Anchor University Lagos

Project Proposal: Unified Code for Collocation Multistep Methods in Solving Stiff Systems of Ordinary Differential Equations

OKWHAROBO, Solomon Monday

supervised by

Prof J.O Fatokun

Department of Mathematics and Statistics

March 10, 2024

Contents

List of Abbreviations

ODE :	Ordinary Differential Equation
BVP :	Boundary Value Problem
IVP :	Initial Value Problem
BDF:	Backward Differentiation Formula
LMM:	Linear Multistep Method

Chapter 1

Introduction

1.1 Background of Study

Mathematical models in a vast range of disciplines, 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.Quantities that change continuously in time or space are often modeled by differential equations. When everything depends on just one independent variable, we call the model an ordinary differential equation (ODE)(*Multistep methods — Fundamentals of Numerical Computation*, n.d.).

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. In many important cases of differential equations, analytic solutions are difficult or impossible to obtain and time consuming. The mathematical modelling of many problems in physics, engineering, chemistry, biology, and many more gives rise to systems of ordinary differential equation. Yet, the number of instances where an exact solution can be found by analytical means is very limited(Lambert, 1977).In many important cases of differential equations, analytic solutions are difficult or impossible to obtain and time consuming, Hence the need for an approximate, or a numerical method.

In contemporary scientific and engineering research, the formulation of complex

mathematical models often leads to the generation of differential equations that defy closed-form solutions. This persistent challenge has underscored the growing significance of approximate, or numerical, methods in tackling intricate mathematical problems. Among these methods, numerical techniques for ordinary differential equations (ODEs) stand out as indispensable tools, providing a robust means to compute numerical approximations to the solutions of these challenging equations.This necessity becomes even more pronounced when dealing with stiff systems of differential equations, where rapid variations in solution components pose additional complexities.Classical analytical methods, while powerful and elegant, encounter limitations when confronted with intricate mathematical formulations. Numerical methods step in precisely where analytical methods fall short, offering a practical avenue to obtain solutions when exact expressions are elusive.

Various advanced numerical techniques, such as implicit methods, exponential integrators, and collocation multistep methods, have proven effective in addressing the challenges posed by stiff systems. These methods excel in capturing the dynamics of stiff ODEs by incorporating strategies that adapt to the varying time scales inherent in the system. Implicit methods, for instance, allow for larger time steps, enhancing stability in the presence of stiffness.

With the advent of powerful computing technologies, numerical methods for ODEs have witnessed significant advancements. High-performance computing allows researchers and engineers to tackle more complex problems, simulate intricate physical phenomena, and explore the behavior of systems over extended time-frames. These simulations not only aid in understanding complex systems but also contribute to the optimization and design of real-world applications.

1.2 Problem Statement

Many studies on solving the equations of stiff ordinary differential equations (ODEs) have been done by researchers or mathematicians specifically. With the number of numerical methods that currently exist,extensive research has been done to unveil the comparison between their rate of convergence, number of computations, accuracy, and capability to solve certain type of test problems (Enright, Hull, & Lindberg, 1975) . The well-known numerical methods that are used widely are from the class of BDFs or commonly understood as Gear's Method (Byrne, Hindmarsh, Jackson, & Brown, 1977). However, many other methods that have evolved to this date are for solving stiff ODEs which arise in many fields of the applied sciences

(Yatim, Ibrahim, Othman, & Suleiman, 2013). The class of methods to consider in this project are Linear Multistep methods for the solutions of Initial and Boundary value problems of Ordinary Differential Equations.

In the field of computational mathematics and scientific computing, the effective analysis and numerical solution of ordinary differential equations (ODEs) play a pivotal role in modeling and understanding various real-world phenomena.ODEs arise in diverse contexts, ranging from modeling physical phenomena to simulating engineering systems, often exhibiting a spectrum of behaviors from stiff to non-stiff dynamics. Furthermore, both boundary value problems (BVPs) and initial value problems (IVPs) are prevalent scenarios requiring accurate and efficient numerical solutions.Linear multistep methods (LMMs) stand as prominent numerical techniques extensively utilized for solving ODEs, offering a balance between accuracy and computational efficiency. However, the implementation, analysis, and utilization of LMMs for tackling diverse ODE scenarios, including stiff and non-stiff problems, BVPs, and IVPs, pose significant challenges to researchers and practitioners in computational mathematics (Butcher, 2009).

