BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF MECHANICAL ENGINEERING
FAKULTA STROJNÍHO INŽENÝRSTVÍ

INSTITUTE OF MATHEMATICS
ÚSTAV MATEMATIKY

RUNGE-KUTTA METHODS
RUNGE-KUTTOVY METODY

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR — **Bc. Tereza Kroulíková**
AUTOR PRÁCE

SUPERVISOR — **Mgr. Jitka Zatočilová, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2017

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF MECHANICAL ENGINEERING

Specification Master's Thesis

Department:	Institute of Mathematics
Student:	**Bc. Tereza Kroulíková**
Study programme:	Applied Sciences in Engineering
Study field:	Mathematical Engineering
Leader:	**Mgr. Jitka Zatočilová, Ph.D.**
Academic year:	2017/18

Pursuant to Act no. 111/1998 concerning universities and the BUT study and examination rules, you have been assigned the following topic by the institute director Master's Thesis:

Runge–Kutta methods

Concise characteristic of the task:

The Runge–Kutta methods are an important family of explicit and implicit iterative methods for the approximation of solutions of ODE's, that were developed around 1900 by the german mathematicians C. Runge and M.W. Kutta. The most widely known member of the explicit Runge–Kutta family is the fourth order Runge–Kutta method generally referred to as "RK4".
But for the solution of stiff equations are explicit Runge–Kutta methods generally unsuitable, because their region of absolute stability is small. Therefore it is better to use for the solution of stiff equations more stable implicit Runge–Kutta methods. The implicit Runge–Kutta methods are divided to several subclasses, all specified by the structure of the A matrix of the Butcher tableau.

Goals Master's Thesis:

1. Introduction of the explicit and implicit Runge–Kutta methods and their subclasses.
2. Order conditions for Runge–Kutta methods.
3. Adaptive Runge–Kutta methods.
4. Stability Runge–Kutta methods.
5. Programming selected Runge–Kutta methods and testing them on examples.

Recommended bibliography:

BUTCHER, J.C. Numerical Methods for Ordinary Differential Equations. Second edition. Chichester: John. Wiley & Sons, Ltd., 2008. ISBN 978-0470723357.

HAIRER, E., S. P. NORSETT and G.WANNER. Solving Ordinary Differential Equations I: Nonstiff Problems. Second edition. Berlin: Springer-Verlag, 1993. ISBN 978-3642051630.

HAIRER, E., S. P. NORSETT and G.WANNER. Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems. Second edition. Berlin: Springer-Verlag, 1996. ISBN 978-3642052200.

Deadline for submission Master's Thesis is given by the Schedule of the Academic year 2017/18

In Brno,

L. S.

prof. RNDr. Josef Šlapal, CSc.
Director of the Institute

doc. Ing. Jaroslav Katolický, Ph.D.
FME dean

Summary
This thesis deals with Runge–Kutta methods for initial value problem. It starts with analysis of Euler method and the order conditions are derived. The modified methods are presented. For two of them is done theoretical examination of order and for all of them, the order is tested numerically. Embedded methods and methods with error estimation based on modified method are presented and numerically tested. In the second part the implicit methods are derived. Then two approaches of constructing implicit embedded methods is presented. Also diagonal implicit method are introduced. Finally, two kinds of stability of presented method is discussed.

Keywords
Runge–Kutta methods, modified methods, methods with error estimation, stiff problems, stability

Abstrakt
Tato práce se zabývá Runge–Kuttovými metodami pro počáteční problém. Práce začíná analýzou Eulerovy metody a odvozením podmínek řádu. Jsou představeny modifikované metody. Pro dvě z nich je určen jejich řád teoreticky a pro všechny je provedeno numerické testování řádu. Jsou představeny a numericky testovány dva typy metod s odhadem chyby, "embedded" metody a metody založené na modifikovaných metodách. V druhé části jsou odvozeny implicitní metody. Jsou představeny dva způsoby konstrukce implicitních "embedded" metod. Jsou zmíněny také diagonální implicitní metody. Na závěr jsou probrány dva druhy stability u metod prezentovaných v práci.

Klíčová slova
Runge–Kuttovy metody, modifikované metody, metody s odhadem chyby, tuhé problémy, stabilita

KROULÍKOVÁ, T. *Runge-Kuttovy metody.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2018. 93 s. Vedoucí diplomové práce Mgr. Jitka Zatočilová, Ph.D..

Rozšířený abstrakt

Tato práce se zabývá Runge–Kuttovými metodami pro počáteční problém obyčejných diferenciálních rovnic prvního řádu:

$$y'(t) = f(t, y(t)), \quad t \in [t_0, T] \subset \mathbb{R},$$
$$y(t_0) = y_0,$$

kde funkce $f : [t_0, T] \times \mathbb{R}^m \to \mathbb{R}^m$, $y : [t_0, T] \to \mathbb{R}^m$ a vektor počátečních hodnot $y_0 \in \mathbb{R}^m$.

Runge–Kuttovy metody jsou jednokrokové metody dané přepisem:

$$k_i = f\left(t_n + c_i h, y_n + h \sum_{j=1}^{s} a_{ij} k_j\right) \qquad i = 1, ..., s$$
$$y_{n+1} = y_n + h \sum_{j=1}^{s} b_j k_j,$$

kde k_i jsou takzvané stupně. Tyto metody se staly velmi oblíbené. Zejména explicitní metody s odhadem chyby jsou součástí softwarů pro řešení počátečních problémů.

Historie Runge–Kuttových metod se začala psát v roce 1895, kdy C. Runge publikoval rozšíření myšlenky Eulerovy metody. Zatímco Eulerova metoda aproximuje řešení v levém bodě intervalu, Runge použil lepší formule, jakými jsou obdélníková a lichoběžníková formule a pro výpočet derivace ve středním nebo koncovém bodě použil Eulerovu metodu. Tímto vznikla základní myšlenka Runge–Kuttových metod. Na práci Rungeho navázali K. Heun a W. Kutta, kteří vytvořili metody řádu tři a čtyři a odvodili pro ně podmínky řádu. Později se objevily implicitní Runge–Kuttovy metody, které na rozdíl od těch klasických – explicitních, závisejí na všech stupních metody. Díky své neomezené oblasti stability jsou vhodné pro tuhé systémy. Kompromisem mezi explicitními a implicitními metodami se zdají být metody diagonální, které lze rozdělit do několika skupin. Tyto metody mohou být, stejně jako implicitní metody, stabilní a zároveň jsou méně výpočetně náročné [9].

V této práci je nejprve provedena analýza Eulerovy metody, poté jsou odvozeny podmínky řádu z Taylorova rozvoje pro metody druhého a třetího řádu. Podmínky řádu jsou také obecně zavedeny pomocí teorie stromů a jsou odvozeny metody čtvrtého řádu.

Dále jsou představeny čtyř-stupňové modifikované metody založené na různých druzích průměrů. Konkrétně jde o metody založené na aritmetickém, geometrickém, harmonickém, heronianském, odmocninovém, kontraharmonickém a centroidálním průměru. Všechny tyto metody jsou uváděny jako metody čtvrtého řádu [36]. Pro metody založené na kontraharmonickém a centroidalním průměru je v práci ukázáno, že jsou čtvrtého řádu pouze pro skalární autonomní problém. V případě neautonomní rovnice či systému jsou pouze řádu dva.

Numerický experiment pro zjištění řádu byl proveden pro všech sedm uvedených modifikovaných metod a bylo zjištěno, že všechny metody mají řád čtyři pro autonomní rovnici. Pro dva testovací problémy (nehomogenní rovnice a nehomogenní systém) modifikované metody vykazovaly druhý řád. V případě testovacího problému tři, numerický řád nekonvergoval k žádné hodnotě. Snižováním velikosti kroku nedocházelo vždy ke zlepšení přesnosti, někdy se přesnost i zhoršila. Dále pak v okolí bodů řešení s nulovou derivací docházelo k odskokům řešení. Toto chování je způsobeno dělením dvou čísel blízkých nule. Proto jsou modifikované metody nevhodné pro většinu úloh.

Poté se práce zabývá metodami s odhadem chyby. Nejprve jsou uvedeny embedded metody, které mají společné stupně, ale rozdílné koeficienty pro výpočet výstupu a řád. Rozdíl jejich výstupů se pak používá k odhadu lokální chyby. Následně jsou oba přístupy numericky porovnány na třech testovacích problémech.

V druhé části práce jsou představeny implicitní metody. Tyto metody mají neomezenou oblast stability, takže je lze použít pro řešení tuhých problémů. Na druhou stranu v každém časovém kroku je potřeba řešit systém obecně nelineárních rovnic. Pomocí Butcherových zjednodušujících vztahů jsou odvozeny Gaussovy, Radau IA, Radau IIA a tři skupiny Lobattových metod. Jsou uvedeny dva přístupy pro vytvoření embedded metod. Tyto přístupy jsou následně numericky porovnány. Ukazuje se, že dvojice embedded metod se stejným počtem stupňů, často nadhodnocuje chybu a díky tomu zkracují délku kroku. Závěrem druhé části jsou představeny diagonální metody, které lze rozdělit do několika podskupin.

Závěrem práce je diskutována stabilita představených metod. Je ukázáno, že explicitní Runge–Kuttovy metody nesplňují ani podmínky A-stability a tímto jsou zcela nevhodné pro řešení tuhých systémů.

I declare that I have written this master's thesis all by myself under the direction of my supervisor Mgr. Jitka Zatočilová, Ph.D. using the literature listed in the bibliography.

Bc. Tereza Kroulíková

I would like to express my gratitude to Mgr. Jitka Zatočilová, Ph.D. for the great help and all the useful tips which made this thesis as complete as it is presented.

Bc. Tereza Kroulíková

Contents

Introduction

History of Runge–Kutta methods started in 1895 when C. Runge published his famous work. He improved Euler method, by using midpoint and trapezidal rule instead of simple rectangular formula. To obtain the value of function in the midpoint or endpoint, he used Euler method. Thus the main idea of Runge-Kutta methods, evaluating function multiple times per step, was established. In 1900 K. Heun presented order conditions up to fourth order and one year later, W. Kutta made the analysis up to order five. In the same paper, he presented methods, nowadays known as classical Runge–Kutta method of fourth order. Later, Kutzmannn and Butcher proposed implicit Runge–Kutta methods. First, gauss implicit methods were established, then the Radau and Lobatto type of methods. Then alternatives to full implicit methods appeared, diagonaly implicit Runge–Kutta methods. Their stages can be evaluated in sequence rather than a big system [9].

The goals of this thesis are to give the overview of Runge–Kutta methods, to derive order conditions, to introduce methods suitable for adaptive step size strategy, to program them and test and to discus the stability. This thesis starts with the analysis of Euler method, then continues with derivation of order conditions and fourth order methods. As alternative to classical explicit Runge–Kutta methods, the modified methods based on various kinds of means are presented. For methods based on the contraharmonic and centroidal mean the order is examined theoretically. The numerical order is determined for all presented modified methods. Then two strategies for estimating the local error are presented and numerically compared. It is important to have some error estimation for step size selection to avoid unnecessary computational work.

The implicit methods are derived, namely, methods based on Gaussian, Radau and Lobatto quadrature. Two approaches of derivation embedded methods are presented and then they are tested. In the end, the stability of presented method is discussed. The stress is put on A-stability. Then the concept of L-stability is mentioned. The part of the thesis is collection of MATLAB codes, providing the numerical experiments, which results are commented in this work.

1 Introduction to Runge–Kutta Methods

Runge–Kutta (RK) methods are family of well-known one-step numerical methods used for approximate solution of ordinary differential equation.

Let us consider the initial value problem:

$$\begin{aligned} y'(t) &= f(t, y(t)), \quad t \in [t_0, T] \subset \mathbb{R}, \\ y(t_0) &= y_0 \end{aligned} \tag{1.1}$$

where functions $f : [t_0, T] \times \mathbb{R}^m \to \mathbb{R}^m$ and $y : [t_0, T] \to \mathbb{R}^m$ and the initial value vector $y_0 \in \mathbb{R}^m$.

Numerical methods for solving (1.1) will find an approximate solution $y(t)$ at a discrete set of nodes

$$t_0 < t_1 < ... < t_N \leq T.$$

The distance between two neighbouring nods $h_n = t_{n+1} - t_n$, $n = 0, 1, ...N - 1$ is called step size. In case that all step sizes are equal $h_n = h = \frac{T - t_0}{N}$ also relation $t_n = t_0 + nh$, $n = 0, 1, ..., N$ holds. In the following the approximate solution in the node t_n will be denoted by y_n whereas the exact solution will be denoted by $y(t_n)$. To obtain values of $y(t)$ at points other than in the set above, it is necessary to use some kind of interpolation.

Now Runge–Kutta methods can be presented.

Definiton 1.1. Let s be an integer and a_{ij}, b_j, c_j for $i, j = 1, ...s$ be real coefficients. Then the method

$$\begin{aligned} k_i &= f\left(t_n + c_i h, y_n + h \sum_{j=1}^{s} a_{ij} k_j\right) \qquad i = 1, ..., s \\ y_{n+1} &= y_n + h \sum_{j=1}^{s} b_j k_j \end{aligned} \tag{1.2}$$

is called an **s-stage Runge–Kutta method**.

Coefficients a_{ij}, b_j, c_j in above definition specify each RK method and are usually written in the scheme called Butcher tableau:

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^T \end{array} = \begin{array}{c|cccc} c_1 & a_{1,1} & a_{1,2} & \cdots & a_{1,s} \\ c_2 & a_{2,1} & a_{2,2} & \cdots & a_{2,s} \\ c_3 & a_{3,1} & a_{3,2} & \cdots & a_{3,s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s,1} & a_{s,2} & \cdots & a_{s,s} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}.$$

Usually the coefficients c_i and a_{ij} are connected by relation

$$c_i = \sum_{j=1}^{s} a_{ij}, \quad i = 1, ..., s. \tag{1.3}$$

These conditions express that all points where f is evaluated are first order approximations to the solution.

According to structure of matrix **A** we can specify the subclasses of RK methods. The most general case is group of **implicit RK methods** (IRK), sometimes they are called fully implicit (FIRK). They have almost no zero element (usually none) in the matrix **A**. If the matrix is strictly lower triangular then the method is called **explicit RK method** (ERK). If also the diagonal elements are non-zero, we speak about **diagonally implicit RK methods** (DIRK). In the case that all diagonal elements are equal we call those methods as **singly diagonal implicit RK methods** (SDIRK). Even more special type of previous subclass is class of **explicit singly diagonal implicit RK methods** (ESDIRK). These methods have only zeros in the first row of matrix **A**. For example if we have three stages RK method, then the matrices **A** are as following:

$$\begin{pmatrix} 0 & 0 & 0 \\ a_{2,1} & 0 & 0 \\ a_{3,1} & a_{3,2} & 0 \end{pmatrix} \quad \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix} \quad \begin{pmatrix} a_{1,1} & 0 & 0 \\ a_{2,1} & a_{2,2} & 0 \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix}$$

Explicit RK (Fully) Implicit RK Diagonaly Implicit RK

$$\begin{pmatrix} \gamma & 0 & 0 \\ a_{2,1} & \gamma & 0 \\ a_{3,1} & a_{3,2} & \gamma \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 0 \\ a_{2,1} & \gamma & 0 \\ a_{3,1} & a_{3,2} & \gamma \end{pmatrix} .$$

Singly Diagonal Implicit RK Explicit Singly Diagonal Implicit RK

One could notice that explicit RK are the simplest among those subclasses, since there is no system of equations which has to be solved. On the other hand with fully implicit RK we get the system of ms non-linear equations in each time step. Naturally a question can arise: why do we not use only explicit RK method? As it will be shown later, explicit methods have small area of stability, which makes problems mainly in case of **stiff** equations.

There is a problem with defining the stiffnes. It is not possible to make a strict borderline between non-stiff and stiff equations. So there also exist problems called mildly-stiff. Despite this, there are several approaches how to recognize the stiff problem. Professor J.C. Butcher in his book [7] said that the stiff problems usually have large Lipschitz constant. Another interesting approach is by Randall J. LeVeque [26]. He characterised the stiff problem as one where $\frac{\partial}{\partial t} f(t, y)$ is much larger in norm value than $y'(t)$. One possibility to check this, is calculating the ratio between the largest and the smallest eigenvalue of the Jacobian matrix $f'(t, y(t))$.

$$\frac{\max|\lambda_p|}{\min|\lambda_p|}, \quad p = 1, ..., m$$

This ratio is called stiffness ratio. If this value is large problem might be very stiff but unfortunately it does not hold every time. For example in case of scalar problem the stiffness ratio is always one, but may be also stiff. Opposite difficulty occurs too, the value may be large even though the problem is not stiff at all.

2 | Explicit Runge–Kutta Methods

This chapter deals with first group of Runge–Kutta methods – Explicit RK methods. We will start with analysis the oldest numerical method for ordinary differential equations, the Euler method. Then we show derivation of RK methods of second, third and fourth order and also order conditions. The methods based on various kinds of means will be introduced and tested on test problems. The important part is methods with error estimation. We will present classical embedded methods and combination of modified method.

2.1 First Order Runge–Kutta Method – Euler Method

The very first method for solving initial value problem was Euler method. It was published in his work *Institutiones Calculi Integralis* in the year 1770. The idea behind is based on very simple principle. For example, we assume a particle moving in such way that in time t_0 is placed in position y_0 and at this time the velocity v_0 is known. In the really short period of time, the velocity does not change significantly from v_0. So the position can be approximated by the time change multiplied by initial velocity v_0:

$$y_1 = y_0 + (t_1 - t_0)v_0.$$

It is possible to compute y_2 analogously with velocity $v_1 = v(t_1, y_1)$ and then continue the sequence of approximations $y_3, y_4, \ldots$ [7].

Formally Euler method can be derived from forward difference approximation of derivative

$$y'(t) \approx \frac{y(t+h) - y(t)}{h}$$

if we apply it to (1.1) at point $t = t_n$ we obtain

$$\begin{aligned} \frac{y(t_n + h) - y(t_n)}{h} &\approx f(t_n, y(t_n)) \\ y(t_{n+1}) &\approx y(t_n) + hf(t_n, y(t_n)). \end{aligned}$$

Now we substitute the exact value $y(t_n)$ with the approximated one y_n and by taking the relation exact we obtain Euler method:

$$y_{n+1} = y_n + hf(t_n, y_n), \tag{2.1}$$

which is Runge–Kutta method with one stage with the following Butcher tableau

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}.$$

But using the difference scheme is not the only way how to derive Euler method. We also can turn the differential equation

$$y' = f(t, y)$$

into integral one

$$\int_{t_n}^{t_n+h} y'(t)\mathrm{d}t = \int_{t_n}^{t_n+h} f(t, y)\mathrm{d}t \tag{2.2}$$
$$y(t_n + h) - y(t_n) = \int_{t_n}^{t_n+h} f(t, y)\mathrm{d}t.$$

Now we approximate the right hand side by the left hand rectangle method

$$\int_{t_n}^{t_n+h} f\mathrm{d}t \approx hf(t_n, y(t_n))$$

which will lead again to Euler scheme (2.1).

Another possibility of derivation is to consider the Taylor expansion of the function y around t_n:

$$y(t_n + h) = y(t_n) + hy'(t_n) + \frac{1}{2}h^2y''(t_n) + \mathcal{O}(h^3). \tag{2.3}$$

From differential equation in (1.1) we know that $y' = f(t, y)$ so we use it in (2.3) and ignore the higher-order terms thus we obtain

$$y(t_n + h) = y(t_n) + hf(t_n, y(t_n))$$

After substituting y_{n+1} and y_n for $y(t_n + h)$ and $y(t_n)$ we get again (2.1).

We know that the Euler method is one stage RK method. Now we show also its order of convergence. We start with the local truncation error. We need to compute how approximated result differs from the exact one assuming that the approximated value in previous step is equal to the exact one $y(t_n) = y_n$. From above derivation we have Taylor expansion of exact solution

$$y(t_n + h) = y(t_n) + hf(t_n, y_n) + \frac{1}{2}h^2y''(t_n) + \mathcal{O}(h^3),$$

from which we subtract approximated solution obtained by (2.1) and we get

$$|y(t_n + h) - y_{n+1}| = \frac{1}{2}h^2y''(t_n) + \mathcal{O}(h^3),$$

thus for bounded second derivation of y we have local truncation error proportional to $\mathcal{O}(h^2)$.

After many steps the local truncation errors will accumulate and we will observe phenomena called global truncation error. It is and actual difference between the exact solution and the approximated one after n steps.

If we subtract euler approximation (2.3) from Taylor expansion of exact solution (2.1) we get

$$e_{n+1} = y(t_{n+1}) - y_{n+1} = y(t_n) - y_n + h(f(t_n, y(t_n)) - f(t_n, y_n)) + \frac{h^2}{2}y''(t_n) + \mathcal{O}(h^3).$$

Assume that f is Lipschitz continuous in second variable $|f(t,y_1)-f(t,y_2)| \leq L|y_1-y_2|$ where L is so-called Lipschitz constant. We take bounds

$$|e_{n+1}| \leq |e_n| + hL|e_n| + \frac{h^2}{2}\max|y''(t)|.$$

If the previous inequality is applied recursively we obtain

$$|e_n| \leq (1+hL)^n|e_0| + \left(1+(1+hL)+...+(1+hL)^{n-1}\right)\frac{h^2}{2}\max|y''(t)|$$

and then we simplify it by using formula for a finite geometric sum

$$|e_n| \leq (1+hL)^n|e_0| + \left(\frac{(1+hL)^n-1}{L}\right)\frac{h^2}{2}\max|y''(t)|. \tag{2.4}$$

To continue we need following lemma:

Lemma 2.1. *For any real t*

$$1+t \leq e^t$$

and for any $t \geq -1$ and any $m \geq 0$

$$0 \leq (1+t)^m \leq e^{mt}.$$

Proof. See [2]. □

Now using Lemma 2.1 we have $(1+hL)^n \leq e^{nhL} = e^{t_n - t_0 L}$ and the relation (2.4) becomes

$$|e_n| \leq e^{(t_n-t_0)L}|e_0| + h\frac{\max|y''(t_n)|\left(e^{(t_n-t_0)L}-1\right)}{2L}.$$

If we have "correct" initial condition $y(t_0)=y_0$, then the first term disappear since $|e_0|=0$. We see that $|e_n| \leq Ch$ so the global error of the Euler's method is proportional to step size h and we can say that it has order 1.

2.2 Methods of Order 2 and 3

In previous section we saw simple method to treat differential equations numerically. But we also showed that the method is quite poor in sense of order. If we want to compute some descent precision, we need a lot of steps (possibly thousands). Generally there are two approaches how to improve Euler method. First one is to make the approximated solution dependent on more values which leads to multi-step method. The second one uses evaluating right hand side f of equation in (1.1) in more points, which is principle of Runge–Kutta methods.

We start this section with the derivation of specific second order method and then we set up general conditions for second and third order RK methods.

For deriving better method, we evolve the idea from the previous section. We turn the differential equation (1.1) into integral one (exactly as we did in the previous section) but now we use trapezoidal rule (since it has better accuracy than left rectangle method used in case of Euler method)

$$\int_{t_n}^{t_{n+1}} f(t,y(t))\mathrm{d}t \approx (t_{n+1}-t_n)\left[\frac{f(t_n,y_n)+f(t_{n+1},y_{n+1})}{2}\right]$$

to approximate the integral on the right hand side (2.2) and we obtain

$$y(t_{n+1}) \approx y_n + \frac{h}{2}\left[f(t_n, y(t_n)) + f(t_{n+1}, y(t_{n+1}))\right].$$

Unfortunately right hand side involves undetermined value $y(t_{n+1})$. For lack of better solution of this problem we use Euler method to determine it. The resulting formula

$$y_{n+1} = y_n + \frac{h}{2}\left[f(t_n, y_n) + f(t_{n+1}, y_n + hf(t_n, y_n)\right]$$

is called Heun's method or improved Euler method. We can also rewrite it into form of RK methods

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f(t_n + h, y_n + hk_1) \\ y_{n+1} &= y_n + h\left(\frac{1}{2}k_1 + \frac{1}{2}k_2\right) \end{aligned} \tag{2.5}$$

so we can easily see that it is 2-stage RK method and its Butcher tableau is

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}.$$

By adding more and more stages, we naturally want new methods to be more accurate, to have higher order. The task of this section will be to set the conditions of coefficients a_{ij}, b_j, c_j to obtain method of order 2 or 3. We will start with two-stage RK method

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f(t_n + c_2 h, y_n + a_{21} h k_1) \\ y_{n+1} &= y_n + h(b_1 k_1 + b_2 k_2) \end{aligned} \tag{2.6}$$

and three-stage RK method

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f(t_n + c_2 h, y_n + a_{21} h k_1) \\ k_3 &= f(t_n + c_3 h, y_n + a_{31} h k_1 + a_{32} h k_2) \\ y_{n+1} &= y_n + h(b_1 k_1 + b_2 k_2 + b_3 k_3). \end{aligned} \tag{2.7}$$

both of them applied to a scalar problem.

To find out the error in the single step of the method we need to compare successive terms in Taylor expansions of the exact solution and approximate one. As we will compare the errors in the single step in this section we assume the equality of the exact value of function $y(t)$ and the approximated one $y(t_n) = y_n$ in the point t_n.

We start with the exact solution. First the Taylor expansion of exact solution $y(t)$ around point t_n is needed

$$y(t_n + h) = y(t_n) + hy'(t_n) + \frac{h^2}{2}y''(t_n) + \frac{h^3}{6}y'''(t_n) + \mathcal{O}(h^4). \tag{2.8}$$

From (1.1) we know that $y' = f$, thus for higher derivatives $y'' = f'$, $y''' = f''$ apply and we use this to evaluate the above Taylor expansion. Since we work with the function of two variables to

avoid confusion the total derivative of function $f(t,y)$ will be denoted by apostrophe $f' = \frac{\mathrm{d}f}{\mathrm{d}t}$ and the partial derivative (according to t) by subscript $f_t = \frac{\partial f}{\partial t}$. Now we make the first derivative of function $f(t,y)$:

$$f' = \frac{\mathrm{d}f}{\mathrm{d}t} = f_t + f_y y' = f_t + f_y f$$

and the second one

$$f'' = f_{tt} + 2f_{ty}y' + f_{yy}\left(y'\right)^2 + f_y y'' = f_{tt} + 2f_{ty}f + f_{yy}\left(f\right)^2 + f_y f'.$$

For simplifying the above expressions we set

$$F = f_t + f_y f \text{ and } G = f_{tt} + 2f_{ty}f + f_{yy}\left(f\right)^2. \tag{2.9}$$

We plug them into (2.8) and obtain Taylor expansion of exact solution y around point t_n:

$$y(t_n + h) = y(t_n) + hf(t_n, y_n) + \frac{h^2}{2}F(t_n, y_n) + \frac{h^3}{6}\left(G(t_n, y_n) + f_y(t_n, y_n)F(t_n, y_n)\right) + \mathcal{O}(h^4). \tag{2.10}$$

Now we need Taylor expansion of approximate solution around the same point. That means need of expansions of stages k_i, $i = 1, 2, 3$. The first one is really easy since there is nothing to expand:

$$k_1 = f(t_n, y_n) \tag{2.11}$$

For the second stage we recall multivariate Taylor expansion of function f around point (t, y):

$$f(t + h, y + k) = f(t, y) + hf_t(t, y) + kf_y(t, y) + \frac{h^2}{2}f_{tt}(t, y) + \frac{2hk}{2}f_{ty}(t, y) + \frac{k^2}{2}f_{yy}(t, y).$$

If we use $a_{21} = c_2$ the Taylor expansion of stage k_2 is

$$\begin{aligned} k_2 = f(t_n, y_n) &+ c_2 h f_t(t_n, y_n) + c_2 h f(t_n, y_n) f_y(t_n, y_n) + \frac{c_2^2 h^2}{2} f_{tt}(t_n, y_n) \\ &+ \frac{2c_2^2 h^2}{2} f(t_n, y_n) f_{ty}(t_n, y_n) + \frac{c_2^2 h^2}{2} f^2(t_n, y_n) f_{yy}(t_n, y_n) + \mathcal{O}(h^3) \end{aligned}$$

and after using (2.9) we get

$$k_2 = f(t_n, y_n) + c_2 h F(t_n, y_n) + \frac{c_2^2 h^2}{2} G(t_n, y_n) + \mathcal{O}(h^3). \tag{2.12}$$

For 2-stage method it is enough, therefore we can write down Taylor expansion of (2.6) around t_n

$$y(t_n + h) = y(t_n) + h(b_1 + b_2)f(t_n, y_n) + h^2(b_2 c_2)F(t_n, y_n) + \frac{c_2^3 h^3}{2} G(t_n, y_n) + \mathcal{O}(h^4). \tag{2.13}$$

After comparison of first three terms of (2.10) and (2.13) we obtain conditions

$$b_1 + b_2 = 1 \tag{2.14}$$

$$b_2 c_2 = \frac{1}{2} \tag{2.15}$$

which need to hold for method to be second order method. If we compare the fourth terms

$$\frac{h^3}{6}\left(G(t_n,y_n)+f_y(t_n,y_n)F(t_n,y_n)\right)=\frac{c_2^3h^3}{2}G(t_n,y_n)$$

we see that the equality does not hold for general function f. So to obtain third order method we need method with more stages.

