Stiff Integrator Problems

Tom Krumpolc and Vibhav Dabadghao
Department of Chemical Engineering

Table of Contents

1 Introduction **1**

2 Method of Solution **3**

2.1 Explicit Euler 3

2.2 Runge-Kutta Methods 4

2.3 Implicit Euler 5

2.4 Backward Difference Formulae (BDF) 6

2.5 Stiff System 7

2.6 DAE Systems 8

3 Results **9**

3.1 Stiff System 9

3.2 DAE Index 2 System 10

3.3 DAE Index 3 System 11

4 Conclusions and Future Directions **13**

References **14**

Appendix **15**

1 Introduction

Numerical methods are a family of solution techniques meant to approximate an Ordinary Differential Equation (ODE) or Differential-Algebraic Equation (DAE) through the use of derivative approximations. The ODE in its general form is given by

$$y'(t) = f(y(t), t), \quad y(0) = y_0 \tag{1}$$

A DAE involves both differential and algebraic equations. A general DAE is given by

$$y'(t) = f(y(t), z(t), t), \quad y(0) = y_0 \tag{2.1}$$
$$g(y, z, t) = 0 \tag{2.2}$$

Solution schemes can be broadly classified into explicit and implicit methods. When the approximation can be calculated in terms of previously known quantities, the method is explicit. The simplest explicit scheme is Euler's forward difference. Providing an initial guess, this algorithm will calculate the next point based on the previous function evaluation. Alternatively, if the approximation requires a system of equations to be solved simultaneously to obtain the necessary quantities, the method is implicit. A simple implicit scheme is Euler's backward difference. This algorithm requires solving a system of equations to solve for $f(y)$ at each step.

Each solution method comes with its own set of advantages and disadvantages which need to be taken into consideration when deciding which to employ depending on the model of interest. Explicit methods solve quicker since all of the information is know at each step, but prove to be less accurate and potentially unstable depending on the problem. Implicit methods require a larger computation time, but with greater accuracy and stability. This paper will investigate different explicit and implicit methods with a focus on a stiff system.

In chemical engineering, modeling a system that involves reaction kinetics is common. A challenge faced is that the kinetics that describe a series of reactions can often vary drastically, with the rate of one reaction reaching equilibrium much faster than another. The relative reaction rates can be calculated by solving for the eigenvalues of the system, with the limiting reactions having the fastest and slowest reaction rates. By taking the ratio of these eigenvalues, one can come up with a stiffness ratio for the system. The larger the stiffness ratio, or relation between the largest and smallest eigenvalue, the more difficult the system can be to solve numerically. In the first part of this paper, we will examine a system of equations that represent the concentration of a species across a series of reactors, with a differential equation representing the reaction occurring in each tank. The system is represented by the following set of equations:

$$\frac{dc_A}{dt} = -0.1c_A - 49.9c_B, \qquad c_A(0) = 3 \tag{3.1}$$
$$\frac{dc_B}{dt} = -50c_B, \qquad c_B(0) = 1.5 \tag{3.2}$$
$$\frac{dc_C}{dt} = 70c_B - 120c_C, \qquad c_C(0) = 3 \tag{3.3}$$

We will test a series of numerical integration methods and study their application on this problem relating to chemical reaction kinetics. In addition to Explicit and Implicit Euler, we will also introduce a class of implicit integration methods known as Linear Multi-step Methods (LMS) and investigate their applicability to stiff problems. We extend these methods to Differential-Algebraic Equations (DAEs), which are a more general class of problems. In particular, we will examine DAE

system of index 2:

$$\frac{dx}{dt} = -y \tag{4.1}$$

$$x = \sin(t) \tag{4.2}$$

and a DAE system of index 3 [2]:

$$\frac{dx}{dt} = y \tag{5.1}$$

$$\frac{dy}{dt} = z \tag{5.2}$$

$$x = \sin(t) \tag{5.3}$$

The definition of index as well as its importance and difficulties in numerical integration is explained in section 2.6 of this paper.

In the next section, we introduce commonly used numerical methods, provide an analysis and study their applicability to these problems.

2 Method of Solution

In numerical integration, we discretize the domain (usually time) and approximate the derivative at each step. Using this approximation, we replace the ODE with an algebraic equation that we solve by marching forward in time. As briefly discussed earlier, the simplest numerical integration scheme is Explicit Euler. Other commonly used schemes include Implicit Euler, Runge-Kutta and Linear Multistep methods. In this section, we describe these methods in detail and provide an analysis of their stability and convergence properties using the integrator problems discussed in Section 1.

2.1 Explicit Euler

In this method, we approximate the derivative in Equation (1) by a first order forward approximation. So, at time step i, the ODE is approximated as

$$y_{i+1} = y_i + \Delta t f(y_i, t_i) \tag{6}$$

Naturally, as we decrease the step size, we expect to observe more accurate results compared with the analytical solution. However, reducing the step size leads to an increase in computational load owning to an increase in the number of time steps. Often, for large-scale problems, there is a trade-off between accuracy and computational expense. Stability graphs are a helpful guide to selecting good steps sizes. These graphs represent the regions of stability in which the numerical method will show good convergence behavior. This region depends both on the numerical method and the nature of the problem itself. To compare the stability regions of numerical methods, we use a ODE

"test" problem:

$$y' = \lambda y, \quad y(0) = y_0 \tag{7}$$

The stability region for Explicit Euler is illustrated in Figure 1.

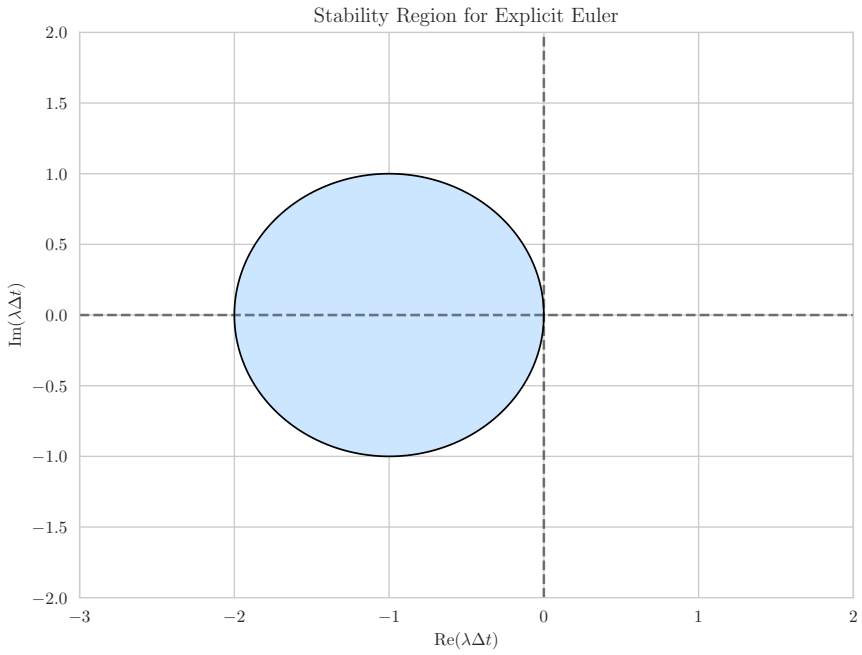

Figure 1: Stability region of Explicit Euler [5]

