Convergence, Accuracy and Efficiency Study of Numerical Solution of ODE

Dr Rishikant Agnihotri*, Thakur Bajrang Kumar Singh**
* Professor, Kalinga University, Raipur,
** Research Scholar, Kalinga University, Raipur,

Abstract:

In this paper we presents two numerical experiments that focus on investigating the accuracy and efficiency properties of the five explicit Runge-Kutta methods. The first numerical experiment is a convergence study and the second numerical experiment is an efficiency study. MATLAB is also used in the convergence study and the efficiency study to solve initial value problems numerically and to visualize the results in figures. This is done by running the scripts mentioned in the convergence study and the efficiency study in MATLAB.

Convergence study

In the convergence study, we measure the order of accuracy of the five explicit Runge-Kutta methods for solving an initial value problem of a first-order linear ODE by verifying the convergence rate p. In this study, the following initial value problem of a first-order linear ODE is considered

$$y'(t) = -2ty(t),\ y(0) = 1,\ \text{for } t \in [0, T]. \tag{1}$$

A similar first-order linear ODE as Equation (1) can be found in [10]. Equation (1) is chosen to be considered in this study because it is a simple initial value problem with a known exact solution which enables us to compute the global truncation error at the final time T.

The exact solution of Equation (1) is

$$y(t) = e^{-t^2}.$$

In the study, we verify the convergence rate p through three steps. The first step is to compute a numerical solution y_n^h of Equation (1) with the step size h. We choose the step sizes h, $\frac{h}{2}$, $\frac{h}{4}$ and $\frac{h}{8}$, which decrease by a factor of 2 because according to Leveque [16] it is a common way to make h small to get highly accurate numerical solutions that we want to obtain. Additionally, other factors can certainly be used as well. The second step is that we want to compute the following global truncation error from [28] at the final time T

$$e^h = |y_n^h - y(t_n)|. \tag{2}$$

The global truncation error e^h with step size h is the difference between the the numerical solution y_n^h and the exact solution $y(t_n)$. The third step is that we want to estimate the convergence rate p by computing three approximations of the convergence rate p using [16].

We solve Equation (1) using the forward Euler method, Heun's method, RK4 and RK5 on the time interval $t \in [0, T]$, where $T = 1$ and with the step sizes $h = 0.1$, $\frac{h}{2}$, $\frac{h}{4}$ and $\frac{h}{8}$. We solve Equation (1) using RK8 on $t \in [0, T]$, where $T = 1.2$ and with the step sizes $h = 0.4$, $\frac{h}{2}$, $\frac{h}{4}$ and $\frac{h}{8}$. We have chosen other step sizes h for RK8 because RK8 reaches faster to the machine precision error 10^{-16} which is the smallest relative error we can get by a computer. Heath [10] describes a relative error as a quotient of the global truncation error and the exact solution. Furthermore, RK8 reaches faster to the machine precision error because it has a high accuracy level compared to for instance the forward Euler method. In the convergence study, we do not want to reach this limit of error because if we reach this limit, then we will not be able to decrease the errors even more. Therefore, for RK8 we need to use step sizes h that are larger than $h = 0.1$, $\frac{h}{2}$, $\frac{h}{4}$ and $\frac{h}{8}$ just to make sure that the computed errors using RK8 do not decrease too fast. This is why the step sizes $h = 0.4$, $\frac{h}{2}$, $\frac{h}{4}$ and $\frac{h}{8}$ are used instead. Furthermore, the time interval $t \in [0, 1.2]$ was used instead of $t \in [0, 1]$ because if we start to solve Equation (1) with $h = 0.4$, then $h = 0.4$ will not fit in $t \in [0, 1]$. However, it does fit in $t \in [0, 1.2]$.

Script 2 is used to define Equation (4.1) which is solved using the forward Euler method (see Script 3), Heun's method (see Script 4), RK4 (see Script 5), RK5 (see Script 6) and RK8 (see Script 7). The data obtained with Script 2 to Script 7 were put in a tabular format in LATEX to get Table 1 to Table 5. The results shown in Table 1 to Table 5 are also presented in Figure 1 which is obtainedby running Script 8.

h	Global truncation errors	Approximations of convergence rate
0.1	1.3827×10^{-2}	—
0.05	6.5045×10^{-3}	≈ 1.0880
0.025	3.1569×10^{-3}	≈ 1.0430
0.0125	1.5554×10^{-3}	≈ 1.0210

Table 1: Global truncation errors and approximations of the convergence rate of the forward Euler method for solving $y'(t) = -2ty(t), \quad y(0) = 1,$ for $t \in [0, 1]$.

h	Global truncation error	Approximation of convergence rate
0.1	1.1739×10^{-3}	—
0.05	3.0109×10^{-4}	≈ 1.9630
0.025	7.6014×10^{-5}	≈ 1.9860
0.0125	1.9085×10^{-5}	≈ 1.9940

Table 2: Global truncation errors and approximations of the convergence rate of Heun's method for solving $y'(t) = -2ty(t)$, $y(0) = 1$, for $t \in [0, 1]$.

h	Global truncation errors	Approximations of convergence rate
0.1	1.6252×10^{-6}	—
0.05	1.0253×10^{-7}	≈ 3.9860
0.025	6.4067×10^{-9}	≈ 4.0000
0.0125	3.9993×10^{-10}	≈ 4.0020

