وزارة التعليم العالي والبحث العلمي
جامعة بابل
كلية التربية للعلوم الصرفة
قسم الرياضيات

Research Title

numerical methods for solving ordinary differential equations

بحث تقدم به الطلب

سجاد علي ساجت

الى مجلس كلية التربية للعلوم الصرفة وهو جزء من

متطلبات نيل شهادة البكالوريوس في الرياضيات

بإشراف

م.م. فاطمة علي عبد الحسين البياتي

٢٠٢٤ م ١٤٤٥ هـ

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

{ يَٰٓأَيُّهَا ٱلَّذِينَ ءَامَنُوٓاْ إِذَا قِيلَ لَكُمۡ تَفَسَّحُواْ فِي ٱلۡمَجَٰلِسِ فَٱفۡسَحُواْ يَفۡسَحِ ٱللَّهُ لَكُمۡۖ وَإِذَا قِيلَ ٱنشُزُواْ فَٱنشُزُواْ يَرۡفَعِ ٱللَّهُ ٱلَّذِينَ ءَامَنُواْ مِنكُمۡ وَٱلَّذِينَ أُوتُواْ ٱلۡعِلۡمَ دَرَجَٰتٖۚ وَٱللَّهُ بِمَا تَعۡمَلُونَ خَبِيرٞ }

سورة المجادلة الاية (١١)

إهــــــــــــداء

اهدي هذا العمل المتواضع

إلى:

◊ الوالدين الكريمين حفظهما الله

◊ و إلى كل أفراد أسرتي

◊ و إلى روح جدي و جدتي رحمهما الله

◊ إلى كل الأصدقاء، و من كانوا برفقتي و مصاحبتي أثناء دراستي في الجامعة

◊ و إلى كل من لم يدخر جهدا في مساعدتي

◊ و إلى كل من ساهم في تلقيني ولو بحرف في حياتي الدراسية.

Chapter 1

1.1 Introducti0n

A differential equation is an equation that relates one or several functions and its derivatives [10]. Differential equations are used to model advanced systems, for instance, mechanical and electrical systems from fields such as biology, social sciences, engineering, and economics. Mechanical systems and electrical systems are two examples of physical systems that change over time [18]. Differential equations are used to describe a continuous change of physical systems mathematically. Partial differential equations (PDEs) and ordinary differential equations (ODEs) are two classes of differential equations that are used to model and characterize the behavior of physical systems. Parabolic PDEs exist in physical problems such as a heat conduction problem for solid bodies. Heat equations describe how the heat temperature distributes in, for instance, a rode as the time changes. Wave equations exist to be used to model physical problems such as physical systems, where wave motions are considered [23]. Furthermore, the above-mentioned examples of systems are complex which according to Houcque [11] are not possible to solve using analytical methods, therefore, numerical methods are used instead. Numerical methods are important to use to solve those differential equations whose exact solutions are not possible to obtain using analytical methods. Analytical methods such as the method of separation of variables give us exact solutions of simple PDEs and ODEs, whereas numerical methods give us approximations of the exact solution. Heath [10] emphasizes that numerical methods are viewed as a significant tool to use to solve problems for example in engineering and industry. Also, Omale et al. [18] describe that numerical methods have been used to solve differential equations of weather and climate forecasts. Numerical methods are explicit or implicit computed in one step or multiple steps. An explicit method computes the numerical solution at the next time point using the previous numerical solution at the previous time point. While an implicit method evaluates a function using the numerical solution at the next time point which is solved for. There are various numerical methods for solving PDEs and ODEs [10]. The method of lines is an example of a numerical method used to find numerical solutions of hyperbolic and parabolic PDEs by transforming the PDEs to a system of first-order ODEs which approximates the original PDEs. Finite difference methods, finite element methods, and finite volume methods are examples of numerical methods that are used to approximate the partial derivatives in the PDEs [23]. Finite difference methods such as Runge-Kutta methods are essential for finding approximate solutions of initial value problems of first-order ODEs [3]. High-order Runge-

Kutta methods are used to solve ODEs because of their high accuracy and efficiency [10]. Moreover, C. Runge and M. W. Kutta developed explicit and implicit RungeKutta methods [1]. The developments of low-order to high-order Runge-Kutta methods started in 1895. In 1895, C. Runge extended the forward Euler method to more elaborate Runge-Kutta methods with higher accuracy. Runge developed the second-order Runge-Kutta method in 1895. K. Heun introduced the third-order Runge-Kutta method in 1900 [6]. Runge-Kutta methods became important in the studies of explicit and implicit methods for solving ODEs through time discretization [7]. In 1901, W. Kutta introduced the fourth-order and the fifth-order Runge-Kutta methods. F. J. Nystr¨om developed Runge-Kutta methods to be used to solve systems of second-order differential equations [27]. In 1957, R. H. Merson proposed the idea to combine Runge-Kutta methods of different orders in Butcher tableau. In 1964, Butcher introduced the sixth-order Runge-Kutta method with seven stages. The seventh-order Runge-Kutta method with nine stages was recognized from 1968. In 1970, the eighth-order Runge-Kutta method with eleven stages was introduced by Curtis [6]. Runge-Kutta methods are active in research [1] and for the past several years, most research papers have relied on the derivation of new Runge-Kutta methods with higher derivatives to obtain more accurate Runge-Kutta methods [30]. In several papers, researchers have investigated, compared lower-order and high-order RungeKutta methods in terms of numerical properties, namely accuracy, stability, and efficiency. In Section 1.2, we review such research

papers that have been published for the past several years

1.2 Research question

we study the accuracy of the five explicit Runge-Kutta methods for solving an initial value problem of a first-order linear ODE by verifying their convergence rates. We derive the stability analysis to investigate which one of the five explicit Runge-Kutta methods has a better stability property for solving an initial value problem of a linear test equation.

1.3 Abstract

In Chapter 2, we present a theoretical background on PDEs and ODEs. Chapter 3 starts with a theoretical background on finite difference methods and it ends with the stability analysis for the five explicit Runge-Kutta methods.