2.2 Runge-Kutta Methods

In Runge-Kutta methods, we take a linear combination of solutions within a single time step. Explicit Runge-Kutta methods take a weighted average over points in each step size. The most popular of these methods is Runge-Kutta of order 4, which is defined as follows:

$$y_{i+1} = y_i + \frac{1}{6}\Delta t(k_1 + 2k_2 + 2k_3 + k_4) \tag{8.1}$$

$$k_1 = f(y_i, t_i) \tag{8.2}$$

$$k_2 = f(y_i + \Delta t\frac{k_1}{2}, t_i + \frac{\Delta t}{2}) \tag{8.3}$$

$$k_3 = f(y_i + \Delta t\frac{k_2}{2}, t_i + \frac{\Delta t}{2}) \tag{8.4}$$

$$k_4 = f(y_i + \Delta t k_3, t_i + \Delta t) \tag{8.5}$$

This scheme divides the step size into four equally spaced regions. Then, each step k_i is calculated and used to solve for the following step, with more of the weight being placed on the slope around

the midpoint. The stability region for Runge-Kutta Order 4 is illustrated in Figure 2.

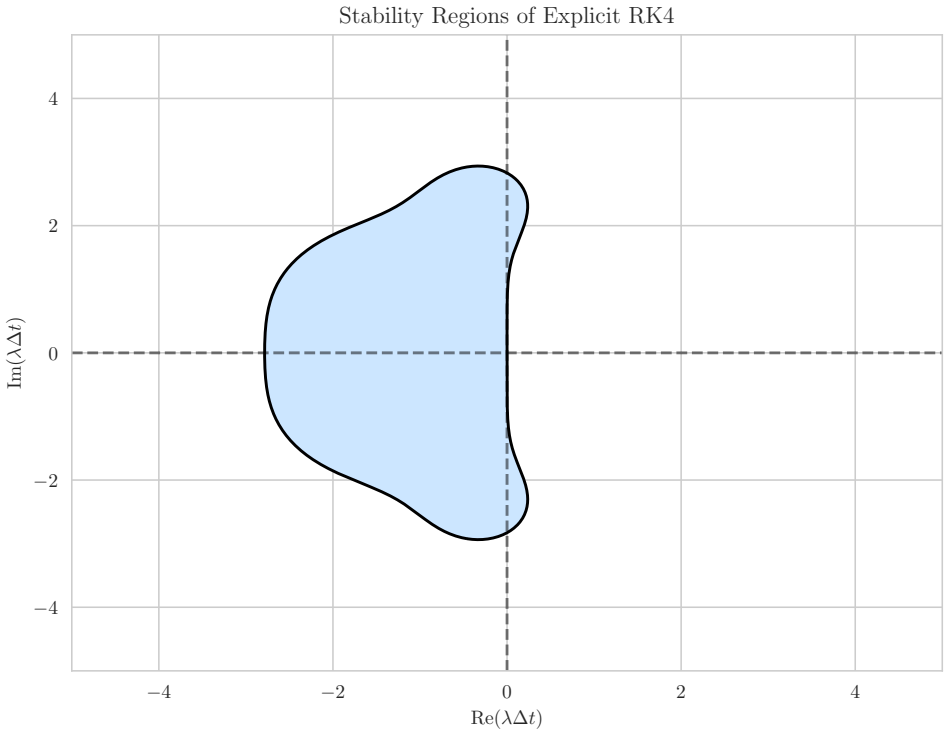

Figure 2: Stability Region Runge-Kutta Order 4

2.3 Implicit Euler

As in Explicit Euler, we approximate the derivative in Equation (1), but with a first order backward difference. So it follows that the ODE is approximated as

$$y_{i+1} = y_i + \Delta t f(y_{i+1}, t_{i+1}) \tag{9}$$

The Implicit Euler formulation is similar to that of Explicit Euler, but at every time step, a system of equations needs to be solved before calculating the next step. In return for the additional computation time to solve this system at every step, Implicit Euler schemes are stable over a much wider range of step sizes. Additionally, it is stable over the entire left-hand size of the real/imaginary plane, a property known as absolute stability. The stability region for Implicit Euler is illustrated in Figure 3.

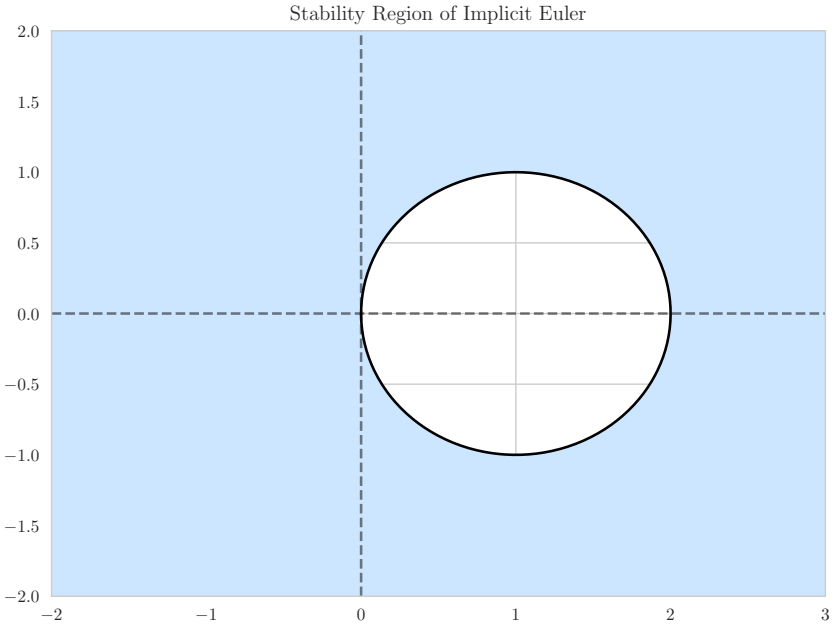

Figure 3: Stability region of Implicit Euler

2.4 Backward Difference Formulae (BDF)

BDF are a family of implicit linear multistep methods in which the derivative y' at time step i is approximated by a linear combinations of function values y at current and previous time steps [1]. The number of previous steps in the approximation decides the order of the method. For instance, Implicit Euler takes into account the y value at only one previous step. Hence, Implicit Euler is a BDF method of order 1. The BDF family takes the following form.

$$hy'_i = \sum_{j=0}^{k} \alpha_{kj} y_{i-j} \tag{10}$$

The coefficients α_{kj} for methods of various orders are given in Table 1.