Table 3: Global truncation errors and approximations of convergence rate of RK4 for solving $y'(t) = -2ty(t)$, $y(0) = 1$, for $t \in [0, 1]$.

h	Global truncation errors	Approximations of convergence rate
0.1	2.8055×10^{-8}	—
0.05	8.3665×10^{-10}	≈ 5.0670
0.025	2.5427×10^{-11}	≈ 5.0401
0.0125	7.8292×10^{-13}	≈ 5.0210

Table 4: Global truncation errors and approximations of the convergence rate of RK5 for solving $y'(t) = -2ty(t)$, $y(0) = 1$, for $t \in [0, 1]$.

h	Global truncation errors	Approximations of convergence rate
0.4	1.1530×10^{-7}	—
0.2	5.2249×10^{-10}	≈ 7.7860
0.1	1.9577×10^{-12}	≈ 8.0600
0.05	7.3552×10^{-15}	≈ 8.0560

Table 5: Global truncation errors and approximations of the convergence rate of RK8 for solving $y'(t) = -2ty(t)$, $y(0) = 1$, for $t \in [0, 1.2]$.

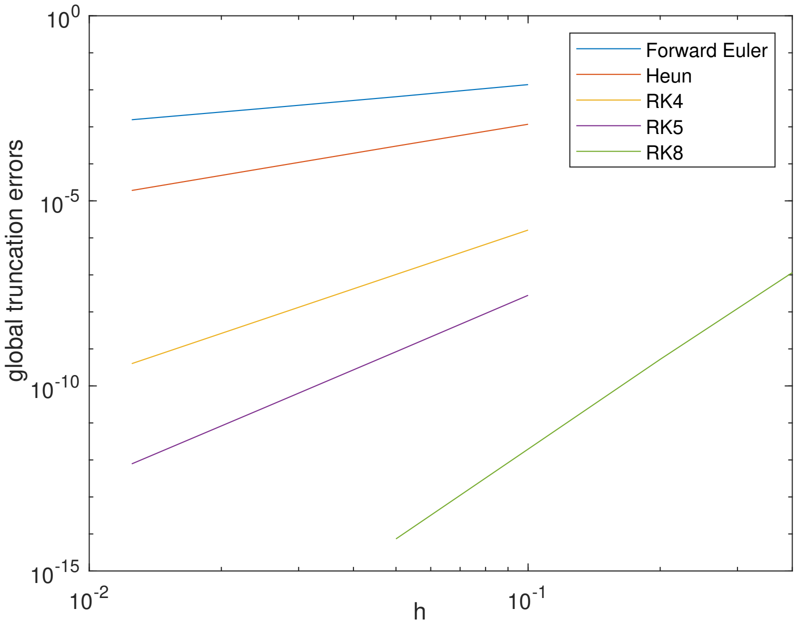

Figure 1: Global truncation errors in the numerical solution of $y'(t) = -2ty(t)$,$y(0) = 1$, for $t \in [0, T]$ using the five explicit Runge-Kutta methods against the step size h on a log-log scale with logarithm of base 10.

Table 1 to Table 5 shows global truncation errors and three approximations of the convergence rate p of the five explicit Runge-Kutta methods. As the step sizes h in Table 1 to Table 5 is halved, the global truncation errors of the five explicit Runge-Kutta methods also decrease by a factor of approximately 2p. According to Leveque [16], if global truncation errors decrease by a factor of approximately 2p, then for a forward Euler method which is first-order accurate with p = 1 and we get that the global truncation errors of the forward Euler method decrease by approximately a factor of 2 because $2^1 = 2$. This means that we should expect that the five explicit Runge-Kutta methods decrease by a factor of approximately 2p. We can now check if the global truncation errors in Table 1 to Table 5 do decreaseby a factor of approximately 2p. We check this by investigating how much the first global truncation error in the tables decreases.

In Table 1, the first global truncation error decreases by a factor of approximately 2 because

$$\frac{1.3827 \times 10^{-2}}{6.5045 \times 10^{-3}} \approx 2.1260. \tag{3}$$

From Table 1, we can see that as the step sizes decrease by half, the global truncation errors in Table 1 decrease by a factor of approximately 2. We can perform this division (3) for all global truncation errors in Table 1 to check how much they decrease. In Table 1, we can see that the forward Euler method has the convergence rate $p \approx 1$, therefore it is first-order accurate. In Figure 1, we can see

that the global truncation errors of the forward Euler method are on a line of slope 1, which shows that the global truncation errors of the forward Euler method are equal to $\mathcal{O}(h)$ which is proportional to h.

In Table 2, we can see that the global truncation errors of Heun's method decrease by a factor of approximately 4. The first global truncation error decrease by a factor of approximately 4 because

$$\frac{1.1739 \times 10^{-3}}{3.0109 \times 10^{-4}} \approx 3.8988.$$

In Table 2, we can see that Heun's method has the convergence rate $p \approx 2$, thereforeit

is second-order accurate. Heun's method is on a line of slope 2, therefore the global truncation errors of Heun's method are equal to $\mathcal{O}(h^2)$, which is proportional to h^2 (see Figure 1).

For RK4, the global truncation errors decrease by a factor of approximately 16, which is in agreement with [4]. According to Atkinson et al. [4] we should expect that RK4 decrease by approximately $2^4 = 16$. The first global truncation error decrease by a factor of approximately 16 (see Table 3) as follows

$$\frac{1.6252 \times 10^{-6}}{1.0253 \times 10^{-7}} \approx 15.851.$$

From Table 3 we can see that RK4 is fourth-order accurate since it has the convergence rate $p \approx 4$. In Figure 4.1, we can see that the global truncation errors of RK4 are on a line of slope 4 and equal to $\mathcal{O}(h^4)$ which is proportional to h^4.

