



WALLAGA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

**NUMERICAL SOLUTION OF INITIAL VALUE PROBLEM
OF BRATU TYPE EQUATIONS USING EIGHTH ORDER
PREDICTOR-CORRECTOR METHOD**

MSc. Thesis

By ABOMA TAMANA GAMACHU

JULY, 2026

NEKEMTE, ETHIOPIA



**WALLAGA UNIVERSITY
SCHOOL OF GRADUATE STUDIES**

**NUMERICAL SOLUTION OF INITIAL VALUE PROBLEM
OF BRATU TYPE EQUATIONS USING EIGHTH ORDER
PREDICTOR-CORRECTOR METHOD**

**A THESIS SUBMITTED TO THE COLLEGE OF NATURAL AND
COMPUTATIONAL SCIENCE, DEPARTMENT OF MATHEMATICS
IN PARTIAL FULFILLMENTS OF THE REQUIREMENTS FOR THE
DEGREE OF MASTERS OF SCIENCE IN MATHEMATICS
(NUMERICAL ANALYSIS)**

By: ABOMA TAMANA GAMACHU

Advisor: MAMO ABEBE ASHEBO (PhD)

**MAY, 2026
NEKEMTE,**

ETHIOPIA

STATEMENT OF THE AUTHOR

I, **Mr. AbomaTamanaGamachu**, declare that the thesis entitled “**Eighth Order Predictor-Corrector Method for Solving Initial Value Problems of Bratu-Type Equations**” is the result of my own academic work carried out under the guidance and supervision of **Dr. MamoAbebeAshebo**.

I confirm that all stages of this research, including the design of the study, data collection, analysis, interpretation of results, and preparation of the thesis, have been conducted in accordance with accepted academic and ethical standards. Appropriate acknowledgment has been given to the ideas, findings, and works of other scholars through proper citation and referencing.

I further certify that no part of this thesis contains fabricated, falsified, or misleading information and that the work presented herein is original. This thesis is submitted to the Postgraduate Program of Wallaga University in partial fulfillment of the requirements for the degree of Master of Science in Mathematics.

I also affirm that this thesis has not previously been submitted, either wholly or partially, to any other university or institution for the award of any degree, diploma, or certificate. I understand that any breach of academic integrity may result in disciplinary measures in accordance with the rules and regulations of the University.

Name: **AbomaTamanaGamachu**

Signature: _____

Date: ____ / ____ / ____

Department: **Mathematics**

College: **Natural and Computational Sciences**

BIOGRAPHICAL SKETCH

Mr. Aboma Tamana Gamachu was born in Kellem Wallaga Zone at Dale Sadi Woreda in Wara Wale Suchi Kebele in 1992 G.C. He joined primary school grade 1-8 at Gnea Kara Belam elementary school during 2000-2007 G.C and secondary school grade 9-12 at Haro Sabu School during 2008-2011 G.C. After he has completed his high school, he joined Adama Science and Technology University during 2012-2014 G.C. by regular to hold his BSc degree. Now his work place is at Kellem Wallaga Zone Dale Sadi Woreda Dale secondary School and he served for 8 years in government employment. Currently, he is married and he has two children.

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to Almighty God, the Son of the Holy Virgin Mary, for granting me life, health, strength, and guidance through out my academic journey, enabling me to reach this stage successfully.

I am sincerely thankful to my advisor, **Dr. MamoAbebe**, for his continuous support, valuable guidance, constructive suggestions, and professional expertise during the entire period of my research and the preparation of this thesis. His encouragement, insightful comments, and dedication greatly contributed to the successful completion of this work.

I would also like to extend my heartfelt appreciation to my family for their unwavering encouragement, understanding, and support throughout my study period. Their motivation played a significant role in helping me achieve my academic goals.

Finally, I would like to acknowledge Wallaga University for providing me with the opportunity to pursue my graduate studies and for offering the necessary academic resources and facilities that supported the successful completion of this research.

ABBREVIATIONS AND ACRONYMS

IVPs	Initial value problems
ODEs	Ordinary differential equations
PCM8	Eighth order predictor-corrector method

SYMBOLS

α :	Initial value of $y(x)$
λ	Parameter
γ	Initial value of $y'(x)$

TABLE OF CONTENTS

Contents	Pages
STATEMENT OF THE AUTHOR.....	I
BIOGRAPHICAL SKETCH.....	II
ACKNOWLEDGMENTS	III
ABBREVIATIONS AND ACRONYMS.....	IV
TABLE OF CONTENTS	V
LIST OF TABLES	VII
<i>ABSTRACT</i>	IX
CHAPTER ONE	1
INTRODUCTION.....	1
1.1. Background of the Study	1
1.2. Statement of the Problem	2
1.3. Objectives of the Study	2
1.3.1. General Objective	2
1.3.2. Specific Objectives	2
1.4. Significance of the Study	3
1.5. Scope of the Study.....	3
1.6. Mathematical Preliminaries and Concepts	3
1.6.1. Interpolation with Equally Spaced Points.....	4
1.6.2. Newton’s Difference Backward Interpolation Formula	5
1.6.3. The six stage fifth-Order Runge-Kutta Method.....	6
1.6.4. Error of numerical method.....	7
1.7. Thesis Organization.....	8
CHAPTER TWO	9
REVIEW OF RELATED LITERATURE	9
CHAPTER THREE	12
RESEARCH METHODOLOGY	12
3.1. Study Area and Period.....	12
3.2. Study Design	12
3.3. Sources of Information.....	12

3.4. Mathematical Procedure.....	12
3.5. Ethical Considerations.....	13
CHAPTER FOUR.....	14
RESULTS AND DISCUSSION	14
4.1. Definition of the Problem.....	14
4.2 Development of the Predictor Method	16
4.3. Development of Corrector Method	19
4.4. Stability and Convergence Analysis of the Developed Method	22
4.5. Numerical Examples and Results.....	25
4.6. Discussion	34
CHAPTER FIVE	36
SUMMARY, CONCLUSION AND RECOMMENDATION.....	36
5.1 Summary	36
5.2. Conclusion.....	37
5.3. Recommendation and Future Work	38
REFERENCES.....	39
APPENDIX.....	41

LIST OF TABLES

Table 4.1: Comparison of exact solution and numerical solution of Example 4.1 for $h = 0.1$ using PCM8	25
Table 4.2: Comparison of exact solution and numerical solution of Example 4.1 for $h = 0.01$ using PCM8	26
Table 4. 3: Comparison of exact solution and numerical solution of Example 4.1 for $h = 0.002$	27
Table 4.4: Comparison of exact solution and numerical solution of Example 4.2 for $h = 0.1$ using PCM8	30
Table 4.5: Comparison of exact solution and numerical solution of Example 4.2 for $h = 0.01$ using PCM8	31
Table 4. 6: Comparison of exact solution and numerical solution of Example 4.2 for $h = 0.002$ using PCM8	32
Table 4.7: The maximum absolute errors for different values of N.....	33
Table 4.8: The Comparison of absolute errors for Example 4.2 at mesh size $h=0.1$ with the existing literature	33

LIST OF FIGURES

Figure 4.1:- The graph of exact and numerical solution of Example 4.1 When $N=10$, $h=0.1$	26
Figure 4.2:- The graph of exact and numerical solution of Example 4.1 when $N=100$, $h=0.01$	27
Figure 4. 3:- The graph of exact and numerical solution of Example 4.1 when $N=100$, $h=0.01$	28
Figure 4.4:- The Error graph for $h=0.1$, $h=0.01$ and $h=0.002$ of Example 4.1.	29
Figure 4.5:- The graph of exact and numerical solution of Example 4.1 when $N=10$, $h=0.1$	30
Figure 4. 6:- The graph of exact and numerical solution of Example 4.2 when $N=100$, $h=0.01$	31
Figure 4.7: The graph of exact and numerical solution of Example 4.2 $N=500$, $h=0.002$..	32
Figure 4.8:- The error graph for $h=0.1$, $h=0.01$ and $h=0.002$ of Example 4.2.....	33

ABSTRACT

The primary purpose of this research was to construct an eighth-order predictor–corrector numerical scheme for solving second-order initial value problems associated with Bratu-type differential equations. The developed method is examined through stability and convergence analysis in order to evaluate its theoretical reliability. Numerical experiments are conducted to test the effectiveness and computational performance of the proposed approach. The approximate numerical results are compared with the corresponding exact solutions to determine the accuracy of the method. The comparison indicates that the proposed scheme produces highly accurate results with very small absolute errors. Furthermore, the influence of different step sizes on the accuracy of the numerical solutions is investigated. The study also demonstrated that smaller mesh sizes improve the precision of the obtained solutions. Several model examples are provided to verify the applicability, efficiency, and robustness of the developed predictor–corrector method.

CHAPTER ONE

INTRODUCTION

1.1. Background of the Study

Differential equations are fundamental mathematical tools used to represent many phenomena occurring in science, engineering, and technology. Numerous real-world applications including fluid flow, heat conduction, chemical kinetics, biological processes, and population growth can be modeled mathematically by differential equations. Since many nonlinear differential equations do not possess exact analytical solutions, numerical techniques have become essential for obtaining approximate solutions.

Among nonlinear differential equations, the Bratu-type equation has attracted considerable attention due to its broad range of applications in thermal combustion theory, electrostatic models, chemical reaction processes, and heat transfer problems. Because of the nonlinear characteristics of Bratu-type equations, finding exact solutions is often difficult, particularly for complicated models.

To address these challenges, researchers frequently employ numerical methods to generate approximate solutions. Predictor–corrector methods are among the most effective numerical techniques because they provide high accuracy, improved convergence, and numerical stability. In particular, high-order predictor–corrector schemes can produce reliable solutions even when relatively small step sizes are used.

This study is focused on the application of an eighth-order predictor–corrector method for solving initial value problems involving Bratu-type equations. The numerical results obtained from the proposed method are compared with exact solutions in order to assess the performance and efficiency of the scheme.

1.2.Statement of the Problem

Many nonlinear ordinary differential equations arising in scientific and engineering applications are difficult to solve analytically. Bratu-type equations are examples of such nonlinear problems. Although analytical solutions may be available for a limited number of cases, deriving these solutions is generally complicated and time-consuming.

As a result, there is a need for reliable numerical methods capable of producing accurate and stable approximations. The major challenge in numerical analysis is to design computational techniques that minimize numerical errors while maintaining efficiency and convergence.

The Bratu-type equation is one of such nonlinear problems. Although analytical solutions may exist for some special cases, obtaining them can be difficult.

Therefore, efficient numerical methods are required to approximate the solutions of such equations.

1.3. Objectives of the Study

1.3.1. General Objective

The general objective of this study is to apply eighth-order predictor–corrector method for solving initial value problems of Bratu-type equations.

1.3.2. Specific Objectives

The specific objectives of this research are:

- ✓ To derives an eighth-order predictor–corrector numerical scheme.
- ✓ To investigates the accuracy of the developed method using absolute error analysis.
- ✓ To examines the stability and convergence properties of the proposed technique.
- ✓ To compares the developed method with existing numerical approaches available in the literature.

1.4. Significance of the Study

This study contributes to the advancement of numerical techniques for solving nonlinear differential equations. The developed method may assist researchers, instructors, and students in understanding the effectiveness of higher-order predictor–corrector algorithms.

The research also demonstrates the practical application of MATLAB in solving nonlinear ordinary differential equations numerically. Consequently, the study may serve as a useful reference for future investigations in numerical analysis, applied mathematics, and engineering sciences.

1.5. Scope of the Study

The research is restricted to the numerical solution of initial value problems for Bratu-type equations using the eighth-order predictor–corrector method. The computational implementation is carried out using MATLAB software with different mesh sizes over a specified interval.

1.6. Mathematical Preliminaries and Concepts

This section introduces some basic concepts, definitions, and mathematical principles required throughout the study.

Definition 1.1. Initial Value Problem: An initial value problem is an ordinary differential equation accompanied by specified initial conditions that determine the value of the unknown function at a particular point.

Definition 1.2. Consistency: A numerical method is said to be consistent if the local truncation error approaches zero as the step size tends to zero.

Definition 1.3. Stability: A numerical scheme is considered stable if small perturbations in the computational process do not produce large deviations in the numerical solution.

Definition 1.4. Convergence: A numerical method is convergent if the approximate solution approaches the exact solution as the step size decreases. Consistency and stability are necessary conditions for convergence.

1.6.1. Interpolation with Equally Spaced Points

Interpolation techniques are useful for constructing approximate functions from known data values. In the case of equally spaced nodes, finite difference interpolation formulas provide efficient approximations.

There are different finite difference operators.

Backward difference operator (∇)

When the operator ∇ is applied on $f(x_i)$, we obtain

$$\nabla f(x_i) = f(x_i) - f(x_{i-1}) = f_i - f_{i-1}$$

That is $\nabla f(x_1) = f(x_1) - f(x_0)$

$$\nabla f(x_2) = f(x_2) - f(x_1)$$

$\nabla f(x_3) = f(x_3) - f(x_2)$, and so on

These differences are called the first backward differences. The second backward difference is defined by

$$\begin{aligned}\nabla^2 f(x_i) &= \nabla(f(x_i) - f(x_{i-1})) = \nabla f(x_i) - \nabla f(x_{i-1}), \\ &= f(x_i) - f(x_{i-1}) - (f(x_{i-1}) - f(x_{i-2})), \\ &= f(x_i) - 2f(x_{i-1}) + f(x_{i-2}) = f_i - 2f_{i-1} + f_{i-2}\end{aligned}$$

The third backward difference is defined by

$$\begin{aligned}\nabla^3 f(x_i) &= \nabla(\nabla^2 f(x_i)) = \nabla(f(x_i) - 2f(x_{i-1}) + f(x_{i-2})) \\ &= \nabla(f(x_i) - 2f(x_{i-1}) + f(x_{i-2})), \\ &= f_i - 3f_{i-1} + 3f_{i-2} - f_{i-3}\end{aligned}$$

Using this relation, we can write the nth backward difference of $f(x_i)$ as

$$\nabla^n f(x_i) = \sum_{k=0}^n (-1)^k \frac{n!}{k!(n-k)!} f_{i-k} \quad (1.1)$$

1.6.2. Newton's Difference Backward Interpolation Formula

Again, we use the newton's divided difference interpolation polynomial to drive the newton's backward difference interpolation formula. Since, the divided differences are symmetric with respect to their arguments, we write the arguments of the divided differences in the order $x_n, x_{n-1}, \dots, x_1, x_0$ (lyengar & Jain, 2009). the newton's divided difference interpolation polynomials can be written as

$$p_n(x) = f(x_n) + (x - x_n)f[x_n, x_{n-1}] + (x - x_n)(x - x_{n-1})f[x_n, x_{n-1}, x_{n-2}] + \dots + (x - x_n)(x - x_{n-1}) \dots (x - x_1)f[x_n, x_{n-1}, \dots, x_0] \quad (1.2)$$

