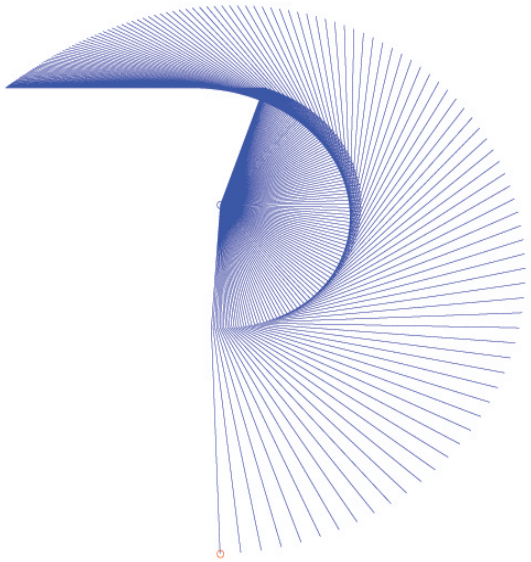
Numerical Solution of Ordinary Differential Equations

– analysis and applications



Allan P. Engsig-Karup & Per Grove Thomsen

APEK & PGT

September 4, 2018.

Preface

Engineers often handle problems that have a dynamic nature. Traditionally this kind of problems are modelled in the form of systems of differential equations where the mathematical formulation is given in terms of one or more equations involving a time-derivative.

In most numerical applications for scientific computing the solution by an analytic approach is very difficult or even impossible. For this reason we concentrate on numerical solutions.

This is a lecture note on the numerical treatment of initial value problems. It is concerned with the introduction of numerical solution methods and their applications in the art of solving problems arising in different fields of engineering. The subject areas are chosen to match the curriculum of course no. 02685 *Scientific Computing for Differential Equations*[1] taught at Technical University of Denmark (DTU), the part concerning ODE's.

These lecture notes are intended for students who have a basic knowledge of Numerical Analysis and Scientific Computing. The text is written mainly for engineering students who wish to know more than just the basics of numerical solution of initial value problems. It introduces classical theory of accuracy and stability, the basic ideas behind the concept of stiffness, adaptive stepsize control and discusses implementation details of the most widely used methods in the environment of Matlab. The derivation of methods can be supplemented by the use of symbolic manipulation software tools such as Maple or Mathematica.

[1] Before 2009, the course was titled *Numerical Analysis of Differential Equations*.

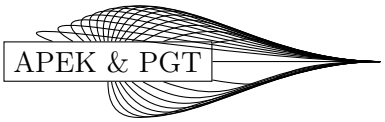
The emphasis is on the modelling and simulation of dynamical systems that appear in connection with engineering applications. The vast subject of solving partial differential equations is although very important not treated in any depth here.

Exercises are chosen from engineering applications and they are kept (mostly) on a level of complexity that graduate students can grasp.

We would also like to acknowledge the contributions to Section 9.1 on the "Dynamics of a rolling railway wheel set" by Hans True serving as an example of ODEs in applications.

The second author of these lecture notes would like to thank an old friend and inspirator, professor J.D. Lambert who taught him that mathematics need not be all that serious. In many cases you can simply enjoy the activity of exercising your knowledge on a new problem. An activity that was often enjoyed in the good company of good friends in the GODESS group.

Good fun.



September 4, 2018
Allan P. Engsig-Karup & Per Grove Thomsen

Contents

1 Introduction **1**

2 Dynamical systems **3**

2.1 The parachute jumper 3

2.2 The flat water kayak sprint 7

2.3 Simulation of ascent and descent for a spacerocket 8

3 Initial Value Problems **11**

3.1 Existence and uniqueness 11

3.2 Solution of Initial Value Problems 13

3.3 Discretisations 14

3.4 Solution by integration 15

3.5 Convergence 16

3.6 Local and global errors 18

4 Linear multistep methods **23**

4.1 Local truncation errors 24

4.2 Consistency 25

4.3 Order of accuracy 25

4.4 Zero-stability 27

4.5 Convergence 28

4.6 Adams Methods 30

4.7 Predictor-Corrector Methods 34

4.8 Stopping criterion for iterations 35

4.9 Milne's Device 35

4.10 Backward Differentiation Formula (BDF) methods 36
4.11 Implementation of Predictor-Corrector Methods 37

5 One-step methods **41**
5.1 Global errors for one-step methods 42
5.2 Runge-Kutta methods 43
5.3 The order of Runge-Kutta methods 47
5.4 The local error 49
5.5 Design of a RK-23 method 53

6 Absolute stability **61**
6.1 Stability for one-step methods 62
6.2 Stability for multistep methods 66
6.3 A-Stability 69
6.4 Stability Matrix 70

7 Control of Error and Convergence **73**
7.1 Asymptotic control 75
7.2 PI-control of step size 75
7.3 Measuring local errors 77
7.4 Implementation of one-step methods 79

8 Special problems for ODE-solvers **87**
8.1 Stiff systems 87
8.2 Semi-implicit Runge-Kutta methods 90
8.3 Systems with changes of state 97
8.4 The Poincaré section 106
8.5 Differential Algebraic Equations 108
8.6 General Linear Methods 113
8.7 Collocation 114

9 ODEs in applications **119**
9.1 Dynamics of a rolling Railway Wheel set 119
9.2 The Van der Pol oscillator 126
9.3 The golfstroke 130

A Prerequisites **139**

A.1 Taylor series 139

A.2 Backward differences 139

A.3 Interpolation 140

A.4 Numerical Integration 141

A.5 Nonlinear equations 141

A.6 Polynomials 143

List of Figures

2.1 Parachute jump from an airplane. 4
2.2 The flight path of the parachute jumper. 5
2.3 Simulation of a 500 m kayak sprint. 9

3.1 The direction field for two coupled ODEs (3.5) with various trajectory solution paths (black lines) and equilibrium points. 13
3.2 The interval $[a, b]$ divided into N subintervals. 15
3.3 Graphical construction of Euler's method. 16
3.4 Euler fails to find a circle - it becomes a spiral. 17

4.1 Convergence test for Milne's method demonstrating fourth order convergence as expected. 27
4.2 The layout of a Predictor-Corrector code. 38

5.1 Geometric construction of a two-stage Runge-Kutta method. 44
5.2 Structure of the **A** matrix for different classes of Runge-Kutta Methods. 46
5.3 The three-stage Runge-Kutta method with error estimate. 52
5.4 Layout of a steps for implementing a one-step method. 54

6.1 Characteristics of solutions to the test problem. 61
6.2 Test equation behaviour. 64
6.3 Stability regions for Runge-Kutta method of order two and three. . . 64
6.4 Stability region for Fehlberg Runge-Kutta methods of order 1-5. . . . 65
6.5 Stability regions for Adams-Bashforth methods of order 1, 2, 3 and 4. 68
6.6 Stability regions for Backward Euler of order 1 together with Adams-Moulton methods of order 2, 3 and 4. 68
6.7 A(α)-stability region. 70

6.8 Stability regions for BDF methods of order 1 to 6. 71

7.1 Phase plane solutions to the Van der Pol solution for a) $\mu = 1$ and b) $\mu = 100$ 74
7.2 Development of solution in time and step sizes for Van der Pol solution using asymptotic control for $\mu = 1$. 74
7.3 Steps versus tolerance test, log-log plot, $\mu = 200$ 77

8.1 The Robinson problem for different values of λ. 88
8.2 The Robinson problem, oscillations. 88
8.3 The Robinson problem, closeup. 89
8.4 The Robinson problem, stepsize history. 90
8.5 Order Star for SDIRK-2. 92
8.6 Discontinuity across curve of state change. 98
8.7 Integrating across a point of state change. 100
8.8 Solution and stepsize history with a discontinuity. 101
8.9 Solution and Continuous Extension. 103
8.10 Solution and determination of a transition point. 103
8.11 State transition diagram and matrix 104
8.12 The tank with heater and thermostat. 105
8.13 State diagram for the tank-heater. 105
8.14 Simulation of the tank-heater. 105
8.15 Lorenz solution. 106
8.16 Lorenz solution. 106
8.17 Poincare section of Duffings equation. 107

9.1 Sketch of contact plane systems for a wheel and a railway flange. . . 121
9.2 Sketch of wheel set and notations in the contact plane. 122
9.3 Bifurcation diagram for the dynamic railway problem. 125
9.4 Oscillator circuit. 126
9.5 Negative resistor diagram. 127
9.6 Van der Pol solution for $\mu = 10$. 128
9.7 Van der Pol solution for $\mu = 10$. 128
9.8 Van der Pol solution for variable μ. 129

9.9 Phase plane solution for variable μ. 129
9.10 Illustration of a double pendulum and notation for angles. 131
9.11 Golfer swings the driver. 132
9.12 Illustration of forces acting on a spinning ball. 133
9.13 Trajectories with constant initial speed and different launch angles. The lower launch angle leads to lower shot. Optimal length of the ball flight is achieved with initial angle, 37°. 135
9.14 Trajectories with constant launch angle and different initial speed. The larger initial speed achieves the ball fly further. 136
9.15 Trajectories with constant launch angle and initial speed as a function of the initial spin rate. Higher spinning rate achieves the ball fly further. 136
9.16 Trajectories with constant launch angle and initial speed as a function of the initial spin rate. Too high spin frequency decreases range. . . . 137
9.17 If initial spin frequency and launch angle are very large, ball rises to a steeper angle than its launch angle (solid line). Smaller launch angle makes the ball fly longer (dash-dot line). 138

Chapter 1

Introduction

Since Isaac Newton in his famous work on differential and integral calculus and the publication of "*De Methodis Serienum et Fluxionum*" in 1671 [34] and the contemporary work by Leibniz [48], scientists and engineers have used differential equations to describe systems where dynamical behavior in different forms is involved. Leibniz was working on dynamics and even started the development of computers in 1671, his computer was named 'Stepped Reckoner' and had concepts like the binary numbers.

Even though it has taken almost 300 years and the development of modern computers to make the theory useful to the general engineer, differential equations have been very useful in the process of understanding physics of the universe and of many everyday processes.

The importance of differential equations in engineering has been increased by the introduction of numerical methods for integration. The first attempt to find a discrete solution was done by Leonhard Euler in the 1730's and his book *Mechanica* [14] laid out some of the principles for later studies. Since then mechanical systems became more and more important and astronomers have become more and more interested in computing methods. It is not surprising that early numerical methods were developed by astronomers and by the physicists, in particular the works by Adams, Bashforth and Moulton [15, 33], Runge and Kutta [37] and later by Butcher [4].

These lecture notes are intended for engineering students. Classical theory of numerical methods for integration of ordinary differential equations is introduced followed by some discussion related to application of the methods for stiff systems, differential algebraic equations and system with discontinuous behavior.

The approach is a combination of theory, algorithms and experiments, where the implementational aspects play an important role. A discussion on implementations with step-size control is intended for the engineer who will understand the finer details of those techniques that are necessary to make reliable software for simulation of dynamical systems.

The final chapter is dedicated to some illustrative applications that have been solved using the techniques that were introduced. These applications belong to different areas of engineering to show that the theory and in particular the implementations are useful in a variety of practical situations. Special emphasis has been made to some of the more difficult areas of applications where the systems can really test the reliability and robustness of the software.

Chapter 2

Dynamical systems

In engineering and science, a vast range of problems are constantly being defined mathematically. Such mathematical formulations of problems are then models that seek to describe phenomena or processes that happens in the real world surrounding us. These models are solved by some means to improve the understanding of the behavior of dynamical systems. In engineering there is usually a practical need motivated by design considerations. In science, the need is rather motivated by research and development efforts, whether it is the mathematical or numerical analysis, or the physical processes themselves.

In this chapter some typical and relatively simple problems are considered together with solutions without going into any details with the applied solution techniques. Although relatively simple, the problems are sufficiently complex in order to make it difficult or impossible to determine a solution by analytical means. Thus, it is our hope that a basis for motivating the reader for studying the numerical solution techniques treated in the proceeding chapters of the notes.

2.1 The parachute jumper

We consider the simple problem given in the form of a second order Ordinary Differential Equation (ODE). This kind of system is typical for applications where the use of Newton's second law of motion is applied to obtain a mathematical model of the physical system. The type of mathematical problem is a second order ordinary differential equation.

We look at a simple application involving the travel through air by a parachute jumper who jumps from an airplane as illustrated in Figure 2.2. The jumper leaves the plane at time $t = 0$ at an initial altitude of $y = y_0$. We simplify the description by looking only at the vertical movement. Applying Newton's second law of motion and taking the gravity and air-resistance as the two forces acting on the jumper we get the equation for vertical acceleration. Here the notation $\ddot{y}$ is used for the second

derivative with respect to time.

$$M\ddot{y} = -Mg + \frac{1}{2}C_d A\rho\dot{y}^2. \tag{2.1}$$

Here we have introduced the formula for drag where A is the projected area on a horizontal plane, g is the gravitational acceleration, C_d is a dimensionless coefficient of drag that depends on shape, and ρ is the specific mass of air. The gravitational acceleration is assumed constant and set to 9.81 m^2/s. Our model has the form of a second order ordinary differential equation.

