QUEEN'S UNIVERSITY BELFAST, SCHOOL OF MATHS & PHYSICS – MTH3023

MTH3023 Project: Choice 3 Solving Differential Equations with Runge-Kutta

Your Name (Student ID: 40415474)

February 18, 2026

1 Introduction

The spread of infectious diseases in a closed population can be modelled using an ordinary differential equation. In this project we focus on the logistic infection model, where the rate of change of the infected population is proportional to both the number of infected and the number of susceptible individuals. This leads to the nonlinear ODE

$$y'(t) = k\,[N - y(t)]\,y(t),$$

where $y(t)$ is the number of infected individuals at time t, N is the total population, and k is the infection rate.

The aim of the project is to analyse this model using both analytical and numerical techniques. We first obtain the exact solution by separating variables, and then implement the fourth–order Runge Kutta (RK4) method to compute a numerical approximation.Using both approaches, we compute infected individuals after 30 days for a specified population and infection rate. We then examine the convergence of RK4 by comparing the numerical results with the exact solution for a range of step sizes, confirming the expected $O(h^4)$ accuracy.

Section 2 develops the analytical solution and introduces the RK4 scheme. Section 3 reports the numerical findings, Section 4 provides a convergence analysis, and Section 5 explores possible extensions such as the Runge Kutta Fehlberg method.

2 Theory and methods

In this section we explore two ways of solving the logistic equation. We begin by deriving the exact analytical solution, which helps us understand the essential behaviour of the model. We then describe the RK4 numerical method, which allows the solution to be approximated efficiently on a computer. Using both approaches shows how the model behaves and how accurately it can be computed

We begin with the logistic infection model,

$$\frac{dy}{dt} = k\,[N - y]\,y, \tag{1}$$

which describes how the number of infected individuals $y(t)$ changes over time within a closed population of size N. The term $y(N - y)$ models the interaction between infected and susceptible individuals: infection spreads fastest when both groups are large, producing the characteristic S-shaped curve [1].

To solve (1) analytically, we first separate variables [1, Ch. 1; Ch. 5.1]:

$$\frac{1}{(N - y)y}\,dy = k\,dt. \tag{2}$$

The left-hand side is a rational function, so we decompose it using partial fractions [1, Ch. 1]:

$$\frac{1}{(N - y)y} = \frac{A}{y} + \frac{B}{N - y}. \tag{3}$$

Multiplying both sides of (3) by $(N - y)y$ gives

$$1 = A(N - y) + By. \tag{4}$$

Setting $y = 0$ yields $A = 1/N$; setting $y = N$ gives $B = 1/N$. Substituting these values back into (3) produces

$$\frac{1}{(N - y)y} = \frac{1}{N}\left(\frac{1}{y} + \frac{1}{N - y}\right). \tag{5}$$

Integrating both sides of (2) using (5) gives the standard separable-ODE solution form [1, Ch. 5.1]:

$$\frac{1}{N}\big(\ln|y| - \ln|N-y|\big) = kt + C_1. \tag{6}$$

Combining logarithms and multiplying through by N yields

$$\ln\left|\frac{y}{N-y}\right| = Nkt + C, \tag{7}$$

where $C = NC_1$. Since $0 < y < N$, the argument of the logarithm is positive, so

$$\frac{y}{N-y} = Ce^{Nkt}. \tag{8}$$

Solving (8) for $y(t)$ gives the familiar closed-form logistic solution [1, Ch. 5.1]:

$$y(t) = \frac{CNe^{Nkt}}{1 + Ce^{Nkt}}. \tag{9}$$

Dividing numerator and denominator by Ce^{Nkt} produces the more common form

$$y(t) = \frac{N}{1 + A_0 e^{-Nkt}}, \tag{10}$$

where $A_0 = 1/C$. Applying the initial condition $y(0) = y_0$ gives

$$A_0 = \frac{N - y_0}{y_0}. \tag{11}$$

Hence, the final expression for the analytic solution is

$$\boxed{y(t) = \frac{N}{1 + \dfrac{N - y_0}{y_0} e^{-Nkt}}.} \tag{12}$$

Substituting the partial fraction expression from equation (5) into (2) makes the equation integrable. The resulting solution in equation (12) provides the exact analytical form, which is later used to check the accuracy of the numerical methods in Section 3.