Existing software solutions tailored for LMM analysis and ODE solving often lack robust capabilities to address the complexity and diversity of real-world ODE problems. For instance, proprietary software solutions like MATLAB and Wolfram Mathematica offer built-in functions for ODE solving, but they may not provide specific support for LMMs, potentially limiting the accuracy or efficiency of LMM-based solutions. Open-source libraries like SciPy and GNU Octave offer broader accessibility but may not offer as extensive support for LMM-specific analysis and customization compared to specialized software.

Furthermore, these software solutions may suffer from limitations such as:

- **Proprietary nature:** restricting access for users who cannot afford licenses or prefer open-source solutions.
- **Limited customization or extension capabilities:** particularly in commercial software, which may restrict users from implementing specialized algorithms or analyses.
- **User interface and documentation:** may not be as intuitive or user-friendly compared to specialized software designed specifically for LMM analysis and ODE solving.

The proposed solution aims to significantly enhance the ease and efficiency with

which users can explore, analyze, and apply Linear Multistep Methods (LMMs) across a wide array of Ordinary Differential Equation (ODE) scenarios. This initiative is driven by the recognition that existing software solutions often lack comprehensive support for the complexity and diversity of real-world ODE problems, particularly when addressing stiff and non-stiff problems, as well as boundary value problems (BVPs) and initial value problems (IVPs).

The development of a unified code aimed at enhancing the capabilities of researchers, educators, and students in utilizing Linear Multistep Methods (LMMs) for solving a wide range of Ordinary Differential Equation (ODE) scenarios is grounded in the critical role of computational tools in advancing computational mathematics education and research. This application seeks to provide a unified platform that facilitates the analysis and numerical solution of ODEs, thereby supporting the advancement of computational mathematics by offering a more efficient and effective means to tackle real-world problems.

The development of such an application requires a deep understanding of the mathematical principles underlying LMMs, including the identification and management of stiffness in ODEs, the selection of appropriate numerical methods, and the implementation of advanced analysis tools for error control and stability analysis. Additionally, the application must be designed to support both boundary value problems (BVPs) and initial value problems (IVPs), necessitating the development of algorithms that can adapt to the specific characteristics of these problem types.

By addressing these challenges, the proposed unified code aims to fill a critical gap in the field of computational mathematics, providing researchers and practitioners with a powerful tool for the analysis of LMM and solution of ODEs. This initiative is expected to contribute significantly to the advancement of computational mathematics education and research, enabling more effective problem-solving and decision-making in various disciplines.

1.3 Aim and Objectives

Aim:

The aim of the application is to develop a robust software tool for analyzing linear multistep methods used in solving stiff ordinary differential equations (ODEs). This tool will encompass functionalities for assessing numerical properties such as zero-stability, consistency, convergence, and error constants, providing valuable insights into the behavior and performance of these methods.

Objectives:

1. **Algorithm Development and Software Implementation**: Develop algorithms to analyze numerical properties of linear multistep methods, implement them into a user-friendly software application with intuitive interfaces.

2. **Validation and Error Analysis**: Validate algorithms by comparing results with analytical solutions, incorporate error analysis functionalities to compute error constants, ensuring accuracy and reliability.

3. **Optimization and Documentation**: Optimize software performance for efficient analysis of large-scale ODE problems, provide comprehensive documentation and user support channels for enhanced usability and understanding.

