Unified Code for Collocation Multistep Methods in Solving Stiff Systems of Boundary Value Problems

OKWHAROBO, Solomon Monday

January 4, 2024

Dedication

I dedicate this project to my Mother,
who have always been a source of inspiration
and unwavering support throughout my academic journey.

Your dedication and encouragement have been my driving force.

Acknowledgments

I would like to express my deepest gratitude to God, the source of all wisdom and strength, for guiding me throughout this academic journey.

My heartfelt thanks go to my mother, whose unwavering support, encouragement, and sacrifices have been the driving force behind my success.

I am immensely grateful to my supervisor, Prof J.O Fatokun, for their invaluable guidance, mentorship, and expertise. Their insights and encouragement significantly contributed to the completion of this project.

Special thanks to Dr. Akewe, Dr. Okoro, Dr. Evans, and Dr. Akinnukawe for their support, constructive feedback, and contributions to my academic and personal development.

Your support has been instrumental, and I am deeply appreciative.

Abstract

This project introduces a unified python numerical code that integrates collocation and multistep methods for solving stiff systems of boundary value problems (BVPs). Stiff BVPs, characterized by rapidly changing components, pose challenges in numerical analysis. The code aims to balance accuracy and computational efficiency by seamlessly combining the strengths of collocation methods in capturing local behavior and multistep methods in handling temporal evolution. The research includes parametric studies to assess the code's versatility and explores its impact on diverse stiff BVPs. The documentation provides clear guidance for users, ensuring accessibility and usability of the developed code.

Contents

List of Figures

List of Tables

List of Abbreviations

ODE :	Ordinary Differential Equation
BVP :	Boundary Value Problem
IVP :	Initial Value Problem

Chapter 1

Introduction

1.1 Background of study

1.1.1 Introduction to Ordinary Differential Equations (ODEs)

Mathematical models in a vast range of discilines, from science and technology to sociology and business, describe how quantities *change*. This leads naturally to a language of ordinary differential equations (ODEs). Ordinary Differential Equations (ODEs) are a type of differential equation that involves an unknown function and its derivatives.
ODEs are of paramount significance in mathematical modeling because they provide a concise and powerful way to describe how a quantity changes concerning time or another independent variable. The ability to capture the rate of change of a variable makes ODEs essential in understanding dynamic processes, predicting future states, and optimizing system behavior. They find applications in various scientific disciplines, engineering, economics, and other fields where changes over time are crucial to the analysis. Differential equations along with a specified value of the unknown function at a given point in the domain of the solution are an Initial Value Problem. This specified value is the initial condition. In many important cases of differential equations, analytic solutions are difficult or impossible to obtain and time consuming.

Although numerical solutions are approximations, the error of approximation is often acceptable and numerical solutions give birth to algorithms that are used to design computer simulated solutions.A major development in the study of numerical methods is the introduction of modern computers for the calculation of functions in the mid $20^{th} century$ [1]. Until this time, numerical methods often depended on hand interpolation in large printed tables. These tables contained data such as interpolation points and function coefficients up to 16 decimal places or more and were used to obtain very good numerical estimates of functions. Although these same algorithms continue to be part of the base of software algorithms for obtaining numerical solutions, the way com-

puter represents and processes numbers gives rise to inexact results from the programs. Inexactness arises for different reasons like the number of decimal places in which results are retrieved and the number of steps required to the final solution, nevertheless, the use of computers and computer applications in numerical methods has become an established part of general numerical analysis with the development of many numerical computing applications such as MATLAB, S-PLUS, LabView, FreeMat, Scilab, GNU Octave and their associated speed and performance in obtaining solutions [6].

They are two main types of problems associated with ODEs: Initial Value Problem (IVPs) and Boundary Value Problem (BVPs).

IVPs is concerned with ODEs that specify the state of the system at a single point, usually at the start of the interval over which the solition is sought. IVPs require the values of the unknown function and its derivatives at a specific initial point. They are of the from

$$y'(x) = f(x, y(x)) \quad x \in [x_o, b] \tag{1.1}$$

$$y(x_o) = y_o \tag{1.2}$$

where prime indicates derivative with respect to x and f satisfies the Lipschitz condition of the existence and uniqueness of solution

BVP, which is our main topic of concern are classes of mathematical problems that involve finding a solution to a differential equation subject to specified conditions at different points within the domain. Unlike Initial Value Problems (IVPs), which require conditions at a single point, BVPs entail conditions at multiple points, typically at the boundaries of the system. BVPs are prevalent in scientific and engineering applications where systems exhibit spatial dependence or are subject to conditions at distinct locations.

