in order to achieve fast convergence rate (Goh, 2009). Moreover, notice that when $H(x_k) = I$ in equation (2.10), the Newton's method reduces to the SD method.

On the other hand, the Hessian matrix can be singular at the starting point. For instance, the famous Rosenbrock function is singular along the curve $x_2 = x_1^2 + \frac{1}{200}$ and hence, the Newton's method cannot be used to solve it at the starting point $(-1, \frac{201}{200})$ (Goh, 2009).

Furthermore, the Hessian matrix may be indefinite further away from the solution. For example, the Hessian matrix of the Powell's function is indefinite at the starting point $(0, 0)$ and so the Newton's method cannot be applied to solve it (Fletcher, 1987). In order to avoid this indefiniteness, one method is to add a positive term $\lambda_k I$ to $H(x_k)$ to get $H(x_k) + \lambda_k I$ which is always positive definite for sufficiently large $\lambda_k > 0$ (Dennis and Schnabel, 1983). A similar method which utilized this strategy is the Levenberg-Marquardt (LM) method which is discussed below.

2.5.3. The Gauss-Newton (GN) method for NLS

The Gauss-Newton (GN) method is a line search method derived from the linearization of the components of the residual vector function $r(x)$ (i.e. a linear model of $r(x)$) in the neighbourhood of x. From the Taylor expansion (2.5), it follows that

$$r(x+p) \simeq \ell(h) \equiv r(x) + J(x)p \tag{2.13}$$

for sufficiently small $\|p\|$. Substituting (2.13) into definition (2.1) of $F(x)$, we have

$$\begin{aligned} F(x+p) \simeq \mathcal{L}(p) &\equiv \frac{1}{2}\ell(p)^T\ell(p) \\ &= \frac{1}{2}r(x)^Tr(x) + p^TJ(x)^Tr(x) + \frac{1}{2}p^TJ(x)^TJ(x)p \\ &= F(x) + p^TJ(x)^Tr(x) + \frac{1}{2}p^TJ(x)^TJ(x)p \end{aligned} \tag{2.14}$$

where $\mathcal{L}(p)$ represents the linear model of $F(x)$. From equation (2.14), one can easily obtain the gradient and Hessian matrix of $\mathcal{L}(p)$ given by

$$\nabla\mathcal{L}(p) = J(x)^Tr(x) + J(x)^TJ(x)p \quad \text{and} \quad \nabla^2\mathcal{L}(p) = J(x)^TJ(x) \tag{2.15}$$

respectively. Notice that $\nabla^2\mathcal{L}(p)$ is a symmetric matrix that is independent of p. It follows that if $J(x)$ has full column rank; i.e. if the columns are linearly independent, then $\nabla^2\mathcal{L}(p)$ is positive definite and hence $\mathcal{L}(p)$ has a unique minimum point. Letting $\nabla\mathcal{L}(p)$=0 in the first equation of (2.15), this minimum point is obtained by solving

$$[J(x_k)^TJ(x_k)]p_k^{GN} = -J(x_k)^Tr(x_k) \tag{2.16}$$

$$[J(x_k)^TJ(x_k)]p_k^{GN} = -g(x_k)\ . \tag{2.17}$$

Again, if $J(x)$ has full column rank, equation (2.17) is actually the normal equations for the linear least squares problem (Hansen et al., 2013)

$$\min_{p_k\in\mathbb{R}^n} \|J(x_k)p_k + r(x_k)\|^2\ . \tag{2.18}$$

From equation (2.16) is easy to check that p_k^{GN} is a descent step. For instance,

$$p_k^{GN^T} \nabla F(x) = p_k^{GN^T} (J(x)^T r(x)) = -p_k^{GN^T} (J(x)^T J(x)) p_k^{GN} < 0.$$

Substituting equation (2.17) into equation (2.2), the GN iterative equation is given by

$$x_{k+1} = x_k - \alpha_k [J(x_k)^T J(x_k)]^{-1} g(x_k). \tag{2.19}$$

By comparing the Newton's iterative equation (2.12) and the GN iterative equation (2.19), it can be seen that the GN method is a simplification of the Newton's method where linearization of components of the residual vector function $r(x)$ results in the disappearance of the tensor terms $S(x)$ of the Hessian matrix $H(x)$ thus leading to a truncated Hessian matrix $H_T(x)$ (see (2.7)). In short, the GN method can be derived directly from the Newton's method by neglecting $S(x)$ completely in its algorithm.

Similar to the Newton's method, the presence of a step length parameter α_k in the iterative equation (2.19) results in the GN method with line search which is normally referred to as the damped GN method (Hansen et al., 2013) where α_k can be found by the backtracking line search method. For the classical GN method, $\alpha = 1$ is used for all iterative steps (Madsen et al., 2004).

The GN method can exhibit quadratic convergence rate provided that the neglected term $S(x^*)$ is negligible. Otherwise, it may be seen to converge linearly in general. However, if $S(x^*)$ is too large, it may not be locally convergent at all (Dennis et al., 1981; Hansen et al., 2013). The convergence proofs of GN method for NLS problem can be found in the paper by Chen and Li (2005). However, Transtrum and Sethna (2012) pointed out that unless the initial guess is very good, the GN method takes large, uncontrolled steps and will fail to converge.

A drawback of the GN method occurs when an NLS problem turns out to have large residuals at a current point. In this case, the truncated Hessian matrix is not a valid approximation of $H(x_k)$ and thus the GN method may fail to work. Another drawback of the GN method is that the matrix product of the Jacobians $J(x_k)^T J(x_k)$ which appears in the GN step (2.17) can be singular at the current iteration or at the solution. This implies that the GN method cannot be used. Therefore, the GN method is not well defined if $J(x_k)$ does not have full column rank (Dennis and Schnabel, 1983).

The main difficulty encountered by the GN method arises in the case when $\|p_k^{GN}\|$ is too large (which occurs when $J(x_k)$ is rank deficient) so that there is only a negligible reduction of $F(x)$ (Yuan, 1999; Hansen et al., 2013). Under such situation, it is a common practice to add an inequality constraint to the linear least square problem (2.18) so that the step $\|p_k^{GN}\|$ is now bounded by some constant. This leads to a trust region based numerical method for NLS, called the Levenberg-Marquardt (LM) method, which improves the quality of the step.

2.5.4. The Levenberg-Marquardt (LM) method for NLS

The Levenberg Marquardt (LM) method, which is also known as the damped least squares (DLS) method, was first published by Kenneth Levenberg in 1944 and later by Donald W. Marquardt in 1963 (Levenberg, 1944; Marquardt, 1963). This method is derived from the GN method where a positive Lagrange (or damping) parameter μ is introduced into the GN step (2.19) to give the LM step

$$p_k^{LM} = -[J(x_k)^T J(x_k) + \mu_k I]^{-1} g(x_k) \quad \text{with} \quad \mu_k > 0 \tag{2.20}$$

so that the LM iteration is now given by

$$x_{k+1} = x_k - [J(x_k)^T J(x_k) + \mu_k I]^{-1} g(x_k). \tag{2.21}$$

The effects of the positive Lagrange parameter μ_k are (Transtrum and Sethna, 2012):

(1) For sufficiently large $\mu_k > 0$, it ensures that the matrix $J(x_k)^T J(x_k) + \mu_k I$ is always positive definite and hence overcomes the problem when $J(x_k)^T J(x_k)$ is an ill-conditioned (or positive semidefinite) matrix.

(2) It ensures that the step p_k^{LM} is a descent step since

$$P_k^{LM^T} g(x_k) = -P_k^{LM^T} \left[J(x_k)^T J(x_k) + \mu_k I \right] P_k^{LM} < 0.$$

Hence, the method is well-defined.

(3) For small values of μ_k, we have $p_k^{LM} \simeq p_k^{GN}$, which is a good step in the final stages of the iterative process for the NLS problem with small residuals at the solution.

(4) For large values of μ_k, one obtains $p_k^{LM} \simeq -\frac{1}{\mu_k} g(x_k)$ which represents a short step in the SD direction. This is a good step in the initial stages of the iteration.

Hence, these show that the Lagrange parameter μ_k influences both the direction and size of the step (Madsen et al., 2004). Thus, this leads to a method without a specific line search since its role is taken over by the Lagrange parameter μ_k. In other words, the LM method is an approximate combination of the SD and the GN methods (Lourakis, 2005; Gavin, 2015).

Note that the LM step p_k^{LM} given by (2.20) is also a solution of the constrained minimization problem (Nocedal and Wright, 2006)

$$\min_{p_k \in \mathbb{R}^n} \|J(x_k)p_k + r(x_k)\|^2 \tag{2.22}$$

$$\text{s.t. } \|p_k\| \leqslant \triangle_k. \tag{2.23}$$

Notice that an equality constraint (2.23) is added into the linear least squares equation (2.18) for the GN iterations to obtain the above constrained minimization problem. This is done to prevent $\|p_k^{GN}\|$ being too large by bounding it with some constants $\triangle > 0$.

In the minimization problem (2.22)–(2.23), the linear model (2.14) is trusted to accurately represent the objective function $F(x)$ inside a ball of radius $\triangle$ about x_k where $p_k = x - x_k$. Hence, $\triangle > 0$ is called the trust region radius and the ball which is represented by the inequality constraint (2.23) is called the trust region. In addition, the step p_k^{LM} in (2.20) and point x_{k+1} in (2.21) are termed the trial step and the trial point of the constrained minimization problem respectively. After obtaining the trial point x_{k+1}, one must now decide whether to accept the point and/or to vary the Lagrange parameter μ. Normally, the trial point x_{k+1} and the Lagrange parameter

μ are tested simultaneously in order to determine how well the linear model (2.14) approximates the function $F(x)$ inside the trust region. This is measured by computing an improvement or gain ratio called the ratio test defined as (Madsen et al., 2004; Kelly, 1999; Hansen et al., 2013)

$$\Gamma_k = \frac{\text{Actual reduction}}{\text{Predicted reduction}} = \frac{F(x_k) - F(x_{k+1})}{\mathcal{L}(0) - \mathcal{L}(p_k^{LM})} \tag{2.24}$$

where the predicted reduction is the reduction in $F(x)$ predicted by the linear model (2.14) computed as follows:

$$\begin{aligned}\mathcal{L}(0) - \mathcal{L}(p_k^{LM}) &= -{p_k^{LM}}^T J(x_k)^T r(x_k) - \frac{1}{2}{p_k^{LM}}^T J(x_k)^T J(x_k) p_k^{LM} \\ &= -\frac{1}{2}{p_k^{LM}}^T \left[2g(x_k) + \left(J(x_k)^T J(x_k) + \mu_k I - \mu_k I\right) p_k^{LM}\right] \\ &= \frac{1}{2}{p_k^{LM}}^T \left[\mu_k p_k^{LM} - g(x_k)\right] > 0\end{aligned}$$

since the terms ${p_k^{LM}}^T \mu_k p_k^{LM}$ and $-{p_k^{LM}}^T g(x_k)$ are both positive.

After the ratio test, three control parameters given by (Kelly, 1999; Hansen et al., 2013)

$$\Gamma_0 \leqslant \Gamma_{low} < \Gamma_{high}$$

are used to determine whether

- the trial point x_{k+1} should be rejected ($\Gamma_k < \Gamma_0$) and/or
- the Lagrange parameter should be increased ($\Gamma_k < \Gamma_{low}$),
- the Lagrange parameter should be decreased ($\Gamma_k > \Gamma_{high}$), or
- left unchanged.

Typical values are $\Gamma_{low} = 0.25$, $\Gamma_{high} = 0.75$ and either $\Gamma_0 = 10^{-4}$ or $\Gamma_0 = \Gamma_{low}$ can be used. The Lagrange parameter μ is increased or decreased by multiplying it with the constants

$$0 < \mu_{down} < 1 < \mu_{up}.$$

Typical values of μ_{down} and μ_{up} are 0.5 and 2 respectively (Kelly, 1999). In addition, a default value μ_0 of the Lagrange parameter is required at the start of the iteration. Whenever the Lagrange parameter $\mu_k < \mu_0$, we set $\mu_k = 0$ so that the LM iteration switches to the fast convergence GN iteration for small-residual problems.

It is interesting to note that in the trust region method for general unconstrained optimization problems, the algorithm for testing the trial point x_{k+1} differs from those described above in that rather than controlling the Lagrange parameter μ_k, the radius of the search region is shrunk or expanded according to the ratio test (Yuan, 1999; Hansen et al., 2013). That is; the radius of the trust region $\triangle_k$ is decreased (increased) if Γ_k is small (large) rather than increasing (decreasing) the Lagrange parameter μ_k. This indicates that the Lagrange parameter μ_k is inversely proportional to the radius of the trust region $\triangle_k$; i.e. $\mu_k \propto \frac{1}{\triangle_k}$.

Similar to the GN method, the LM method can exhibit quadratic convergence rate provided that the neglected term $S(x_k)$ is negligible. Otherwise, it converges linearly. The LM method is more robust than the GN method in the case of an ill-conditioned Jacobian and in many cases it converges to the minimum point x^* even if the starting point is far away from it. For instance, Powell (1975), Osborne (1976), and Moré (1977) have proved the global convergence of several versions of LM algorithm with various sets of assumptions. Yuan (2011) reviewed some recent results of the LM methods and presented some theoretical results on its local convergence. However, according to Transtrum and Sethna (2012), the LM method can exhibit slow convergence, especially when it must navigate a narrow canyon en route to a best fit. Moreover, when the objective function is very flat, the algorithm may easily become lost in parameter space. Thus, several improvements to the LM algorithm are introduced by Transtrum and Sethna (2012) in order to improve both its convergence speed and robustness to initial parameter guesses. However, despite these improvements, the LM method still uses a truncated Hessian matrix in its algorithm and it approaches the slow convergence SD method for large μ_k.

2.6. Conclusion

The nonlinear least squares (NLS) problem is considered as a special class of unconstrained optimization problem. Since numerical methods are used to solve the NLS problem, an approximation to the objective function is normally used. All the existing numerical methods use to solve the NLS problem use either an approximate linear or quadratic model of the objective function.

From the analysis of the multi-stage network optimization path (see Figure 2.1), one can conclude that a long-term optimal iteration to reach the minimum point x^* should always be considered since it is more cost-effective in the long run. However, in the existing numerical methods, most numerical iterations used to solve NLS problem are at best, just short-term optimal. In addition to that, both the GN and the LM methods use a truncated Hessian matrix to compute their iterative steps. The truncated Hessian matrix, which is obtained by ignoring the tensor terms $S(x_k)$ of the Hessian matrix $H(x_k)$ completely, is an inadequate approximation of $H(x_k)$ under two situations. The first occurs when either the number of data points m or the number of parameters n or both are large and the second happens when the residuals at the current iteration are large. As a consequence, it is not surprising that these numerical methods either converge very slowly or fail to work when solving the NLS problem. Thus, the truncated Hessian matrix should not be used in numerical algorithms without justification.

The numerical methods for NLS can be classified into line search methods or trust region methods. Three line search methods are discussed in this chapter; namely the SD method, the Newton's method and the GN method. The LM method is the only trust region numerical method considered in this chapter. All these existing numerical methods have their own strengths and weaknesses when they are applied to solve the NLS problem. The SD method is cost-effective since it only requires the evaluation of the first derivatives. However, its convergence speed is normally very slow. In contrast, the Newton's method has a fast quadratic convergence rate but it is very expensive to compute since it requires the evaluation of the second derivatives. Due to the use of a truncated Hessian matrix, the GN and the LM method work well with quadratic convergence rates provided that the residuals are sufficiently small. However, for large-residual problems, the truncation Hessian matrix is an inadequate approximation of the original Hessian matrix and hence these methods may either converge very slowly or fail to converge. Nonetheless, the LM method is considered to be the most successful numerical approach for NLS problem due to its robustness in handling the ill-conditioned Jacobian by introducing a positive Lagrange parameter into its algorithm.

In this thesis, a new systematic numerical method is introduced to solve the NLS problem. This method, called the approximate greatest descent (AGD) method, is discussed in the next chapter.

CHAPTER 3

LYAPUNOV-BASED NUMERICAL METHODS FOR SOLVING NLS PROBLEM

In this chapter, some new approaches used to solve the NLS problem are proposed. An overview of the Zoutendijk theorem and the Lyapunov function theorem as convergence analyses of a numerical method are first presented. Numerical differentiation is introduced to compute the numerical derivatives needed in the iterative procedures. The implementation of numerical differentiation avoids the needs to compute the derivatives of a function analytically and hence it save a tremendous amount of time and effort. The use of the stiff ODE method for plotting the level sets of an NLS objective function is also discussed. Following that, the existing numerical methods discussed in Chapter 2 are modified by implementing the Lyapunov function theorem and numerical differentiation in their algorithms. Furthermore, the AGD and the AGDN methods are introduced as new numerical approaches to solve the NLS problem. All the numerical methods discussed in this chapter follow the convergence analysis of the Lyapunov function theorem so that monotonic decreasing property of the objective function of the NLS problem can be achieved to guarantee convergence towards a minimum point.

3.1. Convergence analysis of numerical methods for NLS

The convergence proof of a numerical method in an unconstrained optimization problem plays a crucial part in the construction of a good numerical algorithm (Goh, 2010). According to Nocedal and Wright (2006), the challenge lies in designing an algorithm which guarantees good global convergence and a rapid rate of convergence. In this section, we discuss two types of convergence analyses that are used to study the convergence of the numerical methods for solving NLS problem. The first convergence analysis, due to Zoutendijk, is used to study the convergence of line search numerical methods. On the other hand, the second convergence analysis, called the Lyapunov function theorem, can be implemented into both line search methods or trust region methods.

3.1.1. Zoutendijk theorem as convergence analysis

In order to establish the convergence of the numerical method for computing the minimum point of an optimization problem, the Zoutendijk theorem is normally used as a set of prototype conditions (Goh et al., 2014). Since the Zoutendijk theorem are used in line search numerical methods, various line search termination conditions are used to establish its convergence proof by ensuring a sufficient decrease in the objective function value. These includes the Wolfe conditions which are stated and briefly explained below (Wolfe, 1969; 1971; Nocedal and Wright, 2006; Hansen et al., 2013).

The Wolfe conditions is a collection of the Armijo and the curvature conditions stated as follows:

$$\begin{aligned} \text{Armijo condition:} \quad & F(x_k + \alpha_k p_k) \leqslant F(x_k) + c_1 \alpha_k \nabla F(x_k)^T p_k, \\ \text{Curvature condition:} \quad & \nabla F(x_k + \alpha_k p_k)^T p_k \geqslant c_2 \nabla F(x_k)^T p_k, \end{aligned} \tag{3.1}$$

where $0 < c_1 < c_2 < 1$. The Armijo condition ensures that the reduction in the objective function $F(x_k)$ is proportional to both the step length α_k and the directional derivative $\nabla F(x_k)^T p_k$. On the other hand, the curvature condition ensures that the slope of $F(x_k + \alpha_k p_k)$ at α_k is greater than c_2 times the initial slope $\nabla F(x_k)^T p_k$.

The next theorem states the Zoutendijk theorem for convergence analysis of numerical methods.

Theorem 3.1 (Zoutendijk theorem)**.** *Suppose the iterative equation* (2.2) *holds such that α_k satisfies the Wolfe conditions* (3.1)*. If $F(x)$ is bounded below in $\mathbb{R}^n$ and that it is continuously differentiable in an open set Ψ containing the level set $\Pi(x, x_0) = \{x | F(x) \leqslant F(x_0)\}$ with its gradient $\nabla F(x)$ satisfying the Lipschitz conditions in Ψ; that is, there exists a positive constant $\varphi > 0$ such that*

$$\|\nabla F(\hat{x}) - \nabla F(x)\| \leqslant \varphi \|\hat{x} - x\|, \quad \forall \quad x, \hat{x} \in \Psi. \tag{3.2}$$

Then,

$$\sum_{k \geqslant 0} \cos^2 \theta_k \|\nabla F(x_k)\|^2 < \infty \tag{3.3}$$

where θ_k is the angle between the search direction p_k and the steepest descent direction $-\nabla F(x_k)$.

For convenience of the reader, the proof of Theorem (3.1), which can be found in Nocedal and Wright (2006) and Goh et al. (2014), are provided as shown.

Proof. From the iterative equation (2.2) and the curvature condition in (3.1), one can obtain

$$[\nabla F(x_{k+1}) - \nabla F(x_k)]^T p_k \geqslant (c_2 - 1) \nabla F(x_k)^T p_k,$$

and the Lipschitz condition (3.2) gives

$$[\nabla F(x_{k+1}) - \nabla F(x_k)]^T p_k \leqslant \alpha_k \varphi \|p_k\|^2.$$

Then, by combining these two inequalities yield

$$\alpha_k \geqslant \frac{c_2 - 1}{\varphi} \frac{\nabla F(x_k)^T p_k}{\|p_k\|^2}.$$

Substituting this into the Armijo condition in (3.1) gives

$$F(x_{k+1}) \leqslant F(x_k) - \frac{c_1(1-c_2)}{\varphi}\left(\frac{\nabla F(x_k)^T p_k}{\|p_k\|}\right)^2.$$

Now, by considering the angle θ_k between the search direction p_k and the steepest descent direction $-\nabla F(x_k)$ defined by

$$\cos\theta_k = \frac{-\nabla F(x_k)^T p_k}{\|\nabla F(x_k)\|\,\|p_k\|},$$

this inequality can be further simplified to give

$$F(x_{k+1}) \leqslant F(x_k) - c\cos^2\theta_k\|\nabla f(x_k)\|^2,$$

where $c = \frac{c_1(1-c_2)}{\varphi}$. Then, by summing this inequality over all indices less than or equal to k yields

$$F(x_{k+1}) \leqslant F(x_0) - c\sum_{j=0}^{k}\cos^2\theta_j\|\nabla F(x_j)\|^2. \tag{3.4}$$

Since $F(x)$ is bounded below, the terms $F(x_0) - F(x_{k+1})$ must be less than some positive constant for all k. It follows that

$$\sum_{k=0}^{\infty}\cos^2\theta_k\|\nabla F(x_k)\|^2 < \infty$$

by taking limits in inequality (3.4). This completes the proof. ❏

The inequality (3.3), which is termed the Zoutendijk condition, implies that

$$cos^2\theta_k\|\nabla F(x_k)\|^2 \to 0. \tag{3.5}$$

If the search direction p_k is not orthogonal with the steepest descent direction $-\nabla F(x_k)$ such that the angle $0 \leqslant \theta_k < 90°$, then there exists a positive constant ξ such that

$$\cos\theta_k \geqslant \xi > 0 \quad \forall \quad k.$$

It follows from (3.5) that

$$\lim_{k\to\infty}\|\nabla F(x_k)\| = 0. \tag{3.6}$$

It is interesting to note that $\theta_k = 0$ occurs in the steepest descent (SD) method where the search direction p_k is parallel to the negative gradient of $F(x)$. In this case, $\cos\theta_k = 1$ for all k and hence (3.6) is immediately satisfied if the method uses a line search which satisfy the Wolfe conditions (Nocedal and Wright, 2006).

The condition (3.3) or (3.6) represents the total trajectory from an initial point x_0. As a result, there is no practical way to predict the outcome of the numerical method if there are numerical errors in the initial state vector x_0 or the current vector x_k as the numerical method progresses. From the perspective of control system theory, this situation is regarded an open-loop control policy where the outcome could be sensitive to numerical errors in the state variable x during the iterative process (Goh et al., 2014). Hence, this suggests that a small variation in the initial state vector x_0 can produce a completely different outcome.

Meanwhile, the limit (3.6) only guarantees the convergence of a trajectory from any initial guessed point x_0 to a stationary point (Nocedal and Wright, 2006). As a consequence, the trajectory may converge to a point which is either a maximum point, a minimum point or even a saddle point.

Due to these reasons, the Zoutendijk theorem is not used as the convergence analysis of the numerical methods in this thesis. The convergence analysis is chosen to follow the Lyapunov function theorem which is discussed in the next section.

3.1.2. Lyapunov function theorem as convergence analysis

The Lyapunov function theorem proves the convergence of a numerical method in a feedback-type manner where all trajectories converge to a minimum point from any initial point provided that the objective function has properly nested level sets globally or in a finite sublevel set containing the minimum point. This is illustrated in Figure 3.1 for the Barbashin and Krasovskii (1952) function (Goh et al., 2014).

According to Goh et al. (2014), since the objective function is monotonic decreasing everywhere, the Zoutendijk theorem can be applied globally. In Figure 3.1, observe that the objective function has properly nested level sets only in the sublevel set

$\{x|F(x) \leqslant a < 1\}$ where a is a positive constant (Goh et al., 2014). Hence, the Lyapunov function theorem ensures convergence to the minimum point $(0,0)$ for any initial point in this sublevel set. Furthermore, all trajectories with initial point $(0, x_2)$ with $x_2 \geqslant 2.61$ converge to $(\infty, 0)$ instead of the minimum point $(0,0)$. The trajectories PQM from $(0, 2.6)$ and RST from $(0, 2.61)$ show sensitivities depending on the initial points (Goh et al., 2014) and thus feedback control is important.

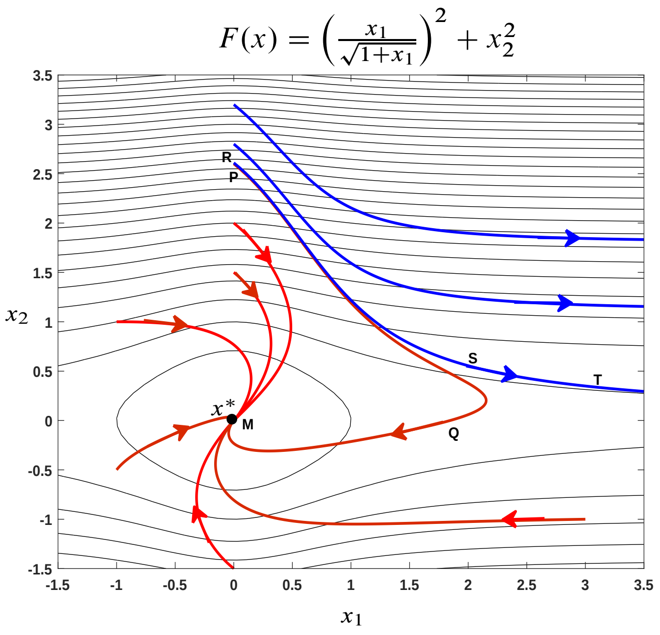


Figure 3.1. The sensitivity of trajectories to small changes in initial conditions which shows the importance of feedback-type control analysis (Goh et al., 2014).

Lyapunov function was first developed in 1892 by a Russian mathematician A.M. Lyapunov (Parks, 1992) and later introduced to the US by LaSalle, Kalman and Bertram in the late 1950's. Since then, it has become a vital tool in the analysis of stability for nonlinear dynamical systems prescribed by systems of differential equations, difference equations and functional equations (Kalman and Bertram, 1960a; 1960b; LaSalle, 1964; Ortega, 1973; LaSalle, 1976). The book by Vincent and Grantham (1997) describes how Lyapunov function ideas can be incorporated when differential equations are used to compute the minimum point of a function.

The following theorem states the Lyapunov function theorem for discrete-time system (Kalman and Bertram, 1960b; LaSalle, 1964; 1976; Goh, 2010; Leong and Goh, 2013; Goh et al., 2014).

Theorem 3.2 (Lyapunov function theorem)**.** *Consider the following iterative equation*

$$x_{k+1} = f(x_k) \quad \textit{for} \quad k = 0, 1, 2, ...$$

$$x(0) = x_0, \quad x^* = f(x^*),$$

where $f(x)$ is a vector of continuous functions. Then, x^* is globally convergent if

(i) *$V(x)$ is a continuous positive definite scalar function with $V(x) > 0$ for $x \neq x^*$ and $V(x^*) = 0$;*

(ii) *All level sets of $V(x)$ are properly nested (i.e. they are topologically equivalent to concentric spherical surfaces); and*

(iii) *$\Delta V(x_k) = V[f(x_k)] - V(x_k) < 0$ for $x_k \neq x^*$ and $\Delta V(x^*) = 0$.*

The proof of this theorem can be found in Goh (2010) and Goh et al. (2014). Note that a level set of $V(x)$ is defined by $\Pi = \{x | V(x) = C\}$ where $C > 0$.

Corollary 3.1. *Suppose that the conditions stated in Theorem* (3.2) *are satisfied only in a finite sublevel set $\Omega(x, x^*, K) = \{x | 0 \leqslant V(x) \leqslant K\}$ where $K > 0$. Then, convergence is only assured in Ω.*

Here, if K is a large positive constant, then it defines a large sublevel set of $V(x)$ and vice-versa.

The paper by Ortega (1973) reviewed several connections between the concept of stability of a discrete-time system, the convergence of iterative equations and Lyapunov function in stability analysis. In other words, iterative equation in numerical method can be viewed as nonlinear difference equation in discrete-time dynamical system.

In general, Lyapunov function cannot be determined easily since there is no systematic way to construct it. However, in NLS, the Lyapunov function can be readily determined from its objective function $F(x)$ (Goh et al., 2014) as shown in the following lemma.

Lemma 3.1 (Lyapunov convergence analysis)**.** *Suppose $F(x)$ has properly nested level sets globally or in a finite sublevel set $\Gamma(x, x^*, L) = \{x | 0 \leqslant F(x) \leqslant L\}$ where $L > 0$ and let $V(x) = F(x) - F(x^*) > 0$. If*

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

then x^* is a minimum point of $F(x)$ and $V(x)$ is a Lyapunov function of $F(x)$ satisfying Theorem (3.2).

In other words, the Lyapunov function theorem requires only a sufficient decrease of the objective function $F(x)$ to ensure convergence towards the minimum point x^* provided that the objective function has properly nested level sets. It is important to note that condition (3.7) is a crucial step which must be computed at every iteration of a Lyapunov-based numerical method even if the Lyapunov function is not stated explicitly in its algorithm. This numerical computation is done to guarantee that the Lyapunov function theorem is satisfied at every iteration so that convergence is assured. If the condition (3.7) is not satisfied, a backtracking line search is performed until a sufficient decrease of $F(x)$ is obtained. This is achieved by using a contraction factor d with $d \in (0, 1)$ to reduce the step length α_k whenever the change in $F(x)$ is positive. The following pseudocode describes the iterative procedures for backtracking line search via Lyapunov function theorem.

Algorithm 1: Backtracking line search via Lyapunov function theorem

```
Initial setting
Choose $\alpha_1 > 0$, $d \in (0,1)$, $N_b = 5000$ (Maximum iteration number)
Compute $\Delta F(x) = F(x_{k+1}) - F(x_k)$
while $\Delta F(x) > 0$ do
    $\alpha_{j+1} = d\alpha_j, \quad j = 1, 2, \dots$ until $N_b$
    Compute $x_{k+1} = x_k + \alpha_k p_k$ (depending on choice of numerical method)
end
```

Notice that the convergence analysis of the Lyapunov function theorem is relatively simpler compared to that of the Zoutendijk theorem since only a sufficient decrease in the objective function is required for each iteration. Throughout this thesis, the Lyapunov function theorem is used as the convergence analysis of the numerical methods for solving NLS problem. In order to implement the Lyapunov function theorem, the line search numerical methods perform a backtracking line search (Algorithm 1) so as to ensure a sufficient decrease in the objective function of the NLS problem at every iteration. Since the Zoutendijk theorem is not used in the convergence analysis, the Wolfe conditions (3.1) are not implemented in line search numerical methods. Obviously, these termination conditions are much more complicated to implement in numerical algorithms compared to the backtracking line search. On the other hand, the sufficient decrease in the objective function required by the Lyapunov function theorem is also ensured implicitly in the trust region LM method through the ratio test.

The use of Lyapunov function theorem as global convergence analysis of the SD method and the Newton's method are discussed in Goh (2010) and Goh and McDonald (2015). In addition, the importance of the Lyapunov function theorem in proving global convergence is also highly emphasized in Goh (1997), Goh (2009), Goh (2011), Leong and Goh (2013), and Goh et al. (2014). The main advantage of the Lyapunov function theorem is its ability to provide a feedback-type analysis and thus the outcomes are robust to small numerical errors in the initial state vector x_0 or the current vector x_k (Goh et al., 2014).

3.2. Numerical differentiation in NLS

In numerical analysis, numerical differentiation describes algorithms for estimating the derivative of a mathematical function or function subroutine using the formulas of the function, the function data or perhaps other knowledge about the function. According to Gill et al. (1983), in numerical optimization, it is not crucial for each component of the gradient to have close-to-maximal accuracy at each iterate as long as the gradient vector has a reasonable level of accuracy.

Numerical differentiation is usually employed when the derivatives of a function cannot be readily determined analytically. This happens especially when dealing with high-dimensional problems where the number of parameters n are large or the number of equations m are large or when the number of derivatives to be computed are large. The implementation of numerical differentiation into numerical algorithms provides a great flexibility where numerical calculations can be performed by just providing the original objective function of the NLS problem. This save a lot of time and effort while preventing any evaluation mistakes done analytically.