Chapter 2

Differential equations

This chapter presents a theoretical background on PDEs and ODEs that are fundamental in this thesis

2.1 Ordinary differential equations

AN ordinary differential equation (ODE) is a differential equation involving an unknown function of derivatives with respect to one independent variable usually t. The order of an ODE is determined by the highest derivative in the ODE. The highest derivative in a first-order linear ODEisone,thereforethistypeofODEisoffirst-order[10].
Consider this first-order linear ODE taken from [28]

$$y'(t) = 5y(t), \tag{2.1}$$

where y is a dependent variable with respect to the independent variable t and y(t) is an unknown solution which we seek for. According to Wang [28] Equation (2.1) is simple, therefore we can use analytical methods to solve it analytically. The general solution which is a family of solutions of Equation (2.1) is

$$y(t) = Ce^{5t} ,$$

where C is a real constant [28]. If we instead want to find an exact solution of Equation (2.1), then we can impose the following initial condition to determine the value of C which according to [28] it is

$$y(0) = 2 ,$$

Wang [28] describes that the initial condition provide us with information that the initial value of y(t) at the initial time point t_0 = 0 is 2. We get that y(0) = Ce^0 = C = 2 and the exact solution that satisfies the initial condition of Equation (2.1) is

$$Y(t) = 2e^{5t}, \tag{2.2}$$

where Equation (2.2) is in agreement with [28]. The first-order ODE with an initial condition is called an initial value problem [28]. Wang [28] describes that a general form of an initial value problem is

$$y'(\mathrm{t}) = f(\mathrm{t,y(t)}), \tag{2.3}$$
$$\mathrm{Y(a) = c}, \ for\ \mathrm{t} \in [\![a, b]\!],$$

where f(t, y(t)) is a function which depend on t and y. From the initial condition in Equation (2.3), we know that the initial value is c. Also, the time interval is . t ∈ [a, b], the initial time point is t_0 = a and the final time point is t_n = b, where n is the number of time points. Heath [10] describes that in most applications of differential equations there exists more than one ODE which is transformed into a system of ODEs. Jung [13] introduce a system of k first-order linear ODEs with coefficients written in this

$$\begin{cases} y_1'(t) = a_{11}y_1(t) + a_{12}y_2(t) + \cdots + a_{k1}y_k(t) + b_1(t)), \\ y_2'(t) = a_{21}y_1(t) + a_{22}y_2(t) + \cdots + a_{2k}y_k(t) + b_2(t)), \\ \vdots \\ y_k'(t) = a_{k1}y_1(t) + a_{2k}y_2(t) + \cdots + a_{kk}y_k(t) + b_k(t)), \end{cases}$$

where a_{ij} (t) and b_j (t) are known functions on the time interval t ∈ [a, b], i = 1, 2, · · · , n and i = 1, 2, · · · , n. The column vector is $\vec{y}$(t) of the unknown functions y_1(t), · · · , y_k(t) which are dependent on t, where k = 1, 2, · · · , n [13]. Jung [13] write this system of k first-order linear ODEs with an initial condition in matrix notation form as

$$\vec{y}'(t) = A\vec{y}(\mathrm{t}) + \vec{b}(t), \tag{2.4}$$
$$\vec{y}(t) = \vec{y}_0\,,\ \text{for} \in [a, b],$$

In Equation (2.4), A is a n × n matrix, $\vec{y}'$ (t) is a column vector and $\vec{b}$(t) is another column vector. The column vectors depend on t and the column vectors have a length n. If $\vec{b}$(t) has length zero, then it is a zero column vector, and Equation (2.4) becomes homogeneous. In Equation (2.4), we have the initial value vector $\vec{y}_0$. In the efficiency study, we use numerical methods for ODEs to solve an initial value problem of a homogenous system of first-order linear ODEs formulated as Equation (2.4)

Chapter 3

Numerical methods for ODEs

In this chapter, we present a theoretical background on finite difference methods and analyze the stability of the five explicit Runge-Kutta methods.

3.1 Finite difference methods

Finite difference methods solve for instance initial value problems of first-order ODEs on the time interval t ∈ [a, b]. The idea of finite difference methods for an initial value problem of a first-order ODE is to divide the time interval t ∈ [a, b] into n subintervals and approximate the first-order derivative in the initial value problem for every discrete time point t_j in t ∈ [a, b]. In time discretization of initial value problems of first-order ODEs, we obtain discrete solutions that are approximate values of the exact solution of the first-order ODE at discrete time points. Time-discretization is a procedure for solving initial value problems of first-order linear ODEs [31].

Now we show how to proceed with time-discretization of Equation (2.3). We want to find an approximate solutions of Equation (2.3) by discretizing the time interval t ∈ [a, b] into n subintervals using n + 1 points [28]

$$a = t_0 < t_1 < t_2 < \cdots < t_n = b\ .$$

Wang [28] describes that the length of each n subinterval is the step size $h_j = t_j - t_{j-1}$, where j = 1, 2, · · · , n. In this procedure, the previous computed numerical solution y_{j-1} at the previous discrete time point y_{j-1} is used to compute the next numerical solution y_j at the next discrete time point tj , where j = 0, 1, · · · , n. An approximate solution of Equation (2.3) at the discrete time point $t_j$is $y_j \approx y(t_{j)}$. The numerical solution at t_0 is $y_0 = y(t_o)$ = c. We can use the numerical solution y_0 = c to compute the next numerical solution y_1 at the next discrete time point t_j. We can use this procedure to get the numerical solution y_n at the final time point t_n. One-step methods such as explicit Runge-Kutta methods use the information from the previously computed numerical solution at a discrete time point to compute the next numerical solution at the next discrete time point. Explicit Runge-Kutta methods are finite difference methods for solving initial value problems of first-order ODEs. Explicit Runge-Kutta methods involve s stages which is equal to the number of function evaluations of the function f(t, y(t)) in Equation (2.3) needed to advance the numerical solution in one time step [10].

3.1.1 Errors

Errors occur when one performs numerical computation on a computer. Errors are classified into round-off errors and truncation errors. Round-off errors are created from the representation of numbers that is approximated. We get round-off errors from numbers such as fractional numbers, π, and the square root of numbers which can-not be represented exactly in the computer. Truncation errors are created when a numerical method is approximating an exact solution. Truncation errors are the difference between the approximate solution and the exact solution. Truncation errors are classified into local truncation errors and global truncation errors [10]. Heath [10] describes that the local truncation error ℓ_j is

$$\ell_j = y_j - u_{j-1}(t_j) \, ,$$

where j = 1, 2, · · · , n, the numerical solution of an initial value problem of a first-order ODE is y_j and u_{j-1}(t) is the solution of the ODE that passes through the point before, which is (t_{j-1}, y_{j-1}).

According to Heath [10] the global truncation error e_n is

$$e_n = y_n - y(t_n)$$

the difference between the numerical solution y_n and the exact solution y(t_n) at the final time point t_n. The differences between a local truncation error and a global truncation error are that we compute the local truncation error in the numerical solution of an ODE at a discrete-time point t_j. While we compute the global truncation error at the final time point t_n and we obtain a total error at the final time point t_n [28]. In this thesis, we only compute global truncation errors because in both the convergence study and the efficiency study we want to compute global truncation errors at the final time point t_n. Heath [10] states that global truncation errors are essential to study when evaluating the performance of numerical methods

3.1.2 Numerical properties

Accuracy, stability, and efficiency are three numerical properties that are used to determine the performance of numerical methods [10].
The following definition of accuracy is from [10].

