

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

STABILITY AND CONVERGENCE OF NUMERICAL COMPUTATIONS

DISERTAČNÍ PRÁCE

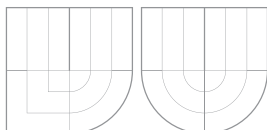
PHD THESIS

AUTOR PRÁCE

AUTHOR

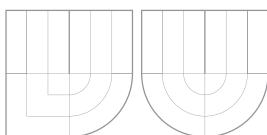
Ing. PAVLA SEHNALOVÁ

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ



FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

STABILITA A KONVERGENCE NUMERICKÝCH VÝPOČTŮ

STABILITY AND CONVERGENCE OF NUMERICAL COMPUTATIONS

DISERTAČNÍ PRÁCE

PHD THESIS

AUTOR PRÁCE

AUTHOR

Ing. PAVLA SEHNALOVÁ

VEDOUCÍ PRÁCE

SUPERVISOR

doc. Ing. JIŘÍ KUNOVSKÝ, CSc.

BRNO 2011

Abstrakt

Tato disertační práce se zabývá analýzou stability a konvergence klasických numerických metod pro řešení obyčejných diferenciálních rovnic. Jsou představeny klasické jednokrokové metody jako je Eulerova metoda, Runge-Kuttovy metody a nepříliš známá, ale rychlá a přesná metoda Taylorovy řady. V práci uvažujeme zobecnění jednokrokových metod do vícekových metod jako jsou Adamsovy metody a jejich implementaci ve dvojicích prediktor-korektor. Dále uvádíme generalizaci do vícekových metod vyšších derivací, jako jsou např. Obreshkovovy metody. Dvojice prediktor-korektor jsou často implementovány v kombinacích módů, v práci uvažujeme tzv. módy PEC a PECE. Hlavním cílem a přínosem této práce je nová metoda čtvrtého řádu, která se skládá z dvoukrokového prediktoru a jednokrokového korektoru, jejichž formule využívají druhých derivací. V práci je diskutována Nordsieckova reprezentace, algoritmus pro výběr proměnlivého integračního kroku nebo odhad lokálních a globálních chyb. Navržený přístup je vhodně upraven pro použití proměnlivého integračního kroku s přístupem vyšších derivací. Uvádíme srovnání s klasickými metodami a provedené experimenty pro lineární a nelineární problémy.

Abstract

The aim of this thesis is to analyze the stability and convergence of fundamental numerical methods for solving ordinary differential equations. These include one-step methods such as the classical Euler method, Runge–Kutta methods and the less well known but fast and accurate Taylor series method. We also consider the generalization to multistep methods such as Adams methods and their implementation as predictor–corrector pairs. Furthermore we consider the generalization to multiderivative methods such as Obreshkov method. There is always a choice in predictor-corrector pairs of the so-called mode of the method and in this thesis both PEC and PECE modes are considered. The main goal and the new contribution of the thesis is the use of a special fourth order method consisting of a two-step predictor followed by an one-step corrector, each using second derivative formulae. The mathematical background of historical developments of Nordsieck representation, the algorithm of choosing a variable stepsize or an error estimation are discussed. The current approach adapts well to the multiderivative situation in variable stepsize formulations. Experiments for linear and non-linear problems and the comparison with classical methods are presented.

Klíčová slova

Numerické metody, obyčejné diferenciální rovnice, stabilita, konvergence, jednokrokové metody, Runge–Kutta metody, metoda Taylorovy řady, vícekrokové metody, lineární vícekrokové metody, Adamsovy metody, metody prediktor–korektor, Obreshkovovy formule, Nordsieckova reprezentace, proměnlivý integrační krok.

Keywords

Numerical methods, ordinary differential equations, stability, convergence, one-step methods, Runge–Kutta methods, Taylor series method, multistep methods, linear multistep methods, Adams methods, predictor–corrector pairs, Obreshkov formulae, Nordsieck representation, variable stepsize.

Citace

Pavla Sehnalová: Stability and convergence of numerical computations, disertační práce, Brno, FIT VUT v Brně, 2011

Acknowledgement

I would like to thank my supervisor Jiří for his guidance, advices and given opportunity for traveling abroad and meeting new people from the field of computer science and mathematics. I also would like to thank my mum for her motivation and beloved support and my partner Zbyněk for his patience and understanding. Many thanks go to my friends John and Annie for their help and mathematical advice.

The work presented in this thesis was supported by the Czech Ministry of Education (project MSM 0021630528) and the internal BUT project FIT-10-1.

© Pavla Sehnalová, 2011.

Contents

1	Introduction	3
1.1	Targets of the thesis	4
1.2	Structure of the thesis	4
2	Ordinary differential equations and one-step methods	6
2.1	Ordinary differential equation	6
2.2	Euler method	16
2.3	Taylor series	23
2.4	Runge–Kutta methods	31
2.5	Taylor series method	44
3	Multistep methods	50
3.1	Improved Euler method	50
3.2	Linear multistep methods	51
3.3	Predictor–corrector methods	56
4	Multiderivative multistep methods	61
4.1	General linear methods	61
4.2	Obreshkov quadrature formulae	63
4.3	Nordsieck representation	63
5	Two-derivative multistep method	66
5.1	Starting method	72
5.2	Stepsize control	75
5.3	Order of the method	81
5.4	Function evaluations	85
5.5	PEC and PECE modes	87
5.6	Stability analysis	90
6	Conclusions	92

A List of publications	103
B TKSL/C code	104
C Starting method	106
D Results of circle test	107

Chapter 1

Introduction

There exists computational systems these days, which can be used to solve huge and time-consuming scientific calculations. Universal computational systems and equipments solve these kinds of special algorithms and problems in less shorter time than in former centuries. One of these problems is the numerical solution of differential equations.

Differential equations have a very long history; they are old as a differential calculus (Newton 1691). Many mathematicians or researchers have tried to calculate them, to discover new methods for solving them and to improve methods' properties since then. Classic application of differential equations is found in many areas of science and technology. They can be used for modelling of physical, technical or biological processes such as in the study of an electric circuit consisting of a resistor, an inductor and a capacitor driven by an electromotive force, in gravitational equilibrium of a star, chemical reactions kinetic, in the psychology, in models of the learning of a task involves the equation, in vibrating strings and propagation of waves, etc. [57, 83]. Movement of celestial bodies, the shape of a ship's wake, stress and lift action on aircraft wings, spread of epidemic through a large population, percolation of crude oil in semi-permeable rock, nuclear processes in the core of stars, transmission of electric pulses down a nerve fibre, the fickle behaviour of stock markets, tear and wear of turbine blades – to all intents and purposes the list is infinite, limited merely by our imagination [66]. Main questions of modern technology are how to increase the accuracy of calculations considering short computational time, how to decrease necessary mathematical operations and all these questions have many aspects and criterion, which we need to explore to get the suitable answer.

My field of study is focused on modelling and simulation of various problem. Each simulation system includes different type of numerical computations. To summarize numerical methods is very demanding task in terms of extensiveness. Therefore the thesis is focused on non-stiff problems described by ordinary differential equations and their solutions using numerical methods.

There have been written many great books about numerical methods and a lot of theory have been discovered, still many mathematicians and engineers research to get better and faster results for problems which can be described by differential equations. In this thesis I would like to describe some mathematic background and theory for ordinary differential equations solved by one-step and multistep methods and pointed out ideas of solving systems of differential equations by Taylor series method [75]. This thesis is focused on essential research and contains only a fragmentary amount of the mathematical background.

1.1 Targets of the thesis

Targets of this thesis are divided into two main goals. The first main goal is an original approach of the predictor–corrector method in Obreshkov quadrature formulae; the new two-derivative multistep numerical method. This goal also includes the implementation of the new method with variable stepsize and the implementation of the new method in modes PEC and PECE.

The second goal is to investigate the convergence and stability analysis for the new method with constant stepsize for various problems as well as to investigate and to compare the convergence and stability analysis for selected numerical methods. The analysis of the new method obtains the variable stepsize analysis.

1.2 Structure of the thesis

The chapter 2 presents the mathematical background of ordinary differential equations and numerical methods. Required fundamentals and basics are presented to introduce the notation and chosen problems. The analytical solution is discussed and the example of analytical solution is given by solving the simple electrical circuit with a resistor, a capacitor and a coil. Then the focus is transferred to numerical solutions and one-step methods. The idea of the generalization of Euler method is shown and through Runge-Kutta methods and Taylor series method we describe interesting notations and approaches.

The attention is dedicated to multistep methods such as linear multistep methods and predictor–corrector methods from the stability point of view as well as the implementation point of view in the chapter 3.

The goal of the chapter 4 is to briefly present multiderivative multistep methods. The predictor–corrector formulae are combined in nontraditional way in the form of Obreshkov quadrature formulae using variable stepsize. The approach of variable stepsize was presented by Nordsieck and it is described in this chapter.

The chapter 5 is dedicated to the new contribution, which is closely linked to my stay at

University of Auckland, Department of Mathematics during the academic year 2008/2009. Many aspects of the thesis were discussed with the mathematician J. C. Butcher, Emeritus Professor at University of Auckland. In this chapter the new two-derivative multistep method is introduced. We begin with a discussion of some important tasks of error estimation and choosing the stepsize. Then the results of test problems are compared with results obtained by classical methods.

Summary and proposed future work is described in the final chapter 6.

Chapter 2

Ordinary differential equations and one-step methods

The study of differential equations is a wide field in mathematics, physics and other disciplines. It is possible to divide differential equations in many groups of different types. Two main groups are ordinary differential equations and partial differential equations. Both of them can be classified as linear and nonlinear. In this thesis, we will assume and solve ordinary differential equations and systems of ordinary differential equations.

2.1 Ordinary differential equation

Ordinary differential equation (ODE) of first order obtains a single independent variable and one or more its derivatives with respect to that variable [8]. The equation is given in the form

$$y'(x) = f(x, y(x)), \quad (2.1)$$

$$y(x_0) = y_0, \quad (2.2)$$

where $y'(x) = \frac{dy}{dx}$, x is independent variable, y is dependent variable. A function $y(x)$ is called a solution of equation (2.1) and the initial value (2.2) is given.

A second order ODE for y is, under mild assumptions for (2.1) together with (2.2), given in the form

$$y'' = f(x, y, y'), \quad (2.3)$$

with two free parameters which represent two uniquely determined initial values

$$y(x_0) = y_0, \quad y'(x_0) = y'_0.$$

Generally, an order n ODE in x with $y^{(n)}$ has the explicit form

$$y^{(n)} = f(x, y, y', \dots, y^{(n-1)}), \quad (2.4)$$

there is a unique solution with n initial values

$$y(x_0) = y_0, y'(x_0) = y'_0, \dots, y^{(n-1)}(x_0) = y_0^{(n-1)}. \quad (2.5)$$

Order of differential equation is an order of the highest-order derivatives presented in the differential equation [98].

To solve the ordinary differential equations we need to ask how we can solve them. We are also interested in a question if a differential equation has more than one solution. Here we talk about the uniqueness of the solution. If it has at least one solution we need to find a solution which satisfies particular conditions. The answer testifies about the existence of the solution. And we try to discover which method should we use for solving the differential equation to get the accurate result in a suitable time. There are other fundamentals which need to be presented. But only in a way to understand the described methods and generalizations. Generally, the mathematical background is very extensive and described in many other books.

Explicit solution

Explicit solution (2.4) is called a function $\Phi(x)$ when substituted for y in equation (2.4) satisfies the equation for all x in the interval I [28].

Sometimes the explicit solution does not suit the purpose of a differential equation because of its properties. Then we need to settle a solution that is defined implicitly.

Implicit solution

A relation (2.6) is said to be an implicit solution to equation (2.4) on the interval I if it defines one or more explicit solutions on interval I [28].

$$f(x, y, y', \dots, y^{(n)}) = 0 \quad (2.6)$$

Analytical solution

Many of ordinary differential equations of arbitrary order can be solved *analytically*. In the most of cases it is very complicated and time-consuming problem.

Generally, it is possible to determine the analytical solution of the differential equation as a composition of particular $y_p(x)$ and homogeneous $y_h(x)$ solutions

$$y(x) = y_p(x) + y_h(x).$$

To present the possibilities of analytical solution, we now introduce a n -th order linear ordinary differential equation which has the general form of

$$a_n(x)y^{(n)}(x) + a_{n-1}(x)y^{(n-1)}(x) + \dots + a_1(x)y(x) = b(x), \quad (2.7)$$

where $a_n(x)$, $a_{n-1}(x), \dots, a_0(x)$ are all functions of x . Assuming that $a_n(x) \neq 0$ and dividing the previous equation by this $a_n(x)$, we can rewrite the equation in the form

$$y^{(n)}(x) + p_{n-1}(x)y^{(n-1)}(x) + \dots + p_1(x)y(x) = g(x), \quad (2.8)$$

where the functions $p_{n-1}(x), \dots, p_1(x), g(x)$ are continuous on interval I . If $g(x) = 0$, the equation is called homogeneous. Otherwise, it is a non-homogeneous differential equation. For the corresponding homogeneous equation (2.8) to (2.7) there exists a set of n linearly independent solutions y_1, y_2, \dots, y_n on I .

Such functions form a fundamental solution set and every solution for (2.8) can be written as a linear combination according to a superposition principle [83]

$$y(x) = c_1y_1(x) + \dots + c_ny_n(x), \quad (2.9)$$

where c_1, c_2, \dots, c_n are constants. The linear independence of solutions to (2.8) is equivalent to the non-vanishing on interval I of the Wroskian

$$W(y_1, \dots, y_n)(x) = \det \begin{pmatrix} y_1(x) & \dots & y_n(x) \\ y_1'(x) & \dots & y_n'(x) \\ \vdots & & \vdots \\ y_1^{(n-1)}(x) & \dots & y_n^{(n-1)}(x) \end{pmatrix}.$$

When homogeneous equations in the form of (2.8) have (real) constant coefficients then the problem of determining a fundamental solution set is reduced to the algebraic problem of solving the auxiliary equation called the *characteristic equation*

$$\lambda^n + p_{n-1}(x)\lambda^{n-1} + \dots + p_2(x)\lambda + p_1(x) = 0.$$

Let $A(x)$ be a $n \times n$ constant matrix of characteristic equations' coefficients. The eigenvalues of A are numbers λ for which

$$(A - \lambda I)\mathbf{v} = 0 \quad (2.10)$$

has at least one nontrivial solution $\mathbf{v}(x)$. The corresponding nontrivial solutions for linear systems are called the *eigenvectors* and have different natures: simple and multiple, real and complex [28].

Let generalize it for a linear homogeneous system of n algebraic equations. The system has a nontrivial solution if and only if the determinant of its coefficients is zero. Hence a necessary and sufficient condition for (2.10) to have a nontrivial solution is the characteristic equation of A such that

$$\det(A - \lambda I) = 0. \quad (2.11)$$

Solutions for homogeneous equations can be divided into groups, see three important special cases:

1. Eigenvalues are real and equal ($\lambda_1 = \lambda_2 = \lambda_n = \lambda$), we have only one eigenvalue and only general solution of the system

$$P(x)e^{\lambda x},$$

where P is an arbitrary polynomial of degree $n - 1$.

2. Eigenvalues are real and distinct ($\lambda_1 \neq \lambda_2 \neq \lambda_n$)

$$c_1 e^{\lambda_1 x}, c_2 e^{\lambda_2 x}, \dots, c_n e^{\lambda_n x}.$$

Since the exponents are negative, both will progressively decay with time. The rates of decay will be given by the respective eigenvalues and the state will move towards the eigenvector associated with the larger eigenvalue and finally converge into the equilibrium point.

If the eigenvalues are real and negative, the system is stable in the sense that any perturbation from an equilibrium points decays exponentially and the system settles back to the equilibrium point. If the real parts of the eigenvalues are positive, any deviation from the equilibrium point grows exponentially and the system is unstable.

If one eigenvalue is real and negative while the others are real and positive, the system is stable along the eigenvector associated with the negative eigenvalues and is unstable away from this. The trajectory starting from any initial condition progressively converges on the eigenvector associated with the positive eigenvalue and moves to infinity along that line in the state space. This point is called a *saddle* and the system with a saddle is globally unstable. The vector fields of the three types of systems are shown in figure 2.1 [98].

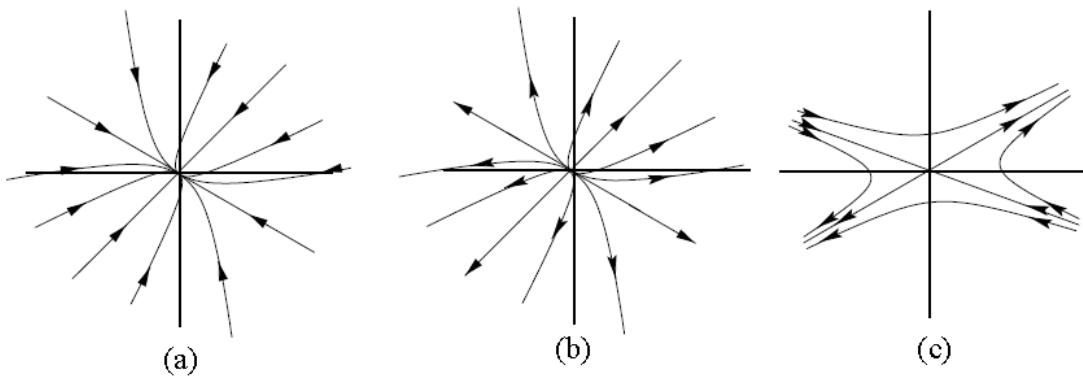


Figure 2.1: Vector fields of linear systems with two real eigenvalues, (a) both eigenvalues negative, (b) both eigenvalues positive and (c) one eigenvalue negative and one positive

3. If the real matrix A has complex conjugate eigenvalues $\alpha \pm i\beta$ with corresponding eigenvectors $a \pm ib$, then two linearly independent real vector solutions are

$$y_1 = e^{\alpha x} \begin{pmatrix} \cos(\beta x_a) - \sin(\beta x_b) \\ \cos(\beta x_b) + \sin(\beta x_a) \end{pmatrix},$$

$$y_2 = e^{\alpha x} \begin{pmatrix} \cos(\beta x_b) + \sin(\beta x_a) \\ \cos(\beta x_a) - \sin(\beta x_b) \end{pmatrix}.$$

For initial conditions at various distances from the origin, the trajectories are circles of various radius and the imaginary part of the eigenvalue gives the period of rotation. The vector field in the state space has the structure shown in Figure 2.2. An equilibrium point with imaginary eigenvalues is called a *centre*.

In general, if the eigenvalues are purely imaginary the orbits are elliptical. For initial conditions at different distances from the equilibrium point the orbits form a family of geometrically similar ellipses which are inclined at a constant angle to the axes, but having the same cyclic frequency.

When the eigenvalues are complex, with α nonzero, the sinusoidal variation of the state variables will be multiplied by an exponential term $e^{\alpha x}$. If α is negative, this term will decay as time progresses. Therefore the waveform in time-domain will be a damped sinusoid and in the state space the state will spiral in towards the equilibrium point. If α is positive the $e^{\alpha x}$ -term will increase with time and so in the state space the behavior will be an outgoing spiral, see figure 2.2 [98].

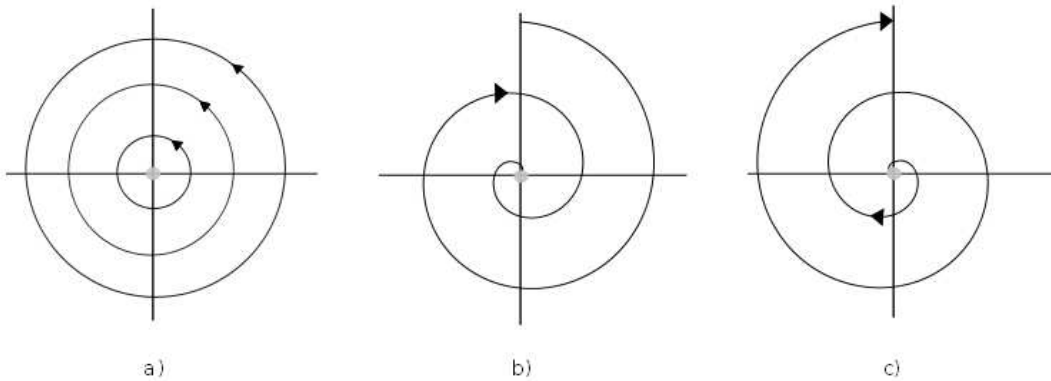


Figure 2.2: Vector fields in state space for (a) imaginary eigenvalues, (b) complex eigenvalues with negative real part and (c) complex eigenvalues with positive real part

Let $y_p(x)$ be a particular solution to the non-homogeneous system on the interval I and let $y_1(x), y_2(x), \dots, y_n(x)$ be a fundamental solution set on I for the corresponding homogeneous system. Then every solution to on I can be expressed as composition of particular and homogeneous solutions

$$y(x) = y_p(x) + c_1 y_1(x) + \dots + c_n y_n(x), \quad (2.12)$$

where c_1, c_2, \dots, c_n are constants.

Two useful techniques [83] for finding particular solutions are

1. the annihilator methods,
2. the method of variation of parameters.

Systems of linear differential equations

We can always reduce an explicit linear differential equation of any order to a first order system of differential equation. We usually use substitutions such as $y_n = y^{(n-1)}$ and we get a first order system of differential equations of dimension n in the form

$$\begin{aligned} y_1' &= y_2, & y_1(x_0) &= y_{01}, \\ y_2' &= f_1(x, y_1, y_2), & y_2(x_0) &= y_{02}, \\ & \vdots & & \\ y_n' &= f_n(x, y_1, y_2, \dots, y_n), & y_n(x_0) &= y_{0n}. \end{aligned}$$

Assume that equation (2.8) has $p_i(x) = a_i(x)/a_n(x)$ and $g(x) = b(x)/a_n(x)$, the system of differential equations is then given by

$$\begin{pmatrix} y_1' \\ y_2' \\ \vdots \\ y_n' \end{pmatrix} = \begin{pmatrix} 0 & 1 & \dots & 0 \\ 0 & 0 & & 0 \\ \vdots & & \ddots & 1 \\ -p_1(x) & -p_2(x) & & -p_n(x) \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ g(x) \end{pmatrix} \quad (2.13)$$

For simplification we denote the vector $(y_1, y_2, \dots, y_n)^T$ by \mathbf{y} , the vector $(0, 0, \dots, g(x))^T$ by $\mathbf{f}(\mathbf{x})$ and a matrix by $A(x)$. The equation (2.13) becomes the system of linear differential equations

$$\mathbf{y}' = A(x)\mathbf{y} + \mathbf{f}(\mathbf{x}), \quad (2.14)$$

$$A(x) = (a_{ij}(x)), \quad \mathbf{f}(\mathbf{x}) = (f_i(x)), \quad y_i(x_0) = (y_{0i}), \quad i, j = 1, \dots, n.$$

Let A be an $n \times n$ matrix. The following statements are equivalent and significant for determining of the solution stability [83]

- A is singular and does not have an inverse. (A matrix that has an inverse is called invertible or nonsingular. If no inverse exists the matrix is said to be singular.)
- The determinant of a singular matrix A is zero.
- The system of equations $A\mathbf{y} = 0$ has a nontrivial solution ($x \neq 0$).
- The columns (rows) of a singular matrix A form a linearly dependent set.

After the representing the basics of differential equations and some of their property, we start to solve them.

Example of analytical solution

Description of circuits using differential equations is very convenient for the electrical circuits' behavior analysis [76]. Electrical circuits are described by differential equations for time-dependent elements (capacitors, inductances) together with equations for linear and non-linear time-independent elements (resistors, diodes and transistors). Well-known Ohm's law and Kirchhoff's laws are part of the electronic circuit description.

Assume the differential equation of second order (2.15) describing the electrical circuit in the figure 2.3. We assume $y' = dy/dt$ for this example.

$$LCu''_C + RCu'_C + u_C = u, \quad u_C(0) = 0, \quad u'_C(0) = 0 \quad (2.15)$$

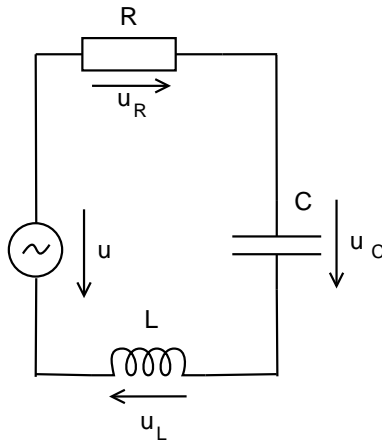


Figure 2.3: Electrical circuit with serial resistor, capacitor and inductor

The homogeneous equation is transferred to the characteristic equation and solved as a quadratic equation in the first step

$$LC\lambda^2 + RC\lambda + 1 = 0$$

$$\lambda_{1,2} = -\frac{RC \mp \sqrt{(RC)^2 - 4LC}}{2LC}.$$

There are three possible choices of the expected eigenvalues according to the value of the determinant $D = (RC)^2 - 4LC$

1. $D > 0 \longrightarrow \lambda_1 \neq \lambda_2 \in \text{Re}$,
2. $D = 0 \longrightarrow \lambda_1 = \lambda_2 \in \text{Re}$,
3. $D < 0 \longrightarrow \lambda_{1,2} = a \pm ib \in \text{Im}$

and due to three possible homogeneous solutions $y_h = u_{Ch}$ are expected

1. $u_{Ch} = C_1 e^{\lambda_1 t} + C_2 e^{\lambda_2 t}$,
2. $u_{Ch} = e^{\lambda t} (C_1 t + C_2)$,
3. $u_{Ch} = e^{at} (C_1 \cos(bt) + C_2 \sin(bt))$.

In this example we assume the multiple root ($\lambda_1 = \lambda_2 = -R/2L$), so the expected solution for the circuit is

$$u_{Ch} = e^{\lambda t} (C_1 t + C_2), \quad (2.16)$$

where C_1 and C_2 are unknown values.

As a second step it is necessary to determine the effect of the right-hand side in the differential equation (2.15). Let us say the electrical circuit has the alternating voltage source and the corresponding equation is $u = U_0 \sin(\omega t)$. We simplify the example for $U_0 = 1 \text{ V}$ and the expected particular equation $y_p = u_{Cp}$ looks like

$$u_{Cp} = A \sin(\omega t) + B \cos(\omega t). \quad (2.17)$$

To determine the unknown values A and B we derive the particular solution (2.17) up to the order the given differential equation

$$\begin{aligned} u'_{Cp} &= A\omega \cos(\omega t) - B\omega \sin(\omega t) \\ u''_{Cp} &= -A\omega^2 \sin(\omega t) - B\omega^2 \cos(\omega t) \end{aligned}$$

and replace u_{Cp} , u'_{Cp} and u''_{Cp} into the given differential equation (2.15)

$$LC\omega^2(-A \sin(\omega t) - B \cos(\omega t)) + RC\omega(A \cos(\omega t) - B \sin(\omega t)) + A \sin(\omega t) + B \cos(\omega t) = \sin(\omega t) \quad (2.18)$$

Comparing functions $\sin(\omega t)$, $\cos(\omega t)$ on both sides of the equation (2.18) we get

$$\begin{aligned} -ALC\omega^2 - BRC\omega + A &= 1 \\ -BLC\omega^2 + ARC\omega + B &= 0 \end{aligned}$$

$$\begin{aligned} A &= \frac{1 - LC\omega^2}{(LC\omega^2)^2 + (RC\omega)^2} \\ B &= -\frac{RC\omega}{(LC\omega^2)^2 + (RC\omega)^2} \end{aligned}$$

In the third step we add both homogeneous and particular parts together

$$u_C = u_{Ch} + u_{Cp}$$

$$u_C = e^{-\frac{R}{2L}t}(C_1t + C_2) + \frac{(1 - LC\omega^2)\sin(\omega t) - RC\omega\cos(\omega t)}{(LC\omega^2)^2 + (RC\omega)^2} \quad (2.19)$$

To determine the unknown C_1 and C_2 we insert the initial value $u_C(0) = 0$ into (2.19)

$$C_2 = \frac{RC\omega}{(LC\omega^2)^2 + (RC\omega)^2}$$

for inserting second initial value $u'_C(0) = 0$ we calculate the derivative of the equation (2.19)

$$u'_C = -\frac{R}{2L}e^{-\frac{R}{2L}t}(C_1t + C_2) + C_1e^{-\frac{R}{2L}t} + \frac{\omega(1 - LC\omega^2)\cos(\omega t) + RC\omega^2\sin(\omega t)}{(LC\omega^2)^2 + (RC\omega)^2} \quad (2.20)$$

and the initial value $u'_C(0) = 0$ is now inserted in (2.20)

$$C_1 = \frac{R^2C\omega - 2L\omega(1 - LC\omega^2)}{2L((LC\omega^2)^2 + (RC\omega)^2)} \quad (2.21)$$

The analytical solution u_C of the differential equation of second order (2.15) with multiple root for RLC circuit is given by

$$u_C = e^{-\frac{R}{2L}t} \left(\frac{R^2C\omega - 2L\omega(1 - LC\omega^2)}{2L((LC\omega^2)^2 + (RC\omega)^2)}t + \frac{RC\omega}{(LC\omega^2)^2 + (RC\omega)^2} \right) + \frac{(1 - LC\omega^2)\sin(\omega t) - RC\omega\cos(\omega t)}{(LC\omega^2)^2 + (RC\omega)^2} \quad (2.22)$$

We set the special values of the circuit as

$$R = 20 \, \Omega, \quad L = 2.5 \cdot 10^{-2} \, \text{H}, \quad C = 5 \cdot 10^{-5} \, \text{F}, \quad \omega = 1000 \, \text{rad/s}, \quad u = \sin(\omega t) \, \text{V}$$

we solve the equation (2.22) and we graphically represent the analytical solution of u_C in the graph 2.4.

Numerical solution

The second way to solve differential equations is the *numerical* solution. The numerical solving is based on approximations and it includes many other aspects of chosen numerical method such as initial conditions, generation and propagation errors, stability and convergence of the method, a variable stepsize etc. By numerical solution of differential equation we mean a sequence of values $y(t_0), y(t_1), \dots, y(t_i)$ for $i = 0, 1, \dots, n$.

As the analytical solution is unknown, we need to have initial values and information about the stability behavior of the solution for all initial values in the neighborhood of

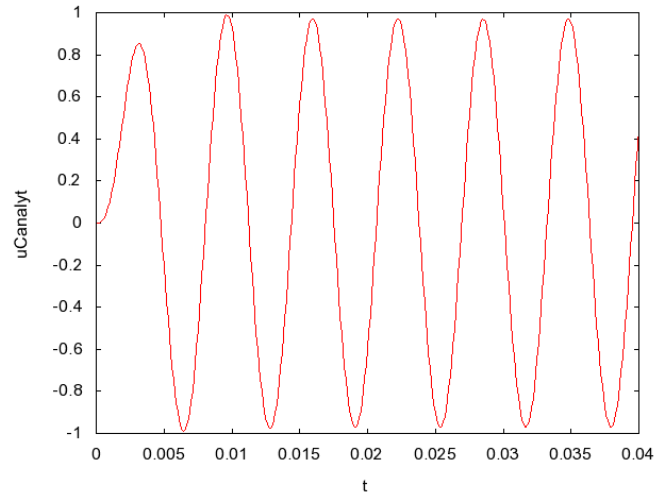


Figure 2.4: Voltage u_C in RLC circuit - computed from the analytical solution

a certain equilibrium point. We can carry the equilibrium point to the origin and define the stability of trivial solution such that

$$y'_i = f_i(y_1, \dots, y_n), \quad i = 1, \dots, n \quad (2.23)$$

is the system with zero initial values. The origin is called *stable* if for any $\epsilon > 0$ exists a $\rho < 0$ such that for the solution $\|y(x_0) < \rho\| \implies \|y(x) < \epsilon\|$ for all $x > x_0$ [57].