Table 1: Coefficients of BDF methods

Order k	α_{k0}	α_{k1}	α_{k2}	α_{k3}	α_{k4}	α_{k5}	α_{k6}
1	1	-1					
2	3/2	-2	1/2				
3	11/6	-3	3/2	-1/3			
4	25/12	-4	3	-4/3	1/4		
5	137/60	-5	5	-10/3	5/4	-1/5	
6	49/20	-6	15/2	-20/3	15/4	-6/5	1/6

As mentioned eariler, BDF of order 1 (BDF1) corresponds to Implicit Euler. The stability regions of higher order BDF methods are similar to that of Implicit Euler, the boundaries of which are illustrated in Figure 4 [3]. The points lying outside the boundaries are stable. Additionally, it should be noted that for BDF methods of order k, another numerical scheme must first be used to generate the first m number of points, where $m = k - 1$ [4]. That is, for BDF of order 6, we employed Implicit Euler to calculate the first 5 steps before the higher order method could start to take advantage of the linear combination of the larger number of previous points.

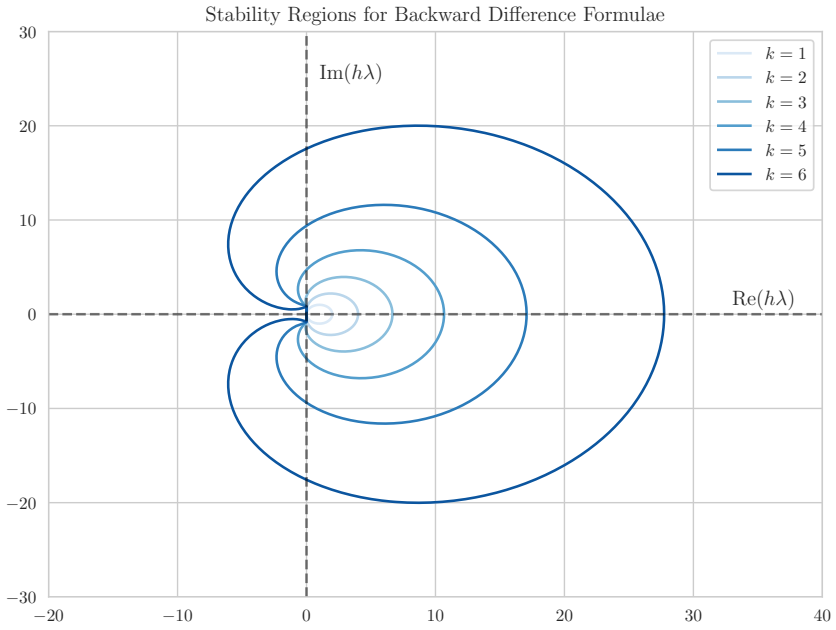

Figure 4: Boundaries of stability regions of k-order BDF methods [3]

2.5 Stiff System

We can represent the linear reaction system in Equation (3) as

$$\mathbf{c}' = A\mathbf{c}, \quad \mathbf{c}(0) = \mathbf{c}_0 \tag{11.1}$$

where,

$$A = \begin{bmatrix} -0.1 & -49.9 & 0 \\ 0 & -50 & 0 \\ 0 & 70 & -120 \end{bmatrix} \quad \text{and } \mathbf{c}(0) = \begin{bmatrix} 3 \\ 1.5 \\ 3 \end{bmatrix} \tag{11.2}$$

The resulting eigenvalues of the system are $\lambda_1 = -0.1$, $\lambda_2 = -50$, $\lambda_3 = -120$. As previously mentioned, the stiffness ratio of a system is defined by the ratio of the largest eigenvalue to that of

the smallest eigenvalue:

$$S = \frac{|\lambda_{max}|}{|\lambda_{min}|} = 1200 \tag{12}$$

Stiff systems are broadly defined as having a large stiffness ratio. As the ratio increases, the ability of numerical schemes to accurately and stably represent the numerical solution to the system becomes more difficult to guarantee.

2.6 DAE Systems

In the limiting case when the stiffness ratio $S \to \infty$, the fast dynamics of the stiff system converge to their steady state almost instantaneously.In this case, we can assume the time dependence ($\frac{dc}{dt} = 0$) and treat it as an algebraic equation. This results in a DAE system; however the numerical behavior of this system may be significantly different from the corresponding stiff ODE system. This behavior depends directly on the structure of the resulting DAE which is quantified by its so-called index.

When we use a numerical method, say BDF of order k, to solve the DAE system in Equation (2), then at step n, we can write the system as

$$\mathcal{F}(y_{n+1}, z_{n+1}) \coloneqq \begin{bmatrix} y_{n+1} - \left(\sum_{j=0}^{k} \alpha_j y_{n-j} + h f(y_{n+1}, z_{n+1}) \right) \\ g(y_{n+1}, z_{n+1}) \end{bmatrix} = 0$$

Now, we use a Newton-type strategy to solve this implicit equation for (y_{n+1}, z_{n+1}). The Jacobian of this system with respect to (y_{n+1}, z_{n+1}) is given by

$$J = \begin{bmatrix} I - h\dfrac{\partial f^\top}{\partial y} & -h\dfrac{\partial f^\top}{\partial z} \\ \dfrac{\partial g^\top}{\partial y} & \dfrac{\partial g^\top}{\partial z} \end{bmatrix}$$

If the step size $h \to 0$, we get

$$J = \begin{bmatrix} I & 0 \\ \dfrac{\partial g^\top}{\partial y} & \dfrac{\partial g^\top}{\partial z} \end{bmatrix}$$

Clearly, it is critical for the sub-matrix $\frac{\partial g^\top}{\partial z}$ to be non-singular for the implicit algorithm to converge. In other words, we need the jacobian of the algebraic part of Equation (2) to be non-singular. When it is singular, the DAE is said to be high-index.

We previously commented on the index of a system of DAEs. The index of a DAE system is index 1 when the Jacobian of the algebraic system, eq. (2.2), is non singular. Another way of thinking

about this is determining the number of times that the system of equations must be differentiated in order for all of the algebraic variables to appear in algebraic equations. For example, in eq. (4), we have algebraic variable x and differential variable y. However, in eq. (4.2), the variable y is not present in the algebraic equation, which makes the system high index. In order to reduce this to an Index 1 system, one must differentiate eq. (4.2) once, and substitute into eq. (4.1). The resulting system is presented below.

$$y = -\cos(t) \tag{13.1}$$

$$x = \sin(t) \tag{13.2}$$

It can now be seen that in the system above, x and y can be calculated explicitly at a point in time. A similar solution process can be followed for the Index 3 system, eq. (5). The index of a system plays an important role in numerical integration. If the index of the system is 2 or higher, then the system is highly sensitive to initial conditions. Differentiating the DAE system to reduce its index increases the degrees of freedom of the resulting system by introducing constants of integration. That is, for the differentiated eq. (2.2), there is only one value for the initial condition that will satisfy the original algebraic equation and the new differential equation. So if the initial condition for the higher index system does not satisfy the differentiated system, the resulting numerical solution can become unstable. Consequences of this can be seen in the results section.

