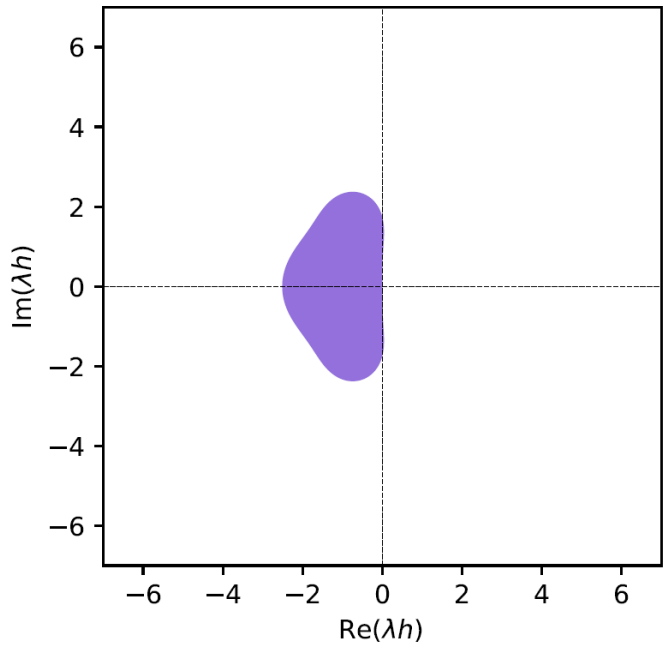
Analysis of Runge-Kutta methods using Butcher tableaus



Charlotte Kauppinen

Bachelor End Project 2023

TU Delft

Analysis of Runge-Kutta methods using Butcher tableaus

by

Charlotte Kauppinen

to obtain the degree of Bachelor of Science in Applied Mathematics
at the Delft University of Technology,
to be defended publicly on Friday July 14, 2023.

Student number:	5339855	
Project duration:	April 25, 2023 – July 14, 2023	
Thesis committee:	D. den Ouden-van der Horst,	TU Delft, supervisor
	C. Kraaikamp,	TU Delft

TU Delft

Abstract

The aim of this thesis is to analyse Runge-Kutta methods using Butcher tableaus. Runge-Kutta method are numerical methods used for approximating initial value problems. A Runge-Kutta method can be classified as either an explicit or an implicit method. A special kind of implicit methods are diagonally implicit methods. The type of method can be recognised by the Butcher tableau.

Using the entries of the Butcher tableau, one can compute the amplification factor of a Runge-Kutta method. The amplification factor can then be used to compute the order of the local truncation error and the stability region. Examples of these computations are given for seven methods. Furthermore, this thesis provides an algorithm to perform time steps for each of the three types of Runge-Kutta methods. Finally, in order to analyse the global truncation error of the seven methods, the algorithm to perform time steps is used with different step sizes.

Preface

This thesis is written as part of my Bachelor of Science in Applied Mathematics at the University of Technology at Delft.

I want to thank Dennis den Ouden-van der Horst for supervising this project. He provided supportive guidance, insight in the project and a relaxed work environment to make sure this thesis came to a good end. I would like to thank Cor Kraaikamp for taking part in my thesis committee.

Finally, I want to thank my friends for their support during this project. Especially Rick, Sam, Merel, Thomas, Niels, Jara, Hanne, Ida and Karel for proofreading and discussing the thesis.

Charlotte Kauppinen
Delft, July 2023

Contents

1 Introduction 1

2 Runge-Kutta methods and Butcher tableaus 3
2.1 An introduction to Runge-Kutta methods and Butcher tableaus . . . 3
2.2 Classification of Runge-Kutta methods . . . 4
2.2.1 Explicit methods . . . 4
2.2.2 Implicit methods . . . 4
2.2.3 Diagonally implicit methods . . . 4
2.3 Examples of Runge-Kutta methods . . . 5
2.3.1 The RK4 method . . . 5
2.3.2 Van der Houwen and Wray's method . . . 5
2.3.3 Kraaijevanger and Spijker's method . . . 6
2.3.4 Third-order Radau IIA method . . . 6

3 Local truncation error and stability analysis 7
3.1 Computation . . . 7
3.1.1 The amplification factor . . . 7
3.1.2 The local truncation error . . . 8
3.1.3 The stability region . . . 9
3.2 Examples of computations . . . 11
3.2.1 The amplification factor and local truncation error . . . 11
3.2.2 Computation of the stability region . . . 12

4 Algorithms for time steps 15
4.1 Time step explicit method . . . 15
4.1.1 Algorithm . . . 15
4.1.2 Van der Houwen and Wray's method . . . 15
4.2 Time step diagonally implicit method . . . 16
4.2.1 Algorithm . . . 16
4.2.2 Kraaijevanger and Spijker's method . . . 17
4.3 Time step implicit method . . . 18
4.3.1 Algorithm . . . 18
4.3.2 Radau's method . . . 19

5 Global truncation error analysis 21
5.1 Absolute error analysis . . . 21

6 Summary and recommendations 23
6.1 Summary . . . 23
6.2 Recommendations . . . 24

A Stability region of some methods 25

1

Introduction

Many processes that occur in nature can be conceptualised as initial value problems (IVPs). These problems arise when the behaviour of a system is determined by its initial state, evolving over time according to specific rules or laws. Braun, 1992 describes how the Italian biologist Umberto D'Ancona came across data on percentages of sharks caught in the Mediterranean Sea. The biologist was interested in finding out how the intensity of fishing affected the fish populations. His colleague analysed the situation and concluded that D'Ancona's predator-prey problem could be described by IVPs.

The presence of IVPs in everyday life is the reason we are interested in methods to solve these problems. There are many methods to analytically find to solution to an initial value problem. Some famous examples include the method of variation of parameters and the method of Laplace transforms, both described in Braun, 1992. Analytical solutions can provide useful insights in the processes described by IVPs. A disadvantage of analytical methods is that they can not be used to solve all IVPs. In some cases, the complexity of the problem or the lack of known analytical methods makes the IVP unsolvable analytically.

Numerical methods come into the picture when analytical methods fall short. Numerical methods are an algorithmic way to approximate the solution of an IVP without knowing the analytic solution. The general approach of a numerical method is to start at the given initial value and approximate the solution at each next time step until the end time is reached.

This thesis will specifically discuss Runge-Kutta methods. Runge-Kutta methods famously compute the value of the next time step using a certain number of stages. The methods will be applied to first order IVPs, although the analysis could be extended to IVPs of higher order.

As numerical methods only provide an approximation of the solution to an IVP, errors are inevitably made. In order to guarantee accuracy and reliability of the methods, we want to analyse the order of the errors. The order of the local truncation error and the global truncation error are treated in this thesis. Moreover, the stability of Runge-Kutta methods is analysed. Stability analysis is concerned with reliability of a method in preserving the solution's behaviour over time.

Chapter 2 introduces Runge-Kutta methods and Butcher tableaus. It presents a classification of Runge-Kutta methods, categorising into explicit, implicit and diagonally implicit methods. Some examples of Runge-Kutta methods are presented in the form of Butcher tableaus. These examples will be used throughout the thesis to perform computations.

Chapter 3 is concerned with the analysis of the local truncation error and the stability of Runge-Kutta methods. An expression for the amplification factor for a Runge-Kutta method is derived, which is then used to find the order of the local truncation error and stability region. Furthermore, an algorithm is given to find the step size for which explicit methods are stable.

Chapter 4 provides algorithms to perform time steps of explicit, implicit and diagonally implicit methods. Each algorithm is executed for a method introduced in Chapter 2 to find numerical approximations to an IVP.

Chapter 5 analyses the global truncation error using approximations of the solution to an IVP computed with the algorithms provided in the previous chapter. Finally, a summary and recommendations for further research are given in Chapter 6.

2

Runge-Kutta methods and Butcher tableaus

This chapter will give an introduction to Runge-Kutta methods and Butcher tableaus. We will first describe Runge-Kutta methods in a general form and convert them into Butcher tableaus. Then we will make a distinction between explicit, implicit and diagonally implicit Runge-Kutta methods and give examples of all three. Finally, four other examples of Runge-Kutta methods will be given to provide the reader with a better understanding of Runge-Kutta methods.