1.4 Scope of Study

The scope of this study encompasses a comprehensive exploration of linear multistep methods as applied to the solution of stiff ordinary differential equations (ODEs). It involves an in-depth analysis and implementation of various linear multistep techniques, including but not limited to Adams-Bashforth and Adams-Moulton methods. The focus is on investigating the numerical properties of these methods, such as stability, accuracy, and convergence, particularly in the context of stiff ODEs. Additionally, the study involves the development of a software tool tailored for the analysis and comparison of linear multistep methods. This entails designing intuitive user interfaces and incorporating features for parameter tuning, result visualization, and interpretation. The software's performance will be optimized to ensure efficient

handling of large-scale stiff ODE problems, with scalability and reliability in numerical computations being paramount. The validation of the developed software will be conducted through rigorous testing against established benchmarks and analytical solutions to guarantee the accuracy and reliability of results. Furthermore, the study will include the application of the developed software to real-world case studies and practical examples, demonstrating its utility and effectiveness across various domains. Finally, the study will identify any limitations encountered and suggest future research directions and enhancements to the software tool to address these limitations and improve its applicability and functionality.

1.5 Significance of Study

The significance of this study lies in its contributions to both theoretical understanding and practical applications in the field of numerical analysis and computational mathematics, specifically focusing on linear multistep methods for solving stiff ordinary differential equations (ODEs). It enhances the theoretical knowledge by providing insights into the numerical properties of these methods, including stability, accuracy, and convergence behavior. Additionally, it offers valuable tools and techniques for practitioners in scientific and engineering domains to accurately and efficiently solve stiff differential equations encountered in their respective fields, thereby advancing computational techniques used in various research, design, and decision-making processes.

1.6 Definition of Terms

1. **Ordinary differential equation:**

2. **Numerical method:** A numerical method is a difference equation involving a number of consecutive approximations $y_{n+j}, j = 0, 1, 2 \ldots k$ from which it be possible to compute sequentially the sequence $y_n | n = 0, 1, 2, \ldots N$. The integer k is called a step-number; if$k = 1$, the method is called a one-step method, while if $k > 1$, the method is called a *one-step method*

3. **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.

4. **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 (Lambert, 1977). 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.

5. **Algorithms or Packages:** These are computer code which implements numerical method, in addition to find the approximate/numerical method, it may perform other task such as estimating the error of a particular method, monitoring and updating the value of the step-length h and deciding which of the family of methods to employ at a particular stage in the solution (Lambert, 1977)

6. **Collocation method:** is a numerical technique used to solve ordinary differential equations, partial differential equations, and integral equations. The method involves selecting a finite-dimensional space of candidate solutions, usually polynomials up to a certain degree, and a number of points in the domain called collocation points. The idea is to select the solution that satisfies the given equation at the collocation points. The method provides high order accuracy and globally continuous differentiable solutions. (Wikipedia contributors, 2023)

7. **Multistep and Singlestep methods:** A single-step method is a numerical method for solving ordinary differential equations that calculates the approximate solution only using the information from the current step.Examples of this are the Taylor algorithm of order K and Runge-Kutta Methods.A multistep method is a numerical method for solving ordinary differential equations that calculates the approximate solution using the information from the current step and one or more previous steps.A crucial characteristic of multistep methods is the necessity to compute prior values of y_n (where n takes on values $1, 2, 3, \ldots, N$) for f_n (where n takes on values $1, 2, 3, \ldots, N$) through alternative methods, such as Runge-Kutta methods. This is essential for obtaining accurate values when starting the utilization of multistep methods. Alternatively, if the exact solution is known, y_n can be directly calculated (Fatokun, 1992).For example:

$$y_{n+1} = y_n + h \cdot f(t_n, y_n) \tag{1.1}$$

$$y_{n+2} = y_{n+1} + \frac{h}{2}\left[3f(t_{n+1}, y_{n+1}) - f(t_n, y_n)\right] \tag{1.2}$$

   A disadvantage of multistep methods is that they are not self-starting.But on the other hand, they are faster than the single-step methods. In addition, Multistep methods can be more stable and efficient for certain types of ODEs, especially when dealing with stiff systems (Fatokun, 1992).

8. **Linear Multistep Method (LMM):** A numerical method for approximating the solution of an ordinary differential equation (ODE) at discrete time points using a linear combination of past and present function values. LMMs typically involve using multiple previous function values to compute the next value, hence the term "multistep."