Now we can check that introduced Heun's method is second order:

$$\begin{aligned}\frac{1}{2}+\frac{1}{2}&=1\\ \frac{1}{2}\cdot 1&=\frac{1}{2}.\end{aligned}$$

Going back to making Taylor expansions of stages. After some similar computation as we did for previous stages we obtain

$$\begin{aligned}k_3=&f(t_n,y_n)+c_3hf_t(t_n,y_n)+(a_{31}hk_1+a_{32}hk_2)f_y(t_n,y_n)+\frac{c_3^2h^2}{2}f_{tt}(t_n,y_n)\\&+\frac{2c_3(a_{31}hk_1+a_{32}hk_2)}{2}f_{ty}(t_n,y_n)+\frac{(a_{31}hk_1+a_{32}hk_2)^2}{2}f_{yy}(t_n,y_n)+\mathcal{O}(h^3)\end{aligned}$$

Now we use $c_3=a_{31}+a_{32}$ and expansions of previous stages k_1,k_2 and plug them into the above expression. We also multiply the brackets, put together terms with the same power of h and neglect terms with higher powers of h. Thus we obtain

$$k_3=f(t_n,y_n)+c_3hF(t_n,y_n)+h^2\left(a_{32}c_2f_y(t_n,y_n)F(t_n,y_n)+\frac{1}{2}c_3^2G(t_n,y_n)\right)+\mathcal{O}(h^3). \quad (2.16)$$

By putting expansions of stages (2.11), (2.12) and (2.16) into expression for y_{n+1} (2.7) we get the Taylor expansion of the approximate solution around point (t_n,y_n):

$$\begin{aligned}y(t_n+h)=&y(t_n)+h(b_1+b_2+b_3)f(t_n,y_n)+h^2(b_2c_2+b_3c_3)F(t_n,y_n) &(2.17)\\&+\frac{h^3}{2}\left[2b_3a_{32}c_2f_y(t_n,y_n)F(t_n,y_n)+(c_2b_2^2+c_3^2b_3)G(t_n,y_n)\right]+\mathcal{O}(h^4). &(2.18)\end{aligned}$$

If we compare expansions (2.10) and (2.17) we get following conditions under which 3-stage method is 3rd order method

$$\begin{aligned}b_1+b_2+b_3&=1\\ b_2c_2+b_3c_3&=\frac{1}{2}\\ b_2c_2^2+b_3c_3^2&=\frac{1}{3}\\ b_3a_{32}c_2&=\frac{1}{6}.\end{aligned} \quad (2.19)$$

An example of three stages RK method is method given by tableau:

$$\begin{array}{c|ccc}0&&&\\ \frac{1}{2}&\frac{1}{2}&&\\ 1&-1&2&\\ \hline &\frac{1}{6}&\frac{2}{3}&\frac{1}{6}\end{array}.$$

We can convince ourself that is also 3rd order RK method.

$$\begin{aligned}
\frac{1}{6}+\frac{2}{3}+\frac{1}{6} &= 1\\
\frac{2}{3}\cdot\frac{1}{2}+\frac{1}{6}\cdot 1 &= \frac{1}{2}\\
\frac{2}{3}\cdot\left(\frac{1}{2}\right)^2+\frac{1}{6}\cdot 1^2 &= \frac{1}{3}\\
\frac{1}{6}\cdot 2\cdot\frac{1}{2} &= \frac{1}{6}.
\end{aligned}$$

In the second part of this section, we will again derive order condition. But in this case it is about vector problem and method without specified number of stages s. Even in previous case of scalar problems, we see that the conditions would be more complicated as desired order would increase. Hence, in next section we will present rooted tree and their correspondence to order conditions. To make this more clear, we will use tensor notation for a while. It means that a component of vector is denoted by superscript and in capitals in order to avoid confusion between partial derivatives and indices of coefficients and stages in approximate solution.

At first, let's make the problem (1.1) autonomous for easier computation

$$\begin{pmatrix} t' \\ y' \end{pmatrix} = \begin{pmatrix} 1 \\ f(t,y) \end{pmatrix}.$$

This can be rewritten in tensor notation as

$$(y^J)' = f^J(y^1,...,y^m), \qquad J = 1,...,m. \tag{2.20}$$

Now we start to make derivatives of the above equality and evaluate them at point y_n. Directly from above equation we have the first derivative of y, we just evaluate it at y_n

$$\left(y^J\right)' = f^J\big|_{y=y_n}. \tag{2.21}$$

Using chain rule we make second and third derivative and again evaluate them at y_n

$$\begin{aligned}
\left(y^J\right)'' &= \sum_{K=1}^{m} f^J_K f^K \Bigg|_{y=y_n}\\
\left(y^J\right)''' &= \sum_{K=1}^{m}\sum_{L=1}^{m} f^J_{KL} f^K f^L \Bigg|_{y=y_n} + \sum_{K=1}^{m}\sum_{L=1}^{m} f^J_K f^K_L f^L \Bigg|_{y=y_n},
\end{aligned} \tag{2.22}$$

where f^K_J denotes $\frac{\partial f^J}{\partial y^K}$.

We want to obtain the derivatives of approximate solution in easier way. In order to get better symmetry, we use another expression of RK methods (1.2). We replace stages k_i with expressions g_i such that $k_i = f(g_i)$ and we obtain

$$\begin{aligned}
g_i^J &= y_n^J + \sum_{j=1}^{i-1} a_{ij} h f^J(g_j^1,...,g_j^m)\\
y_{n+1}^J &= y_n^J + \sum_{j=1}^{s} b_j h f^J(g_j^1,...,g_j^m) \quad i = 1,...,s.
\end{aligned} \tag{2.23}$$

If the autonomous system (2.20) was originally non-autonomous then

$$g_i^1 = y_n^1 + \sum_{j=1}^{i-1} a_{ij} h = t_n + c_i h. \tag{2.24}$$

Thus we see that methods (1.2) and (2.23) are the same.

Expressions in (2.23) are really similar. They differ just by the coefficients thus it is enough to compute just derivatives of g_i with respect to h at $h = 0$. The first derivative according to h

$$\left(g_i^J\right)' = \sum_{j=1}^{i-1} a_{ij} f^J + \sum_{j=1}^{i-1} a_{ij} h \left(f^J\right)' \tag{2.25}$$

and computed at $h = 0$

$$\left(g_i^J\right)'\Big|_{h=0} = \sum_{j=1}^{i-1} a_{ij} f^J \Big|_{y=y_n} \tag{2.26}$$

The second derivative obtained by differentiating the first one (2.25)

$$\left(g_i^J\right)'' = 2 \sum_{j=1}^{i-1} a_{ij} (f^J)' + \sum_{j=1}^{i-1} a_{ij} h (f^J)'' \tag{2.27}$$

to obtain (2.27) at $h = 0$ the derivative of f^J is needed

$$(f^J)' = \sum_{K=1}^{m} f_K^J \left(g_j^K\right)'. \tag{2.28}$$

And by inserting (2.26) into (2.28) we obtain $(f^J)'|_{h=0}$ and we can write down the second derivative of g_i^J at $h = 0$

$$\left(g_i^J\right)''\Big|_{h=0} = 2 \sum_{j=1}^{i-1} \sum_{k=1}^{j-1} a_{ij} a_{jk} \sum_{K=1}^{m} f_K^J f^K \Big|_{y=y_n} \tag{2.29}$$

We continue with the third derivative

$$\left(g_i^J\right)''' = 3 \sum_{j=1}^{i-1} a_{ij} \left(f^J\right)'' + \sum_{j=1}^{i-1} a_{ij} h \left(f^J\right)'''.$$

In this case the second derivative of f^J is needed

$$\left(f^J\right)'' = \sum_{K=1}^{m} \sum_{L=1}^{m} f_{KL}^J (g_j^K)' (g_j^L)' + \sum_{K=1}^{m} f_K^J (g_j^K)'' \tag{2.30}$$

and finally for evaluation in $h = 0$ we evaluate (2.30) with the help of (2.26) and (2.29). We get

$$\left(g_i^J\right)'''\Big|_{h=0} = 3 \sum_{j=1}^{i-1} \sum_{k=1}^{j-1} \sum_{l=1}^{j-1} a_{ij} a_{jk} a_{jl} \sum_{K=1}^{m} \sum_{L=1}^{m} f_{KL}^J f^K f^L \Big|_{y=y_n} + 6 \sum_{j=1}^{i-1} \sum_{k=1}^{j-1} \sum_{l=1}^{k-1} a_{ij} a_{jk} a_{kl} \sum_{K=1}^{m} \sum_{L=1}^{m} f_K^J f_L^K f^L \Big|_{y=y_n}. \tag{2.31}$$

If we substitute a_{ij} by b_j in derivatives of g_i (2.26), (2.29), (2.31) and use condition $c_i = \sum_j a_{ij},\ \ i = 1, ..., s$ we obtain following expressions of derivatives of y_{n+1}.

$$\begin{aligned}
\left(y_{n+1}^J\right)'\Big|_{h=0} &= \sum_{j=1}^{s} b_j \sum_{J=1}^{m} f^J \Bigg|_{y=y_n} \\
\left(y_{n+1}^J\right)''\Big|_{h=0} &= 2\sum_{j=1}^{s} b_j c_j \sum_{J=1}^{m}\sum_{K=1}^{m} f_K^J f^K \Bigg|_{y=y_n} \\
\left(y_{n+1}^J\right)'''\Big|_{h=0} &= 3\sum_{j=1}^{s} b_j c_j^2 \sum_{J=1}^{m}\sum_{K=1}^{m}\sum_{L=1}^{m} f_{KL}^J f^K f^L \Bigg|_{y=y_n} + 6\sum_{j=1}^{s}\sum_{k=1}^{j-1} b_j a_{jk} c_k \sum_{J=1}^{m}\sum_{K=1}^{m} f_K^J f_L^K f^L \Bigg|_{y=y_n}
\end{aligned} \tag{2.32}$$

The comparison of (2.21),(2.22) and (2.32) will lead to these four conditions

$$\begin{aligned}
\sum_{j=1}^{s} b_j &= 1 \\
\sum_{j=1}^{s} b_j c_j &= \frac{1}{2} \\
\sum_{j=1}^{s} b_j c_j^2 &= \frac{1}{3} \\
\sum_{j=1}^{s}\sum_{k=1}^{j-1} b_j a_{jk} c_k &= \frac{1}{6}
\end{aligned}$$

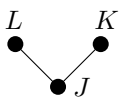
which for choice $s = 3$ give the same conditions as we derived above for the scalar problem (2.19).

2.3 Order Conditions

In previous section, we saw that direct derivation by comparing Taylor expansions of exact and approximated solution is turning more complicated as the order is rising. In this section we want to show how to derive the order conditions with help of labelled rooted trees.

In the previous section in the derivatives of g_i^J the indices j, k, l and J, K, L are linked together as pairs of indices in coefficients $a_{jk}, a_{jl}, ...$ in the same way as upper and lower indices in the derivatives f_{KL}^J, f_K^J. The way how they are linked corresponds to rooted labelled trees.

A labelled rooted tree $\mathbf{t}$ is a connected acyclic graph with one vertex designed as root and every vertex of tree is labelled by index. The symbol τ denotes the tree with only one vertex. We will also consider an empty tree which will be denoted by the symbol of empty set $\emptyset$. Labelled trees can be also represented as mappings. This mapping indicates to which lower letter the corresponding vertices are attached. For example coefficients a_{jk}, a_{jl} and derivative f_{KL}^J correspond to the tree $\mathbf{t}_{31}$ graphically represented as



and as mappings is represented

$$L \mapsto J, \quad K \mapsto J.$$

Definiton 2.1. Let A be an ordered chain of indices $A = \{J < K < L < M < ...\}$ and denote by A_q the subset consisting of the first q indices. *A rooted labelled tree of order* q $(q \geq 1)$ is a mapping

$$\mathbf{t} : A_q \setminus \{J\} \to A_q$$

such that $\mathbf{t}(z) < z$ for all $z \in A_q \setminus \{J\}$.

The set of all labelled trees of order q is denoted by LT_q. We call z the "child" of $\mathbf{t}(z)$ and the $\mathbf{t}(z)$ "parent" of z. The vertex J is called the root of $\mathbf{t}$ and in graphical representation we put it as the lowest vertex. The order q of a labelled tree is equal to the number of its vertices and is denoted by $q = r(\mathbf{t})$.

Definiton 2.2. For a labelled tree $t \in LT_q$ we call

$$F^J(\mathbf{t})(y) = \sum_{K,L,...} f^J_{K,...}(y) f^K_{...} f^L_{...}(y)..$$

the corresponding elementary differential. The summation is over $q-1$ indices $K, L, ...$ (which correspond to $A_q \setminus \{J\}$) and the summand is a product of q *f's* where the upper index runs through all vertices of $\mathbf{t}$ and the lower indices are the corresponding children. We denote by $F(\mathbf{t})(y)$ the vector $(F^1(\mathbf{t})(y), ..., F^m(\mathbf{t})(y))$.

Example 2.1. For the tree $\mathbf{t}_{31}$ the corresponding elementary differential is $\sum_{K,L} f^J_{KL} f^K f^L$.

We already saw elementary differentials, in previous section, when we computed the derivatives of solution. In (2.32) we have f^J in expression for first derivative of y^J_{n+1}, which is elementary differential of tree τ, elementary differential of $\mathbf{t}_{21}$ appeared in second derivative of y^J_{n+1} and even two elementary differentials, corresponding to $\mathbf{t}_{31}$ and $\mathbf{t}_{32}$ (see Table 2.2), are in third derivative of y^J_{n+1}. Graphically we just add a branch to each vertex with new summation index, as is represented, up to fourth order, in the Figure 2.1.

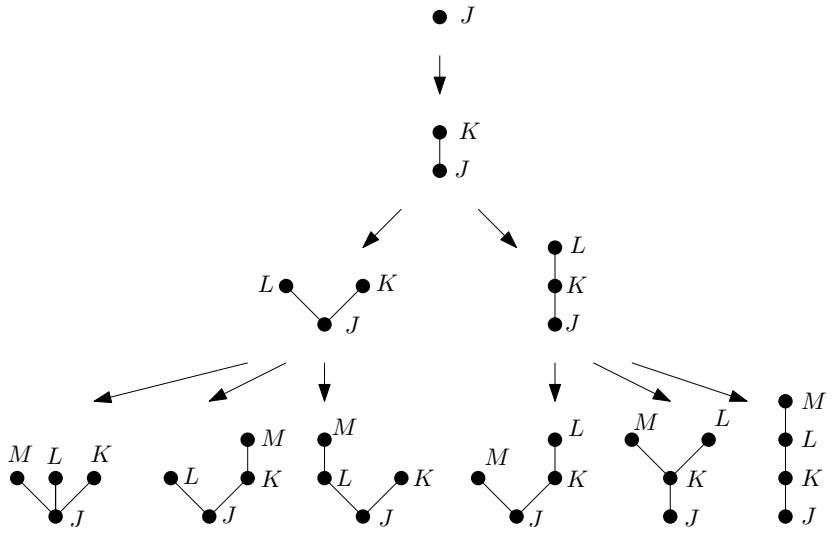


Figure 2.1: Derivatives of exact solution

We can notice that the second, third and fourth labelled tree in the last layer look topologically alike. Also their corresponding elementary differentials

$$\sum_{K,L,M} f^J_{KM} f^M f^K_L f^L, \quad \sum_{K,L,M} f^J_{KL} f^L f^K_M f^M, \quad \sum_{K,L,M} f^J_{LK} f^K f^L_M f^M$$

differ just by an exchange of the summation indices. This bring us to two following definitions.

Definiton 2.3. Two labelled trees $\mathbf{t}_i$ and $\mathbf{t}_j$, $i \neq j$ are *equivalent*, if they have the same order, say q, and if there exists a permutation $\sigma : A_q \to A_q$, such that $\sigma(J) = J$ and $\mathbf{t}_i\sigma = \sigma\mathbf{t}_j$ on $A_q \setminus \{J\}$.

Definiton 2.4. An equivalence class of qth order labelled trees is called *a rooted tree* of order q. The set of all trees of order q is denoted by T_q. The order of a tree is defined as the order of a representative and is again denoted by $r(\mathbf{t})$. Furthermore we denote by $\alpha(\mathbf{t})$ for $\mathbf{t} \in T_q$ the number of elements in the equivalence class $\mathbf{t}$.

In graphical representation a tree is distinguished from a labelled tree by omitting the labels. Rooted trees up to order 5, their corresponding elementary differentials and values $\alpha(\mathbf{t})$ can be found in Table 2.2 and number of trees up to order 10 can be found in Table 2.1.

order q	1	2	3	4	5	6	7	8	9	10
number of trees T_q	1	1	2	4	9	20	48	116	286	719

Table 2.1: Number of trees

In this moment we state the theorem for derivatives of the exact solution.

Theorem 2.2. *The exact solution of (1.1) satisfies*

$$(y)^{(q)}(t_n) = \sum_{\mathbf{t}\in LT_q} F(\mathbf{t})(y_n) = \sum_{\mathbf{t}\in T_q} \alpha(\mathbf{t})F(\mathbf{t})(y_n). \tag{2.33}$$

Proof. See [21]. □

To state similar theorem for approximated solution we need introduce two another function on trees.

Definiton 2.5. Let $\mathbf{t}$ be a labelled tree with root j, we denote by

$$\Phi_j(\mathbf{t}) = \sum_{k,l} a_{jk}a_{\dots}\dots$$

the sum over the $q-1$ remaining indices $k, l, \dots$. The summand is a product of $q-1$ a's, where all parents stand two by two with their children as indices. This sum is called *weighted function*.

Weighted functions $\Phi_j(\mathbf{t})$ for trees up to order 5 are presented in Table 2.2.

Definiton 2.6. For $\mathbf{t} \in LT_q$ let $\gamma(\mathbf{t})$ be the product of $r(\mathbf{t})$ and all orders of the trees which appear, if the roots, one after another, are removed from $\mathbf{t}$.

This definition is demonstrated in Fig. 2.2

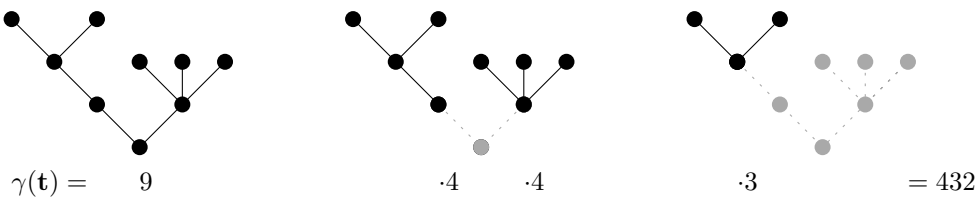


Figure 2.2: Example of computation $\gamma(\mathbf{t})$

Theorem about derivatives of approximated solution follows.

order	$\mathbf{t}$	graph	$\gamma(\mathbf{t})$	$\alpha(\mathbf{t})$	$F^J(\mathbf{t})(y)$	$\Phi_j(\mathbf{t})$
0	$\emptyset$	$\emptyset$	1	1	y^J	
1	τ		1	1	f^J	1
2	$\mathbf{t}_{21}$		2	1	$\sum_K f^J_K f^K$	$\sum_k a_{jk}$
3	$\mathbf{t}_{31}$		3	1	$\sum_{K,L} f^J_{KL} f^K f^L$	$\sum_{k,l} a_{jk}a_{jl}$
3	$\mathbf{t}_{32}$		6	1	$\sum_{K,L} f^J_K f^K_L f^L$	$\sum_{k,l} a_{jk}a_{kl}$
4	$\mathbf{t}_{41}$		4	1	$\sum_{K,L,M} f^J_{KLM} f^K f^L f^M$	$\sum_{k,l,m} a_{jk}a_{jl}a_{jm}$
4	$\mathbf{t}_{42}$		8	3	$\sum_{K,L,M} f^J_{KM} f^K_L f^L f^M$	$\sum_{k,l,m} a_{jk}a_{kl}a_{jm}$
4	$\mathbf{t}_{43}$		12	1	$\sum_{K,L,M} f^J_K f^K_{LM} f^L f^M$	$\sum_{k,l,m} a_{jk}a_{kl}a_{km}$
4	$\mathbf{t}_{44}$		24	1	$\sum_{K,L,M} f^J_K f^K_L f^L_M f^M$	$\sum_{k,l,m} a_{jk}a_{kl}a_{lm}$
5	$\mathbf{t}_{51}$		5	1	$\sum f^J_{KLMP} f^K f^L f^M f^P$	$\sum_{k,l,m,p} a_{jk}a_{jl}a_{jm}a_{jp}$
5	$\mathbf{t}_{52}$		10	6	$\sum f^J_{KMP} f^K_L f^L f^M f^P$	$\sum_{k,l,m,p} a_{jk}a_{kl}a_{jm}a_{jp}$
5	$\mathbf{t}_{53}$		15	4	$\sum f^J_{KMP} f^K_L f^L f^M f^P$	$\sum_{k,l,m,p} a_{jk}a_{kl}a_{km}a_{jp}$
5	$\mathbf{t}_{54}$		30	4	$\sum f^J_{KP} f^K_L f^L_M f^M f^P$	$\sum_{k,l,m,p} a_{jk}a_{kl}a_{lm}a_{jp}$
5	$\mathbf{t}_{55}$		20	3	$\sum f^J_{KM} f^K_L f^L f^M_P f^P$	$\sum_{k,l,m,p} a_{jk}a_{kl}a_{jm}a_{mp}$
5	$\mathbf{t}_{56}$		20	1	$\sum f^J_K f^K_{LMP} f^L f^M f^P$	$\sum_{k,l,m,p} a_{jk}a_{kl}a_{km}a_{kp}$
5	$\mathbf{t}_{57}$		40	3	$\sum f^J_K f^K_{LP} f^L_M f^M f^P$	$\sum_{k,l,m,p} a_{jk}a_{kl}a_{lm}a_{kp}$
5	$\mathbf{t}_{58}$		60	1	$\sum f^J_K f^K_L f^L_{MP} f^M f^P$	$\sum_{k,l,m,p} a_{jk}a_{kl}a_{lm}a_{lp}$
5	$\mathbf{t}_{59}$		120	1	$\sum f^J_K f^K_L f^L_M f^M_P f^P$	$\sum_{k,l,m,p} a_{jk}a_{kl}a_{lm}a_{mp}$

Table 2.2: Trees up to order 5

Theorem 2.3. *The derivatives of g_i satisfy*

$$g_i^{(q)}|_{h=0} = \sum_{\mathbf{t}\in LT_q} \gamma(\mathbf{t}) \sum_j a_{ij}\Phi_j(\mathbf{t})F(\mathbf{t})(y_n).$$

The numerical solution y_1 of (1.1) satisfies

$$(y_{n+1})^{(q)}|_{h=0} = \sum_{\mathbf{t}\in LT_q} \gamma(\mathbf{t}) \sum_j b_j\Phi_j(\mathbf{t})F(\mathbf{t})(y_n) = \sum_{\mathbf{t}\in T_q} \alpha(\mathbf{t})\gamma(\mathbf{t}) \sum_j b_j\Phi_j(\mathbf{t})F(\mathbf{t})(y_n). \tag{2.34}$$

Proof. See [21]. □

Finally we state the result about order conditions. If we compare expressions (2.33) and (2.34) in Theorems 2.2 and 2.3 we can conclude the following theorem about the order of RK method.

Theorem 2.4. *A Runge-Kuta method (1.2) is of order p iff*

$$\sum_{j=1}^{s} b_j \Phi_j(\mathbf{t}) = \frac{1}{\gamma(\mathbf{t})} \tag{2.35}$$

for all trees of order $\leq p$.

Proof. See [21]. □

We see that for order 1 method we need from theorem above just one condition. For order 2 method 2 conditions - the equation (2.35) has to hold for tree of order 1 τ and tree of order 2 $\mathbf{t}_{21}$. In case of third order method we have two conditions mentioned and another two corresponding to two trees of order 3. The number of order conditions increase very quickly as can be seen in Table 2.3, which gives number of order conditions up to order 10.

order p	1	2	3	4	5	6	7	8	9	10
number of conditions	1	2	4	8	17	37	85	200	486	1205

Table 2.3: Number of order conditions

2.4 Methods of Order 4

In this section we want to construct fourth order explicit method with four stages, $s = p = 4$. Practically it means, that we need determine the coefficients to satisfy the order conditions. According to Theorem 3.1, equations $\sum_{j=1}^{4} b_j \Phi_j(\mathbf{t}) = \frac{1}{\gamma(\mathbf{t})}$ has hold for all trees $\tau, \mathbf{t}_{21}, \mathbf{t}_{31}, \mathbf{t}_{32}, \mathbf{t}_{41}, \mathbf{t}_{42}$, $\mathbf{t}_{43}, \mathbf{t}_{44}$. Since the method will be an explicit method, we also use condition $a_{ij} = 0$ for $i < j$, and take into account condition (1.3)

$$c_i = \sum_{j=1}^{i-1} a_{ij}, \quad i = 1, 2, ...s.$$

From this condition follows $c_1 = 0$. Elementary differentials and values $\gamma(\mathbf{t})$ corresponding to above mentioned trees can be found in Table 2.2. With all aspects mentioned above we get eight

conditions:

$$\sum_{i=1}^{4} b_i = b_1 + b_2 + b_3 + b_4 = 1 \tag{2.36a}$$

$$\sum_{i=1}^{4} b_i c_i = b_2 c_2 + b_3 c_3 + b_4 c_4 = \frac{1}{2} \tag{2.36b}$$

$$\sum_{i=1}^{4} b_i c_i^2 = b_2 c_2^2 + b_3 c_3^2 + b_4 c_4^2 = \frac{1}{3} \tag{2.36c}$$

$$\sum_{i=1}^{4} \sum_{j=1}^{i-1} b_i a_{ij} c_j = b_3 a_{32} c_2 + b_4 (a_{42} c_2 + a_{43} c_3) = \frac{1}{6} \tag{2.36d}$$

$$\sum_{i=1}^{4} b_i c_i^3 = b_2 c_2^3 + b_3 c_3^3 + b_4 c_4^3 = \frac{1}{4} \tag{2.36e}$$

$$\sum_{i=1}^{4} \sum_{j=1}^{i-1} b_i c_i a_{ij} c_j = b_3 c_3 a_{32} c_2 + b_4 c_4 (a_{42} c_2 + a_{43} c_3) = \frac{1}{8} \tag{2.36f}$$

$$\sum_{i=1}^{4} \sum_{j=1}^{i-1} b_i a_{ij} c_j^2 = b_3 a_{32} c_2^2 + b_4 (a_{42} c_2^2 + a_{43} c_3^2) = \frac{1}{12} \tag{2.36g}$$

$$\sum_{i=1}^{4} \sum_{j=1}^{i-1} \sum_{k=1}^{j-1} b_i a_{ij} a_{jk} c_k = b_4 a_{43} a_{32} c_2 = \frac{1}{24}. \tag{2.36h}$$

Solving this system of equitations is almost exhausting. It is possible to make this system easier to solve with results of J.C. Butcher which are called *simplifying assumptions* [8]. Following theorem is actually simplifying assumption $D(1)$.

Theorem 2.5. *If*

$$\sum_{i=j+1}^{s} b_i a_{ij} = b_j (1 - c_j), \quad j = 1, 2, ...s, \tag{2.37}$$

then the equations (2.36d), (2.36g) and (2.36h) follow from the others.