In the case of a linear system

$$y' = Ay \quad (2.24)$$

a particular stability property of any solution is equivalent to that stability property of the trivial solution. Thus one may transfer this property to the equation (2.24) and say that the equation (2.24) is stable. The stability concepts can be expressed by a matrix A .

A square matrix A is *stable* if there exists a constant C such that for all $n = 0, 1, 2, \dots$ $\|A_n\| \leq C$. Equivalent statements for matrix A of $n \times n$ are

- The minimal polynomial of A has all its zeros in the closed unit disc and all its multiple zeros in the open unit disc.
- The Jordan canonical form of A has all its eigenvalues in the closed unit disc with all eigenvalues of magnitude 1 lying in 1×1 blocks.
- There exists a non-singular matrix S such that $\|S^{-1}AS\|_\infty \leq 1$ [8].

In case of linear system with constant coefficients and with $\lambda_1, \dots, \lambda_n$ as eigenvalues of coefficients matrix A , one can write down equivalent statements:

- The solutions of the system are stable if for all $\text{Re}(\lambda_n) < 0$.

- All solutions of the system are unstable if exists $\text{Re}(\lambda_n) > 0$.
- For all eigenvalues such that $\text{Re}(\lambda_n) < 0$ and for $\text{Re}(\lambda_i) = 0$, λ_i are simple roots, all solutions are stable [88].

The convergence is the point of the interest together with stability. This attribute of numerical methods guaranties the solution reaches the exact solution after few steps of calculation. The stability and convergence determine the consistency of the method [28].

From this part of the work numerical methods for the solution of the *initial value problem* in ordinary differential equations are evaluated and compared. An initial value problem is specified as follows

$$y'(x) = f(y(x)), \quad y(x_0) = y_0. \quad (2.25)$$

There exist two main types of numerical methods, the first types use for the next approximation y_n only the current already known approximation y_{n-1} , we call them *one-step methods*. The other ones called multistep methods solve the next approximation using current and previous approximations $y_n, y_{n-1}, y_{n-2}, \dots$

We proceed from introduction of chosen one-step methods such as the simplest Euler method through generalizations to chosen multistep methods. These generalizations are based on more computations in a step, use of more previous values or higher derivatives. We will see these procedures later.

2.2 Euler method

The simplest and the most analyzed numerical method for solving ordinary differential equations is *Euler method* (proposed in the 18th century by Euler). It is the simple recursion formula which studies the solution for only certain values $x = 0, h, 2h, \dots$, where h is called an integration step or a stepsize and assumes that dy/dx is constant between points. The recursion formula is given by

$$y_n = y_{n-1} + hf(y_{n-1}), \quad y(0) = y_0. \quad (2.26)$$

The sequence of values starting from the initial value x_0 is used for computation and stepsizes between each values of sequence $x_1 - x_0, x_2 - x_1, \dots$ are denoted as h_1, h_2, \dots , the highest is denoted by h . For each value of n , each approximation of y_n is computed using a previous value y_{n-1} which is exactly equal to $y(x_{n-1})$. We see that the quality of approximations of y_{n-1} depend on the magnitude of h . To analyze how the function $hf(y_n)$ varies, the Lipschitz constant L can be used [45].

The function f satisfied a Lipschitz condition if there is a Lipschitz constant L if for all $u, v \in R^N$

$$\|f(u) - f(v)\| \leq L\|u - v\|. \quad (2.27)$$

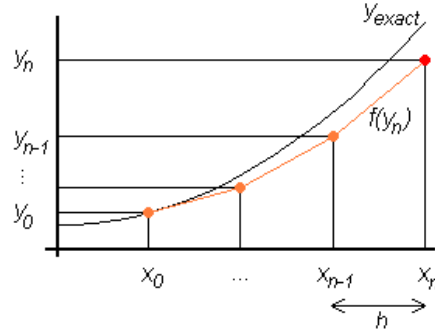


Figure 2.5: Graphic interpretation of Euler method

Symbols $\|$ in $\|f(u) - f(v)\|$ represent the norm.

The function f satisfied a „one-side Lipschitz condition“, with „one-side Lipschitz constant“ λ if for all $x \in [a, b]$ and all $u, v \in R^N$

$$\langle f(x, u) - f(x, v), u - v \rangle \leq \lambda \|u - v\|^2. \quad (2.28)$$

The condition for the Lipschitz constant L guarantees the existence and uniqueness of the solution. We will assume this for the present section also for other methods [16].

The Euler method is based on a truncated Taylor series expansion which implies the *local truncation error* l_n (or *discretization error*) of the method as a $O(h^2)$. The local truncation error is an error committed by the method in a single step when the values at the beginning of that step are assumed to be exact. From this fact we can say, that the Euler method is first order technique, generally a method with local truncation error equals to $O(h^{p+1})$ is said to be of p -th order. At the n -step the error is defined by

$$\begin{aligned} l_n &= y(x_{n-1} + h) - y(x_{n-1}) - hf(x_{n-1}, y(x_{n-1})) \\ &= y(x_{n-1}) + hy'(x_{n-1}) + \frac{h^2}{2}y''(x_{n-1} + \theta h) - y(x_{n-1}) - hy'(x_{n-1}) \\ &= \frac{h^2}{2}y''(x_{n-1} + \theta h), \quad 0 < \theta < 1. \end{aligned}$$

The truncation error is different from *the global error* ϵ_n [28], which is defined as

$$\begin{aligned} \epsilon_n &= y(x_{n-1} + h) - y_n = y(x_{n-1}) - hf(x_{n-1}, y(x_{n-1})) + l_n - y_{n-1} - hf(x_{n-1}, y_{n-1}) \\ &= \epsilon_{n-1} - hf(x_{n-1}, y(x_{n-1})) - hf(x_{n-1}, y_{n-1}) + l_n. \end{aligned} \quad (2.29)$$

In the most cases, the exact solution is unknown and hence the global error cannot be evaluated. Evaluations of errors are closely linked to a variable stepsize determination, but we will discuss it later. The magnitude of stepsize is important for the convergence of the method. A convergent numerical method is the one where the numerically computed

solution approaches the exact solution as the stepsize approaches 0. For problems with unknown exact solution, we choose the solution obtained with a sufficiently small time step as the „exact“ solution to study the convergence characteristics. So taking the norm of global error in (2.29) and applying the triangle inequality, the Lipschitz condition and the bound on the local error, we get the first-order inequality

$$\|\epsilon_n\| = (1 + hL)\|\epsilon_{n-1}\| + \frac{Mh^2}{2}.$$

Since $\epsilon_0 = 0$, the inequality has solution given by

$$\|\epsilon_n\| \leq \frac{Mh}{2L}(1 + hL)^n$$

where as $n \rightarrow \infty$ and $h \rightarrow 0$, we have $\epsilon_n \rightarrow 0$ and $y_n \rightarrow y(x_n)$ for some $M < \infty$ that is the numerical solution converges to the exact solution. Then we can say that methods of order higher than one are also convergent [36].

For the Euler methods there are stepsize limitations such as to ensure numerical stability, reasonable required accuracy, also fast convergence behaviour. A bit of improvement is given by *implicit Euler method*

$$y_n = y_{n-1} + hf(y_n), \quad y(0) = y_0 \tag{2.30}$$

For better understanding of stepsize and convergence of the method, have a look to a simple example also called Dahlquist problem with known exact solution [39].

Example

Consider a Dahlquist problem

$$y' = qy, \quad y(0) = 1 \tag{2.31}$$

with known analytical solution given by $y(x) = \exp(qx)$. In this case we choose constant $q = 1$.

The solution is computed for stepsizes $h = 0.2$ (a first case) and $h = 0.1$ (a second case), exact solution and solutions for first and second case are plotted in picture 2.6. The comparison with the exact solution is presented in in the table 2.1 where the second column represents the exact value in the value x_n . The third column, the fifth column y_n respectively show computed values by Euler method with stepsizes 0.2, 0.1 respectively. And the fourth column, the sixth column respectively determines error between the exact solution and the computed solution. The Euler method computes less steps for stepsize $h = 0.2$ which also cause the bigger global error than error for $h=0.1$. For the first case the solution converges slower to the exact value.

Table 2.1: Solution for Dahlquist problem (2.31) using explicit Euler method

x_n	$y(x_n)$	$h = 0.2$		$h = 0.1$	
		y_n	err	y_n	err
0.0	1.000000	1.000000	0.000000	1.000000	0.000000
0.1	1.105171			1.100000	0.005171
0.2	1.221403	1.200000	0.021403	1.210000	0.011403
0.3	1.349859			1.331000	0.018859
0.4	1.491825	1.440000	0.051825	1.464100	0.027725
0.5	1.648721			1.610510	0.038211
0.6	1.822119	1.728000	0.094119	1.771561	0.050558
0.7	2.013753			1.948717	0.050558
0.8	2.225541	2.073600	0.151941	2.143589	0.081952
0.9	2.459603			2.357948	0.101655
1.0	2.718282	2.488320	0.229962	2.593742	0.124540

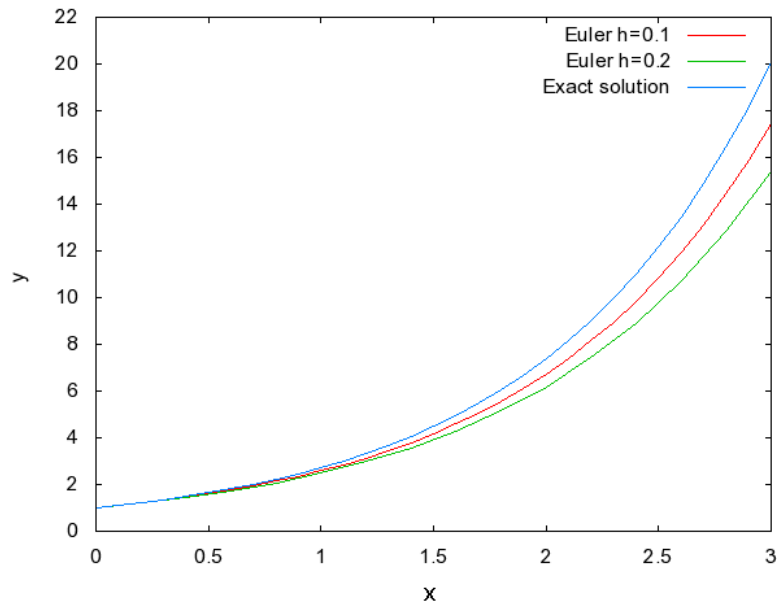


Figure 2.6: Solutions for different stepsizes by Euler method

To check the order of Euler method with the fixed stepsize, we determine the error each time for n steps and set the stepsize such as $h = (t_{max} - t_{min})/n$ for different n values as $n = 10, 20, 40, \dots, 10240$, see table 2.2. We plot the order graph with the log of stepsizes on the x -axis and the log of absolute values of errors on the y -axis.

From now we will use the notation $1e-02, 1e-03, 1e-04, \dots, 1e+03, 1e+04, \dots$ re-

spectively for numbers $1 \cdot 10^{-2}$, $1 \cdot 10^{-3}$, $1 \cdot 10^{-4}$, \dots , $1 \cdot 10^3$, $1 \cdot 10^4$, \dots respectively.

Errors give us an order illustrates the rate at which the numerical error decreases with stepsize, see picture 2.7. Note that the order plot is from now always a log-log plot because the size of the error spans orders of magnitude. The slope of the error curve on a log-log plot gives the order of accuracy of the method. If the slope is unity, the error scales linearly with the stepsize. If the slope is two, then the error scales as the square of the stepsize.

Checking the slope of lines through the points we can say that the order of Euler method is 1. This means that results are consistent with order 1. Generally holds, that if the method has order p , the error for small h approximately satisfies an equation

$$E \approx Ch^p \tag{2.32}$$

assuming that E is the norm of the error and C is some constant so that everything is scalar and taking logs, we find

$$\log(E) = \log(C) + p \log(h). \tag{2.33}$$

This means the graph of E versus h , on a log-log scale should have slope p .

$$\begin{aligned} \frac{16.79689 \cdot 10^{-3}}{8.446252 \cdot 10^{-3}} &\cong 1.988680 \\ \frac{8.446252 \cdot 10^{-3}}{4.235185 \cdot 10^{-3}} &\cong 1.994305 \\ \frac{4.235185 \cdot 10^{-3}}{2.120621 \cdot 10^{-3}} &\cong 1.997144 \end{aligned}$$

The slope p is computed in the column called ratio in the table 2.2. To show the practical and fast approach of order checking of Euler method we use errors from the table 2.2. Taking the error for $h = 0.1 \cdot 2^{-3}$ and divide it by the error for $h = 0.1 \cdot 2^{-4}$ we get 1.988680. Taking the error for $h = 0.1 \cdot 2^{-4}$ and divide it by the error for $h = 0.1 \cdot 2^{-5}$ we get 1.994305, etc. We see that the order of Euler method holds, the ratios are approximately $2^1 = 2^p$ where p is the order of the method.

Knowing that the Euler method converges and the error increases for increasing time over the tolerable limit, let us study the behaviour of the method over extended interval [39]. Assume the linear system of equations of constant coefficients

$$y'(x) = My(x) \tag{2.34}$$

where M is the constant matrix. This problem can be transformed using a few assumptions according to [21] to the simpler form

$$y'(x) = q(x) \tag{2.35}$$

Table 2.2: Errors and the order of Euler method for different fixed stepsizes

h	err	ratio
0.2	0.2299618	
0.1	0.1245394	1.846
$0.1 \cdot 2^{-1}$	6.498412e-02	1.916
$0.1 \cdot 2^{-2}$	3.321799e-02	1.956
$0.1 \cdot 2^{-3}$	1.679689e-02	1.978
$0.1 \cdot 2^{-4}$	8.446252e-03	1.989
$0.1 \cdot 2^{-5}$	4.235185e-03	1.994
$0.1 \cdot 2^{-6}$	2.120621e-03	1.997
$0.1 \cdot 2^{-7}$	1.061069e-03	1.999

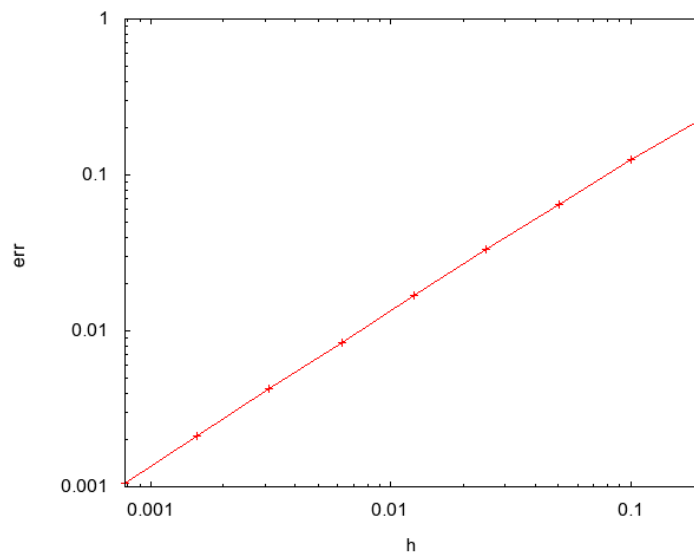


Figure 2.7: Order of Euler method for Dahlquist problem

where $z = hq$ with the exact solution $y_{n+1} = \exp(zn)y_0$ and z is a complex number. Using fixed stepsize it was said that $(1 + hq)^n$ is an acceptable approximation to $\exp(nhq)$, where both expressions as $n \rightarrow \infty$ are bounded. That also means that if the stability function defined as

$$R(z) = \frac{y_{n+1}}{y_n} \tag{2.36}$$

meet the condition $R(z) \leq 1$, then $|1 + hq|$ is bounded by 1. The set of values for the exact

solution is bounded in the non-positive half-plane $z \in C : R(z) \leq 0$. For this condition is the set of points for Euler method equals to $|1 + z| \leq 1$ (set of points is the closed disc in the complex plane with the centre in -1 and radius of 1). This property is also called boundedness. Property of converging is less strict then the unify, the exact solution lays in the negative left-plane $z \in C : R(z) < 0$, so the set of points for Euler method lays in the open disc with the centre in -1 and radius of 1. For Euler method and implicit Euler method have been derived stability regions as follows

$$R(z) = \begin{cases} 1 + z, & \text{(Euler method)} \\ \frac{1}{1-z}. & \text{(implicit Euler method)} \end{cases}$$

Stability regions of both methods are plotted and colored in figure 2.8.

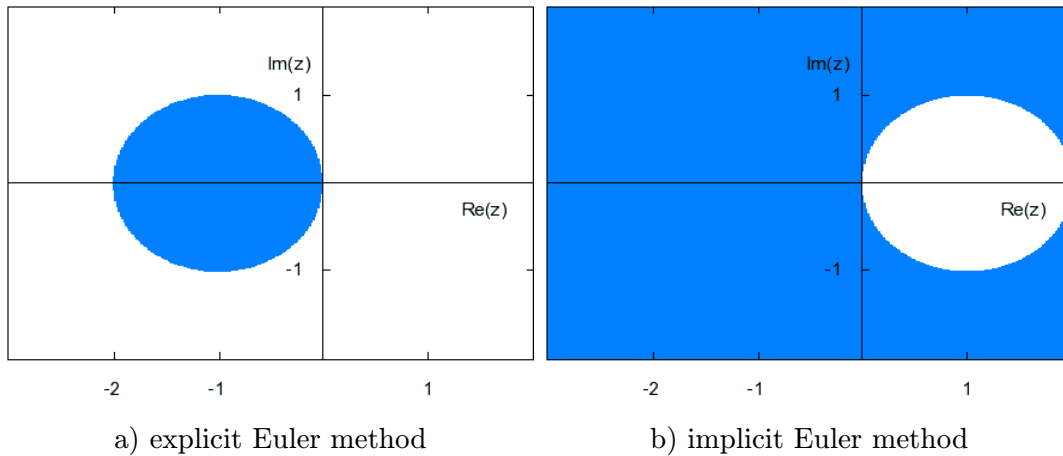


Figure 2.8: Stability regions for Euler method

We say that the stability region is defined as a set of points in the complex plane, z should stay in the disc for other problems. It can be achieved only by reducing h . This causes many limitations. For example to solve stiff problems with very negative eigenvalues it means to decrease h so much that it makes explicit method unusable. If the stability function has no poles in the left half-plane, this means the stability region includes all zeros of the left half-plane and the method is said to be A-stable. It also holds that the magnitude $|R(z)|$ must be bounded by 1 for z on the imaginary axes. A-stability is a very desirable property for any numerical algorithm, particularly if initial value problems were to be stiff or stiff oscillatory [64].

Another interesting way how to study the stability region is using *order stars* technique [57], see colored regions in figure 2.9. This property of multiplying the stability function by $\exp(-z)$ should make no difference in the characteristic of the method stability. Notice the behaviour near $z = 0$ and $z = -1$. For $|\text{Re}(z)|$ large, the behaviour is effected by the exponential function, the behaviour around zero is the same as for the absolute stability

region and the behaviour at $z = -1$ is determined by a pole. The regions intersect with zero and $\text{Re}(R(z) \exp(-z))$ positive are called *fingers*. Regions with negative $\text{Re}(R(z) \exp(-z))$ are known as *dual fingers*. Similar technique as order stars is the *order arrows*. The technique of order arrows is about to plot the paths in the complex plane where $\omega(z) = \exp(-z)R(z)$ is real and positive.

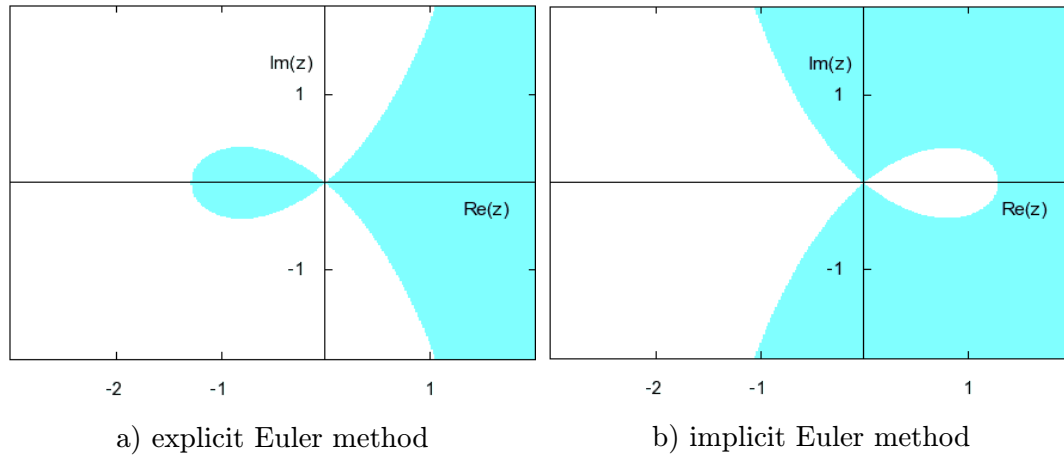


Figure 2.9: Order stars for Euler method

2.3 Taylor series

As we mentioned, Euler method can be generalized in few different ways. Consider the way of use of higher derivatives. The Euler method can be extended or be a stepwise process. When an acceptable estimate for $y(x_n)$ is given, the coefficients in a series in powers of $(x - x_n)$ can be determined. This series is then used in approximately computing $y(x_{n+1})$ where the size of $(x_{n+1} - x_n)$ is chosen so that acceptable accuracy is obtain with a reasonable number of terms in the series. The Taylor polynomial can be used to approximate function and to bound the error of the approximation [7].

The polynomial

$$T_n(x) = p(a) + p'(a)(x - a) + \frac{p''(a)}{2}(x - a)^2 + \frac{p'''(a)}{3!}(x - a)^3 + \dots + \frac{p^n(a)}{n!}(x - a)^n \quad (2.37)$$

is called the n -th degree **Taylor polynomial** for p at a and the expression

$$\frac{p^{n+1}(a)}{n + 1!}(x - a)^{n+1}$$

is called the Lagrange form of the reminder and it is the error of approximating $p(x)$ by $T_n(x)$.

Very interesting concept of rooted trees were established by Butcher, Wanner, Hairer et al. [16, 57]. We write T for the set of all trees and τ for the trivial tree with exactly one vertex. Let $t_1 = (V_1, R_1)$, $t_2 = (V_2, R_2), \dots, t_s = (V_s, R_s)$ as trees where vertices V_s are suppose to be disjoint and we denote r_1, r_2, \dots, r_s as roots of these trees. Let $t = (V, R)$ where $V = r \cup V_1 \cup V_2 \cup \dots \cup V_s$ with $r \notin V_1 \cup V_2 \cup \dots \cup V_s$ and $R = (r, r_1), (r, r_2), \dots, (r, r_s) \cup R_1 \cup R_2 \cup \dots \cup R_s$. Now we see that t is a tree with the root r such as

$$t = [t_1 t_2 \dots t_s]. \quad (2.38)$$

For the power notation it is convenient to operate with decomposition and conciseness. If we have a tree t with more than one vertex, then we can find a decomposition of the form (2.38) by simply removing the root and all arcs from it and identifying t_1, t_2, \dots, t_s with the connected components that remain. If we have repeated trees among t_1, t_2, \dots, t_s , we combine them using the power notation which is the first aspect of the conciseness. That is e. q. $[t_1 t_1 t_1 t_2 t_2]$ can be written as $[t_1^3 t_2^2]$. The second aspect of the conciseness is to indicate the matching or repetitions of symbols. Thus we will write for the given example $[[[\tau]\tau]]$, that can be simplified by $[_2[\tau]\tau]_2$ only [16]. We always need to have pairs of brackets $[,]$ with the same subscripts. This example will be presented by tree in figure 2.10. Notice that the symmetric tree has different notation.

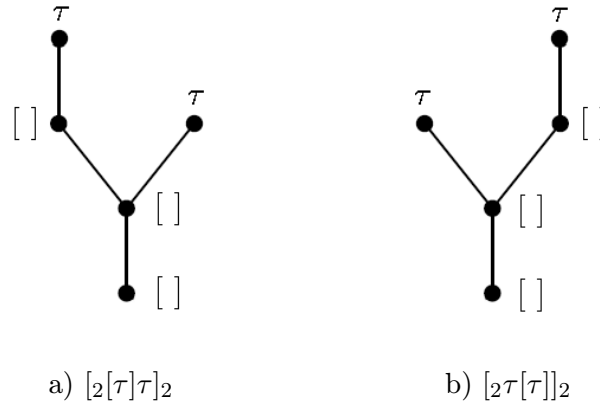


Figure 2.10: Examples of rooted trees

Different notation is based on a binary operation on T . Define the operation „ \cdot “ as $T \times T \rightarrow T$ by the formulae

$$\begin{aligned} \tau \cdot u &= [u] \\ [t_1 t_2 \dots t_s] \cdot u &= [t_1 t_2 \dots t_s u]. \end{aligned} \quad (2.39)$$

For simplification it is common to elide the symbol „ \cdot “ in cases which can not be confusion of the notation, such as $\tau \cdot \tau\tau$ means $\tau \cdot (\tau \cdot \tau)$. From equations (2.39) it is easy to see that we can write any tree (excepting τ) as the product of two trees with fewer vertices. That

is we can write any tree in terms of τ and the binary operation \cdot . According to the rule $tu \cdot v = tv \cdot u$ where $u, v, t \in T$ trees mentioned above have the same description in this notation, which is $\tau \cdot (\tau\tau \cdot \tau\tau)$. All trees with less than six vertices and their notations are shown in table 2.3 [16].

Now some functions on trees will be defined. Denote $r(t)$ *the order* of the tree t or the number of vertices in a tree t .

For the order holds following recursions: for a tree with root $r(\tau) = 1$ and

$$\begin{aligned} r([t_1 t_2 \cdots t_s]) &= 1 + r(t_1) + r(t_2) + \cdots + r(t_s), \\ r(tu) &= r(t) + r(u). \end{aligned}$$

Another function which can be defined is *height* H of the tree t which is given by length of the longest possible path in the tree

$$\begin{aligned} H(\tau) &= 1 \\ H([t_1 t_2 \cdots t_s]) &= 1 + \max_i H(t_i), \\ H(tu) &= \max[H(t), 1 + H(u)]. \end{aligned}$$

By the *width of the tree* $w(t)$ we understand the number of terminal vertices that is the number of vertices with no successors. There are two known conventions, one defines the width of simply tree τ as a 0, second convention is using $w(t) = 1$, to distinguish it denote the second convention as $\bar{w}(t) = 0$. We can imply that if $w(\tau) = 0$, $\bar{w}(t) = 1$ and $t \neq \tau$, then $w(t) = \bar{w}(t)$.

Then there is a *density* γ of the tree which does not have any obvious significance as the previous functions, but we can simplify the description as a measure of the non-bushiness of a tree. The *density* γ of the tree satisfies





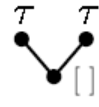
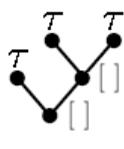

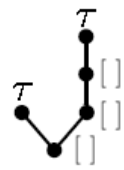
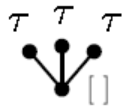
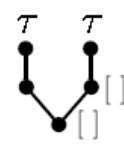
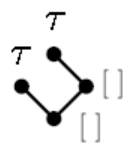
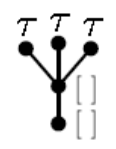
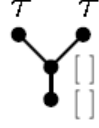
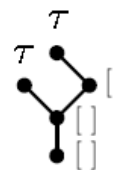

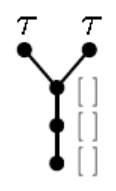

$$\begin{aligned} \gamma(\tau) &= 1, \\ \gamma([t_1 t_2 \cdots t_s]) &= r([t_1 t_2 \cdots t_s])\gamma(t_1)\gamma(t_2)\cdots\gamma(t_s). \end{aligned}$$

Two extreme cases could happen, first is the type of tree with $H(t) = 2$ and the second for a tree with $H(t) = r(t)$, where the density is $\gamma(t) = r(t)$ and $\gamma(t) = r(t)!$ respectively.

Next function the *symmetry* σ of the tree is defined as the order of its symmetry group which is the group of isomorphisms of the tree with itself. If $t = [t_1^{n_1} t_2^{n_2} \cdots t_s^{n_s}]$, where t_1, t_2, \cdots, t_s are distinct, then for the symmetry holds

$$\begin{aligned} \sigma(\tau) &= 1, \\ \sigma(t) &= n_1!n_2!\cdots n_s!\sigma(t_1)^{n_1}\sigma(t_2)^{n_2}\cdots\sigma(t_s)^{n_s}. \end{aligned}$$

Table 2.3: Notations for trees [16]

	τ	τ		$[\tau^4]$	$(\tau\tau \cdot \tau)\tau \cdot \tau$
	$[\tau]$	$\tau\tau$		$[\tau^2[\tau]]$	$(\tau\tau \cdot \tau) \cdot \tau\tau$
	$[\tau^2]$	$\tau\tau \cdot \tau$		$[\tau[\tau^2]]$	$\tau\tau \cdot (\tau\tau \cdot \tau)$
	$[2\tau]_2$	$\tau \cdot \tau\tau$		$[\tau[2\tau]_2]$	$\tau\tau \cdot (\tau \cdot \tau\tau)$
	$[\tau^3]$	$(\tau\tau \cdot \tau)\tau$		$[[\tau]^2]$	$(\tau \cdot \tau\tau) \cdot \tau\tau$
	$[\tau[\tau]]$	$\tau\tau \cdot \tau\tau$		$[2\tau^3]_2$	$\tau \cdot (\tau\tau \cdot \tau)\tau$
	$[2\tau^2]_2$	$\tau(\tau\tau \cdot \tau)$		$[2\tau[\tau]_2]$	$\tau(\tau\tau \cdot \tau\tau)$
	$[3\tau]_3$	$\tau(\tau \cdot \tau\tau)$		$[3\tau^2]_3$	$\tau \cdot \tau(\tau\tau \cdot \tau)$
				$[4\tau]_4$	$\tau \cdot \tau(\tau \cdot \tau\tau)$

We are also interested in labelling trees. It is described by the function $\beta(t)$ denoting the number of ways of labelling a tree t with $r(t) - 1$ distinct labels where a root of the tree is not labelled. Or we can use the notation $\bar{\beta}(t)$ for a number of ways of labelling a tree with carrying out this labelling with $r(t)$ labels where every vertex is labelled. Then it holds

$$\begin{aligned}\beta(t) &= \frac{[r(t) - 1]!}{\sigma(t)}, \\ \bar{\beta}(t) &= \frac{r(t)!}{\sigma(t)}.\end{aligned}\tag{2.40}$$

And the final interesting function is a function $\alpha(t)$ which define a number of ways of labelling a tree t with a given totally ordered set V in such a way that if (m, n) is an arc then $m < n$. The function is given by

$$\alpha(t) = \frac{r(t)!}{\gamma(t)\sigma(t)}.$$

For our exemplar trees the functions are determined in table 2.4. Functions for trees up to order 5 are determined in table 2.5.

