Middle East Technical University

Department of Electrical and Electronics Engineering

Final Project Report

Control System Design and Simulation

Part 2: Simulation

Student Name: Aylar Roshan Nahad

Course: EE498 – SPECIAL TOPICS: CONTROL SYSTEM DESIGN AND SIMULATION

June 22, 2025

Contents

1 Overview and Introduction **2**

2 System's Differential Equations **2**

3 Model Parameters **2**

4 Part a: Simulink Solvers **3**

4.1 Variations of the Maximum Step Size 3

5 Part b: Implementing Solvers **10**

5.1 Fixed Step Size Implementation and Simulation 10

5.2 Variable Step Size Implementation and Simulation 11

6 Conclusion **16**

1 Overview and Introduction

In this part of the project, we perform a simulation study of eutrophication in aquatic ecosystems using a fifth-order compartmental model. It consists of nitrogen concentration $n(t)$, algal biomass $a(t)$, zooplankton biomass $Z(t)$, detritus $S(t)$, and dissolved oxygen $C(t)$.

2 System's Differential Equations

The model is based on the formulation in [1]:

$$\begin{aligned}
\frac{dn(t)}{dt} &= q + \pi_0 \delta S(t) - \alpha_0 n(t) - \beta_1 n(t) a(t) \\
\frac{da(t)}{dt} &= \theta_1 \beta_1 n(t) a(t) - \alpha_1 a(t) - \beta_{10} a^2(t) - \beta_2 a(t) Z(t) \\
\frac{dZ(t)}{dt} &= \theta_2 \beta_2 a(t) Z(t) - \alpha_2 Z(t) - \beta_{20} Z^2(t) \\
\frac{dS(t)}{dt} &= \pi_1 \alpha_1 a(t) + \pi_2 \alpha_2 Z(t) - \delta S(t) \\
\frac{dC(t)}{dt} &= q_c - \alpha_3 C(t) - \lambda_{11} a(t) - \delta_1 S(t)
\end{aligned}$$

Initial conditions: $n(0) = a(0) = Z(0) = S(0) = C(0) = 0.1$

3 Model Parameters

The model is realized using the following numerical values:

$$\begin{aligned}
&q = 5.0, \quad \pi_0 = 0.1, \quad \delta = 1.0, \quad a_0 = 0.5, \quad \beta_1 = 1.0, \\
&\theta_1 = 1.0, \quad \alpha_1 = 0.5, \quad \beta_{10} = 2.0, \quad \beta_2 = 1.0, \\
&\theta_2 = 1.0, \quad \alpha_2 = 0.5, \quad \beta_{20} = 2.0, \quad \pi_1 = 0.9, \quad \pi_2 = 0.9, \\
&q_c = 10.0, \quad \alpha_3 = 2.0, \quad \lambda_{11} = 0.25, \quad \delta_1 = 10.0.
\end{aligned}$$

We initialize states as:

$$n(0) = 0.1, \quad a(0) = 0.1, \quad Z(0) = 0.1, \quad S(0) = 0.1, \quad C(0) = 0.1$$

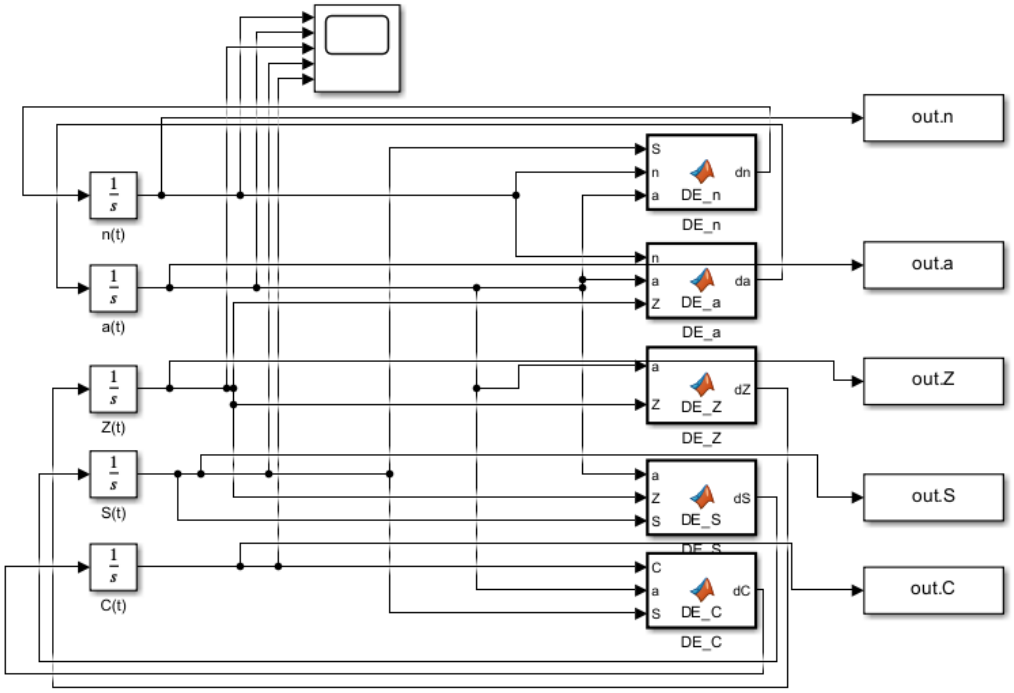
From the details explored in the paper, it is clear that the model is not stiff. Therefore, using an explicit runge-kutta method such as the ode45 is sufficient. For the rest of this report, the reference solution is taken with the following settings:

Table 1: Reference Solution's Configuration Parameters

Maximum Step Size	Relative Tolerance	Solver
auto	1e-5	ode45 (variable step size)

4 Part a: Simulink Solvers

For this part, we use Simulink's built-in solvers.



