Addis Ababa University
College of Natural and Computational Science
Computational Science

Graduate Program

Application of Newton and Quasi-Newton Techniques to Non-Linear problems in Computational Mechanics

By

Fikru Habtamu Shibeshi

Supervisor

Prof. Okey Oseloka Onyejekwe

A Thesis submitted to the School of graduate studies of Addis Ababa University in partial fulfillment for the degree of masters in Computational science

November 2020

Addis Ababa, Ethiopia

ABSTRACT

In the field of engineering, researches often come across strong nonlinear boundary value problems (BVPs) that cannot solve easily. Numerical convergence for many problems, typically solved by the Newton-Raphson algorithm, is sensitive to the initial guess and need computations of Jacobi and its inverse at each iteration. Emphasis in the present work is placed on the alternative approach, such as quasi-Newton, HM and optimization method.

Many problems in applied mechanics are reduced to the solutions of systems of nonlinear algebraic, transcendental equations containing an explicit parameter. These are problems in the areas of thermo-fluids, gas dynamics, deformable solids, heat transfer, biomechanics, optimal control and others. A parameter found in these models is not unique, and may be easily identified artificially. An important aspect of these problems is a question of the variation of the solution when parameter is incrementally changed.

The numerical solution of BVPs of ordinary differential equations (ODE's) relies heavily on methods for solving systems of algebraic equations. The choice of the optimal numerical method, which ensures the best convergence rate with minimum error for the corresponding system of nonlinear equations, is discussed. Some modifications of quasi-Newton's method for systems of ordinary nonlinear differential equations are apply and suggested. Effectiveness of the method is demonstrated by comparing the results with the analytic solution for model boundary value problem implemented using a MATLAB Program. The objective of the research is to investigate applicability of the method to the wide range of nonlinear boundary value problems in different areas of mechanics.

Different problems of applied mechanics and physics with dominant nonlinearities due to constituent models, and others are analyzed and solve in the present work. In this paper, the Newton's Homotopy analysis method (NHAM) is also applied on nonlinear boundary value problems(BVPs) of mechanics problems. The result from the method prove NHAM with Runge-Kutta steps, were significantly reliable and more accurate however, computation cost is high.

ACKNOWLEDGEMENT

First and foremost, thanks to God, for His ultimate blessings and guidance at each step of this work.

I would like to express my deepest gratitude, sincere thanks and appreciation to my supervisor Prof**.** Okey Oseloka Onyejekwe for his patience, motivation, valuable suggestions, guidance, continuing encouragement and constructive criticism throughout the period of this work. I also thank him for help in the preparation of this thesis and for his fatherly guidance.

Finally, I must express my very profound gratitude to my parents, specially my sister Eden Habtamu and her husband Nick for their laptop support for the study. My wife Fasik Alemayehu and lovely children for giving their time, support and all my best friends 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. Thank you.

LIST OF CONTENT

LIST OF TABLES

LIST OF FIGURE

ACRONYMS

- BVPs - Boundary value problems
- BCs - Boundary Conditions
- ODEs - Ordinary Differential Equations
- NLAEs – Nonlinear Algebraic Equations.
- HM - Homotopy Method.
- NHAM -Newton Homotopy Analysis Method
- TSMM - Trapezoidal, Simpson and Midpoint Method.
- RK - Runge Kutta
- NHDE - Newton Homotopy Differential Equation
- FD - Finite Difference Method
- DFP – Davidon-Fletcher-Powell.
- BFGS – Broyden-Fletcher-Gloldfarb-Shanno.
- LMM - Levenberg-Marquardt Method
- PDEs - Partial Differential Equations
- CPU - Central Processing Unit
- BC – Broyden Class.
- MBC – Modified Broyden Class.
- F - Function.
- u - Initial Value.
- u^* - Approximate Solution.
- J - Jacobian Matrix.
- A^{-1} - Invers of Matrix.
- λ - Lambda Parameter.
- I - Identity matrix

CHAPTER ONE

1. INTRODUCTION

A Boundary Value Problems (BVPs) is a differential equation together with a set of additional constraints, called the Boundary Conditions (BCs). There are several real physical phenomena in which BVPs problems represent the model for them, we found different areas of engineering and science examples can be found in Determine the deflection of a uniformly loaded beam with variable stiffness and supported at both endpoint [**1**] and heat flow problem in [**2**] which are mechanics problem.

When studying mechanics problem model by nonlinear BVPs, one might be interested in finding solutions, which also satisfy its constraint or boundary conditions (BCs). These BVPs, in most cases, do not have exact solutions and so, we need to use numerical algorithms to generate approximate solutions.

In this paper, we will describe many numerical methods to solve mechanics problem of boundary value problems (BVPs) and compare the methods depending on many criteria.

Consider a general second order nonlinear boundary value problems (BVPs) have a form

$$u'' = f\left(x,u,u'\right), \quad a \le x \le b \ . \tag{1.1}$$

subject to the boundary conditions

$$u\left(a\right)=\alpha \quad and \quad u\left(b\right)=\beta \tag{1.2}$$

The numerical solution of BVPs of ordinary differential equations (ODE's) relies heavily on methods for solving systems of algebraic equations. If the ODE system is nonlinear, then so are the algebraic systems that one must solve.

Newton–Raphson method is the most popular technique for solving nonlinear equations. It has quadratic rate of convergence, but one of its major disadvantages of this methods is that a computation of Jacobian matrix at each iteration. In order to diminish this problem, quasi-Newton method of rank-1can be used, which is presented by Broyden [**3**] .

Quasi-Newton methods, those that are generalizations of the one-dimensional secant method, have been found to be very successful methods for solving nonlinear algebraic systems. Over the last decade, great deals of progress have made in determining very effective quasi-Newton methods that declines its weakness. One weakness of Broyden

method is that, it is not self-correcting, this problem also gets improvement by Abu-Hour [**4**] that declines this problem.

The Newton and quasi-Newton methods, which are widely used, have advantage on their high speed of convergence once the initial guess of the root, which is sufficiently close to exact solution to ensure convergence. In order to overcome, the choice of a better guesses of exact solutions of both methods several effort and progress have made, HM by **Liao** [**5**] and further modified by different authors in [**6**,**7**], HM can solve nonlinear algebraic systems and converge to a desired solution without close choice of initial guess. Another method that avoid initial guess problem of the methods is Optimization methods, like Steepest Decent, BFGS, DFP, Newton and Levenberg-Marquardt propose by authors see [**8**,**9**,**10**,**11**].

Both Newton and Quasi-Newton methods are locally convergent algorithms, Damped Newton methods by [**12**] and Damped Broyden's that improves local convergence method to global convergent.

In the following, we consider the application of Newton's, its variation like quasi-Newton methods and their modification to the solution of model of mechanics problems of BVP's. These methods have intended primarily for use on large nonlinear algebraic systems with sparse Jacobians. The focus here is on ODE's for which the associated algebraic systems have sparse Jacobians and are so large that not only function and Jacobian evaluations but also storage and the cost of arithmetic are major concerns. The remainder of this introduction provides a very brief background on BVP's of ODE, certain procedures for solving them numerically, and basic concept associated with algebraic systems.

1.1. Background

BOUNDARY VALUE PROBLEMS

Definition -1, a boundary value problem is a differential equation together with a set of additional restraints, called the boundary conditions.

Definition -2, a solution to a boundary value problem is a solution to the differential equation, which also satisfies the boundary conditions.

Boundary Conditions come in many shapes

- **"Dirichlet" boundary conditions**

$y(0) = \alpha$	Straightforward to implement

❖ **"Neumann" boundary conditions**

$y'(0) = \gamma$	Requires special attention

❖ **"Robin" conditions**

$y(0) + c \cdot y'(0) = \kappa$	Requires same attention

Taylor's theorem: the Taylor series expansion of y(x) around x_0 [**13**]

$$y(x)=y(x_0)+y'(x_0).(x-x_0)+\frac{1}{2}y''(x_0).(x-x_0)^2+ \\ \ldots+\frac{1}{n!}y^n(x_0).(x-x_0)^n+R_n. \qquad (1.3)$$

The remainder R_n is given by

$$R_n=\frac{1}{(n+1)!}y^{(n+1)}(\xi).(x-x_0)^{n+1} \qquad (1.4)$$

Where $\xi \in$

1.1.1. Tridiagonal matrices

Tridiagonal matrices are those that have nonzero elements only on the diagonal and in the positions adjacent to the diagonal, they will be of special importance in certain differential equations.

For a tridiagonal matrix, only the nonzero values need to be recorded, and that means that the $n\times$ matrix can be compressed into a matrix of three columns and n rows [**2**].

It is often possible to reduce the computational work necessary for solving $Ax=b$ by taking into account special features of the coefficient matrix A, such as symmetry or sparseness. As an example, we now discuss briefly the solution of tridiagonal systems.

We say that the matrix $A=(a_{ij})$ of order n is tridiagonal if

$$a_{ij}=0 \quad whenever \quad |i-j|>1 \qquad (1.5)$$

In words, A is tridiagonal if the only nonzero entries of A lie on the diagonal of A. a_{ij}, $i,j=1,2,3,...n$ or the sub diagonal of A. $a_{i-1,i}$, $i=2,3,...n$ or the super diagonal of A. $a_{i,i-1}$, $i=2,3,...n$

1.1.2. A system of nonlinear equations

Definition_3. A function $f : R^n \to R$ is define as being nonlinear when it does not satisfy the *superposition principle* that is

$$f(x_1 + x_2 + ...) \neq f(x_1) + f(x_2) + ... \qquad (1.6)$$

Now that we know what the term nonlinear refers to, we can define a system of nonlinear *equations*.

Definition_4. [**14**] A system of nonlinear equations is a set of equations as the following:

$$\begin{aligned} f_1(x_1, x_2, ...x_n) &= 0 \ , \\ f_2(x_1, x_2, ...x_n) &= 0 \ , \\ &\vdots \\ f_n(x_1, x_2, ...x_n) &= 0 \ , \end{aligned} \qquad (1.7)$$

Where $(x_1, x_2, ..., x_n) \in R^n$ and each f_i is a nonlinear real function, $i = 1, 2, ..., n.$

Because systems of nonlinear equations cannot be solving as nicely as linear systems, we use procedures called iterative methods.

Definition_5. An iterative method is a procedure that is repeated over and over again, to find the root of an equation or find the solution of a system of equations.

In this paper, we must add some "regularizing" assumptions about f, to solve the problem f well posed. Typical assumptions include the following:

- ***Continuity***: A function f is continuous if it can be draw without lifting up a pen; more formally, f is continuous if the difference $f(x) - f(y)$ vanishes as $x \to y$.
- **Lipschitz**: A function f is Lipschitz continuous if there exists a constant c such that $|f(x) - f(y)| \leq c|x - y|$; Lipschitz functions need not be differentiable but are limited in their rates of change.
- ***Differentiability***: A function f is differentiable if its derivative f' exists for all x.

1.1.3. Convergence

One of the things we will discuss in this study for comparison purpose is the convergence of each of the numerical methods.

Definition_6. We say that a sequence *converges* if it has a limit.

Definition_7. Let p_n be a sequence that converges to p, where $p_n \neq p$. If constants $\lambda, \alpha > 0$ exist such that

$$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|^{\alpha}} = \lambda. \qquad (1.8)$$

Then, it is said that p_n *converges to* p *of order* α *with a constant* λ

- **Orders of convergences.**

To measure the speed of convergence, we use a concept called the "order of convergence." The three different orders of convergence are define here [**15**].

Definition_8. **Linear Convergence**.

It requires that the error be reduce by at least a constant factor less than 1 at each iteration:

$$|e_i + 1| \leq c|e_i| \qquad (1.9)$$

Definition_9. **Quadratic Convergence**.

It requires that the error occurred at each iteration is proportional to the square of the error on the previous iterations:

$$|e_i + 1| \leq c|e_i|^2 \qquad (1.10)$$

Definition_10. **Superlinear Convergence**.

It requires that the ratio of successive errors go to zero, specifically,

$$\lim_{i\to\infty} \frac{e_i + 1}{e_i} = 0. \qquad (1.11)$$

Note: In the case of numerical methods, the sequence of approximate solutions is converging to the root. If the convergence of an iterative method is more rapid, then a

solution may be reach in less iteration in comparison to another method with a slower convergence.

1.1.4 Jacobian and Hessian Matrix

The Jacobian matrix is a key concept of numerical methods in this work.

Definition_11. The Jacobian *matrix* is a matrix of first order partial derivatives of system of equations.

$$J(x)=\begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x) & \frac{\partial f_1}{\partial x_2}(x) & \dots & \frac{\partial f_1}{\partial x_n}(x) \\ \frac{\partial f_2}{\partial x_1}(x) & \frac{\partial f_2}{\partial x_2}(x) & \dots & \frac{\partial f_2}{\partial x_n}(x) \\ \vdots & & & \\ \frac{\partial f_n}{\partial x_1}(x) & \frac{\partial f_n}{\partial x_2}(x) & \dots & \frac{\partial f_n}{\partial x_n}(x) \end{bmatrix} \qquad (1.12)$$

Definition_12. The Hessian matrix is a matrix of second order partial derivatives of system of equations.

$$H(x)=\begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 x_2} & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 x_1} & \frac{\partial^2 f}{\partial x_2^2} & \dots & \frac{\partial^2 f}{\partial x_2 x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_n x_1} & \frac{\partial^2 f}{\partial x_n x_2} & \dots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \qquad (1.13)$$

1.1.5. Norms of Vectors

Let $x \in \Re^n \in \Re$ where

$x = [x_1, x_2, \ldots x_n]^T$.

We will discuss two types of vector norms in this paper, the l_2 *and* l_∞ norms.

Definition_13. The l_2 norm for the vector x is called, the *Euclidean norm* because it represents the length of the vector denoted by

$$\|x\| = \|x\|_2 = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2} \qquad (1.14)$$

Definition_15. The l_∞ norm represents the absolute value of the largest component in the vector x . It is denote by

$$\|x\|_\infty = \max_{1\le i\le n} |x_i| \qquad (1.15)$$

In the next units of this study, we will examine different numerical methods that will apply the terms we discussed in this section. These methods include: Newton's, Broyden's , modified of Broyden, safeguarded Newton and Broyden , Damped Newton and Broyden, Homotopy and unconstrained optimization methods with the Finite Difference method.

1.2. Problem Definitions

The main purpose of this paper is solving different mechanics problems of highly nonlinear boundary value problems (BVPs) by using different numerical methods. These problems have variety of applications in real life, in this section we are try to define each of these problems:

Problem-1- Bratu problem

The Bratu problem appears in a large variety of applications such as: Fuel ignition model of thermal combustion, radioactive heat transfer, thermal reaction, the Chandrasekhar model of the expansion of the universe, chemical reactor theory and nanotechnology [**16**].. The one-dimensional Bratu boundary value problem written as

$$u''(x) + \lambda e^{u(x)} = 0 \qquad (1.2.1)$$

Subject to the boundary conditions:

$$u(0) = u(1) = 0 \qquad (1.2.2)$$

The Analytic solution for this problem is

$$u(x) = -2\ln\left[\frac{\cosh(x-1/2)\theta/2}{\cosh(\theta/4)}\right] \qquad (1.2.3)$$

Where θ satisfies

$$\theta = \sqrt{2\lambda}\cosh\left(\frac{\theta}{4}\right) \qquad (1.2.4)$$

This problem derives its importance first from the combustion theory where it has been used for several applications and secondly, from the fact that its exact solution is

well known so that it has been widely applied to test the accuracy and efficiency of many numerical methods. This importance motivates the reason to apply different numerical methods to Bratu problem (1.2.1).

Problem-2- Troesch's problem

This problem arises in the investigation of the confinement of a plasma column by radiation pressure that was described and solved by [**17**]. Troesch's problem, which is given by

$$u'' = \lambda \sinh(\lambda u) \qquad (1.2.5)$$

Subject to the boundary conditions:

$$u(0) = 0 \quad and \quad u(1) = 1. \qquad (1.2.6)$$

Different researchers pointed out that this two-point boundary value problem is unstable and difficult to solve. The equation can be effectively solved by combination of different methods, [**18**]

Whose general solution given by [**19**]

$$v0 = w * sinh(\lambda x) \ where \quad w = \frac{\tanh(\lambda/4)}{\sinh(\lambda)},$$

$$v1 = -((w^3)\lambda exp(-\lambda x)/(2*exp(2\lambda)-2))((x-1)*(exp(2\lambda(x+1))-1)+(x+1)*\ldots$$
$$(exp(2\lambda)-exp(2\lambda x)));$$

$$v2 = m+(((q*exp(-3\lambda x))/(2*lam))*(((3/4)+(-x^2+6*x)*n^2+(-(3/2)*x-3)\lambda)*$$
$$exp((4*x+2)\lambda)-(1/8)*exp((6*x+2)\lambda)+(-(3/4)+(6*x+x^2)*n^2+(-(3/2)*x+3)\lambda)*$$
$$exp(2\lambda(x+1))+((3/4)+(-x^2+6*x)*nsq+(3+(3/2)\lambda)\lambda)*exp(2\lambda x)+(-(3/4)+$$
$$(6*x+xsq)*nsq+((3/2)*x-3)\lambda)*exp(4\lambda x)-(1/8)+(1/8)*exp(6\lambda x)+(1/8)*exp(2\lambda)));$$

$$u2(x)=v0+v1+v2,$$

$$u(x)=4\tanh^{-1}(u2(x))/n, \quad 0 \le x \le 1, \quad \lambda \ge 0. \qquad (1.2.7)$$

Problem -3- Heat Transfer problem

Consider a thin wire of length L, and radius r. Let the ends of the wire have a fixed temperature of 900 and let the surrounding region be $u_{sur} = 300$. Suppose the surface of the wire be cooled via radiation. The lateral surface area of a small cylindrical portion of

the wire has area $A = 2\pi rh$. Therefore, the heat leaving the lateral surface in Δt time is given by [**20**].

$$\Delta t\,(2\pi rh)\left(\varepsilon\sigma\left(u_{sur}^4 - u^4\right)\right)$$

Assume steady state heat diffusion in one direction and apply the Fourier heat law to get:

$$0 \approx \left(2\varepsilon\sigma / r\left(u_{sur}^4 - u^4\right)\right) + ku'' \qquad (1.2.8)$$

The continuous model for the heat transfer is

$$\frac{d^2u}{dx^2} = c\left(u_{sur}^4 - u^4\right), \qquad (1.2.9)$$

Where $c = 2\varepsilon\sigma / r$ and subject to boundary conditions:

$$u(0) = 900 \ \textit{and} \ \ u(L) = 900 \qquad (1.2.10)$$

The thermal conductivity will also be temperature dependent, but for simplicity assume k is constant and will be incorporate into c.

Problem-4- Thermal Explosion Problem

Thermal explosion denotes the rapid build of energy in systems subject to exothermic reactions, the rate of which rises with temperature. It is of practical importance in problems of fire and explosion safety. There are few models which this problem accompanied by chemical reaction [**21**].

Consider the following Thermal explosion model of one-dimensional nonlinear BVP

$$u'' + \lambda e^{u} = 0, \quad x \in (0,1), \qquad (1.2.11)$$

with the associated boundary conditions

$$u'(0) = 0,\ u(1) = 0.$$

The analytic solution of equation (1.2.11) can be found by subsequently multiplying equation (1.2.11) with derivative of u, integrate, isolate derivative of u and integrate again. This solution is given by

$$u(x) = \ln\left(\frac{2\mu^2}{\lambda}\right) - 2\ln\left(\cosh\left(\mu x\right)\right), \qquad (1.2.12)$$

Where the parameter μ satisfies the relation $\cosh\mu = \sqrt{2/\lambda\mu}$

A number of numerical methods are apply to obtain a numerical solution to the above real life problems, such as the Newton's method, modifications and its different variant like quasi-Newton of rank 1 and 2 and different HM and Optimization methods are some of the method used to approximate numerical solutions.

1.3. Objectives of the Study

General objective

Applying different nonlinear algebraic system solvers numerical methods on mechanics problems model by nonlinear BVPs and compare the methods based on their output.

Specific Objectives

This paper was conduct to achieve the following specific objectives:

- ✓ To transform our nonlinear BVP to NLAE's by Finite difference method.
- ✓ To code Finite difference approximation, Newton-Raphson's, Quasi-Newton (Broyden's), Safeguarded and Damped for Newton's and Broyden's, Modified Broyden's, Trapezoidal, Simpson and Midpoint (TSMM), NHAM with Euler, RK2,RK4,RK5 and RK6 NHDE steps, steepest Decent, BFGS, DFP, Newton's optimization and Levenberg-Marquardt of optimization methods using MATLAB.
- ✓ To apply the above numerical methods in solving mechanics problems model by BVP like Bratu, Troesch's, Heat Transfer and Thermal Explosion Problems.
- ✓ To compare the performance of algorithms by a certain criteria.
- ✓ Analyze the results of simulation and determine the efficiency of each method.

1.4. Scope of the Study

This study focuses on solving mechanics problems of BVP's by using, Newton's method and its variant methods and its modifications, NHAM and its variant and different optimization methods are used to handle this problems by approximation of the Jacobian according to the formula considered and then injected into the algorithm. The algorithm for all the methods is coded using MATLAB program.

1.5. Thesis Outline

The remainders of the thesis chapters are organize as follows.

Chapter 2 contains an overview of literature that supports our study. Its focus on review different numerical methods by different authors in different years, which are used as a

base for our study, like Newton-Raphson and Quasi-Newton and their modifications, the global convergent method HAM, different optimization methods and other related previous works.

Chapter 3 establishes the main framework for all the numerical methods shown in the paper, i.e. Finite difference approximations. It comprises first how to change nonlinear BVPs to system of nonlinear algebraic equations. Moreover, explain the basic numerical method, Newton, and their safeguard and Damped Newton methods. Finally, apply these methods to nonlinear BVPs that are defined in section (1.2).

Chapter 4 gives an overview of the numerical methods, Broyden's and their Safeguarded and Damped Broyden methods, starting from pure Broyden method, and its improvements are applied on selected problems defined in section (1.2). The tested problems result with these methods are compared by using their output from a MATLAB program.

Chapter 5 gives an overview of HAM and NHAM. The different variant of NHAM; which are solving with different RK methods in order to solve NHDE. Finally presents the application of the numerical methods, NHAM and its variant on the selected nonlinear BVPs and their numerical results present in both numeric and graphical forms, finally comparison of methods also done here.

Chapter 6 gives overview of Optimization methods, Newton, Quasi-Newtons (Davidon-Fletcher-Powell's (DFP), Broyden, Fletcher, Goldfarb, and Shannon (BFGS), Levenberg-Marquardt (LM) methods are discussed in order to solve NLSAEs. These methods apply on nonlinear BVPs, their numerical results are display for these methods by using a MATLAB program, and their results are compared.

Chapter 7 gives an overview of a Continuation by Parameter (CBP) method which can be used to solve BVPs like the other chapters, it also test the numerical output by using problems defined in section (1.2).

Chapter 8 summarizes the overall methods, comparison of the overall methods by using numerical results and outlines possible directions for further research.

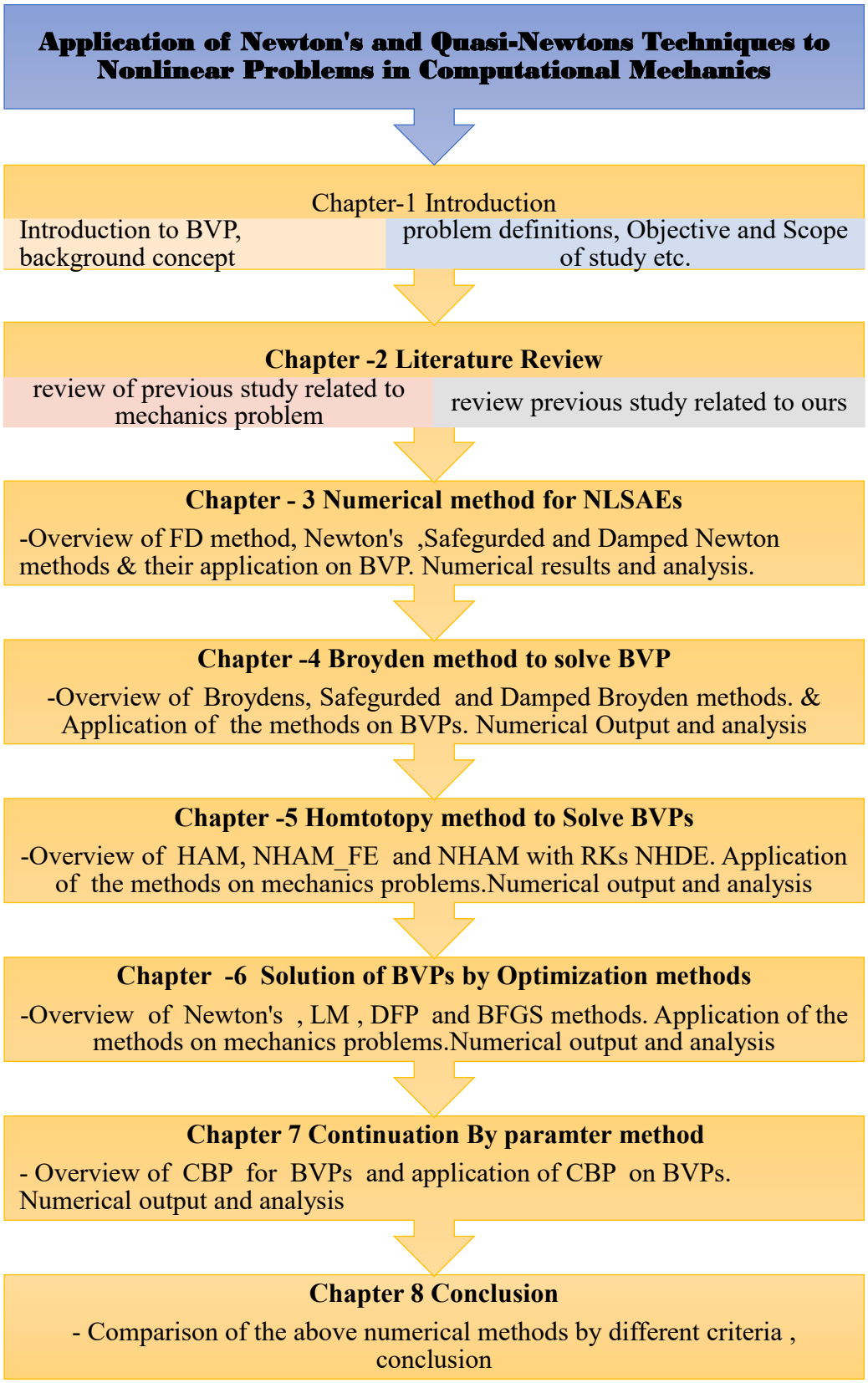

Figure 1.5.1: A Diagram of the main component of the Thesis

CHAPTER TWO

2. LITERATURE REVIEW

Introduction

Boyd [**22**]apply Chebychev pseudo spectral method to solve nonlinear BVPs of Bratu Problem, which models thermal Explosion, Caglar et.al. [**23**] Apply B-Spline method for solving the same problem. Troesch [**24**], apply Shooting method and Mirmoradi ***et.al*** [**25**] apply Homotopy perturbation method to solve nonlinear BVPs of Troesch's problem which models radiation pressure of confinement of a plasma column. Gerald and Wheatley [**2**] use different numerical methods, to solve heat flow problem in uniformly loaded beam model by second order nonlinear BVPs.

Finite Difference Approximation

Luenberger [**26**] shows Numerical Methods of solving ordinary differential equations, with basic techniques of finite difference methods for linear boundary value problem. Studies by **Mathews and Strang** [**27**]also shows basic techniques of FD method for linear BVP and **Lakshmi and Muthuselvi** [**28**], shows the basic techniques of FD method to solve linear BVP and they found numerical solution using MATLAB program and compare it with analytic solution.

Books by **Chapra** [**29**] and **King** *et al*. [**30**], shows how to transform the Boundary value problem to system of algebraic equations by using FD approximation and shows also how to solve the transformed BVP by using Thomas Algorithm, which solves tridiagonal matrix.

Recently, **Pandey** [**31**] presented an exponential finite difference scheme for solving nonlinear two point boundary value problems with Dirichlet's boundary conditions and discussed the local truncation error and the convergence of the proposed method. In the same year, **Becker** *et al*. [**32**]used finite difference to convert 1D implicit heat equation of Neumann and Dirichlet boundary condition to system of nonlinear algebraic equation and solve it with MATLAB with backward slash operator.

In this paper, we are taking the main concept of the researchers for FD method of approximation, to transform our nonlinear BVP's model with appropriate difference-quotient approximation to system of NLAE's.

Newton's Method

Dennis [**33**] shows the prominent method for solving nonlinear equation is the classical Newton's method, which converges quadratically, but the numerical implementation of algorithm of Newton's Methods turns to be expensive due to the computation for storing the Jacobian matrix, as well as solving the Newton's system at each iteration.

Recently, **Kisabo** *et al*. [**34**] Present an approach to explaining and implementing Newton's method as a numerical approach for solving Nonlinear System of Algebraic Equations (NLSAEs) by using MATLAB program. Due to their convergence rate and their applicability in different areas we are interested and applying these method to apply and solve our nonlinear BVP's models.

Quasi-Newton (Broyden) Method

The basic shortcomings of Newton's, like computation of Jacobian matrix have attracted the attention of many scholars, in 1965 a well-known Quasi Newton method proposed by **Broyden** [**10**] , called Broyden's which approximate the Jacobian matrix or its inverse by other matrix he also proved the superlinear convergence of the method.

Kelly [**35**] show a number of revised Newton's -type methods that is being introduced which includes fixed Newton's, inexact Newton's and quasi-Newton's to diminish the weakness of Newton's method .

One of variations of Newton's method known as the secant method, which is replacing derivative in newton step by the secant updating formula to solve equations in one dimension **Griewank** [**36**]. This idea led Broyden to develop three methods that use secant information instead of derivative information to solve nonlinear systems (two of them are implement in this paper). We are using this Broyden method to solve our BVP's models.

DAMPED NEWTON's Method

Robinson [**37**] shows damped Newton method for solving non- smooth equations. This damping, via the so-called path search instead of the traditional line search , enlarges the domain of convergence of Newton's method and therefore is said to be ***globally*** convergent.

Burdakov [**38**] shows Newton's method which is damped by a line search; that is by choosing x_{k+I} on the line segment from x_k to x'_{k+1} such that the decrease in norm of function.

Recently, **Zhang and Yu** [**39**]compare damped Newton method and Newton Homotopy method, these known methods for circumventing the drawback of local convergence of the standard Newton method.

In this paper, we are using both the Damped Newton and Newton Homotopy analysis method to avoid the local convergence of classical Newton method.

Two improved classes of Broyden's methods

Al-Towaiq *et al.* [**40**] , propose an algorithm that improve and modify Broyden's methods to increase the accuracy and the order of convergence.

In this paper, we are interested to apply this algorithm to solve our nonlinear BVPs and compare the result with pure Broyden's.

Trapezoidal, Simpson and Midpoint method (TSMM)

Quasi-Newton method reduce the amount of calculation at each iteration of Newton's method but subsequently lead to an increase in the number of iterations to be performed to converge to a solution of NLAE's .

Mohammed and **Waziri** [**41**] proposed quadrature based Broyden-like methods to solve systems of non-linear equations, **Mama**t *et al.* [**42**]also show a Broyden-like method using the Trapezoidal rule in order to reduce the number of iterations of the classical Broyden method. **Mohammad** and **Waziri** [**43**]used the weighted combination of the Midpoint and Simpson quadrature formulas to achieve the same goal as above.

Recently, **Osinuga** *et al*. [**44**], show the weighted combination of the Trapezoidal, Simpson and Midpoint quadrature rules for NL system of equations. Which is variation of Broyden-Like method and compare its result with previous Broyden-like methods.

In this paper, we are applying this weighted combination of Trapezoidal, Simpson and Midpoint Method (TSMM) to solve our nonlinear BVPs models and compare the result with pure Broyden Method.

Homototopy Analysis Methods

Newton's method still attracts attention of researchers. Even it is well-known, one of its drawback of the methods is that the initial approximation x_0 must be chosen sufficiently close to exact solution in order to guarantee their convergence. Finding a criterion for choosing x_0 is quite difficult and therefore effective and globally convergent algorithms has shown by **Yamamato** [**6**].

Liao [**7**] employed the basic ideas of Homotopy as a general method for nonlinear problems, namely the Homotopy analysis method (HAM), and then modified it systematically.

Recently, **Abbasbandy** *et al*. [**45**] , shows Homotopy Analysis method (HAM) to solve some nonlinear algebraic equations and show the solution by comparing with HPM and ADM solutions. Furthermore, combining the Newton scheme with HAM construct numerical algorithm named Newton_Homotopy Analysis method (NHAM). Finally test by using example, he suggest NHAM has powerful improvement for solving nonlinear equations.

To solve the nonlinear algebraic or differential equations by the NHAM, the existing (linear and nonlinear) terms need to be differentiated using the homotopy derivative concept named as Newton Homotopy Differential equation (NHDE), this first order differential equation can be solved by using Forward Euler method by a book Keskin [**20**]. The book also proposes that, the Euler step can be replace by Runge Kutta fourth order.

In this paper, we apply NHAM with Forward Euler step to solve nonlinear BVPs models. In addition to NHAM, which solve NHDE by forward Euler step like Keskin, we solve this NHDE with RK_2, RK_4, RK_5, RK_6 step and compare their result.

Optimization

Review of Optimization methods are summarized by **Brune** *et al.* [**46**]. He explains the idea of solving systems of nonlinear equations $f(x)=0$ by using Optimization methods derived from nonlinear preconditioning ideas.

Grosan and **Abraham** [**47**], shows how to transformed the system of nonlinear equations into a multi-objective optimization problem and solved it by computation technique. Recently **Song** *et al.* [**48**] proposed a transformation technique from the system of

nonlinear equations into a bi-objective optimization problem, and solve it by employing computation technique.

A book [**20**], present a transformation technique, that transform the system of nonlinear equations into an objective function of optimization problem, and solve it by using steepest Decent method. In this paper, we are interested in using these techniques to solve our system of nonlinear equations by using different methods of optimization methods like Steepest Decent, BFGS, DFP, Newton and Levenberg -Marquardt.

Quasi-Newton Methods.

There are various quasi-Newton methods proposed by different researchers. They differ in how they define and construct the quasi-Newton matrices $\{B_k\}$, how the search directions are computed and how the parameters of the model are updated **Nocedal** and **Wright** [**49**].

DFP Method

The first quasi-Newton method was, **Davidon** [**50**] and , **Fletcher** and **Powell** [**51**]. The DFP method updates the approximation Hessian matrix H_k *to* H_{k+1}, to satisfy the secant equation:

$$H_{k+1}\alpha_k = \gamma_k, \ \ where \ \ \alpha_k = g_{k+1} - g_k \ \ and \ \ \gamma_k = x_{k+1} - x_k$$

Powell [**52**], show that the DFP method with exact line searches is globally convergent for uniformly convex functions.

BFGS Method

The most efficient quasi-Newton method is the BFGS method, which was proposed by **Broyden** [**10**], by **Fletcher** [**51**], by **Goldfarb** [**53**], and by **Shanno** [**54**], independently, which produce a positive-definite matrix H_K for each iteration.

Levenberg-Marquardt (LM) Method

The classical Levenberg-Marquardt method by **Marquardt** [**55**] for solving nonlinear equation, $F(x) = 0$. Which computes a systematic modification of Gauss_Newton's step and also shows LM method has quadratic rate of convergence, if Jacobian at the solution x^* is nonsingular.

Yamashita and **Fukishima** [**56**]**,** shows that under the weaker condition of ||F(x)|| provides a local error bound near solution, but Levenberg_Marquardt method has quadratic convergence if the parameter is chosen properly.

CHAPTER THREE

3. Numerical Methods for Solution of NLAEs.

There are two main types of methods available to find the roots of algebraic and transcendental equations of the form $f(x)=0$ i.e. Direct and Indirect methods **Rao** [**57**].

The common characteristics of direct methods are that they transform the original equation into equivalent equations (equations that have the same solution) that can be solving more easily. The transformation is carrying out by applying certain operations.

Iterative or indirect methods, start with a guess of the solution x, and then repeatedly refine the solution until a certain convergence criterion is reached.

According to ***Rao*** Iterative procedures are self-correcting, meaning that round off errors (or even arithmetic mistakes) in one iteration are corrected in next consecutive cycles. The solution contains truncation error. However, iterative methods may not always converge to the solution. Choice of initial guess affects the number of iterations to convergence solution.

In this paper, we examined different indirect (iterative) methods and applied it to our mechanics problem modeled by nonlinear BVPs.

Before we solve BVP, let us see first how to transform the BVP into system of algebraic equations, by replacing all derivatives with finite difference approximations, called Finite difference method.

❖ Finite Difference Method for BVP

A finite difference method transforms BVPs to system of equations by replacing the derivatives in the differential equations with finite difference approximations. This gives a large but finite algebraic system of equations to be solving in place of the differential equation, something that can be done on a computer. [**58**]

Consider a simple two-point BVP:

$$y'' = f(x,y), \qquad a < x < b, \tag{3.1}$$

With boundary condition

$$y(a) = \alpha, \qquad y(b) = \beta. \tag{3.2}$$

The function $f(x)$ is specified and we wish to determine $y(x)$ in the interval $a<x<b$,we will attempt to compute a grid function consisting of values $y_0, y_1, ..., y_m, y_{m+1}$, where

y_m is our approximation to the solution $y(x_m)$. Here $x_m = m*h$ and $h=(b-a)/(m+1)$ is the mesh width, the distance between grid points. From the boundary conditions we know that $y_0 = \alpha$ ، and $y_{m+1} = \beta$, and so we have m unknown values $y_1,...,y_m$ to compute. If we replace $y''(x)$ in (3.1) by the centered difference approximation

$$y'' = \frac{1}{h^2}\left(y_{j-1} - 2y_j + y_{j+1}\right), \qquad (3.3)$$

then we obtain a set of algebraic equations

$$\frac{1}{h^2}\left(y_{j-1} - 2y_j + y_{j+1}\right) = f(x,y) \quad for \quad j = 1,2,...,m \qquad (3.4)$$

Note that the first equation when $(j=1)$ involves the value $y_0 = \alpha$ and the last equation $(j=m)$, involves the value $y_{m+1} = \beta$. We have a system of m equations for the m unknowns, which can be written in the form

$$Ay = F, \qquad (3.5)$$

Where y the vector of unknown $y = [y_1, y_2, ..., y_m]^T$ and

$$A = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{bmatrix} \qquad F = \begin{bmatrix} f(x_1) - \alpha/h^2 \\ f(x_2) \\ f(x_3) \\ \vdots \\ f(x_{m-1}) \\ f(x_m) - \beta/h^2 \end{bmatrix} \qquad (3.6)$$

As shown above, FD method convert nonlinear boundary value problems (BVP's) into a system of nonlinear algebraic equations that would be solve by matrix algebra techniques. The reduction of the differential equation to a system of algebraic equations makes the problem of finding the solution to a given ODE/PDE ideally suited to modern computers.

Due to this, Finite difference methods are the dominant approach to numerical solutions of BVPs by the same reason we are applying FD method to transform our nonlinear BVPs to system of NLAE's that can be solved by Newton's method, Quasi-Newton's method, modification of them, HAM and Optimization methods which are presented in the next sections.

Newton's Method

Newton's method is one of the most popular numerical methods, and referred by **Burden** and **Faires** [**59**] is the most powerful method that used to solve for the equation $f(x)=0$. This method originates from the Taylor's series expansion of the function $f(x)$ about the point x_1:

$$f(x)=f(x_1)+(x-x_1)f'(x_1)+\frac{1}{2!}(x-x_1)^2 f''(x_1)+... \qquad (3.7)$$

Where f, and its first and second order derivatives, f' *and* f'' are calculated at x_1. If we take the first order terms of the Taylor's series expansion of (3.7) and the remainder terms constitute the truncation error.

$$f(x)=f(x_1)+(x-x_1)f'(x_1) \qquad (3.8)$$

We then set (3.8) to zero (i.e. $f(x) = 0$) to find the root of the equation which gives us:

$$f(x_1)+(x-x_1)f'(x_1)=0 \qquad (3.9)$$

Rearranging equation (3.9), we obtain the next approximation to the root, giving us:

$$x=x_2=x_1-\frac{f(x_1)}{f'(x_1)} \qquad (3.10)$$

Thus generalizing (3.10), we obtain Newton's iterative method:

$$x_i=x_{i-1}-\frac{f(x_{i-1})}{f'(x_{i-1})},\ i\in N \qquad (3.11)$$

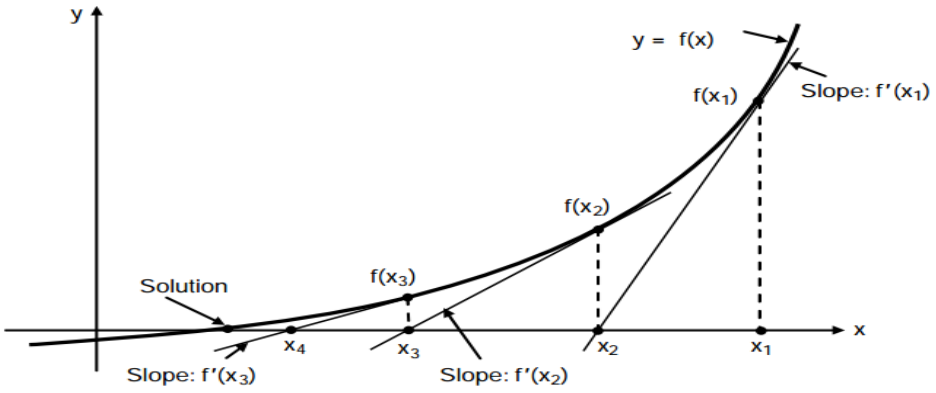

Figure 3.1: Graphical interpretation of the Newton–Raphson formula.

A graphical depiction of the Newton–Raphson formula by book **Kiusalaas**, [**60**] in Fig.3.1 shows, the formula approximates $f(x)$ by the straight line that is tangent to the curve at x_i .Thus x_{i+1} is at the intersection of the x-axis and the tangent line.

Where $x_i \to \bar{x}$ $(as \quad i \to \infty)$, and $\bar{x}$ is the approximation to a root of the function $f(x)$.

Remark-1-: As the iterations begin to have the same repeated values i.e. as $x_i = x_{i+1} = x$ this is an indication that $f(x)$ converges to $\bar{x}$ .Thus, x_i is the root of the function $f(x)$.

The algorithm for the Newton–Raphson method repeatedly applies to equation (3.11), starting with an initial value x_0, until the convergence criterion equation (3.12)

$$\left|x_{i+1} - x_i\right| < \varepsilon \qquad (3.12)$$

is reached, ε being the error tolerance. Only the latest value of x has to be stored.

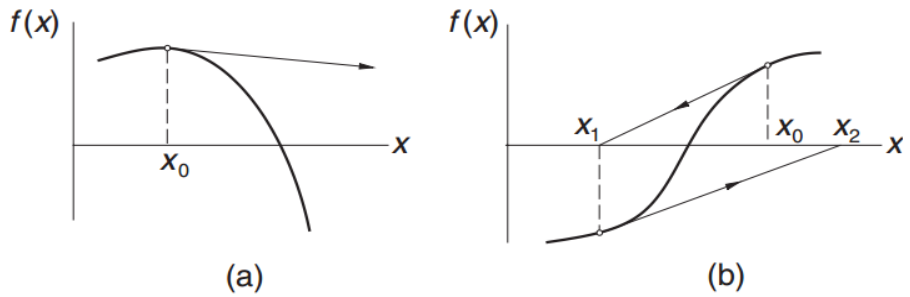

Figure 3.2. Examples where the Newton–Raphson method diverges.

Although the Newton–Raphson method converges fast near the root, its global convergence characteristics are poor. The reason is that the tangent line is not always an acceptable approximation of the function, as illustrated in the two examples in Fig.3.2.

3.1. Newton-Raphson Method for BVPs

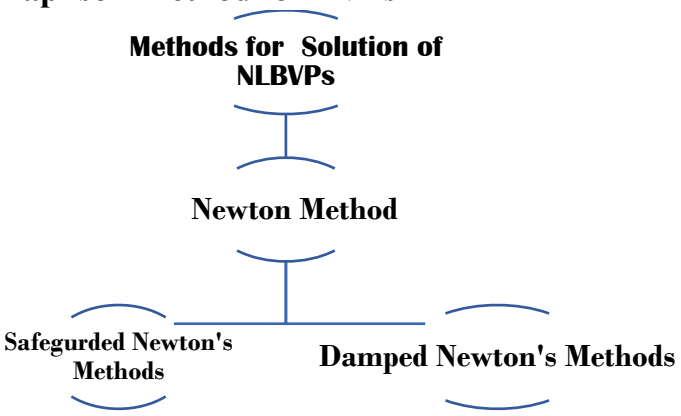

Figure 3.3. : A Schematic Diagram Newton's and modifications Methods

3.1.1. Newton–Raphson Method for Systems of Equations

However, equation (3.11) used to solve nonlinear equations involving only a single variable. This means we have to take equation (3.11) and alter it, in order to use it to solve a set of nonlinear algebraic equations involving multiple variables

Let us now consider the n dimensional version of the same problem, namely

$$F(x) = 0 \qquad (3.1.1.0)$$

We know from Linear Algebra that we can take systems of equations and express those systems in the form of matrices and vectors. With this in mind and using Definition 2, we can express the nonlinear system as a matrix with a corresponding vector. Thus, the following equation is deriving:

$$x^{(k)} = x^{(k-1)} - J\left(x^{(k-1)}\right)^{-1} F\left(x^{(k-1)}\right) \qquad (3.1.1.1)$$

Where $k = 1,2,...,n$ represents the iteration; $x \in R^n, F$ is a vector function and $J(x)^{-1}$ is the inverse of the Jacobian matrix. This equation represents the procedure of Newton's method for solving nonlinear algebraic systems. However, instead of solving the equation $f(x) = 0$, we are now solving the system $F(x) = 0$.

Now we describe the steps of Newton's method:

Step 1: Let $x^{(0)} = \left(x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)}\right)$ be a given initial vector.

Step 2: Calculate $J\left(x^{(0)}\right)$ *and* $F\left(x^{(0)}\right)$.

Step 3: We now have to calculate the vector $y^{(0)}$, where

$$y = [y_1, y_2, ..., y_n]^T$$

In order to find $y^{(0)}$, we solve the linear system $J(x^{(0)})y^{(0)} = -F(x^{(0)}),$.

Remark -2-: Rearranging the system in **Step 3**, we get that $y^{(0)} = -J(x^{(0)})^{-1}F(x^{(0)}).$, we can replace $-J(x^{(0)})^{-1}F(x^{(0)})$ in our iterative formula with y^{0} this result will yield that

$$x^{(k)} = x^{(k-1)} - J\left(x^{(k-1)}\right)^{-1} F\left(x^{(k-1)}\right) = x^{(k-1)} - y^{(k-1)} \qquad (3.1.1.2)$$

Step 4:

Once $y^{(0)}$ is found, we can now proceed to finish the first iteration by solving for $x^{(1)}$.

Thus, using the result from **Step 3**, we have that

$$x^{(1)} = x^{(0)} + y^{(0)} = \left[x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)}\right]^T + \left[y_1^{(0)}, y_2^{(0)}, ..., y_n^{(0)}\right]^T \qquad (3.1.1.3)$$

Step 5: Once we have calculated $x^{(1)}$, we repeat the process again, until $x^{(k)}$ converges to solution $\bar{x}$. The above process is continued until $\|x_k -$ $<$, where ε is the error tolerance.

Remark -3-: When a set of vectors converges, the norm $\left\|x^{(k)} - x^{(k-1)}\right\| \approx 0.$ this means that

$$\left\|x^{(k)} - x^{(k-1)}\right\| = \sqrt{\left(x_1^{(k)} - x_1^{(k-1)}\right)^2 + \ldots + \left(x_n^{(k)} - x_n^{(k-1)}\right)^2} = 0. \qquad (3.1.1.4)$$

3.1.2. Convergence of Newton's Method

Newton's method converges quadratically, if the initial guess solution is near to the exact solution of the system.

- **Advantages and Disadvantages of Newton's Method**

One of the advantages of Newton's method is that it is not too complicated in form and it can be used to solve a variety of problems.

The major ***disadvantage*** associated with Newton's method, are [**61**]:

1. It needs a near initial guess solution $x(0)$ close to the solution x*.
2. Requires $n^2 + n$ function evaluation at each iteration (n^2 for Jacobian matrix and n for $F(x)$).
3. $\mathrm{J}\left(\mathrm{x}^{(\mathrm{k})}\right)$ must be nonsingular for all k and $J(x^*)$ is invertible .
4. Need to compute n^2 partial derivative for $J\left(x^{(k)}\right)$ and $J^{-1}\left(x^{(k)}\right)$ at each step.
5. To solve the linear systems at each iteration require $O(n^3)$ arithmetic operations.

To approximate the solution of a general nonlinear BVP, equation (3.1) with boundary condition eq. (3.2). First, transform it in to NLSAEs by centered difference approximation equation (3.3).

$$\frac{1}{h^2}\left(u_{j-1} - 2u_j + u_{j+1}\right) = f(x,u) \quad for \;\; j = 1,2,\ldots,N \qquad (3.1.1.5)$$

equations (3.1.1.5) can be written us:

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \; for \;\; j = 1,2,\ldots,N \qquad (3.1.1.6)$$

Where $f = f(x,u_j)$ .Generally, NLSAEs (3.1.1.6) cannot not be solve exactly and then $u_1,\ldots,u_N$ are solve iteratively, starting at $u_0 = \left[u_1^{(0)},\ldots,u_N^{(0)}\right]^T$ and the Jacobi Matrix of F

$$FP(u_j) = J_F\left(u_j\right) = \left[\frac{\partial F_i\left(u_i\right)}{\partial u_j}\right] \qquad (3.1.1.7)$$

3.1.3. Application of the Newton's Method

To approximate the solution of mechanics problems defined in section (1.2), we wrote a general algorithm (3.1) for Newton method.

- **Algorithm(3.1): Solves Nonlinear BVPs by using Newton's Method**

INPUT: - L – length of problem domain (b-a)

u - Initial guess of problem solution

λ - Parameter values of lambda

n - Number of spaces

STEP- 1. Calculate magnitude of uniform space between nodes (h)

$$h = \frac{L}{n+1}$$

STEP- 2. *for* m = 1 *to* max_*iteration* do step 3 up to step -6-

STEP- 3. Call functions to compute $F\left(u_i\right)$,and 1st derivative $Fp(u_i)$

$F\left(u_0\right) =$ FC(u_0, h, n, λ) *and* $Fp\left(u_0\right) =$ FJJ(u_0, h, n, λ);

STEP- 4. Compute the Newton step du and update solution

$$\left(Fp\right)^m \left(du\right)^m = F^m \quad \text{for} \quad \left(du\right)^m$$

Update solution at each m (iteration):

$$u^{m+1} = u^m - \left(du\right)^m$$

STEP- 5. if $\|F\| \le tolerance$ is true stop else $m = m+1$

OUTPUT: u – solution Vector

Based on Algorithm (3.1) we wrote a MATLAB program, that solve problems 1 up to 4.

3.1.3.1. Numerical and Graphical output by using Newton's method

Newton's method are applied and tested on following BVPs that is defined in section (1.2) and the numerical result and norm error are displayed by using MATLAB program

Problem-1- consider a Bratu 1-D nonlinear problem defined in section (1.2)

$$u''\left(x\right) + \lambda e^{u(x)} = 0 \qquad (3.1.3.0)$$

Subject to the Dirichlet boundary conditions:

$$u(0) = u(1) = 0 \qquad (3.1.3.1)$$

The Analytic solution for this problem given by, equation (1.2.3)

Using equations (3.1.3.0) and (3.3), we can translate equation (3.1.3.0) into NLSAEs:

$$\frac{1}{h^2}\left(u_{j-1} - 2u_j + u_{j+1}\right) = -\lambda e^{u(j)} \quad for \;\; j = 1,2,...,N \qquad (3.1.3.2)$$

This can be written in the form

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1} \qquad (3.1.3.3)$$

Where $f = -\lambda e^{u(j)}$

To solve eq. (3.1.3.3) by using Newton's method and inputs: initial guess solution (u=ones(n,1)*0.5), error tolerance for F (tol= 10^{-}), number of iterations=50 and value of parameter $\lambda = \{1,1.5,2\}$ by a MATLAB Program for Bratu problem gives the following results:

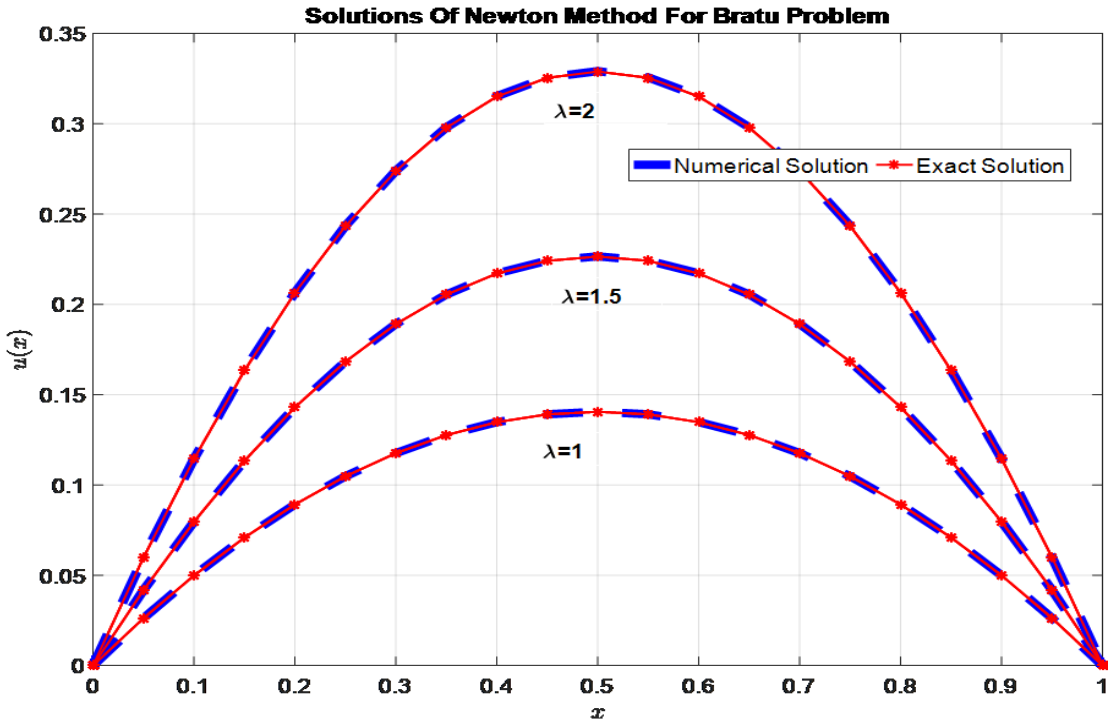

Figure 3.4 : Comparison between the approximated solution and exact solution of Bratu problem using Newton method.

