

INVESTIGATE A RANGE OF NUMERICAL IMPLICIT METHODS FOR SOLVING STIFF DIFFERENTIAL EQUATIONS

NAZNEEN PARVEEN¹, NAJMUDDIN AHMAD^{1*}

Manuscript received: 27.10.2025; Accepted paper: 15.06.2026;

Published online: 30.09.2026.

Abstract. The phenomena of stiffness and general-purpose methods for solving stiff differential equations are the primary topics of this investigation. We are attempting to characterize the distinct features of stiff differential equations as well as the precise location of the problem. A numerical technique is required since analytical solutions do not exist for the majority of practical stiff systems. Numerous computer programs such as DIFSUB, GEAR, EPISODE, and others provide implementations of these methods. For ordinary differential equations, the backward differentiation formulas (BDFs) are the go-to tools for solving tough starting value issues. This study focuses on EPISODE, an extremely efficient algorithm, using a variable-coefficient backward differentiation formula as its foundation. Even though the strategy is generally effective, our research reveals that novice users may encounter difficulties in some areas. All of those issues have been identified, and solutions are being suggested for future revisions.

Keywords: Stiff differential equations; implicit numerical methods; stability analysis.

Mathematics Subject Classification: 65L04; 65L06; 65L50.

1. INTRODUCTION

The occurrence of events with drastically diverse time scales gives rise to a significant subset of differential equations used in initial value problems, known as stiff differential equations. The concept of stiffness lacks a consensus definition. Stiffness is a crucial but nuanced term when numerically solving ordinary differential equations [1]. The starting condition, the interval in question, and the differential equation all have a role.

Engineers study electrical circuits, vibrations, chemical processes, and many more areas where stiff ordinary differential equation systems cause initial value issues. Many non-industrial fields, including biology and weather prediction, as well as astrochemical kinetics, control systems, and electronics, rely on stiff differential equations.

A differential equation set is said to be "stiff" if only a very small step is required for proper integration. Simply put, a "stiff" set of differential equations consists of two or more "time constants" that differ significantly from one another. In this example, time serves as the joint independent variable. Along with this, Shampine and Gear provided a stricter definition of stiffness: We say that a problem is stiff if there is at least one extremely stable component and no unstable component in the solution (i.e., no big, positive real part in any eigenvalue of the Jacobian matrix) [2]. Also, a problem won't be considered stiff until the solution has

¹Integral University, Department of Mathematics and Statistics, 226026 Lucknow, India.

E-mail: knaznnenkhan786@gmail.com.

* Corresponding author: najmuddinahmad33@gmail.com.

positive and negative eigenvalue correlations. Therefore, a condition could be mild for certain periods and severe for others.

In the process of resolving the (vector) equation system

$$y' = f(t, y), y(t_0) = a(\text{given}) \quad (1)$$

When looking for a solution, it's important to think about how similar ones behave. This is because when we move from $y_n = y(x_n)$ to y_{n+1} approaching $y(x_n + h)$, we will undoubtedly make mistakes that will move us away from the ideal integral curve and toward one that is closer to it. Following this new curve will cause the resultant error to be dependent on the relative behavior of the two solution curves, assuming no more mistakes are made [3]. As an illustration, consider the one-equation equation.

$$y' = A(y - p(t)) + p'(t), y(t_0) = a \quad (2)$$

Given that A is a constant. This analytical problem's answer is

$$y(t) = (a - p(0))\exp(At) + p(t) \quad (3)$$

Equation (3) indicates a stiff issue after the transient At has passed (i.e., when At is less than the error tolerance of interest), but it is not stiff during the transient area if A is very negative and $p(t)$ is slowly changing. If (1) is linear with a constant Jacobian J , where $J = f_y$ is the corresponding Jacobian matrix, it will become stiff after the quickest transient has passed. However, it will not be stiff during the first transient. Finding a good numerical solution to a stiff differential equation problem requires baby steps since, as we can see, the sought-after solution varies slowly while surrounding solutions fluctuate fast [4]. Stiffness is an efficiency problem. There would be no need to worry about stiffness if computation time were not an issue. Although they are time-consuming, nonstiff solutions may tackle stiff challenges.

