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

OKWHAROBO, Solomon Monday

January 17, 2024

0.1 Chapter 1: Introduction

0.1.1 Background

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.
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.

0.1.2 Problem Statement

Many studies on solving the equations of stiff ordinary differential equations (ODEs) have been done by researchers or mathematicians specifically. With the numbers of numerical methods that currently exist in the literature, 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 problems considered in this paper are for the numerical solution of the boundary value problem.

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.

0.1.3 Aim and Objectives

Aim:

The primary goal of this project is to develop a unified Python-based numerical code that seamlessly integrates collocation and multistep methods for the efficient and accurate solution of stiff systems of boundary value problems (BVPs)

Objectives:

1. Develop a unified code that seamlessly integrates collocation and multistep methods.
2. Assess the accuracy, efficiency, and versatility of the proposed code.
3. Provide a user-friendly tool for researchers and practitioners working with stiff BVPs.

0.1.4 Scope of Study

The project centers on the implementation of a numerical code using the Python programming language that integrates collocation and multistep methods. The code aims to solve stiff systems of boundary value problems (BVPs) efficiently and accurately and also designing the code with a user-friendly interface to cater to researchers and practitioners working with stiff BVPs in flutter

0.1.5 Significance of Study

This codebase tends to provide a tool that combines the strengths of collocation and multistep methods, offering a more comprehensive approach to solving stiff BVPs, and also a user-friendly interface for researchers and practitioners working with stiff BVPs in flutter.

0.1.6 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 solition (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)

0.2 Chapter 2: Literature Review

0.2.1 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).

In quantum mechanics, stiff equations are used to model the behavior of particles in potential wells. For example, the finite square well problem, where a particle is confined in a box, is a classic application of stiff boundary value problems (BVPs) (Cappellaro, Year of Publicationa). In nuclear physics, stiff BVPs are employed to model the decay of radioactive substances. An example of this is the unbound problem in quantum mechanics, where a particle moves freely in a potential barrier (Cappellaro, Year of Publicationb), and many other fields where stiff BVPs arise.Hence there is a need to develop numerical methods to solve this cases.

0.2.2 Stiff BVPs in Scientific and Engineering Applications

Stiff Boundary Value Problems (BVPs) are prevalent in various scientific and engineering fields due to their complex nature. These problems often arise in areas such as mathematical physics, fluid dynamics, and electrical engineering. They pose challenges because of the difficulty in finding exact solutions and the sensitivity to small changes in the problem parameters.

One example of a stiff BVP is the Troesch's problem, which is often used in numerical analysis for studying the behavior of stiff differential equations. Consider the second-order linear ordinary differential equation (ODE):

$$y''(x) + p(x) \cdot y'(x) + q(x) \cdot y(x) = f(x) \tag{1}$$

For Troesch's problem, specific functions are chosen for $p(x)$, $q(x)$, and $f(x)$ that make the solution challenging for numerical methods. The exact solution is provided as part of the problem formulation.The problem exhibits a stiffness, which makes it a challenging.In a research by (Dragunov, 2021), stiff BVPs are addressed through transformation-based approach, where the auth (Dragunov, 2021) proposed series of transformations that can reduce the stiffness of the problem, making it easier to find a numerical solution. The methodology framework presented in this work can be seen as an approach that enhances the "stiffness resistance" of virtually all existing numerical techniques designed for the solution of two-point boundary value problems (BVPs) (Dragunov, 2021).In this chapter we focus on different numerical methods that can be used to solve stiff problems

0.2.3 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:

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

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 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)

0.2.4 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).

0.2.5 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).

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(MATHLAB).

0.3 Chapter 3: Methodology

0.3.1 Implementation of Collocation and Multistep Methods in Python

1. Define the Problem

- Clearly define the stiff Boundary Value Problem (BVP) with the differential equation, boundary conditions, and parameters. This study will be limited to a two-point boundary value problem.

2. Choose Collocation and Multistep Methods

- Select appropriate methods based on problem characteristics (e.g., Gaussian, Chebyshev, Legendre collocation, Adams-Bashforth, Adams-Moulton).

0.3.2 3. Discretize the Domain

- Divide the problem domain into discrete points or intervals. - Choose collocation points where the differential equation will be enforced.

4. Formulate the System of Equations

- Use the chosen collocation method to formulate a system of algebraic equations based on the discretized differential equation. - For multistep methods, set up initial conditions and recurrence relations.

5. Implement the Solver

- Write functions or classes representing the solver. - Implement numerical algorithms (e.g., Newton-Raphson iteration, direct solvers).

6. Time Stepping for Multistep Methods

- Implement the time-stepping process for multistep methods. - Update the solution using recurrence relations at each time step.

7. Boundary Conditions

- Incorporate boundary conditions into the solver to satisfy both the differential equation and specified conditions.

8. Post-Processing

- Implement post-processing steps to visualize and analyze results. - Plot the solution to observe system behavior.

9. Validate and Optimize

- Validate by comparing results with analytical solutions or benchmarks. - Optimize for efficiency, considering vectorization and parallelization.

10. Documentation and Testing

- Provide documentation, including algorithm details. - Conduct thorough testing for accuracy and reliability.

11. Use Libraries and Frameworks

- Leverage existing Python libraries and frameworks (e.g., NumPy, SciPy) for efficient mathematical operations.

12. Iterate and Refine

- Iterate based on feedback, experiments, or identified areas of improvement.

Example Code Snippet (Python, NumPy, SciPy)

```
import numpy as np
from scipy.optimize import fsolve

def collocation_solver(problem_parameters):
    # Define collocation points and weights
    collocation_points, weights = np.polynomial.legendre.leggauss(
        problem_parameters['num_collocation_points'])

    def system_equations(y, *args):
        # System of algebraic equations based on the collocation method
        # (Replace with your specific equations)
        return np.array([...])

    # Initial guess for the solution
    initial_guess = np.zeros(problem_parameters['num_collocation_points'])

    # Solve the system of equations using fsolve from SciPy
```

```
    solution = fsolve(system_equations, initial_guess)

    return solution

# Example usage
problem_parameters = {'num_collocation_points': 5}
result = collocation_solver(problem_parameters)
print(result)
```

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.

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

Cappellaro. (Year of Publicationa). *Bound problems.* Retrieved from https://phys.libretexts.org/Bookshelves/Nuclear_and_Particle_Physics/Introduction_to_Applied_Nuclear_Physics_(Cappellaro)/04%3A_Energy_Levels/4.01%3A_Bound_Problems

Cappellaro. (Year of Publicationb). *Unbound problems in quantum mechanics.* Retrieved from https://phys.libretexts.org/Bookshelves/Nuclear_and_Particle_Physics/Introduction_to_Applied_Nuclear_Physics_(Cappellaro)/03%3A_Radioactive_Decay_Part_I/3.02%3A_Unbound_Problems_in_Quantum_Mechanics

Dragunov, D. (2021). A transformation-based approach for solving stiff two-point boundary value problems. *arXiv*. (Submitted on 18 Jun 2021 (v1), last revised 28 Nov 2021 (this version, v2))

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

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.

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