Table 3.1: Newton's method Euclidian norm of error, number of iteration and CPU time for each value of parameter of Bratu problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
4	1	1.87585e -7	4.201e-9	0.78125
3	1.5	1.50368e- 7	3.106e-9	0.53125
3	2	1.242883e-6	2.3052e-8	0.671875

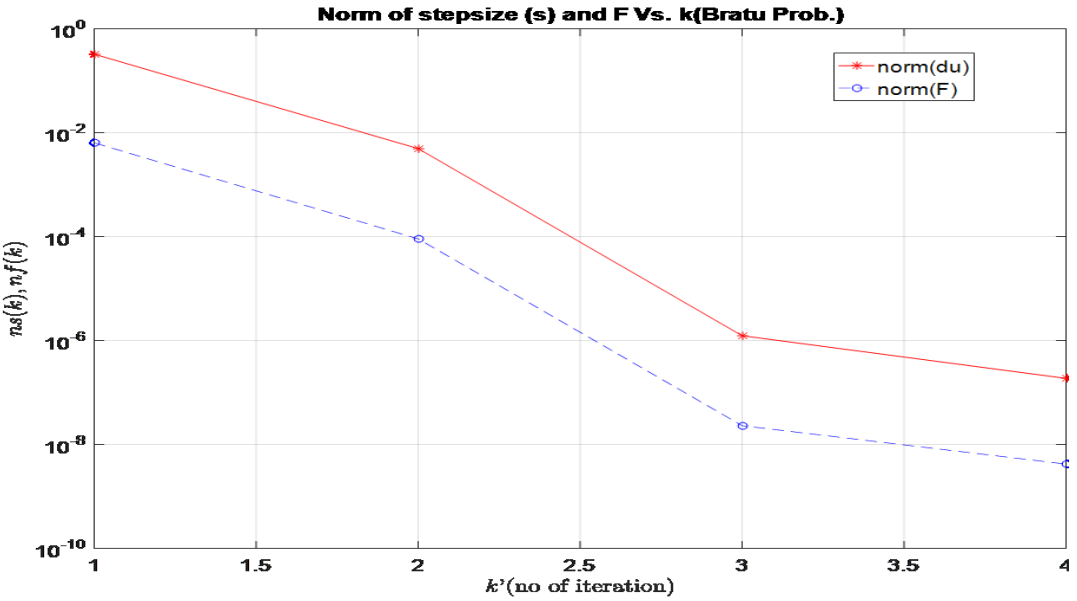

Figure 3.5 Newton's Method Euclidian Norm of Errors versus number of iterations for Bratu problem.

Figure 3.4 cites a comparison between the approximate solution obtained by Newton method and analytic solution for three different values of parameters 1, 1.5 and 2. From this comparison, one can see a good agreement between the analytic solution and the Newton method (approximate) solutions. We can also see the influence of parameters lambda on the solution $u(x)$.

As shown in Table 3.1 Newton method converges for Bratu problem for 3 different values of parameters 1,1,5 and 2 within 4,3 and 3 iterations , which also displays their corresponding CPU time and norm error of Newton step and norm error of function are displaying. Moreover as shown in Fig. 3.5 which proves the convergence of Newton method for Bratu problem and shows the norm errors of a function (F) and Newton step (du) are continuously decreasing and reach an error that is below predefined error tolerance.

Problem-2- consider Troesch's 1-D nonlinear problem defined in section (1.2)

$$u'' = \lambda \sinh(\lambda u) \qquad (3.1.3.3)$$

subject to the Dirichlet boundary conditions:

$$u(0) = 0 \ \ and \ \ u(1) = 1. \qquad (3.1.3.4)$$

Whose general solution given by, equation (1.2.7)

Using equations (3.1.3.3) and (3.3), we can translate Troesch's problem in to NLSAEs:

$$F(u_j) = -fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1,2,...,N \qquad (3.1.3.5)$$

Where $f = \lambda \sinh(\lambda u(j))$

Based on Algorithm (3.1), equation (3.1.3.5) is solve by using Newton's method and inputs: initial guess solution (u=ones(n,1)), error tolerance for function(tol= 10^{-}), number of iterations=50 and value of parameter $\lambda = \{0.5,1,1.5\}$ using a MATLAB Program for Troesch's problem gives the following results:

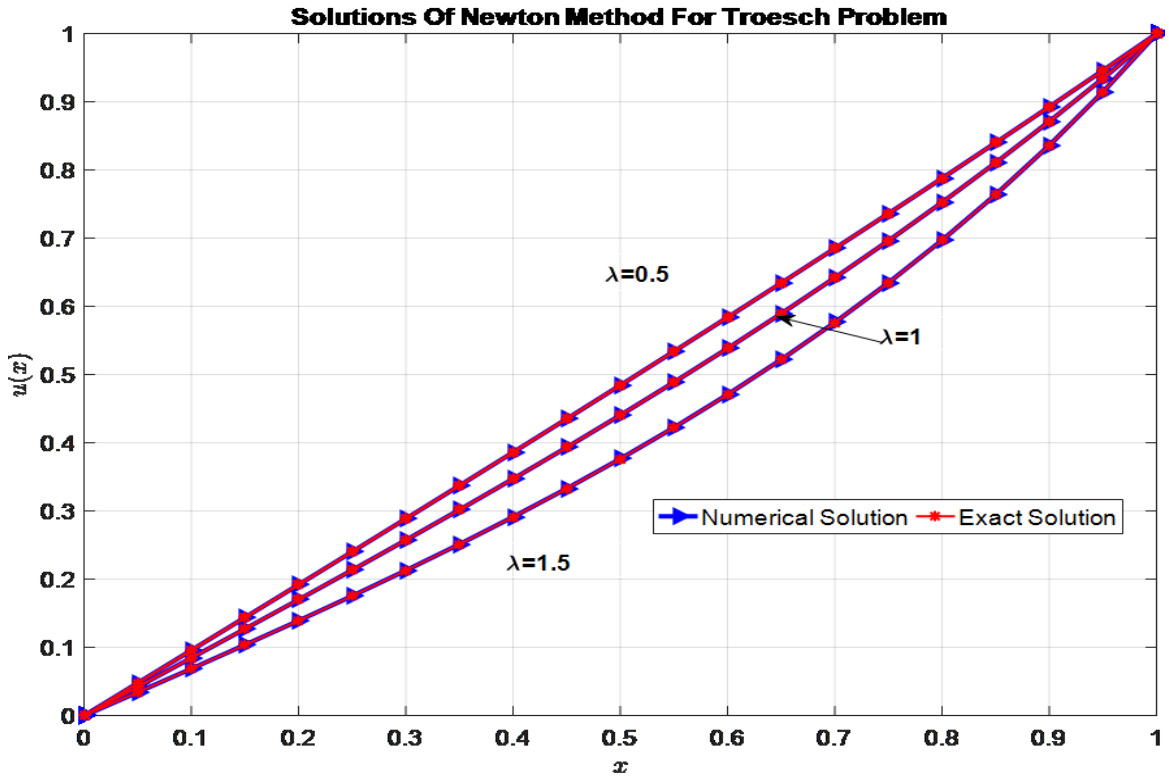

Figure 3.6 Comparison between the approximated solution and exact solution of Troesch problem using Newton's method .

Table 3.2 : Newton's method Euclidian norm of error, number of iteration and CPU time for each value of parameter of Troesch's problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of fuction (F)	CPU time
3	0.5	3.45e -9	8.9e -11	0.0625
3	1	1.23e-10	4e -12	2.078125
3	1.5	4.4489e-8	1.64e -9	2.25

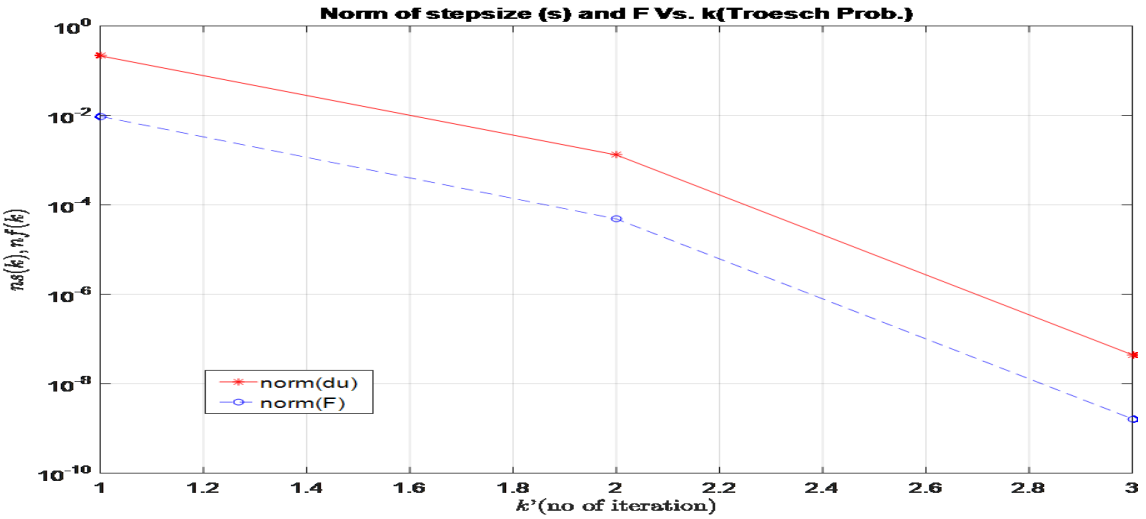

Figure 3.7 Newton's Method Euclidian Norm of Errors versus number of iteration for Troesch problem.

As shown in Figure 3.6 a comparison between the approximate solution obtained by Newton method and analytic solution. From this comparison, one can see a good agreement between the analytic solution and the Newton method (approximate) solution for Troesch's problem and the influence of parameter lambda on the solution $u(x)$.

As shown in Table 3 .2 Newton methods converge for Troesch problem for 3 different values of parameters, which also takes different CPU time for different value of parameters. Moreover as shown in Figure 3.7, which proves the convergence of newton method for Troesch's problem and displays the norm errors of a function and Newton step (du) are continuously decreasing and reach an error, which is below predefined error tolerance for all the three different parameters.

Problem -4 - consider Thermal Explosion nonlinear BVPs defined in section (1.2)

$$u''+\lambda e^{u}=0, \quad x\in(0,1), \qquad (3.1.3.6)$$

With the associated Neumann boundary conditions

$$u'(0) = 0,\ u(1) = 0.$$

The analytic solution given by, equation (1.2.12)

Using equations (3.1.3.6) and (3.3), we can translate eq. (3.1.3.6) in to NLSAEs

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1},\ for\ \ j = 1,2,...,N \qquad (3.1.3.7)$$

Where $f = -\lambda e^{u_j}$

Based on algorithm (3.1), equation (3.1.3.7) is solved, by using Newton's methods and inputs: initial guess solution (u=0.5*ones(n,1)), error tolerance for function(tol= 10^{-}), number of iterations=50 and value of parameter , $\lambda = \{0.1, 0.2\}$ using a MATLAB Program for this problem gives the following results:

Table 3.3:Newton's method Euclidian norm of error, number of iteration and CPU time for each value of parameter of Thermal Explosion problem.

Number of iteration(k)	Parameter Lambda	Euclidian norm of error(du)	Euclidian norm of function (F)	CPU time
14	0.1	2.2877e -4	9.137e -5	0. 09375
10	0.2	2.52e -4	9.98e -5	1.296875

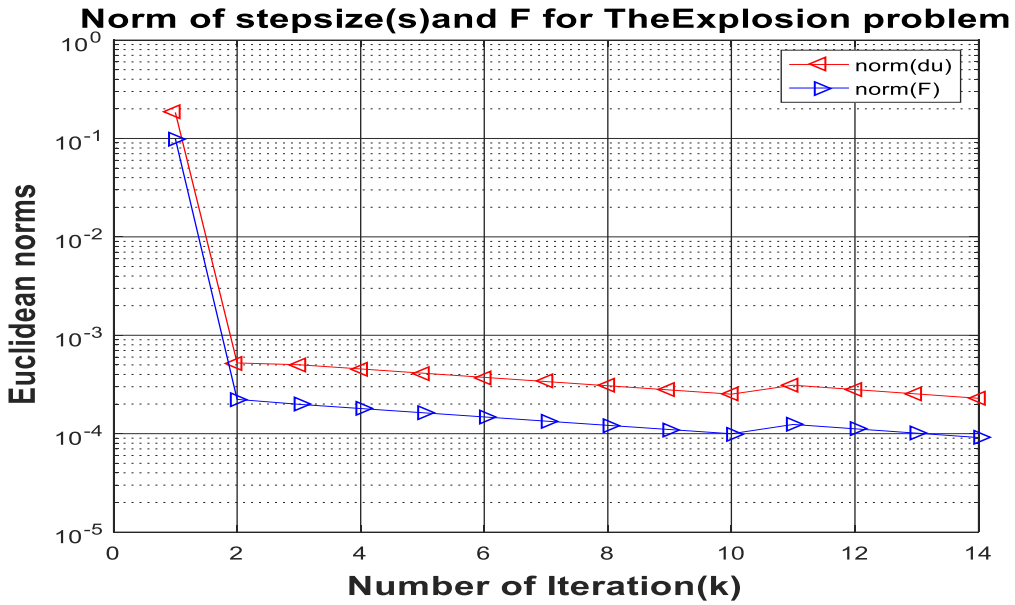

Figure 3.8 Newton's Method Euclidian Norm of Errors versus number of iteration for Thermal Explosion problem.

As shown in Table 3.3 Newton's method converges for Thermal Explosion problem and terminate within 14 iteration for $\lambda = 0.1$ and within 10 iteration for λ=0.2. Moreover as shown in Figure 3.8, which proves the convergence of Newton's method for Thermal Explosion problem and displays the norm errors of a function and Newton step (du) are continuously decreasing. However, up to second iteration, its convergence speed is fast but finally; its convergence speed is decreases.

3.2. Safeguarded Newton's Method for BVP

Several difficulties commonly arise in the application of Newton and quasi-Newton methods in solving multivariate non-linear equations.

One of the main causes of failure of Newton-type methods is failure in the specification of a starting point that is not sufficiently close to a root. These problems can be diminishing by appropriate action called Safeguarded method.

The norm of a function is zero at a root, one may view an iterate as yielding an improvement if it reduces this norm. However, if an iterate increases the norm of the function value, then one can cut the step length prescribed by the newton method in half, and continue cutting it in half, until the revised iterate yields an improvement.

Generally $\|f(x)\|_2$ is zero at a root and positive elsewhere an improvement over the previous iterate if it reduces the function norm,

$$\|f(x)\| > \|f(x+d)\| \qquad (3.2.0)$$

Where d is the Newton step. Back stepping of Safeguarded Newton prevents, Newton and quasi-Newton methods from taking a large step in the wrong direction, substantially improving their robustness.

$$\|f(x)\| > \|f(x+d)\| \; or \|f(x+d/2)\| > \|f(x+d)\| \qquad (3.2.1)$$

3.2.1. Application of the Safeguarded Newton's (SNewton's)

To approximate the solution of mechanics problems defined in section (1.2) and general NLSAEs (3.1.1.6), we wrote a general algorithm (3.2) for SNewton method.

➢ **Algorithm (3.2): Solves nonlinear BVPs by using SNewton's Method.**

INPUT: - L – length of problem domain (b-a)

u - Initial guess of problem solution

λ - Parameter values of lambda

$n, MaxIt$ - Number of spaces, Maximum iteration

STEP- 1. Compute F,F' for (u).

$F = FC(u,h,n,\lambda), \quad Fp = FC(u,h,n,\lambda)$

do step-2 up to 6 until K<MaxIt else Stop

Compute $norol = \|F\|_2$

STEP- 2. Compute Du,

$Du = Fp \setminus F$

when k > MaxIt and norol> tol satisfy stop else do step 3

STEP- 3. Update solution (u), $u_{k+1} = u_k - Du$ and

$F(u_{k+1}) =$ FC(u_{k+1},h,n,λ) *and* $F'(u_{k+1}) =$ FJJ(u_{k+1},h,n,λ);

Compute $norf = \|F_{k+1}\|$

STEP- 4. If $norf < norol$ is satisfied stop else go to step 5

STEP- 5. Compute

$Du = Du / 2$

OUTPUT: u – solution Vector

Based on Algorithm (3.2), we wrote a MATLAB program, that solve problems 1and 2.

Numerical and Graphical output of problems using SNewton method

Safeguarded Newton's method are applied and tested on following BVPs that is define in section (1.2) and the numerical result and norm error are displayed by using MATLAB program.

Problem-1- consider Bratu 1-D nonlinear problem defined in section (1.2)

$$u''(x) + \lambda e^{u(x)} = 0 \qquad (3.2.2)$$

Subject to the Dirichlet boundary conditions:

$$u(0) = u(1) = 0 \qquad (3.2.3)$$

The analytic solution for this problem given by, equation (1.2.3).

Using equations (3.2.2) and (3.3), we can create NLSAEs for Bratu problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1,2,...,N \qquad (3.2.4)$$

Where $f = -\lambda e^{u_j}$

To solve equation (3.2.4), by using Safeguarded Newton's method and take similar inputs of Newton method for Bratu problem gives the following results:

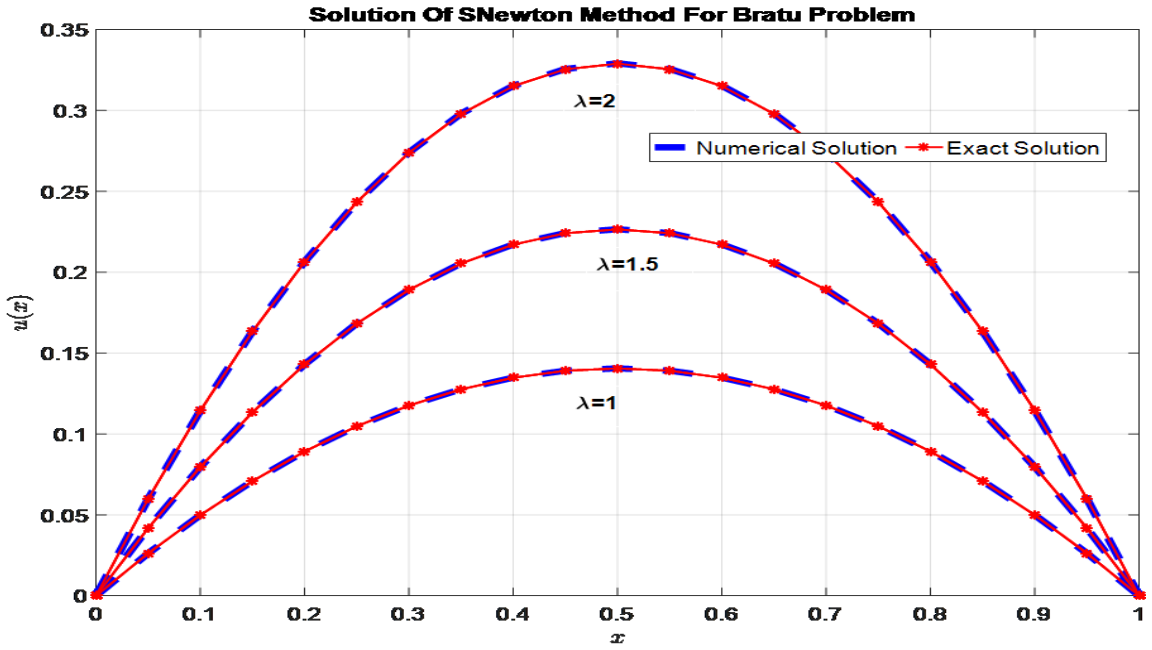

Figure 3.9 Comparison between the approximated solution and exact solution of Bratu problem.

Table 3.4: Safeguarded Newton method Euclidian norm of error, number of iteration and CPU time for each value of parameter of Bratu problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
18	1	7.997e-7	6.69e-8	0.18750
10	1.5	6.79e-7	7.39e-7	1.1875
10	2	7.97e-7	7.915e-7	2.05625

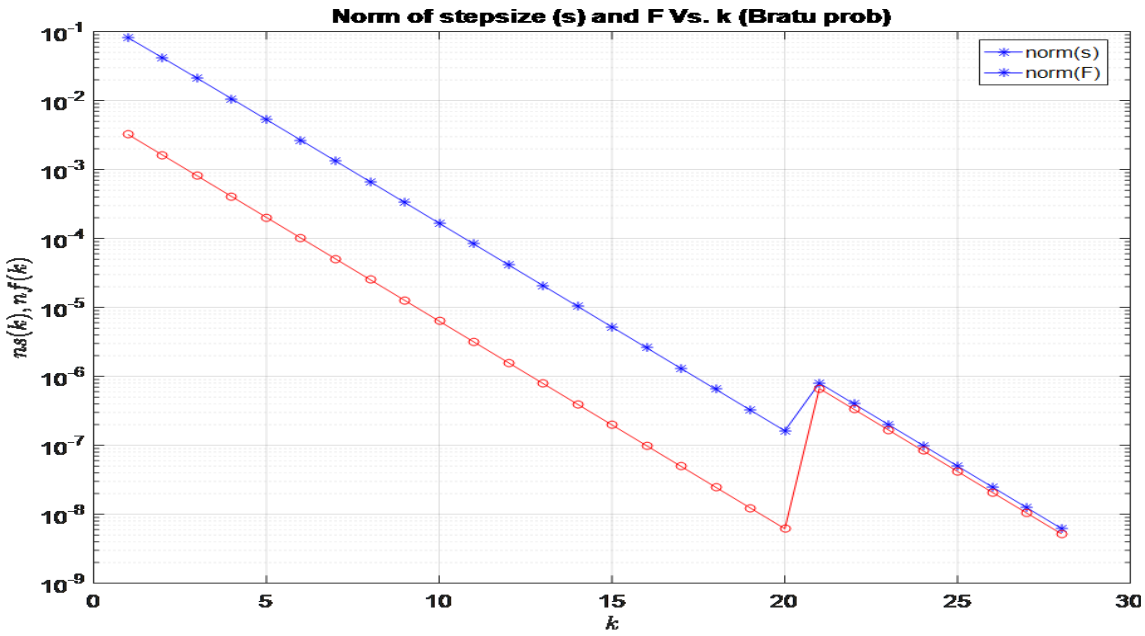

Figure 3.10 Safeguarded Newton Method Euclidian Norm of Errors Vs. number of iterations for Bratu's problem

Figure 3.9 cites a comparison between the approximate solution and exact solution for three different values of parameter (λ), we can see a good agreement between the exact solution and the SNewton method (approximate) solution.

As shown in Table 3.4, SNewton method converges for Bratu problem and terminates within 18, 10, and 10 iterations for three different values of parameters (lambda). The CPU-time taken for each parameter also indicates SNewton method takes more time for computation than Newton method, however its norm error step(du) are less than pure Newton. Moreover as shown in Figure 3.10, which proves the convergence of SNewton method for Bratu problem and Euclidean norm error for Newton step (du) and function are continuously decreasing. This Shows Back stepping used to reduce a large step (du) in the wrong direction by comparing two consecutive norms of a function.

Problem-2- consider Troesch's 1-D nonlinear problem defined in section (1.2)

$$u'' = \lambda \sinh(\lambda u) \qquad (3.2.5)$$

Subject to the Dirichlet boundary conditions:

$$u(0) = 0 \ \ and \ \ u(1) = 1. \qquad (3.2.6)$$

Whose general solution for the problem given by, equation (1.2.7).

Using equation (3.2.5) and (3.3), we can create NLSAEs for Troesch's problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1,2,...,N \qquad (3.2.7)$$

Where $f = \lambda \sinh(\lambda u(j))$

To solve equation (3.2.7) by using Safeguarded Newton's method and takes similar inputs of Newton method for Troesch's problem gives the following results

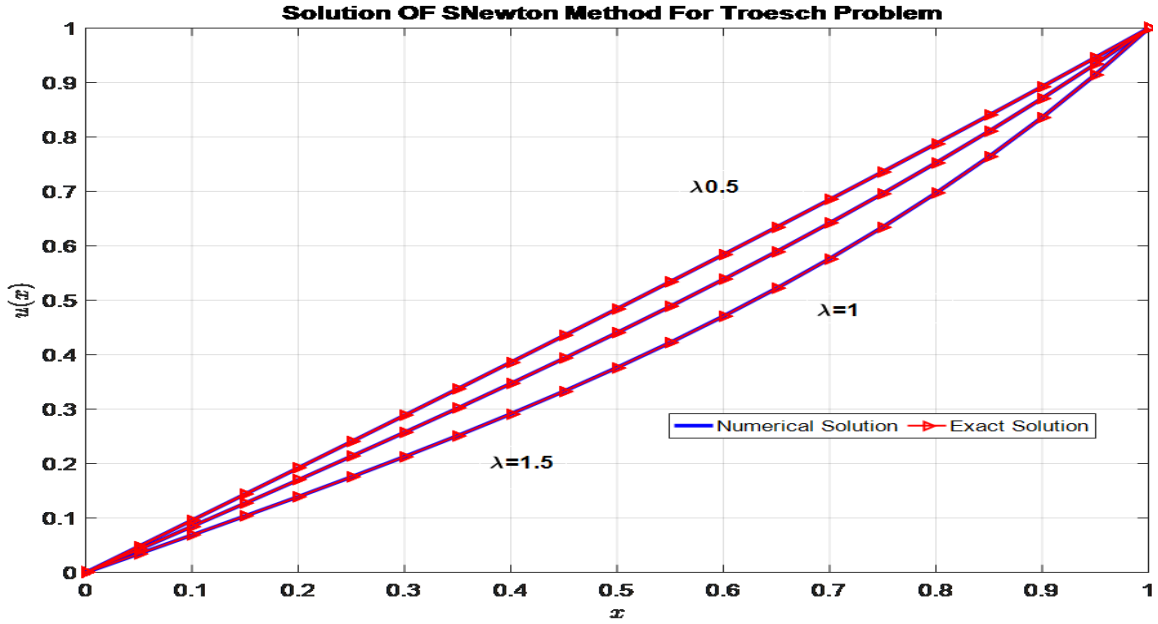

Figure 3.11 Comparison between the approximated solution and exact solution of SNewton method for Troesch's problem.

Table 3.5: Safeguarded Newton method Euclidian norm of error, number of iteration and CPU time for each value parameter of Troesch's problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
20	0.5	9.41e-9	7.4558e-9	0.03125
14	1	1.011e-8	7.478e-9	1.109375
14	1.5	1,209e-8	7.547e-9	1.84375

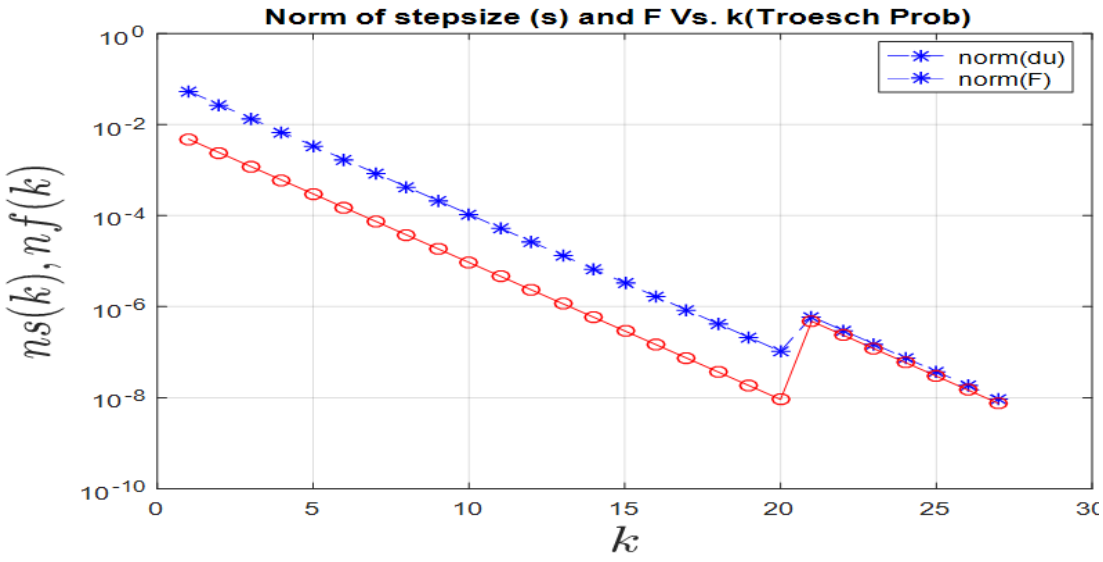

Figure 3.12 Safeguarded Newton Method Euclidian Norm of Errors versus number of iterations for Troesch's problem.

As shown in Figure 3.11 a comparison between the approximate solution obtained by Safeguarded Newton method and exact solution for Troesch's problem. From this comparison, we can see a good agreement between the exact solution and the Safeguarded Newton method (approximate) solution for Troesch's problem for three different values of parameter.

As shown in Table 3.5 Safeguarded Newton method converges for Troesch problem and terminate within 20, 14, and 14 iterations for three different values of parameters (lambda). The CPU-time taken for each parameter also indicates Safeguarded Newton method takes more time for computation; however, its errors of Newton's steps are less than pure Newton method. Moreover as shown in Figure 3.12 for Troesch problem the error getting smaller in between successive iterates implies that getting closer to the answers. Euclidian norm of error of both Newton step (du) and function are continuously decreasing by back stepping that is used to reduce a large step (du) in the wrong direction by comparing two consecutive norms of a function.

3.3. Damped Newton's Method for BVPs

Locally convergent algorithms like Newton Raphson require initial iterate "sufficiently" close to "unknown" zero of the function under consideration. When this requirement is not meet, the algorithm can fail to converge to a "zero" (solution).

Damped Newton Method enables pure (not damped) Newton Method ***Globally convergent*** by algorithms that allow arbitrary starting point. We embark in a study of such methods even for non-differentiable equations but whose Jacobi is estimated numerically.

An improvement of local convergence of Newton Method, following iteration can be used

$$x^{m+1} = x^m + \lambda z^m \qquad (3.3.0)$$

Here, damping factor is $0 \le \lambda \le 1$ and z solves the linear system

$$J(x^m)z = -f\left(x^m\right) \qquad (3.3.1)$$

The computation of jacobian matrix J numerically by using Armijo step size rule as

$$J(x^m) = \frac{f(x^{m+1}) - f(x^m)}{h} \qquad (3.3.2)$$

Where, J is an $n \times n$ matrix and equations (3.3.1) and (3.3.2) can be written as

$$z = -\left(J(x^m)\right)^{-1} f(x^m)$$

Or
$$z = -\left(\frac{f(x^{m+1}) - f(x^m)}{h}\right)^{-1} f(x^m) \qquad (3.3.3)$$

The values of λ changes at each iteration. In order to find its value, the following condition checked:

$$\max|z| < \max|\lambda z| \quad and \quad \lambda < \lambda(\min) \qquad (3.3.4)$$

Initially, a test value of λ is accepted if $\max|z_i| > \max|\lambda z_{i+1}|$, where $i = 0,1,\ldots$ Then λ is accepted, otherwise $\lambda_i = \lambda_i / 2$ is taken and check the second condition:

$$\lambda_i < \lambda(\min) \qquad (3.3.5)$$

Where $\lambda(\min)$, minimum value of λ which is given, if (3.3.5) satisfied, we are using too many damping then stop, otherwise we are computing the new iterate using equation (3.3.0) and compute λ_{i+1}:

$$\lambda_{i+1} = \min\left(1., 2\lambda\right) \qquad (3.3.6)$$

iteration are repeated until λ is accepted or no convergence is detected. Using Stopping criteria is $\|F\|_2 > tolerance$ satisfied or until iteration limit is reached.

3.3.1. Application of the Damped Newton's Method

General Algorithm of Damped Newton method, to approximate the solution of our mechanics problems defined in section (1.2).

- **Algorithm (3.3): Solves nonlinear BVPs by using DNewton's Method.**

INPUT: - L – length of problem domain (b-a)

u - Initial guess of problem solution

λ - Parameter values of lambda

n, MaxIt, α - Number of spaces,MaxIt, relaxation factor

STEP- 1. Compute F, for (u).

$$F = FC(u, h, n, \lambda),$$

STEP-2. Compute numerical Jacobi step,'hh' and solution u for iteration 1 up to N.

$$hh = \sqrt{eps} * u_i \ , u = u + hh$$

Compute F and Numerical Jacobi J

$$F = FC(u, h, n, lam), \quad J = (F_{new} - F_{old}) / hh$$

STEP-3. Compute Newton step size s_{old} s

$$s_{old} = -J \setminus F$$

Update solution(u) & compute error

$$u = u + \alpha s, \ error = \|F\|_2$$

STEP-4 . When error > Convcrit & K < maxiter

$k = k + 1$ & do STEP-3

Compute s_{temp} by

$$s_{temp} = -J \setminus F$$

STEP-5. If $\max|s_{temp}| < \max|s|$ satisfied compute $\alpha = \alpha / 2$

Compute and update Newton step by

$$u_{new} = u_{old} + \alpha s_{temp}$$

increase relaxation factor $\alpha = \min(1., \alpha * 2)$

OUTPUT: display Numerical and Graphical Output

➢ **Numerical and Graphical output of problems using Damped Newton method**

The numerical method, Newton Damp is applied and tested on the following nonlinear BVPs that are define in section (1.2) and the numerical result and norm error are displayed by using MATLAB program.

Problem-1- Consider Bratu 1-D nonlinear problem defined in section (1.2.)

$$u''(x)+\lambda e^{u(x)}=0 \qquad (3.3.1.0)$$

Subject to the Dirichlet boundary conditions:

$$u(0)=u(1)=0 \qquad (3.3.1.1)$$

The analytic solution for this problem given by, equation (1.2.3).

Using equations (3.3.1.0) and (3.3), we can create NLSAEs for Bratu problem

$$F(u_j)=fh^2+u_{j-1}-2u_j+u_{j+1},\ for\ \ j=1,2,...,N \qquad (3.3.1.2)$$

Where $f=-\lambda e^{u_j}$

Equation (3.3.1.2) is solving by using Damped Newton method and using similar inputs of previous Newton's method for the same problem provides the following outputs:

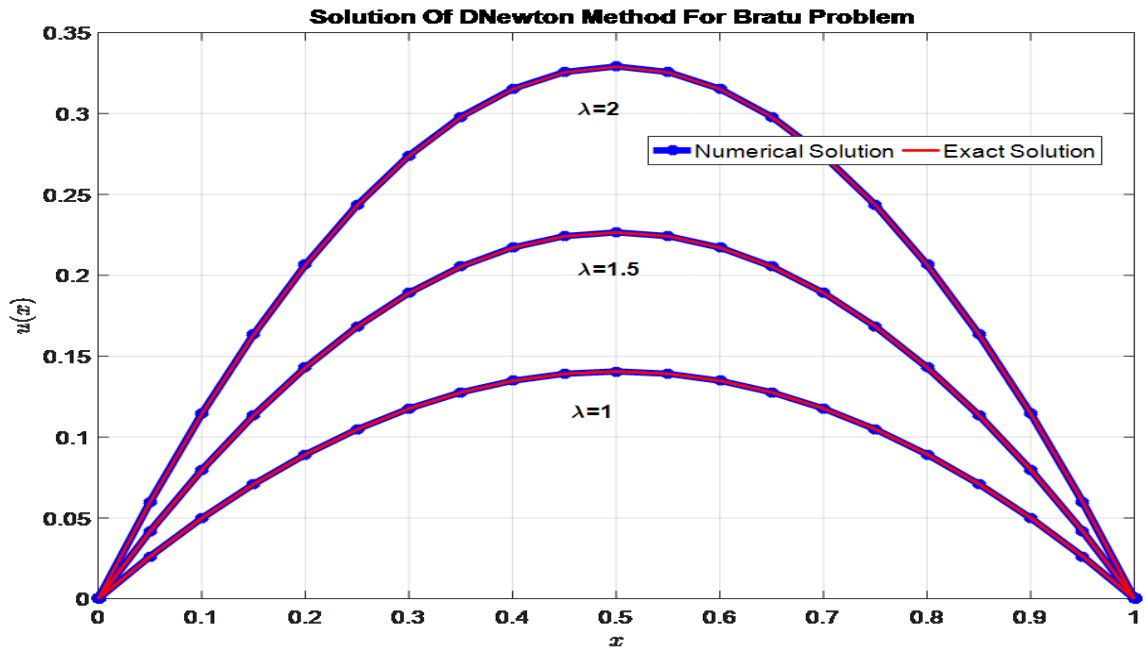

Figure 3.13 Comparison between the approximated solution and exact solution of Bratu problem.

Table 3. 6: Damped Newton method Euclidian norm of error, number of iteration and CPU time for each value of parameter of Bratu problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of fuction (F)	CPU time
3	1	3.119e -3	3.955e - 9	0.53125
4	1.5	5.5877e -5	1.3897e -9	1.890625
4	2	8.465e- 5	1.9914e -9	2.59375

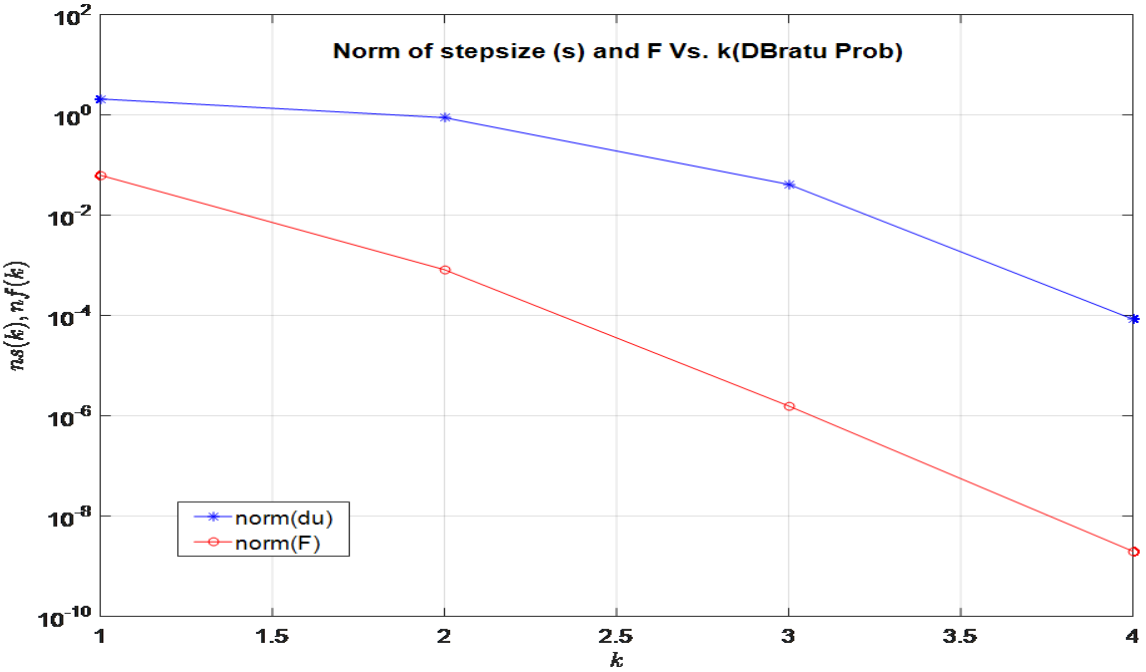

Figure 3.14 Damped Newton Method Euclidian Norm of Errors versus number of iterations for Bratu problem.

Figure 3.13 cites a comparison between the approximate solutions obtained by Damped Newton method and exact solution. From this comparison, one can see a good agreement between the exact solution and the Damped Newton method (approximate) solution.

As shown in Table 3.6, Damped Newton method converges for Bratu problem and terminates within 3, 4, and 4 iterations for three different values of parameters (lambda). The CPU-time taken for each parameter also indicates Damped Newton method takes almost nearest computation time with pure Newton except the first parameter result and the number of iteration takes for each parameter is almost the same as Newton method. Moreover as shown in Figure 3.14, for Bratu problem the error getting smaller in between successive iterates implies that getting closer to the answers. We observe norm error of Step(du) are more than both Newton and SNewton methods because it decreases

relaxation factor only, however, it converges for big initial guess solution and it takes a maximum of 4 iteration to converge .

Problem-2- consider Troesch's 1-D nonlinear problem defined in section (1.2)

$$u'' = \lambda \sinh(\lambda u) \qquad (3.3.1.3)$$

Subject to the Dirichlet boundary conditions:

$$u(0) = 0 \ \ and \ \ u(1) = 1. \qquad (3.3.1.4)$$

The theoretical solution for this problem given by, equation (1.2.7).

Using equation (3.3.1.3) and (3.3), we can create NLSAEs for Troesch's problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1,2,...,N \qquad (3.3.1.5)$$

Where $f = \lambda \sinh(\lambda u(j))$

Equation (3.3.1.5) is solve by using Damped Newton method and by taking Newton method inputs of similar problem gives the following results:

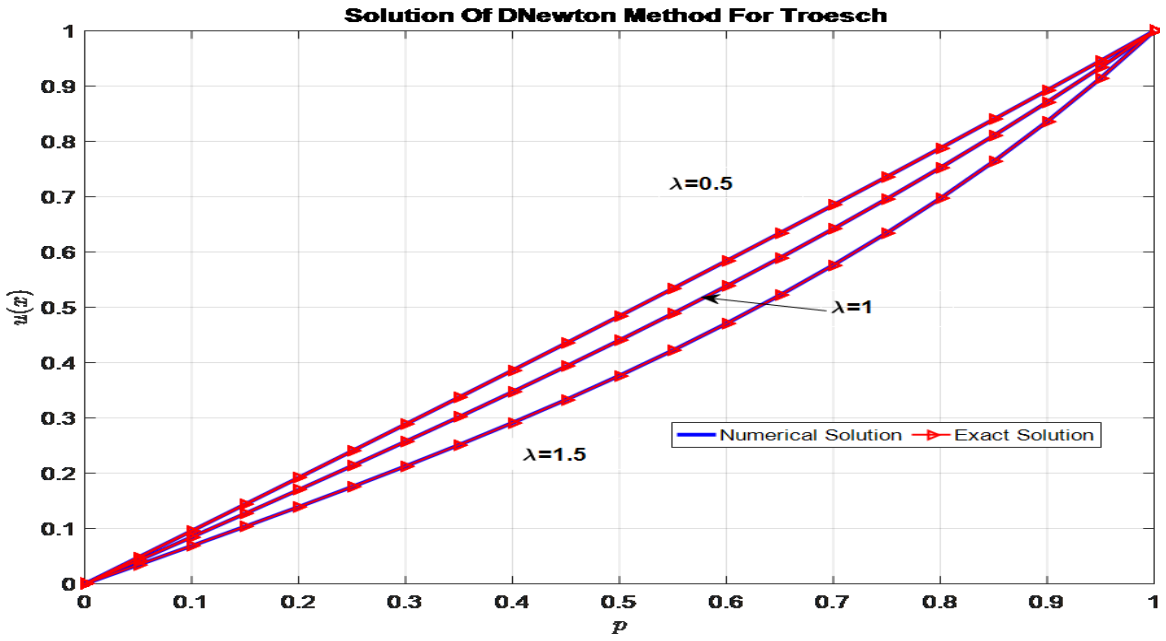

Figure 3.15 Comparison between the approximated solution and exact solution of Troesch problem.

Table 3. 7: Damped Newton method Euclidian norm of error, number of iteration and CPU time for each value of parameter of Troesch's problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of fuction (F)	CPU time
32	0.5	4.388e -5	3.1534e -8	0. 140625
2	1	1.302e -3	3.11895e -9	1.59375
3	1.5	2.121e -6	3.0624e -8	2.250

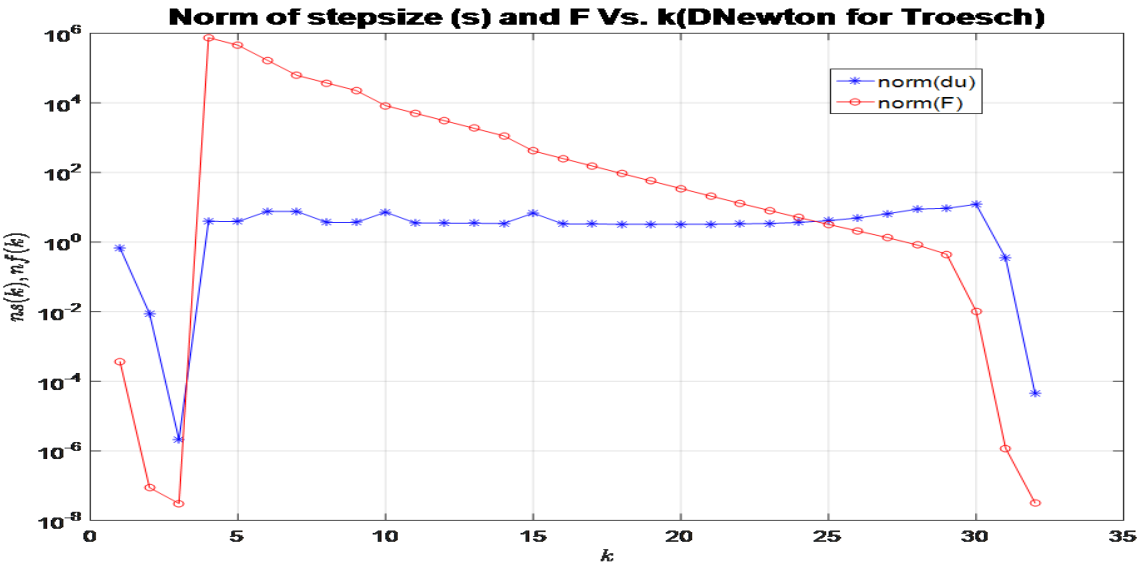

Figure 3. 16 Damped Newton Method Euclidian Norm of Errors versus number of iterations for Troesch's problem.

As shown in Fig. 3.15 a comparison between the approximate solution obtained by DNewthon method and exact solution. From this comparison, one can see a good agreement between the exact solution and the DNewton method (approximate) solution.

As shown in Table 3.7 Damped Newton method converges for Troesch's problem and terminate within 32, 2, and 3 iterations for 3 values of parameters 0.5,1 and 1.5 respectively. From this, we can see that the first parameter needs 32 iterations, which is big as compared to other parameter values due to numerical approximation of Jacobean matrix that leads solution to wrong direction, in order to correct the solution path it takes more iterations. Moreover, as shown in Figure 3.16 the error is not stable, around fifth iteration Euclidian norm of errors increase rather decrease.

However, the method converges even the initial guess solution, is far from true solution. That is a better advantage as compared to local pure Newton method.

CHAPTER FOUR

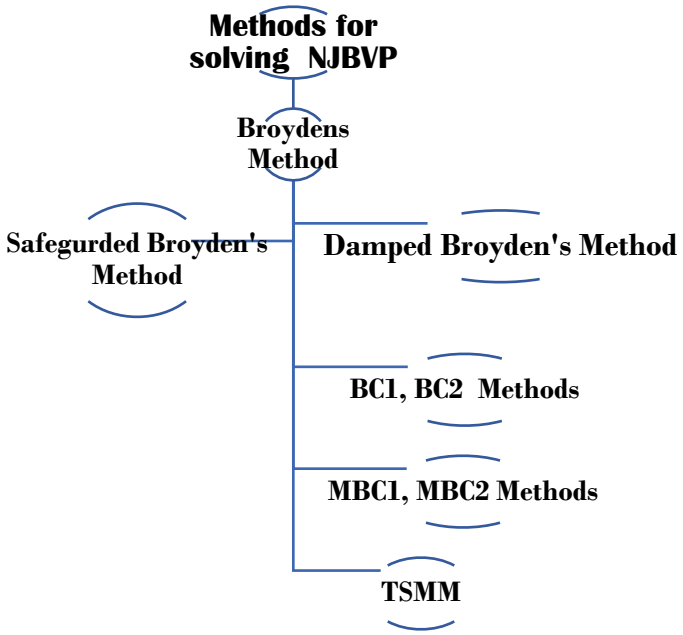

Figure 17 : A Schematic Diagram Broydens and its modifications Methods

4. Broyden's Methods

In the previous section (3.1), we examined the numerical method known as Newton's method. We established that one of the major disadvantages of this method was that Jacobian matrix, $J(x)$ and its inverse must be compute at each iteration. We therefore want to avoid this problem by the methods known as Quasi-Newton methods that use an approximation matrix that is updating at each iteration in place of the Jacobian matrix. This implies that the form of the iterative procedure for Broyden's method is almost identical to that used in Newton's method. The only exception being that an approximation matrix A_i is implementing instead of $J(x)$.

The Secant Method

When finding the derivative $f'(x)$ in Newton's method is problematic, or when function evaluations take too long; we may adjust the method slightly. Instead of using tangent lines to the graph, we may use secants .The approach be referred as the secant method.

The idea of the secant method is to think as in Newton's method, but instead of using $f'(x)$, we approximate this derivative by a finite difference or the secant, i.e., the slope

of the straight line that goes through the points $\left(x_n, f\left(x_n\right)\right)$ and $\left(x_{n-1}, f\left(x_{n-1}\right)\right)$ on the graph, given by the two most recent approximations x_n and x_{n-1}. This slope reads

$$\frac{f\left(x_n\right)-f\left(x_{n-1}\right)}{x_n-x_{n-1}} \qquad (4.0)$$

Inserting equation (4.0), in place of expression $f'\left(x_n\right)$ in Newton's method equation (3.11) simply gives us the secant method:

$$x_{n+1}=x_n-f(x_n)\frac{x_n-x_{n-1}}{f\left(x_n\right)-f\left(x_{n-1}\right)} \qquad (4.1)$$

From (4.1), we see how two chose starting points (x_n, x_{n-1}, and corresponding function values) are used to compute x_{n+1}. Once we have x_{n+1}, we similarly use x_n and x_{n+1} to compute x_{n+2}. As with Newton's method, the procedure is repeated until $\|f(x_n)\|$ is below some chosen limit value or some limit on the number of iterations has been reached. We use an iteration counter here too, based on the same thinking as in the implementation of Newton's method.

The Secant Method used, when the derivative is not available or is too expensive to evaluate.

Now, that we have a version of Newton's Method, for systems of nonlinear equations. Although there is no simple extension of Newton's Method to a Secant Method for systems, [**62**]suggested a method that is generally considered the next best thing.

Suppose A_i is the best approximation available at step i to the Jacobian matrix, and that it has been used to create

$$F\left(x\right)=0, \qquad x_{i+1}=x_i-A_i^{-1}F\left(x_i\right). \qquad (4.2)$$

To update A_i to A_{i+1} for the next step, we would like to show the derivative aspect of the Jacobian F', and satisfy

$$A_{i+1}\,s_{i+1}=y_{i+1}, \qquad (4.3)$$

Where $s_{i+1} = x_{i+1} - x_i$ and $y_{i+1} = F(x_{i+1}) - F(x_i)$. On the other hand, for the orthogonal complement of s_{i+1}, we have no new information. Therefore, we ask that

$$A_{i+1} w = A_i w \qquad (4.4)$$

for every w satisfying $s_{i+1}^T w = 0$, a matrix that satisfies both (4.3) and (4.4) is

$$A_{i+1} = A_i + \frac{(y_{i+1} - A_i s_i) s_{i+1}^T}{s_{i+1}^T s_{i+1}}. \qquad (4.5)$$

Moreover, the relation between the inverses of A_i and A_{i+1} is, in this case,

$$A_{i+1}^{-1} = A_i^{-1} + \frac{(s_i - A_i^{-1} y_i) s_i^T A_i^{-1}}{s_i^T A_i^{-1} y_i}. \qquad (4.6)$$

This formula shows that iteration (4.3) can be compute without solving a linear system at each iteration.

Broyden's Method uses the Newton's Method step (3.11) to advance the current guess, while updating the approximate Jacobian by (4.6). Summarizing, the algorithm starts with an initial guess x_0 and an initial approximate Jacobian A_0, which can be choose to be the identity matrix if there is no better choice.

4.1. Broyden's Method I

x_0 =initial vector

A_0 =initial approximate Jacobian matrix ($A_0 = B_0$)

For $i = 0,1,2,...$

$$x_{i+1} = x_i - A_i^{-1} F(x_i)$$

$$(*) B_{i+1}^{-1} = B_i^{-1} + \frac{(s_{i+1} - B_i^{-1} y_{i+1}) s_{i+1}^T B_i^{-1}}{s_{i+1}^T B_i^{-1} y_{i+1}}$$

end

where $s_{i+1} = x_{i+1} - x_i$ and $y_{i+1} = F(x_{i+1}) - F(x_i)$.

Note that the Newton-type step is carried out by solving $A_i s_{i+1} = F(x_i)$. Also like Newton's Method, Broyden's Method is not guarantee to converge.

4.1.1. Application of the Broyden's method I

To approximate the solution of mechanics problems define in section (1.2), we wrote a general algorithm (4.1) for Broyden method I.

- **Algorithm(4.1): Solves nonlinear BVPs by using Broyden I method**

INPUT: L – length of problem domain (b-a)

uI, uF - initial & final boundary value

u - Initial guess of problem solution

λ - Parameter values of lambda

n - Number of spaces

maxiter - Maximum number of iterations

B - Initial approximation of Jacobian matrix ($B =$

STEP 1: Do steps-2 up to 6 until iteration(k) <maxiter and err < tolerance

STEP 2: call function F for u_0.

$$F = \text{FC}(u_0, h, n, \lambda);$$

Compute du(Newton step) and update u

$$du = B * F, \quad u_{new} = u_{old} - du$$

STEP 3: call function F for u_{new}

$$F = \text{FC}(u_{new}, h, n, \lambda);$$

Compute dL and DL.

$$dL = u_{new} - u_{old}, \; DL = F_{new} - F_{old}$$

STEP 4: Update Jacobian matrix B by (*)

STEP 5: Compute norm error of du and F

STEP 6: Display Numerical and graphical output

- **Numerical and Graphical output of problems using Broyden method I**

The numerical method, Broyden method I applying and testing on the following nonlinear BVPs that are define in section (1.2) and the numerical result and norm error are display by using MATLAB program.