Interpretation of the analytical solution. For small times, the exponential term e^{-Nkt} is large, and the solution grows almost exponentially. As time increases, the exponential term decays, causing the growth rate to slow. Eventually, $y(t)$ approaches the carrying capacity N, meaning that almost the entire population becomes infected. The model also has two equilibrium points, $y = 0$ (unstable) and $y = N$ (stable), consistent with the standard analysis of logistic growth models presented in [1].

3 Runge Kutta Methods

The Runge Kutta family of methods offers a practical and reliable way to obtain accurate numerical solutions of ordinary differential equations, without the need to compute higher derivatives. The basic idea is straightforward: instead of relying on a single slope estimate, as in simpler methods, we sample the function $f(t, y)$ at several points within each step. These intermediate evaluations give progressively better estimates of how the solution is changing, and by combining them in a suitable weighted average, the method closely reproduces the effect of higher order Taylor expansions using only function evaluations. Because of this balance between accuracy and ease of use, Runge Kutta methods have become one of the most popular tools for solving differential equations numerically [1, 2].

3.1 Runge Kutta Methods of Order Two (RK2)

The construction of RK methods is based on matching their numerical update rule with the Taylor expansion of the exact solution [1, 2]. Expanding $y(t)$ to second order about t_i gives

$$y(t_{i+1}) = y(t_i) + hf(t_i, y(t_i)) + \frac{h^2}{2}\Big[f_t(t_i, y_i) + f_y(t_i, y_i)\, f(t_i, y_i)\Big] + O(h^3). \tag{13}$$

To avoid calculating the partial derivatives f_t and f_y, we introduce a general two–stage RK method [1, 2]:

$$\begin{aligned} k_1 &= f(t_i, y_i), \\ k_2 &= f(t_i + \beta_1 h,\ y_i + \beta_2 h k_1), \\ y_{i+1} &= y_i + h(a_1 k_1 + a_2 k_2), \end{aligned} \tag{14}$$

where the constants a_1, a_2, β_1, and β_2 are chosen to reproduce the Taylor expansion.

Expanding k_2 about (t_i, y_i) gives

$$k_2 = f(t_i, y_i) + \beta_1 h f_t(t_i, y_i) + \beta_2 h f_y(t_i, y_i) f(t_i, y_i) + O(h^2).$$

Substituting this into the update rule yields

$$y_{i+1} = y_i + h(a_1 + a_2) f(t_i, y_i) + a_2 h^2 \big[\beta_1 f_t(t_i, y_i) + \beta_2 f_y(t_i, y_i) f(t_i, y_i)\big] + O(h^3).$$

Matching coefficients with the Taylor expansion (13) gives the **order conditions** [1, 2]:

$$a_1 + a_2 = 1, \tag{15}$$
$$a_2\beta_1 = \frac{1}{2}, \tag{16}$$
$$a_2\beta_2 = \frac{1}{2}. \tag{17}$$

Any constants satisfying these three equations produce a second–order RK method.

Midpoint Method (a standard RK2 scheme)

One convenient choice is [1, 2]:

$$\beta_1 = \beta_2 = \frac{1}{2}, \qquad a_1 = 0, \qquad a_2 = 1,$$

which yields the **midpoint method**:

$$\begin{aligned} k_1 &= f(t_i, y_i), \\ k_2 &= f\left(t_i + \frac{h}{2},\ y_i + \frac{h}{2}k_1\right), \\ y_{i+1} &= y_i + hk_2. \end{aligned} \tag{18}$$

Here, the slope is evaluated at the midpoint of the interval $[t_i, t_{i+1}]$. This method requires two function evaluations per step and achieves global error $O(h^2)$ [1, 2].