2.1. An introduction to Runge-Kutta methods and Butcher tableaus

Runge-Kutta methods are well-known numerical methods to solve initial value problems of the following form:

$$\frac{dy}{dt} = f(t, y),\ y(t_0) = y_0. \tag{2.1}$$

We call w_n the approximated value at time t_n. That is, $w_n \approx y(t_n)$. Runge-Kutta methods famously compute the value of the next time step using a certain number of stages, which we call $k_1, k_2, \dots, k_s$. The value w_{n+1} is computed by Equation (2.2). In this equation, $h > 0$ is the step size between two consecutive values and stages k_j are given as in Equation (2.3). Furthermore, coefficients a_{ij}, b_i and c_j specify the Runge-Kutta method.

$$w_{n+1} = w_n + h\sum_{i=1}^{s} b_i k_i. \tag{2.2}$$

$$\begin{cases} k_1 &= f(t_n + c_1 h, w_n + h\sum_{i=1}^{s} a_{1i}k_i), \\ k_2 &= f(t_n + c_2 h, w_n + h\sum_{i=1}^{s} a_{2i}k_i), \\ &\vdots \\ k_s &= f(t_n + c_s h, w_n + h\sum_{i=1}^{s} a_{si}k_i). \end{cases} \tag{2.3}$$

The coefficients of the method can be written in a so-called Butcher tableau, which is displayed in general form in Table 2.1.

c_1	a_{11}	a_{12}	$\cdots$	a_{1s}
c_2	a_{21}	a_{22}	$\cdots$	a_{2s}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
c_s	a_{s1}	a_{s2}	$\cdots$	a_{ss}
	b_1	b_2	$\cdots$	b_s

Table 2.1: Butcher tableau of a Runge-Kutta method.

2.2. Classification of Runge-Kutta methods

Each Runge-Kutta method can be defined by the coefficients given in the previous section. Runge-Kutta methods can be divided into explicit and implicit methods. Furthermore, we establish a distinct classification for diagonally implicit methods. All methods described in this section can be found in Vuik et al., 2007.

2.2.1. Explicit methods

A Runge-Kutta method is considered explicit when the expression for w_{n+1} only depends on w_n and t_n. As a consequence, the coefficient matrix $[a_{ij}]$ of explicit methods is a lower triangular matrix with zeros on the diagonal.

The simplest example of an explicit method is the Forward Euler method, which is given by $w_{n+1} = w_n + hf(t_n, w_n)$. The corresponding Butcher tableau is given by Table 2.2.

0	0
	1

Table 2.2: Butcher tableau of the Forward Euler method.

2.2.2. Implicit methods

Implicit methods can be recognised by expressions for w_{n+1} that do not solely depend on w_n and t_n. Specifically, each k_m in Equation (2.3) may depend on any k_i for $1 \leq i \leq s$.

The Backward Euler method is the simplest example of an implicit method. This method is given by $w_{n+1} = w_n + hf(t_{n+1}, w_{n+1})$ and its Butcher tableau is shown in Table 2.3.

1	1
	1

Table 2.3: Butcher tableau of the Backward Euler method.

It is not immediately evident that the method corresponds with the Butcher tableau. Substitution of coefficients of Table 2.3 in Equation (2.2) and (2.3) results in a system of equations that represents the Backward Euler method:

$$w_{n+1} = w_n + hk_1, \tag{2.4a}$$
$$k_1 = f(t_{n+1}, w_n + hk_1). \tag{2.4b}$$

To rewrite the system in the familiar form, one can substitute Equation (2.4a) into Equation (2.4b) to find $k_1 = f(t_{n+1}, w_{n+1})$. This can be used to find that Equation (2.4) is indeed equivalent to the expression $w_{n+1} = w_n + hf(t_{n+1}, w_{n+1})$.

Implicit methods require more computations, since for every time step a system of equations needs to be solved. However, properties of implicit methods described in Rehman et al., 2020 make them more suitable for solving stiff differential equations.

2.2.3. Diagonally implicit methods

Diagonally implicit methods can be recognised by lower triangular coefficient matrices $[a_{ij}]$. Unlike for explicit methods, the entries of the diagonal may be non-zero and one of the entries of the diagonal must be non-zero. Consequently, the expression for w_{n+1} does not solely depend on w_n. Diagonally implicit methods are therefore a subcategory of implicit methods.

The Backward Euler method described in the previous subsection is an example of a diagonally implicit method. Another well-known example is the Trapezoidal method, which is given by $w_{n+1} = w_n + \frac{h}{2}(f(t_n, w_n) + f(t_{n+1}, w_{n+1}))$. The Butcher tableau of the method is given in Table 2.4.

0	0	0
1	$\frac{1}{2}$	$\frac{1}{2}$
	$\frac{1}{2}$	$\frac{1}{2}$

Table 2.4: Butcher tableau of the Trapezoidal method.

Again it is not immediately clear that the Butcher tableau corresponds with the method. To show the correspondence, we substitute the coefficients of the tableau in Equation (2.2) and (2.3) to find

$$w_{n+1} = w_n + h(\frac{1}{2}k_1 + \frac{1}{2}k_2), \tag{2.5a}$$

$$k_1 = f(t_n, w_n), \tag{2.5b}$$

$$k_2 = f(t_{n+1}, w_n + h(\frac{1}{2}k_1 + \frac{1}{2}k_1)). \tag{2.5c}$$

The substitution of Equation (2.5a) in Equation (2.5c) yields $k_2 = f(t_{n+1}, w_{n+1})$. Then the substitution of k_1 and k_2 in Equation (2.5a) results in the given method: $w_{n+1} = w_n + \frac{h}{2}(f(t_n, w_n) + f(t_{n+1}, w_{n+1}))$.

The next section will discuss four other examples of Runge-Kutta methods. Two explicit methods, one diagonally implicit method and one implicit method. These methods, along with the methods discussed in this section, will be used in the rest of this thesis to perform computations.

2.3. Examples of Runge-Kutta methods

This section describes four examples of Runge-Kutta methods: RK4, Van der Houwen and Wray's method, Kraaijevanger and Spijker's method and a third-order Radau IIA method.

2.3.1. The RK4 method

Perhaps the most familiar Runge-Kutta method is RK4, because it offers a good balance between computation cost and order of accuracy. It is a fourth order explicit method that only requires four stages. The Butcher tableau of this method is given in Table 2.5. The method is described in Vuik et al., 2007.

0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	0	0	0
$\frac{1}{2}$	0	$\frac{1}{2}$	0	0
1	0	0	1	0
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

Table 2.5: Butcher tableau of the RK4 method.

2.3.2. Van der Houwen and Wray's method

Van der Houwen and Wray's method is an explicit method with two stages. The method is described in van der Houwen, 1972 and the Butcher tableau is given in Table 2.6.

0	0	0	0
$\frac{8}{15}$	$\frac{8}{15}$	0	0
$\frac{2}{3}$	$\frac{1}{4}$	$\frac{5}{12}$	0
	$\frac{1}{4}$	0	$\frac{3}{4}$

Table 2.6: Butcher tableau of van der Houwen and Wray's method.

2.3.3. Kraaijevanger and Spijker's method

Kraaijevanger and Spijker's method is a two-stage diagonally implicit method. The method is described in Kraaijevanger and Spijker, 1989. The Butcher tableau of the method is given in Table 2.7.

$\frac{1}{2}$	$\frac{1}{2}$	0
$\frac{3}{2}$	$-\frac{1}{2}$	2
	$-\frac{1}{2}$	$\frac{3}{2}$

Table 2.7: Butcher tableau of Kraaijevanger and Spijker's method.

2.3.4. Third-order Radau IIA method

The third-order Radau IIA method, from now on referred to as Radau's method, is an implicit method. Because the method is implicit, it is expensive to implement. The method is described in Hairer and Wanner, 2015 and its Butcher tableau is given in Table 2.8.

$\frac{1}{3}$	$\frac{5}{12}$	$-\frac{1}{12}$
1	$\frac{3}{4}$	$\frac{1}{4}$
	$\frac{3}{4}$	$\frac{1}{4}$

Table 2.8: Butcher tableau of the third order Radau IIA method.

The seven Runge-Kutta methods discussed in this chapter will be used in the following chapters as examples to compute amplification factors, local truncation errors and stability regions.