Problem-1- consider Bratu 1-D nonlinear problem defined in section (1.2)

$$u''(x) + \lambda e^{u(x)} = 0 \qquad (4.1.1.0)$$

Subject to the Dirichlet boundary conditions:

$$u(0) = u(1) = 0 \qquad (4.1.1.1)$$

The Analytic solution for this problem given by, equation (1.2.3).

Using equations (4.1.1.0) and (3.3), we can create NLSAEs for Bratu problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1},\ for\ \ j = 1,2,...,N \qquad (4.1.1.2)$$

Where $f = -\lambda e^{u_j}$

Equation (4.1.1.2) is solved by using Broyden method I and inputs: initial guess solution (u=ones(n,1)), error tolerance for function(tol= 10^{-}), number of iterations=50 and value of parameter $\lambda =$ using a MATLAB Program for Bratu problem gives the following results:

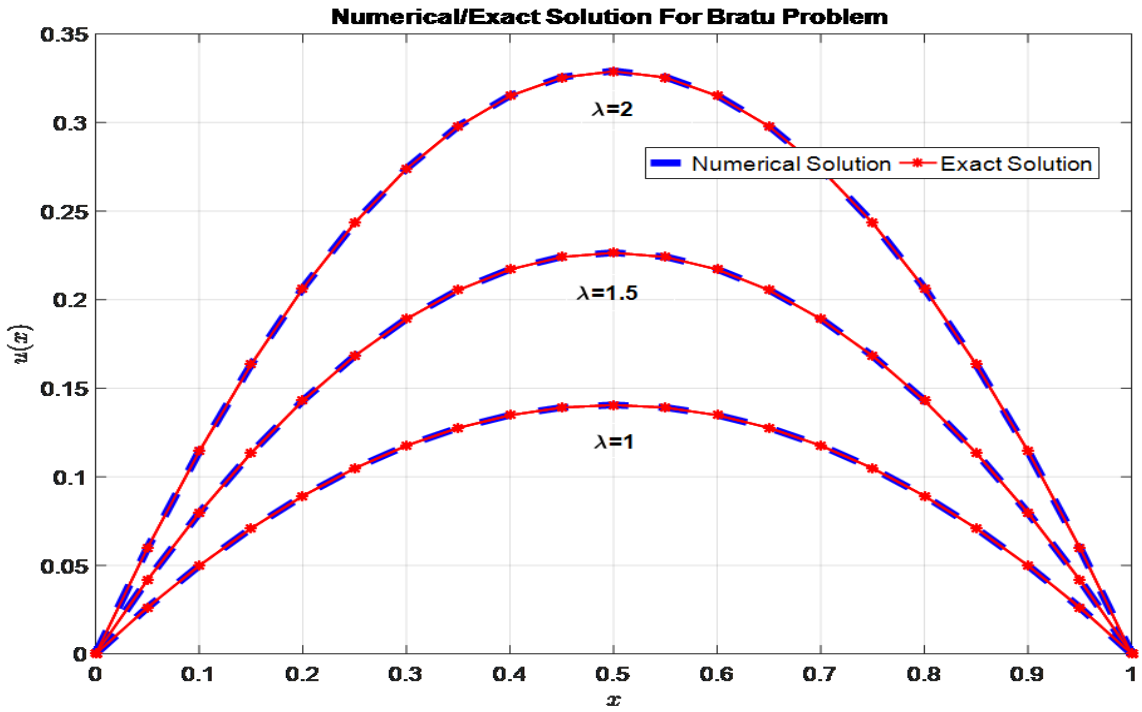

Figure 4.18 Comparison between the approximated solution and exact solution of Bratu problem using Broyden method I for 3 different parameters (lambda).

Table 4. 8: Broyden method I Euclidian norm of error, number of iteration and CPU time for each value of parameter of Bratu problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
22	1	2.9726e -5	4.7964e -8	0.265
14	1.5	3.6774e -7	1.5788e -7	2.1875
12	2	3.7764e -6	9.3153e -7	2.34375

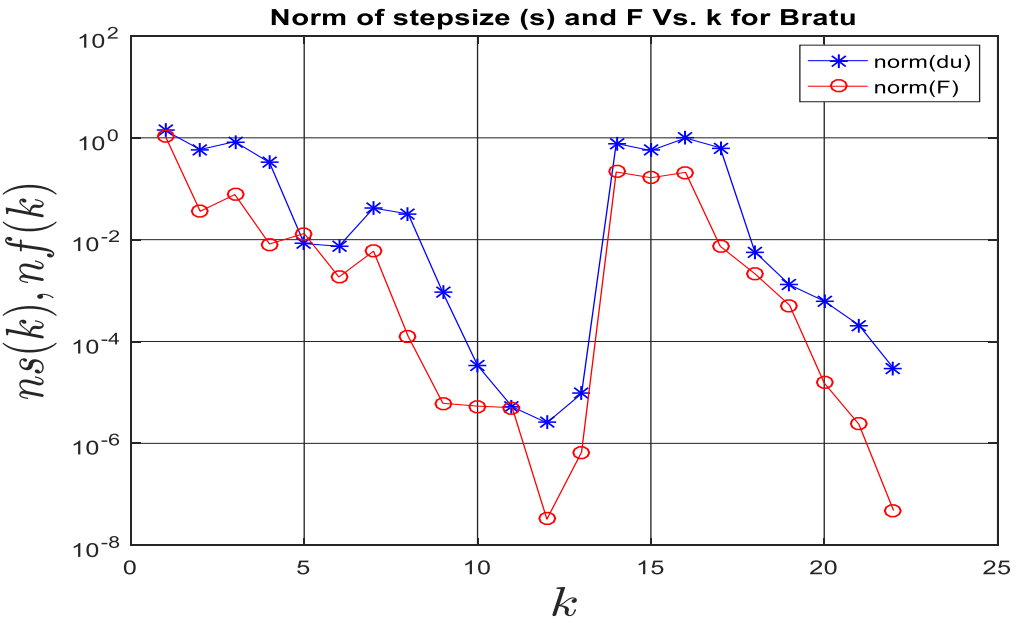

Figure 4.19 Broyden Method I Euclidian Norm of Errors versus number of iteration for Bratu problem.

Analysis

Figure 4.18 cites a comparison between the Broyden Method I solution obtained by and its analytic solution. From this comparison, one can see a good agreement between the analytic solution and the Broyden method I (approximate) solution.

As shown in Table 4.8, Broyden Method I converges for Bratu problem and terminates at the 22, 14 and 12 iterations for 3 different values of parameters 1, 1.5 and 2 respectively. The CPU-time taken for each parameter also indicates Broyden method I takes more time and iteration for computation than Newton Method. Moreover as shown in Figure 4.19, which proves the convergence of Broyden method I for Bratu problem but the norm errors are not continuously decreasing, it is not stable. This is because of Numerical approximation of Jacobian matrix, which also increases the number of iteration and CPU time needed for computation of solution than pure Newton method.

Problem-2- consider Troesch's 1-D nonlinear problem defined in section (1.2)

$$u'' = \lambda \sinh(\lambda u) \qquad (4.1.1.3)$$

Subject to the dirichlet boundary conditions:

$$u(0) = 0 \ \ and \ \ u(1) = 1. \qquad (4.1.1.4)$$

The analytical solution for this problem given by, equation (1.2.7).

Using equation (4.1.1.3) and (3.3), we can create NLSAEs for Troesch's problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1,2,...,N \qquad (4.1.1.5)$$

Where $f = \lambda \sinh(\lambda u(j))$

To solve eq. (4.1.1.5) by using Broyden method I and inputs of Newton method for the same problem gives the following results:

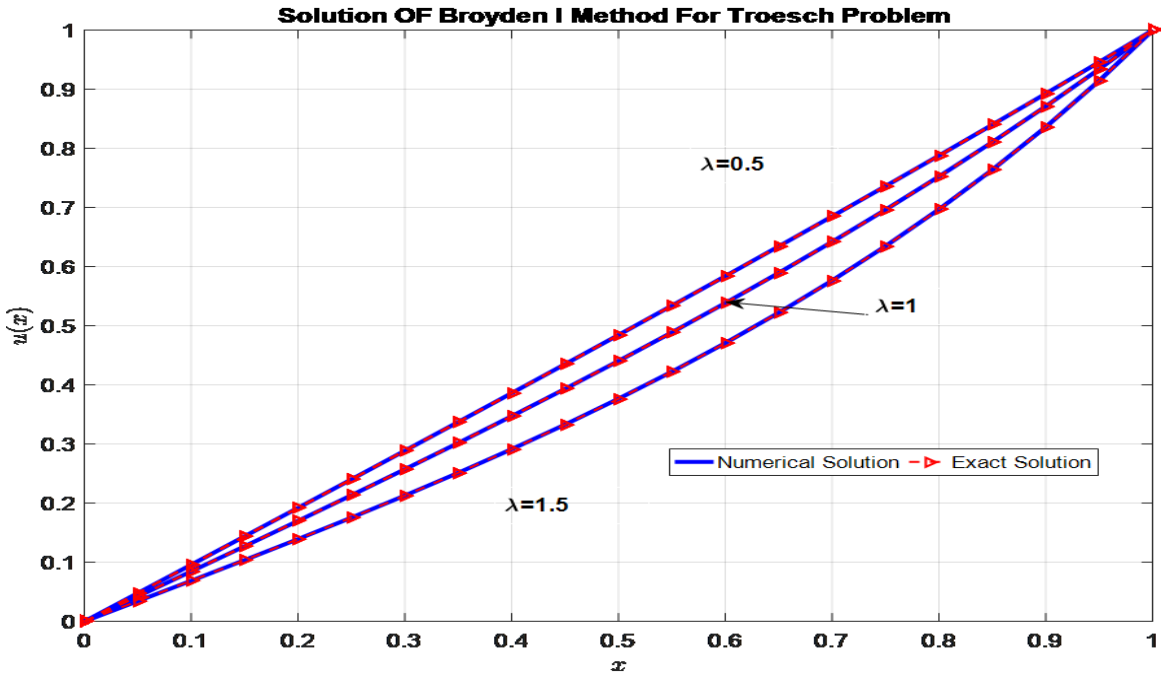

Figure 4.20 Comparison between the approximated solution and exact solution of Troesch problem using Broyden method I for 3 different parameters (lambda).

Table 4.9: Broyden method I Euclidian norm of error, number of iteration and CPU time for each value of parameter of Troesch's problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
33	0.5	3.10367e -6	1.7527e -7	0.203125
15	1	1.01149e -3	4.4439e -7	2.390625
15	1.5	9.2402e -4	5.304e -7	2.65625

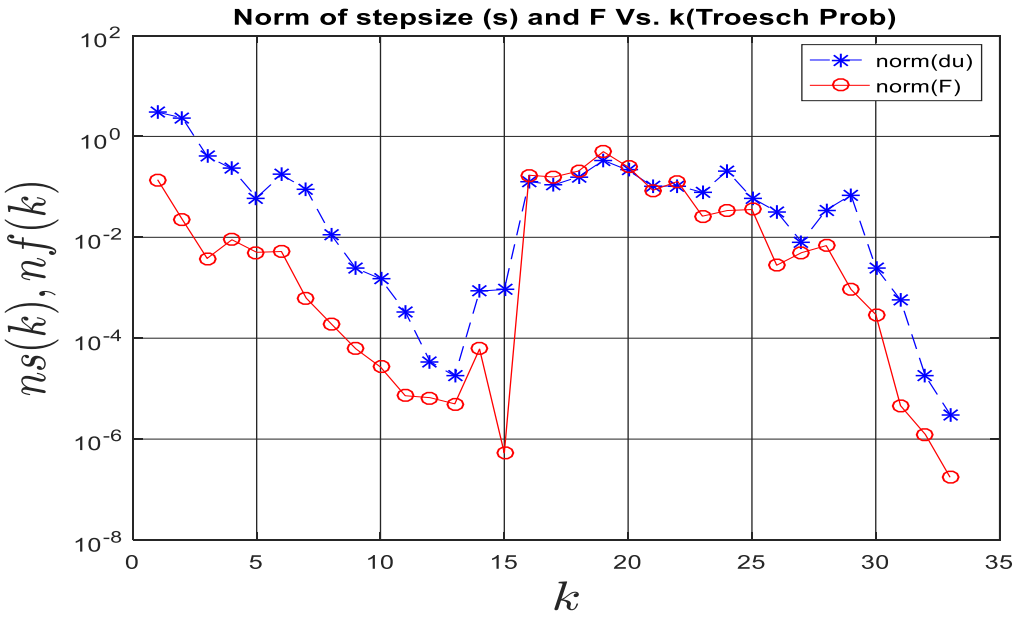

Figure 4.21 Broyden Method I Euclidian Norm of Errors versus number of iteration for Troesch's problem.

Figure 4.20 shows, a comparison between the approximate solution obtained by Broyden method I and its exact solution. From this comparison, one can see a good agreement between the exact solution and the Broyden method I (approximate) solution.

As shown in Table 4.9 Broyden method I converge for Troesch problem and terminate within 33, 15, and 15 iterations. The CPU-time taken for each parameter also indicates Broyden method I takes more time and iteration for computation than Newton Method for Troesch's problem. Moreover as shown in Figure 4.21, which proves the convergence of Broyden method I for Troesch problem and like previous problem, error is not stable. This is because of approximation of Jacobian matrix, which also increases both the number of iteration and CPU time needed for computation of solution than Newton method.

4.2. Broyden's Method II

A second approach to Broyden's Method avoids the relatively expensive matrix solver step $A_i s_{i+1} = F(x_i)$. Since we are at best only approximating the derivative F' during the Broyden I, we may as well be approximating the inverse of F' instead, which is what is needed in the Newton step.

The derivation of Broyden from the point of view of $B_i = A_i^{-1}$. We would like to have

$$s_{i+1} = B_{i+1} y_{i+1}, \qquad (4.2.0)$$

Where $s_{i+1} = x_{i+1} - x_i$ and $y_{i+1} = F(x_{i+1}) - F(x_i)$, still satisfy

$$A_{i+1} w = A_i w, \qquad (4.2.1)$$

A matrix that satisfies both (4.2.0) and (4.2.1) is

$$B_k = B_{k-1} + \frac{\left(s^{(k)} - B_{k-1} s^{(k)}\right)}{\left\|y^{(k)}\right\|^2} (y^{(k)})^T \ldots \qquad (4.2.2)$$

The new version of the iteration, which needs no matrix solve, is

$$x_{i+1} = x_i - B_i F(x_i). \qquad (4.2.3)$$

The resulting algorithm is called Broyden's Method II.

x_0 =initial vector

A_0 =initial Jacobian matrix ($B_i = A_i^{-1}$.)

For i =0,1,2,...

$$x_{i+1} = x_i - B_i F(x_i)$$

$$B_i = B_{i-1} + \frac{\left(s^{(i)} - B_{i-1} s^{(i)}\right)}{\left\|y^{(i)}\right\|^2} (y^{(i)})^T.$$

where $s_i = x_i - x_{i-1}$ and $y_i = F(x_i) - F(x_{i-1})$.

To begin, an initial vector x_0 and an initial guess for B_0 are needed. If it is impossible to compute derivatives, the choice $B_0 = I$ can be used.

A perceived disadvantage of Broyden II [**62**] is that estimates for the Jacobian, needed for

some applications, are not easily available. The matrix B_i is an estimate for the matrix inverse of the Jacobian. Broyden I, on the other hand, keeps track of A_i, which estimates the Jacobian. For this reason, in some circles Broyden I and II are referred to as "Good Broyden'' and "Bad Broyden,'' respectively.

Both versions of Broyden's Method converge superlinearly , slightly slower than the quadratic convergence of Newton's Method. If a formula for the Jacobian is available, it usually speeds convergence to use the inverse of $F'(x_0)$ for the initial matrix B_0.

4.2.1. Application of the Broyden's method II

- **Algorithm(4.2): Solves nonlinear BVPs by using Broyden Method II**

INPUT: -

L	**– length of problem domain (b-a)**
u	**- Initial guess of problem solution**
λ	**- parameter values of lambda**
n	**- number of spaces**
maxiter	**- maximum number of iterations**
B	**- Initial approximation of Jacobian matrix**

STEP 1: Do steps-2 up to 6 until iteration(k) <maxiter and err < tolerance

STEP 2: call function F for u_0 .

$$F_k = \text{FC}(u_0, h, n, \lambda);$$

Update solution (u) & compute F_{k+1}

$$u_k = u_0 - B \backslash F_k \text{ and } \quad F_{k+1} = \text{FC}(u_k, h, n, \lambda);$$

STEP 3: compute change in u(s) & change in F(y)

$$s = u_{k+1} - u_k \ , \ y = F_{k+1} - F_k$$

STEP 4: Update Jacobian matrix **B** by equation (4.2.2)

STEP 5: Compute norm error of $du = BF$ and F

STEP 6: Display Numerical and graphical output

Based on Algorithm (4.2), we wrote a MATLAB program for solving problems 1 and 4.

- **Numerical and Graphical output of problems using Broyden method II**

Broyden method II applied and tested on the following nonlinear BVPs that are define in section (1.2).

Problem-1- consider Bratu 1-D nonlinear problem defined in section (1.2)

$$u''(x) + \lambda e^{u(x)} = 0 \qquad (4.2.1.0)$$

Subject to the dirichlet boundary conditions:

$$u(0) = u(1) = 0 \qquad (4.2.1.1)$$

The Analytic solution for this problem given by, equation (1.2.3)
Using equations (4.2.1.0) and (3.3), we can create NLSAEs for Bratu problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1,2,...,N \qquad (4.2.1.2)$$

Where $f = -\lambda e^{u_j}$

Equation (4.2.1.2) solved by using Broyden method **II** and similar inputs to Broyden I method for the same problem using a MATLAB Program gives the following results:

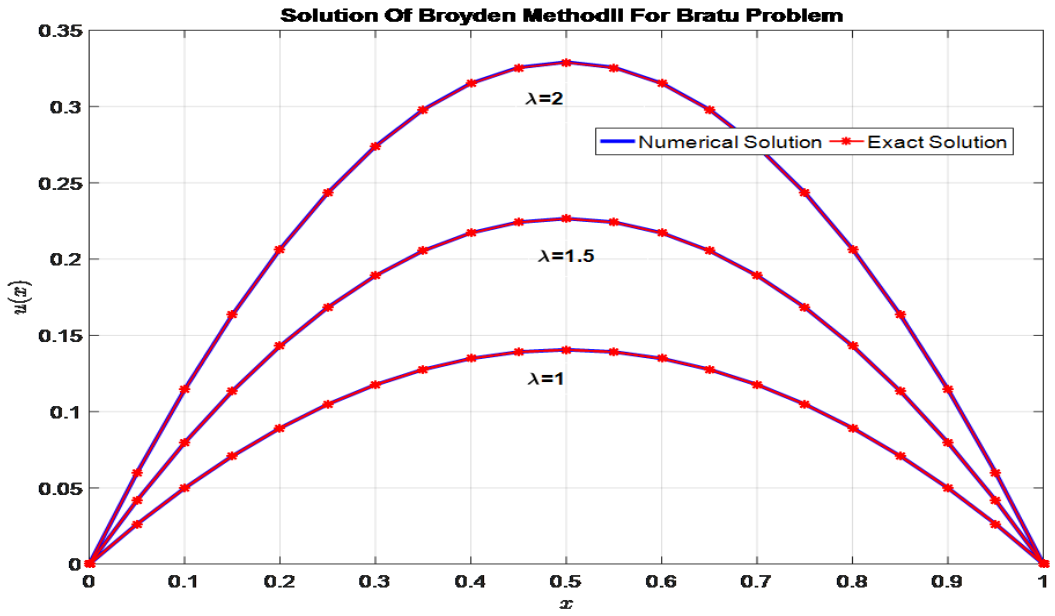

Figure 4.22 Comparison between the approximated solution and exact solution of Bratu problem using Broyden method II for 3 different parameters.

Table 4.10 : Broyden method II Euclidian norm of error, number of iteration and CPU time for each value of parameter of Bratu problem.

Number of iteration(k)	Parameters λ	Euclidian norm of Newton step (du)	Euclidian norm of fuction (F)	CPU time
24	1	6.551e -9	1.662e -9	0.421875
17	1.5	2.3258e -7	1.3832e -7	1.703125
16	2	2.95515e -7	1.6689e -7	2.421875

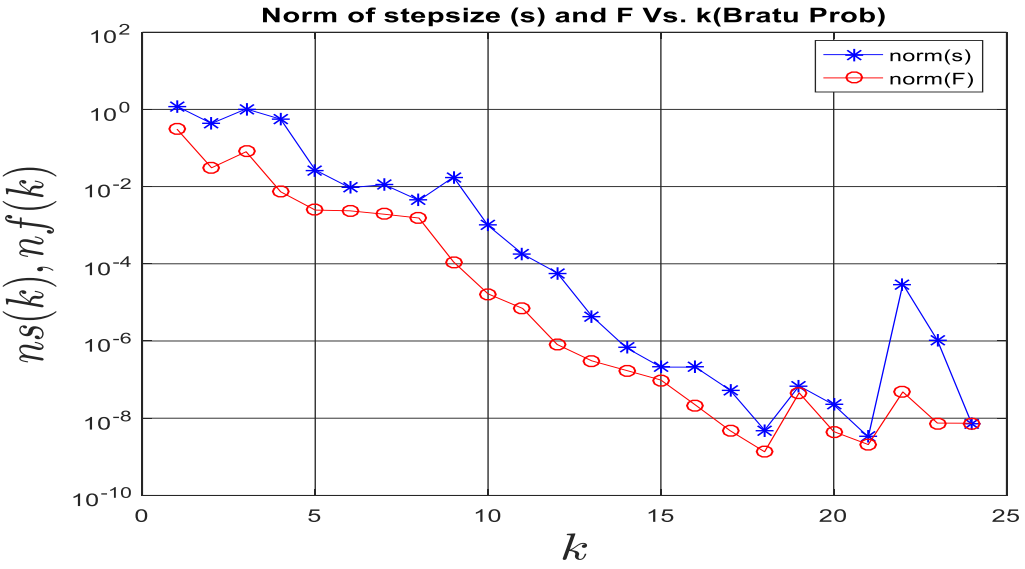

Figure 4.23 Broyden method II Euclidian Norm of Errors versus number of iterations for Bratu problem.

Fig.4.22 cites a comparison between the Broyden method II and analytic solution for three different parameters 1, 1.5 and 2 for Bratu problem. It shows good agreement between the analytic solution and the Broyden method II solution for all three different cases. We can also see the influence of parameters lambda on the solution u(x).

As shown in Table 4.10 Broyden method II converges for Bratu problem and terminate within 24, 17, and 16 iterations for three different values of parameters (λ). The CPU-time taken for each parameters also indicates Broyden method II takes more time and iteration than Newton Method but less computation time and iteration than Broyden method I for Bratu problem. Moreover, as shown in Figure 4.23, that proves the convergence of Broyden method II for Bratu problem and as if previous problems solved by Broyden's, method I, Euclidean norm error is not continuously decreasing.

Problem -4 - consider Thermal Explosion nonlinear BVPs defined in section (1.2)

$$u''+\lambda e^{u}=0, \quad x\in(0,1), \tag{4.2.1.3}$$

With the associated Neumann boundary conditions

$$u'(0)=0,\ u(1)=0.$$

The Analytic solution for this problem by, equation (1.2.12)

Using equations (4.2.1.3) and (3.3), we can create NLSAEs for Thermal Explosion problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1,2,...,N$$

Where $f = -\lambda e^{u_j}$

The above NLSAEs can be solve by using Broyden method II and take similar inputs of Broyden I method of similar problem using a MATLAB program gives the following result:

Table 4.11 : Broyden method II Euclidian norm of error, number of iteration and CPU time for each value of parameter for Thermal Explosion problem.

Number of iteration(k)	Parameter Lambda	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
48	0.1	4.8495 X 10^-7	7.598 X 10^-9	0.078125
22	0.2	4.667 X 10^-7	5.301X 10^-10	0.640625

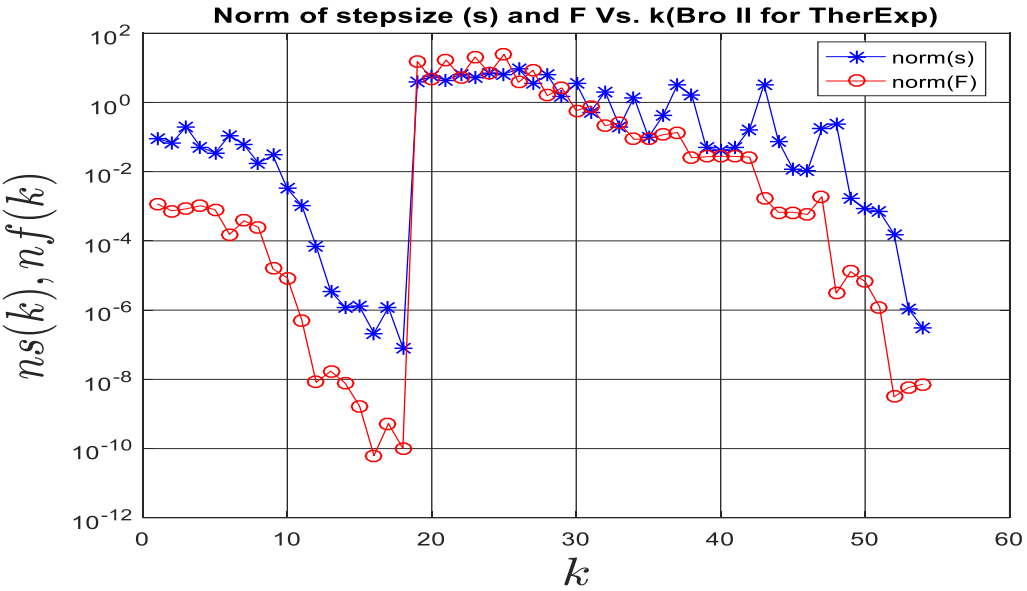

Figure 4.24 Broyden's method II Euclidian Norm of Errors versus number of iteration for Thermal Explosion problem.

As shown in Table 4.11 Broyden's method II converges for Thermal Explosion problem and terminate within 48, and 22 iterations for $\lambda = 0.1$ and $\lambda = 0.2$ respectively. The CPU-time also shows Broyden method II takes more time and iteration for computation than Newton's method. Moreover as shown in Figure 4.24 which proves the convergence of Broyden method II for Thermal Explosion problem but the Norm error is not continuously decrease, like Broden I, it is not stable.

4.3. Safeguarded Broyden Method for BVP

Broyden's method is the most popular multivariate generalization of the univariate secant method. Broyden's method generates sequence of vectors $x^{(k)}$ and matrices $A^{(k)}$ that approximate the root of f and the Jacobian f' at the root, respectively. Broyden's method begins with the analyst supplying a guess $x^{(0)}$ for the root of the function and a guess $A^{(0)}$ for the Jacobian of the function at the root. Often, $A^{(0)}$ can be approximate by the numerical Jacobian of f at $x^{(0)}$.

The Newton's method reads by (3.1.1.1) replace by :

$$x_{k+1} \approx x_k - B_k F(x_k), \qquad (4.3.0)$$

Where B_k is an approximation of $\left[J(x_k)\right]^{-1}$.

Like Newton method, the same problem also occurs in Broyden's, here we are also concentrated on the problem, a poor starting value.

If taking Broyden iteration as,

$$x_{k+1} \approx x_k - d * step, \quad where \quad d = B_k F(x_k), \qquad (4.3.1)$$

step of $(x - d * step)$ does not offer an improvement over the current iterate x, then one back-steps toward the current iterate x by repeatedly cutting *step in half* until $x - d * step$ does offer an improvement like Newton Safeguarded method an improvement is measured by the Euclidean norm.

$$\|f(x)\|_2 = \frac{1}{2} f(x)^T f(x) \qquad (4.3.2)$$

$\|f(x)\|_2$ is zero at a root and positive elsewhere an improvement over the previous iterate, if it reduces the function norm. Back stepping also prevents quasi-Newton methods from taking a large step in the wrong direction, substantially improving their robustness.

Like safeguarded Newton's method, we use a very simple condition in which the back steps continue until conditions (4.3.3) are satisfied

$$\|f\ x\| < \|f\ x + \qquad \| \ and\ \ step > Threshold \qquad (4.3.3)$$

Here Threshold are approximated by user, in our case we use 2^{-4} and step initially start with 1 and improve in each iteration ($step/2$) until the condition in (4.3.3) satisfied.

4.3.1. Application of the Safeguarded Broyden's Method

To approximate the solution of mechanics problems defined in section (1.2), we wrote an algorithm (4.3) for Safeguarded Broyden method.

- **Algorithm (4.3): Solves nonlinear BVPs by using SBroyden Method**

INPUT: L – length of problem domain (b-a)

u, λ, - initial guess and parameter values of lambda,

n, h -- number of spaces and space between nodes

Maxiter, tol, α – max iteration, error tolerance and Relax factor

STEP-1 Compute F for initial guess solution (u)

$$F = FC(u, h, n, lam)$$

STEP-2 DO STEP- 3 until iter(k) < maxiter else go to STEP- 8

STEP-3 Compute numerical Jacobi step, hh and solution u .

$$hh = \sqrt{eps} * u_i \ , u = u + hh$$

Compute F and Numerical Jacobi J

$$F = FC(u, h, n, lam), \quad J = (F_{new} - F_{old}) / hh$$

STEP-4 Compute Newton step size s_{old}

$$s_{old} = -J \setminus F$$

Update solution(u) & compute error

$$u = u + \alpha s, \ error = \|F\|_2$$

STEP-5 When error > Convcrit & K < maxiter

$$k = k + 1 \ \& \ \text{do STEP-3}$$

Compute s_{temp} by $s_{temp} = -J \setminus F$

STEP-6 If $\max|s_{temp}| < \max|s|$ satisfied compute $\alpha = \alpha / 2$

Compute and update Newton step by

$$u_{new} = u_{old} + \alpha s_{temp}$$

STEP-7 update Jacobi by equation (4.2.2) and α

$$\alpha = \min(1., \alpha * 2)$$

STEP-8 display Numerical and Graphical Output

- **Numerical and Graphical output of problems by using SBroyden method**

The numerical method, Safeguarded Broyden is applied and tested on the following nonlinear BVPs that are define in section (1.2) and the numerical result and norm error are displayed by using MATLAB program.

Problem-2- consider Troesch's 1-D nonlinear problem defined in section (1.2)

$$u'' = \lambda \sinh(\lambda u) \qquad (4.3.1.0)$$

Subject to the dirichlet boundary conditions:

$$u(0) = 0 \ \ and \ \ u(1) = 1. \qquad (4.3.1.1)$$

Whose general solution given by, equation (1.2.7),

Using equations (4.3.1.0) and (3.3), we can create NLSAEs for Bratu problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1, 2, \ldots, N \qquad (4.3.1.2)$$

Where $f = -\lambda e^{u_j}$

To solve eq. (4.3.1.2), by using Safeguarded Broyden method and inputs (Broyden II method inputs for the same problem) using a MATLAB Program gives the following results:

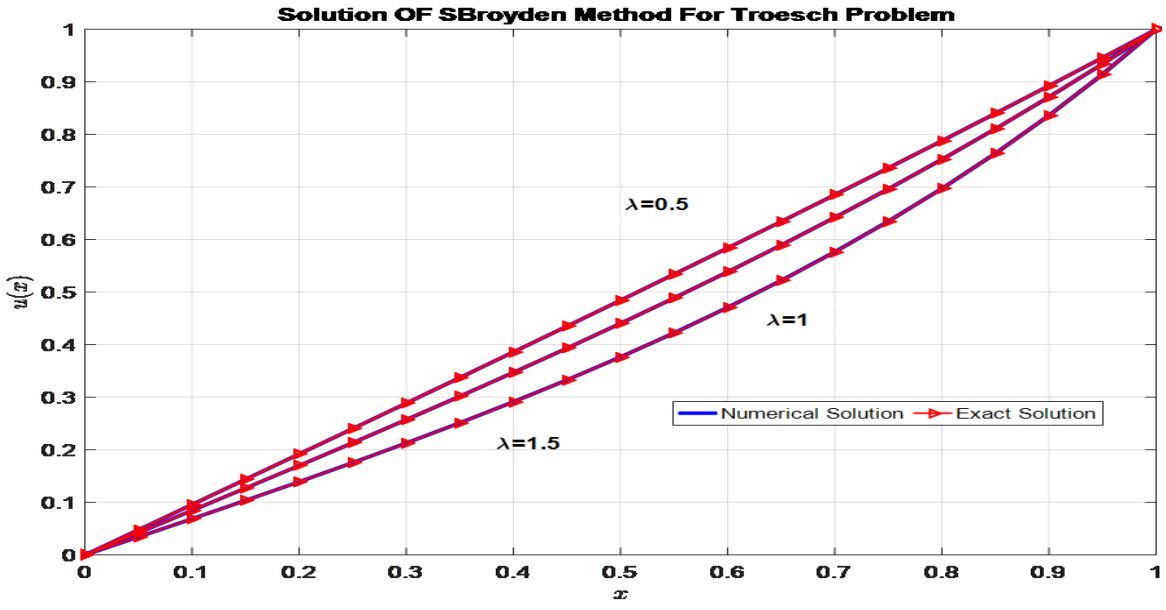

Figure 4.25 Comparison between the approximated solution and exact solution of Troesch problem.

Table 4.12: Safeguarded Broyden method Euclidian norm of error, number of iteration and CPU time for each value of parameter of Troesch's problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of fuction (F)	CPU time
4	0.5	4.008e -6	4.198e -11	0.109375
3	1	1.3202e -6	1.11861e-13	2.578125
4	1.5	3.91387e -10	1.80122e-11	2.734375

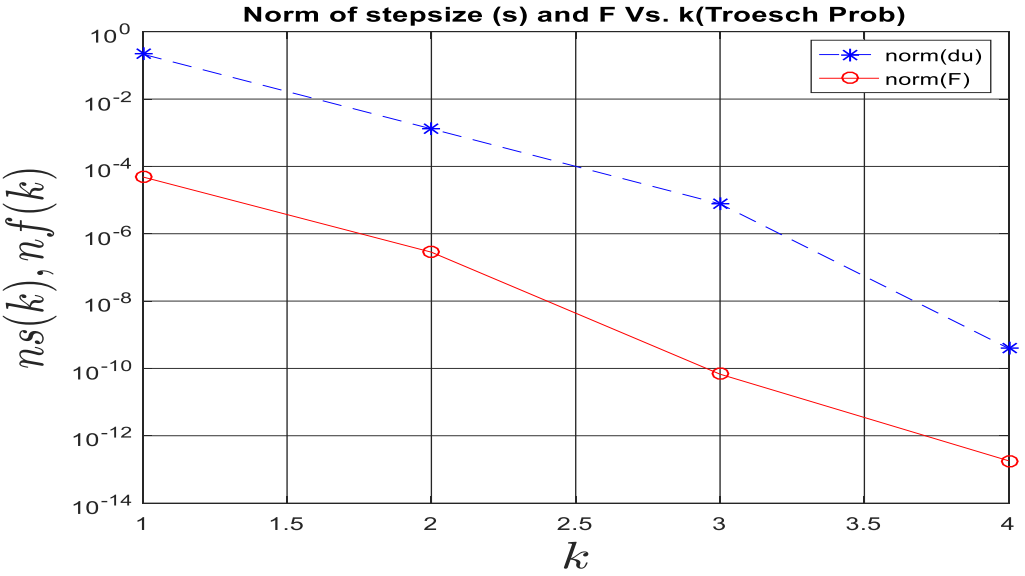

Figure 4.26 Safeguarded Broyden Method Euclidian Norm of Errors versus number of iterations for Troesch problem

As shown in Figure 4.25 a comparison between the approximate solution obtained by SBroyden method and exact solution. From this comparison, one can see a good agreement between them.

As shown in Table 4.12 Safeguarded Broyden method converges for Troesch's problem and terminate within 4, 3, and 4 iterations for three different values of parameter's (lambda). Safeguarded Broyden method takes less iteration for computation than Broyden method II. Moreover as shown in Figure 4.26 for Troesch's problem the error getting smaller in between successive iterates implies that getting closer to the answers. We observe that Safeguarded Broyden method error is stable as compared to Broyden method II for Troesch problem and Euclidian norm of error of both Newton step (du) and function

are continuously decreasing by back stepping that is used to reduce a large step (du) in the wrong direction by comparing two consecutive norms of a function.

Problem-3- consider Heat Transfer nonlinear BVPs defined in section (1.2)

$$\frac{d^2u}{dx^2} = c\left(u_{sur}^4 - u^4\right), \qquad (4.3.1.3)$$

Subject to the boundary conditions:

$$u(0) = 900 \ \ and \ \ u(1) = 900. \qquad (4.3.1.4)$$

Using equation (4.3.1.3) and (3.3), we can create NLSAEs for Heat transfer problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1,2,\ldots,N \qquad (4.3.1.5)$$

Where $f = c\left(u_{sur}^4 - u_j^4\right)$

To solve eq.(4.3.1.3), by using Safeguarded Broyden method and inputs: initial guess solution (u=ones(n,1)*900), error tolerance for function(tol= 10^{-}), number of iteration=50 and value of parameter $c = \left\{10^{-10}, 10^{-9}.10^{-8}\right\}$ using a MATLAB Program for Heat Transfer problem gives the following results:

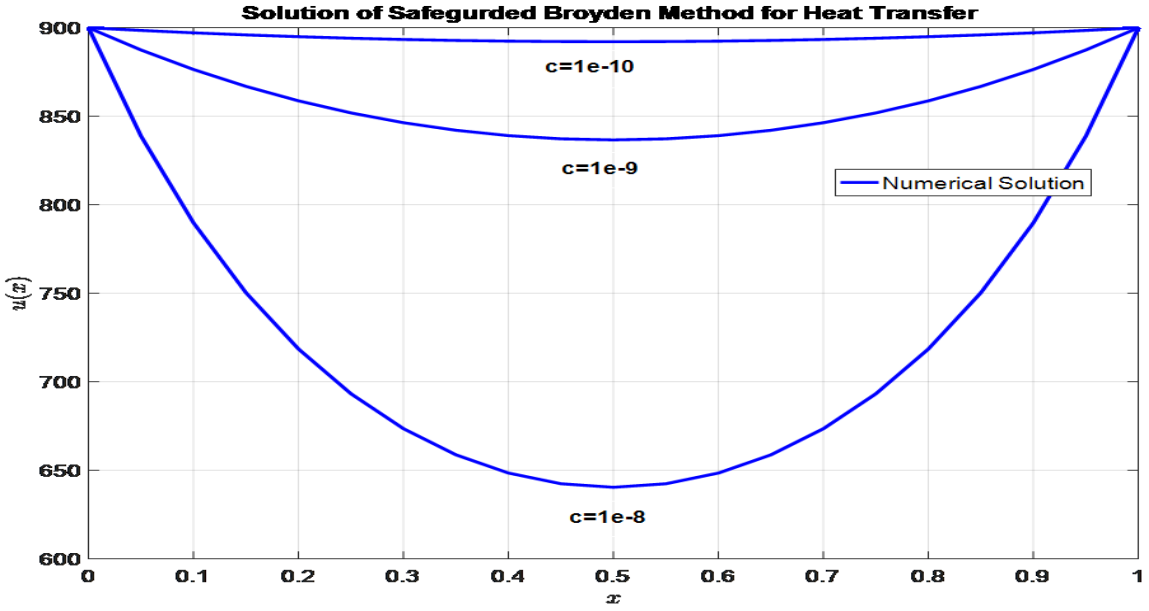

Figure 4.27 Safeguarded Broyden method approximated solution of HT problem.

Table 4.13: Safeguarded Broyden method Euclidian norm of error, number of iteration and CPU time of HT problem.

Number of iteration(k)	Parameter C	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
4	1e-10	4.94139 X 10^ -12	1.60778 X 10^-13	0.140625
5	1e-9	1.9765 X10^-8	8.197 X10^-12	1.703125
9	1e-8	1.32064 X10^ -9	1.08611 X 10^ -11	2.484375

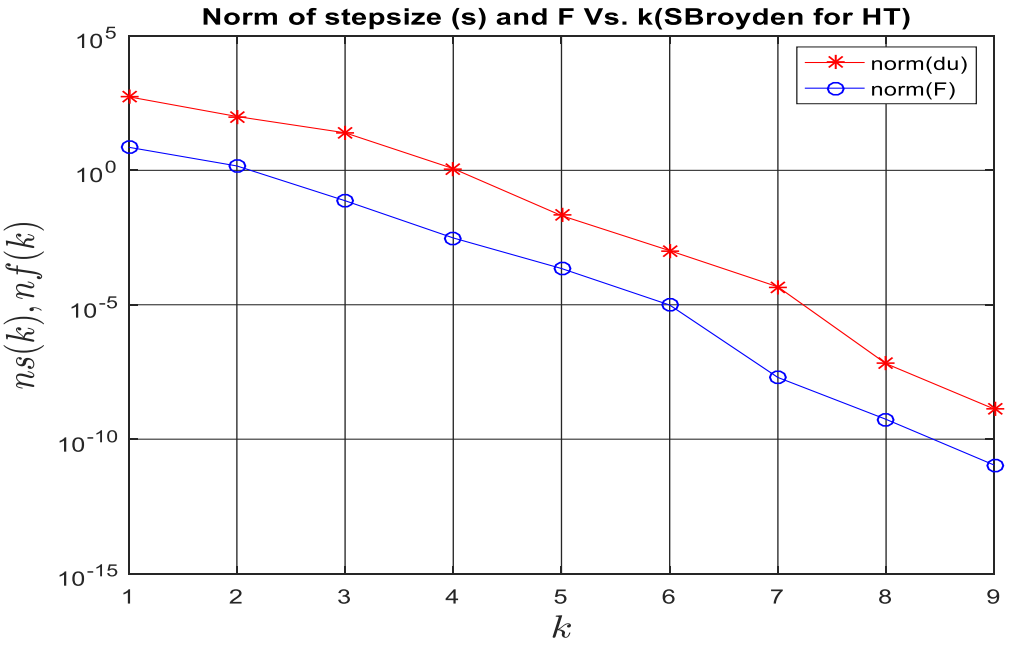

Figure 4.28 Safeguarded Broyden Method Euclidian Norm of Errors versus number of iterations for HT problem

As shown in Figure 4.27, the approximate solution obtained by Safeguarded Broyden method for Heat transfer problem, result agree with the result on book [**20**]..

As shown in Table 4.13 Safeguarded Broyden method converges for HT problem and terminate within 4, 5, and 9 iterations for three different values of parameter's (c). Moreover as shown in Figure 4.28 for HT problem the error getting smaller in between successive iterates implies that getting closer to the answers. The main reason is that Euclidian norm of error of both Newton step (du) and function are continuously decreasing by back stepping.

4.4. Damped (Softening) Broyden Method for BVP

One main problem of Broyden Method is when its increment is too large, which is also a problem of Newton Method. One way to diminish this problem is to "soften" or "dampen" the iterate by putting a fractional factor on the iterate. First let us start with Newton iterate:

$$u^{(k+} = \quad - \quad {}^{-} J(u^{(k)}) \qquad (4.4.0)$$

Where α is a number smaller than one. It should be clear that convergence for Newton's method that introducing the softening factor α destroys the quadratic convergence of the method. This case also affects the superlinear convergence of Broyden Method. This raises the question of stopping. In our case, we stop iteration when Euclidian norm of F gets less than the desired criteria and the number of iteration is up to a certain limit , but if increment has been multiplied by alpha α , then convergence could happen immediately if alpha is very small. It is important to make sure $J\ u^{(k)\ -1} f(u^{(k)})$ (or increment) is not multiplied by alpha before the test is done.

In this section, we present, the Broyden softening by considering Newton Damping as a base because both of them use increment factor, we present our examined method as follows:

Compute the first step by Armijo rule:

$$h = \sqrt{(eps)}\ \ *u(i) \qquad (4.4.1)$$

Where 'eps' is small constant number and $u(i)$ is the initial guess value of the nonlinear system of equations $F(u) = 0$.

Then update the solution by using (4.4.0) we get

$$u* = u + h; \qquad (4.4.2)$$

Compute Numerical Jacobian of f by using (4.4.0) and (4.4.1):

$$J = \frac{f(u*) - f(u)}{h} \qquad (4.4.3)$$

Compute Newton step size, and then update solution by newton step

$$s = -J \setminus f(u) \qquad (4.4.4)$$

$$u = u + \lambda s \qquad (4.4.5)$$

Where λ is softening factor whose value starts from 1 up to a certain minimum limit. Iteratively softening or damping of the step (s) until the norm of the function F satisfies

$$\|F\|_2 < \qquad (4.4.6)$$

In addition to that; if equation (4.4.6) is not satisfied the following condition is checked and makes the following actions.

$$\max\|s\| < \max\|\lambda step\| \qquad (4.4.7)$$

If condition (4.4.7) is satisfied damping Lambda (step size) is needed then we decrease it by $\lambda/2$ else the value of softening factor (Lambda) is suitable for our system. In this way, we can decrease the step size by multiplying it by optimal softening factor.

4.4.1. Application of the Damped Broyden Method

To approximate the solution of mechanics problems define in section (1.2), we wrote an algorithm (4.4) for Damped Broyden method.

Algorithm (4.4): Solves nonlinear BVPs by using DBroyden Method.

INPUT: L – length of problem domain (b-a)
u, ul, uf - guess solution ,initial & final boundary value
λ, n, h - parameter, number of spaces and space between nodes
Maxiter,tol- maximum number of iteration, tolerance
α --- Relaxation factor

STEP-1 Compute F for initial guess solution (u)
$F = FC(u, h, n, lam)$

STEP-2 DO STEP-3 until iter(k) < maxiter else go to step 8

STEP-3 Compute numerical Jacobi step,hh and update solution u .
$hh = \sqrt{eps} * u_i$, $u = u + hh$
Compute F and Numerical Jacobi J
$F = FC(u, h, n, lam)$, $J = (F_{new} - F_{old})/hh$

STEP-4 Compute Newton step size s_{old} s
$s_{old} = -J \backslash F$
Update solution(u) & compute error
$u = u + \alpha s$, $error = \|F\|_2$

STEP-5 When error > Convcrit & K < maxiter
$k = k + 1$ & do STEP-3
Compute s_{temp} by
$s_{temp} = -J \backslash F$

STEP-6 If $\max|s_{temp}| < \max|s|$ satisfied compute α
$\alpha = \alpha / 2$
Compute and update Newton step by

$u_{new} = u_{old} + \alpha s_{temp}$

STEP-7 update Jacobi by equation (4.2.1) and α

$\alpha = \min(1., \alpha * 2)$

STEP-8 display Numerical and Graphical Output

Based on Algorithm (4.4), we wrote a MATLAB program that solves problems 1 and 2.

Numerical and Graphical output of problems using Damped Broyden method

The numerical method, Damped Broyden is applied and tested on the following nonlinear BVPs that are define in section (1.2) and the numerical result and norm error are displayed by using MATLAB program.

Problem-1- consider Bratu 1-D nonlinear problem defined in section (1.2)

$$u''(x) + \lambda e^{u(x)} = 0 \qquad (4.4.1.0)$$

Subject to the dirichlet boundary conditions:

$$u(0) = u(1) = 0 \qquad (4.4.1.1)$$

The Analytic solution for this problem given by, equation (1.2.3).

Using equations (4.4.1.0) and (3.3), we can create NLSAEs for Bratu problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1, 2, ..., N \qquad (4.4.1.2)$$

Where $f = -\lambda e^{u_j}$

To solve eq. (4.4.1.2), by using Damped Broyden method using similar inputs of Broyden II method of similar problem gives the following results:

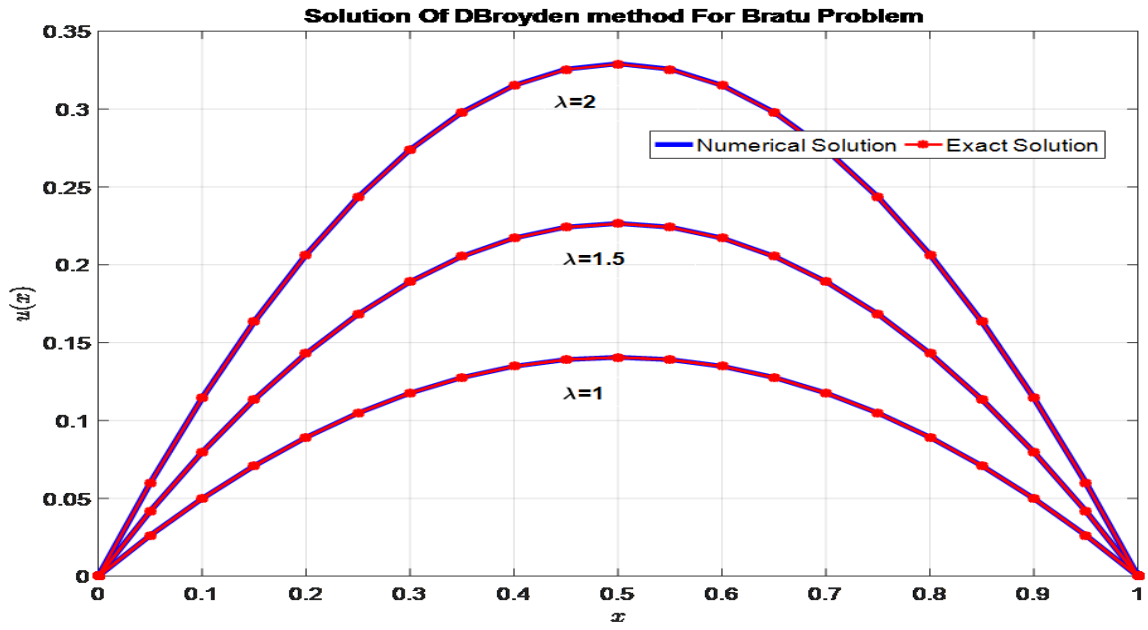

Figure 4.29 Comparison between the approximated solution and exact solution of Bratu problem.

Table 4.14: Damped Broyden method Euclidian norm of error, number of iteration and CPU time for each value of parameter of Bratu problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
3	1	3.181e -3	4.055e -9	0.70312
4	1.5	5.57e -5	1.3897e -9	2.375
4	2	8.466e -5	1.991e -9	2.5625

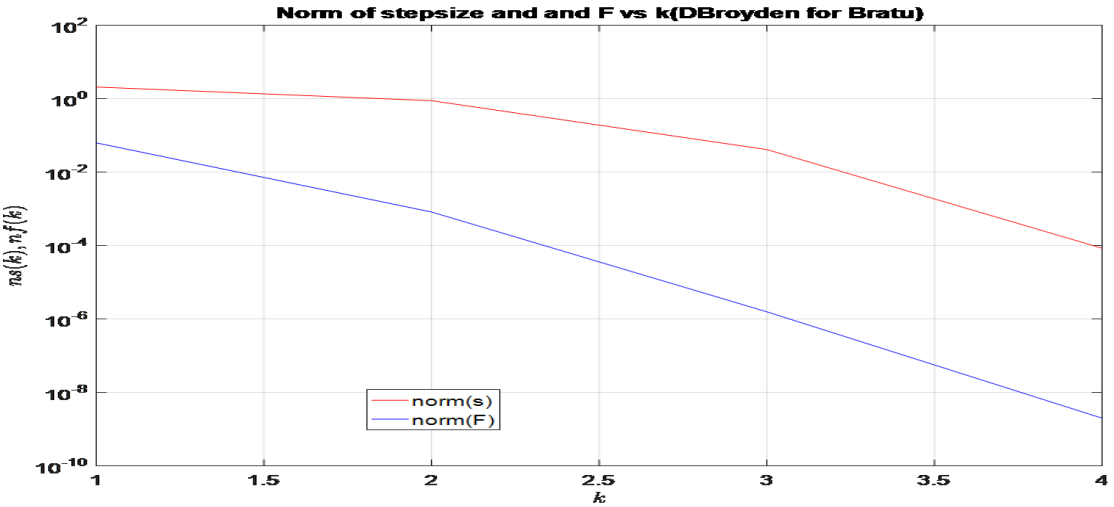

Figure 4.30 Damped Broyden Method Euclidian Norm of Errors versus number of iterations for Bratu problem.

Fig.4.29 cites a comparison between Damped Broyden method and exact solution. From this comparison, one can see a good agreement between the exact solution and the Damped Broyden method solution, even much better than pure Broyden methods.

As shown in Table 4.14 Damped Broyden method converges for Bratu problem and terminate within 28, 20, and 20 iterations for three different values of parameters (lambda). The CPU-time taken for each parameter also indicates Damped Broyden method takes less time and errors for computation than Broyden method II. Moreover, as shown in Figure 4.30, that proves the convergence of Damped Broyden method for Bratu problem and Euclidean norm error for Newton step (du) and function are continuously decreasing.

However, the method converges even the initial guess solution is far from exact solution. That is a better advantage as compared to pure Broyden method.

Problem-2- consider Troesch's 1-D nonlinear problem defined in section (1.2)

$$u'' = \lambda \sinh(\lambda u) \qquad (4.4.1.3)$$

Subject to the dirichlet boundary conditions:

$$u(0) = 0 \ \ and \ \ u(1) = 1. \qquad (4.4.1.4)$$

The theoretical solution for this problem given by, equation (1.2.7).