Since, the divided differences are symmetric with respect to their arguments we have

$$f[x_n, x_{n-1}] = f[x_{n-1}, x_n] = \frac{1}{h} \nabla f_n$$

$$f[x_n, x_{n-1}, x_{n-2}] = f[x_{n-2}, x_{n-1}, x_n] = \frac{1}{2! h^2} \nabla^2 f_n, \dots$$

$$f[x_n, x_{n-1}, \dots, x_0] = f[x_0, x_1, \dots, x_n] = \frac{1}{n! h^n} \nabla^n f_n.$$

Substituting these in equation, we obtain the newton's backward difference interpolation formula

$$f(x) = f(x_n) + (x - x_n) \frac{1}{h} \nabla f_n + (x - x_n)(x - x_{n-1}) \frac{1}{2! h^2} \nabla^2 f_n + \dots + (x - x_n)(x - x_{n-1}) \dots (x - x_1) \frac{1}{n! h^n} \nabla^n f_n. \text{ And gives}$$

$$\begin{aligned} f(x) = f(x_i + n \cdot h) &= f_i + n \nabla f_i + \frac{n(n+1)}{2!} \nabla^2 f_i + \frac{n(n+1)(n+2)}{3!} \nabla^3 f_i \\ &\quad + \frac{n(n+1)(n+2)(n+3)}{4!} \nabla^4 f_i \\ &\quad + \frac{n(n+1)(n+2)(n+3)(n+4)}{5!} \nabla^5 f_i \\ &\quad + \frac{n(n+1)(n+2)(n+3)(n+4)(n+5)}{6!} \nabla^6 f_i \\ &\quad + \frac{n(n+1)(n+2)(n+3)(n+4)(n+5)(n+6)}{7!} \nabla^7 f_i \\ &\quad + \dots + \frac{n(n+1)(n+2)(n+3)(n+4)(n+5)(n+6)(n+7)(n+8)(n+9)(n+10)(n+11)(n+12)(n+13)(n+14)(n+15)(n+16)(n+17)(n+18)(n+19)(n+20)(n+21)(n+22)(n+23)(n+24)(n+25)(n+26)(n+27)(n+28)(n+29)(n+30)(n+31)(n+32)(n+33)(n+34)(n+35)(n+36)(n+37)(n+38)(n+39)(n+40)(n+41)(n+42)(n+43)(n+44)(n+45)(n+46)(n+47)(n+48)(n+49)(n+50)(n+51)(n+52)(n+53)(n+54)(n+55)(n+56)(n+57)(n+58)(n+59)(n+60)(n+61)(n+62)(n+63)(n+64)(n+65)(n+66)(n+67)(n+68)(n+69)(n+70)(n+71)(n+72)(n+73)(n+74)(n+75)(n+76)(n+77)(n+78)(n+79)(n+80)(n+81)(n+82)(n+83)(n+84)(n+85)(n+86)(n+87)(n+88)(n+89)(n+90)(n+91)(n+92)(n+93)(n+94)(n+95)(n+96)(n+97)(n+98)(n+99)(n+100)}{101!} \nabla^{101} f_i \end{aligned} \quad (1.3)$$

Note that $n = \frac{x - x_i}{h}$ and

$$\tau_k^{(j)} = \frac{n(n+1)(n+2)(n+3)(n+4)(n+5)(n+6)(n+7)(n+8)(n+9)(n+10)(n+11)(n+12)(n+13)(n+14)(n+15)(n+16)(n+17)(n+18)(n+19)(n+20)(n+21)(n+22)(n+23)(n+24)(n+25)(n+26)(n+27)(n+28)(n+29)(n+30)(n+31)(n+32)(n+33)(n+34)(n+35)(n+36)(n+37)(n+38)(n+39)(n+40)(n+41)(n+42)(n+43)(n+44)(n+45)(n+46)(n+47)(n+48)(n+49)(n+50)(n+51)(n+52)(n+53)(n+54)(n+55)(n+56)(n+57)(n+58)(n+59)(n+60)(n+61)(n+62)(n+63)(n+64)(n+65)(n+66)(n+67)(n+68)(n+69)(n+70)(n+71)(n+72)(n+73)(n+74)(n+75)(n+76)(n+77)(n+78)(n+79)(n+80)(n+81)(n+82)(n+83)(n+84)(n+85)(n+86)(n+87)(n+88)(n+89)(n+90)(n+91)(n+92)(n+93)(n+94)(n+95)(n+96)(n+97)(n+98)(n+99)(n+100)}{101!} \quad (1.4)$$

is the error term where τ lies in some interval containing the points $x_i, x_{i-1}, \dots, x_{i-k+1}$ and x . This is the newton's backward interpolation formula.

1.6.3. The six stage fifth-Order Runge-Kutta Method

Agbebohet *al.*, (2020): The six stage fifth-Order explicit Runge-Kutta method of system of first order IVP is given by:

$$y_{i+1} = y_i + \frac{1}{90}(7k_1 + 32k_2 + 12k_3 + 32k_4 + 7k_5)$$

$$z_{i+1} = z_i + \frac{1}{90}(7l_1 + 32l_2 + 12l_3 + 32l_4 + 7l_5)$$

Where the parameters

$$k_1 = f(x_i, y_i, z_i)$$

$$l_1 = f(x_i, y_i, z_i)$$

$$k_2 = f\left(x_i + \frac{1}{2}, y_i + \frac{k_1}{2}, z_i + \frac{l_1}{2}\right)$$

$$l_2 = f\left(x_i + \frac{1}{2}, y_i + \frac{k_1}{2}, z_i + \frac{l_1}{2}\right)$$

$$k_3 = f\left(x_i + \frac{1}{4}, y_i + \frac{1}{16}(3k_1 + k_2), z_i + \frac{1}{16}(3l_1 + l_2)\right)$$

$$l_3 = f\left(x_i + \frac{1}{4}, y_i + \frac{1}{16}(3k_1 + k_2), z_i + \frac{1}{16}(3l_1 + l_2)\right)$$

$$k_4 = f\left(x_i + \frac{3}{4}, y_i + \frac{k_3}{2}, z_i + \frac{l_3}{2}\right)$$

$$l_4 = f\left(x_i + \frac{3}{4}, y_i + \frac{k_3}{2}, z_i + \frac{l_3}{2}\right)$$

$$k_5 = f\left(x_i + \frac{3}{4}, y_i + \frac{1}{16}(3k_2 + 6k_3 + 9k_4), z_i + \frac{1}{16}(3l_2 + 6l_3 + 9l_4)\right)$$

$$l_5 = f\left(x_i + \frac{3}{4}, y_i + \frac{1}{16}(3k_2 + 6k_3 + 9k_4), z_i + \frac{1}{16}(3l_2 + 6l_3 + 9l_4)\right)$$

$$k_6 = f\left(x_i + 1, y_i + \frac{1}{7}(k_1 + 4k_2 + 6k_3 + 12k_4 + 8k_5), z_i + \frac{1}{7}(l_1 + 4l_2 + 6l_3 + 12l_4 + 8l_5)\right)$$

$$l_6 = f\left(x_i + 1, y_i + \frac{1}{7}(k_1 + 4k_2 + 6k_3 + 12k_4 + 8k_5), z_i + \frac{1}{7}(l_1 + 4l_2 + 6l_3 + 12l_4 + 8l_5)\right)$$

Definition 1.5 (Stability) GashuGamachis., (2019): Methods that satisfy the roots condition in which $\lambda = 1$ is the only root of the characteristic equation with magnitude one are called strongly stable.

a) **(Consistency)**: Robaet *al.*, (2017): The method is consistent, if the local truncation error $\tau_k(h) \rightarrow 0$ as $h \rightarrow 0$

b) **(Accuracy)**: Accuracy refers to the closeness of numerical solution to the exact solution (Ibrahim W., 2018).

Definition 1.6 (Convergence) Ibrahim., (2018): Convergence of a numerical method means that the numerical solution approaches to true solution as step sizes are reduced. The necessary and sufficient conditions for convergence are consistency and stability

1.6.4. Error of numerical method

a. **Error (E)**: If $y(x_i)$ is the exact or true value of the quantity and y_i is its approximate value, then $y(x_i) - y_i$ is called error

b. **Absolute error**: The absolute of the given error is called absolute error which is denoted by $||E||$ (Dukkipati V. R., 2010)

$||E|| = |y(x_i) - y_i|$, where $x_i = x_0 + i \cdot h$, $i = 1, 2, \dots, N$ and h is step size

c. **Truncation Error**: Error which is caused by using approximate formulae in computations such as the ones that arises when a truncated infinite series is used (Trivandrum, 2012)

Two types of truncation errors occur.

d. **Local truncation error**: Local truncation error is simply the difference between true value and approximate value from x_i to x_{i+1} (Kroshko A, 2018)

e. **Global truncation error (maximum absolute error)**: Global truncation error is error evaluated at time x_{i+1} (Kroshko A, 2018)

1.7. Thesis Organization

This thesis is subdivided into five chapters. The first chapter contains the background of the study which is written to indicate applicability of Initial Value Problems of Bratu-type equations in real life and the most popularity of predictor corrector method to solve real live applicable of some differential equations.

The ambiguity to solve Bratu- type equations using some numerical method is stated in statement of problem and eighth order predictor –corrector method is proposed to overcome. The objectives which are not achieved yet were written under this chapter and to win them is stated as significance of the study.

The particularity of the problem and method is also written under scope of the study and mathematical preliminary concepts which are used in the other body of this thesis are also written under this chapter. Secondly, Literature review is written under chapter two to illustrate the focus, gap, parameters and finding of recently addressed numerical method to find numerical solution of Biratu type equations via debates made between mathematician professions.

Again, next to chapter two study areas and period, study design, sources of information, mathematical procedure and ethical considerations are listed under the topic research methodology in chapter three. As well as, chapter four is written next to chapter three on the progress terms of this thesis definition of the problem, description of the method, derivation of eighthorder predictor-corrector method, stability and convergence analysis, results via model example and error analysis of method are briefly explained by following mathematical procedure given in chapter three to achieve objectives of this thesis we mentioned and discussion will be made at the end of this chapter.

Finally, chapter five is written on summary, conclusion, recommendation and future works to illustrate achievement of the study.

CHAPTER TWO

REVIEW OF RELATED LITERATURE

Differential equations play a fundamental role in describing many phenomena encountered in science, engineering, and applied mathematics. They establish mathematical relationships between an unknown function and its derivatives with respect to one or more independent variables. Depending on the number of independent variables involved, differential equations are generally categorized as ordinary differential equations (ODEs) and partial differential equations (PDEs). Furthermore, ODEs are commonly classified into initial value problems (IVPs) and boundary value problems (BVPs) based on the nature of the conditions imposed on the solution.

In practical applications, obtaining exact analytical solutions for differential equations is often difficult. Consequently, numerical techniques have become essential tools for approximating solutions. Over the years, numerous numerical methods have been developed, including Taylor series methods, Euler methods, Runge–Kutta methods, and predictor–corrector techniques. Among these approaches, Runge–Kutta methods are widely recognized for providing a suitable balance between computational efficiency and solution accuracy (Butcher, 2008). Likewise, predictor–corrector methods have received considerable attention because of their ability to achieve high-order accuracy while maintaining favorable stability characteristics (Shampine, 1994).

One of the most extensively studied nonlinear boundary value problems is the Bratu–Gelfand equation, commonly referred to as the Bratu problem. This problem has attracted significant interest due to its relevance in several scientific and engineering fields. According to Abukhaled *et al.*, (2012), Bratu-type equations arise in mathematical models associated with thermal combustion phenomena, heat transfer processes involving radioactive materials, nanotechnology applications, and astrophysical investigations such as Chandrasekhar’s model of cosmic expansion. Similar equations are also encountered in nonlinear diffusion studies and chemical reaction systems.

Various analytical and numerical techniques have been proposed for solving Bratu-type equations. For instance, Wazwaz (2005) investigated the application of the Adomian Decomposition Method (ADM) and the Variational Iteration Method (VIM) for obtaining approximate solutions of nonlinear problems. Similarly, Batiha *et al.*, (2007) utilized the Homotopy Perturbation Method (HPM) to address Bratu-type equations effectively. Another notable contribution was made by He (2006), who introduced the Homotopy Analysis Method (HAM) as a flexible framework for solving nonlinear differential equations. These studies demonstrate the effectiveness of modern approximation techniques in dealing with nonlinear mathematical models.

In their study, Debela *et al.*, (2017) employed a higher-order Runge–Kutta method to obtain numerical solutions for second-order Bratu-type initial value problems. By comparing the numerical and exact solutions, they observed that the method provided accurate approximations with reduced errors. Based on the results, the authors affirmed the suitability of higher-order Runge–Kutta methods for handling nonlinear differential equations.

In his work, Mengistu (2017) utilized a Predictor–Corrector approach to obtain approximate solutions for second-order Bratu-type problems. The reliability of the method was assessed through a comparison of numerical and analytical results. The close agreement between the two sets of solutions suggested that the method possesses desirable accuracy and convergence properties, making it a practical option for handling Bratu-type equations.

The increasing availability of advanced computational resources has further enhanced the applicability of numerical methods. Software environments such as MATLAB, Mathematica, and Maple provide powerful platforms for implementing high-order numerical algorithms and examining their convergence and stability behavior. Research conducted at *Wallaga University* has also highlighted the importance of higher-order predictor–corrector methods in solving nonlinear ordinary differential equations, particularly with respect to accuracy, convergence, and computational performance (WallagaUniversity, 2021).

The present study focuses on the derivation and implementation of an eighth-order predictor–corrector method for solving second-order initial value problems associated with Bratu-type equations.

In addition, the convergence and stability properties of the proposed numerical scheme are investigated to evaluate its effectiveness and suitability for scientific and engineering computations.

Although numerous numerical methods have been developed for solving Bratu-type differential equations, including the Adomian Decomposition Method, Variational Iteration Method, Homotopy Perturbation Method, Homotopy Analysis Method, Runge–Kutta methods, and predictor–corrector techniques, there remains a need for numerical schemes that provide higher-order accuracy with improved convergence and stability properties. Previous studies such as Debela *et al.*, (2017) and Mengistu (2017) focused on higher-order Runge–Kutta methods and predictor–corrector methods, respectively; however, they did not develop or investigate an eighth-order predictor–corrector scheme for second-order Bratu-type initial value problems. Furthermore, limited attention has been given to a detailed analysis of the convergence and stability characteristics of such high-order predictor–corrector methods. Therefore, the development and analysis of an eighth-order predictor–corrector method for solving second-order Bratu-type differential equations constitute the main research gap addressed in this study.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1. Study Area and Period

The study was conducted in the Department of Mathematics under the College of Natural and Computational Sciences at Wallaga University from February 2025 to May 2026.

3.2. Study Design

This research employed both documentary review and experimental computational approaches. Relevant literature was reviewed, and numerical experiments were performed using MATLAB.

3.3. Sources of Information

The information used in this study was obtained from textbooks, journal articles, research papers, online resources, and numerical outputs generated through MATLAB programming.

3.4. Mathematical Procedure

To achieve the stated objectives, the study will be followed the following procedures:

- 1) Identification of the research problem.
- 2) Development of the eighth-order predictor–corrector scheme.
- 3) Stability analysis of the proposed method.
- 4) Convergence analysis of the developed numerical algorithm.
- 5) MATLAB implementation of the numerical method.
- 6) Validation of the method using model examples.
- 7) Presentation of numerical results through tables and graphical illustrations.
- 8) Comparison of the proposed method with existing numerical methods.