Block diagram realization of the differential equations.

4.1 Variations of the Maximum Step Size

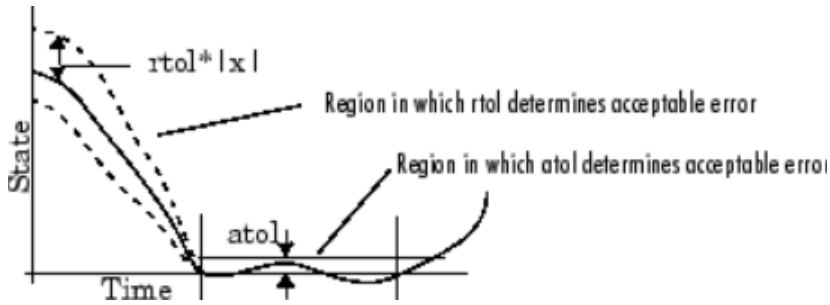
In this section, we focus on the effect of varying the following three parameters: maximum step size, relative tolerance, and absolute tolerance.

First, we briefly define how each parameter affects our simulation settings [2]:

- **Absolute tolerance:** The user can define a global limit on the local error which applies to all states via this parameter.
- **Relative tolerance:** The solvers require the error for the local error for the i^{th} state e_i to satisfy:

$$e_i \leq \max(\text{rtol} \times |x_i|, \text{atol}_i)$$

- **Maximum step size:** By default, it is set to auto, which indicates that the solver determines the maximum step size to obey constraints such as:
  - $\leq$ the smallest discrete sample time in the model
  - $\leq$ one-third of the period of the highest specified frequency in the model
  - ≤ 0.2 if the stop time is Inf
  - Defined such that the simulation takes at least 50 time steps
  - Defined such that the stop time is an integer multiple of the step size



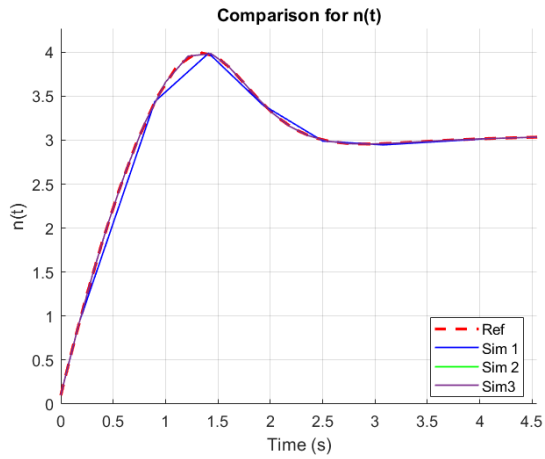
Relative and absolute tolerance margins. Picture from [2].

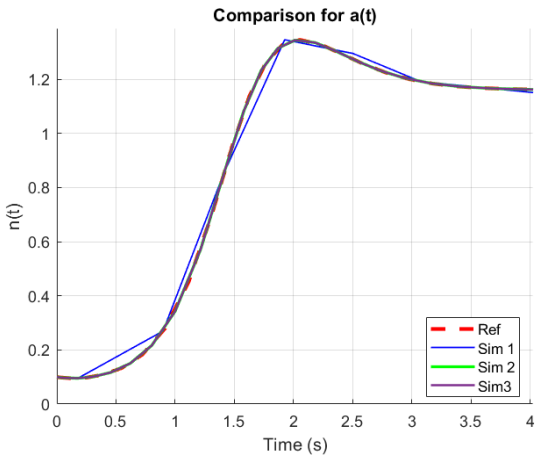
To understand how modifying these parameters affects our proposed numerical solution, three different parameter sets are selected as follows:

Table 2: Reference Solution's Configuration Parameters

	Maximum Step Size	**Relative Tolerance**	**Absolute Tolerance**
Sim 1	1	0.1	auto
Sim 2	1	0.1	0.1
Sim 3	5	1e-5	auto

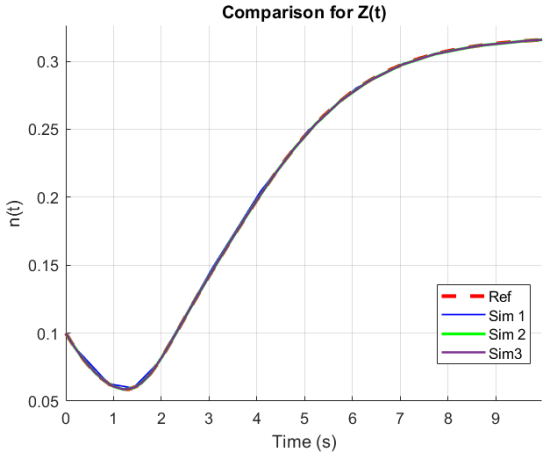
The progress plots for all states are shown below:

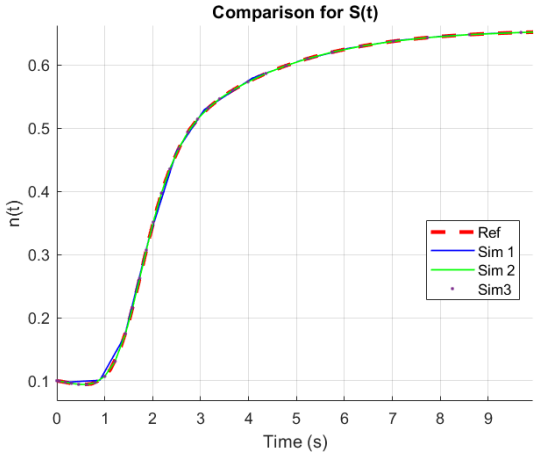




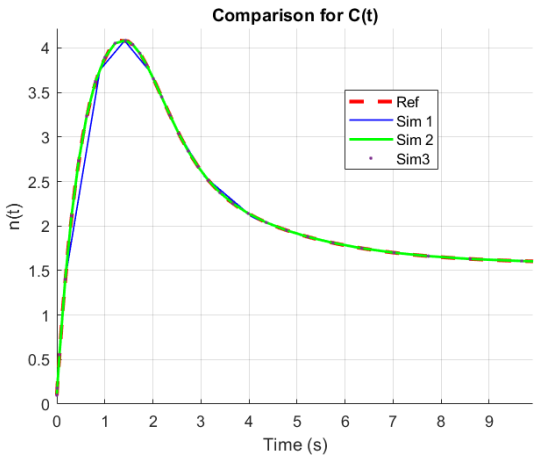
Comparison of $n(t)$ and $a(t)$ for different solver settings.

The following figures show how different solver settings affect other system states.





Comparison of $Z(t)$ and $S(t)$ for different solver settings.

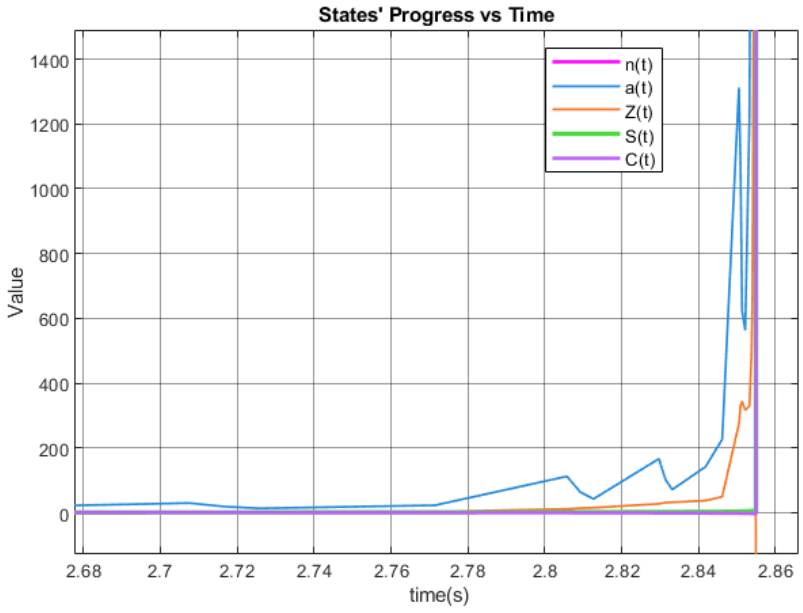


Comparison of $C(t)$ for different solver settings.

Additionally, since `ode45` is an explicit Runge-Kutta method, it is possible for the solver's solution to diverge or become unstable. For the (5, 1, 1) configuration parameter set, the following warning was issued:

> **Derivative of state '1' at time 2.8616381763344192 is not finite. The simulation will be stopped.**

The simulation results up to that point are shown below:



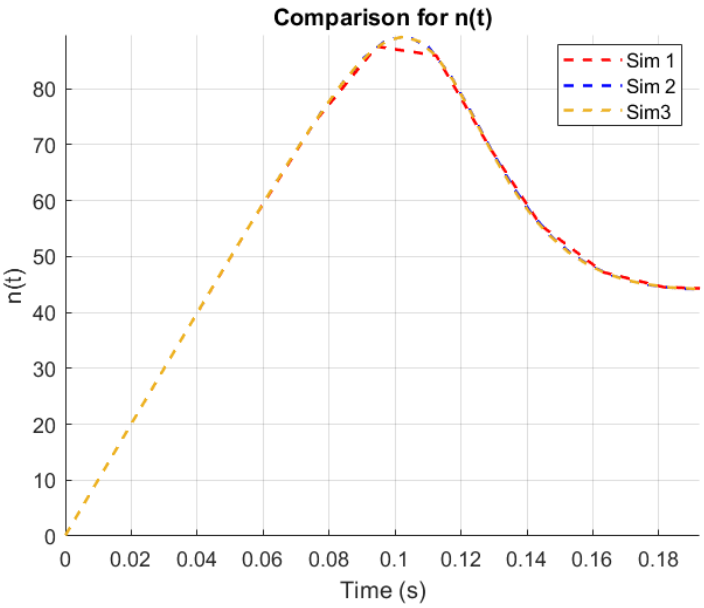
Simulation results until error point.