3.2 Runge Kutta Methods of Order Four (RK4)

The same principles used to derive RK2 can be extended to obtain higher order methods. The most widely used scheme is the **classical fourth order Runge Kutta method (RK4)** [1, 2], which matches the Taylor expansion up to $O(h^4)$.

The resulting scheme uses four slope evaluations [1, 2]:

$$\begin{aligned}
k_1 &= hf(t_i, y_i), \\
k_2 &= hf\left(t_i + \frac{h}{2},\ y_i + \frac{k_1}{2}\right), \\
k_3 &= hf\left(t_i + \frac{h}{2},\ y_i + \frac{k_2}{2}\right), \\
k_4 &= hf(t_i + h,\ y_i + k_3), \\
y_{i+1} &= y_i + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right).
\end{aligned} \tag{19}$$

The interpretation is straightforward [1, 2]:
k_1 is the initial Euler slope,
k_2 and k_3 sample the slope at refined midpoint predictions,
k_4 is the slope at the end of the interval.
The update formula is a weighted average that cancels lower–order errors and achieves [1, 2]:

$$\text{local truncation error } = O(h^5), \qquad \text{global error } = O(h^4).$$

With only four function evaluations per step, RK4 provides excellent accuracy for its computational cost and is widely used in scientific computing.

4 Results and Discussion

We now apply the analytical and numerical methods to the problem with parameters $N = 1{,}500{,}000$, initial infection $y_0 = 100$, spread rate $k = 2 \times 10^{-7}$, and time interval $0 \leq t \leq 30$ days. The numerical solution uses $M = 30$ steps, giving a step size of $h = 1$ day.

Using the analytical solution [1]

$$y(t) = \frac{N}{1 + A_0 e^{-Nkt}}, \qquad A_0 = \frac{N - y_0}{y_0}, \tag{20}$$

we compute

$$\begin{aligned}
A_0 &= \frac{1{,}500{,}000 - 100}{100} = 14{,}999, \\
-Nkt &= -(1{,}500{,}000)(2 \times 10^{-7})(30) = -9.0,
\end{aligned}$$

which gives

$$y(30) = \frac{1{,}500{,}000}{1 + 14{,}999e^{-9.0}} \approx 526{,}126.82. \tag{21}$$

The 4th-order Runge–Kutta (RK4) method was then applied to the same model, giving the numerical approximation [1, 2]

$$y_{30} \approx 525{,}990.62. \tag{22}$$

This corresponds to an absolute error of 136.20 and a relative error of 0.0259%, demonstrating very close agreement between the numerical and analytical solutions [1, 2].

Table 1: Comparison of analytical and RK4 solutions at $t = 30$ days for the logistic infection model. The table shows excellent agreement between the two methods, with a relative error below 0.03%.

Method	Infected at $t = 30$	Absolute Error
Analytical solution	526,126.82	–
RK4 ($M = 30$)	525,990.62	136.20

Figure 1 illustrates the excellent match between the analytical and RK4 solutions [2]. The RK4 curve almost perfectly overlaps the exact solution, confirming the high accuracy of the method even with a step size of $h = 1$ day.