Proof. For (2.36g) we get

$$\sum_{i=1}^{s} \sum_{j=1}^{s} b_i a_{ij} c_j^2 = \sum_{j=1}^{s} b_j c_j^2 - \sum_{j=1}^{s} b_j c_j^3 = \frac{1}{3} - \frac{1}{4} = \frac{1}{12}$$

by (2.36c) and (2.36e).

Similarly for (2.36d) we obtain

$$\sum_{i=1}^{s} \sum_{j=1}^{s} b_i a_{ij} c_j = \sum_{j=1}^{s} b_j c_j - \sum_{j=1}^{s} b_j c_j^2 = \frac{1}{2} - \frac{1}{3} = \frac{1}{6}$$

by (2.36b) and (2.36c).

And in the end for (2.36h)

$$\sum_{i=1}^{s} \sum_{j=1}^{s} b_i a_{ij} a_{jk} c_j = \sum_{j=1}^{s} \sum_{k=1}^{s} b_j a_{jk} c_k - \sum_{j=1}^{s} \sum_{k=1}^{s} b_j c_j a_{jk} c_k = \frac{1}{6} - \frac{1}{8} = \frac{1}{24}$$

holds by (2.36d) and (2.36f). □

Theorem 2.6. *For $s = 4$, equations (2.36) and (1.3) imply (2.37)*

To prove Theorem 2.6 we need following lemma:

Lemma 2.7. *If P and Q are each 3×3 matrices such that their product has the form*

$$PQ = \begin{pmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \textit{where} \quad \det \begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{pmatrix} = \det(R) \neq 0,$$

then either the last row of P is zero or the last column of Q is zero.

Proof. See [21]. □

Proof of Theorem 2.6. Define

$$d_j = \sum_{i=1}^{4} b_i a_{ij} - b_j(1 - c_j), \quad \text{for} \quad j = 1, 2, 3, 4,$$

so we have to prove $d_j = 0$. Now we apply Lemma 2.7 with

$$P = \begin{pmatrix} b_2 & b_3 & b_4 \\ b_2 c_2 & b_3 c_3 & b_4 c_4 \\ \sum_{i=1}^{4} b_i a_{i2} - b_2(1 - c_2) & \sum_{i=1}^{4} b_i a_{i3} - b_2(1 - c_3) & \sum_{i=1}^{4} b_i a_{i4} - b_4(1 - c_4) \end{pmatrix}$$

and

$$Q = \begin{pmatrix} c_2 & c_2^2 & \sum_{j=1}^{4} a_{2j} c_j - \frac{1}{2} c_2^2 \\ c_3 & c_3^2 & \sum_{j=1}^{4} a_{3j} c_j - \frac{1}{2} c_3^2 \\ c_4 & c_4^2 & \sum_{j=1}^{4} a_{4j} c_j - \frac{1}{2} c_4^2 \end{pmatrix}$$

Multiplication of these two matrices, using the conditions of (2.36), gives

$$PQ = \begin{pmatrix} \frac{1}{2} & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{1}{4} & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

where $\det(R) = \frac{1}{8} - \frac{1}{9} \neq 0$. Thus the last row of P or the last column of Q is zero. The first term in last column of Q is

$$\sum_{j=1}^{4} a_{2j} c_j - \frac{1}{2} c_2^2 = -\frac{1}{2} c_2^2,$$

cannot be equal to zero since c_2 has to satisfy order condition (2.36h). It implies that the last row of P is equal to 0, which actually are d_2, d_3 and d_4. From (2.36a), (2.36b) we obtain $d_1 + d_2 + d_3 + d_4 = 0$. Thus also $d_1 = 0$. □

From Theorems 2.5 and 2.6 follows:

Theorem 2.8. *Under the assumption* $c_i = \sum_{j=1}^{4} a_{ij}$, $i = 1, 2, ...4$ *the equations (2.36) are equivalent to equations*

$$b_1 + b_2 + b_3 + b_4 = 1 \tag{2.38a}$$
$$b_2c_2 + b_3c_3 + b_4c_4 = \frac{1}{2} \tag{2.38b}$$
$$b_2c_2^2 + b_3c_3^2 + b_4c_4^2 = \frac{1}{3} \tag{2.38c}$$
$$b_2c_2^3 + b_3c_3^3 + b_4c_4^3 = \frac{1}{4} \tag{2.38d}$$
$$b_3c_3a_{32}c_2 + b_4c_4(a_{42}c_2 + a_{43}c_3) = \frac{1}{8} \tag{2.38e}$$
$$b_3a_{32} + b_4a_{42} = b_2(1 - c_2) \tag{2.38f}$$
$$b_3a_{43} = b_3(1 - c_3) \tag{2.38g}$$
$$0 = b_4(1 - c_4) \tag{2.38h}$$

From the last equation we have $c_4 = 1$, since $b_4 \neq 0$ from (2.36h). Also b_3 cannot be zero by (2.36h) and (2.38g). Now the sketch of solving the system (2.38) in Theorem 2.8 is presented. First we choose c_2, c_3. Then we compute b_1, b_2, b_3, b_4 to satisfy (2.38a),(2.38b),(2.38c) and (2.38d). Coefficient a_{43} is possible to compute from (2.38g). Equations (2.38e) and (2.38f) form a linear system of two equations for a_{32} and a_{42}. This system is regular, so it has a solution. Missing coefficients a_{21}, a_{31}, a_{41} can be obtained from (1.3).

In case that c_i, $i = 2, 3$ are not chosen distinct, there is possibility of no solution for b_i, $i = 1, 2, ..., 4$ and it might also happen that b_3 or b_4 can be found zero (which we showed is not possible). W. Kutta in [24] distinguished five cases where a solution exists. Those cases with their Butcher tableaus can be found in [7].

In conclusion of this section some fourth order four stages method are mentioned. In [24] Kutta also presented his two famous fourth order methods. First is classical Runge–Kutta method with Butcher tableau:

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array} .$$

The second method is also well-known 3/8 rule:

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{3} & \frac{1}{3} & & & \\ \frac{2}{3} & -\frac{1}{3} & 1 & & \\ 1 & 1 & -1 & 1 & \\ \hline & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \end{array} .$$

Another interesting fourth order RK method was derived by Gill. This is methods is special for purpose of reducing memory requirements for large problems and its Butcher tableau is

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

2.5 Order Barriers

In Table 2.3 we saw that number of order condition significantly grows with higher order. In previous section methods with $s = p$, $s = 1, 2, 3, 4$ were presented. In case of 5-stages method it is needed to hold 17 order conditions. This is not possible with just 5 stages. Those results about highest possible order of s-stage methods were presented by J.C.Butcher, thus they are also called Butcher barriers.

Theorem 2.9. *For $p \geq 5$ no explicit RK method exists of order p with $s = p$ stages.*

Proof. See [21]. □

If desired order is higher and higher with number of order conditions to satisfy, number of stages also grows.

Theorem 2.10. *For $p \geq 7$ no explicit RK method exists of order p with $s = p + 1$ stages.*

Proof. See [4]. □

Theorem 2.11. *For $p \geq 8$ no explicit RK method exists of order p with $s = p + 2$ stages.*

Proof. See [6]. □

Explicit RK method with highest order, tenth order, was constructed by E.Hairer in [20] and it has 17 stages. But it is unknown if it is possible to have even less stages.

2.6 Modified Runge–Kutta Methods

During last few decades RK methods based on various kinds of means, as geometrical, harmonic, contra-harmonic, heroian or centroidal appeared. Among the first work was thesis of B. B. Sanugi [31] who constructed several methods based on geometrical mean with one up to four stages. Or A. M. Wazwaz who combined arithmetic and geometric mean, which lead to method based on harmonic mean. He also did a comparison of modified methods with three stages [36]. Also extensive work of D. J. Evans should be mentioned [13–15].

We will present seven methods with four stages based on various kinds of means. For two of them we will make the theoretical examination of order and for all of the methods we will test the numerical order.

As the first modified method we start with the one based on arithmetic mean (RKAM). Those methods have output in form

$$y_{n+1} = y_n + \frac{h}{s-1} \sum_{i=1}^{s-1} \frac{k_i + k_{i+1}}{2}.$$

This is not so far from the usual explicit RK methods. In the case of four stage method, we actually have the classical RK method.

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

with the output rewritten as

$$y_{n+1} = y_n + \frac{h}{3}\left(\frac{k_1+k_2}{2} + \frac{k_2+k_3}{2} + \frac{k_3+k_4}{2}\right).$$

In case of modified RK methods we will use incomplete Butcher tableau and output of method to avoid confusion with traditional RK methods.

As the second we present method based on geometrical mean (RKGM). It was derived by B. Sanugi in his thesis [31]:

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ \frac{1}{2} & -\frac{1}{16} & \frac{9}{16} & \\ 1 & -\frac{1}{8} & \frac{5}{24} & \frac{11}{12} \\ \hline \end{array}.$$

$$y_n + \frac{h}{3}\left(\sqrt{|k_1k_2|} + \sqrt{|k_2k_3|} + \sqrt{|k_3k_4|}\right). \tag{2.39}$$

But (2.39) is not able to approximate the decreasing solution. Therefore we introduce factor δ such as $\delta_i = -1$ if one of k_i, k_{i+1} is negative. Then the output is given by

$$y_n + \frac{h}{3}\left(\delta_1\sqrt{|k_1k_2|} + \delta_2\sqrt{|k_2k_3|} + \delta_3\sqrt{|k_3k_4|}\right).$$

If we take square of geometrical mean and dived it by arithmetical mean we obtain harmonical mean (RKHaM). Method based on this mean was presented by A. M. Wazwaz in [35] and has Butcher tableau:

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ \frac{1}{2} & -\frac{1}{8} & \frac{5}{8} & \\ 1 & -\frac{1}{4} & \frac{7}{20} & \frac{9}{10} \\ \hline \end{array}$$

$$y_n + \frac{2h}{3}\left(\frac{k_1k_2}{k_1+k_2} + \frac{k_2k_3}{k_2+k_3} + \frac{k_3k_4}{k_3+k_4}\right) \tag{2.40}$$

The method based on heronian mean (RKHeM) was published by D. J. Evans and N. Yaacob [13] and has the form

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ \frac{1}{2} & -\frac{1}{48} & \frac{25}{48} & \\ 1 & -\frac{1}{24} & \frac{47}{600} & \frac{289}{300} \\ \hline \end{array}.$$

$$y_n + \frac{h}{9}\left(k_1 + 2\left(k_2 + k_3\right) + k_4 + \sqrt{|k_1 k_2|} + \sqrt{|k_2 k_3|} + \sqrt{|k_3 k_4|}\right).$$

As in the case of RKGM method, for practical implementation we involve factor δ:

$$y_n + \frac{h}{9}\left(k_1 + 2\left(k_2 + k_3\right) + k_4 + \delta_1\sqrt{|k_1 k_2|} + \delta_2\sqrt{|k_2 k_3|} + \delta_3\sqrt{|k_3 k_4|}\right).$$

The following three methods are results of two researchers D. J. Evans and A. R. Yaakub. The first method of their work is based on root mean square (RKRM):

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ \frac{1}{2} & \frac{1}{16} & \frac{7}{16} & \\ 1 & \frac{1}{8} & -\frac{17}{56} & \frac{33}{28} \\ \hline \end{array}.$$

$$y_n + \frac{h}{3}\left(\sqrt{\frac{k_1^2 + k_2^2}{2}} + \sqrt{\frac{k_2^2 + k_3^2}{2}} + \sqrt{\frac{k_3^2 + k_4^2}{2}}\right)$$

And again for good approximating of decreasing solution we will need the factor δ to multiply the square roots.

$$y_n + \frac{h}{3}\left(\delta_1\sqrt{\frac{k_1^2 + k_2^2}{2}} + \delta_2\sqrt{\frac{k_2^2 + k_3^2}{2}} + \delta_3\sqrt{\frac{k_3^2 + k_4^2}{2}}\right)$$

The second method is based on contraharmonic mean (RKCoM):

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ \frac{1}{2} & \frac{1}{8} & \frac{3}{8} & \\ 1 & \frac{1}{4} & -\frac{3}{4} & \frac{3}{2} \\ \hline \end{array}.$$

$$y_n + \frac{h}{3}\left(\frac{k_1^2 + k_2^2}{k_1 + k_2} + \frac{k_2^2 + k_3^2}{k_2 + k_3} + \frac{k_3^2 + k_4^2}{k_3 + k_4}\right) \tag{2.41}$$

The last method presented in this work is based on centroidal mean (CeM)

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ \frac{1}{2} & \frac{1}{24} & \frac{11}{24} & \\ 1 & \frac{1}{12} & -\frac{25}{132} & \frac{73}{66} \\ \hline \end{array}.$$

$$y_{n+1} = y_n + \frac{2h}{9}\left(\frac{k_1^2 + k_1 k_2 + k_2^2}{k_1 + k_2} + \frac{k_2^2 + k_2 k_3 + k_3^2}{k_2 + k_3} + \frac{k_3^2 + k_3 k_4 + k_4^2}{k_3 + k_4}\right).$$

All above method are referred in [29] as fourth order method. It will be check for two of them in the following. Unfortunately, we cannot use all results of section 2.3 because of the much more complicated output. But a few results from section 2.2 will be useful.

The RKAM method is in fact the classical Runge-Kutta method of order 4. Its derivation was sketched in section 2.4 therefore here will not be done any further theoretical examination.

First the examination for autonomous scalar equation

$$y' = f(y) \tag{2.42}$$

will be done and then the same for non-autonomous scalar equation (1.1). For the systems only system of autonomous equations will be examined. Since the modified methods satisfy the condition (1.3) we can make the problem autonomous without loss of generality as we have seen in (2.24).

In case of autonomous scalar problem the Taylor expansion of exact solution around the point t_n is

$$y(t_n + h) = y_n + hf(y_n) + \frac{h^2}{2}f'(y_n) + \frac{h^3}{6}f''(y_n) + \frac{h^4}{24}f'''(y_n) + \mathcal{O}(h^5), \tag{2.43}$$

where the derivatives of function f are

$$\begin{aligned} f'(y_n) &= f_y(y_n)f(y_n) \\ f''(y_n) &= f_{yy}(y_n)f^2(y_n) + f_y^2(y_n)f(y_n) \\ f'''(y_n) &= f_{yyy}(y_n)f^3(y_n) + 4f_{yy}(y_n)f_y(y_n)f^2(y_n) + f_y^3(y_n)f(y_n). \end{aligned}$$

If RKCoM method is applied to autonomous scalar problem (2.42), we get

$$\begin{aligned} k_1 &= f(y_n) \\ k_2 &= f(y_n + \frac{1}{2}hk_1) \\ k_3 &= f(y_n + \frac{1}{8}hk_1 + \frac{3}{8}hk_2) \\ k_4 &= f(y_n + \frac{1}{4}hk_1 - \frac{3}{4}hk_2 + \frac{3}{2}hk_3) \\ y_{n+1} &= y_n + \frac{h}{3}\left(\frac{k_1^2 + k_2^2}{k_1 + k_2} + \frac{k_2^2 + k_3^2}{k_2 + k_3} + \frac{k_3^2 + k_4^2}{k_3 + k_4}\right). \end{aligned} \tag{2.44}$$

To obtain Taylor expansion of (2.44) it is necessary to make directly the derivatives of output which needs also derivatives of stages. Theoretically clear procedure requires a lot of computation in case of derivatives of output. Fortunately there are software programs which can make the job.

To get the following Taylor expansion software Maple was involved. By using command `taylor(`y_{n+1}`,h=0,6)` was obtained result

$$\begin{aligned} y(t_n + h) =& y_n + hf(y_n) + \frac{h^2}{2}f_y(y_n)f(y_n) + \frac{h^3}{6}\left(f_{yy}(y_n)f^2(y_n) + f_y^2(y_n)f(y_n)\right) && (2.45) \\ &+ \frac{h^4}{24}\left(f_{yyy}(y_n)f^3(y_n) + 4f_{yy}(y_n)f_y(y_n)f^2(y_n) + f_y^3(y_n)f(y_n)\right) && (2.46) \\ &+ \frac{h^5}{41472}\left(296f_y^4(y_n)f(y_n) + 47347f_y^2(y_n)f^2(y_n)f_{yy}(y_n) + ...\right) + \mathcal{O}(h^6) && (2.47) \end{aligned}$$

thus if we compare (2.43) and (2.45) we can see that the RKCoM method is of order 4 for autonomous scalar problem.

Now we apply RKCoM method on (1.1), so we have

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1) \\ k_3 &= f(t_n + \frac{1}{2}hy_n + \frac{1}{8}hk_1 + \frac{3}{8}hk_2) \\ k_4 &= f(t_n + h, y_n + \frac{1}{4}hk_1 - \frac{3}{4}hk_2 + \frac{3}{2}hk_3) \\ y_{n+1} &= y_n + \frac{h}{3}\left(\frac{k_1^2 + k_2^2}{k_1 + k_2} + \frac{k_2^2 + k_3^2}{k_2 + k_3} + \frac{k_3^2 + k_4^2}{k_3 + k_4}\right). \end{aligned} \tag{2.48}$$

The Taylor expansion of exact solution for scalar problem around point t_n is

$$\begin{aligned} y(t_n + h) = y(t_n) + hf(t_n, y_n) + \frac{h^2}{2}F(t_n, y_n) + \frac{h^3}{6}\left(G(t_n, y_n) + f_y(t_n, y_n)F(t_n, y_n)\right) \\ + \frac{h^4}{24}\left(f_{ttt}(t_n, y_n) + 3f_{tty}(t_n, y_n)f(t_n, y_n) + 3f_{tyy}(t_n, y_n)f^2(t_n, y_n) + f_{yyy}(t_n, y_n)f^3(t_n, y_n)\right. \\ \left. + 3(f_{ty}(t_n, y_n) + f_{yy}(t_n, y_n))F(t_n, y_n) + f_y(t_n, y_n)(G(t_n, y_n) + f_y(t_n, y_n)F(t_n, y_n))\right) + \mathcal{O}(h^5), \end{aligned} \tag{2.49}$$

where G,F are defined same as in Section 2.2, i.e.

$$F = f_t + f_y f \text{ and } G = f_{tt} + 2f_{ty}f + f_{yy}\left(f\right)^2.$$

From Maple we obtain Taylor expansion of (2.48)

$$\begin{aligned} y(t_{n+h}) =& y_n + hf(t_n, y_n) + \frac{h^2}{2}\left(f_t(t_n, y_n) + f_y(t_n, y_n)\right) \\ &+ \frac{h^3}{6}\left(f_{tt}(t_n, y_n) + 2f_{ty}(t_n, y_n)f(t_n, y_n) + f_{yy}(t_n, y_n)f^2(t_n, y_n) + \frac{5}{4}f_t(t_n, y_n)f_y(t_n, y_n)...\right. \\ &\left. + f_y^2(t_n, y_n)f(t_n, y_n) + \frac{1}{4}\frac{f_t^2(t_n, y_n)}{f(t_n, y_n)}\right) + \mathcal{O}(h^4). \end{aligned} \tag{2.50}$$

By comparing (2.49) and (2.50) we see that the method is only second order for non-autonomous scalar problem.

The second method which we are going to check for the order is RKCeM method. We apply it on the autonomous scalar problem

$$\begin{aligned} k_1 &= f(y_n) \\ k_2 &= f(y_n + \frac{1}{2}hk_1) \\ k_3 &= f(y_n + \frac{1}{24}hk_1 + \frac{11}{24}hk_2) \\ k_4 &= f(y_n + \frac{1}{12}hk_1 - \frac{25}{132}hk_2 + \frac{73}{66}hk_3) \\ y_{n+1} &= y_n + \frac{2h}{9}\left(\frac{k_1^2 + k_1k_2 + k_2^2}{k_1 + k_2} + \frac{k_2^2 + k_2k_3 + k_3^2}{k_2 + k_3} + \frac{k_3^2 + k_3k_4 + k_4^2}{k_3 + k_4}\right) \end{aligned} \tag{2.51}$$

and by Maple we obtain the following Taylor expansion

$$\begin{aligned} y(t_n+h) =& y_n + hf(y_n) + \frac{h^2}{2} f_y(y_n)f(y_n) + \frac{h^3}{6}\left(f_{yy}(y_n)f^2(y_n) + f_y^2(y_n)f(y_n)\right) \\ &+ \frac{h^4}{24}\left(f_{yyy}(y_n)f^3(y_n) + 4f_{yy}(y_n)f_y(y_n)f^2(y_n) + f_y^3(y_n)f(y_n)\right) \\ &+ h^5\left(\frac{5}{576}f_{yyyy}(y_n)f^4(y_n) + \frac{203}{3456}f_y(y_n)f^3(y_n)f_{yyy}(y_n) + ...\right) + \mathcal{O}(h^6). \end{aligned}$$

This method is also of fourth order for autonomous scalar problem.

But for the non-autonomous scalar problem Taylor expansion is the following:

$$\begin{aligned} y(t_n+h) =& y_n + hf(t_n,y_n) + \frac{h^2}{2}\left(f_t(t_n,y_n) + f_y(t_n,y_n)\right) \\ &+ \frac{h^3}{6}\left(f_{tt}(t_n,y_n) + 2f_{ty}(t_n,y_n)y_n)f(t_n,y_n) + f_{yy}(t_n,y_n)f^2(t_n,y_n)...\right. \\ &\left.+ \frac{13}{12}f_t(t_n,y_n)f_y(t_n,y_n) + f_y^2(t_n,y_n)f(t_n,y_n) + \frac{1}{12}\frac{f_t^2(t_n,y_n)}{f(t_n,y_n)}\right) + \mathcal{O}(h^4). \end{aligned}$$

So this method is also only second order for non-autonomous equation.

If we make the system (1.1) autonomous, then the result should be the same as in case of non-autonomous scalar problem because every non-autonomous problem can be written as system of two autonomous equation. To confirm it, we will just briefly show it for RKCoM method.

From (2.21) and (2.22) we can write down Taylor expansion of exact solution for vector problem

$$\begin{aligned} y^J(t+h) = y_n^J + h\ f^J\big|_{y=y_n} + \frac{h^2}{2}\sum_{K=1}^{m} f_K^J f^K\Big|_{y=y_n} \\ + \frac{h^3}{3}\left(\sum_{K=1}^{m}\sum_{L=1}^{m} f_{KL}^J f^K f^L\Big|_{y=y_n} + \sum_{K=1}^{m}\sum_{L=1}^{m} f_K^J f_L^K f^L\Big|_{y=y_n}\right) + \mathcal{O}(h^4). \end{aligned} \tag{2.52}$$

Therefore, RKCeM method is also only second order for non-autonomous equation.

Similarly as in Section 2.2, we rewrite the (1.1) into form

$$y_{n+1}^J = y_n^J + \frac{1}{3}h\Phi^J(h)$$

then its Taylor expansion can be written as

$$y_{n+1}^J = y_n^J + h\left(\frac{1}{3}\Phi^J(0)\right) + \frac{h^2}{2}\left(\frac{2}{3}\left(\Phi^J\right)'(0)\right) + \frac{h^3}{6}\left(\left(\Phi^J\right)''(0)\right) + \mathcal{O}(h^4) \tag{2.53}$$

Then we need to compute the derivatives of function Φ. This has been done again with the help

of Maple software.

$$\Phi(0) = \sum_{i=1}^{3} \frac{k_i^2 + k_{i+1}^2}{k_i + k_{i+1}}\bigg|_{y=y_n}$$

$$\Phi'(0) = \sum_{i=1}^{3} \frac{(2k_ik_i' + 2k_{i+1}k_{i+1}')(k_i + k_{i+1}) - (k_i^2 + k_{i+1}^2)(k_i' + k_{i+1}')}{(k_i + k_{i+1})^2}\bigg|_{y=y_n}$$

$$\Phi''(0) = \sum_{i=1}^{3} \left(\frac{(2(k_i')^2 + 2k_ik_i'' + 2(k_{i+1}')^2 + 2k_{i+1}k_{i+1}'')}{(k_i + k_{i+1})}\bigg|_{y=y_n} - \frac{2(2k_ik_i' + 2k_{i+1}k_{i+1}')(k_i' + k_{i+1}')}{(k_i + k_{i+1})^2}\bigg|_{y=y_n} \right.$$
$$\left. + \frac{2(k_i^2 + k_{i+1}^2)(k_i' + k_{i+1}')^2}{(k_i + k_{i+1})^3}\bigg|_{y=y_n} - \frac{(k_i^2 + k_{i+1}^2)(k_i'' + k_{i+1}'')}{(k_i + k_{i+1})^2}\bigg|_{y=y_n} \right)$$

With the help of relations (2.28), (2.26), (2.30) and (2.27) we obtain derivatives of stages k_i, $i = 1, 2, 3, 4$. The Taylor expansion of solution obtained by RKCoM method become:

$$y(t_n + h)^J = y_n^J + hf^J + \frac{h^2}{2} \sum_{K=1}^{m} f_K^J f^K$$
$$+ \frac{h^3}{6} \left(\sum_{K=1}^{m} \sum_{L=1}^{m} f_{KL}^J f^K f^L + \frac{3}{4} \sum_{K=1}^{m} \sum_{L=1}^{m} f_K^J f_L^K f^L + \sum_{K=1}^{m} \frac{(f_K^J)^2 (f^K)^2}{4f^J} \right) + \mathcal{O}(h^4).$$

Comparing with the Taylor expansion of exact solution (2.52) we see that this method is only of second order for vector problem, as we expected.

2.7 Numerical Testing of Modified Runge–Kutta Methods

Now we move to testing the modified RK methods numerically. The methods will be compared according the absolute error. For this purpose the solution was computed with constant step size $h = \frac{t_f - t_0}{100}$.

Then, we will also numerically determine the order of all methods from previous section. The global error e satisfy following:

$$|e| = Ch^p$$
$$|e_{\frac{1}{2}}| = Ch_{\frac{1}{2}}^p$$

where we denote by $h_{\frac{1}{2}} = \frac{1}{2}h$. Then if we make logarithm and subtract those two equations. Hence we get

$$\ln\left(\frac{|e|}{|e_{\frac{1}{2}}|}\right) = \ln\left(\frac{h}{h_{\frac{1}{2}}}\right)^p = p\ln(2).$$

From above we have relation for the numerical order of method

$$p = \frac{\ln\left(\frac{|e|}{|e_{\frac{1}{2}}|}\right)}{\ln(2)}. \tag{2.54}$$

The solution was computed with step sizes $h_i = \frac{1}{2^i}$, $\quad i = 5, ...9$. Since we know the exact solution we can compute the error and plug it into (2.54). The sequence of $\{p_i\}$ should converge to the order of method.

All modified methods were implemented in MATLAB (ComparisonModifiedMethod.m).

Problem 1

The first problem we have chosen autonomous scalar equation

$$\begin{aligned} y' &= -y, \qquad t \in [0,1] \\ y(0) &= 1 \end{aligned} \tag{2.55}$$

with exact solution

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

In Table 2.4 are the global errors of all modified RK methods at times $0, 0.1, ..., 1$.

Time	AM	GM	HaM	HeM	RM	CeM	CoM
0.0	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
0.1	7.603e-12	1.768e-11	2.910e-11	1.089e-11	2.179e-12	1.054e-12	1.511e-11
0.2	1.376e-11	3.200e-11	5.266e-11	1.970e-11	3.943e-12	1.908e-12	2.735e-11
0.3	1.868e-11	4.343e-11	7.147e-11	2.674e-11	5.352e-12	2.590e-12	3.712e-11
0.4	2.253e-11	5.239e-11	8.622e-11	3.226e-11	6.458e-12	3.124e-12	4.478e-11
0.5	2.548e-11	5.926e-11	9.752e-11	3.648e-11	7.304e-12	3.534e-12	5.065e-11
0.6	2.767e-11	6.434e-11	1.059e-10	3.961e-11	7.930e-12	3.837e-12	5.499e-11
0.7	2.921e-11	6.792e-11	1.118e-10	4.182e-11	8.372e-12	4.051e-12	5.805e-11
0.8	3.021e-11	7.024e-11	1.156e-10	4.324e-11	8.657e-12	4.189e-12	6.003e-11
0.9	3.075e-11	7.150e-11	1.177e-10	4.402e-11	8.812e-12	4.264e-12	6.111e-11
1.0	3.091e-11	7.188e-11	1.183e-10	4.426e-11	8.860e-12	4.287e-12	6.144e-11

Table 2.4: The global errors of the Problem 1 (2.55)

We assumed all modified methods to be fourth order for autonomous scalar problem. In Table 2.5 we see that it is true.