As mentioned in Section (2.3.1), when solving an NLS problem, the expensive computation of the tensor terms $S(x)$ of the Hessian matrix has led to the unjustified use of the truncated Hessian matrix $H_T(x)$. Thus, numerical differentiation can be used to calculate the Hessian matrix numerically and hence avoid the need of the truncated Hessian matrix. Similarly, the use of numerical differentiation can also be applied to compute the gradient vector $g(x)$ of $F(x)$ and the Jacobian matrix $J(x)$ of the residual function vector $r(x)$ numerically. Both the GN and the LM algorithms require the computation of the Jacobian matrix as discussed in the previous chapter.

A number of numerical differentiation approaches can be used to compute the derivatives required by numerical algorithms. According to Nocedal and Wright (2006), some of the most important approaches include the finite differencing, the automatic differentiation and the symbolic differentiation. The finite differencing, which is motivated by the Taylor's theorem (2.1), approximates the derivatives of a function from estimating the response to infinitesimal perturbations through examining the differences in function values in response to small (or finite) perturbation in the values of x. The automatic differentiation applies the chain rule to obtain the derivatives by breaking down the computer code for function evaluation into a composition of elementary arithmetic operations. In the symbolic differentiation, new algebraic expressions for each component of gradient is obtained by manipulating the algebraic specification for the function $F(x)$ using symbolic manipulation tools.

Among all, the simplest method to compute the derivatives of a function numerically is to use the finite differencing (Gill et al., 1983). There are three different approaches associated with the finite difference approximations: the forward difference, the central difference and the backward difference approximations (Dennis and Schnabel, 1983). According to Nocedal and Wright (2006), whilst the approximate derivatives obtained using the central difference approximation is more accurate than the forward difference approximation, the former is about twice as expensive in its computation.

Gill et al. (1983) have observed that the relative error bound in forward difference approximation increases as $|\nabla F(x)|$ decreases. In other words, the approximation of $\nabla F(x)$ calculated by the forward difference approximation becomes unreliable when $|\nabla F(x)|$ becomes significantly small. This happens when the current point is near the optimal solution (i.e. when $\nabla F(x) \approx 0$). As a consequence, it is recommended that when the iterations are near the optimal solution, the central difference approximation should be used.

3.2.1. Numerical gradient and Hessian

According to Gill et al. (1983), a forward difference formula can be applied to compute the first derivative of an objective function to solve numerical optimization problems. However, due to the increase in the relative error bound of the forward difference formula as the numerical method progresses, we will switch to the central difference formula when the iterations are close to the solution. This is shown in Algorithm 2 below.

3.2.2. Numerical Jacobian

As discussed earlier, the computation of the Jacobian matrix of the residual functions $r(x)$ of the NLS problem is required for both the GN and LM algorithms due to the presence of the truncated Hessian matrix in their algorithms (2.7). The Jacobian matrix is obtained by computing all the first partial derivatives of $r(x)$ with respect x. Algorithm 3 below provides the steps to construct the numerical Jacobian of the NLS

Algorithm 2: Numerical gradient and Hessian

```
Evaluate $h_i = \sqrt{\epsilon}\max(|x_i|, 1)$, $i = 1, 2, \ldots, n$ ($\epsilon$ is machine precision number)
Evaluate $F(x)$ and compute $g(x)$ using the forward difference formula
    $g_i(x_i, h_i) = \frac{F(x_i + h_i) - F(x_i)}{h_i}$
if $\|g_i\| < 1$ (iterations near solution) then
    Compute $g(x)$ using the central difference formula
        $g_i(x_i, h_i) = \frac{f(x_i + h_i) - f(x_i - h_i)}{2h_i}; \quad h_i = \sqrt[3]{\epsilon}\max(|x_i|, 1)$
    Compute $H(x)$ using the central difference formulas
        (a) For $i = j, \quad H_{ij}(x_{ij}, h_{ij}) = \frac{F(x_i+h_i)+F(x_i-h_i)-2F(x_i)}{h_i^2}$
        (b) For $i \neq j, \quad H_{ij}(x_{ij}, h_{ij}) = \frac{F(x_i+h_i+h_j)-F(x_i+h_i-h_j)-F(x_i-h_i+h_j)+F(x_i-h_i-h_j)}{4h_ih_j}$
    where $h_i = \sqrt[4]{\epsilon}\max(|x_i|, 1)$
else
    Compute $H(x)$ using the forward difference formulas
        (a) For $i = j, \quad H_{ij}(x_{ij}, h_{ij}) = \frac{F(x_i+2h_i)-2F(x_i+h_i)+F(x_i)}{h_ih_j}$
        (b) For $i \neq j, \quad H_{ij}(x_{ij}, h_{ij}) = \frac{F(x_i+h_i+h_j)-F(x_i+h_i)-F(x_i+h_j)+F(x_i)}{h_ih_j}$
    where $h_i = \sqrt[4]{\epsilon}\max(|x_i|, 1)$
end
```

problem.

Algorithm 3: Numerical Jacobian

```
Evaluate $r_i(x)$, $i = 1, 2, \ldots, n$
Use Algorithm 2 to compute $g(x)$
if $\|g_i\| > 1$ (iterations far away from solution) then
    Compute $J(x)$ using the forward difference formula
        $J_i(x_i, h_i) = \frac{r(x_i + h_i) - r(x_i)}{h_i}; \quad h_i = \sqrt{\epsilon}\max(|x_i|, 1)$
else
    Compute $J(x)$ using the central difference formula
        $J_i(x_i, h_i) = \frac{r(x_i + h_i) - r(x_i - h_i)}{2h_i}; \quad h_i = \sqrt[3]{\epsilon}\max(|x_i|, 1)$
end
```

3.3. The stiff ODE method to plot level sets

The stiff ordinary differential equation (ODE) package in MATLAB is relatively easy to implement since the codes and syntaxes involved are short and simple to understand. It gives the user great control by plotting any level set of an objective function through a specific point. For instance, consider plotting a level curve of a two-variable objective function $F(x_1, x_2) = K$ through a point (a, b) where $K \in \mathbb{R}$. Differentiating $F(x_1, x_2)$ w.r.t t yields

$$\frac{\mathrm{d}F}{\mathrm{d}t} = \frac{\partial F}{\partial x_1} \cdot \frac{\mathrm{d}x_1}{\mathrm{d}t} + \frac{\partial F}{\partial x_2} \cdot \frac{\mathrm{d}x_2}{\mathrm{d}t} = 0, \tag{3.8}$$

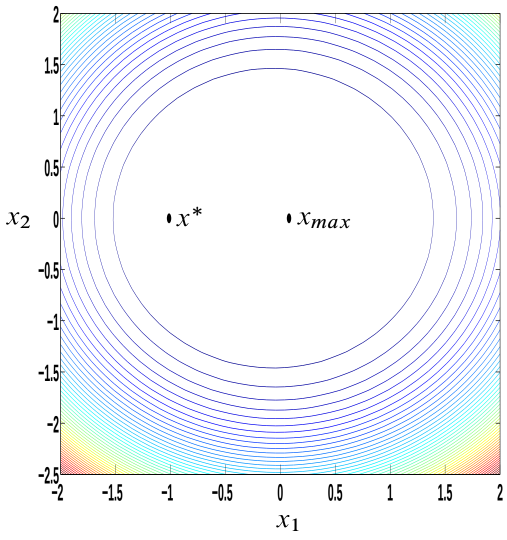
which implies that

$$-\frac{\partial F}{\partial x_1} \cdot \frac{\mathrm{d}x_1}{\mathrm{d}t} = \frac{\partial F}{\partial x_2} \cdot \frac{\mathrm{d}x_2}{\mathrm{d}t}.$$

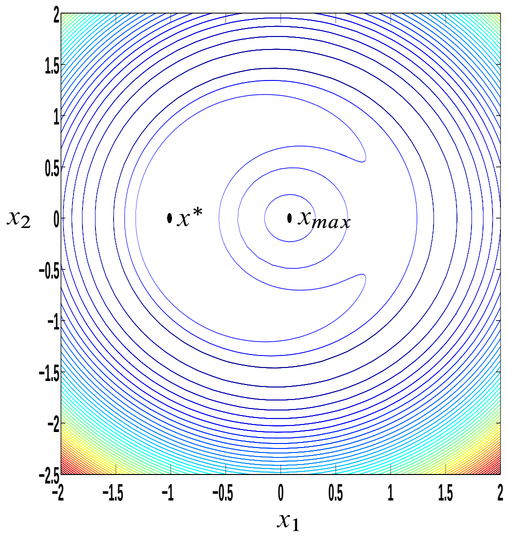
It then follows that equation (3.8) can be converted into an initial value problem (IVP) ODE system given by:

$$\begin{cases} \dfrac{\mathrm{d}x_1}{\mathrm{d}t} = \dfrac{\partial F}{\partial x_2}; \\ \dfrac{\mathrm{d}x_2}{\mathrm{d}t} = -\dfrac{\partial F}{\partial x_1} \end{cases} \quad \text{with} \quad \begin{array}{l} x_1(0) = a; \\ x_2(0) = b. \end{array} \tag{3.9}$$

This technique of employing the stiff ODE method to plot the level sets of an NLS objective function will be utilized whenever the level curves near the stationary points do not appear in a MATLAB plot. This is particularly useful when the objective function has multiple stationary points that are close together. Thus, with the advantage of choosing a specific point through which a level curve passes through, the user is able to produce a more desirable visualization of the figure. Two examples of using the stiff ODE method to plot the missing level curves near different stationary points in MATLAB plots are illustrated below.

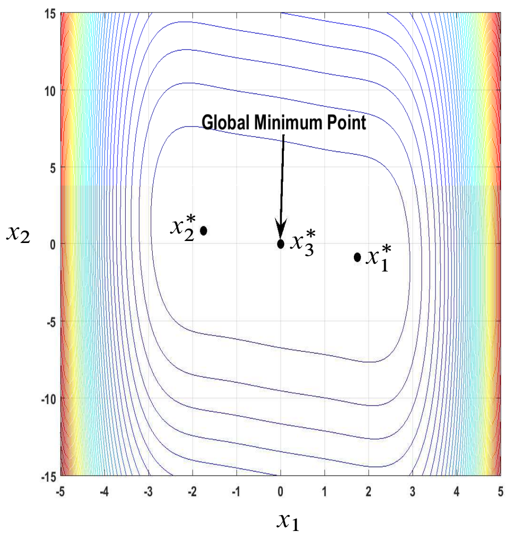


(a) Plot of level sets of $F(x)$ without stiff ODE.

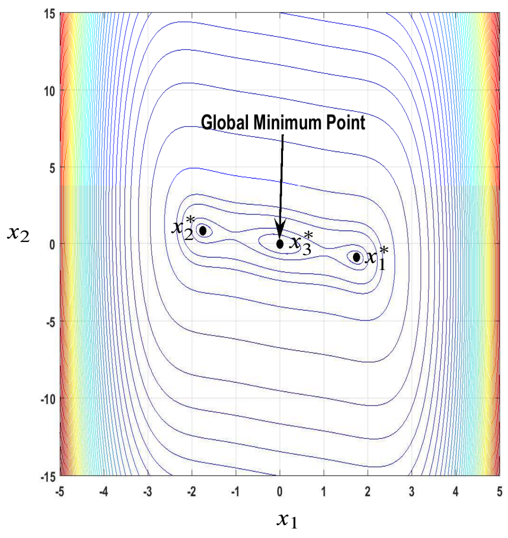


(b) Plot of level sets of $F(x)$ with stiff ODE.

Figure 3.2. A function $F(x)$ with a minimum point x^* and a maximum point x_{max}.



(a) Plot of level sets of $F(x)$ without stiff ODE.



(b) Plot of level sets of $F(x)$ with stiff ODE.

Figure 3.3. A function $F(x)$ with three minimum points. The points x_1^* and x_2^* are the local minimum points while x_3^* is the global minimum point.

3.4. Lyapunov-based line search numerical methods for NLS

In this section, the Lyapunov function theorem will be incorporated into the algorithms of the existing methods for NLS so that convergence towards a minimum point x^* is assured. Other than that, numerical differentiation is also implemented in these algorithms to avoid the need to compute derivatives analytically. The pseudocode for these modified SD algorithm and GN algorithm are given in Algorithms 4 and 5 respectively.

As discussed in Chapter 2, the Newton's method only works well at the final stage of the iterations. Therefore, for iterations computed far away from the solution, the Hessian matrix may be indefinite and so the Newton's method may fail to converge. In order to overcome this difficulty, the Newton's method is defined explicitly in a two-phase manner. In Phase-I, when the iterations are computed far away from the minimum point x^*, backtracking line search with inexact step length is employed to ensure that $F(x)$ is monotonic decreasing and then switches to Newton's method with $\alpha = 1$ in Phase-II for iterations near x^* when the gradient is sufficiently small. This is because the objective function $F(x)$ is approximately quadratic near x^*. Again, backtracking line search is employed to ensure that $F(x)$ is monotonic decreasing. In addition, there is a possibility that the Hessian matrix is singular and hence the Newton's method cannot be applied. The Newton's method described here is stated in Algorithm 6.

Algorithm 4: The modified SD method for NLS

```
Initial setting
Initialize $\epsilon = 10^{-6}$, $N = 50000$ (maximum iteration number)
Choose an initial step length $\alpha_0 > 0$
for $k = 0, 1, 2, 3, \ldots N$ or $\|g(x)\| > \epsilon$ do
    repeat
        Evaluate $F(x_k)$ and $g(x_k)$ using Algorithm 2
        Compute $x_{k+1} = x_k - \alpha_k g(x)$
        while $\Delta F(x) > 0$ do
            Perform backtracking line search using Algorithm 1
        end
    until $k$=$N$ or $\|g(x)\| < \epsilon$
end
```

3.5. Lyapunov-based trust region LM method for NLS

As discussed in Section (2.5.4), the LM method is a trust region method where a Lagrange parameter μ_k is varied in order to obtain a good ratio between the values of the predicted and the actual functions. This ratio is measured using a ratio test defined in (2.24). In this thesis, the MATLAB program for the ratio test is adopted from that developed by C.T. Kelly. This program is available online under the file name *trtestlm* and a description of it can be found in Algorithm 3.3.4. of his book (see Kelly (1999), pg 57). For the convenience of the reader, this program is provided in Algorithm 7 below. Note that the sufficient decrease in the function value which

Algorithm 5: The modified GN method for NLS

```
Initial setting
Initialize $\epsilon = 10^{-6}$, $N = 50000$ (maximum iteration number)
Choose an initial step length $\alpha_0 > 0$
for $k = 0, 1, 2, 3, \ldots N$ or $\|g(x)\| > \epsilon$ do
    repeat
        Evaluate $F(x_k)$, $g(x_k)$ using Algorithm 2
        Evaluate $J(x_k)$ using Algorithm 3
        Compute $x_{k+1} = x_k - \alpha_k [J(x_k)^T J(x_k)]^{-1} g(x)$
        while $\Delta F(x) > 0$ do
            Perform backtracking line search using Algorithm 1
        end
    until $k$=$N$ or $\|g(x)\| < \epsilon$
end
```

is required by the Lyapunov function theorem is ensured implicitly in the trust region LM method through the ratio test.

The MATLAB program for LM method used in this thesis was developed by C.T. Kelly in December 1997 and later updated on July 23, 2016. This MATLAB program is readily available online under the file name *levmar*. A description of this program can also be found in Algorithm 3.3.5 of his book (see Kelly (1999), pg 58). For the convenience of the reader, this program is provided in Algorithm 8 below.

3.6. The new approximate greatest descent (AGD) method for NLS

The Approximate Greatest Descent (AGD) method was first proposed by Goh (2009) for unconstrained optimization problems and later extended to optimization problems with equality constraints in Goh (2011). However, to date, it has not been applied to solve NLS problem. Unlike other numerical methods, the AGD method uses the original objective function (2.1) to construct its iterations instead of an approximate

Algorithm 6: The modified Newton's method for NLS

```
Phase-I: Far away from Solution
Initial setting
Initialize $\epsilon_1 = 1$ (stopping criteria for Phase-I), $N = 50000$
Choose an initial step length $\alpha_0 > 0$
for $k = 0, 1, 2, 3, \ldots N$ or $\|g(x)\| > \epsilon_1$ do
    repeat
        Evaluate $F(x_k)$, $g(x_k)$ and $H(x_k)$ using Algorithm 2
        Compute $x_{k+1} = x_k - \alpha_k H^{-1}(x_k) g(x_k)$
        while $\Delta F(x) > 0$ do
            Perform backtracking line search using Algorithm 1
        end
    until $k$=$N$ or $\|g(x)\| < \epsilon_1$
end
Phase-II: Close to solution
Initialize $\epsilon = 10^{-6}$ (stopping criteria for Phase-II)
Set initial step length $\alpha_0 = 1$
for $t = 0, 1, 2, 3, \ldots N$ or $\|g(x)\| > \epsilon$ do
    repeat
        Evaluate $F(x_t)$, $g(x_t)$ and $H(x_t)$ using Algorithm 2
        Set $\alpha_t = 1$
        Compute $x_{t+1} = x_t - \alpha_t H^{-1}(x_t) g(x_t)$
        while $\Delta F(x) > 0$ do
            Perform backtracking line search using Algorithm 1
        end
    until $k$=$N$ or $\|g(x)\| < \epsilon$
end
```

Algorithm 7: Ratio test for the LM method

```
Initial setting
Initialize $N = 3000$ (maximum iteration number)
Set $z = x_{k+1}$
Choose an initial $\mu_0 > 0$
while $k = 0, 1, 2, 3, \ldots N$ or $z = x_{k+1}$ do
    repeat
        Evaluate $F(x_k)$
        Compute $p_k = x_{k+1} - x_k$
        Compute $\Gamma_k$ using Equation (2.24)
        if $\Gamma_k < \Gamma_0$ then
            Set $z = x_{k+1}$ and $\mu_k = \max(\mu_{up}\mu_k, \mu_0)$
            Recompute $x_{k+1}$ with new $\mu_k$
        else if $\Gamma_0 \leqslant \Gamma_k < \Gamma_{low}$ then
            Set $z = x_{k+1}$ and $\mu_k = \max(\mu_{up}\mu_k, \mu_0)$
        else
            Set $z = x_{k+1}$
            if $\Gamma_k > \Gamma_{high}$ then
                Set $\mu_k = \mu_{down}\mu_k$
                if $\mu_k < \mu_0$ then
                    Set $\mu_k = 0$
                end
            end
        end
    until $k = N$
end
```

Algorithm 8: The LM method for NLS

```
Initial setting
Initialize $\epsilon = 10^{-6}$ and $N = 50000$ (maximum iteration number)
Choose $\mu_0 = 10^{-4}$  Choose an initial $\mu_1 > \|g_(x_1)\|$
for $k = 0, 1, 2, 3, \ldots N$ or $\|g_(x_k) < \epsilon\|$ do
    repeat
        Evaluate $F(x_k)$, $g(x_k)$ using Algorithm 2
        Evaluate $J(x_k)$ using Algorithm 3
        Compute $x_{k+1} = x_k - [H_T(x_k) + \mu_k I]^{-1} g(x_k)$
        Use Algorithm 7 to perform ratio test
    until $k$=$N$ or $\|g(x)\| < \epsilon$
end
```

quadratic model in (2.4). According to Goh (2009), in the computation of a numerical solution, we are interested in finding the optimal trajectory, which is obtained by joining an initial guessed point to the minimum point, in a finite time.

3.6.1. The AGD method for NLS

The long-term optimal trajectory can be achieved by reformulating the numerical unconstrained optimization problem as a multi-stage decision problem (i.e. by considering it as a sequence of optimization problems similar to a trust region method). As a result, we seek the minimum points x_{k+1} in a sequence of neighbourhoods. It follows that the long-term optimal trajectory may be constructed by linking up the solutions x_{k+1} of these subproblems in the sequence of neighbourhoods as depicted in Figure 3.4. A similar figure can be found in Goh (2009).

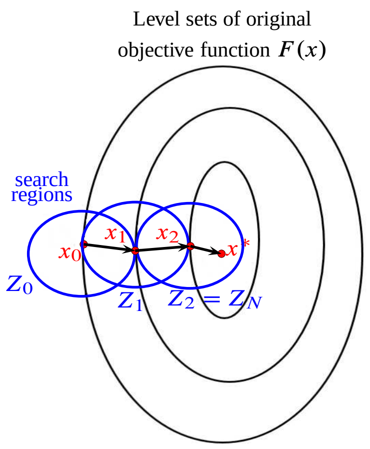


Figure 3.4. A long-term optimal trajectory from x_0 to x^* in three iterations. The first two iterations are the greatest descent steps while the final iteration approximates the Newton's step.

Consider a sequence of spherical neighbourhoods $Z_0, Z_1, \dots, Z_N$ where the minimum point x^* of $F(x)$ is located in Z_N after $N+1$ iterations. Assume that $F(x)$ has a unique minimum point $x^* \in \mathbb{R}^n$. In each of the neighbourhoods $Z_0, Z_1, \dots, Z_{N-1}$, i.e. except the last neighbourhood, the AGD iteration will generate points on the boundary of the these search region. This formulates the AGD search direction p_k^{AGD} as given in the following theorem.

Theorem 3.3. *Suppose a point is computed at every boundary of the search regions $Z_0, Z_1, \dots, Z_{N-1}$ of radius R such that the next objective function value $F(x_{k+1})$ is minimized and assume $F(x)$ has a unique minimum point $x^* \in Z_N$. Then, the search direction*

$$p_k^{AGD} = -g(x_{k+1}) \quad \textit{for} \quad k = 0, 1, \dots, N-1. \tag{3.10}$$

must be satisfied.

Proof. Mathematically, this is formulated as

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} \quad F(x_{k+1}) \\ &\text{s.t.} \quad \|x_{k+1} - x_k\|^2 = R^2. \end{aligned}$$

Then, the Lagrange function $L(x)$ is given by

$$L(x_k) = F(x_{k+1}) + \lambda_k[u_k^T u_k - R^2] \tag{3.11}$$

where $u_k = \alpha_k p_k = x_{k+1} - x_k$ is the step taken and λ_k is the Lagrange multiplier. Taking the partial derivative of (3.11) and applying the optimality condition, one obtains

$$\frac{\partial L}{\partial u_k} = \nabla F(x_{k+1}) + 2\lambda_k u_k = 0. \tag{3.12}$$

Strictly speaking, equation (3.12) is a nonlinear equation which has multiple solutions that is difficult to solve. For simplicity, we let $2\lambda_k \alpha_k = 1$ to obtain

$$p_k^{AGD} = -\nabla F(x_{k+1}) = -g(x_{k+1})$$

which completes the proof. ❏

It is important to note that there are other ways to solve equation (3.12) instead of letting $2\lambda_k\alpha_k = 1$. In order to obtain the AGD iterative equation, the AGD search direction p_k^{AGD} in (3.10) is approximated using the Taylor's series expansion to give

$$p_k^{AGD} = -[I + \alpha_k H(x_k)]^{-1}\nabla F(x_k) = -[I + \alpha_k H(x_k)]^{-1} g(x_k) \tag{3.13}$$

and so from (2.2), the AGD iterative equation given by

$$x_{k+1} = x_k - \alpha_k[I + \alpha_k H(x_k)]^{-1} g(x_k). \tag{3.14}$$

By letting $\mu_k = \frac{1}{\alpha_k}$, the AGD iterative equation (3.14) is simplified to give

$$x_{k+1} = x_k - [\mu_k I + H(x_k)]^{-1} g(x_k). \tag{3.15}$$

In the last search region Z_N, we seek the minimum point x^* inside Z_N. Hence, the direction p_k^{AGD} must satisfy the stationary condition

$$g(x_N + \alpha_N p_N^{AGD}) = g(x_N) + \alpha_N H(x_N) p_N^{AGD} = 0 \tag{3.16}$$

which is simply the Newton's method. This leads to a greatest descent direction in each sequence of neighbourhoods (Goh, 2011; 2012).

From the AGD iterative equation (3.15), it is obvious that the AGD method approximates the slow linear convergence SD method for small α_k. Conversely, it approximates the fast quadratic convergence Newton's method for large α_k. Hence, in order to achieve a fast convergence, the AGD method must approximate or merge with the Newton's method near the minimum point x^*. This is done by choosing the step length α_k such that $\alpha_k \to \infty$ as $x \to x^*$. Such a step length can be derived in a systematic way mathematically as given by the next lemma.

Lemma 3.2. *Assume condition* (3.10) *holds and suppose* $u_k = \alpha_k p_k = R$ *is true. Then, the relative step length* α_k *is approximated as*

$$\alpha_k = \frac{R}{\|g(x_k)\|}. \tag{3.17}$$

Proof. Since $u_k^T u_k = R^2$ and $u_k = \alpha_k p_k$, we have

$$\alpha_k^2 = \frac{R^2}{\|p_k\|^2} \qquad \Rightarrow \quad \alpha_k = \frac{R}{\|p_k\|} \tag{3.18}$$

$$\therefore \quad \alpha_k = \frac{R}{\|g(x_{k+1})\|}. \qquad \text{[from (3.10)]}$$

However, it is not possible to obtain the value of $g(x_{k+1})$; i.e. the value of the gradient at the next iterative step. As a result, $g(x_{k+1})$ is approximated by $g(x_k)$ to obtain

$$\alpha_k = \frac{R}{\|g(x_k)\|}.$$

Substituting (3.18) into the iterative equation (2.2) yields

$$x_{k+1} = x_k + \frac{R}{\|p_k\|} p_k = x_k + R\hat{p}_k \tag{3.19}$$

where $\hat{p}_k$ is a unit vector in the direction of p_k. It follows that the parameter α_k is the relative step length of the iterative equation (2.2) and hence equation (3.14). This concludes the proof. ❏

Remark 3.1. The parameter α_k is strictly speaking the relative step length of the iterative equation (2.2). It is a step length only if $\|p_k\| = 1$. For simplicity, we normally use the term step length only. However, it should be reminded that this step length is actually a relative step length instead of just a step length.

Since$\mu_k = \frac{1}{\alpha_k}$, it can be deduced from (3.17) that

$$\mu_k = \frac{\|g(x_k)\|}{R}. \tag{3.20}$$

Equation (3.20) suggests that the step length μ_k is inversely proportional to the radius of the search region R. Note that this finding has already been discussed in Section (2.5.4) of the LM method. Similar choices of step length are also considered by Kelly (1999) for the LM method with $\mu_k = \|g(x_k)\|$ (see Algorithm 8) and Grantham (2003) and Grantham (2007) for a continuous-time LM method with $\mu_k = \gamma\|g(x_k)\|$.

From the above discussion, the pseudocode for the AGD method is given in Algorithm 9.

Algorithm 9: The AGD method for NLS

```
Initial setting
Initialize $\epsilon = 10^{-6}$, $N = 50000$ (maximum iteration number)
Choose an initial step length $\alpha_0 > 0$
for $k = 0, 1, 2, 3, \ldots N$ or $\|g(x)\| > \epsilon$ do
    repeat
        Evaluate $F(x_k)$, $g(x_k)$ and $H(x_k)$ using Algorithm 2
        Evaluate $R = \|g(x_k)\|$, $\alpha_k = \frac{R}{\|g(x_k)\|}$ and $\mu_k = \frac{1}{\alpha_k}$
        Compute $x_{k+1} = x_k - [\mu_k I + H]^{-1} g(x_k)$
        while $\Delta F(x) > 0$ do
            Perform backtracking line search using Algorithm 1
        end
    until $k$=$N$ or $\|g(x)\| < \epsilon$
end
```

3.6.2. The AGDN method for NLS

In the previous section, we have discussed that the AGD method will approximate the Newton's method implicitly for large α_k. It follows that there is a potential to develop a two-phase AGD method. It consists of two explicitly defined phases with the AGD method in Phase-I when the current iterations are far away from the minimum point x^* and then switches to the Newton's method in Phase-II when the gradient is sufficiently small (i.e. near the minimum point). This two-phase AGD method (abbreviated as AGDN) will have a faster convergence rate compared to the single phase AGD method discussed in the previous section since the Newton's method is used explicitly near x^* instead of an approximate version of it. This AGDN method is stated below in Algorithm 10.

It is important to note that the AGDN method may fail to work in Phase-II if the Hessian matrix is singular. This is because the Newton's method fails to work whenever the Hessian matrix is singular. In this case, it is advisable to use the single phase AGD method (see Algorithm 9) to solve the NLS problem under studied. An advantage of the AGD method over the AGDN method is that the presence of the parameter μ_k in equation (3.15) ensures the non-singularity of $\mu I + H$ so that its inverse always exists.

3.7. Conclusion

The Zoutendijk theorem is normally used as a set of prototype conditions for establishing the convergence of a numerical method towards a minimum point x^*. However, the Zoutendijk theorem only ensures the convergence of a trajectory from an initial point to a stationary point in an open-loop manner. This suggests that it is possible to achieve an undesirable convergence towards a maximum point or a saddle point. Moreover, in an open loop policy, the outcome could be sensitive to numerical errors in the initial state vector x_0 or the current vector x_{k+1}. Due to these reasons, the Lyapunov function theorem, which ensures the convergence of a numerical method in a feedback-type manner, are incorporated in all the algorithms discussed in this chap-

Algorithm 10: The AGDN method for NLS

```
Phase-I: AGD method — Far away from Solution
Initial setting
Initialize $\epsilon_1 = 10^{-3}$, $N = 50000$ (maximum iteration number)
Choose an initial step length $\alpha_0 > 0$
for $k = 0, 1, 2, 3, \ldots N$ or $\|g(x)\| > \epsilon_1$ do
    repeat
        Evaluate $F(x_k)$, $g(x_k)$ and $H(x_k)$ using Algorithm 2
        Evaluate $R = \|g(x_k)\|$, $\alpha_k = \frac{R}{\|g(x_k)\|}$ and $\mu_k = \frac{1}{\alpha_k}$
        Compute $x_{k+1} = x_k - [\mu_k I + H]^{-1} g(x_k)$
        while $\Delta F(x) > 0$ do
            Perform backtracking line search using Algorithm 1
        end
    until $k$=$N$ or $\|g(x)\| < \epsilon_1$
end
Phase-II: Newton's method — Close to solution
Initialize $\epsilon = 10^{-6}$ (stopping criteria for Phase-II)
Set initial step length $\alpha_0 = 1$
for $t = 0, 1, 2, 3, \ldots N$ or $\|g(x)\| > \epsilon$ do
    repeat
        Evaluate $F(x_t)$, $g(x_t)$ and $H(x_t)$ using Algorithm 2
        Set $\alpha_t = 1$
        Compute $x_{t+1} = x_t - \alpha_t H^{-1}(x_t) g(x_t)$
        while $\Delta F(x) > 0$ do
            Perform backtracking line search using Algorithm 1
        end
    until $k$=$N$ or $\|g(x)\| < \epsilon$
end
```

ter to ensure the monotonic decreasing property of the objective function $F(x)$ of the NLS problem. If the level sets of the objective function are properly nested, all trajectories will converge to a minimum point x^* provided that the iterations stay within the properly nested region containing x^*. This is depicted in Figure 3.1 when the Lyapunov function theorem is applied to investigate the convergence of the Barbashin and Krasovskii (1952) function.

Furthermore, numerical differentiation, which uses the finite difference approximations, is also implemented into numerical algorithms to avoid tedious calculation of derivatives of functions. This is done by employing the forward difference formula when the iterations are far away from the solution and then switches to the central difference formula when the solution is near for sufficiently small $\|g(x_k)\|$ (see Algorithm 2).

Besides that, the stiff ODE method is used as a technique to plot the missing level curves near multiple stationary points of an objective function in a MATLAB plot. It allows the user to plot a level curve through a specific point so that a more informative plot can be obtained.

The pseudocodes of the existing numerical methods use to solve the NLS problem is provided in Algorithm 4–10 where the Lyapunov function theorem is implemented to ensure the convergence of the numerical methods towards a minimum point and numerical differentiation is used for calculating the derivatives of the functions numerically. It is important to note that the sufficient decrease in the objective function, which is required by the Lyapunov function theorem, is ensured implicitly in the trust region LM method through the ratio test.

The approximate greatest descent (AGD) method, which is a new numerical method to solve NLS problem, is the main focus of this research. It has shown great results when it is applied to solve unconstrained optimization problems. In this research, a modified two-phase AGD method, abbreviated as AGDN, is proposed to solve the NLS problem (see Algorithm 10). It is constructed based on employing the AGD method in Phase-I when the current iterations are far away from the minimum point x^* and then switches to the Newton's method in Phase-II when x^* is near for sufficiently $\|g(x_k)\|$. In the original AGD method (see Algorithm 9), instead of switching to the Newton's method, the AGD method approximates or merges with the Newton's method for large α_k near x^*.