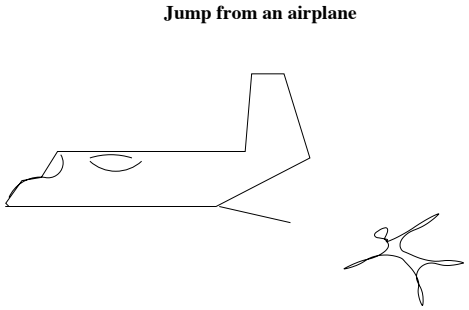


Figure 2.1: Parachute jump from an airplane.

If we introduce the velocity $v = \dot{y}$ and therefore $\dot{v} = \ddot{y}$ we can transform the second order equation into a system of two first order equations. After scaling by the mass M we have

$$\begin{aligned} \dot{y} &= v \\ \dot{v} &= -g + \tfrac{C_d A\rho}{2M} v^2 \end{aligned} \quad \cdot \tag{2.2}$$

The solution can be found analytically or numerically. We will try first to obtain insight in the behavior of the system by looking at the properties by simple analysis.

Let us look at the equation for the acceleration $\dot{v}$. It will have a value zero when the gravity equals the air resistance, this leads to the equation

$$v^2 = \frac{2Mg}{C_d A\rho}. \tag{2.3}$$

A positive value of velocity in this formula can be interpreted as the limiting velocity of the jumper when he no longer accelerate due to fore equilibrium between the gravitational and drag forces. This velocity is shown to depend on the model parameters introduced in the model. Let us consider four different situations (scenarios) for a jumper having a mass of $M = 75\ kg$, jumping from an altitude of 100 m.

Case 1: Falling like a ball. If he folds together we have $A = 0.5\ m^2$ and $C_d = 1$.

Case 2: Spreading arms and legs. $A = 1.0\ m^2$ and $C_d = 1.2$.

Case 3: Unfolding the parachute at time $t = 0$ results in $A = 20\ m^2$ and $C_d = 1.2$.

Case 4: Like case 1 for three seconds and then unfolding the parachute like in case 3.

Without more computing we can find that when the jumper hits the ground his vertical velocity in the four cases will be as shown in Table 2.1. To predict the

Case	1	2	3	4
v_{impact}	49.01	31.64	6.33	6.33

Table 2.1: Computed impact velocities (in m/s) for the different test cases just before a jumper hits the ground.

different flight paths or flight times it is possible to solve the system with proper initial conditions. In Figure 2.2 the solution has been determined numerically and the result is plotted as altitude against time. The velocity or falling speed is the slope of the flight path. Since the path becomes linear the velocity becomes constant and can be measured from the graph.

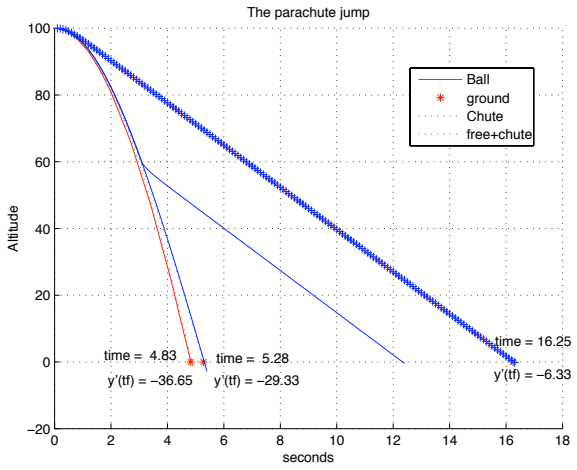


Figure 2.2: The flight path of the parachute jumper.

.

In Figure **??** it is illustrated that for a person falling from an airplane at 100 m without parachute there is not much hope for survival. The velocity at impact v_{impact} with the ground will be about 180 km/t, (36.65 m/s) while it is possible

to survive with a parachute even when it is released after 3 seconds since the final velocity is reached and in this case is only 22 km/t (6.33 m/s).

An exact analytical expression for the velocity $v(t)$ can be determined if we first rewrite the second equation in (2.2) into

$$\dot{v} = -\tfrac{b}{m}(v^2 - k^2), \quad k^2 = \tfrac{Mg}{b}, \quad b = \tfrac{C_d A \rho}{2}$$

and then integrate the equation such as

$$\int_{v(0)}^{v(t)} \frac{1}{(v^2 - k^2)} dv = -\tfrac{b}{M} \int_0^t dt$$

to find the following general expression for the velocity of the parachute jumper with time

$$v(t) = k\tfrac{1+c\cdot e^{-st}}{1-c\cdot e^{-st}}, \quad s = \tfrac{2kb}{M}, \quad c = \tfrac{v(0)-k}{v(0)+k}$$

If we wait long enough, the parachute jumper should reach the limiting velocity mentioned above

$$\lim_{t\to\infty} v(t) = k = \sqrt{\tfrac{Mg}{b}} = \sqrt{\tfrac{2Mg}{C_d A \rho}}$$

which indeed is equivalent to the expression found in (2.3).

To conclude this section, it can be noted that this example is a special case of a very general problem type, namely, problems described by a differential equation of order p. In the case where Newton's second law of motion is used we have $p = 2$. This is a very typical case for dynamical systems. The general problem where $y^{(p)}$ is the p'th derivative may be written

$$G(t, y, y', \ldots, y^{(p)}) = 0, \quad y^{(s)}(t_0) = y_0^{(s)} \text{ for } s = 0, 1, \ldots, p-1. \tag{2.4}$$

This initial value problem can be transformed into a system of first order ODE's by introducing the vector of variables

$$\mathbf{z} = (z_1, z_2, \ldots, z_p)^T \equiv \left(y, y', y'', \ldots, y^{(p-1)}\right)^T \tag{2.5}$$

We now make use of the definition for derivatives of $\boldsymbol{z}$ to form the system

$$\begin{aligned}
&z_1' = y' = z_2 \\
&z_2' = y'' = z_3 \\
&\vdots \qquad\qquad\qquad\qquad , \quad \text{for } q = 3, \ldots, p-1. \\
&z_q' = z_{q+1} \\
&G(t, \mathbf{z}) = 0
\end{aligned} \tag{2.6}$$

Very often the last equation can be put into an explicit form whereby the system simplifies to

$$\begin{aligned} z_1' &= y' = z_2 \\ z_2' &= y'' = z_3 \\ &\vdots \qquad\qquad\qquad\qquad , \quad \text{for } q = 3, \ldots, p-1. \\ z_q' &= z_{q+1} \\ z_{p-1}' &= z^{(p)} = f(t, z_1, z_2, \ldots, z_{p-1}) \end{aligned} \tag{2.7}$$

This equations of this system may be interpreted in many ways depending on the application. For example, in the case where $p = 2$, the variable $z_2 = y'$ can often be interpreted as the velocity. To an engineer this is a very obvious choice while to a mathematician it may just represent a convenient substitution when rewriting a second order differential equation into two first order equations. In any case, a goal of these lecture notes is to find methods for the efficient solution of systems of ODEs which can be represented in on of the general forms above. Thereby we also have methods for higher order systems of ODEs, since these can always be rewritten mathematically to a system of equations containing differential equations of lower order.

2.2 The flat water kayak sprint

We consider a simple model for the prediction of flat water kayaking sprint times for a kayak paddler who considers using a new proto-type kayak from a manufacturer. The example illustrates how experimental investigations can be used in conjunction with numerical modeling for developing simple predictions of flat water sprint timings for a paddler. The assumptions may be very crude and call for improvements, however, the following example illustrates the types of decisions we may encounter if we want to set up a simple dynamics model based on a ODE.

Consider the case of a person sprinting a distance of 500 m on flat water in a kayak using a paddle. The propulsive force of the kayak is delivered by the kayaker using a paddle at maximum effort over the sprint distance. The moving kayak is subject to a total drag force which is mainly due to wave resistance, friction between the wetted surface of the kayak and the water and could also be subject to exterior forces such as those produced by wind and currents. The first two contributions to the drag force is usually sought minimized by the kayak design criteria (including standards set in for example competition rules) resulting in a relatively long and narrow kayak hull.

Using Newton's second law, we can state this problem as

$$M\dot{v} = \sum_i F_i = F_{paddle} - F_{resistance} - F_{friction} \tag{2.8}$$

where $v = dx/dt$ is velocity, m is the total mass of the paddler, the kayak, clothes and equipment and we assume that the flat water sprint is in a place where there is no currents, e.g. a lake, and on a day with no wind.

In a sprint the person doing flat water kayak sprinting will usually try to produce a maximum force within his or her capabilities from start to finish. Here we assume that the force exerted through the paddle on the water for the forward propulsion is a constant force. In 500 m trial tests on an ergometer a former world champion was measured to produce a paddle force of 266 N at the moment of peak power [43]. We assume that our paddler is somehow a more average paddler that is able to produce a constant maximum work force over the 500 m that equals 90 % of this estimate, i.e. we set $F_{paddle} = 239.4\ N$.

We also assume that our paddler has morphological characteristics matching characteristics of some Olympic sprint paddlers at Sydney 2000, see [32]. Thus, our paddler is assumed to be a male having a body mass of 85 kg. The kayak has a mass of 12 kg matching international competition rules. Additional clothes and equipment is assumed to have a total mass of 2 kg.

Further, the friction force is assumed to be proportional to the product of the total weight and the gravitational acceleration factor as $F_{friction} = \mu g M$ where μ is some constant friction factor which for the case is assumed to be $\mu = 0.22$ (crude assumption). This friction factor can be shown to be proportional to the wetted surface of the kayak, the density of the water and the total mass, however, for the purpose of modeling, we chosen the before-mentioned simplistic expression for the friction force. Further, the wave resistance can be shown to be proportional to the square velocity and thus we express it as $F_{resistance} = kv^2$ with k is some coefficient which may also have some dependence on the velocity. Also, we assume that the manufacturer of the kayak has provided a resistance curve that relates the wave resistance coefficient to the velocity and have been obtained for the proto-type kayak, e.g. as seen in the rightmost sub-figure of Figure 2.3. With the assumptions made, we find using our numerical model that the kayaker will be able to do the 500 m sprint in approximately 106 s. The kayaker may now decide whether he would seriously consider using the new kayak. If this estimate is far from the kayaker's personal best times one could seek to improve the model and reduce crude assumptions.

2.3 Simulation of ascent and descent for a spacerocket

The first time a human was successfully sent to space using a space rocket to orbit Earth and return again was April 12 in 1961 with the Soviet Union cosmonaut

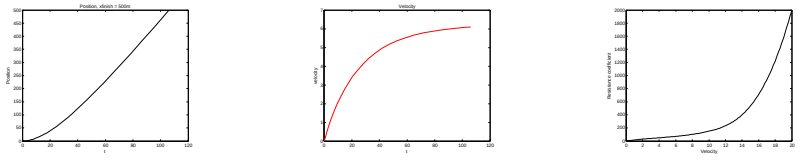


Figure 2.3: Simulation of a 500 m kayak sprint.

Yuri Gagarin. Since then the interest in the exploration of space has been steadily increasing as a result of the human curiosity combined with the emerging of new advanced technologies and potential (astronomical!) economical returns in successful technology advancement and understanding. However, the purpose of the following example is to describe a simple model for predicting trajectories of space rockets.

In the following, we are concerned with the simulation of ascent and descent for a small spacerocket from the Earth surface. The position of the space rocket can be determined by solving the equations for Newton's second law which can be stated as

$$M(t)\dot{v} = \sum_i F_i(t) \tag{2.9}$$

where the forces $F_i(t)$ will change according to the four phases: a) launch from a ramp, b) propulsion phase, c) ballistic phase and d) descent. During the first two phases the mass will change due to the burning of propellant. During the third phase, the space rocket will reach its maximum altitude before the gravity of Earth pulls the rocket back toward the Earth surface.

In the first phase (I), the net force can be expressed as

$$F_I = F_g + F_t + F_n \tag{2.10}$$

where F_g is the gravitational force, F_t is the thrust force and F_n is a normal force exerted through the contact with the ramp. The time duration of the phase is very short and fictive forces such as the coriolis and centrifugal forces due to the rotating Earth frame of reference can be neglected during this phase.

In the second phase (II), the net force can be expressed as

$$F_{II} = F_g + F_t + F_d \tag{2.11}$$

where F_d is the net drag force experienced by the rocket due to air resistance. In the case of a wind force, there will be an additional F_n normal force component proportional to the relative speed and angle of attack due to the wind.

In the third phase (III), the net force can be expressed as

$$F_{III} = F_g + F_t + F_d + F_c + F_{cf} \tag{2.12}$$

where F_c is the coriolis force and F_{cf} is the centrifugal force. The latter two forces needs to be taken into account in the case where the grounding site is to be predicted since the the rocket trajectory will be affected by the Earth rotation relative to the Earth frame of reference. If we are just interested in the maximum altitude these fictitious forces can be disregarded.

In the fourth and last phase (IV), the net force can be expressed as

$$F_{IV} = F_g + F_d + F_c + F_{cf} \tag{2.13}$$

where there is no longer any thrust due to the burning of rocket propellant. To reduce the speed of the rocket during the descent through the Earth atmosphere it is possible to equip the rocket with a parachute which can be unfolded at some specific altitude for the rocket to experience a soft landing rather than crashing onto the ground with high speed. The effect of such a parachute can be included in the model by modifying the drag force.

Chapter 3

Initial Value Problems

In many areas of application physical problems are modeled by the solutions to a system of Ordinary Differential Equations which we can refer to as ODEs. We may consider a variety of applications ranging from chemical reactions over electrical circuits to multibody problems.