Definition 3.1.1. The accuracy of a numerical method is said to be of order p if

$$\ell_j = \mathcal{O}(h_j^{p+1}).$$

The motivation for this definition, with the order of accuracy one less than the exponent of the step size in the local error, is that if the local error is $\mathcal{O}(h_j^{p+1}$), then the local error per unit step, $\frac{\ell_j}{h_j}$, is $\mathcal{O}(h_j^p)$, and it can be shown that under reasonable conditions the global error $e_n\ is\ \mathcal{O}(h^p$), where h is the average step size.

Leveque [16] describe that $\mathcal{O}$ is a big-oh notation and it is also known as an asymptotic notation which describe a function that is bounded asymptotically by upper bounds. In Definition 3.1.1, j = 1, 2, · · · , n and the global truncation error 16 $\mathcal{O}(h^p$) is independent of h. We expect the following from a numerical method that has order of accuracy p

$$e^h = Ch^p + o(h^p) when\ h \to 0\ ,$$

where e^h is an error on a grid of M points and C is in this case an error constant. The little-oh notation o describe a function that is not bounded asymptotically by upper bounds [16]. According to Leveque [16] if h is too small, then

$$e^h \approx Ch^p.$$

A refinement of the grid by a factor of 2, we get.

$$e^{\frac{h}{2}} \approx C\left(\frac{h}{2}\right)^p$$

Other factors can certainly be used as well to refine the grid, but refining a grid by a factor of 2 is a standard way. The ratio of the errors e^h and $e^{\frac{h}{2}}$ is .

$$\frac{e^h}{e^{\frac{h}{2}}} \approx 2^p$$

where p is a positive integer and the errors decrease approximately by a factor of 2^p . Hence,

$$p \approx log_2\left(\frac{e^h}{e^{\frac{h}{2}}}\right).$$

Furthermore, Leveque [16] describes that we can estimate p with two grid spa-cings h_1 and h_2 as the following

$$p \approx \frac{log\left(\frac{e^{h_1}}{e^{h_2}}\right)}{log\left(\frac{h_1}{h_2}\right)}\ , \tag{3.1}$$

If the positive integer p > 0, then the global truncation error $\mathcal{O}(h^p\ ) \to 0$, when h → 0, and the numerical method converges with the convergence rate p. The rate p of convergence is known as the order of accuracy of a numerical method. The convergence rate p shows how fast the numerical method converges to the exact solution when the step size h → 0. We can measure the accuracy of a numerical method by verifying the convergence rate p of a numerical method [28].

According to Söderlind [26], it is not possible to obtain accuracy in numerical solutions if a numerical method is not stable. For this reason, a numerical method must be stable too. The following definition of stability is from [10]

Definition 3.1.2. numerical method is said to be **stable** if small perturbations do not cause the resulting numerical solutions to diverge away without bound.

Perturbations are created in numerical computation. Small perturbations are due to round-off errors or truncation errors in the initial data. A small perturbation is when a small change in the initial value of an ODE leads to often a small change in the numerical solutions of the ODE. We are perturbing the initial value by doing small changes to them. A stable numerical method is not sensitive to small perturbations, therefore the numerical solutions do not diverge from the exact solution. For a stable numerical method, the errors in the numerical solutions decrease to zero. Small perturbations to the numerical solutions of a stable ODE get diminished as the time increases because the numerical solution curves of the ODE converge to the exact solution of the ODE. Consequently, the numerical solutions will bound the exact solution.

Furthermore, an unstable numerical method is sensitive to small perturbations, where a small change in the initial value does result in a major change in the numerical solution which causes the numerical solutions to diverge from the exact solution. Small perturbations to a numerical solution of an unstable ODE will grow as the time increases because the numerical solution curves of the ODE diverge. If the numerical solutions of an unstable ODE diverge from the exact solution, then the numerical solutions are unbounded the exact solution [10].

Furthermore, an unstable numerical method is sensitive to small perturbations, where a small change in the initial value does result in a major change in the numerical solution which causes the
numerical solutions to diverge from the exact solution.Small perturbations to a numerical solu
of an unstable ODE will grow as the time increases because the numerical solution curves

of the ODE diverge. If the numerical solutions of an unstable ODE diverge from the exact solution, then the numerical solutions are unbounded the exact solution [10].

Definition 3.1.3.Numerical efficiency means here the combination of short computational time and an acceptable error level.

An important aspect of numerical methods is to verify how much computational time is required for numerical methods to compute errors [22]. We can measure the efficiency of numerical methods by computing errors in the numerical solution of ODEs and computational time [25]. Several researchers have improved the efficiency of Runge-Kutta methods for solving ODEs by decreasing the number of function evaluations

3.2 Stability of explicit Runge-Kutta methods

Söderlind [26] emphasizes that the stability property has a crucial role in solving first-order ODEs using numerical methods. We can verify the stability of numerical methods by analyzing how the numerical methods behave on a linear test equation. In this section, we derive the stability analysis for the five explicit Runge-Kutta methods for solving a linear test equation from [28] defined as

$$y'(t) = -\boldsymbol{\lambda} y(t), \qquad (3.2)$$
$$Y(0) = 1 \; t \in [0, T],$$

where λ ∈ C.

We choose Equation (3.2) to be considered in the stability analysis for the five explicit Runge-Kutta methods because it is a simple first-order ODE which according to Söderlind [26] one choose in the stability analysis to be able to solve it analytically by pen and paper

The known exact solution of Equation (3.2) is

$$Y(t)=e^{-\lambda t}\,, \qquad (3.3)$$

The exact solution (3.3) is also an exponential solution of Equation (3.2) which decrease exponentially to zero when $t \to \infty$ and Re(λ) > 0. The notation, Re(λ) > 0 means positive real values of λ on the real coordinate (Re) axis in a complex plane [28]. A complex plane also consists of an imaginary coordinate (Im) axis which visualizes complex numbers z = hλ. A numerical method is stable for different values of h$\boldsymbol{\lambda}$, where h is the step size. Different values of hλ are bounded in a region called stability region which we can visualize in a complex plane. An numerical method is stable if |S(z)|< 1, where S(z) is a stability function. The stability function S(z) is a series in power of z which approximate the exponential solution of a linear test equation [20].