Observations and Comments

- As explored in lectures, there is a direct relation between step size and the accuracy of the solution. When the step size is larger, the solution tends to have higher local/global error. Additionally, explicit solvers can become unstable for high step sizes. We observed that increasing the maximum step size and tolerances can lead to solver divergence.
- Simulation results for Sim 3, where the relative tolerance matches the reference, are nearly identical to the reference. This shows that even though the maximum step size was increased, the low relative tolerance enforced smaller step sizes.
- The solution from Sim 1 is less accurate than Sim 2, as expected from how relative and absolute tolerances are defined.
- Not all states are equally affected by changes in solver parameters. Depending on the system dynamics, the error can vary. For example, for $n(t)$, the Sim 1 curve is clearly different from the reference, while for $Z(t)$, the difference is minimal.
- Errors are more pronounced when the signal slope is higher. Thus, systems with fast dynamics require more accurate solvers and stricter tolerance values. Variable step size methods adjust for this by reducing the step size during rapid transitions.

Exploring Different Solvers' Response

Given that our current model's state responses are relatively smooth, we redefine the model's parameter $q = 50$ and $\alpha_0 = 0.05$ to make $n(t)$ dynamics faster, generating a pronounced sharp peak around 0.1s. This enhances the comparison of solver responses.



Comparison of $n(t)$ for different solvers.

Sim 1, Sim 2, and Sim 3 use `ode45`, `ode113`, and `ode15s` with default settings, respectively.

`ode45` and `ode113` are explicit solvers, while `ode15s` is an implicit solver, often used for stiff systems. Around the peak, Sim 3 (using `ode15s`) appears the most stable and accurate. All methods converge to the same steady-state solution after the system's fast dynamics decay.

Solver Selection Heuristics

The following image provides a guideline for selecting solvers based on model characteristics.

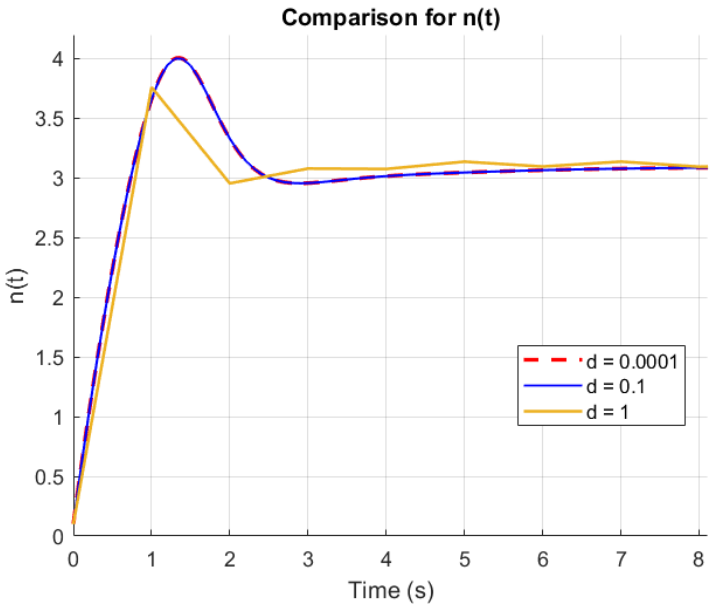
Auto Solver Heuristics

The solver type is \ The system has	Only discrete states	Continuous states	
		The system is an ordinary differential equation (ODE)	The system has differential algebraic equations (DAE)
Fixed-Step	FixedStepDiscrete	ode3	ode14x
Variable-Step	VariableStepDiscrete	The model is stiff: ode15s; The model is non-stiff: ode45	ode23t

Auto solver heuristics. Picture from [2].

Variations of the Step Size

Using MATLAB's `ode3` solver, we plot $n(t)$ versus time for different step size values:



Comparison of $n(t)$ for different step sizes.

For $d = 1$, the error becomes more visible. Additionally, with larger step sizes, the simulation may

become unstable, highlighting the need for step size tuning.