Extending the problem area to Partial Differential Equations (PDEs) covering non-stationary problems in fluid flows or heat conduction after discretization in space we may end up solving large systems of ODEs. These examples are very important to many areas of engineering making it important to study methods in detail for the numerical solution of such problems.

In recent years the study of Differential Algebraic Equations (DAEs) has grown in importance and the introduction to these types of systems of equations is post-phoned to Section 8.5.

3.1 Existence and uniqueness

Let us consider a model of a dynamical system given in the form of a single first order ordinary differential equation

$$y' = f(t, y). \tag{3.1}$$

An equation of this type will in general have infinitely many solutions. Take a point P in the (t, y)-plane, then the function $f(t, y)$ maps this point into the space of the derivative through P. We want to find a curve that has a derivative through P that is equal to the value $f(t, y)$. This may exist for any point P in the (t, y)-space. Since P is arbitrarily chosen there will be a solution through all points where $f(t, y)$ is defined.

If we want the solution to pass through a particular point P_0 denoted as the initial point, then a very modest condition on the function $f(t, y)$ will ensure that such a solution exists and is unique. This is the so-called *Lipschitz condition.*

Theorem 3.1.1. Let the function $f(t, y)$ be defined and continuous for all points (t, y) in the strip D, $a \leq t \leq b$, $-\infty < y < \infty$, where a and b are finite. If f satisfies a *Lipschitz condition*, ie. if there exists a constant L such that

$$|f(t, y) - f(t, \widetilde{y})| \leq L|y - \widetilde{y}| \quad \text{for all} \quad t \in [a, b] \text{ and all } y, \widetilde{y} \ ,$$

then for any initial value α there exists a unique solution in the domain D of the initial value problem

$$y' = f(t, y), \qquad y(a) = \alpha \ .$$

The constant L is called the *Lipschitz constant*.

This condition is weaker than differentiability of the function $f(t, y)$ with respect to y and gives an upper bound for changes in $f(t, y)$ when y varies and t is assumed fixed.

Let us consider the important special case where the function $f(t, y)$ is continuously differentiable with respect to y everywhere in our domain of solution, then it will satisfy a Lipschitz condition since from the mean value theorem

$$f(t, y) - f(t, y^*) = \frac{\partial f(t, \tilde{y})}{\partial y}(y - y^*), \tag{3.2}$$

where $\tilde{y}$ is a point in the interior of the interval with end points y and y^*. Clearly the Lipschitz condition in Theorem 3.1.1 is satisfied if

$$L = \sup_{(t,y) \in D} \left| \frac{\partial f(t, y)}{\partial y} \right|. \tag{3.3}$$

The above theorem can be generalized to systems of ODEs and this leads to quite similar statements where absolute values are exchanged with vector norms.

Unless otherwise stated, it is assumed throughout these lecture notes that a Lipschitz condition is always satisfied for the initial value problems considered. One exception to this rule is in the discussion of problems with discontinuities where special properties of the right hand side function will be discussed.

The satisfaction of the Lipschitz condition implies that a solution always exist and that it is unique. This is essential for numerical computations to make sense. If there is no solution we cannot find one. If the solution to the system of differential equations is not unique we do not know which one we have found and whether we have found all of the solutions.

3.2 Solution of Initial Value Problems

For systems of n equations, where we may have a set of coupled equations, we can use the same notation as in (3.1)

$$\mathbf{y}' = \mathbf{f}(t, \mathbf{y}), \tag{3.4}$$

where we have introduced a solution vector $\mathbf{y}$ and a vector of right hand side functions

$$\mathbf{y} = (y_1, y_2, ..., y_n)^T, \quad \mathbf{f}(t, \mathbf{y}) = (f_1, f_2, ..., f_n)^T.$$

With this notation we can write general systems in a compact form.

We will therefore assume that an initial condition is given and that we can compute values of the function $f(t, y)$ in every point of the solution space. In order to get some visual information of the solution we can use a mapping of the field of vectors in our domain. This is called the *direction field* since it indicates the directions of the tangents to solution curves. The length of the vectors in the direction field represent the magnitude of the tangents and are often scaled. Thus, large vectors indicates fast variation in the solution in the direction shown. Figure 3.1 illustrates a direction

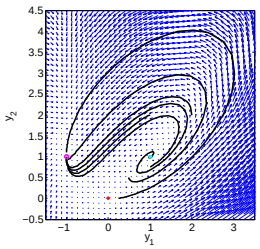


Figure 3.1: The direction field for two coupled ODEs (3.5) with various trajectory solution paths (black lines) and equilibrium points.

field[1] for the system of two simultaneous ODEs

$$\begin{aligned} y_1' &= y_1 - y_2 + y_1^2 - y_1 y_2, \\ y_2' &= y_1^2 - y_2. \end{aligned} \tag{3.5}$$

Also shown in Figure 3.1 are a number of solution curves which have different initial points. Some of the initial points are very close, but the solution curves are behaving very differently. The point $(y_1, y_2) = (1, 1)$ is an unstable equilibrium point where solutions will run away while the point $(y_1, y_2) = (-1, 1)$ is a stable equilibrium point where solutions are attracted. The point $(y_1, y_2) = (0, 0)$ is also an equilibrium point. Is it stable? A clue to the answer of this question can be found by inspecting the direction field in the immediate neighborhood of the point in question.

3.3 Discretisations

Common to all the methods that will be presented and discussed is that the numerical solution will appear basically as a table of discrete solution points. Each of these discrete solution points will represent an approximation to the exact solution to the initial value problem at a single point. Our goal is to find appropriate algorithms for calculating the approximate solution points and provide some kind of information about the quality of the solution. The quality of the solution is often assessed in the form of estimates of the local or global errors.

Consider a discrete set of solution points, eventually extended by a continuous extension or interpolant through the discrete set of points. We define the discrete point set as shown in Figure 3.2

$$\{t_n : n = 0, 1, 2, \ldots, N\}, \tag{3.6}$$

where $t_{n-1} < t_n$ and the stepsize $h_n = t_n - t_{n-1} > 0$. Thus, the point distribution can be both uniform and nonuniform over the interval. The corresponding solution points are defined as the set

$$y(t_n), \quad n = 0, 1, 2, \ldots, N. \tag{3.7}$$

For the function values defining the derivatives at the solution points we use the notation

$$y'(t_n) = f(t_n, y(t_n)), \quad n = 0, 1, 2, \ldots, N. \tag{3.8}$$

To represent computed values we introduce a compact indexed notation where y_n is the computed value which can be compared to the exact solution $y(t_n)$. In some cases the computed value y_n can be shown or assumed to be equivalent to the exact solution $y(t_n)$. This is for example the case in local error analysis.

[1] A matlab package for doing plots like this can be found at `http://math.rice.edu/~dfield/`.

Figure 3.2: The interval $[a, b]$ divided into N subintervals.

3.4 Solution by integration

If we do an integration of the initial value problem (3.1) over the interval $[t, t + h]$

$$\int_t^{t+h} y'(\tau)d\tau = \int_t^{t+h} f(\tau, y)d\tau \tag{3.9}$$

we can find an integration formula for the exact solution over one step of our discrete set of points of the form

$$y(t + h) = y(t) + \int_t^{t+h} f(\tau, y)d\tau. \tag{3.10}$$

This formula is however not easy to use for practical computations since it assumes we already know the solution $y(t)$ to complete the integral on the right. However, this is in general not the case and therefore we need other ways of defining computational methods.

To obtain this sort of computational scheme we will apply the above integration formula in different ways in later sections. A first attempt is obtained by using simple rectangular integration where the integral is approximated by the function value at the lower integration limit $f(t, y(t))$ multiplied with the length of the interval h. By this approximation of the integral in (3.10) we find

$$y(t + h) \cong y(t) + hf(t, y(t)). \tag{3.11}$$

From this approximation we define the method called *Euler's* method.

Definition 3.4.2. Euler's method:

$$\begin{aligned} y_0 &= \alpha, \\ y_{n+1} &= y_n + hf(t_n, y_n), \quad n = 0, 1, \ldots, N - 1. \end{aligned} \tag{3.12}$$

Using this method we can compute the sequence of approximate solutions step by step. The geometry behind Euler's method is that a step corresponds to moving along the tangent to the local solution through the point (t_n, y_n) as shown in Figure 3.3 for the ODE system for a harmonic oscillator (or circle if we examine solutions

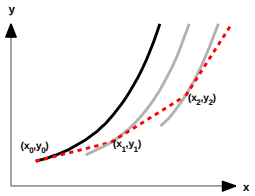


Figure 3.3: Graphical construction of Euler's method.

in the phase plane)

$$\begin{aligned} y_1' &= y_2, \\ y_2' &= -y_1 \end{aligned} \tag{3.13}$$

with some specified initial condition (y_1, y_2). In Figure 3.4 it is shown how the Euler steps progress from the same starting point for two different choices of step sizes h. Clearly, the Euler method for this problem fails to exactly represent the orbital motions of the solution in the phase plane. The reason for the failure can be understood by the simple geometric argument mentioned above and is due to that the method does not take into account the curvature of the solution.

3.5 Convergence

We will consider methods for finding approximate solutions at all N grid points in our discretization x_n, $n = 1, 2, \ldots, N$. A basic property of all our methods to be acceptable is, that the sequence of solution values $\{y_n\}$ generated by the method converges to the exact solution $y(t)$ as the stepsize goes to zero. Let us assume the special case of constant stepsize $h \to 0$ and consider the following experiment

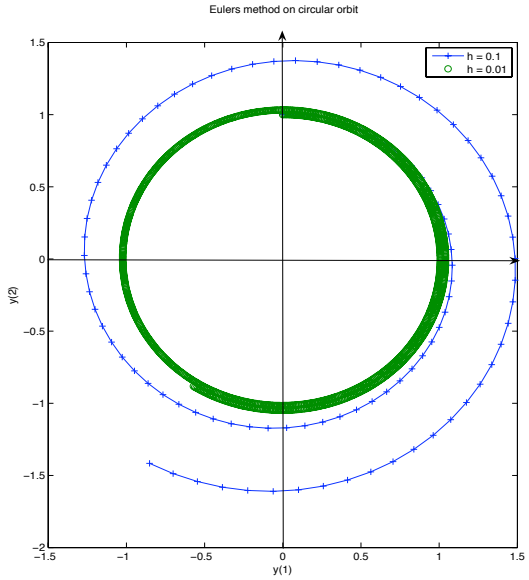


Figure 3.4: Euler fails to find a circle - it becomes a spiral.

1: Consider a fixed point t^* in our interval of integration, let h_0 be such that $t^* = a + h_0$.
2: Construct a sequence of experiments by successively halving the stepsize $h_k = h_0/2^k$ for $k = 0, 1, ...$
3: Consider the sequence of computed solution values y_n obtained by using the stepsize h_k to compute the approximation to $y(t^*)$ using n steps starting from $y(a)$.
4: Then consider the limiting process $h_k \to 0$ which implies $k \to \infty$.

For this process to converge we require that the method is convergent.

Definition 3.5.3. Convergence. A method is *convergent* if for all initial value problems subject to the hypothesis of Theorem 3.1.1 we have that

$$\lim_{h \to 0, nh = t-a} y_n = y(t) \tag{3.14}$$

holds for all $t \in [a, a + nh]$ and for all solutions $\{y_n\}$ generated by our process. This holds if any starting values needed also converges toward the exact solution.

For practical purposes it is important to establish that a method is convergent as this give confidence that we can expect a numerical method to not only produce a

solution, but also converge toward the unique solution in the asymptotic limit $h \to 0$. For practical reasons it may also be of interest to determine the *rate of convergence*, since this is a key factor in evaluating the efficiency of a numerical method.

Definition 3.5.4. Rate of convergence. The *rate of convergence* for a convergent method can be determined from

$$|y_n - y(t)| < Ch^p, \quad \text{for } h \to 0 \tag{3.15}$$

where C is some constant independent on the constant step size h, $y_n \approx y(t)$ after n steps for any $t \in [a, a + nh]$ where $t - a = nh$ and for all solutions $\{y_n\}$ generated in the process. The rate of convergence is determined by the order of accuracy p.

The rate of convergence is often stated in a form using "Big-O" notation as $\mathcal{O}(h^p)$. The faster the rate of convergence, the larger p is. To compare and choose between two convergent methods with the same rate of convergence, it can be of useful to determine the actual constant C in front of the term. The rate of convergence is not the only property of numerical methods that we need to pay attention to. Other requirements are for example memory use and stability issues.

3.6 Local and global errors

From the examples in the previous section we see that errors that are introduced in one step will influence the next step and accumulate over the full interval of integration. In order to see how the accumulation of local errors influence the convergence of numerical methods we start by examining Euler's method. We can determine an expression for the local error when we assume that $y_n = y(t_n)$ and ignore rounding errors. The local truncation error e_n for one step of the Euler method is then found to be

$$e_n = y_{n+1} - y(t_n + h) = -\frac{h^2}{2}y''(t_n) + \mathcal{O}(h^3) \tag{3.16}$$

where it is assumed that the true solution $y(t_n+h)$ can be expanded in a Taylor series and the computed solution y_{n+1} is determined from the recursion formula (3.12) for Euler's method.