Table 2.4: Functions and enumerations of exemplar tree

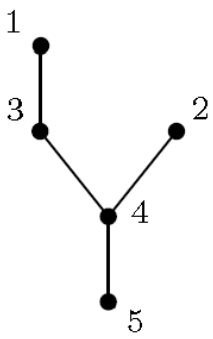
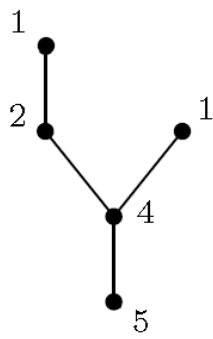

















 <p style="text-align: center;">a)</p>	 <p style="text-align: center;">b)</p>	<p>$r(t) = 5$, (see a)</p> <p>$\sigma(t) = 2! = 2$</p> <p>$\gamma(t) = 5 \cdot 1 \cdot 2 \cdot 1 \cdot 2 = 40$, (see b)</p> <p>$\omega(t) = 2$</p> <p>$H(t) = 4$</p> <p>$\alpha(t) = \frac{r(t)!}{\sigma(t)\gamma(t)} = \frac{5!}{2 \cdot 40} = \frac{3}{2}$</p> <p>$\beta(t) = \frac{r(t)!}{\sigma(t)} = \frac{5!}{2} = 60$</p>
--	--	---

Table 2.5: Functions for trees up to order 5 [16]

t	$r(t)$	$\sigma(t)$	$\gamma(t)$	$\omega(t)$	$H(t)$	$\alpha(t)$	$\beta(t)$
	1	1	1	0	1	1	1
	2	1	2	1	2	1	2
	3	2	3	2	2	1	3
	3	1	6	1	3	1	6
	4	6	4	3	2	1	4
	4	1	8	2	3	3	24
	4	2	12	2	3	1	12
	4	1	24	1	4	1	24
	5	24	5	4	2	1	1
	5	2	10	3	3	6	12
	5	2	15	3	3	4	12
	5	1	30	2	4	4	24
	5	2	20	2	3	3	12
	5	6	20	3	3	1	4
	5	1	40	2	4	3	24
	5	2	60	2	4	1	12
	5	1	120	1	5	1	24

The trees notation is very convenient for Taylor series. The exact solution of the differential equation can be expressed by Taylor series and it becomes increasingly complicated as we evaluate higher derivatives. We need to formulate the second, third, fourth, ... derivatives. Hence we look for a systematic pattern

$$\begin{aligned}
 y'(x) &= f(y(x)) \\
 y''(x) &= f'(y(x))y'(x) = f'(y(x))f(y(x)) \\
 y'''(x) &= f''(y(x))\left(f(y(x)), y'(x)\right) + f'(y(x))f'(y(x))y'(x) \\
 &= f''(y(x))\left(f(y(x)), f(y(x))\right) + f'(y(x))f'(y(x))f(y(x)) \quad (2.41)
 \end{aligned}$$

Next higher derivatives are increasingly complicated. But using systematic patterns we write $f(y(x)) = \mathbf{f}$, $f'(y(x)) = \mathbf{f}'$, $f''(y(x)) = \mathbf{f}''$ and we found they depend on the following four vectors: \mathbf{f} , $\mathbf{f}'\mathbf{f}$, $\mathbf{f}''(\mathbf{f},\mathbf{f})$, $\mathbf{f}'\mathbf{f}'\mathbf{f}$. The motivation of using the systematic patterns is to write the Taylor series in elementary differentials notation and also in rooted trees notation called operation diagram.

The elementary differentials noted in operation diagram are presented in the way that $\mathbf{f}^{(n)}$ is an n -ary operator and is always attached to a vertex in a particular diagram and also to a vertex which has n outward branching arcs. The n operands on which $\mathbf{f}^{(n)}$ is to act in a particular diagram are found from the n subdiagrams rooted to these n outward branching arcs. In the case $n = 0$, corresponding to terminal vertices, $\mathbf{f}^{(0)} = \mathbf{f}$ [16].

It is clear that for any rooted tree it is possible to form the operation diagram and a corresponding elementary differential. Elementary differentials for trees up to order 5 are given in table 2.6 in the second column.

Our exemplar tree is presented as on picture 2.11a.

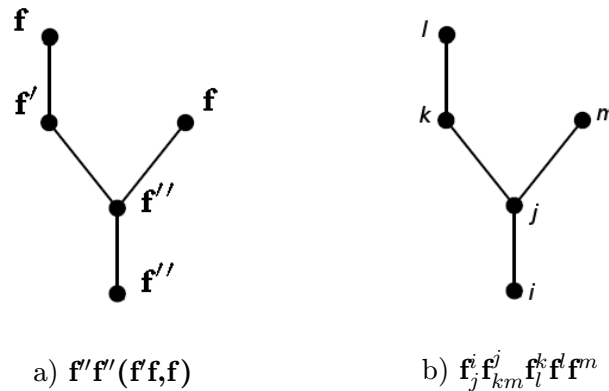



















Figure 2.11: Examples of rooted trees notations

Table 2.6: Elementary differentials up to order 5

t	elementary differentials	partial derivative
	f	f^i
	$f'f$	$f_j^i f^j$
	$f''(f, f)$	$f_{jk}^i f^j f^k$
	$f'f'f$	$f_{jk}^i f_l^j f^k f^l$
	$f'''(f, f, f)$	$f_{jkl}^i f^j f^k f^l$
	$f''(f, f'f)$	$f_{jk}^i f_l^j f^k f^l$
	$f''f''(f, f)$	$f_{jk}^i f_{kl}^j f^k f^l$
	$f'f'f'f$	$f_{jk}^i f_l^j f^k f^l$
	$f^{(4)}(f, f, f, f)$	$f_{jklm}^i f^j f^k f^l f^m$
	$f'''(f, f, f'f)$	$f_{jkl}^i f^j f^k f_l^l f^m$
	$f''(f, f''(f, f))$	$f_{ijk}^i f_l^j f_m^k f^l f^m$
	$f''(f, f'f'f)$	$f_{jk}^i f_l^j f_m^k f^l f^m$
	$f''(f'f, f'f)$	$f_{jl}^i f_k^j f_m^k f^l f^m$
	$f'''f'''(f, f, f)$	$f_{jklm}^i f^j f^k f^l f^m$
	$f''f''(f, f'f)$	$f_j^i f_{kl}^j f_m^k f^l f^m$
	$f''f''f''(f, f)$	$f_{jk}^i f_{lm}^j f^k f^l f^m$
	$f'f'f'f'f$	$f_{jk}^i f_l^j f_m^k f^l f^m$

We can rewrite previous equations (2.41) in elementary differential notation and form following relation such as

$$\begin{aligned}y'(x) &= \mathbf{f} \\y''(x) &= \mathbf{f}'\mathbf{f} \\y'''(x) &= \mathbf{f}''(\mathbf{f}, \mathbf{f}) + \mathbf{f}'\mathbf{f}'\mathbf{f} \\y^{(4)}(x) &= \mathbf{f}'''(\mathbf{f}, \mathbf{f}, \mathbf{f}) + \mathbf{f}''(\mathbf{f}, \mathbf{f}'\mathbf{f}) + \mathbf{f}'\mathbf{f}''(\mathbf{f}, \mathbf{f}) + \mathbf{f}'\mathbf{f}'\mathbf{f}'\mathbf{f}\end{aligned}$$

It is possible to write the notation \mathbf{f}' , \mathbf{f}'' , \mathbf{f}''' , \dots as $\mathbf{f}^{(1)}$, $\mathbf{f}^{(2)}$, $\mathbf{f}^{(3)}$, \dots and then the vectors compose in the sense of partial derivative notation. Let $\mathbf{f}_{j_1 j_2 \dots j_n}^i = f_{j_1 j_2 \dots j_n}^i(y(x))$ denote as an n -th order partial derivative of component number i of \mathbf{f} . The summation convention applies, it means an implicit summation over every superscript j , k , \dots which appears also as a subscript. Then the root of the tree is attached by label i and to the other vertices are labeled j , k , l, \dots For each vertex we write down the \mathbf{f} and we attach to it a superscript equal to its label and subscript equal to the labels of each outwardly connected vertex. The (summed) product of these factors is the expression for the i -th component of the vector [16]. Our exemplar tree is presented in the partial derivative notation in picture 2.11b. Partial derivatives for trees up to order 5 are given in table 2.6 in the third column.

The next generalization of the Euler method assumes instead of computing f once in a step that the method computes f two or more times with different arguments. This approach defines an important class of one-step method known as *Runge-Kutta methods*.

2.4 Runge–Kutta methods

Suppose we know $y(x_n)$ and we want to determine an approximation y_{n+1} to $y(x_n + h)$. The idea behind the Runge-Kutta methods is to compute the value of $f(x, y)$ at several conveniently chosen points near to the solution in the interval $(x_n, x_n + h)$ and to combine these values in such a way as to get good accuracy in the computed increment.

Generally, this important section of numerical methods can be written in the form of equations such as

$$\begin{aligned}y_n &= y_{n-1} + h \sum_{j=1}^s b_j F_j, \\Y_i &= y_{n-1} + h \sum_{j < i}^s a_{ij} F_j,\end{aligned}\tag{2.42}$$

where $F_i = f(Y_i)$ is evaluated by approximations y_n to $y(x_n)$ for $i = 1, 2, \dots, s$, constants b_j , a_{ij} can be written into table 2.7. Types of methods could be specified by different values of those coefficients. The tableau was defined by J. C. Butcher [19].

Table 2.7: Butcher tableau of Runge–Kutta methods

0					
c_2	a_{21}				
c_3	a_{31}	a_{32}			
\vdots	\vdots	\vdots	\ddots		
c_s	a_{s1}	a_{s2}	\cdots	$a_{s,s-1}$	
	b_1	b_2	\cdots	b_{s-1}	b_s

To specify some types of the method, one needs to provide the constant number s , which determines the number of internal stages, and constants a_{ij} (for $1 \leq j < i \leq s$), constants b_i (for $i = 1, 2, \dots, s$) and constants c_i (for $i = 2, 3, \dots, s$) [8]. For the demonstration we will use two types of notations to show the whole theory of some methods in this section. The second notation 2.43 is also quite known, it is given such as

$$y_{n+1} = y_n + h \sum_{i=1}^s b_i k_i, \tag{2.43}$$

where k_i represent internal stages

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f(t_n + c_2 h, y_n + a_{21} h k_1) \\ &\vdots \\ k_i &= f(t_n + c_s h, y_n + a_{s1} h k_1 + a_{s2} h k_2 + \cdots + a_{s,s-1} h k_{s-1}) \end{aligned}$$

Runge–Kutta methods can be classified into three main classes: explicit, semi-implicit and implicit. The separation is given by the characteristics in table 2.8.

Table 2.8: Types of Runge–Kutta methods

Type	Coefficients	No. of coefficients
Explicit	$a_{ij} = 0, \quad j \geq i$	$\frac{s(s+1)}{2}$
Semi-Implicit	$a_{ij} = 0, \quad j > i$	$\frac{s(s+3)}{2}$
Implicit	$a_{ij} \neq 0$ for at least one $j > i$	$s(s+1)$

The local truncation error of Runge–Kutta methods cannot be worse than the Euler method from the view of the consistency condition and it is $O(h^2)$. The consistency condition guarantees that at least one independent variable is computed correctly. Due to the dependency of the local truncation error on constants a_{ij} and b_i the conditions for a given order accuracy are determined. Explicit low-order schemes are presented first [11].

To start with the methods of order 2, methods are already introduced by the work of Runge. This work was extended by Heun (1900) and by Kutta (1901). Heun who completed order 3 methods and started the classification for order 4 method and Kutta finished it. The main idea behind the order of each method is the number of stages s required to achieve this order and the number of computed free parameters for given number of stages, for low-order methods are given in table 2.9. The relationship between those numbers is given by conditions, so-called order conditions. And again we can use the approach of the rooted trees presented earlier and apply it for the order condition description for all classes of Runge–Kutta algorithms [19].

Table 2.9: Numbers of stages to achieve specified order of low-order methods

order p	1	2	3	4	
number of conditions	1	2	4	8	
number of stages	1	2	3	4	5
number of parameters	1	3	6	10	15

For Runge–Kutta explicit method order 1 holds an order condition

$$b_1 + b_2 = 1, \tag{2.44}$$

for second order second order to the condition (2.44) second condition need to be add, it is

$$b_2 a_{21} = \frac{1}{2}, \tag{2.45}$$

for third order needs to hold other two conditions etc. Now we show how to obtain those specific numbers.

At first we need to present the *elementary weight* $\Phi_i(t)$ for stage i where $i = 1, 2, \dots, s + 1$. The elementary weight for the rooted tree τ is defined by

$$\begin{aligned} \Phi_i(\tau) &= c_i, \\ \Phi_i([t_1 t_2 \cdots t_m]) &= \sum_{j=1}^s a_{ij} \Phi_j(t_1) \Phi_j(t_2) \cdots \Phi_j(t_m). \end{aligned}$$

The notation rule for formula $\Phi(t)$ corresponding to the tree says that the root of the tree is labeled by i and the other vertices are labeled by j, k, l, \dots . For each arc write down a factor a_{uv} where u, v are labels at the end of each arc. Insert a further factor b_i and then sum each indexes i, j, k, l, \dots through the numbers $1, 2, \dots, s$. For example, our exemplar tree from figure 2.10 first used in the subsection 2.3, is now represented by the

formula

$$\Phi(t) = \sum_{i,j,k,l,m=1}^s b_i a_{ij} a_{jk} a_{kl} a_{jm}.$$

To simplify it, we use the numbers c_1, c_2, \dots, c_s to give

$$\Phi(t) = \sum_{i,j,k=1}^s b_i a_{ij} a_{jk} c_k c_j$$

by summing over l, m . For a standard Runge–Kutta method of order p for all $t \in T$ with $r(t) \leq p$ holds the equation (2.46) [16].

$$\Phi(t) = \frac{1}{\gamma(t)} \tag{2.46}$$

To generalize the approach of rooted trees in subsection 2.3 with using the elementary weight Φ , we find the order conditions for an arbitrary Runge–Kutta method. The list of order conditions related to trees for explicit Runge–Kutta methods up to order 5 is given in table 2.10, where $\gamma(t)$ was already computed in the previous section.

To illustrate the use of conditions for computing coefficients, we present several methods up to order 4 of various parameters. The method of order 1 and stage 1, which has only one possible variation, is Euler method (already mentioned above). The method is described in table 2.11 by its scheme in the first column, by the Butcher tableau in the second column and we check that the order conditions are satisfied for given coefficients of specific method in the third column.

For second order method we obtain two equations (2.47) which gives us three possible choices for computing coefficients.

$$\begin{aligned} b_1 + b_2 &= 1 \\ b_2 c_2 &= \frac{1}{2} \end{aligned} \tag{2.47}$$

The usual approach is to choose coefficient c_2 and then solve coefficients b_i . According to the system of two equations (2.47) and the condition that $a_{21} = c_2$ we have options for RK2 given by

$$\begin{aligned} 1) \quad c_2 &= \frac{1}{2}, & b_1 &= 0, & b_2 &= 1, \\ 2) \quad c_2 &= 1, & b_1 &= b_2 = \frac{1}{2}, \\ 3) \quad c_2 &= \text{arbitrary}, & b_1 &= 1 - \frac{1}{2c_2}, & b_2 &= \frac{1}{2c_2}. \end{aligned}$$

The second option is called Heun’s method or improved Euler method, more details are described later in the subsection 3.1. For presenting the order conditions we choose the

Table 2.10: Order conditions related to trees for explicit RK methods up to order 5


















t	$r(t) \leq p$	$\Phi(t) = 1/\gamma(t)$
	1	$\sum_{i=1}^s b_i = 1$
	2	$\sum_{i=1}^s b_i c_i = \frac{1}{2}$
	3	$\sum_{i=1}^s b_i c_i^2 = \frac{1}{3}$
	3	$\sum_{i=1}^s \sum_{j=1}^s b_i a_{ij} c_j = \frac{1}{6}$
	4	$\sum_{i=1}^s b_i c_i^3 = \frac{1}{4}$
	4	$\sum_{i=1}^s \sum_{j=1}^s b_i c_i a_{ij} c_j = \frac{1}{8}$
	4	$\sum_{i=1}^s \sum_{j=1}^s b_i a_{ij} c_j^2 = \frac{1}{12}$
	4	$\sum_{i=1}^s \sum_{j=1}^s \sum_{k=1}^s b_i a_{ij} a_{jk} c_k = \frac{1}{24}$
	5	$\sum_{i=1}^s b_i c_i^4 = \frac{1}{5}$
	5	$\sum_{i=1}^s \sum_{j=1}^s b_i c_i^2 a_{ij} c_j = \frac{1}{10}$
	5	$\sum_{i=1}^s \sum_{j=1}^s b_i c_i a_{ij} c_j^2 = \frac{1}{15}$
	5	$\sum_{i=1}^s \sum_{j=1}^s \sum_{k=1}^s b_i c_i a_{ij} a_{jk} c_k = \frac{1}{30}$
	5	$\sum_{i=1}^s b_i \left(\sum_{j=1}^s c_i a_{ij} c_j^2 \right) = \frac{1}{20}$
	5	$\sum_{i=1}^s \sum_{j=1}^s b_i a_{ij} c_j^3 = \frac{1}{20}$
	5	$\sum_{i=1}^s \sum_{j=1}^s \sum_{k=1}^s b_i a_{ij} c_j a_{jk} c_k = \frac{1}{40}$
	5	$\sum_{i=1}^s \sum_{j=1}^s \sum_{k=1}^s b_i a_{ij} a_{jk} c_k^2 = \frac{1}{60}$
	5	$\sum_{i=1}^s \sum_{j=1}^s \sum_{k=1}^s b_i a_{ij} a_{jk} a_{kl} c_l = \frac{1}{120}$

Table 2.11: Runge–Kutta order 1, stage 1

Scheme	B. tableau	Order condition
$y_{n+1} = y_n + hf(y_n)$	$\begin{array}{c c} 0 & 0 \\ \hline & 1 \end{array}$	$b_1 = 1$

Table 2.12: Runge–Kutta order 2, stage 2

Scheme	B. tableau	Order conditions
$y_{n+1} = y_n + \frac{1}{4}h(k_1 + 3k_2)$	$\begin{array}{c cc} 0 & & \\ \hline \frac{2}{3} & & \frac{2}{3} \\ 0 & \frac{1}{4} & \frac{3}{4} \end{array}$	$b_1 + b_2 = \frac{1}{4} + \frac{3}{4} = 1$
$k_1 = f(t_n, y_n)$		$b_2c_2 = \frac{3}{4}\frac{2}{3} = \frac{1}{2}$
$k_2 = f(t_n + \frac{2}{3}h, y_n + \frac{2}{3}hk_1)$		

third option for $c_2 = 2/3$ and we obtain the two-stage Runge–Kutta scheme given by table 2.12.

For the third order method we have four order conditions according to table 2.9 that assumption gives us four equations

$$\begin{aligned} b_1 + b_2 + b_3 &= 1, \\ b_2c_2 + b_3c_3 &= \frac{1}{2}, \\ b_2c_2^2 + b_3c_3^2 &= \frac{1}{3}, \\ b_3a_{32}c_2 &= \frac{1}{6}. \end{aligned} \tag{2.48}$$

Solving the system of equations (2.48) with conditions $a_{21} = c_2$ and $c_3 = a_{31} + a_{32}$ give us several options for coefficients options for RK3 given by

$$\begin{aligned} 1) \quad & c_2 \neq \frac{2}{3}, \quad c_2 \neq c_3, \quad b_1 = 1 - b_2 - b_3, \quad b_2 = \frac{c_2 - \frac{2}{3}}{2c_2(c_3 - c_2)}, \quad b_3 = \frac{\frac{2}{3} - c_2}{2c_2(c_3 - c_2)}, \quad a_{32} = \frac{1}{6b_3c_2}, \\ 2) \quad & c_2 = \frac{2}{3}, \quad c_3 = 0, \quad b_3 \neq 0, \quad b_1 = \frac{1}{4} - b_3, \quad b_2 = \frac{3}{4}, \quad a_{32} = \frac{1}{4b_3}, \\ 3) \quad & c_2 = c_3 = \frac{2}{3}, \quad b_3 \neq 0, \quad b_1 = \frac{1}{4}, \quad b_2 = \frac{3}{4} - b_3, \quad a_{32} = \frac{1}{4b_3}. \end{aligned}$$

If we choose the third option for $b_3 = 3/4$, we obtain the three-stage Runge–Kutta scheme given by table 2.13.

Table 2.13: Runge–Kutta order 3, stage 3

Scheme	B. tableau	Order conditions
$y_{n+1} = y_n + \frac{1}{6}h(k_1 + 4k_2 + k_3)$	$\begin{array}{c ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ 1 & -1 & 2 & \\ \hline & \frac{1}{6} & \frac{4}{6} & \frac{1}{6} \end{array}$	$b_1 + b_2 + b_3 = \frac{1}{6} + \frac{4}{6} + \frac{1}{6} = 1$
$k_1 = f(t_n, y_n)$		$b_2c_2 + b_3c_3 = \frac{4}{6}\frac{1}{2} + \frac{1}{6}1 = \frac{1}{2}$
$k_2 = f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1)$		$b_2c_2^2 + b_3c_3^2 = \frac{4}{6}\frac{1}{4} + \frac{1}{6} = \frac{1}{3}$
$k_3 = f(t_n + h, y_n - hk_1 + 2hk_2)$		$b_3a_{32}c_2 = \frac{1}{6}2\frac{1}{2} = \frac{1}{6}$

For fourth order method we have eight order conditions according to table 2.9. That gives us eight equations for computing coefficients

$$\begin{aligned}
 b_1 + b_2 + b_3 + b_4 &= 1 \\
 b_2c_2 + b_3c_3 + b_4c_4 &= \frac{1}{2} \\
 b_2c_2^2 + b_3c_3^2 + b_4c_4^2 &= \frac{1}{3} \\
 b_3a_{32}c_2 + b_4a_{42}c_2 + b_4a_{43}c_3 &= \frac{1}{6} \\
 b_2c_2^3 + b_3c_3^3 + b_4c_4^3 &= \frac{1}{4} \\
 b_3c_3a_{32}c_2 + b_4c_4a_{42}c_2 + b_4c_4a_{43}c_3 &= \frac{1}{8} \\
 b_3a_{32}c_2^2 + b_4a_{42}c_2^2 + b_4a_{43}c_3^2 &= \frac{1}{12} \\
 b_4a_{43}a_{32}c_2 &= \frac{1}{24}
 \end{aligned} \tag{2.49}$$

Finding results for given system of equations is more complicated, the problem lays in the last equation. But it was derived and proved that condition for $c_4 = 1$ gives the famous RK4, which was classified by Kutta, see table 2.14. Other options are presented as

$$\begin{aligned}
 1) \quad c_2 = c_3 = a_{21} = \frac{1}{2}, \quad b_3 \neq 0 : \\
 b_1 = \frac{1}{6}, \quad b_2 = \frac{2}{3} - b_3, \quad b_4 = \frac{1}{6}, \quad a_{31} = \frac{3b_3 - 1}{6b_3}, \quad a_{32} = \frac{1}{6b_3}, \\
 a_{41} = 0, \quad a_{42} = 1 - 3b_3, \quad a_{43} = 3b_3,
 \end{aligned}$$

$$2) \quad c_2 = a_{21} \neq 0, \quad c_3 = \frac{1}{2} : \\ b_1 = \frac{1}{6}, \quad b_2 = 0, \quad b_3 = \frac{2}{3}, \quad b_4 = \frac{1}{6}, \quad a_{31} = \frac{4c_2 - 1}{8c_2}, \quad a_{32} = \frac{1}{8c_2}, \\ a_{41} = \frac{1 - 2c_2}{2c_2}, \quad a_{42} = -\frac{1}{2c_2}, \quad a_{43} = 2,$$

$$3) \quad c_2 = a_{21} = \frac{1}{2}, \quad c_3 = 0, \quad b_3 \neq 0 : \\ b_1 = \frac{1}{6} - b_3, \quad b_2 = \frac{2}{3}, \quad b_4 = \frac{1}{6}, \quad a_{31} = -\frac{1}{12b_3}, \quad a_{32} = \frac{1}{12b_3}, \\ a_{41} = -\frac{1}{2} - 6b_3, \quad a_{42} = \frac{3}{2}, \quad a_{43} = 6b_3,$$

$$4) \quad c_2 = a_{21} = 1, \quad c_3 = \frac{1}{2}, \quad b_4 \neq 0 : \\ b_1 = \frac{1}{6}, \quad b_2 = \frac{1}{6} - b_4, \quad b_3 = \frac{2}{3}, \quad a_{31} = \frac{3}{8}, \quad a_{32} = \frac{1}{8}, \\ a_{41} = 1 - \frac{1}{4b_4}, \quad a_{42} = -\frac{1}{12b_4}, \quad a_{43} = \frac{1}{3b_4},$$

$$5) \quad c_2, c_3, 0, 1 \text{ all distinct, } c_2 \neq \frac{1}{2} \text{ and } 3 - 4(c_2 + c_3) + 6c_2c_3 \neq 0 :$$

$$b_1 = \frac{1 - 2(c_2 + c_3) + 6c_2c_3}{12c_2c_3}, \quad b_2 = \frac{2c_3 - 1}{12c_2(c_3 - c_2)(1 - c_2)}, \\ b_3 = \frac{1 - 2c_3}{12c_2(c_3 - c_2)(1 - c_2)}, \quad b_4 = \frac{3 - 4(c_2 + c_3) + 6c_2c_3}{12(1 - c_2)(1 - c_3)}, \\ a_{21} = c_2, \quad a_{41} = \frac{c_2^3(12c_2^2 - 12c_2 + 4) - c_3(12c_2^2 - 15c_2 + 5) + 4c_2^2 - 6c_2 + 2}{2c_2c_3[3 - 4(c_2 + c_3) + 6c_2c_3]}, \\ a_{42} = \frac{(-4c_3^2 + 5c_3 + c_2 - 2)(1 - c_2)}{2c_2(c_3 - c_2)(3 - 4(c_2 + c_3) + 6c_2c_3)}, \quad a_{43} = \frac{(1 - 2c_2)(1 - c_3)(1 - c_2)}{c_3(c_3 - c_2)(3 - 4(c_2 + c_3) + 6c_2c_3)}.$$

If we choose the option 1 with $b_3 = 1/3$, we get very known RK4, see table 2.14.

The Runge–Kutta methods are divided into two groups according to the number of stages for a reason. We start with a theorem for s -stage method that the order of the method cannot exceeds s . It also holds that for low-order Runge–Kutta methods ($p \leq 4$) the greatest achievable order of the s -stage method is equal to s . It was determined and proved, e. g. by Butcher (1987). We are interested in the high-order Runge–Kutta methods. Which order can be reached by s -stage method? It has been shown in the table 2.9 that the order 4 can be reached by 4 stages and 5 stages Runge–Kutta methods. The theorem holds that there is no p -stage p th-order method with $p \geq 5$. Hence, there are determined bounds for explicit Runge–Kutta methods high-order [16].

For a given number p there exists a method of order p with s_1 stages where

$$s_1 = \frac{p^2 - 7p + 20}{2}$$

Table 2.14: Runge–Kutta order 4, stage 4

Scheme	B. tableau	Order conditions
$y_{n+1} = y_n + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4)$ $k_1 = f(t_n, y_n)$ $k_2 = f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1)$ $k_3 = f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_2)$ $k_4 = f(t_n + h, y_n + hk_3)$	$ \begin{array}{c ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ \frac{1}{2} & 0 & \frac{1}{2} & \\ 1 & 0 & 0 & 1 \\ \hline & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array} $	$\frac{1}{6} + \frac{2}{6} + \frac{2}{6} + \frac{1}{6} = 1$
		$\frac{2}{6}\frac{1}{2} + \frac{2}{6}\frac{1}{2} + \frac{1}{6} = \frac{1}{2}$
		$\frac{1}{12} + \frac{1}{12} + \frac{1}{6} = \frac{1}{3}$
		$\frac{1}{12} + 0 + \frac{1}{12} = \frac{1}{6}$
		$\frac{1}{24} + \frac{1}{24} + \frac{1}{6} = \frac{1}{4}$ $\frac{1}{24} + 0 + \frac{1}{12} = \frac{1}{8}$ $\frac{1}{24} + 0 + \frac{1}{24} = \frac{1}{12}$

and if $p \geq 10$ the method exists with the required order with

$$s_2 = \frac{p^2 - 7p + 10}{2}$$

stages [35].

To fulfill the theory for high-order explicit methods numbers of orders and stages are presented in table 2.15. Notice that for order 5 method 17 order conditions to determine coefficients need to be accomplished, for method of order 6 there are 37 conditions etc.

Table 2.15: Numbers of values for specific order of high-order methods

order p	5	6	7	8		
number of conditions	17	37	85	200		
number of stages	6	7	8	9	10	11
number of parameters	21	28	36	45	55	66

We present conditions only for the method of order 5 that means we add another 9 conditions to conditions already mentioned above such as

$$\begin{aligned}
\sum_{i=1}^s b_i c_i^4 &= \frac{1}{5} \\
\sum_{i=1}^s \sum_{j=1}^s b_i c_i^2 a_{ij} c_j &= \frac{1}{10} \\
\sum_{i=1}^s \sum_{j=1}^s b_i c_i a_{ij} c_j^2 &= \frac{1}{15} \\
\sum_{i=1}^s \sum_{j=1}^s \sum_{k=1}^s b_i c_i a_{ij} a_{jk} c_k &= \frac{1}{30} \\
\sum_{i=1}^s \sum_{j=1}^s \sum_{k=1}^s b_i a_{ij} c_j a_{ik} c_k &= \frac{1}{20} \\
\sum_{i=1}^s \sum_{j=1}^s b_i a_{ij} c_j^3 &= \frac{1}{20} \\
\sum_{i=1}^s \sum_{j=1}^s \sum_{k=1}^s b_i a_{ij} c_j a_{jk} c_k &= \frac{1}{40} \\
\sum_{i=1}^s \sum_{j=1}^s \sum_{k=1}^s b_i a_{ij} a_{jk} c_k^4 &= \frac{1}{60} \\
\sum_{i=1}^s \sum_{j=1}^s \sum_{k=1}^s \sum_{l=1}^s b_i a_{ij} a_{jk} a_{kl} c_l &= \frac{1}{120}
\end{aligned}$$

There are known numbers of stages and orders for the method order 9 and 10. The lower bound for order 9 is the number of 12 stages and upper bound should accomplish number of 17 stages, to reach the order 10 we need at least 13 stages but no more than 17 stages. The number of conditions for the high-dimensional general problem to have order p is equal to the number of rooted-trees with less than or equal to p -vertices.

Bounds of explicit methods, also called as barriers, say that p -order Runge–Kutta method requires more than p -stages if $p > 4$. According to the fact solving order conditions becoming more difficult for higher p . Hence, here arise the *implicit Runge–Kutta methods*. Implicit Runge–Kutta method is given by same the formula as explicit Runge–Kutta method (2.42) but defining internal approximations as

$$Y_i = y_{n-1} + h \sum_{j=1}^s a_{ij} f(Y_j), \quad i = 1, 2, \dots, s \quad (2.50)$$

with coefficients $a_{ij} = 0$ for all $i \geq j$. The implicit Runge–Kutta methods can be sorted out to different families. There are Gauss, Radau and Lobatto families. A special case with $a_{ij} = 0$ for $i > j$ but with at least one of a_{ii} is non-zero, lays somewhere between explicit and implicit are called semi-implicit RK methods. Other groups are *singly implicit*

Runge-Kutta methods (SIRK, [23]) and *diagonally implicit Runge-Kutta methods* (DIRK, [1]). SIRK can be generalized into *diagonally extended singly implicit Runge-Kutta methods* using efficient order (DESIRE, [25]) and *diagonally-implicit multi-stage integration methods* (DIMSIM, [17]).

Stability analysis

To illustrate the analysis of the grown of numerical errors in a computed solution to a differential equation, we consider the equation (2.35) again as in Euler method stability analysis. As we write $hq = z$ the analysis generalizes in the case of explicit Runge-Kutta methods to give a result y_n computed after n steps from $y(0) = 1$. The result is given by $y_n = r(z)^n$. The r is a particular polynomial determined by the coefficients in the method. In the case of implicit Runge-Kutta methods, r is not in general a polynomial but a rational function.

A Runge-Kutta method is said to be A -stable if its stability region contains C^- , the non-positive half-plane. This definition has been redefined in different ways during the time. More requirements on the qualitative behaviour of numerical solutions were proposed. Let us introduce some of the requirements. One of them is that a method to be such that $|r(z)| \leq 1$ for all C^- and in addition that $\lim_{|z| \rightarrow \infty} |r(z)| = 0$ and it is known as a L -stability. Quite standard requirement of A -stability is that the stability region include the set $C(\alpha) = z \in C : |\arg(-z)| \leq \alpha$ and the stability region contains some left half-plane together with the intersection of the negative half-plane with some open set containing the real axis. This properties was named $A(\alpha)$ -stability (Widlund, 1967) and later named as stiff stability (Gear, 1969) [47, 104].