5 Part b: Implementing Solvers

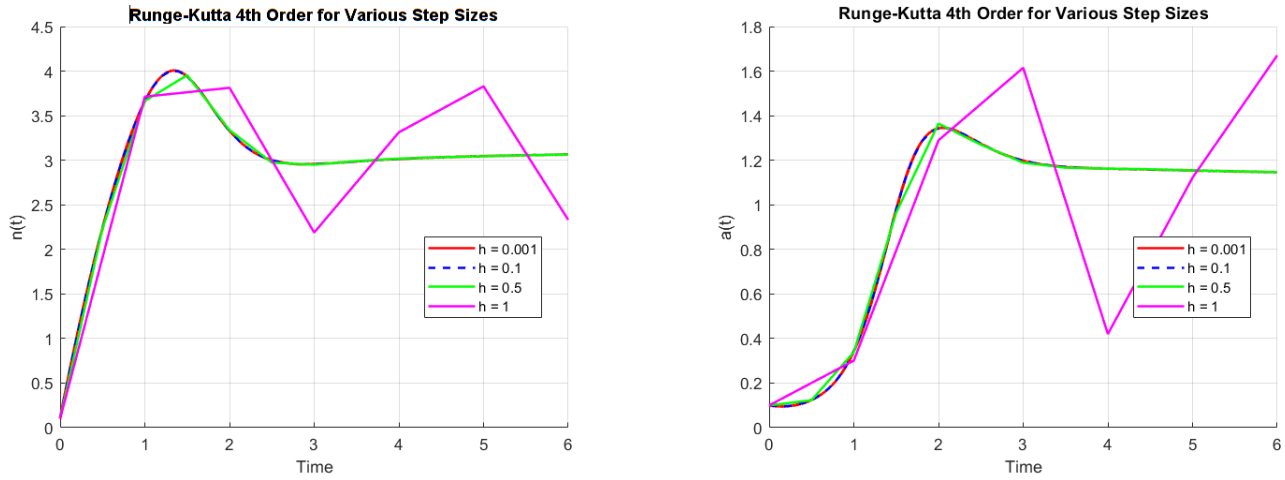
For this part, we implement the Runge-Kutta method introduced in the lecture notes, using the following Butcher Tableau:

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ \frac{1}{2} & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array}$$

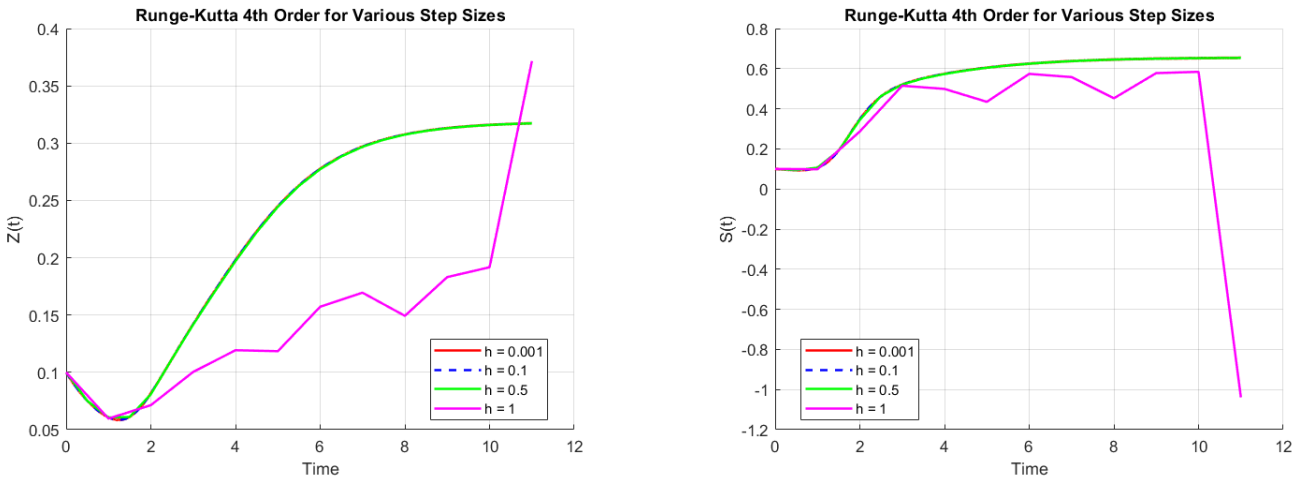
Additionally, we extend this method by introducing a step-size adaptation mechanism and compare fixed vs variable step performance.

5.1 Fixed Step Size Implementation and Simulation

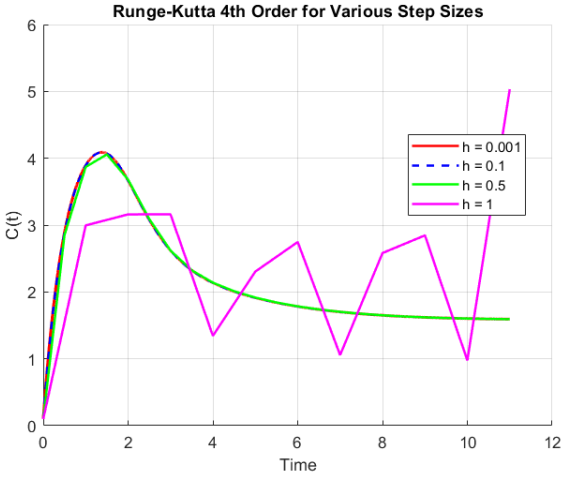
On the same model, we implement the method with four different step sizes.



Comparison of $n(t)$ and $a(t)$ for different step sizes.



Comparison of $Z(t)$ and $S(t)$ for different step sizes.



Comparison of $C(t)$ for different step sizes.