If we define the global error e_n after n steps from the initial point $y_0 = y(t_0)$ as

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

and

$$e_{n+1} = y_{n+1} - y(t_{n+1}) = y_n + hf(t_n, y_n) - (y(t_n) + hf(t_n, y(t_n)) + \frac{h^2}{2}y''(\xi)$$

directly we obtain

$$e_{n+1} = e_n + h(f(t_n, y_n) - f(t_n, y(t_n))) - \frac{h^2}{2} y''(\xi)$$

where the Mean Value Theorem has been used to replace the high-order terms with a leading order term defined in terms of $\xi \in [t_n, t_n + h]$. Further taking absolute values and making use of the triangular inequality, we find

$$\begin{aligned}
|e_{n+1}| &= \mid e_n + h(f(t_n, y_n) - f(t_n, y(t_n))) - \frac{h^2}{2} y''(\xi) \mid \\
|e_{n+1}| &\leq |e_n| + h \mid (f(t_n, y_n) - f(t_n, y(t_n))) \mid + \frac{h^2}{2} \mid y''(\xi) \mid \\
|e_{n+1}| &\leq (1 + hL)|e_n| + \frac{h^2}{2} M
\end{aligned}$$

Here L is the Lipschitz constant and M is an upper bound on $|y''(t)|$. We also assume that the higher order derivatives will be bounded over the solution interval so that we can ignore the influence of the high-order terms.

We will study the solution of this difference equality and seek a bound for the global error. In a slightly reduced form we obtain, after introducing $A = (1 + hL)$ and $B = \frac{h^2}{2} M$.

$$|e_1| \leq A|e_0| + B$$

and after n applications of the process we get

$$|e_n| \leq A^n |e_0| + B \sum_{k=0}^{n-1} A^k \tag{3.17}$$

This expression shows that the initial error, that may result from the starting process of obtaining the initial values and from representing initial values in our computer, is present throughout the solution. We cannot disregard this effect on the global error unless we assume that this error can be set to zero.

We now make the assumption that $e_0 = 0$ this means that we represent the initial value(s) exactly.

$$|e_n| \leq \frac{A^n - 1}{A - 1} B \tag{3.18}$$

This can be even more reduced since

$$A^n = (1 + hL)^n \leq e^{nLh} = e^{L(t_n - t_0)} \tag{3.19}$$

From these expressions we find the upper bound for the global error to be of the form

$$|e_n| \leq \frac{hM}{2L} (e^{L(t_n - t_0)} - 1) = \mathcal{O}(h). \tag{3.20}$$

This shows that in the limit $h \to 0$ our global error $e_n \to 0$ since the other parts are bounded when we look at a fixed point t_n. This is confirming that Euler's method is convergent. More generally for the type of processes we will consider, we refer to the following Theorem by Dahlquist [8].

Theorem 3.6.5. Global error bound. Suppose that in a step-by-step solution, the magnitudes of the local errors at the points $t_n, n = 1, 2, \ldots$ are smaller than ϵ. Then the global error will satisfy the inequalities

$$|e_n| \leq \epsilon \frac{e^{L(t_n - t_0)} - 1}{e^{Lh} - 1} \tag{3.21}$$

It is common to distinguish between the local and global errors. When local error are discussed we always refer to the error we commit in a single step. Thus, any errors introduced in the past are neglected and the exact solution is assumed at the initial point. Furthermore, for the local errors it is useful to distinguish between the *Local Truncation Error* (LTE) and the *one-step error*. The LTE is defined from the error expression that results when we substract what we want to approximate with the introduced approximation arising when the function derivatives in the differential equation are replaced with difference approximations. The one-step error is defined from the recursive formula for advancing the solution to the next discrete point from past points. The global errors are determined from the accumulation of truncation errors, representation errors and modeling errors.

To clarify the difference between local and global truncation errors, we consider the ODE

$$y' = y \tag{3.22}$$

The local truncation error for this ODE can be determined from a discretization of the form

$$\frac{y_{n+1} - y_n}{h} + \mathcal{O}(h) = y_n$$

or

$$\frac{y_{n+1} - y_n}{h} - y_n = \mathcal{O}(h) \tag{3.23}$$

where the derivative y' has been replaced with a first order finite difference approximation (this is equivalent to (3.11)) and this lead to a first-order LTE. The one-step error can be determined by rewriting the formula as

$$y_{n+1} = (1 + h)y_n + \mathcal{O}(h^2) \tag{3.24}$$

and shows that the one-step error is second-order accurate. Formally, the one-step error is one order more accurate than the LTE and global error.

Exercises

1. Write the following ODE

$$\frac{d^2y}{dx^2} + 2sin(y) = 0$$

   as a system of first-order ODEs.

2. Explain the difference between the local truncation error and the one-step error for a numerical method.

3. Derive the expression for the local one step error of Euler's method given in (3.16).

4. Solve the ODE system (3.13) using simple geometric considerations on a piece of paper by visualizing the first few steps of Euler's method.

5. Use the Trapezoidal rule to replace the integral of (3.10) to determine a numerical method for solving ODEs. Use this method to solve the ODE system (3.13) with the initial conditions $(y_1, y_2) = (0, 1)$.

Chapter 4

Linear multistep methods

Historically, what is referred to as linear multistep methods have been given a lot of attention mainly because of their good properties with respect to obtaining high-accuracy solutions. As the name indicates the methods make use of information from a multitude of steps for the determination of the next step.

The general form is given by the formula for a linear k-step method

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h \sum_{j=0}^{k} \beta_j f(t_{n+j}, y_{n+j}), \tag{4.1}$$

where h is the step size, α_j and β_j are coefficients for respectively the number of forward steps and the function value at the forward steps, k is the number of steps included in the method and thus $k+1$ solution values are included in the formula. If the coefficient $\beta_k = 0$ the method is called *explicit* since the value of y_{n+k} can be computed explicitly from the formula. The method is *implicit* if $\beta_k \neq 0$ and then it will be necessary to calculate the function value in the point that has to be approximated using some iterative procedure. For a k-step formula we need to assume that $|\alpha_0| + |\beta_0| \neq 0$. It is standard to normalize the coefficients of the formula such that $\alpha_k = 1$. Finally, it is noted that a linear multistep method with only one step, i.e. $k = 1$, also belongs to the class of one-step methods which will be treated in Chapter 5.

An example of an implicit linear multistep method with two steps is

$$y_{n+2} = y_n + \tfrac{h}{3}(f_n + 4f_{n+1} + f_{n+2}), \tag{4.2}$$

which is called the *Milne method* [51] and it has been expressed using the compact notation for the function value $f_{n+j} = f(t_{n+j}, y_{n+j})$ taken at time t_{n+j}.

4.1 Local truncation errors

The Local Truncation Error (LTE) for a linear multistep method applied to solve an ODE is defined as

$$\tau(t_{n+r}) = \frac{1}{h}\left(\sum_{j=0}^{k}\alpha_j y(t_{n+j}) - h\sum_{j=0}^{k}\beta_j f(t_{n+j}, y(t_{n+j}))\right), \tag{4.3}$$

which by making use of the ODE by substitution of $f(t_{n+j}, y(t_{n+j})) = y'(t_{n+j})$ can be expressed as

$$\tau(t_{n+r}) = \frac{1}{h}\left(\sum_{j=0}^{k}\alpha_j y(t_{n+j}) - h\sum_{j=0}^{k}\beta_j y'(t_{n+j})\right).$$

If we assume that $y(t)$ is a smooth function it is possible to employ the following Taylor series expansions of the function and it's first derivative in a local analysis

$$\begin{aligned} y(t_{n+j}) &= y(t_n) + (jh)y'(t_n) + \tfrac{1}{2}(jh)^2 y''(t_n) + \cdots \\ y'(t_{n+j}) &= y'(t_n) + (jh)y''(t_n) + \tfrac{1}{2}(jh)^2 y'''(t_n) + \cdots \end{aligned}$$

By inserting these Taylor series expansions into the expression for the LTE and collecting terms of same power in h, the LTE for a linear multistep method can be expressed as

$$\begin{aligned} \tau(t_{n+r}) = \frac{1}{h}&\left(\sum_{j=0}^{k}\alpha_j\right) y(t_n) + \left(\sum_{j=0}^{k}(j\alpha_j - \beta_j)\right) y'(t_n) \\ &+ h\left(\sum_{j=0}^{k}\left(\tfrac{1}{2}j^2\alpha_j - j\beta_j\right)\right) y''(t_n) + \cdots \\ &+ h^{q-1}\left(\sum_{j=0}^{k}\left(\frac{1}{q!}j^q\alpha_j - \frac{1}{(q-1)!}j^{q-1}\beta_j\right)\right) y^{(q)}(t_n) + \cdots \end{aligned} \tag{4.4}$$

having collected terms with same power dependence on the step size. If we introduce the difference operator $\mathcal{L}$ for the LTE of a linear k-step method

$$\mathcal{L}(y(t); h) = \frac{1}{h}\left(\sum_{j=0}^{k}\left(\alpha_j y(t+jh) - h\beta_j y'(t+jh)\right)\right) \tag{4.5}$$

or written compactly as

$$\mathcal{L}(y(t); h) = \frac{1}{h}C_0 y(t) + C_1 y'(t) + C_2 h y''(t) + \cdots + C_q h^{q-1} y^{(q)} + \cdots \tag{4.6}$$

By comparing with the terms in (4.4), we find that the constants C_j, $j = 0, 1, ..., k$ are defined as

$$C_0 = \sum_{j=0}^{k} \alpha_j, \quad C_q = \sum_{j=0}^{k} \left(\frac{j^q}{q!} \alpha_j - \frac{j^{q-1}}{(q-1)!} \beta_j \right), \quad q \geq 1, \tag{4.7}$$

where the factorial function is defined as $n! \equiv n \cdot (n-1) \cdots 2 \cdot 1$ and $0! \equiv 1$.

Thus, the LTE can then be expressed as

$$\tau(t_{n+r}) = \mathcal{L}(y(t_n); h). \tag{4.8}$$

4.2 Consistency

A linear multistep method is called *consistent* with the initial value problem on which it is applied if it is possible to make the local errors of the computed solution vanish when the step size h is decreased toward the asymptotic limit

$$\tau(t_{n+r}) \to 0, \quad \text{for } h \to 0. \tag{4.9}$$

This condition implies that a consistent linear multistep method must satisfy the conditions

$$C_0 = C_1 = 0. \tag{4.10}$$

with the two conditions defined in (4.7) and referred to as the *consistency conditions*.

The consistency property of a linear multistep method ensures that there is a connection between the initial value problem and the numerical solution in the asymptotic limit $h \to 0$. This implies that for decreasing step sizes h the solution should approach the exact solution to the mathematical problem. It can be shown that this is a necessary condition for convergence of the numerical solution toward an exact solution to the initial value problem.

4.3 Order of accuracy

The formal order of accuracy of a linear multistep method is determined using a local analysis by comparing terms with a Taylor series expansion of the local truncation error of the difference equation.

Theorem 4.3.6. A linear multistep method is order q if the $q + 1$ *order conditions* are fulfilled

$$C_0 = C_1 = \cdots = C_q = 0, \quad \text{and } C_{q+1} \neq 0 \tag{4.11}$$

where the constants are defined in (4.7).

Thus, the order of accuracy is determined by the leading order terms in a Taylor series expansion of the local error. The first non-zero constant is called *the error constant*. Remark, the order conditions depend only on the coefficients of the linear multistep method and not on the particular differential equations being solved.

To serve as an example, we can determine the order of accuracy of the *Milne method* given in (4.2). We recognize the method to be a two-step method with coefficients

$$\begin{aligned} \alpha_0 = -1, \quad \alpha_1 = 0, \quad \alpha_2 = 1, \\ \beta_0 = \frac{1}{3}, \quad \beta_1 = \frac{4}{3}, \quad \beta_2 = \frac{1}{3}. \end{aligned}$$

The order of accuracy for the method can be determined from the explicit evaluation of the order conditions (4.7) where we find

$$C_0 = C_1 = C_2 = C_3 = C_4 = 0, \quad C_5 = -\frac{1}{90}.$$

Thus, we can conclude that the Milne method is a consistent fourth order accurate method with a local truncation error of $\mathcal{O}(h^4)$ and principal local error constant $-\frac{1}{90}$.

In practical implementations it is recommended to demonstrate the expected and predicted order of accuracy by estimating the rate of convergence in a *convergence test*. The goal of a convergence test is to validate an implementation by checking that the error behaves as expected when the step size approaches the asymptotic limit $h \to 0$. Such a test can be devised using an initial value problem having a known analytical solution, e.g. the test IVP. Test results are produced by and then compare the computed solution to the known exact solution to check the error behavior towards the asymptotic limit $h \to 0$, where we expect the error to behave as $e \cong Ch^p$ for a convergent numerical method of order p. The results of a convergence test for Milne's method (4.2) based on solving the IVP $y' = y$ with $y(0) = 1$ up to $t = 1$ is shown in Figure 4.1.

The order conditions (4.11) can be used in designing a linear multistep method of a given order by the Method of Undetermined Coefficients by solving the set of coupled order conditions. For an k-step method we would have a total of $2(k+1)$ free unknown parameters α_j, β_j, $j = 0, 1, ..., k$ to determine. By setting $\alpha_k = 1$ we are left with $2r+1$ free unknown parameters if the method is implicit. If the method is explicit we also set $\beta_k = 0$ and we are left with $2r$ free unknown parameters. These free parameters can then be used to satisfy as many of the order conditions (4.7) possible. The maximum attainable order of accuracy for a linear multistep method is stated in Theorem 4.3.7.