The requirements which refers to the qualitative behaviour of numerical solutions to certain non-linear problems are given by B -stability (Butcher, 1975). The property says that for two particular solutions to such a problem the difference between them is non-increasing and could be applied to numerical solution. This property can be considered also for non-autonomous differential equations and the method preserves it is called BN -stable (Burrage and Butcher, 1979) [12, 22].

Consider a Runge-Kutta method given by

$$\begin{aligned} Y_1 &= y_{n-1}, \\ Y_2 &= y_{n-1} + ha_{21}f(Y_1), \\ &\dots \\ Y_s &= y_{n-1} + h\left(a_{s1}f(Y_1) + a_{s2}f(Y_2) + \dots + a_{s,s-1}f(Y_{s-1})\right), \\ y_n &= y_{n-1} + h\left(b_1f(Y_1) + b_2f(Y_2) + \dots + b_sf(Y_s)\right) \end{aligned}$$

using the Dahlquist problem (2.31), $z = hq$ and s the number of stages. We rewrite it as

$$\begin{aligned} Y &= y_{n-1}e + zAY, \\ y_n &= y_{n-1} + zb^T Y, \end{aligned}$$

where $e = [1, 1, \dots, 1]^T$, $Y = [Y_1, Y_2, \dots, Y_s]^T$ and $b^T = [b_1, b_2, \dots, b_s]$.

The polynomial r which determines the stability of the method is given by

$$R(z) = \frac{y_n}{y_{n-1}} = 1 + zb^T(y_{n-1}^{-1}Y).$$

Due to some assumptions [16], we find

$$R(z) = 1 + z + \frac{z^2}{2!} + \dots + \frac{z^p}{p!} + c_{p+1}z^{p+1}, \quad (2.51)$$

A method is said to be A -stable if its stability function is bounded by 1 in the left half-plane. It is said to be L -stable if it is A -stable and $R(\infty) = 0$. A method of order p has a stability function with a series that agrees with e^z up to terms in h^p [21]. Hence we obtain the stability regions described in the table 2.16 and plotted in the picture 2.12 [13].

Table 2.16: Stability functions for Runge–Kutta methods up to order 4

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

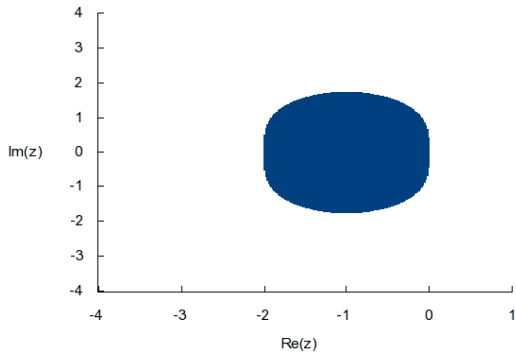
Notice that stability regions up to order 4 are same for Runge–Kutta method and for Taylor series method. The A -stability of the implicit algorithms can be tested by the same procedure as the explicit methods.

The stability function for s -stage implicit Runge–Kutta method is a rational function

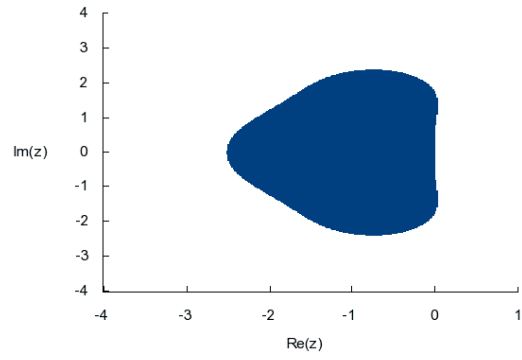
$$R(z) = 1 + zb^T(I - zA)^{-1}e,$$

where $z = \lambda h$ and e is the s -vector $e = (1, 1, \dots, 1)^T$ [13].

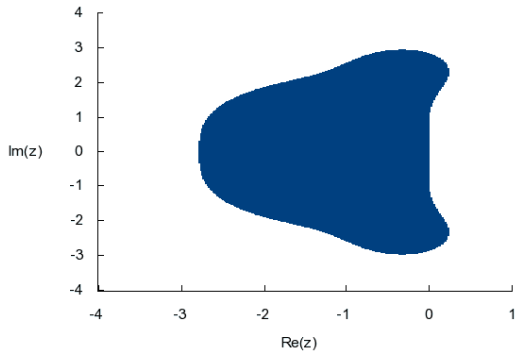
The disadvantages of the explicit Runge–Kutta methods include relatively large number of function evaluations at every integration step. A p -th order explicit formula requires at least p function evaluation per integration step, whereas a corresponding Adams method will require one function evaluation per step, but usually with more overhead costs and smaller stepsizes. The second disadvantage lays in the relatively small interval of absolute



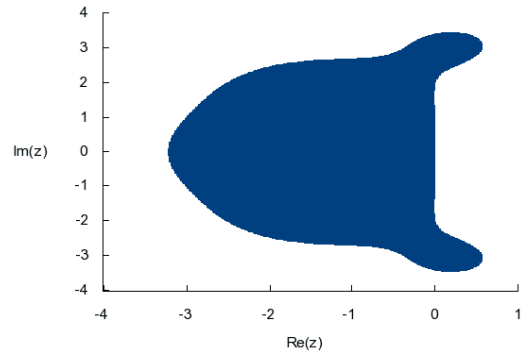
a) second order



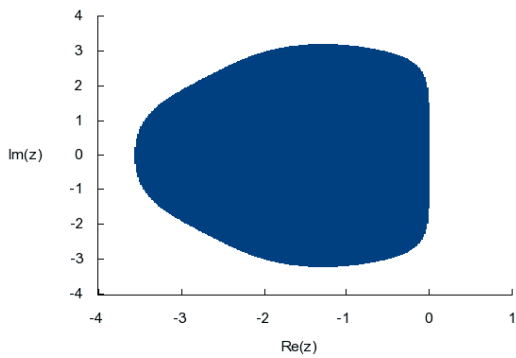
b) third order



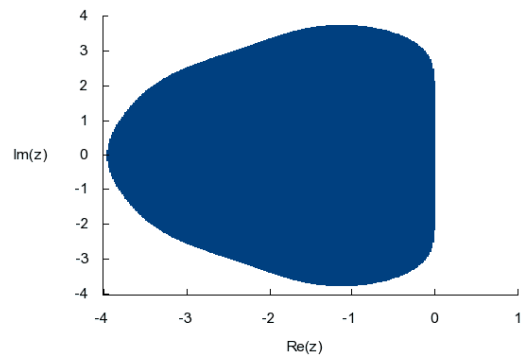
c) fourth order



d) fifth order



e) sixth order



f) seventh order

Figure 2.12: Stability regions of Runge–Kutta method up to order 7

stability render them unsuitable for stiff problems. These disadvantages can be handled with Adams methods such as with classes of implicit and semi-implicit methods proposed by different people (Butcher, Cash, Nørsett, Rosenbrock etc.) [26, 57].

2.5 Taylor series method

Substituting derivations and initial values into the formula for the Taylor polynomial (2.37), we then obtain a representation of the solution as a power series about the initial point x_0 . This procedure, called the Taylor series method, is illustrated of power series (2.52). The mathematical background was widely described in the history [4, 5, 75].

$$y_n = y_{n-1} + hy'_{n-1} + \frac{h^2}{2!}y''_{n-1} + \dots + \frac{h^p}{p!}y^{(p)}_{n-1} + O(h^{p+1}) \quad (2.52)$$

The method has been implemented in simulators TKSL/386, TKSL/C (Kunovsky, 1991, 1998) with different approach than the approach brought by Barton, Willers and Zahar [4]. It uses so-called forming differential equations which implement higher orders more effectively. The Taylor series method can be used for solving a large number of various problems and it has an automatic integration method using Taylor series. It could be used for variable order; the order p is set automatically using as many Taylor series terms for computing as needed to achieve the required accuracy.

The absolute value of the relative error of the computation is the main criterion to choose the order. Maximum order of this method is computed up to 63 of Taylor series terms. The advantage is in the speed of computation, that is functions are generated by adding, multiplying and superposing elementary functions. The disadvantage of the method is the need to generate higher derivatives.

We again present the example of RLC electrical circuit (2.22) as a first test problem to show the power of Taylor series method. We focus on the numerical solution of the circuit and we compare it with the analytical solution. We have same constants and we solve the circuit numerically using differential equations. The numerical solution is plotted in the graph 2.13. The numerical solution of u_C is labelled as u_Cnumer in the graph. When we compare the numerical solution u_C to an analytical solution $u_Canalyt$ we get very small numbers of the error around values 10^{-17} . Hence, the Taylor series method proves high accuracy of calculation. For those interested in specific equations TKSL/C source code is given in the Appendix B.

As second test problem we present *Kepler problem* also known as one-body problem which describes the motion of a single planet around a heavy sun. The problem is given by

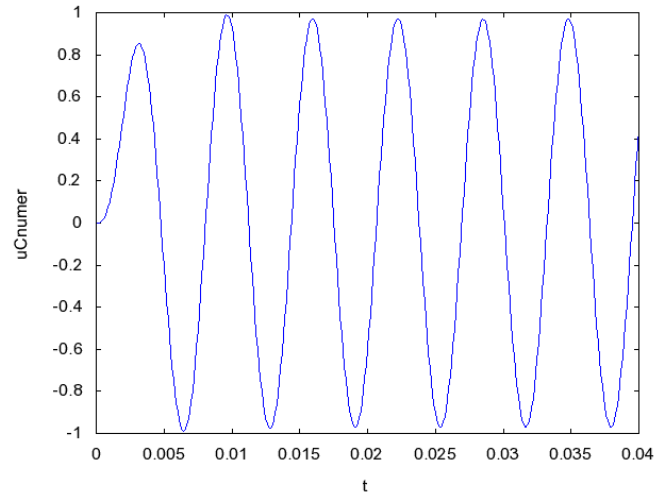


Figure 2.13: RLC circuit (2.22) - numerical solution and error

the system of differential equations

$$\begin{aligned}
 y_1' &= y_3, \\
 y_2' &= y_4, \\
 y_3' &= \frac{-y_1}{r^3}, \\
 y_4' &= \frac{-y_2}{r^3}, \quad r = \sqrt{y_1^2 + y_2^2}.
 \end{aligned} \tag{2.53}$$

We are already assuming that if the M denotes the mass of the sun, γ the gravitational constant and m the mass of the planet, the attractive force on the planet with the magnitude

$$\frac{\gamma M m}{y_1^2 + y_2^2}$$

can be removed from the given system (2.53) by adjusting the scales of the variables. The results of the system are known as ellipses, parabolas or hyperbolas, if we ignore the possibility that the trajectory is a straight line directed either towards or away from the sun. The initial values have been derived as

$$\begin{aligned}
 y_1 &= 1 - e, \\
 y_2 &= 0, \\
 y_3 &= 0, \\
 y_4 &= \sqrt{\frac{1+e}{1-e}},
 \end{aligned} \tag{2.54}$$

where e is the eccentricity of an ellipse on which the orbit lies [18]. The solution of the Kepler problem is described by that all points on the trajectory lie on the ellipse

$$(y_1 + e)^2 + \frac{y_2^2}{1 - e^2} = 1$$

with centre $(-e, 0)$, eccentricity e and major and minor axis lengths 1 and $\sqrt{1 - e^2}$ respectively [21]. The case of $e = 0.75$ is described in figure 2.14.

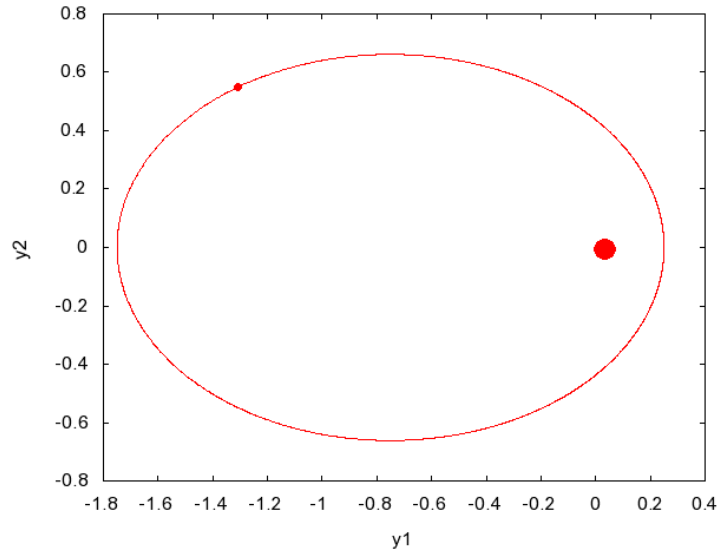
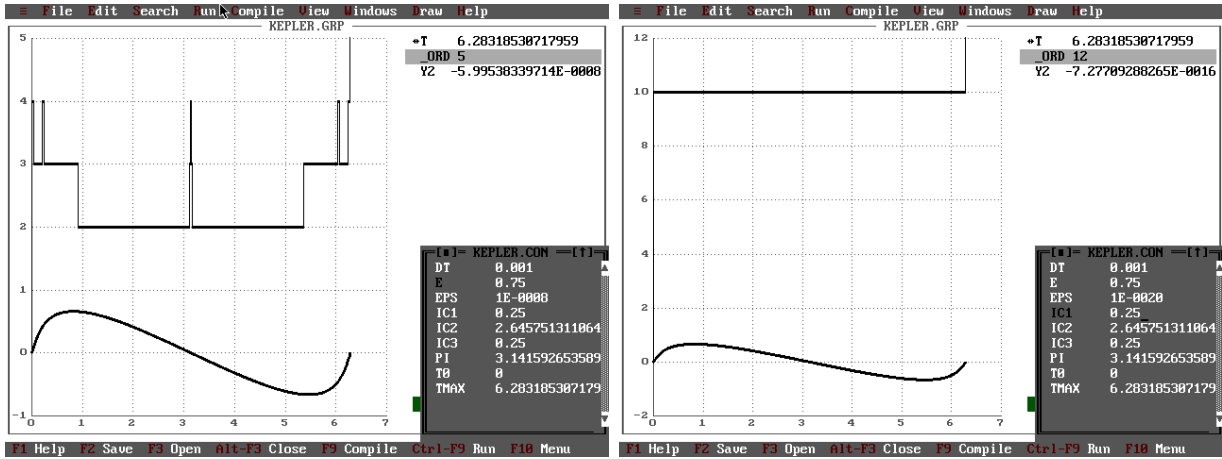


Figure 2.14: Solution of Kepler problem

We demonstrate the variable order setting of Taylor series method by solving the Kepler problem in program TKSL/386. The program automatically uses variable stepsize and variable order via number of used Taylor series terms during the calculation to satisfy the required error tolerance given by the user. Kepler problem is specified by stepsize $DT = 0.001$ and the eccentricity $e = 0.75$. The maximum timesteps of computation is specified for $T = 2\pi$ so we can easily check the accuracy from the graph and plotted values at the end of the calculation. At first we force the program to use the maximum order 5, see picture 2.15a where time is plotted on the x -axis and variable $Y2$ and used order ORD are plotted on the y -axis. We observe that the plotted value $Y2$ corresponding to unknown value y_2 in the system of equations (2.53) is very close to the exact value. Value y_2 should be at $t = 2\pi$ equals to 0, the global error is $EPS = 5.99538339714e-08$ in this case.

If we increase the maximum order to 10, the global error is even smaller. From the graph 2.15b we see $EPS = 7.27709288265e-16$ and t is specified on x -axis, $Y2$ and ORD on y -axis. Notice that in case a) the number of Taylor series terms differs during the calculation, but in case b) the program is forced to use tenth terms of the Taylor series and it uses the maximum order as a minimum order. It is linked with the efficiency of the computation. The calculation specified by chosen parameters is very time demanding. The program TKSL/386 is implemented to achieve the best accuracy possible.

We focus on choosing the stepsize in the second experiment. Implementation of Taylor series method uses the method of halving the stepsize. Experiment with Kepler problem



a) fifth order

b) tenth order

Figure 2.15: Different orders of the Taylor series method for Kepler problem

specified by the eccentricity $e = 0.75$, maximum time interval $T = 50\pi$ and starting value of the stepsize $DT = 1$ is given by the graph 2.16. The method of halving the stepsize seems to be quite effective. The accuracy of the calculation is achieved.

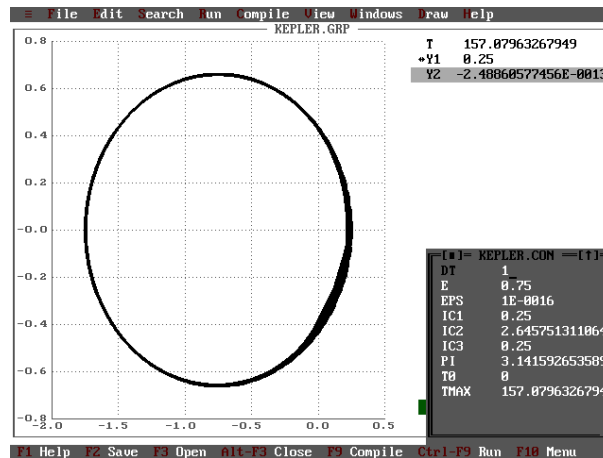


Figure 2.16: Taylor series method for Kepler problem with a computing time $t = 50\pi$

It has been shown that generally the method is A-stable [102]. Stability regions for different numbers of Taylor series terms are shown in the section 2.4, where Runge–Kutta methods were presented, in figure 2.12.

Dahlquist problem (2.31) was chosen for next experiment to check the order of the Taylor series method. Errors of Dahlquist problem with fixed stepsize are displayed in table 2.17. The errors of Euler method was also added into the table for comparison. The order of Euler method and orders of Taylor series method are displayed in the graph 2.17.

The Taylor series method applies to nonlinear as well as linear equations. One dis-

Table 2.17: Errors of Euler method and Taylor series method for Dahlquist problem

h	err_{Euler}	err_{Ts2}	ratio	err_{Ts4}	ratio
0.1	0.1245394	4.200982e-03		2.084324e-06	
$0.1 \cdot 2^{-1}$	6.498412e-02	1.090774e-03	3.851	1.358027e-07	15.348
$0.1 \cdot 2^{-2}$	3.321799e-02	2.778841e-04	3.925	8.666185e-09	15.670
$0.1 \cdot 2^{-3}$	1.679689e-02	7.012736e-05	3.963	5.473053e-10	15.834
$0.1 \cdot 2^{-4}$	8.446252e-03	1.761434e-05	3.981	3.438494e-11	15.917
$0.1 \cdot 2^{-5}$	4.235185e-03	4.413927e-06	3.991	2.155165e-12	15.955
$0.1 \cdot 2^{-6}$	2.120621e-03	1.104776e-06	3.995	1.350031e-13	15.964
$0.1 \cdot 2^{-7}$	1.061069e-03	2.763559e-07	3.997	4.884981e-15	27.636

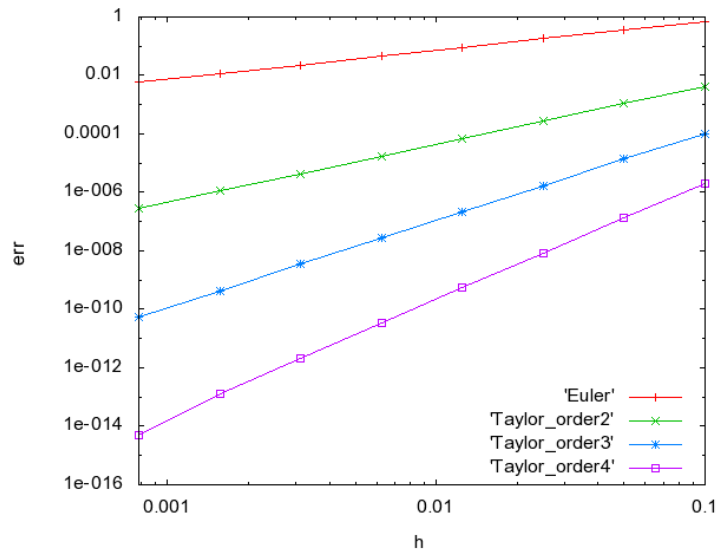


Figure 2.17: Orders of Euler method and Taylor series method for Dahlquist problem

advantage of the Taylor series method is that by computing finitely many terms of the Taylor expansion, one still has no way of knowing the radius of convergence of the series. Fortunately, when the differential equation is linear, there are existence theorems that give a minimum value for its radius [83].

In this chapter the techniques for solving problems by using the single step with one or more computing and functional evaluating in a step were introduced. The next chapter will continue in the generalization in such a way to bring more previous values such as the value y_n depends not only on y_{n-1} and $f(y_{n-1})$ but also on y_{n-2} and $f(y_{n-2})$, y_{n-3} and $f(y_{n-3}), \dots$

Chapter 3

Multistep methods

Generally, the method is called k -step method. To this group of methods belong linear multistep methods, which will be presented in the section 3.2 and their combinations to predictor–corrector pairs, which are described in the section 3.3.

3.1 Improved Euler method

We present the first step of generalization from one-step methods. Let us introduce the method called *improved Euler method*. Given an initial value problem (2.25) the improved Euler method with stepsize h consists in applying the iterative formulae (3.1). The procedure of this method is following; as a first step we evaluate the function $f(x_{n-1}, y_{n-1})$ and we use it to compute the predicted value of y noted as y_n^* . In the second step we use this precomputed value of the solution for evaluating f -function in x_n . Last step is to correct the result value using the new function evaluation. Equations which describe the predict-evaluate-correct process are given by

$$\begin{aligned}F_1 &= f(x_{n-1}, y_{n-1}) \\y_n^* &= y_{n-1} + hF_1 \\F_2 &= f(x_n, y_n^*) \\y_n &= y_{n-1} + \frac{1}{2}h(F_1 + F_2)\end{aligned}\tag{3.1}$$

The method computes successive approximations y_1, y_2, y_3, \dots to the values $y(x_1), y(x_2), y(x_3), \dots$ of the (exact) solution $y = y(x)$ at the points x_1, x_2, x_3, \dots respectively [41].

The improved Euler method is one of a class of numerical techniques known as predictor–corrector methods. First a predictor y_n^* of the next y -value is computed; then it is used to correct itself in y_n . If we rewrite it, the improved Euler method with stepsize h consists in using predictor

$$y_n^* = y_n + hf(x_{n-1}, y_{n-1})\tag{3.2}$$

and the corrector

$$y_n = y_{n-1} + \frac{1}{2}h \left(f(x_{n-1}, y_{n-1}) + f(x_n, y_n^*) \right). \quad (3.3)$$

The stopping rule may be controlled either by comparing the difference of both formulae to chosen tolerance or by predetermining the number of iterations [101]. More information about predictor–corrector pairs are presented later in section 3.3.

3.2 Linear multistep methods

The linear multistep method is essentially a polynomial interpolation procedure whereby the solution or its derivative is replaced by a polynomial of appropriate degree in the independent variable x , whose derivative or integral is readily computed. The linear multistep method for the initial value problem is given by

$$y_n = \sum_{i=1}^k \alpha_i y_{n-i} + h \sum_{i=0}^k \beta_i f(y_{n-i}, y_{n-i}). \quad (3.4)$$

According to the coefficient b_0 one separates methods into *Gear methods* and *Adams methods*: explicit *Adams–Bashforth* ($b_0 = 0$) and implicit *Adams–Moulton* ($b_0 \neq 0$).

To construct arbitrary linear multistep methods we use order conditions. We replace the values and their derivatives on the right-hand side of equation (3.4) by the exact values and apply Taylor series expansions about the point $(x_n, y(x_n))$ [55].

$$\begin{aligned} & a_1 y(x_n - h) + a_2 y(x_n - 2h) + \dots + a_k y(x_n - kh) + h \left(b_0 y'(x_n) + b_1 y'(x_n - h) + \dots + b_k y'(x_n - kh) \right) \\ &= a_1 \left(y(x_n) - h y'(x_n) + \frac{h^2}{2} y''(x_n) - \frac{h^3}{6} y'''(x_n) + \frac{h^4}{24} y^{(4)}(x_n) - \dots \right) \\ & \quad + a_2 \left(y(x_n) - 2h y'(x_n) + \frac{4h^2}{2} y''(x_n) - \frac{8h^3}{6} y'''(x_n) + \frac{16h^4}{24} y^{(4)}(x_n) - \dots \right) \\ & \quad \vdots \\ & \quad + a_k \left(y(x_n) - kh y'(x_n) + \frac{k^2 h^2}{2} y''(x_n) - \frac{k^3 h^3}{6} y'''(x_n) + \frac{k^4 h^4}{24} y^{(4)}(x_n) - \dots \right) \\ & \quad + h b_0 y'(x_n) \\ & \quad + h b_1 \left(y'(x_n) - h y''(x_n) + \frac{h^2}{2} y''(x_n) - \frac{h^3}{6} y'''(x_n) + \frac{h^4}{24} y^{(4)}(x_n) - \dots \right) \\ & \quad + h b_2 \left(y'(x_n) - 2h y''(x_n) + \frac{4h^2}{2} y''(x_n) - \frac{8h^3}{6} y'''(x_n) + \frac{16h^4}{24} y^{(4)}(x_n) - \dots \right) \\ & \quad \vdots \\ & \quad + h b_k \left(y'(x_n) - kh y''(x_n) + \frac{k^2 h^2}{2} y''(x_n) - \frac{k^3 h^3}{6} y'''(x_n) + \frac{k^4 h^4}{24} y^{(4)}(x_n) - \dots \right) \end{aligned}$$

$$\begin{aligned}
& a_1y(x_n - h) + a_2y(x_n - 2h) + \cdots + a_ky(x_n - kh) + h\left(b_0y'(x_n) + b_1y'(x_n - h) + \cdots + b_ky'(x_n - kh)\right) \\
& = y(x_n)(a_1 + a_2 + \cdots + a_k) \\
& \quad - hy'(x_n)(a_1 + 2a_2 + \cdots + ka_k - b_0 - b_1 - b_2 - \cdots - b_k) \\
& \quad + \frac{h^2}{2}y''(x_n)(a_1 + 4a_2 + \cdots + k^2a_k - 2b_1 - 4b_2 - \cdots - 2kb_k) \\
& \quad \vdots \\
& \quad + \frac{h^k}{k!}y^{(k)}(x_n)\left(a_1 + 2^ka_2 + \cdots + k^ka_k - pb_1 - 2^{(p-1)}pb_2 - \cdots - k^{(p-1)}pb_k\right)
\end{aligned} \tag{3.5}$$

If we equate the left-hand side of equation (3.5) to zero, we find the following order conditions and we get order conditions for linear multistep method.

The linear multistep method is of order m if

$$\begin{aligned}
& a_1 + a_2 + \cdots + a_k = 1, \\
\text{Order 1 :} & \quad a_1 + 2a_2 + \cdots + ka_k = b_0 + b_1 + b_2 + \cdots + b_k, \\
\text{Order 2 :} & \quad a_1 + 2^2a_2 + \cdots + k^2a_k = 2(b_0 + 2b_1 + 3b_2 + \cdots + kb_k), \\
\text{Order 3 :} & \quad a_1 + 2^3a_2 + \cdots + k^3a_k = 3(b_0 + 4b_1 + 8b_2 + \cdots + k^2b_k), \\
\text{Order 4 :} & \quad a_1 + 2^4a_2 + \cdots + k^4a_k = 4(b_0 + 8b_1 + 27b_2 + \cdots + k^3b_k), \\
& \quad \vdots \\
\text{Order } p : & \quad a_1 + 2^pa_2 + \cdots + k^pa_k = p(b_0 + 2^{(p-1)}b_1 + 3^{(p-1)}b_2 + \cdots + k^{(p-1)}b_k).
\end{aligned} \tag{3.6}$$

The two-step explicit method can attain order 3 while the two-step implicit method can attain order 4.

Gear methods, especially Gear formula also called *backward differentiation formula*, have a great importance within the multistep methods. The conditions are $p = k - 1$ and $b_0 = b_1 = \cdots = b_{k-1} = 0$. Using the following equations (3.7) we are able to compute the coefficients for Gear formula up to order 4, see Table 3.1 [50].

$$\begin{aligned}
& \sum_{i=0}^p a_i = 1 \\
& \sum_{i=1}^p (-i)^j a_i + jb_{-1} = 1, \quad j = 1, \cdots, k
\end{aligned} \tag{3.7}$$

There is not a stable second order integration method than the Gear's method of second order. Only implicit Gear methods with order $k \leq 6$ are zero stable.

Adams–Bashforth method is an explicit multistep method whence

$$p = k - 1, \quad a_1 = a_2 = \cdots = a_{k-1} = 0, \quad b_{-1} = 0$$

Table 3.1: Gear method's coefficients up to $k = 5$ steps

k	b_{-1}	a_0	a_1	a_2	a_3
2	1	1			
3	$\frac{2}{3}$	$\frac{4}{3}$	$-\frac{1}{3}$		
4	$\frac{6}{11}$	$\frac{18}{11}$	$-\frac{9}{11}$	$\frac{2}{11}$	
5	$\frac{12}{25}$	$\frac{48}{25}$	$-\frac{36}{25}$	$\frac{16}{25}$	$-\frac{3}{25}$

defined by

$$y_n = y_{n-1} + h(a_1 f_{n-1} + a_2 f_{n-2} + \cdots + b_k f_{n-k}).$$

The coefficients a_k (see Tab. 3.2) can be determined from (3.6) and rewritten also by

$$j \sum_{i=0}^p (-i)^{j-1} b_i = 1, \quad j = 1, \dots, k \quad (3.8)$$

The Adams-Bashforth formula of order 1 for $k = 1$ yields the (explicit) Euler method, see table 3.2.