$h/h_{\frac{1}{2}}$	AM	GM	HaM	HeM	RM	CeM	CoM
$\frac{1}{32}/\frac{1}{64}$	4.0188	4.0256	4.0273	4.0225	4.0754	3.9346	4.0402
$\frac{1}{64}/\frac{1}{128}$	4.0094	4.0128	4.0137	4.0113	4.0380	3.9695	4.0200
$\frac{1}{128}/\frac{1}{256}$	4.0050	4.0062	4.0066	4.0064	4.0193	3.9839	4.0095
$\frac{1}{256}/\frac{1}{512}$	4.0073	3.9997	4.0046	4.0056	4.0353	3.8256	4.0007

Table 2.5: Numerical order for Problem 1 (2.55)

Problem 2

As the second problem we took equation with non-constant coefficients

$$\begin{aligned} y' &= -3t^2 y, \qquad t \in [0,1] \\ y(0) &= 1 \end{aligned} \tag{2.56}$$

with exact solution

$$y(t) = e^{-t^3}.$$

In the Table 2.6 the global errors are presented. The RKAM method performed much better result than the other methods. In case of order, we observe that all methods except RKAM have second order, see Table 2.7, which agree with results of previous section.

Time	AM	GM	HaM	HeM	RM	CeM	CoM
0.0	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
0.1	3.124e-13	2.497e-06	4.817e-06	8.323e-07	2.377e-06	1.606e-06	4.817e-06
0.2	6.206e-13	4.947e-06	9.717e-06	1.649e-06	4.827e-06	3.239e-06	9.718e-06
0.3	9.180e-13	7.233e-06	1.429e-05	2.411e-06	7.115e-06	4.765e-06	1.429e-05
0.4	1.162e-12	9.170e-06	1.817e-05	3.057e-06	9.057e-06	6.059e-06	1.818e-05
0.5	8.613e-13	1.054e-05	2.092e-05	3.514e-06	1.044e-05	6.977e-06	2.093e-05
0.6	2.992e-12	1.115e-05	2.215e-05	3.717e-06	1.105e-05	7.387e-06	2.217e-05
0.7	2.215e-11	1.088e-05	2.162e-05	3.626e-06	1.079e-05	7.211e-06	2.164e-05
0.8	8.938e-11	9.747e-06	1.938e-05	3.250e-06	9.679e-06	6.465e-06	1.940e-05
0.9	2.725e-10	7.948e-06	1.580e-05	2.650e-06	7.894e-06	5.272e-06	1.582e-05
1.0	6.752e-10	5.800e-06	1.153e-05	1.934e-06	5.757e-06	3.845e-06	1.154e-05

Table 2.6: The lobal errors of Problem 2 (2.56)

$h/h_{\frac{1}{2}}$	AM	GM	HaM	HeM	RM	CeM	CoM
$\frac{1}{32}/\frac{1}{64}$	3.9868	2.0227	2.0092	2.0254	2.0047	2.0084	2.0104
$\frac{1}{64}/\frac{1}{128}$	3.9955	2.0106	2.0038	2.0114	2.0019	2.0040	2.0047
$\frac{1}{128}/\frac{1}{256}$	3.9982	2.0051	2.0017	2.0054	2.0009	2.0020	2.0022
$\frac{1}{256}/\frac{1}{512}$	3.9991	2.0025	2.0008	2.0026	2.0004	2.0010	2.0011

Table 2.7: Numerical order for Problem 2 (2.56)

Problem 3

Our next problem is a linear system

$$y' = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} y, \qquad t \in [0, 10]$$
$$y_0 \begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{2.57}$$

with exact solution

$$y_1(t) = \sin(t) + \cos(t)$$
$$y_2(t) = \cos(t) - \sin(t).$$

In matter of global error we have the biggest error among our test problems, see Table 2.9. But really disturbing results brought the numerical examination of order. For y_2 (Table 2.8) we can clearly determine the order in case of RKAM (oder 4), RKGM and RKHeM (both order 2). For the rest of methods we can say that they have the second order of accuracy but not so sure. On the other hand the numbers in numerical order of y_1 are complete mess and we cannot say anything about order of methods (again except RKAM). It seems that contraction of time step size has no impact on the accuracy.

$h/h_{\frac{1}{2}}$	AM	GM	HaM	HeM	RM	CeM	CoM
	Numerical order for component y_1						
$\frac{1}{32}/\frac{1}{64}$	4.0803	1.7421	-0.0868	1.7177	-1.4668	-1.4897	-2.4551
$\frac{1}{64}/\frac{1}{128}$	4.0420	4.1585	9.1204	4.1775	0.8870	5.1823	4.6311
$\frac{1}{128}/\frac{1}{256}$	4.0205	4.7640	-0.7310	4.6609	2.6784	3.9335	4.9688
$\frac{1}{256}/\frac{1}{512}$	3.9948	-1.8733	0.7395	-1.7972	2.5224	0.0844	-0.3603
	Numerical order for component y_2						
$\frac{1}{32}/\frac{1}{64}$	3.9956	2.0225	3.6032	2.0254	1.8905	3.7117	3.8160
$\frac{1}{64}/\frac{1}{128}$	3.9979	1.8732	0.6144	1.8740	2.3206	0.4886	0.4051
$\frac{1}{128}/\frac{1}{256}$	3.9991	1.9607	1.9532	1.9612	1.9096	1.8654	1.8112
$\frac{1}{256}/\frac{1}{512}$	4.0013	2.0780	2.1419	2.0780	1.9569	2.1322	2.1275

Table 2.8: Numerical order of Problem 3 (2.57)

This is connected with another phenomena. Figure 2.3 presents the solution and the global error of method RKHaM. At time $t = 7.1$ we can see a jump of approximated solution of y_1. The same happened in case of RKCoM method, Figure 2.4, at time $t = 4$. Both jumps appeared near point where the solution changes from ingreasing do decreasing or vice versa. Thus the derivative of solution is close to zero and the stages also. If we look back to outputs of methods (2.40), (2.41), we find out that we dived $k_i k_{i+1}$ by the sum of the same stages or sum of square of k_i and k_{i+1} is again divided by its sum. Thus when the stages are really close to zero it

may happen that we divide by number which is almost zero. Even not to be so dramatic, we divide two close numbers which causes bigger numerical errors than in case of classical explicit RK methods where the stages are just sum. Therefore non-monotonous solution are difficult to approximate by modified method.

Time	AM	GM	HaM	HeM	RM	CeM	CoM
	Global error for component y_1						
0.0	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
1.0	1.542e-07	1.443e-03	2.532e-03	4.810e-04	1.725e-03	9.060e-04	2.807e-03
2.0	2.133e-06	5.598e-04	1.239e-03	1.879e-04	1.716e-03	3.866e-04	1.129e-03
3.0	2.994e-06	1.563e-03	3.662e-03	5.277e-04	1.881e-03	1.316e-03	4.092e-03
4.0	4.867e-08	4.245e-04	7.789e-03	1.457e-04	2.517e-03	1.123e-02	2.302e-02
5.0	4.925e-06	3.006e-03	1.329e-03	1.007e-03	1.318e-03	4.223e-03	1.794e-02
6.0	6.459e-06	2.888e-03	9.164e-03	1.065e-03	2.569e-03	6.742e-03	3.118e-03
7.0	1.248e-06	1.394e-04	8.564e-03	2.681e-04	1.630e-04	1.162e-02	2.062e-02
8.0	7.070e-06	3.624e-03	2.143e-02	1.088e-03	4.361e-03	4.049e-03	2.328e-02
9.0	1.020e-05	5.088e-03	2.563e-03	1.752e-03	4.459e-03	6.324e-03	4.410e-03
10.0	3.409e-06	1.504e-03	1.857e-02	6.527e-04	7.497e-04	1.123e-02	1.567e-02
	Global error for component y_2						
0.0	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
1.0	1.168e-06	8.849e-04	1.686e-03	2.956e-04	2.291e-04	5.722e-04	1.736e-03
2.0	1.003e-06	1.804e-03	3.262e-03	6.020e-04	1.216e-03	1.146e-03	3.525e-03
3.0	1.879e-06	8.808e-04	9.261e-04	2.877e-04	2.059e-04	1.629e-04	2.920e-04
4.0	4.713e-06	2.375e-03	4.741e-03	7.953e-04	2.275e-03	1.583e-03	4.772e-03
5.0	3.234e-06	7.919e-04	8.847e-03	2.639e-04	3.226e-03	1.022e-02	1.702e-02
6.0	2.876e-06	2.387e-03	2.934e-03	6.012e-04	2.168e-03	9.006e-03	2.396e-02
7.0	8.154e-06	4.246e-03	7.171e-03	1.397e-03	3.855e-03	1.158e-03	1.135e-02
8.0	6.235e-06	3.192e-03	1.224e-02	1.221e-03	1.621e-03	9.443e-03	1.267e-02
9.0	2.903e-06	1.815e-03	2.439e-02	4.627e-04	2.319e-03	9.051e-03	2.216e-02
10.0	1.128e-05	5.798e-03	1.642e-02	1.903e-03	5.535e-03	7.878e-04	1.673e-02

Table 2.9: The global errors for Problem 3 (2.57)

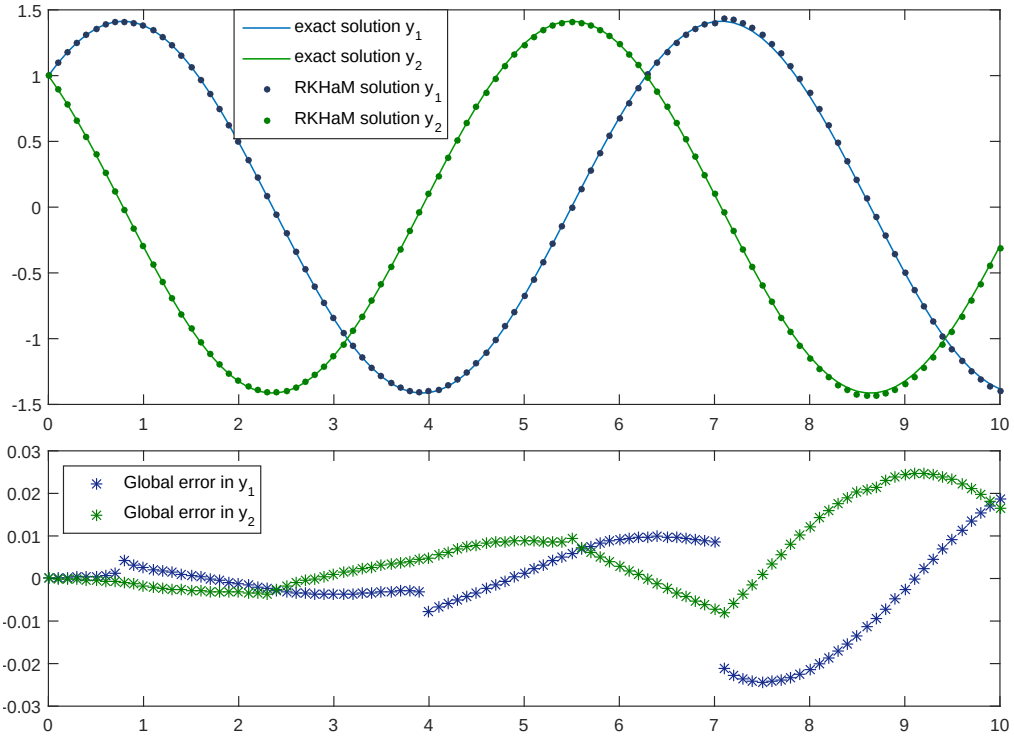


Figure 2.3: Approximated solution of Problem 3 (2.57) obtained by method RKHaM

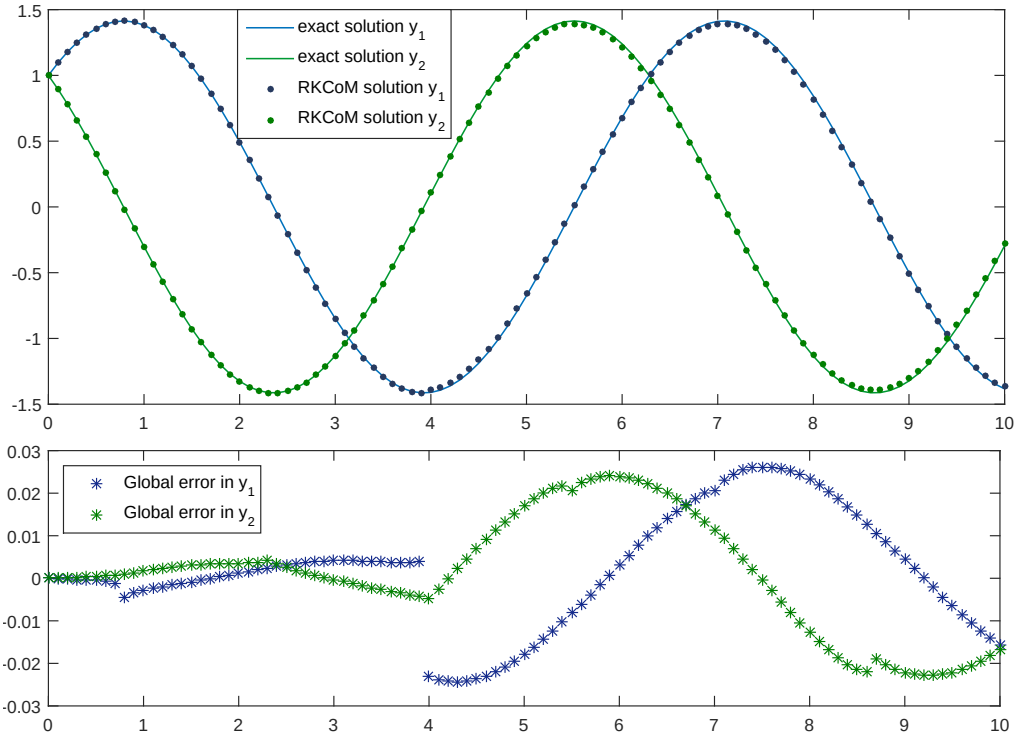


Figure 2.4: Approximated solution of Problem 3 (2.57) obtained by method RKCoM

Problem 4

As the last problem, in this section, the following inhomogeneous system was chosen

$$\begin{aligned} y_1' &= -4y_1 - 3y_2 - 14 \\ y_2' &= -2y_1 - 5y_2 + 5e^{-2t}, \quad t \in [0,1] \\ y(0) &= \begin{pmatrix} 0 \\ -2 \end{pmatrix} \end{aligned} \tag{2.58}$$

and with exact solution

$$\begin{aligned} y_1(t) &= 6e^{-2t} - e^{-7t} - 3te^{-2t} \\ y_2(t) &= -3e^{-2t} - e^{-7t} + 2te^{-2t} + 2. \end{aligned}$$

Except the RKAM, all methods reached the second order in both components, as can be seen in Table 2.10. The global errors are the lowest again for RKAM method. For the rest of the methods is similar to each other and comparable to those ones we obtained in Problem 2.

Modified methods are of fourth order accuracy only in case of autonomous equation. In any other case they are only second order. The group of autonomous scalar problem is not so wide thus we would not appreciate much the fourth order for autonomous equation in practise. Also, we can conclude that the performance of modified method heavily depends on the monotony of solution. This make them for many problem unsuitable and would be better to avoid them.

$h/h_{\frac{1}{2}}$	AM	GM	HaM	HeM	RM	CeM	CoM
	Numerical order for component y_1						
$\frac{1}{32}/\frac{1}{64}$	4.0373	2.0983	2.0911	2.1108	2.1031	2.0986	2.1159
$\frac{1}{64}/\frac{1}{128}$	4.0193	2.0410	2.0374	2.0455	2.0457	2.0439	2.0506
$\frac{1}{128}/\frac{1}{256}$	4.0098	2.0186	2.0168	2.0203	2.0215	2.0207	2.0236
$\frac{1}{256}/\frac{1}{512}$	4.0046	2.0088	2.0079	2.0095	2.0104	2.0101	2.0114
	Numerical order for component y_2						
$\frac{1}{32}/\frac{1}{64}$	4.0551	2.0893	2.0833	2.0960	2.0995	2.0966	2.1102
$\frac{1}{64}/\frac{1}{128}$	4.0248	2.0385	2.0352	2.0413	2.0445	2.0432	2.0488
$\frac{1}{128}/\frac{1}{256}$	4.0118	2.0178	2.0161	2.0191	2.0210	2.0204	2.0229
$\frac{1}{256}/\frac{1}{512}$	4.0056	2.0085	2.0077	2.0091	2.0102	2.0099	2.0111

Table 2.10: Numerical order of Problem 4 (2.58)

Time	AM	GM	HaM	HeM	RM	CeM	CoM
	Global error for component y_1						
0.0	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
0.1	6.666e-08	3.031e-05	6.036e-05	1.010e-05	3.078e-05	2.051e-05	6.184e-05
0.2	6.340e-08	3.243e-05	6.458e-05	1.081e-05	3.290e-05	2.192e-05	6.609e-05
0.3	4.408e-08	2.789e-05	5.555e-05	9.299e-06	2.826e-05	1.882e-05	5.677e-05
0.4	2.603e-08	2.270e-05	4.521e-05	7.573e-06	2.297e-05	1.530e-05	4.613e-05
0.5	1.315e-08	1.835e-05	3.655e-05	6.125e-06	1.854e-05	1.234e-05	3.723e-05
0.6	5.076e-09	1.499e-05	2.987e-05	5.007e-06	1.512e-05	1.007e-05	3.037e-05
0.7	4.592e-10	1.244e-05	2.478e-05	4.155e-06	1.253e-05	8.343e-06	2.516e-05
0.8	1.935e-09	1.046e-05	2.085e-05	3.497e-06	1.053e-05	7.013e-06	2.115e-05
0.9	3.000e-09	8.898e-06	1.774e-05	2.975e-06	8.951e-06	5.960e-06	1.797e-05
1.0	3.318e-09	7.624e-06	1.520e-05	2.549e-06	7.665e-06	5.103e-06	1.539e-05
	Global error for component y_2						
0.0	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
0.1	6.964e-08	3.706e-06	7.394e-06	1.198e-06	3.924e-06	2.631e-06	7.837e-06
0.2	6.820e-08	2.709e-07	5.458e-07	5.125e-08	4.604e-07	3.242e-07	8.841e-07
0.3	4.986e-08	3.610e-06	7.192e-06	1.235e-06	3.498e-06	2.316e-06	7.051e-06
0.4	3.222e-08	6.278e-06	1.251e-05	2.116e-06	6.232e-06	4.142e-06	1.253e-05
0.5	1.936e-08	7.648e-06	1.524e-05	2.566e-06	7.646e-06	5.087e-06	1.536e-05
0.6	1.106e-08	8.050e-06	1.605e-05	2.695e-06	8.073e-06	5.373e-06	1.621e-05
0.7	6.054e-09	7.841e-06	1.563e-05	2.623e-06	7.876e-06	5.243e-06	1.581e-05
0.8	3.191e-09	7.297e-06	1.455e-05	2.439e-06	7.335e-06	4.884e-06	1.472e-05
0.9	1.620e-09	6.601e-06	1.316e-05	2.206e-06	6.638e-06	4.420e-06	1.332e-05
1.0	7.930e-10	5.864e-06	1.169e-05	1.959e-06	5.898e-06	3.928e-06	1.184e-05

Table 2.11: The global errors for Problem 4 (2.58)

2.8 Methods with Error Estimation

In practical computation we want to choose step size h_i such as on one hand is sufficiently small to achieve required precision but on the other hand sufficiently large to avoid unnecessary computational work. For this reason we need some error estimation.

There are rigorous results about error of RK methods, see [21, p. 156], but their use for practical computing is quite limited. Rigorous bounds on error estimation require the computation and majorization of several partial derivatives of high orders. Since RK methods do not need

computation of derivatives, theoretical approach is quite impractical.

First approach was by Runge – repeat the computations with halved step size. Then results are compared and those digits which have not change are assumed to be correct. Then the idea of using Richardson extrapolation came. Now it is widely accepted that embedded methods are efficient. They usually combine two methods. In this work will be discussed two type of this approach. First case is embedded method using two methods of different order. And then alternative idea will be presented – method involving two methods of the same order.

Embedded Methods

Idea of embedded methods is to combine two RK methods with different order, but with the same stages i.e. a_{ij} and c_i coefficients are same for both method, but with different coefficients b_i and order. Thus we have common stages:

$$k_i = f\left(t_n + c_i h, y_n + h\sum_{j=1}^{s} a_{ij}k_j\right) \qquad i = 1, ..., s$$

and first method

$$y_{n+1} = y_n + h\sum_{j=1}^{s} b_j k_j \tag{2.59}$$

of order p and second method

$$\hat{y}_{n+1} = y_n + h\sum_{j=1}^{s} \hat{b}_j k_j \tag{2.60}$$

with order $\hat{p}$ (usually $\hat{p} = p + 1$ or $\hat{p} = p - 1$). Coefficients of both method can be written in extended Butcher tableau

$\mathbf{c}$	$\mathbf{A}$
	$\mathbf{b}^T$
	$\hat{\mathbf{b}}^T$

.

Let's now assume that $\hat{p} = p + 1$, in this case the local error, used for automatic step size selection, is estimated by

$$est_{n+1} = |\hat{y}_{n+1} - y_{n+1}| = h\sum_{j=1}^{s}(\hat{b}_j - b_j)k_j = h\sum_{j=1}^{s} d_j k_j. \tag{2.61}$$

If we use the value y_{n+1} to continue the sequence of approximated solution, we call it method without local extrapolation. In this case, we do not need to compute values of $\hat{y}_{n+1}$ but only the error estimation, sometimes the Butcher tableau for embedded methods contains also coefficients d_i, thus looks following:

$\mathbf{c}$	$\mathbf{A}$
	$\mathbf{b}^T$
	$\hat{\mathbf{b}}^T$
	$\mathbf{d}^T$

.

One of the first who derived embbeded methods was Fehlberg. In [18] he introduced also method known as Runge–Kutta–Fehlberg 4(5), where numbers 4(5) indicate orders of y_{n+1} and $\hat{y}_{n+1}$ respectively. This method is given by following tableau:

$$
\begin{array}{c|cccccc}
0 & & & & & & \\
\frac{1}{4} & \frac{1}{4} & & & & & \\
\frac{3}{8} & \frac{3}{32} & \frac{9}{32} & & & & \\
\frac{12}{13} & \frac{1932}{2197} & -\frac{7200}{2197} & \frac{7296}{2197} & & & \\
1 & \frac{439}{216} & -8 & \frac{3680}{513} & -\frac{845}{4104} & & \\
\frac{1}{2} & -\frac{8}{27} & 2 & -\frac{3544}{2565} & \frac{1859}{4104} & -\frac{11}{40} & \\
\hline
 & \frac{25}{216} & 0 & \frac{1408}{2565} & \frac{2197}{4104} & -\frac{1}{5} & 0 \\
\hline
 & \frac{16}{135} & 0 & \frac{6656}{12825} & \frac{28561}{5630} & -\frac{9}{50} & \frac{2}{55}
\end{array}
.
$$

Practically we usually use higher order approximation $\hat{y}_{n+1}$ as an approximation of solution. Then we use $\hat{y_n}$ in (2.59), (2.60) instead of y_n. This approach is called local extrapolation. Dormand and Prince in [12] presented RK method of order 5(4) for purpose of local extrapolation with additional property to reduce computational cost. This property is called FSAL - 'first same as last'. It means that last row of A is the same as the $\hat{b}_i$ coefficients of output, as can be seen in the following tableau:

$$
\begin{array}{c|ccccccc}
0 & & & & & & & \\
\frac{1}{5} & \frac{1}{5} & & & & & & \\
\frac{3}{10} & \frac{3}{40} & \frac{9}{40} & & & & & \\
\frac{4}{5} & \frac{44}{45} & -\frac{56}{15} & \frac{32}{9} & & & & \\
\frac{8}{9} & \frac{19372}{6561} & -\frac{25360}{2187} & \frac{644448}{6561} & -\frac{212}{729} & & & \\
1 & \frac{9017}{3168} & -\frac{355}{33} & \frac{46732}{5247} & \frac{49}{176} & -\frac{5103}{18656} & & \\
1 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & \\
\hline
 & \frac{5179}{57600} & 0 & \frac{7571}{16695} & \frac{393}{640} & -\frac{92097}{339200} & \frac{187}{2100} & \frac{1}{40} \\
\hline
 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & 0
\end{array}
.
$$

That makes Dormand–Prince method same expensive in terms of computational cost as Runge–Kutta–Fehlber 4(5), even if it has more stages because the last stage k_7 can be used as first stage k_1 in next time step.

Another examples of embedded RK methods are methods by Verner [33], or Cash and Karp method of order 4(5) [10] and also well know method of order 3(2) with local extrapolation by Bogacki and Shampine [3].

Embedded methods with automatic step size control are wildly used in software for solving ODE problems. As an example MATLAB can be mentioned, where are two functions with implemented embedded RK method. First is `ode23` using Bogacki–Shampine 3(2) method and the second one is function `ode45` based on Dormand–Prince 5(4) method.

Methods of Same Order

As another possibility of estimating error appeared idea using two different RK methods but of the same order. D. J. Evans and A. R. Yaakub presented in [16] error estimation strategy

based on pair RKAM and RKCoM. And in [28] the combination of RKAM and RKCeM were published. Now we will present the error estimation for both of these pairs of methods and discussed them in light of result from sections 2.6 and 2.7.