Theorem 4.3.7. Any k-step method defined by (4.1) has maximum attainable order $2k$ if the method is implicit and $2k-1$ if the method is explicit.

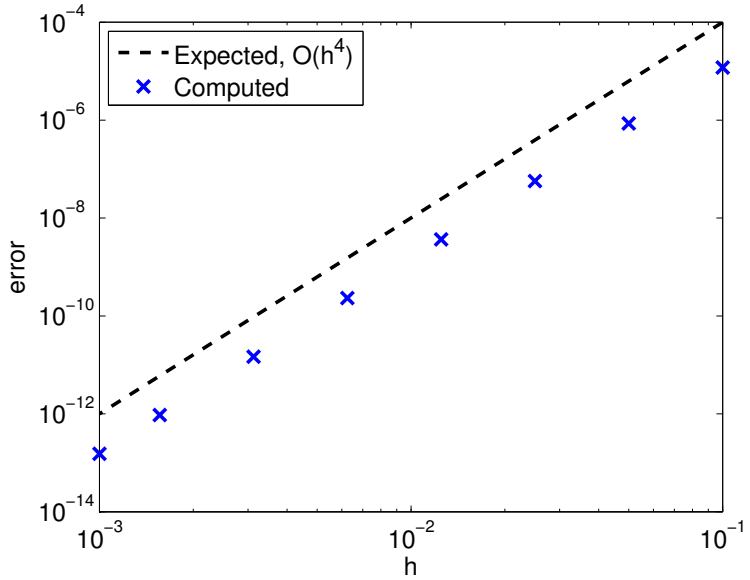


Figure 4.1: Convergence test for Milne's method demonstrating fourth order convergence as expected.

4.4 Zero-stability

When we apply a linear multistep method to solve a general initial value problem a sequence of approximate discrete solutions is determined. We are interested in approximate solution which is stable and bounded for any number of steps N. Thus, we need to introduce the concept of *zero-stability*. When we refer to zero-stability for a resulting difference system we will be concerned with stability in the asymptotic limit $h \to 0$. Thus, 'zero' refers to the limit $h \to 0$, which is equivalent to consider the stability of initial value problems of the form

$$y' = 0, \quad y(0) = \alpha \tag{4.12}$$

where α is some initial value and the right hand side function of the general IVP is defined to be zero. This initial value problem is called the *trivial test problem*. Zero-stability can be shown to be an important necessary condition for a linear multistep method to be convergent.

In the limit $h \to 0$ the linear multistep method tends to the linear constant coefficient difference system

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = 0. \tag{4.13}$$

Thus, it is clear that zero-stability of a linear multistep method is controlled by the location of the roots of the first characteristic polynomial $\rho(\zeta)$ for this difference

system

$$\rho(\zeta) = \sum_{j=0}^{k} \alpha_j \zeta^j \tag{4.14}$$

Definition 4.4.8. Root condition: A linear multistep method satisfies the *root condition* if all the roots of the first characteristic polynomial $\rho(\zeta)$ have modulus less than or equal to unity, and those of modulus unity are simple.

To satisfy the root condition the k roots $\zeta_i \in \mathbb{C}$, $i = 1, 2, ..., k$ of the first characteristic polynomial satisfies

$$\begin{aligned} &|\zeta_i| \leq 1 \text{ for } i = 1, 2, ..., k. \\ &\text{If } \zeta_i \text{ is a repeated root, then } |\zeta_i| < 1. \end{aligned} \tag{4.15}$$

which implies that all roots should be located inside or on the unit circle in the complex plane, and roots of unit modulus should be simple.

The stability of a linear multistep method is connected to the roots of the first characteristic polynomial $\rho(\zeta)$ and we call the k-step method *strongly stable* if the method satisfies the root condition with $\zeta = 1$ the only root of magnitude one. We call the method *weakly stable* if the root condition is satisfied with more than one distinct root of magnitude one. If a k-step method does not satisfy the root condition it is denoted *unstable*.

Theorem 4.4.9. Zero-stability: The necessary and sufficient condition for a k-step linear multistep method to be *zero-stable* is that it satisfies the root condition. If a linear multistep method fails to satisfy the root condition it is *zero-unstable*.

4.5 Convergence

One important question to ask is whether a numerical method will converge toward an exact solution of a given initial value problem in the asymptotic limit when the step size approaches zero, i.e. when $h \to 0$. If this is the case, then it implies that the convergent numerical method makes it possible to not only bound the global error at a fixed time by appropriate choices of the step size h, but also makes it possible to achieve a solution which has the desired accuracy.

Definition 4.5.10. Convergence: A numerical method is said to be convergent when applied to solve the initial value problem if the computed solution approaches the exact solution at a fixed time t in the limit $h \to 0$

$$\lim_{\substack{h\to 0 \\ t=t_0+Nh}} y_n = u(t) \tag{4.16}$$

It is important to remark that we want our numerical methods to not only be convergent on a single problem, but on all initial value problems for any reasonable starting values.

Let us consider the trivial test problem (4.12), which has the almost trivial constant solution $y(t) = \alpha$. We may ask what is required for a linear multistep method to reproduce this solution?

To solve this trivial test problem, we can employ a linear multistep method (4.1) to this test problem where $f(y,t) = 0$ and we find the following difference equation

$$y_{n+k} + \alpha_{k-1} y_{n+k-1} + \cdots + \alpha_0 y_n = 0 \tag{4.17}$$

This is a linear homogeneous difference equation of order k. In order to find all solutions to this equation it is standard to try solutions of the form $y_n = \zeta^n$. By this assumption, we find the so-called *characteristic equation* of the difference equation

$$\rho(\zeta) = \zeta^k + \alpha_{k-1}\zeta^{k-1} + \cdots + \alpha_0 = 0, \tag{4.18}$$

having divided by the common factor ζ^n. Thus, the resulting equation can be considered as a polynomial equation in ζ. The solutions ζ_j, $j = 1, 2, ..., k$ to this polynomial equation is the set of roots of the first characteristic polynomium $\rho(\zeta)$. The solution to the difference equation (4.17) is then a linear combination of such solutions because of the linear character of the difference equation

$$y_n = \sum_{j=1}^{k} d_j \zeta_j^n, \tag{4.19}$$

where the coefficients d_j, $j = 1, 2, ..., k$ have to be determined from the initial conditions of the problem. The nature of the solution (4.19) is that it stays bounded if and only if the roots $\mid \zeta_j \mid \leq 1$ for all $j = 1, 2, \ldots, k$. We say that they satisfy the *Root Condition*, cf. Definition 4.4.8. One such root is already known since a linear multistep method can only be consistent if $\rho(1) = 0$, which means that for a consistent linear multistep method we will always have a *principal root* $\zeta_1 = 1$. If in addition, all other complex roots of the difference equation (4.18) satisfy the root condition it is possible to show that the solution will be bounded which is necessary for convergence.

Theorem 4.5.11. Dahlquists Equivalence Theorem.
A linear multistep method (4.1) is convergent if and only if it is consistent and zero-stable.

The theorem is more general than just for the trivial differential equation with zero right hand side but we will not go deeper in the proof for the general ODE-problem. Interested readers are referred to [20, 8].

4.6 Adams Methods

We will now consider the most popular methods belonging to the family of Multistep formulas. We follow the development that has been described in the previous sections and develop formulas for integration of the general initial value problem

$$y' = f(t, y) \tag{4.20}$$

By an exact integration of this formula from t_n to t_{n+1} we find

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(t, y(t))dt \tag{4.21}$$

To determine the integral, we interpolate the integrand $f(t, y)$ using a polynomial $I(t)$ to determine a quadrature formula to approximate the integral. The polynomial can be defined in terms of Newton's backward divided difference formula assuming equi-distant position of nodes as described in Section A.3. We replace the exact solution with $y_n = y(t_n)$ and find

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} I(t)dt \tag{4.22}$$

To define the Adams family of methods we apply interpolation by *backward differences* (cf. Section A.3) and replace the integrand $I(t)$ with a polynomial approximation of order $k-1$ interpolating the function $f(t, y)$ at the points $t_{n-i} = t_n + ih$, $i = -k+1, -k+2, ..., 1$, of the form

$$I(t) \cong I_{k-1}(t) = I_{k-1}(t_n + rh) = \sum_{i=0}^{k-1} (-1)^i \begin{pmatrix} -r \\ i \end{pmatrix} \nabla^i f_n \tag{4.23}$$

where the Binomial coefficient is defined as

$$\begin{pmatrix} n \\ k \end{pmatrix} \equiv \frac{n!}{k!(n-k)!} = \frac{n(n-1)(n-2)\cdots(n-k+1)}{k!} \tag{4.24}$$

where k is a nonnegative integer and $n \in \mathbb{R}$.

The methods that results from (4.22) will be referred to as Adams methods. If the point t_{n+1} is included as an interpolation point in the polynomial approximation then the method will be implicit and otherwise explicit. However, note that in either case the integration interval is the same. Thus, for the explicit cases the integral is determined by extrapolation of the interpolating polynomial to the integration interval between $t_n \leq t \leq t_{n+1}$. The integral is hereby approximated by a k'th order polynomial approximation which determines the order of accuracy of the resulting Adams method.

The integral of (4.22) can by transforming the integration limits of the integral be rewritten into

$$\int_{t_n}^{t_{n+1}} f(t)dt = h \int_0^1 f(t_n + rh)dr \tag{4.25}$$

where the time step size is $h = t_{n+1} - t_n$. By replacing the integrand with a polynomial approximation, we find that we can express the integral in the form

$$\int_0^1 I(t_n + rh)dr = \int_0^1 \sum_{i=0}^{k-1} (-1)^i \begin{pmatrix} -r \\ i \end{pmatrix} \nabla^i f_n dr \tag{4.26}$$

Since polynomials are easily integrated, it is then possible to express the Adams family of methods in the form

$$y_{n+1} = y_n + h \sum_{i=0}^{k} \gamma_i^* \nabla^i f_n \tag{4.27}$$

where the methods are defined by the coefficients

$$\gamma_i^* \equiv \int_0^1 (-1)^i \begin{pmatrix} -r \\ i \end{pmatrix} dr \tag{4.28}$$

We can compute the coefficients for the explicit Adams-Bashforth method as

$$\begin{aligned}
\gamma_0^* &= \int_0^1 1dr = 1 \\
\gamma_1^* &= -\int_0^1 -rdr = \left[\tfrac{1}{2}r^2\right]_0^1 = \tfrac{1}{2} \\
\gamma_2^* &= \int_0^1 \tfrac{-1}{2} r(-r-1)dr = \tfrac{1}{2}\left[\tfrac{1}{3}r^3 + \tfrac{1}{2}r^2\right]_0^1 = \tfrac{5}{12}
\end{aligned}$$

It can be shown, e.g. see [9], that we need to satisfy conditions in the form

$$\sum_{s=0}^{i} \frac{1}{i-s+1} \gamma_s^* = 1 \tag{4.29}$$

which can be used to generate the unknown coefficients using the following explicit *recursion formula*

$$\gamma_0^* = 1, \quad \gamma_i^* = 1 - \sum_{s=0}^{i-1} \frac{1}{i-s+1}\gamma_s^*, \quad i = 1, 2, ... \tag{4.30}$$

The first three coefficients can then be derived explicitly using this recursion formula as

$$\begin{aligned}
\gamma_0^* &= 1 \\
\gamma_1^* &= 1 - \left(\tfrac{1}{2}\gamma_0^*\right) = \tfrac{1}{2} \\
\gamma_2^* &= 1 - \left(\tfrac{1}{3}\gamma_0^* + \tfrac{1}{2}\gamma_1^*\right) = \tfrac{5}{12}
\end{aligned}$$

In a similar procedure, the coefficients for the implicit Adams-Moulton method can be computed from the integral formula,

$$\int_{t_n}^{t_{n+1}} f(t)dt = h\int_{-1}^{0} f(t_n + rh)dr \tag{4.31}$$

By replacing the integrand with a polynomial approximation we find

$$\int_{-1}^{0} I(t_n + rh)hdr = h\int_{-1}^{0} \sum_{i=0}^{k} (-1)^i \begin{pmatrix} -r \\ i \end{pmatrix} \nabla^i f_{n+1} dr \tag{4.32}$$

and then integrate the terms of the polynomial such that we find the scheme

$$y_{n+1} = y_n + h\sum_{i=0}^{k} \gamma_i \nabla^i f_{n+1} \tag{4.33}$$

The coefficients for the first coefficients are

$$\begin{aligned}
\gamma_0 &= \int_{-1}^{0} 1dr = 1 \\
\gamma_1 &= \int_{-1}^{0} -rdr = \left[\frac{r^2}{2}\right]_{-1}^{0} = -\tfrac{1}{2} \\
\gamma_2 &= \int_{-1}^{0} \frac{-r(-r-1)}{2}dr = \frac{1}{2}\left[\frac{1}{3}r^3 - \frac{1}{2}r^2\right]_{-1}^{0} = -\tfrac{1}{12}
\end{aligned}$$

A recurrence relation for generating the implicit coefficients can be shown to be given as

$$\gamma_0 = 1, \quad \sum_{s=0}^{i} \frac{1}{i-s+1}\gamma_s = 0. \tag{4.34}$$

Using this recurrence relation, we can compute the coefficients for the implicit Adams methods using

$$\gamma_0 = 1, \quad \gamma_i = -\sum_{s=0}^{i-1} \frac{1}{i-s+1}\gamma_s. \tag{4.35}$$

We emphasize that the expressions for the linear multistep methods derived using the backward difference formulas for the Adams methods can be rewritten into equivalent forms for general linear multistep methods.