CHAPTER 4

NUMERICAL EXPERIMENTS

In this chapter, some numerical experiments are carried out to test and compare the efficiency, reliability and robustness of the numerical methods discussed in the previous chapters. All the experiments are conducted using the MATLAB programming language where the codes and syntaxes are constructed based on Algorithms 4–10 described in Chapter 3. These numerical methods are tested using two-variable and multi-variable NLS test problems. For two-variable NLS test problems with $n = 2$ variables (or parameters), $n^2 = 4$ initial points are used for testing the efficiency, reliability and robustness of the numerical methods. However, for multi-variable NLS test problems with $n \geqslant 3$ variables, only the standard initial (or starting) points are used since it is a tedious task to choose and run computer simulations for n^2 number of initial points when $n \geqslant 3$. Based on the experimental results, these methods are compared and critically analyzed in terms of the number of iterations and the CPU times required for convergence. Besides that, two logarithmic scaled performance profiles are plotted for an overall analysis of the numerical methods.

4.1. The NLS test problems

According to Moré et al. (1981), there has been too much emphasis on testing the efficiency of the numerical methods rather than on the reliability and robustness of these methods. This is because only one standard initial (or starting) point are tested for each test problem. In addition, this standard initial point is normally close to the solution (or minimum point). As a result, numerical methods that work for the standard initial point may fail for other initial points; especially for points that are chosen far away from the minimum point. Moreover, Moré et al. (1981) further stated that the use of initial points that are chosen far away from the minimum point will normally reveal drastic differences in reliability and robustness between similar algorithms (e.g. two AGD methods).

Strictly speaking, the efficiency, reliability and robustness of a numerical method have distinct qualitative meanings which determine the quality of a numerical method. The efficiency of a numerical method is a measure of the time taken (i.e. the CPU time) it takes to achieve convergence. For instance, a higher number of iterations may require a longer amount of time to reach the minimum point since more computative steps are needed during the numerical process. In other words, the longer a numerical method takes to achieve convergence, the less efficient is the method. On the other hand, the reliability of a numerical method refers to the successful rate or the ability of the method to reach the minimum point. It is normally measured in terms of probability (between 0 and 1) or percentage (between 0 and 100) of solved problems. For example, if a numerical method can solve nine out of ten of the test problems, then it is reliable most of the time with a probability of 0.9 and solves 90% of all the problems. As a result, the higher the probability or percentage of solved problems, the more reliable is the numerical method. In contrast, the robustness of a numerical method denotes the sensitivity of the method towards parameter variations. For instance, a method which is highly sensitive can lead to a false or different solution by small variations of the parametric values.

In order to address this issue, the numerical methods are tested using the two-variables NLS test problems with $n = 2$ for four chosen initial points. These points are chosen by first dividing the 2-dimensional plane into four regions using the lines $x_1 = x_{1e}^*$ and $x_2 = x_{2e}^*$ where $x^* = [x_{1e}^*, x_{2e}^*]$ is the minimum point of the objective function $F(x)$. Then, four initial points are chosen from each region. This can easily be done since we are able to visualize and choose desired initial points from a two-dimensional plane. However, for multi-variable test problem with $n \geqslant 3$, this technique is a tedious task since it involves choosing and running computer experiments for n^2 chosen initial points with $n \geqslant 3$. Furthermore, it is normally hard to visualize an n-dimensional space for $n > 3$. Hence, only the standard initial points are used to test the numerical methods for muti-variable NLS test problems. Hillstrom (1977) first suggested the use of nonstandard initial points by choosing random points from a box surrounding the standard initial point. This approach, which is much more satisfactory, leads to a simulation of huge amount of data that are hard to interpret and reproduce since the initial points are generated randomly. Therefore, further research should be done to address this issue.

In this chapter, each NLS test problem is tested using the same initial point(s) under the same testing environment (i.e. using the same version of MATLAB with the same computer) in order to established an unbiased comparison between the six numerical methods; i.e the SD method, the Newton's method, the GN method, the LM method, the AGD method and the two-phase AGD method (abbreviated as AGDN), as described by Algorithms 4–10. Hence, the codes and syntaxes are written using the MATLAB programming language. As stated in the algorithms, the maximum number of iterations is set to be 50000 and the stopping criteria is set to be $\|g(x_k)\| < 10^{-6}$. Thus, whenever the maximum number of iterations is reached before $\|g(x_k)\| < 10^{-6}$, we declare that this run as a failure.

4.2. Numerical experiments on two-variable NLS test problems

Two-variable functions are the most fundamental and simplest test functions used in testing numerical algorithms. They are important test functions since it is always possible to plot the level sets of such functions. The level sets of a function provides valuable information regarding the behaviors and structures of such functions. According to Lemma (3.1), a function which has properly nested level sets should converge to the minimum point x^* in a finite time provided that the iterations stay within the properly nested region containing x^*. Failure to do so suggests that the method fails to converge.

Remark 4.1. According to Goh et al. (2014), the properly nested condition of the level sets of a function $F(x)$ can be easily verified for a two-variable function. This is achieved by plotting samples of the level sets of the function and by invoking the assumption that the function is continuous. If the level sets of a function are properly nested, then they are topologically equivalent to concentric spherical surfaces.

4.2.1. The two-variable NLS test problems

Table 4.1 provides the two-variable NLS test problems used in numerical experiments that are obtained from Moré et al. (1981) and Adorio (2005) and available in the constrained and unconstrained testing environment, revisited/safe threads (CUTEr/CUTEst). A detailed information of these test problems are given in Appendix A.

Table 4.1. A list of two-variable NLS test problems used in numerical experiments where the abbreviations "TP" and "Dim" denote Test Problem and the dimension of the problem respectively.

TP	**Function**	**Dim**		**TP**	**Function**	**Dim**	
No.	**Name**	***n***	***m***	**No.**	**Name**	***n***	***m***
1.	NF 1	2	3	7.	Mod. BK 2	2	3
2.	NF 2	2	4	8.	Mod. RF 1	2	3
3.	NF 3	2	3	9.	Mod. RF 2	2	3
4.	3-hump CF	2	4	10.	Mod. RF 3	2	3
5.	BBSF	2	3	11.	BF	2	3
6.	Mod. BK 1	2	3	12.	J&S	2	10

4.2.2. Experimental results for two-variable NLS test problems

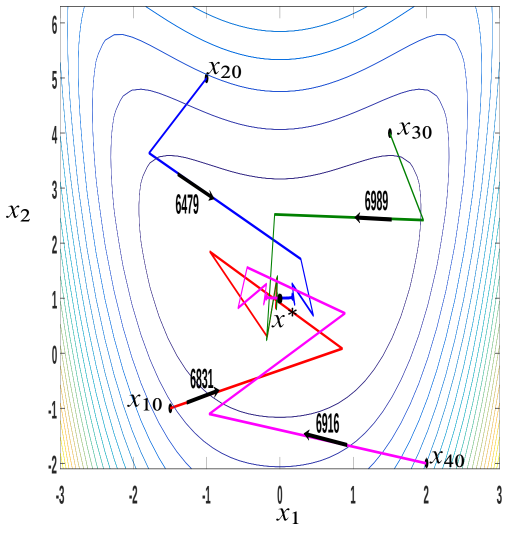
Each test problem listed in Table 4.1 are tested using the six numerical methods (as described by Algorithms 4–10) for four initial points. In order to compare between the efficiency, reliability and robustness of these methods, the experimental results are displayed by adopting the following pattern. For each test problem, the experimental data obtained from numerical simulations are first recorded in a table where x_{j0} denotes the different initial points used in the simulations and k stands for the number of iterations. This is followed by plotting the phase portraits of the test problems. The phase portraits depict the behaviours of the trajectories of the test problems when different numerical methods are applied to solve them. Since similar behaviours of the trajectories are observed for these test problems, only the phase portraits for the first four test problems are plotted. Finally, two graphs are plotted respectively to compare between the number of iterations and the CPU times required for each numerical methods to converge to the minimum points. Whenever a method fails to converge, the number of iterations and the CPU times are recorded as zeros in these graphs.

Table 4.2. Record of experimental results for Test Problem 1.

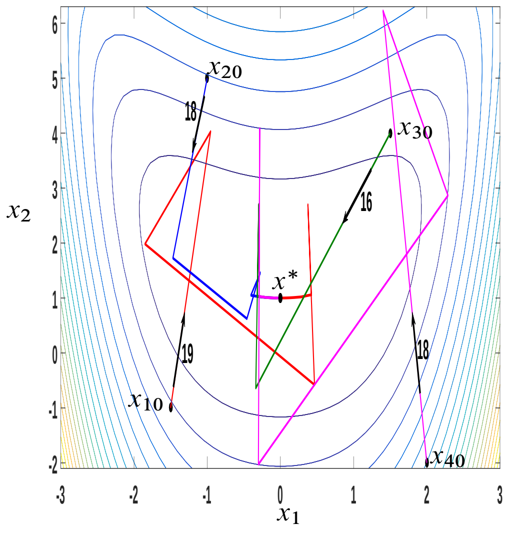
TP No.	**Method**	$x_{j0};\ j=1,2\ldots$	k	$F(x^*)$	**CPU Times (s)**
1.	SD	$[-1.5,-1]$	6831	9.91×10^{-10}	0.9146
		$[-1,5]$	6479	1.13×10^{-9}	0.8740
		$[1.5,4]$	6989	9.31×10^{-10}	0.9346
		$[2,-2]$	6916	9.96×10^{-10}	0.9228
	Newton's	$[-1.5,-1]$	19	4.56×10^{-12}	0.1952
		$[-1,5]$	18	5.94×10^{-7}	0.1612
		$[1.5,4]$	16	6.91×10^{-12}	0.2265
		$[2,-2]$	18	4.99×10^{-12}	0.2067
	GN	$[-1.5,-1]$	6	3.21×10^{-18}	0.1761
		$[-1,5]$	10	2.34×10^{-20}	0.1771
		$[1.5,4]$	12	3.74×10^{-22}	0.1861
		$[2,-2]$	6	5.86×10^{-18}	0.1817
	LM	$[-1.5,-1]$	21	1.03×10^{-13}	0.2084
		$[-1,5]$	20	1.58×10^{-13}	0.1966
		$[1.5,4]$	19	6.88×10^{-13}	0.1933
		$[2,-2]$	22	1.43×10^{-13}	0.2012
	AGD	$[-1.5,-1]$	18	3.29×10^{-12}	0.1833
		$[-1,5]$	18	4.67×10^{-12}	0.1706
		$[1.5,4]$	16	5.52×10^{-12}	0.1769
		$[2,-2]$	19	2.47×10^{-12}	0.1773
	AGDN	$[-1.5,-1]$	18	3.21×10^{-12}	0.1998
		$[-1,5]$	18	4.47×10^{-12}	0.1927
		$[1.5,4]$	16	5.08×10^{-12}	0.1980
		$[2,-2]$	19	2.44×10^{-12}	0.1972

The phase portraits in Figure 4.1 depict the behaviours of the trajectories of Test

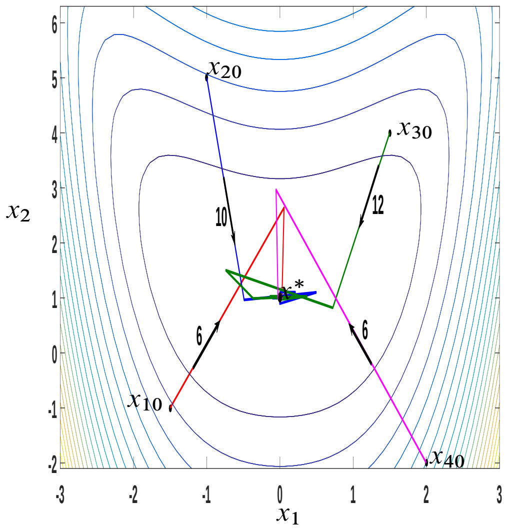
Problem 1 for four initial points using the six numerical methods. Observe that the function of Test Problem 1 has properly nested level sets and hence convergence is assured. The direction of the trajectories are shown by arrows and the numbers beside each arrow denote the number of iterations used to converge to x^*.



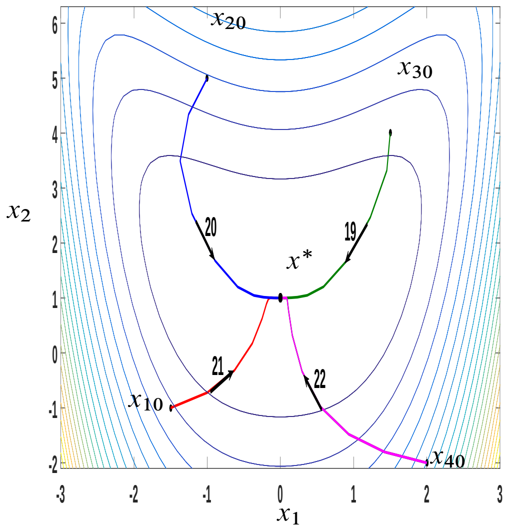
(a) Phase portrait of Test Problem 1 using the SD method.



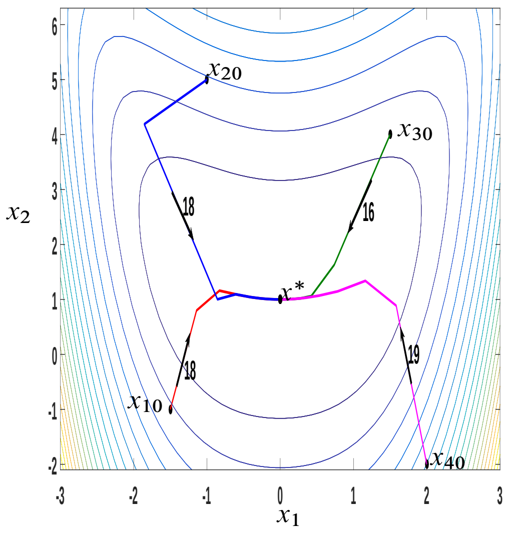
(b) Phase portrait of Test Problem 1 using the Newton's method.



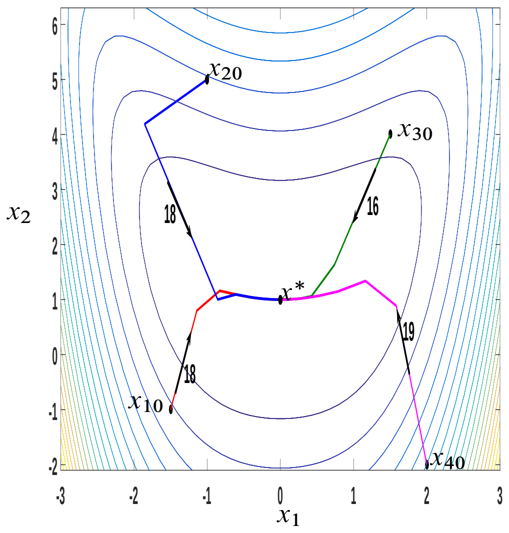
(c) Phase portrait of Test Problem 1 using the GN method.



(d) Phase portrait of Test Problem 1 using the LM method.



(e) Phase portrait of Test Problem 1 using the AGD method.



(f) Phase portrait of Test Problem 1 using the AGDN method.

Figure 4.1. The phase portraits of Test Problem 1 using the six numerical methods for four initial points where $x^* = [0, 1]$. The direction of the trajectories are shown by arrows and the numbers beside each arrow denote the number of iterations used to converge to x^*.

Figures 4.2 and 4.3 shows the comparisons between the number of iterations and the CPU times when the six different numerical methods are applied to solve Test Problem 1 for four initial points.

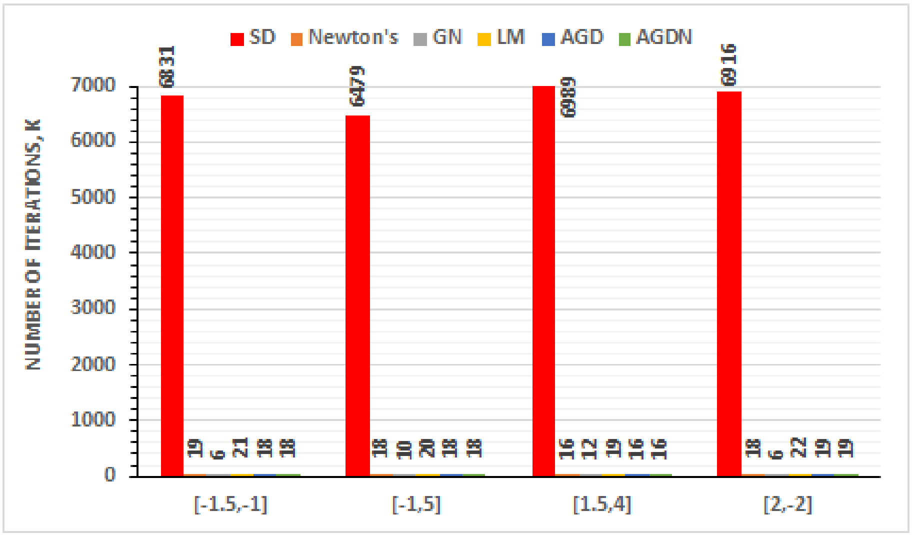


Figure 4.2. A comparison between the number of iterations for six numerical methods using four initial points; i.e. $x_{10} = [-1.5, -1]$, $x_{20} = [-1, 5]$, $x_{30} = [1.5, 4]$ and $x_{40} = [2, -2]$ for Test Problem 1.

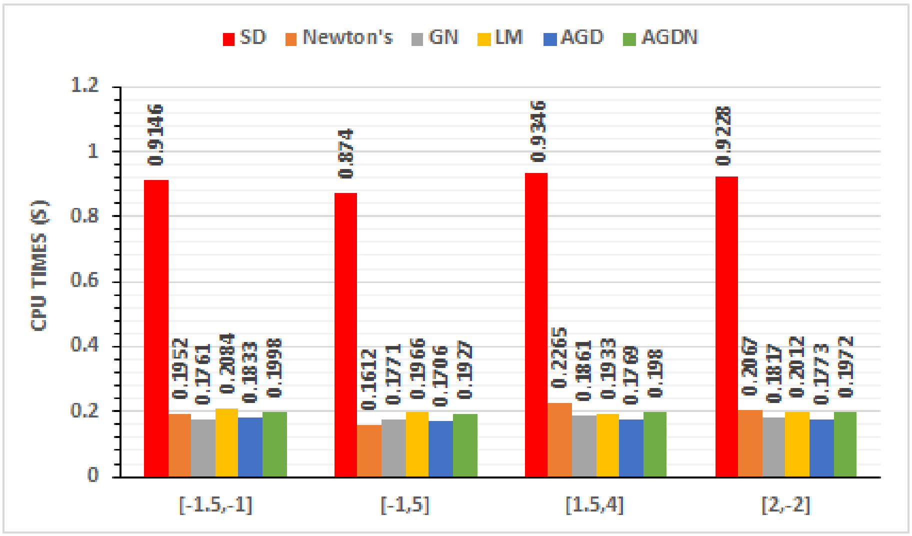


Figure 4.3. A comparison between the CPU times for six numerical methods using four initial points; i.e. $x_{10} = [-1.5, -1]$, $x_{20} = [-1, 5]$, $x_{30} = [1.5, 4]$ and $x_{40} = [2, -2]$ for Test Problem 1.

From Table 4.2 and Figure 4.1, observe that all the numerical methods show convergence towards $x^* = [0, 1]$ as expected since the function of Test Problem 1 has properly nested level sets. From Figure 4.2, notice that the SD method requires a comparatively large number of iterations for convergence compared to all the other methods. Hence, it requires the longest amount of time to reach x^* as illustrated in Figure 4.3.

On the other hand, the number of iterations required by the Newton's, the GN, the LM, the AGD and the AGDN methods are comparable with the GN method having the least number of iterations as shown in Figure 4.2. Nonetheless, it can be observed in Figure 4.3 that the LM method takes the longest amount of time on average to achieve convergence. In addition, numerical results have shown that the GN method has shown numerical termination towards the minimum point $x^* = [0, 1]$ for three chosen initial points $x_{10} = [-1.5, -1]$, $x_{20} = [-1, 5]$ and $x_{30} = [1.5, 4]$ as compared to the other methods which only show numerical convergence towards x^*.

The phase portraits in Figure 4.1(a) illustrate that the phase trajectories generated by the SD method produces erratic zigzag behaviours near x^*. This explains why the SD method requires a comparatively high number of iterations to achieve convergence with the slowest convergence rate compared to the other methods. In addition, the zigzag behaviours of these phase trajectories may also indicate that the SD method may become unreliable for higher dimensional problems. Similarly, the phase trajectories generated by the Newton's method also behave erratically upon reaching x^* as illustrated in Figure 4.1(b). However, due to the fast quadratic convergence of the Newton's method, convergence can be achieved in a very short amount of time with a few number of iterations. In contrast, the phase trajectories generated by the LM, the AGD and the AGDN methods behave steadily before approaching x^*. As a result, fewer number of iterations are needed to reach x^* in a shorter amount of time. However, the phase trajectories generated by the GN method have shown some erratic behaviours near x^* despite its good numerical outcomes.

The AGD method outperforms all the other numerical methods on average in terms of execution time with an average time of 0.1770 seconds. This is followed by the AGDN method with an average time of 0.1969 seconds. Notice that the results obtained from the AGD and the AGDN methods are similar since the AGD method merges with the Newton's method near x^* while the AGDN method switches to the Newton's method in Phase-II when the gradient is sufficiently small. Nonetheless, the AGD method has a faster convergence rate compared to the AGDN method.

Table 4.3 and Figures 4.4–4.6 show the experimental results for Test Problem 2.

Table 4.3. Record of experimental results for Test Problem 2.

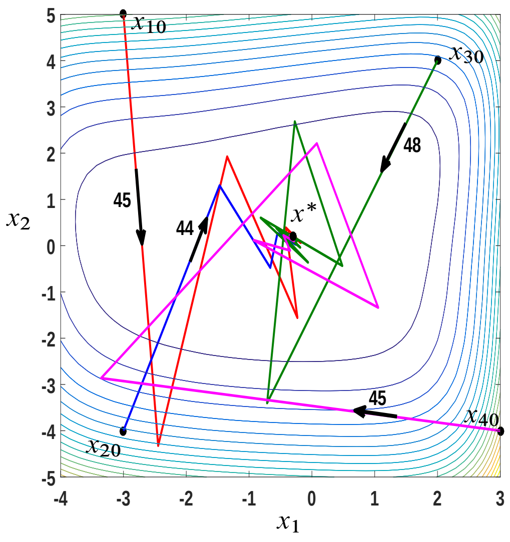
TP No.	Method	$x_{j0};\ j = 1, 2 \ldots$	k	$F(x^*)$	CPU Times (s)
2.	SD	$[-3, 5]$	45	1.250	0.1509
		$[-3, -4]$	44	1.250	0.1452
		$[2, 4]$	48	1.250	0.1457
		$[3, -4]$	45	1.250	0.1451
	Newton's	$[-3, 5]$	6	1.250	0.1469
		$[-3, -4]$	6	1.250	0.1467
		$[2, 4]$	6	1.250	0.1392
		$[3, -4]$	10	1.250	0.1438
	GN	$[-3, 5]$	25	1.250	0.1995
		$[-3, -4]$	19	1.250	0.1908
		$[2, 4]$	21	1.250	0.1908
		$[3, -4]$	20	1.250	0.1920
	LM	$[-3, 5]$	33	1.250	0.2258
		$[-3, -4]$	24	1.250	0.2038
		$[2, 4]$	21	1.250	0.1978
		$[3, -4]$	23	1.250	0.1978
	AGD	$[-3, 5]$	10	1.250	0.1769
		$[-3, -4]$	8	1.250	0.1827
		$[2, 4]$	8	1.250	0.1667
		$[3, -4]$	10	1.250	0.1744
	AGDN	$[-3, 5]$	10	1.250	0.1870
		$[-3, -4]$	8	1.250	0.1833

Continued on next page

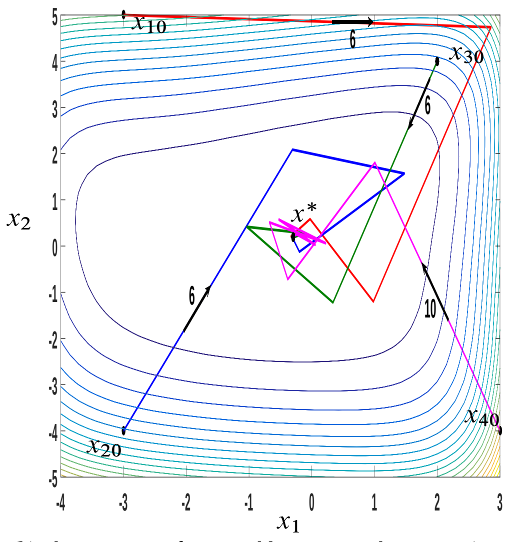
Table 4.3 – *Continued from previous page*

TP No.	Method	$x_{j0};\ j = 1,2\dots$	k	$F(x^*)$	CPU Times (s)
2.		$[2, 4]$	8	1.250	0.1810
		$[3, -4]$	10	1.250	0.1859

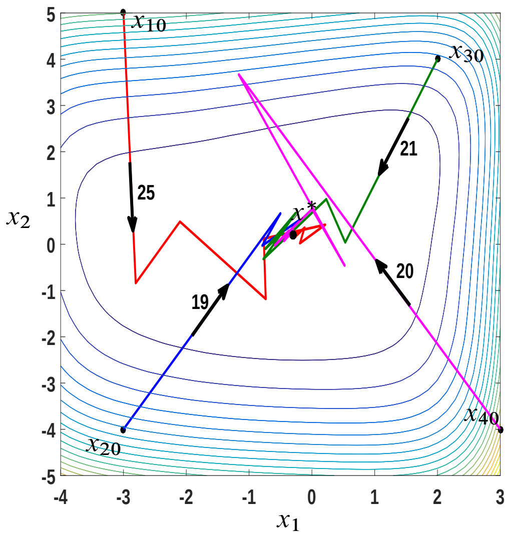
Figure 4.4 (a)–(f) illustrate the phase portraits of Test Problem 2 for four initial points using the six numerical methods.



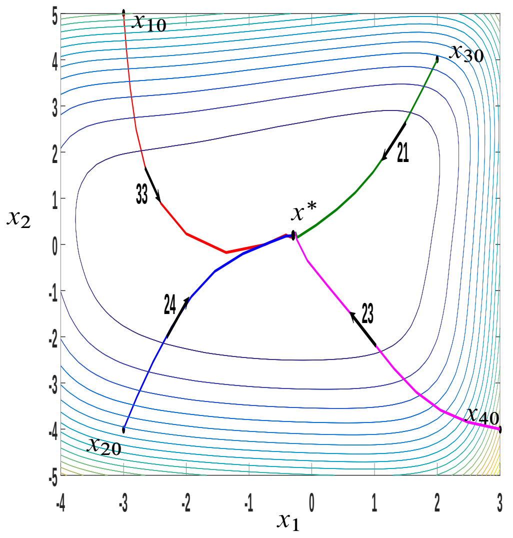
(a) Phase portrait of Test Problem 2 using the SD method.



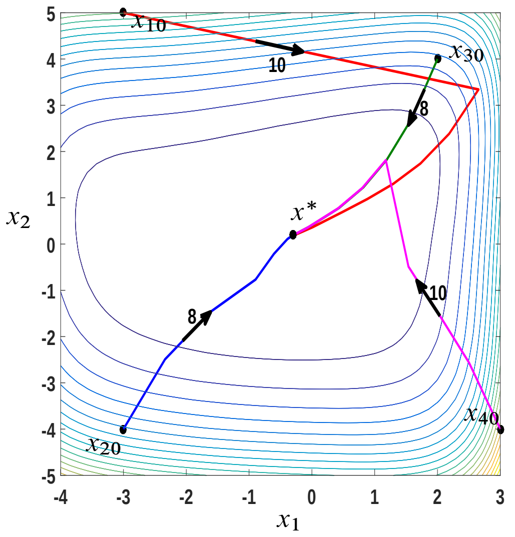
(b) Phase portrait of Test Problem 2 using the Newton's method.



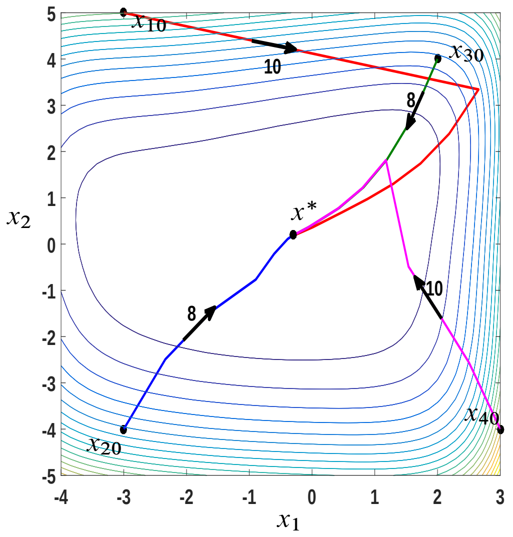
(c) Phase portrait of Test Problem 2 using the GN method.



(d) Phase portrait of Test Problem 2 using the LM method.



(e) Phase portrait of Test Problem 2 using the AGD method.



(f) Phase portrait of Test Problem 2 using the AGDN method.

Figure 4.4. The phase portraits of Test Problem 2 using the six numerical methods for four initial points where $x^* = [-0.2950, 0.1980]$. The direction of the trajectories are shown by arrows and the numbers beside each arrow denote the number of iterations used to converge to x^*.

Figures 4.5 and 4.6 shows the comparisons between the number of iterations and the CPU times when the six different numerical methods are applied to solve Test

Problem 2 for four initial points.

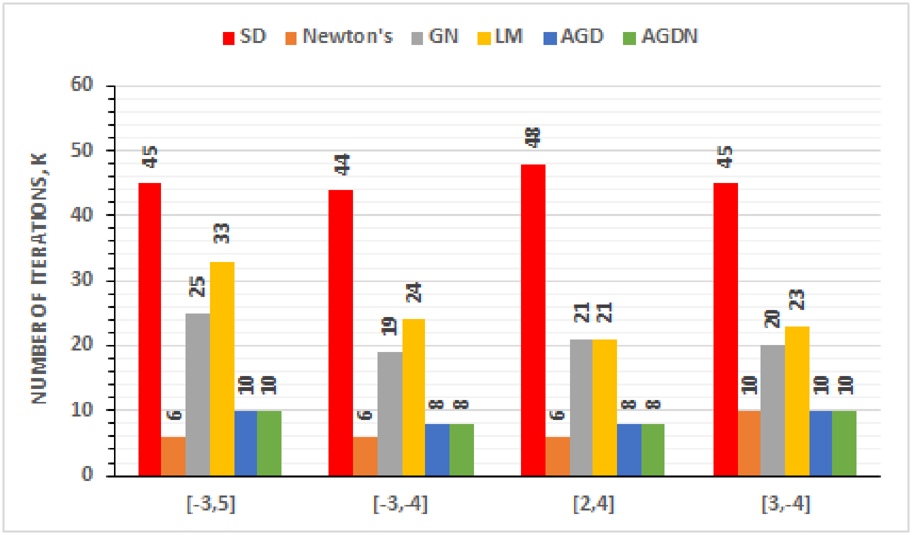


Figure 4.5. A comparison between the number of iterations for six numerical methods using four initial points; i.e. $x_{10} = [-3, 5]$, $x_{20} = [-3, -4]$, $x_{30} = [2, 4]$ and $x_{40} = [3, -4]$ for Test Problem 2.

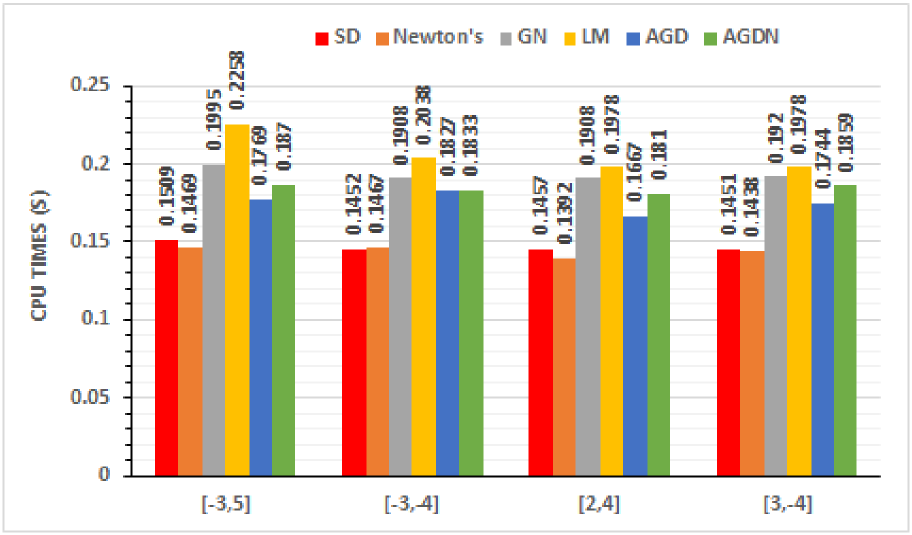


Figure 4.6. A comparison between the CPU times for six numerical methods using four initial points; i.e. $x_{10} = [-3, 5]$, $x_{20} = [-3, -4]$, $x_{30} = [2, 4]$ and $x_{40} = [3, -4]$ for Test Problem 2.