If we combine RKAM and RKCeM methods we get RKACeM method

$$\begin{aligned}
k_1 = k_1^* &= f(t_n, y_n) \\
k_2 = k_2^* &= f\left(t_n + \frac{h}{2}, y_n + \frac{1}{2}hk_1\right) \\
k_3 &= f\left(t_n + \frac{h}{2}, y_n + \frac{1}{2}hk_2\right) \\
k_4 &= f\left(t_n + h, y_n + hk_3\right) \\
k_3^* &= f\left(t_n + \frac{h}{2}, y_n + \frac{1}{24}hk_1 + \frac{11}{24}hk_2\right) \\
k_4^* &= f\left(t_n + \frac{h}{2}, y_n + \frac{1}{12}hk_1 - \frac{25}{132}hk_2 + \frac{73}{66}hk_3^*\right) \\
y_{n+1} &= y_n + \frac{h}{3}\left(\sum_{i=1}^{3}\frac{k_i + k_{i+1}}{2}\right) \\
y_{n+1}^* &= y_n + \frac{h}{3}\left(\frac{2}{3}\sum_{i=1}^{3}\frac{(k_i^*)^2 + k_i^*k_{i+1}^* + (k_{i+1}^*)^2}{k_i^* + k_{i+1}^*}\right).
\end{aligned}$$

The authors in [28] established the error estimation as follows. For local truncation errors of each method the following holds

$$\begin{aligned}
y_{n+1}^{AM} &= y(t_{n+1}) + LTE_{AM} \\
y_{n+1}^{CeM} &= y(t_{n+1}) + LTE_{CeM}
\end{aligned}$$

therefore taking difference between RKAM and RKCeM gives

$$y_{n+1}^{AM} - y_{n+1}^{CeM} = LTE_{AM} - LTE_{CeM}. \tag{2.62}$$

The local truncation error of RKAM is given by

$$LTE_{AM} = \frac{h^5}{2880}\left(-24ff_y^4 + f^4f_{yyyy} + 2f^3f_yf_{yyy} - 6f^3f_{yy}^2 + 36f^2f_y^2f_{yy}\right)$$

while the local truncation error of the RKCeM is given by

$$LTE_{CeM} = \frac{h^5}{69120}\left(-762ff_y^4 + 8f^4f_{yyyy} + 36f^3f_yf_{yyy} - 744f^3f_{yy}^2 + 273f^2f_y^2f_{yy}\right)$$

Then the absolute difference between LTE_{AM} and LTE_{CeM} is obtained as

$$|LTE_{AM} - LTE_{CeM}| = \frac{h^5}{69120}\left(186ff_y^4 + 16f^4f_{yyyy} + 12f^3f_yf_{yyy} + 600f^3f_{yy}^2 + 591f^2f_y^2f_{yy}\right) \tag{2.63}$$

Now we bound f and its partial derivative using the following relations suggested in [27]:

$$\begin{aligned}
|f(t,y)| &< Q \\
\left|\frac{\partial^{i+j}f(t,y)}{\partial t^i\partial y^j}\right| &< \frac{P^{i+j}}{Q^{j-1}},\ i+j \le p
\end{aligned}$$

where P, Q are positive constants, p is the order of the method and this holds for $t \in [t_0, T]$ and $y \in (-\infty, \infty)$. In this case $p = 4$ we obtain

$$|f f_y^4| < QP^4$$
$$|f^4 f_{yyyy}| < \frac{Q^4 P^4}{Q^3}$$
$$|f^3 f_y f_{yyy}| < Q^3 \frac{P P^3}{Q^2}$$
$$|f^3 f_{yy}^2| < Q^3 \left(\frac{P^2}{Q}\right)^2$$
$$|f^2 f_y^2 f_{yy}| < Q^2 P^2 \frac{P^2}{Q}$$

thus all derivatives in (2.63) are bounded by QP^4, hence it becomes

$$LTE_{AM} - LTE_{CeM} \leq \frac{281}{13824} P^4 Q h^5$$

and thus by (2.62) also

$$|y_{n+1}^{AM} - y_{n+1}^{CeM}| \leq \frac{281}{13824} P^4 Q h^5.$$

And as error estimation we use difference between the RKAM method and RKCeM method together with the constant derived above

$$est_n = |y_n^{AM} - y_n^{CeM}| \frac{281}{13824}. \tag{2.64}$$

The method RKACoM combines RKAM and RKCoM methods.

$$\begin{aligned}
k_1 = k_1^* &= f(t_n, y_n) \\
k_2 = k_2^* &= f\left(t_n + \frac{h}{2}, y_n + \frac{1}{2} h k_1\right) \\
k_3 &= f\left(t_n + \frac{h}{2}, y_n + \frac{1}{2} h k_2\right) \\
k_4 &= f\left(t_n + h, y_n + h k_3\right) \\
k_3^* &= f\left(t_n + \frac{h}{2}, y_n + \frac{1}{8} h k_1 + \frac{3}{8} h k_2\right) \\
k_4^* &= f\left(t_n + \frac{h}{2}, y_n + \frac{1}{4} h k_1 - \frac{3}{4} h k_2 + \frac{3}{2} h k_3^*\right) \\
y_{n+1} &= y_n + \frac{h}{3}\left(\sum_{i=1}^{3} \frac{k_i + k_{i+1}}{2}\right) \\
y_{n+1}^* &= y_n + \frac{h}{3}\left(\sum_{i=1}^{3} \frac{(k_i^*)^2 + (k_{i+1}^*)^2}{k_i^* + k_{i+1}^*}\right).
\end{aligned}$$

and by the same procedure as for RKACeM the error estimation was established as

$$est_n = |y_n^{AM} - y_n^{CeM}| \frac{1405}{23040}. \tag{2.65}$$

The above derivation of the bound on the local truncation errors are done under assumption that both method are of order 4. It was shown in Section 2.6 that the methods are fourth order only in case of autonomous scalar problem. In any other case the RKCeM method is only second order and therefore the constant is not needed. The difference between those two method should be sufficient error estimation but probably overrated, as can be seen from numerical experiments.

2.9 Automatic Step Size Selection

If we have some local error estimation, we can use it for step size selection to achieve a prescribed tolerance of the local error. The prescribed tolerance can be written component wise in form

$$\varepsilon_i = max\left\{Atol_i, \max\{|y_{n,i}|, |y_{n+1,i}|\} \cdot Rtol_i\right\}, \quad i = 1, 2, ..., m,$$

where $Atol$ is chosen absolute tolerance and $Rtol$ is chosen relative tolerance. For $Atol = 0$ only relative errors are considered and for $Rtol = 0$ the absolute ones. But usually both tolerances are different from zero.

To consider the step from y_n to y_{n+1} as successful, we want to an estimation of the local error satisfy $|est_{n,i}| \leq \varepsilon_i$. To measure the error we use Euclidean norm

$$\|err\| = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\frac{est_{n,i}}{\varepsilon_i}\right)^2}, \tag{2.66}$$

but also maximum norm is frequently used. Since local error of the method of order p is proportional to h^{p+1} also $\|err\| \approx Ch^{p+1}$ holds. For optimal step size $1 \approx Ch_{opt}^{p+1}$. From that, by eliminating the constant C, we have the optimal time step:

$$h_{opt} = h\left(\frac{1}{\|err\|}\right)^{\frac{1}{q+1}}$$

where $q = \min(p, \hat{p})$. To prevent rejection in the next step the optimal step is multiplied by a safety factor θ, usually is equal to 0.8 or 0.9. In our numerical experiments we will use $\theta = 0.8$ Also we do not want h to increase or decrease too fast, thus we introduce factors $\theta_{min}, \theta_{max}$ and as the new time step we put

$$h_{new} = h \min\left\{\theta_{max}, \max\left\{\theta_{min}, \theta\left(\frac{1}{\|err\|}\right)^{\frac{1}{q+1}}\right\}\right\}. \tag{2.67}$$

We reject the step if $\|err\| \leq 1$ does not hold. We do a new computation of y_{n+1} with h_{new}. When the inequality $\|err\| \leq 1$ is satisfied, we pass over to the next step also with h_{new}.

Factor θ_{max} is usually chosen between 1.5 and 5, but in addition to this, for the step right after the rejected one $\theta_{max} = 1$. Factor θ_{min} is chosen 0.2. Sometimes is add even the maximal step size h_{max} and minimal step size h_{min}.

Starting Step Size

The last thing we are missing to complete automatic step size control is the length of starting step. It will be presented algorithm from [22] which is based on the hypothesis that

$$\text{local error} \approx Ch^{p+1}(y)^{p+1}(t_0).$$

But we do not know $(y)^{p+1}(t_0)$ and thus we need to replace it by approximations of the first and second derivative of the solution. This is done as follows: First we compute $d_0 = \|y_0\|$ and $\|f(t_0, y_0)\|$ using the same norm as we used above in err with $\varepsilon_i = Atol_i + |y_0|Rtol_i$. Then the first guess $h_0 = 0.001\frac{d_0}{d_1}$ is made. When d_0 or d_1 is smaller than 10^{-5} we set $h_0 = 10^{-6}$. Using Euler method we compute $y_1 = y_0 + h_0 f(t_0, y_0)$ and use it for estimation of second derivative

$$d_2 = \frac{\|f(t_0 + h_0, y_1) - f(t_0, y_0)\|}{h_0}.$$

Then we compute h_1 from relation $h_1^{p+1}\max\{d_1, d_2\} = 0.01$. If $\max\{d_1, d_2\} \leq 10^{-15}$ we put $h_1 = \max\{10^{-6}, h_0\, 10^{-3}\}$, and finally we set the starting step size as

$$h_{start} = \min\{100\, h_0, h_1\}.$$

2.10 Numerical Testing of Methods with Error Estimation

In this section some numerical results about error estimation methods will be presented. Methods RKACoM and RKACeM were compared with embedded methods Runge–Kuta–Fehlberg 4(5) (RKF45), Cash–Karp 4(5) (CK45) and local extrapolation method Dormand–Prince 5(4) (DP54). The problems were also computed by MATLAB solvers ode23 (based on Bogacki–Shampine method) and by MATLAB implementation of Dormand–Prince method ode45.

All methods were implemented in MATLAB (ExplicitEmbeddedMethods). The algorithms described in Section 2.9 were used for step size selection with absolute tolerance $Atol = 10^{-6}$ and relative tolerance $Rtol = 10^{-3}$. Euclidean norm was used. For every test problem we present the number of accepted steps, the number of rejected steps and the maximal error in each component.

Problem 1

We start our testing with autonomous scalar equation

$$\begin{aligned} y' &= \sqrt{y}, \quad t \in [1,4] \\ y(1) &= 1 \end{aligned} \tag{2.68}$$

with exact solution

$$y(t) = \frac{1}{4}(t+1)^2.$$

Since the problem is autonomous equation as error estimator for methods RKACoM and RKACeM the estimation with the constants (2.64) and (2.65), respectively, were used. In Figures 2.5 and 2.6 we present behaviour of the solution and selection of time steps these methods

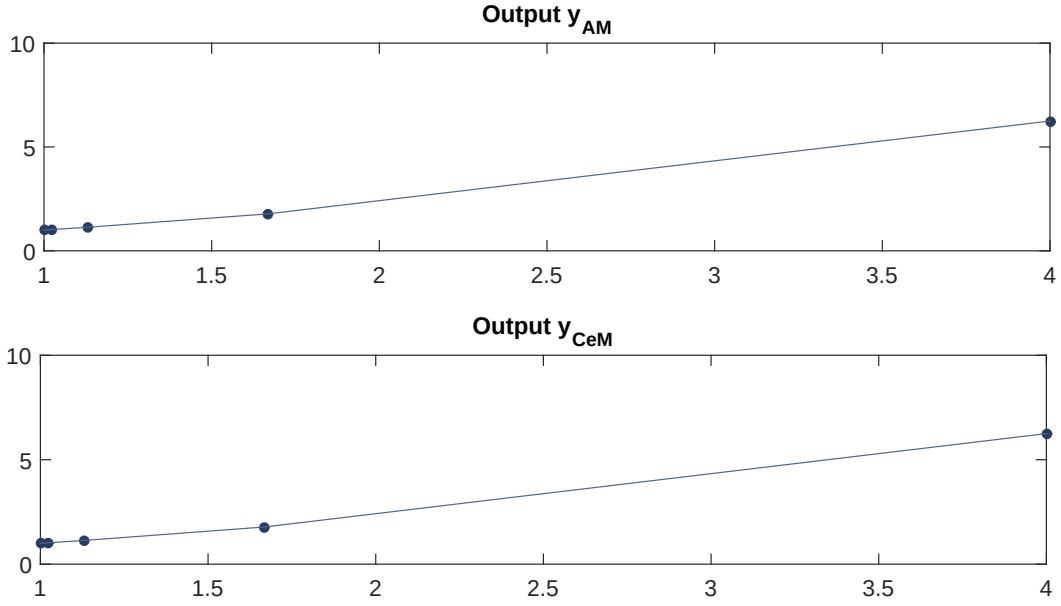


Figure 2.5: Numerical solution obtained by RKACeM with step size control of Problem 1 (2.68)

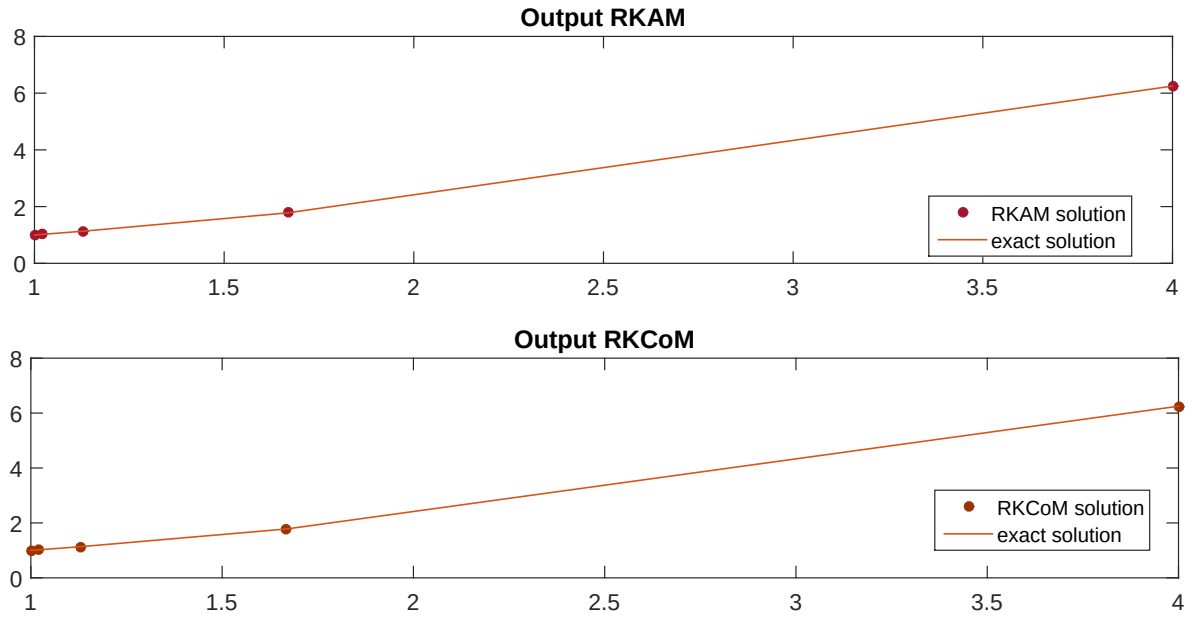


Figure 2.6: Numerical solution obtained by RKACoM with step size control of Problem 1 (2.68)

If we compare the all the methods they performed quite similar results. For modified method we would like to see if there is a difference in which method is used as output. That we can see in Figures 2.5 for RKACeM and 2.6 for RKCoM. It seems that all of the solutions are the same. Actually, in all four cases the solution was computed at points:

$$t_0 = 1,\ t_1 = 1.0216,\ t_2 = 1.1293,\ t_3 = 1.16681,\ t_4 = 4,$$

which means that the chosen output method did not influenced the error estimation. But the values of y_n are not the same.

Problem 2

The second problem for testing of error estimating methods was chosen scalar equation with non-constant coefficient

$$\begin{aligned} y' &= y\cos(t), \quad t \in [0, 8], \\ y(0) &= 1 \end{aligned} \tag{2.69}$$

with exact solution

$$y(t) = e^{\cos(t)},$$

which is non-monotonous function on the given interval.

For this problem the local error was estimated for RKACoM method as

$$est_n = |y_{AM} - y_{CoM}| \tag{2.70}$$

and for RKACeM method analogously. In Table 2.14 we can see the significant difference in number of steps in case of modified methods and the embedded methods. As was assumed the modified methods performed much more steps (accepted and rejected) than the embedded methods.

Because the modified methods (RKCoM, RKCem) do not give a good approximation around points with zero derivative, as have seen in Section 2.7, the step size around those points is for error estimating RKACoM and RKaCeM automatically chosen smaller, see Figure 2.7. Therefore we got the approximated solution at points where the solution changes the most. This is not generally bad but on the other hand much more computation was needed.

We wanted to verify the idea that using error estimation (2.64) or (2.65) for RKACoM and RKCeM methods is not working well. As we can see in Table **??**, in this case the number of steps decreased. On the other hand the errors increased. In the Figure 2.8 is presented the solution obtained by RKACeM with both outputs. In case of RKCeM output the result is really inaccurate. For the RKAM output we got bigger maximal error than in computation with *est* given by (2.70) however comparable with the results of embedded method RKF45, CK45 and DP45.

	RKACeM		RKACoM	
output	RKAM	RKCeM	RKAM	RKCoM
No. of accepted steps	12	11	19	19
No. of rejected steps	4	2	5	5
Maximal error	1.5695e-02	2.2384e-01	3.4522e-03	1.4135e-01

Table 2.12: Table for Problem 2 (2.69) modified methods with error estimation (2.64),(2.65)

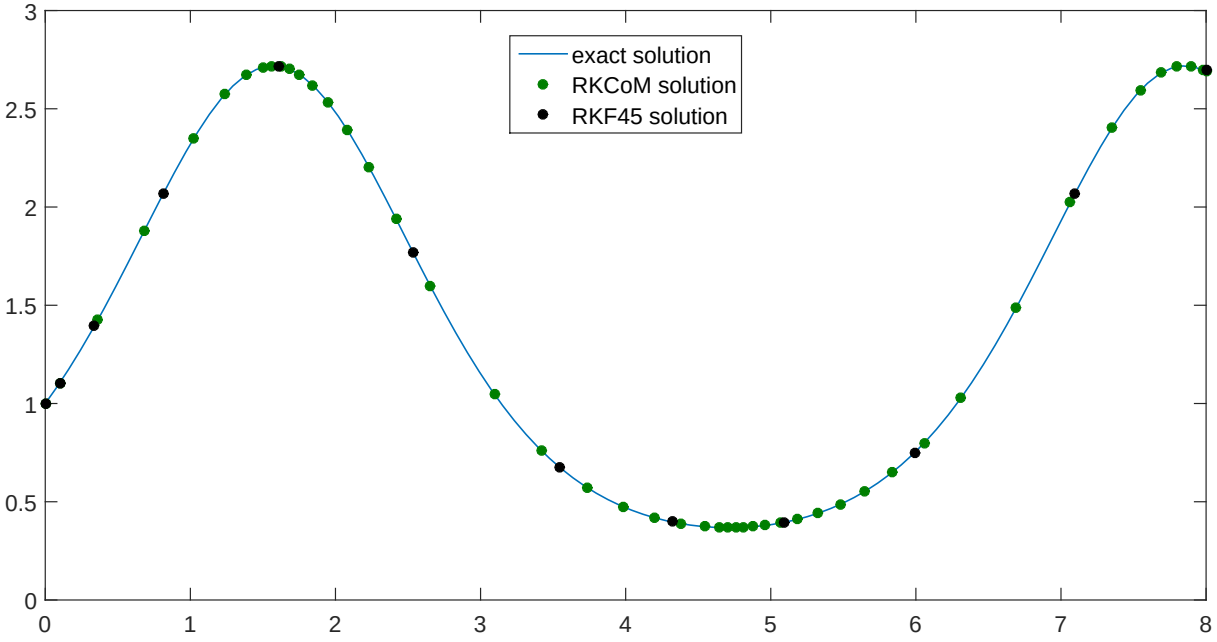


Figure 2.7: The solutions computed by RKACoM and RKF45

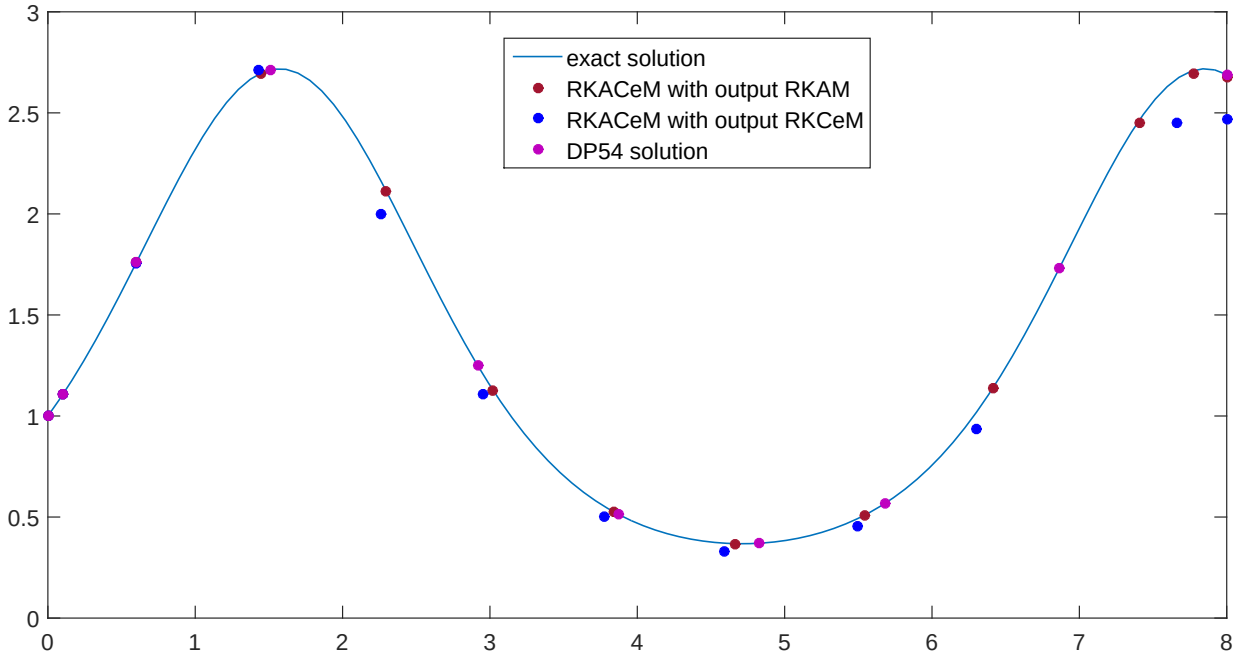


Figure 2.8: The solutions computed by RKACeM and DP54

Problem 3

The last problem we chose a linear system of two equations

$$\begin{aligned} y' &= \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} y + \begin{pmatrix} \mathrm{tg}^2(t) + 1 \\ \mathrm{tg}(t) \end{pmatrix}, \quad t \in [0, 1.5], \\ y(0) &= \begin{pmatrix} 1 \\ 3 \end{pmatrix} \end{aligned} \tag{2.71}$$

with exact solution

$$y(t) = \begin{pmatrix} \cos(t) + \sin(t) + \mathrm{tg}(t) \\ -\sin(t) + \cos(t) + 2 \end{pmatrix}.$$

As in previous problem modified methods RKACeM and RKACoM performed more steps than other embedded methods but since the solution is monotonous, the difference is not so significant an in Problem 2. From Table 2.15 it is also seem the better precision mainly in case of RKAM output.

The modified method with error estimation overestimated the actual error and because of the inaccuracy of modified methods, around points with zero derivative. Accurate results are obtained only in case of RKAM output. The number of steps corresponds with method of lower order (ode23 - method of order 3). The possible application could be in case we need better illustration of the solution and also we want to avoid some unnecessary computational work in parts where solution does not varies a lot.

output	RKACeM		RKACoM		CK45	RKF45	DP54	ode45	ode23
	RKAM	RKCeM	RKAM	RKCoM					
Number of accepted steps	4	4	4	4	4	5	4	3	4
Number of rejected steps	0	0	0	0	0	0	0	0	0
Maximal error	1.5417e-02	1.2081e-02	3.6901e-03	7.4162e-04	7.3576e-06	3.6335e-03	4.2706e-05	3.2842e-03	1.1597e-02

Table 2.13: Table for problem 1 (2.68)

output	RKACeM		RKACoM		CK45	RKF45	DP54	ode45	ode23
	RKAM	RKCeM	RKAM	RKCoM					
Number of accepted steps	35	35	47	46	8	11	9	10	28
Number of rejected steps	10	12	14	13	0	1	1	3	5
Maximal error	1.1032e-03	8.3985e-03	1.1623e-04	1.1404e-02	1.3067e-02	1.1031e-02	5.4524e-04	1.6515e-03	7.4176e-03

Table 2.14: Table for problem 2 (2.69)

output	RKACeM		RKACoM		CK45	RKF45	DP54	ode45	ode23
	RKAM	RKCeM	RKAM	RKCoM					
Number of accepted steps	15	15	19	19	10	10	9	6	16
Number of rejected steps	8	8	5	5	7	6	6	4	7
Maximal error in y_1	1.9399e-03	4.0395e-02	6.3715e-04	6.4648e-02	3.3420e-04	4.2334e-03	5.9691e-05	1.4099e+01	1.4088e+01
Maximal error in y_2	2.0197e-04	1.5688e-03	9.5860e-05	2.4670e-03	1.0034e-04	2.2513e-03	1.0457e-05	1.6957e-04	1.0191e-03

Table 2.15: Table for problem 3 (2.71)

3 | Implicit Runge–Kutta Methods

Explicit Runge–Kutta methods are easy to implement formula. Unfortunately, they suffer from lack of stability (as we will see in Chapter 4), therefore they are not suitable for treating stiff problems also as singular perpetuated problems or differential-algebraic systems.

Implicit Runge–Kutta methods have several subgroup of methods. First group is fully implicit methods. These methods have almost none of coefficients of matrix **A** in Butcher tableau equal to zero. Every stage k_i, $i = 1, 2, ..., s$ depends on previous and also following stages therefore there is needed solution of generally non-linear system of ms equations. To make it easier the idea of diagonal implicit Runge–Kutta methods arise. If the coefficients above the diagonal are zero, the solution of system of equations is easier, instead of whole system of ms equations we can solve one equation after another.

3.1 Collocation Methods

As the first we will present derivation of collocation methods. First methods as Implicit Euler, implicit midpoint rule or trapezoidal rule are collocation methods. To derive the collocation method we start from differential equation

$$y'(t) = f(t, y(t))$$

and we convert it to integral equation by integrating over interval $[t_n, t]$

$$\int_{t_n}^{t} y'(r)\mathrm{d}r = \int_{t_n}^{t} f(r, y(r))\mathrm{d}r$$

and we compute the integral on the left hand side we obtain

$$y(t) = y(t_n) + \int_{t_n}^{t} f(r, y(r))\mathrm{d}r.$$

Now we replace the exact value $y(t_n)$ by approximated one y_n and the integrand by a polynomial interpolation

$$y(t) \approx y_n + \int_{t_n}^{t} p(r)\mathrm{d}r, \tag{3.1}$$

where $p(r)$ is a unique interpolation polynomial of degree $< s$ which interpolates points $[t_{n,i}, f(t_{n,i}, y(t_{n,i})],\ i = 1, 2, ..., s$, where $t_{n,i} \equiv t_n + \tau_i h,\ i = 1, ..., s.$ and $0 \leq \tau_1 < ... < \tau_s \leq 1$.

Using Lagrange interpolating polynomial we have

$$p(r) = \sum_{j=1}^{s} f(t_{n,j}, y(t_{n,j}))l_j(r), \tag{3.2}$$

where $l_j(r)$ are fundamental Lagrange polynomials defined as

$$l_j(r) = \prod_{\substack{i=1 \\ i \neq j}}^{s} \left(\frac{r - r_i}{r_j - r_i} \right), \qquad j = 1, ..., s.$$

Plug (3.2) into (3.1) we obtain

$$y(t) \approx y_n + \sum_{j=1}^{s} f(t_{n,j}, y(t_{n,j})) \int_{t_n}^{t} l_j(r) \mathrm{d}r.$$

Now the key of collocation method will come. We force equality of above expression at all the points $t_{n,j}$, $i = 1, 2, ..., s$. Thus approximations $y_{n,j}$ for $j = 1, ..., s$ of exact values $y(t_{n,j})$ at collocation node points are determined by following system of non-linear equations

$$y_{n,i} = y_n + \sum_{j=1}^{s} f(t_{n,i}, y_{n,j}) \int_{t_n}^{t_{n,i}} l_j(r) \mathrm{d}r, \qquad i = 1, ..., s. \tag{3.3}$$

In case $\tau_s = 1$ we put $y_{n+1} = y_{n,s}$. Otherwise we set

$$y_{n+1} = y_n + \sum_{j=1}^{s} f(t_{n,j}, y_{n,j}) \int_{t_n}^{t_{n+1}} l_j(r) dr. \tag{3.4}$$

For example, case $s = 2$ we will have Lagrange interpolation polynomial

$$p(r) = f(t_{n,1}, y(t_{n,1})) \frac{(t_{n,2} - r)}{h(\tau_2 - \tau_1)} + f(t_{n,2}, y(t_{n,2})) \frac{(r - t_{n,1})}{h(\tau_2 - \tau_1)}$$

and thus the resulting method will have Butcher tableau in the following form

τ_1	$\frac{2\tau_2 - \tau_1^2}{2(\tau_2 - \tau_1)}$	$-\frac{\tau_1^2}{2(\tau_2 - \tau_1)}$
τ_2	$\frac{\tau_2^2}{2(\tau_2 - \tau_1)}$	$\frac{\tau_2^2 - 2\tau_1}{2(\tau_2 - \tau_1)}$
	$\frac{2\tau_2 - 1}{2(\tau_2 - \tau_1)}$	$\frac{1 - 2\tau_1}{2(\tau_2 - \tau_1)}$

where coefficients a_{ij} and b_i are obtained from integrals in (3.3) and (3.4) respectively. The particular examples of collocation methods will be presented in the following section.