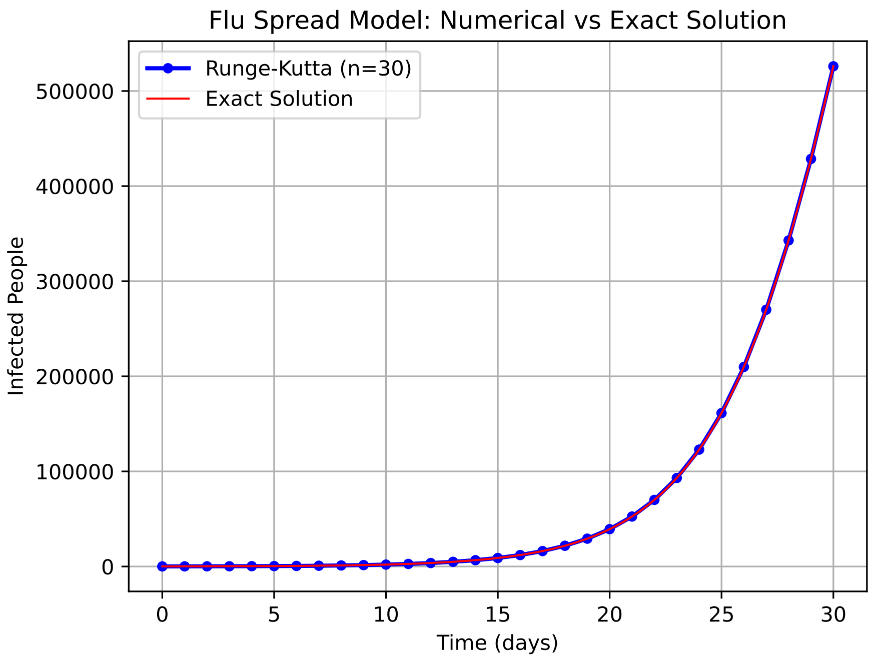

Figure 1: Comparison between the exact logistic solution and the RK4 numerical approximation ($M = 30$, $h = 1$ day). The RK4 curve closely follows the analytical solution over the 30-day interval, illustrating the accuracy of the method even for relatively large step sizes [1, 2]. The vertical axis shows the number of infected individuals, and the horizontal axis shows time in days.

The absolute error between the exact solution (21) and the RK4 approximation (22) is

$$\text{Error} = |526{,}126.82 - 525{,}990.62| = 136.20, \tag{23}$$

corresponding to a relative error of only 0.0259%. This shows RK4 remains highly accurate even with relatively few steps[1, 2].

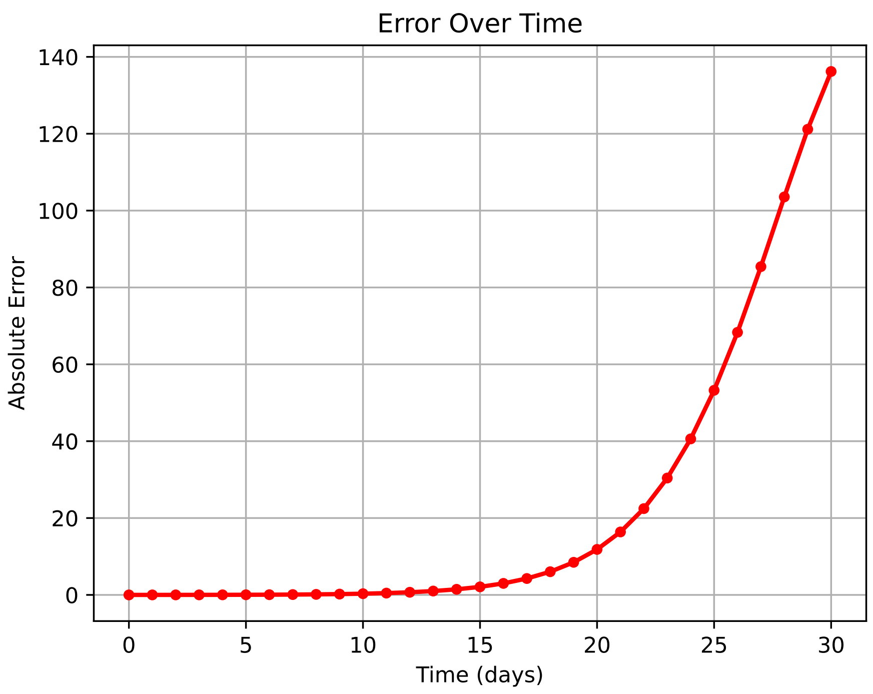

Figure 2: Absolute error of the RK4 approximation over time for step size $h = 1$ day. The error remains small throughout the simulation, increasing slightly as t grows due to the accumulation of local truncation error [1].