3.5. Ethical Considerations

Ethical principles were carefully observed throughout the research process. All materials collected from books, journals, and online sources were properly acknowledged through citation and referencing. Academic honesty and integrity were maintained during the preparation and presentation of the thesis.

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1. Definition of the Problem

The Bratu–type equation is presented as

$$y''(x) + \lambda e^{y(x)} = 0, \quad 0 < x < 1. \quad (4.1)$$

Subject to the initial condition $y(0) = \alpha$, $y(\alpha) = \gamma$ where α and γ are constant numbers for $y(x)$ is unknown function.

The classical Bratu BVP

$$y''(x) + \lambda e^{y(x)} = 0, \quad y(0) = 0, y(1) = 0$$

Can be interpreted as a family of IVPs

$$y''(x) + \lambda e^{y(x)} = 0, \quad y(0) = 0, y(1) = s$$

Where the unknown initial slope s is adjusted until the terminal condition $y(1)=0$ is satisfied. Thus, the BVP is transformed into an IVP through the shooting parameter s . The multiple-solution nature of the Bratu BVP corresponds to the possibility of multiple values of s producing the required boundary condition.

The mathematical modeling of many scientific and engineering processes frequently results in Bratu-type differential equations. These equations are commonly formulated as second-order initial value problems and can be solved using numerical techniques such as the eighth-order predictor-corrector method. Mathematical models provide a detailed description of physical phenomena and help researchers understand their behavior under different conditions.

Finding solutions to real-world problems often requires solving Bratu-type equations, making them an important area of study in applied mathematics. In many practical situations, obtaining exact analytical solutions for differential equations is difficult or impossible. As a

result, numerical methods have become essential tools for generating approximate solutions with a high degree of accuracy.

For a second-order initial value problem, the objective is to determine the dependent variable y as a function of the independent variable x , subject to specified initial conditions. Bratu-type equations are generally categorized into two groups: initial value problems (IVPs) and boundary value problems (BVPs). The present study focuses specifically on the numerical solution of Bratu-type initial value problems.

Among the available numerical techniques, predictor-corrector methods are widely used because of their efficiency and accuracy. These methods operate through two successive stages. In the first stage, known as the predictor step, an initial approximation of the solution is generated. In the second stage, called the corrector step, the predicted value is refined to obtain a more accurate approximation. By repeatedly applying these two stages, the predictor-corrector approach produces highly accurate numerical solutions for Bratu-type differential equations.

The mathematical modeling of many scientific and engineering processes frequently results in Bratu-type differential equations. These equations are commonly formulated as second-order initial value problems and can be solved using numerical techniques such as the eighth-order predictor-corrector method.

Mathematical models provide a detailed description of physical phenomena and help researchers understand their behavior under different conditions.

for Bratu-type differential equations.

For example: consider the initial value problem in equation (4.1). Using simple Euler's and modified Euler's method, we can write down a simple predictor- corrector (P-C) to solve a problem in equation (2) as follows:-

Predictor (p): $y_{n+1}^{(p)} = y_n + hf(x_n, y_n)$ (4.2)

Corrector(c): $y_{n+1} = y_n + \frac{h}{2} [f(x_n, y_n) + f(x_{n+1}, y_{n+1}^p)]$ (4.3)

where h is the step length. Most of the second order initial value problems of Bratu-type equations are derived from real life applications. Numerical method for the solution of second order Initial value problem of Bratu-type equation is a part of numerical analysis which describes the numerical solution of ordinary differential equations (ODEs). Numerical methods form an important part of solving second order Initial value problem of Bratu-type equation. In most especially in cases where there is no closed form solution. There are several numerical methods proposed to solve second order Initial value problem of Bratu-type equation, approximately. However, the matter is getting better accurate numerical solutions.

4.2 Development of the Predictor Method

Consider k data values $(x_i, f_i^{(j)}), (x_{i-1}, f_{i-1}^{(j)}), \dots, (x_{i-k+1}, f_{i-k+1}^{(j)})$, for $j=1, 2$. For this data we fit Newton's backward difference interpolating polynomial of degree $k-1$ and obtain

$$\begin{aligned} f^{(j)}(x_i + n) = & f_i^{(j)} + (n-1)\nabla f_i^{(j)} + \frac{n(n-1)}{2!}\nabla^2 f_i^{(j)} + \frac{n(n-1)(n+1)}{3!}\nabla^3 f_i^{(j)} \\ & + \frac{n(n-1)(n+1)(n+2)}{4!}\nabla^4 f_i^{(j)} + \frac{n(n-1)(n+1)(n+2)(n+3)}{5!}\nabla^5 f_i^{(j)} + \\ & \frac{n(n-1)(n+1)(n+2)(n+3)(n+4)}{6!}\nabla^6 f_i^{(j)} + \frac{n(n-1)(n+1)\dots(n+k-2)}{k!}\nabla^k f_i^{(j)} \quad (4.4) \end{aligned}$$

This is the Newton's backward interpolation formula.

Note that $n = \frac{x - x_i}{h}$ and

$$\tau_k^{(j)} = \frac{n(n+1)\dots(n+k-2)}{k!} \nabla^k f_i^{(j)}(\varepsilon) \quad (4.5)$$

Is the error term where ε lies in some interval containing the points $x_i, x_{i-1}, \dots, x_{i-k+1}$, and x .

Using (4.5) and (4.6) we get:

$$\begin{aligned} y_{i+1}^{(j)} = & y_i^{(j)} + \int_{x_i}^{x_{i+1}} [f_i^{(j)} + n\nabla f_i^{(j)} + \frac{n(n+1)}{2!}\nabla^2 f_i^{(j)} + \frac{n(n+1)(n+2)}{3!}\nabla^3 f_i^{(j)} \\ & + \frac{n(n+1)(n+2)(n+3)}{4!}\nabla^4 f_i^{(j)} + \frac{n(n+1)(n+2)(n+3)(n+4)}{5!}\nabla^5 f_i^{(j)} \\ & + \frac{n(n+1)(n+2)(n+3)(n+4)(n+5)}{6!}\nabla^6 f_i^{(j)} + \frac{n(n+1)(n+2)(n+3)(n+4)(n+5)(n+6)}{7!}\nabla^7 f_i^{(j)} \\ & + \dots] dx \end{aligned}$$

Putting $x = x_i + n \rightarrow dx = dn$

As $x \rightarrow x_i, n \rightarrow 0$, as $x \rightarrow x_{i+1}, n \rightarrow 1$

$$\begin{aligned}
y_{i+1}^{(j)} = & y_i^{(j)} + \int_0^1 [f_i^{(j)} + n \nabla f_i^{(j)} + \frac{n(n+1)}{2} \nabla^2 f_i^{(j)} + \frac{n(n+1)(n+2)}{6} \nabla^3 f_i^{(j)} \\
& + \frac{n(n+1)(n+2)(n+3)}{24} \nabla^4 f_i^{(j)} \\
& + \frac{n(n+1)(n+2)(n+3)(n+4)}{120} \nabla^5 f_i^{(j)} \\
& + \frac{n(n+1)(n+2)(n+3)(n+4)(n+5)}{720} \nabla^6 f_i^{(j)} \\
& + \frac{n(n+1)(n+2)(n+3)(n+4)(n+5)(n+6)}{5040} \nabla^7 f_i^{(j)} \dots] dn \\
& + \frac{n(n+1)(n+2)(n+3)(n+4)(n+5)(n+6)(n+7)}{40320} \nabla^8 f_i^{(j)} (4.6)
\end{aligned}$$

$$\begin{aligned}
\int_0^1 n dn &= \frac{1}{2}, \quad \int_0^1 n(n+1) dn = \int_0^1 (n^2 + n) = \frac{5}{6}, \\
\int_0^1 n(n+1)(n+2) dn &= \int_0^1 (n^3 + 3n^2 + 2n) = \frac{9}{4} \\
\int_0^1 n(n+1)(n+2)(n+3) dn &= \int_0^1 (n^4 + 6n^3 + 11n^2 + 6n) = \frac{251}{30}, \\
\int_0^1 n(n+1)(n+2)(n+3)(n+4) dn &= \int_0^1 (n^5 + 10n^4 + 35n^3 + 50n^2 + 24n) = \frac{475}{12} \\
\int_0^1 n(n+1)(n+2)(n+3)(n+4)(n+5) dn \\
&= \int_0^1 (n^6 + 15n^5 + 85n^4 + 225n^3 + 274n^2 + 120n) = \frac{19087}{84} \\
\int_0^1 n(n+1)(n+2)(n+3)(n+4)(n+5)(n+6) dn \\
&= \int_0^1 (n^7 + 21n^6 + 175n^5 + 735n^4 + 1624n^3 + 1764n^2 + 720n) = \frac{36799}{24}
\end{aligned}$$

$$\int_0^1 n(n+1)(n+2)(n+3)(n+4)(n+5)(n+6)(n+7)dn$$

$$\int_0^1 (n^8 + 28n^7 + 322n^6 + 1960n^5 + 6769n^4 + 13132n^3 + 12368n^2 + 504n) = \frac{1070017}{90}$$

Hence, from Eq. (1.5), we get:

$$y_{i+1}^{(j)} = y_i^{(j)} + [f_i^{(j)} + \frac{1}{2}\nabla f_i^{(j)} + \frac{5}{12}\nabla^2 f_i^{(j)} + \frac{3}{8}\nabla^3 f_i^{(j)} + \frac{251}{720}\nabla^4 f_i^{(j)} + \frac{475}{1440}\nabla^5 f_i^{(j)} + \frac{19087}{60480}\nabla^6 f_i^{(j)} + \frac{36799}{120960}\nabla^7 f_i^{(j)} + \frac{1070017}{90}\nabla^8 f_i^{(j)}] \quad (4.7).$$

For $j = 1, 2$

We obtain the error term as:

$$\tau_k^p = {}_{k+1} \int_0^1 \frac{n(n+1)(n+2) \dots (n+k-1)}{k!} f^{(k)}(\xi_k) d \quad (4.8).$$

By choosing different values for k , we get different methods. But for this particular study, we choose the value for $k = 8$ which is of seventh method. Thus, we get:

$$y_{i+1}^{(j)} = y_i^{(j)} + \left[f_i^{(j)} + \frac{1}{2}\nabla f_i^{(j)} + \frac{5}{12}\nabla^2 f_i^{(j)} + \frac{3}{8}\nabla^3 f_i^{(j)} + \frac{251}{720}\nabla^4 f_i^{(j)} + \frac{475}{1440}\nabla^5 f_i^{(j)} + \frac{19087}{60480}\nabla^6 f_i^{(j)} \right] + \frac{5257}{17280}\nabla^7 + \tau_8 \quad (4.9).$$

Expressing $\nabla f_i^{(j)}, \nabla^2 f_i^{(j)}, \nabla^3 f_i^{(j)}, \nabla^4 f_i^{(j)}, \nabla^5 f_i^{(j)}, \nabla^6 f_i^{(j)}, \nabla^7 f_i^{(j)}$ in terms of the function, we get

$$\begin{aligned} y_{i+1}^{(j)} = y_i^{(j)} + & \left[f_i^{(j)} + \frac{1}{2}(f_i^{(j)} - f_{i-1}^{(j)}) + \frac{5}{12}(f_i^{(j)} - 2f_{i-1}^{(j)} + f_{i-2}^{(j)}) \right. \\ & + \frac{3}{8}(f_i^{(j)} - 3f_{i-1}^{(j)} + 3f_{i-2}^{(j)} - f_{i-3}^{(j)}) \\ & + \frac{251}{720}(f_i^{(j)} - 4f_{i-1}^{(j)} + 6f_{i-2}^{(j)} - 4f_{i-3}^{(j)} + f_{i-4}^{(j)}) \\ & + \frac{475}{1440}(f_i^{(j)} - 5f_{i-1}^{(j)} + 10f_{i-2}^{(j)} - 10f_{i-3}^{(j)} + 5f_{i-4}^{(j)} - f_{i-5}^{(j)}) \\ & + \frac{19087}{60480}(f_i^{(j)} - 6f_{i-1}^{(j)} + 15f_{i-2}^{(j)} - 20f_{i-3}^{(j)} + 15f_{i-4}^{(j)} - 6f_{i-5}^{(j)} + f_{i-6}^{(j)}) + \frac{5257}{17280}(f_i^{(j)} - f_{i-1}^{(j)} + \\ & \left. 21f_{i-2}^{(j)} - 35f_{i-3}^{(j)} + 35f_{i-4}^{(j)} - 21f_{i-5}^{(j)} + 7f_{i-6}^{(j)} - f_{i-7}^{(j)}) \right] + \tau_8 \end{aligned}$$

For j=1, 2

$$y_{i+1}^{(j)} = y_i^{(j)} + \frac{1}{120960} (434241f_i^{(j)} - 11521f_{i-1}^{(j)} + 2183877f_{i-2}^{(j)} - 2664477f_{i-3}^{(j)} + 2102243f_{i-4}^{(j)} - 1041723f_{i-5}^{(j)} + 295767f_{i-6}^{(j)} - 36799f_{i-7}^{(j)}) + \tau_7 \quad (4.10)$$

This is called a predictor formula.

$$\tau_8^{(j)} = \frac{1070017}{3628800} h^9 u^{(8)}(\epsilon) \text{ for } j=1, 2 \text{ is the local truncation error (4.11)}$$

and is called **eighth order predictor method**.

4.3. Development of Corrector Method

Consider the k+1 data values $(x_{i+1}, f_{i+1}^{(j)})$, $(x_i, f_i^{(j)})$, ..., $(x_{i-k+1}, f_{i-k+1}^{(j)})$, for j=1,2. Which include the current data points. For this data we fit newtons backward difference interpolating polynomial of degree k and obtain

$$\begin{aligned} f^{(j)}(x_i + n) = & f_{i+1}^{(j)} + (n-1)\nabla f_{i+1}^{(j)} + \frac{n(n-1)}{2!} \nabla^2 f_{i+1}^{(j)} + \frac{n(n-1)(n+1)}{3!} \nabla^3 f_{i+1}^{(j)} \\ & + \frac{n(n-1)(n+1)(n+2)}{4!} \nabla^4 f_{i+1}^{(j)} + \frac{n(n-1)(n+1)(n+2)(n+3)}{5!} \nabla^5 f_{i+1}^{(j)} \\ & + \frac{n(n-1)(n+1)(n+2)(n+3)(n+4)}{6!} \nabla^6 f_{i+1}^{(j)} + \\ & \frac{n(n-1)(n+1)(n+2)(n+3)(n+4)(n+5)}{7!} \nabla^7 f_{i+1}^{(j)} + \dots + \frac{n(n-1)(n+1)\dots(n+k-2)}{k!} \nabla^k f_{i+1}^{(j)} \end{aligned} \quad (4.12)$$

This is the newton's backward interpolation formula.