Using equation (4.4.1.3) and (3.3), we can create NLSAEs for Troeschs problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1, 2, ..., N \qquad (4.4.1.5)$$

Where $f = \lambda \sinh(\lambda u(j))$

Equation (4.4.1.5) solved by using Damped Broyden method and using inputs of Broyden method II for the same problem gives the following results:

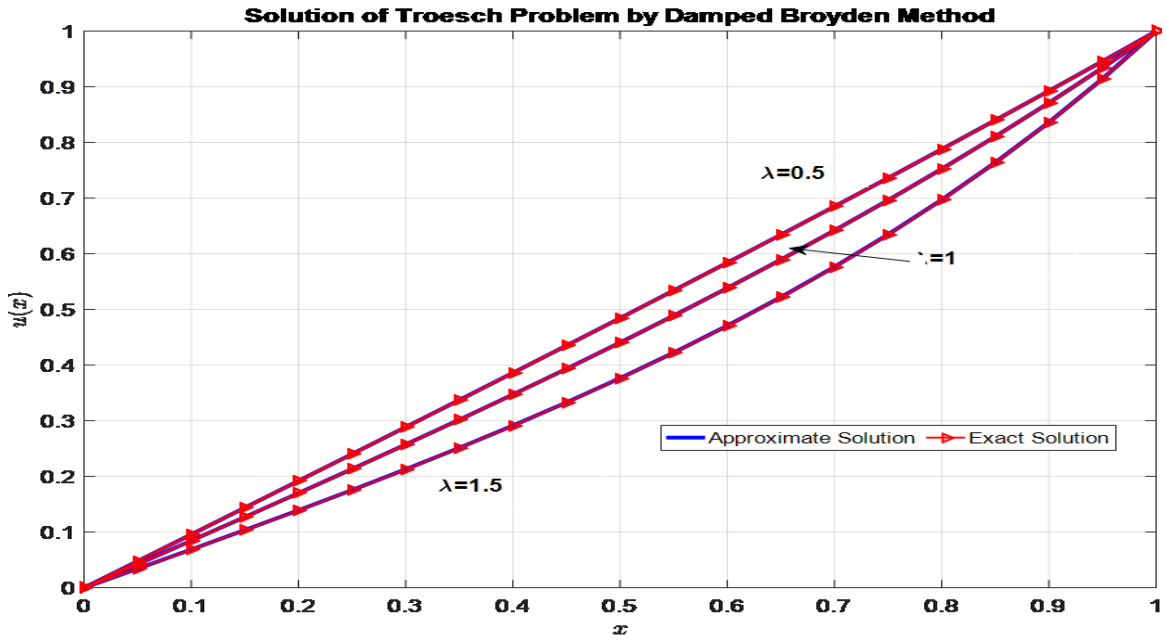

Figure 4.31Comparison between the approximated solution and exact solution of Troesch problem.

Table 4.15: Damped Broyden method Euclidian norm of error, number of iteration and CPU time for each value of parameter of Troesch's problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of fuction (F)	CPU time
32	0.5	4.38831 X 10^- 5	3.11534 X10^-8	0.406250
2	1	1.30169 X 10^-3	3.1189 X10^-8	1.37500
3	1.5	2.12052 X10^-6	3.0624 X10 ^-8	1.90625

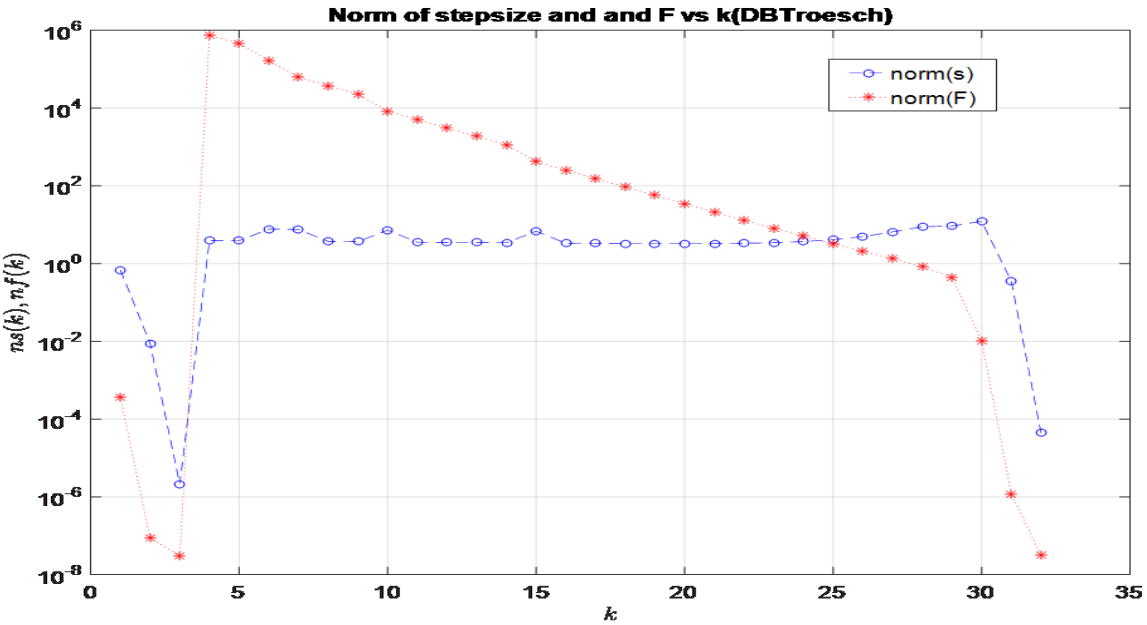

Figure 4.32 Damped Broyden Method Euclidian Norm of Errors versus number of iterations for Troesch problem.

As shown in Fig. 4.31, a comparison between the approximate solution obtained by DBroyden method and exact solution. This shows a good agreement between the exact solution and the DBroyden method solution for Troesch's problem.

As shown in Table 4.15, Damped Broyden method converges for Troesch problem and terminates within 32, 2, and 3 iterations for three different values of parameters (lambda). This shows the first parameter computation takes repeated damping process to correct the path to solution. Generally, the CPU-time taken for each parameter also indicates Damped Broyden method takes less time for computation than Broyden method II. Moreover as shown in Figure 4.32 for Troesch problem the error is not stable, around fifth iteration Euclidian norm of errors increase rather decrease. This is because of the inverse of the Numerical approximation of Jacobian matrix.

4.5. Modified Broyden's Classes of Method for BVPs

Introduction

One of the problems of Broyden's method is that it is not self-correcting in order to avoid this we propose two efficient algorithms based on Broyden's methods by using the central finite difference and modification of Newton's method for solving systems of nonlinear equations by Abu-Hour [**63**].

In this section, we will see how to improve and modify Broyden's methods to increase the accuracy and the order of convergence. The algorithms of modified Broyden's with central finite difference of types I and II are presented.

Broyden's classes with central finite difference (BC)

Broyden can use central finite difference to approximate the Jacobian matrix

$$\frac{\partial f_j}{\partial x_k}\left(x^{(i)}\right) \cong \frac{f_j(x^{(i)} + e_k h) - f_j\left(x^{(i)}\right)}{h} \qquad (4.5.0)$$

- **First type of BC (BC1)**

In the first type of Broyden's we will use central finite difference to approximate Jacobian matrix (i.e. $B_k \cong F_k'$), to improve the order of convergence and to avoid computing the Jacobian matrix.

By using Taylor's expansion and by similar explanation in [**4**]

$$F(x+h) = F(x) + F'(x)h + \frac{1}{2!}F''(\xi_1)h^2, \qquad (4.5.1)$$

$$F(x-h) = F(x) - F'(x)h + \frac{1}{2!}F''(\xi_2)h^2. \qquad (4.5.2)$$

Subtract (4.5.2) from (4.5.1), and by the mean value theorem of vector-valued function, we get

$$F(x+h) - F(x-h) = F'(x)2h + O(\|h\|)^2. \qquad (4.5.3)$$

Equation (4.5.3) indicates that the secant equation of Broyden's method replaced by

$$F(x+h) - F(x-h) = B_k 2h, \qquad (4.5.4)$$

from this, two assumptions Broyden's set on B_k:

I. Central secant equation $F(x+h) - F(x-h) = B_k 2h,$

II. No change condition $B_k q = B_{k-1} q, where \quad h^t q = 0.$

Now we need to determine x, $x - h$ and $x + h$.

Let $x = x^{(k)}$ and $2h = x^{(k)} - x^{(k-2)} = s^{(k)}$ then

$$x+h=\frac{3}{2}x^{(k)}-\frac{1}{2}x^{(k-2)} \quad and \quad x-h=\frac{1}{2}(x^{(k)}-x^{(k-2)})$$

$$y^{(k)}=F\left(\frac{3}{2}x^{(k)}-\frac{1}{2}x^{(k-2)}\right)-F\left(\frac{1}{2}(x^{(k)}-x^{(k-2)})\right)$$

By (I) and (II) we get

$$B_k = B_{k-1} + \frac{(y^{(k)} - B_{k-1}s^{(k)})(s^{(k)})^t}{\left\|s^{(k)}\right\|^2}, \; k = 2,...,m \qquad (4.5.5)$$

In (4.5.5) B_0 and B_1 are not defined, we can define both as Newton's method, like $B_0 = F_0'$ and $B_1 = F_1'$. We can use Sherman-Morrison formula [**64**] to make the above updating depend on the inverse of B_k,

$$B_{k+1}^{-1} = B_k^{-1} + \frac{\left(s_{k+1} - B_k^{-1}y_{k+1}\right)s_{k+1}^T B_i^{-1}}{s_{k+1}^T B_i^{-1} y_{k+1}} \qquad (4.5.6)$$

Second type of BC (BC2)

For this type, we use the central finite difference to approximate F_K^{-1} (*i.e*, $H_K \cong F_K^{-1}$) .One can easily show that the conditions (i) and (ii) in BC1 method become [**63**]:
Now, we use the same procedures in BC1 to determine x, $x - h$ *and* $x + h$ for BC2.
By (I) and (I) we obtain

$$H_k = H_{k-1} + \frac{\left(s_k - H_{k-1}s^{(k)}\right)}{\left\|y^{(k)}\right\|^2}\left(y^{(k)}\right)^t \qquad (4.5.7)$$

We can use H_0 and H_1 as Newton's method, $H_0 = F_0'^{-1}$ and $H_1 = F_1'^{-1}$

The above approximation of Jacobian matrix (BC1 and BC2) increases the accuracy and the new algorithms of Broyden's have the same arithmetic operations and mathematical computations of pure Newton's methods up to the first iteration, but its accuracy increases.

4.5.1. Application of the Broyden Classes Methods

To approximate the solution of mechanics problems define in section (1.2), we wrote an algorithm (4.5) for Broyden Class I method.

➢ **Algorithm (4.5): Solves nonlinear BVPs by using BC1 Method**

INPUT: - L – length of problem domain (b-a)
u, λ - Initial guess solution and parameter values
n - number of spaces
maxiter - maximum number of iterations
B - Initial approximation of Jacobian matrix

STEP 1: Do for 1st iteration, compute F and F' by

$$F_k = \mathrm{FC}(u_k, h, n, \lambda), \quad Fp_k = \mathrm{F}JJ(u_k, h, n, \lambda);$$

Update solution by $u_{k+1} = u_k - (Fp_k) \backslash F_k$

Compute F and F' for u_{k+1} and update solution

$$F_{k+1} = \mathrm{FC}(u_{k+1}, h, n, \lambda), \quad Fp_{k+1} = \mathrm{F}JJ(u_{k+1}, h, n, \lambda);$$

$$u_{new} = u_{k+1} - (Fp_{k+1}) \backslash F_{k+1}$$

STEP 2: for iteration > 2 do .step -3 up to step-8

STEP 3: compute s,u2. and u3

$$s = u_{k+1} - u_k \ , \ y = F_{k+1} - F_k$$

STEP 4: Call and compute F and y

$$F2 = \mathrm{FC}(y2, h, n, \lambda), F3 = \mathrm{FC}(y3, h, n, \lambda), \quad y = F2 - F3$$

STEP 5: update Jacobian matrix by eq.(4.5.6)

STEP 6: update solution by $u_{k+1} = u_k - B_k^{-1} F(u_k)$

STEP 7: If $\|u_{k+1} - u_k\|_2 \le Tol, Stop$ else $k = k + 1$

STEP 6: Display Numerical and graphical output

STOP

➢ **Numerical and Graphical output of problems using BC1 method**

The numerical method, BC1 is applied and tested on the following nonlinear BVPs that are define in section (1.2) and the numerical result and norm error are displayed by using MATLAB program.

Problem-1- consider Bratu 1-D nonlinear problem defined in section (1.2)

$$u''(x) + \lambda e^{u(x)} = 0 \qquad (4.5.1.1)$$

Subject to the boundary conditions:

$$u(0) = u(1) = 0 \qquad (4.5.1.2)$$

The Analytic solution for this problem given by, equation (1.2.3).

Using equations (4.5.1.1) and (3.3), we can create NLSAEs for Bratu problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1, 2, \ldots, N \qquad (4.5.1.3)$$

Where $f = -\lambda e^{u_j}$

To solve eq. (4.5.1.3) by using BC1 method and inputs of Broyden method I for the same problem give the following result:

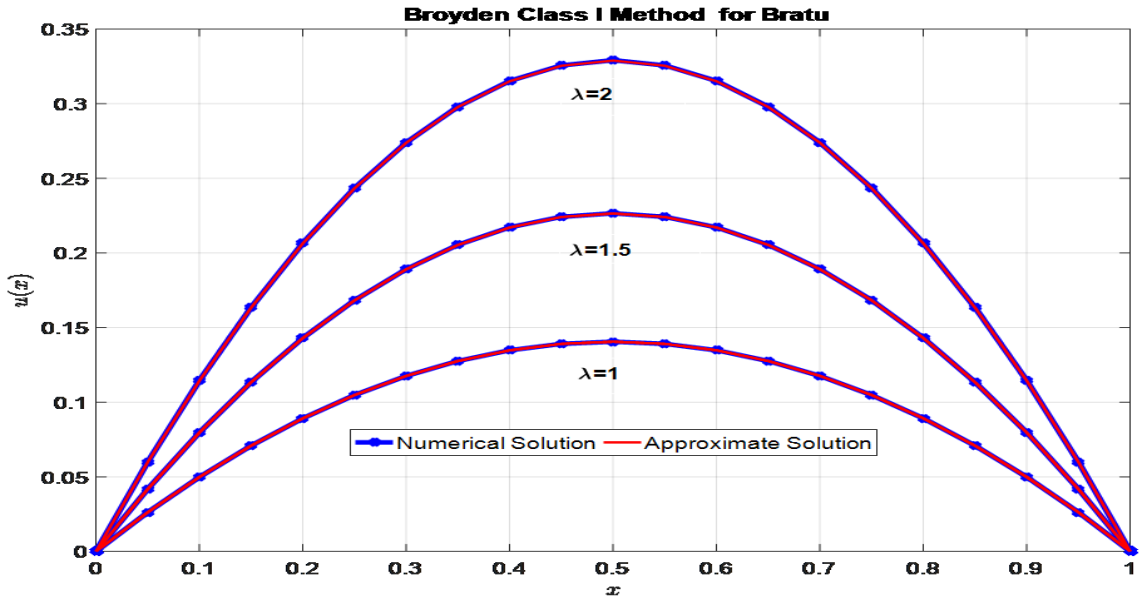

Figure 4.33 Comparison between the approximated solution and exact solution of Bratu problem.

Table 4.16:Broyden Class I(BC1) method Euclidian Norm of Error, number of iteration and CPU time for each value of parameter of Bratu problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
2	1	3.45e-4	2.144e -6	0.28125
2	1.5	9.0341e-8	7.319e-9	2.3125
2	2	2.895e-7	2.74e-8	2.453125

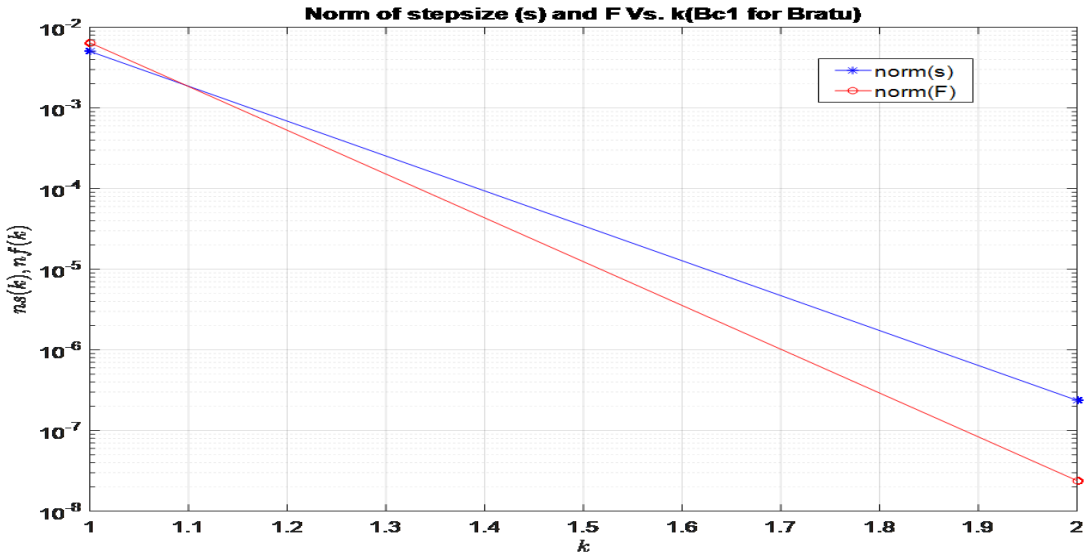

Figure 4.34 Broyden Class I Method Euclidian Norm of Errors versus number of iterations for Bratu problem.

Fig.4.33 cites a comparison between the BC1 method solution and exact solution. From this comparison, one can see a good agreement between the exact solution and the BC1 method (approximate) solution.

As shown in Table 4.16, Broyden Class 1 method converges for Bratu problem and terminates within 2 iterations for each of three different values of parameters (lambda). This shows that all computation takes around 1/10 times the number of iteration of Pure Broyden. Generally, the CPU-time taken for each parameter also indicates BC1 method takes more time for computation than Broyden method due to additional computation in BC1. Moreover, as shown in Figure 4.34, for Bratu problem the norm errors of function and Newton step are continuously decreasing, it is stable.

Problem-2- consider Troesch's 1-D nonlinear problem defined in section (1.2)

$$u'' = \lambda \sinh(\lambda u) \qquad (4.5.1.4)$$

Subject to the boundary conditions:

$$u(0) = 0 \ \ and \ \ u(1) = 1. \qquad (4.5.1.5)$$

The theoretical solution for this problem given by, equation (1.2.7).

Using equation (4.5.1.4) and (3.3), we can create NLSAEs for Troesch's problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1,2,...,N \qquad (4.5.1.6)$$

Where $f = \lambda \sinh(\lambda u(j))$

Equation (4.5.1.6), solve by using BC1 method and take similar input of Broyden I method for the same problem gives the following result:

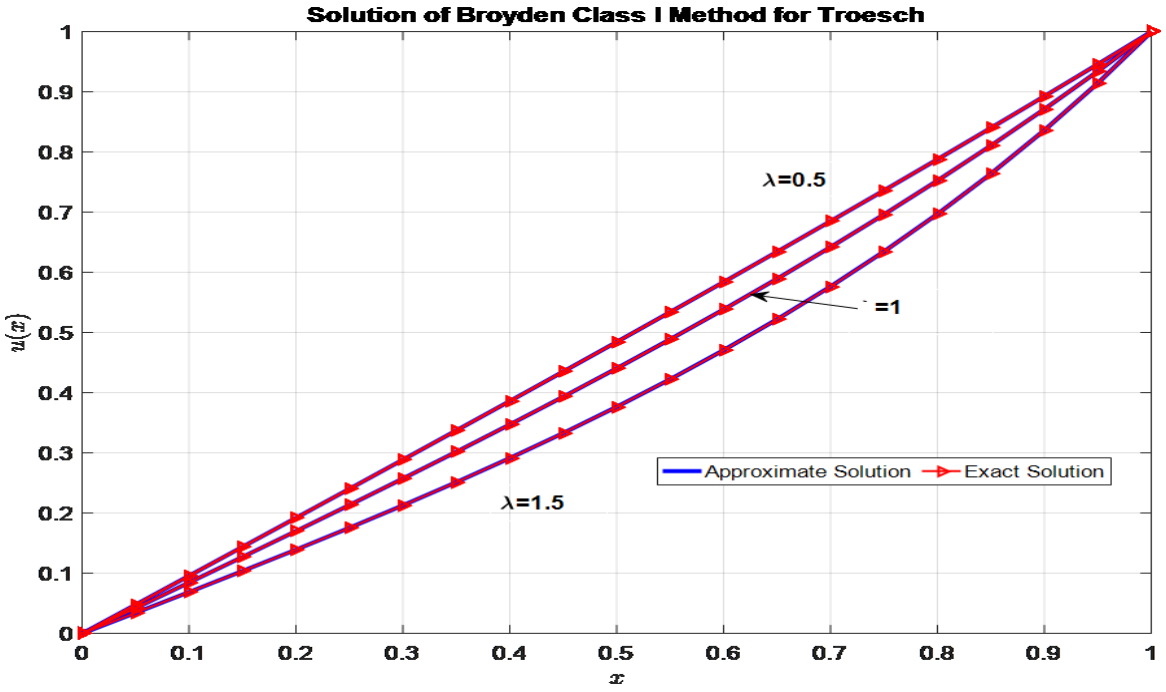

Figure 4.35 Comparison between the approximated solution and exact solution of Troesch problem.

Table 4.17:Broyden Class 1(BC1) method Euclidian Norm of error, number of iteration and CPU time for each value of parameter of Troesch problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
2	0.5	1.5e- 11	4.42e- 13	0. 03125
2	1	3e- 12	4.168e- 12	0.421875
2	1.5	3.007e- 9	1.824e- 9	0.5625

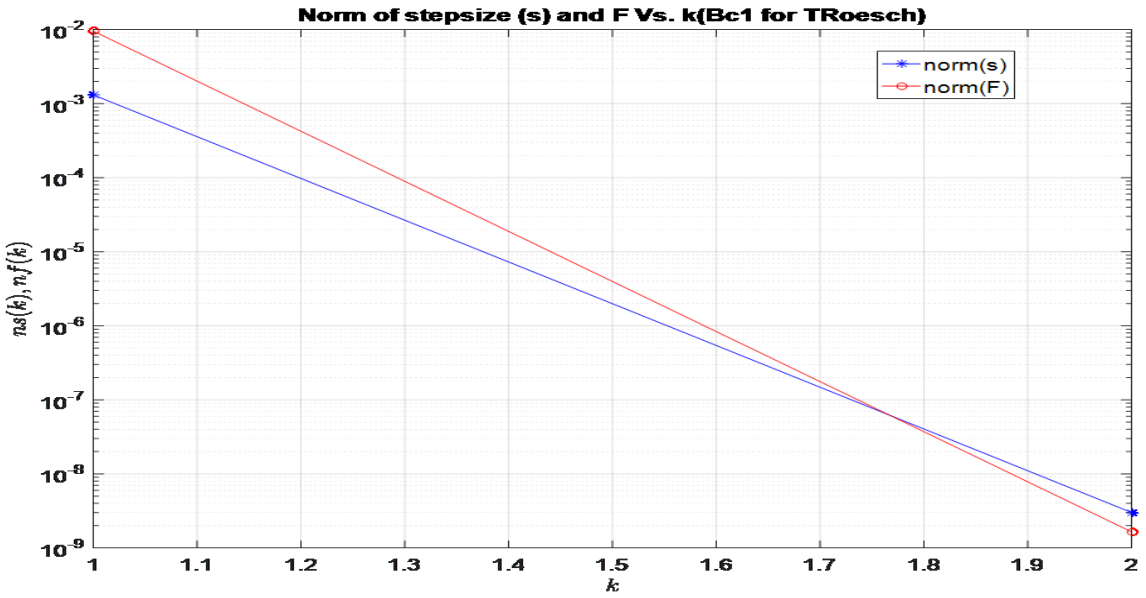

Figure 4.36 Broyden Class 1 Method Euclidian Norm of errors versus number of iterations for Troesch problem.

As shown in Figure 4.35, a comparison between the approximate solution obtained by BC1 method and exact solution. This shows a good agreement between the exact solution and the BC1 method (approximate) solution for Troesch's problem.

As shown in Table 4.17, BC1 methods converge for Troesch problem and terminate within 2 iterations for each of three different values of parameters (lambda). This shows that all computation takes around 1/8 times the number of iteration of Pure Broyden. Generally, the CPU-time taken for each parameter also indicates BC1 method takes more time for computation than Broyden method due to additional computation in BCI. Moreover, as shown in Figure 4.36 for Troesch problem, the norm errors of function and Newton step are continuously decreasing, and it is stable.

➢ **Algorithm (4.6): Solves nonlinear BVPs by using BC2 Method.**

INPUT: L – length of problem domain (b-a)/h

u, λ - Initial guess of problem solution

n , h - number of spaces and space between nodes

Maxiter- maximum number of iteration
Tol – magnitude of max error tolerance

STEP 1: for 1st iteration do

Call both function and jacobian function to compute $F(u)$ *and* $Fp(u)$

$$F = \text{FC}(u,h,n,\lambda), \quad Fp = \text{F}JJ(u,h,n,\lambda);$$

Compute the new value of u by:

$$u_{new} = u_{old} - (Fp) \backslash F$$

Call both function and jacobian function for u_{new} ,and update u_{new}

$$F_{new} = \text{FC}(u_{new},h,n,\lambda), \quad Fp_{new} = \text{F}JJ(u_{new},h,n,\lambda),$$

$$u_{new} = u_{new} - Fp_{new} \backslash F_{new}$$

STEP 2: for iteration 2 to maxiter do step-3 up to 8

STEP 3: Compute s ,u2 and u3

$$s_i = u_i - u_{i-2} \qquad u2 = 1.5u_i - 0.5u_{i-2} \text{ , } u3 = 0.5u_i - u_{i-2}$$

STEP 4 call F and compute y

$$F2 = \text{FC}(u2,h,n,\lambda), F3 = \text{FC}(u3,h,n,\lambda), \quad y = F2 - F3$$

STEP 5: update the Jacobi matrix by eq.(4.5.7)

STEP 6: Compute the new value of u by:

$$u_{i+1} = u_i - (Fp)_i \text{ '} \backslash F_i \text{ '};$$

STEP 7: check stopping criteria of iteration

STEP 8: Display graphical and Numerical output

➢ **Numerical and Graphical output of problems using BC2 method**

BC2 method is applied and tested on nonlinear BVPs that are define in section (1.2) and the numerical result and norm error are displayed by using MATLAB program.

Problem-1- consider Bratu 1-D nonlinear problem defined in section (1.2)

$$u''(x) + \lambda e^{u(x)} = 0 \qquad (4.5.1.7)$$

Subject to the dirchlet boundary conditions:

$$u(0) = u(1) = 0 \qquad (4.5.1.8)$$

The Analytic solution for this problem given by, equation (1.2.3)

Using equations (4.5.1.7) and (3.3), we can create NLSAEs for Bratu problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1,2,...,N \qquad (4.5.1.9)$$

Where $f = -\lambda e^{u_j}$

To solve eq. (4.5.1.9), by using Broyden Class 2 (BC2) method and using previous (BC1) method inputs for the same problem gives the following results:

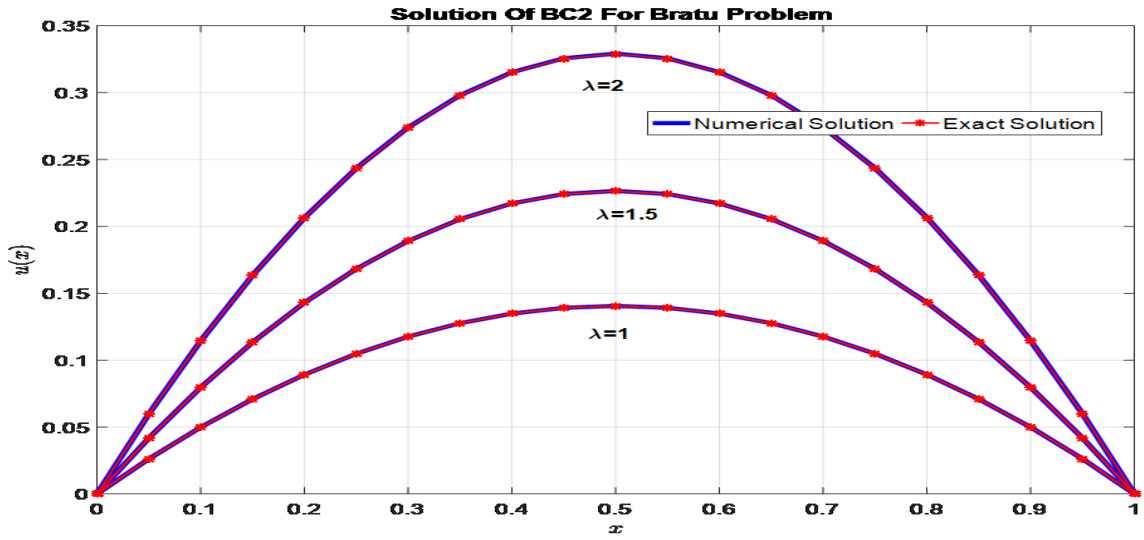

Figure 4.37 Comparison between the approximated solution and exact solution of Bratu problem.

Table 4.18: Broyden Class 2 (BC2) method Euclidian norm of error, number of iteration and CPU time for each value of parameter of Bratu problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
2	1	7.01e -6	4.2015e -9	0.140625
2	1.5	2.4956e -6	3.2505e -9	1.390625
2	2	1.802676e -5	2.366e -8	2.25

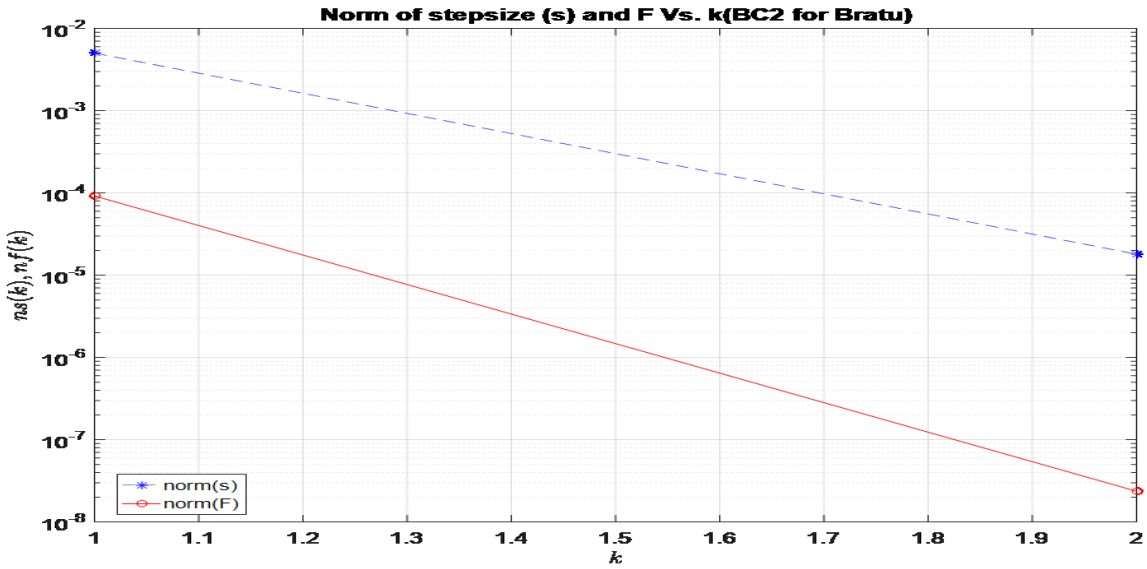

Figure 4.38 Broyden Class 2 Method Euclidian Norm of errors versus number of iterations for Bratu problem.

Analysis

Fig.4.37 cites a comparison between the approximate solution obtained by BC2 method and exact solution. From this comparison, one can see a good agreement between the exact solution and the BC2 method (approximate) solution, even much better than pure Broyden method.

As shown in Table 4.18, BC2 methods converge for Bratu problem and terminate within 2 iterations for each of three different values of parameters (lambda). This shows that all computation takes around 1/10 times the number of iteration of Pure Broyden. Generally, the CPU-time taken for each parameter also indicates BC2 method takes more time for computation than pure Broyden method due to additional computation in BC2. Moreover, as shown, in Figure 4.38, for Bratu problem, the norm errors of function and Newton step are continuously decreasing, and it is stable.

Problem-2- consider Troesch's 1-D nonlinear problem defined in section (1.2)

$$u'' = \lambda \sinh(\lambda u) \qquad (4.5.1.10)$$

Subject to the dirichlet boundary conditions:

$$u(0) = 0 \ \ and \ \ u(1) = 1. \qquad (4.5.1.11)$$

The theoretical solution for this problem given by, equation (1.2.7).

Using equation (4.5.1.10) and (3.3), we can create NLSAEs for Troeschs problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1,2,...,N$$

Where $f = \lambda \sinh(\lambda u(j))$

The above NLSAEs are solving by using BC2 method and similar inputs BC1 for the same problem gives the following output:

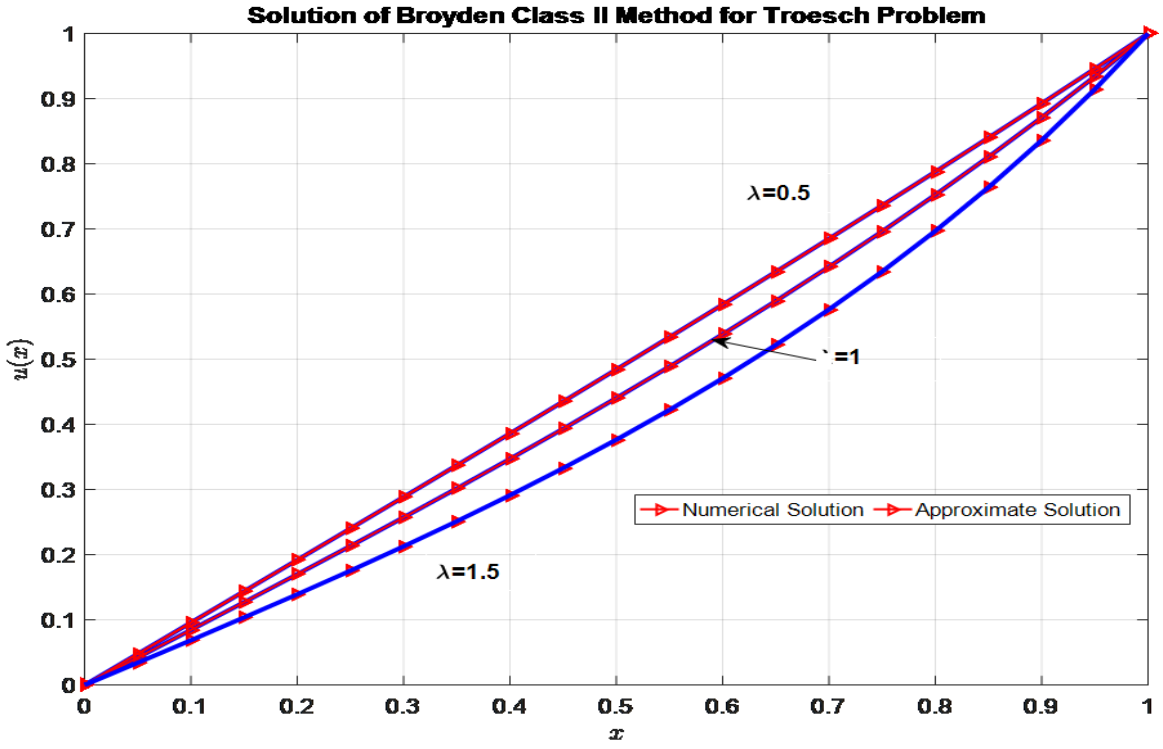

Figure 4.39 Comparison between the approximated solution and exact solution of Troesch problem.

Table 4.19: Broyden Class 2 (BC2) method Euclidian norm of error, number of iteration and CPU time for each value of Troesch's problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
2	0.5	3.38e -11	9.6e -13	0. 0375
2	1	1.238e-10	3.83e -12	1.6718750
2	1.5	4.4343e - 8	1.6456e -9	2.359375

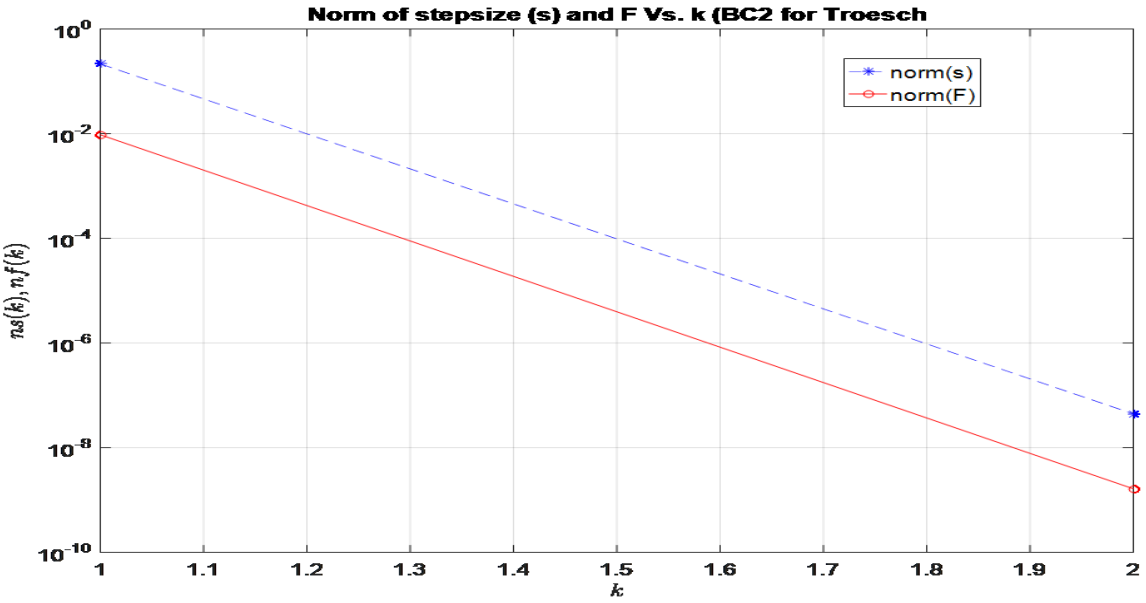

Figure 4.40 Broyden Class 2 Method Euclidian Norm of errors versus number of iterations for Troesch problem.

As shown in Figure 4.39, a comparison between BC2 method solution and exact solution. From this comparison, one can see a good agreement between the exact solution and the BC2 method (approximate) solution for Troesch's problem.

As shown in Table 4.19, Broyden Class 2 method converges for Troesch problem and terminates within 2 iterations for each of the three different values of parameters (lambda). This shows that all computation takes around 1/8 times number of iteration of Pure Broyden. Generally, the CPU-time taken for each parameter also indicates BC2 method takes a little more time for computation than Broyden method due to additional computation in BC2. Moreover, as shown, in Figure, 4.40 for Troesch problem, the norm errors of function and Newton step are continuously decreasing, and it is stable.

Modified Broyden Class I (MBC1)

In MBC1, we follow two steps of iteration formula to approximate solution $x^{(k+1)}$, which requires $(2Xn)$ functions evaluation and free of partial derivative by using:

$$x^{(k+1)} = x^{(k)} - B_k^{-1} F\left(x^{(k)}\right), \qquad (4.5.1.12)$$

$$x^{(k+1)} = x^{(k+1)} - B_k^{-1} F\left(x^{(k+1)}\right). \qquad (4.5.1.13)$$

Where, $B_K =$.

Modified Broyden Class II (MBC2)

We follow the same procedures of MBC1 of two steps as above and replacing $B_K^- =$, we obtain the following modification of type two of MBC2 to approximate solution $x^{(k+1)}$ as follow:

$$x^{(k+1)} = x^{(k)} - M_k F\left(x^{(k)}\right), \quad (4.5.1.14)$$

$$x^{(k+1)} = x^{(k+1)} - M_k F\left(x^{(k+1)}\right). \quad (4.5.1.15)$$

4.5.2. Application of the Modified Broydens Classes (MBC) Methods

To approximate the solution of mechanics problems defined in section (1.2), we wrote an algorithm (4.7) for Modified Broyden Class I method.

- **Algorithm (4.7): Solves nonlinear BVPs by using MBC1 Method.**

Algorithm (4.7) contains all the steps of Algorithm (4.5) by only changing step-6 by the following Predicator corrector steps:

$u_{k+1} = u_k - B_k^{-1}F(u_k)$ (Predicator) then $u^*_{k+1} = u_{k+1} - B_k^{-1}F(u_{k+1})$ (corrector)

- **Numerical and Graphical output by using MBC1 method.**

MBC1 is apply and tested on the following nonlinear BVPs that are define in section (1.2) and the numerical result and norm error are displayed by using MATLAB program.

Problem-1- consider Bratu 1-D nonlinear problem defined in section (1.2)

$$u''(x) + \lambda e^{u(x)} = 0 \quad (4.5.2.0)$$

subject to the dirichlet boundary conditions:

$$u(0) = u(1) = 0 \quad (4.5.2.1)$$

The Analytic solution for this problem given by, equation (1.2.3).

Using equation (4.5.2.0) and (3.3), we can create NLSAEs for Bratu problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1,2,...,N \quad (4.5.2.2)$$

Where $f = -\lambda e^{u_j}$

Equation (4.5.2.2), solve by using MBC1 method and similar inputs of BC1 for Bratu problem gives the following result:

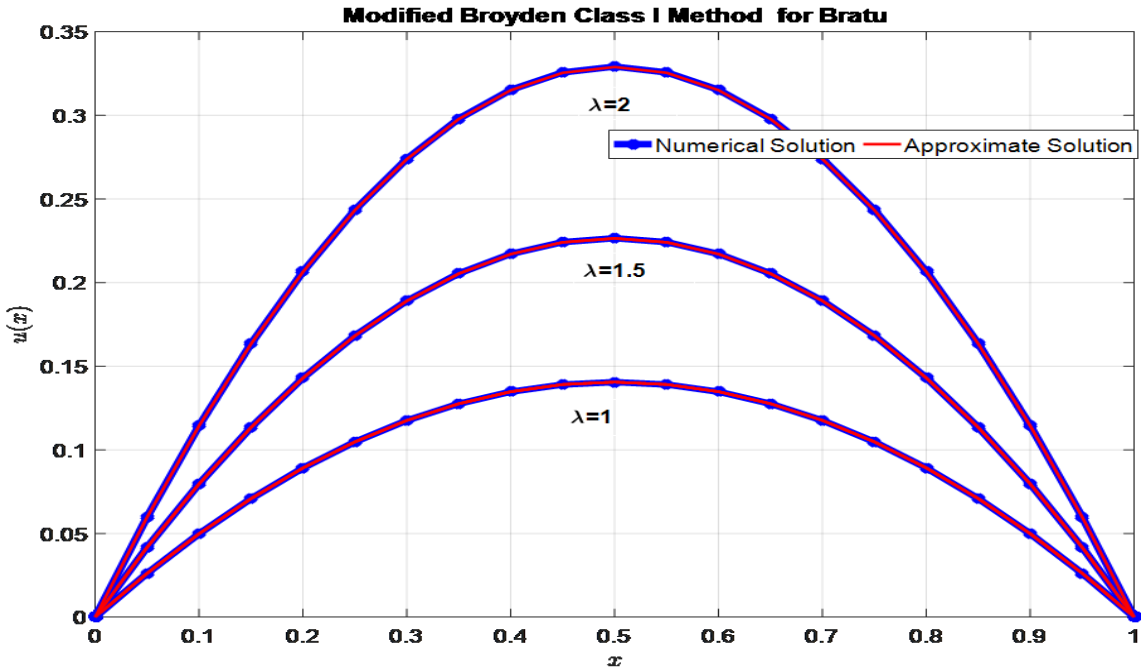

Figure 4.41 Comparison between the approximated solution and exact solution of Bratu problem.

Table 4.20: Modified Broyden Class 1 (MBC1) method Euclidian norm of error, number of iteration and CPU time for each value of parameter of Bratu problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
2	1	1.34e-4	1.063e-6	0.203125
2	1.5	8.0501e-8	6.09e-9	1.578125
2	2	2.8647e-7	2.3633e-8	2.125

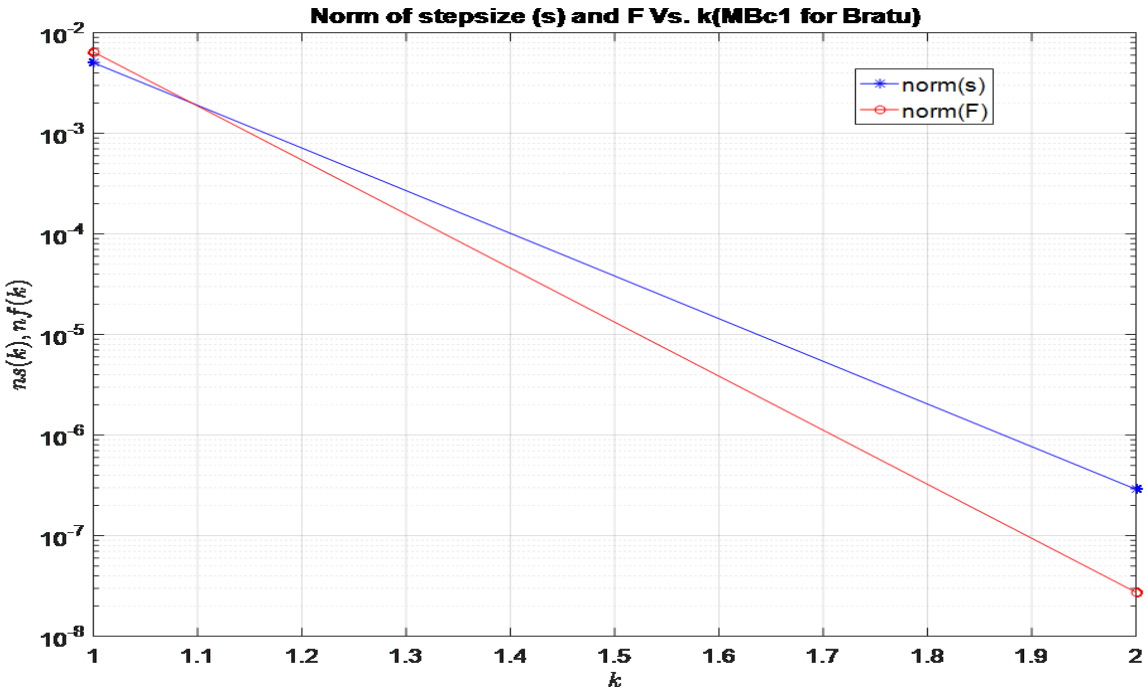

Figure 4.42 Modified Broyden Class 1 Method Euclidian Norm of errors versus number of iterations for Bratu problem.

Fig.4.41 cites a comparison between the approximate solution obtained by MBC1 method and exact solution. This shows a good agreement between the exact solution and the MBC1 method (approximate) solution.

As shown in Table 4.20 MBC1, methods converge for Troesch problem and terminate within 2 iterations for each of the three different values of parameter's (lambda). This shows that all computation takes above 1/8 times the number of iteration of Pure Broyden and MBC1 have the same number of iteration with BC1. However, the Euclidean norm error of Newton step (du) and F of MBC1 are lesser than both pure Broyden and BC1, which is the effect of predicator and corrector steps. Moreover as shown in Figure 4.42 the norm errors of function and Newton step are continuously decreasing.

Problem-2- consider Troesch's 1-D nonlinear problem defined in section (1.2)

$$u'' = \lambda \sinh(\lambda u) \qquad (4.5.2.3)$$

Subject to the dirichlet boundary conditions:

$$u(0) = 0 \ \ and \ \ u(1) = 1. \qquad (4.5.2.4)$$

The theoretical solution for this problem given by, equation (1.2.7).

Using equation (4.5.2.3) and (3.3), we can create NLSAEs for Troesch's problem.

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1},\ for\ \ j = 1,2,...,N \qquad (4.5.2.5)$$

Where $f = \lambda \sinh(\lambda u(j))$

The NLSAEs solve by using MBC1 method and similar input of BC2 for Troesch's problem gives the following output:

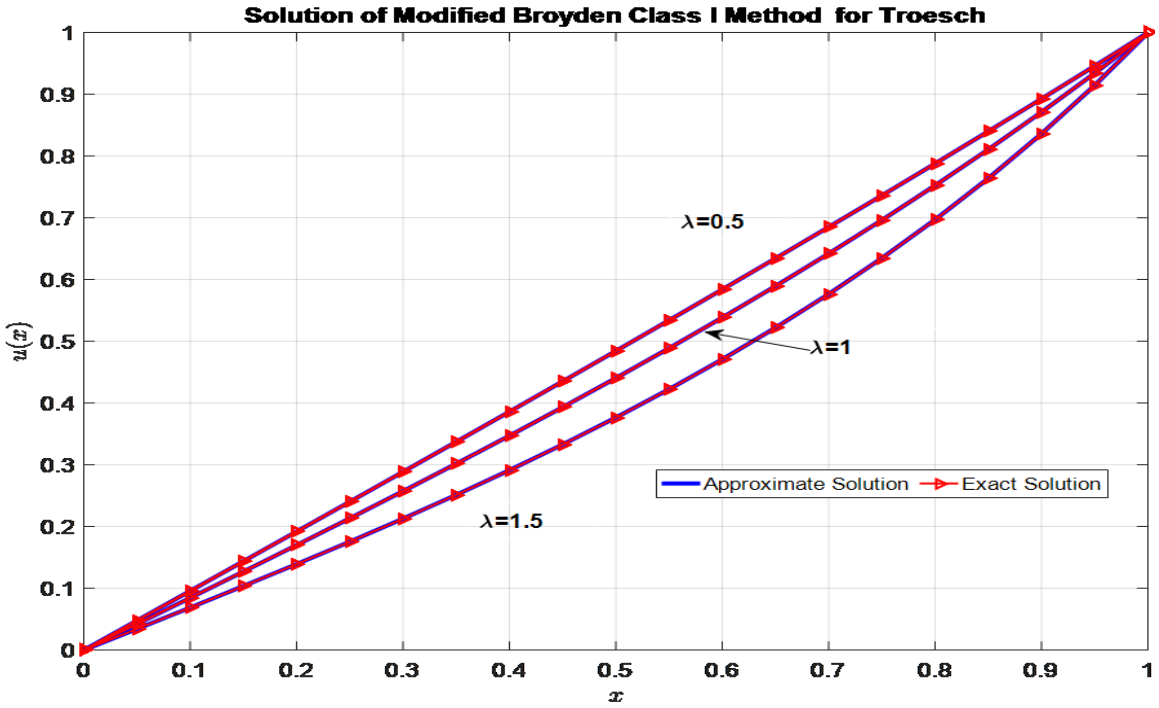

Figure 4.43 Comparison between the approximated solution and exact solution of Troesch problem.

Table 4.21: Modified Broyden Class 1 (MBC1) method Euclidian Norm of error, number of iteration and CPU time for each value of parameter of Troesch problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
2	0.5	1.5e- 11	4.4222e- 13	0. 03125
2	1	3e- 12	4.168e- 12	0.328125
2	1.5	1.984e- 9	1.824e- 9	0.50

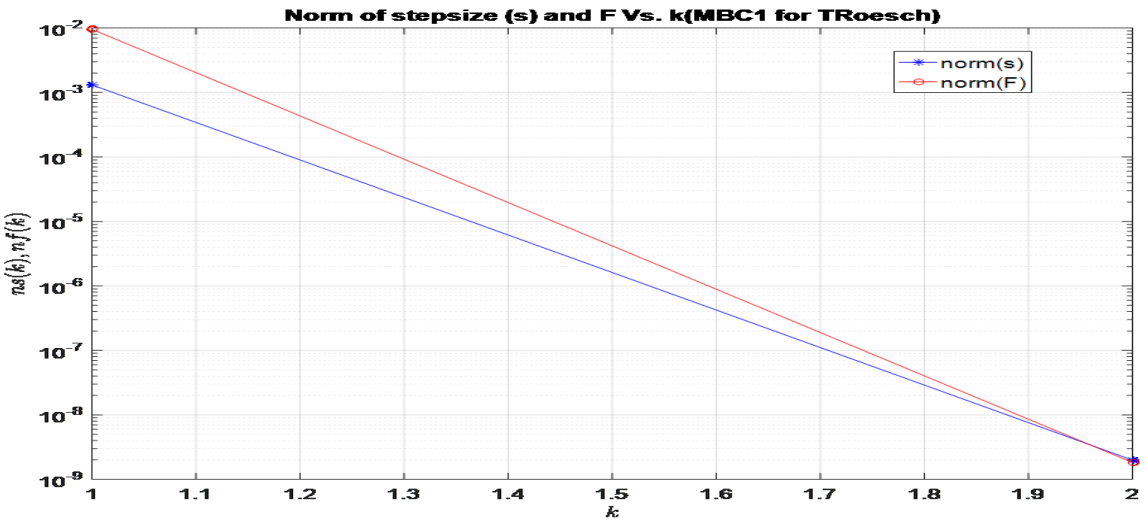

Figure 4.44 Modified Broyden Class 1 Method Euclidian Norm of errors versus number of iterations for Troesch problem.

As shown in Fig.4.43, a comparison between the approximate solution obtained by MBC1 method and exact solution. This shows a good agreement between the exact solution and the MBC1 method (approximate) solution for Troesch's problem.

As shown in Table 4.21, a modified Broyden Class 1(MBC1) method converges for Troesch problem and terminates within 2 iterations for each of the three different values of parameter's (lambda). MBC1 have the same number of iteration with BC1, however the Euclidean norm error of Newton step (du) of MBC1 is lesser than both pure Broyden and BC1 like previous Bratu problem. Moreover as shown in Figure 4.44 for Troesch problem, the norm errors of function and Newton step are continuously decreasing.

Algorithm (4.8): Solves nonlinear second order BVPs by using (MBC2) Method.

Algorithm (4.8) contains all the steps of Algorithm (4.6) by only changing step-6 by the following Predicator corrector step:

$u_{k+1} = u_k - HF(u_k)$ (Predicator) then $u^*_{k+1} = u_{k+1} - HF(u_{k+1})$ (corrector)

Based on Algorithm (4.8) we wrote a MATLAB program, which we will use in solving problems 1 and 2.

- **Numerical and Graphical output by using Modified BC2(MBC2) method**

Problem-1- consider Bratu 1-D nonlinear problem defined in section (1.2)

$$u''(x) + \lambda e^{u(x)} = 0 \qquad (4.5.2.6)$$

Subject to the dirichlet boundary conditions:

$$u(0) = u(1) = 0 \qquad (4.5.2.7)$$

The Analytic solution for this problem given by, equation (1.2.3).