For example, Adams-Bashforth's coefficients of order 6 ($p = 6$, $k = 7$) are derived by

$$j \sum_{i=0}^7 (-i)^{j-1} b_i = 1, \quad j = 1, 2, \dots, 7$$

which gives us

$$\begin{aligned} 1[(-0)^0 b_0 + (-1)^0 b_1 + (-2)^0 b_2 + (-3)^0 b_3 + (-4)^0 b_4 + (-5)^0 b_5 + (-6)^0 b_6] &= 1 \\ 2[(-0)^1 b_0 + (-1)^1 b_1 + (-2)^1 b_2 + (-3)^1 b_3 + (-4)^1 b_4 + (-5)^1 b_5 + (-6)^1 b_6] &= 1 \\ 3[(-0)^2 b_0 + (-1)^2 b_1 + (-2)^2 b_2 + (-3)^2 b_3 + (-4)^2 b_4 + (-5)^2 b_5 + (-6)^2 b_6] &= 1 \\ 4[(-0)^3 b_0 + (-1)^3 b_1 + (-2)^3 b_2 + (-3)^3 b_3 + (-4)^3 b_4 + (-5)^3 b_5 + (-6)^3 b_6] &= 1 \\ 5[(-0)^4 b_0 + (-1)^4 b_1 + (-2)^4 b_2 + (-3)^4 b_3 + (-4)^4 b_4 + (-5)^4 b_5 + (-6)^4 b_6] &= 1 \\ 6[(-0)^5 b_0 + (-1)^5 b_1 + (-2)^5 b_2 + (-3)^5 b_3 + (-4)^5 b_4 + (-5)^5 b_5 + (-6)^5 b_6] &= 1 \\ 7[(-0)^6 b_0 + (-1)^6 b_1 + (-2)^6 b_2 + (-3)^6 b_3 + (-4)^6 b_4 + (-5)^6 b_5 + (-6)^6 b_6] &= 1 \end{aligned}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & -2 & -4 & -6 & -8 & -10 & -12 \\ 0 & 0 & 3 & 12 & 27 & 48 & 75 & 108 \\ 0 & 0 & -4 & -32 & -108 & -256 & -500 & -864 \\ 0 & 0 & 5 & 80 & 405 & 1280 & 3125 & 6480 \\ 0 & 0 & -6 & -192 & -1458 & -6144 & -1875 & -46656 \\ 0 & 0 & 7 & 448 & 5103 & 28672 & 109375 & 326592 \end{bmatrix} \begin{bmatrix} a_7 \\ b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

and the result for Adams-Bashforth method order $p = 6$, $k = 7$ is

$$y_n = y_{n-1} + h \left(\frac{28369}{8634} f_{n-1} - \frac{5421}{733} f_{n-2} + \frac{46083}{395} f_{n-3} - \frac{10754}{945} f_{n-4} + \frac{5547}{824} f_{n-5} - \frac{2697}{1213} f_{n-6} + \frac{421}{1334} f_{n-7} \right)$$

Table 3.2: Adams-Bashforth method's coefficients up to $k = 7$ steps

k	b_1	b_2	b_3	b_4	b_5	b_6	b_7
1	1						
2	$\frac{3}{2}$	$-\frac{1}{2}$					
3	$\frac{23}{12}$	$-\frac{16}{12}$	$\frac{5}{12}$				
4	$\frac{55}{24}$	$-\frac{59}{24}$	$\frac{37}{24}$	$-\frac{9}{24}$			
5	$\frac{1901}{720}$	$-\frac{2774}{720}$	$\frac{2616}{720}$	$-\frac{1274}{720}$	$\frac{251}{720}$		
6	$\frac{4277}{1440}$	$-\frac{7923}{1440}$	$\frac{9982}{1440}$	$-\frac{7298}{1440}$	$\frac{2877}{1440}$	$-\frac{475}{1440}$	
7	$\frac{28369}{8634}$	$-\frac{5421}{733}$	$\frac{46083}{395}$	$-\frac{10754}{945}$	$\frac{5547}{824}$	$-\frac{2697}{1213}$	$\frac{421}{1334}$

Adams–Moulton method is an implicit multistep method whence

$$p = k - 2, \quad a_1 = a_2 = \dots = a_{k-2} = 0$$

defined by

$$y_n = y_{n-1} + h(b_0 f_n + b_1 f_{n-1} + \dots + b_{k-1} f_{n-k+1}).$$

Similarly, coefficients are obtained for the highest order possible, see table 3.3. And however, the Adams-Moulton are implicit methods, thus reach order $p + 1$. The Adams-Moulton formula of order 1 yields the (implicit) backward Euler integration method and the formula of order 2 yields method known as the trapezoidal rule.

Table 3.3: Adams-Moulton method's coefficients for $k = 7$ steps for constant stepsize

k	b_0	b_1	b_2	b_3	b_4	b_5	b_6
1	1						
2	$\frac{1}{2}$	$\frac{1}{2}$					
3	$\frac{5}{12}$	$\frac{8}{12}$	$-\frac{1}{12}$				
4	$\frac{9}{24}$	$\frac{19}{24}$	$-\frac{5}{24}$	$\frac{1}{24}$			
5	$\frac{251}{720}$	$\frac{646}{720}$	$-\frac{264}{720}$	$\frac{106}{720}$	$-\frac{19}{720}$		
6	$\frac{475}{1440}$	$\frac{1427}{1440}$	$-\frac{798}{1440}$	$\frac{482}{1440}$	$-\frac{173}{1440}$	$\frac{27}{1440}$	
7	$\frac{925}{2931}$	$\frac{2713}{2520}$	$-\frac{2015}{2623}$	$\frac{586}{945}$	$-\frac{1321}{3953}$	$\frac{263}{2520}$	$-\frac{37}{2593}$

A comparison of tables 3.2 and 3.3 reveals that the coefficients of the implicit formula are smaller than those of the corresponding explicit formulas. The smaller coefficients lead to smaller local truncation errors and, hence, to improved accuracy over the explicit Adams-Bashforth methods [45].

Stability analysis

A linear multistep method $[\alpha, \beta]$ is stable if the difference equation (3.9) has only bounded solution. The difference equation represents an one-dimensional problem to equation (3.4) with $f(x, y) = 0$ gives

$$y_n = \alpha_1 y_{n-1} + \alpha_2 y_{n-2} + \dots + \alpha_k y_{n-k}. \quad (3.9)$$

A linear multistep method is said to be stable if all solution of the difference equation (3.9) are bounded as $n \rightarrow \infty$. Let $p(\lambda)$ be the corresponding characteristic polynomial

$$p(\lambda) = \lambda^k - \alpha_1 \lambda^{k-1} - \alpha_2 \lambda^{k-2} - \dots - \alpha_k.$$

A method is said to satisfy the root condition if $|\lambda_j| \leq 1$ for all j , and if $|\lambda_i|$ is a repeated root then $|\lambda_j| < 1$. That is, all roots must lie within the unit circle and those on the boundary must be simple [21].

The Adams-Bashforth formulae are straightforward computations of y_n , but they are handicapped by lower attainable order compared with the corresponding implicit methods. Their stability is also quite inferior to that of the corresponding implicit processes [45]. According to Dahlquist (1963) the theorem about the A-stability is given by Barrier theorem [38]

1. An explicit k -step method cannot be A-stable.

2. The order p of an A-stable method, linear multistep method cannot exceed 2. The smallest error constant $c = 1/12$ is obtained for the trapezoidal rule.

The stability regions for Adams–Bashforth formulae of order up to 6 are presented in figure 3.1. The stability regions for Adams–Moulton formulae of order up to 6 are presented in figure 3.2.

3.3 Predictor–corrector methods

Predictor–corrector methods constitute an important algorithm in implementation of linear multistep methods and the most successful codes for the solution of initial value problems of ordinary differential equations. Briefly, these methods are successful because they occur in naturally arising families covering a range of orders, they have reasonable stability properties, and they allow an easy control via suitable stepsize or order changing policies and techniques. The major advantage of the multistep methods is that fewer functional evaluations are usually required per integration step [46].

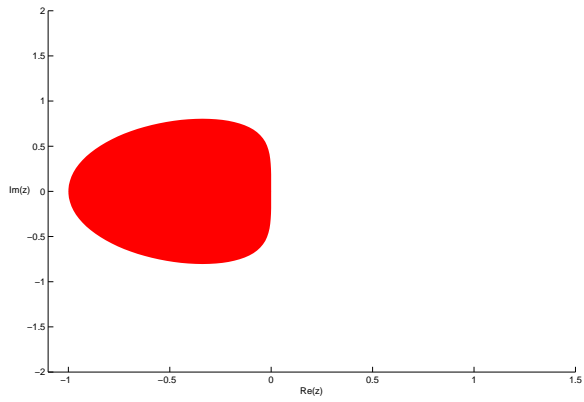
We obtain different types by combinations of explicit and implicit methods. Usually the predictor is an Adams-Bashforth formula and it predicts first approximation value of the solution. The derivative evaluated from this approximation is used in Adams-Moulton corrector formula in the next step. Apart from the better stability of the predictor-corrector formulae over the explicit formulae, the predictor-corrector formulae are generally more accurate and provide reasonable and adequate error estimators [45].

In the calculation of predictor-corrector pairs are three stages:

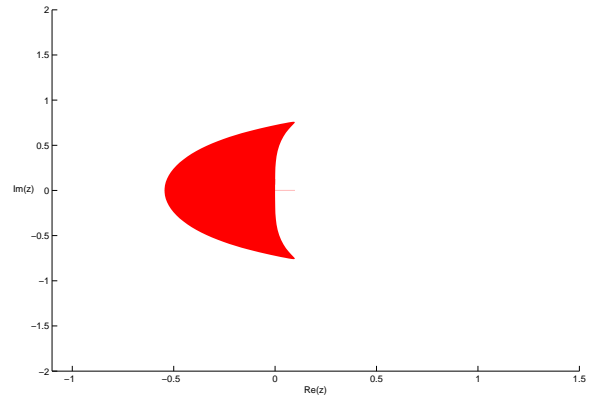
1. Predict the starting value for the dependent variable y_{n+k} as y_{n+k}^* .
2. Evaluate the derivative at (x_{n+k}, y_{n+k}^*) .
3. Correct the evaluated predicted value.

A combination of three stages is called PEC (predict–evaluate–correct) mode. It is often more desirable in terms of stability considerations to incorporate one additional function evaluation per integration step, thus calculate the PECE (predict–evaluate–correct–evaluate) mode [70]. Other options of repeating stages are possible but we have in mind that it is generally considered that functional evaluations are the most expensive part of the predictor–corrector procedure.

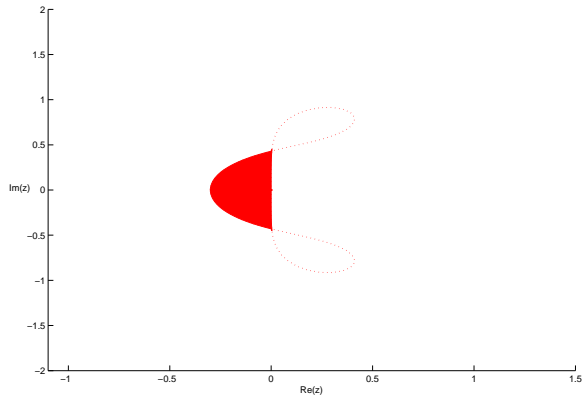
See how the stability regions of different combinations look like. On the picture 3.3 we can see the stability regions of predictor-corrector methods based on Adams-Bashforth method order 2 and 3 and Adams–Moulton method order 2 and 3.



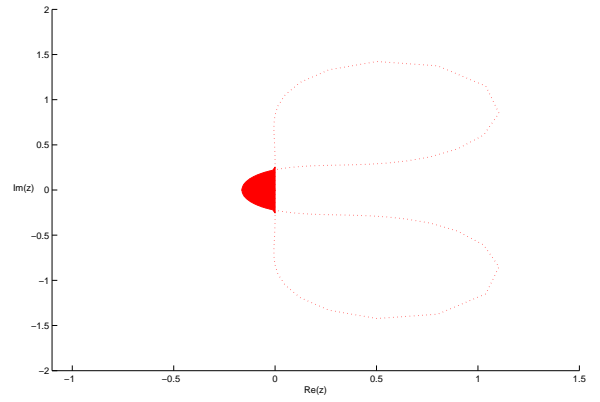
a) second order AB



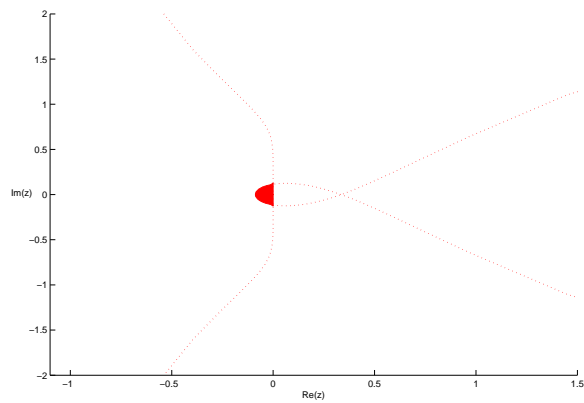
b) third order AB



c) fourth order AB

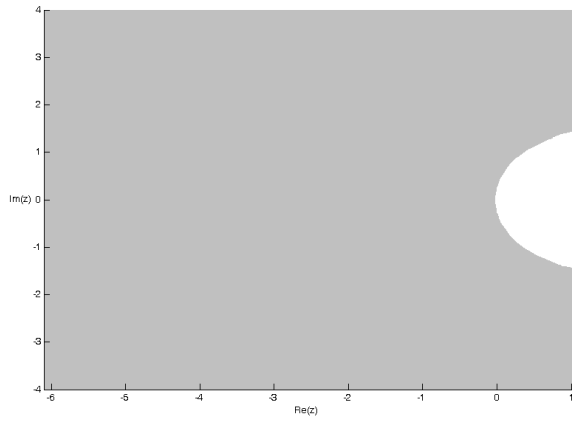


d) fifth order AB

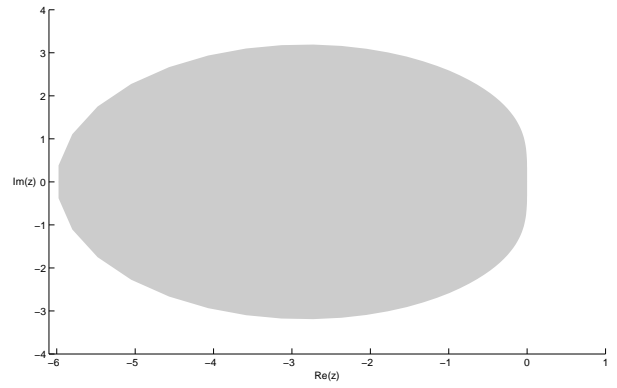


e) sixth order AB

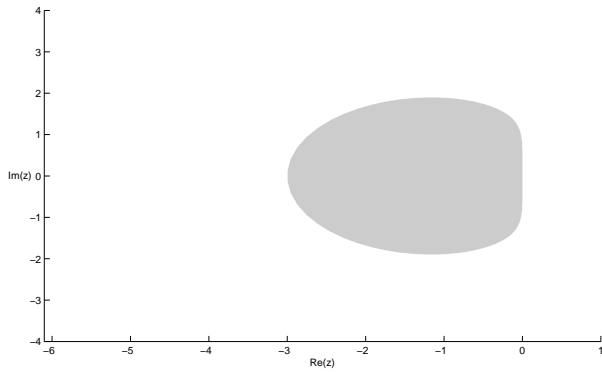
Figure 3.1: Regions of absolute stability - Adams–Bashforth method up to order 6



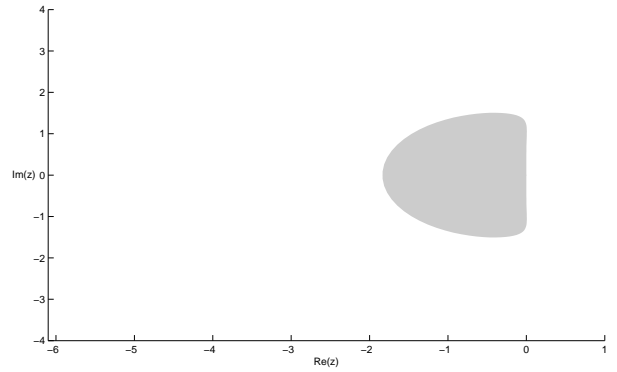
a) second order AM



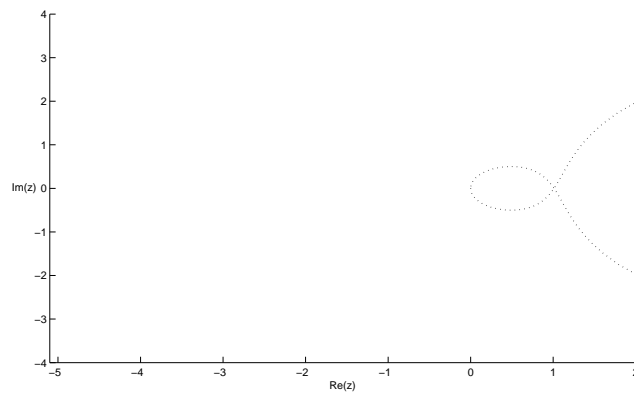
b) third order AM



c) fourth order AM

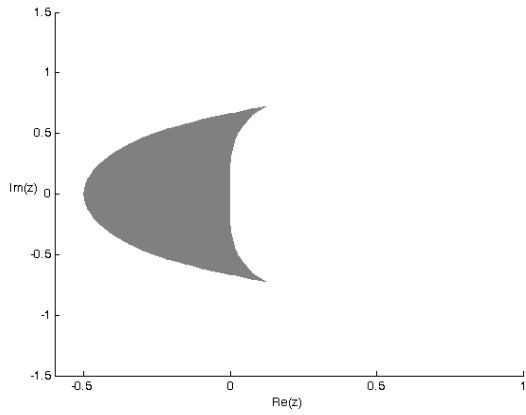


d) fifth order AM

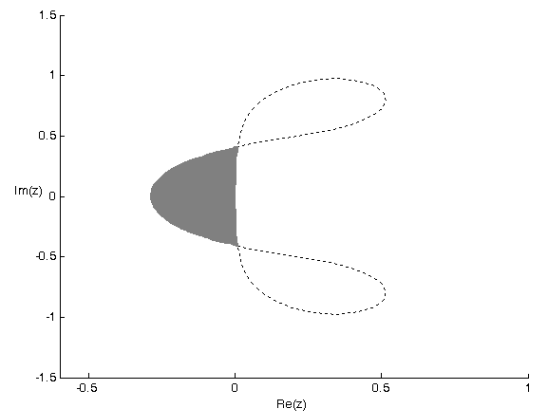


e) sixth order AM

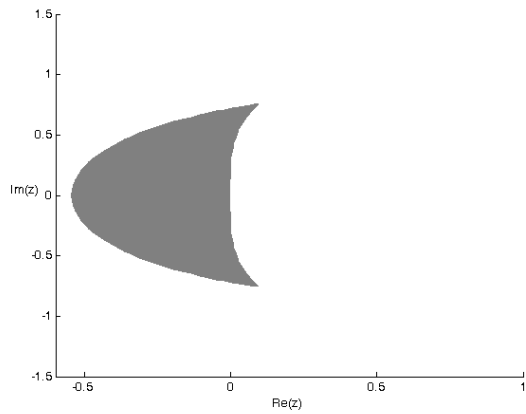
Figure 3.2: Regions of absolute stability - Adams–Moulton method up to order 6



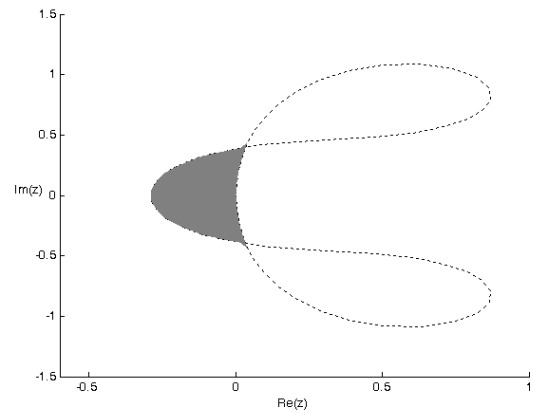
a) PEC (AB2+AM2)



b) PEC (AB3+AM3)



c) PEC (AB2+AM3)



d) PEC (AB3+AM2)

Figure 3.3: Regions of absolute stability: predictor–corrector methods

The stability improvement is given by PECE mode. One more evaluation on the end of each computational step makes the stability region more wide [71]. Notice the difference in stability region for Adams-Bashforth method order 2 (AB2) and Adams–Moulton method order 2 (AM2) in PEC mode presented in picture 3.4a and stability region of same methods orders 3 (AB3, AM3) in PECE mode in picture 3.4b.

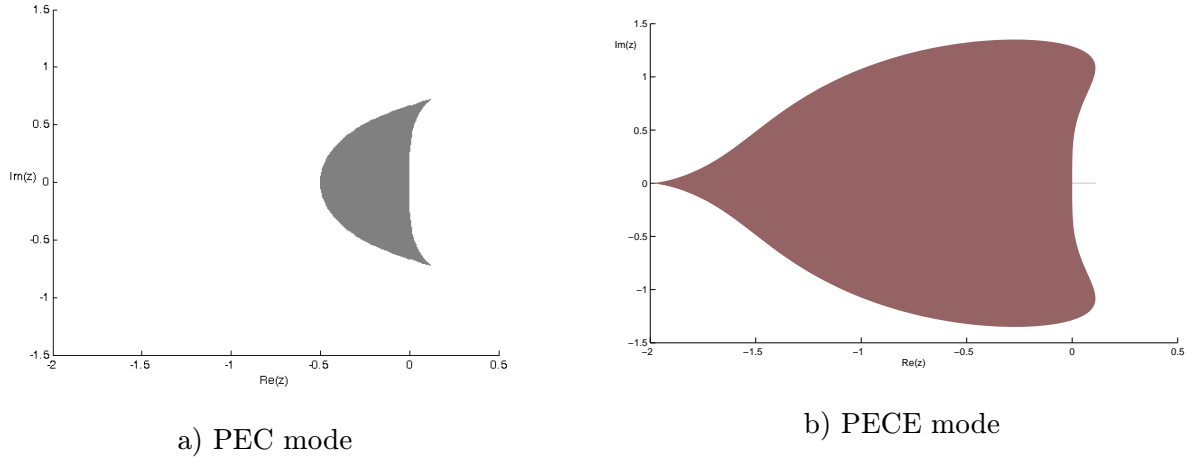


Figure 3.4: Regions of absolute stability - Adams-Bashforth method and Adams–Moulton method of order 2

In this chapter we presented the generalization of using previous values such as the value y_n depends on y_{n-1} , $f(y_{n-1})$ and on y_{n-2} , $f(y_{n-2})$, y_{n-3} , $f(y_{n-3}), \dots$. The next generalization is given by higher derivatives of these previous values, we obtain methods called multiderivative multistep methods.

Chapter 4

Multiderivative multistep methods

To obtain the *General linear methods* we have two options. We generalize Runge–Kutta methods in case of using more previous values, or we generalize Linear multistep methods in case of using more stages in the calculation per step. So we have a range of possibilities from 1 input quantity, as in Runge–Kutta methods, to a large number as in multistep methods.

4.1 General linear methods

We denote the number of input quantities by r , the number of stages by s and then the quantities input at the start of step n is $y_i^{(n-1)}$ for $i = 1, 2, \dots, r$ we got

$$y^{(n-1)} = \begin{bmatrix} y_1^{(n-1)} \\ y_2^{(n-1)} \\ \vdots \\ y_r^{(n-1)} \end{bmatrix}. \quad (4.1)$$

We also need the notation for the stage values computed in step n as Y_i for $i = 1, 2, \dots, s$, which can be represented by

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_s \end{bmatrix} \quad (4.2)$$

and the stage derivatives computed in step n is F_i also for $i = 1, 2, \dots, s$ such as

$$F = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_s \end{bmatrix}. \quad (4.3)$$

Finally the quantities exported at the end of each step is $y_i^{(n)}$ for $i = 1, 2, \dots, r$

$$y^{(n)} = \begin{bmatrix} y_1^{(n)} \\ y_2^{(n)} \\ \vdots \\ y_r^{(n)} \end{bmatrix}. \quad (4.4)$$

The formulation of general linear methods for the various steps is given by

$$\begin{aligned} Y_i &= \sum_{j=1}^s a_{ij} h F_j + \sum_{j=1}^r u_{ij} y_j^{(n-1)}, & F_i &= f(Y_i), & i &= 1, 2, \dots, s \\ y_i^{(n)} &= \sum_{j=1}^s b_{ij} h F_j + \sum_{j=1}^r u_{ij} y_j^{(n-1)}, & & & i &= 1, 2, \dots, r \end{aligned} \quad (4.5)$$

or using a compact notation

$$\begin{aligned} Y &= (A \otimes I) h F + (U \otimes I) y^{(n-1)} \\ y^{(n)} &= (B \otimes I) h F + (V \otimes I) y^{(n-1)} \end{aligned} \quad (4.6)$$

To show the process of carrying out the step we follow the given algorithm [20]

1. The subvectors r comprising $y_i^{(n-1)}$ are imported at the start of step n .
2. The subvectors in Y_i and F_j are computed.
3. Each of the Y_i is a linear combination of the hF_j and the $y_j^{(n-1)}$, we obtain the matrices A and U .
4. The subvectors r comprising $y_i^{(n)}$ are computed corresponding to the $u_{ij}^{(n-1)}$ subvectors.
5. The $y_i^{(n)}$ are linear combinations of the hF_j and the $y_j^{(n-1)}$ and we obtain the matrices B and V .

The special case of general multistep methods is Obreshkov quadrature formulae. We briefly introduce them in the next section due to the approach of the contribution in the next chapter.

4.2 Obreshkov quadrature formulae

It has been described that the suitable way to obtain the s -stage q -derivative method is to use the collocation method with multiple nodes. For ordinary differential equations it consists in searching for a polynomial of degree s whose derivative coincides („co-locates“) at s given points with the vector field of the differential equation.

For s a positive integer and c_1, c_2, \dots, c_s distinct real numbers (typically between 0 and 1), the corresponding collocation polynomial $u(x)$ of degree s is defined by

$$u'(x_0 + c_i h) = f(x_0 + c_i h, u(x_0 + c_i h)), \quad i = 1, \dots, s, \quad u(x_0) = y_0 \quad (4.7)$$

with the numerical solution given by

$$y_1 = u(x_0 + h). \quad (4.8)$$

If some of the c_i coincide, the collocation condition will contain higher derivatives so to obtain the s -stage q -derivative method we can replace equation (4.7) by

$$u^{(l)}(x_0 + c_i h) = (D^l y)(x_0 + c_i h, u(x_0 + c_i h)), \quad i = 1, \dots, s, \quad l = 1, \dots, q_i, \quad (4.9)$$

where $u(x)$ is a polynomial of degree $q_1 + q_2 + \dots + q_s$, q_1, q_2, \dots, q_s are multiplicities of nodes c_1, c_2, \dots, c_s and D is a differential operator which, when applied to a function $\Psi(x, y)$, is given by

$$(D\Psi)(x, y) = \frac{\partial\Psi}{\partial x}(x, y) + \frac{\partial\Psi}{\partial y^1}(x, y) \cdot f^1(x, y) + \dots + \frac{\partial\Psi}{\partial y^n}(x, y) \cdot f^n(x, y). \quad (4.10)$$

Under the generalization we have to replace the Lagrange interpolation by Hermite interpolation and now we consider the special case of collocation methods with $s = 2$, $c_1 = 0$, $c_2 = 1$. The multi-derivative method of order m is then defined by

$$\sum_{j=0}^m h^j (D^j y)(x_1, y_1) P^{(m-1)}(0) = \sum_{j=0}^m h^j (D^j y)(x_0, y_0) P^{(m-1)}(1), \quad (4.11)$$

where $P(t)$ is a polynomial of exact degree m [27, 86].

The next section brings the theory about the Nordsieck representation as an important approach for variable stepsize.

4.3 Nordsieck representation

There are two important considerations when selecting a multistep method for solving the initial value problems, the speed and the accuracy. There are three important considerations

for developing and using the predictor–corrector pairs for solving the initial value problems, the speed, the truncation error and the stability. The basic factor for speed determination is the number of functional evaluations required per step. An accuracy is given by the stability, by the error and by the truncation error of the method. It has been shown that Adams methods satisfy those requirements in predictor–corrector formulae. The major problem with these formulae is that interval modification and interpolation is difficult.

Equivalent form to Adams methods has been proposed by Nordsieck (1962). The form allows to change the stepsize of the method very easily and it accomplishes changing of stepsize simply and inexpensively. The derivation of Nordsieck form is based on Taylor’s theorem. The Nordsieck form stores the current values of the higher derivatives of a polynomial approximating the solution. The Nordsieck form of Adams-Moulton method can be given by

$$Y_n = PY_{n-1} + LG_n, \quad (4.12)$$

where

$$Y_n = \begin{bmatrix} y_n \\ hy'_n \\ \frac{h^2}{2!}y''_n \\ \vdots \\ \frac{h^{k-1}}{(k-1)!}y_n^{(k-1)} \end{bmatrix}, \quad (4.13)$$

$$L = \begin{bmatrix} l_{k,0} \\ l_{k,1} \\ l_{k,2} \\ \vdots \\ l_{k,k-1} \end{bmatrix}, \quad l_{k,0} = \beta_{k,0}, \quad l_{k,1} = 1, \quad (4.14)$$

where $P = (a_{ij})_{k \times k}$ is a Pascal matrix with elements

$$\begin{aligned} a_{1,j} &= 1, & j &= 1, 2, \dots, k, \\ a_{i,j} &= 0, & i &= 2, 3, \dots, k, \quad j < i, \\ a_{i,j} &= C_{j-1}^{i-1}, & i &= 2, 3, \dots, k, \quad j \geq i \end{aligned}$$

and where $y(x_n)$ is the exact solution of the initial value problem and

$$G_n = hf_n - h\tilde{f}_n, \quad h\tilde{f}_n = \sum_{j=1}^{k-1} \frac{j h^j y_{n-1}^{(j)}}{j!},$$

the $y_n^{(j)}$ are meant to be approximations to $y^{(j)}(x_n)$. Nordsieck determined that values $(l_{k,0}, l_{k,1}, \dots, l_{k,k-1})$ accomplish the desirable property of an accuracy. Coefficients are

chosen such that the error constant of the method vanishes. For each method or order the coefficients are determined specifically [30, 84].

The construction of a p -th order general linear method requires more specific order conditions. It can be demonstrated using the theory of B -series [10].

The next chapter is dedicated to the contribution of the thesis and it is based on methods, forms or formulae already mentioned as well as approaches or ideas which will be described now.

Chapter 5

Two-derivative multistep method

The main contribution of this thesis is to extend Adams methods to higher derivative methods by using Obreshkov quadrature formulae. We consider a two-step predictor followed by a one-step corrector, in each case using second derivative formulae. There is always a choice in predictor–corrector pairs of the so-called mode of the method and we consider both PEC and PECE methods. The Nordsieck representation of Adams methods adapts well to the multiderivative situation and it makes the variable stepsize convenient.

We start with a generalization of Adams methods to second derivative methods. We consider problems for which it is efficient to calculate first and second derivatives at any solution point. We denote first and second derivatives by

$$y'(x) = f(x, y), \quad y''(x) = g(x, y).$$

At the start of the step we assume that we already have computed values in previous points

$$y_{n-1}, y_{n-2}, \dots, y_{n-k},$$

obtained from the starting method. The question of starting method will be discussed later. We also know from the given problem first and second derivative values,

$$f_{n-1}, f_{n-2}, \dots, f_{n-k}, g_{n-1}, g_{n-2}, \dots, g_{n-k},$$

which are given by $f_i = f(x_i, y_i)$, $g_i = g(x_i, y_i)$.

Adams methods are usually implemented as predictor–corrector pairs. That is, a predicted part is evaluated by the formula using an explicit method and corrected part is evaluated by the formula using an implicit formula. The predictor step is generally given by

$$y_n = y_{n-1} + h(a_1 f_{n-1} + a_2 f_{n-2} + \dots + a_k f_{n-k})$$

and the corrector step by

$$y_n = y_{n-1} + h(b_0 f_n + b_1 f_{n-1} + \dots + b_{k-1} f_{n-k+1}).$$

Since coefficients are independent of h and n , the order conditions are given by

$$\int_{-1}^0 \phi(t) dt = a_1 \phi(-1) + a_2 \phi(-2) + \cdots + a_k \phi(-k),$$

$$\int_{-1}^0 \phi(t) dt = b_1 \phi(0) + b_2 \phi(-1) + \cdots + b_k \phi(-k+1),$$

whenever $\phi(t)$ is a polynomial of degree not exceeding $k-1$. We will restrict our attention to the case $k=2$ and to calculate the coefficients a_i and b_i we can rewrite them in the Lagrange interpolation formulae

$$\phi(t) = \alpha_1(t)\phi(-1) + \alpha_2(t)\phi(-2) + \cdots + \alpha_k(t)\phi(-k),$$

$$\phi(t) = \beta_1(t)\phi(0) + \beta_2(t)\phi(-1) + \cdots + \beta_k(t)\phi(-k+1)$$

and we can find coefficients in Adams methods as

$$a_i = \int_{-1}^0 \alpha_i(t) dt,$$

$$b_i = \int_{-1}^0 \beta_i(t) dt, \quad i = 1, 2, \dots, k.$$

Obreshkov method becomes available if, in addition to a formula for $f(x, y)$, a formula for $g(x, y)$ is also available. The extension to two-derivative methods uses the order $2k$ -formulae for the predictor equation

$$y_n = y_{n-1} + h(a_1 f_{n-1} + a_2 f_{n-2} + \cdots + a_k f_{n-k}) + h^2(c_1 g_{n-1} + c_2 g_{n-2} + \cdots + c_k g_{n-k}) \quad (5.1)$$

and the order $2k$ -formulae for the corrector equation

$$y_n = y_{n-1} + h(b_0 f_n + b_1 f_{n-1} + \cdots + b_{k-1} f_{n-k+1}) + h^2(d_0 g_n + d_1 g_{n-1} + \cdots + d_{k-1} g_{n-k+1}). \quad (5.2)$$

With the restriction to the case $k=2$ we can replace the Lagrange integration polynomial by the Lagrange-Hermite integration polynomial

$$\phi(t) = \alpha_1(t)\phi(-1) + \alpha_2(t)\phi(-2) + \gamma_1(t)\phi'(-1) + \gamma_2(t)\phi'(-2),$$

$$\phi(t) = \beta_0(t)\phi(0) + \beta_1(t)\phi(-1) + \delta_0(t)\phi'(0) + \delta_1(t)\phi'(-1),$$

hence

$$a_i = \int_{-1}^0 \alpha_i(t) dt, \quad c_i = \int_{-1}^0 \gamma_i(t) dt, \quad i = 1, 2,$$

$$b_i = \int_{-1}^0 \beta_i(t) dt, \quad d_i = \int_{-1}^0 \delta_i(t) dt, \quad i = 1, 2.$$

The coefficients of predictor and corrector formulae are found

$$\begin{aligned} a_1 &= -\frac{1}{2}, & b_0 &= \frac{1}{2}, \\ a_2 &= \frac{3}{2}, & b_1 &= \frac{1}{2}, \\ c_1 &= \frac{17}{12}, & d_0 &= -\frac{1}{12}, \\ c_2 &= \frac{7}{12}, & d_1 &= \frac{1}{12}. \end{aligned}$$

and the final formula of predictor equation is

$$y(x_n) = y_{n-1} - \frac{1}{2}hf_{n-1} + \frac{3}{2}hf_{n-2} + \frac{17}{12}h^2g_{n-1} + \frac{7}{12}h^2g_{n-2} \quad (5.3)$$

and the corrector equation is

$$y(x_n) = y_{n-1} + \frac{1}{2}hf_n + \frac{1}{2}hf_{n-1} - \frac{1}{12}h^2g_n + \frac{1}{12}h^2g_{n-1}. \quad (5.4)$$

Due to we assume for the new method usage of the variable stepsize, we implement the new method in Nordsieck representation.