Note that $n = \frac{x-x_i}{h}$ and

$$\tau_k^{(j)} = \frac{n(n-1)(n+1)\dots(n+k-2)}{k!} h^k f^{(k)}(\epsilon) \quad (4.13)$$

Is the error term where ϵ lies in some interval containing the points $x_i, x_{i-1}, \dots, x_{i-k+1}$, and x .

Using (4.12) and (4.13) we get:

$$\begin{aligned} y_{i+1}^{(j)} = & y_i^{(j)} + \int_{x_i}^{x_{i+1}} [f_{i+1}^{(j)} + (n-1)\nabla f_{i+1}^{(j)} + \frac{n(n-1)}{2!} \nabla^2 f_{i+1}^{(j)} + \frac{n(n-1)(n+1)}{3!} \nabla^3 f_{i+1}^{(j)} \\ & + \frac{n(n-1)(n+1)(n+2)}{4!} \nabla^4 f_{i+1}^{(j)} + \frac{n(n-1)(n+1)(n+2)(n+3)}{5!} \nabla^5 f_{i+1}^{(j)} \end{aligned}$$

$$\begin{aligned}
& + \frac{n(n-1)(n+1)(n+2)(n+3)(n+4)}{6!} \nabla^6 f_{i+1}^{(j)} \\
& + \frac{n(n-1)(n+1)(n+2)(n+3)(n+4)(n+5)}{7!} \nabla^7 \dots \\
& + \frac{n(n-1)(n+1) \dots (n+k-2)}{k!} \nabla^k f_{i+1}^{(j)}] dx
\end{aligned}$$

As $x \rightarrow x_i$, $n \rightarrow 0$, as $x \rightarrow x_{i+1}$, $n \rightarrow 1$

$$\begin{aligned}
y_{i+1}^{(j)} &= y_i^{(j)} + \int_0^1 [f_{i+1}^{(j)} + (n-1) \nabla f_{i+1}^{(j)} + \frac{n(n-1)}{2} \nabla^2 f_{i+1}^{(j)} \\
& + \frac{n(n-1)(n+1)}{6} \nabla^3 f_{i+1}^{(j)} + \frac{n(n-1)(n+1)(n+2)}{24} \nabla^4 f_{i+1}^{(j)} \\
& + \frac{n(n-1)(n+1)(n+2)(n+3)}{120} \nabla^5 f_{i+1}^{(j)} \\
& + \frac{n(n-1)(n+1)(n+2)(n+3)(n+4)}{720} \nabla^6 f_{i+1}^{(j)} \\
& + \frac{n(n-1)(n+1)(n+2)(n+3)(n+4)(n+5)}{5040} \nabla^7 f_{i+1}^{(j)} \\
& + \frac{n(n-1)(n+1)(n+2)(n+3)(n+4)(n+5)(n+6)}{40320} \nabla^8 f_{i+1}^{(j)} \dots \\
& \dots \dots + \frac{n(n-1)(n+1) \dots (n+k-2)}{k!} \nabla^k f_{i+1}^{(j)}] dx
\end{aligned} \tag{4.14}$$

$$\int_0^1 (n-1) ds = \frac{1}{2}, \int_0^1 (n-1) n dn = \int_0^1 (n^2 - n) dn = \frac{1}{6},$$

$$\int_0^1 (n-1) n(n+1) dn = \int_0^1 (n^3 - n) dn = \frac{1}{4}$$

$$\int_0^1 (n-1) n(n+1)(n+2) dn = \int_0^1 (n^4 + 2n^3 - n^2 - 2n) dn = \frac{19}{30},$$

$$\int_0^1 (n-1) n(n+1)(n+2)(n+3) dn = \int_0^1 (n^5 + 5n^4 + 5n^3 - 5n^2 - 6n) dn = \frac{9}{4}$$

$$\begin{aligned}
& \int_0^1 (n-1)n(n+1)(n+2)(n+3)(n+4)dn \\
&= \int_0^1 (n^6 + 9n^5 + 25n^4 + 15n^3 - 26n^2 - 24n)dn = \frac{863}{84} \\
& \int_0^1 (n-1)n(n+1)(n+2)(n+3)(n+4)(n+5)dn \\
&= \int_0^1 (n^7 + 14n^6 + 70n^5 + 140n^4 + 49n^3 - 154n^2 - 120n)dn = \frac{1375}{24} \\
& \int_0^1 (n-1)n(n+1)(n+2)(n+3)(n+4)(n+5)(n+6)dn \\
&= \int_0^1 (n^8 + 13n^7 + 56n^6 + 70n^5 + 91n^4 - 199n^3 - 162n^2 - 121n)dn \\
&= \frac{33953}{90}
\end{aligned}$$

Hence from Eq. (15), we get:

$$\begin{aligned}
y_{i+1}^{(j)} = & y_i^{(j)} + \left[\frac{1}{2} \nabla f_{i+1}^{(j)} - \frac{1}{12} \nabla^2 f_{i+1}^{(j)} + \frac{1}{24} \nabla^3 f_{i+1}^{(j)} \right. \\
& \left. - \frac{19}{720} \nabla^4 f_{i+1}^{(j)} + \frac{27}{1440} \nabla^5 f_{i+1}^{(j)} - \frac{863}{60,480} \nabla^6 f_{i+1}^{(j)} - \frac{1375}{120,960} \nabla^7 f_{i+1}^{(j)} + \frac{5257}{17280} \nabla^8 f_{i+1}^{(j)} + \dots \right]
\end{aligned}
\tag{4.15}$$

For $j=1,2$

We obtain the error term as:

$$\begin{aligned}
& \tau_k^c \\
&= \int_0^1 \frac{n(n-1)(n+1)(n+2) \dots (n+k-1)}{(k+1)!} f^{(k)}(\xi_k) d
\end{aligned}
\tag{4.16}$$

By choosing different values for k , we get different methods. But for this particular study, we choose the value for $k = 7$ which is of seventh method. Thus, we get:

$$y_{i+1}^{(j)} = y_i^{(j)} + \left(f_{i+1}^{(j)} - \frac{1}{2} \nabla f_{i+1}^{(j)} + \frac{1}{12} \nabla^2 f_{i+1}^{(j)} - \frac{1}{24} \nabla^3 f_{i+1}^{(j)} \right.$$

$$\begin{aligned} & \frac{19}{720} \nabla^4 f_{i+1}^{(j)} - \frac{27}{1440} \nabla^5 f_{i+1}^{(j)} + \frac{863}{60,480} \nabla^6 f_{i+1}^{(j)} - \frac{1375}{120,960} \nabla^7 f_{i+1}^{(j)} + \frac{5257}{17280} \nabla^8 f_{i+1}^{(j)} + \tau_8 \\ (4.17) \end{aligned}$$

Expressing $\nabla f_{i+1}^{(j)}, \nabla^2 f_{i+1}^{(j)}, \nabla^3 f_{i+1}^{(j)}, \nabla^4 f_{i+1}^{(j)}, \nabla^5 f_{i+1}^{(j)}, \nabla^6 f_{i+1}^{(j)}, \nabla^7 f_{i+1}^{(j)}, \nabla^8 f_{i+1}^{(j)}$, in terms of the function, we get

$$\begin{aligned} y_{i+1}^{(j)} = y_i^{(j)} + & \left[\left(f_{i+1}^{(j)} - \frac{1}{2} (f_{i+1}^{(j)} - f_i^{(j)}) - \frac{1}{12} (f_{i+1}^{(j)} - 2f_i^{(j)} + f_{i-1}^{(j)}) \right. \right. \\ & \left. \left. - \frac{1}{24} (f_{i+1}^{(j)} - 3f_i^{(j)} + 3f_{i-1}^{(j)} - f_{i-2}^{(j)}) \right. \right. \\ & \left. \left. - \frac{19}{720} (f_{i+1}^{(j)} - 4f_i^{(j)} + 6f_{i-1}^{(j)} - 4f_{i-2}^{(j)} + f_{i-3}^{(j)}) \right. \right. \\ & \left. \left. - \frac{3}{160} (f_{i+1}^{(j)} - 5f_i^{(j)} + 10f_{i-1}^{(j)} - 10f_{i-2}^{(j)} + 5f_{i-3}^{(j)} - f_{i-4}^{(j)} \right. \right. \\ & \left. \left. - \frac{27}{1440} (f_{i+1}^{(j)} - 5f_i^{(j)} + 10f_{i-1}^{(j)} - 10f_{i-2}^{(j)} + 5f_{i-3}^{(j)} - 6f_{i-4}^{(j)} + f_{i-5}^{(j)}) \right. \right. \\ & \left. \left. - \frac{275}{24192} (f_{i+1}^{(j)} - 7f_i^{(j)} + 21f_{i-1}^{(j)} - 35f_{i-2}^{(j)} + 35f_{i-3}^{(j)} - 21f_{i-4}^{(j)} + 7f_{i-5}^{(j)} - f_{i-6}^{(j)} + \tau_7^{(j)}) \right) \right. \\ & \left. y_{i+1}^{(j)} = y_i^{(j)} + \frac{1}{120960} (36799f_{i+1}^{(j)} + 139849f_i^{(j)} - 121797f_{i-1}^{(j)} + 123133f_{i-2}^{(j)} \right. \\ & \left. - 88547f_{i-3}^{(j)} + 41499f_{i-4}^{(j)} - 11351f_{i-5}^{(j)} + 1375f_{i-6}^{(j)}) + \tau_7^{(j)} \right] \quad (4.18) \end{aligned}$$

This is corrector formula.

$$T_7^{(j)} = \frac{33953}{3628800} h^9 (f)^{(j)8}(\epsilon). \text{ For } j = 1, 2 \text{ is the local truncation error } (4.19)$$

and is called **eighth order corrector method**.

Remark 3.2. For using these methods, we require the starting values $y_1^{(j)}, y_2^{(j)}, y_3^{(j)}, y_4^{(j)}, y_5^{(j)}$ and $y_6^{(j)}$ for $j = 1, 2$. So in this study, we used the sixth stage fifth order Runge–Kutta method of system of first order IVP for the first six nodal points.

4.4. Stability and Convergence Analysis of the Developed Method

The eighth order predictor method in equation (4.11) can be expressed as:

$$y_{i+1}^{(j)} = y_i^{(j)} + F(x_i, y_{i+1}^{(j)}, y_i^{(j)}, \dots, y_{i-6}^{(j)})$$

Where

$$F(x_i, y_{i+1}^{(j)}, y_i^{(j)}, \dots, y_{i-6}^{(j)})$$

$$= \frac{1}{120960} (434241f_i^{(j)} - 1152169f_{i-1}^{(j)} + 2183877f_{i-2}^{(j)} - 266477f_{i-3}^{(j)} + 2102243f_{i-4}^{(j)} - 1041723f_{i-5}^{(j)} + 295767f_{i-6}^{(j)} - 36799f_{i-7}^{(j)} + \tau_8^{(j)})$$

In this case $k=8$ so we have $a_0^{(j)} = a_1^{(j)} = a_2^{(j)} = a_3^{(j)} = a_4^{(j)} = a_5^{(j)} = 0$ and $a_6^{(j)} = 1$, for $j=1, 2$ $y_{n+1} - y_n = 0$.

Therefore the coefficients are $\alpha_0 = 1, \alpha_1 = 1$, with all others zero. Hence, $\rho(\lambda) = \lambda^7 - 1$ or, depending on indexing convention

$$p(\lambda) = \lambda^7 - \lambda^6 = \lambda^6(\lambda - 1)$$

These are equivalent polynomials because multiplying by a power of λ does not change the nonzero roots.

For a 7-step predictor (which is usually called an eighth-order predictor because it uses seven previous values and attains order eight,

$$p(\lambda) = \lambda^7 - \lambda^6 = \lambda^6(\lambda - 1)$$

The roots are $\lambda^1 = 1$, and $\lambda^2 = \lambda^3 = \lambda^4 = \lambda^5 = \lambda^6 = \lambda^7 = 0$

$\lambda^2 = \lambda^3 = \lambda^4 = \lambda^5 = \lambda^6 = 0$ should be

$$\lambda^2 = \lambda^3 = \lambda^4 = \lambda^5 = \lambda^6 = \lambda^7 = 0$$

A . Zero-Stability Analysis: The root condition requires:

1. Every root satisfies $|\lambda^i| \leq 1$.
2. Any root on the unit circle is simple.

The roots are 1, 0, 0, 0, 0, 0, 0.

Hence, $|1| = 1$, $|0| < 1$. $|1| = 1$, The root at 1,1,1 has multiplicity one.

Therefore the method satisfies the root condition. Thus the predictor is **zero-stable**.

b. Strong Stability: A linear multistep method is strongly stable if:

1. $\lambda=1$ is a simple root;
2. All remaining roots satisfy $|\lambda^i| < 1$.

Since $\lambda^1 = 1$ and $\lambda^2 = \dots = \lambda^7 = 0$, all other roots lie strictly inside the unit disk. Hence the predictor is strongly stable.

Theorem 4.4.2 The eighth-order predictor method is strongly stable.

Proof. The first characteristic polynomial of the predictor method is

$$p(\lambda) = \lambda^7 - \lambda^6 = \lambda^6(\lambda - 1)$$

The roots are $\lambda^1 = 1, \lambda^2 = \lambda^3 = \lambda^4 = \lambda^5 = \lambda^6 = \lambda^7 = 0$

Since $|\lambda^i| < 1$ for $i = 2, \dots, 7$, $\lambda = 1$ is simple, the root condition is satisfied. Therefore the method is zero-stable. Moreover, all roots other than $\lambda=1$ lie strictly inside the unit circle. Hence the predictor method is strongly stable.

Theorem 4.4.3: The eighth-order corrector method is strongly stable.

Proof. The first characteristic polynomial of the corrector method is

$$p(\lambda) = \lambda^8 - \lambda^7 = \lambda^7(\lambda - 1)$$

The roots are $\lambda^1 = 1, \lambda^2 = \lambda^3 = \lambda^4 = \lambda^5 = \lambda^6 = \lambda^7 = 0$

Since $|\lambda^i| < 1$ for $i = 2, \dots, 8, \lambda = 1$

Since $|\lambda^i| < 1$ for $i = 2, \dots, 8$, $\lambda = 1$ is simple, the root condition is satisfied. Therefore the method is zero-stable. Moreover, all roots other than $\lambda=1$ lie strictly inside the unit circle. Hence the corrector method is strongly stable.

Definition 4.4.3: (Consistency) (Robaet *al.*, 2017): The method is consistent, if the local truncation error $\tau_k(\epsilon) \rightarrow 0$ as $\epsilon \rightarrow 0$

Theorem 4.4.3: The predictor equation (4.9) and the corrector equation (4.15) are both eighth order convergence.

From equations (4.11) and (4.19), we have

$$\tau_8^{(j)} = \frac{1070017}{3628800} \tau_7^{(j)} \text{ and } \tau_7^{(j)} = \frac{33953}{3628800} \tau_6^{(j)}(\epsilon)$$

$$\tau_k^{(j)}(\epsilon) \rightarrow 0 \text{ as } \epsilon \rightarrow 0, \text{ for } k=7, 8 \text{ and } j=1, 2$$

Therefore the predictor in equation (4.11) and corrector method in equation (4.19) are consistent and hence they are Eighth-order convergence.