Using equation (4.5.2.6) and (3.3), we can create NLSAEs for Bratu problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1,2,...,N \qquad (4.5.2.8)$$

Where $f = -\lambda e^{u_j}$

Equation (4.5.2.8) solves by using MBC2 method and similar input of BC2 for Bratu problem gives the following output:

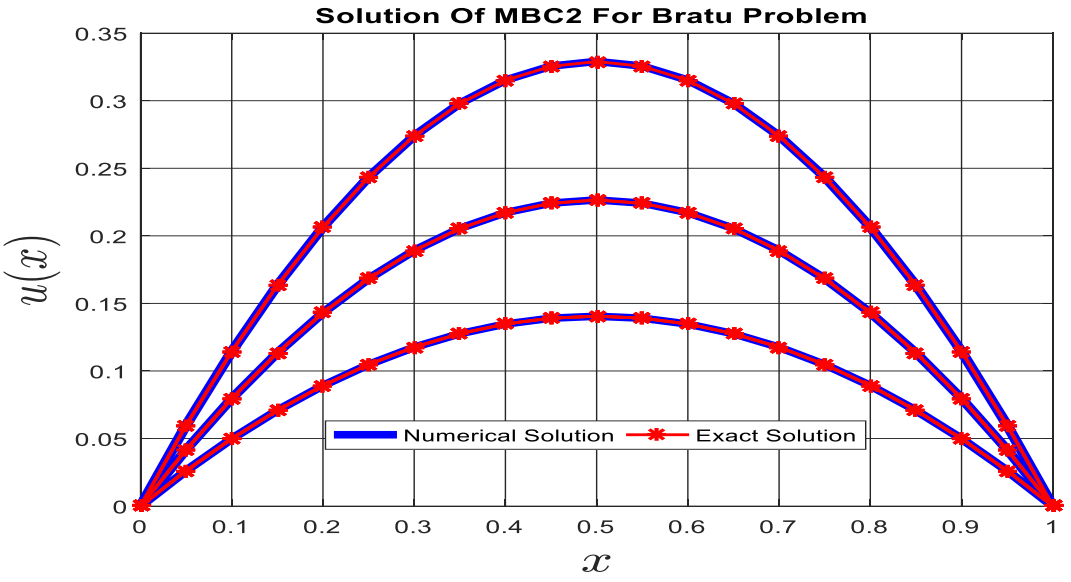

Figure 4.45 Comparison between the approximated solution and exact solution of Bratu problem.

Table 4.22: Modified Broyden Class 2 (MBC2) method Euclidian norm of error, number of iteration and CPU time for each value of parameter of Bratu problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
2	1	2.167e -8	3.79e -9	0.0375
2	1.5	2.374e -6	9.2e -8	0.46875
2	2	1.801e -5	6.24e -7	0.71875

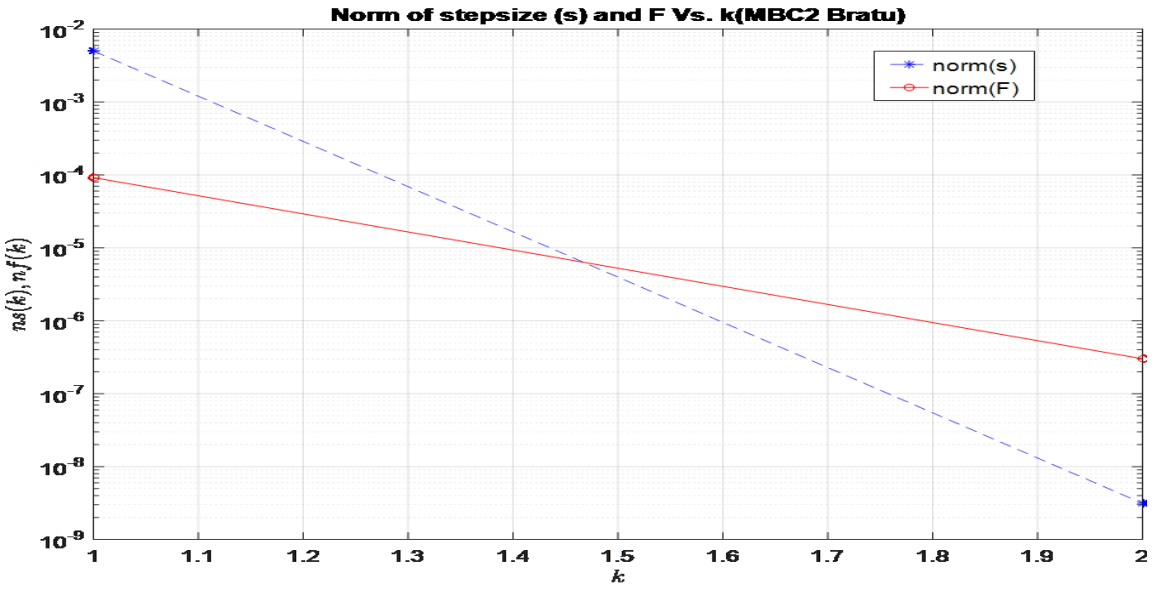

Figure 4.46 Modified Broyden Class 2 Method Euclidian Norm of errors versus number of iterations for Bratu problem.

Analysis

Fig.4.45 cites a comparison between the solution obtained by MBC2 method and exact solution. This shows a good agreement between the exact solution and the MBC2 method (approximate) solution, even much better than pure Broyden method and BC2.

As shown in Table 4.22 MBC2, method converges for Bratu problem and terminates within 2 iterations for each of the three different values of parameter (λ). The computation takes the same number of iteration with BC2, however, the Euclidean norm error of Newton step (du=F\DF) of MBC2 is less than both pure Broyden and BC2 methods like previous problems. Moreover as shown in Figure 4.46 for Bratu problem, the norm errors of function and Newton step are continuously decreasing.

Problem-2- consider Troesch's 1-D nonlinear problem defined in section (1.2)

$$u'' = \lambda \sinh(\lambda u) \qquad (4.5.2.9)$$

Subject to the dirichlet boundary conditions:

$$u(0) = 0 \ \ and \ \ u(1) = 1. \qquad (4.5.2.10)$$

The theoretical solution for this problem given by, equation (1.2.7).

Using equation (4.5.2.9) and (3.3), we can create NLSAEs for Troesch problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1,2,...,N \qquad (4.5.2.11)$$

Where $f = \lambda \sinh(\lambda u(j))$

The above NLSAEs solve by using MBC2 method and with input similar to BC2 of Troesch's problem by a MATLAB program gives the following output:

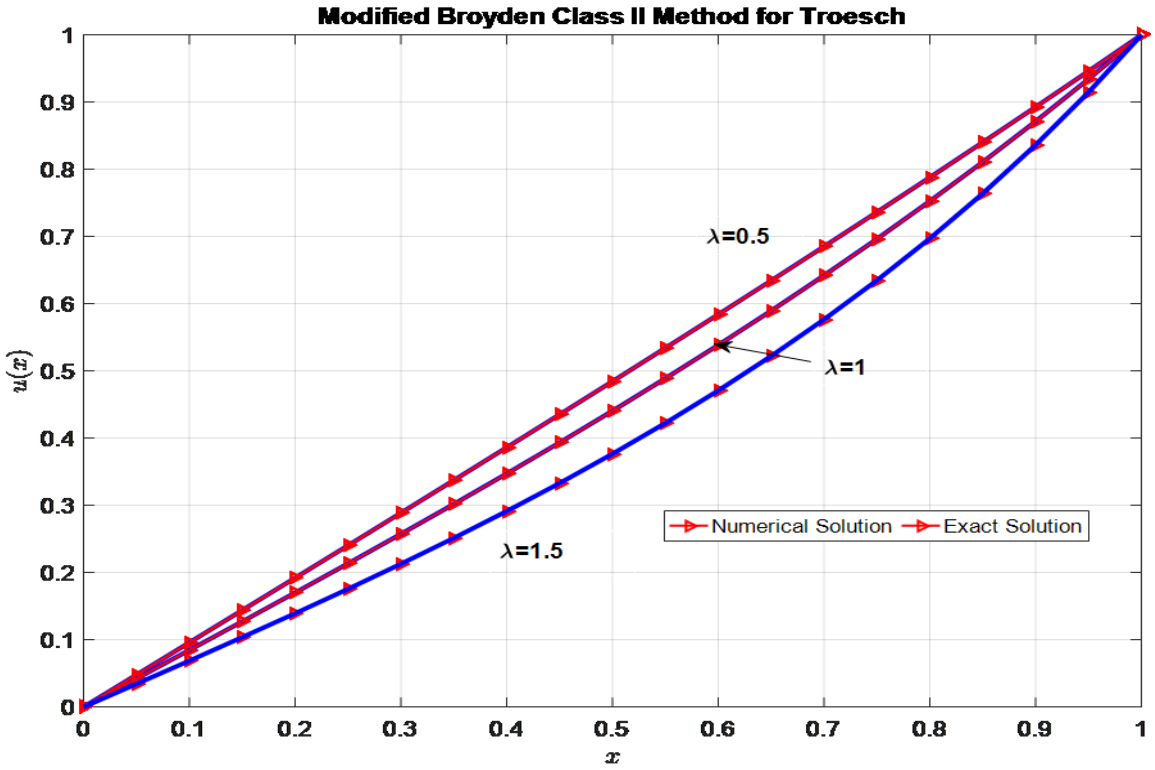

Figure 4.47 Comparison between the approximated solution and exact solution of Troesch problem.

Table 4.23: Modified Broyden Class 2 (MBC2) method Euclidian norm of error, number of iteration and CPU time for each value of parameter of Troesch's problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
2	0.5	5.6e- 13	6e-14	0. 421875
2	1	1.2e- 13	4.0279e- 15	1.8125
2	1.5	2.659e- 10	1.2116e- 11	1.984375

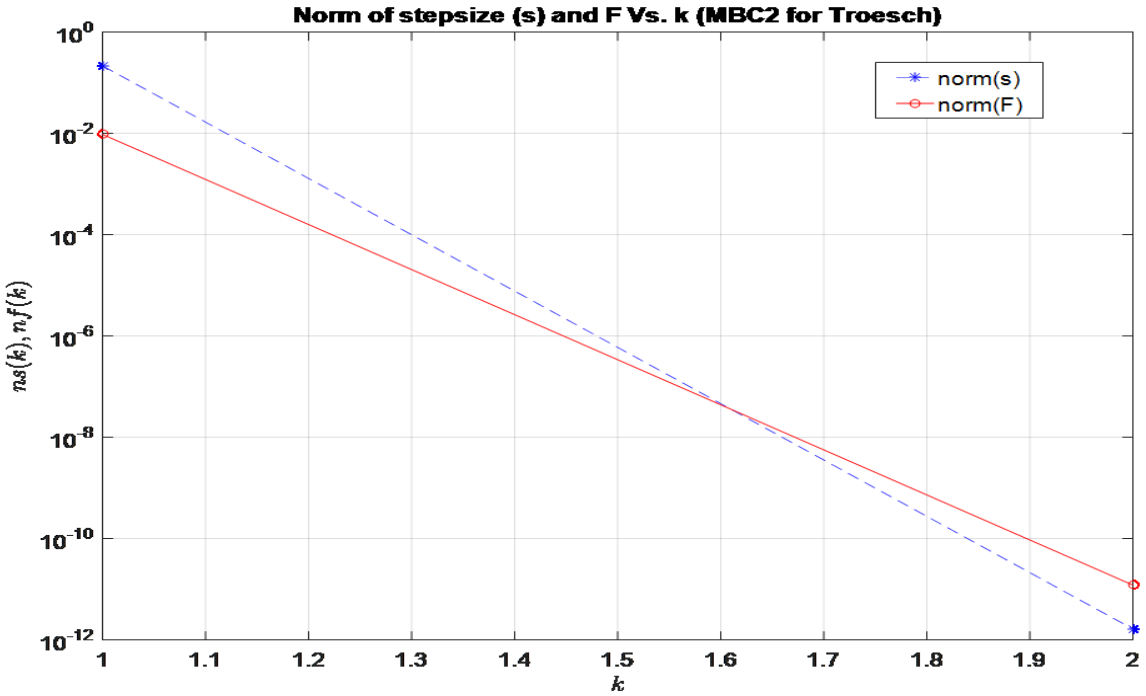

Figure 4.48 Modified Broyden Class 2 Method Euclidian Norm of errors versus number of iterations for Troesch problem.

Analysis

As shown in Fig.4.47, a comparison between the approximate solution obtained by MBC2 method and exact solution. From this comparison, one can see a good agreement between the exact solution and the MBC2 method (approximate) solution for Troesch's problem.

As shown in Table 4.23, MBC2 method converges for Troesch problem and terminates within 2 iterations for each of the three different values of parameters. MBC2 have the same number of iteration with BC2, however, the Euclidean norm error of Newton step (du) of MBC2 is much less than both pure Broyden and BC2 methods like previous problems. Moreover as shown in Figure 4.48, for Troesch problem, the norm errors of function and Newton step are continuously decreasing

4.6. Trapezoidal, Simpson and Mid-point Method (TSMM)

The classical Newton method is one of the commonly used iterative schemes for the solution of equation (1.1.7) but confronted with many demerits, like the need to calculate and invert the Jacobian matrix $J(x)$ at each iteration.

The main reason, we use of the Broyden-like methods by **Mamat** *et al*. [**41**], by **Waziri & Mohammad** [**65**] and the need to avert the shortcomings of the classical Newton methods especially on large scale problems are presented in this paper.

4.6.1. Derivation process of TSMM

For Single, variable Newton's method originates from the Taylor's series expansion of the function $f(x)$ about the point :

$$f(x)=f(x_1)+(x-x_1)f'(x_1)+\frac{1}{2!}(x-x_1)^2 f''(x_1)+... \quad (4.6.1.0)$$

Where f, and its first and second derivatives, f' and f'' are calculated at x_1. For multiple variable function F from the above equation, it is obvious that

$$F(x)=F(x_k)+\int_{x_k}^{x} F'(t)dt \quad (4.6.1.1)$$

Approximating the integral in Equation (4.6.1.1) by the weighted combination of Trapezoidal, Simpson and Midpoint quadrature rules yields:

$$x_{k+1}=x_k-24[5F'(x_k)+14F'\left(\frac{x_k+x_{k+1}}{2}\right)+5F'(x_{k+1})]^{-1}F(x_k) \quad (4.6.1.2)$$

Replacing $F'(x_k), F'(x_{k+1})$ by $B(x_k), B(x_{k+1})$ respectively like in [**66**] .

$$x_{k+1}=x_k-24[5B(x_k)+14B\left(\frac{x_k+x_{k+1}}{2}\right)+5B(x_{k+1})]^{-1}F(x_k) \quad (4.6.1.3)$$

In order to avoid the implicit nature of this equation, we can do

$$x_{k+1}=x_k-24[5B(x_k)+14B\left(\frac{x_k+m_k}{2}\right)+5B(m_{k+1})]^{-1}F(x_k) \quad (4.6.1.4)$$

Where m_k is given as

$$m_k=x_k-B_k^{-1}F(x_k) \quad (4.6.1.5)$$

Therefore, we have

$$x_{k+1}=x_k-24[5B(x_k)+14B(z_k)+5B(m_k)]^{-1}F(x_k) \quad (4.6.1.6)$$

for $z_k=\frac{x_k+m_k}{2}$ Suppose we set $B_k=[5B(x_k)+14B(z_k)+5B(m_k)]$, then we have

$$x_{k+1} = x_k - 24B_k^{-1}F(x_k) \qquad (4.6.1.7)$$

Hence, with the formulation above and selecting Predictor-Corrector of Broyden method we have the following two-step iterative scheme of the Trapezoidal-Simpson-Midpoint method for solving system of equations (1.1.7). For a given x_0 using initial matrix $B_0 = I$ and compute the approximates solution x_{k+1} by the iterative schemes

$$\begin{aligned} m_k &= x_k - B_k^{-1}F(x_k) \\ x_{k+1} &= x_k - 24[5B(x_k) + 14B(z_k) + 5B(m_k)]^{-1}F(x_k) \end{aligned} \qquad (4.6.1.8)$$

Where, $z_k = \dfrac{x_k + m_k}{2}$, $k = 0,1,...$

We are interested to apply the weighted combination of the Trapezoidal, Simpson and Midpoint quadrature rules to our model BVPs.

4.6.1.1. Application of the Trapezoidal, Simpson and Mid-point (TSMM)

General Algorithm of TSMM method, to approximate the solution of our mechanics problems defined in section (1.2).

Algorithm (4.9): Solves nonlinear BVPs by using TSMM Method.

INPUT: L – length of problem domain (b-a)

u, λ - Initial guess solution, parameter

n , h - number of spaces and space between nodes

Maxiter - maximum number of iteration

Tol – magnitude of max error tolerance

1. Compute $F(u_k)$ if $\|F(u_k)\| < tol$ and k > maxiter are satisfied stop, Else go to step 3,
2. Compute m_k from Equation (4.6.1.5).
3. Compute $B(m_k)$ using.

$$B(m_k) = B(u_k) + \frac{(y_k - B(x_k)s_k)s_k^T}{s_k^T s_k}$$

Where $y_k = F(m_k) - F(u_k)$ and $s_k = m_k - u_k$

4. Compute $B(z_k)$ using

$$B(z_k) = B(u_k) + \frac{(g_k - B(x_k)b_k)b_k^T}{b_k^T b_k}$$

Where $z_k = \frac{m_k + u_k}{2}, g_k = F(z_k) - F(u_k)$ & $b_k = z_k - u_k$

5. Compute u_{k+1} from Equation

$$u_{k+1} = u_k - 24\left((5B + 14Bz + 5Bm) \setminus F_k\right)$$

6. Set k = k + 1 and go to step 2.

$$B_{k+1} = B_k + \frac{\left(y_k - B(u_k)s_k\right)s_k^T}{s_k^T s_k}$$

Where $y_k = F(u_{k+1}) - F(u_k)$

$$s_k = u_{k+1} - u_k$$

Stop

Based on Algorithm (4.9), we wrote a MATLAB program, that solve problems defined in section (1.2).

- **Numerical and Graphical output by using TSMM**

The numerical method, TSMM applied and tested on the following nonlinear BVPs that are define in section (1.2).

Problem-1- consider Bratu 1-D nonlinear problem defined in section (1.2)

$$u''(x) + \lambda e^{u(x)} = 0 \qquad (4.6.1.1.0)$$

Subject to the dirichlet boundary conditions:

$$u(0) = u(1) = 0 \qquad (4.6.1.1.1)$$

The Analytic solution for this problem given by, equation (1.2.3).

Using equation (4.6.1.1.0) and (3.3), we can create NLSAEs for Bratu problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1},\ for\ \ j = 1, 2, ..., N \qquad (4.6.1.1.2)$$

Where $f = -\lambda e^{u_j}$

The above NLSAEs solve by using TSMM method and similar inputs of Broyden method for Bratu problem gives:

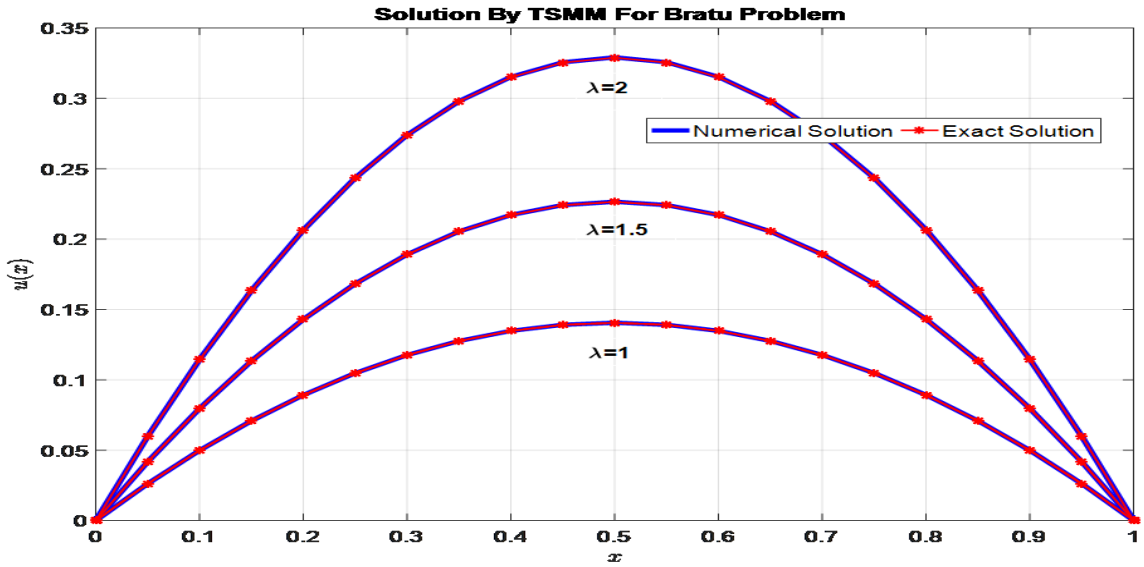

Figure 4.49 Comparison between the approximated solution and exact solution of Bratu problem.

Table 4.24: TSMM Euclidian norm of error, number of iteration and CPU time for each value of parameter of Bratu problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
25	1	2.4116e -11	3.6164e -9	0.078125
12	1.5	2.8964e -11	4.6286e -9	1.40625
21	2	5.112e -10	1.9451e -9	1.953125

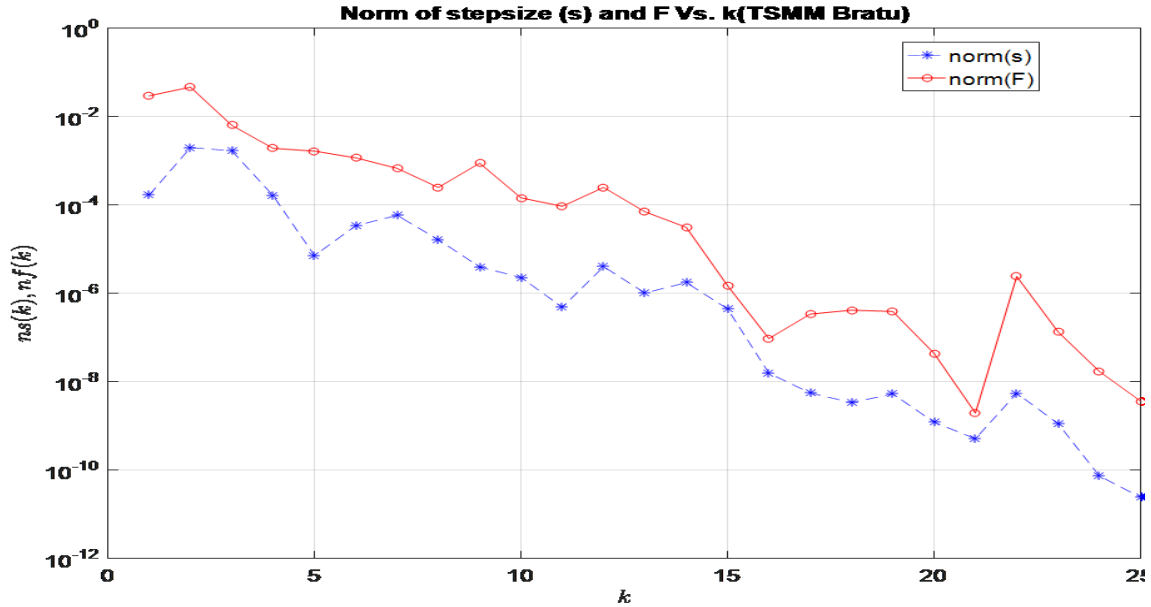

Figure 4.50 TSMM Euclidian Norm of errors versus number of iterations for Bratu problem.

Fig.4.49 cites a comparison between the approximate solution obtained by TSM method and exact solution. From this comparison, we can see a good agreement between the exact solution and the TSMM (approximate) solution.

As shown in Table 4.24, TSM method converges for Bratu problem and terminate within 25,12 and 21 iterations for three different values of parameters (lambda). This shows that the number of iteration is less than Broyden method II. The Euclidean norm error of Newton step (du=F\DF) of TSM method is lesser than Broyden method II. Moreover as shown in Figure 4.50, TSM method converges, but it is not stable at different iteration point, i.e. Euclidian norm of error increase rather decrease on different point due to the inverse of the approximation Jacobian matrix.

Problem-2- consider Troesch's 1-D nonlinear problem defined in section (1.2)

$$u'' = \lambda \sinh(\lambda u) \qquad (4.6.1.1.3)$$

Subject to the boundary conditions:

$$u(0) = 0 \ \ and \ \ u(1) = 1. \qquad (4.6.1.1.4)$$

The theoretical solution for this problem given by, equation (1.2.7).

Using equation (4.6.1.1.3) and (3.3), we can create NLSAEs for Troeschs problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1, 2, ..., N \qquad (4.6.1.1.5)$$

Where $f = \lambda \sinh(\lambda u(j))$

Equation (4.6.1.1.5) solve by using TSMM and similar inputs of BC1 for Troesch problem gives

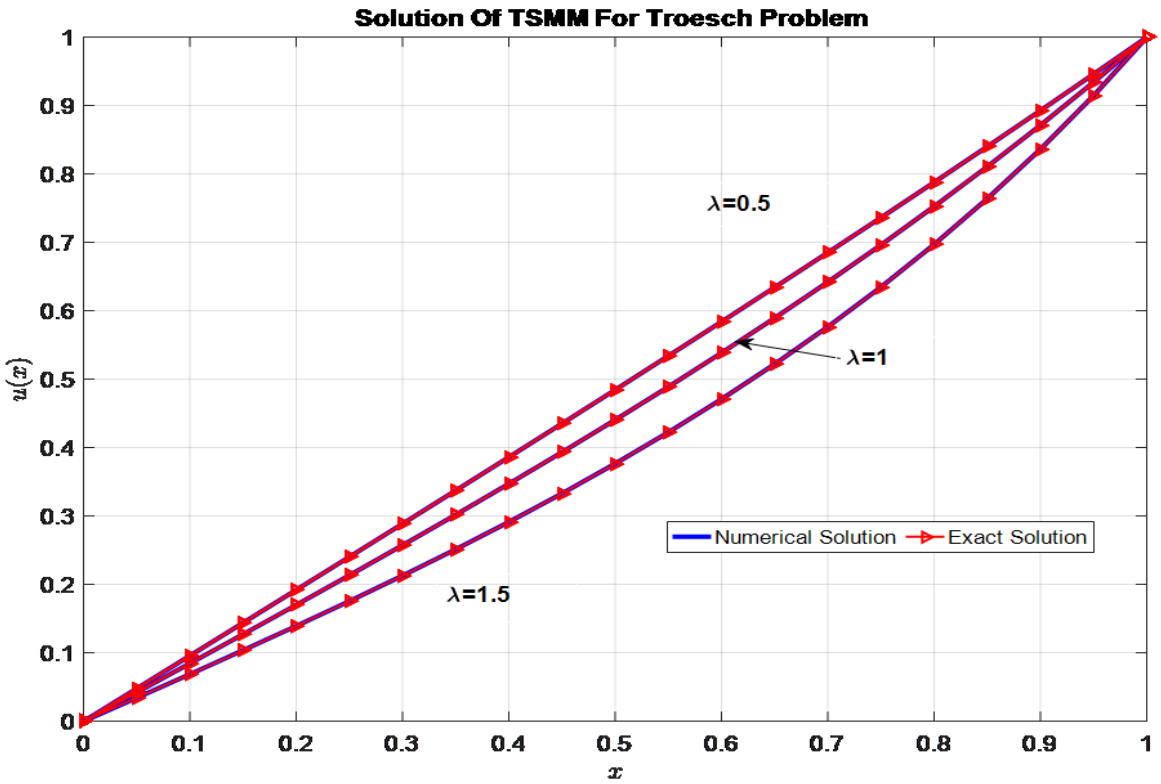

Figure 4.51Comparison between the approximated solution and exact solution of Troesch problem.

Table 4.25: TSMM Euclidian norm of error, number of iteration and CPU time for each value of parameter of Troesch's problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of fuction (F)	CPU time
46	0.5	1.38e -10	8.998e -9	0. 03125
20	1	1.0168e -10	8.54e -9	0.515625
16	1.5	2.4669e - 10	4.773e -9	0.734375

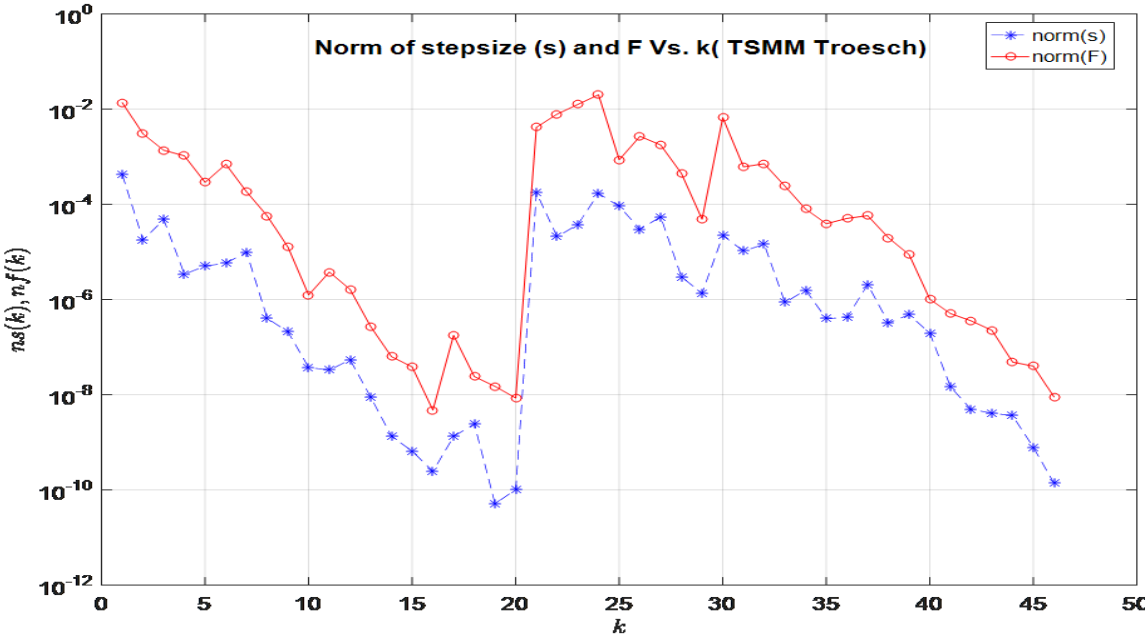

Figure 4.52 TSMM Euclidian Norm of errors versus number of iterations for Troesch problem.

As shown in Fig.4.51, a comparison between the approximate solution obtained by TSMM and exact solution. This shows a good agreement between the exact solution and the TSMM (approximate) solution for Troesch's problem.

As shown in Table 4.25, TSMM method converges for Troesch problem and terminates within 46, 20 and 16 iterations for three different values of parameters (lambda). This shows that the number of iteration is almost near to Broyden method II except the first parameter computation, however the Euclidean norm error of Newton step (du) of TSMM is lesser than Broyden method II which enables to decrease wrong path to solution of Trapezoidal Broyden method. Moreover as shown in Figure 4.51, TSMM converges, but its error is not stable, at many iteration point error increases rather than decreasing, i.e. Euclidian norm of error increase rather decrease on that point due to the inverse of the approximation Jacobian matrix.

CHAPTER FIVE

5. Homotopy Method (HM) to Solve BVPs

Homotopy methods are the family of continuation methods. They represent a way to find a solution to a problem by constructing a new problem simpler than the original one, and then gradually deforming this simpler problem into the original one by keeping track of the series of zeros that connect the solution of the simpler problem to that of the original, harder one.

In addition, to solve nonlinear system of equations by HM, its result can be used as a good initial approximation to both Newton's and Broyden Methods see [**20**].

Description of problem and the methods

Consider the following nonlinear algebraic or transcendental system of equations

$$F(x)=0, \qquad F=(f_1,f_2,...,f_n), \tag{5.0}$$

Recalling from section (3.1) that the Newton Method for solving (5.0) is formulated as follows

$$x^{(k+1)}=x^{(k)}-\left[DF\left(x^{k}\right)\right]^{-1}F\left(x^{(k)}\right), \quad k=0,1,2,..., \tag{5.1}$$

Where DF is the Jacobian matrix of F and $x^{(0)}$ is an initial guess of x^{*}. For more details, see [**67**] . As a second choice for solving (5.0), the Homotopy method for the system of nonlinear equations is recalled [**68**].

Consider the Homotopy function

$$H:\ 0,1\times\Re\ \rightarrow\Re \quad .... \tag{5.2}$$

is defined by

$$H(q,x)\ =\ F(x)\ +(q-1)\left(F(x)-F(x^{(0)})\right) \tag{5.3}$$

Here, $x^{(0)}$ is an initial guess of the solution x^{*}; q is called Homotopy parameter, obviously, at $q=0$ *and* $q=1$, the HM can be define as:

$$H\left(0,x\right)=F\left(x\right)-\ F\ \left(x^{(0)}\right), \qquad H\left(1,x\right)=F\ \left(x\right). \tag{5.4}$$

If q increases from 0 to 1 then the function $\mathrm{H}(\mathrm{q},\mathrm{x})$ varies continuously from $\mathrm{F}(\mathrm{x})\text{-}\mathrm{F}(\mathrm{x}^{(0)})$ to $F(x)$. The function H, with respect parameter q, provides us a family of function that can lead from the known value $x^{(0)}$ to unknown (solution) x^* The solution x^* *of* $F(x) = 0$ can be obtain by solving the following system of equations

$$\phi'(q) = -\left[J(\phi(q))\right]^{-1} F(\phi(0)), \qquad 0 \le q \le 1, \qquad (5.5)$$

With the initial condition $\phi(0) = x^{(0)}$ and $J(\phi(q))$ is jacobian matrix of H with respect to x. This method will be referred as Homotopy Method (HM) [**67**].

If q increases from 0 to 1 then the function $H(q,x)$ varies continuously from $F(x^{(0)})$ to $F(x)$. The function H with respect to parameter q, lead from the known value $x^{(0)}$ to solution x^* [**20**].

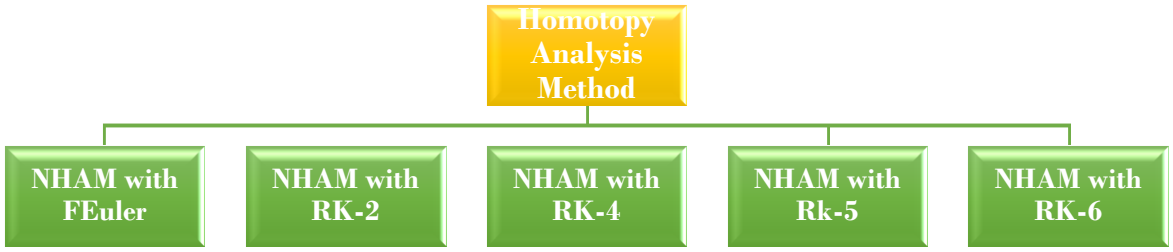

Figure 53 : A Schematic Diagram Homotopy Analysis Methods

5.1. Newton Homotopy Analysis Method (NHAM_FE)

Lio introduces the Homotopy Method (HM) in 2005 [**7**] . In this method, one introduces a Homotopy function for solving equation (5.0). To derive the Homotopy differential equation, like **Hasan** *et al.* [**69**], we can follow the following mathematical steps:

Consider the Homotopy function equation (5.2), defined by equation (5.4). Where $x^{(0)}$ is an initial guess and q is called Homotopy parameter, x^* is a unique solution of the equation

$$H(x,t) = 0 \qquad (5.1.0)$$

By chain rule, we obtained

$$\frac{\partial H}{\partial x}\frac{dx}{dq} + \frac{\partial H}{\partial q} = 0 \qquad (5.1.1.)$$

$$\text{Or} \quad \frac{dx}{dq} = \frac{-\frac{\partial H}{\partial q}}{\frac{\partial H}{\partial x}} = \frac{-H_q}{H_x} = F(q,x) \qquad (5.1.2)$$

Equation (5.1.2) is a first order differential equation named as Newton Homotopy Differential Equation (NHDE). This equation can be solve numerically using Euler's method to find x for parameter $q \in [0,1]$ the numerical solution can be formulated as :

$$x^{(k)} = x^{(k-1)} + hf(q,x); \quad h = rq \qquad (5.1.3)$$

Substitute equation (5.1.2) in equation (5.1.3), we get

$$x^{(k)} = x^{(k-1)} + h\left(-\frac{H_q}{H_x}\right); \quad k = 0,1,2,... \qquad (5.1.4)$$

Homotopy methods give different roots depending on selection of initial guess [**20**], these properties makes HM used to obtain initial guess values for methods that need good initial guess.

5.1.1. Application of the Newton Homotopy Analysis Method (NHAM_FE)

General Algorithm of NHAM method, to approximate the solution of our mechanics problems defined in section (1.2).

Algorithm (5.1): Solves nonlinear second order BVPs by using NHAM_FE.

INPUT: L – length of problem domain (b-a)

u, λ - Initial guess of problem solution

n , h - no of spaces and space between nodes

Maxiter - maximum number of iteration

dL – step length of homotopy parameter

N,hi - no of Euler step & step length

K - no of computation of function (k=1/dL)

STEP- 1. Compute f for iteration from 1 to k

$$f_i = u_{i-1} - 2u_i + u_{i+1} + \frac{1}{h^2}F$$

STEP- 2. Compute Jacobian of F by built in function

$$J = jacobian(F)$$

STEP- 3. Compute NHDE, dxdL by

$$dxdL = \frac{dx}{dL} = -J \setminus F$$

STEP- 4. Compute Euler step by

$$u = u + hidxdL$$

STEP- 5. Display Numerical and Graphical output

Based on Algorithm (5.1) and a MATLAB program, we solved problems 1and 4.

➢ **Numerical and Graphical output by using NHAM(with Forward Euler)**

The numerical method, NHAM_FE is applied and tested on the following nonlinear BVPs that are define in section (1.2) and the numerical result and norm error are displayed by using MATLAB program.

Problem-1- consider Bratu 1-D nonlinear problem defined in section (1.2)

$$u''(x)+\lambda e^{u(x)}=0 \qquad (5.1.1.0)$$

Subject to the dirichlet boundary conditions:

$$u(0)=u(1)=0 \qquad (5.1.1.1)$$

The Analytic solution for this problem given by, equation (1.2.3).

Using equations (5.1.1.0) and (3.3), we can create NLSAEs for Bratu problem

$$F(u_j)=fh^2+u_{j-1}-2u_j+u_{j+1},\ for\ \ j=1,2,...,N \qquad (5.1.1.2)$$

Where $f=-\lambda e^{u_j}$

Equation (5.1.1.2), solve by using NHAM and similar inputs of Newton method for Bratu problem gives the following results:

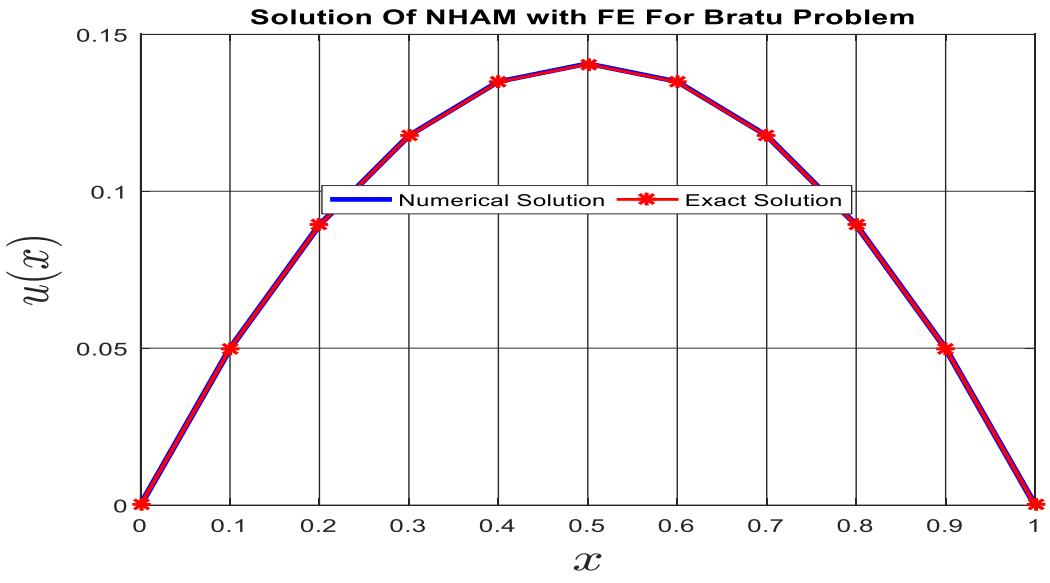

Figure 5.54 Comparison between the approximated solution and exact solution of NHAM for Bratu problem.

Table 5.26: NHAM Euclidian norm of error, number of iteration and CPU time of Bratu problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
9	1	1.26e-40	3.5884-40	8.453125

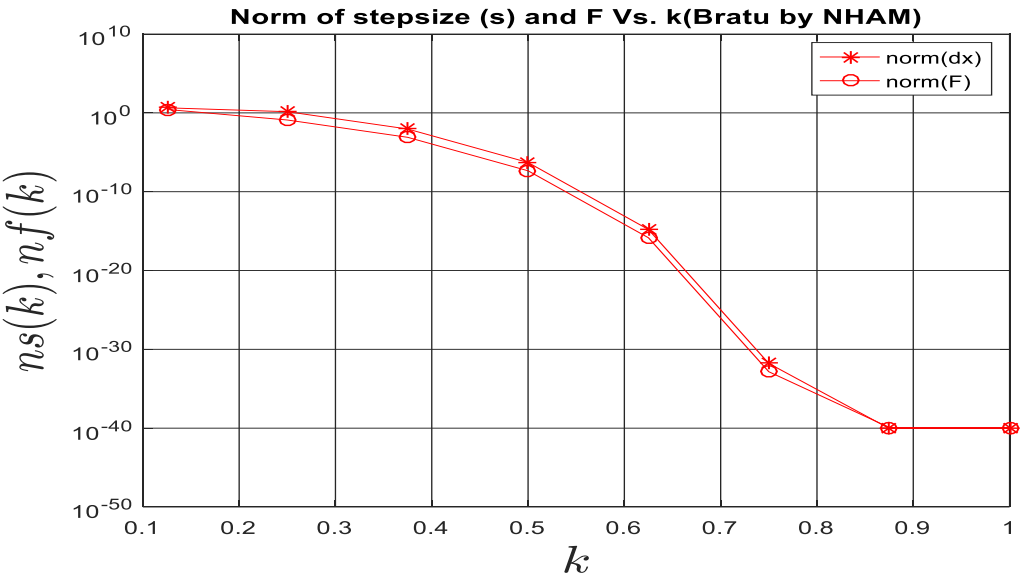

Figure 5.55 NHAM Euclidian Norm of errors Vs number of iterations for Bratu problem.

Fig. 5.54 cites a comparison between the approximate (NHAM) solution and exact solution. This shows a good agreement between the exact solution and the NHAM (approximate) solution for Bratu problem.

As shown in Table 5.26, the output of NHAM converges for Bratu problem and terminates at the ninth iteration. The CPU-time indicates NHAM takes more time than all the previous methods. The Euclidean norm error of both Newton step (du) and the function are very small. Moreover as shown in Figure 5.55, which proves the convergence of NHAM for Bratu problem and the norm error getting smaller in between successive iterates and reach small norm error which is very small than Newton method.

Problem -4 - consider Thermal Explosion nonlinear BVPs defined in section (1.2)

$$u''+\lambda e^{u}=0, \quad x\in(0,1), \tag{5.1.1.3}$$

With the associated Neumann boundary conditions

$$u'(0)=0,\ u(1)=0.$$

The Analytic solution for this problem given by, equation (1.2.12).

Using equations (5.1.1.3) and (3.3), we can create NLSAEs for Thermal Explosion problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1,2,...,N \qquad (5.1.1.4)$$

Where $f = -\lambda e^{u_j}$

Equation (5.1.1.4) solve by using NHAM_FE method by taking similar inputs of Newton method, the following result are obtain for Thermal Explosion method:

Table 4.27: NHAM Euclidian norm of error, number of iteration and CPU time for Thermal Explosion problem.

Number of iteration(k)	Parameter Lambda	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
9	1e-1	1.7e -40	4.38e -40	13.69075

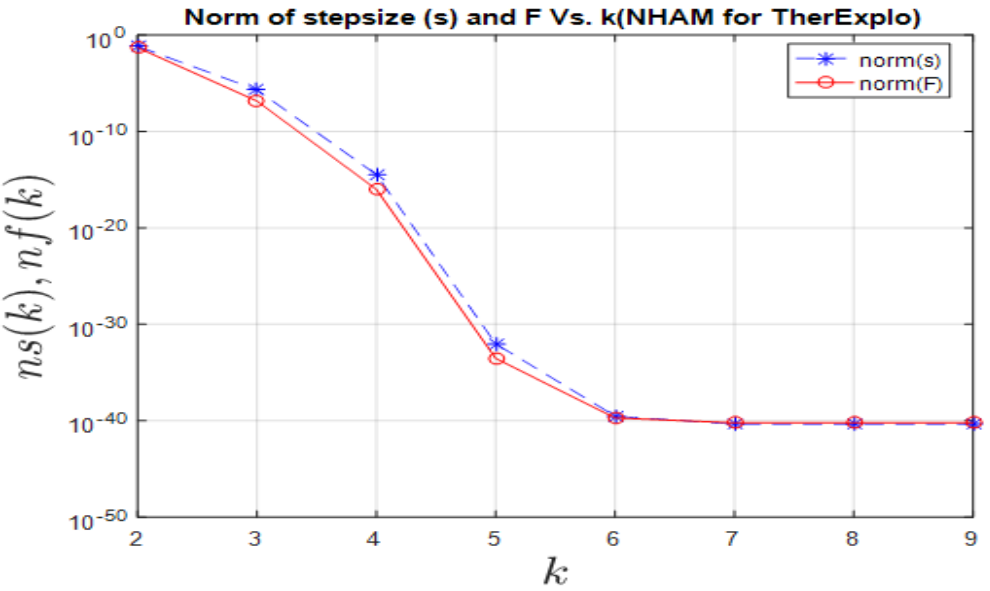

Figure 5.56 NHAM Euclidian Norm of Errors versus number of iteration for Thermal Explosion problem.

As shown in Table 5.56, NHAM with Forward Euler step converges for Thermal Explosion problem and terminate within 9 iteration for $\lambda = 0.1$. Moreover as shown in Figure 5.54, which proves the convergence of NHAM for Thermal Explosion problem and displays the norm errors of a function (F) and Newton step (du) are almost

continuously decreasing and reach an error (10^{-40}), which is below any previous methods error, however its CPU time taken for computation is big (it is expensive) .

5.2. Newton Homotopy Analysis Method by RK2 NHDE (RK2_NHAM)

First order differential equation in equation (5.1.2) named as Newton Homotopy Differential Equation (NHDE) can be written as:

$$\frac{dx}{dq} = \phi(q, x) \qquad (5.2.0)$$

This system of differential equations that we need to solve for our continuation problem has the general form (take $\lambda = q$)

$$\left[\frac{dx_i}{d\lambda}\right]^T = \left[\phi_i\left(\lambda, x_1, x_2, ..., x_n\right)\right]^T, \qquad (5.2.1)$$

Where

$$\left[\phi_i\left(\lambda, x_1, x_2, ..., x_n\right)\right]^T = -J\left(x_1, ..., x_n\right)^{-1}\left[f_i\left(x(0)\right)\right]^T \qquad (5.2.2)$$

To solve (5.2.2) using the Runge-Kutta method of order two (improved Euler), we first choose an integer $N > 0$ and $h = (1-0)/N$. Partition the interval $[0, 1]$ into N subintervals with the mesh points

$$\lambda_j = jh, \quad for\ each\ j = 0, 1, ..., N.$$

We use the notation w_{ij} , for each j =0, 1,...,N and i =1,...,n, to denote an approximation to $x_i(\lambda_j)$. for the initial conditions, set

$$w_{1,0} = x_1(0), \quad w_{2,0} = x_2(0), \quad ..., \quad w_{n,0} = x_n(0). \qquad (5.2.3)$$

Suppose $w_{1,j}, w_{2,j}, ..., w_{n,j}$ have been computed. We obtain $w_{1,j+1}, w_{2,j+1}, ..., w_{n,j+1}$ using the equations

$$k_{1,i} = h\phi_i(\lambda_j, w_{1,j}, w_{2,j}, ..., w_{n,j}),\ \ for\ each\ \ i = 1,\ 2, ..., n;$$

$$k_{2,i} = h\phi_i(\lambda_j + \frac{h}{2}, w_{1,j} + \frac{1}{2}k_{1,1},\ ..., w_{n,j} + \frac{1}{2}k_{1,n})\ \ for\ each\ \ i = 1,\ 2, ..., n;$$

Finally,

$$w_{i,j+1} = w_{i,j} + k_2 h,\quad for\ each\ \ i = 1,\ 2, ..., n.$$

The vector notation

$$k_1 = [k_{1,1}, ..., k_{1,n}]^T, k_2 = [k_{2,1}, ..., k_{2,n}]^T\ \ , and\ \ \ w_j = [w_{1,j}, ..., w_{n,j}]^T$$

Then simplifies equation (5.2.2), gives us $x(0) = x(\lambda_0) = w_0$, and for each $j = 0,1, ...n$

$$k_1 = h\begin{bmatrix} \phi_1(\lambda_j, w_{1,j}, ..., w_{n,j}) \\ \phi_2(\lambda_j, w_{1,j}, ..., w_{n,j}) \\ \vdots \\ \phi_n(\lambda_j, w_{1,j}, ..., w_{n,j}) \end{bmatrix} = h\left[-J(w_{1,j}, ..., w_{n,j})\right]^{-1} F(x(0))$$

$$= h\left[-J(w_j)\right]^{-1} F(x(0));$$

$$k_2 = h\left[-J\left(w_j + \frac{1}{2}k_1\right)\right]^{-1} F(x(0));$$

$$x(\lambda_{j+1}) = x(\lambda_j) + k_2 h = w_j + k_2 h. \qquad (5.2.4)$$

Finally, $x(\lambda_n) = x(1)$) is our approximation to x^*.

Therefore, in the Runge-Kutta method of order two, the calculation of each w_j requires two linear systems to be solve, one each when computing k_1, and k_2 .

An alternative is to use a Runge-Kutta method of order four (RK4), RK5 or RK6 that shows the number of linear systems that needed to be solved.

5.2.1. Application of NHAM with RK_2

To approximate the solution of mechanics problems defined in section (1.2), we wrote an algorithm (5.2) for NHAM with RK2 method.

Algorithm (5.2): Solves nonlinear BVPs by using NHAM with RK_2 step.

INPUT: L **– length of problem domain (b-a)**

u, λ, **- Initial guess of problem solution, parameter**

n , h **- number of spaces and space between nodes**

Maxiter - maximum number of iteration

dL **– step length of homotopy parameter**

N,hi - no of Euler step & step length

K - no of computation of function (k=1/dL)

STEP-1 Compute f for iteration from 1 to k

$$f_i = u_{i-1} - 2u_i + u_{i+1} + \frac{1}{h^2} F$$

Where F is second order ODE of F

STEP-2 Compute Jacobian of F by built in function

$$J = jacobian(F)$$

STEP-3 Compute NHDE, dxdL by .

$$dxdL = \frac{dx}{dL} = -J \setminus F$$

STEP-4 Compute 1st slope and update solution

$$k1 = hi * dL, \; u_1 = u + k1$$

Compute new Jacobian and NHDE

$$J_1 = F'(u_1), \quad dxdL_1 = -J_1 \setminus F$$

STEP-5 Compute slope-2 $k2 = hi * dxdL_1$

STEP-6 Compute Improved Euler step by

$$u = u_1 + (k1 + k2)/2$$

stop

- **Numerical and Graphical output by using NHDE with RK2**

The numerical method, NHAM_RK2 method is apply and tested on the following nonlinear BVPs that are define in section (1.2) and the numerical result and norm error are displayed by using MATLAB program.

Problem-1- consider Bratu 1-D nonlinear problem defined in section (1.2)

$$u''(x)+\lambda e^{u(x)}=0 \qquad (5.2.1.0)$$

Subject to the boundary conditions:

$$u(0)=u(1)=0 \qquad (5.2.1.1)$$

The Analytic solution for this problem given by, equation (1.2.3).

Using equation (5.2.1.0) and (3.3), we can create NLSAEs for Bratu problem

$$F(u_j)=fh^2+u_{j-1}-2u_j+u_{j+1},\ for\ \ j=1,2,...,N$$

Where $f=-\lambda e^{u_j}$

The above NLSAEs solve by using NHAM with RK2 step and take inputs of Newton method for Bratu problem we get the following output:

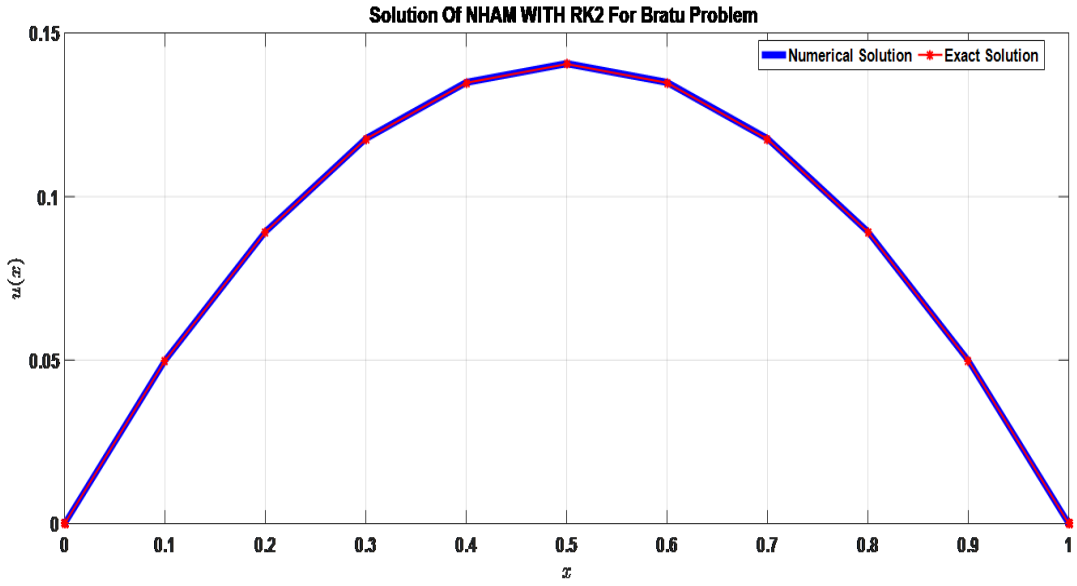

Figure 5.57Comparison between the approximated solution and exact solution of Bratu problem

Table 5.28: NHAM with RK2 Euclidian norm of error, number of iteration and CPU time of Bratu problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of fuction (F)	CPU time
9	1	3.173e-41	3.5883e-40	13.03125

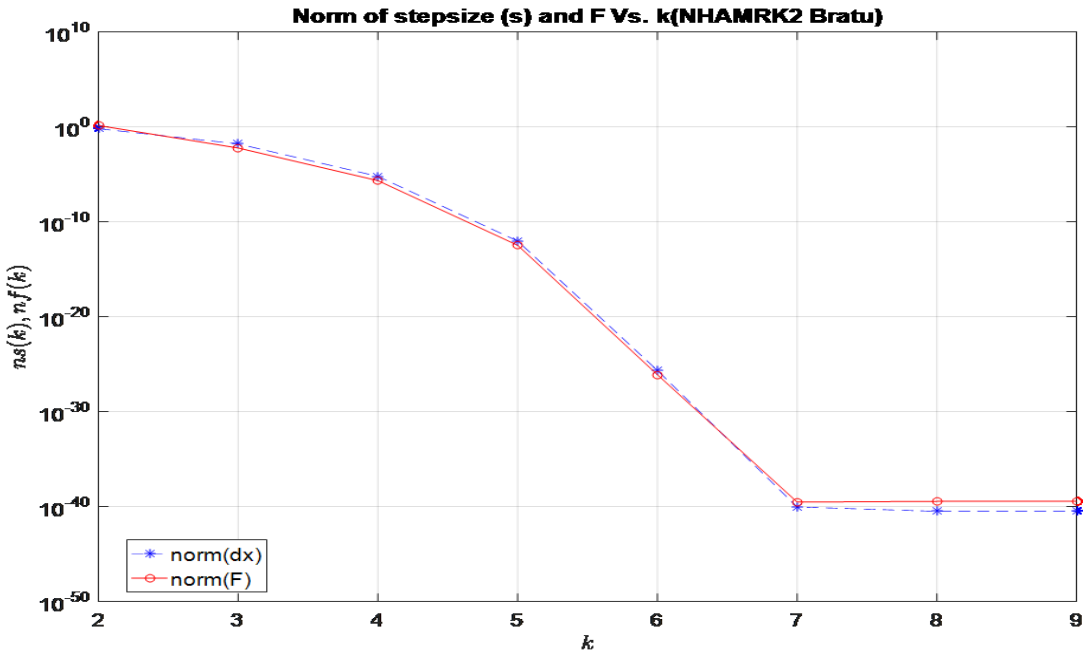

Figure 5.58 NHAM with RK_2 Euclidian Norm of errors versus number of iterations for Bratu problem.