Boundary value problems (BVPs) comes in various types, each characterized by the specific conditions imposed on the solution at the boundaries of the domain. This is categorized based on the number and location of boundary conditions:

1. Two-Point Boundary Value Problems (2PBVP)
2. Multipoint Boundary Value Problems
3. Periodic Boundary Value Problems
4. Eigenvalue problems e.g Sturm-liouville problem
5. $\ldots e.t.c$

Most cases which arises in the analysis of different engineering and mathematical physics problems are of second-order differential equation with boundary conditions prescribed at each of two ends of finite interval $[a, b]$, which is often called the Two-point boundary-value problem.

1.2 Statement of the problem

Stiff systems of Boundary Value Problems (BVPs) pose significant challenges in numerical analysis and computational mathematics. These systems, characterized by solutions with rapidly varying components, require specialized numerical methods to ensure accurate and efficient solutions. While collocation methods excel in capturing local behavior, they may face limitations in handling the temporal evolution of stiff systems. On the other hand, multistep methods provide effective time-stepping techniques but may not fully exploit the spatial information captured by collocation.

The absence of a unified and comprehensive numerical framework that seamlessly integrates both collocation and multistep methods hinders the efficient solution of stiff BVPs. Existing numerical approaches often necessitate a trade-off between accuracy and computational efficiency, limiting their applicability to a broad range of stiff systems.

Therefore, the problem at hand is to develop a unified code that amalgamates the strengths of collocation and multistep methods, aiming to provide a versatile, accurate, and computationally efficient tool for solving stiff BVPs. This research seeks to bridge the gap in existing methodologies by creating a unified numerical framework capable of addressing the unique challenges posed by stiff systems, ultimately contributing to advancements in the numerical solution of stiff BVPs across diverse scientific and engineering disciplines.

1.3 Objectives

The overarching objectives of this research are:

1. To develop a unified code that integrates collocation and multistep methods for solving stiff systems of Boundary Value Problems (BVPs).

   Explanation: The primary goal is to create a comprehensive numerical tool capable of efficiently handling stiff BVPs. The unified code (Using python and dart) will bring together the strengths of collocation and multistep methods, providing a robust framework for solving two-point boundary value problems.

2. To assess the accuracy, efficiency, and versatility of the proposed unified code.

   Explanation: This objective involves a thorough evaluation of the developed code. Accuracy will be examined by comparing solutions with known analytical solutions or benchmarks. Efficiency will assess the computational performance of the code, and versatility will gauge its applicability to two-point boundary value problem

3. To provide a user-friendly tool for researchers and practitioners working with stiff BVPs.

Explanation: In addition to technical performance, emphasis will be placed on creating a user-friendly interface for the unified code which will be tackled with flutter framework. This ensures accessibility for researchers and practitioners from diverse backgrounds, facilitating its use in both academic and practical settings.

1.4 Scope and Limitations

1.5 Definition of Terms

1. Step Length (Mesh-Size): A point within the solution domain where the solution is approximated or calculated. The **step length** (h) is the size of the interval between consecutive points in the independent variable (e.g., time or space) at which the solution of a differential equation is calculated. It plays a crucial role in determining the granularity of the numerical approximation and impacts the accuracy and efficiency of the solution. A smaller step length typically leads to a more accurate but computationally expensive solution, while a larger step length may sacrifice accuracy for computational efficiency. The choice of an appropriate step length is a critical consideration in the numerical solution of differential equations.

2. Stiff and Non-Stiff system: In the context of research, J. D. Lambert characterizes stiffness as follows:

   When employing a numerical method possessing a finite region of absolute stability on a system with arbitrary initial conditions, if the method necessitates the utilization of an exceptionally small step length within a specific integration interval, relative to the smoothness of the exact solution in that range, then the system is identified as stiff during that interval [5]. A system is considered stiff if it contains components or features that vary widely in terms of their natural frequencies or time scales. Stiff systems often involve rapid and slow modes of response, and the stiffness of the system can lead to numerical challenges in solving the associated differential equations.

   It can be deduced that stiffness in a dynamic system refers to the difference in time scales or natural frequencies of its components. Stiff systems require special consideration in numerical simulations due to the challenges associated with solving the corresponding stiff ODEs. Non-stiff systems, on the other hand, are generally easier to simulate numerically.

3. Stiff ODE: A stiff ordinary differential equation (ODE) is a type of differential equation that requires very small step sizes in numerical solutions, often leading to numerical instability unless the step size is taken to be extremely small. This is due to certain terms in the equation that can lead to rapid variation in the solution [1].

For some ODE problems, the step size taken by the solver is forced down to an unreasonably small level in comparison to the interval of integration, even in a region where the solution curve is smooth. These step sizes can be so small that traversing a short time interval might require millions of evaluations. This can lead to the solver failing the integration, but even if it succeeds it will take a very long time to do so. [3]. This can lead to the solver failing the integration, but even if it succeeds it will take a very long time to do so.