4.1 Convergence Analysis (Task 5)

To assess how accurately the fourth order Runge Kutta method approximates the solution, the value of $y(30)$ was computed using a sequence of step counts M, with step sizes $h = 30/M$. For each choice of M, the numerical result $y_h(30)$ was compared with the exact analytical value [1]

$$y(30) = 526{,}126.81737.$$

A useful way to understand how a numerical method behaves is to examine how its error changes as the step size becomes smaller. When the step size is large, such as $h = 15$ or $h = 7.5$, the numerical solution uses only one or two steps across the whole 30-day interval. Unsurprisingly, the approximation in this regime is very rough, and the error is correspondingly large. The method simply does not have enough resolution to follow the steep early growth of the logistic curve [2].

However, as soon as the step size is reduced, the picture changes dramatically. Refining h even a little produces a noticeably more accurate result, and once h enters the range where the solution is sampled many times during the fast growth phase, the error drops very rapidly. This is the regime where the theoretical fourth-order accuracy becomes visible: halving h reduces the global error by approximately a factor of 16, consistent with $O(h^4)$ error behaviour [1, 2].

This behaviour appears clearly in the log–log plot of error versus the number of steps M in Figure 3. As M increases (so $h = 30/M$ decreases), the points lie close to a straight line, showing that the error follows a power law in the step size.

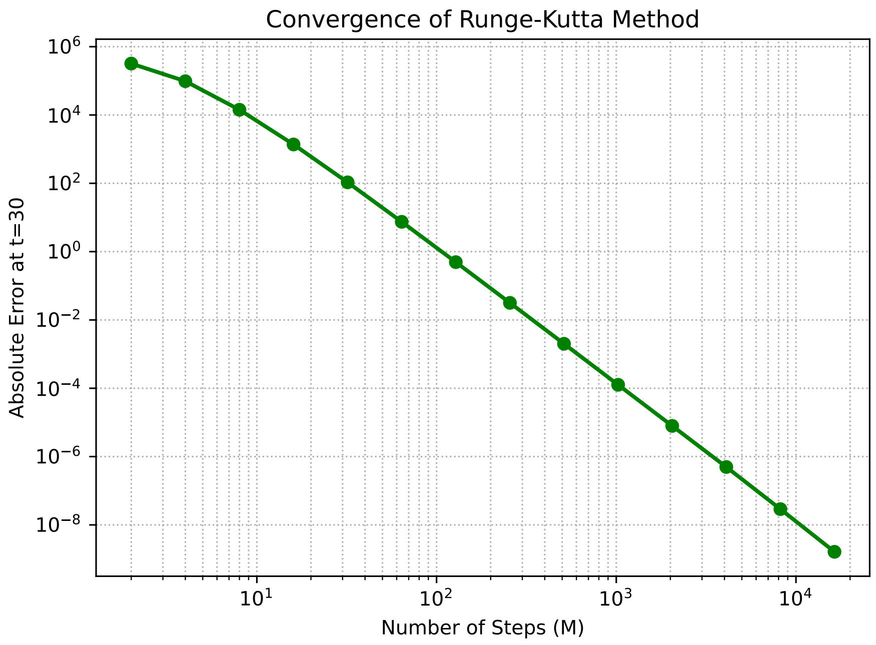

Figure 3: Log–log plot of absolute error at $t = 30$ versus number of steps M for the RK4 method, using $M = 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192$, and 16384 (equivalently step sizes $h = 30/M$). The points lie approximately on a straight line, consistent with the theoretical $O(h^4)$ convergence of the 4th–order Runge–Kutta method [1, 2].

When h is extremely small, the convergence eventually levels off. The RK4 method has not stopped improving; instead, floating-point round-off becomes comparable to the remaining discretisation error [1], so further reducing h gives little extra accuracy.