9. **Explicit Method:** A type of linear multistep method in which the value of the unknown function at the next time step is explicitly computed using only known values at previous time steps. The formula for the next value does not involve solving any equations or iterative procedures.

10. **Implicit Method:** A type of linear multistep method in which the value of the unknown function at the next time step is computed using known values at previous time steps, as well as the value at the next time step itself. The formula for the next value involves solving equations or iterative procedures, making implicit methods more computationally intensive than explicit methods.

11. **Adams-Bashforth Method:** A specific family of explicit linear multistep

methods used for numerical integration of ordinary differential equations. Adams-Bashforth methods use interpolation of previous function values to approximate the derivative of the function, allowing for the computation of future function values.

12. **Adams-Moulton Method:** A specific family of implicit linear multistep methods used for numerical integration of ordinary differential equations. Adams-Moulton methods involve using interpolation of previous function values, as well as the value at the next time step, to compute the next function value, typically requiring solving equations or iterative procedures.

13. **Stability:** A property of linear multistep methods indicating the behavior of the numerical solution with respect to small perturbations or errors. A stable method produces a solution that does not grow exponentially with time and remains bounded, ensuring accuracy and reliability of the numerical solution.

14. **Convergence:** A property of linear multistep methods indicating the behavior of the numerical solution as the step size approaches zero. A convergent method produces a solution that approaches the true solution of the ordinary differential equation as the step size decreases, ensuring accuracy and consistency of the numerical solution.

15. **Order of Accuracy:** A measure of the accuracy of a linear multistep method, indicating the rate at which the numerical solution approaches the true solution as the step size decreases. Higher-order methods have higher order of accuracy, meaning they converge to the true solution faster as the step size decreases.

Chapter 2

Literature Review

2.1 Introduction

One of the more challenging classes of problems in numerical computation is the solution of stiff equations and stiff systems. These problems arise from various physical situations but were likely first identified in chemical kinetics. Finding numerical solutions to stiff systems has been a significant challenge for numerical analysts. A potentially good numerical method for solutions of stiff systems must possess certain qualities in terms of its region of absolute stability and accuracy (Quresh, Ramos, Soomro, Akinfenwa, & Akanbi, 2024).

2.2 Exploring Stiff Systems of Ordinary Differential Equations: Characteristics, Solutions, and Numerical Methods

Stiff ordinary differential equations (ODEs) pose significant challenges in scientific and engineering applications due to their unique properties, which can lead to numerical instability and inaccuracy in traditional numerical methods.Stiff ODEs are characterized by a rapid change in the solution over a short period, followed by a period of slow change. This characteristic can lead to numerical instability in traditional numerical methods, making it difficult to accurately solve these equations over long time scales.

2.3 Multistep Methods

Multistep methods are a category of numerical methods used to solve ordinary differential equations (ODEs), including stiff systems. They are called "multistep" because they use information from multiple previous steps to compute the next step. This makes them particularly effective for problems with stiff behavior, where the slope of the solution changes rapidly (Jeon, Bak, & Bu, 2019). One of the main strengths of multistep methods is their ability to handle temporal evolution. Because they use information from previous steps, they can adapt to changes in the behavior of the solution over time. This makes them particularly effective for problems where the solution evolves in a complex way, such as stiff systems (Jeon et al., 2019).

In terms of applications, multistep methods are widely used in various fields, including physics, engineering, and economics. They are used to solve a wide range of problems, from simulating the motion of celestial bodies to modeling economic growth. In the context of boundary value problems (BVPs), multistep methods can be used to solve problems where the solution varies over time and space (Jeon et al., 2019).

A general linear multistep method can be expressed as:

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h \sum_{j=0}^{k} \beta_j f(t_{n+j}, y_{n+j})$$

where:

- k is the number of previous steps to use
- h is the step length
- t_n is the current time
- y_n is the solution at the current time
- $f(t_n, y_n)$ is the derivative of the solution at the current time
- α_j and β_j are the coefficients of the method
- y_{n+j} is the solution at the previous time steps

Linear multistep methods are generally defined by their coefficients α_j and β_j, and the choice of these coefficients determines the order and stability properties of the method. Common examples of linear multistep methods include the backward Euler method, the Adams-Bashforth methods, and the Adams-Moulton methods.