The explicit equivalent Adams-Bashforth predictor formulas can be stated as

$$y_{n+1} = y_n + h\sum_{i=0}^{k-1} \gamma_i^* \nabla^i f_n \tag{4.36}$$

or in the general form as

$$y_{n+1} = y_n + h\sum_{i=0}^{k-1} \beta_{k,i}^* f_{n-i} \tag{4.37}$$

The connections between γ_i^* and $\beta_{k,i}^*$ coefficients for the Adams-Bashforth methods are defined by the formulas

$$\beta_{k,i}^* = (-1)^i \left[\sum_{j=i}^{k} \begin{pmatrix} j \\ i \end{pmatrix} \gamma_j^*\right], \quad i = 0, 1, ..., k-1 \tag{4.38}$$

Notice that for each value of k there is a vector of coefficients of length k that defines the k-step method considered.

k	p	C_{p+1}	$\beta_{k,1}^*$	$\beta_{k,2}^*$	$\beta_{k,3}^*$	$\beta_{k,4}^*$
1	1	$\frac{1}{2}$	1			
2	2	$\frac{5}{12}$	$\frac{3}{2}$	$-\frac{1}{2}$		
3	3	$\frac{3}{8}$	$\frac{23}{12}$	$-\frac{16}{12}$	$\frac{5}{12}$	
4	4	$\frac{251}{720}$	$\frac{55}{24}$	$-\frac{59}{24}$	$\frac{37}{24}$	$-\frac{9}{24}$

Table 4.1: Coefficients defining the Adams-Bashforth k-step methods of order $p = k$.

Similarly, the implicit Adams-Moulton corrector formulas can be stated as

$$y_{n+1} = y_n + h\sum_{i=0}^{k} \gamma_i \nabla^i f_{n+1} \tag{4.39}$$

or in the equivalent general form as

$$y_{n+1} = y_n + h \sum_{i=0}^{k} \beta_{k,i} f_{n-i+1} \tag{4.40}$$

The conversion formula for the Adams-Moulton coefficients is given by

$$\beta_{k,i} = (-1)^i \left[\sum_{j=i}^{k} \begin{pmatrix} j \\ i \end{pmatrix} \gamma_j \right] \tag{4.41}$$

The resulting table for the Adams-Moulton method is given here. Notice that the vector for the order k formula is of length $k+1$.

k	p	C_{p+1}	$\beta_{k,1}$	$\beta_{k,2}$	$\beta_{k,3}$	$\beta_{k,4}$
0	1	$-\frac{1}{12}$	1			
1	2	$-\frac{1}{24}$	$\frac{1}{2}$	$\frac{1}{2}$		
2	3	$-\frac{19}{720}$	$\frac{5}{12}$	$\frac{8}{12}$	$-\frac{1}{12}$	
3	4	$-\frac{3}{160}$	$\frac{9}{24}$	$\frac{19}{24}$	$-\frac{5}{24}$	$\frac{1}{24}$

Table 4.2: Coefficients defining the Adams-Moulton k-step methods of order $p = k+1$.

The set consisting of an explicit and an implicit method is called a pair because they are often used in combination as a predictor-corrector pair, cf. Section 4.7.

4.7 Predictor-Corrector Methods

The most efficient way of using the relatively superior properties of the implicit methods is in a combination of two methods, one explicit to give a predicted solution value with the implicit that corrects the solution to a more accurate one. In the so called Predictor-Corrector schemes we apply the two methods as follows:
Predictor, explicit of order k-1:

$$\tilde{y}_{n+1} = \tilde{y}_n + h \sum_{i=1}^{k-1} \beta^*_{ki} f_{n+i-k+1}$$

Corrector, implicit of order k :

$$y_{n+1} = y_n + h \sum_{i=1}^{k} \beta_{ki} f_{n+i-k+1}$$

Iteration in the Corrector:

$$y_{n+1}^{s+1} = y_n + h\sum_{i=1}^{k-1} \beta_{ki} f_{n+i-k+1} + h\beta_{kk} f(x_{n+1}, y_{n+1}^{s})$$

The iterative use of the corrector formula needs a criterion for stopping the iterations when the accuracy is acceptable using the following form of test.

$$\mid y_{n+1}^{s+1} - y_{n+1}^{s} \mid \leq \texttt{tol}$$

where `tol` is some small positive number that represents the accuracy threshold.

4.8 Stopping criterion for iterations

In determining when to stop iterating we will use a relative error criterion to compute the value of `tol`,

$$\mid y_{n+k}^{[\nu+1]} - y_{n+k}^{[\nu]} \mid < \texttt{tol} \mid y_{n+k}^{[\nu]} \mid$$

The tolerance `tol`> 0 has the same effect as a rounding error and will influence the global error accordingly. This means that we must choose `tol` small enough not to influence our results significantly. As a rule of thumb we select `tol`$< \sqrt{\mu}$, where μ is the rounding precision of the computer. This choice ensures a positive value and a number that is not unrealistically small.

4.9 Milne's Device

Let P represent application of the Predictor and C application of the Corrector and let E represent one evaluation of the right-hand-side function, we can then form sequences of methods in the following way

$P(EC)^{\mu}E$: one Predictor followed by μ times iterating the Corrector.

The Principal Local Truncation Error (PLTE) can be computed from the equation formed by taking the difference of the predicted value and the corrected value

$$y_{n+1}^{\mu} - y_{n+1}^{0} = PLTE/W$$

We can then, assuming the corrector is one order higher than the predictor obtain the following estimate:

$$PLTE = \frac{C_{p+1}}{C_{p+1}^{*} - C_{p+1}}(y_{n+1}^{\mu} - y_{n+1}^{0})$$

For error control we may require that the PLTE is kept below some specified tolerance ϵ

$$| PLTE | \leq \epsilon$$

The estimated PLTE can be used in extrapolation as

$$\hat{y}_{n+1} = y^{\mu}_{n+1} + PLTE$$

to modify the estimated solution value.

We can choose between using the estimated error to either give a more accurate result or to make it possible to adjust the step size so as to produce an error within a given tolerance. This technique will be treated in more detail later in the implementation chapter.

4.10 Backward Differentiation Formula (BDF) methods

The Backward Differentiation Formula (BDF) methods is an family of linear multistep methods in which the function value only will be calculated in the point that is going to be found. A variable number of points calculated in the steps before can be used. This means that a BDF method is an implicit multistep method where $\beta_0 \neq 0$ but $\beta_i = 0$, $i = 1, ..., k$. The class of BDF methods got its name because of the general formula for the methods, that can be written in the form

$$hy'_n = \sum_{j=1}^{k} \frac{1}{j} \nabla^j y_n,$$

where ∇ is the backward difference operator. The BDF methods are particular useful for solving stiff ODEs due to their stability properties.

By normalizing the coefficients which have been generated such that the normalization condition is $\alpha_k = 1$ is satisfied, we can then write the method in the general form of a linear multistep method as

$$\frac{1}{h} \sum_{j=0}^{k} \alpha_j y_{n-j} = \beta_0 f(t_n, y_n)$$

The simplest BDF method is the Implicit Backward Euler method

$$y_n = y_{n-1} + hf(t_n, y_n)$$

which is a first order accurate method.

The family of BDF methods of order $1 \leq p \leq 6$ is given in Table 4.10.

r	α_6	α_5	α_4	α_3	α_2	α_1	α_0	β_r	p	C_{p+1}
1						1	-1	1	1	$-\frac{1}{2}$
2					1	$-\frac{4}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	2	$-\frac{2}{9}$
3				1	$-\frac{18}{11}$	$\frac{9}{11}$	$-\frac{2}{11}$	$\frac{6}{11}$	3	$-\frac{3}{22}$
4			1	$-\frac{48}{25}$	$\frac{36}{25}$	$-\frac{16}{25}$	$\frac{3}{25}$	$\frac{12}{25}$	4	$-\frac{12}{125}$
5		1	$-\frac{300}{137}$	$\frac{300}{137}$	$-\frac{200}{137}$	$\frac{75}{137}$	$-\frac{12}{137}$	$\frac{60}{137}$	5	$-\frac{10}{137}$
6	1	$-\frac{360}{147}$	$\frac{450}{147}$	$-\frac{400}{147}$	$\frac{225}{147}$	$-\frac{72}{147}$	$\frac{10}{147}$	$-\frac{60}{147}$	6	$-\frac{20}{343}$

Table 4.3: Coefficients of BDF methods of order $1 \leq p \leq 6$.

4.11 Implementation of Predictor-Corrector Methods

The actual use of a multistep method is always in the shape of a Predictor-Corrector method. The diagram below shows the basic layout of such a scheme.

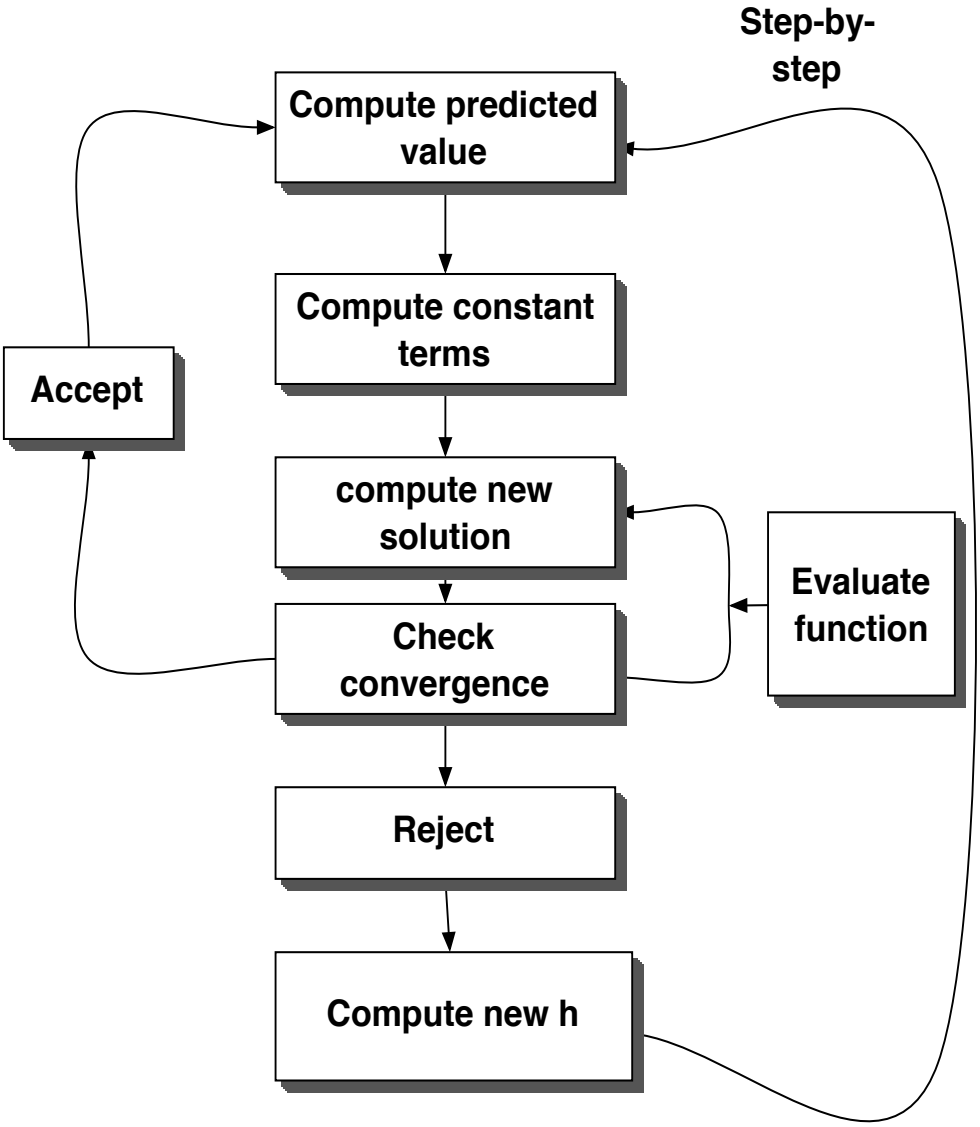


Figure 4.2: The layout of a Predictor-Corrector code.

Exercises

1. Show that Euler and Trapez methods both satisfy the conditions for Dahlquists Equivalence Theorem.

2. Show that due to their design all Adams methods are convergent.

3. Derive the BDF-k formulas for $k = 2$ and $k = 3$.

4. Show that after only one iteration using the Adams PECE scheme the order of the corrector is achieved.

5. Discuss what step sizes that can be used for a Linear Multistep Method of order p for the solution of an initial value problem where the solution is known to be a polynomial of the same order.

6. Is it possible to find a Linear Multistep Method which can provide absolutely stable solutions to the initial value problem $y' = y$ with initial condition $y(0)$?

Chapter 5

One-step methods

In this chapter we will introduce the family of methods that unlike the multistep methods do not make use of any past steps to compute the next solution point. These methods are known as one-step methods and include the family of Runge-Kutta methods and are all *self-starting*. These were used in the beginning of the 1900's by Runge and Kutta [37], two german mathematicians, to compute accurate solutions to problems from astronomy.