RK4 is more accurate than the forward Euler method and Heun's method because the computed global truncation errors of RK4 are smaller than the global truncation errors of the forward Euler method and Heun's method. The global truncation errors of RK4 decrease faster than for the global truncation errors of the forward Euler method and Heun's method. Ascher [2] also have computed the convergence rate of the forward Euler method and RK4 using formulas that are similar to the formula (3.1). The convergence rate of the forward Euler method and RK4 obtained in this section is also obtained in [2].

In Table 4, we can see that for RK5 we expect that the global truncation errors should decrease by a factor of approximately 32 since $2^5 = 32$. However, we get that first global truncation error decrease by a factor of approximately 34 because

$$\frac{2.8055 \times 10^{-8}}{8.3665 \times 10^{-10}} \approx 33.5329.$$

The first global truncation error decreased more than what we should expect. RK5 has the convergence rate $p \approx 5$, therefore it is fifth-order accurate. The global truncation errors of RK5 are on a line of slope 5 and are equal to $\mathcal{O}(h^5)$ which is proportional to h^5 (see Figure 1). It is difficult to find in the literature about the convergence rate p of RK5.

In Table 5, we can see that for RK8, the global truncation errors should decrease by a factor of approximately 256 because $2^8 = 256$, but we get that the first global truncation error decrease by

$$\frac{1.1530 \times 10^{-7}}{5.2249 \times 10^{-10}} \approx 220.6740,$$

which is approximately 221. This means that the first global truncation error decreases less than 256. RK8 has the convergence rate $p \approx 8$, therefore it is eighth-order accurate (see Table 5). In Figure 1, the global truncation errors of RK8 are on aline of slope 8 and it is equal to $\mathcal{O}(h^8)$ which is proportional to h^8. It is difficult to find in the literature about the convergence rate p of RK8. From Table 1 to Table 5 we see that the global truncation errors of the five explicit Runge-Kutta methods decrease differently because the order of accuracy of the five explicit Runge-Kutta methods is different.

Moreover, Figure 1 shows that the five explicit Runge-Kutta methods converge and the global truncation errors are linear with the slope of the convergence rate p of the five explicit Runge-Kutta methods on a log-log scale with the logarithm of base 10. A log-log scale with the logarithm of base 10 is used because according to Papadopoulos and Simos [19] when we measure accuracy by computing errors at the final time point we get decimal digits which we can present in the logarithm of base 10. In Figure 1, we can see that an explicit Runge-Kutta method converges faster to the exact solution of Equation (1) as the order of accuracy of the explicit Runge-Kutta method gets higher. The forward Euler method and Heun's method do have a low-order of accuracy, therefore they do not converge as fast to the exact solution as RK4, RK5, and RK8 of high-order of accuracy. The forward Euler method has the lowest order of accuracy and RK8 has the highest order of accuracy of all the five explicit Runge-Kutta methods. RK8 converges faster to the exact solution with the smallest global truncation errors, therefore it is the most accurate explicit Runge-Kutta method. While the forward Euler method, Heun's method, RK4, and RK5 have greater global truncation errors than RK8, therefore they are less accurate. The forward Euler method is the least accurate explicit Runge-Kutta method. There are smaller global truncation errors in RK5 than the forward Euler method, Heun's method, and RK4, therefore RK5 is more accurate. We can see in Figure 1 that the slope for the forward Euler method and Heun's method involving greater global truncation errors are shown in the upper part of Figure 1. While the slope for RK4, RK5, and RK8 involving smaller global truncation errors are shown in the lower part of Figure 1.

Efficiency study

In this efficiency study, we investigate the efficiency of the five explicit Runge-Kutta methods for solving an initial value problem for a system of first-order linear ODEs. This study is carried out to find which one of the five explicit Runge-Kutta methods is most efficient for solving a system of first-order linear ODEs.

The system of first-order linear ODEs is defined as the following,

$$\begin{aligned} \vec{y}\,'(t) &= A\vec{y}(t) + \vec{b}(t), \\ \vec{y}(0) &= \vec{y}_0,\ t \in [0,1]. \end{aligned} \tag{4}$$

The exact solution of Equation (4) is a 40401×1 column vector at the final time point t =1. This column vector is the exact solution of the first 40401 elements in the initial column vector $\vec{y}_0$. Also, $\vec{y}_0$ is a 80802×1 column vector of initial values. In this efficiency study, we want to find the numerical solution of Equation (4) at the final time point t =1. The matrix A is a 80802×80802 matrix involving 80802×80802 real numbers, $\vec{y}(t)$ is a column vector of unknown functions $y_1(t), \cdots, y_k(t)$ dependent on t. Also, $\vec{b}(t)$ is a 80802×1 zero column vector of 80802 zeros and it has length zero. The matrix A of Equation (4) is already constructed and it is originally obtained by the finite difference method for the two-dimensional wave equation in space. Constructing this matrix A is not part of this thesis but the theoryfor how to construct this matrix A can be found in [17]. Script 9 is used to definethe Equation (4).