The solution at the next time step, y_{n+1}, can be obtained by rearranging the terms in the above formula:

$$y_{n+1} = \frac{1}{\alpha_0}\left(h\sum_{j=0}^{k}\beta_j f(t_{n+j}, y_{n+j}) - \sum_{j=1}^{k}\alpha_j y_{n+j}\right)$$

title

However, like all numerical methods, multistep methods have their limitations. For example, they can suffer from numerical diffusion, where the solution becomes smoother than expected due to roundoff errors. This can lead to inaccuracies in the solution, especially for problems with stiff behavior. Furthermore, the choice of the number of previous steps to use can significantly affect the performance of the method. More steps can lead to more accurate solutions, but they also increase the computational cost (Jeon et al., 2019). The ode15s and ode23 solvers in MATLAB are examples of multistep methods used for solving stiff systems of BVPs or IVPs. The ode15s solver uses an implicit Runge-Kutta method of order 15, with the embedded 6th order BDF method as a predictor. It is able to handle stiff and nonstiff problems and can be used with either the Jacobian of the system or a numerical approximation. On the other hand, the ode23 solver uses an implicit Runge-Kutta method of order 2, with the embedded 3rd order BDF method as a predictor. It is also able to handle stiff and nonstiff problems (Wong, 2020). The bvp4c and bvp5c solvers in MATLAB are examples of multistep methods used for solving BVPs. They use a collocation method with a finite difference code that implements the Lobatto IIIa formula. This is a collocation formula, and the collocation polynomial provides a C1-continuous solution that is fourth-order or fifth-order accurate uniformly in the interval of integration(MatLab).

2.4 Implicit Methods

Implicit methods are commonly used to solve stiff systems of ordinary differential equations (ODEs). They involve the solution at the next step, which requires solving a nonlinear equation at each step. These methods are generally more stable than explicit methods, which only use the current step's solution. However, they can be more computationally expensive due to the need to solve a system of equations at each step (Thohura & Rahman, 2013).

One of the mostly widely used implicit methods for stiff ODEs is the Backward Differentiation Formula (BDF).Backward Differentiation Formulas (BDFs) are a family of implicit numerical methods commonly used to solve stiff systems of ordinary differential equations (ODEs).BDFs use information from the future (at the next time level) to update the solution at the current time level. The backward nature of the method enables stability for stiff problems (Press, Teukolsky, Vetterling, & Flannery, 2007).

The general form of a BDF of order k is given by:

$$\alpha_0 y_n + \alpha_1 y_{n-1} + \alpha_2 y_{n-2} + \ldots + \alpha_k y_{n-k} = h \cdot f(t_n, y_n)$$

For example, the BDF of order 1 (Backward Euler method) is:

$$y_n = y_{n-1} + h \cdot f(t_n, y_n)$$

And the BDF of order 2 is:

$$\frac{3}{2} y_n - 2y_{n-1} + \frac{1}{2} y_{n-2} = h \cdot f(t_n, y_n)$$

It works by approximating the solution at the next step using a polynomial of degree less than or equal to the method order. This makes BDF suitable for stiff ODEs, as it avoids the loss of accuracy associated with steep slopes in the solution.BDFs are widely implemented in numerical software packages for solving stiff ODEs. Popular implementations include ode23s and ode45s (Shampine & Reichelt, 1997),the Livermore Solver for Ordinary Differential Equations (LSODA),the Differential Algebraic System Solver (DASSL),GEAR, DIFSUB, and EPISODE (Yatim et al., 2013) which is a collection of FORTRAN subroutines designed to facilitate the automated resolution of problems, minimizing the level of effort needed when

encountering potential challenges in the problem-solving process (Thohura & Rahman, 2013).Each of this tools have their strength and weakness; DIFSUB has no graphical user interface which makes it harder for users who are not comfortable working from command-line or with text-based interfaces and it a very old package, therefore finding support becomes challenging; GEAR uses FORTRAN package, uses need to have some knowledge of FORTRAN to use it effectively; EPISODE is a deprecated package which performs faster than GEAR in solving waves or active solutions, but the reverse for linear or decaying problems (Byrne et al., 1977),With this limitations comes the aim of this project.