As shown in Figure 4.4, the function of test problem 2 has properly nested level sets and hence convergence is guaranteed. Similar to Test Problem 1, the phase trajectories of Test Problem 2 generated by the SD, the Newton's and the GN methods show erratic behaviours before approaching x^* while those generated by the LM, the AGD and the AGDN methods behave steadily before reaching x^*.

All the numerical methods have shown satisfactory results when they are applied to solve Test Problem 2. All these methods terminate at the minimum point $x^* = [-0.2954, 0.1980]$ within 0.23 seconds as depicted in Figure 4.6. Thus, the number of iterations required by these methods to achieve convergence are relatively small as shown in Figure 4.5.

From Figures 4.5 and 4.6, observe that the SD method has a fast convergence rate despite its high number of iterations. This is because the computation of the SD method is relatively cheap since it only involves the evaluation of the first derivatives (i.e. $p_k^{SD} = -g(x_k)$). In contrast, the LM method has the slowest rate of convergence despite the steady behaviours of its phase trajectories as observed in Figure 4.4(d). This may be due to the use of a truncated Hessian matrix in the LM iterative equation. Similarly, the GN method, which also uses a truncated Hessian matrix, ranked the second slowest in terms of execution times. Among all, the Newton's method exhibits the best convergence rate with the least number of iterations. This is due to the fast quadratic convergence of the Newton's method. Similar to Test Problem 1, the AGD and the AGDN methods show similar numerical results but the rate of convergence of the AGD method is faster than that of the AGDN method.

Table 4.4 and Figures 4.7–4.9 shows the experimental results for Test Problem 3.

Table 4.4. Record of experimental results for Test Problem 3.

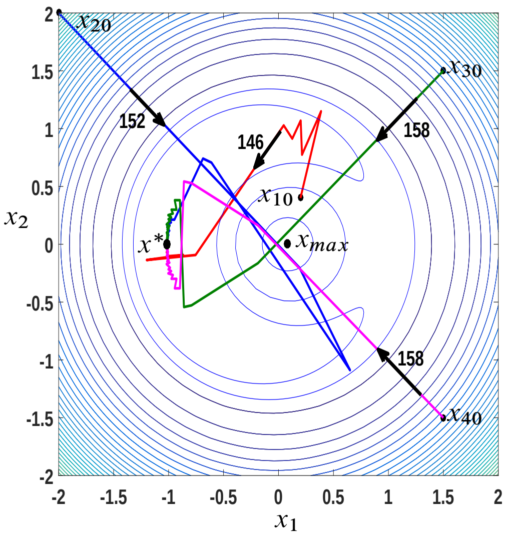
TP No.	**Method**	$x_{j0};\ j = 1, 2 \ldots$	k	$F(x^*)$	**CPU Times (s)**
3.	SD	$[0.2, 0.4]$	146	0.1220	0.1617
		$[-2, 2]$	152	0.1220	0.1603
		$[1.5, 1.5]$	158	0.1220	0.1542
		$[1.5, -1.5]$	158	0.1220	0.1566
	Newton's	$[0.2, 0.4]$	FAILED		

Continued on next page

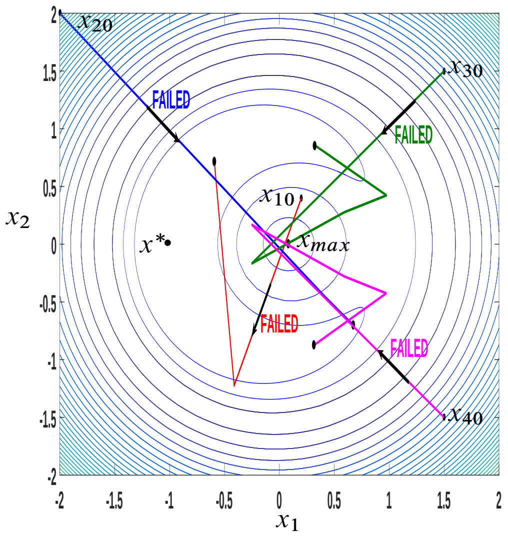
Table 4.4 – *Continued from previous page*

TP No.	**Method**	$x_{j0};\ j = 1, 2 \ldots$	k	$F(x^*)$	**CPU Times (s)**
3.	Newton's	$[-2, 2]$	FAILED		
		$[1.5, 1.5]$	FAILED		
		$[1.5, -1.5]$	FAILED		
	GN	$[0.2, 0.4]$	25	0.1220	0.1833
		$[-2, 2]$	30	0.1220	0.1876
		$[1.5, 1.5]$	29	0.1220	0.1882
		$[1.5, -1.5]$	29	0.1220	0.1849
	LM	$[0.2, 0.4]$	15	0.1220	0.1986
		$[-2, 2]$	19	0.1220	0.2095
		$[1.5, 1.5]$	20	0.1220	0.1962
		$[1.5, -1.5]$	20	0.1220	0.2099
	AGD	$[0.2, 0.4]$	10	0.1220	0.1838
		$[-2, 2]$	10	0.1220	0.1805
		$[1.5, 1.5]$	14	0.1220	0.1813
		$[1.5, -1.5]$	14	0.1220	0.1890
	AGDN	$[0.2, 0.4]$	10	0.1220	0.1945
		$[-2, 2]$	10	0.1220	0.1880
		$[1.5, 1.5]$	14	0.1220	0.1948
		$[1.5, -1.5]$	14	0.1220	0.1979

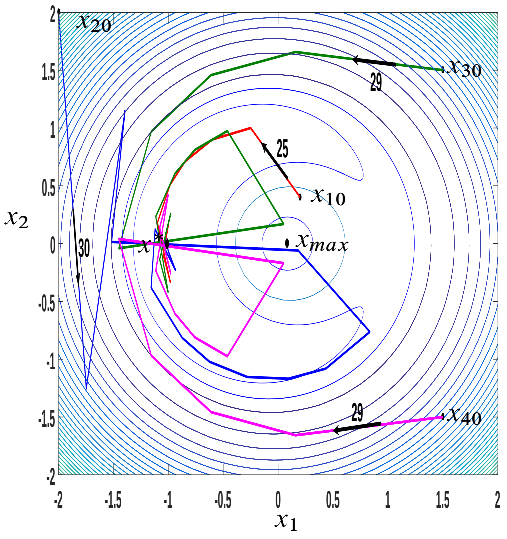
The phase portrait of Test Problem 3, which are obtained using the six numerical methods, are shown in Figures 4.7(a)–(f). Since the function of Test Problem 3 has a global minimum point and a global maximum point that are close to each other, the stiff ODE method is employed to plot the missing level curves near these points.



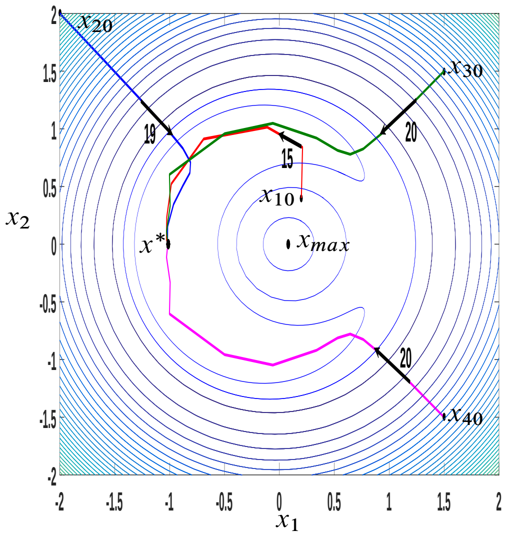
(a) Phase portrait of Test Problem 3 using the SD method.



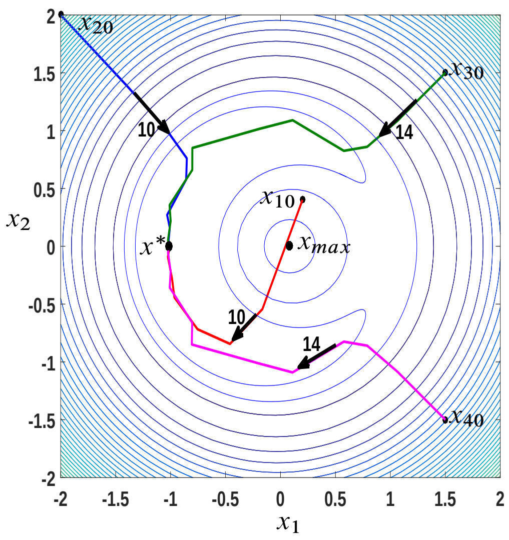
(b) Phase portrait of Test Problem 3 using the Newton's method.



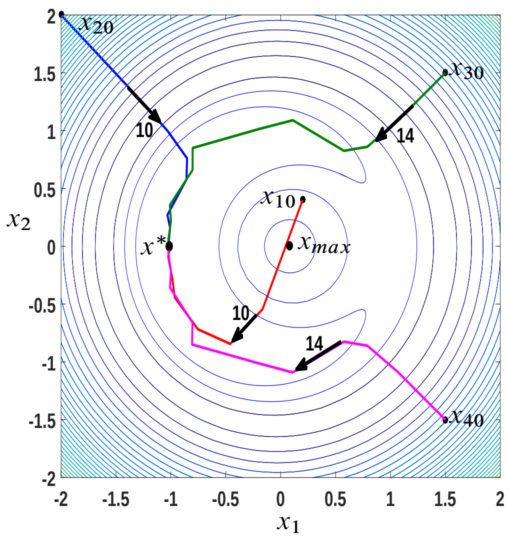
(c) Phase portrait of Test Problem 3 using the GN method.



(d) Phase portrait of Test Problem 3 using the LM method.



(e) Phase portrait of Test Problem 3 using the AGD method.



(f) Phase portrait of Test Problem 3 using the AGDN method.

Figure 4.7. The phase portraits of Test Problem 3 using the six numerical methods for four initial points where $x^* = [-1.120, 0]$. The direction of the trajectories are shown by arrows and the numbers beside each arrow denote the number of iterations used to converge to x^*.

Figures 4.8 and 4.9 shows the comparisons between the number of iterations and the CPU times when the six different numerical methods are applied to solve Test Problem 3 for four initial points.

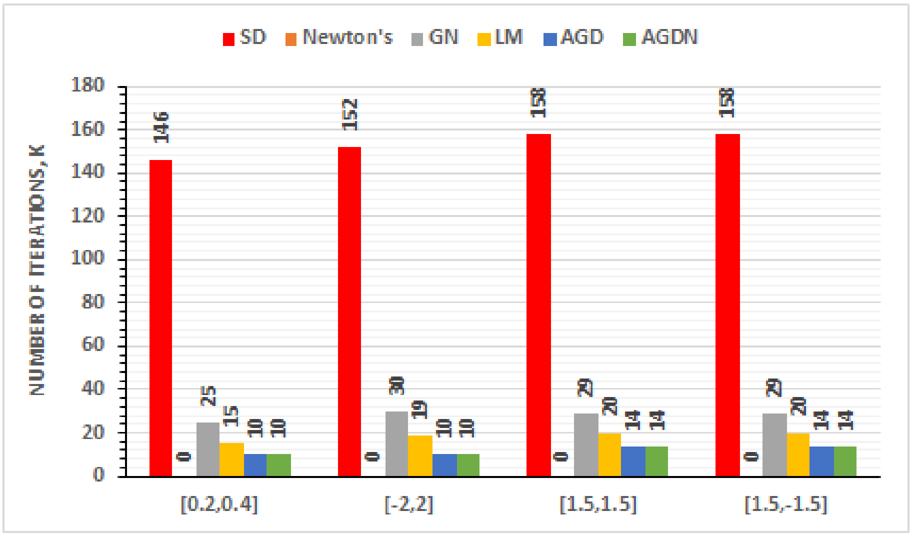


Figure 4.8. A comparison between the number of iterations for six numerical methods using four initial points; i.e. $x_{10} = [0.2, 0.4]$, $x_{20} = [-2, 2]$, $x_{30} = [1.5, 1.5]$ and $x_{40} = [1.5, -1.5]$ for Test Problem 3.

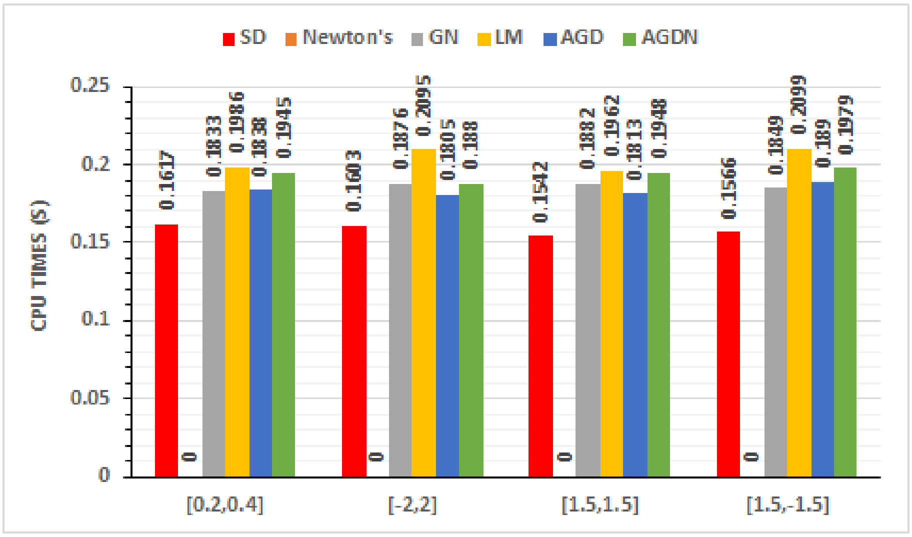


Figure 4.9. A comparison between the CPU times for six numerical methods using four initial points; i.e. $x_{10} = [0.2, 0.4]$, $x_{20} = [-2, 2]$, $x_{30} = [1.5, 1.5]$ and $x_{40} = [1.5, -1.5]$ for Test Problem 3.

From Figure 4.7, observe that the function of Test Problem 3 has properly nested sets and hence a method applied to solve it must converge to the minimum point x^*. However, from Table 4.4 and Figure 4.7(b), notice that the Newton's method fails to converge when it is applied to solve Test Problem 3 for all the four chosen initial points since the maximum number of iterations is reached before $\|g_k\| < 10^{-6}$. These failures of the Newton's method may be due to the almost singularity of the Hessian matrix.

From Figure 4.8, observe that the SD method requires the highest number of iterations to reach convergence but with the fastest convergence rate as depicted in Figure 4.9. This is because the SD method only requires the evaluation of the first derivatives for each iteration which has a relatively low computational cost. This situation is also observed for Test Problem 2.

Meanwhile, the computational times of the the GN, the LM, the AGD and the AGDN methods are comparable as shown in Figure 4.9. However, the trajectories of the LM, the AGD and the AGDN methods behave more steadily than those of the GN method and hence fewer number of iterations are required to reach x^*. Conversely, similar to the previous test problems, the LM method has the slowest rate of convergence despite the small number of iterations it requires to reach convergence as illustrated in Figure 4.9.

Furthermore, from Figure 4.7, it is important to note that the function of Test Problem 3 has a minimum point $x^* = [-1.012, 0]$ and a maximum point $x_{max} = [0.07948, 0]$. However, all the numerical iterations have shown the ability to jump over the global maximum point x_{max} in order to avoid an undesirable convergence towards a maximum point. This is also observed in Figure 4.7(b) for the Newton's method where all its phase trajectories just passed by the maximum point x_{max}. This observation shows the importance of incorporating the Lyapunov function theorem as convergence analysis where only a sufficient decrease in the objective function is required to ensure the convergence of a numerical method towards a minimum point.

Table 4.5 records the experimental data when the six numerical methods are applied to solve Test Problem 4. Since the function of Test Problem 4 has multiple minimum points (i.e. one global minimum point and two local minimum points), five initial points are chosen to test the numerical methods.

Table 4.5. Record of experimental results for Test Problem 4.

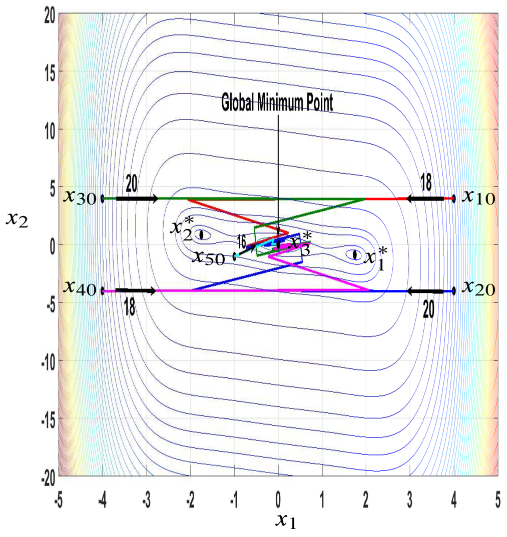
TP No.	Method	x_{j0}; $j = 1, 2 \dots$	k	$F(x^*)$	CPU Times (s)
4.	SD	$[4, 4]$	18	2.51×10^{-14}	0.1515
		$[4, -4]$	20	2.46×10^{-7}	0.1449
		$[-4, 4]$	20	1.23×10^{-14}	0.1478
		$[-4, -4]$	18	2.51×10^{-14}	0.1471
		$[-1, -1]$	16	4.73×10^{-14}	0.1462
	Newton's	$[4, 4]$	FAILED		
		$[4, -4]$	5	0.2986	0.1404
		$[-4, 4]$	5	0.2986	0.1416
		$[-4, -4]$	FAILED		
		$[-1, -1]$	FAILED		
	GN	$[4, 4]$	28	0.2986	0.3292
		$[4, -4]$	28	0.2986	0.3205
		$[-4, 4]$	28	0.2986	0.3043
		$[-4, -4]$	28	0.2986	0.2896
		$[-1, -1]$	34	0.2986	0.3912
	LM	$[4, 4]$	76	0.2986	0.2338
		$[4, -4]$	80	0.2986	0.2275
		$[-4, 4]$	80	0.2986	0.2271
		$[-4, -4]$	76	0.2986	0.2255
		$[-1, -1]$	FAILED		
	AGD	$[4, 4]$	9	0.2986	0.1851
		$[4, -4]$	9	0.2986	0.1772
		$[-4, 4]$	9	0.2986	0.1797
		$[-4, -4]$	9	0.2986	0.1764
		$[-1, -1]$	6	4.14×10^{-14}	0.1832
	AGDN	$[4, 4]$	9	0.2986	0.1950

Continued on next page

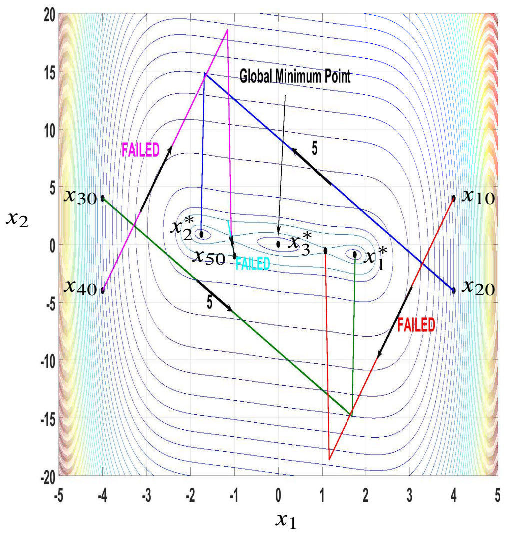
Table 4.5 – *Continued from previous page*

TP No.	Method	$x_{j0};\ j = 1,2...$	k	$F(x^*)$	CPU Times (s)
4.	AGDN	$[4,-4]$	9	0.2986	0.1908
		$[-4,4]$	9	0.2986	0.1888
		$[-4,-4]$	9	0.2986	0.1847
		$[-1,-1]$	6	4.14×10^{-14}	0.1805

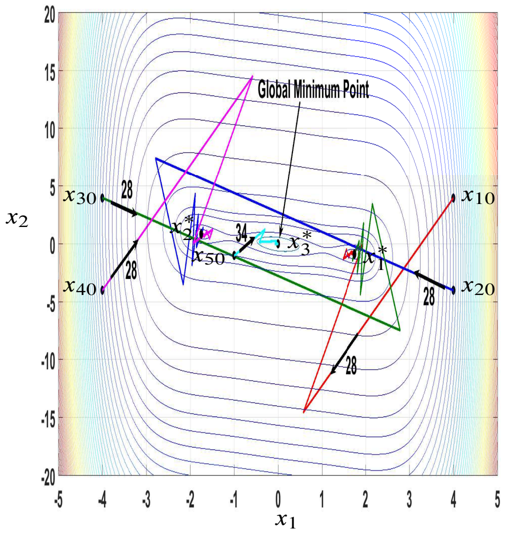
Figures 4.10 (a)–(f) illustrate the phase portrait of Test Problem 4 when the six numerical methods are used to solve it. Since the function of Test Problem 4 has multiple minimum points that are close to each other, the stiff ODE method is applied to plot the missing level curves surrounding these minimum points.



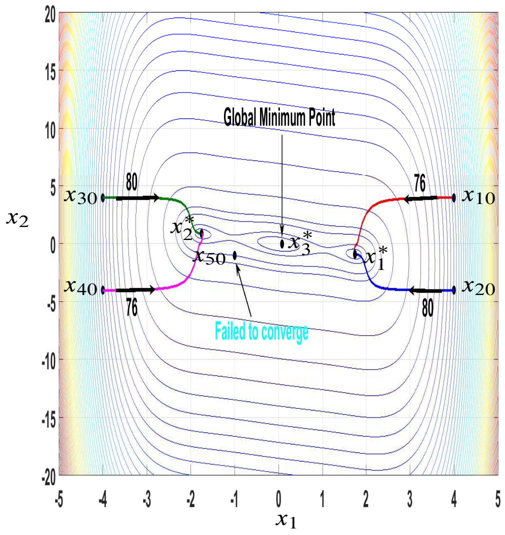
(a) Phase portrait of Test Problem 4 using the SD method.



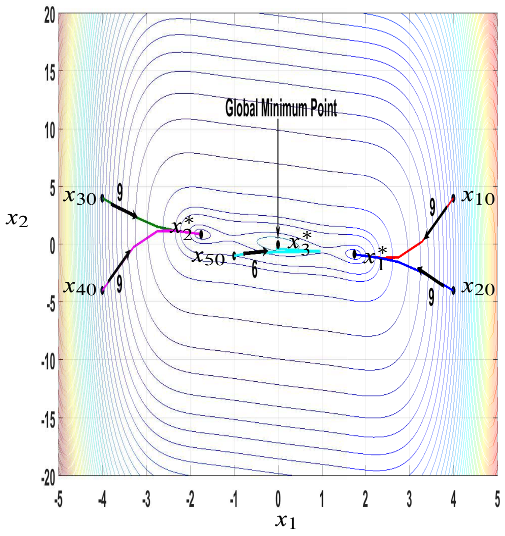
(b) Phase portrait of Test Problem 4 using the Newton's method.



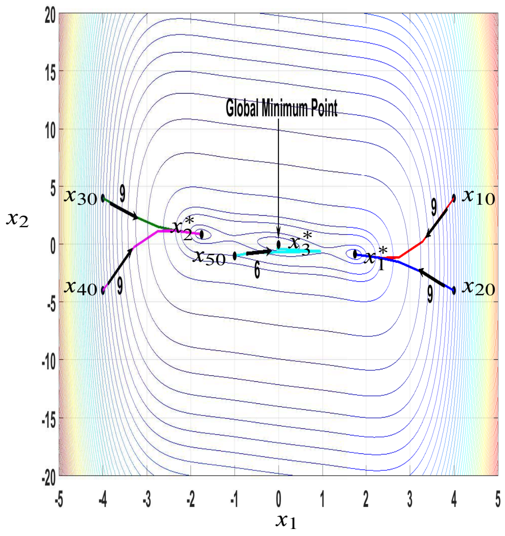
(c) Phase portrait of Test Problem 4 using the GN method.



(d) Phase portrait of Test Problem 4 using the LM method.



(e) Phase portrait of Test Problem 4 using the AGD method.



(f) Phase portrait of Test Problem 4 using the AGDN method.

Figure 4.10. The phase portraits of Test Problem 4 that are generated when the six numerical methods are applied to solve it for five initial points. The two local minimum points are given by $x_1^* = [1.7476, -0.87378]$ and $x_2^* = [-1.7476, 0.87378]$ while the global minimum point is $x_3^* = [0, 0]$. The direction of the trajectories are shown by black arrows and the numbers beside each arrow denote the number of iterations required to reach x^*.

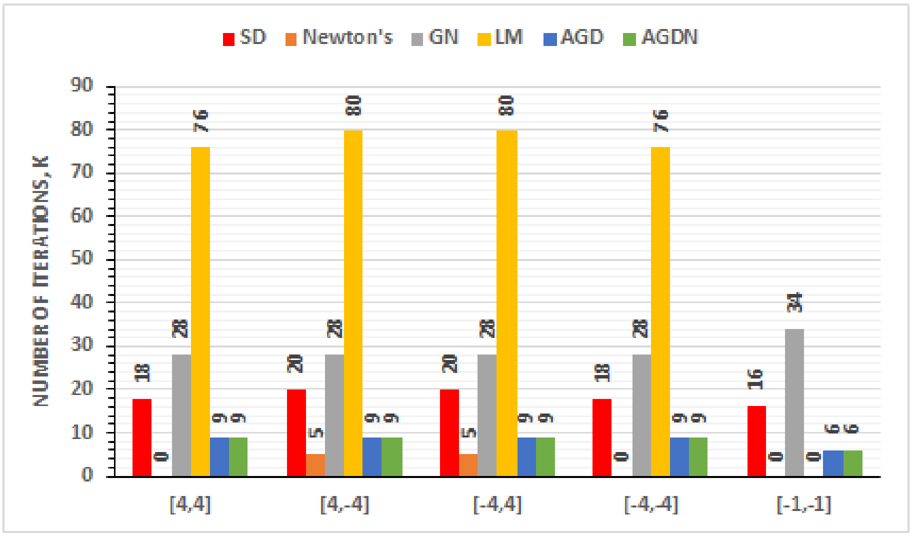


Figure 4.11. A comparison between the number of iterations required when the six numerical methods are applied to solve Test Problem 4 using five initial points; i.e. $x_{10} = [4,4]$, $x_{20} = [4,-4]$, $x_{30} = [-4,4]$, $x_{40} = [-4,-4]$ and $x_{50} = [-1,-1]$.

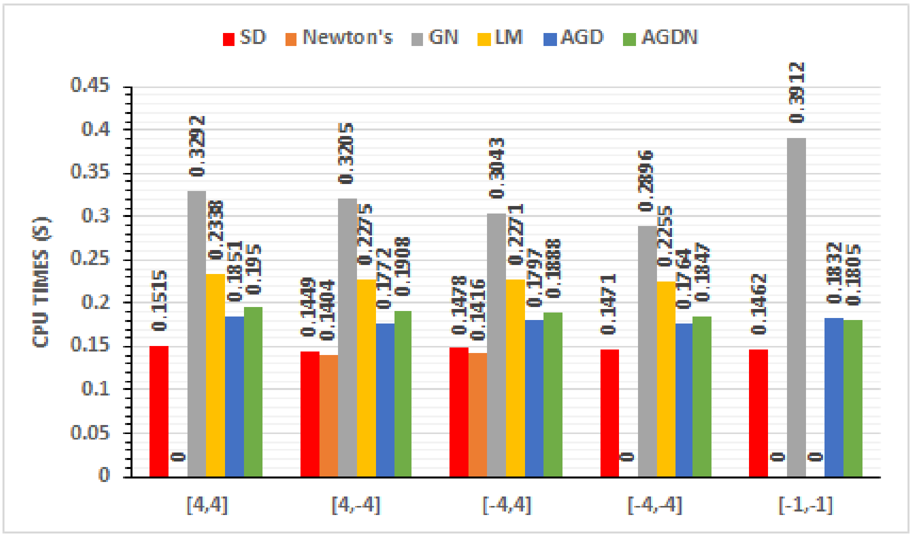


Figure 4.12. A comparison between the CPU times required for convergence when the six numerical methods is applied to solve Test Problem 4 using five initial points; i.e. $x_{10} = [4,4]$, $x_{20} = [4,-4]$, $x_{30} = [-4,4]$, $x_{40} = [-4,-4]$ and $x_{50} = [-1,-1]$.

From Table 4.5 and Figure 4.10, observe that all the numerical methods show convergence towards either of the minimum points except for the Newton's and the LM methods. The Newton's method fails to converge for three of the chosen initial points;

i.e. $x_{10} = [4, 4]$, $x_{40} = [-4, -4]$ and $x_{50} = [-1, -1]$ while the LM method fails to converge for x_{50}. These failures of the Newton's method may be due to the almost singularity of the Hessian matrix during the iterative process. However, the Newton's method is able to converge to either of the local minimum points for the other two initial points. On the other hand, the failure of the LM method for x_{50} may indicate that the truncated Hessian matrix is an inadequate approximation of the actual Hessian matrix due to the presence of large residuals during the iterative process. As a result, the GN and the LM methods, which use a truncated Hessian matrix in their iterative equations, have the slowest convergence rates as shown in Figure 4.12. With the presence of large residuals, the convergence rates of the GN and the LM methods are only linear. Meanwhile, the SD method, which shows global convergence for all the five initial points, has the best convergence rate among all methods in general.

Both the AGD and the AGDN methods outperform all other methods since convergence is achieved for all the five initial points and their phase trajectories behave very steadily before approaching the minimum points with a few number of iterations. Similar to the previous test problems, the SD and the GN methods tend to generate phase trajectories with erratic behaviours before converging to x^*.

An important feature displayed by Test Problem 4 is that for an objective function with multiple minimum points, convergence towards the minimum points from an initial point closest to it is not guaranteed (see Figures 4.10(a)–(c)). This feature has already been mentioned before in Section (2.1) of Chapter 2.

All the numerical methods fail to work for all initial points of Test Problem 5. This is because the function of Test Problem 5; i.e. the Brown badly scaled function, does not have properly nested level sets and hence convergence is not guaranteed. The level sets of this function are illustrated in Figure 4.13. Furthermore, it was found that its Hessian matrix is tridiagonal.

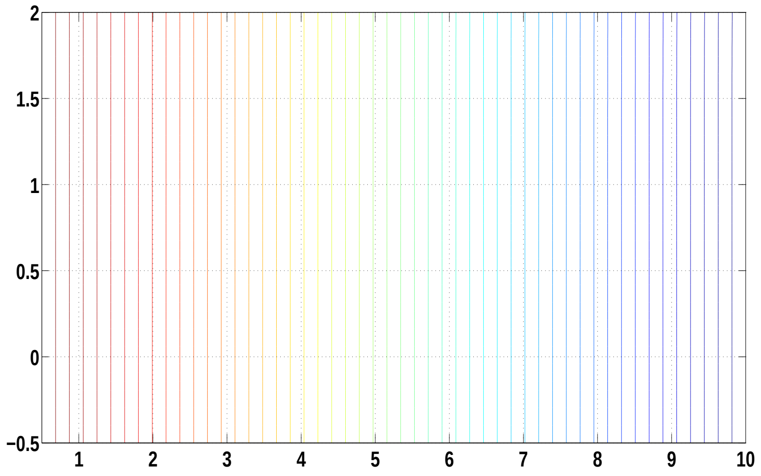


Figure 4.13. The level sets of Brown badly scaled function (i.e. Test Problem 5). Observe that this function does not have properly nested level sets.

Table 4.6 and Figures 4.14–4.15 show the numerical results for Test Problem 6.

Table 4.6. Record of experimental results for Test Problem 6.