2. NUMERICAL SOLUTION OF STIFF DIFFERENTIAL EQUATIONS

Efficient solution of stiff differential equations is necessary since they appear in several scientific and engineering domains. A numerical technique is required since analytical solutions do not exist for the majority of practical stiff systems. To ensure that the solution remains stable, traditional techniques like Euler, explicit Runge-Kutta, and Adams-Moulton are limited to very short step sizes [5]. This might necessitate spending a lot of time in front of a computer. Even though they are mathematically well-conditioned, traditional numerical methods are severely constrained by the step size constraint imposed by numerical stability, making it nearly impossible to solve systems of ordinary differential equations in many areas, particularly chemical applications. One feature of these stiff equations is the existence of transient components. These components are minor compared to the numerical solution, but they force typical numerical techniques to use step sizes on the scale of the problem's shortest time constant.

Over the past three decades, numerical stiff ODE solvers, methods for solving such equations, and the linear algebra that goes along with them have all seen significant advancements. This led to the development of several ODE solutions that are both efficient and dependable. To make the most of the state-of-the-art solvers and effectively and correctly

handle computationally demanding models in diverse fields, a deep grasp of the problem formulation is necessary. The numerical solution method for systems of ODEs and the solution of the resultant linear algebraic system are the two main parts of the numerical solution algorithm of a typical stiff ODE solver package.

The structure of the resulting matrix and its relationship to the linear system has important implications for computing [6]. Before delving into the specifics of sophisticated ODE solvers, it's helpful to take a quick look at the linear algebra and solution techniques that underpin stiff systems of ODEs. Most often, a given starting condition, $y(t_0) = a$, and an N -dimensional system of ordinary differential equations (ODEs) of type (1) are solved using certain multistep techniques. To move the solution from one mesh point to the next in time t , multistep methods use multiple discrete time mesh past values of the variable y and its rate of change f with respect to time t (i.e., the abundances' past values and the rate equations). A multistep procedure with k steps often takes the following form:

$$\sum_{i=0}^k \alpha_i y_{n-i} = h \sum_{i=0}^k \beta_i f_{n-i} \quad (4)$$

The time step size, the mesh number, and the constants α_i and β_i are all dependent on the method's order. Techniques commonly used by Adams that heavily rely on the historical values of

$$y_n = y_{n-1} + h \sum_{i=0}^k \beta_i f_{n-i} \quad (5)$$

are the most widely used, step-by-step strategies for dealing with nonstiff issues. At each stage, a nonlinear system must be solved. To progress the integration, a basic functional iteration with an initial guess, or predictor estimate, is often used. A convergence test then marks the end of the process. Functional iteration fails to converge, and basic iteration restricts the step size too much when dealing with stiff issues where variables may change suddenly (for instance, the rate equations f are strongly dependent on abundances y in very short time intervals). Therefore, in the absence of a step size constraint, stiffness necessitates the employment of implicit techniques including indefinite stability areas [7].

One of the first numerical approaches to solving stiff ordinary differential equations (ODEs) was the backward difference formulae (BDF) technique, which has an unlimited zone of absolute stability (Curtiss and Hirschfelder, 1952). ODE solvers use a standard form for their BDF.

$$y_n = \sum_{i=0}^k \alpha_i y_{n-i} + h \beta_0 f_n \quad (6)$$

The k -step BDF procedures, where α_i and β_0 are the coefficients of the k -th order. It has already been stated that when problems are stiff, a basic functional iteration will often fail to converge, leading to the adoption of a Newton iteration or similar method to solve the

ensuing nonlinear system. The solution of a matrix P with dimensions $N \times N$ is involved in the Newton iteration.

$$P \approx I - h\beta_0 J \quad (7)$$

When the related Jacobian matrix, denoted as $J = \partial f / \partial y$, and an identity matrix of size $N \times N$ are present. The precision of the solution, in turn influenced by the solution to this linear algebraic system, has a significant impact on the computational time required to solve stiff problems. For difficult problems, the ordinary differential equation (ODE) solvers use a modified Newton iteration that makes it possible to use the Jacobian matrix in an effective way when computing, storing, and using it. Direct techniques and iterative methods are typically the two types of approaches used to solve linear algebraic systems. When solving linear systems, the most popular direct approach is the Gaussian elimination technique, which divides the matrix into lower and upper triangular elements. The GEAR, LSODE, and VODE solvers employ this strategy to solve the resulting linear system. While the Jacobi iteration is the most basic iterative scheme for solving linear systems, Krylov subspace methods have seen extensive use in computational fluid dynamics and other real-world applications due to their sophisticated iterative solution techniques based on a sequence of orthogonal vectors and matrix-vector multiplications [8]. After that, the LSODPK and VODPK ODE solvers solve the resultant linear system using Preconditioned Krylov iterative approaches.