4.5. Numerical Examples and Results

In order to demonstrate the validity and efficiency of the developed method, two Bratu-type equations subject to initial conditions were considered. Numerical experiments were conducted to investigate the accuracy and convergence behavior of the method. MATLAB was utilized for the implementation of the algorithm, computation of numerical results, and graphical visualization. The resulting plots were used to compare the numerical approximations with the exact solutions and to examine the corresponding errors.

The proposed eighth-order predictor-corrector method was applied to solve the problem numerically. The obtained numerical results were then compared with the exact solution to determine the accuracy and reliability of the method. Tables and graphical plots were used to present the comparison and to illustrate the effectiveness of the proposed numerical scheme.

Example 4.1 (Debelaet *al.*, 2017). Find the solution of Bratu-type initial value problem $y'' - e^{2y} = 0$, $0 < x < 1$ subject to the initial condition $y(0) = 0$, $y'(0) = 0$ with the exact solution $y(x) = \ln(\sec(x))$

Table 4.1: Comparison of exact solution and numerical solution of Example 4.1 for $h = 0.1$ using PCM8

x	Exact solution	Numerical solution	Absolute Error
---	----------------	--------------------	----------------

0	0	0	0
0.1	5.0084e-03	5.0084 e-03	2.4514e-09
0.2	2.0135e-02	2.0135 e-02	5.0353e-09
0.3	4.5692e-02	4.5692 e-02	7.9863e-09
0.4	8.2229e-02	8.2229 e-02	1.1616e-08
0.5	1.3058e-01	1.3058 e-01	1.6395e-08
0.6	1.9197e-01	1.9197 e-01	2.3102e-08
0.7	2.6809e-01	2.6809 e-01	3.3168e-08
0.8	3.6139e-01	3.6139 e-01	6.8282e-07
0.9	4.7544e-01	4.7544 e-01	2.1626e-06
1.0	6.1563e-01	6.1562 e-01	6.3936e-06

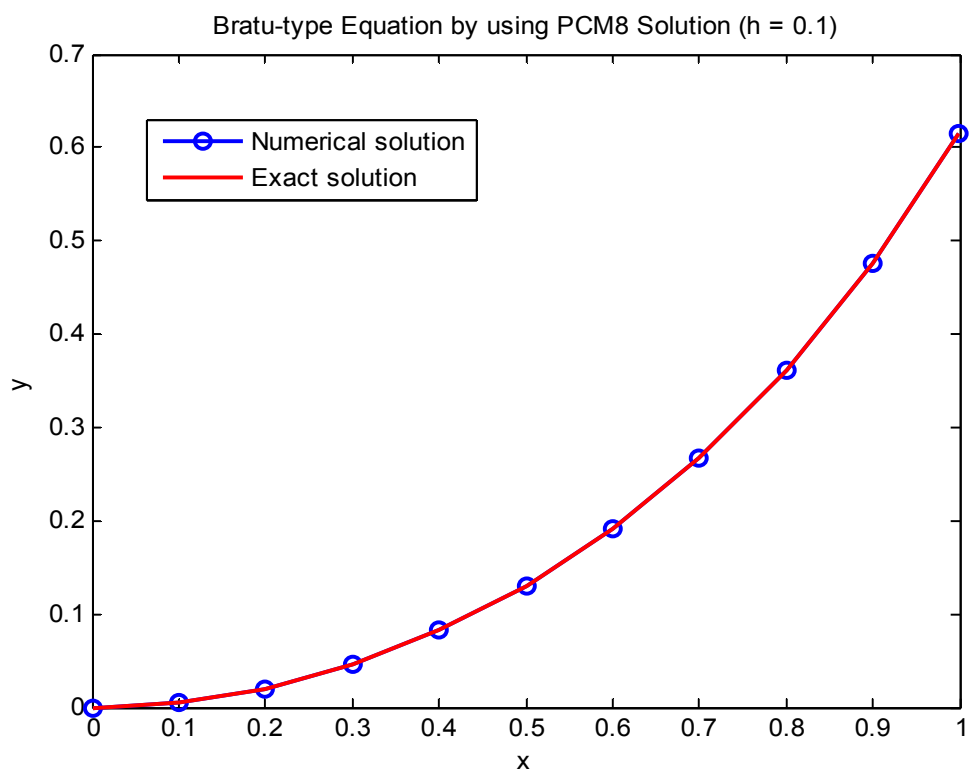


Figure 4.1:-The graph of exact and numerical solution of Example 4.1 When $N=10$, $h=0.1$

Table 4.2: Comparison of exact solution and numerical solution of Example 4.1 for $h = 0.01$ using PCM8

x	Exact solution	Numerical solution	Absolute Error
0	0	0	0
0.1	5.0084e-03	5.0084 e-03	1.7038e-14
0.2	6.0622e-03	6.0622 e-03	1.7244e-14
0.3	2.0135e-02	2.0135 e-02	1.7694e-14
0.4	4.5692e-02	4.5692 e-02	1.8180e-14

0.5	8.2229e-02	8.2229 e-02	1.9040e-14
0.6	1.3058e-01	1.3058 e-01	1.9346e-14
0.7	1.9197e-01	1.9197 e-01	1.8707e-14
0.8	2.6809e-01	2.6809 e-01	1.4544e-14
0.9	4.7544e-01	4.7544 e-01	4.5963e-14
1	6.1563e-01	6.1563 e-01	1.9840e-13

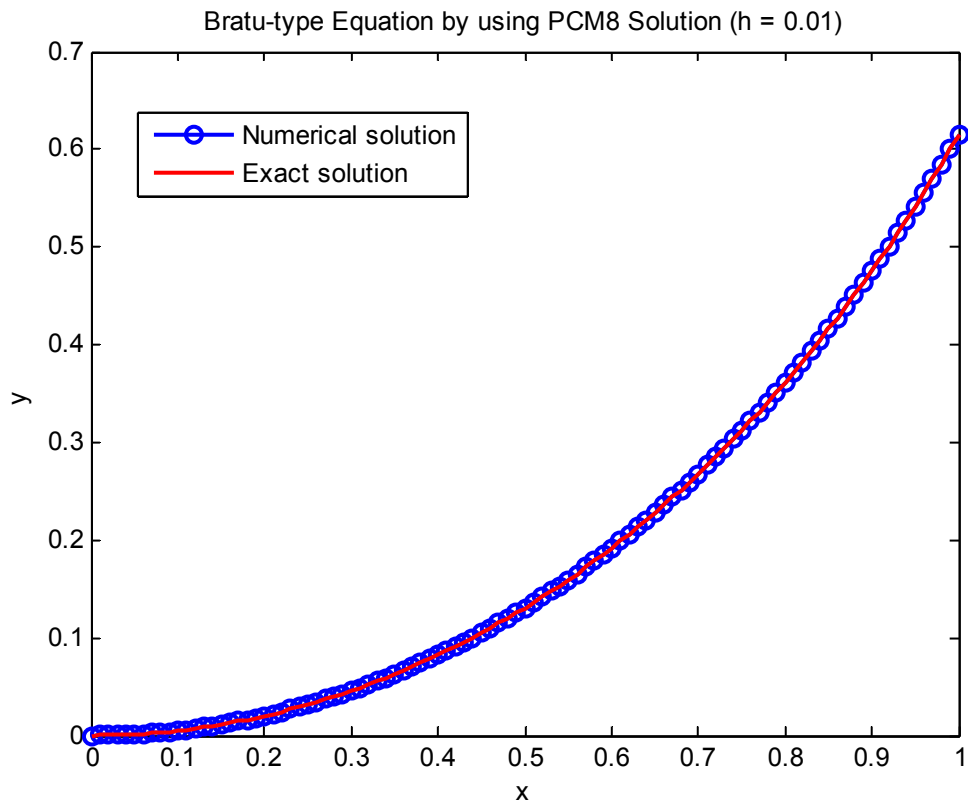


Figure 4.2:-The graph of exact and numerical solution of Example 4.1 when $N=100$, $h=0.01$

Table4.3: Comparison of exact solution and numerical solution of Example 4.1 for $h = 0.002$

X	Exact solution	Numerical solution	Absolute Error
0	0	0	0
0.1	5.0084e-03	5.0084 e-03	5.8981e-17
0.2	2.0135e-02	2.0135 e-02	6.9389e-17
0.3	4.5692e-02	4.5692 e-02	5.5511e-17

0.4	8.2229e-02	8.2229 e-02	1.5266e-16
0.5	1.3058e-01	1.3058 e-01	2.4980e-16
0.6	1.9197e-01	1.9197 e-01	1.6653e-16
0.7	2.6809e-01	2.6809 e-01	3.3307e-16
0.8	3.6139e-01	3.6139 e-01	2.2204e-16
0.9	4.7544e-01	4.7544 e-01	6.1062e-16
1.0	6.1563e-01	6.1563 e-01	7.7716e-16

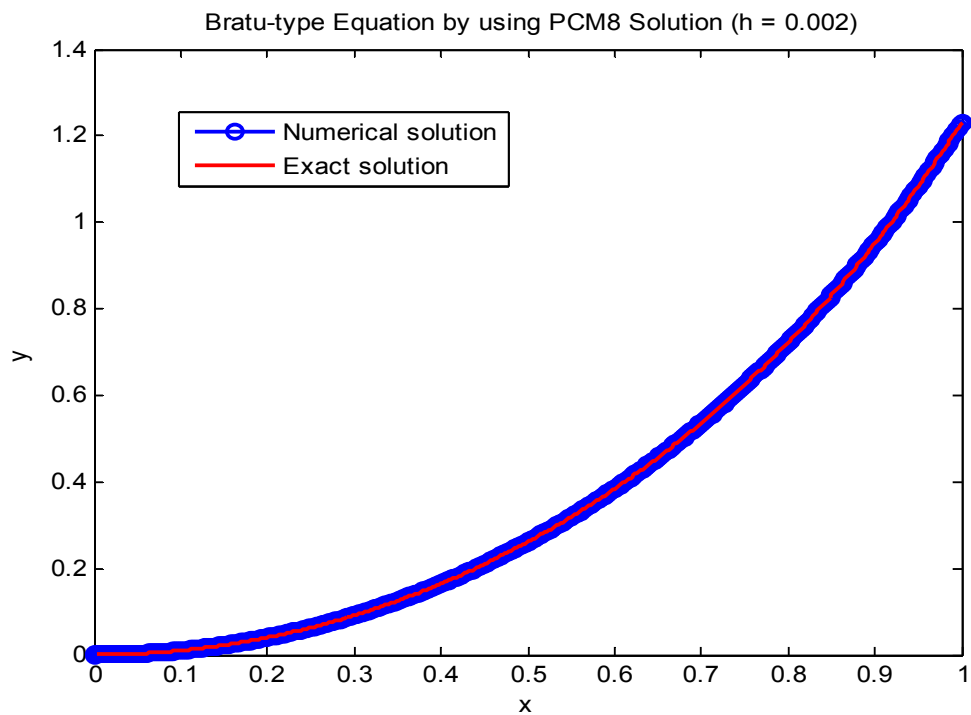


Figure 4.3:-The graph of exact and numerical solution of Example 4.1 when $N=100$, $h=0.01$

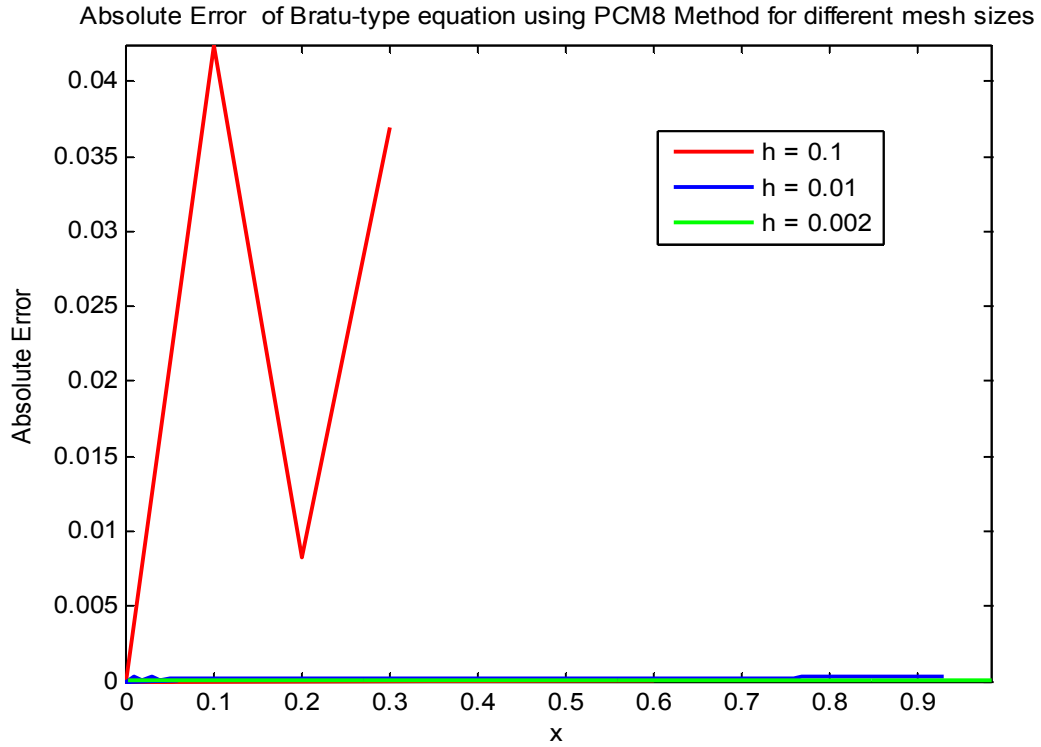


Figure 4.4:-The Error graph for $h=0.1$, $h=0.01$ and $h=0.002$ of Example 4.1.