Furthermore, in Section 3.2.1 to Section 3.2.5, we analyze each explicit Runge-Kutta method in terms of stability for solving Equation (3.2). We obtain a stability condition for each explicit Runge-Kutta method in the stability analysis. In Section 3.2.6, we plot the stability conditions to visualize the stability regions of the five explicit Runge-Kutta methods in a complex plane. Lastly, we compare the stability regions to determine which explicit Runge-Kutta method has the smallest and the largest stability region.

3.2.1 The forward Euler method

Atkinson et al. [4] describe that the forward Euler method is a first-order Runge-Kutta method and it is defined as

$$y_j = y_{j-1} + hf(t_{j-1}, y_{j-1}),$$

The forward Euler method is used to compute the numerical solution yj at the discrete time point tj . The forward Euler method is a one-stage explicit RungeKutta method which means that to advance the numerical solution in one time step, then one function evaluation is needed [28].

We can now derive the stability analysis for the forward Euler method for solving Equation (3.2) as follows

$$y_j = y_{j-1} + h(-\lambda y_{j-1}),$$
$$y_j = (1 - h\boldsymbol{\lambda})\boldsymbol{y_{j-1}},$$

where y_j , y_{j-1} are numerical solutions and 1 − hλ is a constant [28]. Furthermore,we have

$$y_{j-1} = (1 - h\lambda)y_{j-2},$$

$$y_{j-2} = (1 - h\lambda)y_{j-3},$$
$$\vdots$$
$$y_1 = (1 - h)y_0 = (1 - h\boldsymbol{\lambda}).$$

Consequently, we get the following numerical solution.

$$y_j = (1 - h\lambda)^j, \tag{3.4}$$

According to Wang [28] the numerical solution (3.4) of Equation (3.2) obtained by the forward Euler method is stable because the numerical solution (3.4) converges to zero. The numerical solution (3.4) decrease in time t as j → ∞. To satisfy this, therefore we need to require the following stability condition

$$|1 - h\lambda| < 1, \tag{3.5}$$

The stability condition (3.5) constraints the value of the step size h. The forward Euler method is stable when it satisfies the stability condition (3.5). The stability condition (3.5) is an inequality less than 1, therefore we can simplify the stability condition by removing the absolute value and we get

$$-1 < 1 - h\lambda < 1 \to 0 < h\lambda < 2 \to 0 < h < \frac{2}{\lambda}.$$

The forward Euler method is stable for solving Equation (3.2) if it satisfies the stability condition $h < \frac{2}{\lambda}$ if λ ∈ R. The inequality 0 < hλ satisfies if λ > 0 according to [28].

We can now investigate how the numerical solution of Equation (3.2) changes for different values of hλ. Assume that we have h = 2, h = 0.1 and λ in Equation (3.2) is λ = 5. If h = 0.1 and λ = 5, then hλ = 0.5 and hλ < 2 which means that the forward Euler method is stable. We can say that the forward Euler method is stable when h < $\frac{2}{\lambda}$ for λ ∈ ℝ. If h = 2 and λ = 5, then hλ =10 and hλ > 2, which means that the forward Euler method is unstable, therefore the numerical solution diverges to infinity.

Furthermore, the forward Euler method has the following stability function S(hλ)

$$S(h\boldsymbol{\lambda})=1-h\boldsymbol{\lambda}.$$

This stability function alternates in sign

Wang [28] has also obtained the stability condition (3.5) for the forward Euler method for solving the Equation (3.2). The stability condition for the forward Euler method presented in [2] does not equal the stability condition (3.5) because the test equation solved in [2] is not the same as Equation (3.2). The test equation solved in [2] is

$$y'(t) = \boldsymbol{\lambda y(t)}, \qquad (\mathbf{3.6})$$

therefore we do not obtain the same stability condition for the forward Euler method as in [2].

3.2.2 Heun's method

Witty [29] describes Heun's method is a second-order Runge-Kutta method and it is defined as .

$$\boldsymbol{y_j = y_{j-1} + \frac{1}{2}h(k_1 + k_2)},$$
$$\boldsymbol{k_1 = f(t_{j-1}, y_{j-1})},$$
$$\boldsymbol{k_2 = f(t_{j-1} + h, y_{j-1} + hk_1)}.$$

Heun's method is a two-stage explicit Runge-Kutta method which means that to advance the numerical solution in one time step, then two function evaluations are needed. Heun's method consists of one more function evaluation than the forward Euler method. We can now derive the stability analysis for Heun's method for solving Equation (3.2) as the following

$$k_1 = (-\boldsymbol{\lambda})y_{j-1}\,,$$
$$k_2 = (-\boldsymbol{\lambda}+\tfrac{1}{2}h\lambda^2)\, y_{j-1}\,,$$
$$y_j = \left(1 - h\boldsymbol{\lambda} + \tfrac{\mathbf{1}}{\mathbf{2}}\boldsymbol{h}^2\boldsymbol{\lambda}^2\right)\boldsymbol{y}_{\boldsymbol{j-1}},$$

Consequently, we get the numerical solution as follows

$$y_j = \left(1 - h\lambda + \frac{1}{2}h^2\lambda^2\right)^j , \tag{3.7}$$

The numerical solution (3.7) decrease in time t as j → ∞. To satisfy this, we require the following stability condition

$$\left|1 - h\lambda + \tfrac{1}{2}h^2\lambda^2\right| < 1 \tag{3.8}$$

If we simplify the stability condition (3.8) by removing the absolute value we get,

$$-1 < 1 - h\boldsymbol{\lambda} + \tfrac{\mathbf{1}}{\mathbf{2}}\boldsymbol{h}^2\boldsymbol{\lambda}^2 < \mathbf{1} \rightarrow \mathbf{0} < \boldsymbol{h\lambda} - \tfrac{\mathbf{1}}{\mathbf{2}}\boldsymbol{h}^2\boldsymbol{\lambda}^2 < \mathbf{2} \rightarrow \mathbf{0} < \boldsymbol{h\lambda} < \mathbf{2} \rightarrow \mathbf{0} < \boldsymbol{h} < \tfrac{\mathbf{2}}{\boldsymbol{\lambda}}$$

,

where h < $\frac{2}{\lambda}$ is the stability condition for Heun's method when λ ∈ R and h > 0. This stability condition is the same as the stability condition for the forward Euler method. The stability function of Heun's method is .

$$S(h\boldsymbol{\lambda}) = \mathbf{1} - \boldsymbol{h\lambda} + \tfrac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{h\lambda})^{\mathbf{2}}\,.$$

The stability function of the forward Euler method and Heun's method are related because the first two terms of the stability function of Heun's method are the same as the terms of the stability function of the forward Euler method. The stability function of Heun's method involves three terms, whereas the stability function of the forward Euler method involves two terms.