The following are examples of more popular approaches to stiff equations: (1) Gear (1969, 1971) analyzed and used variable-order approaches based on backward differentiation multistep formulae; Hindmarsh (1974) and Byrne and Hindmarsh (1975) revised and examined these methods. (2) Dahlquist (1963) suggested, and Lindberg (1971, 1972) investigated, methods based on the trapezoidal rule. (3) Ehle (1968) investigated implicit Runge-Kutta algorithms that work for stiff equations based on the formulations of Butcher (1964). (4) Approaches that, like those investigated and put into practice by Lawson and Ehle (1972), rely on solving the changed problem using conventional procedures after applying preliminary mathematical transformations to eliminate stiffness. (5) Techniques developed by Enright (1974) and Linger and Willoughby (1967) that rely on second derivative multistep formulae.

Despite the development of several approaches and the suggestion of numerous more fundamental formulae for stiff equations, practitioners have had little access to assistance or advice on how to choose an appropriate method for their specific situation until recently [9].

Stability constraints in stiff differential equations cause the solver to perform many short time steps. This occurs when we integrate an independent variable across two or more scales that are substantially distinct from each other in a system of linked differential equations. Another approach is to think about what has to happen when two solution components call for drastically different time steps. Consider the case when our answer is the result of combining two exponential decay curves, one with a very fast decay and one with a very slow decline. The quick curve will have faded away; thus, the slowly decaying curve will dominate, except for the few time steps distant from the original state. However, we will be confined to tiny time steps despite the dominant component's need for larger ones due to the variable time step routine's need to satisfy stability criteria for both components [10]. We are unable to stray from taking very small-time steps for a solution component that contributes an infinitesimally small amount to the overall solution when dealing with stiff equations. That is to say, instead of briskly progressing toward a solution, we are compelled to proceed slowly.

Backward differentiation formula-based methods, such as DIFSUB (Gear (1971a, 1971b)) and GEAR. Rev. 3. This work evaluates (Hindmarsh, 1974) and EPISODE (Byrne and Hindmarsh, 1975). You may utilize GEAR and EPISODE, which are universal ODE programs, to solve Stiff and nonstiff issues. When using the nonstiff technique option, they seem to outperform other advanced nonstiff system solvers in the nonstiff scenario. These programs enable the usage of the Jacobian matrix in the stiff situation and provide procedures for solving the related linear problems in full matrix form.

Table 1. This is a condensed version of the list of general-purpose solver software that can solve ODE problems.

Solver Name	Release Year	Description	Special Numerical Methods	Language
GEAR	1974	Standard solver, supersedes DIFSUB-Gear 1968	Adams methods for nonstiff, fixed-coefficient BDF for stiff	Fortran
GEARB	1974	Banded Jacobian version ('banded jean')	Adams methods for nonstiff, BDF for stiff	Fortran
GEARS	1974	Jacobian with sparsity structure	Adams and BDF methods	Fortran
LSODE	1982	Combines features of GEAR and GEARB	Adams and BDF methods	Fortran
LSODES	1982	Sparse Jacobian extension of LSODE	Adams and BDF methods	Fortran
LSODPK	Not specified	Uses preconditioned Krylov iteration to solve linear systems	BDF, Preconditioned Krylov iterations	Fortran
VODE	1989	BDF with variable-coefficient and fixed-leading-coefficient forms	BDF (variable and fixed-coefficient forms)	Fortran
VODPK	1992	Uses preconditioned Krylov iteration for solving linear systems	Preconditioned Krylov iterative approach	Fortran
CVODE	Not specified	ANSI C implementation of VODE and VODPK	Newton, Functional iteration, Diagonal Approximation Jacobian	C
PVODE	1995	Parallel version of VODE using preconditioned Krylov iterations; SPGMR Newton iteration	SPGMR Newton iteration, Preconditioned Krylov, Functional iteration	C

2.1. SOLVER PACKAGE EPISODE

Automatic problem-solving with little human intervention is the goal of the EPISODE program, which is a collection of FORTRAN subroutines. The software uses a generalized backward differentiation formula (BDF) and a generalized Adams technique. The former is good for nonstiff situations, while the latter is good for stiff ones [11]. The implicit multistep procedure can be found in these two cases. The software heavily relies on the $N \times N$ matrix, where $i f$ and $j y$ are the vector components of f and y , respectively, to solve difficult problems using the Jacobian matrix.

$$j = \frac{\partial f}{\partial y} = \left(\frac{\partial f_i}{\partial y_j} \right), j=1$$

In [11], the use of EPISODE is thoroughly explained. However, fundamental parameter definitions are needed to provide the examples. As can be seen in examples 1 and possibly 2, the method flag, or MF, is the most important input parameter to EPISODE, even

more so than the issue specification itself. This could take any one of the following values: 10, 11, 12, 13, 20, 21, 22, or 23. Two fundamental approaches, implicit Adams and BDF, are shown by the first digit of MF, METH.