Clearly, for $h = 1$, the simulation becomes unstable. Therefore, smaller values are required for convergence. While $h = 0.5$ still results in a noticeable error, $h = 0.1$ and $h = 0.001$ yield nearly identical results. Thus, for efficiency, $h = 0.1$ is preferred.

An important observation is that due to the nonlinear behavior and different smoothness characteristics of the state variables, some states are more sensitive to inaccuracies when the step size increases. For instance, the instability in $S(t)$ and $Z(t)$ takes longer to manifest for $h = 1$ than in other states.

5.2 Variable Step Size Implementation and Simulation

We adopt a similar approach to the embedded Runge-Kutta methods discussed in [3]. However, the classical RK4 does not have a standard embedded pair, so we modify the algorithm accordingly. It

is important to note that:

This method is not computationally efficient.

In practice, methods such as Dormand-Prince (used in `ode45`) are preferable.

First, we define an error estimator. Any single-step third-order method can be used. For any method, the local error can be approximated as:

$$|\eta^{k+1}| = |x_i - \hat{x}_i| \approx |\hat{C}h^3| > \epsilon$$

This estimate is used to update the step size dynamically, based on a user-defined tolerance (ϵ). In our implementation, we use **Heun's third-order method** with the following Butcher Tableau [4]:

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{3} & \frac{1}{3} & & \\ \frac{2}{3} & 0 & \frac{2}{3} & \\ \hline & \frac{1}{4} & 0 & \frac{3}{4} \end{array}$$

In the next section, the general structure of the modified variable step-size algorithm is outlined.

Variable Step Size Algorithm Implementation

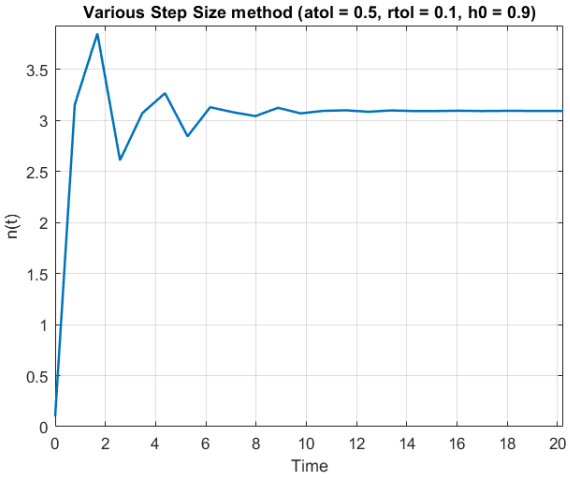
The following algorithm outlines the step-by-step process for implementing a variable step size Runge-Kutta method.

Algorithm 1 Variable Step Size Implementation for RK4

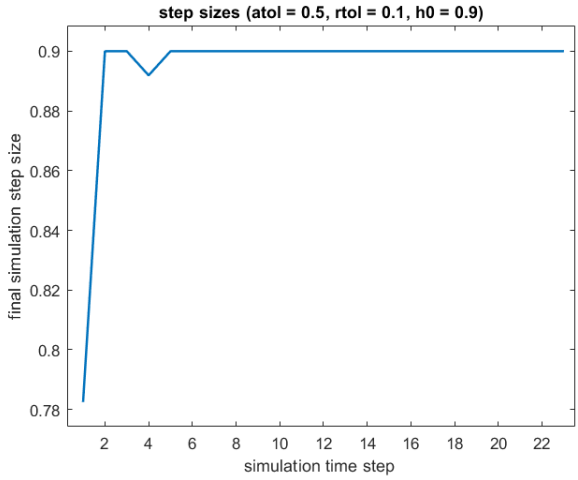
Initialize step size $h \leftarrow h_0$ (e.g., $h_0 = 0.2$)
while $E > 1$ **do**
    Compute candidate solution $x_i^{(4)}$ using RK4
    Compute $x_i^{(3)}$ using a third-order method for error estimation
    Compute element-wise error: $\text{err} \leftarrow |x_i^{(4)} - x_i^{(3)}|$
    Compute tolerance: $\text{tol} \leftarrow \text{atol} + \text{rtol} \cdot |x_i^{(4)}|$
    Compute normalized error: $E \leftarrow \frac{1}{N} \left\| \frac{\text{err}}{\text{tol}} \right\|_2$
    if $E \leq 1$ **then**
        Accept $x_i^{(4)}$ for current step size h
        break
    else
        Update step size: $h \leftarrow h \cdot \min\left(5, \max\left(0.1, \left(\frac{1}{E}\right)^{1/4}\right)\right)$
    end if
end while

We apply this algorithm to our model with two different parameter sets:

Case 1: `rtol = 0.5, atol = 0.1,` h_0 `= 0.9`

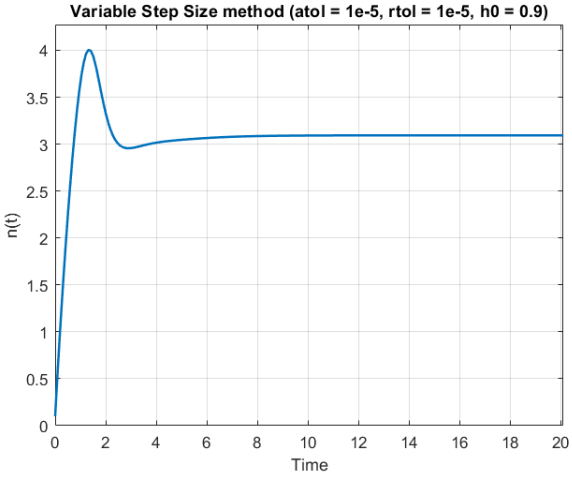


$n(t)$ vs time.