3.2.3 The fourth-order Runge-Kutta method

Roslan [21] describes that the fourth-order Runge-Kutta method (RK4) is also known as the
classical Runge-Kutta method defined as

$$y_j = y_{j-1} + \tfrac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4),$$
$$k_1 = f\left(t_{j-1}\,, y_{j-1}\right),$$

$$k_2 = f\left(t_{j-1}, \frac{1}{2}h, y_{j-1} + \frac{1}{2}hk_1\right),$$
$$k_3 = f\left(t_{j-1} + \frac{1}{2}h, y_{j-1} + \frac{1}{2}hk_2\right),$$
$$k_4 = f\left(t_{j-1} + h, y_{j-1} + hk_3\right),$$

RK4 is a four-stage explicit Runge-Kutta method which means that to advance the numerical solution in one time step four function evaluations are needed. For RK4 two more function evaluations are needed than for Heun's method. We can now derive the stability analysis for RK4 for solving Equation (3.2) and we get

$$k_1 = (-\lambda)y_{j-1},$$
$$k_2 = \left(-\lambda + \frac{1}{2}h\lambda^2\right)y_{j-1},$$
$$k_3 = \left(-\lambda + \tfrac{1}{2}hh^2 - \tfrac{1}{4}hh^3\right)y_{j-1},$$
$$k_4 = \left(-\lambda + h\lambda^2 - \frac{1}{2}h^2\lambda^3 + \frac{1}{4}h^3\lambda^4\right)y_{j-1},$$
$$y_j = \left(1 - h\lambda + \frac{1}{2}h^2\lambda^2 - \frac{1}{6}h^3\lambda^3 + \frac{1}{24}h^4\lambda^4\right)y_{j-1},$$

Consequently, we get the following numerical solution

$$y_j = \left(1 - h\lambda + \tfrac{1}{2}h^2\lambda^2 - \tfrac{1}{6}h^3\lambda^3 + \tfrac{1}{24}h^4\lambda^4\right)^j, \tag{3.9}$$

which decrease in time t as j $\rightarrow\infty$. To satisfy this, we require the following stability condition

$$\left|1 - h\lambda + \tfrac{1}{2}h^2\lambda^2 - \tfrac{1}{6}h^3\lambda^3 + \tfrac{1}{24}h^4\lambda^4\right| < 1.$$

The stability function of RK4 is

$$S(h) = 1 - h\lambda + \tfrac{1}{2}(h\lambda)^2 - \tfrac{1}{6}(h\lambda)^3 + \tfrac{1}{24}(h\lambda)^4.$$

This stability function involves five terms which means two more terms than the terms of the stability function of Heun's method. The first three terms of this stability function are also involved in the stability function of Heun's method.

3.2.4 The fifth-order Runge-Kutta method

Gopal et al. [8] define the fifth-order Runge-Kutta method (RK5) as

$$y_j = y_{j-1} + \frac{1}{90}h(7k_1 + 32k_3 + 12k_4 + 32k_5 + 7k_6),$$
$$k_1 = f\left(t_{j-1}, y_{j-1}\right),$$
$$k_3 = f\left(t_{j-1} + \frac{1}{4}h, y_{j-1} + \frac{1}{4}hk_1\right),$$

$$k_3 = f\left(t_{j-1} + \frac{1}{4}h\,, y_{j-1} + \frac{1}{8}hk_1 + \frac{1}{8}hk_2\right),$$
$$k_4 = f\left(t_{j-1} + \frac{1}{2}h\,, y_{j-1} - \frac{1}{2}hk_2 + \frac{1}{8}hk_3\right),$$
$$k_5 = f\left(t_{j-1} + \frac{3}{4}h\,, y_{j-1} + \frac{3}{16}hk_1 + \frac{9}{16}hk_4\right),$$
$$k_6 = f\left(t_{j-1} + h\,, y_{j-1} - \frac{3}{7}hk_1 + \frac{2}{7}hk_2 + \frac{12}{7}hk_3 - \frac{12}{7}hk_4 + \frac{8}{7}hk_5\right).$$

a six-stage explicit Runge-Kutt method which means six function evaluations are needed to advance the numerical solution in one time step. We see that RK5 consists of two more function evaluations than RK4. We can now derive the stability analysis for RK5 for solving the Equation (3.2) and we getWe can now derive the stability analysis for RK5 for solving the Equation (3.2) and we get

$$k_1 = (-\lambda)y_{j-1},$$
$$k_2 = \left(-\lambda + \frac{1}{4}h\lambda^2\right)y_{j-1}\,,$$
$$k_3 = \left(-\lambda + \frac{1}{4}h\lambda^2 - \frac{1}{32}h^2\lambda^3\right)y_{j-1}\,,$$
$$k_4 = \left(-\lambda + \frac{1}{2}h\lambda^2 - \frac{1}{8}h^2\lambda^3 + \frac{1}{32}h^3\lambda^4\right)y_{j-1}\,,$$
$$k_5 = \left(-\lambda + \frac{12}{16}h\lambda^2 - \frac{9}{32}h^2\lambda^3 + \frac{9}{128}h^3\lambda^4 - \frac{9}{514}h^4\lambda^5\right)y_{j-1}\,,$$
$$k_6 = \left(-\lambda + h\lambda^2 - \frac{1}{2}h^2\lambda^3 + \frac{9}{56}h^3\lambda^4 - \frac{3}{112}h^4\lambda^5 + \frac{72}{3584}h^5\lambda^6\right)y_{j-1}\,,$$
$$y_j = \left(1 - h\lambda + \frac{1}{2}h^2\lambda^2 - \frac{1}{6}h^3\lambda^3 + \frac{1}{24}h^4\lambda^4 - \frac{1}{120}h^5\lambda^5 + \frac{1}{640}h^6\lambda^6\right)y_{j-1}\,,$$

As a consequence, we get the following numerical solution

$$y_j = \left(1\lambda - h\lambda + \tfrac{1}{2}h^2\lambda^2 - \tfrac{1}{6}h^3\lambda^3 + \tfrac{1}{24}h^4\lambda^4\right)^j\,, \qquad (3.10)$$

which decrease in time t as j → ∞. To satisfy this, we require the following stability condition

$$\left|1 - h\lambda + \tfrac{1}{2}h^2\lambda^2 - \tfrac{1}{6}h^3\lambda^3 + \tfrac{1}{24}h^4\lambda^4 - \tfrac{1}{120}h^5\lambda^5 + \tfrac{1}{640}h^6\lambda^6\right| < 1\,.$$

The stability function of RK5 is

$$S(h\lambda) = 1 - h\boldsymbol{\lambda} + \tfrac{1}{2}h^2\lambda^2 - \tfrac{1}{6}h^3\lambda^3 + \tfrac{1}{24}h^4\lambda^4 - \tfrac{1}{120}h^5\lambda^5 + \tfrac{1}{640}h^6\lambda^6\,.$$

The stability function of RK5 involves seven terms which means three more terms than the terms in the stability function of RK4. The first three terms in this stability function are also involved in the stability function of RK4 and Heun's method but not in the stability function of the forward Euler method.