Examples of stiff ODEs arise naturally in a wide variety of applications, including the study of spring and damping systems, the analysis of control systems, and problems in chemical kinetics.

4. Boundary value problems: In the context of differential equations, a *Boundary Value Problem* (BVP) is a mathematical problem that involves finding a solution to a differential equation subject to specified boundary conditions. Unlike an Initial Value Problem (IVP), where the solution is sought at a single point given initial conditions, a BVP requires determining the solution over a specified interval or domain with conditions specified at both ends.

   Mathematically, a BVP often takes the form:

$$\frac{d^2y}{dx^2} = f(x, y, \frac{dy}{dx})$$

   subject to boundary conditions:

$$y(a) = \alpha, \quad y(b) = \beta$$

   Here, y is the unknown function to be determined, and $f(x, y, \frac{dy}{dx})$ represents a given function. The boundary conditions $y(a) = \alpha$ and $y(b) = \beta$ specify the values of the solution at the endpoints $x = a$ and $x = b$, respectively [2].

5. Python libraries and packages: Several Python libraries play a crucial role in providing tools and functions to perform various numerical computations efficiently. Numerical analysis involves solving mathematical problems using numerical methods and algorithms. There are specific Python libraries designed to aid in numerical analysis projects. Some of these libraries include: Scipy,NumPy, SymPy, Matplotlib, Pandas, and so on. These libraries are used to perform various numerical computations such as interpolation, integration, differentiation, and so on. They also provide tools for solving differential equations, linear algebra, and other mathematical problems. These libraries are used in this project to perform numerical computations and solve differential equations.

6. Numerical Methods: Numerical methods are techniques used to approximate solutions to mathematical problems that are difficult or impossible to solve analytically. These methods rely on a combination of mathematical theory, computational algorithms, and computer programming to obtain approximate solutions. Numerical methods are used to solve a wide range of problems in science, engineering, and mathematics. They are also used in the analysis of complex systems where mathematical models are used to describe the behavior of the system. Numerical methods are used in this project to solve differential equations and perform other numerical computations [4].

7. Convergence: A numerical method is said to be convergent if the numerical solution approaches the exact solution as the step-size(h) tend to zero 0. Convergence expresses the property by using small step, an accurate computation of the numerical solution can be made arbitrary close to the solution.

8. Consistency and Order: Suppose the numerical method is

$$y_{n+1} = \phi(x_n, y_n, y_{n+1}, \ldots, y_{n+k}, h) \tag{1.3}$$

The local(truncation error) of the method is the error committed by one step of the method. It is the difference between the exact solution and the numerical solution at the next step. The local error is given by

$$\tau_{n+1} = y(x_{n+1}) - \phi(x_n, y_n, y_{n+1}, \ldots, y_{n+k}, h) \tag{1.4}$$

The method is said to be consistent if the local error tends to zero as the step-size h tends to zero. The method is said to be of order p if the local error is of order $p + 1$ as h tends to zero. The order of a method is the largest integer p for which the method is of order p [4].

Consistency is a crucial property ensuring that the numerical approximation converges to the actual solution as the discretization is refined.
The term "order" refers to the order of accuracy of a numerical method. The order of accuracy quantifies how rapidly the numerical solution produced by the method approaches the true solution as the step size or grid spacing decreases. It is a measure of the speed of convergence of the numerical method.

Chapter 2

Explanation

2.1 Project Description

2.2 Objectives

2.3 Scope and Limitations

2.4 Methodology

2.5 Organization of the Report

2.6 Conclusion

References

Bibliography

[1] Wikipedia contributors. Numerical analysis, 12 2023.

[2] Wikipedia contributors. Boundary value problem, 2024. Accessed: January 3, 2024.

[3] Mathworks Inc. Solve Stiff ODEs -MATLAB and Simulink, 12 2023.

[4] Uwaifo Nelson Isibor. Third derivative block hybrid Obrechkoff method for Numerical solution of stiff initial value problems. pages 224–252. 1 2020.

[5] J. D. Lambert. The initial value problem for ordinary differential equations. In D. Jacobs, editor, *The State of the Art in Numerical Analysis*, pages 451–501. Academic Press, New York, 1977.

[6] M.O. Ogwo, D. Omale P.B. Ojih. MATHEMATICAL ANALYSIS OF STIFF AND NON-STIFF INITIAL VALUE PROBLEMS OF ORDINARY DIFFERENTIAL EQUATION USING MATLAB, 9 2014.

Bibliography

Appendix A

Appendix A