TP No.	**Method**	$x_{j0};\ j = 1, 2 \dots$	k	$F(x^*)$	**CPU Times (s)**
6.	SD	$[4, 4]$	534	3.94×10^{-15}	0.2153
		$[5, -6]$	672	3.43×10^{-15}	0.2182
		$[-4, 8]$	560	3.04×10^{-15}	0.2057
		$[-5, -8]$	653	3.76×10^{-15}	0.2150
	Newton’s	$[4, 4]$	7	7.67×10^{-29}	0.1522
		$[5, -6]$	8	6.42×10^{-29}	0.1415
		$[-4, 8]$	8	7.29×10^{-29}	0.1415
		$[-5, -8]$	7	1.54×10^{-23}	0.1405
	GN	$[4, 4]$	FAILED		
		$[5, -6]$	FAILED		
		$[-4, 8]$	FAILED		
		$[-5, -8]$	FAILED		

Continued on next page

Table 4.6 – *Continued from previous page*

TP No.	Method	$x_{j0};\ j = 1, 2 \ldots$	k	$F(x^*)$	CPU Times (s)
6.	LM	$[4, 4]$	15	1.15×10^{-16}	0.1750
		$[5, -6]$	14	2.76×10^{-17}	0.1763
		$[-4, 8]$	16	2.95×10^{-15}	0.1705
		$[-5, -8]$	14	5.26×10^{-17}	0.1637
	AGD	$[4, 4]$	9	2.92×10^{-28}	0.1541
		$[5, -6]$	9	8.51×10^{-28}	0.1476
		$[-4, 8]$	7	6.14×10^{-29}	0.1424
		$[-5, -8]$	11	1.67×10^{-27}	0.1513
	AGDN	$[4, 4]$	9	2.92×10^{-28}	0.1662
		$[5, -6]$	9	8.52×10^{-28}	0.1628
		$[-4, 8]$	7	6.14×10^{-29}	0.1607
		$[-5, -8]$	11	1.67×10^{-27}	0.1621

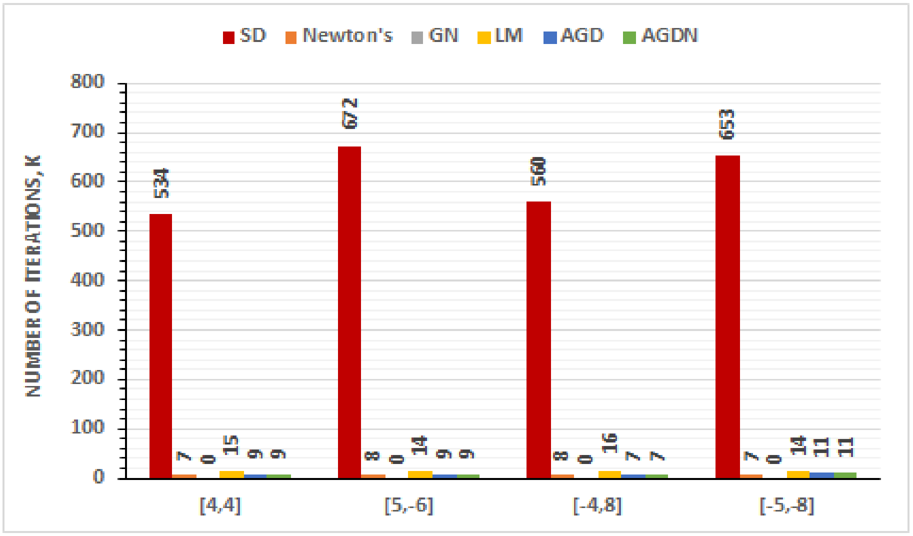


Figure 4.14. A comparison between the number of iterations required when the six numerical methods are applied to solve Test Problem 6 using four initial points; i.e. $x_{10} = [4, 4]$, $x_{20} = [5, -6]$, $x_{30} = [-4, 8]$ and $x_{40} = [-5, -8]$.

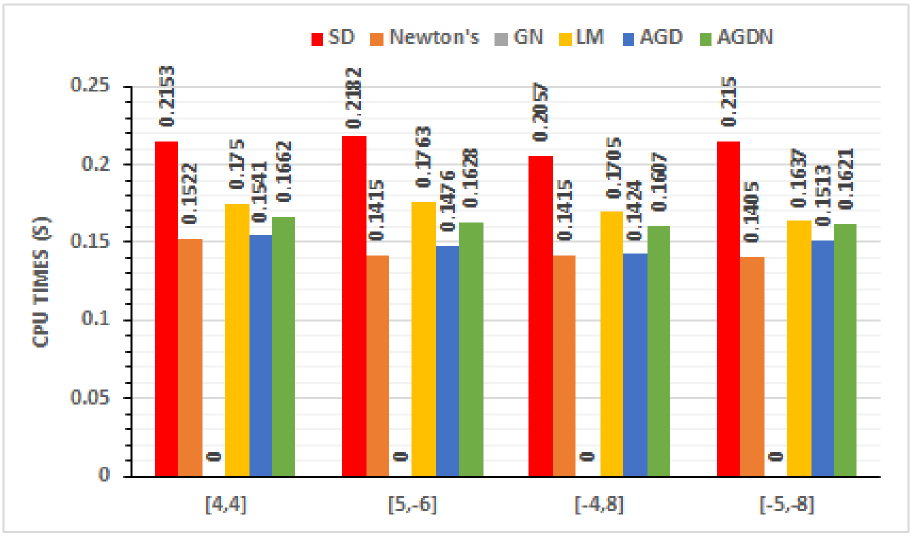


Figure 4.15. A comparison between the CPU times required for convergence when the six numerical methods is applied to solve Test Problem 6 using four initial points; i.e. $x_{10} = [4, 4]$, $x_{20} = [5, -6]$, $x_{30} = [-4, 8]$ and $x_{40} = [-5, -8]$.

As discussed in Section (2.5.1) of Chapter 2, the SD method creates zigzag iterations towards the minimum point x^*. This behaviour, which requires a high number of iterative steps, can be seen in Table 4.6 and Figure 4.14 when it is applied to solve Test Problem 6. From Figure 4.14, it is obvious that the SD method possesses a much higher iteration number when compared with the other methods. Despite this large number of iterations, the SD iterative process is comparatively faster than these methods. It can be seen from Figure 4.15 that a comparatively shorter amount of time is required to compute a very large number of iterations. This is because the cost of computation per iteration is relatively low for this method since it only requires the evaluation of the first derivatives. This situation is also obvious when the SD method is applied to solve Test Problem 3. Following the SD method, the LM method is ranked the second slowest in terms of convergence rates.

On the other hand, the Newton's method exhibits the fastest convergence rates among all the methods as depicted in Figure 4.15. This can be seen from the relatively small number of iterations k required to reach the minimum point x^* in a very short amount of time. This is because the Newton's method exhibits a fast quadratic convergence rate as discussed in Section (2.5.2) of Chapter 2.

Meanwhile, the GN method is declared as a failure when it is applied to solve Test Problem 6 as recorded in Table 4.6. For all the four initial points, the iteration limit 50000 is reached before the gradients reach 10^{-6}. In other words, if a higher iteration limit is allowed, convergence may be achieved by the GN method.

Similar to the Newton's method, both the AGD method and the AGDN method have shown comparatively good experimental results when they are applied to solve Test Function 6. This is because these methods either merge with or switch to the Newton's method near the minimum point x^* and hence they are able to produce fast quadratic convergence rates in their final iterative processes.

Table 4.7 records the experimental data obtained when the numerical methods are applied to solve Test Problem 7. Following this, two graphs are plotted to compare between the number of iterations and the CPU times among these methods.

Table 4.7. Record of experimental results for Test Problem 7.

TP No.	**Method**	$\boldsymbol{x_{j0};\ j = 1, 2 \ldots}$	$\boldsymbol{k}$	$\boldsymbol{F(x^*)}$	**CPU Times (s)**
7.	SD	$[4, 4]$	539	0.5	0.2167
		$[5, -6]$	644	0.5	0.2168
		$[-4, 8]$	563	0.5	0.2063
		$[-5, -8]$	683	0.5	0.2204
	Newton's	$[4, 4]$	7	0.5	0.1159
		$[5, -6]$	8	0.5	0.1186
		$[-4, 8]$	8	0.5	0.1149
		$[-5, -8]$	7	0.5	0.1188
	GN	$[4, 4]$	7	0.5	0.2465
		$[5, -6]$	297	0.5	0.3320
		$[-4, 8]$	1825	0.5	0.7388
		$[-5, -8]$	342	0.5	0.3431
	LM	$[4, 4]$	15	0.5	0.1877

Continued on next page

Table 4.7 – *Continued from previous page*

TP No.	**Method**	$x_{j0};\ j = 1, 2\ldots$	k	$F(x^*)$	**CPU Times (s)**
7.	LM	$[5, -6]$	14	0.5	0.1696
		$[-4, 8]$	16	0.5	0.1718
		$[-5, -8]$	14	0.5	0.1797
	AGD	$[4, 4]$	9	0.5	0.1527
		$[5, -6]$	9	0.5	0.1446
		$[-4, 8]$	7	0.5	0.1396
		$[-5, -8]$	11	0.5	0.1486
	AGDN	$[4, 4]$	9	0.5	0.1581
		$[5, -6]$	9	0.5	0.1680
		$[-4, 8]$	7	0.5	0.1684
		$[-5, -8]$	11	0.5	0.1783

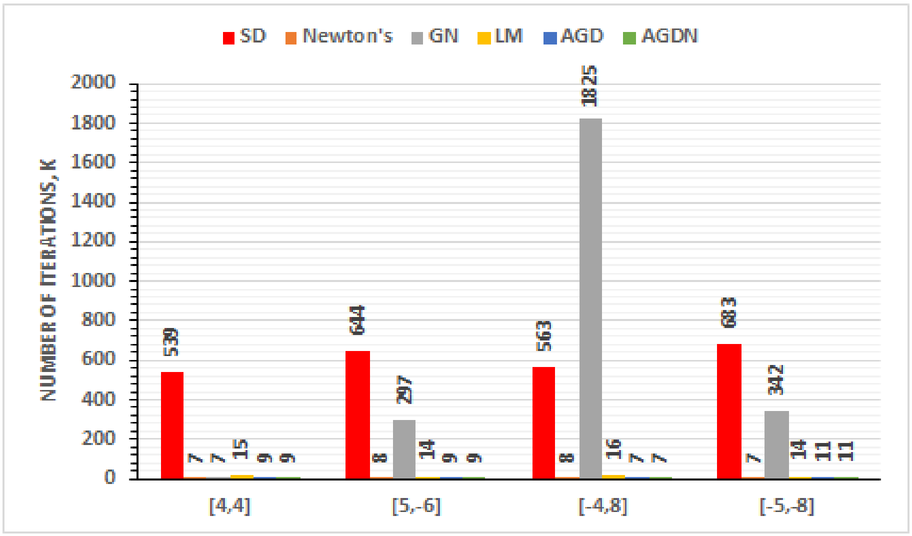


Figure 4.16. A comparison between the number of iterations required when the six numerical methods are applied to solve Test Problem 7 using four initial points; i.e. $x_{10} = [4, 4]$, $x_{20} = [5, -6]$, $x_{30} = [-4, 8]$ and $x_{40} = [-5, -8]$.

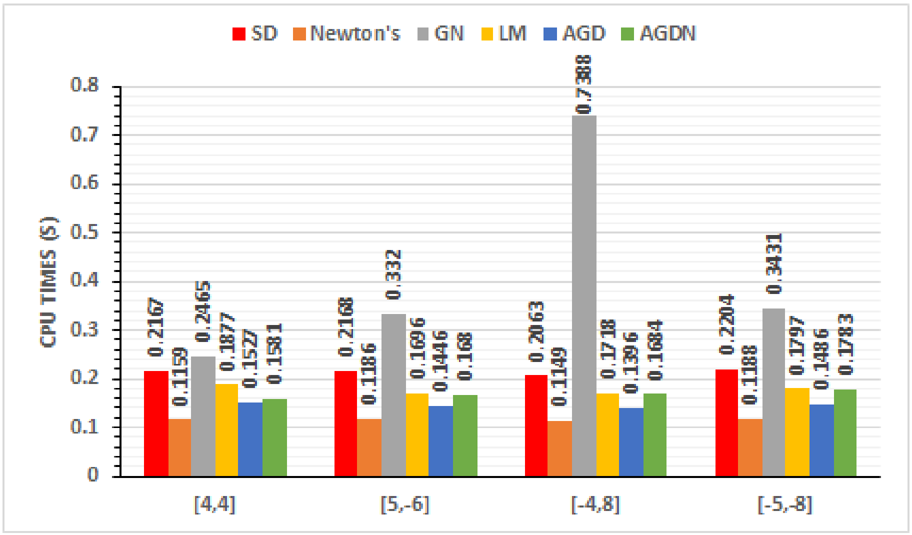


Figure 4.17. A comparison between the CPU times required for convergence when the six numerical methods is applied to solve Test Problem 7 using four initial points; i.e. $x_{10} = [4,4]$, $x_{20} = [5,-6]$, $x_{30} = [-4,8]$ and $x_{40} = [-5,-8]$.

From Table 4.7, it can be seen that all the numerical methods converge to the minimum point x^* when they are applied to solve Test Problem 7. Furthermore, from Figures 4.16 and 4.17, observe that the Newton's method outperforms all the other numerical methods in terms of number of iterations and the CPU times required to achieve convergence. In contrast, regardless of the number of iterations, the GN method has the slowest convergence rates among all the methods. The numerical results obtained from the SD, the LM, the AGD and the AGDN methods are comparable in terms of convergence rates as depicted in Figure 4.17.

Table 4.8 and Figures 4.18–4.19 record and illustrate the numerical results for Test Problem 8.

Table 4.8. Record of experimental results for Test Problem 8.

TP No.	Method	$x_{j0};\ j = 1,2\ldots$	k	$F(x^*)$	CPU Times (s)
8.	SD	$[-1.2, 1]$	58	0.6513	0.1184
		$[2, 2]$	59	0.6513	0.1186

Continued on next page

Table 4.8 – *Continued from previous page*

TP No.	Method	$x_{j0};\ j = 1, 2 \ldots$	k	$F(x^*)$	CPU Times (s)
8.	SD	$[-2, -3]$	61	0.6513	0.1318
		$[2, -3]$	60	0.6513	0.1217
	Newton's	$[-1.2, 1]$	5	0.6513	0.1066
		$[2, 2]$	8	0.6513	0.1216
		$[-2, -3]$	8	0.6513	0.1211
		$[2, -3]$	6	0.6513	0.1164
	GN	$[-1.2, 1]$	20	0.6513	0.2483
		$[2, 2]$	19	0.6513	0.2477
		$[-2, -3]$	20	0.6513	0.2515
		$[2, -3]$	20	0.6513	0.2631
	LM	$[-1.2, 1]$	10	0.6513	0.1682
		$[2, 2]$	9	0.6513	0.1620
		$[-2, -3]$	11	0.6513	0.1659
		$[2, -3]$	18	0.6513	0.1664
	AGD	$[-1.2, 1]$	6	0.6513	0.1442
		$[2, 2]$	7	0.6513	0.1398
		$[-2, -3]$	6	0.6513	0.1451
		$[2, -3]$	7	0.6513	0.1387
	AGDN	$[-1.2, 1]$	6	0.6513	0.1568
		$[2, 2]$	7	0.6513	0.1536
		$[-2, -3]$	6	0.6513	0.1489
		$[2, -3]$	7	0.6513	0.1586

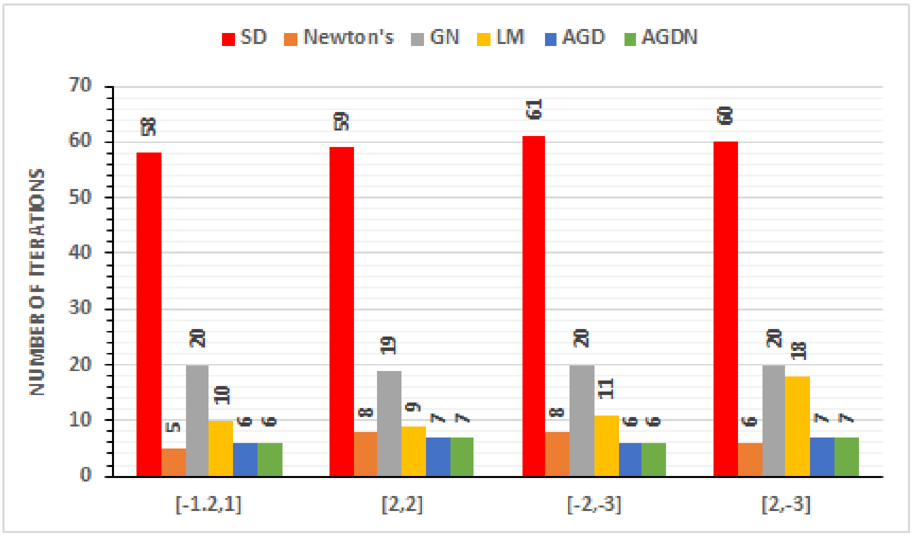


Figure 4.18. A comparison between the number of iterations required when the six numerical methods are applied to solve Test Problem 8 using four initial points; i.e. $x_{10} = [-1.2, 1]$, $x_{20} = [2, 2]$, $x_{30} = [-2, -3]$ and $x_{40} = [2, -3]$.

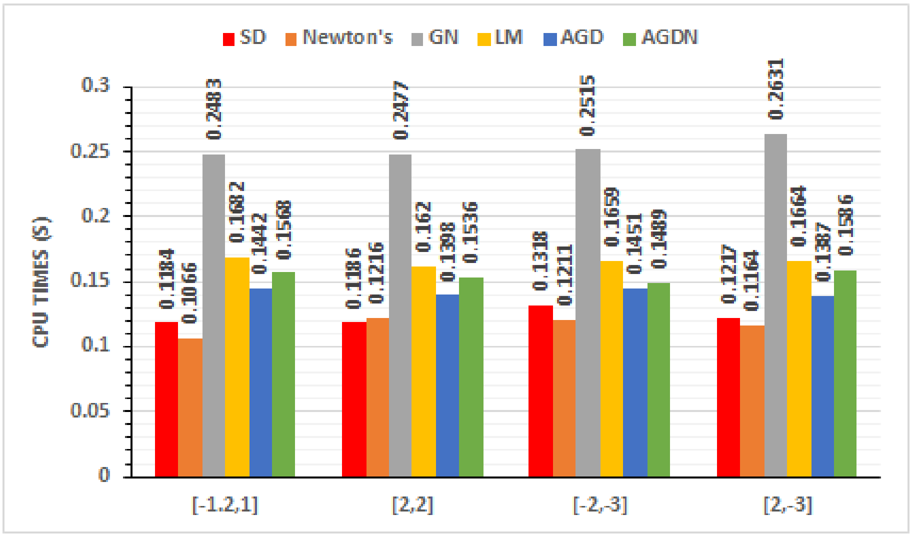


Figure 4.19. A comparison between the CPU times required for convergence when the six numerical methods is applied to solve Test Problem 8 using four initial points; i.e. $x_{10} = [-1.2, 1]$, $x_{20} = [2, 2]$, $x_{30} = [-2, -3]$ and $x_{40} = [2, -3]$.

From Appendix A, note that Test Problem 8 is a weaker version of Test Problems 9 and 10 since the coefficient of the residual functions are $a = b = c = 1$ only. Hence, it is not surprising that all the numerical methods work well for Test Problem 8 as shown in Table 4.1. From Figures 4.18 and 4,19, it can be seen that the SD method has a very fast convergence rate regardless of its high number of iterations. This fast convergence of the SD method is due to its cost-effective requirement of computing the first derivatives of the objective function only at every iterative step.

In addition, the numerical results also shown that the Newton's method has the fastest rate of convergence with a few number of iterations due to its fast quadratic convergence rates. Conversely, the GN and the LM methods exhibit the slowest convergence rates among all the methods despite their low number of iterations. This may indicate the presence of large residuals during the iterative processes and thus the truncated Hessian matrix is an inadequate approximation to the actual Hessian matrix. Similar to the previous test problems, the AGD and the AGDN methods show similar results but the rate of convergence of the AGD method is slightly faster compared to the AGDN method.

Table 4.9 and Figures 4.20–4.21 record and illustrate the numerical results for Test Problem 9.

Table 4.9. Record of experimental results for Test Problem 9.

TP No.	**Method**	$x_{j0};\ j = 1, 2\ldots$	k	$F(x^*)$	**CPU Times (s)**
9.	SD	$[-1.2, 1]$	132	1.9467	0.1286
		$[2, 2]$	143	1.9467	0.1309
		$[-2, -3]$	107	1.9467	0.1283
		$[2, -3]$	89	1.9467	0.1277
	Newton's	$[-1.2, 1]$	6	1.9467	0.1153
		$[2, 2]$	10	1.9467	0.1226
		$[-2, -3]$	11	1.9467	0.1254
		$[2, -3]$	7	1.9467	0.1204
	GN	$[-1.2, 1]$	61	1.9467	0.3044
		$[2, 2]$	58	1.9467	0.2970
		$[-2, -3]$	49	1.9467	0.2632

Continued on next page

Table 4.9 – *Continued from previous page*

TP No.	Method	$x_{j0};\ j = 1, 2 \ldots$	k	$F(x^*)$	CPU Times (s)
9.	GN	$[2, -3]$	47	1.9467	0.2623
	LM	$[-1.2, 1]$	14	1.9467	0.1984
		$[2, 2]$	15	1.9467	0.1952
		$[-2, -3]$	18	1.9467	0.1903
		$[2, -3]$	18	1.9467	0.1995
	AGD	$[-1.2, 1]$	10	1.9467	0.1509
		$[2, 2]$	10	1.9467	0.1512
		$[-2, -3]$	12	1.9467	0.1494
		$[2, -3]$	11	1.9467	0.1441
	AGDN	$[-1.2, 1]$	10	1.9467	0.1562
		$[2, 2]$	10	1.9467	0.1585
		$[-2, -3]$	12	1.9467	0.1606
		$[2, -3]$	11	1.9467	0.1614

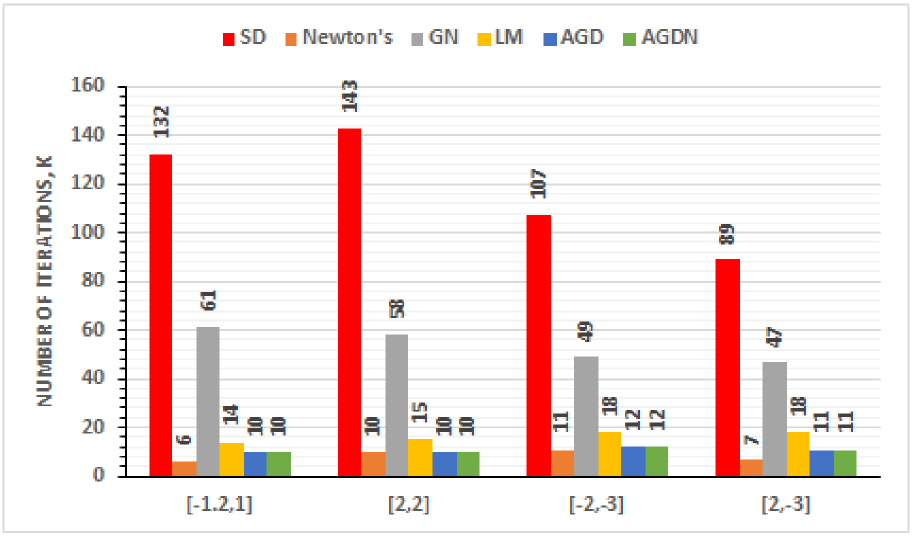


Figure 4.20. A comparison between the number of iterations required when the six numerical methods are applied to solve Test Problem 9 using four initial points; i.e. $x_{10} = [-1.2, 1]$, $x_{20} = [2, 2]$, $x_{30} = [-2, -3]$ and $x_{40} = [2, -3]$.

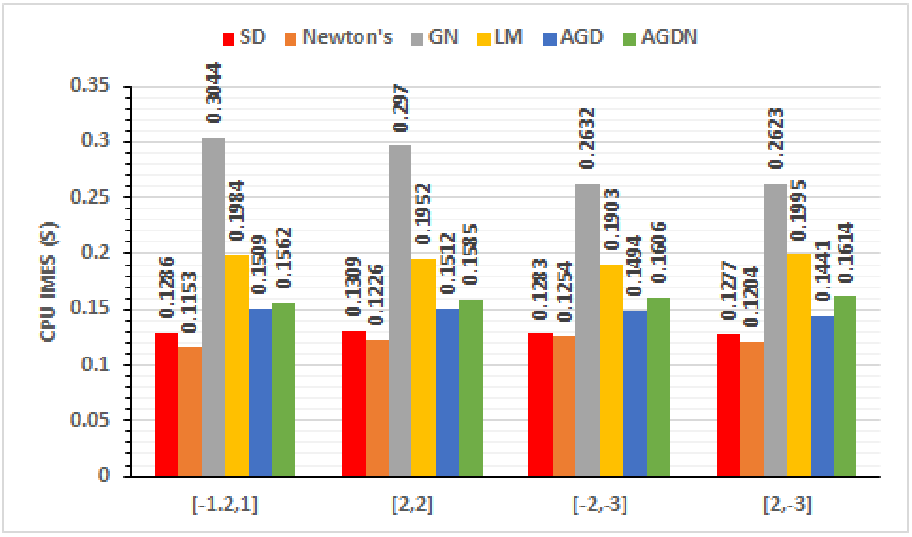


Figure 4.21. A comparison between the CPU times required for convergence when the six numerical methods is applied to solve Test Problem 9 using four initial points; i.e. $x_{10} = [-1.2, 1]$, $x_{20} = [2, 2]$, $x_{30} = [-2, -3]$ and $x_{40} = [2, -3]$.

By comparing the experimental results of Test Problems 8 and 9, notice that the iteration numbers and the CPU times required by each method increase as the parameter value of a increases from 1 to 10. This indicates that Test Problem 9 is a harder problem to solve compared to Test Problem 8. The increase in number of iterations are particularly obvious for the SD and the GN methods. The Newton's, the LM, the AGD and the AGDN methods only show a slight increase in the number of iterations and the CPU times which suggest that these methods are robust to parameter variations and are reliable in terms of time. As usual, the GN method requires the highest number of iterations and the longest amount of time to reach x^* as shown in Figure 4.20 and 4.21. Conversely, the Newton's method shows the best results with the fastest convergence rates and the least number of iterations. Generally, it can be observed from Figures 4.18–4.21 that all numerical methods show similar results when they are applied to solve Test Problems 8 and 9.

Table 4.10 and Figures 4.22–4.23 record and illustrate the numerical results for Test Problem 10.

Table 4.10. Record of experimental results for Test Problem 10.

TP No.	**Method**	$x_{j0};\ j = 1, 2 \ldots$	k	$F(x^*)$	**CPU Times (s)**
10.	SD	$[-1.2, 1]$	FAILED		
		$[2, 2]$	FAILED		
		$[-2, -3]$	FAILED		
		$[2, -3]$	FAILED		
	Newton's	$[-1.2, 1]$	FAILED		
		$[2, 2]$	15	5.9771	0.1272
		$[-2, -3]$	21	5.9771	0.1245
		$[2, -3]$	15	5.9771	0.1330
	GN	$[-1.2, 1]$	28	5.9771	0.2812
		$[2, 2]$	15	5.9771	0.2515
		$[-2, -3]$	33	5.9771	0.2661
		$[2, -3]$	36	5.9771	0.2656
	LM	$[-1.2, 1]$	24	5.9771	0.1926
		$[2, 2]$	21	5.9771	0.1884

Continued on next page

Table 4.10 – *Continued from previous page*

TP No.	**Method**	$x_{j0};\ j = 1, 2 \ldots$	k	$F(x^*)$	**CPU Times (s)**
10.	LM	$[-2, -3]$	24	5.9771	0.2012
		$[2, -3]$	24	5.9771	0.2014
	AGD	$[-1.2, 1]$	18	5.9771	0.1524
		$[2, 2]$	13	5.9771	0.1510
		$[-2, -3]$	21	5.9771	0.1651
		$[2, -3]$	13	5.9771	0.1541
	AGDN	$[-1.2, 1]$	18	5.9771	0.1648
		$[2, 2]$	13	5.9771	0.1648
		$[-2, -3]$	21	5.9771	0.1786
		$[2, -3]$	13	5.9771	0.1677

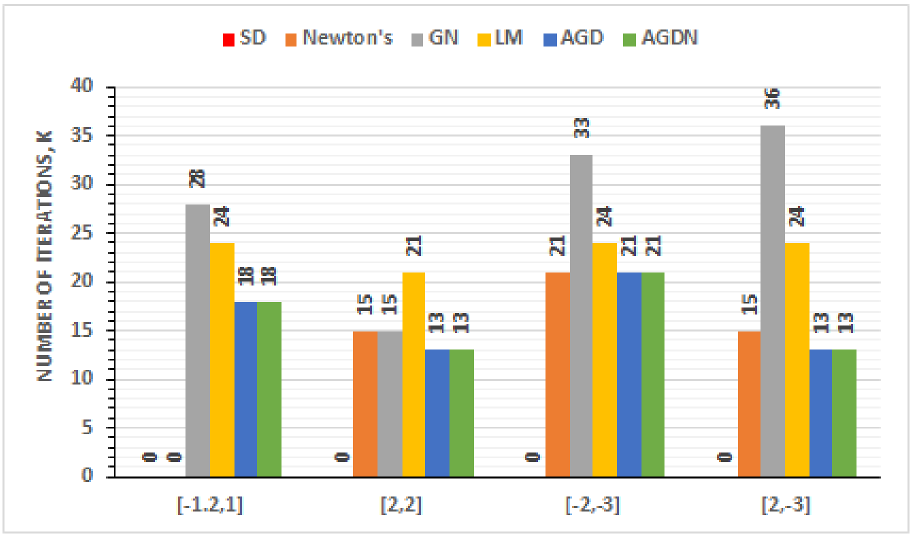


Figure 4.22. A comparison between the number of iterations required when the six numerical methods are applied to solve Test Problem 10 using four initial points; i.e. $x_{10} = [-1.2, 1]$, $x_{20} = [2, 2]$, $x_{30} = [-2, -3]$ and $x_{40} = [2, -3]$.

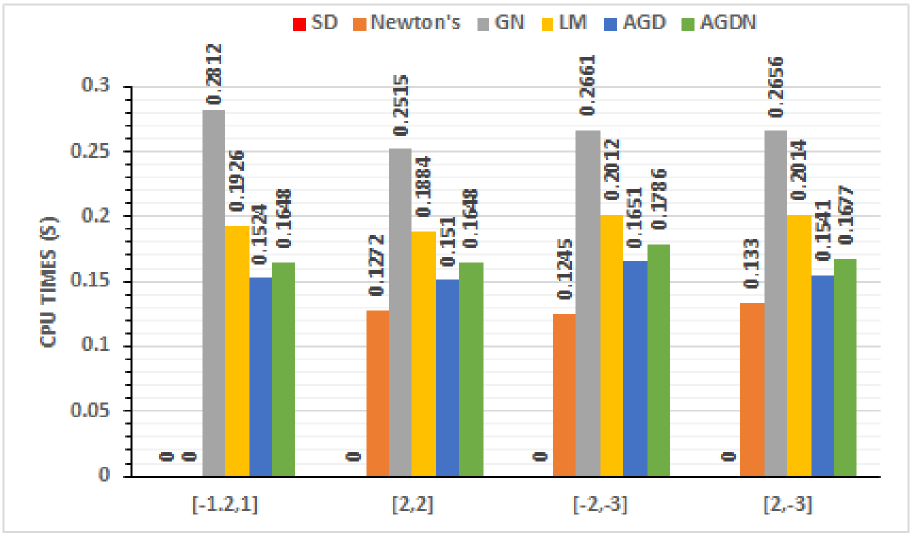


Figure 4.23. A comparison between the CPU times required for convergence when the six numerical methods is applied to solve Test Problem 10 using four initial points; i.e. $x_{10} = [-1.2, 1]$, $x_{20} = [2, 2]$, $x_{30} = [-2, -3]$ and $x_{40} = [2, -3]$.

The function of Test Problem 4 is considered as a severe test function of the original Rosenbrock function (see Goh (2009) and Goh and McDonald (2015)). In the original Rosenbrock function, we have $a = 10$, $b = 1$ and $c = 0$ for the coefficients of the residual functions while in Test Function 4, these coefficients are increased significantly to give $a = 100$, $b = 10$ and $c = 1$. Thus, the difficulty of the test problem has been increased remarkably. Therefore, it is not surprising that the SD method fails to work as recorded in Table 4.10. In fact, the SD method does converge to $x^* = [0.8493, 0.7203]$ for all the initial points after 50000 iterations. Since $\|g(x_{50000})\| > 10^{-6}$ for all initial points, it is declare as a failure. As usual, its phase trajectories are seen to exhibit slow zigzag behaviours towards x^*.

On the other hand, the Newton's method fails to converge with initial point $x_{10} = [-1.2, 1]$ due to the almost singularity of the Hessian matrix. This indicates that the Newton's method loses its robustness and reliability for large parameter variations. Unlike all other methods, the GN method has shown a significant reduction in the number of iterations when solving Test Problem 10 compared to Test Problem 9. However, similar to the previous test problems, it has the slowest rate of convergence with the highest number of iterations. This is followed by the LM method which has the second slowest rate of convergence and the second highest number of iterations.