Predictor-corrector Obreshkov method in Nordsieck representation

To distinguish between Nordsieck and non-Nordsieck representation of our equations, we use the notation N_n for Nordsieck vector now. For our purposes we need to specify the Nordsieck vector as five components due to five terms of the input vector including previous point x_n, x_{n-1}

$$N_n = \begin{bmatrix} y(x_n) \\ y'(x_n) \\ y''(x_n) \\ y'''(x_n) \\ y^{(4)}(x_n) \end{bmatrix}. \quad (5.5)$$

For the non-Nordsieck representation of our two-step second derivative method with two derivatives we get

$$Y_n = \begin{bmatrix} y(x_n) \\ y'(x_n) \\ y''(x_n) \\ y'(x_{n-1}) \\ y''(x_{n-1}) \end{bmatrix}. \quad (5.6)$$

To establish the relation between input vector Y_n and the Nordsieck vector N_n , we consider the Taylor expansion of each component. Approximations are derived and the transformation matrix is given

$$\begin{bmatrix} y(x_n) \\ hy'(x_n) \\ h^2y''(x_n) \\ hy'(x_{n-1}) \\ h^2y''(x_{n-1}) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 1 & -2 & 3 & -4 \\ 0 & 0 & 2 & -6 & 12 \end{bmatrix} \cdot \begin{bmatrix} y(x_n) \\ hy'(x_n) \\ \frac{1}{2!}h^2y''(x_n) \\ \frac{1}{3!}h^3y'''(x_n) \\ \frac{1}{4!}h^4y^{(4)}(x_n) \end{bmatrix} \quad (5.7)$$

The relation between non-Nordsieck and Nordsieck is now described by

$$Y_n = TN_n. \quad (5.8)$$

To derive our method, we use the construction of predict–evaluate–correct algorithm and we approximate the first step using predictor equation without correction. The coefficients from the predictor equation are written in a the first row of the matrix and the evaluation of f -function in the second row, and the evaluation of g -function in the third row respectively. For convenience, from now we will have in our mind that we are using the Nordsieck representation and we will denote the vector Y_n instead of N_n .

The first step in developing the algorithm for a new method is to use the predictor equation

$$Y_n^* = \begin{bmatrix} 1 & -\frac{1}{2} & \frac{17}{12} & \frac{3}{2} & \frac{7}{12} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} Y_{n-1}, \quad (5.9)$$

where Y_n^* means the vector of predicted values, Y_{n-1} is the vector of values in the previous step and the matrix is denoted as H .

The second step is to evaluate f -function and g -function and add them into the formula (5.9), we get

$$Y_n^* = \begin{bmatrix} 1 & -\frac{1}{2} & \frac{17}{12} & \frac{3}{2} & \frac{7}{12} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} Y_{n-1} + \begin{bmatrix} 0 \\ hf(Y_n^*) \\ h^2g(Y_n^*) \\ 0 \\ 0 \end{bmatrix}. \quad (5.10)$$

The term $f(Y_n^*)$ is the f -function evaluation of predicted value Y_n^* . The term $g(Y_n^*)$ is the g -function evaluation of the predicted value.

Before including the corrector equation into the method we must say that output does not rely on the second derivatives of previous points at all, so the output of the predicted value still needs to be uncovered completely. We determine the errors of the algorithm as

$$\delta = hf(Y_{n_1}^*) - [0 \ 1 \ 2 \ 3 \ 4]Y_{n-1}^* \quad (5.11)$$

$$\epsilon = h^2g(Y_{n_1}^*) - [0 \ 0 \ 1 \ 3 \ 6]Y_{n-1}^* \quad (5.12)$$

where $Y_{n_1}^*$ means the first component of a vector with predicted values. The error estimations δ and ϵ will give us the corrected part of the predicted value. Both terms are evaluations of functions of predicted values. The error δ represents the correction of the first derivative, the second error ϵ corrects the error in the second derivative.

To summarize described steps above we write them in the Nordsieck representation as

$$P = T^{-1}HT + T^{-1}e_2[0 \ 1 \ 2 \ 3 \ 4] + 2T^{-1}e_3[0 \ 0 \ 1 \ 3 \ 6] \quad (5.13)$$

which gives us the upper triangular matrix known as the Pascal matrix

$$P = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 1 & 3 & 6 \\ 0 & 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

This matrix computes the predicted value Y_n^* from the given input Y_{n-1} .

To complete the algorithm for the new method, we include the final step which is a correction equation. The new method is now described by

$$Y_n = PY_{n-1} + \delta T^{-1}\alpha + \epsilon T^{-1}\beta, \quad (5.14)$$

where α is equal to vector e_2 with one more correction, that is the corresponding coefficient $b_1 = 1/2$ from the term $1/2hf_{n-1}$ in equation (5.4). Similarly, β is equal to vector e_3 with one more correction, which is the coefficient $d_1 = -1/12$ from the term $-1/12h^2g_{n-1}$ in equation (5.4). Vectors are defined such as

$$\alpha = \begin{bmatrix} \frac{1}{2} \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \beta = \begin{bmatrix} -\frac{1}{12} \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}.$$

The final formula in Nordsieck representation for predictor-corrector Obreshkov method is given by

$$Y_n = PY_{n-1} + \delta \begin{bmatrix} \frac{1}{2} \\ 1 \\ 0 \\ -1 \\ -\frac{1}{2} \end{bmatrix} + \epsilon \begin{bmatrix} -\frac{1}{12} \\ 0 \\ 1 \\ \frac{4}{3} \\ \frac{1}{2} \end{bmatrix}, \quad (5.15)$$

where it holds

$$\begin{aligned} \delta &= hf(Y_{n_1}^*) - [0 \ 1 \ 2 \ 3 \ 4]Y_{n-1}^*, \\ \epsilon &= h^2g(Y_{n_1}^*) - [0 \ 0 \ 1 \ 3 \ 6]Y_{n-1}^*. \end{aligned}$$

We now introduce problems which were tested.

Test problems

Differential equations and some systems of differential equations with initial conditions from DETEST [64] were successfully tested. Problems with known analytical solution were easily checked, problems with unknown analytical solution were compared with given results presented in [64]. The program Scilab (version 5.0.3) was chosen for the implementation of various problems.

Beyond those tests the new method was also successfully tested on a „circle test“ given by

$$y'' + y = 0, \quad y(0) = 1, \quad y'(0) = 0,$$

which is for our method rewritten to the system of differential equations

$$y_1' = y_2, \quad y_1(0) = 1, \quad (5.16)$$

$$y_2' = -y_1, \quad y_2(0) = 0. \quad (5.17)$$

Our method needs to have second derivatives of given problem, we add two equations

$$y_1'' = -y_1, \quad y_1'(0) = 0,$$

$$y_2'' = -y_2, \quad y_2'(0) = -1,$$

with analytical solutions $y_1 = \cos(x)$ and $y_2 = -\sin(x)$. Results of circle test are shown in the Appendix D.

As the first problem from DETEST we present results of very known and already mentioned *Dahlquist problem* given previously in subsection 2.2 by equation (2.31) such as

$$y' = qy, \quad y(0) = 1,$$

where q is a complex number with a negative real part. The exact solution is given by $y = \exp(qx)$ and decays exponentially from 1. We specify the constant $q = 1$ and obtain

$$y' = y, \quad y(0) = 1 \quad (5.18)$$

with second derivatives

$$y'' = y, \quad y'(0) = 1.$$

The second test problem known as *Prothero-Robinson problem* is generally given by

$$y' = L(y - q(x)) + q'(x), \quad y(0) = 0.$$

We specify the constant $L = -1$ and the function $q(x) = \sin(x)$ which is also the analytical solution of the problem in this case. We get the system of two equations for our method specified by

$$\begin{aligned} y_1' &= -y_1 + \sin(y_2) + \cos(y_2), & y_1(0) &= 0, \\ y_2' &= 1, & y_2(0) &= 1 \end{aligned} \quad (5.19)$$

and second derivatives easily determined as

$$\begin{aligned} y_1'' &= y_1 - 2\sin(y_2), & y_1'(0) &= 0, \\ y_2'' &= 0, & y_2'(0) &= 1. \end{aligned}$$

The third problem is *Kepler problem* already mentioned in section 2.5. The Kepler problem is given by four differential equations of first order (2.53). We must add second derivatives for our method which are given by

$$\begin{aligned} y_1'' &= \frac{-y_1}{r^3}, & y_1'(0) &= 0, \\ y_2'' &= \frac{-y_2}{r^3}, & y_2'(0) &= 0, \\ y_3'' &= \frac{y_3(2y_1^2 - y_2^2) + 3y_1y_2y_4}{r^5}, & y_3'(0) &= 0, \\ y_4'' &= \frac{y_4(2y_2^2 - y_1^2) + 3y_1y_2y_3}{r^5}, & y_4'(0) &= 0. \end{aligned}$$

Results will be shown in the next few sections. We now introduce the starting method for our method.

5.1 Starting method

It has been already mentioned that the given problem should be described by its first and second derivatives for solving by the new method. But it should be emphasized that the

method requires only initial values for the first derivative. The starting method described in this subsection determines all necessary information for the next calculation.

As a starting method can be used classical one-step methods such as Runge–Kutta method of order 4 or Adams–Bashforth method of order 4, but it seems reasonable to use the predictor–corrector Obreshkov method itself. Due to known initial values of 3 components of our input vector, we need to derive only the first and second derivative in previous steps. Same two steps are switched by fixed stepsize, ones it is positive, then it is negative and repeating until converge or until given number of iterations. The speed of convergence is given by the chosen tolerance and the starting algorithm will produce results which become a part of the input vector for our method.

We present a Prothero–Robinson problem (5.19) for describing details due to the fact that we are able to determine exact initial value for comparison. If we compare results of the starting procedure for stepsizes after 1, 2, 3, 4, 6, 8, 16 or 32 repetitions (cycles) we find the natural behaviour for the error between the exact initial values and the computed initial values. With decreasing value of the stepsize the error decreases with the slope corresponding to the fifth order. By the cycle we mean 1 step forward and 1 step backward. Errors for different stepsizes and cycles are represented in table 5.1. We see used stepsizes in the first column of the table, the error after 1 cycle of the starting method is presented in the second column. The error after 2 cycles of the starting method is presented in the third column etc.

Table 5.1: Errors between exact initial values and computed initial values for Prothero–Robinson problem

h	1 cycle	2 cycles	8 cycles
0.1	2.550e-05	1.101e-06	2.489e-07
$0.1 \cdot 2^{-1}$	1.523e-06	3.205e-08	7.80e-09
$0.1 \cdot 2^{-2}$	9.305e-08	9.666e-10	2.442e-10
$0.1 \cdot 2^{-3}$	5.751e-09	2.968e-11	7.636e-12
$0.1 \cdot 2^{-4}$	3.574e-10	9.192e-13	2.387e-13
$0.1 \cdot 2^{-5}$	2.228e-11	2.860e-14	7.461e-15
$0.1 \cdot 2^{-6}$	1.390e-12	8.918e-16	2.332e-16
$0.1 \cdot 2^{-7}$	8.683e-14	2.792e-17	7.338e-18

The starting method uses the following procedure:

1. Evaluate the f -function and g -function for given initial value $y(0)$.

2. Calculate the PEC mode of a new method.
3. Change the sign of the stepsize.
4. If the step is even, set the output values as input values for the next step, else reset first three components of the Nordsieck vector and replace them by known initial values $y(0)$, $f(y(0))$, $g(y(0))$ and set last two components of the Nordsieck vector by computed values in this step.
5. Repeat until converge or until given number of cycles (iterations).

Hence, only two components have an influence on the precision of the calculation, those are first and second derivatives in a previous point. It has been observed that the global error of the numerical method is also influenced by the chosen stepsize for a starting method. Global errors and the speed of convergence of the new method is represented in table 5.2. Global errors of the new method with exact initial values are represented in the second column of the table. Then errors of the new method with computed initial values in one cycle of the starting procedure are represented in the third row, etc. It can be easily calculated that the method preserve the order 4.

Table 5.2: Errors of the new method given by different number of cycles for starting method

h	exact IV's	1 cycle	2 cycles	8 cycles
0.1	5.1975119719e-07	6.6524014852e-07	5.0145286223e-07	5.0870973866e-07
$0.1 \cdot 2^{-1}$	2.7172841532e-08	3.1526897537e-08	2.6743608438e-08	2.6851363132e-08
$0.1 \cdot 2^{-2}$	1.5282720644e-09	1.6607170083e-09	1.5170131817e-09	1.5186459867e-09
$0.1 \cdot 2^{-3}$	9.0206508929e-11	9.4284802187e-11	8.9887652876e-11	8.9911966760e-11
$0.1 \cdot 2^{-4}$	5.4728443998e-12	5.5991877800e-12	5.4628523926e-12	5.4629634149e-12
$0.1 \cdot 2^{-5}$	3.3595348725e-13	3.3983926783e-13	3.3562042034e-13	3.3573144264e-13
$0.1 \cdot 2^{-6}$	1.9428902930e-14	1.9428902930e-14	1.9428902930e-14	1.9428902930e-14
$0.1 \cdot 2^{-7}$	6.3282712403e-15	6.3282712403e-15	6.3282712403e-15	6.3282712403e-15

The suggested approach uses the fixed stepsize for the starting method in effectiveness point of view, but the value of the stepsize should be at least $2.5e-02$ with 8 cycles for the starting algorithm. Then we get very close approximations to the exact initial values and very fast convergence. The starting procedure code is presented in the Appendix C.

5.2 Stepsize control

For the most of multistep methods it is very efficient to use the variable stepsize. It is very uneconomical to keep the fixed stepsize during computation. As the number of steps increases, the local error decreases and the stability properties become worse. The task is to use an automatic procedure that continually adjusts the stepsize to achieve some target level of accuracy. Choosing the right stepsize is one of the main challenging task in designing a numerical integration scheme. Many sophisticated procedures have been described already [97].

In the beginning, the algorithm of choosing the variable stepsize is simplified and presented

1. Estimate the local error.
2. Decide if the computed value of y can be accepted or if the step needs to be recalculated with smaller stepsize from the previous point.
3. Determine the new stepsize to use for the recalculation.
4. Adjust the data according to the new stepsize for the next step.

To control the stepsize each part of the algorithm is now described in detail.

Error estimation

The problem of error estimation is the first task for controlling the stepsize during the computation so that the error in a step is approximately constant. There are few ways how to determine the local error. For example, starting at (x_n, y_n) we take one step h and we denote the result as (x_{n+1}, y_{n+1}^*) . Then again, we take two steps starting from the same point (x_n, y_n) , each step of size $h/2$ and the result denote as (x_{n+1}, y_{n+1}^{**}) . The local error of y_{n+1}^{**} is approximately bounded by

$$\epsilon_L = \frac{|y_{n+1}^* - y_{n+1}^{**}|}{2^p - 1},$$

where p is the order of the method [39]. When the local error ϵ_L is an estimate, the standard procedure is to keep it below a chosen tolerance per unit step. A bound of the global error could be tolerated, but classical methods are usually designed to control some measure of the local error. The error tolerance is usually required to be specified.

We use following approach for choosing the stepsize in our method. Assume that our goal is to compute the error in y_n after some short time t and the value of the error should be less than some constant eps , where $eps \ll 1$ and $y_e x$ is some reference position in time

t_{ex} . Firstly we move from y_n to y_{n+2} by taking two time steps of the length $h \ll t$. Then we return to y_n and take one step of the length $2h$ to reach y_{n+1} . Suppose that the correct position after an interval $2h$ is \hat{y}_n and our method has order p , the errors in t_{n+1} and t_{n+2} may be written

$$\begin{aligned} y_{n+1} - \hat{y}_n &\cong (2h)^{p+1} E, \\ y_{n+2} - \hat{y}_n &\cong (4h)^{p+1} E, \end{aligned}$$

where E is unknown error vector. Subtracting previous equations to eliminate \hat{y}_n , we find that

$$E = \frac{\|y_{n+1} - y_{n+2}\|}{2(2^{p+1} - 1)h^{p+1}}.$$

As we assume that for a time t we use $n = t/h_2$ time steps of the length h_2 , the error will be

$$\begin{aligned} \epsilon &= nh_2^{p+1} E, \\ E &= \|y_{n+1} - y_{n+2}\| \frac{th_2^p}{2(2^{p+1} - 1)h^{p+1}}. \end{aligned}$$

Our goal that $|\epsilon| \leq \text{eps}$ is satisfied if

$$h_2 < h_{max} \equiv \left(2(2^{p+1} - 1) \frac{h \cdot \text{eps}}{t(y_{n+1} - y_{n+2})} \right)^{\frac{1}{p}} h.$$

For some predictor–corrector methods the error can be expressed as the difference between the predicted and the corrected value multiplied by

$$\epsilon_L = \frac{C^C}{C^P - C^C},$$

where C^P , C^C are known constants. If the inequality between local error and chosen tolerance is not satisfied, the new stepsize needs to be determined. There are again several procedures, the important part is the safety factor which assures that the stepsize is upper and lower bounded [37, 54, 99]. Commonly, safety factors are numbers 0.8 or 0.9.

Assuming we are trying to control the error for our predictor–corrector method associated with just the single step from x_{n-1} to x_n . Ignoring the higher order terms we approximately write

$$\begin{aligned} y(x_n) &\approx y_n^P - C^P h^{p+1} y^{(p+1)}(x_n) \\ y(x_n) &\approx y_n^C - C^C h^{p+1} y^{(p+1)}(x_n) \end{aligned}$$

We want to know the error in the corrected value $C^C h^{p+1} y^{(p+1)}(x_n)$ and we find this by subtracting the predicted and corrected approximations

$$y_n^P - y_n^C \approx (C^P - C^C) h^{p+1} y^{(p+1)}(x_n)$$

hence

$$C^C h^{p+1} y^{(p+1)}(x_n) \approx \frac{C^C}{C^C - C^P} (y_n^C - y_n^P)$$

in our case it is

$$h^6 y^{(6)}(x_n) \approx \frac{1}{30} (y_n^P - y_n^C).$$

The proof is given by power series. The predictor (5.3) of our method is given by

$$C^P = 1 - e^{-z} + \frac{1}{2} z e^{-z} - \frac{17}{12} z^2 e^{-z} - \frac{3}{2} z e^{-2z} - \frac{7}{12} z^2 e^{-2z}$$

as a series

$$\begin{aligned} C^P &= 1 - (1 - z + \frac{1}{2} z^2 - \frac{1}{6} z^3 + \frac{1}{24} z^4 - \frac{1}{120} z^5) + \frac{1}{2} z (1 - z + \frac{1}{2} z^2 - \frac{1}{6} z^3 + \frac{1}{24} z^4) \\ &\quad - \frac{17}{12} z^2 (1 - z + \frac{1}{2} z^2 - \frac{1}{6} z^3) - \frac{3}{2} z (1 - 2z + 2z^2 - \frac{4}{3} z^3 + \frac{2}{3} z^4) - \frac{7}{12} z^2 (1 - 2z + 2z^2 - \frac{4}{3} z^3) \\ &= (1 - 1) + z(1 + \frac{1}{2} - \frac{3}{2}) + z^2(-\frac{1}{2} - \frac{1}{2} - \frac{17}{12} + 3 - \frac{7}{12}) \\ &\quad + z^3(\frac{1}{6} + \frac{1}{4} + \frac{17}{12} - 3 + \frac{7}{6}) + z^4(-\frac{1}{24} - \frac{1}{12} - \frac{17}{24} + 2 - \frac{7}{6}) + z^5(\frac{1}{120} + \frac{1}{48} + \frac{17}{72} - 1 + \frac{7}{9}) \\ &= \frac{31}{720} z^5 + O(z^6) \end{aligned}$$

and for the corrector (5.4) it is

$$C^C = 1 - e^{-z} - \frac{1}{2} z + \frac{1}{12} z^2 - \frac{1}{2} z e^{-z} - \frac{1}{12} z^2 e^{-z}$$

as a series

$$\begin{aligned} C^C &= 1 - (1 - z + \frac{1}{2} z^2 - \frac{1}{6} z^3 + \frac{1}{24} z^4 - \frac{1}{120} z^5) - \frac{1}{2} z + \frac{1}{12} z^2 \\ &\quad - \frac{1}{2} z (1 - z + \frac{1}{2} z^2 - \frac{1}{6} z^3 + \frac{1}{24} z^4) - \frac{1}{12} z^2 (1 - z + \frac{1}{2} z^2 - \frac{1}{6} z^3) \\ &= (1 - 1) + z(1 - \frac{1}{2} - \frac{1}{2}) + z^2(-\frac{1}{2} - \frac{1}{12} + \frac{1}{2} - \frac{1}{12}) + z^3(\frac{1}{6} - \frac{1}{4} + \frac{1}{12}) \\ &\quad + z^4(-\frac{1}{24} + \frac{1}{12} - \frac{1}{24}) + z^5(\frac{1}{120} - \frac{1}{48} + \frac{1}{72}) \\ &= \frac{1}{720} z^5 + O(z^6) \end{aligned}$$

Hence, the corrector is 31 times as accurate as the predictor. If we say that Y^* is the result obtained after the predicted part and Y is the result obtained after the correction, it looks as though the error of the method err can be approximated by

$$err = \frac{\|Y^* - Y\|}{30}. \quad (5.20)$$

Choosing stepsize

The next step for controlling stepsize lays in choosing the stepsize for the next step. We need to take in account the accuracy and the efficiency of the computation which defects the tolerance for chosen stepsize. If we say that the new stepsize is given by

$$h_{new} = r \cdot h$$

the task is now to find the constant r . There are known different approaches, one of the classical ones is to test when the error of the method err exceeds the chosen tolerance eps [72]. The following expression was determined for constant r such as

$$r = \left(\max \left(\min \left(0.9 \cdot \left(\frac{eps}{err} \right)^{\frac{1}{5}}, 2 \right), \frac{1}{2} \right) \right). \quad (5.21)$$

The error tolerance eps is unknown value and it differs in different examples. An estimated error err is always given by the difference between predicted value and corrected value in one step. The „magic“ numbers 2 and 1/2 guarantee the reasonable interval for chosen stepsize and the number 0.9 is a safety factor and also guarantee the good direction of choosing the new stepsize. The constant $1/(p + 1) = 1/5$ holds the order $p = 4$ of our method. This is an approach called Milne device [81].

The size of the tolerance is made to vary in such a way that methods will choose stepsize sequences that are reasonably close to the optimal ones. For standard tests we choose the tolerance to be

$$eps = 10^{-i}, \quad i = 2, \dots, 9.$$

The last step of choosing a variable stepsize algorithm is in adjusting data. After the new stepsize acceptance, data needs to be adjusted according to the chosen constant r . In our method using the Nordsieck implementation it means to multiply the data by the matrix

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & r & 0 & 0 & 0 \\ 0 & 0 & r^2 & 0 & 0 \\ 0 & 0 & 0 & r^3 & 0 \\ 0 & 0 & 0 & 0 & r^4 \end{bmatrix}$$

To show the algorithm of choosing the stepsize we experiment with Kepler problem. It is known that Kepler problem belongs to the group of Hamiltonians problem and the big question is if the method is symplectic and if it preserves the features of the structure. The method does not preserve the structure for the tolerance equals to $eps = 10^{-7}$ according to the figure 5.1a. The parasitic behaviour can be observed. But when we increase the requirement for the accuracy and decrease the tolerance to $eps = 10^{-10}$, we obtain better results. The error decreases even for longer time period 1000π , see figure 5.1b. And when we decrease the tolerance to the value $eps = 10^{-14}$ we can see the solution almost without errors, figure 5.2.

To discover how the variable stepsize is chosen, the algorithm is presented for one cycle with plotting each step, see figure 5.3. If we imagine the sun at coordinates $[0,0]$, we see

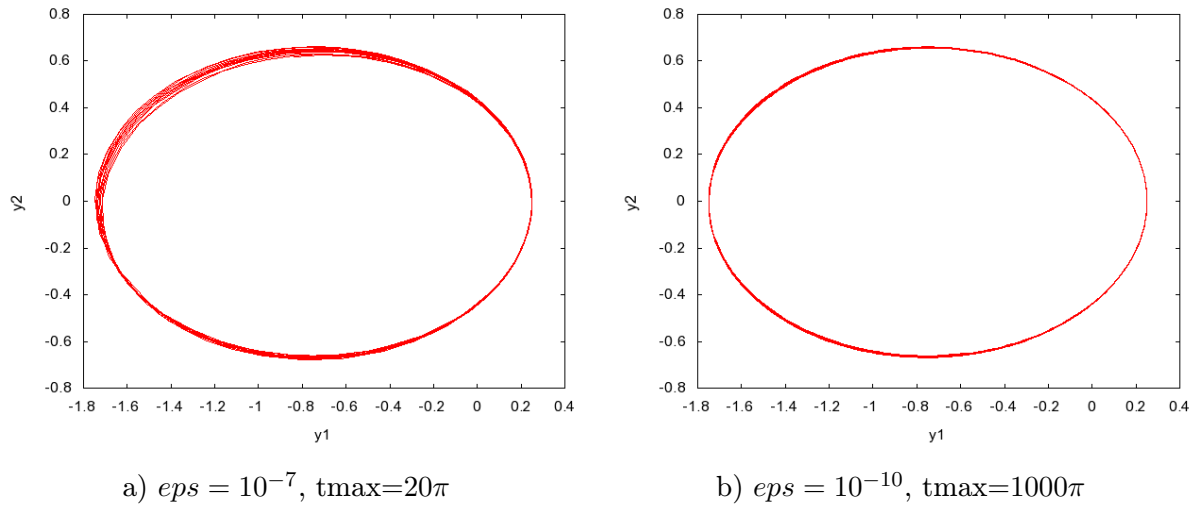


Figure 5.1: Kepler problem solutions

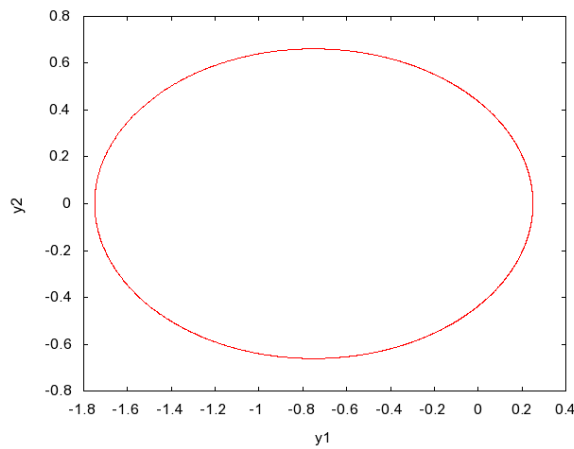


Figure 5.2: Kepler problem solution with $eps = 10^{-14}$, $tmax=1000\pi$

that the method is forced to calculate the trajectory of the planet using small stepsizes near the sun, the imaginary gravity is the biggest. The stepsize is bigger on the opposite side of the ellipse according to the smaller imaginary gravity.

Values of accepted and rejected stepsizes for same parameters of the problem are shown in figure 5.4. On part a) of the figure the algorithm is shown for time interval equals to 2π . Then we extend the time for algorithm to 8π and we observe different values of stepsize in each cycle, see figure 5.4b. We say that the big tolerance allows to choose stepsize during the algorithm differently. According to this fact the error of the method grows and the trajectory after few cycles differs as we saw in figure 5.1a.

Hence we prefer to preserve the high accuracy, so the chosen tolerance should reach the small tolerance possible. Results for choosing the stepsize with tolerance $eps = 10^{-14}$ are shown in figure 5.4c,d. We observe the algorithm accepts same stepsizes in each cycle.