Example 4.2 (Mekonnenet *al.*, 2020) Find the solution of Bratu-type initial value problem $y'' - 2e^y = 0$, $0 < x < 1$ subject to the initial condition $y(0) = 0$, $y'(0) = 0$ with the exact solution $y(x) = 2\ln(\cos(x))$

Table 4.4: Comparison of exact solution and numerical solution of Example 4.2 for $h = 0.1$ using PCM8

x	Exact Solution	Numerical Solution	Absolute Error
0	0	0	0
0.1	1.0017e-02	1.0017 e-02	4.9027e-09
0.2	4.0270e-02	4.0270 e-02	1.0071e-08
0.3	9.1383e-02	9.1383 e-02	1.5973e-08
0.4	1.6446e-01	1.6446 e-01	2.3233e-08
0.5	2.6117e-01	2.6117 e-01	3.2789e-08
0.6	3.8393e-01	3.8393 e-01	4.6204e-08
0.7	5.3617e-01	5.3617 e-01	6.6335e-08
0.8	7.2278e-01	7.2278 e-01	1.3656e-06
0.9	9.5088e-01	9.5088 e-01	4.3252e-06
1	1.2313e+00	1.2312e+00	1.2787e-05

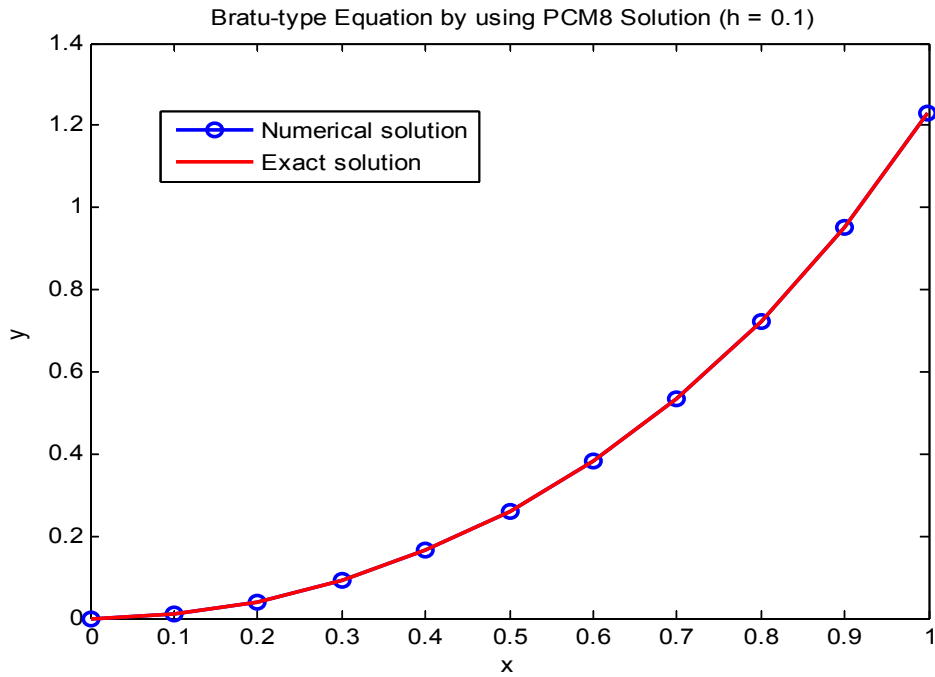


Figure 4.5:-The graph of exact and numerical solution of Example 4.1 when $N=10$, $h=0.1$

Table 4.5: Comparison of exact solution and numerical solution of Example 4.2 for $h = 0.01$ using PCM8

x	<i>Exact Solution</i>	<i>Numerical Solution</i>	<i>Absolute Error</i>
0	0	0	0
0.1	1.0017e-02	1.0017 e-02	3.4292e-14
0.2	4.0270e-02	4.0270 e-02	3.5263e-14
0.3	9.1383e-02	9.1383 e-02	3.6540e-14
0.4	1.6446e-01	1.6446 e-01	3.7942e-14
0.5	2.6117e-01	2.6117 e-01	3.8636e-14
0.6	3.8393e-01	3.8393 e-01	3.7359e-14
0.7	5.3617e-01	5.3617 e-01	2.9088e-14
0.8	7.2278e-01	7.2278 e-01	2.2204e-16
0.9	9.5088e-01	9.5088 e-01	9.1926e-14
1.0	1.2313e+00	1.2313 e+00	3.9679e-13

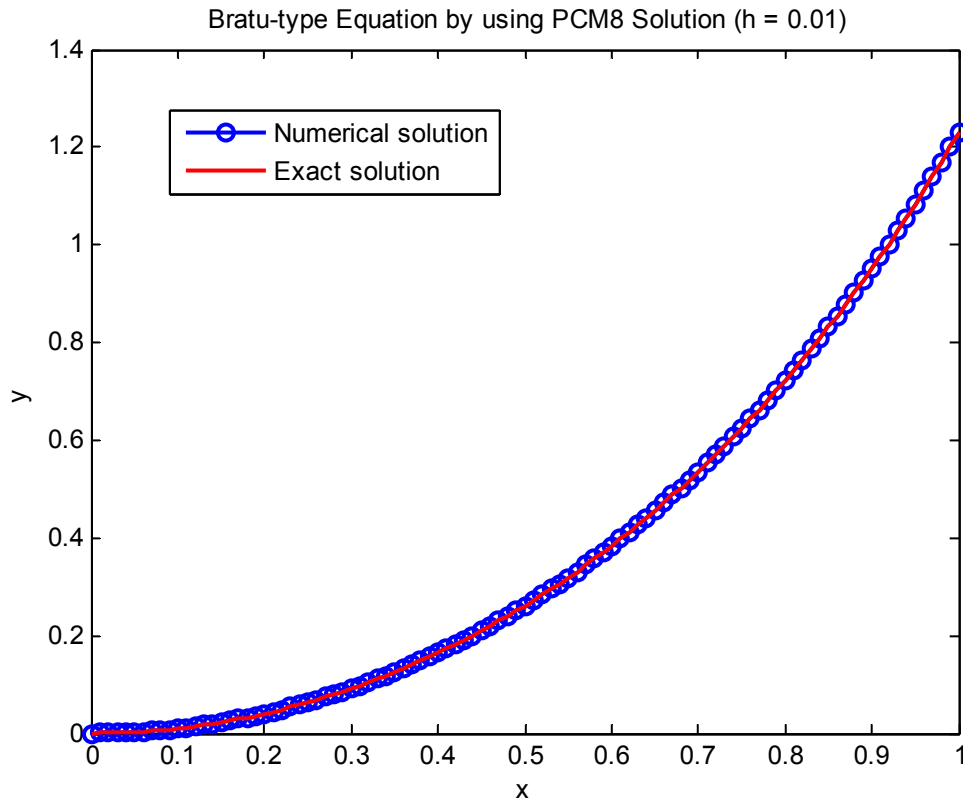


Figure 4.6:-The graph of exact and numerical solution of Example 4.2 when $N=100$, $h=0.01$

Table 4.6: Comparison of exact solution and numerical solution of Example 4.2 for $h = 0.002$ using PCM8

x	Exact Solution	Numerical Solution	Absolute Error
0	0	0	0
0.1	1.0017e-02	1.0017 e-02	1.1449e-16
0.2	4.0270e-02	4.0270 e-02	1.3878e-17
0.3	9.1383e-02	9.1383 e-02	6.9389e-17
0.4	1.6446e-01	1.6446 e-01	1.6653e-16
0.5	2.6117e-01	2.6117 e-01	2.7756e-16
0.6	3.8393e-01	3.8393 e-01	2.7756e-16
0.7	5.3617e-01	5.3617 e-01	5.5511e-16
0.8	7.2278e-01	7.2278 e-01	3.3307e-16
0.9	9.5088e-01	9.5088 e-01	1.2212e-15
1.0	1.2313e+00	1.2313 e+00	1.5543e-15

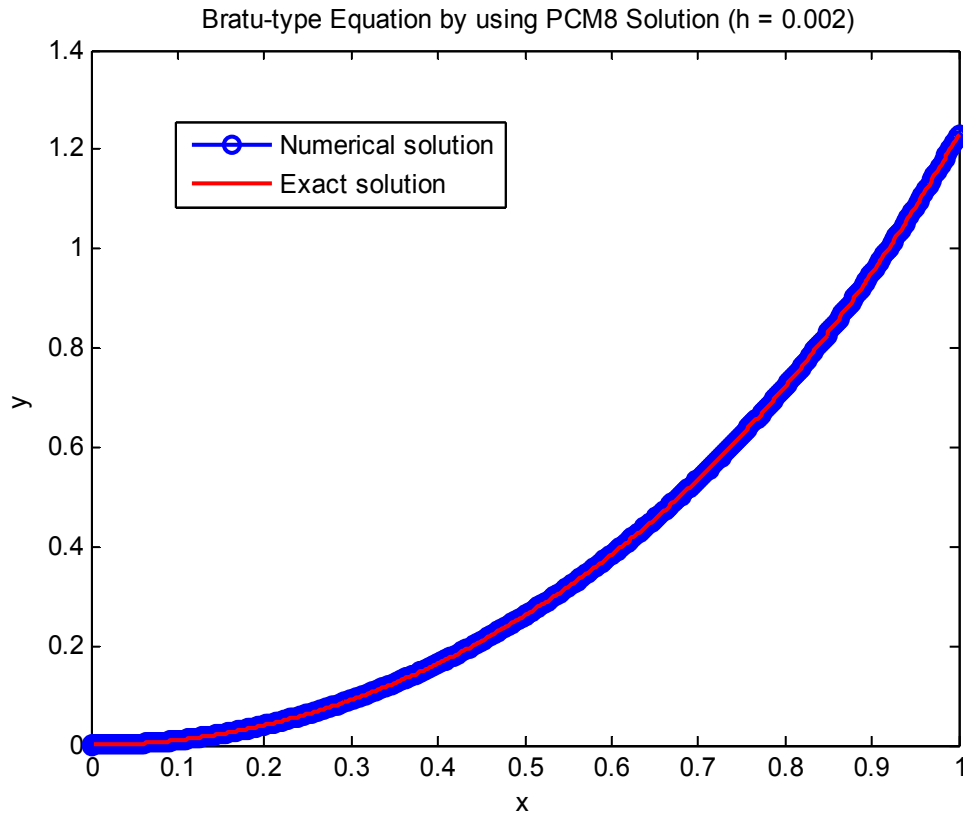


Figure 4.7: The graph of exact and numerical solution of Example 4.2 $N=500, h=0.002$

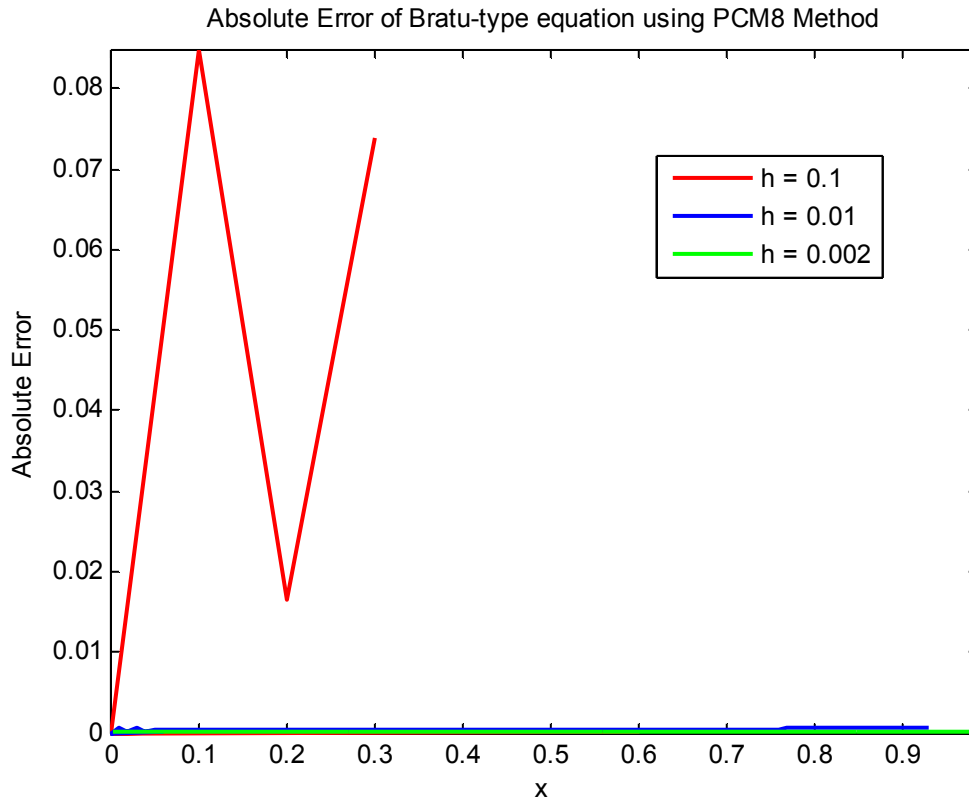


Figure 4.8:-The error graph for $h=0.1$, $h=0.01$ and $h=0.002$ of Example 4.2.

Table 4.7: The maximum absolute errors for different values of N .

Examples	$N=10$	$N=100$	$N=500$	Order of convergence
Example 4.1	6.39360e-06	1.98397e-13	1.11022e-15	8
Example 4.2	1.27872e-05	3.96794e-13	2.22045e-15	8

Table 4.8: The Comparison of absolute errors for Example 4.2 at mesh size $h=0.1$ with the existing literature

x	Mekonnen M. et.al (2020)	Present Method
0	0	0
0.1	2.8436e-09	4.9027e-09
0.2	1.2788e-07	1.0071e-08
0.3	3.9593e-07	1.5973e-08
0.4	3.4141e-06	2.3233e-08
0.5	4.4904e-06	3.2789e-08
0.6	6.8988e-06	4.6204e-08
0.7	1.1741e-05	6.6335e-08
0.8	2.1580e-05	1.3656e-06
0.9	4.2756e-05	4.3252e-06
1	9.2517e-05	1.2787e-05

4.6. Discussion

From Tables 4.1 up to Table 4.6, it can be observed that the eighth order predictor–corrector method provides highly accurate numerical solutions for the Bratu-type equation.

From Table 4.1, for mesh size $h = 0.1$, The absolute error increases gradually as x moves away from the initial point because the numerical solution accumulates local truncation errors at every integration step. Moreover, the nonlinear term e^{2x} becomes larger as the solution increases, making the problem more sensitive to small perturbations. Consequently, although the eighth order predictor- corrector method maintains very high accuracy throughout the interval, the global error naturally grows with x , leading to slightly larger absolute errors near $x=1$.

From Table 4.4, The absolute error increases gradually as x increases because the local truncation errors introduced at each integration step accumulate, producing a larger global error. Furthermore, the nonlinear exponential term e^y increases with the solution, making the differential equation more sensitive to perturbations. Since a fixed step size $h=0.1$ is employed, the numerical approximation becomes less accurate in the region where the exact solution changes more rapidly near $x=1$. Nevertheless, the maximum absolute error is only 1.2787×10^{-5} , demonstrating that the eighth order predictor- corrector method maintains high accuracy over the entire interval.

From Tables 4.2, 4.3, 4.5, 4.6 and 4.7, When the mesh size is reduced to $h = 0.01$, the error decreases significantly. Furthermore, when $h = 0.002$, the numerical solution almost coincides with the exact solution.

The graphical comparisons also confirm that the numerical solution converges to the exact solution as the mesh size decreases. This demonstrates the stability and efficiency of the eighth order predictor–corrector method for solving initial value problems of Bratu-type equations.

Therefore, the proposed method can be considered as effective numerical technique for solving Bratu-type equations.

Furthermore, comparison with existing methods revealed that the proposed method generally yielded smaller errors and better numerical performance. The convergence analysis confirmed that the method possesses the expected order of accuracy, while the stability analysis indicated that the method is stable for the problems considered.

Therefore, the numerical results demonstrate that the proposed eighth-order predictor–corrector method is an accurate, stable, and efficient technique for solving Bratu-type initial value problems and can serve as a reliable alternative to existing numerical methods.

From figure 4.4 and 4.8, For large step size ($h=0.1$) produced oscillatory error behavior, while smaller step sizes ($h=0.01$ and $h=0.002$) yielded stable error distributions. This indicates that the method becomes more stable as the step size decreases.