Fig.5.57 cites a comparison between NHAM solution with RK2 step and exact solution. From this comparison, one can see a good agreement between the exact solution and the NHAM with RK2 (approximate) solution.

As shown in Table 5.28, the output of the above MATLAB programing code of NHAM with RK2 step converges for Bratu problem and terminates at the ninth iteration. The CPU-time indicates NHAM with RK2 takes more time than NHAM with Forward Euler. The Euclidean norm error of both Newton step and the function are lesser than NHAM with Forward Euler. Moreover, as shown in Figure 5.58, which proves the convergence of NHAM_RK2 for Bratu problem and the norm error getting smaller in between successive iterates until 7^{th} iteration and finally becomes a smaller change after 7^{th} iteration.

Problem -4 - consider Thermal Explosion nonlinear BVPs defined in section (1.2)

$$u''+\lambda e^{u}=0, \quad x\in(0,1), \qquad (5.2.1.2)$$

With the associated Neumann boundary conditions

$$u'(0)=0,\ u(1)=0.$$

The Analytic solution for this problem given by, equation (1.2.12).

Using equations (5.2.1.2) and (3.3), we can create NLSAEs for Thermal Explosion problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1},\ for\ \ j = 1,2,...,N$$

Where $f = -\lambda e^{u_j}$

The above NLSAEs solve by using NHAM_RK2 and take inputs of Newton method for Thermal Explosion problem gives the following result:

Table 5.29: NHAM with RK2 Euclidian norm of error, number of iteration and CPU time for Thermal Explosion problem.

Number of iteration(k)	Parameter Lambda	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
9	0.1	6.033 X 10^ -41	5.322 X 10^-41	11.00

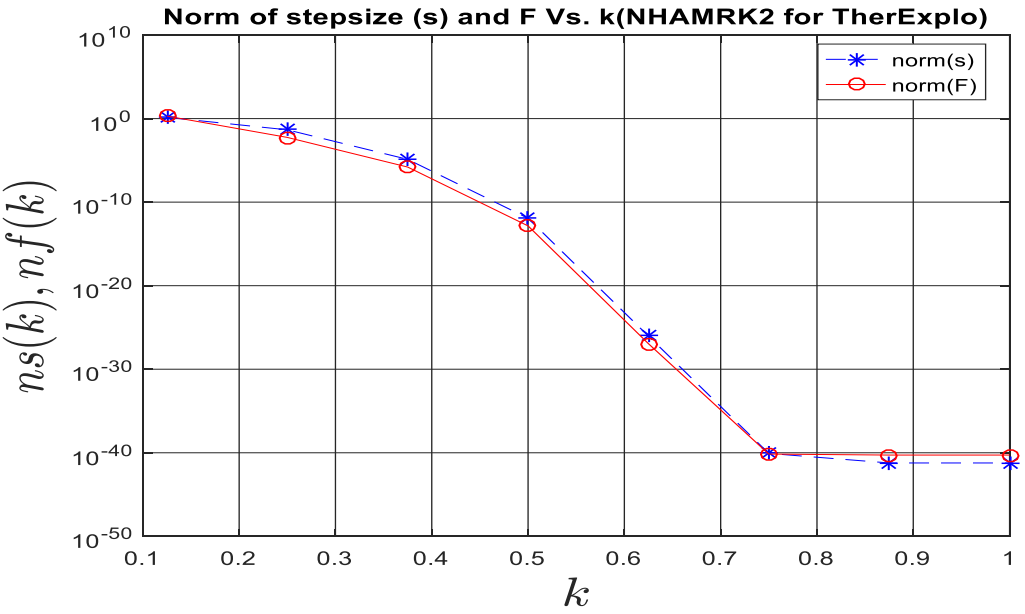

Figure 5.59 NHAM with RK2 Method Euclidian Norm of Errors versus number of iteration for Thermal Explosion problem.

Analysis

As shown in Table 5.29, NHAM with RK2 converges for Thermal Explosion problem and terminate within 9 iteration for $\lambda = 0.1$. Moreover, as shown in Figure 5.59, this proves the convergence of NHAM_RK2 for Thermal Explosion problem and displays the norm errors of a function (F) and Newton step (du). Norm Errors are continuously decreasing and reach an error (10^{-41}), which is better accuracy than NHAM with forward Euler step; however, its CPU time taken for computation is bigger than NHAM with Forward Euler step.

5.3. Newton Homotopy Analysis Method with RK4 NHDE (RK4_NHAM)

NHAM with RK4 NHDE method works, with the same procedure like section 5.2, by only change RK_2 by Runge-Kutta method of order four to solve equation (5.1.2).

Suppose $w_{1,j},\ w_{2,j},\ ...,\ w_{n,j}$ have been computed, we obtain $w_{1,j+1},\ w_{2,j+1},\ ...,\ w_{n,j+1}$ using the equations

$$
(*)\quad
\begin{aligned}
k_{1,i} &= h\phi_i(\lambda_j, w_{1,j}, w_{2,j}, ..., w_{n,j}),\ \ for\ each\ \ i = 1,\ 2, ..., n;\\
k_{2,i} &= h\phi_i\left(\lambda_j + \frac{h}{2}, w_{1,j} + \frac{1}{2}k_{1,1},\ ..., w_{n,j} + \frac{1}{2}k_{1,n}\right)\ \ for\ each\ \ i = 1,\ 2, ..., n;\\
k_{3,i} &= h\phi_i\left(\lambda_j + \frac{h}{2}, w_{1,j} + \frac{1}{2}k_{2,1},\ ..., w_{n,j} + \frac{1}{2}k_{2,n}\right)\ \ for\ each\ \ i = 1,\ 2, ..., n;\\
k_{4,i} &= h\phi_i(\lambda_j + h, w_{1,j} + k_{3,1}, w_{2,j} + k_{3,2},\ ..., w_{n,j} + k_{3,n})\ \ for\ each\ \ i = 1,\ 2, ..., n;
\end{aligned}
$$

and, finally

$$
w_{i,j+1} = w_{i,j} + \frac{1}{6}\left(k_{1,i} + 2k_{2,i} + 2k_{3,i} + k_{4,i}\right),\quad for\ each\ \ i = 1,\ 2, ..., n.
$$

Then simplifies Eq. (*) gives us $x(0) = x(\lambda_0) = w_0$, and for each $j = 0, 1, ...n$

$$
k_1 = h\begin{bmatrix} \phi_1\left(\lambda_j, w_{1,j}, ..., w_{n,j}\right) \\ \phi_2\left(\lambda_j, w_{1,j}, ..., w_{n,j}\right) \\ \vdots \\ \phi_n\left(\lambda_j, w_{1,j}, ..., w_{n,j}\right) \end{bmatrix} == h\left[-J\left(w_j\right)\right]^{-1} F(x(0));\quad k_2 = h\left[-J\left(w_j + \frac{1}{2}k_1\right)\right]^{-1} F(x(0));
$$

$$
k_3 = h\left[-J\left(w_j + \frac{1}{2}k_2\right)\right]^{-1} F(x(0));\qquad k_4 = h\left[-J\left(w_j + k_3\right)\right]^{-1} F(x(0));
$$

$$
x\left(\lambda_{j+1}\right) = x\left(\lambda_j\right) + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) = w_j + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right). \qquad (5.3.1)
$$

Finally, $x\left(\lambda_n\right) = x(1)$) is our approximation to x^*.

Therefore, in the Runge-Kutta method of order four, the calculation of each requires four linear systems to be solve, one each when computing $k_1,\ k_2,\ k_3,\ and\ k_4$.

5.3.1. Application of the RK_4 in NHAM Method

Algorithm (5.3) for NHAM_RK4 used to approximate the solution of mechanics problems defined in section (1.2).

- **Algorithm (5.3): Solves nonlinear second order BVPs by using NHAM with RK4.**

INPUT: L – length of problem domain (b-a)

u, λ - Initial guess solution, paramter

n , h -number of spaces and space between nodes

Maxiter - maximum number of iteration

dL – step length of homotopy parameter

N,hi - no of Euler step & step length

K - no of computation of function (k=1/dL)

STEP-1 Compute f for iteration from 1 to k

$$f_i = u_{i-1} - 2u_i + u_{i+1} + \frac{1}{h^2} F$$

Where F is second order ODE of F

STEP-2 Compute Jacobian of F by built in function

$$J = jacobian(F)$$

STEP-3 Compute NHDE, dxdL by .

$$dxdL = \frac{dx}{dL} = -J \setminus F$$

STEP-4 Compute 1st slope and update solution

$$k1 = hi * dxdL,\ u_1 = u + 3k1/4$$

Compute new Jacobian and NHDE

$$J_1 = F'(u_1),\quad dxdL_1 = -J_1 \setminus F, k2 = hidxdL_1$$

STEP-5 update u,

$$u_2 = u + k1/3 + k2/6$$

Compute new Jacobian and NHDE

$$J_2 = F'(u_2),\quad dxdL_2 = -J_2 \setminus F, k3 = hidxdL_2$$

STEP-6 Compute u by

$$u_3 = u - k1/3 - 2k2/6 + 2k3$$

Compute new Jacobian and NHDE

$$J_3 = F'(u_3), \quad dxdL_3 = -J_3 \setminus F, k4 = hidxdL_3$$

STEP-7 Compute RK4 solution

$$u = u + (k1 + 4k3 + k4)/6$$

STEP-8 Display Numerical and Graphical output

stop

- **Numerical and Graphical output by using NHAM with RK4**

The numerical method, NHAM_RK4 NHDE is applied and tested on the following nonlinear BVPs that are define in section (1.2) and the numerical result and norm error are displayed by using MATLAB program.

Problem-1- consider Bratu 1-D nonlinear problem defined in section (1.2)

$$u''(x) + \lambda e^{u(x)} = 0 \qquad (5.3.1.0)$$

Subject to the dirchlet boundary conditions:

$$u(0) = u(1) = 0 \qquad (5.3.1.1)$$

Using equation (5.3.1.0) and (3.3), we can create NLSAEs for Bratu problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1, 2, ..., N$$

Where $f = -\lambda e^{u_j}$

The above NLSAEs solve by using NHAM with RK4 step and similar inputs of NHAM for the same problem we obtain the following result:

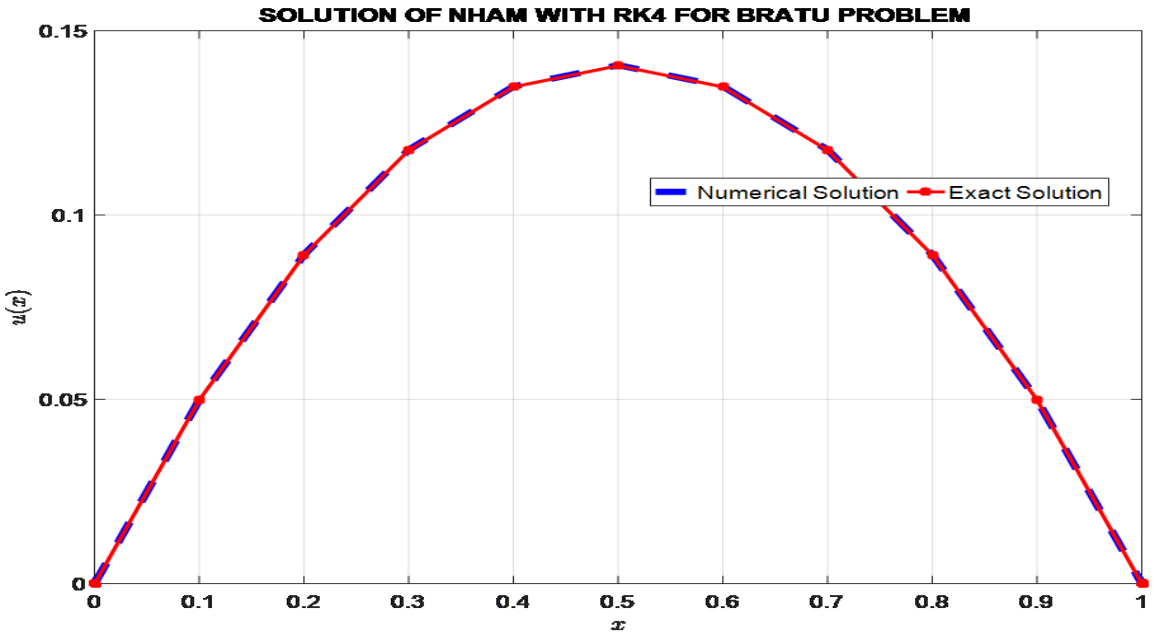

Figure 5.60 Comparison between the approximated solution and exact solution of Bratu problem.

Table 4.30: NHAM with RK4 Euclidian norm of error, number of iteration and CPU time for Bratu problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of fuction (F)	CPU time
9	1	9.75e - 42	8.05e-41	16.421875

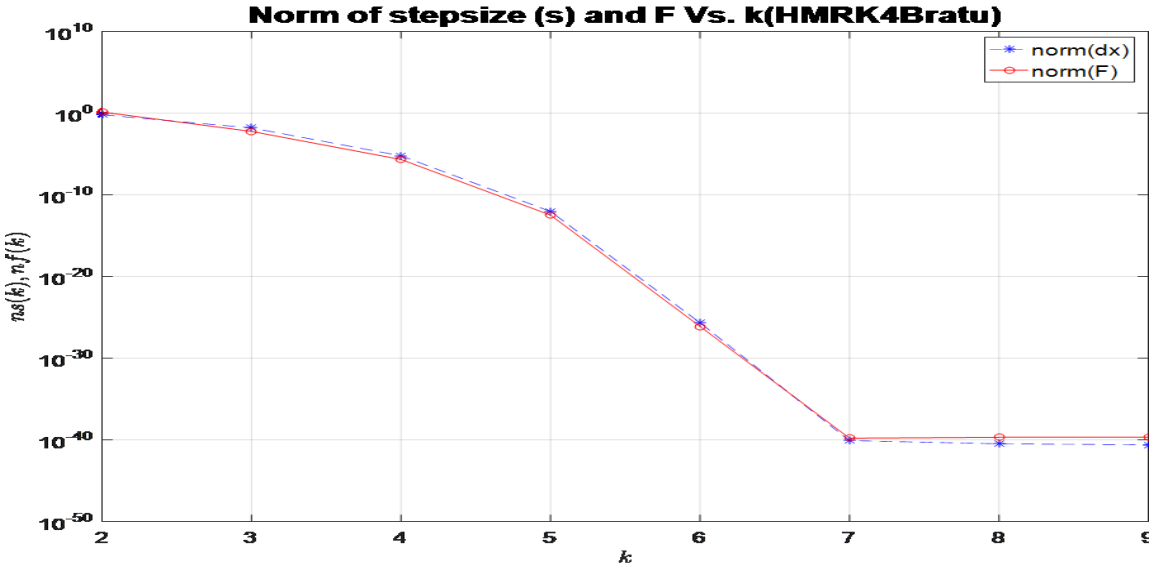

Figure 5.61 NHAM with RK_4 Euclidian Norm of errors versus number of iterations for Bratu problem.

Figure 5.60 cites a comparison between the approximate solution obtained by NHAM with RK4 and exact solution. From this comparison, one can see a good agreement between the exact solution and the NHAM with RK4 (approximate) solution.

As shown in Table 5.30, the output of the MATLAB code of NHAM with RK4 step converges for Bratu problem and shows convergence at nine iterations. The CPU-time indicates NHAM with RK4 takes more than NHAM with RK2 for the same problem. The Euclidean norm error of Newton step (du) and function is also less than NHAM with RK2. Moreover as shown in Figure 5.61 which proves the convergence of NHAM with RK4 for Bratu problem and the norm error getting smaller in between successive iterates until 7^{th} iteration and finally its change of error is not fast.

Problem -4 - consider Thermal Explosion nonlinear BVPs defined in section (1.2)

$$u''+\lambda e^{u}=0, \quad x\in(0,1), \qquad (5.3.1.2)$$

With the associated Neumann boundary conditions

$$u'(0)=0,\ u(1)=0.$$

The Analytic solution for this problem given by, equation (1.2.12).

Using equations (5.3.1.2) and (3.3), we can create NLSAEs for Thermal Explosion problem

$$F(u_j)=fh^2+u_{j-1}-2u_j+u_{j+1},\ for\ \ j=1,2,...,N$$

Where $f=-\lambda e^{u_j}$

These NLSAEs can be solve by using NHAM with RK4 step and inputs of NHAM for the same problem gives the following result:

Table 5.31: NHAM with RK4 step Euclidian norm of error, number of iteration and CPU time for Thermal Explosion problem.

Number of iteration(k)	Parameter C	Euclidian norm of Newton step (du)	Euclidian norm of fuction (F)	CPU time
9	0.1	7.0482 X 10^ -42	8.69 X 10^-41	16.765625

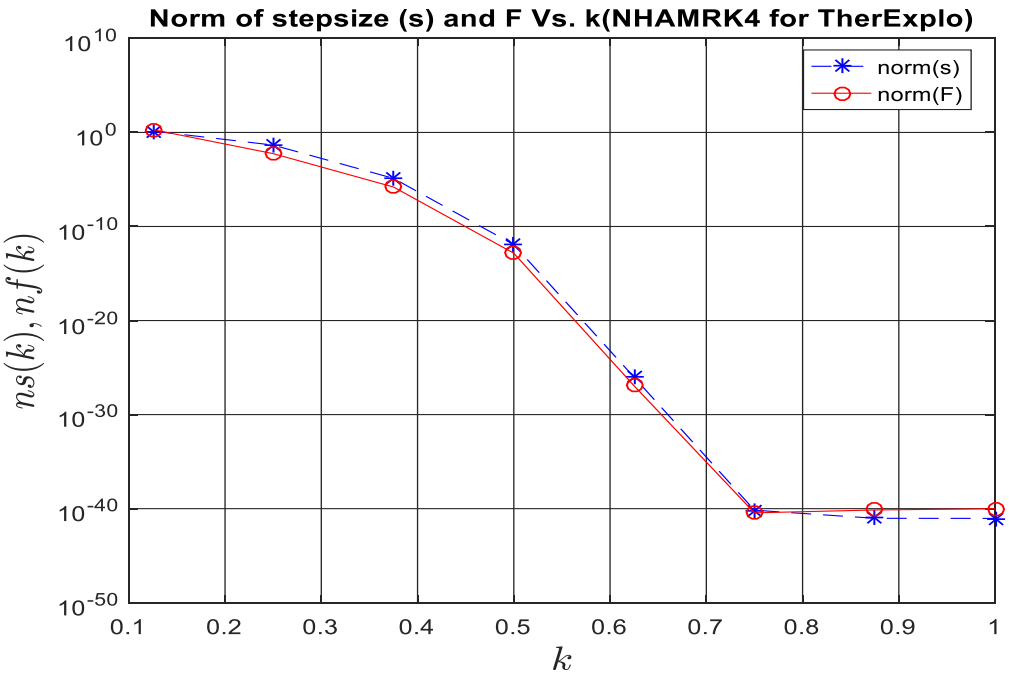

Figure 5.62 NHAMRK4 method Euclidian Norm of Errors versus number of iteration for Thermal Explosion problem.

As shown in Table 5.31, NHAM with RK4 converges for Thermal Explosion problem and terminate within 9 iteration for $\lambda = 0.1$. Moreover as shown in Figure 5.62, this proves the convergence of NHAM with RK4 step for, Thermal Explosion problem. It displays the norm errors of a function (F) and Newton steps (du) is continuously decreasing and reach an error (10^{-41}), which is better accuracy than NHAM with RK2 step, however its CPU time taken for computation is bigger than NHAM with RK2 step.

5.4. Newton Homotopy Analysis Method with RK5 NHDE (RK5_NHAM)

NHAM with RK5_NHDE method works, with the same procedure like section 5.2 by only change RK2 by Runge-Kutta method of order five to solve equation (5.1.2).

Suppose $w_{1,j}, w_{2,j}, ..., w_{n,j}$ have been computed, we obtain $w_{1,j+1}, w_{2,j+1}, ..., w_{n,j+1}$ using the equations

$$k_{1,i} = h\phi_i(\lambda_j, w_{1,j}, w_{2,j}, ..., w_{n,j}),\ \textit{for each}\ i = 1, 2, ..., n;$$

$$k_{2,i} = h\phi_i(\lambda_j + \frac{1}{4}, w_{1,j} + \frac{1}{4}k_{1,1}, ..., w_{n,j} + \frac{1}{4}k_{1,n})\ \textit{for each}\ i = 1, 2, ..., n;$$

$$k_{3,i} = h\phi_i(\lambda_j + \frac{1}{4}, w_{1,j} + \frac{1}{8}k_{1,1}, ..., w_{n,j} + \frac{1}{8}k_{2,n})\ \textit{for each}\ i = 1, 2, ..., n;$$

$$k_{4,i} = h\phi_i(\lambda_j + \frac{1}{2}, w_{1,j} - \frac{1}{2}k_{2,1}, w_{2,j} + k_{3,2}, ..., w_{n,j} + k_{3,n})\ \textit{for each}\ i = 1, 2, ..., n;$$

$$k_{5,i} = h\phi_i(\lambda_j + \frac{3}{4}, w_{1,j} + \frac{3}{16}k_{1,1}, w_{2,j}, ...w_{n,j} + \frac{9}{16}k_{4,2})\ \textit{for each}\ i = 1, 2, ..., n;$$

$$k_{6,i} = h\phi_i(\lambda_j + 1, w_{1,j} - \frac{3}{7}k_{1,1} + \frac{2}{7}k_{2,1} + \frac{12}{7}k_{3,1}, \frac{12}{7}k_{4,1} + \frac{8}{7}k_{5,1})\ \textit{for each}\ i = 1, 2, ..., n;$$

finally

$$w_{i,j+1} = w_{i,j} + \frac{1}{90}\left(7k_{1,i} + 32k_{3,i} + 12k_{4,i} + 32k_{5,i} + 7k_6\right),\quad \textit{for each}\ i = 1, 2, ..., n.$$

Simplifies Eq. (5.2.2) gives us $x(0) = x(\lambda_0) = w_0$, and for each $j = 0,1,...n$

$$k_1 = h\begin{bmatrix} \phi_1\left(\lambda_j, w_{1,j}, ..., w_{n,j}\right) \\ \phi_2\left(\lambda_j, w_{1,j}, ..., w_{n,j}\right) \\ \vdots \\ \phi_n\left(\lambda_j, w_{1,j}, ..., w_{n,j}\right) \end{bmatrix} = h\left[-J\left(w_{1,j}, ..., w_{n,j}\right)\right]^{-1} F(x(0))$$

$$= h\left[-J\left(w_j\right)\right]^{-1} F(x(0));$$

$$k_2 = h\left[-J\left(w_j + \frac{1}{2}k_1\right)\right]^{-1} F(x(0));$$

$$k_3 = h\left[-J\left(w_j + \frac{1}{2}k_2\right)\right]^{-1} F(x(0));$$

$$k_4 = h\left[-J\left(w_j + k_3\right)\right]^{-1} F(x(0));$$

$$k_5 = h\left[-J\left(w_j + k_4\right)\right]^{-1} F(x(0));$$

$$k_6 = h\left[-J\left(w_j + k_5\right)\right]^{-1} F(x(0));$$

$$x\left(\lambda_{j+1}\right) = x\left(\lambda_j\right) + \frac{1}{90}\left(7k_1 + 32k_3 + 12k_4 + 32k_5 + 7k_6\right)$$

$$= w_j + \frac{1}{90}\left(7k_1 + 32k_3 + 12k_4 + 32k_5 + 7k_6\right). \qquad (5.4.0)$$

Finally, $x(\lambda_n) = x(1)$) is our approximation to x^*.

Therefore, in the Runge-Kutta method of order five, the calculation of each w_j requires five linear systems to be solve, one each when computing k_1, k_2, k_3, k_4, k_5 *and* k_6.

5.4.1. Application of the RK_5 in NHAM Method

To approximate the solution of mechanics problems defined in section (1.2), we wrote an algorithm (5.4) for NHAM_RK5 method.

- **Algorithm (5.4): Solves nonlinear BVPs by using NHAM with RK5.**

INPUT: L **– length of problem domain (b-a)**

u, λ **- Initial guess solution,paramter**

n , h **- number of spaces and space between nodes**

Maxiter - maximum number of iteration

dL **– step length of homotopy parameter**

N,hi - no of Euler step & step length

K - no of computation of function (k=1/dL)

STEP-1 Compute f for iteration from 1 to k

$$f_i = u_{i-1} - 2u_i + u_{i+1} + \frac{1}{h^2} F$$

Where F is second order ODE of F

STEP-2 Compute Jacobian of F by built in function

STEP-3 Compute NHDE, dxdL by .

$$dxdL = \frac{dx}{dL} = -J \setminus F$$

STEP-4 Compute 1st slope and update solution

$$k1 = hi * dxdL,\ u_1 = u + k1/4$$

Compute new Jacobian and NHDE

$$J_1 = F'(u_1),\quad dxdL_1 = -J_1 \setminus F, k2 = hidxdL_1$$

STEP-5 update u,

$$u_2 = u + (k1 + k2)/8$$

Compute new Jacobian and NHDE

$$J_2 = F'(u_2),\quad dxdL_2 = -J_2 \setminus F, k3 = hidxdL_2$$

STEP-6 Compute u by

$$u_3 = u - k2/2 + k3$$

Compute new Jacobian and NHDE

$$J_3 = F'(u_3), \quad dxdL_3 = -J_3 \backslash F, k4 = hidxdL_3$$

STEP-7 Compute u by

$$u_4 = u - 3k1/16 + 9k4/16$$

Compute new Jacobian and NHDE

$$J_4 = F'(u_4), \quad dxdL_4 = -J_4 \backslash F, k5 = hidxdL_4$$

STEP-8 Compute u by

$$u_5 = u - 3k1/7 + 2k2/7 + 12k3/7 - 12k4/7 + 8k5/7$$

Compute new Jacobian and NHDE

$$J_5 = F'(u_5), \quad dxdL_5 = -J_5 \backslash F, k6 = hi * dxdL_5$$

STEP-9 Compute RK5 solution

$$u = u + (7k1 + 32k3 + 12k4 + 32k5 + 7k6)/90$$

STEP-10 Display Numerical and Graphical output

- **Numerical and Graphical output by using NHAM with RK5**

NHAM_RK5 of NHDE is apply and tested on the nonlinear BVPs that are define in section (1.2) and the numerical result and norm error are display by using MATLAB program.

Problem-1- consider Bratu 1-D nonlinear problem defined in section (1.2)

$$u''(x) + \lambda e^{u(x)} = 0 \qquad (5.4.1.0)$$

Subject to the boundary conditions:

$$u(0) = u(1) = 0 \qquad (5.4.1.1)$$

The Analytic solution for this problem given by, equation (1.2.3).

Using equation (5.4.1.0) and (3.3), we can create NLSAEs for Bratu problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1, 2, ..., N$$

Where $f = -\lambda e^{u_j}$

These NLSAEs can be solve by using NHAM with RK5 step and by taking similar inputs of NHAM for the same problem gives the following result:

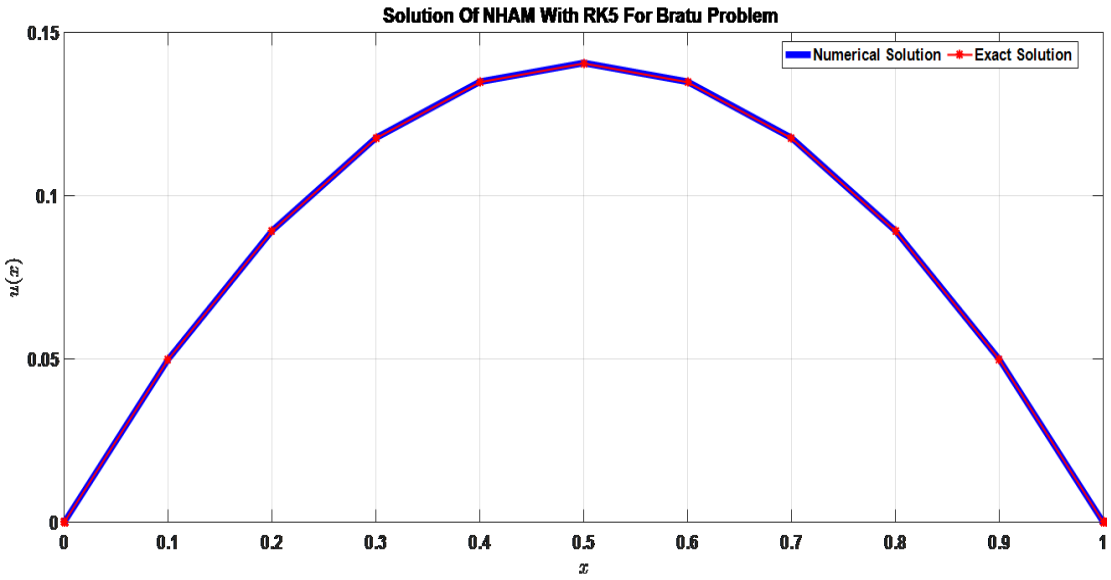

Figure 5.63 Comparison between the approximated solution and exact solution of Bratu problem.

Table 5.32: NHAM with RK5 Euclidian norm of error, number of iteration and CPU time for Bratu problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
9	1	2.2698e - 41	1.4481e - 40	20.375

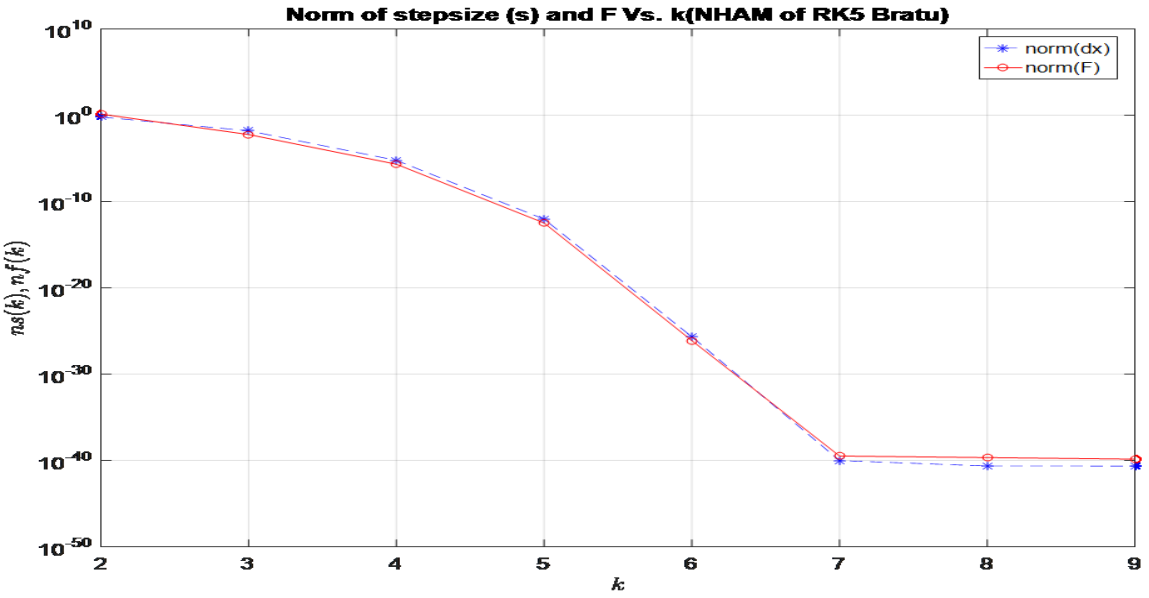

Figure 5.64 NHAM with RK5 Euclidian Norm of errors versus number of iterations for Bratu problem.

Fig.5.63 cites a comparison between NHAM with RK5 solution and exact solution. This shows a good agreement between the exact solution and the NHAM with RK5 (approximate) solution.

As shown in Table 5.32, the output of the MATLAB code of NHAM with RK5 step converges for Bratu problem and terminates at the ninth iteration. The CPU-time indicates NHAM with RK5 takes more time than NHAM with RK4. The Euclidean norm errors of Newton step is also less than NHAM with RK4 of the same problem. Moreover as shown in Figure 5.64, which proves the convergence of NHAM with RK5 for Bratu problem and the norm error getting smaller in between successive iterates until 7^{th} iteration and finally it becomes almost no change after 7^{th} iteration.

Problem -4 - consider Thermal Explosion nonlinear BVPs defined in section (1.2)

$$u''+\lambda e^{u}=0, \quad x\in(0,1), \qquad (5.4.1.2)$$

With the associated Neumann boundary conditions

$$u'(0)=0,\ u(1)=0.$$

The Analytic solution for this problem given by, equation (1.2.12).

Using equations (5.4.1.2) and (3.3), we can create NLSAEs for Thermal Explosion problem

$$F(u_j)=fh^2+u_{j-1}-2u_j+u_{j+1},\ for\ \ j=1,2,...,N \qquad (5.4.1.3)$$

Where $f=-\lambda e^{u_j}$

Eq. (5.4.1.3) can be solving by using NHAM with RK5 step and using inputs of NHAM for the same problem give the following output:

Table 5.33: RK5 step of NHAM Euclidian norm of error, number of iteration and CPU time for Thermal Explosion problem.

Number of iteration(k)	Parameter Lambda	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
9	0.1	9.21e -42	9.215e -41	18.01562

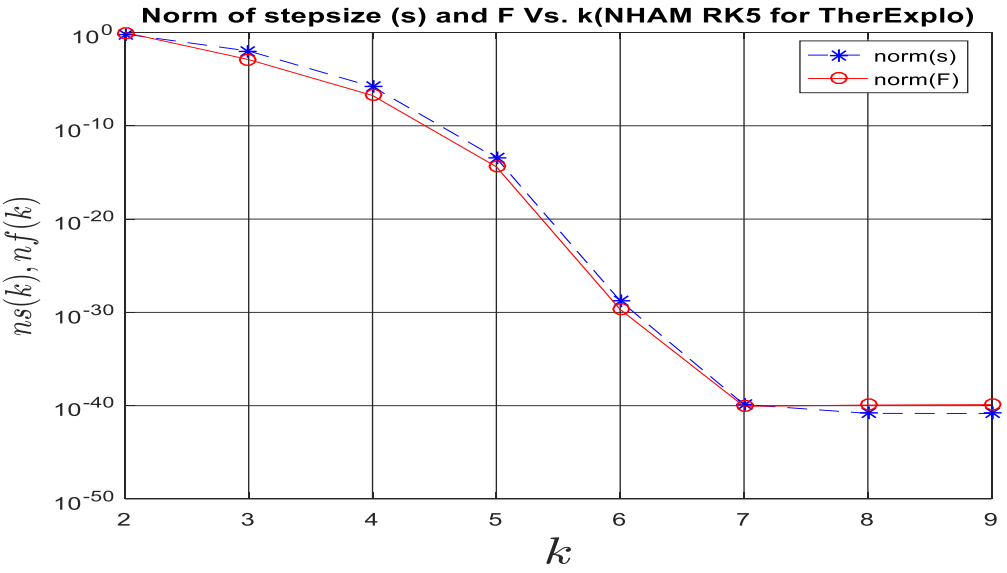

Figure 5.65 8 HM of RK5 Euclidian Norm of Errors versus number of iteration for Thermal Explosion problem.

As shown in Table 5.33, NHAM of RK5 converges for Thermal Explosion problem and terminates within 9 iteration for $\lambda = 0.1$. Moreover, as shown in Figure 5.65, this proves the convergence of NHAM with RK5 steps for Thermal Explosion problem. It displays the norm errors of a function (F) and Newton step (du) are continuously decreasing and reach an error (10^{-41}), which is better accuracy than NHAM with RK4 step; however, its CPU time taken for computation is bigger than NHAM with RK4 step.

5.5. Newton Homotopy Analysis Method with RK6 NHDE (RK6_NHAM)

This method works with the same procedure like section 5.2 but by only change RK2 by Runge-Kutta method of order six to solve equation (5.1.2).

Suppose $w_{1,j}, w_{2,j}, ..., w_{n,j}$ have been computed, we obtain $w_{1,j+1}, w_{2,j+1}, ..., w_{n,j+1}$ using the equations

$$k_{1,i} = h\phi_i(\lambda_j, w_{1,j}, w_{2,j}, ..., w_{n,j}), \text{ for each } i = 1, 2, ..., n;$$

$$k_{2,i} = h\phi_i\left(\lambda_j + \frac{1}{3}, w_{1,j} + \frac{1}{3}k_{1,1}, ..., w_{n,j} + \frac{1}{3}k_{1,n}\right) \text{ for each } i = 1, 2, ..., n;$$

$$k_{3,i} = h\phi_i\left(\lambda_j + \frac{2}{3}, w_{1,j} + \frac{2}{3}k_{2,1}, ..., w_{n,j} + \frac{2}{3}k_{2,n}\right) \text{ for each } i = 1, 2, ..., n;$$

$$k_{4,i} = h\phi_i\left(\lambda_j + \frac{1}{3}, w_{1,j} + \frac{1}{12}k_{1,1}, w_{2,j} + \frac{1}{3}k_{2,1} - \frac{1}{12}k_{3,1}\right) \text{ for each } i = 1, 2, ..., n;$$

$$k_{5,i} = h\phi_i(\lambda_j + \frac{1}{2}, w_{1,j} - \frac{1}{16}k_{1,1} + \frac{9}{8}k_{2,1} - \frac{3}{16}k_3 - \frac{3}{8}k_4) \quad for\ each\ \ i = 1,\ 2,...,n;$$

$$k_{6,i} = h\phi_i(\lambda_j + \frac{1}{2}, w_{1,j} + \frac{9}{8}k_{2,1} - \frac{3}{8}k_{3,1} - \frac{3}{4}k_4 - \frac{1}{2}k_5) \quad for\ each\ \ i = 1,\ 2,...,n;$$

$$k_{7,i} = h\phi_i(\lambda_j + 1, w_{1,j} + \frac{9}{44}k_{1,1} - \frac{9}{11}k_{2,1} + \frac{63}{44}k_3 + \frac{18}{11}k_4 - \frac{16}{11}k_6) \quad for\ each\ \ i = 1,\ 2,...,n;$$

and, finally

$$w_{i,j+1} = w_j + \left(\frac{11}{120}k_1 + \frac{27}{40}k_3 - \frac{4}{15}k_5 - \frac{4}{15}k_6 + \frac{11}{120}k_7\right).$$

Simplifies Eq. (5.2.2) gives us $x(0) = x(\lambda_0) = w_0$, and for each $j = 0,1,...n$

$$k_1 = h\begin{bmatrix} \phi_1(\lambda_j, w_{1,j},..., w_{n,j}) \\ \phi_2(\lambda_j, w_{1,j},..., w_{n,j}) \\ \vdots \\ \phi_n(\lambda_j, w_{1,j},..., w_{n,j}) \end{bmatrix} = h\left[-J\left(w_{1,j},..., w_{n,j}\right)\right]^{-1} F(x(0))$$

$$= h\left[-J\left(w_j\right)\right]^{-1} F(x(0));$$

$$k_2 = h\left[-J\left(w_j + \frac{1}{2}k_1\right)\right]^{-1} F(x(0));$$

$$k_3 = h\left[-J\left(w_j + \frac{1}{2}k_2\right)\right]^{-1} F(x(0));$$

$$k_4 = h\left[-J\left(w_j + k_3\right)\right]^{-1} F(x(0));$$

$$k_5 = h\left[-J\left(w_j + k_4\right)\right]^{-1} F(x(0));$$

$$k_6 = h\left[-J\left(w_j + k_5\right)\right]^{-1} F(x(0));$$

$$x(\lambda_{j+1}) = w_j + \left(\frac{11}{120}k_1 + \frac{27}{40}k_3 - \frac{4}{15}k_5 - \frac{4}{15}k_6 + \frac{11}{120}k_7\right). \qquad (5.5.1)$$

Finally, $x(\lambda_n) = x(1)$ is our approximation to x^*

Therefore, in the Runge-Kutta method of order six, the calculation of each w_j requires six linear systems to be solve, one each when computing k_1, k_2, k_3, k_4, k_5, k_6 *and* k_7 [**67**].

5.5.1. Application of the RK_6 in NHAM Method

To approximate the solution of mechanics problems defined in section (1.2), we wrote an algorithm (5.5) for NHAM_RK6 method.

- **Algorithm (5.5): Solves nonlinear BVPs by using NHAM with RK6.**

INPUT: L – length of problem domain (b-a)/**h**

u, λ - Initial guess solution and paramter values
n , h - number of spaces and space between nodes
Maxiter - maximum number of iteration
dL – step length of homotopy parameter
N,hi - no of Euler step & step length
K - no of computation of function (k=1/dL)

STEP-1 Compute f for iteration from 1 to Maxiter

$$f_i = u_{i-1} - 2u_i + u_{i+1} + \frac{1}{h^2} F$$

Where F is second order ODE of F

STEP-2 Compute Jacobian of F by built in function

$$J = jacobian(F)$$

STEP-3 Compute NHDE, dxdL by .

$$dxdL = \frac{dx}{dL} = -J \setminus F$$

STEP-4 Compute 1st slope and update solution

$$k1 = hi * dxdL,\ u_1 = u + k1/3$$

Compute new Jacobian and NHDE

$$J_1 = F'(u_1),\quad dxdL_1 = -J_1 \setminus F, k2 = hidxdL_1$$

STEP-5 update u,

$$u_2 = u + 2k2/3$$

Compute new Jacobian and NHDE

$$J_2 = F'(u_2),\quad dxdL_2 = -J_2 \setminus F, k3 = hidxdL_2$$

STEP-6 Compute u by

$$u_3 = u - k1/12 + k2/3 + k3/12$$

Compute new Jacobian and NHDE

$$J_3 = F'(u_3), \quad dxdL_3 = -J_3 \setminus F, k4 = hidxdL_3$$

STEP-7 Compute u by

$$u_3 = u - k1/16 - 9k2/8 - 3k3/16 + 3k4/8$$

Compute new Jacobian and NHDE

$$J_4 = F'(u_4), \quad dxdL_4 = -J_4 \setminus F, k5 = hidxdL_4$$

STEP-8 Compute u by

$$u_4 = u + 9k2/8 - 3k3/8 - 3k4/4 + k5/2$$

Compute new Jacobian and NHDE

$$J_5 = F'(u_5), \quad dxdL_5 = -J_5 \setminus F, k6 = hi * dxdL_5$$

Compute u by

$$u_5 = u + 9k1/44 - 9k2/11 - 63k3/44 + 18k4/11 - 16k6/11$$

Compute new Jacobian and NHDE

$$J_6 = F'(u_6), \quad dxdL_6 = -J_6 \setminus F, k7 = hi * dxdL_6$$

STEP-9 Compute RK6 solution

$$u_6 = u + 11k1/120 + 27k3/40 + 27k4/40 - 4k5/15 - 4k6/15 + 11k7/120$$

STEP-10 Display Numerical and Graphical output

stop

- **Numerical and Graphical output by using NHDE with RK6**

NHAM_RK6 of NHDE is applied and tested on nonlinear BVPs that are define in section (1.2) and the numerical result and norm errors are displayed by using MATLAB program.

Problem-1- consider Bratu 1-D nonlinear problem defined in section (1.2)

$$u''(x) + \lambda e^{u(x)} = 0 \qquad (5.5.1.0)$$

Subject to the boundary conditions:

$$u(0) = u(1) = 0 \qquad (5.5.1.1)$$

The Analytic solution for this problem given by, equation (1.2.12).

Using equation (5.5.1.0) and (3.3), we can create NLSAEs for Bratu problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1,2,...,N$$

Where $f = -\lambda e^{u_j}$

The above NLSAEs solve by using NHAM with RK6 steps and by taking similar inputs of NHAM for Thermal Explosion problem give the following result:

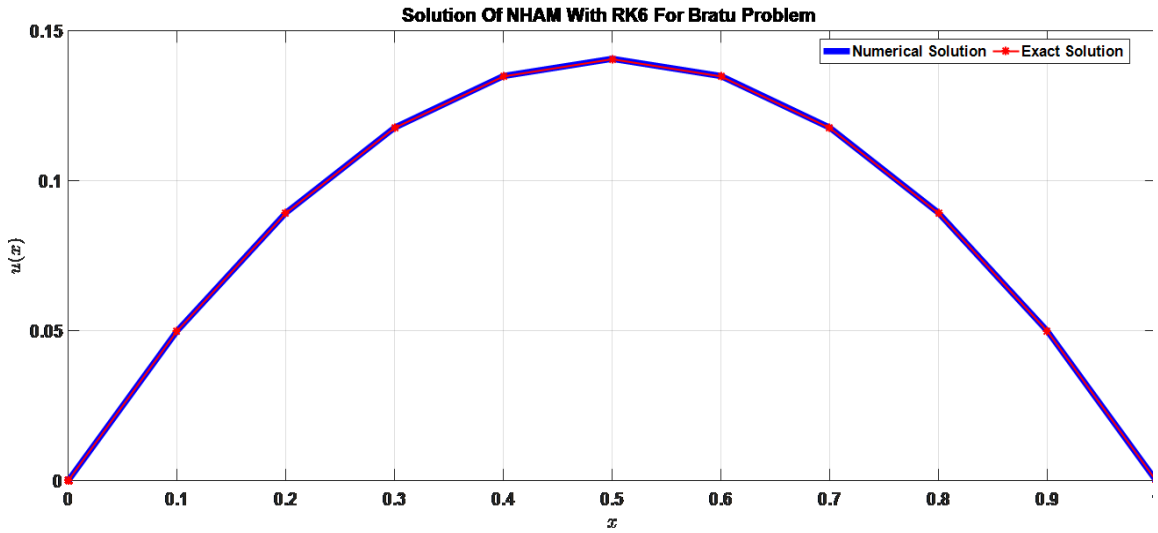

Figure 5.66 Comparison between the approximated solution and exact solution of Bratu problem.

Table 5.34: NHAM with RK6 Euclidian norm error, number of iteration and CPU time for Bratu problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of fuction (F)	CPU time
9	1	3.3233e - 41	3.091e - 40	22.203125

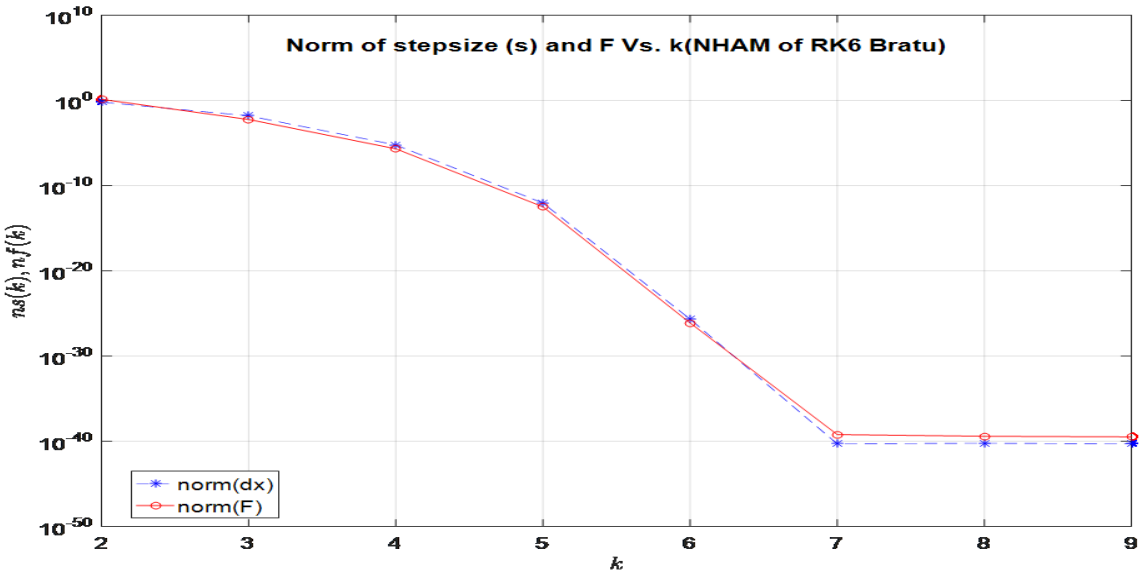

Figure 5.67 NHAM with RK6 Euclidian Norm of errors versus number of iterations for Bratu problem.

Fig.5.66 cites a comparison between the approximate solution obtained by NHAM with RK6 and exact solution. From this comparison, one can see a good agreement between the exact solution and the NHAM with RK6 solution.

As shown in Table 5.34, the output of the MATLAB of NHAM with RK6 step converges for Bratu problem and terminates at the ninth iteration. The CPU-time indicates NHAM with RK6 takes more time than NHAM with RK5. However, the Euclidean norm errors of Newton step of NHAM with RK6 are less than NHAM with RK5 of the same problem. Moreover as shown in Figure 5.67 which proves the convergence of NHAM with RK6 for Bratu problem and the norm error getting smaller in between successive iterates until 7^{th} iteration and finally its change of error becomes very small.

Problem -4 - consider Thermal Explosion nonlinear BVPs defined in section (1.2)

$$u''+\lambda e^{u}=0, \quad x\in(0,1), \qquad (5.5.1.2)$$

With the associated boundary conditions

$$u'(0)=0,\ u(1)=0.$$

The Analytic solution for this problem given by, equation (1.2.12).

Using equations (5.5.1.2) and (3.3), we can create NLSAEs for Thermal Explosion problem

$$F(u_j)=fh^2+u_{j-1}-2u_j+u_{j+1},\ for\ \ j=1,2,...,N \qquad (5.5.1.3)$$

Where $f = -\lambda e^{u_j}$

Equations (5.5.1.3) solve by using NHAM with RK6 steps and take similar inputs of NHAM for the same problem gives the following output:

Table 5.35: NHAM of RK6 Euclidian norm of error, number of iteration and CPU time for Thermal Explosion problem.

Number of iteration(k)	Parameter C	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
9	0.1	2.53 X 10^ -41	4.36 X 10^-40	41.546875

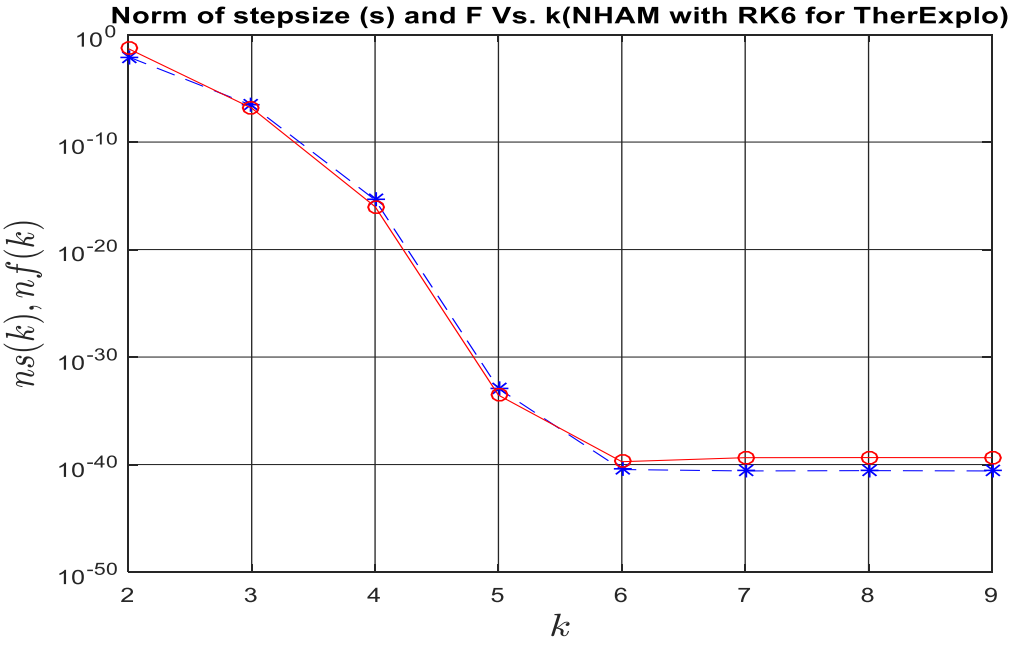

Figure 5.68 NHAM of RK6 Euclidian Norm of Errors versus number of iteration for Thermal Explosion problem.