This efficiency study is carried out through two steps. The first step is to find out the smallest number of time points n such that the five explicit Runge-Kutta methods are stable. In the first step, we want to obtain the stability limit n for the five explicit Runge-Kutta methods. In the second step, we solve Equation (4) using the stability limit n and we compute global truncation errors at the final time point in the numerical solution of Equation (4) using the five explicit Runge-Kutta methods. We are computing the global truncation errors as we increase the stability limit n until the global truncation errors stop decreasing. In the second step, we also measure the time required for the five explicit Runge-Kutta methods to compute the global truncation errors as the stability limit n increases.

We measure the global truncation errors at the final time point using L_2 norm which according to Heath [10] is a vector norm. Vector norms are used to measure errors and there are different types of vector norms that are useful in different contexts. There are m-norms of a vector $\vec{x}$, where the integer m is $m > 0$. If $m = 1$, then we have a L_1 norm which is used to measure the magnitude of the distance between two points. If $m = 2$, then we have L_2 norm which computes the magnitude of the difference between the numerical solution and the exact solution at a time point. If $m = \infty$, then we have a ∞-norm which is used to compute the maximum error over a time interval.

Assume that we have an global truncation error vector defined by

$$\vec{e_n} = \vec{y}_n - \vec{y}(t_n),$$

where $\vec{y}_n$ is the numerical solution vector and $\vec{y}(t_n)$ is the exact solution vector at the final time point t_n. Moreover, a L_2 norm of $\vec{e_n}$ at the final time point t_n is

$$\|\vec{e_n}\|_2 = \|\vec{y}_n - \vec{y}(t_n)\|_2, \tag{5}$$

which we use. A similar global truncation error as Equation (5)can be found in [16].

Stability limits

In this section, we want to find when the five explicit Runge-Kutta methods are stable because we do not know at what n we should start to solve Equation (4). Therefore, we solve Equation (4) using different values of n and solve Equation (4) in order to check when the five explicit Runge-Kutta methods become stable. We run the MATLAB codes in Script 10, Script 12, Script 14, Script 16, and Script 18 to obtain the stability limit n of the forward Euler method, Heun's method, RK4, RK5, and RK8, respectively. We run the MATLAB codes several times until the five explicit Runge-Kutta methods get started to become stable, which is when the numerical solution converges to the exact solution. It is given that the exact solution of the Equation (4) is a magnitude near 5. As a result, we obtain different values of n for which the five explicit Runge-Kutta methods are stable and unstable as presented in Table 6. Table 6 shows the stability limits of the five explicit

Method	Stable	Unstable
Forward Euler	$n \geq 22000$	$n \leq 21800$
Heun	$n \geq 2100$	$n \leq 1980$
RK4	$n \geq 435$	$n \leq 430$
RK5	$n \geq 580$	$n \leq 560$
RK8	$n \geq 328$	$n \leq 300$

Table 6: Stability limit n of the five explicit Runge-Kutta methods for $\vec{y}'(t) = A\vec{y}(t) + \vec{b}(t), \quad \vec{y}(0) = \vec{y}_0, \quad$ for $t \in [0, 1]$.

Runge-Kutta methods for solving Equation (4). We can determine that an explicit Runge-Kutta method is stable when the numerical solution move closer to the exact solution. Also, if the numerical solution does not move closer to the exact solution, then the explicit Runge-Kutta method is unstable. We can see that when $n \geq 22000$ the forward Euler method is stable but unstable when $n \leq 21800$. This means that for the forward Euler method it takes $n \geq 22000$ number of time points in order to be stable for solving Equation (4). Also, $n \leq 21800$ is the number of time points for which the forward Euler method is unstable for solving Equation (4). Heun's method is stable when $n \geq 2100$, but unstable when $n \leq 1980$. RK4 is stable when $n \geq 435$, but unstable when $n \leq 430$. RK5 is stable when $n \geq 580$ but unstable when $n \leq 560$. RK8 is stable when $n \geq 328$ and unstable when $n \leq 300$.

When the forward Euler method, Heun's method, RK4, RK5, and RK8 is stable, then the numerical solution of Equation (4) obtained by these five explicit Runge-Kutta methods converge to the exact solution and bound the exact solution. When the five explicit Runge-Kutta methods are unstable, the numerical solution diverges from the exact solution and the numerical solution unbound the exact solution. The forward Euler method and Heun's method have the highest n number of time points in order to be stable. This means that it takes more time steps for the forward Euler method and Heun's method to converge to the exact solution than for the RK4, RK5, and RK8. These results are in agreement with what Heath [10] defines a stable and unstable numerical method as discussed.

Time-error efficiency

In this efficiency study, we want to find out the computational time required for the five explicit Runge-Kutta methods to compute the global truncation errors at the final time point t_n using the stability limit n in Table 6. We also compute the global truncation errors in the numerical solution of Equation (4) at increasing stability$n$ using Equation (5) and measure the computational time of the global truncation errors. We run the MATLAB codes in Script 11, Script 13, Script 15, Script 17, and Script 19 to compute the global truncation errors and the computational time. The data obtained by running these scripts were plotted in tables and then the data in the tables were visualized using figures in MATLAB. This section presents two figures for each explicit Runge-Kutta method. The first figure visualizes the global truncation errors at increasing values of the stability limit n. The second figure visualizes the computational time in seconds.