3.2.5 The eighth-order Runge-Kutta method

The eighth-order Runge-Kutta method (RK8) is defined in [5] with Butcher tableau. The RK8 is

$$y_j = y_{j-1} + h\left(\frac{34}{105}k_6 + \frac{9}{35}k_7 + \frac{9}{35}k_8 + \frac{9}{280}k_9 + \frac{9}{280}k_{10} + \frac{41}{840}k_{12} + \frac{41}{840}k_{13}\right),$$

$$k_2 = f\left(t_{j-1} + \frac{2}{27}h\,, y_{j-1} + \frac{2}{27}hk_1\right),$$

$$k_3 = f\left(t_{j-1} + \tfrac{1}{9}h\,, y_{j-1} + \tfrac{1}{36}h(k_1 + 3k_2)\right),$$

$$k_4 = f\left(t_{j-1} + \tfrac{1}{6}h\,, y_{j-1} + \tfrac{1}{24}h(k_1 + 3k_3)\right),$$

$$k_5 = f\left(t_{j-1} + \tfrac{5}{12}h\,, y_{j-1} + \tfrac{1}{48}h(20k_1 - 75k_3 + 75k_4)\right),$$

$$k_6 = f\left(t_{j-1} + \frac{1}{2}h\,, y_n + \frac{1}{20}h(k_1 + 5k_4 + 4k_5)\right),$$

$$k_7 = f\left(t_{j-1} + \frac{5}{6}h\,, y_{j-1} + \frac{1}{180}h(-25k_1 + 125k_4 - 260k_5 + 250k_6)\right),$$

$$k_8 = f\left(t_{j-1} + \frac{1}{6}h\,, y_{j-1} + h\left(\frac{31}{300}k_1 + \frac{61}{225}k_5 - \frac{2}{9}k_6 + \frac{13}{900}k_7\right)\right),$$

$$k_9 = f\left(t_{j-1} + \frac{2}{3}h\,, y_{j-1} + h\left(2k_1 - \frac{53}{6}k_4 + \frac{704}{45}k_5 - \frac{107}{9}k_6 + \frac{67}{90}k_7 + 3k_8\right)\right),$$

$$k_{10} = f\left(t_{j-1} + \tfrac{1}{3}h\,, y_{j-1} + h\left(-\tfrac{91}{108}k_1 + \tfrac{23}{108}k_4 - \tfrac{976}{135}k_5 + \tfrac{311}{54}k_6 - \tfrac{19}{60}k_7 + \tfrac{17}{6}k_8 - \tfrac{1}{12}k_9\right)\right)$$

$$k_{11} = f\left(t_{j-1} + h\,, y_{j-1} + h\left(\tfrac{3283}{4100}k_1 - \tfrac{341}{164}k_4 + \tfrac{4496}{1025}k_5 - \tfrac{301}{82}k_6 + \tfrac{2133}{4100}k_7 + \tfrac{45}{82}k_8 + \tfrac{45}{164}k_9 + \tfrac{18}{41}k_{10}\right)\right),$$

$$k_{12} = f\left(t_{j-1}\ , y_{j-1} + h\left(\frac{3}{205}k_1 - \frac{6}{41}k_6 - \frac{3}{205}k_7 - \frac{3}{41}k_8 - \frac{3}{41}k_9 + \frac{6}{41}k_{10}\right)\right),$$

$$k_{13} = f\left(t_{j-1} + h\ , y_{j-1}\ + \left(-\frac{1777}{4100}k_1 - \frac{314}{164}k_4 + \frac{4496}{1025}k_5 - \frac{289}{82}k_6 + \frac{2193}{4100}k_7 + \frac{51}{82}k_8 + \frac{33}{164}k_9 + \frac{12}{41}k_{10} + k_{12}\right)\right),$$

RK8 is a thirteen-stage explicit Runge-Kutta method which means that thirteen function evaluations are needed to advance the numerical solution in one time step. RK8 consists of nine more function evaluations than RK5. We can now derive the stability analysis for RK8 for solving Equation (3.2) and we get

$$k_1 = (-\lambda)y_{j-1}\ ,$$

$$k_2 = \left(-\lambda + \frac{2}{27}h\lambda^2\right)y_{j-1},$$

$$k_3 = \left(-\lambda\ + \frac{1}{9}h\lambda^2 - \frac{1}{162}h^2\lambda^3\right)y_{j-1}\ ,$$

$$k_4 = \left(-\lambda\ + \frac{1}{6}h\lambda^2 - \frac{1}{72}h^2\lambda^3 + \frac{1}{296}h^3\lambda^4\right)y_{j-1}\ ,$$

$$k_5 = \left(-\lambda\ + \frac{5}{12}h\lambda^2 - \frac{25}{288}h^2\lambda^3 + \frac{125}{10368}h^3\lambda^4 - \frac{25}{20736}h^4\lambda^5\right)y_{j-1}\ ,$$

$$k_6 = \left(-\lambda\ + \frac{1}{2}h\lambda^2 - \frac{1}{8}h^2\lambda^3 + \frac{1}{48}h^3\lambda^4 - \frac{1}{384}h^4\lambda^5 + \frac{5}{20736}h^5\lambda^6\right)y_{j-1}\ ,$$

$$k_7 = \left(-\lambda\ + \frac{5}{6}h\lambda^2 - \frac{25}{72}h^2\lambda^3 + \frac{125}{1296}h^3\lambda^4 - \frac{625}{31104}h^4\lambda^5 + \frac{875}{279936}h^5\lambda^6 - \frac{625}{1119744}h^6\lambda^7\right)y_{j-1}\ ,$$

$$k_8 = \left(-\lambda\ + \frac{1}{6}h\lambda^2 - \frac{1}{72}h^2\lambda^3 + \frac{1}{1296}h^3\lambda^4 - \frac{1}{31104}h^4\lambda^5 + \frac{43}{1119744}h^5\lambda^6 + \frac{85}{10077696}h^6\lambda^7 + \frac{325}{40310784}h^7\lambda^8\right)y_{j-1}\ ,$$