Definition 5.0.12. Explicit one-step methods:

$$y_{n+1} = y_n + h\Phi(t_n, y_n, h) \tag{5.1}$$

This definition is for an explicit method where the right hand side can be computed explicitly from the values of the solution at point (t_n, y_n). We will later generalise the definition to also include implicit methods. The function $\Phi(t, y, h)$ needs to be defined, and we will apply the following

Definition 5.0.13. Consistency of One-step methods:
The method (5.1) is **consistent** with the initial value problem if

$$\Phi(t, y, 0) = f(t, y) \tag{5.2}$$

Further we will study the accuracy of the one-step method using the following

Definition 5.0.14. Order of accuracy for explicit one-step methods: The method (5.1) has order p if p is the largest integer for which

$$y(t+h) - y(t) - h\Phi(t, y(t), h) = \mathcal{O}(h^{p+1}) \tag{5.3}$$

We see that the numerical solution satisfies (5.1) exactly while the exact solution inserted in the equation will lead to a residual of order p in the stepsize h. The difference between the two is the local truncation error.

If we put the function Φ equal to the right hand side and insert the taylor expansion for $y(t+h)$ we get

$$\begin{aligned} y(t+h) - y(t) - h\Phi(t, y, h) &= y(t+h) - y(t) - h\Phi(t, y, 0) \\ &= hy'(t) - h\Phi(t, y, 0) + \mathcal{O}(h^2) = \mathcal{O}(h^2) \end{aligned} \tag{5.4}$$

This shows that using Eulers method we get first order, and that any explicit one-step method that is consistent has local truncation order of accuracy at least one. Furthermore, it is seen that the one-step error in (5.4) is $\mathcal{O}(h^2)$, i.e. one order larger than the local truncation error of $\mathcal{O}(h)$.

5.1 Global errors for one-step methods

The case of a general one-step method (5.1) of order q can be treated in a way similar to Euler's method. The local one step error is defined by

$$d_n = y(t_n) - y_n = h^{q+1}\Psi(t_n, y_n) = \mathcal{O}(h^{q+1}) \tag{5.5}$$

We now consider the exact solution and the numerical solution

$$\begin{aligned} y(t_{n+1}) &= y(t_n) + h\Phi(t_n, y(t_n), h) + h^{q+1}\Psi(t_n, y(t_n)) \\ y_{n+1} &= y_n + h\Phi(t_n, y_n, h) \\ e_{n+1} &= y(t_{n+1}) - y_{n+1} \\ &= y(t_n) + h\Phi(t_n, y(t_n), h) + h^{q+1}\Psi(t_n, y(t_n)) - y_n - h\Phi(t_n, y_n, h) \\ &= y(t_n) - y_n + h(\Phi(t_n, y(t_n), h) - \Phi(t_n, y_n, h)) + h^{q+1}\Psi(t_n, y(t_n)) \\ &= e_n + h\frac{\partial \Phi}{\partial y}e_n + h^{q+1}\Psi(t_n, y(t_n)) \end{aligned}$$

We now assume that we can make the following bounds

$$\left|\frac{\partial \Phi}{\partial y}\right| \leq L, \quad \text{and } |\ \Psi(t_n, y(t_n))\ | \leq M \tag{5.6}$$

The first bound comes from the consistency requirement on the function $\Phi(t, y, h)$. This leads to the difference inequality

$$\mid e_{n+1} \mid \leq (1 + hL) \mid e_n \mid + h^{q+1} M \tag{5.7}$$

This is very similar to what we got for Euler's method. We now apply the assumption that our initial error is zero and we can derive the general bound for the one-step method

$$\mid e_{n+1} \mid \leq \frac{A^{n+1} - 1}{A - 1} h^{q+1} M = \frac{(1 + hL)^{n+1} - 1}{hL} h^{q+1} M = [(1 + hL)^{n+1} - 1] h^q \frac{M}{L} \tag{5.8}$$

Using the comparison with the exponential function we can simplify this into the bound after n steps

$$\mid \epsilon_n \mid \leq \frac{h^q M}{L} (e^{L(t_n - t_0)} - 1) = \mathcal{O}(h^q) \tag{5.9}$$

This result shows that the order of the global error is one order lower than the order of the local truncation error. We also remark that for $q \geq 1$ we get convergence in the limit as $h \to 0$.

We will define a one-step method using Taylors expansion of the right hand side function

Definition 5.1.15. Taylor's One-step method:

$$\Phi_T(t, y, h) = f(t, y) + \frac{h}{2!} f'(t, y) + \cdots + \frac{h^q}{(q+1)!} \frac{d^q}{dt^q} f(t, y) + \ldots \tag{5.10}$$

Let us apply Taylor expansion with p terms in our order defining expression (5.3) this leads to

$$y(t + h) - y(t) - h\Phi_T(t, y, h) = \frac{h^{p+1}}{(p+1)!} \frac{d^p}{dt^p} f(t, y) + \cdots = \mathcal{O}(h^{p+1}) \tag{5.11}$$

This shows that we obtain a method of order p.

5.2 Runge-Kutta methods

The class of one-step methods that have been mostly used in the past are the Runge-Kutta methods [37] and they will be used here to illustrate the theory. As an example of a one-step method we consider an explicit Runge-Kutta method with two stages. The following geometric construction illustrates the general idea.

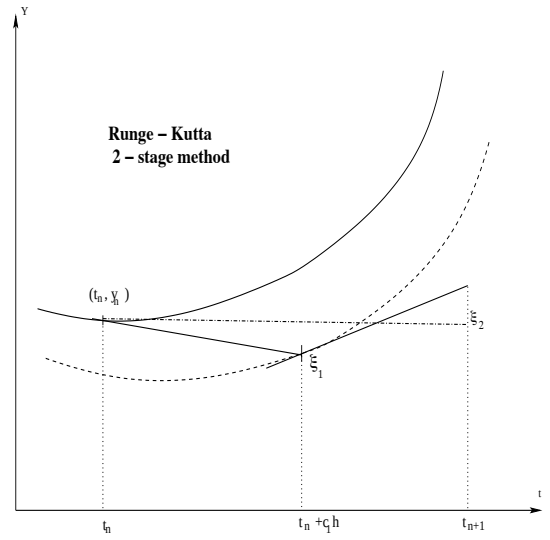


Figure 5.1: Geometric construction of a two-stage Runge-Kutta method.

A general two-stage explicit Runge-Kutta method can be expressed as

$$\begin{aligned} \xi_1 &= y_n \\ \xi_2 &= y_n + a_{2,1} h f(t_n, \xi_1) \\ y_{n+1} &= y_n + h(b_1 f(t_n, \xi_1) + b_2 f(t_n + c_2 h, \xi_2)) \end{aligned} \tag{5.12}$$

Here the four coefficients c_2, $a_{2,1}$, b_1 and b_2 will define the method among a whole family of possible choices. Let us for example use $c_2 = \frac{1}{2}$, $a_{2,1} = \frac{1}{2}$, $b_1 = 0$ and $b_2 = 1$. This gives the following method

$$\begin{aligned} \xi_1 &= y_n \\ \xi_2 &= y_n + \tfrac{1}{2} h f(t_n, \xi_1) \\ y_{n+1} &= y_n + h f(t_n + \tfrac{1}{2} h, \xi_2) \end{aligned} \tag{5.13}$$

This construction corresponds to the case where we evaluate the function at the midpoint of the step and use the tangent at this point all the way from the left endpoint to the right endpoint. The principle of combining several stages in the form of function- or slope-values can be generalized as in the following definition of

a ν-stage explicit Runge-Kutta method

$$\begin{aligned}
\xi_1 &= y_n \\
\xi_2 &= y_n + a_{2,1} h f(t_n, \xi_1) \\
\xi_i &= y_n + h \sum_{j=1}^{i-1} a_{i,j} f(t_n + c_i h, \xi_j) \\
y_{n+1} &= y_n + h \sum_{i=1}^{\nu} b_i f(t_i, \xi_i)
\end{aligned}$$

The explicitness of the method makes it possible to evaluate each stage evaluation when the previous stages have been evaluated. The dependency is backwards. In order for the function values to be consistent the $a_{i,j}$ coefficients must satisfy the *row-sum conditions*

$$c_i = \sum_{j=1}^{\nu} a_{i,j} \text{ for } i = 1, 2, \ldots, \nu \tag{5.14}$$

Together with this definition we consider the so-called Butcher tableau [4] holding the coefficients of the one-step method. This tableau is a standard way to present Runge-Kutta Methods. The last line of coefficients are here for generality and later use but defines like the **b** coefficients a weighted sum of function values. The general Butcher tableau for a Runge-Kutta method is written as

c_1	$a_{1,1}$	$a_{1,2}$	$\cdots$	$a_{1,\nu}$
c_2	$a_{2,1}$	$a_{2,2}$	$\cdots$	$a_{2,\nu}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
c_ν	$a_{\nu,1}$	$a_{\nu,2}$	$\cdots$	$a_{\nu,\nu}$
$\boldsymbol{b}$	b_1	b_2	$\cdots$	b_ν
$\boldsymbol{d}$	d_1	d_2	$\cdots$	d_ν

This shows, that a Runge-Kutta method can be defined from three vectors **b**, **c**, **d** and one matrix **A**. **d** denotes a row of coefficients used in a step for error estimation which will be considered in section 5.4. Thus, the shorthand version of the general Butcher scheme can be written as

c	**A**
	b
	d

For a general explicit two-stage Runge-Kutta method (5.12) the Butcher tableau is given as

c_1	0	0
c_2	$a_{2,1}$	0
$\boldsymbol{b}$	b_1	b_2

The tableau shows that the explicit scheme has a matrix $\mathbf{A}$ that is lower triangular, all elements above the diagonal are born to be zero. For completeness we present an important four-stage one-estep method which is known in the litterature as the explicit four-stage Runge Kutta method (ERK4). The method is a fourth order accurate method, and it has the following Butcher tableau

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

There is some confusion in the literature about the nomenclature for Runge-Kutta methods. In the following, we adopt the nomenclature illustrated in Figure 5.2 which is based on the structure of the $\mathbf{A}$ matrix. The family of Runge-Kutta methods has many famous members, in the appendix we list some of the best known methods, in particular those implemented in the Matlab ODE-solver toolbox.

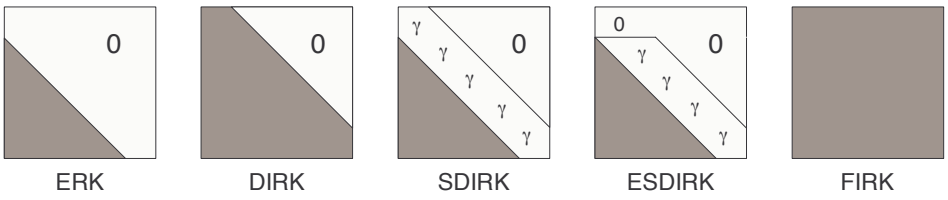


Figure 5.2: Structure of the $\mathbf{A}$ matrix for different classes of Runge-Kutta Methods.

For explicit methods (ERK), $\mathbf{A}$ is strictly lower triangular, and all calculations are done explicitly with low computational cost, but ERK methods are not suited for stiff problems. All implicit methods require in general some sort of iterative solution. For diagonal implicit Runge-Kutta (DIRK) methods the iterations in the internal stages are carried out one at a time, lowering the cost compared to fully implicit methods, but still attaining good stability properties. If the diagonal elements are identical asin the singly diagonally implicit Runge-Kutta Methods (SDIRK), the iteration matrix can be reused, which saves a significant number of (expensive) $\mathbf{LU}$ factorizations in the solution. The computational effort can be reduced further by reusing the last computed value in the first step, which is done in the explicit SDIRK method (ESDIRK). Finally, if the $\mathbf{A}$ matrix is dense, we have the class of fully implicit Runge-Kutta methods (FIRK). In choosing between the different classes of Runge-Kutta method for a specific initial value problem, we will need to consider both the

properties and performance of different methods to decide what is optimal for the problem at hand.

5.3 The order of Runge-Kutta methods

We are interested in analysing the accuracy of the general Runge-Kutta method. This leads to rather complicated expressions and we limit ourselves to a method with three stages. For the error analysis we apply the scheme to the exact solution $y(t)$ and derive the Taylor expansion over one step from the point $(t_n, y(t_n))$. In the following expressions we leave out the arguments of the function evaluations and in order to shorten the expressions we make use of the following notation:

$$F = f_t + f f_y \ , \quad G = f_{tt} + 2f f_{ty} + f^2 f_{yy} \tag{5.15}$$

Taylor expansion of the exact solution leads to

$$y(t_{n+1}) = y(t_n) + hf + \frac{h^2}{2}F + \frac{h^3}{6}[F f_y + G] + \mathcal{O}(h^4) \tag{5.16}$$

The first three stages can be expanded as follows

$$\begin{aligned} k_1 &= f(t_n, \xi_1) = f \\ k_2 &= f(t_n + c_2 h, \xi_2) \\ &= f + hc_2(f_t + k_1 f y) + \frac{h^2}{2}c_2^2[f_{tt} + 2k_1 f_{ty} + k_1^2 f_{yy}] + O(h^3) \\ k_3 &= f(t_n + c_3 h, \xi_3) = f + hc_3 F + h^2[c_2 a_{32} F f_y + \frac{1}{2}c_3^2 G] + \mathcal{O}(h^3) \end{aligned} \tag{5.17}$$

We insert these expressions in the formula for the new y-value and obtain

$$\begin{aligned} y_{n+1} = y(t_n) &+ h(b_1 + b_2 + b_3)f + h^2(b_2 c_2 + b_3 c_3)F \\ &+ \frac{h^3}{2}[2b_3 c_2 a_{32} F f_y + (b_2 c_2^2 + b_3 c_3^2)G] + \mathcal{O}(h^4) \end{aligned}$$

If the terms for each power of h should agree this leads to the following set of conditions

$$\begin{aligned} \mathcal{O}(h): &\quad (b_1 + b_2 + b_3) = 1 \\ \mathcal{O}(h^2): &\quad (b_2 c_2 + b_3 c_3) = \frac{1}{2} \\ \mathcal{O}(h^3): &\quad (b_2 c_2^2 + b_3 c_3^2) = \frac{1}{3} \\ \text{and} &\quad b_3 c_2 a_{3,2} = \frac{1}{6} \end{aligned}$$

If we want higher order, e.g. order four, then we must satisfy four additional conditions. However, with only six coeficients it implies that there are no chances of

getting higher order. We can conclude that with three stages the maximum order we can obtain is three and any set of coefficients that satisfies the four conditions will result in this order of accuracy.