The Runge-Kutta-Fehlberg (RKF45), is another widely used implicit method for solving ordinary differential equations, including stiff ODEs. it combines both explicit and implicit methods to achieve high accuracy and stability (Stone, Alferman, & Niemeyer, 2017).

$$
\begin{aligned}
k_1 &= h \cdot f(t_n, y_n), \\
k_2 &= h \cdot f(t_n + \frac{1}{4}h, y_n + \frac{1}{4}k_1), \\
k_3 &= h \cdot f(t_n + \frac{3}{8}h, y_n + \frac{3}{32}k_1 + \frac{9}{32}k_2), \\
k_4 &= h \cdot f(t_n + \frac{12}{13}h, y_n + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3), \\
k_5 &= h \cdot f(t_n + h, y_n + \frac{439}{216}k_1 - 8k_2 + \frac{3680}{513}k_3 - \frac{845}{4104}k_4), \\
k_6 &= h \cdot f(t_n + \frac{1}{2}h, y_n - \frac{8}{27}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5).
\end{aligned}
$$

$y_{n+1} = y_n + \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6.$

$\texttt{Error} = \frac{1}{360}h(-127k_1 + 845k_3 - 28561k_4 + 9k_5 - 2k_6).$

Although there exist no specific stiff system solver that utilizes the RKF45 method exclusively.The RKF45 method is a variant of the Dormand-Prince method, which is a popular implicit Runge-Kutta method used in MATLAB's ode15s solver (Burkardt, 2010). It's worth noting that while the RKF45 method is not used by a specific stiff system solver, it is a powerful tool for solving stiff systems of ODEs. Its ability to accurately estimate the local truncation error and adapt the step size accordingly allows it to handle stiff problems effectively.(Burkardt, 2010)

2.5 Collocation Methods

Collocation methods are a class of numerical methods used to solve ordinary differential equations (ODEs), including stiff systems. They work by choosing specific points (collocation points) within the domain where the solution is sought. The solution is then approximated as a polynomial at these points, and the differential equation is converted into a set of algebraic equations by enforcing the equations at these points. Numerous studies have demonstrated the efficacy of collocation methods in various scientific and engineering applications. Examples include the modeling of chemical reactions, structural dynamics, and climate phenomena. The ability of collocation methods to efficiently capture rapid changes in the system dynamics makes them well-suited for problems characterized by stiff components. In the realm of stiff systems, collocation methods exhibit notable advantages, offering enhanced efficiency by directly manipulating the coefficients of the differential equation, ensuring heightened accuracy in capturing stiff behavior, and providing increased stability; however, their implementation complexity and sensitivity to the choice of collocation points present challenges(Faleichik, 2009).

Let's consider a simple second-order ordinary differential equation (ODE) as an example:

$$y''(t) = f(t, y, y')$$

with boundary conditions $y(a) = \alpha$ and $y(b) = \beta$. The objective is to find the function $y(t)$ that satisfies the differential equation and boundary conditions.

The collocation method involves selecting a set of collocation points $\{t_1, t_2, \ldots, t_n\}$ within the domain $[a, b]$. At these collocation points, the differential equation is enforced. This results in a system of algebraic equations that can be solved to obtain the values of $y(t_i)$ at the collocation points.

Let $y_i = y(t_i)$ and $y'_i = y'(t_i)$. Applying the collocation method to the differential equation, we have:

$$y''_i = f(t_i, y_i, y'_i)$$

This equation is enforced at each collocation point t_i, resulting in a set of algebraic equations:

$$y_1'' = f(t_1, y_1, y_1')$$
$$y_2'' = f(t_2, y_2, y_2')$$
$$\vdots$$
$$y_n'' = f(t_n, y_n, y_n')$$

Together with the boundary conditions, these equations form a system that can be solved for the unknowns y_i and y_i'. The accuracy and stability of the collocation method depend on the choice of collocation points and the method used to solve the resulting system of equations.