The second digit, MITER, denotes a method for iteratively solving the implicit equations created by the chosen formula. Each of the four possible values for the parameter MITER—0, 1, 2, and 3—indicates one of the following: (i) a Jacobian matrix is useless in functional (or fixed-point) iteration; (ii) a chord method, also known as a generalized Newton method or semi-stationary Newton iteration, where the user-supplied subroutine determines the Jacobian; (iii) an internal finite difference-generated Jacobian chord technique; (iv) an internal chord method that approximates the Jacobian with a diagonal approximation (which is less effective but requires less storage and calculation).

Calling the driver function EPSODE invokes the EPISODE package's other procedures, which work together to resolve the issue. A user-defined subroutine called DIFFUN is used to transmit the function f . In addition, the PEDERV subroutine, which handles the Jacobian, has to be constructed. EPSODE receives an additional name for each output point that is specified by the user. When the desired output value (TOUT) is achieved, control is returned to the caller program with the value of y at $t = \text{TOUT}$. The output is specified by passing a time value into the TOUT parameter of EPSODE. The INDEX input tells EPSODE whether this is the first call to the issue, which determines whether to initialize certain variables. It can also be used as an output parameter to indicate to the package whether or not it completed the task you gave it. How EPISODE handles errors is defined by two additional input parameters, EPS and IERROR [12].

The user's calling program and Subroutines DIFFUN and PEDERV should be coupled with the eight FORTRAN subroutines that make up the EPISODE package. As said before, the user only calls Subroutine EPSODE; the rest are called from within the package. In a nutshell, these are the eight package procedures' purposes: (a) when issues arise, EPSODE handles them, sets up storage, calls TSTEP, the core integrator, checks for error returns, and writes out error messages. (b) INTERP uses an array of multistep history data to calculate interpolated values of $y(t)$ at the output points that the user specifies. (c) TSTEP controls the local error for a single integration step, which includes choosing the step size and sequence. (d) the fundamental integration step and error control are both carried out by TSTEP using coefficients that are specified by COSET. (e) when the order is lowered, ADJUST modifies the history array. (f) where $I = -\beta \text{PSET}$ sets up the matrix J , h , I , p , and 0 is the identity matrix, h is the step size, 0β is a scalar related to the method, and J is the Jacobian matrix. It then processes P for subsequent solution of a linear algebraic system with P for subsequent solution of linear algebraic systems with P as coefficient matrix. (g) using the LU (lower-upper triangular) decomposition, DEC breaks down a $N \times N$ matrix. If the matrix was factored by DEC, then SOL can solve the linear algebraic system. (h) EPSODE is a subroutine that makes use of a variable coefficient-based backward differentiation algorithm. (i) using an Adams-Bashforth predictor and an Adams-Moulton corrector, the nonstiff alternative is implemented. (j) predictor: $y_{n+1} = y_n + h \sum_{i=1}^k \beta_i y'_{n+1-i}$ & corrector $y_{n+1} = y_n + h \sum_{i=1}^k \beta_i y'_{n+1-i}$. From one to seven, the sequence might change.

3. NUMERICAL IMPLEMENTATION

This example from chemical kinetics serves as an illustration of how the EPISODE package can be utilized to resolve difficult initial value issues. The correct FORTRAN code and output for each example issue that can be solved with EPISODE can be found here.

Example 1. The following Robertson case is frequently cited as an illustration of a kinetics issue. The three nonlinear rate equations that are involved are as follows:

$$y_1' = -.04y_1 + 10^4 y_2 y_3 \quad (8)$$

$$y_2' = .04y_1 - 10^4 y_2 y_3 - 3.10^7 y_2 \quad (9)$$

$$y_3' = 3.10^7 y_2 \quad (10)$$

At time zero, the starting values are

$$y_1(0)=1, y_2(0)=y_3(0)=0 \quad (11)$$

The answer must fulfill $\sum y_i = 1$ in the same way since $\sum y_i' = 0$. This may serve as an error check.