Moreover, the results of the global truncation errors and the computational time are presented in Figure 2 to Figure 11. We can see from Figure 2 to Figure

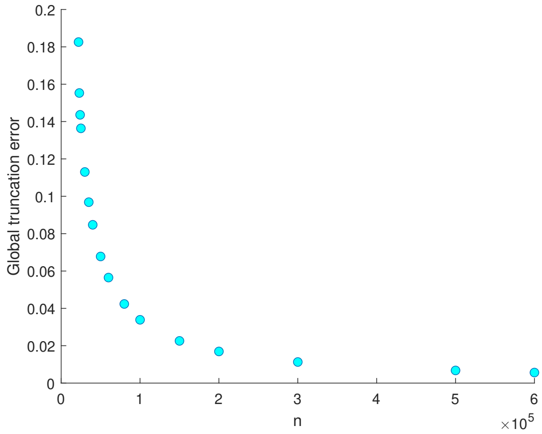

Figure 2: Global truncation errors in the numerical solution of $\vec{y}\,'(t) = A\vec{y}(t) + \vec{b}(t), \quad \vec{y}(0) = \vec{y}_0, \quad$ for $t \in [0, 1]$ computed at increasing stability n number of time points using the forward Euler method.

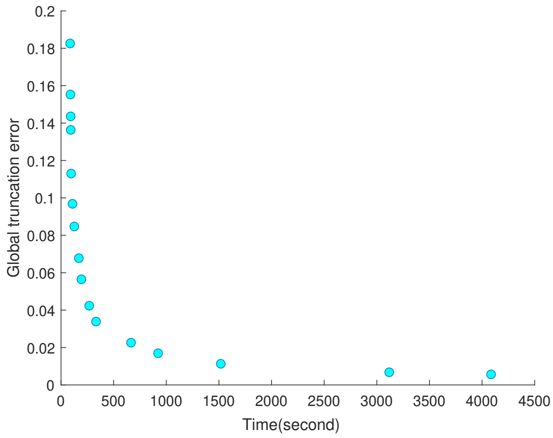

Figure 3: Time (seconds) required for the forward Euler method to compute global truncation errors in the numerical solution of the system of ODEs.

11 that all the five explicit Runge-Kutta methods are stable because the global truncation errors decrease asymptotically. This is because as the stability limit n

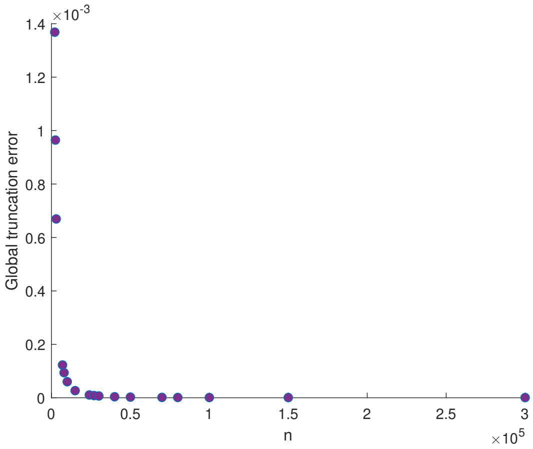

Figure 4: Global truncation errors in the numerical solution of $\vec{y}'(t) = A\vec{y}(t) + \vec{b}(t), \quad \vec{y}(0) = \vec{y}_0, \quad \text{for } t \in [0, 1]$ computed at increasing stability n number of time points using Heun's method.

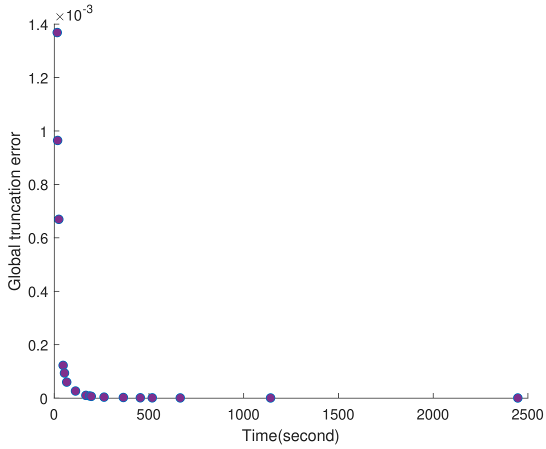

Figure 5: Time (seconds) required for the Heun's method to compute global truncation errors in the numerical solution of the system of ODEs.

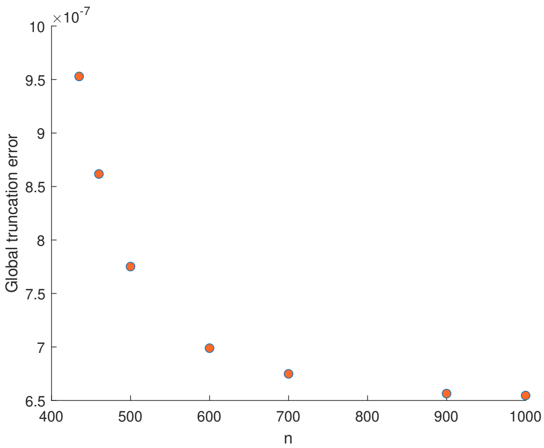

Figure 6: Global truncation errors in the numerical solution of $\vec{y}'(t) = A\vec{y}(t) + \vec{b}(t), \quad \vec{y}(0) = \vec{y}_0, \quad \text{for } t \in [0, 1]$ computed at increasing stability n number of time points using RK4.