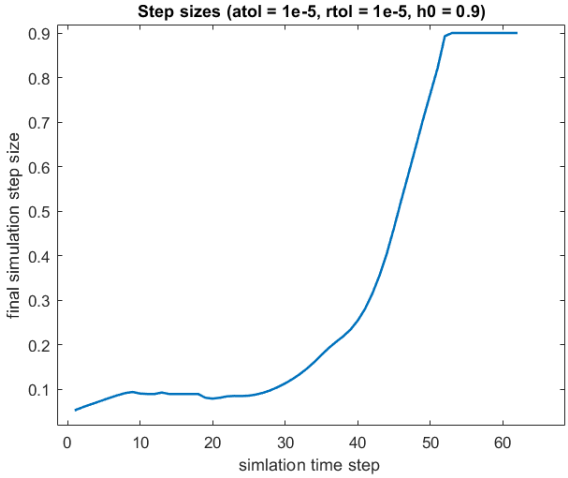


Step size vs time.

Case 2: `rtol = 1e-5, atol = 1e-5,` h_0 `= 0.9`



$n(t)$ vs time.



Step size vs time.

Comments and Observations:

- These three user-defined parameters (atol, rtol, h_0) enable flexible control over the trade-off between accuracy and computational efficiency.
- As expected, smaller step sizes are needed during fast dynamics (large slopes), while larger step sizes suffice during slow dynamics.

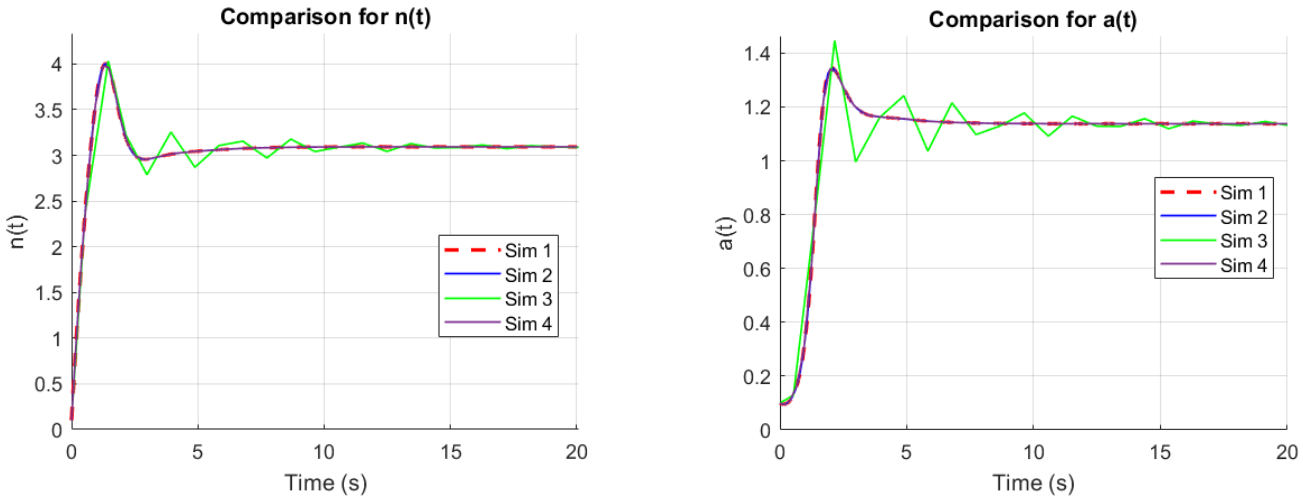
- The total number of simulation steps is directly tied to the tolerance settings. The second case, while more accurate, is computationally more expensive.