Overall, the results follow the theoretical expectations very closely: slow improvement for very coarse step sizes, followed by extremely rapid error reduction once h is sufficiently small, and finally a plateau when numerical round-off limits the achievable precision. This behaviour confirms both the correct implementation of the method and the characteristic fourth-order accuracy of the classical Runge Kutta scheme [1, 2].

5 Future Work

A natural extension is to explore adaptive Runge–Kutta methods, in particular the Runge Kutta Fehlberg scheme (RKF45). Unlike the fixed step methods considered in this report Euler's method, RK2 and the classical RK4, the RKF45 method does not commit to a single step size across the whole interval. Instead, it automatically adjusts the step size according to how the solution behaves. When the solution varies slowly, the method takes large steps to save computational effort; when the solution changes more rapidly, the method decreases the step size to maintain accuracy. Studying RKF45 would therefore provide a natural progression from the fixed step schemes analysed in this work, as it illustrates how modern ODE solvers combine higher order Runge Kutta formulas with built in error estimation to deliver both accuracy and efficiency [1, 2].

5.1 Adaptive Runge Kutta Fehlberg Method (RKF45)

All of the Runge Kutta methods discussed earlier used a fixed step size $h = (b - a)/N$, meaning the interval $[a, b]$ is divided uniformly and the method commits to this spacing regardless of how the solution behaves. This simplicity is convenient, but not always efficient. If the solution changes slowly, a large step size would still be accurate, and if the solution changes rapidly, a smaller step is needed to avoid losing accuracy. A constant step size cannot react to this changing behaviour [1].

Adaptive Runge Kutta methods address this limitation by adjusting the step size dynamically based on an estimate of the local truncation error at each step. The most widely used example is the Runge Kutta Fehlberg method of order $(4, 5)$ (RKF45), developed by Fehlberg in 1969, which embeds two formulas of different orders into a single algorithm [1, 2].

Embedded Formulas

RKF45 uses a single group of six function evaluations,

$$k_1, k_2, k_3, k_4, k_5, k_6,$$

to compute *two* approximations of $y(t_{i+1})$: one of order four and a more accurate one of order five. Because both approximations share the same $k_1, \dots, k_6$, the higher order estimate is obtained essentially for free, with no additional evaluations of $f(t, y)$ [1].

The stage values are

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

Using these values, the fourth order approximation is

$$y_{i+1} = y_i + \frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 \frac{1}{5}k_5,$$

while the fifth order approximation is

$$\tilde{y}_{i+1} = y_i + \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 \frac{9}{50}k_5 + \frac{2}{55}k_6.$$

Local Error Estimate

Because $\tilde{y}_{i+1}$ is one order more accurate than y_{i+1}, their difference provides a direct estimate of the local truncation error *per unit step*:

$$\tau_{i+1}(h) \approx \frac{1}{h}|\tilde{y}_{i+1} - y_{i+1}|$$

[1, 2]. We define

$$R = \frac{1}{h} \left| \tilde{y}_{i+1} - y_{i+1} \right|,$$

which serves as a real-time indicator of whether the most recent step was sufficiently accurate.

Step Acceptance and Adjustment

Let $\varepsilon > 0$ be the chosen tolerance. If $R \leq \varepsilon$, the step is accepted. If $R > \varepsilon$, the step is rejected and recomputed with a smaller step size.

To determine the new step size, RKF45 uses

$$h_{\text{new}} = 0.84 \left(\frac{\varepsilon}{R} \right)^{1/4} h.$$

The exponent $1/4$ reflects the order of the error estimate, which is fourth order in this embedded pair [1].

Benefits of RKF45

RKF45 automatically takes small steps when the solution varies rapidly and larger steps when the solution is smooth. Unlike fixed step RK4, which must use the same step size everywhere (even when it is unnecessarily small), RKF45 distributes computational effort where it is needed most. For the nonlinear epidemic model studied earlier, this means very small steps near the steep initial growth phase and much larger steps later on. As a result, RKF45 often achieves a target accuracy with fewer total evaluations of $f(t, y)$ than any fixed step method, while maintaining reliability through continuous error monitoring [2].