To resolve this issue, we intend to employ the user-supplied Jacobian (MITER=1), BDF, and chord of iteration approaches. Let's say there is a local error constraint of EPS of 10^6 and a control absolute error of 1 (IERROR=1). We select an H0 initial step size between 10 and 8 power units. The Jacobian J f/y must be computed and programmed before MITER=1 can be used. It is provided by

$$J = \begin{pmatrix} -.04 & 10^4 y_3 & 10^4 y_2 \\ .04 & -10^4 y_3 - 6.10^7 y_2 & -10^4 y_2 \\ 0 & 6.10^7 y_2 & 0 \end{pmatrix}$$

The true value of t is 40. At $t = 410k$, we might consider taking output if $k = -1, 0, 1, 2, \dots$. The main point of the argument will be based on these. You can use the following code in conjunction with the EPISODE package to resolve these issues with the options listed above. Here is the tabular output of the program:

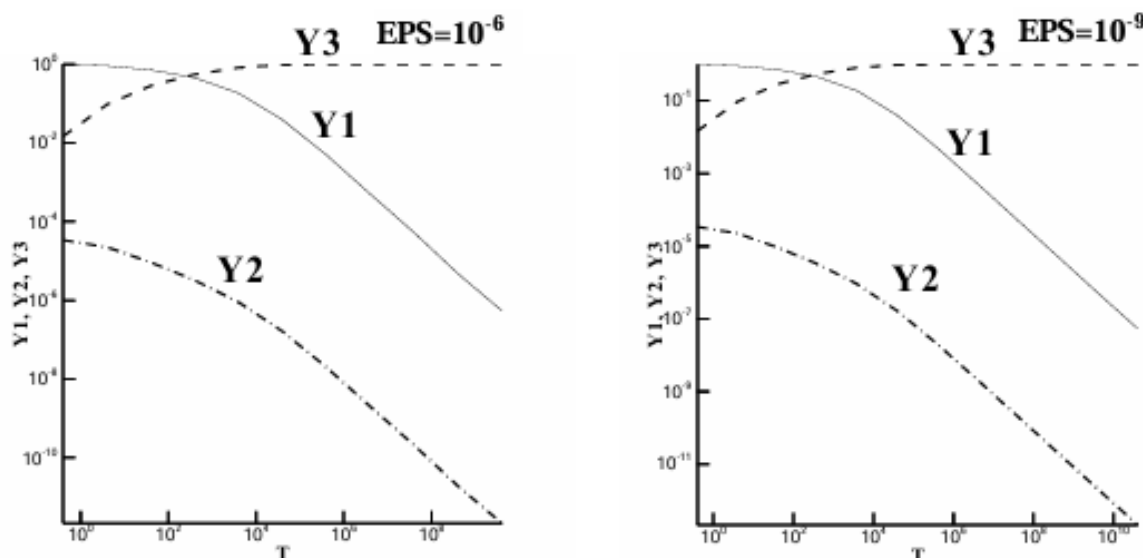
Table 2. MF=21, EPS= 10^{-6} .

T	H	y_1	y_2	y_3	SUM(y) - 1
0.4E+00	0.16E+00	0.98517E+00	0.33864E-04	0.14794E-01	-0.4E-15
0.4E+01	0.56E+00	0.90552E+00	0.22405E-04	0.94462E-01	-0.5E-15
0.4E+02	0.23E+01	0.71582E+00	0.91851E-05	0.28417E+00	-0.3E-16
0.4E+03	0.20E+02	0.45051E+00	0.32228E-05	0.54949E+00	-0.5E-15
0.4E+04	0.24E+03	0.18320E+00	0.89423E-06	0.81680E+00	-0.2E-15
0.4E+05	0.33E+04	0.38986E-01	0.16219E-06	0.96101E+00	0.3E-15
0.4E+07	0.38E+06	0.50319E-03	0.20660E-08	0.99948E+00	0.3E-15
0.4E+10	0.12E+10	0.54561E-06	0.37316E-11	0.10000E+01	-0.1E-13

The approach to equilibrium is rather gradual, and we can see that the equilibrium values are $y_1 < y_2 < 0$, and $y_3 < 1$. As can be seen above, the time step, H, rises gradually with time, T. We also note that the algorithm has been giving negative results that are physically incorrect for the past ten years. This indicates that the issue is unstable or very sensitive to numerical mistakes to the left, and if the integration is kept going, the results will diverge to $\pm\infty$. The following tabular output can be used to verify the accuracy of the previous result by rerunning the program with a lower value of EPS= 10^{-9} and no other changes:

Table 3. MF=21, EPS=10⁻⁹.