3 Results

3.1 Stiff System

For the stiff system of equations, eq. (3), we tested several different numerical schemes with varying step sizes to determine which would be able to stably represent the solution. Additionally, we wanted to compare computational solve times to see if the more robust methods would be necessary, which are reported in table 2. Because of the various number of numerical methods tested, the graphs have been placed in section 4. It is important to note that because of the fast dynamics of two of the species B and C, there are two time scales so that the accuracy of each method can be better visualized. Species B and C reach equilibrium almost instantaneously compared to A. The first observation is that both explicit schemes, (Explicit Euler and Runge-Kutta 4), required finer discretization before the solutions were stable as seen in Figures 10, 13 and 14. Furthermore, both implicit methods, (Implicit Euler and BDF2), were stable across all ranges of Δt values, Figures 11, 12, 15 and 16. Solution times highlighted in red in table 2 are those which were numerically unstable.

Table 2: Solution times for the stiff system using different methods

Algorithm	Solution times (s)				
	$\Delta t = 1$	$\Delta t = 10^{-1}$	$\Delta t = 10^{-2}$	$\Delta t = 10^{-3}$	$\Delta t = 10^{-4}$
Explicit Euler	0.001	0.004	0.033	0.351	3.612
Runge-Kutta (RK4)	0.002	0.016	0.142	1.541	16.627
Implicit Euler	0.004	0.037	0.410	4.093	49.249
BDF2	0.005	0.054	0.634	6.309	68.556

3.2 DAE Index 2 System

For the DAE system Equation (4), we tested two different numerical methods, Implicit Runge-Kutta Order 2 (Midpoint Euler) and Implicit Euler (BDF1) with different sets of initial conditions. The first set is the correct set of initial conditions, the second and third have small perturbations in x or y. Both numerical schemes solved the solution correctly when provided with the correct set of initial conditions, the top graph on Figure 5 and Figure 6. For Midpoint Euler, the numerical solution started to become unstable when the initial condition on y was perturbed slightly, and became very unstable when x was perturbed. Implicit Euler was resilient through each of the perturbations, and can also be seen to recover from the inconsistent initial conditions on the bottom graph of Figure 6.