3

Local truncation error and stability analysis

This chapter will deal with the local truncation error that is made in numerical methods, which in our case applies to Runge-Kutta methods. This will be done by computing the amplification factor of a Runge-Kutta method, which can then be used to find the local truncation error, stability region and bounds for the step size for explicit methods. The computations will then be applied to the methods introduced in Chapter 2. This will result in a table with the amplification factor and local truncation error of each of the seven methods. The stability region of the methods will be plotted in the complex plane. Finally, an example for the computation of the bounds of a step size will be given.

3.1. Computation

Using the test equation $y' = \lambda y$, one can rewrite Equation (2.2) in the form $w_{n+1} = w_n Q(\lambda h)$. $Q(\lambda h)$ is then referred to as the amplification factor. The amplification factor of a Runge-Kutta method can be computed using its Butcher tableau. The amplification factor can then be used to find the local truncation error and stability region of the method.

3.1.1. The amplification factor

The amplification factor $Q(\lambda h)$ of a numerical method is the ratio of the approximated value at the next time step w_{n+1} to the current approximation w_n. That is, $Q(\lambda h) = \frac{w_{n+1}}{w_n}$. The amplification factor is computed using the test equation $y' = \lambda y$. The substitution $f(t, y) = \lambda y$ can be used to find

$$\begin{cases} k_1 &= \lambda h(w_n + h\sum_{i=1}^{s} a_{1i}k_i), \\ k_2 &= \lambda h(w_n + h\sum_{i=1}^{s} a_{2i}k_i), \\ &\vdots \\ k_s &= \lambda h(w_n + h\sum_{i=1}^{s} a_{si}k_i). \end{cases} \tag{3.1}$$

Define the following s-dimensional vectors: $\vec{e} = [1, \dots, 1]^T$, $\vec{k} = [k_1, \dots, k_s]^T$ and $\vec{b} = [b_1, \dots, b_s]$. Define the $s \times s$ matrix $A = \begin{pmatrix} a_{11} & \dots & a_{1s} \\ \vdots & \ddots & \vdots \\ a_{s1} & \dots & a_{ss} \end{pmatrix}$. Then $\vec{k}$ can be written as $\vec{k} = \lambda w_n \vec{e} + \lambda h A\vec{k}$. Solving this equation for $\vec{k}$ yields:

$$\vec{k} = \lambda w_n (I - \lambda h A)^{-1}\vec{e}. \tag{3.2}$$

Substituting Equation (3.2) in Equation (2.2) gives $w_{n+1} = w_n(1 + \lambda h\vec{b}(I - \lambda h A)^{-1}\vec{e})$. Therefore the amplification factor of a general Runge-Kutta method is given by:

$$Q(\lambda h) = 1 + \lambda h\vec{b}(I - \lambda h A)^{-1}\vec{e}. \tag{3.3}$$

Theorem 1. *The amplification factor $Q(\lambda h) = 1 + \lambda h\vec{b}(I - \lambda h A)^{-1}\vec{e}$ of an explicit Runge-Kutta method can be written as a polynomial of at most order s in the indeterminate λh.*

Proof. Assume A and $\vec{b}$ are the matrix and vector corresponding to the Butcher tableau of an explicit Runge-Kutta method with s stages. Then A is an $s \times s$ lower triangular matrix with zeros on the diagonal.

Consider any square matrix M. We claim that

$$\forall s \in \mathbb{N} \setminus \{0\} : I - M^s = (I - M)(I + M + M^2 + ... + M^{s-1}). \tag{3.4}$$

We prove this claim using induction. For the base case, we notice that both the left hand side and the right hand side are equal to $I - M$ for $s = 1$. Assume $I - M^k = (I - M)(I + M + M^2 + ... + M^{k-1})$ for some $k \geq 1$. Then

$$\begin{aligned} I - M^{k+1} &= I - M + (I - M^k)M \\ &= I - M + (I - M)(I + M + M^2 + ... + M^{k-1})M \\ &= (I - M)I + (I - M)(M + M^2 + M^3 ... + M^k) \\ &= (I - M)(I + M + M^2 + ... + M^k), \end{aligned}$$

which proves the induction step.

Next we consider the characteristic polynomial $p(\mu)$ of matrix λhA. Since matrix λhA is a lower triangular matrix with zeros on the diagonal, the eigenvalues all equal zero.

$$p(\mu) = \det(\lambda hA - \mu I) = \mu^s = 0.$$

The Cayley-Hamilton theorem (Straubing, 1983) states that the analogous polynomial $p(\lambda hA)$ is equal to the zero matrix. That is,

$$p(\lambda hA) = (\lambda hA)^s = 0_{s \times s}.$$

Then by Equation (3.4) we find

$$I = I - (\lambda hA)^s = (I - \lambda hA)(I + \lambda hA + (\lambda hA)^2 + ... + (\lambda hA)^{s-1}).$$

This is then used to find the inverse matrix $(I - \lambda hA)^{-1}$. As matrix $(I - \lambda hA)$ is a lower triangular matrix with 1 on each entry of the diagonal, the matrix is non-singular and therefore invertible.

$$(I - \lambda hA)^{-1} = I + \lambda hA + (\lambda hA)^2 + ... + (\lambda hA)^{s-1} = \sum_{k=0}^{s-1} (\lambda h)^k A^k.$$

Finally we notice that $\vec{b}A^k\vec{e}$ is a scalar $\forall k$ and therefore

$$\begin{aligned} Q(\lambda h) &= 1 + \lambda h \vec{b}(I - \lambda hA)^{-1}\vec{e} \\ &= 1 + \lambda h \sum_{k=0}^{s-1} (\lambda h)^k \vec{b}A^k\vec{e} \\ &= 1 + \sum_{k=0}^{s} (\lambda h)^k \vec{b}A^{k-1}\vec{e} \end{aligned} \tag{3.5}$$

is a polynomial of at most order s in the indeterminate λh. □

It is plausible that the amplification factor of any implicit Runge-Kutta method can be written as a rational function. However, a proof currently eludes the author. A proof possibly includes similar steps to the proof of Theorem 1, except that the eigenvalues of matrix λhA do not all equal zero by the nature of implicit methods. Therefore it may not be concluded that $(\lambda hA)^s = 0_{s \times s}$.

Section 3.2 provides several examples of amplification factors computed for explicit and implicit methods. For each explicit method, the amplification factor is a polynomial in the indeterminate λh, as states in Theorem 1. For each implicit method in Section 3.2, the amplification factor is a rational function.

3.1.2. The local truncation error

The local truncation error of a numerical method is the error caused by one iteration. The order of the local truncation error, as defined in Vuik et al., 2007 can be computed using Taylor polynomials. The local truncation error τ_{n+1} is given by

$$\tau_{n+1} = \frac{y_{n+1} - w_{n+1}}{h}. \tag{3.6}$$

The analytical solution y does not need to be known in order to compute the order of the local truncation error. The value y_{n+1} can be written as a Taylor polynomial around y_n:

$$y_{n+1} = y_n + hy'_n + \frac{1}{2}h^2y''_n + ... + \frac{1}{p!}h^p y_n^{(p)} + \mathcal{O}(h^{p+1}). \tag{3.7}$$

Using the test equation $y' = \lambda y$ we find the expression $y'_n = \lambda y_n$. Furthermore, $y''_n = \lambda y'_n = \lambda^2 y_n$ and inductively $y_n^{(p)} = \lambda^p y_n$. These expressions can be substituted in Equation (3.7) to rewrite y_{n+1} as

$$\begin{aligned} y_{n+1} &= y_n + \lambda h y_n + \frac{1}{2}(\lambda h)^2 y_n + ... + \frac{1}{p!}(\lambda h)^p y_n + \mathcal{O}(h^{p+1}) \\ &= \left(1 + \lambda h + \frac{1}{2}(\lambda h)^2 + ... + \frac{1}{p!}(\lambda h)^p\right) y_n + \mathcal{O}(h^{p+1}). \end{aligned} \tag{3.8}$$

The term w_{n+1} in Equation (3.6) can be rewritten using the amplification factor $Q(\lambda h)$. The resulting equation $w_{n+1} = Q(\lambda h)y_n$ and Equation (3.8) can be substituted in Equation (3.6) to find

$$\begin{aligned} \tau_{n+1} &= \frac{1}{h}\left[\left(1 + \lambda h + \frac{1}{2}(\lambda h)^2 + ... + \frac{1}{p!}(\lambda h)^p\right) y_n + \mathcal{O}(h^{p+1}) - Q(\lambda h)y_n\right] \\ &= \frac{1}{h}\left[\left(1 + \lambda h + \frac{1}{2}(\lambda h)^2 + ... + \frac{1}{p!}(\lambda h)^p - Q(\lambda h)\right) y_n + \mathcal{O}(h^{p+1})\right]. \end{aligned} \tag{3.9}$$

For explicit methods, the amplification factor is a polynomial and the term $Q(\lambda h)$ may directly be used. For implicit methods, one can rewrite $Q(\lambda h)$ using its Taylor expansion.