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 Summary

The overall of the Thesis is summarized as the following manner, for discussing the importance and the numerical properties of eighth order predictor corrector methods is the main topics of this thesis and we tried to explain the Bratu-type equations and due to finding and studying same initial value problems of Bratu-type equations which helps to know definitions numerical properties of eighth order predictor corrector methods.

In this thesis, the numerical solutions of Bratu-type equations have been presented depending on same basic mathematical backgrounds(notation) such as, statement of the problem, objective of the study, definition for studies,derivation of the method and analysis. We also to determine the numerical properties of eighth order predictor-corrector methods for solving Bratu-type equations.

Furthermore, comparison with existing methods revealed that the proposed method generally yielded smaller errors and better numerical performance. The convergence analysis confirmed that the method possesses the expected order of accuracy, while the stability analysis indicated that the method is stable for the problems considered.

Therefore, the numerical results demonstrate that the proposed eighth-order predictor–corrector method is an accurate, stable, and efficient technique for solving Bratu-type initial value problems and can serve as a reliable alternative to existing numerical methods.

Finally, the proposed method is stable, more accurate and effective methods for solving Bratu-type equations.

5.2. Conclusion

In this study, the Eighth Order Predictor–Corrector Method was applied to solve the initial value problem of the Bratu-type differential equation. The main objective of the study was to investigate the accuracy and efficiency of this numerical method for solving nonlinear differential equations.

The numerical results were obtained using different mesh sizes $h = 0.1$, $h = 0.01$, and $h = 0.002$ over the interval $0 < x < 1$. The approximate numerical solutions were compared with the exact solution in order to evaluate the performance of the method.

The results obtained from the numerical computations show that the Eighth Order Predictor–Corrector Method provides highly accurate approximations to the exact solution of the Bratu-type equation. The absolute errors between the numerical and exact solutions were found to decrease significantly as the mesh size becomes smaller.

The absolute error varies across the interval because it is the point wise difference between the exact and numerical solutions. As the numerical method advances, truncation and round-off errors accumulate and propagate, causing the absolute error to change, and in this case, to generally increase with x .

From the results presented in Chapter Four, it was observed that when the mesh size decreases from $h = 0.01$, to $h = 0.002$ the numerical solutions become closer to the exact solution. The graphical comparisons also confirmed that the numerical solution converges to the exact solution as the mesh size decreases. Furthermore, comparison with existing methods revealed that the proposed method generally yielded smaller errors and better numerical performance. The convergence analysis confirmed that the method possesses the expected order of accuracy, while the stability analysis indicated that the method is stable for the problems considered.

Therefore, the numerical results demonstrate that the proposed eighth-order predictor–corrector method is an accurate, stable, and efficient technique for solving Bratu-type initial value problems and can serve as a reliable alternative to existing numerical methods.

Large step size ($h=0.1$) produced oscillatory error behavior, while smaller step sizes ($h=0.01$ and $h=0.002$) yielded stable error distributions. This indicates that the method becomes more stable as the step size decreases.

Therefore, it can be concluded that the Eighth Order Predictor–Corrector Method is an efficient, stable, and accurate numerical technique for solving initial value problems of Bratu-type equations. The method is suitable for solving nonlinear differential equations where exact analytical solutions may be difficult to obtain.

5.3. Recommendation and Future Work

In this thesis eighth order predictor corrector method is presented for solving initial values of problem Bratu-type equations. The stability and convergence of the method on two model examples with exact solutions by taking different values for N and the computational results are presented in the tables. Furthermore, from the tables it is significant that all of the absolute errors decreases which in turn shows the convergence of the computed solution.

Briefly, the present method is stable, more accurate and effective method for solving initial value of Bratu-type equations. In a future work, we will apply the proposed method to the third and fourth order of ordinary differential equations.

REFERENCES

- Abukhaled,M,khuri,S.A,sayfly,A,&Siddiqi,S.S.(2012).Numerical solutions of the Bratu problem by collocation methods. *Applied Mathematics and computation* 218(21),10683-10692.
- Batiha,B.,Noorani,M.S.M.,Hashim,I,&Batiha,K.(2007).Application of the variational iteration method to Bratu-type equations. *Chaos,Solitons& Fractals*,32(5),1523-1528.
- Butcher, J.C.(2008).Numerical methods for ordinary differential equations(2nd ed.).
- He,J.h.(2006).Homotopy perturbation method for solving boundary value problems. *Physics Letters A*, 350 (1 - 2) 87-88.
- Shampine,L.F.(1994).Numerical solution of ordinary differential equations.
- Chapman Hall..Wazwaz,A.M. (2005). The variational iteration method for solving Bratu-type equations. *Applied Mathematics and Computation*,167(2),1196-1210.
- Abdelmajid El hajaji,KhlidHilal,ElmerzguioiMhamed and ElghordafJahla,Acubic Spline Collocation Method for solving Biratu problem,IISTEJ..a Mathematic al theory and modelling3(14),2013.
- AksoyY,PardemirliM; NewPerturbation-iteration solutions for Biratu-type equations .*Computers and Mathematics With applications* 59(8),2802-2808,2010.
- BathaB,Numerical solution of Biratu – Type equations by the variational iteration method,HacettepeJ.Math.Stat,39(1),22-29,2010.
- Butcher,J.C. , Numerical methods for Ordinary differential equations , John Wiley and sons Ltd, New Zealand.
- Debela, H. G., Yadeta, H. B., &Kejela, S. B. (2017). Numerical solutions of second order initial value problems of Bratu-type equation using higher ordered Runge-Kutta method. *International Journal of Scientific and Research Publications*, 7(10), 187–197.

- DarwishMA,KashkariBS. ,Numerical solutions of second order initial value problems of Biratu-type via optimal homotopy asymptotic method, American Journal of Computational Mathematics, 4(2);47,2014.
- Duan JS,R BaleanuD,WazwazaM,Areview of the AdomianDecompositionmethod and its applications to fractional differential equations, Communications in Fractional calculus.3(2),73-99,2012.
- Habtamu,G.D, Habtamu B,Y.,SolomonBK.,Numerical solutions of second orderinital value problems of Biratu –type equation using higher ordered Rungu – Kutta method, International Journal of scientific and research Publications,7(10),187-197,2017.
- King A.C. BillinghamJ, and Otto SR, Differential equations: Linear, Non linearordinary,partial.Camridge University Press,New York,(2003).
- MoradiE.BabolianE,Javadis.,The explicit formulas for reproducing kernel of some Hilbert space.Miskok Mathematical Notes . 16(2),1041-1053,2015.
- MengistuMekonnen, GenanewGofe, &HabbtamuGaroma(2020): Numerical solutions of second order initial value Problems of Bratu-Type Equations Using Predictor-Corrector Method, International Journal of Engineering & Applied Sciences, Vol 12(1), 15-31

APPENDIX

MATLAB CODE FOR Example 4.1

```
Format short e
clc; clear;

% Functions
f = @(x,y,z) z;
g = @(x,y,z) exp(2*y);
ex = @(x) log(sec(x));

% Step sizes
h_values = [0.1, 0.01, 0.002];

for hh = 1:length(h_values)

    h = h_values(hh);

    % Initial values
    x0 = 0; xf = 1;
    y0 = 0; z0 = 0;
    n = round((xf - x0)/h);

    % Preallocation
    x = zeros(n+1,1);
    y = zeros(n+1,1);
    z = zeros(n+1,1);
    yex = zeros(n+1,1);
    e = zeros(n+1,1);

    % Initial conditions
    x(1)=x0; y(1)=y0; z(1)=z0;
    yex(1)=ex(x0);

    %% ---- RK6 STARTING VALUES ----
    for i = 1:7
        x(i+1) = x(i) + h;

        k1 = f(x(i),y(i),z(i));
        l1 = g(x(i),y(i),z(i));

        k2 = f(x(i)+h/2, y(i)+h*k1/2, z(i)+h*l1/2);
        l2 = g(x(i)+h/2, y(i)+h*k1/2, z(i)+h*l1/2);

        k3 = f(x(i)+h/4, y(i)+h*(3*k1+k2)/16, z(i)+h*(3*l1+l2)/16);
```

```

l3 = g(x(i)+h/4, y(i)+h*(3*k1+k2)/16, z(i)+h*(3*l1+l2)/16);

k4 = f(x(i)+h/2, y(i)+h*k3/2, z(i)+h*l3/2);
l4 = g(x(i)+h/2, y(i)+h*k3/2, z(i)+h*l3/2);

k5 = f(x(i)+3*h/4, y(i)+h*(-3*k2+6*k3+9*k4)/16,...
z(i)+h*(-3*l2+6*l3+9*l4)/16);
l5 = g(x(i)+3*h/4, y(i)+h*(-3*k2+6*k3+9*k4)/16,...
z(i)+h*(-3*l2+6*l3+9*l4)/16);

k6 = f(x(i)+h, y(i)+h*(k1+4*k2+6*k3-12*k4+8*k5)/7,...
z(i)+h*(l1+4*l2+6*l3-12*l4+8*l5)/7);
l6 = g(x(i)+h, y(i)+h*(k1+4*k2+6*k3-12*k4+8*k5)/7,...
z(i)+h*(l1+4*l2+6*l3-12*l4+8*l5)/7);

y(i+1) = y(i) + h*(7*k1+32*k3+12*k4+32*k5+7*k6)/90;
z(i+1) = z(i) + h*(7*l1+32*l3+12*l4+32*l5+7*l6)/90;

yex(i+1) = ex(x(i+1));
e(i+1) = abs(y(i+1)-yex(i+1));
end

%%% ---- PCM8 METHOD ----
for i = 8:n
x(i+1) = x(i) + h;

% Predictor
yp = y(i) + h/120960 * ( ...
434241*f(x(i),y(i),z(i)) ...
-1152169*f(x(i-1),y(i-1),z(i-1)) ...
+2183877*f(x(i-2),y(i-2),z(i-2)) ...
-2664477*f(x(i-3),y(i-3),z(i-3)) ...
+2102243*f(x(i-4),y(i-4),z(i-4)) ...
-1041723*f(x(i-5),y(i-5),z(i-5)) ...
+295767*f(x(i-6),y(i-6),z(i-6)) ...
-36799*f(x(i-7),y(i-7),z(i-7)) );

zp = z(i) + h/120960 * ( ...
434241*g(x(i),y(i),z(i)) ...
-1152169*g(x(i-1),y(i-1),z(i-1)) ...
+2183877*g(x(i-2),y(i-2),z(i-2)) ...
-2664477*g(x(i-3),y(i-3),z(i-3)) ...
+2102243*g(x(i-4),y(i-4),z(i-4)) ...
-1041723*g(x(i-5),y(i-5),z(i-5)) ...
+295767*g(x(i-6),y(i-6),z(i-6)) ...
-36799*g(x(i-7),y(i-7),z(i-7)) );

```

```

% Corrector
y(i+1) = y(i) + h/120960 * ( ...
    36799*f(x(i+1),yp,zp) ...
    +139849*f(x(i),y(i),z(i)) ...
    -121797*f(x(i-1),y(i-1),z(i-1)) ...
    +123133*f(x(i-2),y(i-2),z(i-2)) ...
    -88547*f(x(i-3),y(i-3),z(i-3)) ...
    +41499*f(x(i-4),y(i-4),z(i-4)) ...
    -11351*f(x(i-5),y(i-5),z(i-5)) ...
    +1375*f(x(i-6),y(i-6),z(i-6)) );
z(i+1) = z(i) + h/120960 * ( ...
    36799*g(x(i+1),yp,zp) ...
    +139849*g(x(i),y(i),z(i)) ...
    -121797*g(x(i-1),y(i-1),z(i-1)) ...
    +123133*g(x(i-2),y(i-2),z(i-2)) ...
    -88547*g(x(i-3),y(i-3),z(i-3)) ...
    +41499*g(x(i-4),y(i-4),z(i-4)) ...
    -11351*g(x(i-5),y(i-5),z(i-5)) ...
    +1375*g(x(i-6),y(i-6),z(i-6)) );

yex(i+1) = ex(x(i+1));
e(i+1) = abs(y(i+1)-yex(i+1));
end

%% ---- RESULTS ----
fprintf('\n===== Results for h = %.3f =====\n', h);
fprintf(' x      ExactsolutionNumericalsolution   Error\n');
disp([x yex y e]);
max_error = max(e);
fprintf('Maximum Absolute Error = %.5e\n', max_error);

%% ---- PLOT ----
figure;
plot(x,y,'-ob',x,yex,'-r','LineWidth',1.5)
legend('Numerical solution','Exact solution ','Location','best')
title(['Bratu-type Equation by using PCM8 Solution (h = ', num2str(h), ')'])
xlabel('x'); ylabel('y');

xlim([0 1]);
end

```

MAT LAB CODE for Absolute Error of different mesh sizes of example 4.1

```
format short e
clc; clear;
% Functions
f = @(x,y,z) z;
g = @(x,y,z) exp(2*y);
ex = @(x) log(sec(x));

% Step sizes
h_values = [0.1, 0.01, 0.002];

% Colors
colors = {'r', 'b', 'g'};

figure; hold on;

for hh = 1:length(h_values)

    h = h_values(hh);

    x0 = 0; xf = 1;
    y0 = 0; z0 = 0;
    n = round((xf - x0)/h);

    x = zeros(n+1,1);
    y = zeros(n+1,1);
    z = zeros(n+1,1);
    yex = zeros(n+1,1);
    e = zeros(n+1,1);

    x(1)=x0; y(1)=y0; z(1)=z0;
    yex(1)=ex(x0);

    %% ---- RK6 START ----
    for i = 1:6

        x(i+1) = x(i) + h;

        k1 = f(x(i),y(i),z(i));
        l1 = g(x(i),y(i),z(i));

        k2 = f(x(i)+h/2, y(i)+h*k1/2, z(i)+h*l1/2);
```

```

l2 = g(x(i)+h/2, y(i)+h*k1/2, z(i)+h*l1/2);

k3 = f(x(i)+h/4, y(i)+h*(3*k1+k2)/16, z(i)+h*(3*l1+l2)/16);
l3 = g(x(i)+h/4, y(i)+h*(3*k1+k2)/16, z(i)+h*(3*l1+l2)/16);

k4 = f(x(i)+h/2, y(i)+h*k3/2, z(i)+h*l3/2);
l4 = g(x(i)+h/2, y(i)+h*k3/2, z(i)+h*l3/2);

k5 = f(x(i)+3*h/4, y(i)+h*(-3*k2+6*k3+9*k4)/16,...
z(i)+h*(-3*l2+6*l3+9*l4)/16);

l5 = g(x(i)+3*h/4, y(i)+h*(-3*k2+6*k3+9*k4)/16,...
z(i)+h*(-3*l2+6*l3+9*l4)/16);

k6 = f(x(i)+h, y(i)+h*(k1+4*k2+6*k3-12*k4+8*k5)/7,...
z(i)+h*(l1+4*l2+6*l3-12*l4+8*l5)/7);

l6 = g(x(i)+h, y(i)+h*(k1+4*k2+6*k3-12*k4+8*k5)/7,...
z(i)+h*(l1+4*l2+6*l3-12*l4+8*l5)/7);

y(i+1) = y(i) + h*(7*k1+32*k3+12*k4+32*k5+7*k6)/90;
z(i+1) = z(i) + h*(7*l1+32*l3+12*l4+32*l5+7*l6)/90;

yex(i+1) = ex(x(i+1));
e(i+1) = abs(y(i+1) - yex(i+1));
end
%% ---- PCM8 METHOD ----
for i = 8:n

x(i+1) = x(i) + h;

yp = y(i) + h/120960 * ( ...
    434241*f(x(i),y(i),z(i)) ...
    -1152169*f(x(i-1),y(i-1),z(i-1)) ...
    +2183877*f(x(i-2),y(i-2),z(i-2)) ...
    -2664477*f(x(i-3),y(i-3),z(i-3)) ...
    +2102243*f(x(i-4),y(i-4),z(i-4)) ...
    -1041723*f(x(i-5),y(i-5),z(i-5)) ...
    +295767*f(x(i-6),y(i-6),z(i-6)) ...
    -36799*f(x(i-7),y(i-7),z(i-7)) );

zp = z(i) + h/120960 * ( ...
    434241*g(x(i),y(i),z(i)) ...
    -1152169*g(x(i-1),y(i-1),z(i-1)) ...
    +2183877*g(x(i-2),y(i-2),z(i-2)) ...
    -2664477*g(x(i-3),y(i-3),z(i-3)) ...