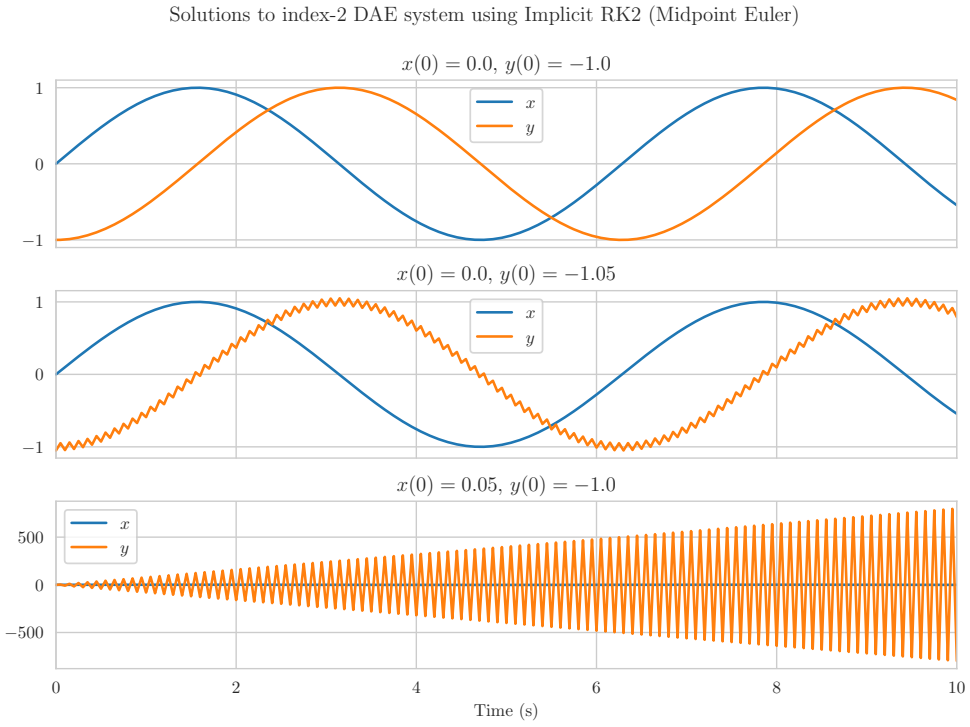

Figure 5: DAE Index 2 Method: Midpoint Euler

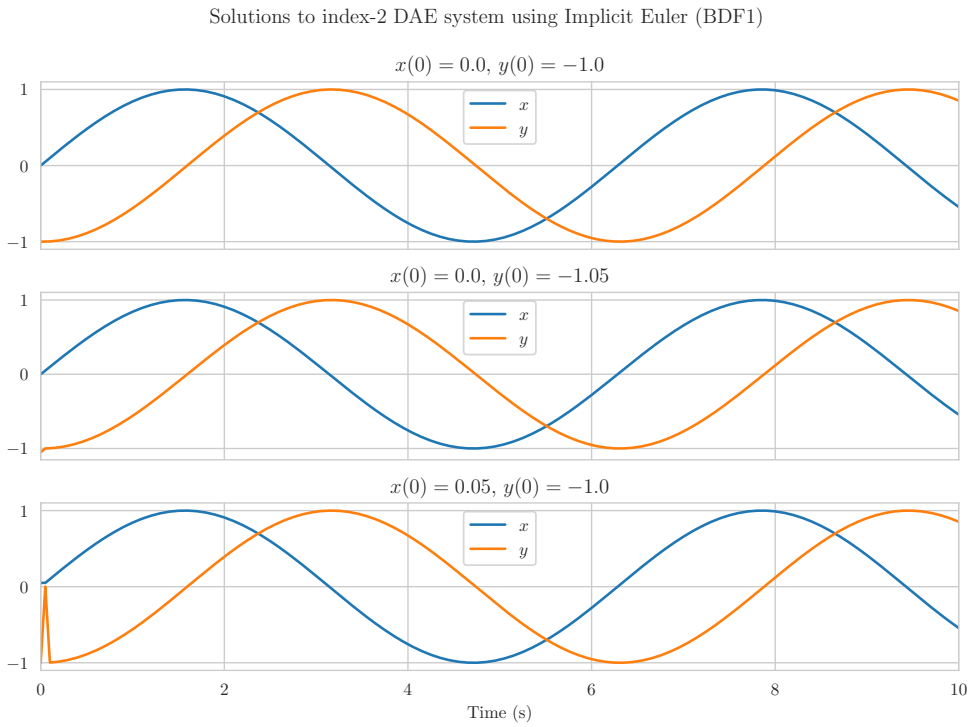

Figure 6: DAE Index 2 Method: Implicit Euler

3.3 DAE Index 3 System

For the DAE system in Equation (5), we tested three different numerical methods: Implicit Runge-Kutta Order 2 (Midpoint Euler), Implicit Euler (BDF1), and BDF2 with different initial conditions. The first set is the correct set of initial conditions, the second and third have small perturbations in x or y. Midpoint Euler was unstable even for consistent initial conditions. This may be attributed to floating point precision and illustrates the importance of having a consistent set of initial conditions to obtain the correct solutions. Implicit Euler was stable across all perturbations in x or y. To our surprise, BDF2 became unstable to perturbations in x. More investigation would need to be done on this problem to determine the exact cause of this behavior. However, it should also be noted that we are solving linear implicit DAE systems where we are able to reformulate them into an explicit equation at each step. If the Equation (5) were to be solved implicitly at each step, all of the above algorithms would fail as described in Section 2.6.

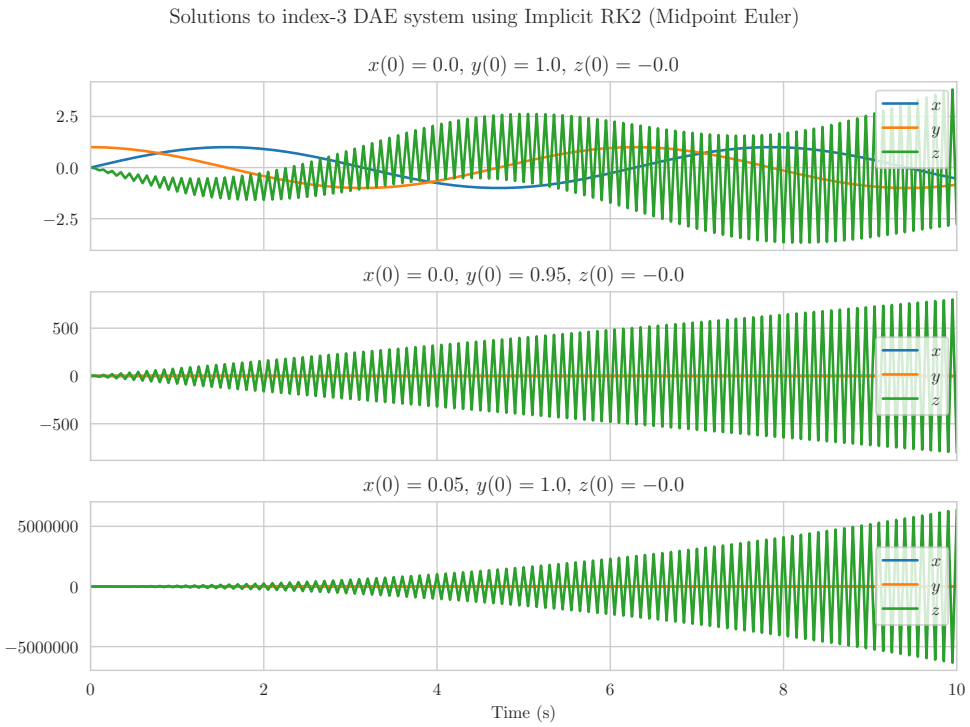

Figure 7: DAE Index 3 Method: Midpoint Euler

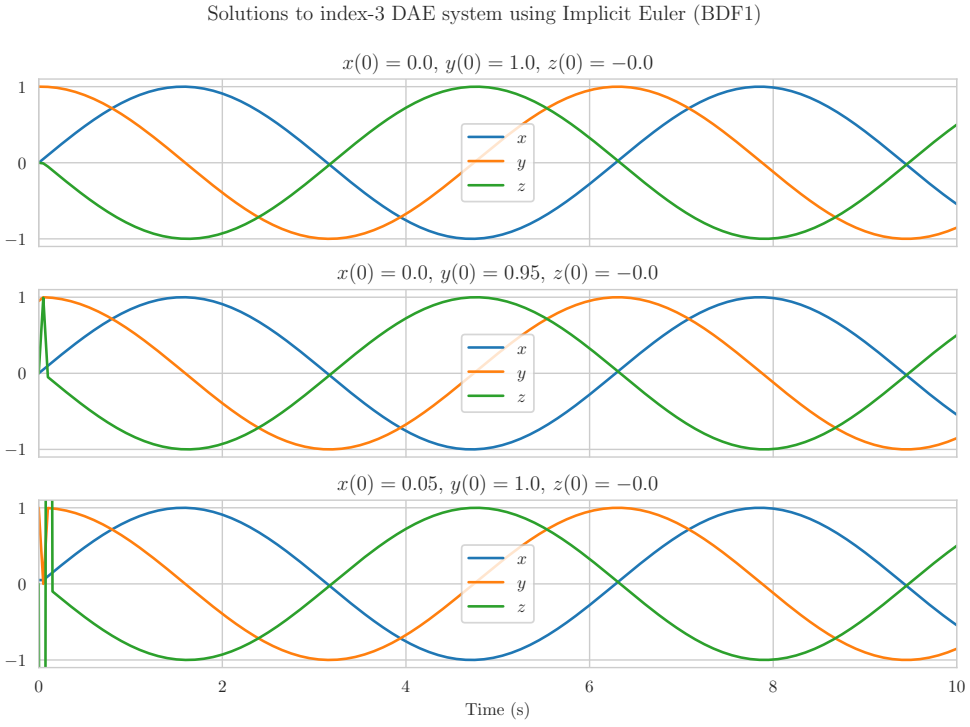

Figure 8: DAE Index 3 Method: Implicit Euler (BDF1)