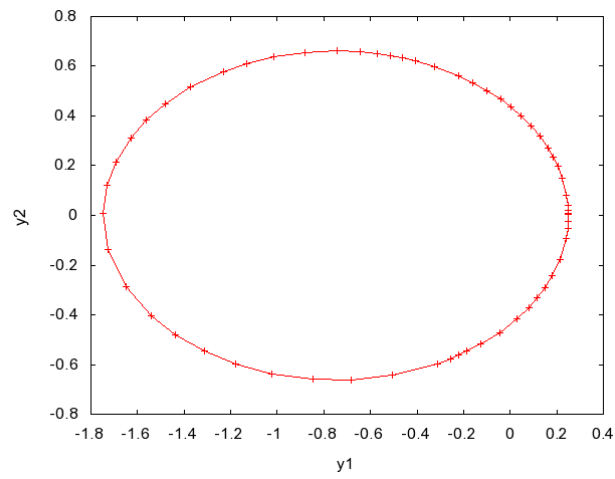


Figure 5.3: Kepler problem - choosing stepsizes - $eps = 10^{-7}$, $tmax=2\pi$

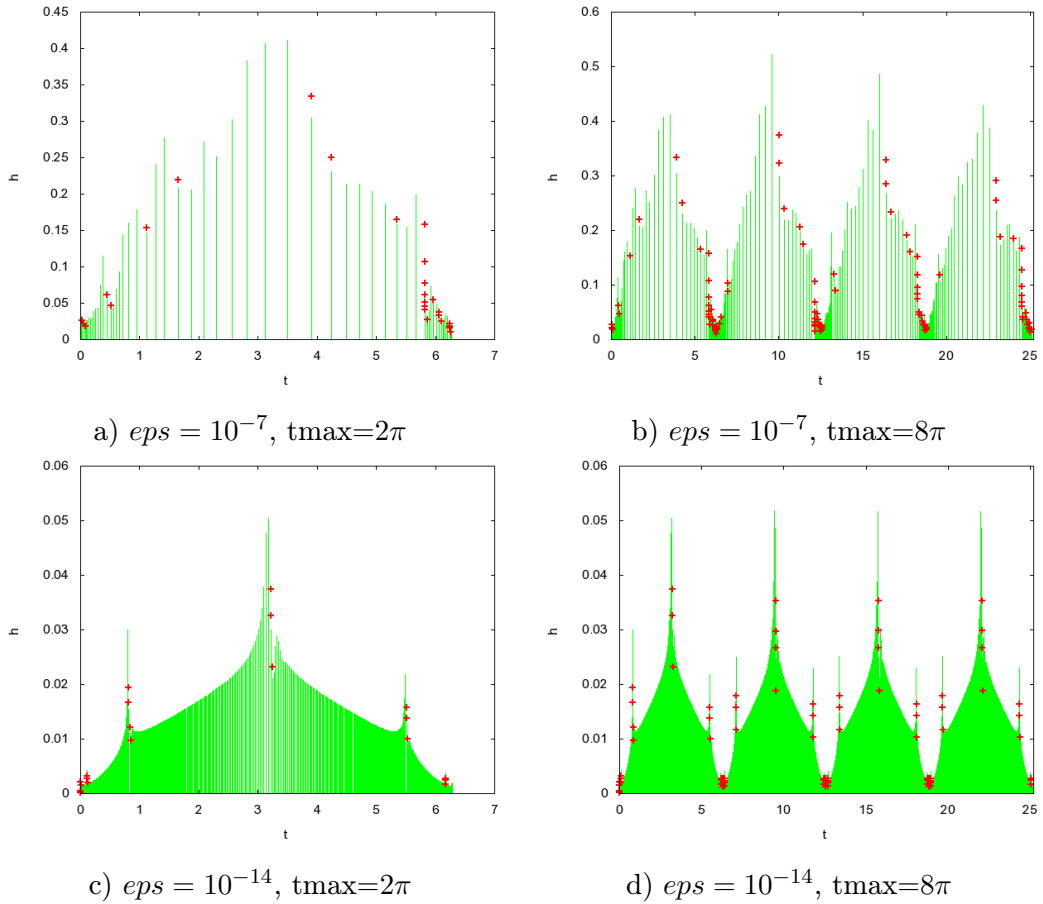


Figure 5.4: Kepler problem - accepted versus rejected stepsizes

5.3 Order of the method

To check the order of the new method we show the simplest Dahlquist problem (2.31) with constant $q = 1$. We proceed uniformly as described previously in the section 2.2. We calculate the error each time for n steps $h = (t_{max} - t_{min})/n$ for different n values such as $n = 10, 20, 40, \dots, 10240$, see table 5.3. We plot the order graph by the log of stepsizes on the x -axis and by the log of absolute values of errors on the y -axis.

Checking the slope of line through points we say that the order of new method (called *vlgm* and represented by violet line in figure 5.5 is 4. For comparison there are also displayed results for the same problem computed by Euler method (red line), which is the method of order 1, and Taylor series method of orders 2 (green line) and Taylor series method of order 4 (blue line). The corresponding errors for Euler method, Taylor series method of order 2 and of order 4 and for our method are determined in table 5.3. The interesting fact is that Taylor series method of order 4 is more precise than the classical Runge–Kutta of order 4.

Ensuring the method stability has order 4 the equation (5.22) should be satisfied.

$$\Phi(\exp(z), z) = O(z^5) \quad (5.22)$$

Table 5.3: Errors for Dahlquist problem of Euler method, Taylor series method and new method

h	err_{Euler}	err_{Ts2}	err_{Ts4}	err_{vlgm}
0.1	6.615560.1	4.200982e-03	2.084324e-06	1.147407e-05
$0.1 \cdot 2^{-1}$	3.490674e-01	1.090774e-03	1.358027e-07	7.462822e-07
$0.1 \cdot 2^{-2}$	1.794883e-01	2.778841e-04	8.666185e-09	4.757726e-08
$0.1 \cdot 2^{-3}$	9.103521e-02	7.012736e-05	5.473053e-10	3.003144e-09
$0.1 \cdot 2^{-4}$	4.584725e-02	1.761434e-05	3.438494e-11	1.886535e-10
$0.1 \cdot 2^{-5}$	2.300691e-02	4.413927e-06	2.155165e-12	1.187850e-11
$0.1 \cdot 2^{-6}$	1.152439e-02	1.104776e-06	1.350031e-13	6.847856e-13
$0.1 \cdot 2^{-7}$	5.767443e-03	2.763559e-07	4.884981e-15	4.279039e-14

Other two multistep methods were implemented and results for the problem were calculated for the comparison. The new method is motivated by Adams methods so to compare the new method with Adams method in PEC mode is natural. Hence, we implemented and calculate with Adams-Bashforth Adams-Moulton formulae of order 4 used in PEC mode (called ABAM4PEC and illustrated only in picture). And the other method is Adams-Bashforth method of order 4 (AdamBash4). Errors are displayed in table 5.4 and corresponding slopes of orders are plotted in the figure 5.6 for comparing those methods via

Table 5.4: Errors for Dahlquist problem of RK4, AB4 and new method *vlgm*

h	err_{RK4}	ratio	$err_{AdamBash4}$	ratio	err_{vlgm}	ratio
0.1	1.133156e-05		3.767292e-04		1.147407e-05	
$0.1 \cdot 2^{-1}$	7.383001e-07	15.348	2.761708e-05	13.641	7.462822e-07	15.375
$0.1 \cdot 2^{-2}$	4.711429e-08	15.670	1.865007e-06	14.808	4.757726e-08	15.686
$0.1 \cdot 2^{-3}$	2.975458e-09	15.834	1.211015e-07	15.400	3.003144e-09	15.842
$0.1 \cdot 2^{-4}$	1.869447e-10	15.916	7.713870e-09	15.699	1.886535e-10	15.919
$0.1 \cdot 2^{-5}$	1.170353e-11	15.973	4.866960e-10	15.849	1.187850e-11	15.882
$0.1 \cdot 2^{-6}$	7.265299e-13	16.109	3.055334e-11	15.929	6.847856e-13	17.346
$0.1 \cdot 2^{-7}$	6.394885e-14	11.361	1.900702e-12	16.075	4.279039e-14	16.003

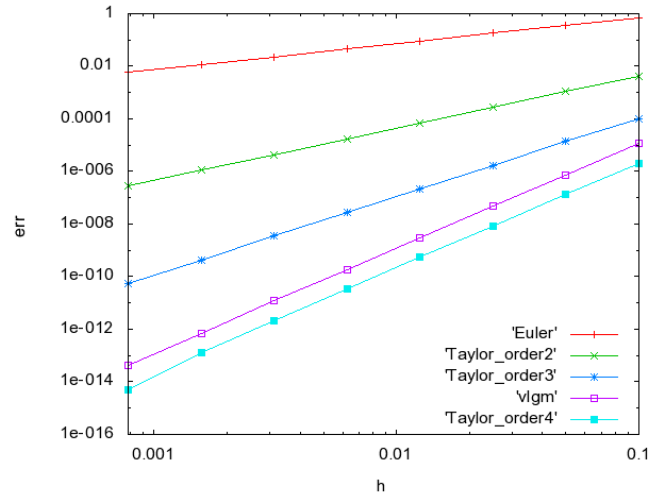


Figure 5.5: Errors and orders of different methods for Dahlquist problem with fixed stepsize

positions of lines. Satisfying fact is that our new method has comparable results with Runge–Kutta method of order four and it is more accurate than Adams-Bashforth Adams-Moulton formulae of order 4 used in PEC mode.

Then all methods are displayed in one graph for the comparison, see figure 5.7.

To show that our method is powerful for more difficult problem, we again present Kepler problem (2.53). Errors of methods are displayed in table 5.5. Graphs of orders are displayed on figure 5.8. For comparison there are Runge–Kutta method of order 4 (RK4) and Adams-Bashforth Adams-Moulton formulae of order 4 used in PEC mode (called ABAMPEC4 in picture, ABM in table) also plotted. Notice that solving the Kepler problem by numerical method such as Runge–Kutta method of order 4 with variable stepsize is difficult. Hence,

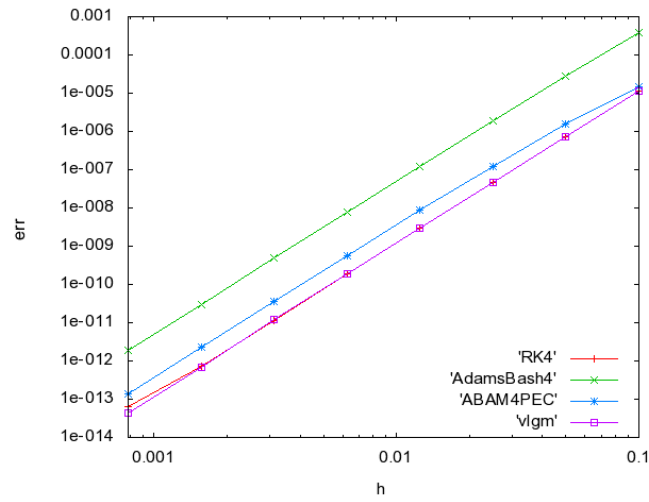


Figure 5.6: Errors and orders of four methods for Dahlquist problem with fixed stepsize

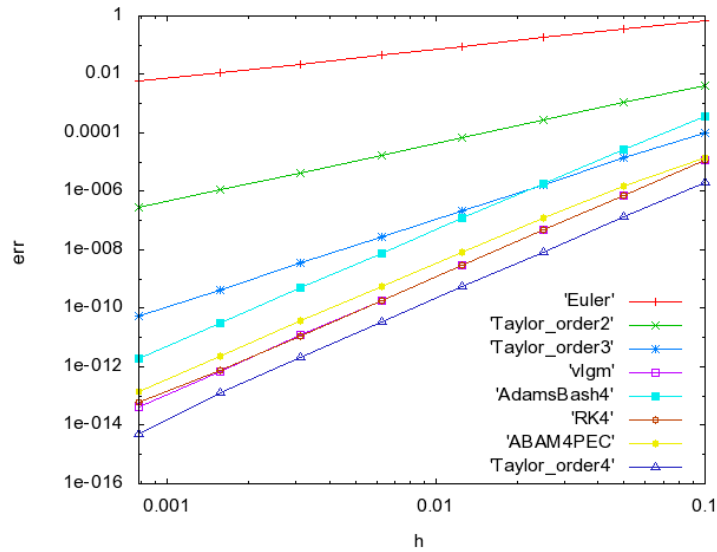


Figure 5.7: Errors and orders of all methods for Dahlquist problem with fixed stepsize

the Kepler problem was simplified by chosen eccentricity $e = 0$ and by usage of the fixed stepsize in this case. Then the fixed stepsize for each method can be used without involving truncation errors in to calculations.

Experiments show that the accuracy of the new method is comparable with classical and efficient Runge–Kutta method of order 4 and it has better stability properties than Adams-Bashforth Adams-Moulton formulae of order 4 used in PEC mode. Those results were expected. The main goal of order graphs was to verify expected results.

Experiments also discovered that the new method is more successful with using the variable stepsize in the calculation of Kepler problem.

Table 5.5: Errors for Kepler problem of methods with fixed stepsize and $e = 0$

stepsize h	error _{ABM}	error _{vlgm}
0.1570796	1.116667e-04	8.902963e-05
7.853982e-02	5.074034e-06	2.567462e-06
3.926991e-02	1.686544e-07	8.563243e-08
1.963495e-02	7.574493e-09	6.182604e-09
9.817477e-03	5.613075e-10	4.670812e-10
4.908739e-03	4.087373e-11	3.222429e-11
2.454369e-03	2.772263e-12	2.094210e-12
1.227185e-03	1.858035e-13	1.916790e-13

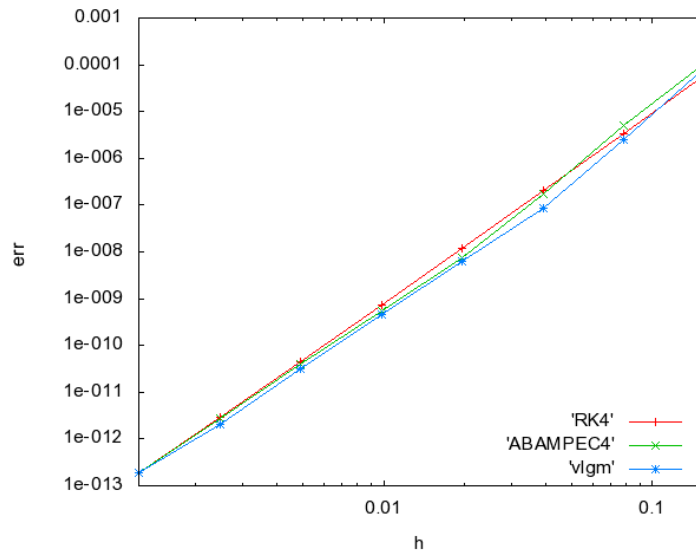


Figure 5.8: Orders for Kepler problem with fixed stepsize and $e = 0$

To show where the new method could be more or less successful than other ones, we present other experiments and tests with function evaluations, PEC and PECE modes and the stability regions. First, we describe the analysis of function evaluations in the next section.

5.4 Function evaluations

The Runge–Kutta method of order 4 requires four function evaluations for each step, which is its major disadvantage. A multistep method requires only one new function evaluation for each step. Reducing the number of function evaluations reduces the number of arithmetic operations involved and therefore reduces the total round-off error. Each evaluation takes time and the integrand may be arbitrarily complicated.

As an example, solving the Dahlquist problem (2.31) using n steps the Runge–Kutta method of order 4 requires $4n$ function evaluations. The Adams-Bashforth multistep method requires 16 function evaluations for the Runge–Kutta method of order 4 for the starting method and $n - 4$ for the n Adams-Bashforth steps, giving a total of $n + 12$ function evaluations for this method. In general the Adams-Bashforth multistep method requires slightly more than a quarter of the number of function evaluations required for the Runge–Kutta method of order 4. If the evaluation of $f(x, y)$ is complicated, the multistep method is more efficient [106].

For our experiment we use Adams-Bashforth Adams-Moulton formulae of order 4 used in PEC mode (called ABAMPEC4 in picture, ABM in table) instead of Adams-Bashforth method, but the number of function evaluations per step is the same. The new method *vlgm* requires 1 function evaluation per step and as it uses itself as the starting method, the total number of function evaluations for our method is $2n$.

We create a table 5.6 according known facts and numbers of function evaluations for each method.

Table 5.6: Function evaluations for Dahlquist problem

h	fe_{RK4}	fe_{vlgm}	fe_{ABM}
0.1	80	40	32
$0.1 \cdot 2^{-1}$	160	80	52
$0.1 \cdot 2^{-2}$	320	160	92
$0.1 \cdot 2^{-3}$	640	320	172
$0.1 \cdot 2^{-4}$	1280	640	332
$0.1 \cdot 2^{-5}$	2560	1280	652
$0.1 \cdot 2^{-6}$	5120	2560	1292
$0.1 \cdot 2^{-7}$	10240	5120	2572

We see that Runge–Kutta method of order 4 is more expensive method than the new method and the new method is two times more expensive than Adams-Bashforth Adams-

Moulton formulae of order 4 in PEC mode. The table 5.6 shows numbers of function evaluations which includes the starting procedures which are necessary for obtaining starting values for multistep method such as the new method and Adams-Bashforth Adams-Moulton formulae of order 4. Results are plotted in figure 5.9. The graph is again plotted by log scale, where the error at the end of the calculation is illustrated on the x -axis and the number of function evaluations in the completed calculation is illustrated on the y -axis. It holds that the lowest line is the most effective one. In this case it is Adams-Bashforth Adams-Moulton formulae of order 4 in PEC mode followed by the new method.

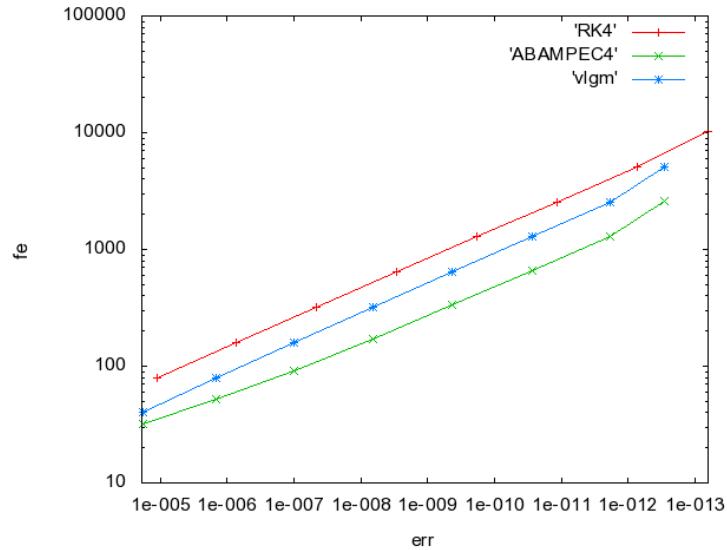


Figure 5.9: Function evaluations for Dahlquist problem

Now we demonstrate numbers of function evaluations for more difficult Kepler problem. Again we simplify the problem using fixed stepsize and eccentricity $e = 0$. And due to the fact that the Kepler problem is defined by four differential equations of first order, numbers of function evaluations are exactly four times bigger. The Runge–Kutta method of order four now requires 16 evaluations per step, which is $16n$ total function evaluations. The Adams-Bashforth Adams-Moulton formulae of order 4 in PEC mode requires 4 evaluations per step and it uses Runge–Kutta method of order 4 for the starting procedure, which includes -16 precomputed evaluations. So the total number of function evaluations for Adams-Bashforth Adams-Moulton formulae of order 4 in PEC mode is $4n + 48$. And our method requires $8n$ of total function evaluations including the starting method. Those facts are demonstrated in the table 5.7 and the graph gives similar comparison as for the Dahlquist problem, the Runge–Kutta method of order 4 is the most expensive method, our method is less expensive and the Adams-Bashforth Adams-Moulton formulae of order 4 in PEC mode costs minimum.

Table 5.7: Function evaluations for Kepler problem

h	fe_{RK4}	fe_{vlgm}	fe_{ABM}
0.1	320	160	128
$0.1 \cdot 2^{-1}$	640	320	208
$0.1 \cdot 2^{-2}$	1280	640	368
$0.1 \cdot 2^{-3}$	2560	1280	688
$0.1 \cdot 2^{-4}$	5120	2560	1328
$0.1 \cdot 2^{-5}$	10240	5120	2608
$0.1 \cdot 2^{-6}$	20480	10240	5168
$0.1 \cdot 2^{-7}$	40960	20480	10288

For a conclusion we need to add that the f -function evaluation and the g -function evaluation cost the same for our new method solving Dahlquist problem. Solving the Kepler problem via new method the g -function evaluation costs more than f -function evaluation. We see that the cost of the calculation descends with the complexity of the problem and the accuracy is preserved.

5.5 PEC and PECE modes

We are concerned about the different modes of the method. Our method is represented in a PEC mode. We discovered that as we repeat the evaluate–correct step one more time after one cycle of predict–evaluate–correct procedure, results will be improved in the accuracy point of view according to experiments. It means in the pseudo code

```
% step 1. predict
Ypred = P * Yinp;

% step 2. evaluate functions f, g;
f=eval(yprime);
g=eval(ydoubleprime);

% step 3. calculate corrections
delta = h * f - Ypred(2);
epsilon = 1/2 * h^2 * g - Ypred(3);
Yout = Ypred + (delta * alpha) + (epsilon * beta);
```

after steps 1., 2. and 3. (called PEC mode) we again calculate steps 2. and 3. (called PECE mode), errors are smaller for PECE mode. Those results are expected and it corresponds to the behaviour of classic PEC–PECE implementation of Adams methods.

As we repeat steps 2. and 3. one more time and we call it PECECE (or $PE(CE)^2$) mode, results are still improved according to a PEC mode, but they are slightly less accurate than errors for a PECE mode. Those result were also expected, this behaviour occurs in some problems even for classical Adams methods in corresponding modes.

The Prothero–Robinson problem (5.19) was chosen for the demonstration. Three modes PEC, PECE and $PE(CE)^2$ are plotted in the graph 5.10a. Slopes represent orders of our methods in three different modes. It is not very clear that error values of PECE and $PE(CE)^2$ are very close to each other, even if the graph is plotted for smaller error interval, see 5.10b.

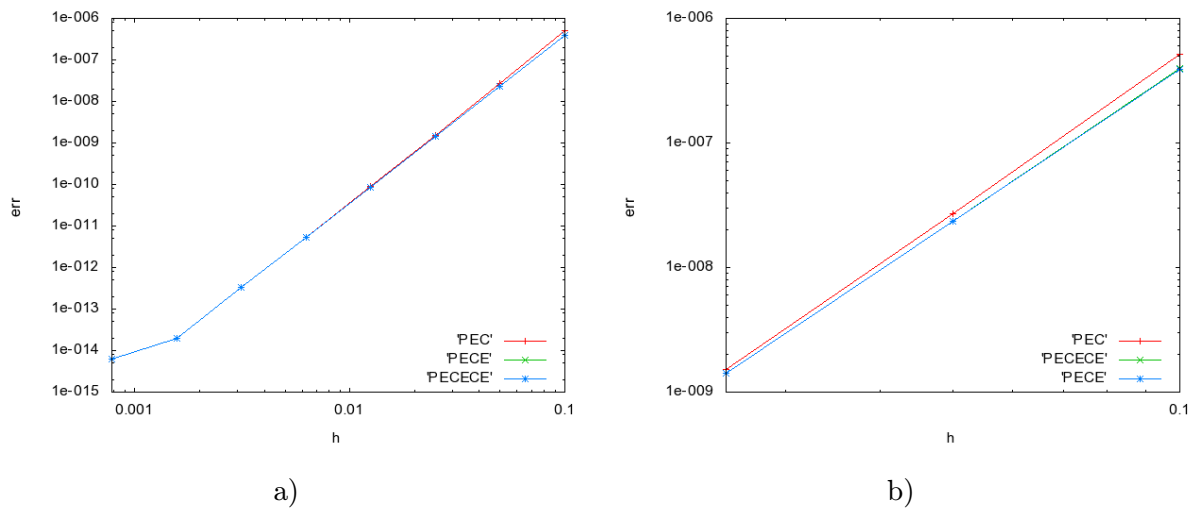


Figure 5.10: New method modes' in Prothero–Robinson problem

To specify the accuracy for Prothero–Robinson problem with fixed stepsize more precisely, errors are written in tables. As expected the PECE mode brings improvement into calculation. Errors of PECE mode are smaller than errors of PEC mode, see the table 5.8. Ratios numbers represent 2^p with the order p of the new method in corresponding mode. The column ratio after the column error $_{PEC}$ represents 2^p with the order p of the new method in PEC mode and it is calculated by the error number divided by the error number below it. The column ratio after the column error $_{PECE}$ represents the order of the new method in PECE mode. Notice that the order is more closer to the expected order 4 of our method for smaller stepsizes. And also the order is more closer to the expected order 4 of our method for the mode PECE than the mode PEC. The value around 3 in the last row is influenced by using very small stepsize and also with round-of error.

The procedure for $PE(CE)^2$ mode with two more evaluate–correct steps also brings improvement into calculation of PEC mode, but it is less effective than using only one repetition, in the table 5.9 you can observe that errors of $PE(CE)^2$ mode are bigger than errors of PECE mode.

Table 5.8: Errors for Prothero–Robinson problem of our method in different modes

h	$error_{PEC}$	ratio	$error_{PECE}$	ratio
0.1	5.197512e-07		3.928242e-07	
$0.1 \cdot 2^{-1}$	2.717284e-08	19.128	2.338723e-08	16.780
$0.1 \cdot 2^{-2}$	1.528272e-09	17.780	1.413994e-09	16.540
$0.1 \cdot 2^{-3}$	9.020651e-11	16.942	8.671031e-11	16.307
$0.1 \cdot 2^{-4}$	5.472844e-12	16.483	5.365375e-12	16.161
$0.1 \cdot 2^{-5}$	3.359535e-13	16.290	3.330669e-13	16.109
$0.1 \cdot 2^{-6}$	1.942890e-14	17.291	1.931788e-14	17.241
$0.1 \cdot 2^{-7}$	6.328271e-15	3.070	6.328271e-15	3.053

Table 5.9: Errors for Prothero–Robinson problem of our method in different modes

h	$error_{PECE}$	ratio	$error_{PE(CE)^2}$	ratio
0.1	3.928242e-07		3.999541e-07	
$0.1 \cdot 2^{-1}$	2.338723e-08	16.780	2.348635e-08	17.029
$0.1 \cdot 2^{-2}$	1.413994e-09	16.540	1.415450e-09	16.593
$0.1 \cdot 2^{-3}$	8.671031e-11	16.307	8.673218e-11	16.320
$0.1 \cdot 2^{-4}$	5.365375e-12	16.161	5.365819e-12	16.164
$0.1 \cdot 2^{-5}$	3.330669e-13	16.109	3.330669e-13	16.110
$0.1 \cdot 2^{-6}$	1.931788e-14	17.241	1.942890e-14	17.143
$0.1 \cdot 2^{-7}$	6.328271e-15	3.053	6.328272e-15	3.070

5.6 Stability analysis

The stability analysis for two-derivative multistep method is presented in this section. For plotting the stability region we use predictor and corrector equations of our method.

We present stability regions of our method in picture 5.11. For the PECE mode holds that only the region to the left of zero is stable. Regions to the right of the zero are very unstable because they have both eigenvalues outside the unit disc.

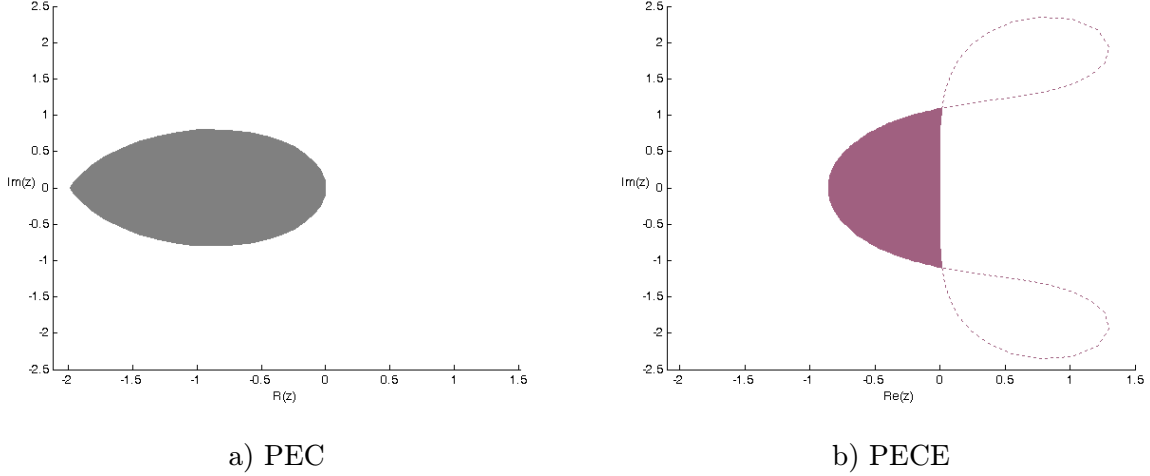


Figure 5.11: Stability regions of the new method

To illustrate how the stability regions can be plotted, we take the PECE mode of our method. We remind the predictor formula of the method

$$y(x_n) = y_{n-1} - \frac{1}{2}hf_{n-1} + \frac{3}{2}hf_{n-2} + \frac{17}{12}h^2g_{n-1} + \frac{7}{12}h^2g_{n-2}$$

and the corrector formula the method

$$y(x_n) = y_{n-1} + \frac{1}{2}hf_n + \frac{1}{2}hf_{n-1} - \frac{1}{12}h^2g_n + \frac{1}{12}h^2g_{n-1}$$

and we rewrite them such as $z = hq$

$$y_n^* = \left(1 - \frac{z}{2} + \frac{17}{12}z^2\right)y_{n-1} + \left(\frac{3z}{2} + \frac{7}{12}z^2\right)y_{n-2}, \quad (5.23)$$

$$y_n = \left(\frac{1}{2}z - \frac{1}{12}z^2\right)y_n^* + \left(1 + \frac{1}{2}z + \frac{1}{12}z^2\right)y_{n-1}, \quad (5.24)$$

where y_n^* represents the predicted value. We use the predicted formula to update the corrector formula only in the current step in the case of PECE mode as already indicated in equation (5.24). Thus we substitute the equation (5.23) into the equation (5.24), we get

$$y_n = \left(1 + z - \frac{1}{4}z^2 + \frac{3}{4}z^3 - \frac{17}{144}z^4\right)y_{n-1} + \left(\frac{3}{4}z + \frac{1}{6}z^3 - \frac{7}{144}z^4\right)y_{n-2}. \quad (5.25)$$

The matrix notation can be used

$$\begin{bmatrix} y_n \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} 1 + z - \frac{1}{4}z^2 + \frac{3}{4}z^3 - \frac{17}{144}z^4 & \frac{3}{4}z + \frac{1}{6}z^3 - \frac{7}{144}z^4 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} y_{n-1} \\ y_{n-2} \end{bmatrix} \quad (5.26)$$

and the stability region is calculated and plotted.

The analysis of PEC modes is given by the comparison of Adams-Bashforth Adams-Moulton method of order 4 (ABAM4) and the new method, see figure 5.12. It is satisfying that the size of stability region is bigger for our method than the stability region of Adams-Bashforth Adams-Moulton method of order 4. The same characteristics holds for the PECE mode.

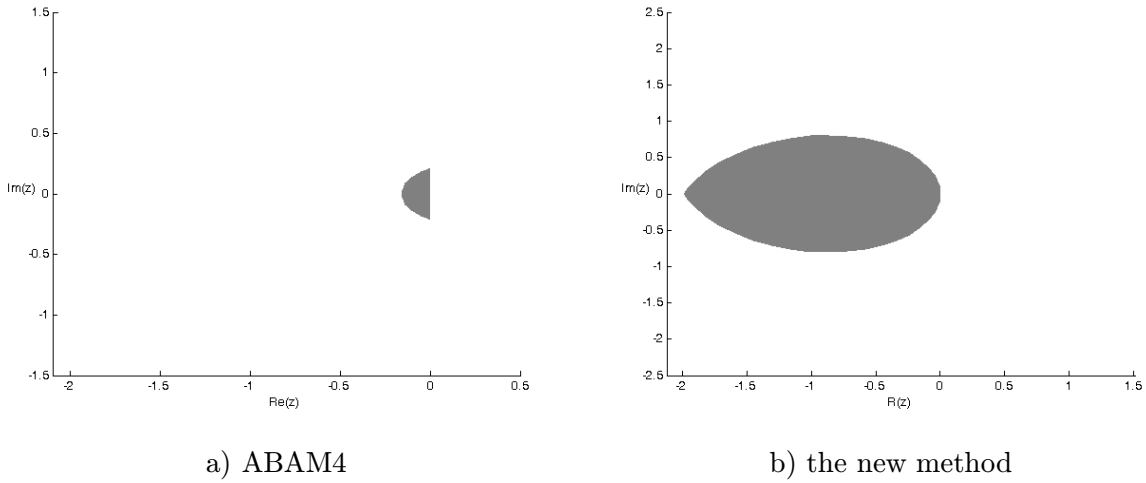


Figure 5.12: Stability regions comparison of PEC modes

Chapter 6

Conclusions

In this thesis, we explored several numerical methods. We have explored the relationship between stability and convergence for chosen numerical methods for solving non-stiff problems. A wide class of Runge–Kutta methods have been described and very interesting approach of rooted trees was introduced to analyze the order of accuracy of these methods. The Taylor series was introduced in the view of the systematic patterns where the motivation was found in elementary differentials notation and in rooted trees notation. Linear multistep methods were also introduced and important stability aspect of the predictor–corrector modes of Adams method was analyzed.

The contribution of the thesis is the new two-derivative method of order four. The method is combined in nontraditional way in the form of Obreshkov formulae using variable stepsize in the predictor–corrector pairs. It has been shown that the method is usable for solving linear as well as non-linear problems.