$$k_9 = \left(-\lambda\ + \frac{2}{3}h\lambda^2 - \frac{2}{9}h^2\lambda^3 + \frac{4}{81}h^3\lambda^4 - \frac{2}{243}h^4\lambda^5 + \frac{1655}{559872}h^5\lambda^6 + \frac{4279}{10077696}h^6\lambda^7 + \frac{7865}{20155392}h^7\lambda^8 - \frac{325}{13436928}h^8\lambda^9\right)y_{j-1}\ ,$$

$$k_{10} = \left(-\lambda\ + \frac{1}{3}h\lambda^2 - \frac{1}{18}h^2\lambda^3 + \frac{1}{162}h^3\lambda^4 - \frac{1}{1944}h^4\lambda^5 - \frac{757}{1119744}h^5\lambda^6 - \frac{439}{1679616}h^6\lambda^7 - \frac{3331}{20155392}h^7\lambda^8 + \frac{65}{6718464}h^8\lambda^9 - \frac{325}{161243136}h^9\lambda^{10}\right)y_{j-1}\ ,$$

$$k_{11} = \left(-\lambda\ + h\lambda^2 - \frac{1}{2}h^2\lambda^3 + \frac{1}{6}h^3\lambda^4 - \frac{1}{24}h^4\lambda^5 + \frac{14767}{1700352}h^5\lambda^6 - \frac{6511}{5101056}h^6\lambda^7 + \frac{53}{186624}h^7\lambda^8 - \frac{7151}{183638016}h^8\lambda^9 + \frac{65}{27205632}h^9\lambda^{10} - \frac{325}{367276035}h^{10}\lambda^{11}\right)y_{j-1}\ ,$$

$$k_{12} = \Big(-\lambda + \frac{10}{566784}h^6\lambda^7 - \frac{17}{45909504}h^7\lambda^8 - \frac{2081}{550914048}h^8\lambda^9 + \frac{65}{183638016}h^9\lambda^{10} - \frac{325}{1101828096}h^{10}\lambda^{11}\Big) y_{j-1}\,,$$

$$k_{13} = \Big(- \ + h\lambda^2 - \frac{1}{2}h^2\lambda^3 + \frac{1}{6}h^3\lambda^4 - \frac{1}{24}h^4\lambda^5 + \frac{14767}{1700352}h^5\lambda^6 - \frac{1585}{1275264}h^6\lambda^7 + \frac{7297}{22954752}h^7\lambda^8 - \frac{599}{17216064}h^8\lambda^9 + \frac{12809}{2203656192}h^9\lambda^{10} + \frac{65}{275457024}h^{10}\lambda^{11} - \frac{325}{1101828096}h^{11}\lambda^{12}\Big) y_{j-1}\,,$$

$$y_j = \Big(1 - h\lambda + \frac{1}{2}h^2\lambda^2 - \frac{1}{6}h^3\lambda^3 + \frac{1}{24}h^4\lambda^4 - \frac{1}{120}h^5\lambda^5 + \frac{1}{720}h^6\lambda^6 - \frac{1}{5040}h^7\lambda^7 + \frac{1}{40320}h^8\lambda^8 - \frac{491}{209018880}h^9\lambda^9 + \frac{1333}{5643509760}h^{10}\lambda^{10} + \frac{13}{501645312}h^{11}\lambda^{11} - \frac{65}{4514807808}h^{12}\lambda^{12}\Big) y_{j-1}\,,$$

Consequently, we get the following numerical solution

$$y_j = \Big(1 - h\lambda + \tfrac{1}{2}h^2\lambda^2 - \tfrac{1}{6}h^3\lambda^3 + \tfrac{1}{24}h^4\lambda^4 - \tfrac{1}{120}h^5\lambda^5 + \tfrac{1}{720}h^6\lambda^6 - \tfrac{1}{5040}h^7\lambda^7 + \tfrac{1}{40320}h^8\lambda^8 - \tfrac{491}{209018880}h^9\lambda^9 + \tfrac{1333}{5643509760}h^{10}\lambda^{10} + \tfrac{13}{501645312}h^{11}\lambda^{11} - \tfrac{65}{4514807808}h^{12}\lambda^{12}\Big)^j \tag{3.11}$$

which decrease in time t as j → ∞. To satisfy this, we require the following stability condition

$$\Big|1 - h\lambda + \frac{1}{2}h^2\lambda^2 - \frac{1}{6}h^3\lambda^3 + \frac{1}{24}h^4\lambda^4 - \frac{1}{120}h^5\lambda^5 + \frac{1}{720}h^6\lambda^6 - \frac{1}{5040}h^7\lambda^7 + \frac{1}{40320}h^8\lambda^8 - \frac{491}{209018880}h^9\lambda^9 + \frac{1333}{5643509760}h^{10}\lambda^{10} + \frac{13}{501645312}h^{11}\lambda^{11} - \frac{65}{4514807808}h^{12}\lambda^{12}\Big| < 1\,.$$

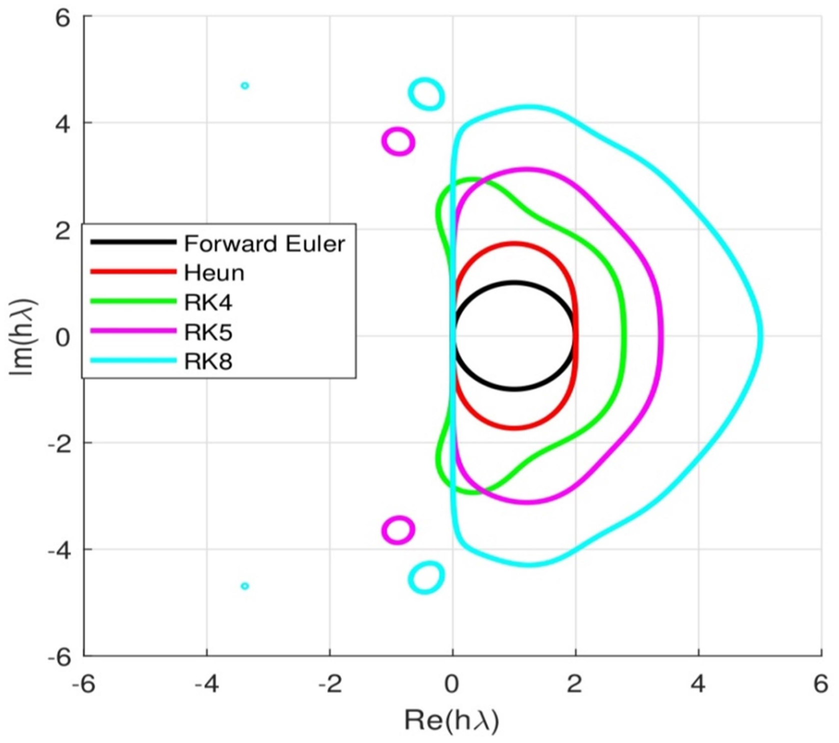
The stability function of RK8 is

$$S(h\lambda) = 1 - h\lambda + \frac{1}{2}(h\lambda)^2 - \frac{1}{6}(h\lambda)^3 + \frac{1}{24}(h\lambda)^4 - \frac{1}{120}(h\lambda)^5 + \frac{1}{720}(h\lambda)^6 - \frac{1}{5040}(h\lambda)^7 + \frac{1}{40320}(h\lambda)^8 - \frac{491}{209018880}(h\lambda)^9 + \frac{1333}{5643509760}(h\lambda)^{10} + \frac{13}{501645312}(h\lambda)^{11} - \frac{65}{4514807808}(h\lambda)^{12}$$