3.2 General Construction of Fully Implicit Runge–Kutta Methods

Since collocation is not the only way how to derive IRK method we will present general approach to construct those methods too. We derived order conditions in Section 2.3, which are valid also for implicit methods. But in that case we need to handle with almost two times number of coefficients. J. C. Butcher presented in [8] five conditions (we already mentioned condition $D(1)$ in Section 2.4) which are known as simplifying assumptions. The construction of implicit RK method relies on three of them

$$B(p) : \sum_{i=1}^{s} b_i c_i^{k-1} = \frac{1}{k}, \quad k = 1, 2, .., p$$

$$C(q) : \sum_{j=1}^{s} a_{ij} c_j^{k-1} = \frac{c_i^k}{k}, \quad k = 1, 2, ..., q, \ i = 1, 2, ..., s$$

$$D(r) : \sum_{i=1}^{s} b_i c_i^{k-1} a_{ij} = \frac{b_j}{k}(1 - c_j^k), \quad k = 1, 2, ..., r, \ j = 1, 2, ..., s, .$$

Condition $B(p)$ means that the quadrature formula

$$\int_t^{t+h} f(s)\mathrm{d}s \approx h\sum_{i=1}^{s} b_i f(t+c_i h)$$

is exact for all polynomials of degree $< p$ and if this condition is satisfied we say that the RK method has quadrature order p.

And the conditions $C(q)$ has similar meaning. If it is satisfied, the corresponding quadratures on $[t, t+c_i h]$

$$\int_t^{t+c_i h} f(s)\mathrm{d}s \approx h\sum_{j=1}^{s} a_{ij} f(t+c_j h)$$

are exact for all polynomials of degree $< q$ and the RK method is called of stage order q. As is proved in [21], methods which satisfy $C(s)$ (where s is as usual number of stages) and have all c_i, $i = 1, 2, ..., s$ distinct are collocation methods.

Also in [8], Butcher stated and proved the following important theorem about order.

Theorem 3.1. *If a Runge–Kutta method satisfies conditions $B(p), C(q)$, and $D(r)$ with $p \leq q + r + 1$ and $p \leq 2q + 2$, its order of convergence is p.*

Proof. See [8]. □

To simplify the construction of implicit method a little bit, we use following lemma.

Lemma 3.2. *Let a Runge–Kutta with s stages have distinct $c_1, c_2, ..., c_s$ and non-zero weights $b_1, b_2, ..., b_s$. Then we have*

1. *$C(s)$ and $B(s+\nu)$ imply $D(\nu)$*
2. *$D(s)$ and $B(s+\nu)$ imply $C(\nu)$.*

Proof. See [22]. □

Hence, we can construct the method with using $B(p)$ condition and $C(q)$ or $D(r)$.

Now we can move to derivation the methods. Fully implicit methods are divided in three groups according to quadrature formula on which are based.

Gauss Methods

Gauss methods are collocation methods based on the Gaussian quadrature formula, which has the maximal order if the nodes are the roots of Legendre polynomial P_s on the interval $[-1, 1]$. We need the formula on the interval $[0, 1]$ therefore we take c_i, $i = 1, 2, ..., s$ as roots of shifted Legendre polynomial of order s:

$$P_s^* = \frac{1}{s!}\frac{\mathrm{d}^s}{\mathrm{d}t^s}\left(t^s\left(t-1\right)^s\right). \tag{3.5}$$

First we take a look on quadrature formula associated to c_i's.

Lemma 3.3. *Let $c_1, c_2, ...c_s$ denote the roots of P_s^*. Then there exist positive numbers $b_1, b_2, ..., b_s$ such that*

$$\int_0^1 \phi(t)\mathrm{d}t = \sum_{i=1}^{s} b_i \phi(c_i) \tag{3.6}$$

for any polynomial of degree less then $2s$. The b_i are unique.

Proof. See [7]. □

Therefore, for c_i chosen as roots of P_s^* the condition $B(2s)$ holds and it is enough to find the s coefficients b_i only from the first s equations of condition $B(2s)$ i.e. it is sufficient to satisfy only $B(s)$.

Thus if we have satisfied $B(2s)$ and $C(s)$, condition $D(s)$ is also satisfied and the method is by the Theorem 3.1 of order $2s$. We obtain the coefficients a_{ij} from the condition $C(s)$.

Let's do it for $s = 1$. From (3.5) we have

$$P_1^* = \frac{\mathrm{d}}{\mathrm{d}t}\left(t(t-1)\right) = 2t - 1$$

we obtain node $c_1 = t = \frac{1}{2}$. Condition $B(1)$ becomes just one equation

$$b_1 c_1^0 = 1,$$

from which we immediately see $b_1 = 1$. This we use in condition $C(1)$: $b_1 a_{11} = b_1 - b_1 c_1$, thus we have $a_{11} = 1 - 1 \cdot \frac{1}{2} = \frac{1}{2}$. So, we will get method given by Butcher tableau

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

which is also know as *implicit midpoint rule*.

For two stage method we need to find c_1 and c_2 as roots of polynomial

$$P_2 = \frac{\mathrm{d}^2}{\mathrm{d}t^2}\left(t^2(t-1)^2\right) = 12t^2 - 12t + 2.$$

After really short computation we get: $c_1 = \frac{1}{2} - \frac{\sqrt{3}}{6}$, $c_2 = \frac{1}{2} + \frac{\sqrt{3}}{6}$. For b's, from $B(2)$ we get system of two equations

$$\begin{aligned} b_1 + b_2 &= 1 \\ b_1 c_1 + b_2 c_2 &= \frac{1}{2} \end{aligned}$$

which solution is $b_1 = b_2 = \frac{1}{2}$ and for coefficients a_{ij}, from $C(2)$ we have the following system of four linear equation is

$$\begin{aligned} a_{11} + a_{21} &= \frac{1}{2} - \frac{\sqrt{3}}{6} \\ a_{21} + a_{22} &= \frac{1}{2} + \frac{\sqrt{3}}{6} \\ a_{11}\left(\frac{1}{2} - \frac{\sqrt{3}}{6}\right) + a_{12}\left(\frac{1}{2} + \frac{\sqrt{3}}{6}\right) &= \frac{1}{2} \cdot \left(\frac{1}{2} - \frac{\sqrt{3}}{6}\right)^2 \\ a_{21}\left(\frac{1}{2} - \frac{\sqrt{3}}{6}\right) + a_{22}\left(\frac{1}{2} + \frac{\sqrt{3}}{6}\right) &= \frac{1}{2} \cdot \left(\frac{1}{2} + \frac{\sqrt{3}}{6}\right)^2 \end{aligned}$$

Its solution with other coefficients are in Butcher tableau

$$\begin{array}{c|cc} \frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{2} & \frac{3-2\sqrt{3}}{12} \\ \frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{3+2\sqrt{3}}{12} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

and is simply called *Gauss 2-stage* method.

If we would done the same procedure for three stages, the resulting method will have following Butcher tableau

$$\begin{array}{c|ccc} \frac{1}{2}-\frac{\sqrt{15}}{10} & \frac{5}{36} & \frac{2}{9}-\frac{\sqrt{15}}{15} & \frac{5}{36}-\frac{\sqrt{15}}{30} \\ \frac{1}{2} & \frac{5}{36}+\frac{\sqrt{15}}{24} & \frac{2}{9} & \frac{5}{36}-\frac{\sqrt{15}}{24} \\ \frac{1}{2}+\frac{\sqrt{15}}{10} & \frac{5}{36}+\frac{\sqrt{15}}{30} & \frac{2}{9}+\frac{\sqrt{15}}{15} & \frac{5}{36} \\ \hline & \frac{5}{18} & \frac{4}{9} & \frac{5}{18} \end{array}$$

and again referred simply as *Gauss 3-stage*.

Theorem 3.4. *The s-stage Gauss method is of order 2s.*

Proof. See [8]. □

Thus above mentioned methods have respectively order 2, 4 and 6.

Radau Methods

The second group of fully implicit methods is based on Radau quadrature formulas and it is divided in two subgroups - Radau I and Radau II methods. We start with Radau I methods, which are based on Radau left quadrature formula of order $2s-1$. We can find c_i as roots of polynomial

$$P_s^* + P_{s-1}^* = \frac{\mathrm{d}^{s-1}}{\mathrm{d}t^{s-1}}\left(t^s\,(t-1)^{s-1}\right) \tag{3.7}$$

and it holds $c_1 = 0$ for any s.

Radau II methods are based on Radau right quadrature formula of order $2s-1$ and coefficients c_i are chosen as roots of polynomial

$$P_s^* - P_{s-1}^* = \frac{\mathrm{d}^{s-1}}{\mathrm{d}t^{s-1}}\left(t^{s-1}\,(t-1)^{s}\right) \tag{3.8}$$

and it follows $c_s = 1$.

First attempt to contract such methods was by J.C. Butcher in [5]. For first group coefficients b_i of his methods satisfy $B(s)$ and coefficients a_{ij} are chosen such that $c_1 = a_{11} = a_{12} = ... = a_{1s} = 0$ therefore $k_1 = f(t_n, y_n)$. This made first stage explicit and condition $C(q)$ hold for any q. An example of Butcher's Radau I 2-stage method is in the following Butcher tableau:

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

In case of methods based on (3.8) coefficients a_{ij} were chosen such that $a_{1s} = a_{2s} = ... = a_{ss} = 0$. This made the final stage explicit and condition $D(r)$ holds for any r. As example of Butcher's 2-stage Radau II method, we present this one

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

But the idea to make the method as close to explicit as possible turn out not be convenient. These methods are not A-stable and nobody used them nowadays.

Then the idea to define a_{ij} in Radau I methods by condition $D(s)$ came. To distinguished this type of method from the Butcher's, there was added A in the name of the method. From order of Radau quadrature formula we have satisfied $B(2s-1)$, from above, we force coefficients to satisfy $D(s)$. By the Lemma 3.2 we have fulfilled condition $C(s-1)$. Therefore, Radau IA methods are not collocation methods. By Theorem 3.1, for $p = 2s-1$, $q = s-1$ and $r = s$, we have $2s-1 \leq 2s$ and $2s-1 \leq 2s$, which means that Radau IA methods are of order $s-1$ [7].

Now we start with $s = 1$. From (3.7) we have

$$\frac{\mathrm{d}^0}{\mathrm{d}t^0}\left(t(t-1)^0\right) = t$$

and $c_1 = 0$ and from the $B(1)$ condition we again have $b_1 = 1$. As the last we determine a_{11} from $D(1)$:

$$b_1 a_{11} = b_1(1 - c_1).$$

The resulting method has Butcher tableau:

$$\begin{array}{c|c} 0 & 1 \\ \hline & 1 \end{array}.$$

When the number of stages equals 2, the coefficients $c_1,\ c_2$ are obtained as roots of polynomial

$$\frac{\mathrm{d}}{\mathrm{d}t}(t^2(t-1)) = 3t^2 - 2t.$$

Those are $c_1 = 0$, $c_2 = \frac{2}{3}$ and again with help of condition $B(2)$ we get system of equations

$$\begin{aligned} b_1 + b_2 &= 1 \\ b_1 \cdot 0 + b_2 \cdot \frac{2}{3} &= \frac{1}{2} \end{aligned}$$

which solve $b_1 = \frac{1}{4}$, and $b_2 = \frac{1}{4}$. Using $D(2)$ we write down the system of equation for desired a's:

$$\begin{aligned} \frac{1}{4}a_{11} + \frac{3}{4}a_{21} &= \frac{1}{4} \\ \frac{1}{4}a_{12} + \frac{3}{4}a_{22} &= \frac{1}{4} \\ \frac{1}{2}a_{21} &= \frac{1}{8} \\ \frac{1}{2}a_{22} &= \frac{15}{72}. \end{aligned}$$

Directly from two last equations we can compute $a_{22} = \frac{5}{12}$ and $a_{21} = \frac{1}{4}$. Then we plug them into first and second equation to obtain all coefficients of two stage Radau IA method:

$$\begin{array}{c|cc} 0 & \frac{1}{4} & -\frac{1}{4} \\ \frac{2}{3} & \frac{1}{4} & \frac{5}{12} \\ \hline & \frac{1}{4} & \frac{3}{4} \end{array}.$$

Analogously Radau IA method with three stages can be constructed:

$$\begin{array}{c|ccc} 0 & \frac{1}{9} & \frac{-1-\sqrt{6}}{18} & \frac{-1-\sqrt{6}}{18} \\ \frac{6-\sqrt{6}}{10} & \frac{1}{9} & \frac{88+7\sqrt{6}}{360} & \frac{88-43\sqrt{6}}{360} \\ \frac{6+\sqrt{6}}{10} & \frac{1}{9} & \frac{88+43\sqrt{6}}{360} & \frac{88-7\sqrt{6}}{360} \\ \hline & \frac{1}{9} & \frac{16+\sqrt{6}}{36} & \frac{16-\sqrt{6}}{36} \end{array}.$$

For Radau IIA coefficients c_i are roots of (3.8).The coeffients b_i are same as in case of Radau IA, based on condition $B(s)$ and for a_{ij} condition $C(s)$ was imposed, therefore we will obtain collocation methods. Since the radau right quadrature formula is of order $2s-1$, condition $B(s-1)$ is satisfied [7]. From Lemma 3.2 holds $D(s-1)$ and by Theorem 3.1 the Radau IIA methods are of order $2s-1$.

Such one stage method has butcher tableau

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

and it is well-known *implicit Euler method* with first order of accuracy.

For $s=2$ we get from (3.8) roots $c_1=\frac{1}{3}$ a $c_2=1$ and $B(2)$ condition gives us

$$\begin{aligned} b_1+b_2 &= 1 \\ b_1\frac{1}{3}+b_2 &= \frac{1}{2} \end{aligned}$$

and then we compute a_{ij} from system of linear equations

$$\begin{aligned} a_{11}+a_{12} &= c_1 \\ a_{21}+a_{22} &= c_2 \\ a_{11}c_1+a_{12}c_2 &= \frac{c_1^2}{2} \\ a_{21}c_1+a_{22}c_2 &= \frac{c_2^2}{2}. \end{aligned}$$

As result we obtain third order method given by tableau:

$$\begin{array}{c|cc} \frac{1}{3} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{3}{4} & \frac{1}{4} \\ \hline & \frac{3}{4} & \frac{1}{4} \end{array}.$$

Three stage Radau IIA method of fifth order has tableau:

$$\begin{array}{c|ccc} \frac{4-\sqrt{6}}{10} & \frac{88-7\sqrt{6}}{360} & \frac{296-169\sqrt{6}}{1800} & \frac{-2+3\sqrt{6}}{225} \\ \frac{4+\sqrt{6}}{10} & \frac{296+169\sqrt{6}}{1800} & \frac{88+7\sqrt{6}}{360} & \frac{-2-3\sqrt{6}}{225} \\ 1 & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \\ \hline & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \end{array}.$$

Lobatto Methods

Lobatto methods, often referred as Lobatto III methods, to preserve some consistency in naming methods, are based on Lobatto quadrature formula. If the coefficients c_i are chosen as roots of polynomials

$$P_s^* - P_{s-2}^* = \frac{\mathrm{d}^{s-2}}{\mathrm{d}t^{s-2}} \left(t^{s-1} (t-1)^{s-1} \right) \tag{3.9}$$

where s is again the number of stages, clearly, $c_1 = 0$ and $c_s = 1$ for every s, the corresponding quadrature formula

$$\int_0^1 \phi(t)\mathrm{d}t \approx \sum_{i=1}^{s} b_i \phi(c_i)$$

is precise for any polynomial of degree less than $2s - 2$ [7].

Obviously the lowest number of stages is two, in that case the polynomial has form

$$\frac{\mathrm{d}^0}{\mathrm{d}t^0}(t(t-1)) = t(t-1)$$

thus we have $c_1 = 0, c_2 = 1$. And in case $s = 3$, we search the roots of polynomial

$$\frac{\mathrm{d}}{\mathrm{d}t}(t^2(t-1)^2) = 4t^3 - 6t^2 + 2t$$

we obtain $c_1 = 0$, $c_2 = \frac{1}{2}$ and $c_3 = 1$. Then coefficients b_i are computed again with help of condition $B(s)$. For case $s = 2$ from solving the system

$$\begin{aligned} b_1 + b_2 &= 1 \\ b_2 &= \frac{1}{2} \end{aligned}$$

we have $b_1 = b_2 = \frac{1}{2}$ and for three stages by system

$$\begin{aligned} b_1 + b_2 + b_3 &= 1 \\ b_2\frac{1}{2} + b_3 &= \frac{1}{2} \\ b_2\frac{1}{4} + b_3\frac{1}{9} &= \frac{1}{3} \end{aligned}$$

we obtain $b_1 = \frac{1}{3}$, $b_2 = \frac{2}{3}$ and $b_3 = \frac{1}{6}$.

Now we are missing only coefficients a_{ij}. Again the first who tried to construct Lobatto method was by J.C. Butcher. He wanted to make close to explicit method as possible, as in case of Radau methods. The elements of the first row and last column of matrix $\mathbf{A}$ have been set as zeros. Thanks to that first and last stage become explicit and number of implicit stages reduces to $s - 2$. But there is the same problem as in case of Radau methods, lack of stability. As an example of Butcher's Lobatto method the three 3-stage method we present only:

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & 0 \\ 1 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array} .$$

Later three groups of Lobatto III methods were established according to way of choosing coefients a_{ij}. Those groups are denoted by letters A, B and C. For Lobatto IIIA coefficients a_{ij} satisfy condition $C(s)$, i.e. in case $s = 2$ have to satisfy following system of equation

$$\begin{aligned} a_{11} + a_{12} &= 0 \\ a_{21} + a_{22} &= 1 \\ a_{21} &= 0 \\ a_{22} &= \frac{1}{2} \end{aligned}$$

thus the method has following Butcher tableau:

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

and we see that this is *trapezoidal method*. Analogously we can get the method with three stages with Butcher tableau:

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{5}{24} & \frac{1}{3} & -\frac{1}{24} \\ 1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

For the second group, Lobatto IIIB method, we impose condition $D(s)$. For $s = 2$ such method does not exist. In case $s = 3$ we need to satisfy system of nine following equations:

$$\begin{aligned} \frac{1}{6}a_{11} + \frac{2}{3}a_{21} + \frac{1}{6}a_{31} &= \frac{1}{6} \\ \frac{1}{6}a_{12} + \frac{2}{3}a_{22} + \frac{1}{6}a_{32} &= \frac{1}{3} \\ \frac{1}{6}a_{13} + \frac{2}{3}a_{23} + \frac{1}{6}a_{33} &= 0 \\ \frac{1}{3}a_{21} + \frac{1}{6}a_{31} &= \frac{1}{12} \\ \frac{1}{3}a_{22} + \frac{1}{6}a_{32} &= \frac{1}{4} \\ \frac{1}{3}a_{23} + \frac{1}{6}a_{33} &= 0 \\ \frac{1}{24}a_{21} + \frac{1}{6}a_{31} &= \frac{1}{18} \\ \frac{1}{24}a_{22} + \frac{1}{6}a_{32} &= \frac{7}{36} \\ \frac{1}{24}a_{23} + \frac{1}{6}a_{33} &= 0. \end{aligned}$$

If we solve this system and add the coefficients c_1, c_2, c_3 and b_1, b_2, b_3 we get Butcher tableau:

$$\begin{array}{c|ccc} 0 & \frac{1}{6} & -\frac{1}{6} & 0 \\ \frac{1}{2} & \frac{1}{6} & \frac{1}{3} & 0 \\ 1 & \frac{1}{6} & \frac{5}{6} & 0 \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}.$$

The 4-stage method can be obtained similarly as the previous one. We present here only resulting Butcher tableau:

$$\begin{array}{c|cccc} 0 & 0 & \frac{-1-\sqrt{5}}{24} & \frac{-1+\sqrt{5}}{24} & 0 \\ \frac{5-\sqrt{5}}{10} & \frac{1}{12} & \frac{25+\sqrt{5}}{120} & \frac{25-13\sqrt{5}}{120} & 0 \\ \frac{1}{12} & \frac{11+\sqrt{5}}{120} & \frac{25+13\sqrt{5}}{120} & \frac{25-\sqrt{5}}{120} & 0 \\ 1 & \frac{1}{12} & \frac{1-\sqrt{5}}{120} & \frac{1+\sqrt{5}}{120} & 0 \\ \hline & \frac{1}{12} & \frac{5}{12} & \frac{5}{12} & \frac{1}{12} \end{array}.$$

Finally for Lobatto IIIC we set $a_{i1} = b_1$, for $i = 1, ..., s$ and the other coefficients a are determined by $C(s-1)$. Therefore, for $s = 2$ we have $a_{11} = a_{21} = \frac{1}{2}$ and from system

$$\frac{1}{2} + a_{12} = 0$$
$$\frac{1}{2} + a_{22} = 1$$

we obtain the rest of coefficients to complete following Butcher tableau:

$$\begin{array}{c|cc} 0 & \frac{1}{2} & -\frac{1}{2} \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}.$$

The Lobatto IIIC method with three stages can be derived in the same manner as the previous 2-stage method and its Butcher tableau is:

$$\begin{array}{c|ccc} 0 & \frac{1}{6} & -\frac{1}{3} & \frac{1}{6} \\ \frac{1}{2} & \frac{1}{6} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}.$$

In the end of this section, we are going to discuss the order of Lobbatto methods using the Lemma 3.2 and Theorem 3.1. By order of quadrature formula, $B(2s-2)$. For Lobatto IIIA we have satisfied $C(s)$, therefore $D(s-2)$ and the order is $2s-2$. For Lobatto IIIB $B(2s-2)$ and $D(s)$ imply $C(s-2)$ and the order is again $2s-2$. For Lobatto IIIC $C(s-1)$ holds, therefore $D(s-1)$ and its order is $2s-2$.

3.3 Simplified Newton Method

It was already mentioned that we need to solve non-linear system of equations for s stages. By (1.2) we obtain following non-linear system

$$\begin{pmatrix} k_1 \\ k_2 \\ \vdots \\ k_s \end{pmatrix} = \begin{pmatrix} f(t_n + c_1 h, y_n + h\sum_{i=1}^{s} a_{1i}k_i) \\ f(t_n + c_2 h, y_n + h\sum_{i=1}^{s} a_{2i}k_i) \\ \vdots \\ f(t_n + c_s h, y_n + h\sum_{i=1}^{s} a_{si}k_i) \end{pmatrix}. \tag{3.10}$$

The vector of stages k_i will by denoted by $\mathbf{K} = (k_1, k_2, \dots\ k_s)^T$ and the right hand side of (3.10) by $\mathbf{F}(\mathbf{K})$. If we apply Newton method to (3.10) we get

$$\begin{pmatrix} I - ha_{11}\frac{\partial f}{\partial y}(t_n + c_1 h, y_n + k_1) & \cdots & -ha_{1s}\frac{\partial f}{\partial y}(t_n + c_1 h, y_n + k_s) \\ \vdots & & \vdots \\ -ha_{s1}\frac{\partial f}{\partial y}(t_n + c_s h, y_n + k_1) & \cdots & I - ha_{ss}\frac{\partial f}{\partial y}(t_n + c_s h, y_n + k_s) \end{pmatrix} \Delta\mathbf{K}^k = -\mathbf{K}^k + \mathbf{F}(\mathbf{K}^k), \tag{3.11}$$

where the superscript denotes the number of Newton iteration. To simplify (3.11) we instead of all Jacobians $\frac{\partial f}{\partial y}(t_n + c_i h, y_n + k_i)$ we use only

$$J \approx \frac{\partial f}{\partial y}(t_n, y_n), \tag{3.12}$$

therefore we get simplified Newton method for (3.10)

$$(I - hA \otimes J)\,\Delta\mathbf{K}^k = -\mathbf{K}^k + \mathbf{F}(\mathbf{K}^k).$$

3.4 Embedded Methods and Step Size Selection

As in case of explicit RK methods we want to established some kind of error estimation. Now we also use the idea of combining two methods with the same stages but different output. In previous section some methods has been derived. Those are of optimal order. In other word it is impossible to embed method of higher order with the same number of stages. Thus we search for lower order method.

We will present two approaches of establishing the implicit embedded methods. First is creating pair of method with the same number of stages. This was presented in [30]. Then the second approach of professor E. Hairer will be presented.

The simplifying assumptions $B(p)$ can be rewritten in matrix-vector notation

$$\mathbf{b}^T\mathbf{c}^{k-1} = \frac{1}{k} \tag{3.13}$$

where the power of vector is taken element-wise $\mathbf{c}^{k-1} = (c_1^{k-1}, ..., c_s^{k-1})^T$

First we choose coefficients c_i according to (3.5) or (3.7) or (3.8) or (3.9) for corespondig kind of method. Then we need conditions $B(1), B(2), ...B(s)$ to be satisfied

$$\begin{aligned} \mathbf{b}^T\mathbf{e} &= 1 \\ \mathbf{b}^T\mathbf{c} &= \frac{1}{2} \\ &\vdots \\ \mathbf{b}^T\mathbf{c}^{s-1} &= \frac{1}{s} \end{aligned}$$

where $\mathbf{e}$ denotes vector of ones $\mathbf{e} = (1, 1, ..., 1)^T$. These conditions can written in matrix form

$$V_s\mathbf{b} = \mathbf{e}_H \tag{3.14}$$

where $\mathbf{e}_H^T$ is defined as $\mathbf{e}_H^T = \left(1, \frac{1}{2}, ..., \frac{1}{s}\right)$ and V_s is so-called Vandermonde matrix:

$$V_s = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ c_1 & c_2 & \cdots & c_s \\ c_1^2 & c_2^2 & \cdots & c_s^2 \\ \vdots & \vdots & \ddots & \vdots \\ c_1^{s-1} & c_2^{s-1} & \cdots & c_s^{s-1} \end{pmatrix}. \tag{3.15}$$

From properties of Vandermonde matrix we know if all c_i are distinct, then the determinant is non-zero and the inverse exist [30].

Pairs of Methods with Same Number of Stages

As first we make the Vandermonde matrix (3.15) for coefficients c_i of Gauss method. From (3.14) we have

$$\hat{\mathbf{b}} = V_s^{-1}\hat{\mathbf{e}}_H \tag{3.16}$$

where $\hat{\mathbf{e}}_H^T = \left(1, \frac{1}{2}, ..., \frac{1}{s-1}, 0\right)$. From (3.16) coefficients $\hat{b}$ satisfy condition $B(s-1)$ therefore resulting method is of order $s-1$. In particular case of 2-stage Gauss method we obtain pair of order (4,1)

$$\begin{array}{c|cc} \frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{2} & \frac{3-2\sqrt{3}}{12} \\ \frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{3+2\sqrt{3}}{12} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} + \frac{\sqrt{3}}{2} & \frac{1}{2} - \frac{\sqrt{3}}{2} \end{array}$$

and for 3-stage Gauss method pair of order (6,2)

$$\begin{array}{c|ccc} \frac{1}{2} - \frac{\sqrt{15}}{10} & \frac{5}{36} & \frac{2}{9} - \frac{\sqrt{15}}{15} & \frac{5}{36} - \frac{\sqrt{15}}{30} \\ \frac{1}{2} & \frac{5}{36} + \frac{\sqrt{15}}{24} & \frac{2}{9} & \frac{5}{36} - \frac{\sqrt{15}}{24} \\ \frac{1}{2} + \frac{\sqrt{15}}{10} & \frac{5}{36} + \frac{\sqrt{15}}{30} & \frac{2}{9} + \frac{\sqrt{15}}{15} & \frac{5}{36} \\ \hline & \frac{5}{18} & \frac{4}{9} & \frac{5}{18} \\ \hline & -\frac{5}{6} & \frac{8}{3} & -\frac{5}{6}. \end{array}$$

Now we will do the same procedure for Radau methods. Radau methods are of $2s-1$ order hence we obtain pair of order (2s-1, s-1). Radau IA methods of orders (3,1) and (5,2) are

$$\begin{array}{c|cc} 0 & \frac{1}{4} & -\frac{1}{4} \\ \frac{2}{3} & \frac{1}{4} & \frac{5}{12} \\ \hline & \frac{1}{4} & \frac{3}{4} \\ \hline & 1 & 0 \end{array} \qquad \begin{array}{c|ccc} 0 & \frac{1}{9} & \frac{-1-\sqrt{6}}{18} & \frac{-1-\sqrt{6}}{18} \\ \frac{6-\sqrt{6}}{10} & \frac{1}{9} & \frac{88+7\sqrt{6}}{360} & \frac{88-43\sqrt{6}}{360} \\ \frac{6+\sqrt{6}}{10} & \frac{1}{9} & \frac{88+43\sqrt{6}}{360} & \frac{88-7\sqrt{6}}{360} \\ \hline & \frac{1}{9} & \frac{16+\sqrt{6}}{36} & \frac{16-\sqrt{6}}{36} \\ \hline & -1 & 1+\frac{7\sqrt{6}}{13} & 1-\frac{7\sqrt{6}}{13} \end{array}.$$

and Radau IIA methods of orders (3,1) and (5,2) are

$\frac{1}{3}$	$\frac{5}{12}$	$\frac{-1}{12}$
1	$\frac{3}{4}$	$\frac{1}{4}$
	$\frac{3}{4}$	$\frac{1}{4}$
	$\frac{3}{2}$	$-\frac{1}{2}$

$\frac{4-\sqrt{6}}{10}$	$\frac{88-7\sqrt{6}}{360}$	$\frac{296-169\sqrt{6}}{1800}$	$\frac{-2+3\sqrt{6}}{225}$
$\frac{4+\sqrt{6}}{10}$	$\frac{296+169\sqrt{6}}{1800}$	$\frac{88+7\sqrt{6}}{360}$	$\frac{-2-3\sqrt{6}}{225}$
1	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$
	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$
	$1-\frac{7\sqrt{6}}{12}$	$1+\frac{7\sqrt{6}}{12}$	-1

.