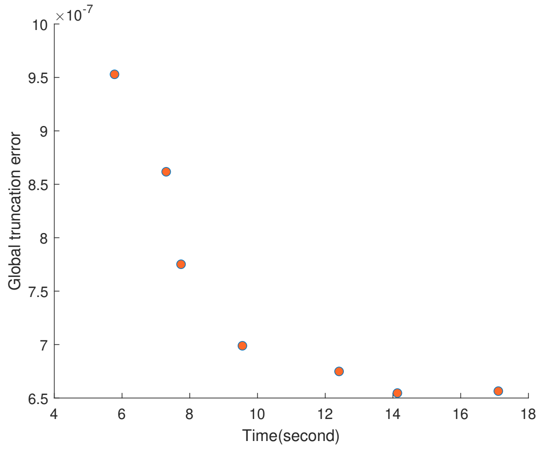

Figure 7: Time (seconds) required for RK4 to compute global truncation errors in the numerical solution of the system of ODEs.

increases, then the global truncation errors in the numerical solution at the final time decrease. This asymptotic behavior is in agreement with what Heath [10] describes how stable numerical methods should behave. From Figure 2 to Figure 11 we also see that the maximum accuracy of RK4, RK5, and RK8 is 10^{-7}. RK4, RK5, and RK8 produce smaller global truncation errors than the forward Euler method and Heun's method. The forward Euler method and Heun's method produce greater global truncation errors. Both the forward Euler method and Heun's method have the maximum accuracy 10^{-3}. Therefore, we can say that Rk4, RK5, and RK8 is more accurate than the forward Euler method and Heun's method. The forward Euler method and Heun's method are the least accurate explicit Runge-Kutta methods

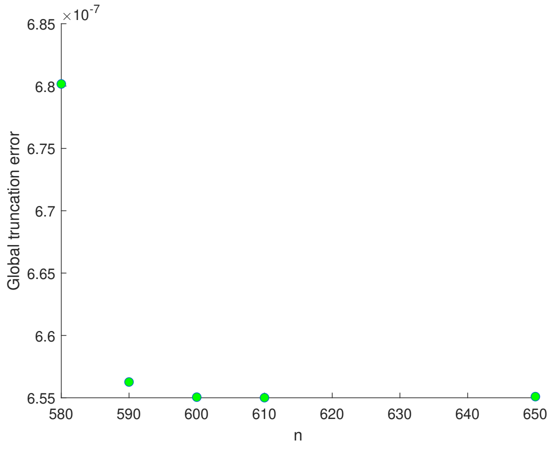

Figure 8: Global truncation errors in the numerical solution of $\vec{y}'(t) = A\vec{y}(t) + \vec{b}(t), \quad \vec{y}(0) = \vec{y}_0, \quad$ for $t \in [0, 1]$ computed at increasing stability n number of time points using RK5.

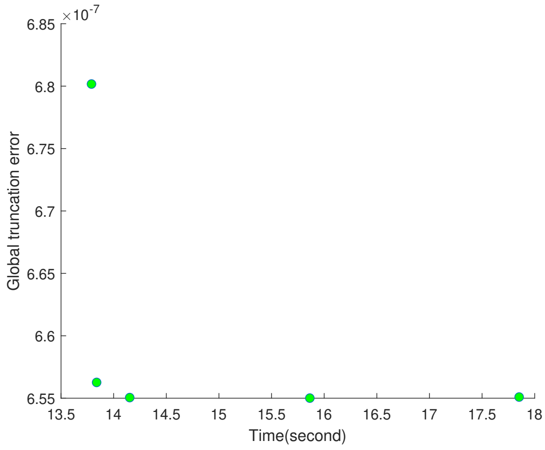

Figure 9: Time (seconds) required for RK5 to compute global truncation errors in the numerical solution of the system of ODEs.

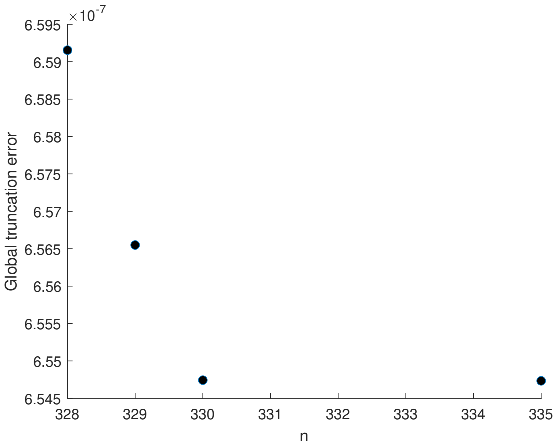

Figure 10: Global truncation errors in the numerical solution of $\vec{y}'(t) = A\vec{y}(t) + \vec{b}(t), \quad \vec{y}(0) = \vec{y}_0, \quad$ for $t \in [0, 1]$ computed at increasing stability n number of time points using RK8.

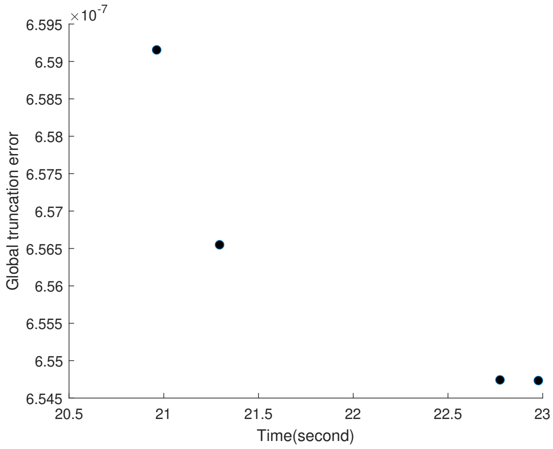

Figure 11: Time (seconds) required for RK8 to compute global truncation errors in the numerical solution of the system of ODEs.

investigated in this thesis. From this accuracy analysis, we can say that the RK4, RK5, and RK8 give more accurate results in the numerical solution which according to Söderlind [26] it is because they are higher-order explicit Runge-Kutta methods.