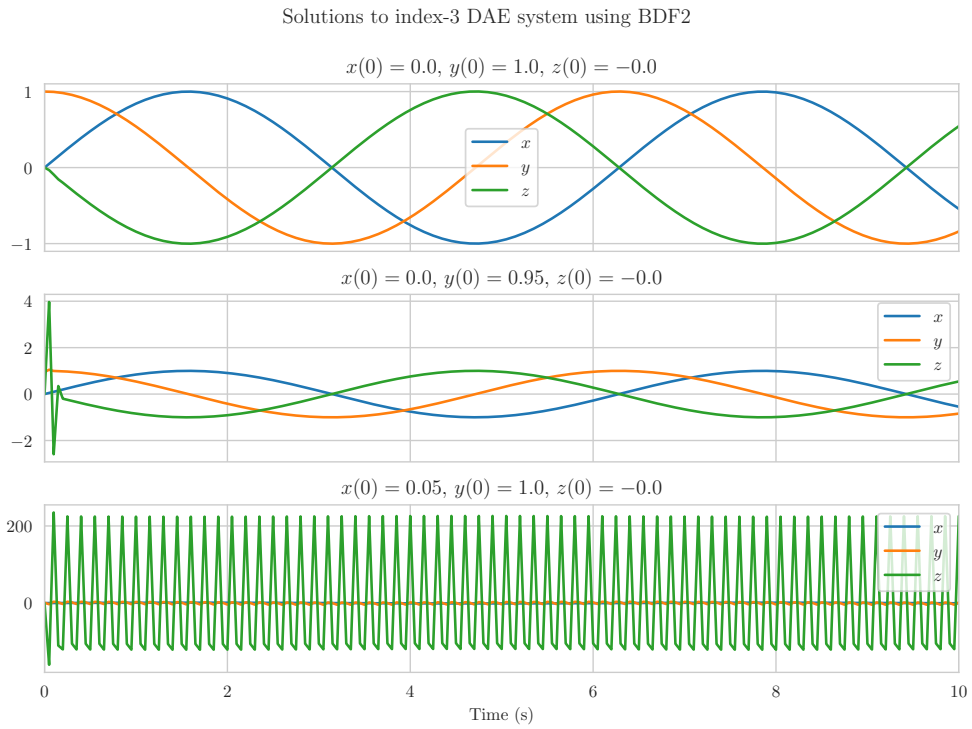

Figure 9: DAE Index 3 Method: BDF2

4 Conclusions and Future Directions

In this paper, we analyzed various numerical integration methods and applied them to stiff ODE systems as well as high-index DAE problems. We demonstrate that BDF methods work better than other methods such as Explicit Euler and the Runge-Kutta family of methods. However, they are computationally much more expensive due to a large number of calculations at each step. We also showed that, for high-index DAE systems, implicit (BDF) methods are able to solve fairly accurately. Further, these systems are highly sensitive to initial conditions, and even small perturbations can cause large fluctuations in the solution. We want to extend this analysis of BDF methods to nonlinear high-index DAE problems. Specifically, since at each step, we would need to solve an implicit equation using a Newton-type algorithm, it would be beneficial to analyze the the effect of changing step size on the accuracy and numerically stability (conditioning of the Jacobian) of the system.

References

[1] C Gear. Simultaneous numerical solution of differential-algebraic equations. *IEEE transactions on circuit theory*, 18(1):89–95, 1971. (cited on page 6)

[2] Sven Erik Mattsson and Gustaf Söderlind. Index reduction in differential-algebraic equations using dummy derivatives. *SIAM Journal on Scientific Computing*, 14(3):677–692, 1993. (cited on page 3)

[3] Jitse Niesen. Stability regions of backward difference formulae. https://en.wikipedia.org/wiki/Backward_differentiation_formula, 2012. (cited on page 7)

[4] Linda Petzold. Differential/algebraic equations are not ode's. *SIAM Journal on Scientific and Statistical Computing*, 3(3):367–384, 1982. (cited on page 7)

[5] Helmut Podhaisky. Stability regions for euler method. https://en.wikipedia.org/wiki/Euler_method, 2010. (cited on page 4)

Appendix

This section includes solution profiles for the stiff system in Equation (3) using different numerical methods. The plots with log-scaled time axis illustrates the stiffness of the system. We can see that one profile converges by about three orders of magnitude slower than the fast dynamics.

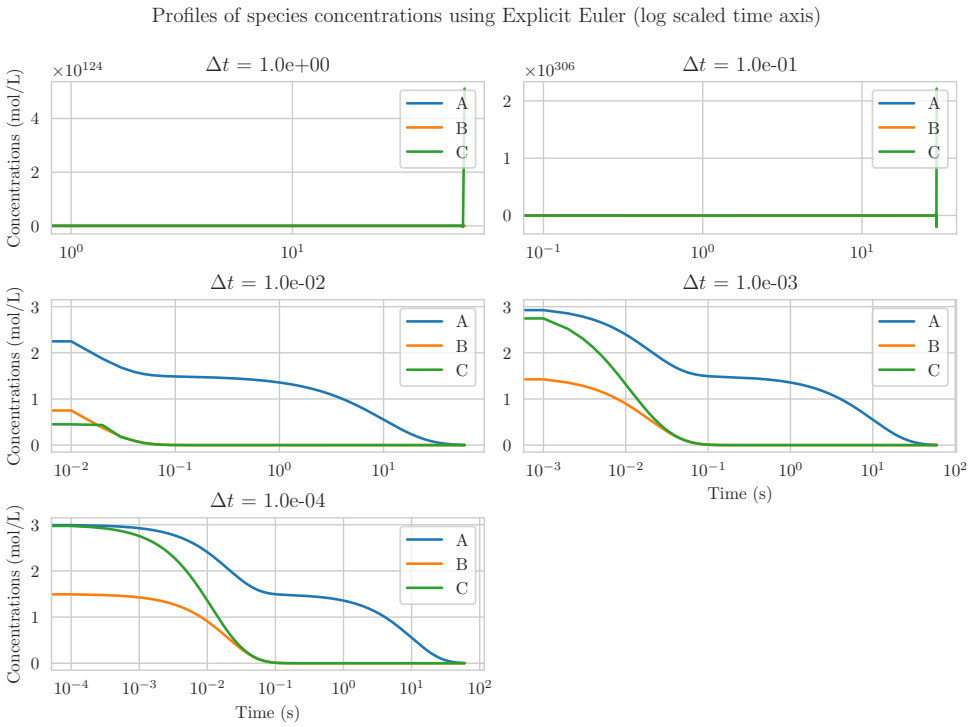

Figure 10: Stiff System solution Plots for Explicit Euler Across Range of Δt Values