Equation (3.9) is used in Section 3.2 to find the local truncation error of several Runge-Kutta methods. Along with the local truncation error, the global truncation error of numerical methods is often analysed. The global truncation error is the cumulative error caused by many iterations. Chapter 5 will discuss global truncation errors.

3.1.3. The stability region

The values of λh that satisfy the condition $|Q(\lambda h)| \leq 1$ form the stability region. Since the eigenvalue λ can have non-zero imaginary part, the stability region can be shown in the complex plane. Some examples of stability regions in the complex plane are given in Subsection 3.2.2.

From the stability region and a given value for λ, one can determine the range for step size h for which the numerical method is stable. In Vuik et al., 2007 it is stated that implicit methods are unconditionally stable when all eigenvalues have non-positive real part. This means that implicit methods are stable for any choice of step size h. We are therefore only interested in finding the range of h for which explicit methods are stable.

To perform the algorithm that will be described in this subsection, we must make an assumption about the form of the stability region of explicit methods. We will assume that the values of the stability region with non-positive real part form a star-shaped subset S of the complex plane about the origin. That is, for all $z \in S$ we have $tz \in S$ for all $t \in [0, 1]$. With this assumption we notice that it suffices to solve the equation $|Q(\lambda h)| = 1$ for h, as any value h_0 with $0 \leq h_0 \leq h$ will then satisfy $|Q(\lambda h_0)| \leq 1$.

A proof to justify the assumption for all explicit methods currently eludes the author. Algorithm 1 may therefore give inaccurate results if the values of the stability region with non-positive real part do not form a star-shaped subset. To solve the equality, we use the Bisection method described in Section 4.3 in Vuik et al., 2007 to find the root of $|Q(\lambda h)| - 1 = 0$. The pseudocode of the method is given in Algorithm 1.

Algorithm 1 Algorithm root $|Q(\lambda h)| - 1$

```
1:  Define λ
2:  Define f(h) = |Q(λh)| − 1                          See Equation (3.3)
3:  Initialise p = 1
4:  if f(p) = 0 then
5:      p_u = p_l = p
6:  else if f(p) < 0 then
7:      while f(p) < 0 do
8:          p = 2p
9:      end while
10:     p_u = p
11:     p_l = p/2
12: else if f(p) > 0 then
13:     while f(p) > 0 do
14:         p = p/2
15:     end while
16:     p_u = 2p
17:     p_l = p
18: end if
19: if f(p_u) = 0 then
20:     r = p_u
21: else if f(p_l) = 0 then
22:     r = p_l
23: else
24:     while |(p_u − p_l)/p_u| > 10^−6 do
25:         p_m = (p_u + p_l)/2
26:         if f(p_m) = 0 then
27:             Break while loop
28:         else if f(p_m) · f(p_l) > 0 then
29:             p_l = p_m
30:         else
31:             p_u = p_m
32:         end if
33:     end while
34:     r = (p_u + p_l)/2
35: end if
36: Return r
```

The Bisection method assumes continuity of a function f on an interval $[a, b]$ where $f(a)$ and $f(b)$ have opposite sign. In Algorithm 1 the value of λ is known and the function f is defined as $f(h) = |Q(\lambda h)| - 1$. From Theorem 1 we know that $Q(\lambda h)$ is a polynomial in the indeterminate λh. The continuity of function f follows from the continuity of polynomials, absolute functions and their compositions. The values of a and b are given by p_l and p_u respectively and they are computed in Line 3-18. The values are chosen such that p_l is a lower bound with $f(p_l) < 0$ and p_u is an upper bound with $f(p_u) > 0$. Therefore the algorithm meets the assumptions of the Bisection method.

From Equation (3.5) one can deduce that $h = 0$ implies $|Q(\lambda h)| = 1$. This value for h corresponds with the origin in the complex plane that shows the stability region. We are therefore only interested in the equality $|Q(\lambda h)| = 1$ with $h > 0$. If the initial guess $p = 1$ on Line 3 satisfies the equality, the algorithm will eventually return $r = 1$. If the initial guess is a lower bound, the algorithm enters the loop that starts on Line 7. The value p is multiplied by 2 until $f(p) \geq 0$. That is, until p is an upper bound. This ensures that both the upper and lower bound are strictly larger than 0. By continuously updating the guess for the lower bound, the algorithm provides bounds that are closer to the root, so that the Bisection method will be completed in less iterations.

Alternatively, if the initial guess $p = 1$ is an upper bound, the algorithm enters the loop that starts on Line 13. The value p is divided by 2 until the value is a lower bound. Similar to the situation before, this approach ensures that both the upper an lower bound are strictly larger than 0. Moreover, updating the guess for the upper bound again allows the Bisection method to be performed with less iterations.

3.2. Examples of computations

In this section, the computations described in the previous section will be applied to seven Runge-Kutta methods. This will be done using the Butcher tableaus given in Chapter 2.

3.2.1. The amplification factor and local truncation error

The amplification factor of the seven methods described in Chapter 2 can be found using vector $\vec{b}$ and matrix A from the Butcher tableaus of the methods. In the case of the Forward Euler method, $\vec{b} = [1]$ and $A = [0]$. Using Equation (3.3), this yields the equation

$$Q(\lambda h) = 1 + \lambda h \cdot [1] \cdot (I - \lambda h \cdot [0])^{-1}\vec{e},$$

which can be reduced to $Q(\lambda h) = 1 + \lambda h$. The amplification factors of the seven methods are tabulated in Table 3.1.

The order of the local truncation error of Van der Houwen and Wray's method can be computed using the amplification factor from Table 3.1 and Equation (3.9). This results in the equation

$$\begin{aligned}\tau_{n+1} &= \frac{1}{h}\left[\left(1 + \lambda h + \frac{1}{2}(\lambda h)^2 + \frac{1}{6}(\lambda h)^3 - \left(1 + \lambda h + \frac{1}{2}(\lambda h)^2 + \frac{1}{6}(\lambda h)^3\right)\right) y_n + \mathcal{O}(h^4)\right] \\ &= \frac{1}{h}\mathcal{O}(h^4) \\ &= \mathcal{O}(h^3).\end{aligned}$$

To find the order of the local truncation error of Kraaijevanger and Spijker's method, the amplification factor should be rewritten as its Taylor expansion. This yields

$$\begin{aligned}Q(\lambda h) &= \frac{1 - \lambda h}{1 - 2\lambda h} \\ &= 1 + \lambda h + 2(\lambda h)^2 + \mathcal{O}(h^3).\end{aligned}$$

The local truncation error of Kraaijevanger and Spijker's method is then given by

$$\begin{aligned}\tau_{n+1} &= \frac{1}{h}\left[1 + \lambda h + \frac{1}{2}(\lambda h)^2 - \left(1 + \lambda h + 2(\lambda h)^2 + \mathcal{O}(h^3)\right) y_n + \mathcal{O}(h^3)\right] \\ &= \frac{1}{h}\left[-\frac{3}{2}(\lambda h)^2 y_n + \mathcal{O}(h^3)\right] \\ &= -\frac{3}{2}\lambda^2 h y_n + \mathcal{O}(h^2) \\ &= \mathcal{O}(h).\end{aligned}$$

The order of the local truncation errors of five other methods are displayed in Table 3.1.

In Table 3.1 ERK is an abbreviation of explicit Runge-Kutta, IRK is an abbreviation of implicit Runge-Kutta and DIRK is an abbreviation of diagonally implicit Runge-Kutta. We notice that the amplification factors of the explicit methods are indeed polynomials in the indeterminate λh, as stated in Theorem 1. Amplification factors of the implicit methods are rational functions, as we suspected in Subsection 3.1.1.