Nevertheless, regardless of the severity of Test Problem 10, both the AGD and the AGDN methods still show good experimental results compared to other methods since they work well for all the chosen initial points with only a slight increase in the number of iterations and the CPU times for large parameter variations. Moreover, the phase portraits reveal that the trajectories of both methods still behave very steadying upon reaching x^* even though the parameters of the test problems are increased significantly. This shows the robustness, efficiency and reliability of the AGD and the AGDN methods despite the severity of the test problem.

Table 4.11 and Figures 4.24–4.25 record and illustrate the numerical results for Test Problem 11.

Table 4.11. Record of experimental results for Test Problem 11.

TP No.	**Method**	$x_{j0};\ j = 1, 2 \dots$	k	$F(x^*)$	**CPU Times (s)**
11.	SD	$[1, 1]$	740	1.52×10^{-12}	0.2683
		$[10, 2]$	FAILED		
		$[-5, -2]$	FAILED		
		$[8, -2]$	13613	1.71×10^{-12}	1.9849
	Newton's	$[1, 1]$	FAILED		
		$[10, 2]$	FAILED		
		$[-5, -2]$	FAILED		
		$[8, -2]$	FAILED		
	GN	$[1, 1]$	FAILED		
		$[10, 2]$	10	7.88×10^{-18}	0.3061
		$[-5, -2]$	FAILED		
		$[8, -2]$	FAILED		
		$[1, 1]$	13	1.82×10^{-13}	0.1984
		$[10, 2]$	30	1.91×10^{-13}	0.2032

Continued on next page

Table 4.11 – *Continued from previous page*

TP No.	Method	$x_{j0};\ j = 1, 2 \ldots$	k	$F(x^*)$	CPU Times (s)
11.	LM	$[-5, -2]$	FAILED		
		$[8, -2]$	26	3.19×10^{-13}	0.2061
	AGD	$[1, 1]$	7	5.67×10^{-19}	0.1614
		$[10, 2]$	37	1.01×10^{-19}	0.2053
		$[-5, -2]$	12	9.54×10^{-15}	0.1959
		$[8, -2]$	18	1.24×10^{-17}	0.2016
	AGDN	$[1, 1]$	7	4.19×10^{-19}	0.1899
		$[10, 2]$	37	1.01×10^{-19}	0.2383
		$[-5, -2]$	12	9.52×10^{-15}	0.2234
		$[8, -2]$	18	1.23×10^{-17}	0.2331

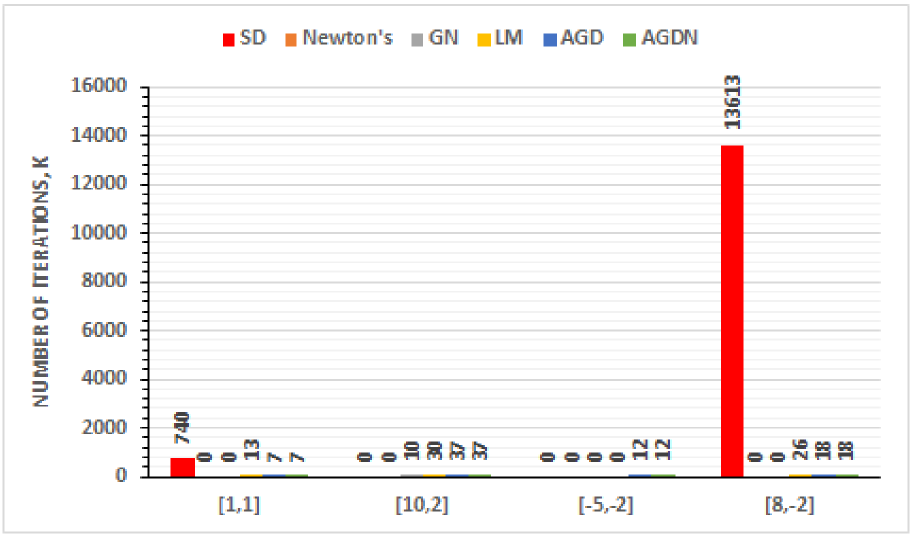


Figure 4.24. A comparison between the number of iterations required when the six numerical methods are applied to solve Test Problem 11 using four initial points; i.e. $x_{10} = [1, 1]$, $x_{20} = [10, 2]$, $x_{30} = [-5, -2]$ and $x_{40} = [8, -2]$.

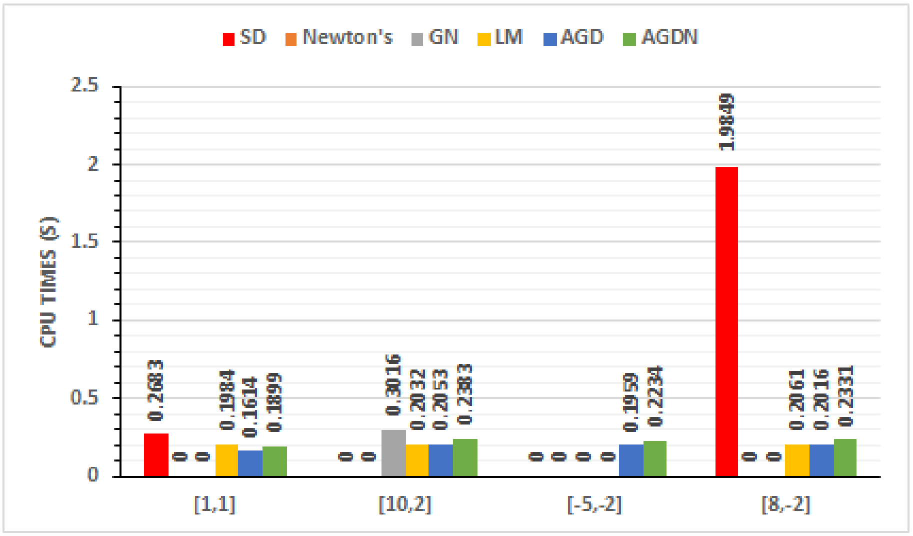


Figure 4.25. A comparison between the CPU times required for convergence when the six numerical methods is applied to solve Test Problem 11 using four initial points; i.e. $x_{10} = [1,1]$, $x_{20} = [10,2]$, $x_{30} = [-5,-2]$ and $x_{40} = [8,-2]$.

Test Problem 11 involves the solving of the Beale function with standard initial point at $x_{10} = [1,1]$; which is very close to the minimum point $x^* = [3,0.5]$. For this test problem, three initial points which are far away from the solution are also chosen to test the reliability of the numerical method. From Table 4.11, observe that both the SD and the LM methods converges to x^* for the standard initial point $[1,1]$. However, for initial points that are farther away from x^*, the SD method fails while the LM method converges for $x_{20} = [10,2]$ and $x_{40} = [8,-2]$.

Meanwhile, the GN method converges only for one out of four chosen initial points. This may be due to the almost singularity of the truncated Hessian matrix. Observe that even though the GN method fails to converge at the standard initial point $x_{10} = [1,1]$, the incorporation of a positive Lagrange parameter in the LM method overcomes the singularity of the truncated Hessian matrix and hence lead to convergence of the LM method at $x_{10} = [1,1]$. Furthermore, observe that the Newton's method fails to converge for all the chosen initial points which may be due to the almost singularity of the Hessian matrix.

Similar to all the previous test problems, both the AGD and the AGDN methods outperforms all other methods in terms of their capabilities to handle severe NLS test problem, the number of iterations and the rate of convergence. Again, the use of Test Problem 11 have further proven the efficiency, reliability and robustness of these methods.

Table 4.12 and Figures 4.26–4.27 record and illustrate the numerical results for Test Problem 12.

Table 4.12. Record of experimental results for Test Problem 12.

TP No.	**Method**	$x_{j0};\ j=1,2\dots$	k	$F(x^*)$	**CPU Times (s)**
12.	SD	$[0.3, 0.4]$	FAILED		
		$[-0.2, 0.4]$	FAILED		
		$[0.5, -0.1]$	FAILED		
		$[0, 0]$	FAILED		
	Newton's	$[0.3, 0.4]$	FAILED		
		$[-0.2, 0.4]$	FAILED		
		$[0.5, -0.1]$	FAILED		
		$[0, 0]$	FAILED		
	GN	$[0.3, 0.4]$	FAILED		
		$[-0.2, 0.4]$	FAILED		
		$[0.5, -0.1]$	FAILED		
		$[0, 0]$	FAILED		
	LM	$[0.3, 0.4]$	16	62.181	0.2106
		$[-0.2, 0.4]$	FAILED		
		$[0.5, -0.1]$	18	62.181	0.2079
		$[0, 0]$	16	62.181	0.2020
	AGD	$[0.3, 0.4]$	10	62.181	0.1811
		$[-0.2, 0.4]$	11	62.181	0.1786
		$[0.5, -0.1]$	13	62.181	0.1811
		$[0, 0]$	5	62.181	0.1739

Continued on next page

Table 4.12 – *Continued from previous page*

TP No.	**Method**	$x_{j0};\ j = 1, 2 \ldots$	k	$F(x^*)$	**CPU Times (s)**
12.	AGDN	$[0.3, 0.4]$	10	62.181	0.1981
		$[-0.2, 0.4]$	11	62.181	0.1966
		$[0.5, -0.1]$	13	62.181	0.2004
		$[0, 0]$	5	62.181	0.1924

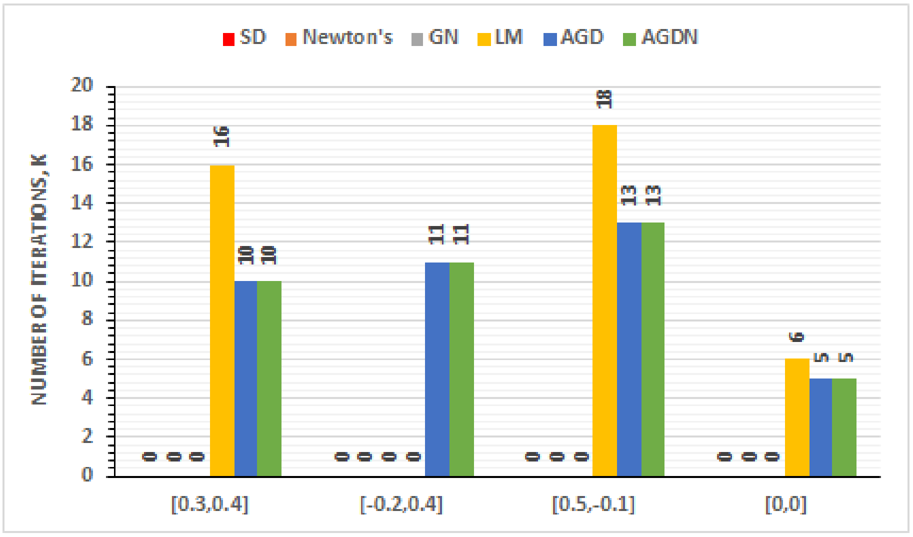


Figure 4.26. A comparison between the number of iterations required when the six numerical methods are applied to solve Test Problem 12 using four initial points; i.e. $x_{10} = [0.3, 0.4]$, $x_{20} = [-0.2, 0.4]$, $x_{30} = [0.5, -0.1]$ and $x_{40} = [0, 0]$.

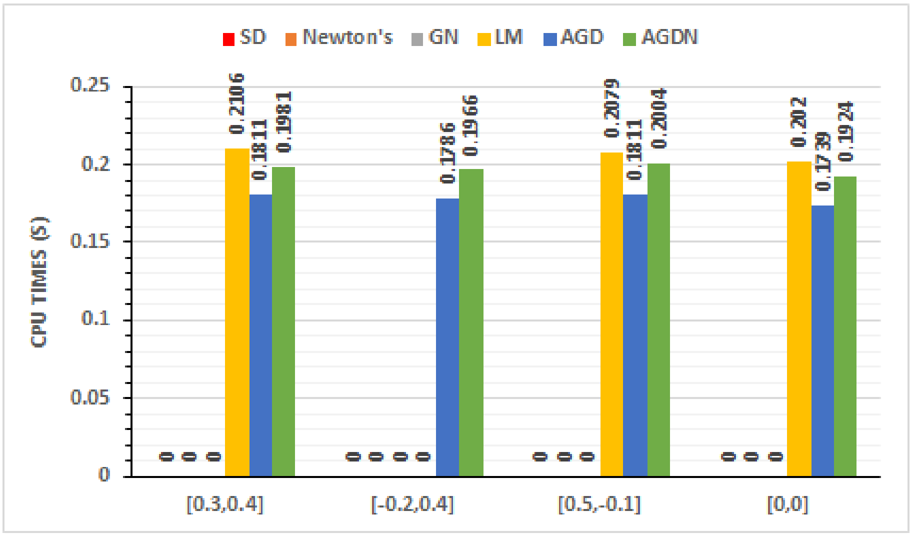


Figure 4.27. A comparison between the CPU times required for convergence when the six numerical methods is applied to solve Test Problem 12 using four initial points; i.e. $x_{10} = [0.3, 0.4]$, $x_{20} = [-0.2, 0.4]$, $x_{30} = [0.5, -0.1]$ and $x_{40} = [0, 0]$.

From Table 4.12, notice that all numerical methods, except for the LM, the AGD and the AGDN methods, fail to solve Test Problem 12. The Newton's method fails because the iteration limit is reached with large values of $\|g(x_k)\|$. For the GN method, its failure may be due to the almost singularity of the truncated Hessian matrix. For the SD method, it converges to a false solution for initial points $x_{10} = [0.3, 0.4]$ and $x_{30} = [0.5, -0.1]$. In fact, it does converges to the minimum point $x^* = [0.2578, 0.2578]$ for initial points $x_{20} = [-0.2, 0.4]$ and $x_{40} = [0, 0]$. However, since $\|g(x_k)\| > 10^{-6}$ at $k = 50000$ for those initial points, the SD method is declared as a failure. Similar result is obtained for the LM method whereby $\|g(x_k)\| > 10^{-6}$ at $k = 50000$ for x_{20}. Conversely, the experimental results of the AGD and the AGDN methods are very encouraging since a few number of iterations are required to reach x^* in a very short amount of time as illustrated in Figures 4.26 and 4.27 respectively.

From the numerical experiments of two-variable NLS test problems, we can conclude that both the AGD and the AGDN methods have shown great successes in solving these test problems compared to other numerical methods. They are more reliable since they are able to solve 11 out of 12 of these test problems with a probability of

0.92. They are robust since they are able to deal with large parameter variations of a test function as can be seen from their results for Test Problems 8–10 when solving the modified Rosenbrock test functions. In addition, they are efficient because they require only a few number of iterations to reach the minimum point x^* in a very short amount of time. Furthermore, the plots of phase portraits have revealed that their phase trajectories behave very steadily before approaching x^*.

In addition to that, note that the AGD and the AGDN methods produce similar numerical results for the two-variable NLS test problems. However, in general, the AGD method has a faster convergence rate than the AGDN method. Nonetheless, these time differences are negligible. For higher dimensional NLS test problems, the numerical results of these two methods may vary. This will be discussed in the next section.

4.3. Numerical experiments on multi-variable NLS test problems

In this thesis, multi-variable test problems involve test functions with $n \geqslant 3$. These functions are higher dimensional test functions which can be used to test the efficiency, reliability and robustness of a numerical method in more vigorous manner. However, it is impossible to plot the level sets of such functions. As a result, it may lead to confusion and skepticism of the failure of a numerical method when it is expected to work.

4.3.1. The multi-variable NLS test problems

Table 4.13 provides the multi-variable NLS test problems used in numerical experiments. A detailed information of these test problems can be found in Appendix A.

Table 4.13. A list of multi-variable NLS test problems used in numerical experiments where the abbreviations "TP" and "Dim" denote Test Problem and the dimension of the problem respectively.

TP No.	**Function Name**	**Dim**		**TP No.**	**Function Name**	**Dim**	
		n	***m***			***n***	***m***
A.	B3DF	3	10	J(ii).	PF I	10	11
B.	GRDF	3	10	K(i).	PF II	4	8
C.	BF	3	15	K(ii).	PF II	10	20
D.	GF	3	15	L.	HeF	8	9
E.	MF	3	16	M.	Os I	5	33
F.	WF	4	6	N.	BEXP6F	6	13
G.	CF	4	7	O.	VDF	8	10
H.	K&OF	4	11	P.	GrF	10	11
I.	B&DF	4	20	Q.	Os II	11	65
J(i).	PF I	4	5	R.	n-D LvF	20	21

4.3.2. Experimental results on multi-variable NLS test problems

The six numerical methods (as described by Algorithms 4–10) are applied to solve each multi-variable NLS test problem in Table 4.13 using the standard initial point. If the standard initial point is not available, then a random point will be chosen as the initial point of the test problem. Table 4.14 records the data obtained from the numerical experiments.

Table 4.14. A record of the numerical results for multi-variable NLS test problems where t denotes the CPU times in seconds.

TP	**SD**			**Newton's**			**GN**			**LM**			**AGD**			**AGDN**		
No.	k	$F(x^*)$	t	k	$F(x^*)$	t	k	$F(x^*)$	t	k	$F(x^*)$	t	k	$F(x^*)$	t	k	$F(x^*)$	t
A.		**FAILED**		8	2.47×10^{-13}	0.2391	975	1.60×10^{-13}	1.5704	21	3.96×10^{-13}	0.4365	16	6.24×10^{-20}	0.2095	16	9.82×10^{-21}	0.2054
B.		**FAILED**			**FAILED**			**FAILED**			**FAILED**			**FAILED**			**FAILED**	
C.	5788	4.11×10^{-3}	1.6839		**FAILED**		72	4.11×10^{-3}	0.5077	19	4.11×10^{-3}	0.3462	8	4.11×10^{-3}	0.2068	8	4.11×10^{-3}	0.2045
D.	57	5.64×10^{-9}	0.2674	1	5.65×10^{-9}	0.2389	2	5.64×10^{-9}	0.4564	2	5.64×10^{-9}	0.3558	3	5.64×10^{-9}	0.1775	2	5.64×10^{-9}	0.1760
E.		**FAILED**			**FAILED**			**FAILED**			**FAILED**			**FAILED**			**FAILED**	
F.	8294	6.64×10^{-13}	2.3385		**FAILED**		10	2.81×10^{-16}	0.3860	81	3.85×10^{-15}	0.3748	36	1.46×10^{-14}	0.2137	36	1.46×10^{-14}	0.2116
G.	8725	6.55×10^{-13}	3.4777	19	7.98×10^{-19}	0.2499	2152	2.06×10^{-13}	3.0513	27	5.25×10^{-15}	0.3931	14	1.77×10^{-17}	0.1989	14	1.77×10^{-17}	0.1967
H.	6330	1.54×10^{-4}	2.3528		**FAILED**		18	1.54×10^{-4}	0.4882	13	1.54×10^{-4}	0.3473	20	1.54×10^{-4}	0.2046	19	1.54×10^{-4}	0.2072
I.		**FAILED**		9	4.29×10^{4}	0.2593		**FAILED**			**FAILED**		11	4.29×10^{4}	0.2326		**FAILED**	
J(i).	34076	1.13×10^{-5}	8.2899	27	1.13×10^{-5}	0.2484	24201	1.13×10^{-5}	21.698	39	1.13×10^{-5}	0.3405	775	1.13×10^{-5}	1.5220	32	1.12×10^{-5}	0.2205
J(ii).	27993	3.54×10^{-5}	34.795	78	3.54×10^{-5}	0.3724	109	3.54×10^{-5}	0.8096	42	3.54×10^{-5}	0.4048	13235	3.54×10^{-5}	52.585	35	3.54×10^{-5}	0.2802
K(i).		**FAILED**			**FAILED**			**FAILED**		36181	4.80×10^{-6}	21.441	194	4.70×10^{-6}	0.4513	101	4.69×10^{-6}	0.2701
K(ii).		**FAILED**		168	1.47×10^{-4}	0.8911	9452	1.47×10^{-4}	54.808	51	1.47×10^{-4}	0.5185	19778	1.47×10^{-4}	120.25	89	1.47×10^{-4}	0.5740
L.		**FAILED**			**FAILED**		1	1.77×10^{7}	0.3675	12	1.77×10^{7}	0.3391	6	1.77×10^{7}	0.1873	6	1.77×10^{7}	0.1866
M.		**FAILED**			**FAILED**			**FAILED**		46	2.73×10^{-5}	0.5349	25	2.73×10^{-5}	0.2974		**FAILED**	
N.		**FAILED**			**FAILED**			**FAILED**			**FAILED**		5632	9.21×10^{-8}	18.728	95	1.89×10^{-13}	0.4966
O.	35	1.95×10^{-15}	0.2613	9	1.52×10^{-17}	0.2520		**FAILED**		19	3.50×10^{-17}	0.3401	13	4.30×10^{-21}	0.2265	13	4.30×10^{-21}	0.2044
P.	7531	1.97×10^{-28}	22.955		**FAILED**		64	3.71×10^{-12}	0.8264	59	4.62×10^{-12}	0.5302	19	4.21×10^{-12}	0.2844	16	1.05×10^{-21}	0.2620
Q.		**FAILED**			**FAILED**		21	2.01×10^{-2}	2.5341	19	2.01×10^{-2}	1.4156	20	2.01×10^{-2}	0.9664	20	2.01×10^{-2}	0.9651
R.	41	$5.85\times^{-13}$	0.8996		**FAILED**		9	$3.81\times^{-17}$	0.7792	10	3.52×10^{-13}	0.5415	41	3.85×10^{-12}	0.9382	23	1.60×10^{-12}	0.5949

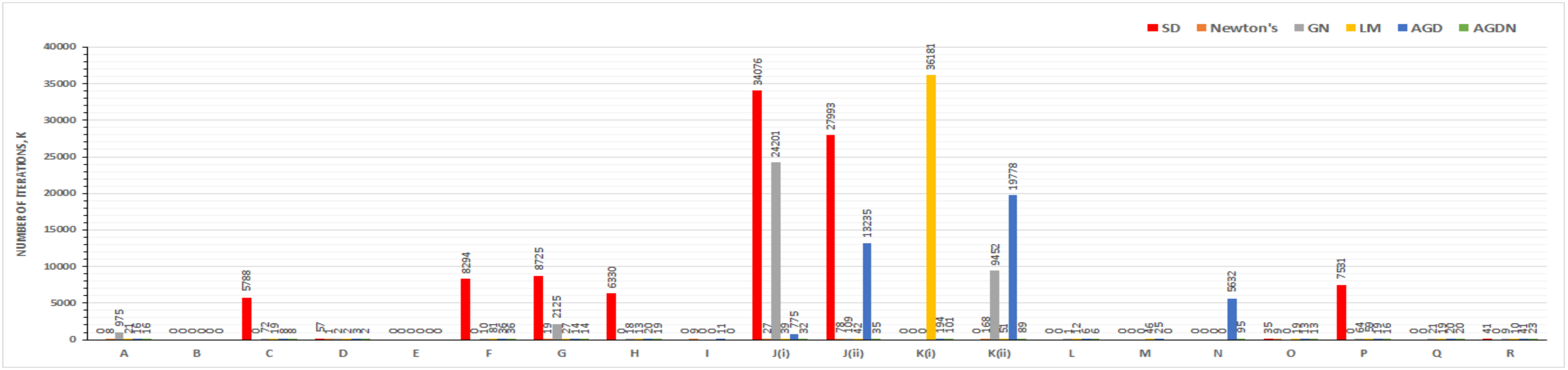


Figure 4.28. A comparison between the iteration numbers required by the six numerical methods for solving the multi-variable NLS test problems.

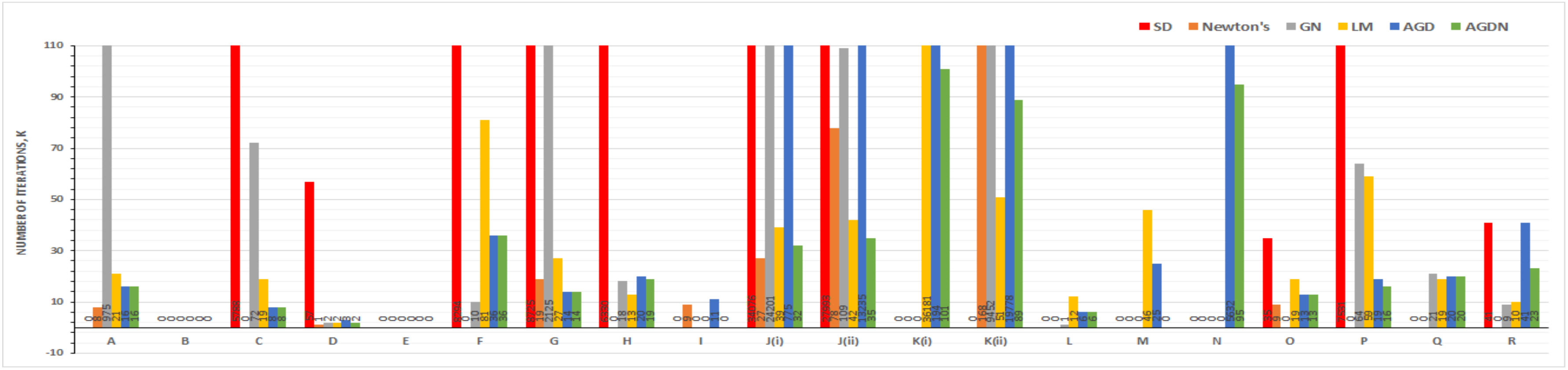


Figure 4.29. A zoomed-in version of Figure 4.28 for all iteration numbers within 110.

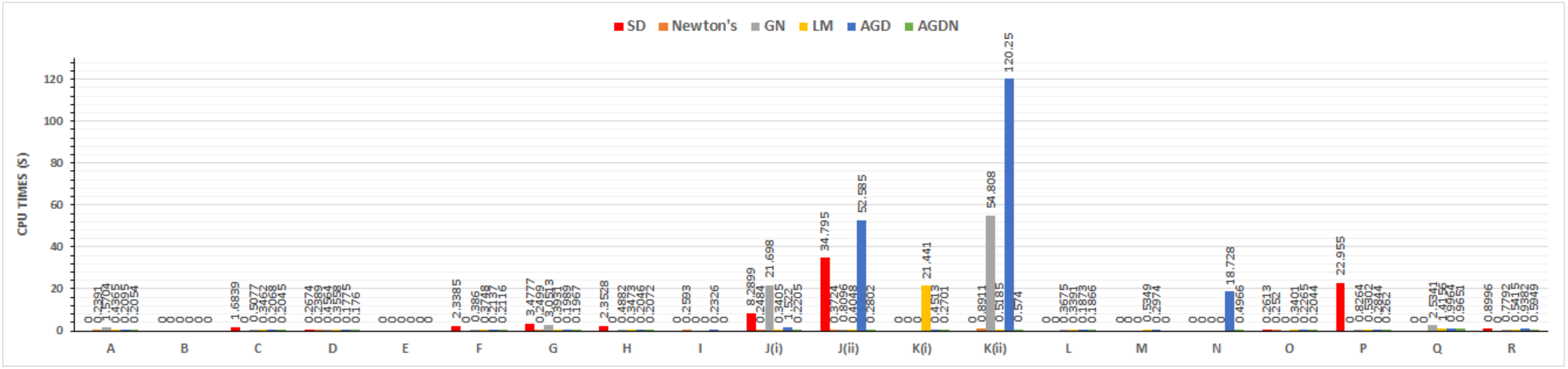


Figure 4.30. A comparison between the CPU times required for the six numerical methods for solving the multi-variable NLS test problems.

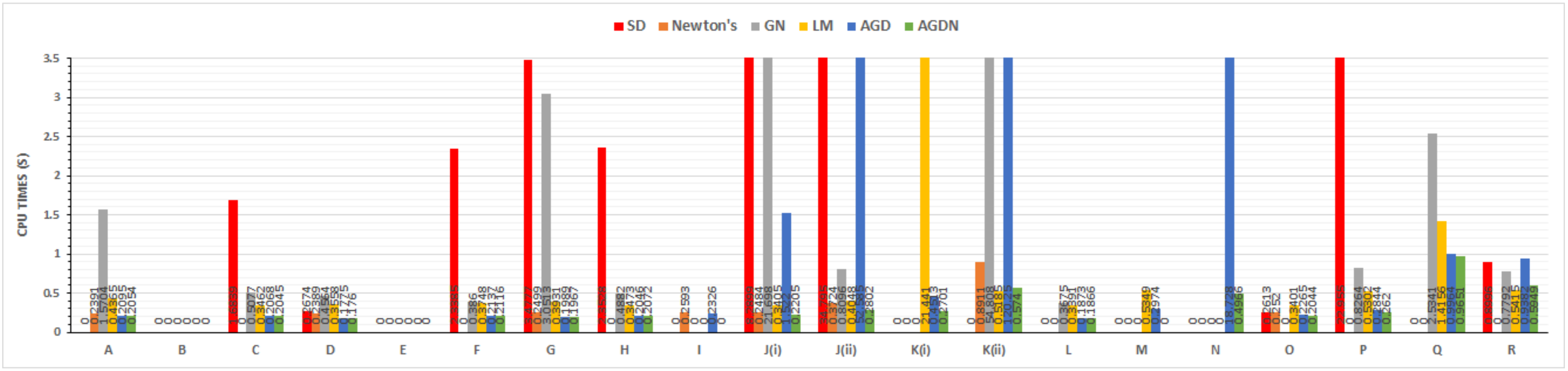


Figure 4.31. A zoomed-in version of Figure 4.30 for all CPU times within 3.5 seconds.

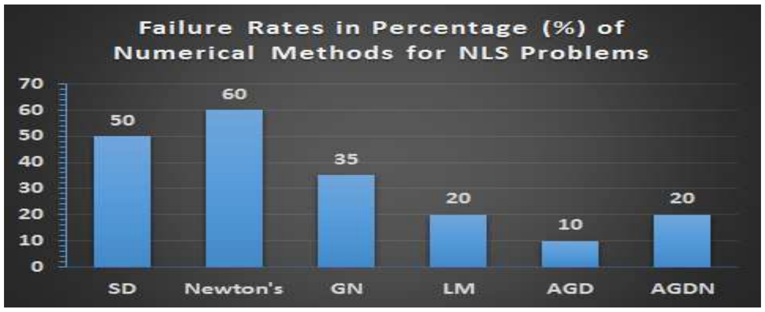


Figure 4.32. Failure rates in percentage (%) of numerical methods for solving the multi-variable NLS test problems.

From Table 4.14 and Figure 4.32, observe that the Newton's method shows the most failures; i.e. 12 out of 20 test problems fail to be solved when compared with other methods. This is followed by the SD and the GN methods with failure rates of 50% and 35% respectively. This shows that the SD, the Newton's and the GN methods are not reliable numerical methods. The LM method, which is regarded as the most famous numerical approach in the NLS literature, shows the best successful rate among the existing methods in solving the multi-variable NLS test problems with only 4 failed test problems. Moreover, observe that the newly developed AGDN method also show similar failure rate of 20%. Hence, this indicate that the LM and the AGDN methods are reliable numerical methods. Nonetheless, the AGD method is the most reliable method with failure rate of only 10%.

In contrast to the results on two-variable test functions, note that the AGD and the AGDN methods do not show similar results in general based on the data recorded in Table 4.14. In addition, both the AGD and the AGDN methods outperform all the existing methods with only 2 and 4 failed test problems respectively. Moreover, the two test problems that the AGD method fails to solve also remain unsolved for all other methods. Meanwhile, notice that the AGDN method fails to solve Test Problem M which involves solving the Osborne I function with dimensions of $n = 5$ and $m = 33$. This is due to the singularity of the Hessian matrix in Phase-II of the AGDN method where the Newton's method is used. This singularity is overcome by a positive parameter μ in the AGD algorithm and hence it works. Therefore, whenever the Hessian matrix is (almost) singular in Phase-II, the AGD method should be used instead.

Figures 4.28 and 4.30 show the plots for the number of iterations and the CPU times required by the six numerical methods when they are applied to solve the multi-variables NLS test problems given in Table 4.13. A zoomed-in version of Figures 4.28 and 4.30 are provided in Figures 4.29 and 4.31 respectively so that a better visualization can be achieved among those methods which require small iteration numbers and short CPU times. For cases where convergence is achieved, it can be seen from these figures that the SD method has the slowest convergence rates with the highest number of iterations in general. This is followed by the AGD and the GN methods. As expected, the Newton's method always has a fast convergence rate but with a very high failure rate.

Furthermore, it can be observed that the AGDN method works the best when compared with all other methods. In general, it shows a faster convergence rate with a relatively fewer number of iterations when compared with the AGD method. From Figure 4.28, notice that the numbers of iterations required by the AGDN method are less than 110 and converge within 1 second as depicted in Figure 4.31. These figures suggest that the AGDN method outperforms all the other numerical methods when applied to solve the multi-variable NLS test problems.