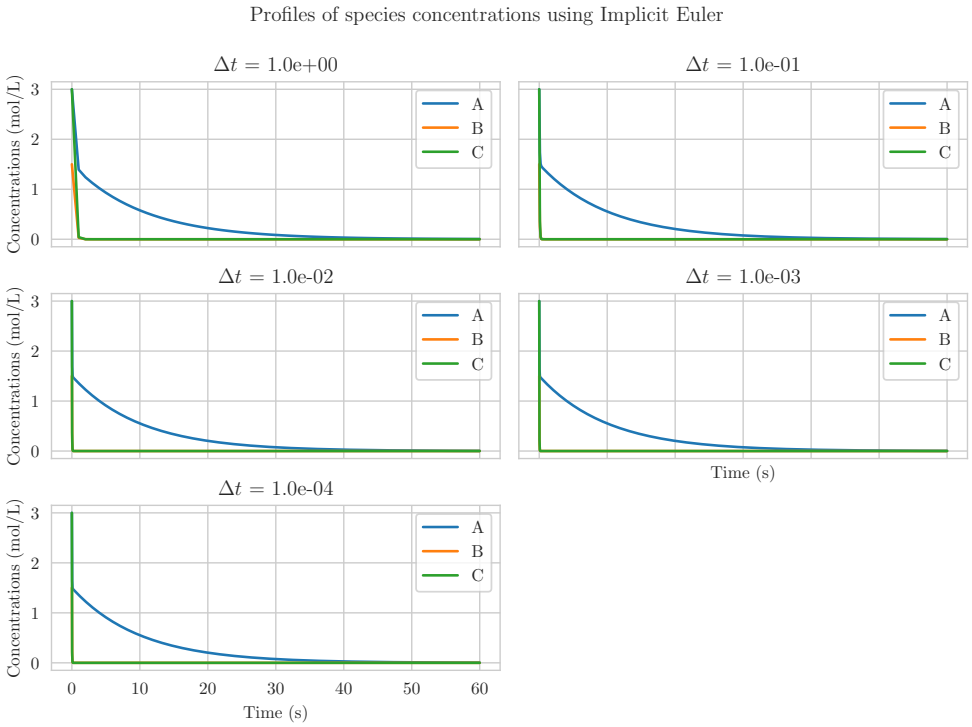

Figure 11: Stiff System solution Plots for Implicit Euler Across Range of Δt Values (log scaled)

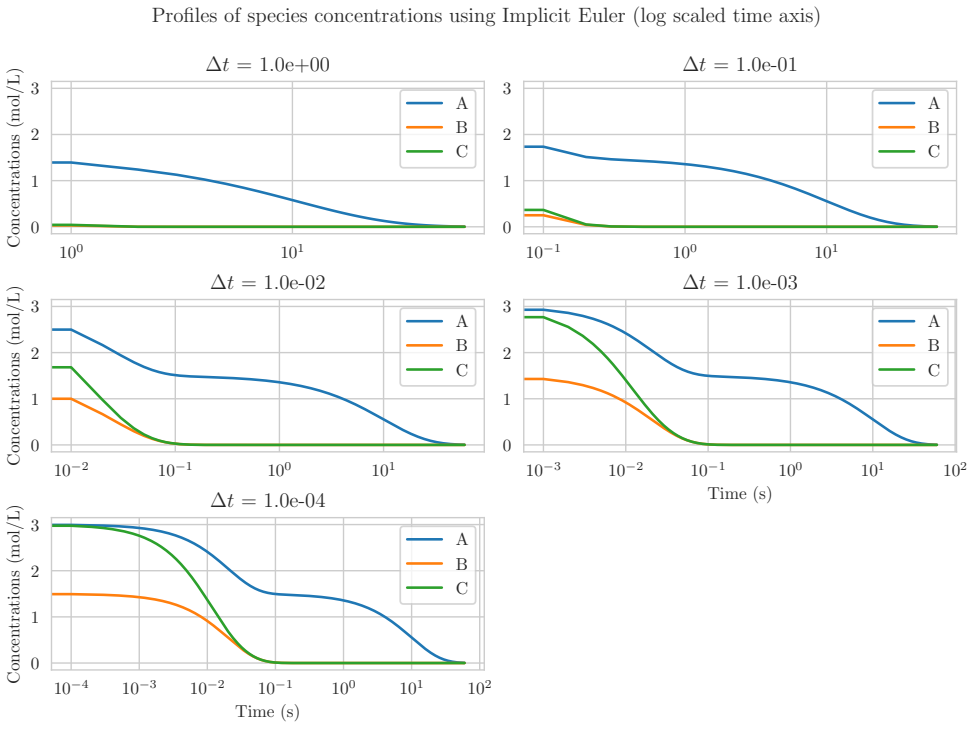

Figure 12: Stiff System solution Plots for Implicit Euler Across Range of Δt Values (log scaled)

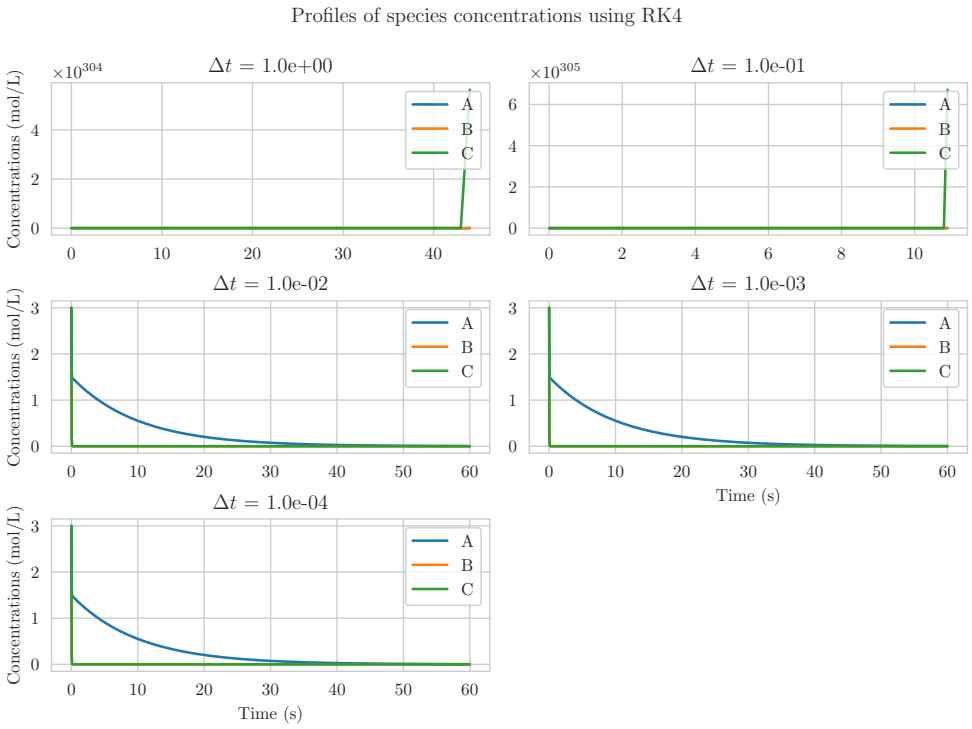

Figure 13: Stiff System solution Plots for Runge-Kutta Order 4 Across Range of Δt Values