5.2 Computational Cost

When comparing numerical methods, it is not enough to look only at their accuracy; we must also consider how expensive they are to run. For fixed step schemes such as the classical RK4 method, the computational cost is directly tied to how many times the function $f(t, y)$ must be evaluated. RK4 always performs four evaluations per step, so if the interval $[a, b]$ is split into N steps, the total cost is proportional to $4N$ [1]. By contrast, the Runge Kutta Fehlberg method (RKF45) performs six evaluations for each attempted step, so at first glance it appears more expensive [1, 2].

However, RKF45 behaves very differently in practice. Because it adapts the step size automatically using its built in error estimate, it takes large steps whenever the solution is behaving smoothly, and only switches to smaller steps when the solution changes rapidly. As a result, although each step costs more, the method takes far fewer steps overall. This distinction becomes clear when comparing the two methods directly.

Figure 4 shows how the absolute error at $t = 30$ varies with the total number of function evaluations for both RK4 and RKF45. Each point on the RK4 curve corresponds to a fixed step size, while each point on the RKF45 curve corresponds to a different error tolerance. The horizontal axis effectively measures computational effort, and the vertical axis measures the resulting accuracy [2].

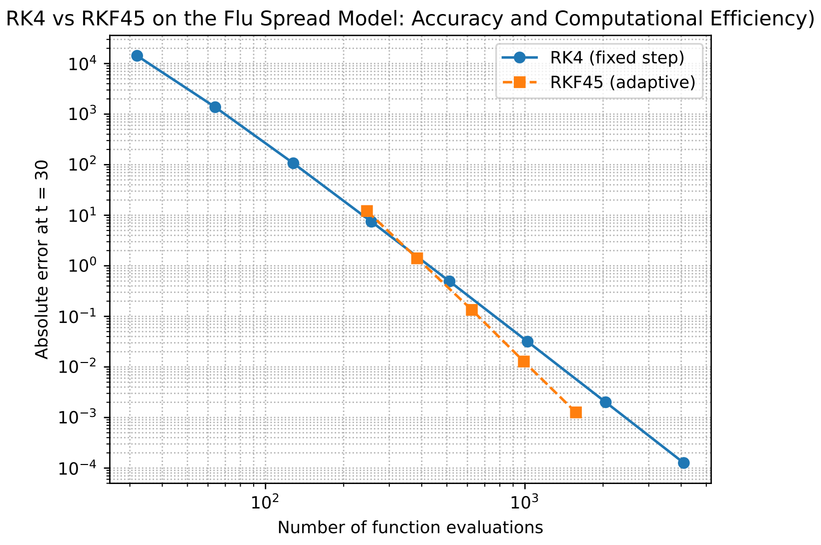

Figure 4: Comparison of RK4 (fixed step) and RKF45 (adaptive) for the flu model. The adaptive RKF45 method achieves much smaller error for the same computational cost, showing its efficiency when the solution has regions of both slow and rapid change [1, 2].

The graph makes the comparison very clear. For the same number of function evaluations, RKF45 consistently produces a far more accurate result than fixed step RK4. In fact, to reach the level of accuracy that RKF45 achieves automatically, RK4 would need to use a step size small enough to be unnecessarily restrictive in smooth parts of the solution. Consequently, even though RK4 is cheaper per individual step, RKF45 is often cheaper overall because it avoids wasting work where high resolution is not needed [1].

In summary, while RK4 is a robust and simple fixed step method, RKF45 is often significantly more efficient for a desired level of accuracy because it automatically minimizes computational effort. The graph clearly illustrates this difference and highlights why adaptive methods are so widely used in modern scientific computing [1, 2].

References

[1] R. L. Burden, J. D. Faires, and A. M. Burden, *Numerical Analysis* (10th ed.). Cengage Learning, 2016.

[2] D. Dundas, *MTH3023: Numerical Analysis. Part 7: Numerical Solution of Ordinary Differential Equations*. Queen's University Belfast.