4.4. Performance profiles

The performance profile for a solver is a nondecreasing piecewise continuous constant function drawn from the right (Dolan and Moré, 2002). It compares the performance of a set of solvers $\mathcal{S}$ (or numerical methods) on a set of test problem $\mathcal{P}$ based on computing a performance ratio defined as

$$r_{p,s} = \frac{t_{p,s}}{\min\left\{t_{p,s} : s \in \mathcal{S}\right\}}$$

where $t_{p,s}$ is the computing time required by the solver s to solve problem p. In addition, the overall assessment of the performance of the solver is obtained from

$$\rho_s(\tau) = \frac{1}{n_p}\text{size}\left\{p \in \mathcal{P} : r_{p,s} \leqslant \tau\right\}$$

where $\rho_s(\tau)$ is the probability for solver $s \in \mathcal{S}$ for which a performance ratio $r_{p,s}$ is within a factor $\tau \in \mathbb{R}$ of the best possible ratio and n_p is the number of test problems in $\mathcal{S}$. Furthermore, ρ_s is the cumulative distribution function for the performance ratio. The value of $\rho_s(1)$ determines the probability that the solver will win over other solvers in comparison. Hence, if the number of wins is the focus of interest in the comparison, only the values of $\rho_s(1)$ need to be compared among all the solvers. On the other hand, if the focus of interest lies in getting the solvers with a high chance of success, then the value of ρ_s^* need to be compared among all the solvers and select the solvers with the largest value. This value of ρ_s^* is obtained from the flat tail of the curve in a performance profile for large values of τ.

According to Dolan and Moré (2002), the performance profiles are not sensitive to the results on a small number of test problems n_p. In addition, if n_p is substantially large, then the result on a particular test problem will not affect the performance profile significantly.

Furthermore, Dolan and Moré (2002) mentioned that a plot of the performance profile shows all of the important performance characteristics of the solvers. Hence, in order to obtain an overall comparison of all the numerical results obtained earlier, two logarithmic scaled performance profiles for the six solvers in terms of the number of iterations and the CPU times are plotted in Figures 4.33 and 4.34 respectively. These performance profiles are plotted by using the combined numerical results obtained from testing the two-variable and the multi-variable NLS test problems.

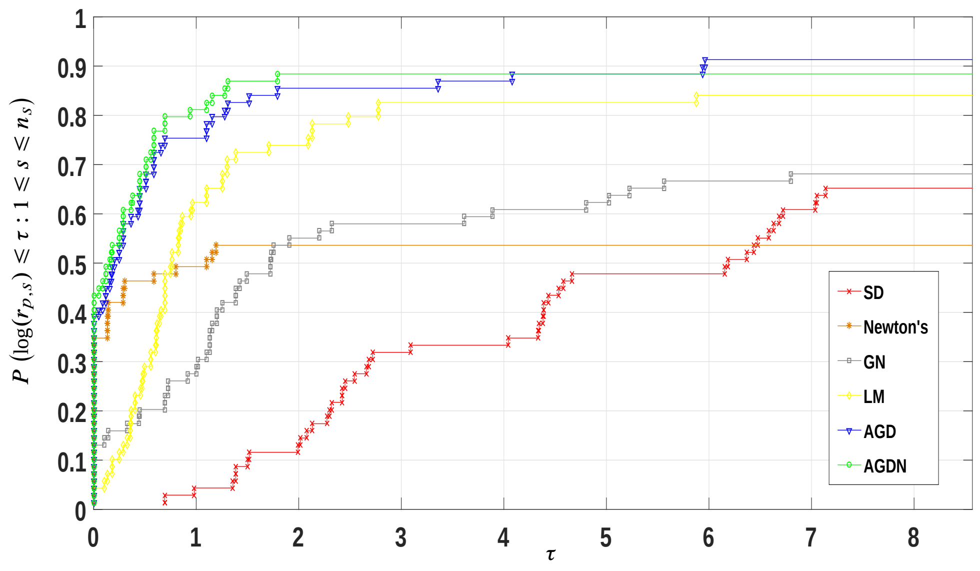


Figure 4.33. Logarithmic scaled performance profile for the six solvers in terms of number of iterations where n_s denotes the number of solvers in $\mathcal{S}$.

Figure 4.33 shows the logarithmic scaled performance profile of the six solvers in terms of number of iterations. From the figure, it can be seen that both the AGD and the AGDN methods require less iterations on average to reach the minimum points compared with the other methods. Furthermore, it is clear that the AGDN method has the most wins (i.e. the highest probability) of being the optimal solver and that the probability that the AGDN method is the winner on a given test problem is 0.44. However, the AGD method shows the highest probability of over 0.9 in solving the NLS test problems successfully, as displayed by the height of its performance profile for $\tau > 6$. This suggests that the AGD method is more robust for $\tau > 6$.

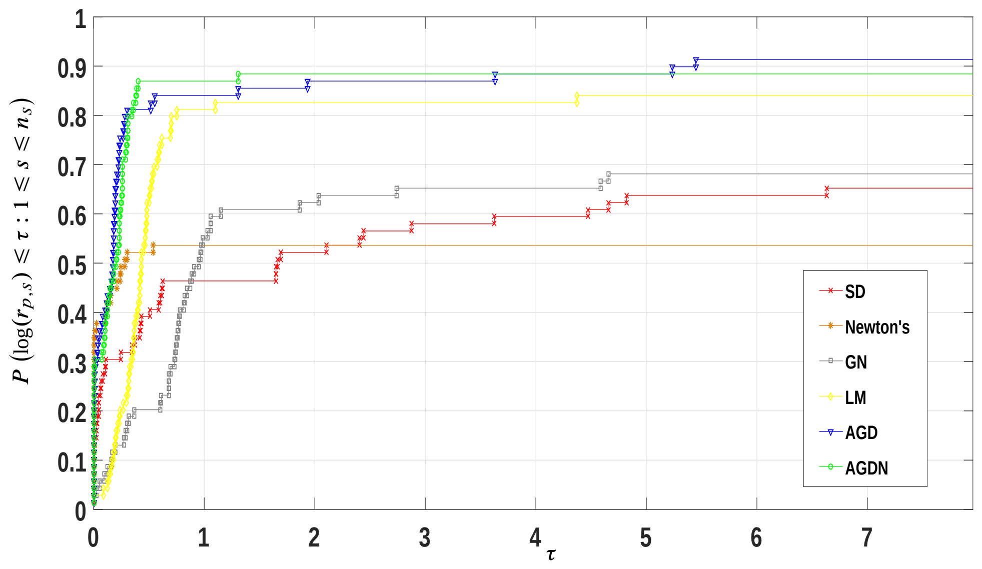


Figure 4.34. Logarithmic scaled performance profile for the six solvers in terms of CPU times where n_s denotes the number of solvers in $\mathcal{S}$.

In terms of execution times, the Newton's method has the most wins with a probability of 0.38 of being the optimal solver as depicted in Figure 4.34. However, its performance is quickly taken over by the AGD and the AGDN methods. These methods become more competitive and outperform all the other methods after $\tau > 0.1$. Again, the AGD shows a probability of over 0.9 in solving the NLS test problems successfully for $\tau > 5.2$ which suggests that it is more robust for $\tau > 5.2$. This is because the positive parameter μ in the AGD algorithm ensures the positive definiteness of the term $\mu I + H$ for iterations near the minimum point and hence overcomes the singularity of the Hessian matrix. Nonetheless, due to the short amount of time required by the AGDN method on average, it should be considered as the priority method for solving the NLS problem. If the Hessian matrix is found to be (almost) singular near the minimum point, then AGD method should be used instead.

4.5. Conclusion

Based on the collection of test problems that are available in the NLS literature, it can be seen that almost all the test problems are concerned with testing functions with only one standard initial point. As a consequence, numerical methods that work for the standard initial point may fail for other initial points; particularly those that are farther away from the minimum point. In other words, there have been too much emphasis on testing the efficiency of the numerical methods rather than on the reliability and robustness of these methods.

In order to overcome this issue, the six numerical methods (as described by Algorithm 4–9) are tested using the two-variable NLS test functions for four chosen initial points. These points are selected by first dividing the 2-dimensional plane into four regions using the lines $x_1 = x_{1e}^*$ and $x_2 = x_{2e}^*$ where $x^* = [x_{1e}^*, x_{2e}^*]$ is the minimum point of $F(x)$. Then, four initial points are selected from each region. This technique is particulary easy for testing two-variable test problems. However, for multi-variable test problems with $n \geqslant 3$, it involves selecting and running computer experiments for n^2 number of initial points. Hence, only the standard initial points will be tested using the six numerical methods for multi-variable NLS test problems. However, further research should be done to address this issue.

A major advantage of using the two-variable test problems in testing numerical algorithm is that it is always possible to plot the level sets of a two-variable function. The level sets of a function reveals important information regarding the behaviour and structures of the function. According to Lemma (3.1), a function which has properly nested level sets should converge to the minimum point in a finite time provided that the iterations stay within the properly nested region. Furthermore, a phase portrait of the function reveals the behaviour of its trajectories before they approach the minimum point. A trajectory which behave steadily along its path shows the stability of the numerical method when solving an NLS problem.

Based on the numerical results obtained from the testing of two-variable test problems, it was found that both the AGD and the AGDN methods outperform all the other methods. From the phase portraits, it was observed that their phase trajectories behave steadily before reaching the minimum point. Furthermore, they can converge to the minimum points with a few number of iterations in a very short amount of time. Besides that, they have a very high successful rate in solving the test problems compared to the other methods. They are also able to deal with large parameter variations in a test problem as shown by their abilities to solve Test Problems 8–10 successfully. In short, the AGD and the AGDN methods have shown great results in terms of efficiency, reliability and robustness of a numerical method.

Based on the results on testing the multi-variable test problems, it was found that the Newton's method has the highest failure rate of 60%. This is mostly due to the singularity of the Hessian matrix during its iterative process. However, in cases where it converges, the convergence rates are normally faster than the other methods. On the other hand, the SD method shows the second highest failure rate of 50% and it requires large number of iterations for convergence. Despite these high iteration numbers, the SD iterative process is comparatively faster than the other methods. This is mainly due to its low computational cost since it only requires the evaluation of the first derivatives. The GN method has shown a moderate result among all the methods with a failure rate of 35%. Its failure is mainly due to the singularity of the truncated Hessian matrix. With the incorporation of a positive Lagrange parameter, the LM method has a lower failure rate compared to the GN method and it performs the best among all the existing methods. Meanwhile, the AGD and the AGDN methods show very encouraging results compared to the other methods with failure rates of only 10% and 20% respectively. These methods outperform all the other methods since they have fast convergence rates with less number of iterations in general.

Both the AGD and the AGDN methods share similar numerical results for two-variable test problems. However, for multi-variable test problems, their results may differ. It was shown that the AGD method has a higher successful rate compared to the AGDN method. The failure of the AGDN method is mostly due to the singularity of the Hessian matrix when the Newton's method is used in Phase-II. This singularity of the Hessian matrix has lead to the failure of the AGDN method when it is applied to solve the Osborne I function (see Test Problem M in Appendix A). In the AGD method, the singularity of the Hessian matrix is overcome by a positive parameter μ in its algorithm. Nevertheless, the AGDN method shows a more encouraging result in terms of number of iterations and the CPU times due to the fast quadratic convergence of the Newton's method in Phase-II of the AGDN method. Hence, it should be chosen as the priority method for solving NLS problem. However, in cases where Hessian matrix is singular, the AGD method should be used instead.

In order to obtain an overall comparison of the six numerical methods, two performance profiles for the numerical methods are plotted by combining the results obtained from testing the two-variable and the multi-variable test problems. The performance graphs of these methods have revealed that the AGD and the AGDN methods outperform the existing methods in terms of iteration numbers and convergence rates.

CHAPTER 5

APPLICATIONS OF NLS

In this chapter, some applications of NLS in data-fitting are discussed. These applications are chosen from some of the test problems used in Chapter 4. Based on the numerical results obtained from Chapter 4, the NLS fitting curves of the test functions are plotted together with the m data points. It was observed that the solution (or minimum point) obtained from the numerical methods have provide a good fitting model to the m data points for each test problem.

5.1. Some Applications of NLS

According to Bongartz et al. (1995), the wide collection of test problems available in the constrained and unconstrained testing environment (CUTE) for numerical optimization consists of test problems gathered from a variety of academic and real-life sources. These sources range from Physics, Chemistry, Biology, Economy to Operations Research. Now, CUTE has been superseded by its latest evolution, CUTEst.

In this section, we discuss some of the test problems available in CUTEst that involve NLS data-fitting. These test problems have already been considered and solved in Chapter 4. Since the solutions obtained from these numerical methods are similar, only one least squares data-fitting plot is drawn for each test problem to show the quality of the solutions obtained from the numerical methods. These solutions are the minimum points of the objective function $F(x)$ where the residual vector function $r(x)$ is minimized so that a best fit curve to the data can be achieved. This procedure is also known as parameter estimations in NLS data-fitting. If a nonlinear mathematical model $M(x,t)$ fits exactly at each data point, then $r(x^*) = 0$ and the problem is termed a zero-residual problem. If the model $M(x,t)$ fits "closely" to most of all the data points, then $r(x^*)$ is small and the problem is called a small-residual problem. Otherwise, it is termed a large-residual problem (Dennis and Schnabel, 1983; Kelly, 1999).

5.1.1. The Bard function

The Bard function is an application of NLS data-fitting with $n = 3$ parameters and $m = 15$ data points or residual functions. Each residual function has a linear and a nonlinear term. A detailed information of the Bard function is given in Appendix A.2 (see Test Problem C). From Table 4.14 of Chapter 4, observe that the Newton's method fails to work when it is applied to solve the Bard function. Hence, no minimum point can be found. Other numerical methods yield the same minimum point given as $x^* = [0.0824, 1.133, 2.344]$. With these parametric values, a nonlinear fitting model of the Bard function is given by

$$M(x,t) = 0.0824 + \frac{t}{1.133(16-t) + 2.344\min(t, 16-t)}.$$

Figure 5.1 displays the least squares fit of the bard function. From the figure, it can be seen that the nonlinear model fits very well to most of the data points.

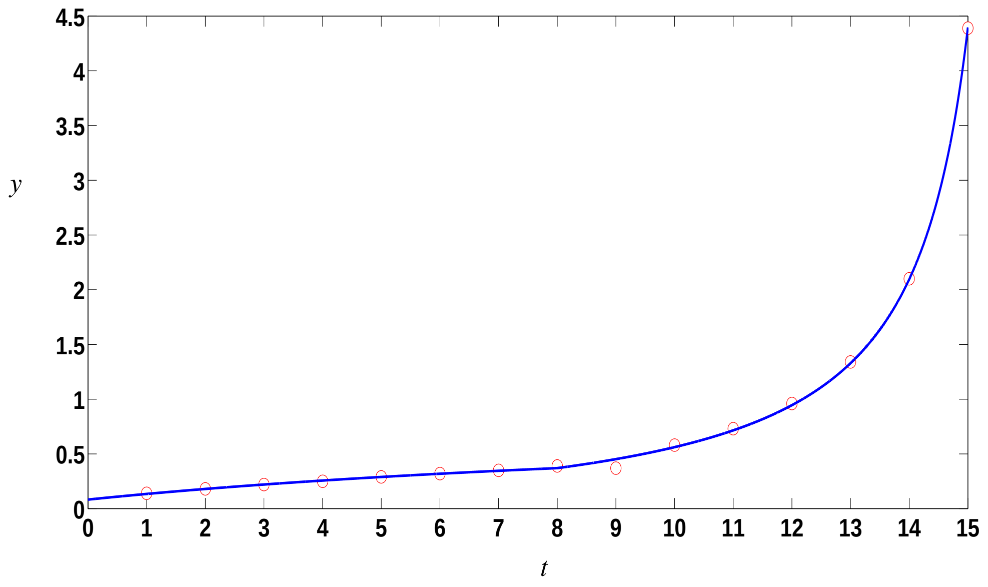


Figure 5.1. A least square fit of the Bard function. The red circled symbols denote the m data points and the blue curve represents the nonlinear fitting model $y = M(x,t)$.

5.1.2. Gaussian function

The Gaussian function is an application of NLS data-fitting with $n = 3$ parameters and $m = 15$ data points. This function is listed in Appendix A.2 as Test problem D. From the numerical experiments conducted in Chapter 4, it was found that all the six numerical methods converges to the same minimum point $x^* = [0.3990, 1, 0]$ when applied to solve the Gaussian function. With these parametric values, a nonlinear fitting model of the Gaussian function is given as

$$M(x,t) = 0.3990\mathrm{e}^{-\frac{1}{2}\left(\frac{8-t}{2}\right)^2}.$$

Figure 5.2 depicts the least squares fit of the Gaussian function. It can be seen that the nonlinear Gaussian fit provides a best fit curve to all the red colored data points.

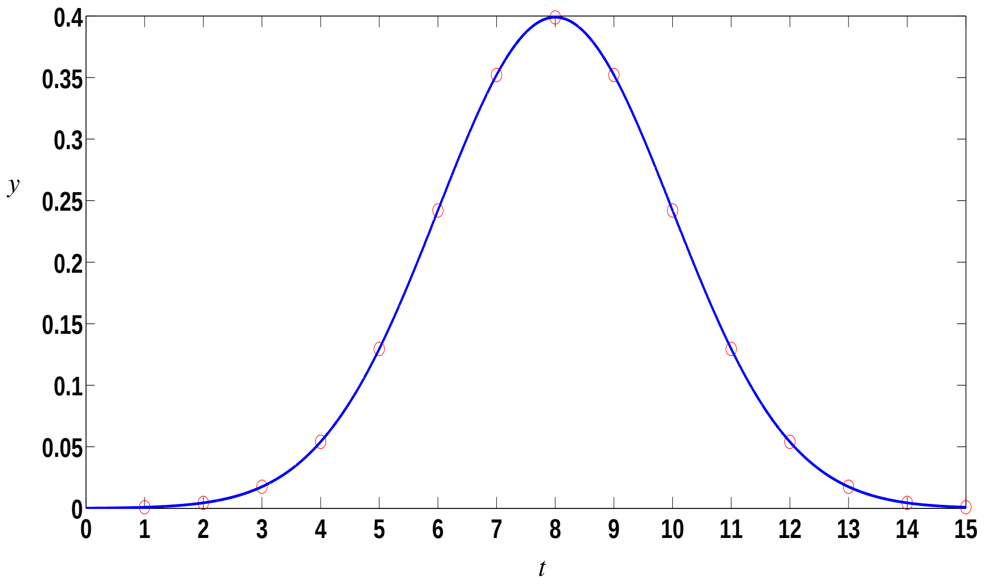


Figure 5.2. A least square fit for the Gaussian function. The red circled symbols denote the m data points and the blue curve represents the nonlinear fitting model $y = M(x, t)$.

5.1.3. Kowalik and Osborne function

The Kowalik and Osborne function is given in Appendix A.2 as Test Problem H with $n = 4$ parameters and $m = 11$ data points. This problem is an application in Physics which models an analysis of kinetic data for an enzyme reaction. From the results of the numerical experiments carried out in Chapter 4, it was found that all the methods, except the Newton's method, converges to $x^* = [0.193, 0.191, 0.123, 0.136]$. Hence, a nonlinear fitting model for this test problem is given as

$$M(x, t) = \frac{0.193(t^2 + 0.191t)}{t^2 + 0.123t + 0.136}.$$

A nonlinear least squares fit for the Kowalik and Osborne function is Figure 5.3. Notice that the nonlinear fitting model only provides a fair fit to the data points.

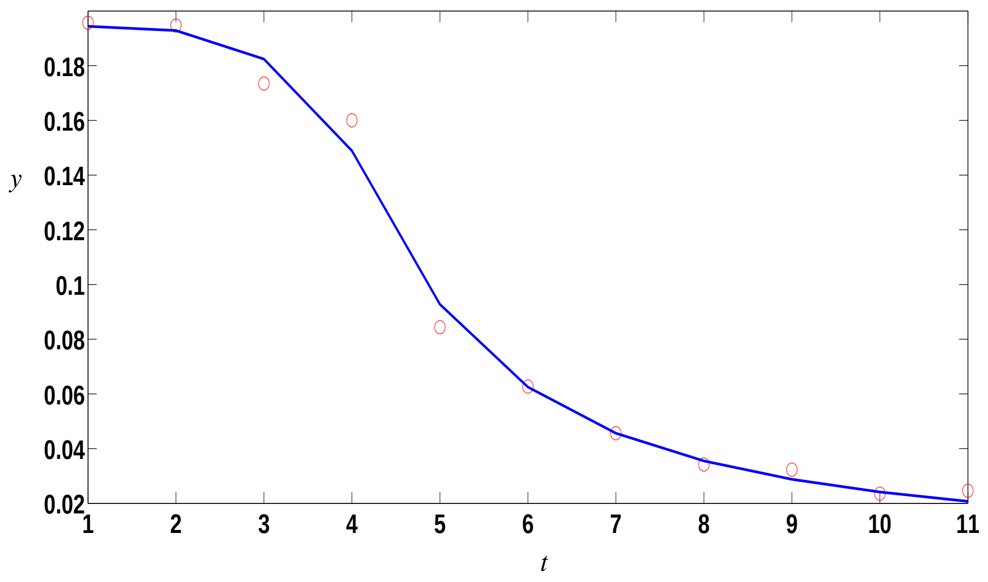


Figure 5.3. A least square fit for the Kowalik and Osborne function. The red circled symbols denote the m data points and the blue curve represents the nonlinear fitting model $y = M(x, t)$.

5.1.4. Osborne I function

The Osborne I function is an application of NLS data fitting with $n = 5$ parameters and $m = 33$ data points or residual functions. Each residual function consists of one linear term and two nonlinear terms. The Osborne I function is listed in Appendix A.2 as Test Problem M. From the numerical analysis conducted in Chapter 4, it was found that the LM and the AGD methods solve the Osborne function successfully with $x^* = [0.375, 1.94, -1.47, 0.0129, 0.0221]$. This may suggest that the Osborne function is an NLS problem that is difficult to solve. Using the parametric values of x^*, a nonlinear fitting model of the Osborne I function is

$$M(x, t) = 0.375 + 1.94e^{-0.129(t-1)} - 1.47e^{-0.221(t-1)}.$$

A nonlinear lest squares fit is depicted in Figure 5.4 below. It can be seen that the parametric values obtained from the AGD method fits perfectly well to all the data points.

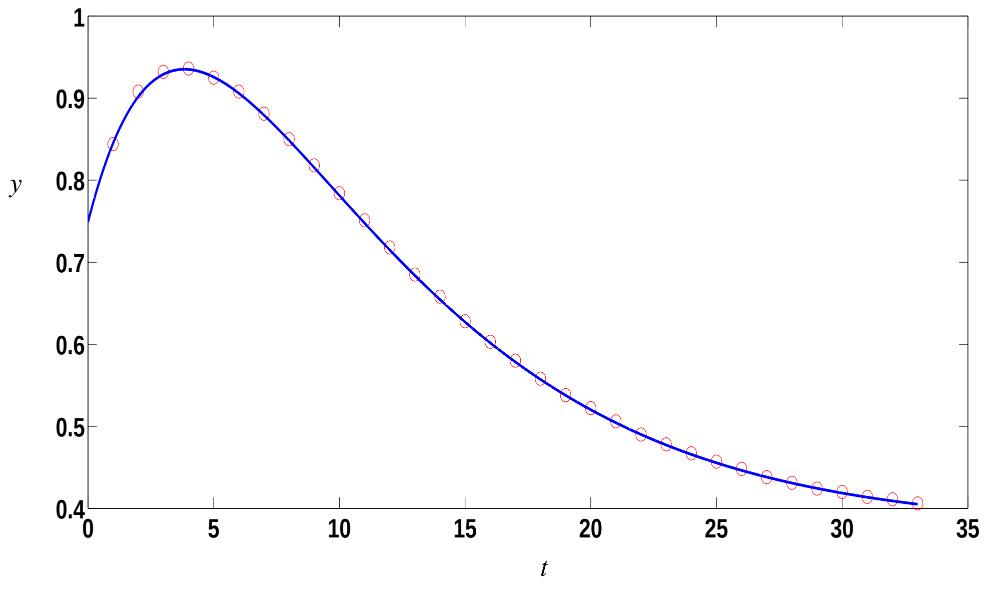


Figure 5.4. A least square fit for the Osborne I function. The red circled symbols denote the m data points and the blue curve represents the nonlinear fitting model $y = M(x,t)$.

5.1.5. Osborne II function

The Osborne II function is considered to be the highest dimensional problem of all the test problems used in this thesis with $n = 11$ parameters and $m = 65$ data points or residual functions. Each residual function consists of four nonlinear elements. The Osborne II function is listed in Appendix A.2 as Test Problem Q. Of all the 6 numerical methods, only the GN, the LM, the AGD and the AGDN methods work when they are applied to solve the Osborne II function. These methods yield the same minimum point given as $x^* = [1.31, 0.432, 0.634, 0.599, 0.754, 0.904, 1.37, 4.82, 2.40, 4.57, 5.68]$. Using these parametric values, a nonlinear fitting model of the Osborne function is given as

$$M(x,t) = 1.31\mathrm{e}^{-0.0754(t-1)} + 0.432\mathrm{e}^{-0.904\left[\frac{t-1}{10}-2.40\right]^2} + 0.634\mathrm{e}^{-1.37\left[\frac{t-1}{10}-4.57\right]^2} + 0.599\mathrm{e}^{-4.82\left[\frac{t-1}{10}-5.68\right]^2}.$$

Figure 5.5 illustrates a least square fit of the Osborne function. It can be seen that the fitting model provides a good fit to the given data points.

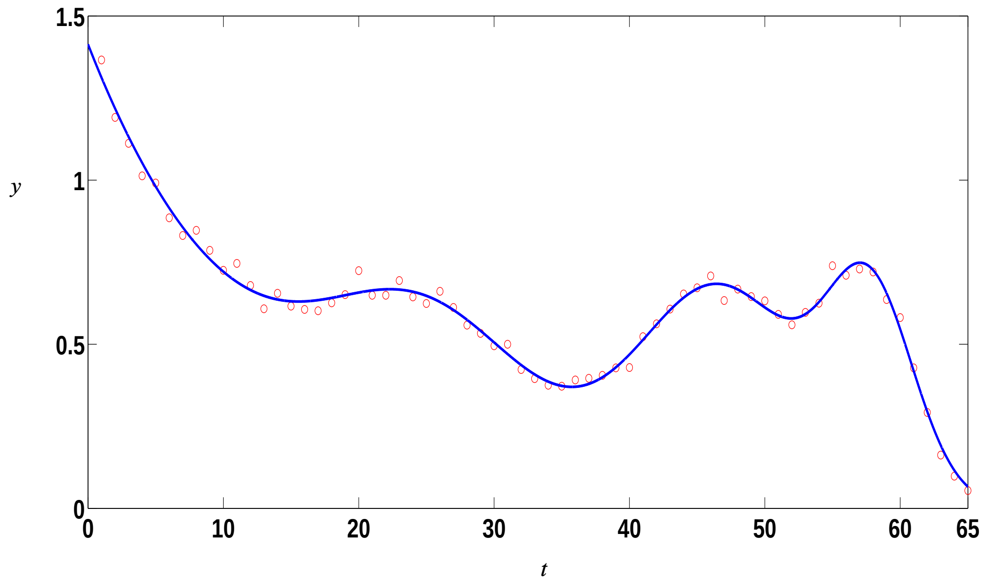


Figure 5.5. A least square fit for the Osborne II function. The red circled symbols denote the m data points and the blue curve represents the nonlinear fitting model $y = M(x,t)$.

5.2. Conclusion

Based on the figures of least squares fitting, one can conclude that the solution (or minimum point) obtained from the numerical methods have provide good parameter estimations for the fitting models. This can be observed from the "closeness" of the fitting curves to the data points which implies that the residuals are almost zeros.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

This chapter provides an overall conclusion for the whole research project. Following that, some suggestions on the possible future research work are discussed.

6.1. Conclusion

The incorporation of numerical differentiation has provides a great flexibility where numerical calculations can be performed by just providing the objective function of the NLS problem. This saves a lot of time and effort while preventing any evaluation mistakes done analytically. The experimental results from Chapters 4 and 5 have shown that the use of numerical differentiation with finite differencing in numerical algorithms has provide useful approximations to the derivatives of the objective function. This is particularly obvious from the plots of the fitting curves in Chapter 5 where the the "closeness" of the fitting curves to the data points implies that the residuals are almost zeros. This in turn shows the accuracy of the solutions (minimum points) obtained from the numerical methods. As a result, the use of truncated Hessian matrix can be avoided in the new AGD and the AGDN methods for solving NLS problem. With the incorporation of numerical differentiation, all the numerical methods can be implemented in practical problems.

All the numerical methods discussed in this thesis follow the Lyapunov function theorem as convergence analysis. The numerical results on Test Problem 3 for a two-variable test function have shown that the implementation of the Lyapunov function theorem in numerical algorithms has avoided an undesirable convergence of the all numerical methods towards the maximum point of the test function. Instead, all the numerical methods which converge show convergence towards the minimum point. This result is also observed for all the other test problems where convergence towards the minimum point is always guaranteed if the methods are convergent. Hence, the Lyapunov function theorem has provide a good convergence analysis for the numerical methods.

Based on all the experimental results conducted in Chapter 4, it can be seen that both the AGD and the AGDN methods outperform all the other numerical methods investigated in this thesis in terms of efficiency, robustness and reliability when they are applied to solve the two-variable and multi-variable test problems. Both methods have shown similar results when they are applied to solve two-variable test problems but as the dimension of the problems get higher, they may yield different outcomes. The AGDN method is a more favourable numerical method compared to the AGD method in terms of number of iterations and convergence rates. This is because the Newton's method, which has a fast quadratic convergence rate, is incorporated into Phase-II of the AGDN method that is activated when the gradient is sufficiently small (i.e. near the minimum point). However, in cases where the Hessian matrix is singular near the minimum point, the AGDN method fails to work. This situation is observed when solving Test Problem M of the Osborne I function. In conclusion, among all the methods, the AGDN method should be chosen as the priority method for solving the NLS problem. In cases where the Hessian matrix is singular near the minimum point, the AGD method should be used instead.

6.2. Future work

As mentioned in Section (4.1) of Chapter 4, the collection of test problems for numerical methods in the current database only consists of one standard initial (or starting) point for each test problem. In addition, this initial starting point is usually close to the solution (or minimum point). As a consequence, numerical methods that work for the standard initial points may fail for initial points that are far away. In other words, there has been to much emphasis on testing the efficiency of numerical methods rather than on the reliability and robustness of these methods. In order to address this issue, a technique which involves choosing four different initial points is used for the two-variable test problems. However, this technique is hard to implement for multi-variable test problems with $n \geqslant 3$. Therefore, further research needs to be done in this area.

Both the AGD and the AGDN methods are new numerical methods in the NLS literature. In addition to that, the AGD method also has a simplified (or reduced) version where a two-dimensional subspace search method is considered (Goh, 2009). This AGD subspace method should be introduced to solve high-dimensional NLS problem where the dimension n or m or when both are large. It uses only two critical components of the gradient vector of the objective function. As a result, it requires only a submatrix of the Hessian matrix of an objective function. This is of considerable advantage in matrix computations for high-dimensional NLS problem. Goh (2009) has shown that the subspace search AGD method is capable of handling very ill-conditioned problems and very high-dimensional problems. For instance, when it is applied to solve a quadratic function with 999 variables, it converges with less than 1600 iterations.

Besides that, the stiff ODE, which has been used to plot the level sets of a two-variable test function, has shown encouraging results when it is applied to solve large-scale nonlinear equations in other areas of numerical optimizations (Luo et al., 2009; Han and Han, 2010). Therefore, this provides us with new insight to use it to solve high-dimensional NLS problem. However, more research needs to be done in this area since NLS requires one to solve an over-determined system of nonlinear equations.

APPENDIX A

THE NLS TEST PROBLEMS

This appendix provides the detailed information of the NLS test problems used in the numerical experiments. Recall that an NLS function takes the form of (2.1); i.e

$$F(x) = \frac{1}{2}\sum_{i=1}^{m} r_i(x)^2; \quad m > n$$

where $r_i(x)$ are the residual functions of an n-variable function $F(x)$. In this section, the NLS test problems are defined using the following format:

(Test Problem (TP) Number) *Function name{} () or []*

(a) Dimensions

(b) Residual functions r_i with $i = 1 \dots m$

(c) The different initial points x_{j0} with $j = 1, 2, \dots$ for $n = 2$

The (standard) initial point x_0 for $n \geqslant 3$

(d) Minimum point x^* or minimum points x_s^* with $s = 1, 2, \dots$ (if available)

(e) Minima $F(x_s^*)$ with $s = 1, 2, \dots$

where an abbreviation for the name of each function is provided in $\{\}$. These abbreviations will be used to denote the function names in Chapter 4. In addition, the number in the round ( ) and square [ ] parentheses after the function name refer to the function numbering in Moré et al. (1981) and Adorio (2005) respectively. However, if it is a new or modified test function, then referencing will be ignored. Otherwise, the function will be cited. It is important to note that these test problems are also available in the constrained and unconstrained testing environment, revisited/safe threads (CUTEr/CUTEst).