<i>T</i>	<i>H</i>	<i>y</i> ₁	<i>y</i> ₂	<i>y</i> ₃	SUM(<i>y</i>) - 1
0.4E+00	0.34E-01	0.985172E+00	0.338641E-04	0.147940E-01	0.2E-15
0.4E+01	0.14E+00	0.905519E+00	0.224048E-04	0.944589E-01	0.5E-15
0.4E+02	0.13E+01	0.715827E+00	0.918552E-05	0.284164E+00	0.6E-15
0.4E+03	0.82E+01	0.450519E+00	0.322290E-05	0.549478E+00	0.8E-15
0.4E+04	0.76E+02	0.183202E+00	0.894237E-06	0.816797E+00	0.1E-14
0.4E+05	0.88E+03	0.389834E-01	0.162177E-06	0.961016E+00	0.9E-15
0.4E+07	0.20E+06	0.516813E-03	0.206835E-08	0.999483E+00	0.1E-14
0.4E+10	0.67E+09	0.522363E-06	0.208942E-11	0.999999E+00	0.1E-14

**Figure 1. The plot of the approximate solution of Example 1 (on the logarithmic scale).**

Example 2. There's just one way to solve the system of initial value problems.

$$u'_1 = 9u_1 + 24u_2 + 5\cos t - \sin t, \quad u_1\left(\frac{1}{3}\right) = \frac{4}{3} \quad (12)$$

$$u'_2 = -24u_1 - 51u_2 - 95\cos t + \sin t, \quad u_2\left(\frac{1}{3}\right) = \frac{2}{3} \quad (13)$$

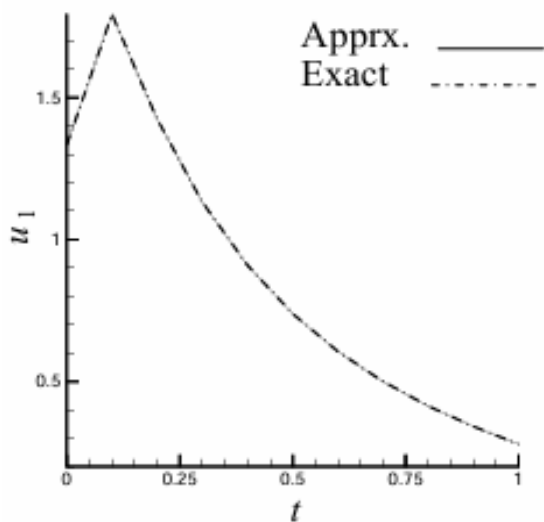
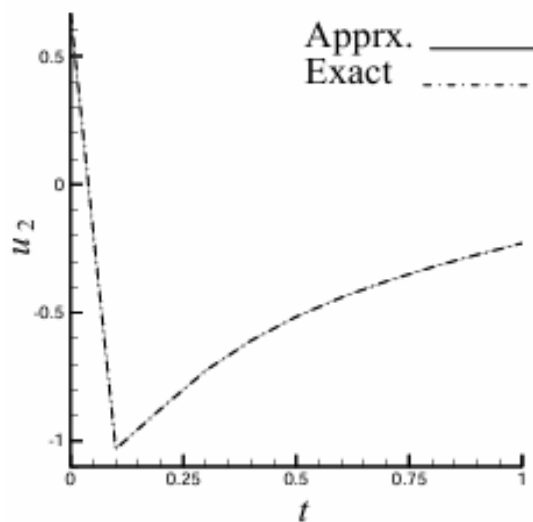
$$u_1(t) = 2e^{-3t} - e^{-39t} + \cos t \frac{1}{3} \quad (14)$$

$$u_2(t) = -e^{-3t} + 2e^{-39t} - \cos t \frac{1}{3} \quad (15)$$

Due to the solution's transient term e^{-39t} , this system fails. The findings shown in the following table are summarized from the EPISODE analysis.