We notice that methods that have more stages often also have a higher order local truncation error. It is in general not that case that the number of stages corresponds with the order of the local truncation error, see for example Radau's method. Kraaijevanger and Spijker's method has a noticeable coefficient in the denominator as most amplification factors have coefficients of absolute values less or equal than 1, but Kraaijevanger and Spijker's method has a coefficient of value -2. The method has a higher number of stages than its order of the local truncation error.

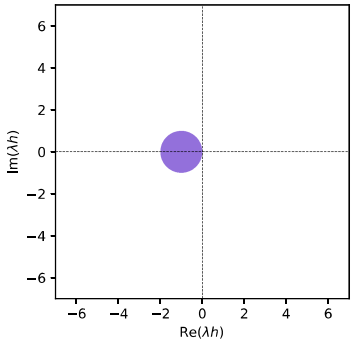
Method	Type	Amplification factor $Q(\lambda h)$	Error	Stages
Forward Euler	ERK	$1+\lambda h$	$\mathcal{O}(h)$	1
Backward Euler	DIRK	$\frac{1}{1-\lambda h}$	$\mathcal{O}(h)$	1
Trapezoidal	DIRK	$\frac{1+\frac{1}{2}\lambda h}{1-\frac{1}{2}\lambda h}$	$\mathcal{O}(h^2)$	2
RK4	ERK	$1+\lambda h+\frac{1}{2}(\lambda h)^2+\frac{1}{6}(\lambda h)^3+\frac{1}{24}(\lambda h)^4$	$\mathcal{O}(h^4)$	4
Van der Houwen and Wray	ERK	$1+\lambda h+\frac{1}{2}(\lambda h)^2+\frac{1}{6}(\lambda h)^3$	$\mathcal{O}(h^3)$	3
Kraaijevanger and Spijker	DIRK	$\frac{1-\lambda h}{1-2\lambda h}$	$\mathcal{O}(h)$	2
Radau	IRK	$\frac{1+\frac{1}{3}\lambda h}{1-\frac{2}{3}\lambda h+\frac{1}{6}(\lambda h)^2}$	$\mathcal{O}(h^3)$	2

Table 3.1: The amplification factor and local truncation error of seven Runge-Kutta methods.

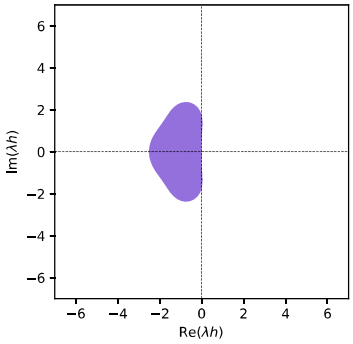
3.2.2. Computation of the stability region

The stability regions of the seven given Runge-Kutta methods can be found by using the amplification factors in Table 3.1 and finding the values of λh for which the inequality $|Q(\lambda h)| \leq 1$ holds.

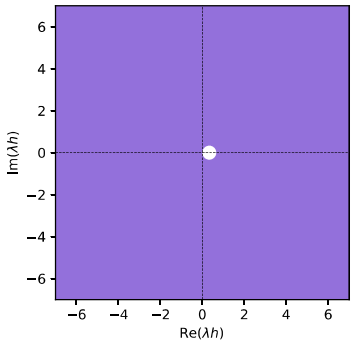
The stability region of four methods is shown in Figure 3.1.



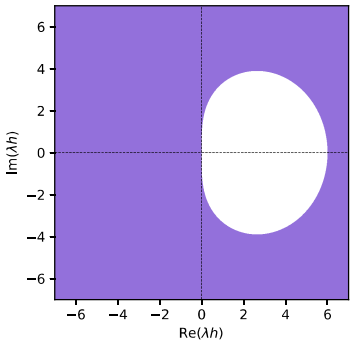
(a) Stability region of the Forward Euler method.



(b) Stability region of van der Houwen and Wray's method.



(c) Stability region of Kraaijevanger and Spijker's method.



(d) Stability region of Radau's method.

Figure 3.1: Stability region of four Runge-Kutta methods.

For the Forward Euler method, the condition $|Q(\lambda h)| \leq 1$ results in the inequality $|1+\lambda h| \leq 1$. In the complex plane, this is a circle with radius 1 and centre point $(0,-1)$. Figure 3.1a shows the stability region of the Forward Euler method.

The stability region of van der Houwen and Wray's method in Figure 3.1b is a shape that touches the origin. This means that the method is stable for small values of step size h. The method is not stable for any values that lie outside of the shape.

The stability region of Kraaijevanger and Spijker's method is shown in Figure 3.1c. We notice that the method is stable for any step size h when the real part of the eigenvalue λ is non-positive. This behaviour is expected from implicit Runge-Kutta methods. Furthermore, we notice that the circle in which the method is not stable is relatively small.

Finally, the stability region of Radau's method is shown in Figure 3.1d. Like Kraaijevanger and Spijker's method, Radau's method is implicit and unconditionally stable for eigenvalues with non-positive real part. The shape in which the method is not stable is significantly larger than the relevant shapes of other methods.

The stability regions of Backward Euler, the Trapezoidal method and RK4 are shown in Appendix A.

Using Algorithm 1 one can determine the range of h for which the inequality $|Q(\lambda h)| \leq 1$ holds. We will perform computations for the explicit methods introduced in Chapter 2. The computations will be done with $\lambda = -3$, which is the eigenvalue of Equation (4.1). The results of the computations rounded to six decimal places are tabulated in Table 3.2.

Method	Amplification factor $Q(-3h)$	h s.t. $\|Q(-3h)\| = 1$
Forward Euler	$1 - 3h$	0.666667
RK4	$1 - 3h + \frac{9}{2}h^2 - \frac{9}{2}h^3 + \frac{27}{8}h^4$	0.928431
Van der Houwen and Wray	$1 - 3h + \frac{9}{2}h^2 - \frac{9}{2}h^3$	0.837582

Table 3.2: The result of Algorithm 1 for $\lambda = -3$.

The result shown in Table 3.2 indicate that the Forward Euler method is stable for $0 \leq h \leq 0.666667$, RK4 is stable for $0 \leq h \leq 0.928431$ and Van der Houwen and Wray's method is stable for $0 \leq h \leq 0.837582$. As the amplification factor of the Forward Euler method is a first degree polynomial, it is simple to validate the result for this method. The inequality $|Q(-3h)| \leq 1$ can be written as

$$-1 \leq 1 - 3h \leq 1,$$

or equivalently

$$-\frac{1}{3} \leq h \leq \frac{2}{3}.$$

Since the step size h is non-negative, the relevant bounds are $0 \leq h \leq \frac{2}{3}$, which is corresponds to the value in Table 3.2.

The next chapter will discuss algorithms to perform time steps of the Runge-Kutta methods. When computing the time steps for explicit methods, one should take into account the bounds for h described in this chapter.

4

Algorithms for time steps

In this chapter, we will provide algorithms to perform time steps of the three types of Runge-Kutta methods. These algorithms will be applied to an initial value problem and the numerical approximation will be compared to the analytical solution.

We will approximate the solution to the IVP

$$\frac{dy}{dt} = -3y + 6t + 5,\ y(0) = 3. \tag{4.1}$$

The analytical solution to the IVP is known and given by $y(t) = 2e^{-3t} + 2t + 1$. The solution is stable, as Equation (4.1) has eigenvalue $\lambda = -3 < 0$. The IVP will be considered on the domain $[0, 2]$.

4.1. Time step explicit method

This section treats the algorithm to perform time steps of explicit Runge-Kutta methods.

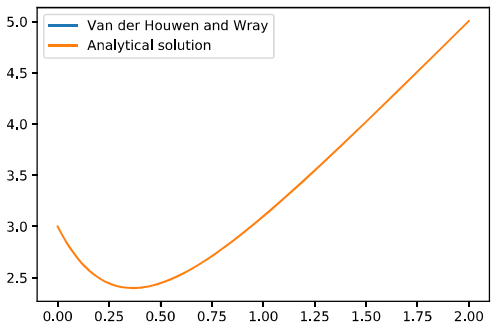
4.1.1. Algorithm

The algorithm for an explicit time step is straightforward. One can define the IVP and the desired explicit method to approximate the solution. Each stage of the method can be computed directly. Consequently, each time step can be computed. The reason that the algorithm for explicit methods is straightforward is because each k_m in Equation (2.3) only depends on previously computed values.