A.1. The two-variable NLS test problems

1. New Function 1 {NF 1}
   (a) $n = 2, m = 3$
   (b) $r_1 = x_2 - \cosh x_1, \quad r_2 = x_2 - \cos x_1, \quad r_3 = x_2 - x_1^2 - 1$
   (c) $x_{10} = [-1.5, -1], \quad x_{20} = [-1, 5], \quad x_{30} = [1.5, 4], \quad x_{40} = [2, -2]$
   (d) $x^* = [0, 1]$
   (e) $F(x^*) = 0$

2. New Function 2 {NF 2}
   (a) $n = 2, m = 4$
   (b) $r_1 = x_2 - \cosh(x_1 + 1) + 1, \quad r_2 = x_2 - \sin x_1 - \cos x_1, \quad r_3 = x_2^2 - x_1 + 1,$
   $r_4 = x_1 + 1$
   (c) $x_{10} = [-3, 5], \quad x_{20} = [-3, -4], \quad x_{30} = [2, 4], \quad x_{40} = [3, -4]$
   (d) $x^* = [-0.2954, 0.1980]$
   (e) $F(x^*) = 1.250$

3. New Function 3 {NF 3}
   (a) $n = 2, m = 3$
   (b) $r_1 = x_1 + \frac{3}{2}, \quad r_2 = x_2, \quad r_3 = \sqrt{10}(x_1^2 + x_2^2 - 1)$
   (c) $x_{10} = [0.2, 0.4], \quad x_{20} = [-2, 2], \quad x_{30} = [1.5, 1.5], \quad x_{40} = [1.5, -1.5]$
   (d) $x^* = [-1.0120, 0]$
   (e) $F(x^*) = 0.1220$

4. Three-hump camel function {3-hump CF}[2.7]

   (a) $n = 2, m = 4$

   (b) $r_1 = \sqrt{4x_1^2 - 2.1x_1^4}, \quad r_2 = \frac{1}{\sqrt{3}}x_1^3, \quad r_3 = \sqrt{2x_1x_2}, \quad r_4 = \sqrt{2}x_2$

   (c) $x_{10} = [4, 4], \quad x_{20} = [4, -4], \quad x_{30} = [-4, 4], \quad x_{40} = [-4, -4],$
   $x_{50} = [-1, -1]$

   (d) $x_1^* = [1.7476, -0.87378], \quad x_2^* = [-1.7476, 0.87378], \quad x_3^* = [0, 0]$

   (e) Local minima: $F(x_1^*) = F(x_2^*) = 0.29864$, Global minima: $F(x_3^*) = 0$

5. Brown badly scaled function {BBSF} (4)

   (a) $n = 2, m = 3$

   (b) $r_1 = x_1 - 10^6, \quad r_2 = x_2 - 2 \cdot 10^{-6}, \quad r_3 = x_1x_2 - 2$

   (c) $x_{10} = [1, 1], \quad x_{20} = [-1, -1], \quad x_{30} = [2, 5], \quad x_{40} = [-3, 2]$

   (d) $x^* = [10^6, 2 \cdot 10^6]$

   (e) $F(x^*) = 0$

6. Modified Barbashin and Krasovskii Function 1 {Mod. BK 1}

   (a) $n = 2, m = 3$

   (b) $r_1 = \frac{100x_1}{\sqrt{1+x_1^2}}, \quad r_2 = 10x_2, \quad r_3 = \sinh x_1$

   (c) $x_{10} = [4, 4], \quad x_{20} = [5, -6], \quad x_{30} = [-4, 8], \quad x_{40} = [-5, -8]$

   (d) $x^* = [0, 0]$

   (e) $F(x^*) = 0$

7. Modified Barbashin and Krasovskii Function 2 {Mod. BK 2}

   (a) $n = 2, m = 3$

   (b) $r_1 = \frac{100x_1}{\sqrt{1+x_1^2}}, \quad r_2 = 10x_2, \quad r_3 = \cosh x_1$

   (c) $x_{10} = [4, 4], \quad x_{20} = [5, -6], \quad x_{30} = [-4, 8], \quad x_{40} = [-5, -8]$

   (d) $x^* = [0, 0]$

   (e) $F(x^*) = 0.5$

8. Modified Rosenbrock Function 1 {Mod. RF 1}

(a) $n = 2, m = 3$

(b) $r_1 = a[x_2 - x_1^2]$, $\quad r_2 = b[x_1 - 1]$, $\quad r_3 = c[(x_2 + 1)^2 - x_1 + 1]$ with $a = b = c = 1$

(c) $x_{10} = [-1.2, 1]$, $\quad x_{20} = [2, 2]$, $\quad x_{30} = [-2, 3]$, $\quad x_{40} = [2, -3]$

(d) $x^* = [0.6423, -0.4127]$

(e) $F(x^*) = 0.6513$

9. Modified Rosenbrock Function 2 {Mod. RF 2}

(a) $n = 2, m = 3$

(b) $r_1 = a[x_2 - x_1^2]$, $\quad r_2 = b[x_1 - 1]$, $\quad r_3 = c[(x_2 + 1)^2 - x_1 + 1]$ with $a = 10$ and $b = c = 1$

(c) $x_{10} = [-1.2, 1]$, $\quad x_{20} = [0.9, 0.9]$, $\quad x_{30} = \left[-1, \frac{201}{200}\right]$, $\quad x_{40} = [1, -0.5]$

(d) $x^* = [0.3224, 0.0653]$

(e) $F(x^*) = 1.9467$

10. Modified Rosenbrock Function 3 {Mod. RF 3}

(a) $n = 2, m = 3$

(b) $r_1 = a[x_2 - x_1^2]$, $\quad r_2 = b[x_1 - 1]$, $\quad r_3 = c[(x_2 + 1)^2 - x_1 + 1]$ with $a = 100$, $b = 10$, and $c = 1$

(c) $x_{10} = [-1.2, 1]$, $\quad x_{20} = [0.9, 0.9]$, $\quad x_{30} = \left[-1, \frac{201}{200}\right]$, $\quad x_{40} = [1, -0.5]$

(d) $x^* = [0.8493, 0.7203]$

(e) $F(x^*) = 5.9771$

11. Beale Function {BF}(5)

(a) $n = 2, m = 3$

(b) $r_i = y_i - x_1(1 - x_2^i)$ where $y_1 = 1.5$, $y_2 = 2.25$, $y_3 = 2.625$

(c) $x_{10} = [1, 1]$, $\quad x_{20} = [10, 2]$, $\quad x_{30} = [-5, -2]$, $\quad x_{40} = [8, -2]$

(d) $x^* = [3, 0.5]$

(e) $F(x^*) = 0$

12. Jenrich and Sampson Function {J&S} (6)

(a) $n = 2, m = 10$

(b) $r_i = 2 + 2i - [\mathrm{e}^{ix_1} + \mathrm{e}^{ix_2}]$

(c) $x_{10} = [0.3, 0.4], \quad x_{20} = [-0.2, 0.4], \quad x_{30} = [0.5, -0.1], \quad x_{40} = [0, 0]$

(d) $x^* = [0.2578, 0.2578]$

(e) $F(x^*) = 62.181$

A.2. The multi-variable NLS test problems

A. Box Three-dimensional Function {B3DF} (12)

(a) $n = 3, m = 10$

(b) $r_i = \mathrm{e}^{-\theta_i x_1} - \mathrm{e}^{-\theta_i x_2} - x_3 \left[\mathrm{e}^{-\theta_i} - \mathrm{e}^{-10\theta_i}\right]$ where $\theta_i = 0.1i$

(c) $x_0 = [0, 10, 20]$

(d) $x_1^* = [1, 10, 1]$, $x_2^* = [10, 1, -1]$ and whenever $[x_1 = x_2 \quad \text{and} \quad x_3 = 0]$

(e) $F(x^*) = 0$ for all x_s^*

B. Gulf Research and Development Function {GRDF} (12)

(a) $n = 3, m = 10$

(b) $r_i = \mathrm{e}^{-\frac{|y_i m i x_2|^{x_3}}{x_1}} - \theta_i$ where $\theta_i = \frac{i}{100} \quad \text{and} \quad y_i = 25 + [-50 \ln(\theta_i)]^{\frac{2}{3}}$

(c) $x_0 = [5, 2.5, 0.15]$

(d) $x^* = [50, 25, 1.5]$

(e) $F(x^*) = 0$

C. Bard Function {BF} (8)

(a) $n = 3, m = 15$

(b) $r_i = y_i - \left(x_1 + \frac{u_i}{v_i x_2 + w_i x_3}\right)$ where $u_i = i$, $v_i = 16 - i$, $w_i = \min(u_i, v_i)$ and

i	y_i	i	y_i	i	y_i
1	0.14	6	0.32	11	0.73
2	0.18	7	0.35	12	0.96
3	0.22	8	0.39	13	1.34
4	0.25	9	0.37	14	2.10
5	0.29	10	0.58	15	4.39

(c) $x_0 = [1, 1, 1]$

(d) $x_1^* = [0.8406\ldots, -\infty, -\infty]$, x_2^* is not available

(e) $F(x_1^*) = 8.7143\ldots$, $F(x_2^*) = 4.107435\ldots \times 10^{-3}$

D. Gaussian Function {GF} (9)

(a) $n = 3, m = 15$

(b) $r_i = x_1 \mathrm{e}^{\frac{-x_2(\theta_i - x_3)^2}{2}} - y_i$ where $\theta_i = \frac{8-i}{2}$ and

i	y_i
1,15	0.0009
2,14	0.0044
3,13	0.0175
4,12	0.0540
5,11	0.1295
6,10	0.2420
7,9	0.3521
8	0.3989

(c) $x_0 = [0.4, 1, 0]$

(d) Not available

(e) $F(x^*) = 5.63965\ldots \times 10^{-9}$

E. Meyer Function {MF} (10)

(a) $n = 3, m = 16$

(b) $r_i = x_1 \mathrm{e}^{\left[\frac{x_2}{\theta_i + x_3}\right]} - y_i$ where $\theta_i = 45 + 5i$ and

i	y_i	i	y_i
1	34780	9	8261
2	28610	10	7030
3	23650	11	6005
4	19630	12	5147
5	16370	13	4427
6	13720	14	3820
7	11540	15	3307
8	9744	16	2872

(c) $x_0 = [0.02, 4000, 250]$

(d) Not available

(e) $F(x^*) = 43.9729\ldots$

F. Wood Function {WF} (14)

(a) $n = 4, m = 6$

(b) $r_1 = 10(x_2 - x_1^2)$, $r_2 = 1 - x_1$, $r_3 = \sqrt{90}(x_4 - x_3^2)$, $r_4 = 1 - x_3$, $r_5 = \sqrt{10}(x_2 + x_4 - 2)$, $r_6 = \frac{1}{\sqrt{10}}(x_2 - x_4)$

(c) $x_0 = [-3, -1, -3, -1]$

(d) $x^* = [1, 1, 1, 1]$

(e) $F(x^*) = 0$

G. Colville Function {CF} [2.10]

(a) $n = 4, m = 7$

(b) $r_1 = 10(x_1^2 - x_2)$, $r_2 = x_1 - 1$, $r_3 = x_3 - 1$, $r_4 = \sqrt{90}(x_3^2 - x_4)$, $r_5 = \sqrt{10.1}(x_2 - 1)$, $r_6 = \sqrt{10.1}(x_4 - 1)$, $r_7 = \sqrt{19.8(x_2 - 1)(x_4 - 1)}$

(c) $x_0 = [10, 10, 10, 10]$

(d) $x^* = [1, 1, 1, 1]$

(e) $F(x^*) = 0$

H. Kowalik and Osborne Function {K&OF} (15)

(a) $n = 4, m = 11$

(b) $r_i = y_i - \frac{x_1(t_i^2+t_i x_2)}{t_i^2+t_i x_3+x_4}$ where

i	y_i	t_i	i	y_i	t_i
1	0.1957	4.0000	7	0.0456	0.1250
2	0.1947	2.0000	8	0.0342	0.1000
3	0.1735	1.0000	9	0.0323	0.0833
4	0.1600	0.5000	10	0.0235	0.0714
5	0.0844	0.2500	11	0.0246	0.0625
6	0.0627	0.1670			

(c) $x_0 = [0.25, 0.39, 0.415, 0.39]$

(d) x_1^* is not available, $x_2^* = [+\infty, -14.07\ldots, -\infty, -\infty]$

(e) $F(x_1^*) = 1.537525\ldots \times 10^{-4}$, $F(x_2^*) = 5.1367\ldots \times 10^{-4}$

I. Brown and Dennis Function {B&DF} (16)

(a) $n = 4, m = 20$

(b) $r_i = (x_1 + \theta_i x_2 - e^{\theta_i})^2 + (x_3 + x_4 \sin\theta_i - \cos^2\theta_i)$ where $\theta_i = \frac{i}{5}$

(c) $x_0 = [25, 5, -5, -1]$

(d) Not available

(e) $F(x^*) = 42911.1\ldots$

J(i). Penalty Function I {PF I} (23)

(a) $n = 4, m = 5$

(b) $r_i = \sqrt{a}(x_1 - 1)$ for $1 \leqslant i \leqslant n$ and $r_{n+1} = \left(\sum_{j=1}^{n} x_j^2\right) - \frac{1}{4}$ where $a = 10^{-5}$

(c) $x_0 = [\xi_j]$ where $\xi_j = j$

(d) Not available

(e) $F(x^*) = 1.124985\ldots \times 10^{-5}$

J(ii). Penalty Function I {PF I} (23)

(a) $n = 4, m = 10$

(b) $r_i = \sqrt{a}(x_1 - 1)$ for $1 \leqslant i \leqslant n$ and $r_{n+1} = \left(\sum_{j=1}^{n} x_j^2\right) - \frac{1}{4}$ where $a = 10^{-5}$

(c) $x_0 = [\xi_j]$ where $\xi_j = j$

(d) Not available

(e) $F(x^*) = 3.543825\ldots \times 10^{-5}$

K(i). Penalty Function II {PF II} (24)

(a) $n = 4, m = 8$

(b) $r_1 = x_1 - 0.2$,

$r_i = \sqrt{a}\left(e^{(0.1x_1)} + e^{0.1(x_{i-1})} - y_i\right)$ for $2 \leqslant i \leqslant n$,

$r_i = \sqrt{a}\left(e^{0.1(x_{i-n+1})} - e^{-0.1}\right)$ for $n < i < 2n$,

$r_{2n} = \left(\sum_{j=1}^{n}(n - J + 1)x_j^2\right) - 1$

where $a = 10^{-5}$ and $y_i = e^{0.1i} + e^{0.1(i-1)}$

(c) $x_0 = [0.5, 0.5, \ldots, 0.5]$

(d) Not available

(e) $F(x^*) = 4.688145\ldots \times 10^{-6}$

K(ii). Penalty Function II {PF II} (24)

(a) $n = 4, m = 10$

(b) $r_1 = x_1 - 0.2$,

$r_i = \sqrt{a}\left(e^{(0.1x_1)} + e^{0.1(x_{i-1})} - y_i\right)$ for $2 \leqslant i \leqslant n$,

$r_i = \sqrt{a}\left(e^{0.1(x_{i-n+1})} - e^{-0.1}\right)$ for $n < i < 2n$,

$r_{2n} = \left(\sum_{j=1}^{n}(n - J + 1)x_j^2\right) - 1$

where $a = 10^{-5}$ and $y_i = e^{0.1i} + e^{0.1(i-1)}$

(c) $x_0 = [0.5, 0.5, \ldots, 0.5]$

(d) Not available

(e) $F(x^*) = 1.4683\ldots \times 10^{-4}$

L. Hyper-ellipsoid Function II {HeF } [2.21]

(a) $n = 8, m = 9$

(b) $r_i = \sqrt{2}x_i^2$ for $1 \leqslant i \leqslant 8$ and $r_9 = \sqrt{2 + 2.2^2 + 2.2^3 + \ldots + 2.2^8}$

(c) $x_0 = [-1, 2, -3, 4, -5, 6, -7, 8]$

(d) $x^* = [0, \ldots, 0]$

(e) $F(x^*) = 1.7650828 \times 10^7$

M. Osborne I Function {Os I} (17)

(a) $n = 5, m = 33$

(b) $r_i = y_i - (x_1 + x_2 e^{-\theta_i x_4}) + x_3 e^{-\theta_i x_5}$ where $\theta_i = 10(i - 1)$ and

i	y_i	i	y_i	i	y_i	i	y_i
1	0.844	10	0.784	19	0.538	28	0.431
2	0.908	11	0.751	20	0.522	29	0.424
3	0.932	12	0.718	21	0.506	30	0.420
4	0.936	13	0.685	22	0.490	31	0.414
5	0.925	14	0.658	23	0.478	32	0.411
6	0.908	15	0.628	24	0.467	33	0.406
7	0.881	16	0.603	25	0.457		
8	0.850	17	0.580	26	0.448		
9	0.818	18	0.558	27	0.438		

(c) $x_0 = [0.5, 1.5, -1, 0.01, 0.02]$

(d) Not available

(e) $F(x^*) = 2.732445\ldots \times 10^{-5}$

N. Biggs EXP6 Function {BEXP6F} (18)

(a) $n = 6, m = 13$

(b) $r_i = x_3 e^{-\theta_i x_1} - x_4 e^{-\theta_i x_2} + x_6 e^{-\theta_i x_5} - y_i$ where $\theta_i = 0.1i$

(c) $x_0 = [1, 2, 1, 1, 1, 1]$

(d) $x_1^* = [1, 10, 1, 5, 4, 3]$, x_2^* is not available

(e) $F(x_1^*) = 0$, $F(x_2^*) = 2.827825\ldots \times 10^{-3}$

O. Variably Dimensioned Function {VDF} (18)

(a) $n = 8, m = 10$

(b) $r_i = x_i - 1$ for $i = 1, \ldots, n$

$r_{n+1} = \sum_{j=1}^{n} j(x_j - 1)$

$r_{n+2} = \left(\sum_{j=1}^{n} j(x_j - 1)\right)^2$

(c) $x_0 = [\xi_j]$ where $\xi_j = 1 - \left(\frac{j}{n}\right)$

(d) $x^* = [1, \ldots, 1]$

(e) $F(x^*) = 0$

P. Griewank Function {GrF} (18)

(a) $n = 10, m = 11$

(b) $r_i = \sqrt{\frac{2}{4000}} x_i$ for $1 \leqslant i \leqslant 10$ and $r_{11} = \sqrt{2 - 2\Pi_{j=1}^{10} \cos\left(\frac{x_j}{\sqrt{j}}\right)}$

(c) $x_0 = [1, -1, 1, -1, 1, -1, 1, -1, 1, -1]$

(d) $x^* = [0, \ldots, 0]$

(e) $F(x^*) = 0$

Q. Osborne II Function {Os II} (19)

(a) $n = 11, m = 65$

(b) $r_i = y_i - \left(x_1 e^{-\theta_i x_5} + x_2 e^{-(\theta_i - x_9)^2 x_6} + x_3 e^{-(\theta_i - x_{10})^2 x_7} + x_4 e^{-(\theta_i - x_{11})^2 x_8}\right)$

where $\theta_i = \frac{i-1}{10}$ and

i	y_i	i	y_i	i	y_i
1	1.366	23	0.694	45	0.672
2	1.191	24	0.644	46	0.708
3	1.112	25	0.624	47	0.633
4	1.013	26	0.661	48	0.668
5	0.991	27	0.612	49	0.645
6	0.885	28	0.558	50	0.632
7	0.831	29	0.533	51	0.591
8	0.847	30	0.495	52	0.559
9	0.786	31	0.500	53	0.597
10	0.725	32	0.423	54	0.625
11	0.746	33	0.395	55	0.739
12	0.679	34	0.375	56	0.710
13	0.608	35	0.372	57	0.729
14	0.655	36	0.391	58	0.720
15	0.616	37	0.396	59	0.636
16	0.606	38	0.405	60	0.581
17	0.602	39	0.428	61	0.428
18	0.626	40	0.429	62	0.292
19	0.651	41	0.523	63	0.162
20	0.724	42	0.562	64	0.098
21	0.649	43	0.607	65	0.054
22	0.649	44	0.653		

(c) $x_0 = [1.3, 0.65, 0.65, 0.7, 0.6, 3, 5, 7, 2, 4.5, 5.5]$

(d) Not available

(e) $F(x^*) = 2.006885\ldots 10^{-2}$

R. $n-$dimensional Levy Function {n-D LvF} [2.29]

(a) $n = 20, m = 21$

(b) $r_1 = \sqrt{2}\sin\left(\frac{(3+x_1)\pi}{4}\right)$,

$r_i = \sum_{j=1}^{20}\left(\frac{3+x_j}{4} - 1\right)\sqrt{2 + 20\sin^2\left(\frac{(3+x_j)\pi}{4} + 1\right)}$ for $2 \leqslant i \leqslant 20$,

$$r_{21} = \left(\frac{3+x_{20}}{4} - 1\right)\sqrt{2 + 2\sin^2\left(\frac{2\pi(3+x_{20})}{4}\right)}$$

(c) $x_0 = [-1, 2, -1, 2, -1, 2, -3, 1, -2, 3, -2, 1, 1, -2, 3, -1, 2, 1, 1, 0]$

(d) $x_1^* = [1, \ldots, 1]$

(e) $F(x^*) = 0$

BIBLIOGRAPHY

Adorio, E.P. 2005. *MVF–Multivariate test functions library in C for unconstrained global optimization*, Department of Mathematics, U.P. Diliman.

Barbashin, E.A. and N.N. Krasovskii. 1952. *On the stability of a motion in the large*, Dokl Akad Nauk SSR **86**, 453–456.

Björck, A. 1996. *Numerical Methods for Least Squares Problems,* SIAM, Philadelphia.

Bongartz, I, A.R. Conn, N. Gould, and Ph.L. Toint. 1995. *CUTE: constrained and unconstrained testing environment*, ACM Transactions on Mathematical Software **21**, no. 1, 123–160.

Boukamp, B.A. 1986. *A nonlinear least squares fit procedure for analysis of immittance data of electrochemical systems,* Solid State Ionics **20**, 31–44.

Chen, J. and W. Li. 2005. *Convergence of Gauss-Newton's method and uniqueness of the solution*, Applied Mathematics and Computation **170**, 686–705.

Chudamani, R., K. Vasudevan, and C.S. Ramalingam. 2009. *Real-time estimation of power system frequency using nonlinear least squares*, IEEE Transcations on Power Delivery **24**, no. 3, 1021–1028.

Dennis, J.E., D.M. Gay, and R.E. Welsch. 1981. *Algorithm 573—NL2SOL, An adaptive nonlinear least-square algorithm,* ACM Transcations on Mathematical Software **7**, 348–368.

Dennis, J.E. and R.B. Schnabel. 1983. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice hall, Englewood Cliffs, N.J.

Dolan, E.D. and J.J. Moré. 2002. *Benchmarking optimization software with performance profiles,* Mathematical Programming **91**, 201–213.

El-Hayek, N., N. Anwer, H. Nouira, O. Gibaru, M. Damak, and P. Bourdet. 2015. *3D measurement and characterization of ultra-precision aspheric surfaces,* 13th CIRP Conference on Computer Aided Tolerancing, 41–46.

Eriksson, J. 1996. *Optimization and Regulation of Nonlinear Least Squares Problems, PhD Thesis,* Umeå University, Sweden.

Fletcher, R. 1987. *Practical Methods of Optimization*, Wiley, New York.

Gander, W., M.J. Gander, and F. Kwok. 2014. *Scientific Computing—An Introduction using Maple and Matlab*, Springer, Switzerland.

Gavin, H.P. 2015. *The Levenberg-Marquardt method for nonlinear least squares curve-fitting problems*, Department of Civil and Environmental Engineering, Duke University.

Gibson, T.E., A. Bashan, H.T. Cao, S.T. Weiss, and Y.Y. Liu. 2016. *On the origins and control of community types in the human microbiome*, PLoS Computational Biology **12**, no. 2.

Gill, P.E., W. Murray, M.A. Saunders, and M.H. Wright. 1983. *Computing Forward-Difference Intervals for Numerical Optimization*, Vol. 4, SIAM Journal on Scientific and Statistical Computing.

Goh, B.S. 1997. *Algorithms for unconstrained optimization problems via control theory*, Journal of Optimization Theory and Applications **92**, 581–604.

Goh, B.S., F. Ye, and S.S. Zhou. 2008. *Steepest descent algorithms in optimization with good convergence properties*, Chinese Control and Decision Conference, 1526–1530.

Goh, B.S. 2009. *Greatest descent algorithms in unconstrained optimization*, Journal of Optimization Theory and Applications **142**, 275–289.

Goh, B.S. 2010. *Convergence of algorithms in optimization and solutions of nonlinear equations*, Journal of Optimization Theory and Applications **144**, 43–55.

Goh, B.S. 2011. *Approximate greatest descent methods for optimization with equality constraints*, Journal of Optimization Theory and Applications **148**, 505–527.

Goh, B.S. 2012. *Numerical method in optimization as a multi-stage decision control system*, Latest Advances in Systems Science and Computational Intelligence.

Goh, B.S., W.J. Leong, and K.L. Teo. 2014. *"Robustness of convergence proofs in numerical methods in unconstrained optimization", in Optimization and Control Methods in Industrial Engineering and Construction, Intelligent System, Control and Automation: Science and Engineering*, Springer Science+Business Media Dordrecht **72**, 1–9.

Goh, B.S. and D.B. McDonald. 2015. *Newton Methods to solve a system of nonlinear algebraic equations*, Journal of Optimization Theory and Applications **164**, 261–276.

Grantham, W.J. 2003. *Trajectory following optimization by gradient transformation differential equations*, In: Proceedings of the 42nd IEEE Conference on Decision and Control, 5496–5501.

Grantham, W.J. 2007. *Gradient transformation trajectory following algorithms for determining stationary min-max saddle points*, In: Jørgensen S., Quincampoix M., Vincent T.L. (eds) Advances in Dynamic Game Theory. Annals of the International Society of Dynamic Games **9**, 639–657.

Han, H., W. Sun, J. Han, and R.J.B. Sampaio. 2005. *An adaptive conic trust-region method for unconstrained optimization*, Optimization Methods and Software **20**, no. 6, 665-667.

Han, T. and Y. Han. 2010. *Solving large scale nonlinear equations by a new ODE numerical integration method*, Applied Mathematics **1**, 222–229.

Hansen, P.C., V. Pereyra, and G. Scherer. 2013. *Least Squares Data Fitting with Applications*, The Johns Hopkins University Press, Maryland.

Hillstrom, K.E. 1977. *A simulation test approach to the evaluation of nonlinear optimzation algorithms*, ACM Transactions on Mathematical Software **3**, 305–315.

Kalman, R.E. and J.E. Bertram. 1960a. *Control system analysis and design via the "second method" of Lyapunov, I: continuous-time system*, Journal of Basic Engineering **82**, no. 2, 371–393.

Kalman, R.E. and J.E. Bertram. 1960b. *Control system analysis and design via the "second method" of Lyapunov, II: discrete-time system*, Journal of Basic Engineering **82**, no. 2, 394–400.

Kelly, C.T. 1999. *Iterative Methods in Optimization*, SIAM, Philadelphia.

Koesler, S. and M. Schymura. 2015. *Substitution elasticities in a constant elasticity of substitution framework — Empirical estimates using nonlinear least squares*, Economic Systems Research **27**, no. 1, 101–121.

LaSalle, J.P. 1964. *Recent advances in Liapunov stability theory*, SIAM Review **6**, no. 1, 1–11.

LaSalle, J.P. 1976. *The Stability of Dynamical Systems*, SIAM, Philadelphia.

Leong, W.J. and B.S. Goh. 2013. *Convergence and stability of line search methods for unconstrained optimization*, Acta Applicandae Mathematicae **127**, 155–167.

Levenberg, K. 1944. *A method for the solution of certain non-linear problems in least squares*, The Quarterly of Apllied Mathematics **2**, 164–168.

Lourakis, M.I.A. 2005. *A Brief Description of the Levenberg-Marquardt Algorithm Implemented by Lavmar*, Institute of Computer Science, Foundation for Research and Technology, Hellas.

Luo, X.L., C.T. Kelly, L.Z. Liao, and H.W. Tam. 2009. *Combining trust-region techniques and Rosenbrock methods to compute stationary points*, Journal of Optimization Thoery and Applications **140**, no. 2, 265–286.

Madsen, K., H.B. Nielsen, and O. Tingleff. 2004. *Methods For Non-linear Least Squares Problems*, Informatics and Mathematical Modelling, Technical University of Denmark.

Marquardt, D.W. 1963. *An algorithm for least-squares estimation of nonlinear parameters*, Journal of the Society for Industrial and Applied Mathematics **11**, no. 2, 431–441.

Moré, J.J., B.S. Garbow, and K.E. Hillstrom. 1981. *Testing unconstrained optimization*, ACM Transactions on Mathematical Software **7(1)**, 17–41.

Moré, J.J. 1977. *"The Levenberg-Marquardt algorithm: implementation and theory", in Numerical Analysis*, Watson, G.A., ed., Vol. 630, Springer-Verlag, Berlin.

Nocedal, J. and S.J. Wright. 2006. *Numerical Optimization*, Springer, Uinted States of America.

Ortega, J.M. 1973. *Stabilty of difference equations and convergence of iterative process*, SIAM Journal on Numerical Analysis **10**, no. 2, 268–282.

Osborne, M.R. 1976. *Nonlinear least squares—the Levenberg algorithm revisited*, Journal of Australian Mathematical Society **19(Series B)**, 343–357.

Parks, P.C. 1992. *A.M. Lyapunov's stability theory-100 years on*, IMA Journal of Mathematical Control and Information **9**, 275–303.

Pav, S.E. 2005. *Numerical Methods Course Notes*, Department of Mathematics, University of California, San Diego, La Jolla.

Powell, M.J.D. 1975. *"Convergence properties of a class of minimization algorithms", in Nonlinear Programming*, Vol. 2, O. Mangasarian, R. Meyer, and S. Robinson, eds. Academic Press, New York.

Sapienza, L., M. Davanço, A. Badolato, and K. Srinivasan. 2015. *Nanoscale optical positioning of single quantum dots for bright and pure single-photon emission*, Nature Communications.

Schafer, R., S. Avery, P. May, D. Rajopadhyaya, and C. Williams. 2002. *Estimation of rainfall drop size distributions from dual-frequency wind profiler spectra using deconvolution and a nonlinear least squares fitting technique*, Journal of Atmospheric and Oceanic Technology **19**, 864–874.

Simionescu, P.A. and M. Mehrubeoglu. 2012. *New concepts on two-dimensional data visualization with applications in engineering analysis and design*, Journal of Computing and Information Science in Engineering, 024501-1–024501-10.

Spalek, T., P. Pietrzyk, and Z. Sojka. 2005. *Application of the genetic algorithm joint with the Powell method to nonlinear least-squares fitting of powder EPR spectra*, Journal of Chemical Information and Modeling **45**, 18–29.

Transtrum, M.K. and J.P. Sethna. 2012. *Improvements to the Levenberg-Marquardt algorithm for nonlinear least-squares minimization*, Preprint submitted to Journal of Computational Physics.

Vincent, T.L. and W.J. Grantham. 1997. *Nonlinear and Optimal Control Systems*, Wiley, New York.

Wang, K., J. Chiasson, M. Bodson, and L.M. Tolbert. 2005. *A nonlinear least-squares approach for identification of the induction motor parameters*, IEEE Transactions on Automatic Control **50**, no. 10, 1622–1628.

Waseda, Y., H. Chen, K.T. Jacob, and H. Shibata. 2008. *On the glass forming ability of liquid alloys*, Science and Technology of Advanced Materials **9**, no. 2.

Weng, Y., W. Xu, Y. Wu, K. Zhou, and B. Guo. 2006. *2D shape deformation using nonlinear least squares optimization*, Visual Computer **22**, 653–660.

Wolfe, P. 1969. *Convergence conditions for ascent methods*, SIAM Review **11**, 226–235.

Wolfe, P. 1971. *Convergence conditions for ascent methods. II: Some corrections*, SIAM Review **13**, 185–188.

Wraith, J.M. and D. Or. 1998. *Nonlinear parameter estimation using spreadsheet software*, Journal of Natural Resources and Life Sciences Education **27**, 13–19.

Yuan, Y. 1999. *Review of trust region algorithms for optimization*, ICM99: Proceedings of the Fourth International Congress on Industrial and Applied Mathematics.

Yuan, Y. 2011. *Recent advances in numerical methods for nonlinear equations and nonlinear least squares*, Numerical Algebra, Control and Optimization **1**, no. 1, 15–34.