Convergence and stability analysis for the predictor–corrector method in Obreshkov quadrature formulae with constant stepsize for various problems have been shown as well as the comparison between the new method, classical Runge–Kutta method of order four, Taylor series method, Adams–Bashforth Adams–Moulton method of order four in PEC mode and others selected numerical methods. The new method with fixed stepsize and also with variable stepsize was implemented and tested in various problems. The implementation of the new method in different modes PEC and PECE was tested.

Predictor–corrector methods are often preferred over Runge–Kutta methods for the numerical solution of ordinary differential equations, since the former may involve fewer derivative evaluations per step. It has been suggested that the number of function evaluations can be reduced in a way of implementing predictor–corrector method in Nordsieck representation. The new two-derivative multistep method has been introduced and the method turned out to be just as reliable as the traditional methods. The cost of new method decreases with the complexity of the problem and the accuracy is preserved. The

higher order of new method will be more accurate than the classical Adams method.

The size of the stability region for the resulting algorithm is still small, but the stability region is larger than commonly used methods such as Adams–Bashforth Adams–Moulton method of order four in PEC mode or Adams–Bashforth method of order 4. Hence, the new algorithm may be of interest of applications where stability is a strong limitation.

The differential equations and some systems of differential equations with initial conditions from DETEST [64] were successfully tested. Tests examples were implemented and some experiments were chosen and described. There were no anomalies in the behaviour of the method for this reason only a few examples were chosen to be described in detail. Detailed results for the individual methods were collected.

The performance of the method can be improved, especially the algorithm for choosing the stepsize. Suggested approach is to implement the PI control in the future. It has been proved that a convergence method of some fixed order is always better than a method of lower order provided the tolerance is stringent enough. If a method is to be a good general-purpose method, and hence perform wide range of tolerances, it must be able to choose its own order. And that is another goal for future work.

Bibliography

- [1] Al-Rabeh, A. H. Optimal order diagonally implicit Runge–Kutta methods. *BIT Numerical Mathematics*, 1993, volume 33, no. 4, pp. 619–633, ISSN 1572-9125.
- [2] Axelsson, O. A class of A-stable methods. *BIT Numerical Mathematics*, 1969, volume 9, no. 3, pp. 185–199, ISSN 1572-9265.
- [3] Barton, D. On Taylor series and stiff equations. *ACM Transactions on Mathematical Software*, 1980, volume 6, no. 32, pp. 280–294, ISSN 0098-3500.
- [4] Barton, D., Willers, I. M., Zahar, R. V. M. Taylor series methods for ordinary differential equations—an evaluation. *Mathematical Software Symposium*, 1970, volume 14, no. 3, pp. 243–248.
- [5] Barton, D., Willers, I. M., Zahar, R. V. M. The automatic solution of ordinary differential equations by the method of Taylor series. *Computing*, 1971, volume 14, no. 3, pp. 243–248, ISSN 2151-9617.
- [6] Bashforth, F., Adams, J. C. An attempt to test the theories of capillary action by comparing the theoretical and measured forms of drops of fluid. With an explanation of the method of integration employed in constructing the tables which give the theoretical forms of such drops, by J. C. Adams. *University Press Cambridge*, 1883.
- [7] Bulirsch, R., Stoer, J. Numerical treatment of ordinary differential equations by extrapolation methods. *Numerical Mathematics*, 1966, volume 8, no. 1, pp. 1 – 13, ISSN 1569-3953.
- [8] Burden, R. L., Faires, J. D. *Numerical analysis*. Brooks Cole, 2004, ISBN 0-534-39200-8.
- [9] Burrage, K. Order and stability properties of explicit multivalued methods. *Applied Numerical Mathematics*, 1985, volume 1, no. 5, pp. 363–379, ISSN 0168-9274.
- [10] Burrage, K., Moss, P. Simplifying assumptions for the order of partitioned multivalued methods. *BIT Numerical Mathematics*, 1980, volume 20, no. 4, pp. 452–465, ISSN 0001-0782.

- [11] Butcher, J. C. The effective order of Runge–Kutta methods. *Lecture Notes in Mathematics*, 1969, volume 109, pp. 133–139, ISSN 0075-8434.
- [12] Butcher, J. C. A stability property of implicit Runge-Kutta methods. *BIT Numerical Mathematics*, 1975, volume 15, no. 4, pp. 358–361, ISSN 1572-9125.
- [13] Butcher, J. C. On A-stable Runge–Kutta methods. *BIT Numerical Mathematics*, 1977, volume 17, pp. 375–378, ISSN 1572-9125.
- [14] Butcher, J. C. On the convergence of numerical solutions to ordinary differential equations. *Mathematics of Computation*, 1985, volume 20, no. 93, pp. 1–10, ISSN 1088-6842.
- [15] Butcher, J. C. Optimal order and stepsize sequences. *IMA Journal of Applied Mathematics*, 1986, volume 6, no. 4, pp. 433–438, ISSN 1464-3634.
- [16] Butcher, J. C. *The numerical analysis of ordinary differential equations: Runge–Kutta and general linear methods*. John Wiley & Sons, 1987, ISBN 0-471-91046-5.
- [17] Butcher, J. C. Diagonally-implicit multi-stage integration methods. *Applied Numerical Mathematics*, 1993, volume 11, no. 5, pp. 347–363, ISSN 0168-9274.
- [18] Butcher, J. C. Some orbital test problems. *Computing*, 1994, volume 53, no. 1, pp. 75 – 94, ISSN 2151-9617.
- [19] Butcher, J. C. A history of Runge–Kutta methods. *Applied Numerical Mathematics*, 1996, volume 20, no. 1, pp. 247–260, ISSN 0168-9274.
- [20] Butcher, J. C. General linear methods for stiff differential equations. *BIT Numerical Mathematics*, 2001, volume 41, no. 2, pp. 240–264, ISSN 1572-9125.
- [21] Butcher, J. C. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2008, ISBN 978-0-470-72335-7.
- [22] Butcher, J. C., Burrage, K. Stability criteria for implicit Runge–Kutta methods. *SIAM Journal on Numerical Analysis*, 1979, volume 16, no. 1, pp. 46–57, ISSN 1095-7170.
- [23] Butcher, J. C., Chen, D. J. A new type of singly-implicit Runge–Kutta method. *Numerical Algorithms*, 2000, volume 34, no. 2-3, pp. 179–188, ISSN 1572-9265.
- [24] Butcher, J. C., Chipman, F. H. Generalized Padé approximations to the exponential function. *BIT Numerical Mathematics*, 1992, volume 32, no. 1, pp. 118–130, ISSN 1572-9125.

- [25] Butcher, J. C., Diamantakis, M. T. DESIRE: diagonally extended singly implicit Runge–Kutta effective order methods. *Numerical Algorithms*, 1998, volume 27, no. 1-2, pp. 121–145, ISSN 1572-9265.
- [26] Butcher, J. C., Heard, A. D. Stability of numerical methods for ordinary differential equations. *Numerical Algorithms*, 2002, volume 31, no. 1-4, pp. 59–73, ISSN 1572-9265.
- [27] Butcher, J. C., Jackiewicz, Z. Error estimation for Nordsieck methods. *Numerical Algorithms*, 2002, volume 31, no. 1-4, pp. 75–85, ISSN 1572-9265.
- [28] Butcher, J. C., Johnson, P. B. Estimating local errors for Runge–Kutta methods. *Computational & Applied Mathematics*, 1993, volume 45, no. 1-2, pp. 203–212, ISSN 1807-0302.
- [29] Chang, Y. F. Automatic solution of differential equations. *Lecture Notes in Mathematics*, 1974, volume 430, pp. 61–94, ISSN 0075-8434.
- [30] Chase, P. E. Stability properties of predictor-corrector methods for ordinary differential equations. *Journal of the ACM*, 1962, volume 9, no. 4, pp. 457–468, ISSN 0004-5411.
- [31] Cooper, G. J., Verner, J. H. Some explicit Runge–Kutta methods of high order. *SIAM Journal on Numerical Analysis*, 1972, volume 9, no. 3, pp. 389–405, ISSN 1095-7170.
- [32] Corliss, G., Chang, Y. F. Solving ordinary differential equations using Taylor Series. *Journal of the ACM*, 1982, volume 8, no. 2, pp. 457–473, ISSN 0004-5411.
- [33] Corliss, G., Lowery, D. Choosing a stepsize for Taylor series methods for solving ODEs. *Lecture Notes in Mathematics*, 1977, volume 3, no. 4, pp. 251–256, ISSN 0075-8434.
- [34] Crane, R. L., Lambert, R. J. Stability of a generalized corrector formula. *Journal of the ACM*, 1962, volume 9, no. 1, pp. 104–117, ISSN 0004-5411.
- [35] Curtis, A. R. High-order explicit Runge–Kutta formulae, their uses, and limitations. *IMA Journal of Applied Mathematics*, 1975, volume 16, no. 1, pp. 35–52, ISSN 1464-3634.
- [36] Dahlquist, G. G. Convergence and stability in the integration of ordinary differential equations. *Mathematica Scandinavica*, 1956, volume 4, pp. 33–50, ISSN 1903-1807.

- [37] Dahlquist, G. G. Stability and error bounds in the numerical integration of ordinary differential equations. *Transactions of the Royal Institute of Technology*, 1959, volume 41, no. 130, ISSN 0374-3624.
- [38] Dahlquist, G. G. A special stability problem for linear multistep methods. *BIT Numerical Mathematics*, 1963, volume 3, no. 1, pp. 1–13, ISSN 1572-9125.
- [39] Dahlquist, G. G. *Numerical methods*. Prentice-Hall Inc., 1974, ISBN 0136273157.
- [40] Dahlquist, G. G. On accuracy and unconditional stability of linear multistep methods for second order differential equations. *BIT Numerical Mathematics*, 1978, volume 18, no. 2, pp. 133–136, ISSN 1572-9125.
- [41] Edwards, C. H., Penney, D. E. *Differential equations & Linear algebra*. Prentice/Hall, Inc., 2001, ISBN 0-13-973751-0.
- [42] Enright, W. H. Second derivative multistep methods for stiff systems. *Journal of the ACM*, 1964, volume 11, no. 2, pp. 188–209, ISSN 0004-5411.
- [43] Enright, W. H., Hull, T. E. Test results on initial value methods for non-stiff ordinary differential equations. *SIAM Journal on Numerical Analysis*, 1976, volume 13, no. 6, pp. 944 – 961, ISSN 1095-7170.
- [44] Enright, W. H., Hull, T. E., Lindberg, B. Comparing numerical methods for stiff systems of O.D.E:s. *BIT Numerical Mathematics 15*, 1975, volume 15, no. 1, pp. 10 – 48, ISSN 1572-9125.
- [45] Fatunla, S. O. *Numerical methods for initial value problems in ordinary differential equations*. Academic Press, Inc., 1988, ISBN 0-12-249930-1.
- [46] Fujii, M. An extension of Milne’s device for the Adams Predictor-Corrector Methods. *Japan Journal of Industrial and Applied Mathematics*, 1991, volume 8, no. 1, pp. 1–18, ISSN 0916-7005.
- [47] Gear, C. W. The automatic integration of ordinary differential equations. *Communications of the ACM*, 1971, volume 14, no. 3, pp. 176–179, ISSN 0001-0782.
- [48] Gear, C. W. *Numerical initial value problems in ordinary differential equations*. Prentice-Hall Inc., 1971, ISBN 0136266061.
- [49] Gear, C. W., Stetter, H. J. Analysis of discretization methods for ordinary differential equations. *Mathematics of Computation*, 1974, volume 28, no. 127, ISSN 1088-6842.

- [50] Gear, C. W., Watanabe, D. S. Stability and convergence of variable order multistep methods. *SIAM Journal on Numerical Analysis*, 1974, volume 11, no. 5, pp. 1044–1058, ISSN 1095-7170.
- [51] Gear, W. C. Runge–Kutta starters for multistep methods. *ACM Transactions on Mathematical Software*, 1980, volume 6, no. 3, pp. 263–279, ISSN 0098-3500.
- [52] Gibbons, A. A program for the automatic integration of differential equations using the method of Taylor series. *Computing*, 1960, volume 3, no. 2, pp. 108–111, ISSN 2151-9617.
- [53] Gragg, W. B., Stetter, H. J. Generalized multistep predictor–corrector methods. *Journal of the ACM*, 1964, volume 11, no. 2, pp. 188–209, ISSN 0004-5411.
- [54] Gustafsson, K., Lundh, M., Söderlind, G. A PI stepsize control for the numerical solution of ordinary differential equations. *BIT Numerical Mathematics*, 1988, volume 28, no. 2, pp. 270–287, ISSN 1572-9125.
- [55] Hairer, E., Lubich, C. Symmetric multistep methods over long times. *Numerical Algorithms*, 2004, volume 97, no. 4, pp. 699–723, ISSN 1572-9265.
- [56] Hairer, E., Nørsett, S. P., Wanner, G. Order stars and stability theorems. *BIT Numerical Mathematics*, 1978, volume 18, no. 4, pp. 475–489, ISSN 1572-9125.
- [57] Hairer, E., Nørsett, S. P., Wanner, G. *Solving ordinary differential equations I: Nonstiff problems*. Springer Verlag, 1993, ISBN 3-540-56670-8.
- [58] Hairer, E., Wanner, G. Multistep-multistage-multiderivative methods for ordinary differential equations. *Computing*, 1973, volume 11, no. 3, pp. 287–303, ISSN 2151-9617.
- [59] Hairer, E., Wanner, G. Algebraically stable and implementable Runge–Kutta methods of high order. *SIAM Journal on Numerical Analysis*, 1981, volume 18, no. 6, pp. 1098–1108, ISSN 1095-7170.
- [60] Hairer, E., Wanner, G. *Solving ordinary differential equations II: Stiff and differential-algebraic problems*. Springer, 1996, ISBN 978-3-540-60452-5.
- [61] Hamming, R. W. Stable predictor-corrector methods for ordinary differential equations. *Journal of the ACM*, 1959, volume 6, no. 1, pp. 37–47, ISSN 0004-5411.
- [62] Henrici, P. K. *Discrete variable methods in ordinary differential equations*. John Wiley & Sons, 1962, ISBN 978-0471372240.

- [63] Hull, T. E., Creemer, A. L. Efficiency of predictor–corrector procedures. *Journal of the ACM*, 1963, volume 10, no. 3, pp. 291–301, ISSN 0004-5411.
- [64] Hull, T. E., Enright, W. H., Fellen, B. M., et al. Comparing numerical methods for ordinary differential equations. *SIAM Journal on Numerical Analysis*, 1972, volume 9, no. 4, pp. 603 – 637, ISSN 1095-7170.
- [65] Hull, T. E., Johnston, R. L. Optimum Runge–Kutta methods. *Mathematics of Computation*, 1964, volume 18, no. 86, pp. 306–306, ISSN 1088-6842.
- [66] Iserles, A., Nørsett, S. P. *Order stars*. Chapman & Hall, 1991, ISBN 0412352605.
- [67] Kahan, W. Pracniques: further remarks on reducing truncation errors. *Communications of the ACM*, 1965, volume 8, no. 1, pp. 1–11, ISSN 0001-0782.
- [68] Kalas, J., Ráb, M. *Obyčejné diferenciální rovnice*. Masarykova universita, Brno, 2001, ISBN 80-210-2589-1.
- [69] Khamsi, M. A. Differential equations.
<http://www.sosmath.com/diffeq/diffeq.html>, 1999 [cit. 2009-12-07].
- [70] Klopfenstein, R. W., Davis, C. B. PECE algorithms for the solution of stiff systems of ordinary differential equations. *Mathematics of Computation*, 1971, volume 25, no. 115, pp. 457–473, ISSN 1088-6842.
- [71] Krogh, F. T. Predictor–corrector methods of high order with improved stability characteristics. *Journal of the ACM*, 1966, volume 13, no. 3, pp. 374–385, ISSN 0004-5411.
- [72] Krogh, F. T. Algorithms for changing stepsize. *SIAM Journal on Numerical Analysis*, 1973, volume 10, no. 5, pp. 949–965, ISSN 1095-7170.
- [73] Krogh, F. T. Stepsize selection for ordinary differential equations. *ACM Transactions on Mathematical Software*, 2010, volume 37, no. 2, pp. 1–11, ISSN 0098-3500.
- [74] Kubíček, M., Dubcová, M., Janovská, D. *Numerické metody a algoritmy*. VŠCHT Praha, 2005, ISBN 80-7080-558-7.
- [75] Kunovský, J. *Modern Taylor series method*. FEI VUT Brno, 1995, habilitation work.
- [76] Kunovský, J., et al. Using differential equations in electrical circuits simulation. *Journal of Autonomic Computing*, 2009, volume 1, no. 2, pp. 192–201, ISSN 1741-8569.

- [77] Kunovský, J., Mikulášek, K. TKSL-Taylor Kunovsky Simulation Language. *Transputers Applications and Systems*, 1992, pp. 161–166, ISBN 0444893318.
- [78] Kunovský, J., Sehnalová, P., Šátek, V. Stability and convergence of the Modern Taylor series method. Proceedings of UKSim 10th International Conference EUROSIM/UKSim2008, 2010, pp. 33–38, ISBN 0-7695-3114-8.
- [79] Kunovský, J., Zbořil, F. Numerical solutions of PDE's by the Modern Taylor series method. Proceedings MOSIS '93, 1993, pp. 43–47, ISBN 80-85988-86-0.
- [80] Meurant, G. *Computer solution of large linear system*. Elsevier Science B.V., 1999, ISBN 0-444-50169-X.
- [81] Milne, W. E. *Numerical calculus*. Princeton University Press, 1949, ISBN 978-0-691-08011-6.
- [82] Milne, W. E. *Numerical solution of differential equations*. Dover Pubns, 1970, ISBN 978-0-486-62437-2.
- [83] Nagle, R. K., Saff, E. B., Snider, A. D. *Fundamentals of differential equations*. Addison-Wesley, 1996, ISBN 0-201-80875-7.
- [84] Nordsieck, A. On numerical integration of ordinary differential equations. *Mathematics of Computation*, 1962, volume 16, pp. 22–49, ISSN 1088-6842.
- [85] Nørsett, S. P. One-step methods of Hermite type for numerical integration of stiff systems. *BIT Numerical Mathematics*, 1974, volume 14, no. 3, pp. 63–77, ISSN 1572-9125.
- [86] Obreshkov, N. *Sochineniya. Tom 2*. Bŭlgar. Akad. Nauk, 1981, ISBN 0933884966.
- [87] Peterson, G. L., Sochacki, J. S. *Linear algebra and differential equations*. Addison-Wesley, 2002, ISBN 0-201-66212-4.
- [88] Piotrowsky, P. Stability, consistency and convergence of variable k-step methods for numerical integration of large systems of ordinary differential equations. *Lecture Notes in Mathematics*, 1969, volume 109, pp. 221–227, ISSN 0075-8434.
- [89] Prothero, A., Robinson, A. On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations. *Mathematics of Computation*, 1974, volume 28, no. 125, pp. 145–145, ISSN 1088-6842.
- [90] Salleh, S., Zomaya, A. Y., Bakar, S. A. *Computing for numerical methods using Visual C++*. Wiley, 2007, ISBN 978-0-470-12795-7.

- [91] Sanz-Serna, J. M. Runge–Kutta schemes for Hamiltonian systems. *BIT Numerical Mathematics*, 1988, volume 28, no. 4, pp. 877–883, ISSN 1572-9125.
- [92] Schmitt, K., Thompson, R. C. Nonlinear analysis and differential equations. <http://www.math.utah.edu/schmitt/ode1.pdf>, 2004 [cit. 2008-11-28].
- [93] Shampine, L. F. Some practical Runge–Kutta formulas. *Mathematics of Computation*, 1986, volume 46, no. 173, pp. 135–150, ISSN 1088-6842.
- [94] Shampine, L. F. *Numerical solution of ordinary differential equations*. Chapman & Hal, 1994, ISBN 978-0-412-05151-7.
- [95] Shampine, L. F., Watts, H. A., Davenport, S. M. Solving nonstiff ordinary differential equations - The state of the art. *SIAM Review*, 1976, volume 18, no. 3, pp. 376–410, ISSN 1095-7200.
- [96] Shanks, E. B. Solutions of differential equations by evaluations of functions. *Mathematics of Computation*, 1966, volume 20, no. 93, pp. 21–38, ISSN 1088-6842.
- [97] Söderlind, G. Automatic control and adaptive time-stepping. *Numerical Algorithms*, 2002, volume 31, no. 1-4, pp. 281–310, ISSN 1572-9265.
- [98] Soumitro, B. Dynamics of physical systems. <http://www.ee.iitkgp.ernet.in/~soumitro/>, 2004-02-18 [cit. 2010-01-16].
- [99] Stetter, H. J. Local estimation of the global discretization error. *SIAM Journal on Numerical Analysis*, 1971.
- [100] Vitásek, E. The numerical stability in solution of differential equations. *Lecture Notes in Mathematics*, 1969, volume 109, pp. 87–111, ISSN 0075-8434.
- [101] Vitásek, E. *Základy teorie numerických metod pro řešení diferenciálních rovnic*. Academia Praha, 1994, ISBN 80-200-0281-2.
- [102] Řezáč, D. *Stiff systems of differential equations and Modern Taylor series method*. FIT VUT Brno, 2004, Ph.D. thesis.
- [103] Watts, H. A. Starting step size for an ODE solver. *Computational & Applied Mathematics*, 1983, volume 9, no. 2, pp. 177–191, ISSN 1807-0302.
- [104] Widlund, O. B. A note on unconditionally stable linear multistep methods. *BIT Numerical Mathematics*, 1967, volume 1, no. 1, pp. 65–70, ISSN 1572-9125.

- [105] Zhang, W. The starting procedure in variable-stepsize variable-order PECE codes. *Computational & Applied Mathematics*, 1994, volume 53, no. 1, pp. 73–86, ISSN 1807-0302.
- [106] Zill, D. G. *A first course in differential equations: with modeling applications*. Brooks/Cole, 2009, ISBN 978-0-495-10824-5.

Appendix A

List of publications

- Butcher, J., Kunovský, J., Sehnalová, P. Predictor–corrector Obreshkov pairs. Accepted in *International Conference of Scientific Computation And Differential Equations*, Toronto, CA, FIELDS, 2011.
- Sehnalová, P., et al. Explicit and implicit Taylor series based computations. In *Proceedings of 8th International Conference of Numerical Analysis and Applied Mathematics*, Tripolis, GR, AIP, 2010, pp. 587-590, ISBN 978-0-7354-0831-9.
- Sehnalová, P., et al. Stability and convergence of the Modern Taylor series method. In *Proceedings of the 7th EUROSIM Congress on Modelling and Simulation*, Praha, CZ, VCVUT, 2010, pp. 6, ISBN 978-80-01-04589-3.
- Sehnalová, P., et al. Technical initial problems and automatic transformation. In *Proceedings of 2009 International Conference on Computational Intelligence, Modelling and Simulation*, Brno, CZ, IEEE CS, 2009, pp. 75-80, ISBN 978-0-7695-3795-5.
- Sehnalová, P., et al. Using differential equations in electrical circuits simulation. In *International Journal of Autonomic Computing*, volume 1, no. 2, 2009, London, GB, pp. 192-201, ISSN 1741-8569.
- Sehnalová, P., et al. Using differential equations in electrical circuits simulation. In *Proceedings of the 42nd Spring International Conference MOSIS '08*, Ostrava, CZ, MARQ, 2008, pp. 150-154, ISBN 978-80-86840-40-6.
- Sehnalová, P., et al. Multi-rate integration and Modern Taylor series method. In *Proceedings of UKSim 10th International Conference EUROSIM/UKSim2008*, Cambridge, GB, IEEE CS, 2008, pp. 386-391, ISBN 0-7695-3114-8.

Appendix B

TKSL/C code

The code for RLC circuit for TKSL/C is presented here.

```
omega=1e+3;
R=20;
L=2.5e-2;
C=5e-5;
u=sin(omega*t);
% numerical solution
uC'=1/C*i &0;
i' =1/L*uL &0;
uL =u-R*i-uC;
% analytical solution
uCanalyt=exp(-400*t)*(16/17*cos(800*t) +13/17*sin(800*t))
        -4/17*sin(omega*t) -16/17*cos(omega*t);
% error between solutions
err=uC-uCanalyt;
```

The program TKSL/C is available on <http://www.fit.vutbr.cz/~kunovsky/TKSL/download.html>. To run the computation copy the code above to the text file named input by the command in the terminal

```
cltksl -t 0.1 -s 1e-4 input > output
```

The output is in the format showed in table B.1 where each column represents a computed variable. Time t in the first column implicitly shows the used stepsize, a variable err describes the local error between analytical and numerical solutions. The error should be in the absolute value, unfortunately the function is not implemented in the program. A variable u is auxiliary and it represents the alternating source of the voltage. There are columns of numerical uC and analytical $uCanalyt$ solutions, the column with a variable uL is the voltage on the conduction coil and numbers in the column $\#$ represent the number of Taylor series terms used in each step.

Table B.1: Results for RLC circuit solved by TKSL/C

t	err	i	u	uC	$uCanalyt$	uL	#
0	0	0	0	0	0	0	0
1e-04	-1.3358644703e-20	1.9447859599e-04	9.9833416646e-02	1.3059108589e-04	1.3059108589e-04	9.5813253640e-02	7
2e-04	-1.0328719430e-19	7.5440946600e-04	1.9866933079e-01	1.0216381129e-03	1.0216381129e-03	1.8255950336e-01	7
3e-04	-3.3615028076e-19	1.6416342446e-03	2.9552020666e-01	3.3664991265e-03	3.3664991265e-03	2.5932102264e-01	7
4e-04	-7.6658988333e-19	2.8146553789e-03	3.8941834230e-01	7.7787653106e-03	7.7787653106e-03	3.2534646941e-01	7
5e-04	-1.4371107721e-18	4.2293067859e-03	4.7942553860e-01	1.4786251761e-02	1.4786251761e-02	3.8005315112e-01	7
6e-04	-2.3778859751e-18	5.8394300519e-03	5.6464247339e-01	2.4826336283e-02	2.4826336283e-02	4.2302753607e-01	7
7e-04	-3.6068178252e-18	7.5975492364e-03	6.4421768723e-01	3.8242650339e-02	3.8242650339e-02	4.5402405216e-01	6
8e-04	-5.1297964683e-18	9.4555375338e-03	7.1735609089e-01	5.5283112557e-02	5.5283112557e-02	4.7296222766e-01	6
9e-04	-6.9411609871e-18	1.1365269293e-02	7.8332690962e-01	7.6099281993e-02	7.6099281993e-02	4.7992224176e-01	6
⋮							
9.97e-02	1.2905973593e-16	-4.2688144771e-02	-7.3858222513e-01	-4.6072268119e-01	-4.6072268119e-01	5.7590335149e-01	6
9.98e-02	1.2977384614e-16	-4.0175105886e-02	-6.6758835430e-01	-5.4365505368e-01	-5.4365505368e-01	6.7956881710e-01	6
9.99e-02	1.2919129897e-16	-3.7260650623e-02	-5.8992416131e-01	-6.2115540458e-01	-6.2115540458e-01	7.7644425572e-01	6
1e-01	1.2731791506e-16	-3.3973899255e-02	-5.0636564110e-01	-6.9244937600e-01	-6.9244937600e-01	8.6556172001e-01	6

Appendix C

Starting method

The code of the starting procedure for Dahlquist problem is presented here. This version is compatible with Scilab version 5.0.3.

```
y=1;
y10=y;
f10=eval(f);
g10=eval(g);
Yinp=[y10 f10*h g10*h^2/2 h^3/6 h^4/24]';

while count < no.cycles
    Ypred = P * Yinp;

    y = Ypred(1);
    f1=eval(f);
    g1=eval(g);
    delta1 = h * f1 - Ypred(2,1);
    epsilon1 = 1/2 * h^2 * g1 - Ypred(3,1);

    Yout = Ypred + (delta1 * alfa) + (epsilon1 * betha);

    r=-1;
    h=-h;
    Yout = diag([1,r,r^2,r^3,r^4])*Yout;
    if (modulo(count,2)==0)
        Yinp = Yout;
    else
        Yinp = [y10 f10*h g10*h^2/2 Yout(4,1) Yout(5,1)]';
    end
    count=count+1;
end
Yinp = [y10 f10*h g10*h^2/2 Yout(4,1) Yout(5,1)]';
```

Appendix D

Results of circle test

Results of circle test (5.17) are computed by the new method for $h = \text{tmax}/n$ where $\text{tmax}=2\pi$ and steps $n = 160$. Results are displayed in table D. In the first column there are numbers of timesteps called x . In the second step there are computed values of y_1 which gives the expected known solution $y_1 = \cos(x)$. In the third column there are numbers of y_2 correspond to $y_1 = \sin(x)$. Results can be easily checked.

Results are plotted in picture D.1.

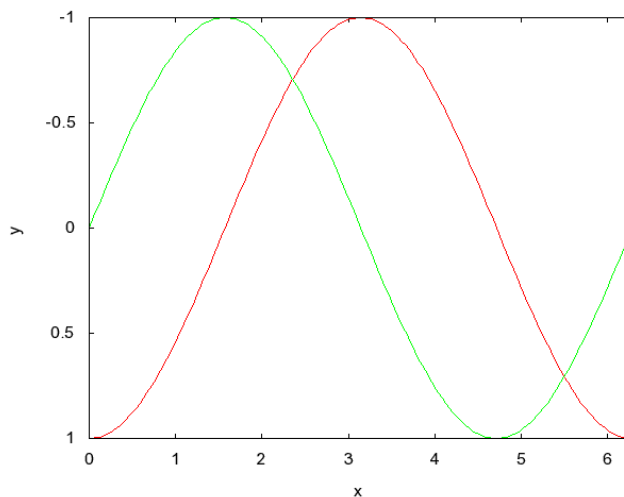


Figure D.1: Circle test results

The order of the new method and errors at the end of the algorithm for different fixed stepsizes are presented in table D.1. The chosen stepsizes are displayed in the first column of the table and the error at the end of the calculation is presented in the second row. The column called ratio represents the order check of the method. As we presented the order of our method is four ($p = 4$), thereof numbers of ratios ($\text{ratio} = 2^p = 2^4$) correspond to our claim.

x	$y_1 = \cos(x)$	$y_2 = -\sin(x)$
0	1	0
3.926991e-02	9.9922903625e-01	-3.9259815629e-02
7.853982e-02	9.9691733377e-01	-7.8459097415e-02
1.178097e-01	9.9306845707e-01	-1.1753740096e-01
1.570796e-01	9.8768834053e-01	-1.5643447037e-01
1.963495e-01	9.8078528001e-01	-1.9509032918e-01
⋮	⋮	⋮
3.063053e+00	-9.9691736340e-01	-7.8458957877e-02
3.102323e+00	-9.9922906079e-01	-3.9259675044e-02
3.141593e+00	-1.0000000192e+00	1.4338736063e-07
3.180863e+00	-9.9922905001e-01	3.9259961617e-02
3.220132e+00	-9.9691734187e-01	7.8459243844e-02
3.259402e+00	-9.9306845930e-01	1.1753754761e-01
⋮	⋮	⋮
6.086836e+00	9.8078537172e-01	1.9509005503e-01
6.126106e+00	9.8768842189e-01	1.5643419292e-01
6.165376e+00	9.9306852796e-01	1.1753712058e-01
6.204645e+00	9.9691739419e-01	7.8458814480e-02
6.243915e+00	9.9922908593e-01	3.9259530548e-02
6.283185e+00	1	-2.8875906233e-07

Table D.1: Order and errors for circle test

h	error	ratio
$0.2 \cdot 2^{-1}$	3.876048e-06	
$0.2 \cdot 2^{-2}$	2.320613e-07	16.70
$0.2 \cdot 2^{-3}$	1.416207e-08	16.39
$0.2 \cdot 2^{-4}$	8.742641e-10	16.20
$0.2 \cdot 2^{-5}$	5.430012e-11	16.10
$0.2 \cdot 2^{-6}$	3.384626e-12	16.04
$0.2 \cdot 2^{-7}$	2.219336e-13	