Table 4. EPISODE analysis.

t	h	Approximated value of $u_1(t)$	Approximated value of $u_2(t)$	Exact value of $u_1(t)$	Exact value of $u_2(t)$
0.0	$0.40 \cdot 10^{-2}$	1.33333333333333	0.66666666666666	1.33333333333333	0.66666666666666
0.1	$0.40 \cdot 10^{-2}$	1.7930614656453	-1.03200020000000	1.79306300000000	-1.03200200000000
0.2	$0.91 \cdot 10^{-2}$	1.4239020595652	-0.8746803395763	1.42390200000000	-0.87468100000000
0.3	$0.12 \cdot 10^{-1}$	1.1315762404057	-0.72499799999999	1.13157700000000	-0.72499860000000
0.4	$0.33 \cdot 10^{-1}$	0.90940824040106	-0.60821345000000	0.90940860000000	-0.60821420000000
0.5	$0.33 \cdot 10^{-1}$	0.73878794444444	-0.51565752070532	0.73878780000000	-0.51565770000000
0.7	$0.66 \cdot 10^{-1}$	0.49986115	-0.37740429999999	0.49986030000000	-0.37740380000000
1.0	$0.66 \cdot 10^{-1}$	0.279680637563552	-0.2298906567848	0.27967490000000	-0.22988780000000

**Figure 2(a). The solution graph for u_1** **Figure 2(b). The solution graph for u_2**

It is clear from this that the graphs representing the precise answer and the estimated solution are in perfect agreement.

4. DISCUSSIONS

This research offers a thorough examination of the nature of stiff differential equations and the implicit numerical methods that have been devised to solve them, with a particular emphasis on the EPISODE solver package. This study confirms that rigidity continues to be a significant obstacle in the resolution of ordinary differential equations (ODEs) in a variety of scientific fields, as evidenced by both theoretical explanation and practical demonstration. The results indicate that implicit multistep methods, particularly backward differentiation formulas (BDFs), provide stability advantages over explicit methods when addressing rigid systems. Euler and explicit Runge-Kutta methods are traditional solvers that are effective in nonstiff contexts. However, their inefficiency for stiff problems is a result of their constrained step size, which is a result of stability constraints. The significance of BDF-based solvers is emphasized by the necessity of methods with broader stability regions. This study's primary solver, EPISODE, employs variable-coefficient BDFs to enhance stability and accuracy. The program's adaptability in accommodating a variety of solver configurations (via the MF flag) allows users to choose from a variety of iterative techniques and combinations of BDF or

Adams methods for Jacobian evaluation. The solver's strengths and limitations are emphasized in the comparative analysis of Example 1 (Robertson kinetics) and Example 2 (a solvable system with transients).

The capacity of EPISODE to adjust to system behaviour and reduce computational burden over time is demonstrated in Example 1, where the time step size (H) increases gradually as time (T) progresses. The Robertson problem's stiff character is also confirmed by the results, which require small time steps for accuracy due to significant disparities in reaction rates. Negative concentrations and numerical divergence were observed during extended simulation periods, despite EPISODE's capabilities.

These issues underscore the method's susceptibility to numerical instability and indicate that the Jacobian approximation and error tolerance parameters (EPS) have a substantial impact on the results. Revaluation with a more stringent error constraint ($\text{EPS} = 10^{-9}$) demonstrated enhanced precision and stability, further substantiating the importance of parameter tuning for effective application. EPISODE is both potent and complex, particularly for novice users, due to its reliance on proper parameter configuration and initialization. The method's robustness when applied to systems with known behaviour was further validated by the solver's exceptional agreement between numerical and analytical solutions in Example 2. EPISODE's ability to produce high-fidelity results in the presence of transient dynamics is attested to by the consistency of exact and computed values across the domain when configured appropriately.

Nevertheless, obstacles persist. The inversion of Jacobian matrices, in particular, remains a performance impediment due to the computational cost associated with solving large linear systems. Optimization in matrix handling is still required to ensure that the solver is scalable for high-dimensional systems, even though it employs LU decomposition and other sophisticated algebraic techniques.

Additionally, the trade-off between storage, computational time, and solution accuracy must be meticulously managed, despite the fact that EPISODE offers options for diagonal and semi-analytical Jacobian approximations. Future development should focus on the following areas: adaptive strategy selection, automatic stiffness detection, and integration with contemporary high-performance computing architectures.

5. CONCLUSIONS

This work aims to investigate the phenomena of stiff differential equations as well as general-purpose approaches to their solution. You can find effective codes to fix these problems, but in order to use them effectively, you need to understand how they work. There has been no assistance or advice to help a user pick an appropriate approach for his situation, despite the development of various ways and the suggestion of many more fundamental formulae for stiff equations. Our study focuses on EPISODE, a highly effective program. We go over what this code can do and show you some real-world applications where it shines.