As shown in Table 5.65 NHAM with RK6 steps converges for Thermal Explosion problem and terminate within 9 iteration for $\lambda = 0.1$. Moreover as shown in Figure 5.68, this proves the convergence of NHAM of RK6 method for Thermal Explosion problem. It displays the norm errors of a function (F) and Newton step (du) are continuously decreasing and reach an error (10^{-41}), which is better accuracy than NHAM with RK5 step however its CPU time taken for computation is bigger than any type of NHAM with RK steps.

CHAPTER SIX

6. Solutions of BVPs Using Optimization Methods

Introduction

Optimization problems are classifying as constrained and unconstrained optimization problems.

Optimization methods diminish the weakness of Newton's and quasi Newton's methods. The weakness of these methods is that accurate initial candidate must be given beforehand to ensure convergence.

In order to find the roots of system of nonlinear equations by Optimization algorithms first, convert the problem into an optimization problem called objective function, and then they look for the desired root as a point that solve the optimization problem.

In book [**20**], a transformation technique from the system of nonlinear equations into an objective functions of optimization problem, and solve it by using steepest decent method are presented.

In this section, we are interested in using this technique to solve our model BVPs by using different methods of optimizations like BFGS, DFP, Newton's and Levenberg–Marquardt.

In addition, to solve nonlinear equations by optimization this result can also be used as a good initial approximation to both Newton's and Broyden Methods.

❖ Systems of Equations and Optimization Problems

Optimization methods are based upon idea that the system $F = 0,\ or\ \ f_i = 0\ \ for$

$i = 1, 2, ..., n$, is solved whenever the function

$$S = f_1^2 + f_2^2 + ... + f_n^2 \qquad (6.0)$$

is minimized, since the minimum clearly occurs when all the f_i are zero. For example, the two-dimensional problem (with a familiar change of notation)

$$f(x, y) = 0 \qquad g(x, y) = 0$$

is equivalent to minimizing this sum

$$S(x,y) = f^2 + g^2$$

Generally, consider system of nonlinear equations $G(x) = 0$; which contains n equations,

$$G(x_1, x_2, ..., x_n) = \begin{bmatrix} g_1(x_1, x_2, ..., x_n) \\ g_2(x_1, x_2, ..., x_n) \\ \vdots \\ g_n(x_1, x_2, ..., x_n) \end{bmatrix} \quad (6.1)$$

And n unknowns of vector x

$$x = [x_1,\ x_2, \ldots x_n]^T \quad (6.2)$$

We can formulate the objective function $F(x)$ by using

$$F(x) = \frac{1}{2} G^T(x) G(x) = g_1^2 + g_2^2 + ... + g_n^2 \quad (6.3)$$

Equivalently we can write by substituting equation (6.1) in (6.3), $F(x)$ is called an Objective function [**70**] of system of equations $G(x)$.

$$F(x) = \left(\left(g_1(x_i)\right)^2 + \left(g_2(x_i)\right)^2 + ... + \left(g_n(x_i)\right)^2 \right) \quad (6.4)$$

The solution methods for unconstrained optimization problems can be broadly classified into gradient-based and non–gradient-based search methods. Once the search direction is identified, we needs to evaluate how much to move in that direction so as to minimize the objective function. The efficiency of solution methods can be gauged by three criteria:

- Number of function evaluations.
- Computational time.
- Rate of convergence.

If the function is continuously differentiable and the derivative should be evaluate then the optimization problem can be express as a root-finding problem [**70**]. We are using the golden section method, for solving the one-dimensional problem. The search direction is

guide by the function evaluations as well as the search directions computed from earlier iterations.

❖ **Golden section search**

The golden section search is the counter part of bisection used in finding roots of equations [**71**] . Suppose that the minimum of $f(x)$ has been bracketed in the interval (a,b) of length h. To telescope the interval, we evaluate the function at $x_1 = b - Rh$ and $x_2 = a + Rh$, as shown in Fig. 6.69 (a).The constant R will be determined shortly.

If $f_1 > f_2$ as indicated in the figure, the minimum lies in (x_1, b), else in (a, x_2).

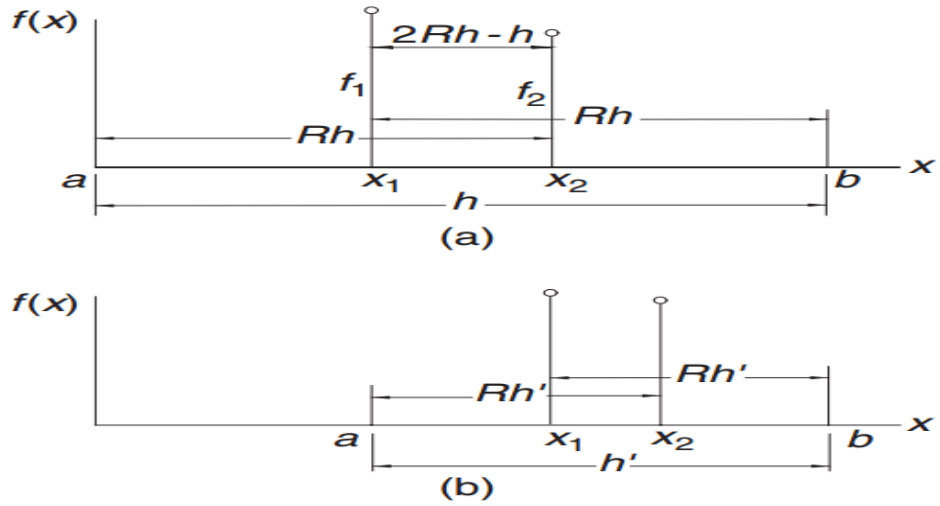

Figure 6.69. Golden section telescoping.

As illustrated in Fig.6.69 (b), to carry out the next telescoping operation we evaluate the function at $x_2 = a + Rh$ and repeat the process.

Referring to Fig.6.69 (a), we note that $x_2 - x_1 = 2Rh - h.$ the same distance in Fig. 6.69 (b) is $x_1 - a = h' - Rh'$. Equating the two, we get

$$2Rh - h = h' - Rh' \qquad (6.5)$$

Substituting $h' = Rh$ and cancelling h yields

$$2R - 1 = R(1 - R) \qquad (6.6)$$

The solution of which is the golden ratio:

$$R = \frac{-1 + \sqrt{5}}{2} = 0.618033989... \qquad (6.7)$$

Note that each telescoping decreases the interval containing the minimum by the factor R, which is not as good as the factor of 0.5 in bisection.

The convergence rate of this method is linear. The interval reduction step is repeated until the width of the bracket falls below the tolerance limit [**71**].

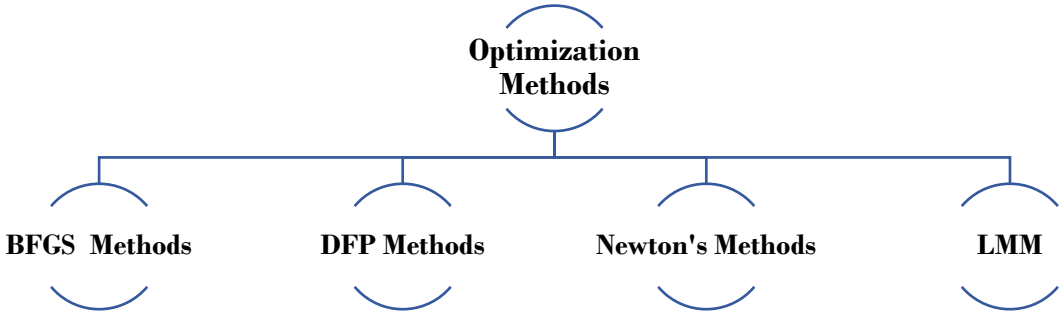

Figure 70: A Schematic Diagram of Optimization Methods

6.1. Newton's Method for solving BVPs

The oldest second order method for minimizing a nonlinear multivariable function in $\Re^n$ is Newton's method. The motivation behind Newton's method is identical to the steepest descent method. The Euclidean norm, however, does not consider the curvature of the surface. Hence, it motivates the definition of a different norm or a metric of the surface. Thus, we pose the problem as finding the direction s that minimizes

$$\nabla f^T s = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} s_i, \qquad (6.1.0)$$

Subject to the condition that

$$s^T Q s = 1 \qquad (6.1.1)$$

The solution of this problem is provide by Newton direction

$$s = -Q^{-1} \nabla f \ , \qquad (6.1.2)$$

Where Q is the Hessian of the objective function, the general form of the update equation of Newton's method for minimizing a function in $\Re^n$ is

$$x_{k+1} = x_k - \alpha_{k+1} Q_k^{-1} \nabla f(x_k). \qquad (6.1.3)$$

Where α_{k+1} is determined by minimizing f along the Newton direction. For quadratic functions it can be shown that the update equation reaches the optimum solution in one step with $\alpha = 1$

$$x^* = x_0 - \left[Q(x_0)\right]^{-1}\left[\nabla f(x_0)\right], \qquad (6.1.4)$$

Newton's method can also show to have a quadratic rate of convergence (see for example [**72**,**73**] , but the serious disadvantages of the method are the need to evaluate the Hessian Q and then solve the system of equations

$$Qs = -\nabla f\ , \qquad (6.1.5)$$

to obtain the direction vector s. For every iteration (if Q is non-sparse), Newton's method involves the calculation of $n(n+1)/2$ elements of the symmetric Q matrix, and n^3 operations for obtaining s from the solution of equation (6.1.5).

6.1.1. Application of Newton's Method

Newton's method is the optimization methods, for which we wrote Algorithm (6.1) to approximate the solution of our mechanics problems defined in section (1.2) .

- **Algorithm (6.1): Solves nonlinear second order BVPs by using Newton's Method.**

INPUT : n, nx - no of variable & no of nodes
x , Δx - guess solution vector & required for gradient computation
ε1, ε2 - constant used for terminating the algorithm
falph_prev - function value at first/previous iteration
deriv - gradient vector
search - search direction(s_i)

STEP-1 start from 1st to last iteration do Step 2 to **6**

STEP-2 compute function, gradient and Hessian ($f, \nabla f, H$) by using initial guess

f_prev = func_multivar()
deriv = grad_vec()
sec_deriv = HESSIAN()

STEP-3 compute search direction

$$s_i = -[H]^{-1}\nabla f$$

update solution vector

$$x_{i+1} = x_i + s_i$$

STEP-4 check $|falpha - falpha\_prev| < \varepsilon_1$ *or* $\|deriv\| < \varepsilon_2$ is satisfied go to step-5 else go to STEP-1

STEP-5 Display Numerical and graphical output

Based on Algorithm (6.1) we wrote a MATLAB program that solve problems defined in section (1.2).

- **Numerical and Graphical output by using Newton's Method**

The numerical method, second order Newton method is apply and tested on the following nonlinear BVPs that are define in section (1.2), the numerical result and norm error are displayed by using MATLAB program.

Problem-1- consider Bratu 1-D nonlinear problem defined in section (1.2)

$$u''(x)+\lambda e^{u(x)}=0 \qquad (6.1.1.0)$$

Subject to the dirichlet boundary conditions:

$$u(0)=u(1)=0 \qquad (6.1.1.1)$$

The Analytic solution for this problem given by, equation (1.2.3).

Using equation (6.1.1.0) and (3.3), we can create NLSAEs for Bratu problem

$$F(u_j)=fh^2+u_{j-1}-2u_j+u_{j+1},\ for\ \ j=1,2,...,N \qquad (6.1.1.2)$$

Eq. (6.1.1.2) solves by Newton method by using a MATLAB program and similar input of pure Newton method for the same problem gives the following result:

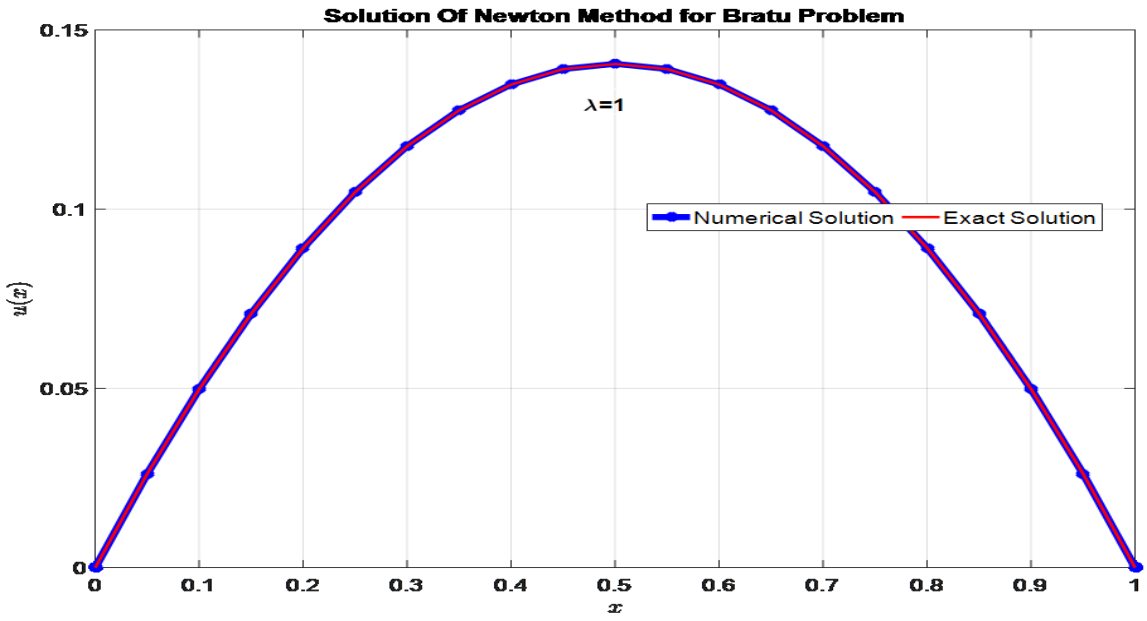

Figure 6.71 Comparison between the approximated solution and exact solution of Bratu problem using Newton's Method.

Table 6.36: Newton's method Euclidian norm of error, number of iteration and CPU time for Bratu problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
4	1	7.77×10^{-10}	1.73×10^{-10}	0.09375

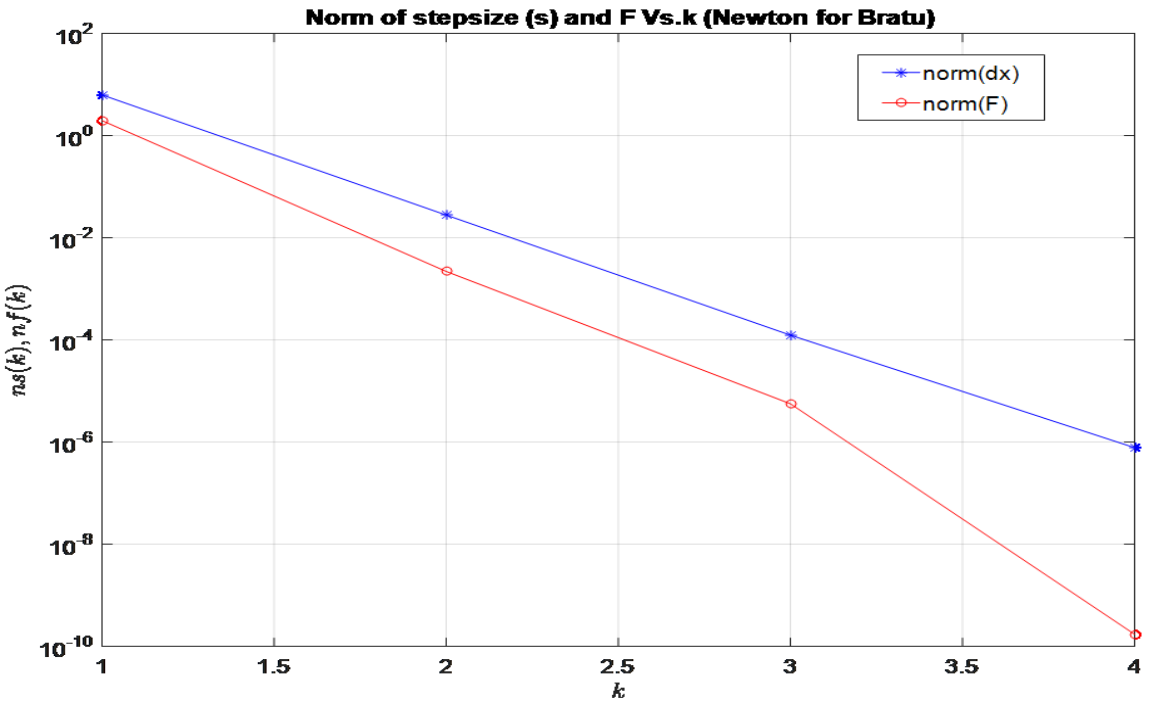

Figure 6.72 Newton's Method Euclidian Norm of Errors versus number of iteration for Bratu problem.

Fig.6.71 cites a comparison between the Newton method solutions and exact solution. From this comparison, one can see a good agreement between the exact solution and the **Newton's** (approximate) solution.

As shown in Table 6.36, **Newton's Method** converges for Bratu problem and terminate within 4 iteration for parameter value lambda=1. The CPU time also indicates **Newton's** method takes 0.0975 for computation. It is fastest method .Moreover as shown in Fig.5.72, which proves the convergence of Newton's Method for Bratu problem and the norm errors of a Newton step (du) and function are continuously decreasing with in four iterations.

Problem-2- consider Troesch's 1-D nonlinear problem defined in section (1.2)

$$u'' = \lambda \sinh(\lambda u) \qquad (6.1.1.3)$$

Subject to the dirichlet boundary conditions:

$$u(0)=0 \ \ and \ \ u(1)=1. \qquad (6.1.1.4)$$

The theoretical solution for this problem given by, equation (1.2.7).

Using equation (6.1.1.3) and (3.3), we can create NLSAEs for Troeschs problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1,2,...,N$$

Where $f = \lambda \sinh(\lambda u(j))$

The above NLSAEs solve by using **optimization technique Newton's Method and by taking similar inputs of pure Newton method for the same problem gives:**

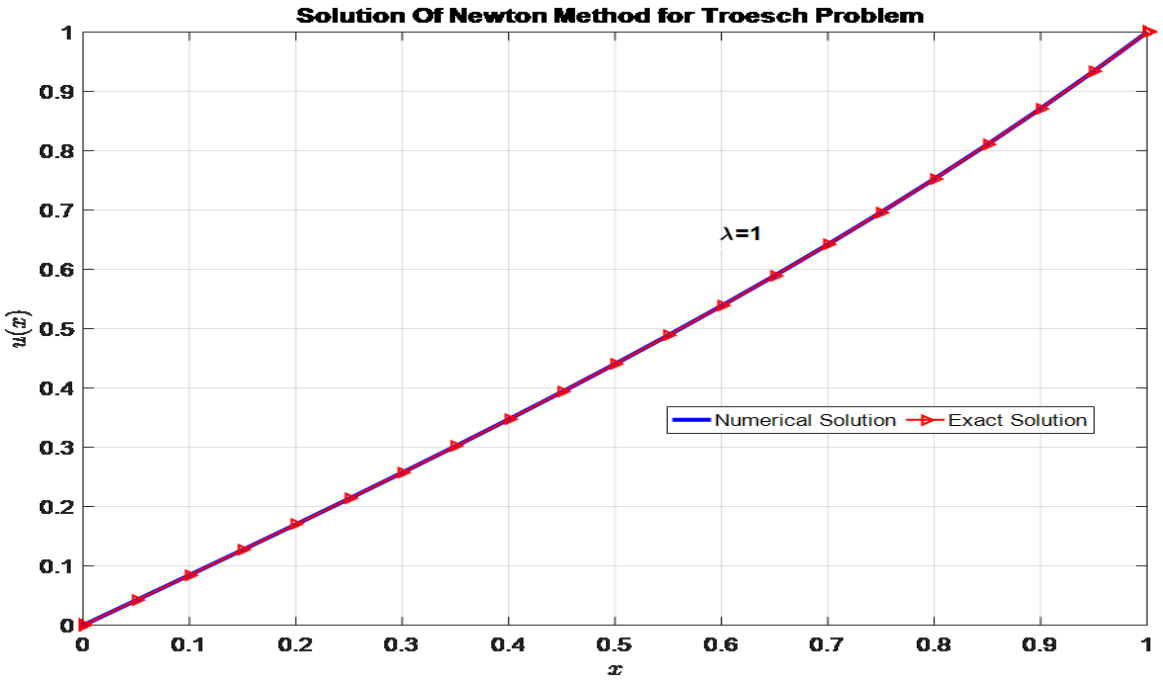

Figure 6.73 Comparison between the approximated solution and exact solution of Troesch problem using Newton's Method.

Table 6.37: Newton method norm error, iteration and CPU time for Troesch's problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
3	1	1.76×10^{-6}	6.61×10^{-10}	0.0625

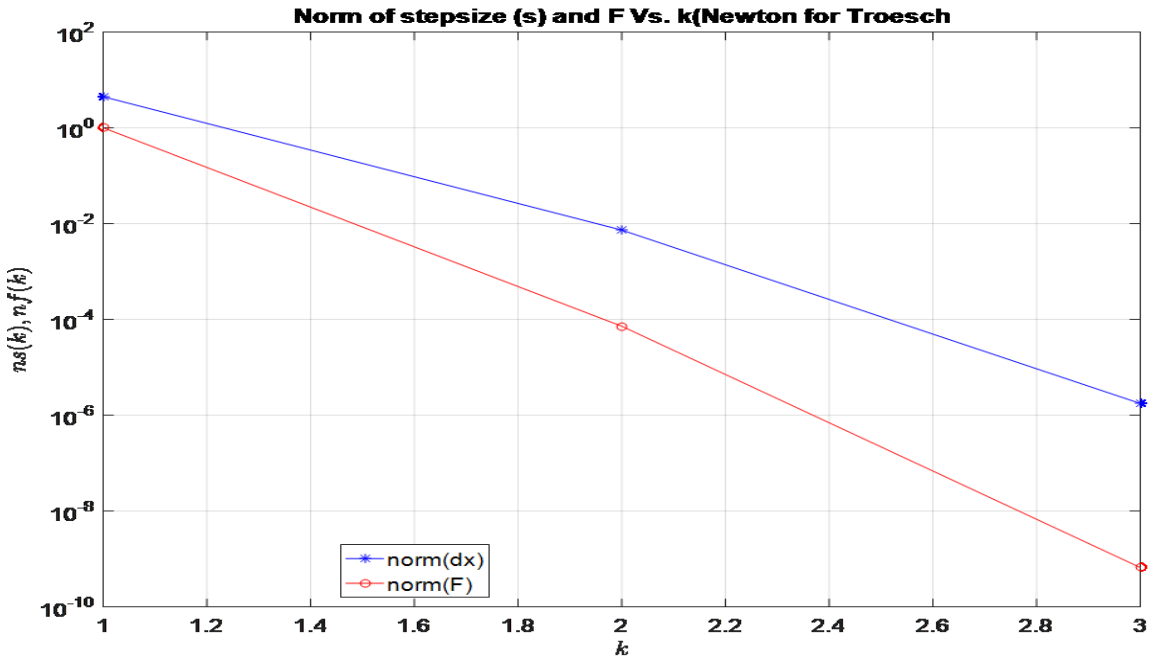

Figure 6.74 Newton's Method Euclidian Norm of Errors versus number of iteration for Troesch problem.

As shown in Fig.6.73, a comparison between the approximate solution obtained by **Newton's Method** and exact solution. From this comparison, one can see a good agreement between the exact solution and the **Newton's Method** (approximate) solution for Troesch's problem.

As shown in Table 6.37, **Newton's Method** converges for Troesch problem and terminate within 3 iteration for parameter value lambda=1. The CPU time shows, it is the fastest method .Moreover as shown in Fig.6.74, which proves the convergence of **Newton's Method** for Troesch problem and the norm errors of a Newton step (du) and function are continuously decreasing and within three iteration it converges to a solution.

6.2. Levenberg-Marquardt Method

The advantage of the steepest descent method is that it reaches closer to the minimum of the function in a few iterations even when the starting guess is far away from the optimum. However, the method shows sluggishness near the optimum point.

The Levenberg–Marquardt method is a kind of hybrid method that combines the strength of both the steepest descent and Newton's methods. The search direction in this method given by

$$S_i = -\left[H + \lambda I\right]^{-1} \nabla f\left(x_i\right) \qquad (6.2.0)$$

Where, I is an identity matrix and λ is a scalar that is set to a high value at the start of the algorithm. The value of λ is altered during every iteration depending on whether the function value is decreasing or not. If the function value decreases in the iteration, λ decreases by a factor (less weightage on steepest descent direction). On the other hand, if the function value increases in the iteration, λ increases by a factor (more weightage on steepest descent direction) [**74**].

6.2.1. Application of Levenberg-Marquardt Method (LMM)

To approximate the solution of mechanics problems define in section (1.2), we wrote an algorithm (6.2) for Levenberg-Marquardt method.

- **Algorithm (6.2): Solves nonlinear second order BVPs by using LM Method.**

INPUT : n, nx - no of variable & no of nodes
x , Δx - guess solution vector & required for gradient computation
ε1, ε2 - constant used for terminating the algorithm
falph_prev - function value at first/previous iteration
deriv - gradient vector
search - search direction(-deriv

STEP-1 start from 1st to last iteration do Step 2 to **7**

STEP-2 compute function, gradient and Hessian ($f, \nabla f, H$) by using initial guess

f_prev = func_multivar()

deriv = grad_vec()

sec_deriv = HESSIAN()

STEP-3 compute search direction

$$s_i = -[H + \lambda I]^{-1} \nabla f$$

update solution vector

$$x_{i+1} = x_i + s_i$$

STEP-4 when $f(x_{i+1}) < f(x_i)$ satisfied change $\lambda = \lambda / 2$ else $\lambda = 2\lambda$

STEP-5 check $|falpha - falpha\_prev| < \varepsilon_1$ *or* $\|deriv\| < \varepsilon_2$ is satisfied go to step-5 else go to STEP-1

STEP-6 Display Numerical and graphical output

- **Numerical and Graphical output by using Levenberg-Marquardt (LM) Method**

The numerical method, LM method is applied and tested on BVPs that are define in section (1.2) and the numerical result are displayed by using MATLAB program.

Problem-1- consider Bratu 1-D nonlinear problem defined in section (1.2)

$$u''(x) + \lambda e^{u(x)} = 0 \qquad (6.2.1.0)$$

Subject to the boundary conditions:

$$u(0) = u(1) = 0 \qquad (6.2.1.1)$$

The Analytic solution for this problem given by, equation (1.2.3).

Using equation (6.2.1.0) and (3.3), we can create NLSAEs for Bratu problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1, 2, \ldots, N$$

Where $f = -\lambda e^{u_j}$

The above NLSAEs solve by using Levenberg–Marquardt (LM) method and similar inputs of Newton method for the same problem we obtain the following result:

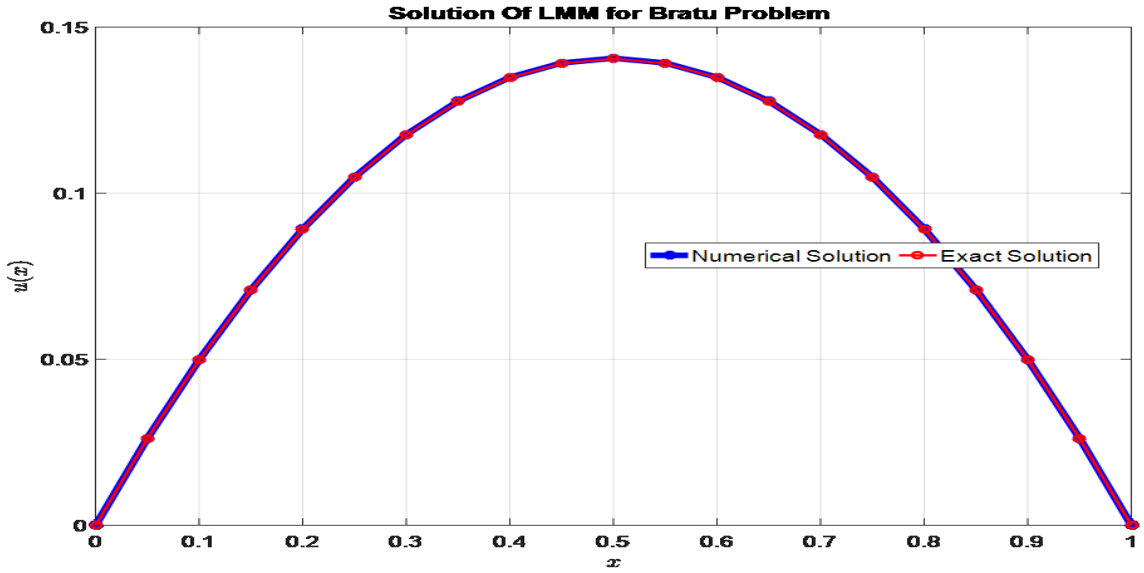

Figure 6.75 Comparison between the approximated solution and exact solution of Bratu problem using LM method.

Table 6.38: LM method Euclidian norm of error, number of iteration and CPU time for Bratu problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
25	1	5.33e-6	1.508e - 8	0.78125

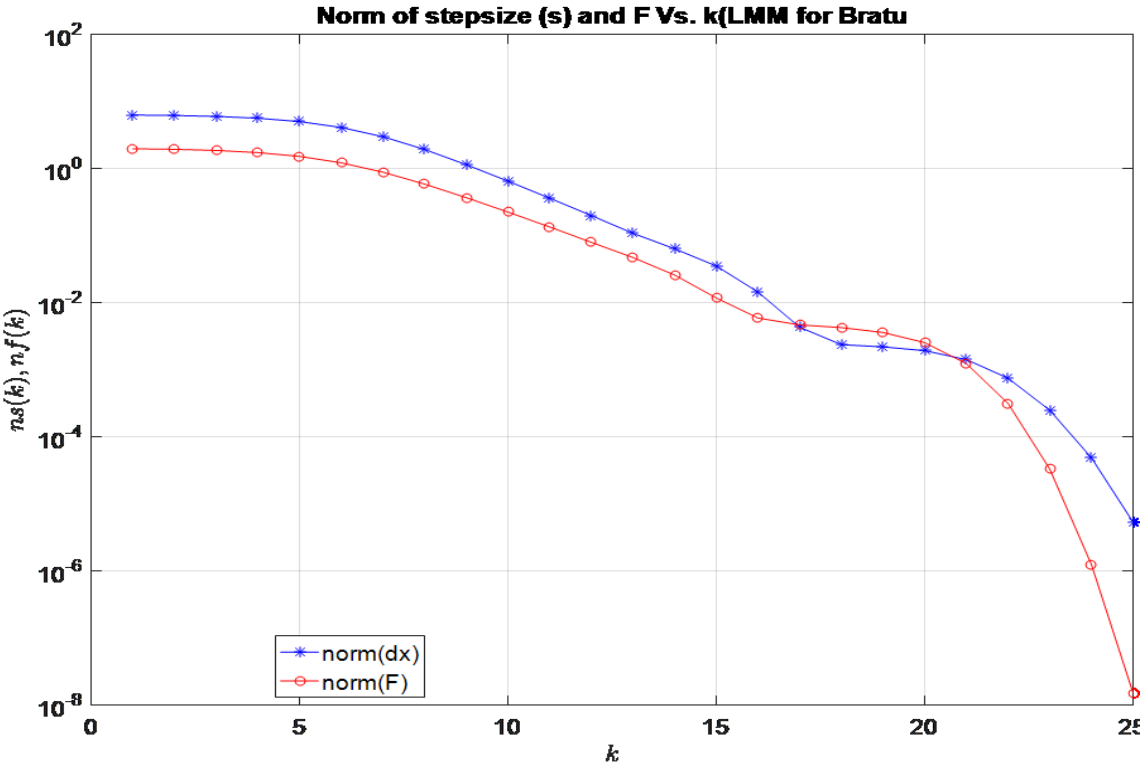

Figure 6.76 LMM Euclidian Norm of Errors versus number of iteration for Bratu problem.

Fig.6.75 cites a comparison between the approximate solution obtained by **LM** method and exact solution. From this comparison, one can see a good agreement between the exact solution and the **LM** (approximate) method solution.

As shown in Table 6.38, **LM method** converges for Bratu problem and terminate within 25 iteration . The CPU time also indicates LM method takes more time than Newton method with this problem. The Euclidean norm error of the function shows LM method has more error than that of Newton's Euclidean norm error .Moreover as shown in Fig. 6.76 that proves the convergence of **LM method** for Bratu problem and the norm errors of a Newton step (du) and function are continuously decreasing below expected error tolerance.

Problem-2- consider Troesch's 1-D nonlinear problem defined in section (1.2)

$$u'' = \lambda \sinh(\lambda u) \qquad (6.2.1.2)$$

Subject to the boundary conditions:

$$u(0) = 0 \ \ and \ \ u(1) = 1. \qquad (6.2.1.3)$$

The theoretical solution for this problem given by, equation (1.2.7).

Using equation (6.2.1.2) and (3.3), we can create NLSAEs for Troeschs problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1,2,...,N$$

Where $f = \lambda \sinh(\lambda u(j))$

The above NLSAEs solve by using **LM method and take similar inputs of Newton method for the same problem gives the following result:**

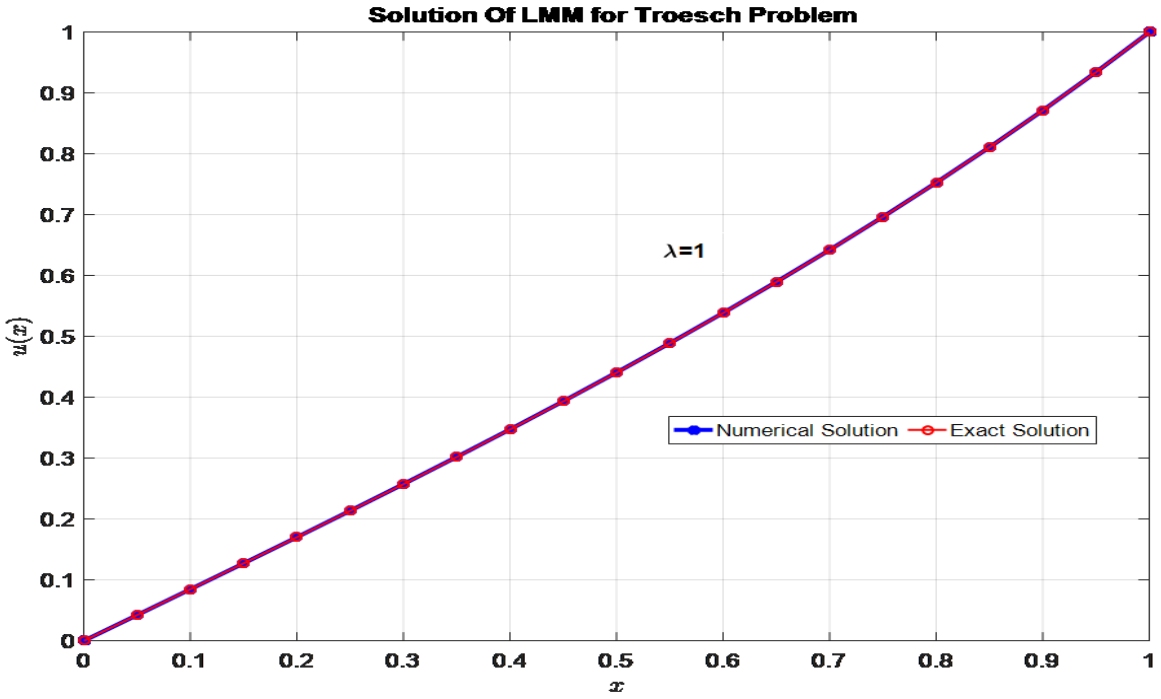

Figure 6.77 Comparison between the approximated solution and exact solution of Troesch problem using LM method.

Table 6.39: LM method Euclidian norm of error, number of iteration and CPU time for Troesch's problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of fuction (F)	CPU time
24	1	1.173e -5	4.564e-8	1.375

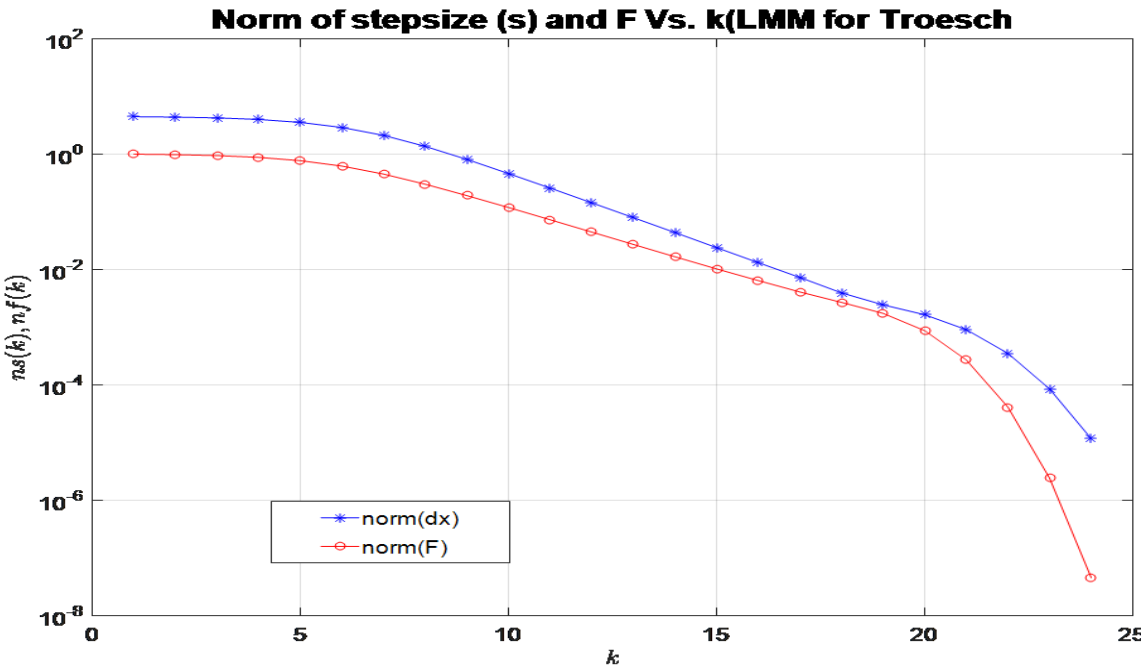

Figure 6.78 LM method Euclidian Norm of Errors versus number of iteration for Troesch problem.

As shown in Fig.6.77, a comparison between the approximate solution obtained by **LM** method and exact solution. This shows a good agreement between the exact solution and the LM (approximate) method solution for Troesch's problem.

As shown in Table 6.39, **LM** method converges for Troesch problem and terminates within 24 iterations. The CPU time also indicates LM method takes more time than Newton method. The Euclidean norm error of the Newton step shows LM method has more Euclidean norm error than Newton method .Moreover as shown in Fig.6.79, which proves the convergence of **LM** method for Troesch problem and the norm errors of the function are continuously decreasing below expected error tolerance.

6.3. Quasi-Newton's or Variable Metric Algorithms

Introduction

Consider the Taylor series expansion of the gradient of f around x_{k+1}

$$\nabla f\left(x_{k+1}\right) \cong \nabla f\left(x_k\right) + Q\left(x_{k+1} - x_k\right), \qquad (6.3.0)$$

Where Q is the actual Hessian of the function f. Assuming $A_k \left(A_k \equiv A\left(x_k\right)\right)$ to approximate the Hessian at the k^{th} iteration, we may write equation (6.3.0) in a more compact form as

$$y_k = A_k p_k \ , \qquad (6.3.1)$$

Where

$$y_k = \nabla f\left(x_{k+1}\right) - \nabla f\left(x_k\right), \quad and \quad p_k = x_{k+1} - x_k, \qquad (6.3.2)$$

Similarly, the solution of Eq. (6.3.2) for p_k be written as

$$B_{k+1} y_k = p_k, \qquad (6.3.3)$$

With B_{k+1} being an approximate inverse of the Hessian (Q), if B_{k+1} is to behave eventually as Q^{-1} then $B_{k+1} A_k = I$. Equation (6.3.3), known as the quasi-Newton or the secant relation.

6.3.1. Davidon-Fletcher-Powell's (DFP) Method

Rank-two updates for the inverse Hessian approximation may generally written as

$$B_{k+1} = \left[B_k - \frac{B_k y_k y_k^T B_k}{y_k^T B_k y_k} + \theta_k v_k v_k^T \right] \rho_k + \frac{p_k p_k^T}{p_k^T y_k}, \qquad (6.3.1.0)$$

Where

$$v_k = \left(y_k^T B_k y_k \right)^{\frac{1}{2}} \left[\frac{p_k}{p_k^T y_k} - \frac{B_k y_k}{y_k^T B_k y_k} \right], \qquad (6.3.1.1)$$

θ_k and ρ_k are scalar parameters that are chosen appropriately. If we set $\theta_k = 0 \ and \ \rho_k = 1$ for all k we obtain the ***Davidon-Fletcher-Powell's*** **(DFP**) update formula [**75**] which is given as

$$B_{k+1} = \left[B_k - \frac{B_k y_k y_k^T B_k}{y_k^T B_k y_k} + \theta_k v_k v_k^T \right] + \frac{p_k p_k^T}{p_k^T y_k}, \qquad (6.3.1.2)$$

However, the step length must guarantee a reduction in the function value, and must be such that $p_k^T y_k > 0$ in order to maintain positive definiteness of B_k . The performance of the algorithm, however shows to deteriorate as the accuracy of the line search decreases

[**76**] . This has led to the introduction of another update formula developed simultaneously by BFGS formula as shown in the next section.

6.3.1.1. Application of Davidon-Fletcher-Powell's (DFP) Method

To approximate the solution of mechanics problems define in section (1.2), we wrote an algorithm (6.3) for DFP method.

- **Algorithm (6.3): Solves nonlinear second order BVPs by using DFP Method.**

INPUT : n, nx - no of variable & no of nodes
x , Δx - guess solution & required for gradient computation
ε1, ε2 - constant used for terminating the algorithm
falph_prev - function value at first/previous iteration
deriv-prev - gradient vector
search - search direction(-deriv

[A] - initialize to identity matrix

STEP-1 call function func_multivar () for falpha_prev at x

falpha_prev = func_multivar(x)

STEP-2 start from 1st to last iteration do Step 3 to **5**

STEP-3 for the 1st iteration do step-3 to step-4 else

compute gradient vector deriv by calling gradient computation function

deriv_prev = grad_vec( )

Compute search direction by

search = -deriv-prev

Determine alpha and falpha by using Golden search method

[alpha, falpha] = golden_funct1( )

STEP-4 $falpha - falpha\_prev < epselon1$,is satisfied go to last else go to step-5

STEP-5 compute Δx and ∇g for update of Hessian matrix approximation

Δx = (alpha*search)

$$A_{i+1} = A_i + \frac{\Delta x \Delta x^T}{\Delta x^T \nabla g} + \frac{A_i \nabla g \nabla g^T A_i}{\nabla f(x_i)^T A_i \nabla g}$$

Where $s_{i+1} = -[A_{i+1}]^{-1} \nabla f(x_i)$

STEP-6 Determine α by using Golden search method

[alpha,falpha] = golden_funct1()

Minimize $f(x_{i+1})$ and update solution x_{i+1}

$$x_{i+2} = x_{i+1} + \alpha s_{i+1}$$

STEP-7 check $|falpha - falpha\_prev| < \varepsilon_1$ *or* $\|deriv\| < \varepsilon_2$ is satisfied go to step-5 else go to STEP-1

STEP-8 Display Numerical and graphical output

- **Numerical and Graphical output by using DFP Method**

DFP method is apply and tested on the following nonlinear BVPs that are define in section (1.2) and the numerical result and norm error are display by using MATLAB program.

Problem-1- consider Bratu 1-D nonlinear problem defined in section (1.2)

$$u''(x) + \lambda e^{u(x)} = 0 \qquad (6.3.1.1)$$

Subject to the dirichlet boundary conditions:

$$u(0) = u(1) = 0 \qquad (6.3.1.2)$$

The Analytic solution for this problem given by, equation (1.2.3).

Using equation (6.3.1.1) and (3.3), we can create NLSAEs for Bratu problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1, 2, \ldots, N$$

Where $f = -\lambda e^{u_j}$

The above NLSAEs solve by using DFP method and similar inputs of Newton method for the same problem gives the following result:

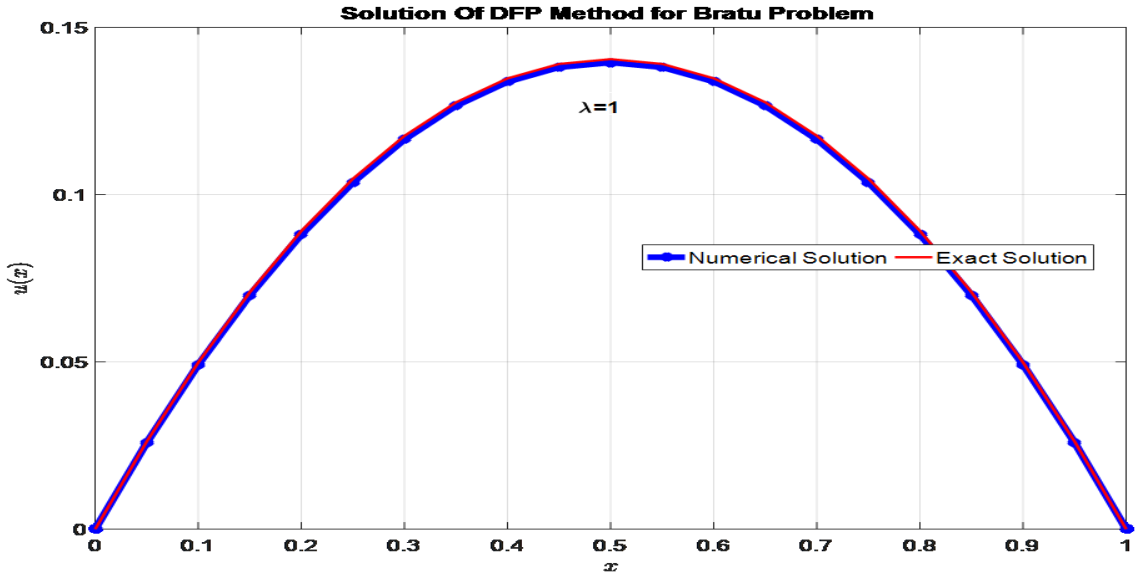

Figure 6.79 Comparison between the approximated solution and exact solution of Bratu problem using DFP method.

Table 6.40: DFP method Euclidian norm of error, number of iteration and CPU time for Bratu problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of fuction (F)	CPU time
78	1	7.051e-3	9.204e – 8	0.421875

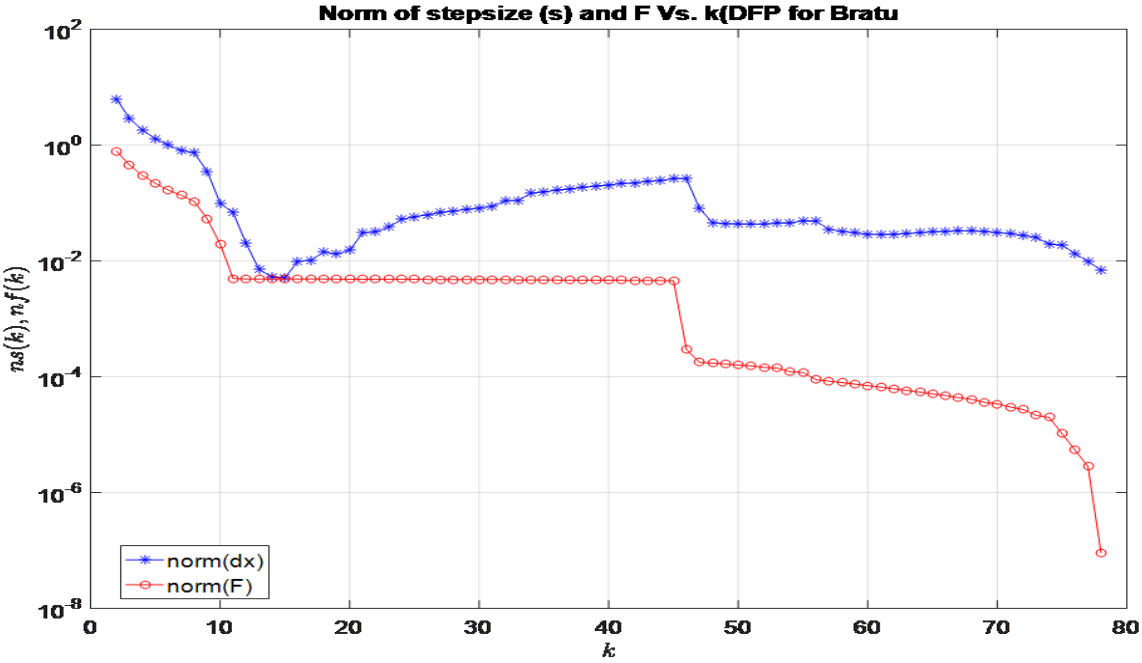

Figure 6.80 DFP Method Euclidian Norm of Errors versus number of iteration.

Fig.6.79 cites a comparison between the approximate solution obtained by **DFP** method and exact solution. From this comparison, one can see an agreement between the exact solution and the **DFP** (approximate) solution.

As shown in Table 6.40 **DFP Method** converges for Bratu problem and terminate within 78 iteration for parameter value λ=1 , which is 20 times more than Newton method. The CPU time also indicates DFP method takes 4.5 times more than Newton method with the same problem. Moreover as shown in Fig.6.80, which proves the convergence of DFP **Method** for Bratu problem and the norm errors of a Newton step (du) are not continuously decreasing. This shows that at some point of iteration, norm of Newton steps are increasing rather than decreasing, due to approximation of hessian matrix.

Problem-2- consider Troesch's 1-D nonlinear problem defined in section (1.2)

$$u'' = \lambda \sinh(\lambda u) \qquad (6.3.1.3)$$

Subject to the dirichlet boundary conditions:

$$u(0) = 0 \ \ and \ \ u(1) = 1. \qquad (6.3.1.4)$$

The theoretical solution for this problem given by, equation (1.2.7).

Using equation (6.3.1.3) and (3.3), we can create NLSAEs for Troeschs problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1,2,...,N$$

Where $f = \lambda \sinh(\lambda u(j))$

The above NLSAEs solve by using **DFP Method and take similar inputs of Newton method for the same problem, give the following output:**

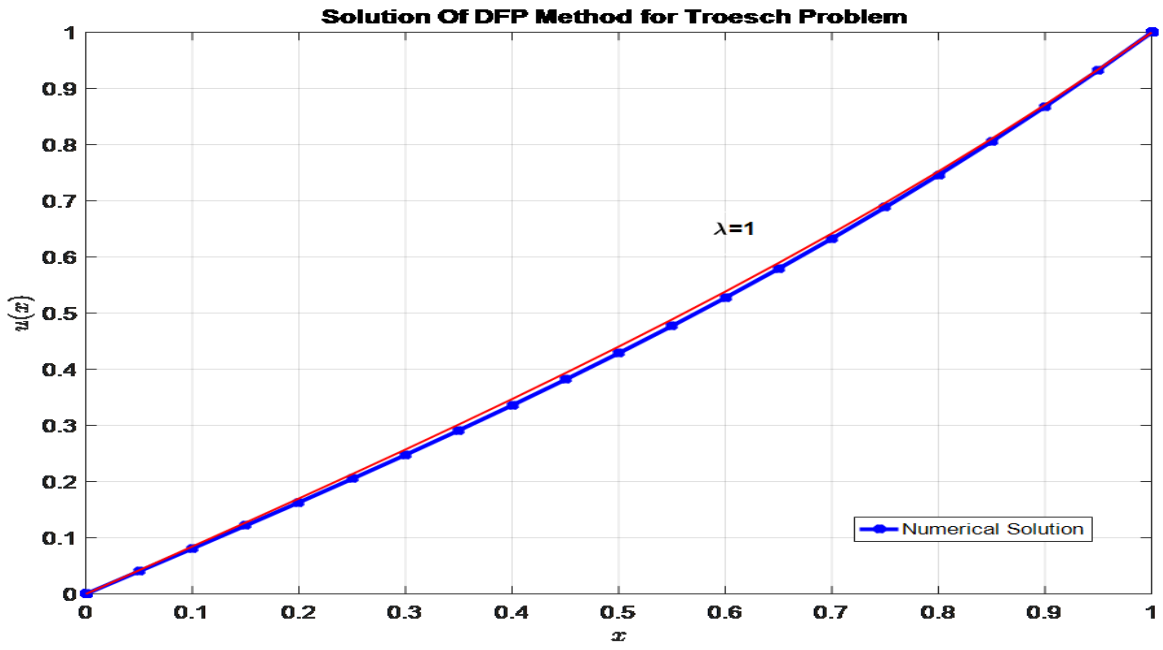

Figure 6.81 Comparison between the approximated solution and exact solution of Troesch problem using DFP Method.

Table 6.41: DFP method Euclidian norm of error, number of iteration and CPU time for Troesch's problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
107	1	1.63e -3	1.05e-6	0.34375

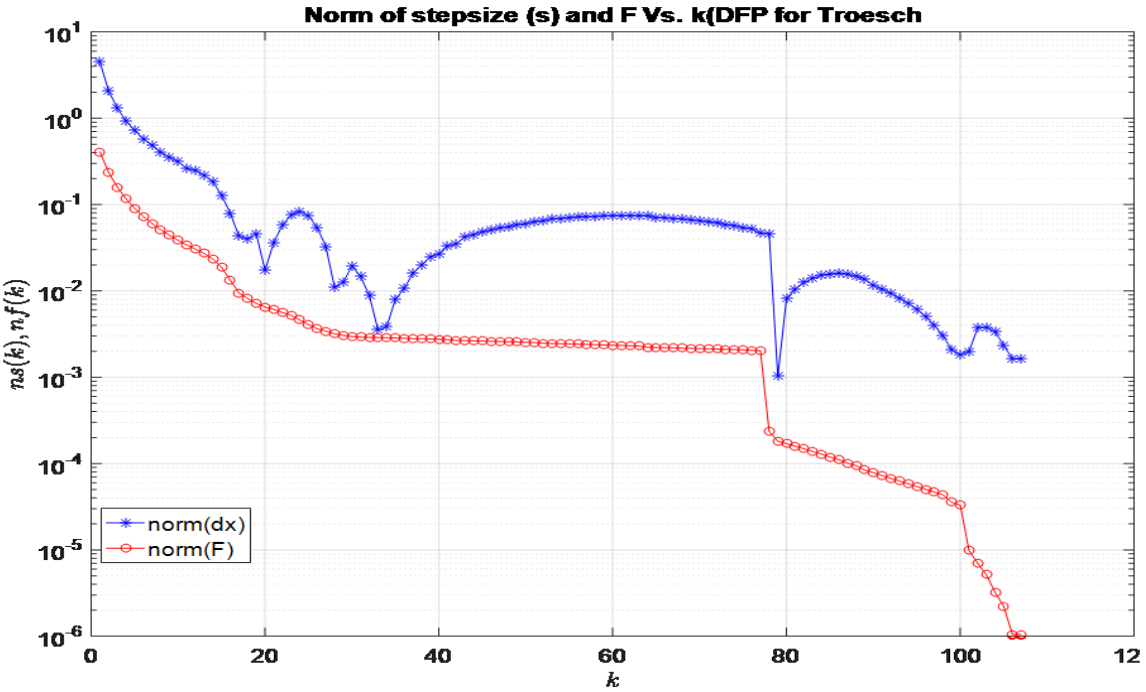

Figure 6.82 DFP Method Euclidian Norm of Errors versus number of iteration.