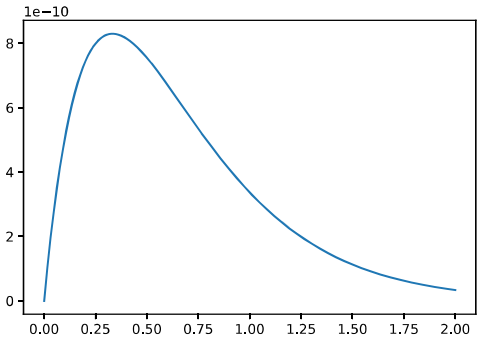
4.1.2. Van der Houwen and Wray's method

We have implemented the algorithm described in 4.1.1 and performed time steps for van der Houwen and Wray's method with step size $h = 0.001$. This choice for step size ensures stability as it lies within the bounds stated in Table 3.2. The analytical solution and numerical approximation are plotted in Figure 4.1a.

The difference between the analytical solution and the numerical approximation is barely visible from Figure 4.1a. For that reason, we have plotted the absolute difference between the two in Figure 4.1b. This difference corresponds with the numerical error of the method. We notice that the size of the error is very small compared to the value of the solution.



(a) Analytical solution and numerical approximation.



(b) Absolute difference between analytical solution and numerical approximation.

Figure 4.1: Analytical solution and numerical approximation of IVP (4.1) using van der Houwen and Wray's method.

4.2. Time step diagonally implicit method

This section treats the algorithm to perform time steps of diagonally implicit Runge-Kutta methods.

4.2.1. Algorithm

The algorithm for a diagonally implicit method is less straightforward than the algorithm for an explicit method. This is due to the fact that each k_m in Equation (2.3) depends on itself. The restriction to diagonally implicit methods allows for an algorithm with less computations than needed for implicit methods. The latter will be discussed in Section 4.3. For each k_m with $1 \leq m \leq s$ the equation to be solved is given by Equation (4.2):

$$k_m = f(t_n + c_j h, w_n + h \sum_{i=1}^{m} a_{2i} k_i). \tag{4.2}$$

Note that the sum over i is only up to m, since the value for k_m does not depend on any k_i with $i > m$. We use the Picard iteration as described in Vuik et al., 2007 to solve the fixed point problem. The algorithm to solve Equation (2.2) at each time t_n with $t_0 \leq t_n < t_{\text{end}}$ for a diagonally implicit method with s stages is given by Algorithm 2. Specifically, the algorithm to perform one time step is described in Lines 8-19.

In Algorithm 2 the initial guess for $\vec{k^a}[i]$ is 0, as can be seen in Line 8. This corresponds with the initial guess for w_{n+1} being w_n. We choose this initial guess to ensure that the first estimate is close to the previous value.

The value for $\epsilon = 10^{-6}$ in Line 11 and the stop condition in Line 17 are chosen such that the ratio of the difference between the previous two guesses $\vec{k^a}[i] - \vec{k^b}[i]$ to the current guess $\vec{k^b}[i]$ is smaller than 1:1,000,000. The value for ϵ is chosen large enough that it does not interfere with machine precision.

There are a few assumptions that the expression for each $\vec{k^b}[i]$, which we will call $g(k_i)$, needs to satisfy. First, the function $g(k_i)$ needs to be continuous on $[a, b]$. This is satisfied if the function f is continuous on $[a, b]$. The domain considered is $(-\infty, \infty)$. The expression should be a map $g(k_i) : (-\infty, \infty) \to (-\infty, \infty)$, which again is satisfied if it is true for the function f. Satisfaction of these assumptions implies the existence of a fixed point.

To ensure uniqueness, the derivative $g'(k_i)$ should exist for all $k_i \in (-\infty, \infty)$. Furthermore, there should be a positive constant $p < 1$ such that $|g'(k_i)| \leq p$ for all $k_i \in (-\infty, \infty)$. If these assumptions are not satisfied, Algorithm 2 may not converge to a unique fixed point.

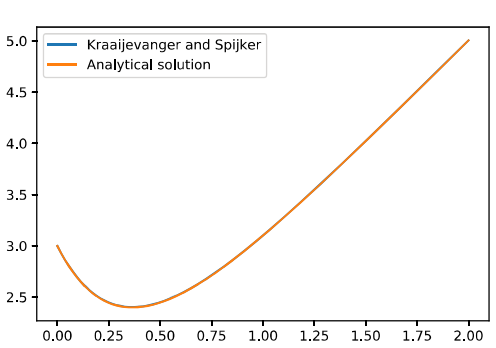
Algorithm 2 Algorithm diagonally implicit method

1: Define A, $\vec{b}$ and $\vec{c}$ — See Table 2.1
2: Define function f — See Equation (2.1)
3: Define t_0
4: Set $w_0 = y_0$
5: Set $h =$ step size
6: Set $\epsilon = 10^{-6}$
7: **for** $n = 0, \ldots, \frac{t_{end} - t_0}{h} - 1$ **do**
8: Initialise $\vec{k^a} = \vec{0}$
9: Initialise $\vec{k^b} = \vec{0}$
10: **for** $i = 1, \ldots, s$ **do**
11: Initialise $S > \epsilon$
12: **while** $S > \epsilon$ **do**
13: $\vec{k^b}[i] = f(t_n + c[i]h, w_n + h\sum_{j=1}^{i-1} A[i,j]k_i + hA[i,i]\vec{k^a}[i])$
14: **if** $\vec{k^b}[i] = 0$ **then**
15: Break while loop
16: **end if**
17: Set $S = \left|\frac{\vec{k^a}[i] - \vec{k^b}[i]}{\vec{k^b}[i]}\right|$
18: Set $\vec{k^a}[i] = \vec{k^b}[i]$
19: **end while**
20: **end for**
21: Set $t_{n+1} = t_n + h$
22: Set $w_{n+1} = w_n + h\sum_{i=1}^{s} b[i]\vec{k^a}[i]$
23: **end for**

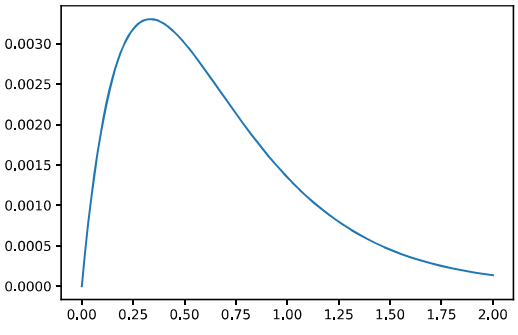
We notice that Algorithm 2 can also be used to compute time steps of explicit Runge-Kutta methods, as they satisfy the conditions of diagonally implicit methods, just with 0 in the diagonal. When we apply the algorithm to an explicit method, we find that the term that contains $\vec{k^a}[i]$ in Line 12 will always be 0, as $\forall i : A[i,i] = 0$. This results in $\vec{k^b}[i]$ being computed as described in Section 4.1. After one more iteration of the loop that starts in Line 12, the value of $\vec{k^a}[i]$ will be equal to $\vec{k^b}[i]$ and the loop ends. This is less efficient than the method of Section 4.1, as it requires the value of $\vec{k^a}[i]$ to be computed twice.

4.2.2. Kraaijevanger and Spijker's method

The algorithm discussed in the previous subsection is used to compute time steps of a diagonally implicit Runge-Kutta method. The results are shown in Figure 4.2



(a) Analytical solution and numerical approximation.



(b) Absolute difference between analytical solution and numerical approximation.

Figure 4.2: Analytical solution and numerical approximation of IVP (4.1) using Kraaijevanger and Spijker's method.

Algorithm 2 is used to compute a numerical approximation of the solution of IVP (4.1) using Kraaijevanger and Spijker's method with step size $h = 0.001$. The choice for step size does not influence the stability of the method, since the method is implicit. The approximation and the analytical solution are plotted in Figure 4.2a.

As the numerical method seems to give a good approximation of the analytical solution, it is unclear from the plot what the size of the error is. We have therefore also plotted the difference between the analytical solution and the numerical solution in Figure 4.2b.