The results that are shown in Figure 2 and Figure 3 are obtained by running Script 11 in MATLAB. Figure 2 shows that when $n \geq 22000$ increases the global truncation errors decrease. Also, when $n \geq 22000$ increases, it takes more time for the forward Euler method to compute the global truncation errors as shown in Figure 3. Due to this high computational time, we stopped to increase the stability limit n of the forward Euler method at the stability limit $n = 600000$, which takes approximately 1 hour and 8 minutes for the forward Euler method to compute.

Furthermore, the results that are shown in Figure 4, and Figure 5 are obtainedby running Script 13 in MATLAB. For Heun's method, we stopped to increase the stability limit n at the stability limit n =300000 because it takes approximately 40 minutes for Heun's method to compute the global truncation errors which are high computational time. It also took too high computational time for the forward Euler method, therefore we can state that both the forward Euler method and Heun's method is inefficient. In Figure 4.4, Heun's method in the beginning when we increase the stability limit n, then the global truncation errors decrease fast. The global truncation errors of Heun's method start to be a constant at the stability limit $n = 7000$. As a consequence, we see in Figure 4 that the global truncation errors become flat because the stability limit $n = 7000$ is large. Consequently, the global truncation errors of Heun's method start to oscillate, which means that as the stability limit n increases, then the global truncation errors can decrease or increase a little bit but it will still be close to the previously computed global truncation error.

The results that are shown in Figure 6 and Figure 7 are obtained by running Script 15 in MATLAB. In Figure 6, we can see that at the stability limit n =1000 the global truncation errors of RK4 stop decreasing. Also, the results that are shown in Figure 8 and Figure 9 are obtained by running the Script 17 in MATLAB. In Figure 8, we see that at the stability limit $n = 700$ the global truncation errors of RK5 stop decreasing. Furthermore, the results that are shown in Figure 10 and Figure 11 are obtained by running the Script 19 in MATLAB. The global truncation errors of RK8 stop decreasing at the stability limit $n = 700$.

In Figure 11, we can, for example, take the global truncation error of magnitude 6.565×10^{-7}, which takes approximately 21.4 seconds for RK8 to compute this global truncation error. If we want to obtain this global truncation error using RK5 it will take approximately 13.8 seconds (see Figure 9), this means that less computational time is required for RK5. Therefore, RK5 is more efficient to compute this global truncation error than RK8. In Figure 7, we can see that for RK4,it takes approximately 12.5 seconds to compute this global truncation error, therefore it takes less computational time for RK4 than RK5 and RK8 to compute the global truncation error of magnitude 6.565×10^{-7}. We can easily compare the computational time required for RK4, RK5, and RK8 to compute the global truncation errors because the smallest global truncation errors of RK4, RK5 and RK8 are of the same magnitude 10^{-7} as we see in Figure 7, Figure, 9 and Figure 11. The smallest global truncation errors of the forward Euler method and Heun's method are of the same magnitude 10^{-3}. This means that the global truncation errors of the magnitude of the forward Euler method and Heun's method are greater than RK4, RK5, and RK8. This means that to get the same global truncation errors of magnitude 10^{-7} using the forward Euler method and Heun's method, then we have to run the Script 11 and Script 13 with much smaller step size. It takes high computational time for the forward Euler method and Heun's method to obtain the global truncation errors of magnitude 10^{-7}, therefore we did not obtain these results.

Moreover, Anidu et al. [1] describe that even though high-order Runge-Kutta methods such as RK4 require more computational effort than low-order Runge-Kutta methods, it is more efficient than low-order Runge-Kutta methods as the forward Euler method. This efficiency is shown in this section because it took more function evaluations to be computed per time step for RK4, RK5, and RK8.

However, RK4, RK5, and RK8 have shown to be more efficient to solve Equation (4) than the forward Euler method and Heun's method, where fewer function evaluations are required per time step.

Conclusion

In this paper, we have compared the performance of the forward Euler method, Heun's method, RK4, RK5, and RK8 in terms of accuracy, stability, and efficiency. We did this through the stability analysis, the convergence study, and the efficiency study. We can conclude from the stability analysis that the forward Euler method, Heun's method, RK4 and RK5 have smaller stability regions than the stability re- gion of RK8. The forward Euler method has the smallest stability region and RK8 has the largest stability region. Therefore, RK8 has shown to have better stability property than the forward Euler method, Heun's method, RK4, and RK5. Therefore, RK8 is the most stable explicit Runge-Kutta method for solving Equation
From the convergence study, RK8 has shown to be the most accurate explicit Runge-Kutta method for solving Equation (1). This is because RK8 hasa higher convergence rate p, therefore produces more accurate numerical solutionsof Equation (1) compared with the forward Euler method, Heun's method, RK4,and RK5 which are the less accurate explicit Runge-Kutta methods.
The forward Euler method is the least accurate explicit Runge-Kutta method investigated in this thesis. Furthermore, from the efficiency study, we can state that the forward Euler method and Heun's method are inefficient explicit Runge-Kutta methods. RK5 and RK8 are less efficient compared with RK4 because RK4 requires less computational time to compute global truncation errors in the numerical solution of Equation (4) than RK5 and RK8. RK5 and RK8 computed the global truncation errors more efficiently than the forward Euler method and Heun's method. RK4 has shown to be the most efficient explicit Runge-Kutta method for solving Equation (4).
Some of the results shown in the stability analysis, the convergence study, and the efficiency study did correspond to previous literature. It was difficult to find previous literature about the stability analysis and convergence of RK5 and RK8 as well as the efficiency properties of the five explicit Runge-Kutta methods for solving a system of ODEs. This means that the results shown in the stability analysis and the convergence study concerning RK5 and RK8 are new results that are derived in this paper.