Efficiency and Comparison

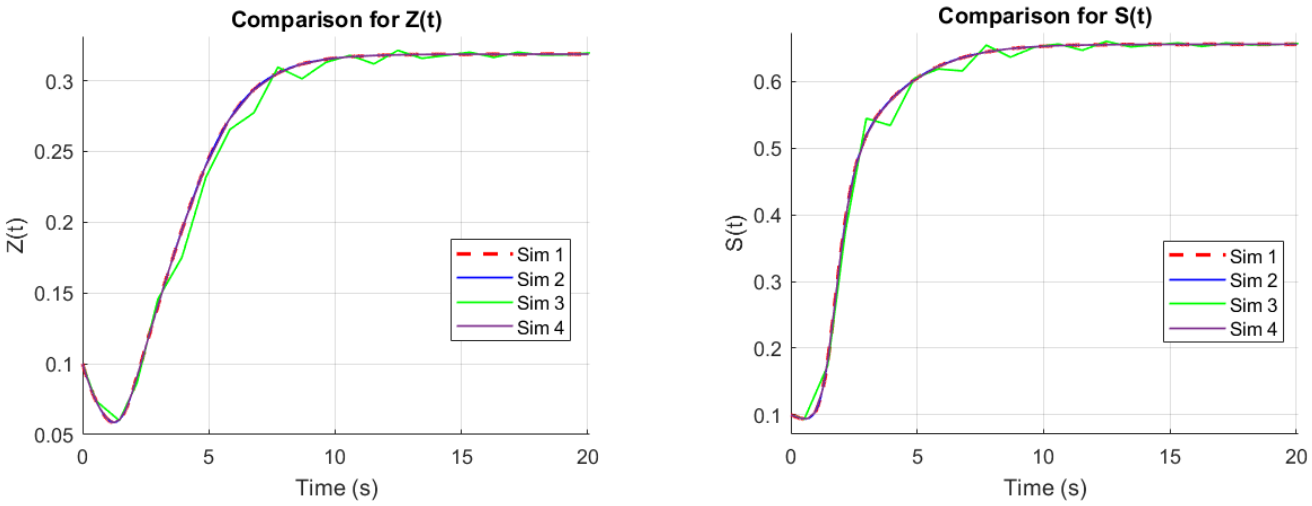
Variable step size methods are generally more efficient than fixed step size methods when system dynamics vary. Although our custom variable step size method is not the most efficient (due to the overhead of computing two RK solutions), it still offers better performance for slowly changing systems.

Table 3: Computation Time Comparison: Fixed vs Variable Step RK4 Solvers

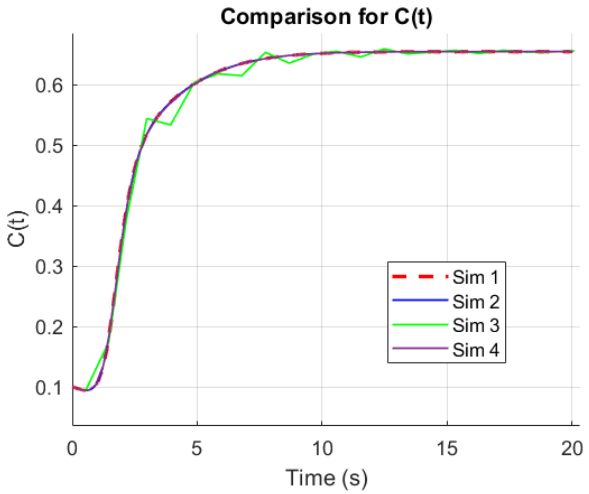
	Type	**Parameters**	**Computation Time**
Sim 1	Fixed	$h = 1\text{e}{-3}$	0.9205 s
Sim 2	Fixed	$h = 0.1$	0.0491 s
Sim 3	Variable	atol = 1e-1, rtol = 1e-1, $h = 0.95$	0.0475 s
Sim 4	Variable	atol = 1e-3, rtol = 1e-3, $h = 0.95$	0.0563 s



Comparison of $n(t)$ and $a(t)$ for different solver settings.



Comparison of $Z(t)$ and $S(t)$ for different solver settings.



Comparison of $C(t)$ for different solver settings.

Final Comments and Observations:

- While Sim 3 is the most efficient, it is also the least accurate. This illustrates the trade-off between speed and precision.
- Sim 1 (with very small fixed step size) takes significantly longer to run than others, with no clear benefit over Sim 2 or Sim 4.
- Even though our variable step method requires computing two RK steps at each iteration, it is still more efficient overall. Thus, when equidistant time steps are not required, variable step methods are preferred.

6 Conclusion

Realizing differential equations through numerical simulations is a widely-used and essential tool in both research and industry. However, choosing solver settings that meet performance goals for stability, accuracy, and efficiency is crucial.

In this report, we:

- Analyzed the effect of Simulink solver configurations.
- Implemented a custom fourth-order Runge-Kutta method.
- Extended the method with variable step size control.
- Compared fixed vs variable step performance based on accuracy and runtime.

The results show that step size adaptation significantly enhances simulation efficiency, especially when handling systems with non-uniform dynamics.

References

[1] Misra, A.K.K., 2007. Mathematical modeling and analysis of eutrophication of water bodies caused by nutrients. Nonlinear Analysis: Modelling and Control, 12(4), pp.511-524.

[2] MathWorks, *MATLAB and Simulink Documentation*, Available at: `https://www.mathworks.com/help/index.html?s_tid=CRUX_lftnav` [Accessed: June 21, 2025].

[3] Griffiths, D.F. and Higham, D.J., 2010. *Numerical Methods for Ordinary Differential Equations: Initial Value Problems*. Springer Undergraduate Mathematics Series. Available at: `https://link.springer.com/book/10.1007/978-0-85729-147-9` [Accessed: June 21, 2025].

[4] Wikipedia contributors. Runge–Kutta methods. Wikipedia, The Free Encyclopedia. Available at: `https://en.wikipedia.org/wiki` [Accessed: June 21, 2025].