From Figure 4.2b we can see that the difference between the analytical solution and numerical approximation is in the same order as the chosen step size. We will analyse the error further in Chapter 5.

4.3. Time step implicit method

This section treats the algorithm to perform time steps of implicit Runge-Kutta methods.

4.3.1. Algorithm

The algorithm to perform time steps of implicit methods is given in Algorithm 3.

Algorithm 3 Algorithm implicit method

Define A, $\vec{b}$ and $\vec{c}$ — See Table 2.1
Define function f — See Equation (2.1)
Define t_0
Set $w_0 = y_0$
Set $h =$ step size
Set $\epsilon = 10^{-6}$
for $n = 0, \ldots, \frac{t_{\text{end}} - t_0}{h} - 1$ **do**
    Initialise $\vec{k^a} = \vec{0}$
    Initialise $\vec{k^b} = \vec{0}$
    Initialise $S > \epsilon$
    while $S > \epsilon$ **do**
        for $i = 1, \ldots, s$ **do**
            $\vec{k^b}[i] = f(t_n + c[i]h, w_n + h\sum_{j=1}^{s} A[i,j]\vec{k^a}[i])$
        end for
        if $||\vec{k^b}||_2 = 0$ **then**
            Break while loop
        end if
        Set $S = \frac{||\vec{k^a} - \vec{k^b}||_2}{||\vec{k^b}||_2}$
        Set $\vec{k^a} = \vec{k^b}$
    end while
    Set $t_{n+1} = t_n + h$
    Set $w_{n+1} = w_n + h\sum_{i=1}^{s} b[i]\vec{k^a}[i]$
end for

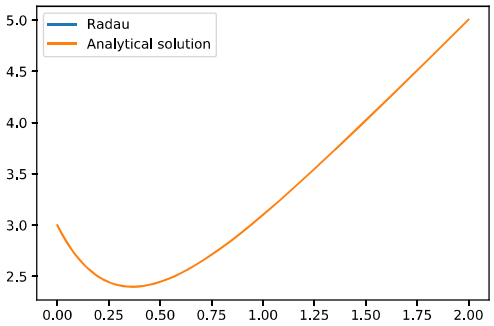
The algorithm for implicit methods has roughly the same set-up as the algorithm for diagonally implicit methods. However, Algorithm 2 only considers k_i for $1 \leq i \leq m$ when computing k_m, while k_m may depend on k_i with $m < i \leq s$ in an implicit method. We again use the Picard iteration to determine the value of every k_m. Due to the dependency, the values are not determined one at the time. The algorithm makes an estimate in the form of a vector $\vec{k}$ and uses the Picard iteration to approximate the values of the entries. The expression for each $\vec{k^b}[i]$ should again satisfy the assumptions described in Subsection 4.2.1. The initial guess for each k_m is again zero, as this corresponds with the first guess for w_{n+1} being w_n. The pseudocode of the algorithm to perform time steps from t_0 to t_{end} is given in Algorithm 3.

The algorithm for implicit methods will also give the desired approximation for an explicit or diagonally implicit method. However, the algorithm then requires more computations than necessary for the method. The initialisation of vector $\vec{k}$ as the zero vector means that each entry of $\vec{k}$ is assumed to be unknown, while the entries are easy to compute, especially for explicit methods.

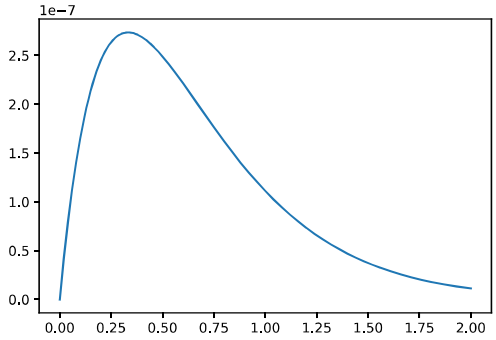
The choice for ϵ in Line 6 is justified by roughly the same argument that the stop condition does not interfere with the problems that may arise due to machine precision. The stop condition makes sure that the length of the difference between $\vec{k^a}$ and $\vec{k^b}$ relative to the length of $\vec{k^b}$ is smaller than 10^{-6}.

4.3.2. Radau's method

We apply Algorithm 3 to IVP (4.1) with Radau's method, again with step size $h = 0.001$. The approximation and the analytical solution are plotted in Figure 4.3a, the difference between the approximation and analytical solution is plotted in Figure 4.3b.



(a) Analytical solution and numerical approximation.



(b) Absolute difference between analytical solution and numerical approximation.

Figure 4.3: Analytical solution and numerical approximation of IVP (4.1) using Radau's method.

We again notice that the algorithm gives a good approximation of the solution. The size of the error is very small in comparison to the value of the solution.

In this chapter, we discussed algorithms for computing time steps of explicit, diagonally implicit and implicit methods. These algorithms were applied to IVP (4.1) for van der Houwen and Wray's method, Kraaijevanger and Spijker's method and Radau's method. The next chapter will discuss the errors made in the approximations more thoroughly and validate the local truncation errors mentioned in Subsection 3.2.1.

5

Global truncation error analysis

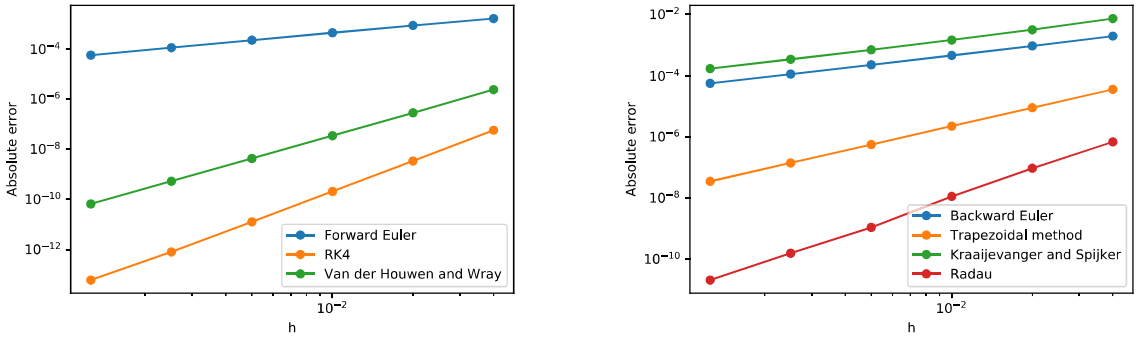
In Chapter 3 the local truncation errors of several Runge-Kutta methods were given. The local truncation error is caused by one iteration of a numerical method. Another relevant error of numerical methods is the global truncation error, which is the cumulative error caused by many iterations and can be found by analysing the difference between the analytical solution and the numerical approximation. Chapter 4 discussed algorithms to compute a numerical approximation of the solution of an IVP. The results of the algorithms will be used in this chapter to analyse the global truncation error.

5.1. Absolute error analysis

The error e_n at time t_n is given by the difference between the analytical value y_n and the numerical approximation w_n. As the numerical approximation may either be an overestimation or an underestimation, the absolute value of the error should be considered, that is $e_n = |y_n - w_n|$. It should be noted that this computation requires the analytical solution to the considered IVP to be known, which is not always true. The error can also be written as $e_n = c \cdot h^p$, where c is a positive constant, h is the step size and p is the order of the global truncation error of the method. Combining the two expressions for e_n, we find $|y_n - w_n| = c \cdot h^p$. To find the order p of the method, one can take the logarithm of both sides and rewrite the equation to find a linear relation between $\log(e_n)$ and $\log(h)$.

$$\begin{aligned} \log(|y_n - w_n|) &= \log(c \cdot h^p) \\ &= \log(c) + p \cdot \log(h). \end{aligned} \tag{5.1}$$

In Equation (5.1) it can be seen that the order p of the truncation error corresponds to the slope of the logarithmic graph. We will consider IVP (4.1) and the computation of the numerical approximation as described in Chapter 4. The logarithmic graph of the absolute error is shown in Figure 5.1. The computations are done for $h \in [0.04, 0.02, 0.01, 0.005, 0.0025, 0.00125]$.



(a) Absolute error of explicit methods.

(b) Absolute error of (diagonally) implicit methods.

Figure 5.1: Absolute error of the numerical approximation of IVP (4.1) of seven Runge-Kutta methods methods.