Since the Lobatto methods have order 2s-2, the pair of embedded methods for $s = 2$ is of order (2,1) and for $s = 3$ (4,2). These Lobbato IIIA methods are

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

0	0	0	0
$\frac{1}{2}$	$\frac{5}{24}$	$\frac{1}{3}$	$-\frac{1}{24}$
1	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$
	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$
	$-\frac{1}{2}$	2	$-\frac{1}{2}$

There is no 2-stage Lobatto IIIB method thus also embedded methods with 2 stages does not exist. So, we introduce only one method (4,2)

0	$\frac{1}{6}$	$-\frac{1}{6}$	0
$\frac{1}{2}$	$\frac{1}{6}$	$\frac{1}{3}$	0
1	$\frac{1}{6}$	$\frac{5}{6}$	0
	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$
	$-\frac{1}{2}$	2	$-\frac{1}{2}$

.

In case of Lobatto IIIC the embedded method for two and three stages of order (2,1) and (4,2) are following:

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

0	$\frac{1}{6}$	$-\frac{1}{3}$	$\frac{1}{6}$
$\frac{1}{2}$	$\frac{1}{6}$	$\frac{5}{12}$	$-\frac{1}{12}$
1	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$
	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$
	$-\frac{1}{2}$	2	$-\frac{1}{2}$

.

There are not so many suitable pairs of methods. Only 2-stage Lobbato IIIA and Lobbatto IIIC have desirable pair of order $(p, p-1)$. In other cases the error estimation would be larger and the step size smaller.

Hairer's Embedded Method

Professor E. Hairer is author of code RADAU5 which is based on fifth order Radau IIA method with three stages. To established the error estimation he embedded method with four stages with explicit first stage

$$\begin{array}{c|cc} c_0 = 0 & 0 & \mathbf{0} \\ \mathbf{c} & \mathbf{0} & \mathbf{A} \\ \hline & \hat{b}_0 & \hat{\mathbf{b}}^T \end{array}$$

where $\hat{b}_0 \neq 0$. Then $\hat{y}_{n+1}$ is computed as

$$\hat{y}_{n+1} = y_n + h\left(\hat{b}_0 f(t_n, y_n) + \sum_{i=1}^{3} \hat{b}_i k_i\right)$$

and the difference is taken as error estimation

$$est = \hat{y}_{n+1} - y_{n+1} = \hat{b}_0 h f(t_n, y_n) + \sum_{i=1}^{3} e_i k_i \tag{3.17}$$

where $e_i = (\hat{b}_i - b_i)$ and then $\mathbf{e} = \left(\frac{-2-3\sqrt{6}}{6}\hat{b}_0, \frac{-2+3\sqrt{6}}{6}\hat{b}_0, -\frac{1}{3}\hat{b}_0\right)$. E. Hairer recomended (and used in his code) to take $\hat{b}_0$ as the real eigenvalue of the matrix $\mathbf{A}$ [22]. But it is not only posibility, Jacques J.B. de Swart and Gustaf Soderlind in [11] used $\hat{b}_0 = 0.02$.

If we assume problem $y' = \lambda y$ then for $h\lambda \to \infty$ the error estimation (3.17) behaves like $est \approx \hat{b}_0 h\lambda y_n$, which is unbounded and therefore not suitable for stiff equations. L. F. Shampine proposed in [32] to take error estimation as

$$est = \left(I - h\hat{b}_0 J\right)^{-1} (\hat{y}_n - y_n),$$

where J is the same Jacobian as in Newton method (3.12).

Step Size Selection

As in case of explicit embedded methods with error estimation we can control the step size using (2.67). But in case of stiff equation this can lead to many rejected steps. Since solving of the stiff problems often required a rapid decrease of step size, the new step size selection strategies have been introduced.

K. Gustafson in [19] presented predictive controller. Where the new step size depend on two previous steps and on two previous error estimations:

$$h_{new} = \theta h_n \left(\frac{1}{\|err_{n+1}\|}\right)^{\frac{1}{4}} \cdot \frac{h_n}{h_{n-1}} \left(\frac{\|err_n\|}{\|err_{n+1}\|}\right)^{\frac{1}{4}}, \tag{3.18}$$

where $\|err\|$ is defined the same way as in (2.66).

3.5 Numerical Testing of Implicit Embedded Methods

In this section the results of testing the implicit embedded methods from previous section will be presented. The embedded methods with same numbers of stages are referred as name of output and the Hairer's estimation is named Radau 5(3). All methods were implemented in MATLAB with step size controlled by (3.18) and as local extrapolation methods (ImplicitEmbeddedMethods.m). The tolerances was taken as $Atol = 1e-6$ and $Rtol = 1e-3$. The implicit embedded methods were also compared to explicit methods. There is no function based on implicit RK method in software MATLAB hence the comparison is only between methods implemented in this work.

Problem 1

The first testing problem is linear equation with non-constant coefficients

$$\begin{aligned} y' &= -2000(y - \cos(t)), \quad t \in [0, 5], \\ y_0 &= 1 \end{aligned} \tag{3.19}$$

with exact solution

$$y(t) = \frac{e^{-2000t} + 2000\sin(t) + 4000000\cos(t)}{4000001}.$$

In Table 3.1 we see the number of accepted and rejected steps for fifth order methods - explicit Dormand–Prince method, 3-stage Radau IA method and Radau IIA method with both error estimating strategies. The DP54 method needed to preform 3232 steps all implicit methods under 60. Thus we have no doubts that problem is stiff. As was assumed the number of steps in case of embedded methods with same stages (the estimation method has order 2) is bigger than in the case of Radau 5(3).

	DP54	Radau IA	Radau IIA	Radau 5(3)
No. of accepted steps	3232	59	48	16
No. of rejected steps	4	6	4	0
Maximal error	6.7849e-07	2.1939e-04	1.1526e-07	2.1967e-05

Table 3.1: Comparison of fifth order methods for Problem 1 (3.5)

The comparison of fourth order method is presented in Table 3.2. The explicit method again preformed much more steps. The Lobatto IIIA and 2-stage Gauss method did almost the same number of steps. On the other hand Lobatto IIIC method computed solution within 16 steps.

	RKF45	Lobatto IIIA	Lobatto IIIC	Gauss
No. of accepted steps	9558	48	16	49
No. of rejected steps	518	5	0	10
Maximal error	4.7091e-06	6.2811e-07	1.3048e-04	6.8026e-06

Table 3.2: Comparison of fourth order methods for Problem 2 (3.5)

In the Figure 3.1 the errors of solution and estimated errors are presented. From that we see that Gauss methods definitely over-estimated the error for this problem as the Radau IIA methods and Lobatto IIIA methods. On contrary Lobatto IIIC methods estimated the error quite well. The embedded Lobatto methods are of order 2(1) in case of 2-stage and 4(2) for 3-stage methods. The difference between order is not big. Lobatto IIIC are, in addition to this, stiffly accurate method, therefore they are suitable for treating stiff problems. Thus the presented Lobatto IIIC method seems like a good option to variable step size solver for stiff problems.

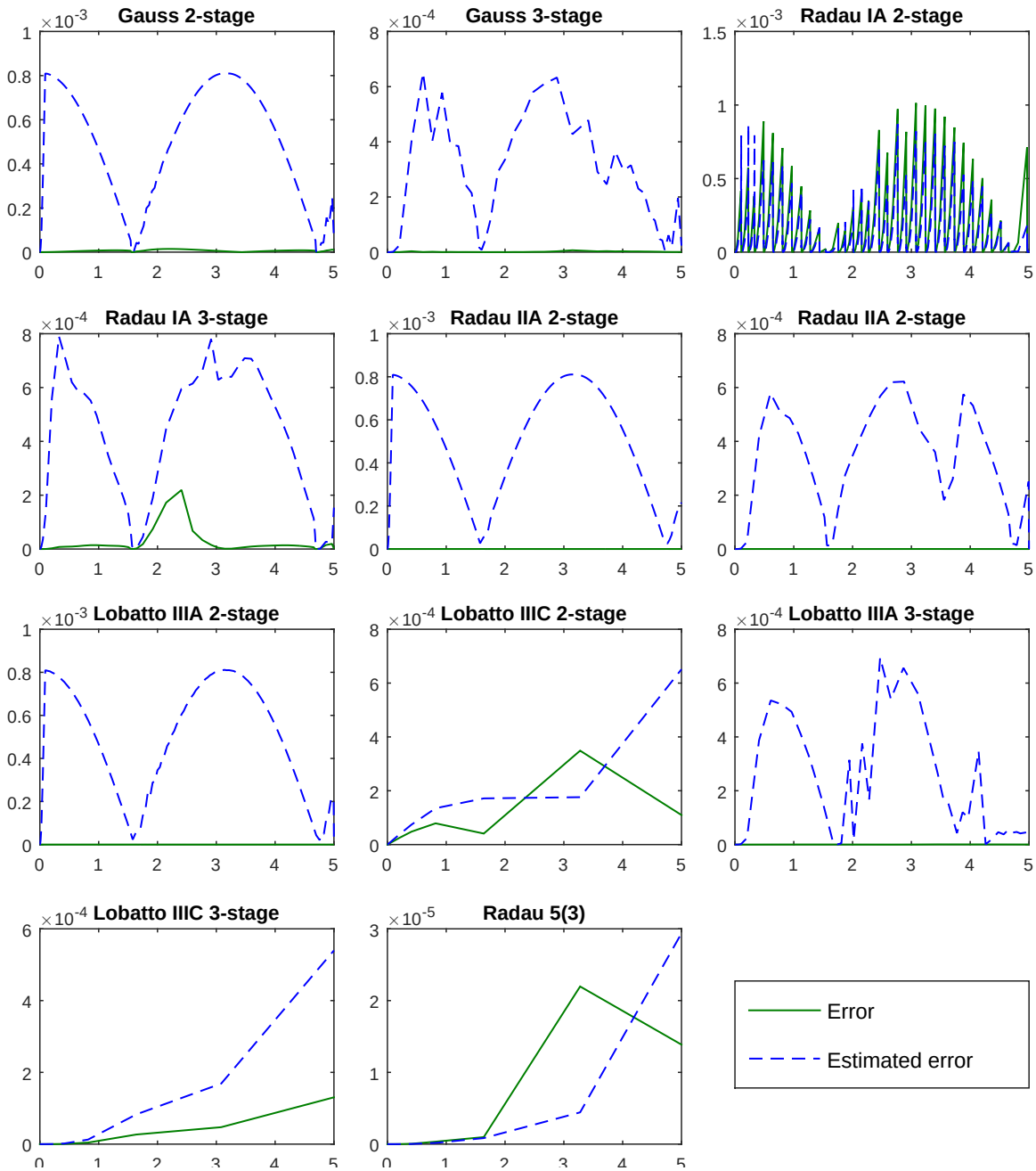


Figure 3.1: The error of solution and the estimated error for Problem 1 (3.5)

Problem 2

The second problem is non-linear system

$$\begin{aligned} y_1 =& -(\mu+2)y_1 + \mu y_2^2 \\ y_2 =& y_1 - y_2 - y_2^2, \qquad t \in [0, 10] \\ y_0 =& \begin{pmatrix} 1 \\ 1 \end{pmatrix} \end{aligned} \tag{3.20}$$

with exact solution independent of stiffness parameter μ

$$\begin{aligned} y_1(t) =& e^{-2t} \\ y_2(t) =& e^{-t}. \end{aligned}$$

Table 3.3 shows again number of accepted and rejected steps and maximal error in both components of solution for fifth order methods. The explicit performed much more steps. The same behaviour of explicit method is observed in case of fourth order methods, Table 3.4. Thus we easily see that the explicit method are not suitable for stiff problems.

Figures 3.2 and 3.3 shows the error and estimated error of solution in both components. Among the implicit fifth order method the best estimation provided the Radau 5(3) method and Lobatto IIIC was the best among the four order methods, as in previous problem.

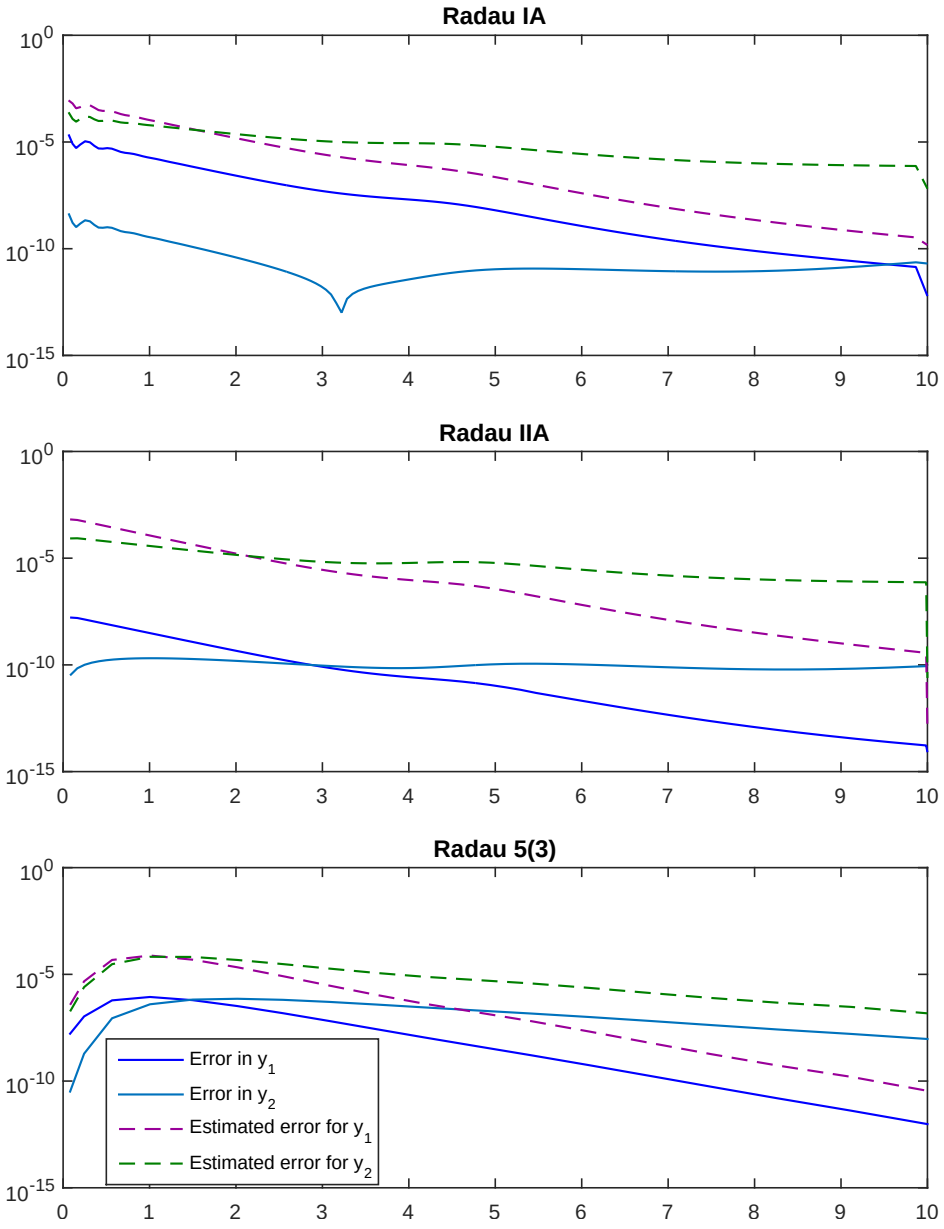


Figure 3.2: The error of solution and the estimated error for problem (3.20)

	DP54	Radau IA	Radau IIA	Radau 5(3)
No. of accepted steps	15396	113	73	18
No. of rejected steps	1016	2	0	0
Maximal error	7.5206e-07	2.2664e-05	1.6585e-08	8.7101e-07
Maximal error	1.5038e-10	4.5238e-09	2.0500e-10	7.1822e-07

Table 3.3: Comparison of fifth order methods for problem (3.20) with $\mu = 5000$

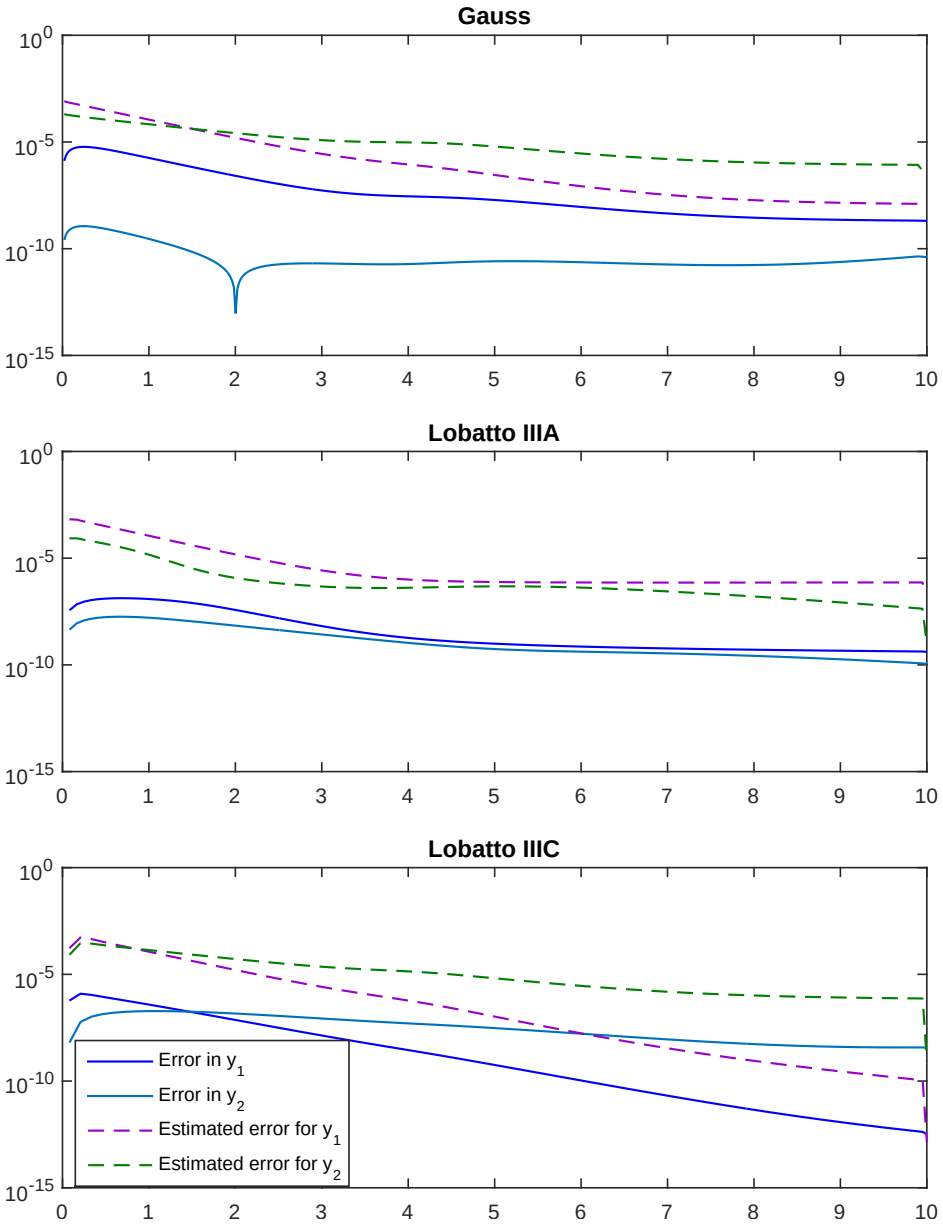


Figure 3.3: The error of solution and the estimated error for problem (3.20)

	RKF45	Lobatto IIIA	Lobatto IIIC	Gauss
No. of accepted steps	16337	144	57	87
No. of rejected steps	2013	0	0	22
Maximal error	1.4429e-04	1.3325e-07	1.2388e-06	1.1199e-06
Maximal error	1.4394e-04	1.8065e-08	1.8965e-07	2.2373e-10

Table 3.4: Comparison of fourth order methods for problem (3.20) with $\mu = 5000$

3.6 Diagonally Implicit Methods

The fully implicit method are very useful for treating stiff equations but they require solving the system of ms implicit equations at each time step. To decrease the cost methods with lower triangular matrix A were established. Some authors called them 'semi-explicit' or 'semi-implicit'. In this work are referred as diagonally implicit method (DIRK). Unfortunately some authors used this term only for methods which have all the diagonal elements equal.

The advantage of DIRK method is that we can solve the system of non-linear equations sequentially rather than as one great implicit system. The earliest method of DIRK type were implicit Euler method and implicit midpoint rule. Those are quite 'trivial' DIRK method. As example of the two methods with nice coefficients are presented (more of them can be found in [23])

$$\begin{array}{c|cc} \frac{1}{3} & \frac{1}{3} & \\ 1 & \frac{3}{4} & \frac{1}{4} \\ \hline & \frac{3}{4} & \frac{1}{4} \end{array} \qquad \begin{array}{c|ccc} 1 & 1 & & \\ \frac{1}{3} & -\frac{1}{12} & \frac{5}{12} & \\ 1 & 0 & \frac{3}{4} & \frac{1}{4} \\ \hline & 0 & \frac{3}{4} & \frac{1}{4} \end{array} .$$

Butcher's two stage Radau I or three stage Lobatto method are examples of first EDIRK methods - diagonal methods with one explicit stage. Hence the number of equation, which needs to be solved, reduces. If in addition the method is stiffly accurate, i.e. have FSAL property, then the value last stage can be used in next step as first one. Such method constructed M. Crouziex:

$$\begin{array}{c|ccccc} 0 & 0 & & & & \\ 2 & 1 & 1 & & & \\ 1 & \frac{5}{12} & -\frac{1}{12} & \frac{2}{3} & & \\ \frac{1}{2} & \frac{5}{24} & 0 & -\frac{1}{24} & \frac{1}{3} & \\ 1 & \frac{1}{6} & 0 & -\frac{5}{12} & \frac{2}{3} & \frac{7}{12} \\ \hline & \frac{1}{6} & 0 & -\frac{5}{12} & \frac{2}{3} & \frac{7}{12} \end{array} .$$

SDIRK

Singly diagonal implicit RK methods have all the diagonal elements of matrix **A** equal. They become very popular for treating the stiff equations. For this reason, they were usually constructed to have desirable properties – A-stability, L-stability or be stiffly accurate.

There is only one 1-stage method, the implicit Euler method, which is first order and well-known A-stable method. M. Crouzeix determined all 2-stage third order and 3-stage fourth order DIRK methods. There is exactly one A-stable method in each group and both are SDIRK methods. The proof of that can be also found in [1]. The 2-stage third order formula has Butcher tableau

$$\begin{array}{c|cc} \frac{1}{2}+\frac{1}{2\sqrt{3}} & \frac{1}{2}+\frac{1}{2\sqrt{3}} & 0 \\ \frac{1}{2}-\frac{1}{2\sqrt{3}} & -\frac{1}{3} & \frac{1}{2}-\frac{1}{2\sqrt{3}} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

and the 3-stage fourth order has following one

$$\begin{array}{c|ccc} \frac{1+\alpha}{2} & \frac{1+\alpha}{2} & & \\ \frac{1}{2} & -\frac{\alpha}{2} & \frac{1+\alpha}{2} & \\ \frac{1-\alpha}{2} & 1+\alpha & -(1+2\alpha) & \frac{1+\alpha}{2} \\ \hline & \frac{1}{6\alpha^2} & 1-\frac{1}{3\alpha^2} & \frac{1}{6\alpha^2} \end{array},$$

where $\alpha = 2\frac{\cos(\frac{\pi}{18})}{\sqrt{3}}$

R. Alexander in [1] presented stiffly accurate SDIRK methods. If we assume such a method with 2 stages, we have Butcher tableau

$$\begin{array}{c|cc} c_1 & \gamma & \\ c_2 & c_2-\gamma & \\ \hline & b_1 & b_2 \end{array}$$

and coefficients have to (2.35) satisfy order conditions

$$\begin{aligned} b_1 + b_2 &= 1 \\ b_1\gamma + b_2 c_2 &= \frac{1}{2} \end{aligned}$$

to be second order. There are infinitely many solutions $b_1 = \frac{2c_2-1}{2(c_2-\gamma)}$, $b_2 = \frac{1-2\gamma}{2(c_2-\gamma)}$. For choice $\gamma \geq \frac{1}{4}$ is the method A-stable and if $\gamma = \frac{2\pm\sqrt{2}}{2}$ the method is also L-stable. For $c_2 = 1$ the method has following Butcher tableau

$$\begin{array}{c|cc} \frac{2-\sqrt{2}}{2} & \frac{2-\sqrt{2}}{2} & 0 \\ 1 & -\frac{\sqrt{2}}{2} & \frac{2-\sqrt{2}}{2} \\ \hline & -\frac{\sqrt{2}}{2} & \frac{2-\sqrt{2}}{2} \end{array}.$$

Popular method is L-stable stiffly accurate SDIRK method with embedded method constructed by J. R. Cash. The construction of this method and some implementation issues are shown in [22]

$$
\begin{array}{c|ccccc}
\frac{1}{4} & \frac{1}{4} & & & & \\
\frac{3}{4} & \frac{1}{2} & \frac{1}{4} & & & \\
\frac{11}{20} & \frac{17}{50} & -\frac{1}{25} & \frac{1}{4} & & \\
\frac{1}{2} & \frac{371}{1360} & -\frac{137}{2720} & \frac{15}{544} & \frac{1}{4} & \\
1 & \frac{25}{24} & -\frac{49}{48} & \frac{125}{16} & -\frac{85}{12} & \frac{1}{4} \\
\hline
 & \frac{25}{24} & -\frac{49}{48} & \frac{125}{16} & -\frac{85}{12} & \frac{1}{4} \\
\hline
 & \frac{59}{48} & -\frac{17}{96} & \frac{225}{32} & -\frac{85}{12} & 0
\end{array}
.
$$

ESDIRK

The last presented subgroup of DIRK methods is SDRIK methods with explicit first stage. We already mentioned that the first explicit stage decrease the number of equations which are need to solve. ESDIRK methods can have the stage order 2 whereas SDIRK method have stage order only 1.