Chapter 3

Methodology

3.1 Introduction

This chapter outlines the detailed methodology employed to address the numerical challenges associated with stiff systems of ordinary differential equations (ODEs). Stiffness arises in systems characterized by vastly different timescales, rapid transitions, or abrupt changes in dynamics. The goal of this methodology is to implement effective numerical techniques, specifically collocation and multistep methods, to provide accurate and stable solutions for stiff ODEs.

3.2 Problem Formulation

The starting point involves a clear definition of the stiff boundary value problem (BVP). The problem is expressed as a set of ordinary differential equations subject to appropriate boundary conditions. In this study, we focus on two-point BVPs, and the differential equation, boundary conditions, and relevant parameters are clearly specified.

3.3 Choice of Methods

To tackle stiffness, a careful selection of numerical methods is crucial. We opt for a combined approach involving collocation and multistep methods. Collocation points are strategically chosen within the problem domain, and the differential equation is

enforced at these points. For time integration, multistep methods, known for their stability and efficiency, are chosen.

3.4 Discretization and System Formulation

The problem domain is discretized into intervals, and collocation points are judiciously selected. The chosen collocation method is then employed to formulate a system of algebraic equations based on the discretized differential equation. Additionally, for multistep methods, initial conditions and recurrence relations are established.

3.5 Implementation

The solver is implemented in Python, leveraging its versatile libraries for efficient mathematical operations. Functions and classes representing the solver are developed, incorporating numerical algorithms such as Newton-Raphson iteration for solving algebraic systems.

3.6 Time Stepping

For multistep methods, the time-stepping process is implemented. The solution is updated at each time step using the recurrence relations derived earlier. This step ensures the accurate propagation of the solution through time.

3.7 Boundary Conditions

Incorporating boundary conditions into the solver is crucial for obtaining physically meaningful solutions. The solver is designed to satisfy both the differential equation and the specified boundary conditions.

3.8 Post-Processing

After obtaining the numerical solution, post-processing steps are implemented. This includes visualization and analysis of results, aiding in a deeper understanding of the system's behavior.

3.9 Validation and Optimization

To validate the implemented solver, comparisons are made with analytical solutions or benchmark problems. Optimization is performed to enhance the code's efficiency, with considerations for vectorization and parallelization.

3.10 Documentation and Testing

Comprehensive documentation is provided, detailing the algorithms and methodologies employed. Rigorous testing is conducted to ensure the accuracy and reliability of the solver under various scenarios.

3.11 Utilizing Libraries and Frameworks

Existing Python libraries, such as NumPy and SciPy, are leveraged for efficient mathematical operations. This ensures that the implementation benefits from optimized, well-established routines.

3.12 Iterative Refinement

The methodology undergoes iterative refinement based on feedback, experiments, and identified areas of improvement. This iterative process aims to enhance the solver's performance and robustness.

Table 3.1: Comparison of Solvers

Solver	**Method Used**	**UI Interface**	**Programming Language**	**Speed**	**Ease of Learning**
Your Project	Collocation & Multistep	Custom UI (if applicable)	Python & Flutter	TBD	Moderate
GEAR	Implicit Methods	Command Line	Fortran	High	Moderate
EPISODE	Explicit Methods	Graphical UI	C++	Moderate	Moderate
MATLAB	Various (ode15s, ode23, etc.)	MATLAB GUI	MATLAB	High	Easy

3.13 Conclusion

This chapter elucidates a systematic and comprehensive methodology for addressing stiff ODEs through a combination of collocation and multistep methods. The methodology is designed to be flexible and adaptable, allowing for the incorporation of additional numerical techniques and algorithms. The next chapter presents the results of applying this methodology to a variety of stiff BVPs and IVPs.

References

Burkardt, J. (2010, March). *Rkf45 - a fortran90 library which implements the runge-kutta-fehlberg ode solver.* https://people.math.sc.edu/Burkardt/f_src/rkf45/rkf45.html.