We notice that the relation between the step size and the error on logarithmic scale seems linear. This behaviour was expected by Equation (5.1). One can use the graphs in Figure 5.1 to find the global truncation error by calculating the slope of each graph. One way to determine the slope is by linear interpolation. We have determined the slope by considering the error for $h = 0.04$ and $h = 0.00125$. The resulting slope of each method is given in Table 5.1.

Method	Type	Slope	Global truncation error	Local truncation error
Forward Euler	ERK	0.97	$\mathcal{O}(h)$	$\mathcal{O}(h)$
Backward Euler	DIRK	1.03	$\mathcal{O}(h)$	$\mathcal{O}(h)$
Trapezoidal	DIRK	2.00	$\mathcal{O}(h^2)$	$\mathcal{O}(h^2)$
RK4	ERK	3.97	$\mathcal{O}(h^4)$	$\mathcal{O}(h^4)$
Van der Houwen and Wray	ERK	3.03	$\mathcal{O}(h^3)$	$\mathcal{O}(h^3)$
Kraaijevanger and Spijker	DIRK	1.09	$\mathcal{O}(h)$	$\mathcal{O}(h)$
Radau	IRK	3.00	$\mathcal{O}(h^3)$	$\mathcal{O}(h^3)$

Table 5.1: The global and local truncation error of seven Runge-Kutta methods.

We notice that the order of the global truncation error and local truncation error of most methods are the same. Vuik et al., 2007 describes Lax's equivalence theorem, which states that the global truncation error and the local truncation error are of the same order if a numerical method is stable and consistent.

The next chapter will summarise the analysis from the previous chapters. Furthermore, it will discuss recommendations for future work.

6

Summary and recommendations

In this chapter, a summary of the thesis will be given. Furthermore, several recommendations for future analysis of Runge-Kutta methods will be discussed.

6.1. Summary

In this thesis we have analysed Runge-Kutta methods using Butcher tableaus. A Runge-Kutta method is a numerical method used to solve initial value problems. Runge-Kutta methods famously compute the value of the next time step using a certain number of stages. A method is specified by certain coefficients which can be displayed in a Butcher tableau.

A Runge-Kutta method can be classified as either an explicit or an implicit method. A special kind of implicit methods are diagonally implicit methods. To provide the reader with a better idea for the types of Runge-Kutta methods, some examples of methods are given. These examples are used throughout the thesis to do computations.

The amplification factor of a Runge-Kutta method can be computed by substitution of the test equation. The amplification factor of an explicit method can be written as a polynomial where the order of the polynomial is at most as high as the number of stages. It is likely, although not proven in this thesis, that the amplification factor of an implicit method can be written as a rational function.

The amplification factor can be used to find the stability region of a Runge-Kutta method. As implicit methods are unconditionally stable, the interest lies in finding the stability region of explicit methods. An algorithm is provided to find the values of the step size for which explicit methods are stable.

Two numerical errors that are analysed are the local truncation error and the global truncation error. The local truncation error, which is caused by one iteration of the method, can be computed using the amplification factor. The global truncation error is the cumulative error caused by many iterations. In order to find the global truncation error, time steps of the Runge-Kutta methods need to be compared to the analytical solution.

Each time step of an explicit Runge-Kutta methods can be found by the straight-forward process of computing the values of each stage. The computation of a time step of an implicit method is less straight-forward. For diagonally implicit methods an algorithm is given that used the Picard iteration as described in Vuik et al., 2007. The algorithm given to perform a time step of an implicit method is similar to that of the algorithm for a diagonally implicit method and also uses the Picard iteration.

To analyse the global truncation error, a graph of the step size as a function of the error is made. By plotting both axes on a logarithmic scale and determining the slope, the global truncation error is approximated.

6.2. Recommendations

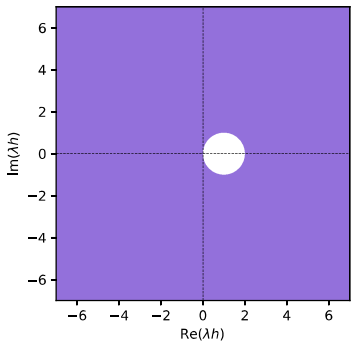
This section provides options to extend analysis of Runge-Kutta methods with Butcher tableaus. Although efforts were made to analyse Runge-Kutta methods thoroughly, there are possibilities for improvements that may be relevant in further research.

- Theorem 1 states that the amplification factor of an explicit Runge-Kutta method can be written as a polynomial of at most order s in the indeterminate λh. It is likely that the amplification factor of an implicit Runge-Kutta method can be written as a rational function. Further research could include a proof to justify this statement.
- To find the root of $|Q(\lambda h)|-1$ for explicit methods we assumed that the values of the stability region with non-positive real part form a star-shaped subset of the complex plane about the origin. In order to justify the assumption, a proof should be given that the statement is true for any explicit method. Alternatively, a classification could be given such that certain methods do satisfy the assumption and Algorithm 1 can be used for those methods.
- Algorithm 2 and Algorithm 3 use the Picard iteration to approximate the value of each k_i. The conditions that need to be met to ensure uniqueness are dependent on the value of step size h. Further research could include an approach to determine which values of h guarantee that the conditions for uniqueness are met.
- The order of the global truncation error as computed in Chapter 5 required the analytical solution to the IVP to be given, which is true for the IVP that was considered. One can also analyse the order of the global truncation error of a Runge-Kutta method for IVPs of which the analytical solution is not known. This can be done by applying the Runge-Kutta method for several value of the step size and considering the approximation at t_{end}. This approach is described in more detail in Section 6.6 of Vuik et al., 2007.
- In this thesis, the computations were only considered for first order initial value problems. The computations could be extended to work for higher order initial value problems. This can be done by writing them as a system of first order initial value problems.

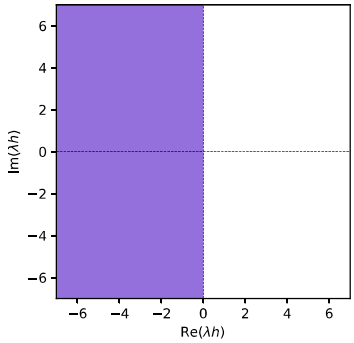
A

Stability region of some methods

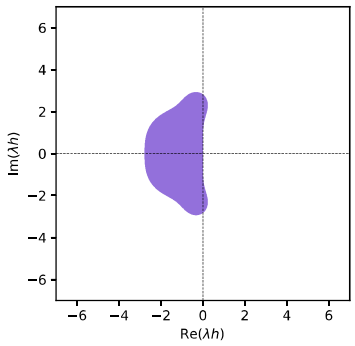
In Chapter 3 the computation of the stability reiong of Runge-Kutta methods was described. We plot the stability regions of the Backward Euler, Trapezoidal and RK4 method.



(a) Stability region of the Backward Euler method.



(b) Stability region of the trapezoidal method.



(c) Stability region of RK4.

Bibliography

Braun, M. (1992). *Differential equations and their applications*. Springer New York, NY. https://doi.org/https://doi.org/10.1007/978-1-4612-4360-1

Hairer, E., & Wanner, G. (2015). Radau methods. https://doi.org/10.1007/978-3-540-70529-1_139

Kraaijevanger, J., & Spijker, M. (1989). Algebraic stability and error propagation in runge-kutta methods. *Applied Numerical Mathematics*, *5*(1), 71–87. https://doi.org/https://doi.org/10.1016/0168-9274(89)90025-1

Rehman, A. U., Rehman, S. U., & Ahmad, J. (2020). Numerical approximation of stiff problems using efficient nested implicit runge–kutta methods. *Punjab University Journal of Mathematics*, *51*(2).

Straubing, H. (1983). A combinatorial proof of the cayley-hamilton theorem. *Discrete Mathematics*, *43*(2), 273–279. https://doi.org/https://doi.org/10.1016/0012-365X(83)90164-4

van der Houwen, P. J. (1972). Explicit runge-kutta formulas with increased stability boundaries. *Numerische Mathematik*, *20*(2), 149–164. https://doi.org/10.1007/bf01404404

Vuik, C., Vermolen, F., van Gijzen, M., & Vuik, M. (Eds.). (2007). *Numerical methods for ordinary differential equations*. TU Delft OPEN. https://doi.org/https://doi.org/10.5074/t.2023.001