The stability function of RK8 involves six more terms than the terms of the stability function of RK5. The stability function of RK5 does also involve the first six terms of the stability function of RK8. The seventh term in this stability function of RK8 does not equal the last term of the stability function of RK5 because the fraction is different.

RK8 has a higher number of function evaluations than the forward Euler method, Heun's method, RK4, RK5, and RK8. The forward Euler method has the lowest number of function evaluations. Söderlind [26] discusses that high-order explicit Runge-Kutta methods involving a high number of function evaluations will have higher computational effort than thelow-orderexplicit-Runge-Kuttamethods.

S´eka and Assui [24] have also done stability analysis for the forward Euler method, Heun's method, and RK4 considering Equation (3.6). In their stability analysis, they obtained equations that almost equal to Equation (3.4), Equation (3.7), and Equation (3.9). The equations in [24] do not alternate in sign which Equation (3.4), Equation (3.7), and Equation (3.9) do. The reason for this is that the authors have solved Equation (3.6) is a nonnegative test equation which we have not done in the stability analysis for the five explicit Runge-Kutta methods. However, since the terms of Equation (3.4), Equation (3.7), and Equation (3.9) equal the terms of the equations in [24], then we can state that Equation (3.4), Equation (3.7), and Equation (3.9) is correct. Furthermore, the first eight terms of Equation (3.11) equal the stability function of RK8 with eleven stages in [24]. The terms of this stability function in [24] do not alternate in sign as the terms of Equation (3.11) does. The stability analysis for RK5 and RK8 with thirteen stages is difficult to



findFigure 3.1. Stability regions of the five explicit Runge-Kutta methods in a complex plane

in the literature. We did the stability analysis for RK5 (see Section 3.2.4) and RK8 (see Section 3.2.5) based on the procedure of the stability analysis for the forward Euler method.

3.2.6 Comparison of stability regions

In Section 3.2.1 to Section 3.2.5, we obtained stability conditions for the five explicit Runge-Kutta methods which are plotted in a complex plane to visualize its stability regions in MATLAB (see Script 1 in Appendix A). MATLAB is a software which we use to perform numerical computation, we visualize the data in figures and tables. We run Script 1 in MATLAB and we get Figure 3.1. We see in Figure 3.1, that the stability region of the forward Euler method is a disk of radius 1. The forward Euler method is centered at the point of $h\lambda = 1$. Söderlind [26] has also analyzed stability regions of the forward Euler method for solving Equation (3.6) and obtained instead that the forward Euler method is centered at $h\lambda = -1$ in a complex plane. The stability region of RK4, RK5, and RK8 have a shape as an ear lobe. The five explicit Runge-Kutta methods are stable inside their regions and unstable outside their regions

In Figure 3.1, we see that the five explicit Runge-Kutta methods are stable on the positive real coordinate R_e(hλ) of the complex plane. The stability regions are on the right-hand side of the complex plane because the five explicit Runge-Kutta methods are stable for values of R_e(hλ) > 0, which are bounded in its stabilityregions. We can see that RK4, RK5, and RK8 have larger stability regions than the forward Euler method and Heun's method. The reason is that as the order of explicit Runge-Kutta methods gets higher, then the explicit Runge-Kutta methods become stable for more values of Re(hλ) and its stability regions get larger which is in agreement with [26]. This means that RK4, RK5, and RK8 are stable for more values of Re(hλ) than the forward Euler method and Heun's method. Additionally, according to Atkinson et al. [4] explicit Runge-Kutta methods get larger stability regions when they are restricted on too small step sizes. The step size of h needs to be small for hλ to be in the stability region and to obtain stable numerical methods. The numerical method with the widest stability region has better stability. This means that RK8 is the most stable of all the five explicit Runge-Kutta methods because it has the largest stability region. Also, the forward Euler method with the smallest stability region is less stable than Heun's method, RK4, RK5, and RK8.

Moreover, Butcher [6] has also plotted the stability regions of the forward Euler method, Heun's method, and RK4. He obtained a similar figure as Figure 3.1 but without the stability regions of RK5 and RK8. He has also plotted the stability regions of the forward Euler method, Heun's method, and RK4 in one complex plane as we have done in Figure 3.1. Therefore, the stability regions of the forward Euler method, Heun's method, and RK4 is equal to the stability regions presented in [6]. In Figure 3.1, all the stability regions of the five explicit Runge-Kutta methods are plotted in one complex plane

Furthermore, S´eka and Assui [24] present the result of their research paper that RK8 with eleven stages has a smaller stability region than Heun's method and RK4. This result does not agree with what we see in Figure 3.1, where we see that RK8 has the largest stability region of all the five explicit Runge-Kutta methods. Even though this result does not agree with what we see in Figure 3.1, this result in [24] is correct because they have investigated the stability of RK8 with eleven stages and not thirteen stages as we have done in this thesis.

Bibliography

[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 RungeKutta 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 RungeKutta 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. Thom´ee. 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 amplifica-tion error deriva-tives for the construction of a modified Runge-Kutta Nystr¨om 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. Kılı¸cman. Solving zhou chaotic system using fourth-order Runge-Kutta method. World Applied Sciences Journal, 21(6):939– 944, 2013.

[22] M. Sch¨afer. 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. S´eka 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. S¨oderlind. 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.

[29] W.H. Witty. A new method of numerical integration of differential equations. Mathematics of Computation, 18(87):497–500, 1964.

[30] A. S. Wusu, M. A. Akanbi, and S. A. Okunuga. A three-stage multiderivative explicit Runge-Kutta method. American Journal of Computational Mathematics, 3(2):121–126, 2013.

[31] W.Y. Yang, W. Cao, J. Kim, K. W. Park, H. H. Park, J. Joung, J. S. Ro, H. L. Lee, C. H. Hong, and T. Im. Applied numerical methods using MATLAB. John Wiley & Sons, 2020.