Butcher, J. (2009). General linear methods for ordinary differential equations. *Mathematics and Computers in Simulation*, *79*(6), 1834-1845. Retrieved from https://www.sciencedirect.com/science/article/pii/S0378475407001462 (Applied and Computational MathematicsSelected Papers of the Sixth PanAmerican WorkshopJuly 23-28, 2006, Huatulco-Oaxaca, Mexico) doi: https://doi.org/10.1016/j.matcom.2007.02.006

Byrne, G. D., Hindmarsh, A. C., Jackson, K. R., & Brown, H. G. (1977). A comparison of two ode codes: Gear and episode. *Computers & Chemical Engineering*, *1*(2), 125–131. Retrieved from https://www.sciencedirect.com/science/article/pii/0098135477800185 doi: 10.1016/0098-1354(77)80018-5

Enright, W. H., Hull, T. E., & Lindberg, B. (1975). Comparing numerical methods for stiff systems of o.d.e:s. *BIT Numerical Mathematics*, *15*(1), 10–48. Retrieved from https://doi.org/10.1007/BF01932994 doi: 10.1007/BF01932994

Faleichik, B. (2009, April). *Explicit implementation of collocation methods for stiff systems with complex spectrum 1.* Retrieved from https://api.semanticscholar.org/CorpusID:9177554

Fatokun, J. O. (1992, August). Power series collocation methods for the initial value problems. *Unilorin*, *1*, 10.

Jeon, Y., Bak, S., & Bu, S. (2019). Reinterpretation of multi-stage methods for stiff systems: A comprehensive review on current perspectives and recommendations. *Mathematics*, *7*(12). Retrieved from https://www.mdpi.com/2227-7390/7/12/1158 doi: 10.3390/math7121158

Lambert, J. D. (1977). The initial value problem for ordinary differential equations. In D. Jacobs (Ed.), *The state of the art in numerical analysis* (pp. 451–501). New York: Academic Press.

Multistep methods — fundamentals of numerical computation. (n.d.). Retrieved from https://fncbook.github.io/fnc/ivp/multistep.html (Accessed: 2023-09-03)

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). *Numerical recipes: The art of scientific computing* (3rd ed.). Cambridge University Press.

Quresh, S., Ramos, H., Soomro, A., Akinfenwa, O. A., & Akanbi, M. A. (2024). Numerical integration of stiff problems using a new time-efficient hybrid block solver based on collocation and interpolation techniques. *Mathematics and Computers in Simulation*. Retrieved from https://www.sciencedirect.com/science/article/pii/S0378475424000016 doi: https://doi.org/10.1016/j.matcom.2024.01.001

Shampine, L. F., & Reichelt, M. W. (1997). The matlab ode suite. *SIAM Journal on Scientific Computing*, *18*, 1–22.

Stone, C. P., Alferman, A. T., & Niemeyer, K. E. (2017). Accelerating finite-rate chemical kinetics with coprocessors: Comparing vectorization methods on gpus, mics, and cpus. *Computational Science and Engineering, LLC*.

Thohura, S., & Rahman, A. (2013). Numerical approach for solving stiff differential equations: A comparative study. *Global Journal of Science Frontier Research. Mathematics and Decision Sciences*, *13*(6), Version 1.0. (Type: Double Blind Peer Reviewed International Research Journal)

Wikipedia contributors. (2023). *Collocation method — Wikipedia, the free encyclopedia.* Retrieved from https://en.wikipedia.org/w/index.php?title=Collocation_method&oldid=1166346639 ([Online; accessed 14-January-2024])

Wong, J. (2020, April 12). *Math 563 lecture notes: Numerical methods for boundary value problems.* Lecture Notes.

Yatim, S. A. M., Ibrahim, Z. B., Othman, K. I., & Suleiman, M. B. (2013). A numerical algorithm for solving stiff ordinary differential equations. *Mathematical Problems in Engineering*, *2013*, Article ID 989381, 11. Retrieved from https://doi.org/10.1155/2013/989381 doi: 10.1155/2013/989381