As shown in Fig.6.81, a comparison between the approximate solution obtained by **DFP Method** and exact solution. From this comparison, one can see an agreement between the exact solution and the **DFP Method** (approximate) solution for Troesch's problem.

As shown in Table 6.41, DFP **Method** converges for Troesch's problem and terminate within 107 iteration for parameter value λ=1, which is 27 times more than Newton method. The CPU time also indicates DFP method takes more time than both Newton and LM method and the Euclidean norm error of function also bigger than both Newton and LM method with the same problem. Moreover as shown in Fig.6.82, which proves the convergence of DFP **Method** for Troesch problem and the norm errors of a Newton step (du) are not continuously decreasing. This shows that, at some point of iteration, norm of Newton steps are increasing rather than decreasing, due to approximation of hessian matrix.

6.3.2. Broyden, Fletcher, Goldfarb, and Shannon (BFGS) Method

This formula can be obtain by putting $\theta_k = 1$ and $\rho_k = 1$ in Eq. (6.3.1.0) which reduces to

$$B_{k+1} = B_k + \left[1 + \frac{y_k^T B_k y_k}{p_k^T y_k}\right]\frac{p_k p_k^T}{p_k^T y_k} - \frac{p_k y_k^T B_k}{p_k^T y_k} - \frac{B_k y_k p_k^T}{p_k^T y_k}. \qquad (6.3.2.0)$$

Equation (6.3.2.0) can also be write in a more compact manner as

$$B_{k+1} = \left[I - \frac{p_k y_k^T}{p_k^T y_k} \right] B_k \left[I - \frac{y_k p_k^T}{p_k^T y_k} \right] + \frac{p_k y_k^T}{p_k^T y_k}. \qquad (6.3.2.1)$$

Using $A_{k+1} = B_{k+1}^{-1}$ and $A_k = B_k^{-1}$ we can invert the above formula to arrive at an update for the Hessian approximations. This update formula reduces to

$$A_{k+1} = A_k - \frac{A_k p_k p_k^T A_k}{p_k^T A_k p_k} + \frac{y_k y_k^T}{y_k^T p_k}. \qquad (6.3.2.2)$$

Which is the analog of the DFP formula (6.3.1.2) with B_k replaced by A_k, p_k and y_k interchanged. Conversely, if the inverse Hessian B_k is update by the DFP formula then the Hessian A_k is update according to an analog of the DFP formula.

Numerical experiments with BFGS algorithm [**76**] suggest that it is superior to all known variable-metric algorithms. We will illustrate its use by solving different nonlinear BVPs.

6.3.2.1. Application of Broyden, Fletcher, Goldfarb, and Shannon Method

To approximate the solution of mechanics problems define in section (1.2), we wrote an algorithm (6.4) for BFGS method.

➢ **Algorithm (6.4): Solves nonlinear BVPs by using BFGS Method.**

INPUT : n, nx - no of variable & no of nodes

x , Δx - guess solution vector & required for gradient computation

ε1, ε2 - constant used for terminating the algorithm

falph_prev - function value at first/previous iteration

deriv-prev - gradient vector

search - search direction(-deriv

[A] - initialize to identity matrix

STEP-1 call function func_multivar () for falpha_prev at x

falpha_prev = func_multivar(x)

STEP-2 start from 1st to last iteration do Step 3 to **5**

STEP-3 for the 1st iteration do step-3 to step-4 else

compute gradient vector deriv by calling gradient computation function

deriv_prev = grad_vec( )

Compute search direction by

search = -deriv-prev

Determine alpha and falpha by using Golden search method

[alpha, falpha] = golden_funct1( )

STEP-4 $falpha - falpha\_prev < tol$,is satisfied go to step-8 else go to step-5

STEP-5 compute Δx and ∇g for update of Hessian matrix approximation

Δx = (alpha*search)

$$A_{i+1} = A_i + \frac{g \nabla g^T}{\nabla g^T \Delta x} + \frac{\nabla f(x_i) \nabla f(x_i)^T}{\nabla f(x_i)^T}$$

Where $s_{i+1} = -[A_{i+1}]^{-1} \nabla f(x_i)$

STEP-6 Determine α by using Golden search method

[alpha,falpha] = golden_funct1()

Minimize $f(x_{i+1})$ and update solution x_{i+1}

$$x_{i+2} = x_{i+1} + \alpha s_{i+1}$$

STEP-7 check $|falpha - falpha\_prev| < \varepsilon_1$ *or* $\|deriv\| < \varepsilon_2$ is satisfied go step-5 else go to - 1

STEP-8 Display Numerical and graphical output

➢ **Numerical and Graphical output by using BFGS Method**

BFGS method is apply and tested on the following BVPs that are define in section (1.2) and the numerical result are displayed by using MATLAB program.

Problem-1- consider Bratu 1-D nonlinear problem defined in section (1.2)

$$u''(x) + \lambda e^{u(x)} = 0 \qquad (6.3..2.1)$$

Subject to the dirchlet boundary conditions:

$$u(0) = u(1) = 0 \qquad (6.3.2.2)$$

The Analytic solution for this problem given by, equation (1.2.3).

Using equation (6.3.2.1) and (3.3), we can create NLSAEs for Bratu problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1}, \ for \ \ j = 1, 2, \ldots, N$$

Where $f = -\lambda e^{u_j}$

The above NLSAEs solve by using BFGS method and inputs similar with DFP method for the same problem gives the following result:

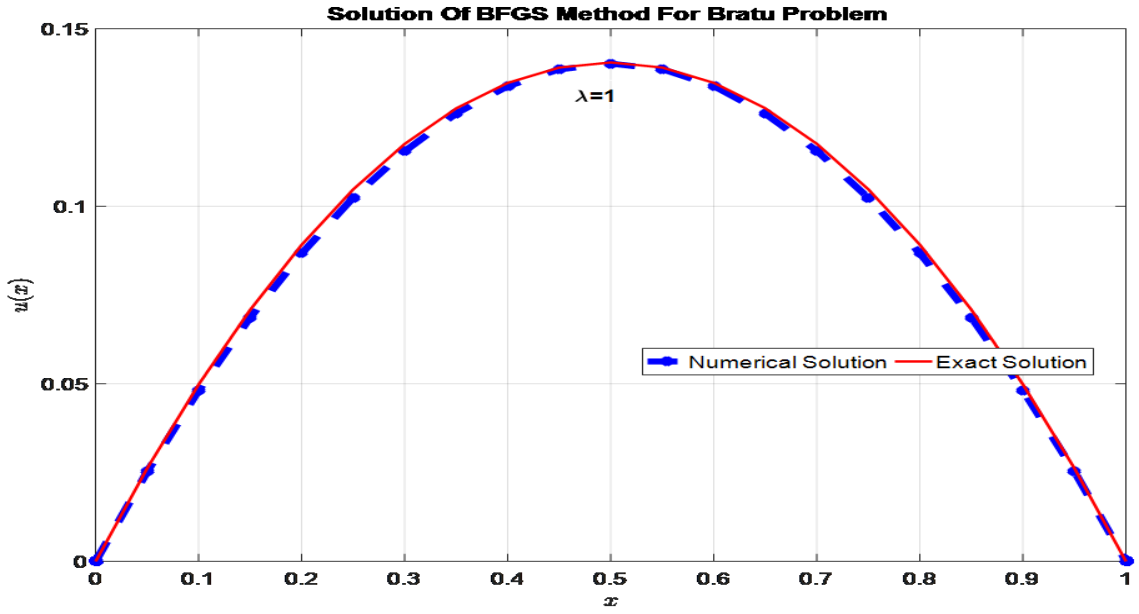

Figure 6.83 Comparison between the approximated solution and exact solution of Bratu problem using BFGS Method .

Table 6.42: BFGS method Euclidian norm of error, number of iteration and CPU time for Bratu problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of function (F)	CPU time
21	1	3.68e-3	1.23e - 6	0.125

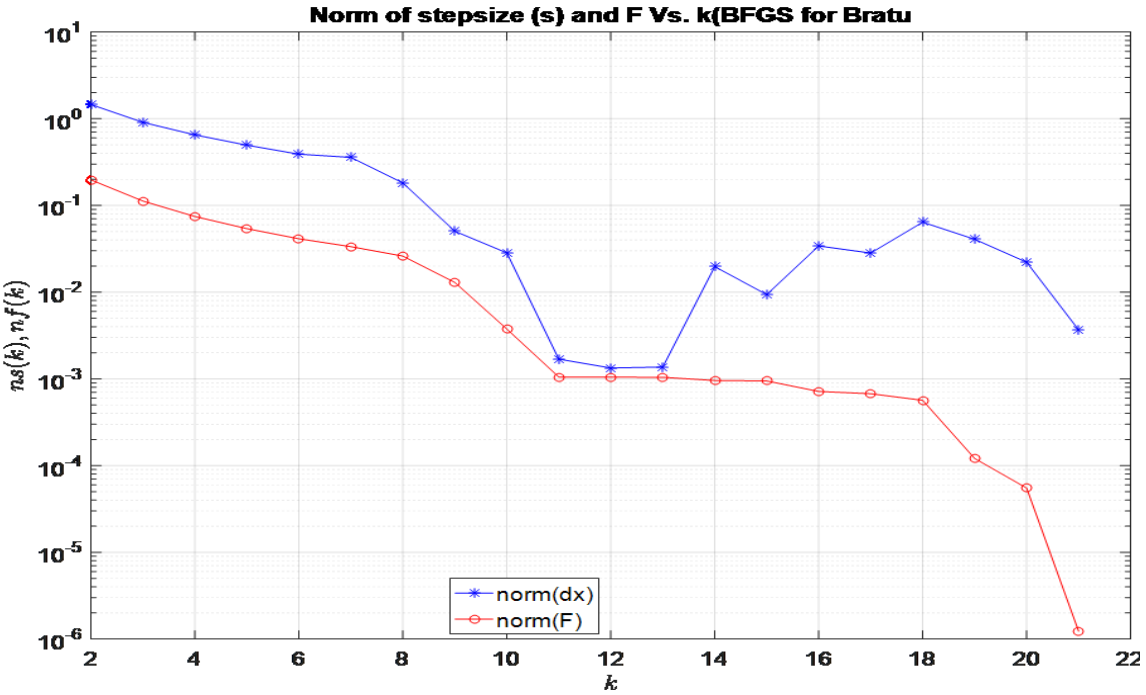

Figure 6.84 BFGS Method Euclidian Norm of Errors versus number of iteration for Bratu problem.

Fig.6.83 cites a comparison between the BFGS method solution of Bratu problem and exact solution. From this comparison, one can see a good agreement between the exact solution and the BFGS (approximate) solution.

As shown in Table 6.42, **BFGS Method** converges for Bratu problem and terminate within 21 iteration for parameter value λ=1, which is better than DFP method iteration. Moreover as shown in Fig.6.81, which proves the convergence of BFGS **Method** for Bratu problem and the norm errors of a Newton step (du) are not continuously decreasing. This shows that at some point of iteration norm of Newton steps are increasing rather than decreasing, it is not stable but the Euclidean norm error of the F are decreases continuously below predefined error tolerance.

Problem-2- consider Troesch's 1-D nonlinear problem defined in section (1.2)

$$u'' = \lambda \sinh(\lambda u) \qquad (6.3.2.3)$$

Subject to the dirichlet boundary conditions:

$$u(0) = 0 \ \ and \ \ u(1) = 1. \qquad (6.3.2.4)$$

The theoretical solution for this problem given by, equation (1.2.7).

Using equation (6.3.2.3) and (3.3), we can create NLSAEs for Troeschs problem

$$F(u_j) = fh^2 + u_{j-1} - 2u_j + u_{j+1},\ for\ \ j = 1,2,...,N$$

Where $f = \lambda \sinh(\lambda u(j))$

The above NLSAEs solve by using **BFGS Method and take inputs of DFP method for the same problem gives the following output:**

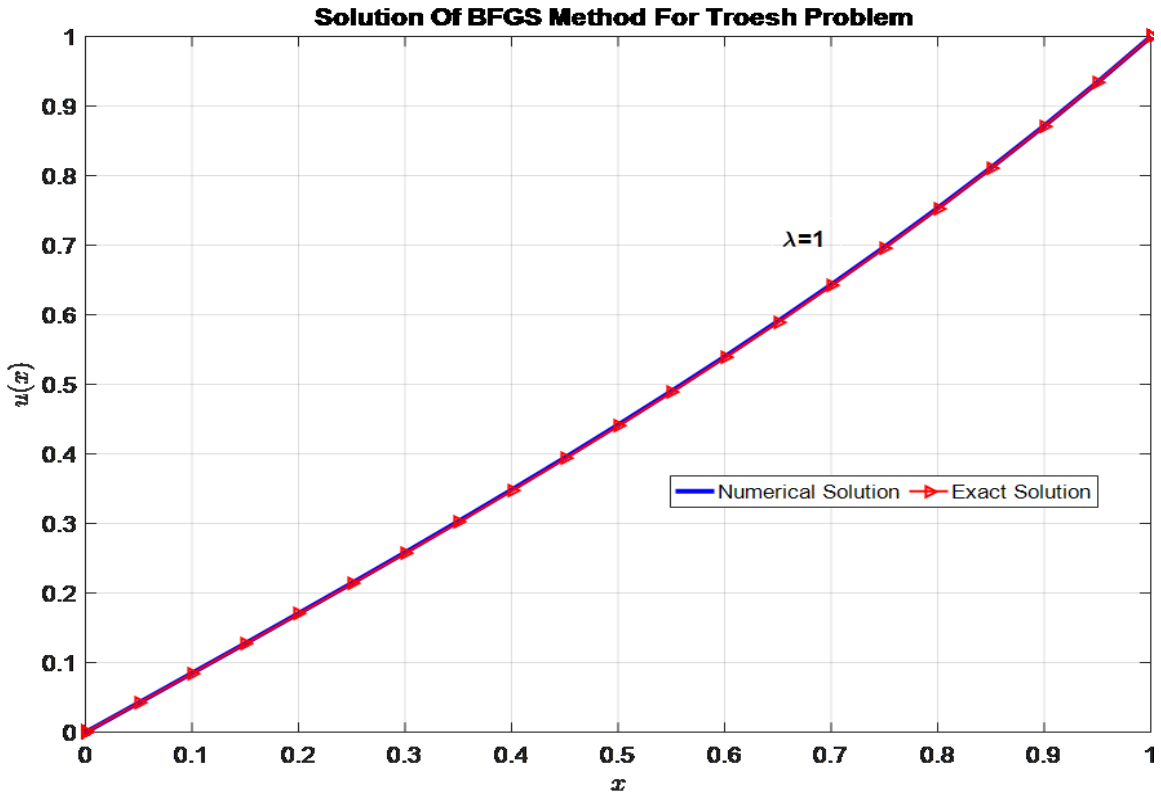

Figure 6.85 Comparison between the approximated solution and exact solution of Troesch problem using BFGS Method.

Table 6.43: BFGS method Euclidian norm of error, number of iteration and CPU time for Troesch's problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of fuction (F)	CPU time
29	1	5.001e-3	4.0295e-8	0.15625

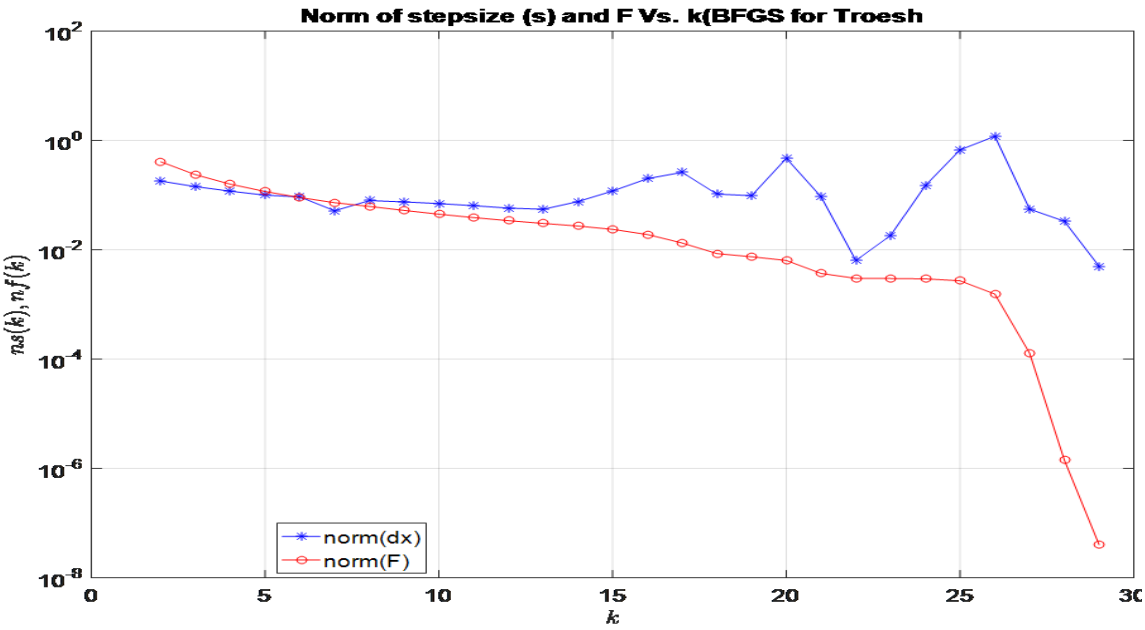

Figure 6.86 BFGS Method Euclidian Norm of errors versus number of iterations for Troesch problem.

As shown in Fig.6.85, a comparison between the approximate solution obtained by **BFGS Method** and exact solution. From this comparison, one can see a good agreement between the exact solution and the **BFGS Method** (approximate) solution for Troesch's problem.

As shown in Table 6.43, **BFGS Method** converges for Troesch problem and terminate within 29 iteration for parameter value λ=1, its CPU time shows BFGS method takes more time for computation than Newton method but less time than DFP method. Moreover as shown in Fig. 6.86, which proves the convergence of **BFGS Method** for Bratu problem and the norm errors of a Newton step (du) are not continuously decreasing. It is not stable due to numerical approximation of inverse hessian matrix, but the Euclidean norm error of a function are stable and decreases continuously below a desired error tolerance.

CHAPTER SEVEN

7. Continuation by Parameter (CBP) Method

When the solutions of system of NLAEs or ODE are very sensitive to changes in the initial values; especially, when the solutions are located in very close proximity. In the majority of cases, it is practically impossible to derive correct diagrams or solutions. Thus, other methods have to be employed, for example, the parametric continuation method, the description and application of which is presented below.

Consider a system of m nonlinear equations with m unknowns, containing a parameter p

$$F_i\left(x_1, x_2, ..., x_m, p\right) = 0, \quad i = 0,1,2,...,m \tag{7.0}$$

Using vector notations,

$$x = \left[x_1, x_2, ..., x_m\right]^T \quad , \quad F = \left[F_1, F_2, ..., F_m\right]^T \tag{7.1}$$

We present the system of equations in a vector form

$$F(\bar{y}, p) = 0 \tag{7.2}$$

Introducing Jacobian matrix

$$\bar{J} = \left\{\frac{\partial F_i}{\partial \bar{y}_i}\right\}, \quad i = 1,2,...,n \tag{7.3}$$

Whereas, $\bar{w}$ is the partial derivative vector calculated in terms of parameter p :

$$\bar{w} = \left\{\frac{\partial F_i}{\partial p}\right\}, \quad i = 1,2,...,n \tag{7.4}$$

By designating the total differential of system (7.2), we obtain

$$J\frac{\partial x}{\partial p} + \frac{\partial F}{\partial p} = 0, \qquad \text{or} \tag{7.5}$$

$$\bar{J}d\bar{y} + \bar{w}dp = 0 \tag{7.6}$$

From equation (7.6), we drive

$$d\bar{y} = -J^{-1}\bar{w}dp \tag{7.7}$$

If the above dependence is to be use in numerical calculations, differential increases $d\bar{y}$ and dp be substitute by the following difference increases:

$$\Delta\bar{y} = \bar{y}_{k+1} - \bar{y}_k, \quad \Delta p = p_{k+1} - p_k \tag{7.8}$$

Which leads to the following steps, which proceed until continuation parameter 'p' reaches the final value, which is p =1.

$$x_{(i+1)} = x_{(i)} - J^{-1}\bar{w}\Delta p, \quad i = 1,2,...,n \tag{7.9}$$

If the computational process is started with precisely determined values of $\bar{y}_0$, and p_0 (that is, fulfilling equation (7.0) with big accuracy), the successive values of vector $\bar{y}_k$, determining the next points of the diagrams, are designated without the necessity of using any other iterations [**77**].

7.1. Application of Continuation By Parameter (CBP)

To approximate the solution of mechanics problems define in section (1.2), we wrote an algorithm (7.1) for CBP method.

- **Algorithm (7.1): Solves nonlinear BVPs by using CBP Method.**

INPUT : N - no of variable &
u - guess solution vector & required for computation
$\Delta\lambda$ - constant used for change the parameter
λ - initial value of parameter
λlimit - final value of parameter

STEP -1 do step 2 until $\Delta\lambda > \lambda$limit

STEP-2 change the second order ODE to two first order ODE and do step-2

$$u'' = f(x,u,\lambda)$$

$u' = z(x,u,\lambda)$, **(a)** $z' = f(x,u,\lambda)$ **(b)**

STEP-3 differentiate the first order ODE (b) with respect to dependent variable U

$$\frac{df(x,u,\lambda)}{du}$$

STEP-4 differentiate the first order ODE (b) with respect to Parameter λ

$$\frac{df(x,u,\lambda)}{d\lambda}$$

STEP-5 solve this system of linear equation by Thomas Algorithm

$$v = Av + f(x,u,\lambda)$$

By Increase $\Delta\lambda$ **by** $\lambda = \lambda + \Delta\lambda$

STEP-6 update solution by

$$u = u + v\Delta\lambda$$

STEP-7 Display Numerical and graphical output

Based on Algorithm (7.1) and a MATLAB programs, we solve problems 1 and 2.

➢ **Numerical and Graphical output by using CBP**

The numerical method, CBP is applied and tested on the following BVPs that are define in section (1.2) and the numerical result are displayed by using MATLAB program.

Problem-1- consider Bratu 1-D nonlinear problem defined in section (1.2)

$$u''(x)+\lambda e^{u(x)}=0 \qquad (7.1.0)$$

Subject to the boundary conditions:

$$u(0)=u(1)=0 \qquad (7.1.1)$$

The Analytic solution for this problem given by, equation (1.2.3).

Using equation (7.1.0) and (3.3), we can create NLSAEs for Bratu problem

$$F(u_j)=fh^2+u_{j-1}-2u_j+u_{j+1},\ for\ \ j=1,2,...,N \qquad (7.1.2)$$

Where $f=-\lambda e^{u_j}$

Eq. (7.1.2) solve by using CBP method by using a MATLAB program and using the inputs: initial guess (u=ones(n,1)*2), error tolerance (tol= 10^{-5}), number of unknowns =19, number of iterations=50 and value of parameter $\lambda=1$, we get the following result:

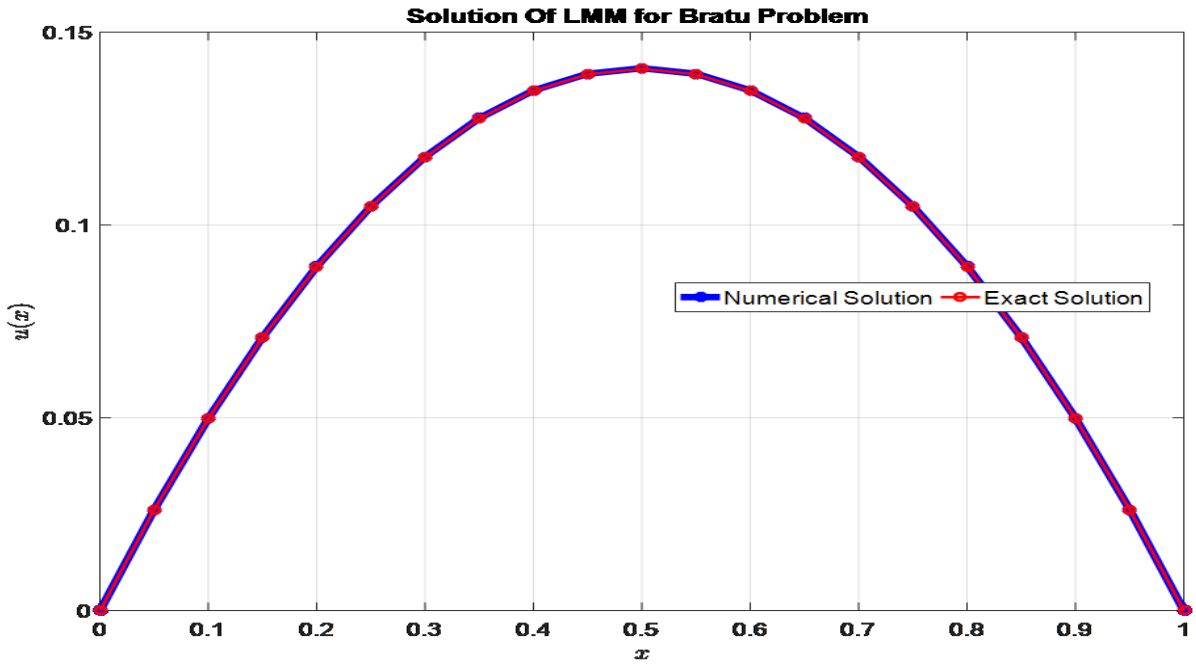

Figure 7.87 Comparison between the approximated solution and exact solution of Bratu problem using CBP method.

Table 7.44: CBP method Euclidian norm of error, number of iteration and CPU time for Bratu problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of fuction (F)	CPU time
25	1	5.33e-6	1.508e - 8	0.78125

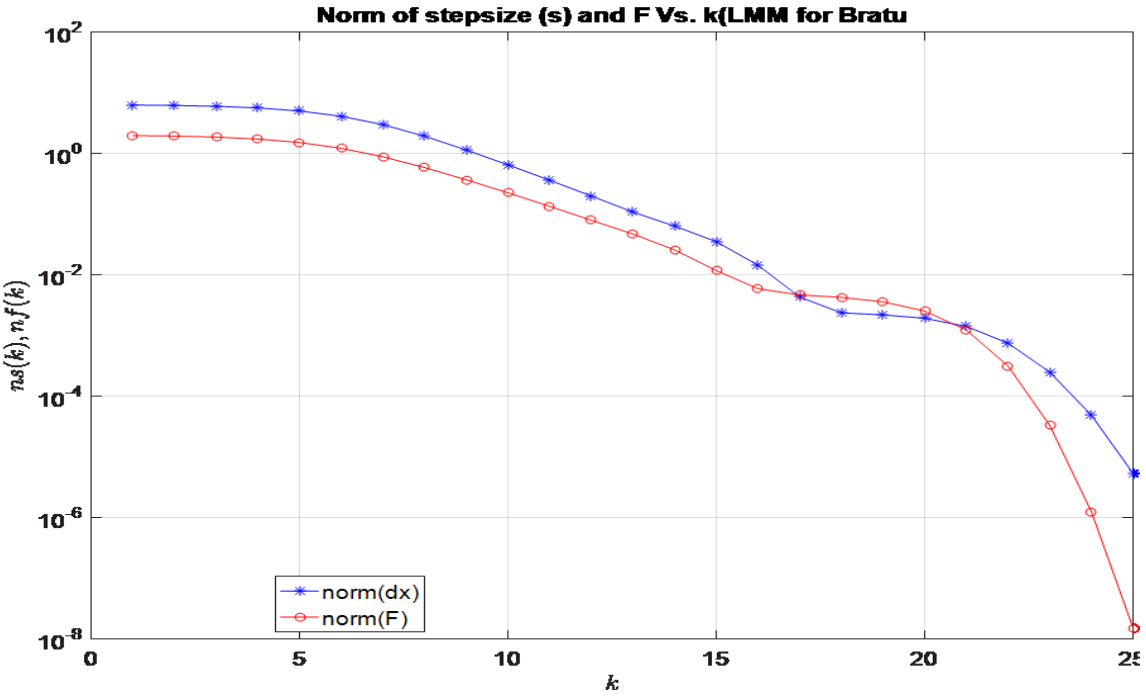

Figure 7.88 CBP Euclidian Norm of Errors versus number of iteration for Bratu problem.

Fig.7.87 cites a comparison between the approximate solution obtained by **CBP** method and exact solution. This shows a good agreement between the exact solution and the CBP (approximate) method solution.

As shown in Table 7.44, **CBP method** converges for Bratu problem and terminates within 25 iterations. The CPU time also indicates CBP method takes 0.7125. The Euclidean norm error of the function and steps below predefined limit. Moreover as shown in Fig.7.88, which proves the convergence of **CBP method** for Bratu problem and the norm errors of a Newton step (du) and function are continuously decreasing below predefined error tolerance.

Problem-2- consider Troesch's 1-D nonlinear problem defined in section (1.2)

$$u'' = \lambda \sinh(\lambda u) \qquad (7.1.2)$$

Subject to the boundary conditions:

$$u(0)=0 \ \ and \ \ u(1)=1. \qquad (7.1.3)$$

The theoretical solution for this problem given by, equation (1.2.7).

Using equation (7.1.2) and (3.3), we can create NLSAEs for Troeschs problem

$$F(u_j)=fh^2+u_{j-1}-2u_j+u_{j+1},\ for\ \ j=1,2,...,N$$

Where $f=\lambda\sinh(\lambda u(j))$

The above NLSAEs solve by using **CBP method and** using the inputs: initial guess (u=ones(n,1)*2), error tolerance (tol= 10^{-5}), number of iterations=30 and value of parameter , $\lambda=1$ we get the following output:

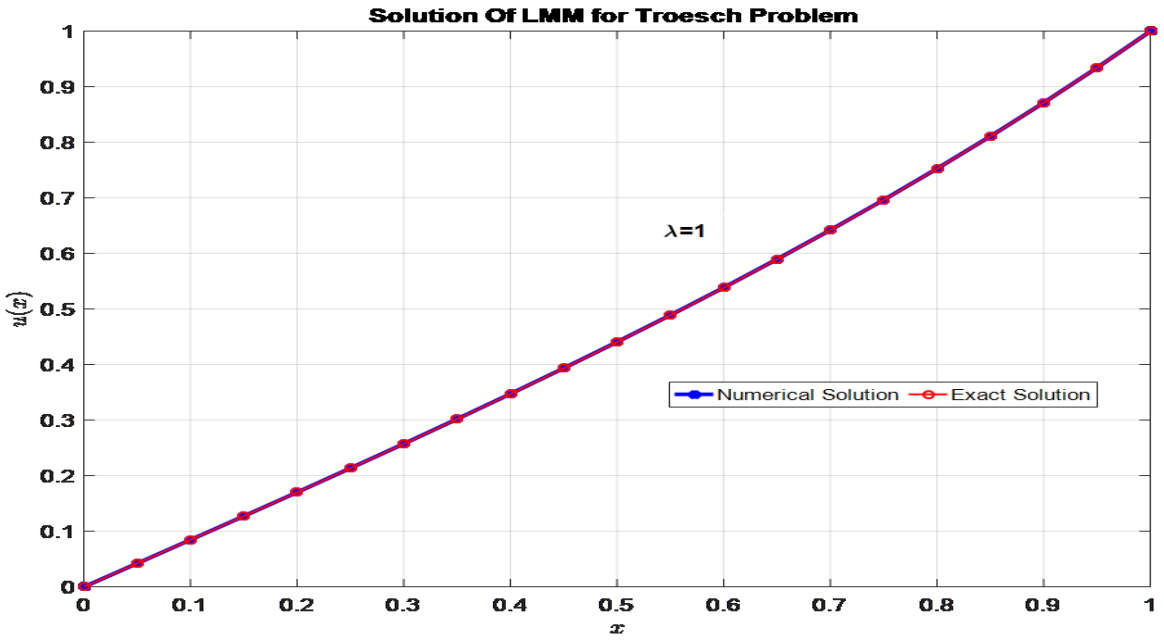

Figure 7.89 Comparison between the approximated solution and exact solution of Troesch problem using CBP.

Table 7.45: CBP method Euclidian norm of error, number of iteration and CPU time for Troesch's problem.

Number of iteration(k)	Parameter λ	Euclidian norm of Newton step (du)	Euclidian norm of fuction (F)	CPU time
24	1	1.173e -5	4.564e-8	1.375

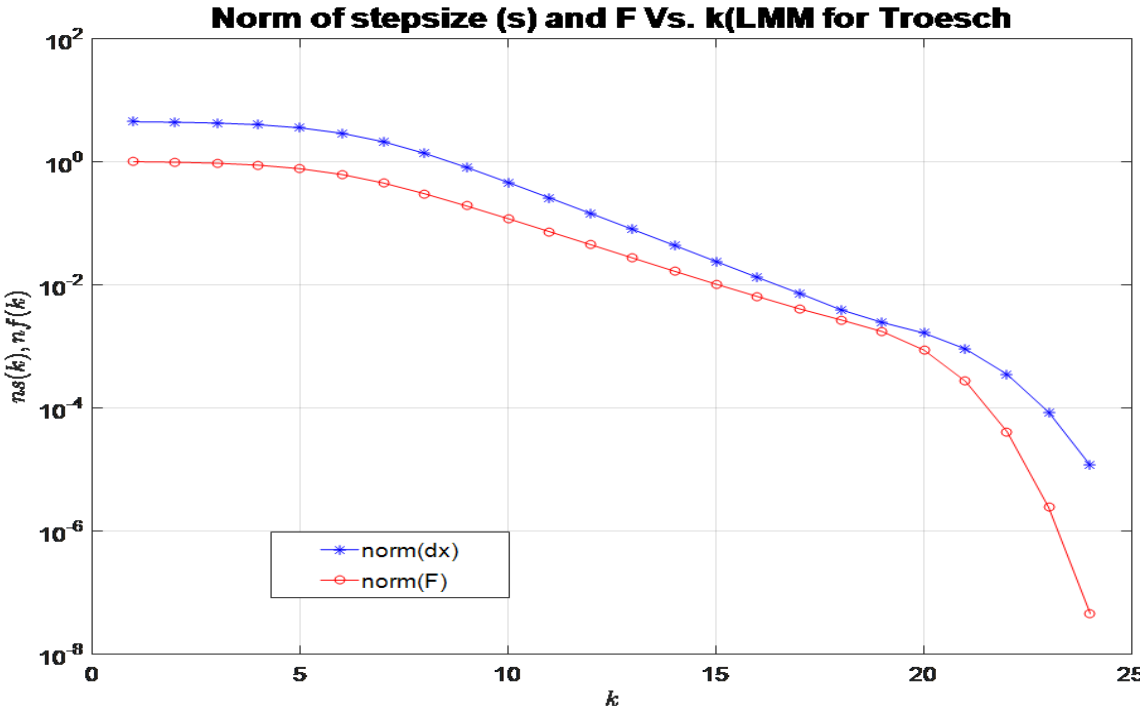

Figure 7.90 CBP method Euclidian Norm of Errors versus number of iteration for Troesch problem.

Analysis

As shown in Fig.7.89 a comparison between the approximate solution obtained by **CBP** method and exact solution. From this comparison, one can see a good agreement between the exact solution and the **CBP** (approximate) method solution for Troesch's problem.

As shown in Table 7.45, **CBP** method converges for Troesch problem and terminates within 24 iterations. The CPU time also indicates CBP method takes 1.375" to compute solution. The Euclidean norm error of the Newton step and function shows, CBP method has better accuracy. Moreover as shown in Fig.7.90, which proves the convergence of **CBP** method for Troesch problem and the norm errors of the function are continuously decreasing below predefined error tolerance.

CHAPTER EIGHT

8. CONCLUSION

Different numerical methods has been applied to different nonlinear boundary value problems from different areas of engineering like fluid mechanics, mechanics of solid, etc. comparison of these methods by the criteria like number of iteration, norm errors of step, norm error of function and CPU time are displayed below.

Table 7.46: Comparison of Newton and Quasi Newton's methods

Numerical Methods	problems	No Iter	Norm(du)	Norm(F)	CPU time	parameter
Newton	Bratu	4	1.88e-7	4.2e-9	0.25	1
	Troesch's	3	1.44e-5	4.03e-7	0.03125	1
	HT	5	8.52e-11	2.71e-12	2.1563	1e-9
	Thermal Explosion	14	2.52e-4	9.13e-5	0.09135	0.1
Broyden_I	Bratu	22	2.97e-5	4.8e-8	0.1563	1
	Troesch's	11	1.01e-3	4.4e -7	1.59	1
Broyden_II	Bratu	24	6.55e-9	1.66e-9	0.25	1
	Thermal Explosion	48	4.8e-7	7.59e-9	0.6406	0.1
BC1	Bratu	2	3.45e -4	2.14e-6	0.28125	1
	Troesch's	2	8e-12	4.16e-12	0.421875	1
BC2	Bratu	2	7.01e-6	4.2e-9	0.1406	1
	Troesch's	2	1.24e-10	3.83e-12	1.4719	1
MBC1	Bratu	1	1.345e-4	1.063-6	0.2031	1
	Troesch's	1	3e-12	4.16e-12	1.6375	1
MBC2	Bratu	2	2.167e-8	3.79e-9	0.0369	1
	Troesch's	2	1.253e-13	4.028e-15	1.8125	1

From Table 6.47, we show that the result obtained by MBC methods converge with few iteration and a better accuracy than Broydens methods and Newton method, however MBC needs more CPU time than other methods. Moreover, MBC methods have a better accuracy than BC methods due to predictor and corrector step.

It is finding that Newton Homotopy Analysis method (NHAM) with forward Euler step solved successfully all the model boundary value problems including notoriously famous 'stiff' Troesch's problem and Thermal Explosion problem with Neumann boundary

condition. This method approximate solution with a very small norm error, that is around 10^{-41}.

The modification of NHAM with RK2, RK4, RK5 and RK6 steps solved this highly nonlinear model BVPs with a better accuracy than NHAM with Euler steps.

Our results from NHAM with RKs method successfully solved nonlinear BVPs appears in a number of applications shows that, the method solved with high accuracy than any methods in this paper. However, the method takes big CPU time than any of the methods listed in this thesis.

Table 7.47: Comparison of optimization methods

Numerical Methods	**problems**	**No Iter**	**Norm(du)**	**Norm(F)**	**CPU time**	**parameter**
BFGS	Bratu	21	3.68e-3	1.23e-6	0.125	1
	Troesch's	29	5e-3	403e-8	0.15625	1
DFP	Bratu	78	7.05e-3	9.21e--8	0.42188	1
	Troesch's	107	1.63e-3	1.05e-6	0.34375	1
Newton	Bratu	4	7.77e-7	1.73e-10	0.40625	1
	Troesch's	3	1.764e-6	6.61e-10	0.3125	1
LM	Bratu	25	5.33e-6	1.508e-8	0.5625	1
	Troesch's	24	1.173e-5	4.564e-8	1.375	1

From Table 7.47, we show that the result obtained by Newton method, converge with few iteration and a better accuracy than other optimization methods.

Finally, it is finding that CBP method solved successfully all the boundary value problems including notoriously famous Troesch's problem. This method provided solution within the range of the parameter, exceeding much the ones appeared to be critical for the solvers used by other methods.

Works Cited

[1] U. M., Mattheij, R. M. M., and Russell, R. D. Ascher, *Numerical solution of boundary value problems for ordinary differential equations, of Classics in Applied Mathematics., PA*, 13th ed., Society for Industrial and Applied Mathematics, Ed. Philadelphia: SIAM, 1995.

[2] P. O. Wheatley C. F. Gerald, *Applied Numerical Analysis*.: Pearson , (2003).

[3] C. G. Broyden, *A class of methods for solving nonlinear simultaneous equations, 577–593.*: Math. Comp., 19, 1965.

[4] Y. Abu-Hour, "Improved Classes of Broyden methods for solving a nonlinear systems of equations," *Jordan University of Science and Technology*, 2016.

[5] S.J. Liao, "The proposed homotopy analysis technique for the solution of nonlinear problems," *Ph.D. Thesis*, p. Shanghai Jiao Tong University, 1992.

[6] T.Yamamoto, "Historical development in convergence analysis for Newton's and Newton-like methods," *Comput. Appl. Math.*, vol. 124, pp. 1-23, 2000.

[7] S.J. Liao, "Notes on the homotopy analysis method: some definitions and theorems,983–997.," in *Commun.*.: Nonlinear Sci. Numer. Simul., 2009.

[8] Jafri, M.D., Suleiman M., Majid, Z.A., Ibrahim Z.B., "Solving directly two point boundary value problems using direct multistep method. : 723-728.," *Sains Malaysiana*, vol. 38(5), pp. 723-728, 2009.

[9] W. C. Davidon, "Variable metric methods for minimization," *SIAM J. Optim*, vol. 1, pp. 1-17, 1991.

[10] C. G. Broyden, *"The convergence of a class of double rank minimizationalgorithms:* , 6th ed.: "J. Inst. Math, no. The new algorithm. pp. 222–231, , April 1970.

[11] R. Fletcher, *"A new approach to variable metric algorithms,"* , 13th ed.: Computer J, pp. 317–322, 1970.

[12] S. M Robinson, *Normal Maps Induced by Linear Transformations.:*.: Math. Oper., Res. 17 691-714. oz.au., 1992.

[13] Leif Rune Hellevik, *Numerical Methods for Engineers*, NTNU, Ed.: Department of Structural Engineering, Jan 26, 2018.

[14] E.S. Weibel, in: R.K.M. Landshof, "The Plasma in Magnetic Field, Stanford," *Stanford University Press*, 1958.

[15] Steve Wright, *introduction to scientific computing,*.: cs416:, 2007.

[16] Ron Buckmire, *On exact and numerical solutions of the one-dimensional planar Bratu problem ,1600 Campus Road Los Angeles, CA 90041-3338,*. U.S.A., Occidental College: Mathematics Department, June 12, 2003.

[17] Z. Pashazadeh Atabakan, and A. KJlJçman A. Kazemi Nasab, *An Efficient Approach for Solving*

Nonlinear Troesch's and Bratu's Problems by Wavelet Analysis Method , , 201382nd ed.: Hindawi Publishing Corporation , 2013.

[18] "A variational iteration method for solving Troesch's problem, ," *Journal of Computational and Applied Mathematics,* , vol. 234, no. Shih-Hsiang Chang., p. 3043–3047, (2010).

[19] Hector Vazquez-Leal,Yasir Khan,Guillermo Fern andez-Anaya, Agust´ ın Herrera-May,Arturo Sarmiento-Reyes,Uriel Filobello-Nino,Vıctor-M. Jimenez-Fernandez, and Domitilo Pereyra-D ıaz, "A General Solution for Troesch's Problem," *Mathematical Problems in Engineering*, vol. 2012, p. 14 pages, october 6 2012.

[20] Ali Umit Keskin, *Boundary Value Problems for Engineers: with MATLAB solution*.: Springer, 2019(19-517).

[21] Promise Mebine* and Erebi Sandra porbeni, "LMM for a Nonlinear Model of Convective Straight Fins with Variable Thermal Conductivity and Heat Transfer Coefficient," *International Journal of Scientific & Engineering Research*, vol. Volume 7, no. Issue 11, pp. 630-651, November 2016.

[22] J.P. Boyd, "One-point pseudospectral collocation for the one dimensional Bratu equation, Comp," *Appl. Math*, vol. 217, pp. 5553-5565, 2011.

[23] N. Caglar, M. Ozer, A. Valarstos, A.N. Anagnostopoulos H. Caglar, "B-spline method for solving Bratu's problem," *Int.J. Comput. Math*, vol. 87, pp. 1885-1891, 2010.

[24] B. A. Troesch, "“Intrinsic difficulties in the numerical solutionof a boundary value problem," ” *Internal Report 142, TRW Inc*, RedondoBeach,Calif,USA,1960.

[25] I. Hosseinpour , S. Ghanbarpour, A. Barari S. H. Mirmoradi, "Application of an Approximate Analytical Method to Nonlinear Troesch's Problem ," *Applied mathmathics science*, vol. vol.3, no. no.32, pp. 1579-1585, 2019.

[26] L. Berenguer, D. Tromeur-Dervout, "Developments on the Broyden procedure to solve nonlinear problems arising in CFD, 891–896.," 2013.

[27] H. Matthies and G. Strang, *The solution of nonlinear finite element equations.*., Int. J. Numer. Meth. Engng 14 (1979)..

[28] R.LAKSHMI, M.MUTHUSELVI, "NUMERICAL SOLUTION FOR BOUNDARY VALUE PROBLEM USING FINITE DIFFERENCE METHOD," vol. 2, no. ISO 3297, 2013.

[29] Steven C.Chapra, *Applied Numerical Method for Engineers and Scientist*. New York: McGraw Hill, 2008.

[30] Michael R. King ,Nipa A.Mody, "Numerical and Statistical Methods for Bioengineering," 2010.

[31] P.K.Pandey, "Solving Nonlinear Two Point Boundary Value Problems Using Exponential Finite Difference Method , (3s.) : 33–44.," vol. 34, 2016.

[32] Becker and Kaus, "Numerical Modeling of Earth Systems ," *Excerpt from GEOL557* , 2016.

[33] J. E. Dennis, "Numerical methods for unconstrained optimization and nonlinear equations. : ," 1983..

[34] Aliyu Bhar Kisabo, Nwojiji Cornelius Uchenna, Funmilayo Aliyu Adebimpe, "Newton's Method for Solving Non-Linear System of Algebraic Equations (NLSAEs) with MATLAB/Simulink and MAPLE (117-131)," vol. 2, (2018).

[35] C.T. Kelly, "Iterative Methods for Linear and Nonlinear Equations. :," 1995.

[36] Andreas Griewank, *Broyden updating, the good and the bad! Documenta Mathematica*.: ,Optimization stories:301–315,., 2012.

[37] Courtney Remani, "Numerical Methods for Solving System of Nonlinear Equations," Lakehead University,Thunder Bay,Canada 2012-2013.

[38] O. P Burdakov, *Some Globally Convergent Modifications of Newton's Method for Solving Systems of Nonlinear Equations*.: Sot iet Math. Dokl. 22 376-379., 1980.

[39] XUPING ZHANG AND BO YU, "A NOTE ON THE RELATION BETWEEN THE NEWTON HOMOTOPY METHOD AND THE DAMPED NEWTON METHOD," *ETNA*, vol. 40, no. ISSN 1069-9613., pp. 378-380, 2013.

[40] Mohammad H. Al-Towaiq∗, Yousef S. Abu hour, , "Two improved classes of Broyden's methods for solving nonlinear systems of equations , ," 2017.

[41] M. Mamat, K. Muhammad, M. Y. Wazir, "Trapezoidal Broyden Method for Solving Systems of Nonlinear Equations. ," *Applied Mathematical Sciences*, vol. 8, pp. 251-260, 2014.

[42] M. Mamat, K. Muhammad, M. Y. Waziri , "Trapezoidal Broyden Method for Solving Systems of Nonlinear Equations. ," *Applied Mathematical Sciences,* vol. 8, pp. 251-260, 2014.

[43] M. Mamat, Y. M. Waziri - K. Muhammad, "A Broyden's-like Method for Solving Systems of Nonlinear Equations. ," *World Applied Sciences Journal* , vol. 21, no. (Special Issue of Applied Math):, pp. 168-173, 2013.

[44] I. A. Osinuga, S. O. Yusuff, "Construction Of a Broyden-Like Method For Nonliner Systems Of Equations, ," *Anale. Seria Informatică.*, vol. Vol. XV, no. fasc. 2 , pp. 128-135, 2017.

[45] Y. Tan, S.J. Liao, S. Abbasbandy, "] Newton-homotopy analysis method for nonlinear equations, Applied Mathematics and Computation, ," *ScienceDirect*, vol. 188, p. 1794–1800, 2007.

[46] Brune PR, Knepley MG, Smith BF, Tu X., "Composing scalable nonlinear algebraic solvers. ; ," vol. 57(4):535–565., SIAM Review 2015.

[47] C. Grosan and A. Abraham, "“A new approach for solving Nonlinear equations systems,”," vol. Vol.38, No.3, pp. 698-714, no. IEEE Trans. Systems, 2008.

[48] W. Song, Y. Wang, H. X. Li, and Z. Cai, "“Locating multiple optimal solutions of nonlinear equations systems based on multi objectiveoptimization,” ".

[49] Wright., J. Nocedal and S. J., *Numerical Optimization*, 2nd ed. New York: Springer, 2006.

[50] W.C. Davidon, *Variable Metric Method for Minimization.*, Atomic Energy Commission Research and Development Report, Ed.: ANL–5990 (Rev.), November November 1959.

[51] R. Fletcher and M. J. D. Powell, "A rapidly convergent descent method for minimization," *Comput. J.,* vol. 6, p. 163–168, 1963.

[52] M. J. D. Powell, "On the convergence of the variable metric algorithm," *J. Inst. Math.*, vol. 6, pp. 21-36, Appl., 7 1971.

[53] D. Goldfarb, "A family of variable metric methods derived by variational means," *Math. Comp*, vol. 24, p. pp. 23–26., 1970.

[54] D. F. Shanno, "Conditioning of quasi-Newton methods for function minimization," *Math. Comp.*, vol. 24, p. pp. 647–650, 1970 1970.

[55] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear inequalities.," *SIAM J.Appl. Math.*, vol. 11, p. 431–441, 1963.

[56] N., Fukushima, M. Yamashita, "On the rate of convergence of the Levenberg-Marquardt method.," *Computing (Suppl. 15)*, p. 237–249 , 2001.

[57] Rao V.Dukkipati, *Numerical Method, New Delhi •Bangalore •Chennai •Cochin •Guwahati*. USA, New Delhi: New Age International, 2010.

[58] C. Remani, *"Numerical Methods for Solving System of Nonlinear Equations"*. Canada, Thunder Bay: Lakehead University, 2012-2013.

[59] R.L,Faires,J.D Burden, "Numerical Solution of Nonlinear system of Equation," in *Numerical Analysis*. Belmount: Thomas Brooks/cole, 2005, pp. 597-640.

[60] Jaan Kiusalaas, *Numerical Methods in Engineering with MATLAB*. New York: Cambridge University Press, 2005,pp 393-435.

[61] J. M. and Rheinboldt, W. C. Ortega, "Iterative solution of nonlinear equations in several variables, reprint of the 1970 original, Classics Appl. Math.," *SIAM, Philadelphia, PA.*, vol. 30, 2000.

[62] Faires,J.D Burden, R.L, *"Numerical Solution of Nonlinear system of Equation," in Numerical Analysis. Belmount:* , Thomas Brooks/cole, Ed., 2005, pp. 597-640.

[63] M. H. Al-Towaiq, Y. S. Abu Hour, "Two Improved Methods Based on Broyden's Newton Methods for the solution of nonlinear systems of equations,(accepted). 1, 3".

[64] C. Y. Deng, "A generalization of the Sherman-Morrison-Woodbury formula," *Appl. Math. Lett.*, vol. 2, p. 1561–1564, 2011.

[65] H. Mohammad, M. Y. Waziri , "- On Broyden-like update via some quadratures for solving nonlinear systems of equations," *Turkish Journal of Mathematics,* vol. 39, pp. 335-345, 2015.

[66] J. R. Torregrosa A. Cordero, "Variants of Newton's method for functions of several variables.

," *Appl Math Comput*, vol. 183, pp. 199-208, 2006.

[67] Faires, J. and Burden, R., *Numerical methods*, 3rd edition, Ed.: Brooks Cole, 2002.

[68] D. and Burden, R.L. Faires, *Numerical Analysis.*. USA: International Thomson Publishing Inc, 1998.

[69] Talib Hashim Hasan, Md. Sazzad Hossien Chowdhury, Prayoto, "Solving Nonlinear Algebraic Problem Using Newton Homotopy Differential Equation, ," *Australian Journal of Basic and Applied Sciences*, vol. 5(4), pp. 56-59, 2011.

[70] R.Brent., *Algorithms for Minimization without Derivatives.* Dover: Dover Books on Mathematics, 2013.

[71] Timothy Sauer, *Numerical analysis* , 2nd ed. New York: Pearson Education, Inc., 2006.

[72] W., Hext, G. R., and Himsworth, F. R. Spendley, *"Sequential Application of Simplex Designs in Optimisation and Evolutionary Operation*, 44441461st ed, Ed.: Technometrics, 1962.

[73] S.Boyd,N.Parikh,E.Chu,B.Peleato and J.Eckstein, "Distributed Optimization and Statistical Learning via the alternating direction method of multipliers,Foundation and Trends in Machine Learning," vol. 3, Jan.2011.

[74] Yamashita, N., Fukushima, M., "On the rate of convergence of the Levenberg-Marquardt method.," no. Computing (Suppl. 15), p. 237–249, 2001.

[75] S.S. and Luenberger, D.] Oren, *"Self–scaling Variable Metric Algorithms, Part I," , 845862nd ed.:*, 2nd ed., Manage. Sci., Ed., 1974, vol. 20(5).

[76] J. Kiefer, *"Sequential Minmax Search for a Maximum,"*.: Proceedings of the American Mathematical Society, 4, 1953. pp. 502–506,.

[77] Marek Berezowski, "The parametric continuation method for determining steady state diagrams," *Silesian University of Technology, Faculty of Applied Mathematic ,Poland, Gliwice,*.