References:

[1] A. O. Anidu, S. A. Arekete, A. O. Adedayo, and A. O. Adekoya. Dynamic computation of Runge-Kutta fourth-order algorithm for first- and second-order ordinary differential equation using java. *International Journal of Computer Science*, 12(13):211–218, 2015.

[2] U. M. Ascher and L. R. Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*. Siam, 1998.

[3] R. Ashino, M. Nagase, and R. Vaillancourt. Behind and beyond the matlab ode suite. *Computers & Mathematics with Applications*, 40(4-5):491–512, 2000.

[4] K. Atkinson, W. Han, and D. E. Stewart. *Numerical solution of ordinary differential equations*. John Wiley & Sons, 2011.

[5] S. Bu, W. Jung, and P. Kim. An error embedded Runge-Kutta method for initial value problems. *Kyungpook Mathematical Journal*, 56(2):311–327, 2016.

[6] J. C. Butcher. A history of Runge-Kutta methods. *Applied numerical mathematics*, 20(3):247–260, 1996.

[7] V. Chauhan and P. K. Srivastava. Computational techniques based on Runge-Kutta method of various order and type for solving differential equations. *International Journal of Mathematical, Engineering and Management Sciences*, 4(2):375–386, 2019.

[8] D. Gopal, V. Murugesh, and K. Murugesan. Numerical solution of second-order robot arm control problem using Runge-Kutta butcher algorithm. *International Journal of Computer Mathematics*, 83(3):345–356, 2006.

[9] A. Hasan. Numerical computation of initial value problem by various techniques. *Journal of Science and Arts*, 18(1):19–32, 2018.

[10] M. T. Heath. *Scientific computing: an introductory survey*. McGraw-Hill, 2002.

[11] D. Houcque. Applications of matlab: ordinary differential equations (ode). *Robert R. McCormick School of Engineering and Applied Science-Northwestern University, Evanston*, 2008.

[12] Md. A. Islam. A comparative study on numerical solutions of initial value problems (ivp) for ordinary differential equations (ode) with euler and Runge-Kutta methods. *American Journal of Computational Mathematics*, 5(3):393–404, 2015.

[13] S. M. Jung. Hyers-ulam stability of a system of first order linear differential equations with constant coefficients. *Journal of Mathematical Analysis and Applications*, 320(2):549–561, 2006.

[14] D. Ketcheson and A. Ahmadia. Optimal stability polynomials for numerical integration of initial value problems. *Communications in Applied Mathematics and Computational Science*, 7(2):247–271, 2013.

[15] S. Larsson and V. Thomée. *Partial differential equations with numerical methods*. Springer Science & Business Media, 2008.

[16] R. J. Leveque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*. Siam, 2007.

[17] K. Mattsson, F. Ham, and G. Iaccarino. Stable boundary treatment for the wave equation on second-order form. *Journal of Scientific Computing*, 41(3):366, 2009.

[18] D. Omale, P. B. Ojih, and M. O. Ogwo. Mathematical analysis of stiff and non-stiff initial value problems of ordinary differential equation using matlab. *International journal of scientific & engineering research*, 5(9):49–59, 2014.

[19] D. F. Papadopoulos and T. E Simos. The use of phase lag and amplification error derivatives for the construction of a modified Runge-Kutta Nyström method. In *Abstract and Applied Analysis*. Hindawi, 2013.

[20] J. S. C. Prentice. Stepsize selection in explicit Runge-Kutta methods for moderately stiff problems. *Applied Mathematics*, 2(6):711–717, 2011.

[21] U. A. M. Roslan, Z. Salleh, and A. Kılıçman. Solving zhou chaotic system using fourth-order Runge-Kutta method. *World Applied Sciences Journal*, 21(6):939–944, 2013.

[22] M. Schäfer. *Computational engineering: introduction to numerical methods*. Springer, 2006.

[23] W. E. Schiesser and G. W. Griffiths. *A compendium of partial differential equation models: method of lines analysis with Matlab*. Cambridge University Press, 2009.

[24] H. Séka and K. R. Assui. Order of the Runge-Kutta method and evolution of the stability region. *Ural Mathematical Journal*, 5(2):64–71, 2019.

[25] M. M. Stabrowski. An efficient algorithm for solving stiff ordinary differential equations. *Simulation Practice and Theory*, 5(4):333–344, 1997.

[26] G. Söderlind. *Numerical methods for differential equations*. Springer, 2017.

[27] P. J. Van der Houwen. The development of Runge-Kutta methods for partial differential equations. *Applied Numerical Mathematics*, 20(3):261–272, 1996.

[28] S. Wang. Numerical methods for ordinary differential equations. 2020.