Nevertheless, more information regarding the experimental package EPISODE is required. First of all, the program is still in its early stages and hasn't seen much usage; therefore, its place among other ordinary differential equation programs is unclear. Second, we've shown that the program spends more time than it should on the linear system of the method for certain issue types. This behavior is connected to the amount to which the matrix is used in issue solving, and it is in this area that the algorithm has to be more efficient.

In conclusion, the research substantiates the effectiveness of EPISODE in the solution of rigid ODEs, particularly in chemical kinetics and systems with a significant disparity in eigenvalue magnitudes. While the solver provides precision and flexibility, its computational demands and complexity necessitate user expertise and meticulous parameter calibration. The efficacy of EPISODE would be enhanced, and its applicability in research and industry would be expanded by addressing these limitations through interface enhancements and algorithmic optimizations.

Acknowledgment: IU/R&D/2025-MCN0003832, Office of Research and Development, Integral University, Lucknow.

REFERENCES

- [1] Hasan, A., Srivastava, R.B., Ahmad, N., An improved iterative method based on cubic spline functions for solving nonlinear equations, *Journal of Chemical, Biological and Physical Sciences*, **4**(1), 528, 2013.
- [2] Atkinson, K.E., *An introduction to Numerical Analysis*, Wiley, New York, 1989.
- [3] Burden, R.L., Faires, J.D., *Numerical Analysis*, PWS Publishers, U.S.A., 2002.
- [4] Byrne, G.D., Hindmarsh, A.C., Stiff ODE Solvers: A review of current and coming attractions, *Journal of Computational Physics*, **70**(1), 1, 1986. [https://doi.org/10.1016/0021-9991\(87\)90001-5](https://doi.org/10.1016/0021-9991(87)90001-5)
- [5] Byrne, G. D., Hindmarsh, A. C., A polyalgorithm for the numerical solution of ordinary differential equations, *ACM Transactions on Mathematical Software*, **1**, 71, 1975. <https://doi.org/10.1145/355626.355636>
- [6] Ansari, A., Ahmad, N., Deebea, F., Application of the direct computation method for solving a general Fredholm integro-differential equation, *Global and Stochastic Analysis*, **11**(1), 65, 2023. <https://doi.org/10.64837/GSA.11.1.6>
- [7] Enright, W.H., Hull, T.E., *Comparing Numerical Methods for the solution of Stiff Systems of ODEs Arising in Chemistry*, Academic Press, Orlando, pp. 45, 1976.
- [8] Gear, C.W., *The automatic integration of stiff ordinary differential equations*, Information Processing 68, A. J. H. Morrell, Ed., North Holland, Amsterdam, pp. 187, 1969.
- [9] Gear, C. W., Algorithm 407: DIFSUB for solution of ordinary differential equations, *Communications of the ACM*, **14**(3), 185, 1971. <https://doi.org/10.1145/362566.362573>
- [10] Gerald, F.C., Wheatley, O.P., *Applied Numerical Analysis*, Pearson Education, Inc., 1999.
- [11] Hindmarsh, A.C., Byrne, G.D., *Episode. An experimental package for the integration of systems of ordinary differential equations*, L.L.L. Report UCID-30112, 1975.
- [12] Hindmarsh, A. C., Gear, C. W., *Ordinary differential equation system solver*, L.L.L. Report UCID-30001, rev. 3, 1974.
- [13] Lapidus, L., Schiesser, W. E., *Numerical Methods for Differential Systems*, Academic Press, INC., London, 1999.
- [14] Nejad, L. A. M., A comparison of stiff ODE solvers for astrochemical kinetics problems, *Astrophysics and Space Science*, **299**(1), 1, 2005. <https://doi.org/10.1007/s10509-005-2100-z>
- [15] Shampine, L.F., Gear, C.W., A user's View of Solving Stiff Ordinary Differential Equations, *SIAM Review*, **21**(1), 1, 1979. <https://doi.org/10.1137/1021001>

- [16] Singh, B.M., Ahmad, N., Study of numerical solution of nonlinear integral equations by using Adomian decomposition method & He's polynomial, *Journal of Mathematical Control Science and Applications*, **6**(2), 93, 2020.
- [17] Ahmad, N., Singh, V.P., New derivative-free iterative methods for solving nonlinear equations using Halley's method, *Journal of Science and Arts*, 19(46), 83, 2019.