```

```

+2102243*g(x(i-4),y(i-4),z(i-4)) ...
-1041723*g(x(i-5),y(i-5),z(i-5)) ...
+295767*g(x(i-6),y(i-6),z(i-6)) ...
-36799*g(x(i-7),y(i-7),z(i-7)) );

y(i+1) = y(i) + h/120960 * ( ...
    36799*f(x(i+1),yp,zp) ...
+139849*f(x(i),y(i),z(i)) ...
-121797*f(x(i-1),y(i-1),z(i-1)) ...
+123133*f(x(i-2),y(i-2),z(i-2)) ...
-88547*f(x(i-3),y(i-3),z(i-3)) ...
+41499*f(x(i-4),y(i-4),z(i-4)) ...
-11351*f(x(i-5),y(i-5),z(i-5)) ...
+1375*f(x(i-6),y(i-6),z(i-6)) );

z(i+1) = z(i) + h/120960 * ( ...
    36799*g(x(i+1),yp,zp) ...
+139849*g(x(i),y(i),z(i)) ...
-121797*g(x(i-1),y(i-1),z(i-1)) ...
+123133*g(x(i-2),y(i-2),z(i-2)) ...
-88547*g(x(i-3),y(i-3),z(i-3)) ...
+41499*g(x(i-4),y(i-4),z(i-4)) ...
-11351*g(x(i-5),y(i-5),z(i-5)) ...
+1375*g(x(i-6),y(i-6),z(i-6)) );

yex(i+1) = ex(x(i+1));
e(i+1) = abs(y(i+1) - yex(i+1));
end
plot(x, e, 'Color', colors{hh}, 'LineWidth', 1.5);
end
% ---- PLOT FORMATTING (FRAME FIX) ----
grid on
box on
axis tight
legend('h = 0.1','h = 0.01','h = 0.002','Location','best');
title('Absolute Error Comparison (PCM8 Method)');
xlabel('x');
ylabel('Absolute Error');
grid off
hold off

```

MATLAB CODE for example 4.2

```
format short e
clc; clear;
% Functions
f = @(x,y,z) z;
g = @(x,y,z) 2*exp(y);
ex = @(x) -2*log(cos(x));
% Step sizes
h_values = [0.1, 0.01, 0.002];

for hh = 1:length(h_values)
    h = h_values(hh);
    % Initial values
    x0 = 0; xf = 1;
    y0 = 0; z0 = 0;
    n = round((xf - x0)/h);

    % Preallocation
    x = zeros(n+1,1);
    y = zeros(n+1,1);
    z = zeros(n+1,1);
    yex = zeros(n+1,1);
    e = zeros(n+1,1);
    % Initial conditions
    x(1)=x0; y(1)=y0; z(1)=z0;
    yex(1)=ex(x0);

    %% ---- RK6 STARTING VALUES ----
    for i = 1:7
        x(i+1) = x(i) + h;

        k1 = f(x(i),y(i),z(i));
        l1 = g(x(i),y(i),z(i));
        k2 = f(x(i)+h/2, y(i)+h*k1/2, z(i)+h*l1/2);
        l2 = g(x(i)+h/2, y(i)+h*k1/2, z(i)+h*l1/2);

        k3 = f(x(i)+h/4, y(i)+h*(3*k1+k2)/16, z(i)+h*(3*l1+l2)/16);
        l3 = g(x(i)+h/4, y(i)+h*(3*k1+k2)/16, z(i)+h*(3*l1+l2)/16);
        k4 = f(x(i)+h/2, y(i)+h*k3/2, z(i)+h*l3/2);
        l4 = g(x(i)+h/2, y(i)+h*k3/2, z(i)+h*l3/2);
        k5 = f(x(i)+3*h/4, y(i)+h*(-3*k2+6*k3+9*k4)/16,...
        z(i)+h*(-3*l2+6*l3+9*l4)/16);
        l5 = g(x(i)+3*h/4, y(i)+h*(-3*k2+6*k3+9*k4)/16,...
        z(i)+h*(-3*l2+6*l3+9*l4)/16);
        k6 = f(x(i)+h, y(i)+h*(k1+4*k2+6*k3-12*k4+8*k5)/7,...
        z(i)+h*(l1+4*l2+6*l3-12*l4+8*l5)/7);
```

```

l6 = g(x(i)+h, y(i)+h*(k1+4*k2+6*k3-12*k4+8*k5)/7,...
z(i)+h*(l1+4*l2+6*l3-12*l4+8*l5)/7);
y(i+1) = y(i) + h*(7*k1+32*k3+12*k4+32*k5+7*k6)/90;
z(i+1) = z(i) + h*(7*l1+32*l3+12*l4+32*l5+7*l6)/90;

```

```

yex(i+1) = ex(x(i+1));
e(i+1) = abs(y(i+1)-yex(i+1));
end

```

```

%%% ---- PCM8 METHOD ----
for i = 8:n
x(i+1) = x(i) + h;

```

```

% Predictor
yp = y(i) + h/120960 * ( ...
    434241*f(x(i),y(i),z(i)) ...
    -1152169*f(x(i-1),y(i-1),z(i-1)) ...
    +2183877*f(x(i-2),y(i-2),z(i-2)) ...
    -2664477*f(x(i-3),y(i-3),z(i-3)) ...
    +2102243*f(x(i-4),y(i-4),z(i-4)) ...
    -1041723*f(x(i-5),y(i-5),z(i-5)) ...
    +295767*f(x(i-6),y(i-6),z(i-6)) ...
    -36799*f(x(i-7),y(i-7),z(i-7)) );

```

```

zp = z(i) + h/120960 * ( ...
    434241*g(x(i),y(i),z(i)) ...
    -1152169*g(x(i-1),y(i-1),z(i-1)) ...
    +2183877*g(x(i-2),y(i-2),z(i-2)) ...
    -2664477*g(x(i-3),y(i-3),z(i-3)) ...
    +2102243*g(x(i-4),y(i-4),z(i-4)) ...
    -1041723*g(x(i-5),y(i-5),z(i-5)) ...
    +295767*g(x(i-6),y(i-6),z(i-6)) ...
    -36799*g(x(i-7),y(i-7),z(i-7)) );

```

```

% Corrector
y(i+1) = y(i) + h/120960 * ( ...
    36799*f(x(i+1),yp,zp) ...
    +139849*f(x(i),y(i),z(i)) ...
    -121797*f(x(i-1),y(i-1),z(i-1)) ...
    +123133*f(x(i-2),y(i-2),z(i-2)) ...
    -88547*f(x(i-3),y(i-3),z(i-3)) ...
    +41499*f(x(i-4),y(i-4),z(i-4)) ...
    -11351*f(x(i-5),y(i-5),z(i-5)) ...
    +1375*f(x(i-6),y(i-6),z(i-6)) );

```

```

z(i+1) = z(i) + h/120960 * ( ...

```

```

        36799*g(x(i+1),yp,zp) ...
        +139849*g(x(i),y(i),z(i)) ...
        -121797*g(x(i-1),y(i-1),z(i-1)) ...
        +123133*g(x(i-2),y(i-2),z(i-2)) ...
        -88547*g(x(i-3),y(i-3),z(i-3)) ...
        +41499*g(x(i-4),y(i-4),z(i-4)) ...
        -11351*g(x(i-5),y(i-5),z(i-5)) ...
        +1375*g(x(i-6),y(i-6),z(i-6)) );

yex(i+1) = ex(x(i+1));
e(i+1) = abs(y(i+1)-yex(i+1));
end

%% ---- RESULTS ----
fprintf('\n===== Results for h = %.3f =====\n', h);
fprintf(' x      Exact y    Numerical y    Error\n');
disp([x yex y e]);

max_error = max(e);
fprintf('Maximum Absolute Error = %.5e\n', max_error);

%% ---- PLOT ----
figure;
plot(x,y,'-ob',x,yex,'-r','LineWidth',1.5)
legend('Numerical solution','Exactsolution','Location','best')
title([' Bratu-type Equation by using PCM8 Solution (h = ', num2str(h), ')'])
xlabel('x'); ylabel('y');

xlim([0 1]);
end

```

Mat LAB CODE for Absolute Error of different mesh sizes of example 4.2

```
format short e
clc; clear;

% Functions
f = @(x,y,z) z;
g = @(x,y,z) 2*exp(y);
ex = @(x) -2*log(cos(x));

% Step sizes
h_values = [0.1, 0.01, 0.002];

% Colors
colors = {'r', 'b', 'g'};

figure; hold on;

for hh = 1:length(h_values)

    h = h_values(hh);

    x0 = 0; xf = 1;
    y0 = 0; z0 = 0;
    n = round((xf - x0)/h);

    x = zeros(n+1,1);
    y = zeros(n+1,1);
    z = zeros(n+1,1);
    yex = zeros(n+1,1);
    e = zeros(n+1,1);

    x(1)=x0; y(1)=y0; z(1)=z0;
    yex(1)=ex(x0);

    %% ---- RK6 START ----
    for i = 1:6

        x(i+1) = x(i) + h;

        k1 = f(x(i),y(i),z(i));
        l1 = g(x(i),y(i),z(i));

        k2 = f(x(i)+h/2, y(i)+h*k1/2, z(i)+h*l1/2);
        l2 = g(x(i)+h/2, y(i)+h*k1/2, z(i)+h*l1/2);
```

```

k3 = f(x(i)+h/4, y(i)+h*(3*k1+k2)/16, z(i)+h*(3*l1+l2)/16);
l3 = g(x(i)+h/4, y(i)+h*(3*k1+k2)/16, z(i)+h*(3*l1+l2)/16);

k4 = f(x(i)+h/2, y(i)+h*k3/2, z(i)+h*l3/2);
l4 = g(x(i)+h/2, y(i)+h*k3/2, z(i)+h*l3/2);

k5 = f(x(i)+3*h/4, y(i)+h*(-3*k2+6*k3+9*k4)/16,...
z(i)+h*(-3*l2+6*l3+9*l4)/16);

l5 = g(x(i)+3*h/4, y(i)+h*(-3*k2+6*k3+9*k4)/16,...
z(i)+h*(-3*l2+6*l3+9*l4)/16);

k6 = f(x(i)+h, y(i)+h*(k1+4*k2+6*k3-12*k4+8*k5)/7,...
z(i)+h*(l1+4*l2+6*l3-12*l4+8*l5)/7);

l6 = g(x(i)+h, y(i)+h*(k1+4*k2+6*k3-12*k4+8*k5)/7,...
z(i)+h*(l1+4*l2+6*l3-12*l4+8*l5)/7);

y(i+1) = y(i) + h*(7*k1+32*k3+12*k4+32*k5+7*k6)/90;
z(i+1) = z(i) + h*(7*l1+32*l3+12*l4+32*l5+7*l6)/90;

yex(i+1) = ex(x(i+1));
e(i+1) = abs(y(i+1) - yex(i+1));
end

%% ---- PCM8 METHOD ----
for i = 8:n

x(i+1) = x(i) + h;

yp = y(i) + h/120960 * ( ...
    434241*f(x(i),y(i),z(i)) ...
    -1152169*f(x(i-1),y(i-1),z(i-1)) ...
    +2183877*f(x(i-2),y(i-2),z(i-2)) ...
    -2664477*f(x(i-3),y(i-3),z(i-3)) ...
    +2102243*f(x(i-4),y(i-4),z(i-4)) ...
    -1041723*f(x(i-5),y(i-5),z(i-5)) ...
    +295767*f(x(i-6),y(i-6),z(i-6)) ...
    -36799*f(x(i-7),y(i-7),z(i-7)) );

zp = z(i) + h/120960 * ( ...
    434241*g(x(i),y(i),z(i)) ...
    -1152169*g(x(i-1),y(i-1),z(i-1)) ...
    +2183877*g(x(i-2),y(i-2),z(i-2)) ...
    -2664477*g(x(i-3),y(i-3),z(i-3)) ...
    +2102243*g(x(i-4),y(i-4),z(i-4)) ...

```

```

-1041723*g(x(i-5),y(i-5),z(i-5)) ...
+295767*g(x(i-6),y(i-6),z(i-6)) ...
-36799*g(x(i-7),y(i-7),z(i-7)) );

y(i+1) = y(i) + h/120960 * ( ...
    36799*f(x(i+1),yp,zp) ...
+139849*f(x(i),y(i),z(i)) ...
-121797*f(x(i-1),y(i-1),z(i-1)) ...
+123133*f(x(i-2),y(i-2),z(i-2)) ...
-88547*f(x(i-3),y(i-3),z(i-3)) ...
+41499*f(x(i-4),y(i-4),z(i-4)) ...
-11351*f(x(i-5),y(i-5),z(i-5)) ...
+1375*f(x(i-6),y(i-6),z(i-6)) );

z(i+1) = z(i) + h/120960 * ( ...
    36799*g(x(i+1),yp,zp) ...
+139849*g(x(i),y(i),z(i)) ...
-121797*g(x(i-1),y(i-1),z(i-1)) ...
+123133*g(x(i-2),y(i-2),z(i-2)) ...
-88547*g(x(i-3),y(i-3),z(i-3)) ...
+41499*g(x(i-4),y(i-4),z(i-4)) ...
-11351*g(x(i-5),y(i-5),z(i-5)) ...
+1375*g(x(i-6),y(i-6),z(i-6)) );

yex(i+1) = ex(x(i+1));
e(i+1) = abs(y(i+1) - yex(i+1));
end

plot(x, e, 'Color', colors{hh}, 'LineWidth', 1.5);

end

% ---- PLOT FORMATTING (FRAME FIX) ----
grid on
box on
axis tight

legend('h = 0.1','h = 0.01','h = 0.002','Location','best');
title('Absolute Error of Bratu-type equation using PCM8 Method');
xlabel('x');
ylabel('Absolute Error');
grid off
hold off;

```