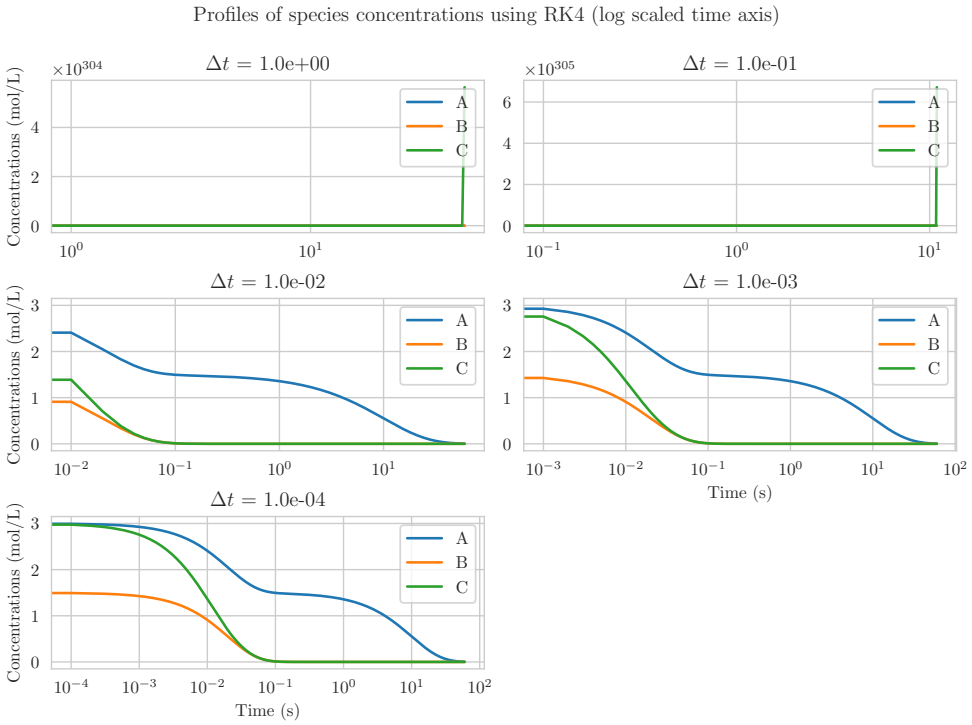

Figure 14: Stiff System solution Plots for Runge-Kutta Order 4 Across Range of Δt Values (log scaled)

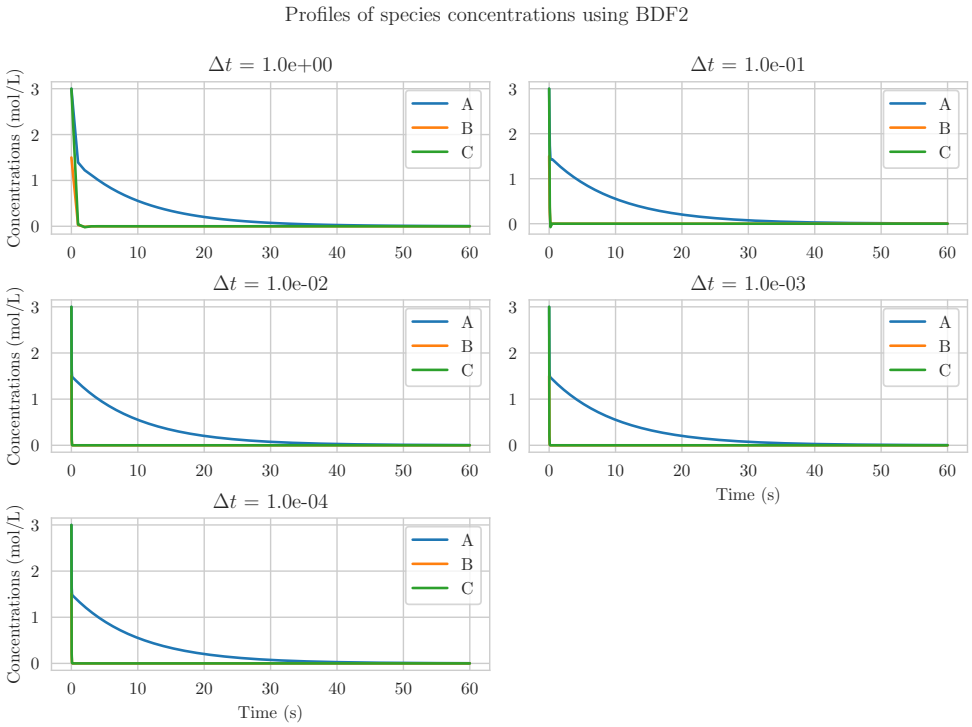

Figure 15: Stiff System solution Plots for BDF Order 2 Across Range of Δt Values

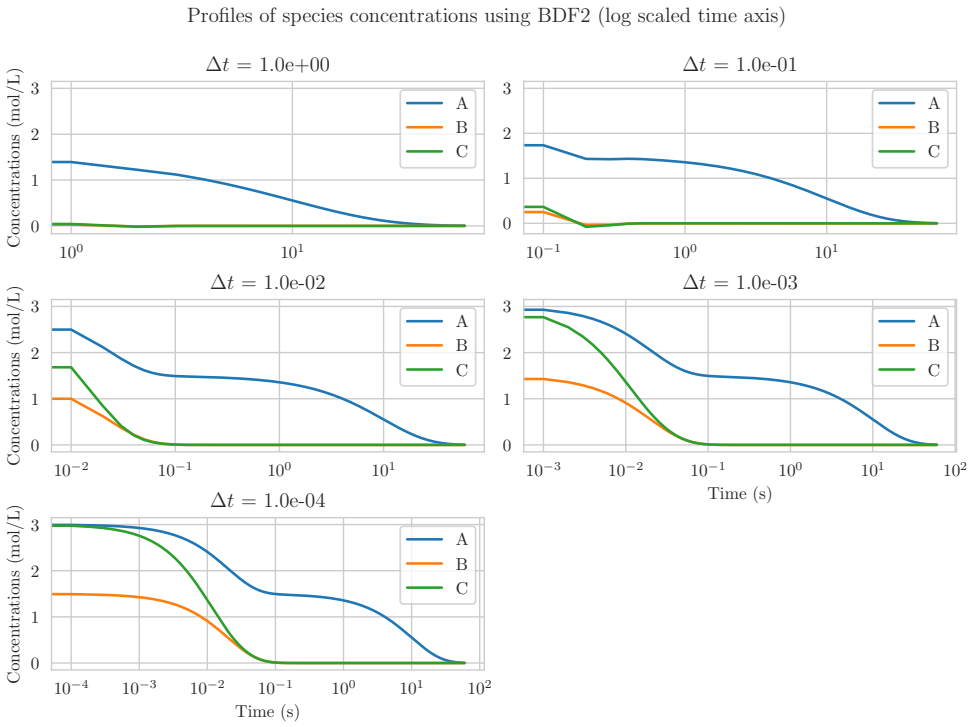

Figure 16: Stiff System solution Plots for BDF Order 2 Across Range of Δt Values (log scaled)