As example present fourth order method with seven stages:

$$
\begin{array}{c|ccccccc}
0 & 0 & & & & & & \\
\frac{1}{2} & \frac{1}{6} & \frac{1}{6} & & & & & \\
\frac{2}{3} & \frac{1}{6} & \frac{1}{3} & \frac{1}{6} & & & & \\
1 & \frac{11}{24} & -\frac{1}{4} & \frac{5}{8} & \frac{1}{6} & & & \\
1 & \frac{11}{36} & -\frac{1}{6} & \frac{11}{12} & -\frac{2}{9} & \frac{1}{6} & & \\
1 & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & -\frac{1}{12} & \frac{1}{24} & \frac{1}{6} \\
\hline
 & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & -\frac{1}{12} & \frac{1}{24} & \frac{1}{6}
\end{array}
$$

and embedded method 3(2)

$$
\begin{array}{c|ccccc}
0 & 0 & & & & \\
\frac{9}{20} & \frac{9}{40} & \frac{9}{40} & & & \\
\frac{4}{5} & \frac{19}{72} & \frac{14}{45} & \frac{9}{40} & & \\
1 & \frac{3337}{11520} & \frac{233}{720} & \frac{207}{1280} & \frac{9}{40} & \\
1 & \frac{7415}{34776} & \frac{9920}{30429} & \frac{4845}{9016} & -\frac{5827}{19320} & \frac{9}{40} \\
\hline
 & \frac{7415}{34776} & \frac{9920}{30429} & \frac{4845}{9016} & -\frac{5827}{19320} & \frac{9}{40} \\
\hline
 & \frac{23705}{104328} & \frac{29720}{91287} & \frac{4225}{9016} & -\frac{69304987}{337732920} & \frac{42843}{233080}
\end{array}
.
$$

In [25] are constructed embedded pairs of methods 3(2), 4(3) and 5(4), where both methods are stiffly accurate. As example, the method 3(2) is given by Butcher tableau

0	0			
2γ	γ	γ		
1	$\frac{-\gamma^2+6\gamma-1}{4\gamma}$	$\frac{-2\gamma+1}{4\gamma}$	γ	
1	$\frac{6\gamma-1}{12\gamma}$	$\frac{-1}{(24\gamma-12)\gamma}$	$\frac{-6\gamma^2+6\gamma-1}{6\gamma-3}$	γ
	$\frac{-\gamma^2+6\gamma-1}{4\gamma}$	$\frac{-2\gamma+1}{4\gamma}$	γ	0
	$\frac{6\gamma-1}{12\gamma}$	$\frac{-1}{(24\gamma-12)\gamma}$	$\frac{-6\gamma^2+6\gamma-1}{6\gamma-3}$	γ

where γ is set according to output. For first output $\gamma = 0.2928932188$ and for the second one $\gamma = 0.435866521$.

4 | Stability

The last chapter deals with important property of numerical methods - stability. Actually, it is better to say important properties. There are more kinds of stability. A-stability was introduced as first.

4.1 A-Stability

The concept of A-stability was introduced by Dahlquist and originally it was considered for multi-step methods.

We assume linear test problem

$$\begin{aligned} y' &= \lambda y \\ y(0) &= 1 \end{aligned} \tag{4.1}$$

where $\lambda \in \mathbb{C}$ and $\mathrm{Re}(\lambda) < 0$. Its exact solution is

$$y(t) = e^{\lambda t}$$

and it approaches zero as $t \to \infty$. Naturally we want the numerical solution also exhibits this behaviour

$$y_n \to 0 \text{ as } n \to \infty \tag{4.2}$$

which we call stability condition.

If we apply a RK method to test equation (4.1) we obtain

$$y_{n+1} = R(\lambda h) y_n$$

where function $R(z)$ is determined by coefficients a_{ij}, b_i, and by induction

$$y_{n+1} = (R(\lambda h))^n y_0.$$

Then the stability condition (4.2) is equivalent to $|R(\lambda h)| < 1$.

Definiton 4.1. The function $R(z)$ is called the *stability function* of the method. It can be interpreted as the numerical solution after one step for (4.1) with $z = h\lambda$.

In the next we need to define the stability domain.

Definiton 4.2. The set

$$S = \{z \in \mathbb{C} : \ |R(z)| \leq 1\}$$

is called the *stability domain* of the method.

Finally we can define the A-stability.

Definiton 4.3. A method, whose stability domain satisfies

$$S \supset \mathbb{C}^- = \{z : \ Re(z) \leq 0\}$$

is called *A-stable.*

Explicit Runge–Kutta Methods

Now we examine the stability of explicit RK methods. We can start with the simplest RK method, explicit Euler method. If we applied it to (4.1), we obtain

$$R(z) = 1 + z$$

and from Definition 4.2 follows $R(z) = 1 + z \leq 1$ thus the stability domain is inside the unit circle with center at $[-1, 0]$.

If we assume 4 stage RK method applied to test problem (4.1), we obtain

$$\begin{aligned} k_1 &= \lambda y_n \\ k_2 &= \lambda(y_n + a_{21}hk_1) \\ k_3 &= \lambda(y_n + a_{31}hk_1 + a_{32}hk_2) \\ k_4 &= \lambda(y_n + a_{41}hk_1 + a_{42}hk_2 + a_{43}hk_3) \\ y_{n+1} &= y_n + h(b_1k_1 + b_2k_2 + b_3k_3 + b_4k_4) \end{aligned}$$

If we plug stages k_i into output y_{n+1}, multiply and put together terms with same power of (λh) we get

$$\begin{aligned} y_{n+1} = (a_{21}a_{32}a_{43}b_4(h\lambda)^4 + (a_{21}a_{32}b_3 + a_{21}a_{42}b_4 + a_{31}a_{43}b_4 + a_{32}a_{43}b_4)(h\lambda)^3 \\ + (a_{21}b_2 + a_{31}b_3 + a_{32}b_3 + a_{41}b_4 + a_{42}b_4 + a_{43}b_4)(h\lambda)^2 + (b_1 + b_2 + b_3 + b_4)h\lambda + 1)y_n \end{aligned}$$

Now we apply order conditions, namely relations (2.36a), (2.36b), (2.36d), (2.36h) and condition (1.3) and we obtain stability function for RK method $s = p = 4$

$$R(z) = 1 + z + \frac{1}{2}z^2 + \frac{1}{6}z^3 + \frac{1}{24}z^4.$$

The stability function for explicit methods is summarized in following theorem.

Theorem 4.1. *If the explicit RK method is of order p, than its stability function is*

$$R(z) = 1 + z + \frac{1}{2!}z^2 + ... + \frac{1}{p!}z^p + \mathcal{O}(z^{p+1}).$$

Proof. The exact solution of (4.1) is e^z and therefore the numerical solution $y_1 = R(z)$ must satisfy

$$e^z - R(z) = \mathcal{O}(h^{p+1}) = \mathcal{O}(z^{p+1}).$$

□

The consequence of this theorem is that all methods with same order such that $s = p$ have the same stability function. The boundaries of stability domain are shown in Figure 4.1.

As examples of RK methods with more stages we present Runge-Kuta-Fehlberg 4(5) method of order 4, mentioned in Section 2.8, with stability function

$$R(z) = \frac{z^5}{104} + \frac{z^4}{24} + \frac{z^3}{6} + \frac{z^2}{2} + z + 1.$$

In the same section mentioned, Dormand-Prince 5(4) method, for which the RK method use for computing solution is fifth order method:

$$R(z) = \frac{z^6}{600} + \frac{z^5}{120} + \frac{z^4}{24} + \frac{z^3}{6} + \frac{z^2}{2} + z + 1$$

and Runge-Kuta-Fehlberg 7(8), its Butcher tableau can be found in [17], of order 7

$$R(z) = -\frac{65z^{11}}{1504935936} + \frac{13z^{10}}{250822656} + \frac{4453z^9}{1881169920} + \frac{269z^8}{11612160} + \frac{z^7}{5040} + \frac{z^6}{720} + \frac{z^5}{120} + \frac{z^4}{24} + \frac{z^3}{6} + \frac{z^2}{2} + z + 1$$

Stability regions of those methods can be found in Figure 4.1.

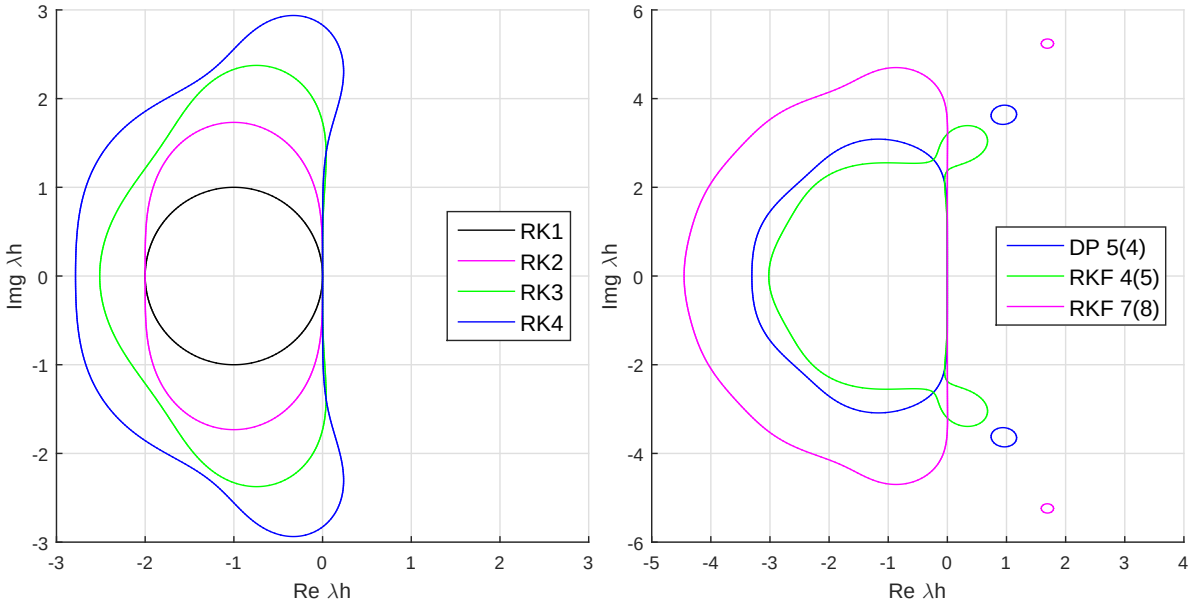


Figure 4.1: Boundaries of stability domains for explicit methods (stable within boundaries)

As we can see, the stability domain is bigger as order of method increase. The explicit RK methods are not A-stable. Therefore the step size is limited and the methods are unsuitable for stiff problems.

Implicit Runge–Kutta Methods

For implicit methods we start with implicit Euler method. Applied to test problem (4.1),

$$y_{n+1} = y_n + h\lambda y_{n+1}$$

hence its stability function is $R(z) = \frac{1}{1-z}$ and its stability domain is the outside of the unit cycle with center at $[1, 0]$.

If we assume general implicit RK method in form

$$g_i = y_n + h\sum_{j=1}^{s} a_{ij} f(t_n + c_j h, g_j), \quad i = 1, \ldots s$$

$$y_{n+1} = y_n + h\sum_{j=1}^{s} b_j f(t_n + c_j h, g_j)$$

then if we apply it to the test problem (4.1), we get

$$g = y_n \mathbf{e} + h\lambda \mathbf{A} g \tag{4.3a}$$

$$y_{n+1} = y_n + h\lambda \mathbf{b}^T g, \tag{4.3b}$$

where g denotes the vector $g = (g_1, g_2, ..., g_s)$ and $\mathbf{e} = (1, 1, ..., 1)^T$ is the s-dimensional vector of all ones. From (4.3a) we have linear system with solution

$$g = (\mathbf{I} - h\lambda\mathbf{A})^{-1} y_n,$$

where $\mathbf{I}$ denotes the unit matrix of dimension s. We plug it into (4.3b) and obtain

$$y_{n+1} = y_n + h\lambda\mathbf{b}^T(\mathbf{I} - h\lambda\mathbf{A})^{-1}y_n$$

then easily, the stability function is given by

$$R(z) = 1 + z\mathbf{b}^T(\mathbf{I} - z\mathbf{A})^{-1}\mathbf{e}. \tag{4.4}$$

Propositon 4.2. *The stability function (4.4) satisfies*

$$R(z) = \frac{\det(\mathbf{I} - z\mathbf{A} + z\mathbf{e}\mathbf{b}^T)}{\det(\mathbf{I} - z\mathbf{A})}. \tag{4.5}$$

Proof. See [22]. □

The stability function of a RK method can be written as the ratio of two polynomials

$$R(z) = \frac{N(z)}{D(z)},$$

and if we define the E-polynomial as

$$E(y) = D(iy)D(-iy) - N(iy)N(-iy)$$

then we can use another criterion for A-stability.

Theorem 4.3. *A Runge–Kutta method with stability function $R(z) = \frac{N(z)}{D(z)}$ is A-stable if and only if*

1. *all poles of R are in the real positive half-plane and*
2. *$E(y) \leq 0$, for all real y.*

Proof. See [7]. □

All Gauss, Radau IA, Radau IIA, Lobatto IIIA, Lobatto IIIB and Lobatto IIIC methods are A-stable [7, 22].

For DIRK methods, from (4.5) the stability function become

$$R(z) = \frac{P(z)}{(1 - a_{11}z)(1 - a_{22}z)...(1 - a_{ss}z)} \tag{4.6}$$

because the determinant of a triangular matrix is the product of its diagonal entries. The numerator $P(z)$ is a polynomial of degree s at most.

The stability function of SDIRK methods is

$$R(z) = \frac{P(z)}{(1 - \gamma z)^s}. \tag{4.7}$$

It can be found bounds on γ for which the method is A-stable [23]. In Table 4.1, we present this bounds for methods up to four stages.

s	p	A-stable
2	2	$\frac{1}{4} \leq \gamma \leq \infty$
2	3	$\gamma = \frac{3+\sqrt{3}}{6}$
3	3	$\frac{1}{3} \leq 1.068579021$
3	4	$\gamma = 1.068579021$
4	4	$0.39433756 \leq \gamma \leq 1.28057976$
4	5	–

Table 4.1: Bounds on γ for SDIRK methods

4.2 L-Stability

The A-stability is not enough for solving stiff equations. It is demanded L-stability, which is sometimes called strong A-stability or stiff A-stability.

Definiton 4.4. A method is called L-stable if it is A-stable and if in addition

$$\lim_{z\to\infty} R(z) = 0.$$

Some of the L-stable method can be identified by the following theorem.

Theorem 4.4. *If and implicit Runge-Kutta method with non-singular* $\mathbf{A}$ *satisfies one of the following conditions:*

$$a_{sj} = b_j, \quad j = 1, ..., s, \tag{4.8}$$
$$a_{i1} = b_1, \quad i = 1, ...s, \tag{4.9}$$

then $R(\infty) = 0$. *This makes A-stable methods L-stable.*

Proof. By (4.4)

$$R(\infty) = 1 - \mathbf{b}^T\mathbf{A}^{-1}\mathbf{e},$$

where $\mathbf{e} = (1, 1, ..., 1)^T$. The condition (4.8) can be written as $\mathbf{A}^T\mathbf{e}_s$, $\mathbf{e}_s = (0, ..., 0, 1)^T$. Therefore $R(\infty) = 1 - \mathbf{e}_s^T\mathbf{e} = 1 - 1 = 0$.

The condition (4.9) means that $\mathbf{A}^T\mathbf{e}_1 = \mathbf{e}b_1$. Therefore $R(\infty) = 1 - \mathbf{b}^T\mathbf{A}^{-1}\mathbf{A}\mathbf{e}_1\frac{1}{b_1} = 1 - 1 = 0$. □

Methods satisfying (4.8) are called stiffly accurate and they important for solution of singularly perturbed problems and differential-algebraic equations. According to the above theorem, the L-stable fully implicit methods are Radau IA, Radau IIA and Lobattto IIIC methods. Among the diagonal RK methods, as we already mentioned, the 2-stage second order method and embedded method with 5 stages are L-stable.

Conclusion

The goals of this thesis were to give the overview of Runge–Kutta methods, to derive order conditions, to introduce methods suitable for adaptive step size strategy and test them and also to discus the stability of methods. In the first chapter, the Runge–Kutta methods were introduced and divided into groups according to the structure of matrix $\mathbf{A}$ from the Butcher tableau. As first, the analysis of Euler method, the oldest numerical method for solving differential equation, was done. There is also defined stiff problem.

The second chapter deals with explicit Runge–Kutta methods. The analysis of the explicit Euler method is done. The way to improve it was sketched and the order conditions for the second and third order are derived from the Taylor expansion. In the Section 2.3, the theory of rooted trees is introduced for application to order conditions of Runge–Kutta methods. Then the methods of fourth order are derived. It is mentioned the highest possible order for which $s = p$ is four. To obtain more accurate methods we need more stages.

In Section 2.6, the modified Runge–Kutta based on the various kinds of means are presented. Namely, geometrical, harmonic, contra-harmonic, heronian, centroidal, square root and arithmetic means. The last one is the classical 4-stage Runge–Kutta methods. All those methods are referred as fourth order methods. The analysis of order was done for methods based on contra-harmonic and centroidal mean and unfortunately, different results were obtained. It turned out that the methods are fourth order only for scalar autonomous equation. In any other case they are only second order.

To confirm the theoretical results, all modified methods were tested numerically on four problems – autonomous and non-autonomous scalar equations, autonomous and non-autonomous systems. In case of autonomous scalar equations, all methods performed solution with similar global errors. The numerical order was determined to be four. For non-autonomous equation and systems, all methods, except the one based on arithmetic mean, performed second order behaviour. In Problem 3, the interesting phenomenon occurred. The jumps in the approximated solution appeared around point, where the derivative of solution is zero. This rapid increase of error in solution can be explained by the division two numbers close to zero. This is probably the reason why was impossible to determine the numerical order. The contraction of step sometimes caused the bigger global error. Therefore the modified Runge–Kutta methods are unsuitable for the most problems.

As the next, two approaches of error estimation were presented. Today the embedded methods are classical way how to approximate the local error for the step size selection. They combine two methods with same stages but have different coefficients $b_i, i = 1, ..., s$. The difference of outputs is used as error estimation. As alternative, the strategy of combination of two methods of same order was presented. This approach combines the method based on arithmetic mean and any other mean. In this thesis were discussed combination with methods based on contra-harmonic and centroidal mean. In the light of previous results of this work, the error estimation provided by authors of methods is justified only for autonomous scalar problem. In other cases, we use the classical estimation, difference of the outputs. The local error is overestimated, mainly around points where the derivative of solution is zero.

The third chapter presents implicit Runge–Kutta methods. First, the collocation methods are derived. Then the fully implicit methods are derived with the help of Butcher's simplifying assumptions $B(p), C(q)$ and $D(r)$. Methods based on Gauss, Radau and Lobatto quadrature formulas are presented. The embedded methods with same number of stages were derived. As alternative, the Hairer's embedded method was presented. He embedded four stage method into Radau IIA method with three stages. Then the methods were compared on two test problems. Methods with same number of stages mainly overestimated the error. The exception was Lobatto IIIC pair of method with orders (2,1) and (4,2). The 3-stage methods performed results comparable with Hairer's method. In the end of the third chapter, the diagonally implicit methods were presented.

The fourth chapter discuses the stability. It is shown that the explicit methods have small domain of stability, therefore the restriction to length of time step is applied and they are not suitable for stiff problems. On the other hand, the fully implicit methods as Gauss, Radau IA and IIA and Lobatto III A, B, C, are A-stable although it is not enough. Therefore, the concept of L-stability is presented.

This work could be extended in many ways. As diagonal methods have big potential to solving the stiff systems, the deeper theoretical research could be presented as well as the numerical experiments. Then method derived from Runge–Kutta methods, as implicit-explicit scheme, could be presented. More kinds of stability could be discussed, for example, the concepts for non-linear problems as AN-stability or B-stability. Other option is S-stability which is required for strongly stiff problems as singularly perturbed problems and differential-algebraic equations.

Bibliography

[1] ALEXANDER, R. Diagonally Implicit Runge—Kutta Methods for Stiff O.D.E.-s. *SIAM Journal on Numerical Analysis*. 1977, **14**(6), 1006-1021. DOI: 10.1137/0714068. ISSN 0036-1429.

[2] ATKINSON, K. E., W. HAN a D. STEWART. *Numerical solution of ordinary differential equations*. Hoboken, N.J.: Wiley, 2009. : Unnumbered). ISBN 978-0-470-04294-6.

[3] BOGACKI, P. a L.F. SHAMPINE. A 3(2) pair of Runge - Kutta formulas. *Applied Mathematics Letters*. 1989, **2**(4), 321-325. DOI: 10.1016/0893-9659(89)90079-7. ISSN 08939659.

[4] BUTCHER, J. C. On the attainable order of Runge-Kutta methods. *Mathematics of Computation*. 1965, **19**(91), 408-408. DOI: 10.1090/S0025-5718-1965-0179943-X. ISSN 0025-5718.

[5] BUTCHER, J. C. Integration Processes Based on Radau Quadrature Formulas. *Mathematics of Computation*. 1964, **18**(86), 233-244. DOI: 10.2307/2003297. ISSN 00255718.

[6] BUTCHER, J. C. The non-existence of ten stage eighth order explicit Runge-Kutta methods. *BIT*. 1985, **25**(3), 521-540. DOI: 10.1007/BF01935372. ISSN 0006-3835.

[7] BUTCHER, J. C. *Numerical methods for ordinary differential equations*. 2nd edition. Hoboken, NJ: Wiley, 2008. ISBN 978-0-470-72335-7.

[8] BUTCHER, J. C. Implicit Runge-Kutta Processes. *Mathematics of Computation*. 1964, **18**(85), 50-64. DOI: 10.2307/2003405. ISSN 00255718.

[9] BUTCHER, J.C. A history of Runge-Kutta methods. *Applied Numerical Mathematics*. 1996, **20**(3), 247-260. DOI: 10.1016/0168-9274(95)00108-5. ISSN 01689274.

[10] CASH, J. R. a A. H. KARP. A variable order Runge-Kutta method for initial value problems with rapidly varying right-hand sides. *ACM Transactions on Mathematical Software*. 1990, **16**(3), 201-222. DOI: 10.1145/79505.79507. ISSN 00983500.

[11] DE SWART, J. J. B. a G. SÖDERLIND. On the construction of error estimators for implicit Runge-Kutta methods. *Journal of Computational and Applied Mathematics*. 1997, **86**(2), 347-358. DOI: 10.1016/S0377-0427(97)00166-0. ISSN 03770427.

[12] DORMAND, J.R. a P.J. PRINCE. A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*. 1980, **6**(1), 19-26. DOI: 10.1016/0771-050X(80)90013-3. ISSN 03770427.

[13] EVANS, D. J. a N. YAACOB. A fourth order runge-kutta method based on the heronian mean formula. *International Journal of Computer Mathematics*. 1995, **58**(1-2), 103-115. DOI: 10.1080/00207169508804437. ISSN 0020-7160.

[14] EVANS, D. J. a A. R. YAAKUB. A Fifth Order Runge-Kutta RK(5, 5) Method With Error Control. *International Journal of Computer Mathematics*. 2002, **79**(11), 1179-1185. DOI: 10.1080/00207160213937. ISSN 0020-7160.

[15] EVANS, D. J. a A. R. YAAKUB. A new fourth order runge-kutta formula based on the contra-harmonic (CoM) mean. *International Journal of Computer Mathematics*. 1995, **57**(3-4), 249-256. DOI: 10.1080/00207169508804428. ISSN 0020-7160.

[16] EVANS, D. J. a A. R. YAAKUB. A new Runge Kutta RK(4, 4) method. *International Journal of Computer Mathematics*. 1995, **58**(3-4), 169-187. DOI: 10.1080/00207169508804442. ISSN 0020-7160.

[17] FEHLBERG, E. *Classical fifth-, sixth-, seventh-, and eighth-order Runge-Kutta formulas with stepsize control*. Washington: National Aeronautics and Space Administration, 1968.

[18] FEHLBERG, E. *Low-order Classical Runge-Kutta Formulas with Stepsize Control and Their Application to Some Heat Transfer Problems*. National Aeronautics and Space Administation, 1969.

[19] GUSTAFSSON, K. Control-theoretic techniques for stepsize selection in implicit Runge-Kutta methods. *ACM Transactions on Mathematical Software*. 1994, **20**(4), 496-517. DOI: 10.1145/198429.198437. ISSN 00983500.

[20] HAIRER, E. A Runge-Kutta Method of Order 10. *IMA Journal of Applied Mathematics*. 1978, **21**(1), 47-59. DOI: 10.1093/imamat/21.1.47. ISSN 0272-4960.

[21] HAIRER, E., S. P. NØRSETT a G. WANNER. *Solving ordinary differential equations I*. 2nd rev. ed. New York: Springer-Verlag, 1996. ISBN 978-3-540-60452-5.

[22] HAIRER, E. a G. WANNER. *Solving ordinary differential equations II*. 2nd rev. ed. Heidelberg: Springer, 2010. ISBN 978-364-2052-217.

[23] KENNEDY, C. A. a M. H. CARPENTER. *Diagonally Implicit Runge-Kutta Methods for Ordinary Differential Equations*. National Aeronautics and Space Administration, 2016.

[24] KUTTA, W. *Beitrag zur naherungsweisen Integration totaler Differentialgleichungen*. B. G. Teubner Verlag, 1901.

[25] KVÆRNØ, A. Singly Diagonally Implicit Runge—Kutta Methods with an Explicit First Stage. *BIT Numerical Mathematics*. 2004, **44**(3), 489-502. DOI: 10.1023/B:BITN.0000046811.70614.38. ISSN 0006-3835.

[26] LEVEQUE, R. J. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2007. ISBN 978-0-89871-629-0.

[27] LOTKIN, M. On the accuracy of Runge-Kutta-s method. *Mathematics of Computation*. 1951, **5**(35), 128-128. DOI: 10.1090/S0025-5718-1951-0043566-3. ISSN 0025-5718.

[28] MURUGESAN, K., D. P. DHAYABARAN, E. C. H. AMIRTHARAJ a D. J. EVANS. A Fourth Order Embedded Runge-Kutta RKACeM(4,4) Method Based on Arithmetic and Centroidal Means with Error Control. *International Journal of Computer Mathematics*. 2010, **79**(2), 247-269. DOI: 10.1080/00207160211925. ISSN 0020-7160.

[29] MURUGESAN, K., D. PAUL DHAYABARAN, E.C. HENRY AMIRTHARAJ a D. J. EVANS. A comparison of extended runge-kutta formulae based on variety of means to solve system of ivps. *International Journal of Computer Mathematics*. 2001, **78**(2), 225-252. DOI: 10.1080/00207160108805108. ISSN 0020-7160.

[30] RANG, J. *Adaptive timestep control for fully implicit Runge–Kutta methods of higer order*. Technische Universität, 2014-03.

[31] SANUGI, B. B. *New numerical strategies for initial value type ordinary differential equations*. Leicester, 1986. A doctoral thesis. Loughborough University of Technology.

[32] SHAMPINE, L. F. a L. S. BACA. Error estimators for stiff differential equations. *Journal of Computational and Applied Mathematics*. 1984, **11**(2), 197-207. DOI: 10.1016/0377-0427(84)90020-7. ISSN 03770427.

[33] VERNER, J. H. Explicit Runge—Kutta Methods with Estimates of the Local Truncation Error. *SIAM Journal on Numerical Analysis*. 1978, **15**(4), 772-790. DOI: 10.1137/0715051. ISSN 0036-1429.

[34] WAZWAZ, A. M. A modified third order Runge-Kutta method. *Applied Mathematics Letters*. 1990, **3**(3), 123-125. DOI: 10.1016/0893-9659(90)90154-4. ISSN 08939659.

[35] WAZWAZ, A. M. Modified numerical methods based on arithmetic and geometric means. *Applied Mathematics Letters*. 1991, **4**(3), 49-52. DOI: 10.1016/0893-9659(91)90034-S. ISSN 08939659.

[36] WAZWAZ, A. M. A comparison of modified runge-kutta formulas based on a variety of means. *International Journal of Computer Mathematics*. 1994, **50**(1-2), 105-112. DOI: 10.1080/00207169408804245. ISSN 0020-7160.

Electronic Appendix Index

- `ComparisonModifiedMethod.m` – script for numerical experiments in Section 2.7
- `ExplicitEmbeddedMethods.m` – script for numerical experiments in Section 2.10
- `ImplicitEmbeddedMethods.m` – script for numerical experiments in Section 3.5
- `ascontroler.m` – function, returns solution obtained by chosen explicit embedded method
- `ButcherTableauE.m` – function, returns extended Butcher tableau for explicit embedded methods
- `ButcherTableauI.m` – function, returns extended Butcher tableau for implicit embedded methods
- `EmIRK.m` – function, returns solution obtained by chosen implicit embedded method
- `jacobian.m` – function, returns Jacobian $f_y(t, y)$
- `modifiedRK.m` – function, returns solution obtained by chosen modified method with constant step size
- `RadauIIA53.m` – function, returns solution obtained by Hairer's embedded method
- `RKAMCeMstep.m` – function, returns one step of solution obtained by RKACeM method
- `RKAMCoMstep.m` – function, returns one step of solution obtained by RKACoM method
- `RKstep.m` – function, returns one step of solution obtained explicit embedded method without FSAL property
- `RKstepFSAL.m` – function, returns one step of solution obtained explicit embedded method with FSAL property
- `startingstepsize.m` – function, returns starting stepsize
- `testproblems.m` – function, returns input data and exact solution of test problems