For completeness and without going into details we list the set of conditions for obtaining fourth order with at least four stages.

$$
\begin{aligned}
\mathcal{O}(h): \quad & \sum_i b_i = 1 \\
\mathcal{O}(h^2): \quad & \sum_i b_i c_i = \tfrac{1}{2} \\
\mathcal{O}(h^3): \quad & \sum_i b_i c_i^2 = \tfrac{1}{3} \\
& \sum_i \sum_k b_i a_{ik} c_k = \tfrac{1}{6} \\
\mathcal{O}(h^4): \quad & \sum_i b_i c_i^3 = \tfrac{1}{4} \\
& \sum_i \sum_k b_i c_i a_{ik} c_k = \tfrac{1}{8} \\
& \sum_i \sum_k b_i a_{ik} c_k^2 = \tfrac{1}{12} \\
& \sum_i \sum_j \sum_k b_i a_{ij} a_{jk} c_k = \tfrac{1}{24}
\end{aligned}
$$

Satisfying all of these conditions is a requirement for at least fourth order. For schemes of higher accuracy similar conditions can be determined.

Attainable order of explicit Runge-Kutta methods

An interesting question is what order can be achieved by an explicit s-stage method. This question has been given some attention by (Butcher 1987) and the result is given in the table below.

Order	1	2	3	4	5	6	7	8	9
min # of stages	1	2	3	4	6	7	9	11	12

Table 5.1: Attainable order for explicit Runge-Kutta methods.

Example of deriving a third order Runge-Kutta method

We illustrate the use of the order conditions by construction a three stage Runge-Kutta method of order three. Initially we see that we have to satisfy four conditions

and the number of coefficients is eight. For the two *row-sum conditions* we get

$$a_{2,1} = c_2 \quad \text{and} \quad a_{3,1} + a_{3,2} = c_3.$$

The resulting total number of conditions is six. This means that we have to more coefficients than the number of conditions. The result is that there exists a two-parameter family of methods of order three with three stages. We can select two coefficients and make the other six be determined by the conditions.

First choose f.ex. $c_3 = 1$ this means that the third function evaluation is at the right end of the step. Then choose the second function evaluation at the midpoint of the step $c_2 = \frac{1}{2}$. The resulting set of conditions to be satisfied is

$$\begin{aligned} b_1 + b_2 + b_3 &= 1 \\ \tfrac{1}{2}b_2 + b_3 &= \tfrac{1}{2} \\ \tfrac{1}{4}b_2 + b_3 &= \tfrac{1}{3} \\ b_3 a_{3,2} &= \tfrac{1}{6} \end{aligned}$$

The first three conditions form a linear system of equations to determine **b** and the last gives $a_{3,2}$ the remaining two are found from the *row-sum conditions*. The resulting method gives the following Butcher tableau

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

Table 5.2: A three-stage Runge Kutta method.

5.4 The local error

The Local truncation Error (LTE) is derived from the following expression for a p'th order method.

$$T_{n+1} = y(t_{n+1}) - \hat{y}_{n+1} = \Psi(y(t_n))h^{p+1} + \mathcal{O}(h^{p+2}) \tag{5.18}$$

where $\Psi(y(t_n))$ is a function of elementary differentials of order $p+1$ evaluated at the point $y(t_n)$. The notation $\tilde{y}_{n+1}$ indicates the value for y at t_{n+1} under the *localizing assumption* that $y_n = y(t_n)$.

A word of caution is that in general we can distinguish between two types of local truncation errors which is essentially the same thing but with a an order of difference

in magnitude. That is we either consider the local truncation error (LTE) or the one step error. The LTE is defined as the approximation error that arises when the derivatives of a mathematical equation is replaced by their discrete approximations. On the other hand, the local one step error is the error that is commited when we advanced the solution one step starting from an exact solution, i.e. the error we commit in one step. It is the accumulation of local errors in additions with representation errors and possibly modeling errors that leads to the concept of global errors.

We do not have a simple expression for the LTE like for multistep methods due primarily to the nonlinear nature of the order conditions. We may however give a relatively simple method for computing estimates of the LTE as we compute the solution. Let us concentrate on the three stage method of order three we derived in the above example shown in Table 5.2. We can embed a second order method in this scheme using a different vector of $\hat{\mathbf{b}}$ coefficients, this is done by satisfying the two order conditions for first and second order

$$\begin{aligned} \hat{b}_1 + \hat{b}_2 + \hat{b}_3 &= 1 \\ \tfrac{1}{2}\hat{b}_2 + \hat{b}_3 &= \tfrac{1}{2} \end{aligned} \tag{5.19}$$

This set of conditions should be used to form a solution value $\hat{y}_{n+1}$ from

$$\hat{y}_{n+1} = y_n + h\left(\hat{b}_1 f(\xi_1) + \hat{b}_2 f(\xi_2) + \hat{b}_3 f(\xi_3)\right) \tag{5.20}$$

Example of deriving a scheme for local error estimation

Using the conditions (5.19) we may chose one of the coefficients $\hat{b}_s$ and find the other two. We select f.ex. $\hat{b}_2 = \frac{1}{2}$ this leaves the two equations

$$\hat{b}_1 + \hat{b}_3 = \tfrac{1}{2}, \quad \hat{b}_3 = \tfrac{1}{4} \quad \Longrightarrow \hat{b}_1 = \tfrac{1}{4}$$

and we then find the resulting expression

$$\hat{y}_{n+1} = y_n + h(\tfrac{1}{4} f(\xi_1) + \tfrac{1}{2} f(\xi_2) + \tfrac{1}{4} f(\xi_3)) \tag{5.21}$$

The estimate for the local truncation error of the second order method is then found by subtracting the two solution expressions

$$\begin{aligned} e_T(n+1) &= \hat{y}_{n+1} - y_{n+1} \\ &= h(\hat{b}_1 - b_1) f(\xi_1) + (\hat{b}_2 - b_2) f(\xi_2) + (\hat{b}_3 - b_3) f(\xi_3)) \\ &= h(d_1 f(\xi_1) + d_2 f(\xi_2) + d_3 f(\xi_3)) \end{aligned}$$

In our example we find the coefficients d_s of the error estimator to be

$$d_1 = \tfrac{1}{12}, \quad d_2 = -\tfrac{1}{6}, \quad d_3 = \tfrac{1}{12}$$

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

Table 5.3: A three-stage Runge-Kutta method with error estimate.

Finally we give the full Butcher tableau for the method with error estimation included.

We carry out the following test for the three-stage method

$$y' = 4t\sqrt{y}, \quad y(0) = 1.$$

The results are presented in Figure 5.3, the lower right figure shows the optimal stepsize computed for a tolerance of 10^{-6}. This stepsize is computed from the following process.

Assume that we have a method where the LTE can be estimated and the order q is known. The estimate can be written

$$\text{LTE} \approx \Psi(y_n)h^q$$

Here h is the stepsize used to take the step and for the estimate of estimate the LTE. We now want to find the stepsize $\hat{h}$ that should have been used to give a LTE equal to a given tolerance τ.

$$\tau = \Psi(y_n)\hat{h}^q$$

If we take the ratio between these two equations we obtain

$$\frac{\tau}{LTE} \cong \frac{\Psi(y_n)\hat{h}^q}{\Psi(y_n)h^q} = \frac{\hat{h}^q}{h^q}$$

This leads to an equation for the estimated optimal stepsize $\hat{h}$ for obtaining the prescribed local accuracys.

$$\hat{h} = h\sqrt[q]{\frac{\tau}{LTE}} \tag{5.22}$$

From the results we see that the local error behaves like a third order process when $h \to 0$ this means that with $q = 3$ we should expect to be able to estimate a stepsize reasonably close to the optimal. For the integration with stepsize $h = 0.01$ we have

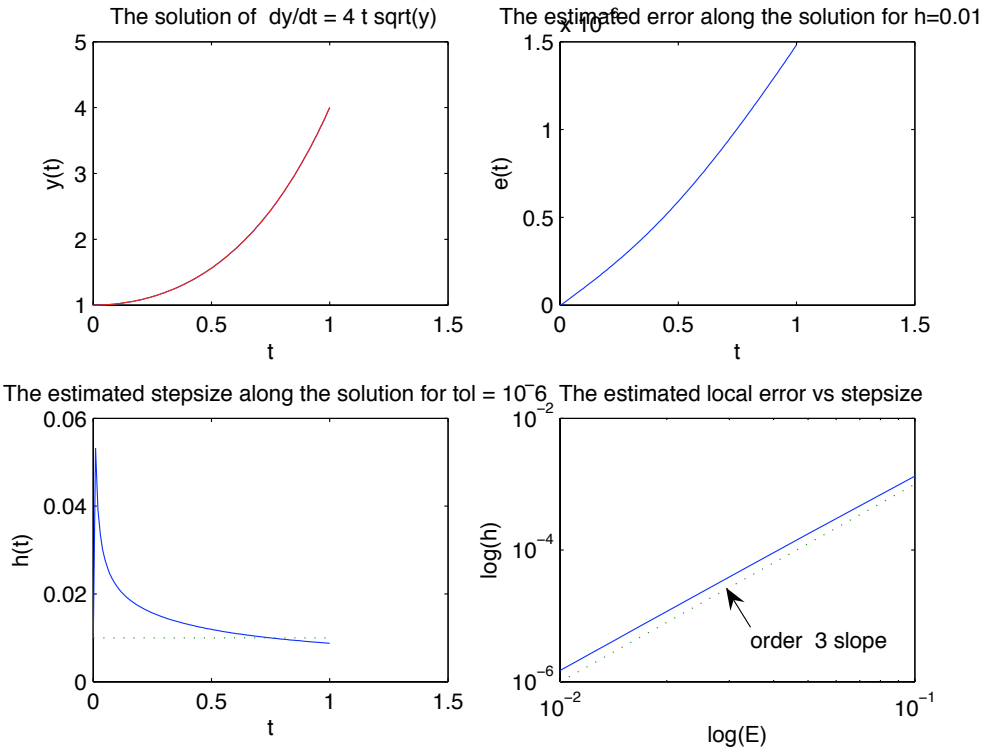


Figure 5.3: The three-stage Runge-Kutta method with error estimate.

computed the optimal stepsizes for each step. The resulting stepsizes are shown in the lower left diagram and we see that in the beginning our stepsize is far too small compared to the optimal but towards the end of the interval the stepsize is close to optimal, this is confirmed by the observed error being close to the tolerance in this region.

This experiment shows that for one-step methods we can step by step compute the optimum stepsize for controlling the size of the local error. This technique is known as *Automatic Stepsize Control* and it is used in commercial software like the integrators in the Matlab ODE-package [38].

A schematic view of a software implementation of the kind of method we have been analysing in the example is shown in Figure 5.4. This represents a general layout for such implementations with *Automatic Stepsize Control*.

5.5 Design of a RK-23 method

This section is an example of the design of a onestep method of Runge-Kutta type intended for implementation as an all-round non-stiff solver. The chapter is intended to illustrate the stages that enter the process of defining all features of a method that will enter the implementation.

The layout of the method

We start with presenting the Butcher scheme that defines the method layout. This example makes use of three stages and the intention is to find information on how much we can expect from accuracy and stability and what tools that are available for an all-round implementation.

The method is defined as follows

$$
\begin{aligned}
\xi_1 &= y_n \\
\xi_2 &= y_n + a_{2,1} h f(t_n, \xi_1) \\
\xi_3 &= y_n + a_{3,1} h f(t_n, \xi_1) + a_{3,2} h f(t_n + c_2 h, \xi_2) \\
y_{n+1} &= y_n + h(b_1 f(t_n, \xi_1) + b_2 f(t_n + c_2 h, \xi_2) + b_3 f(t_n + c_3 h, \xi_3)) \\
y_{n+1} &= y_n + h \sum_{i=1}^{3} b_i f(t_i, \xi_i), \quad t_i = t_n + c_i h \\
e_{n+1} &= h(d_1 f(t_n, \xi_1) + d_2 f(t_n + c_2 h, \xi_2) + d_3 f(t_n + c_3 h, \xi_3))
\end{aligned}
$$

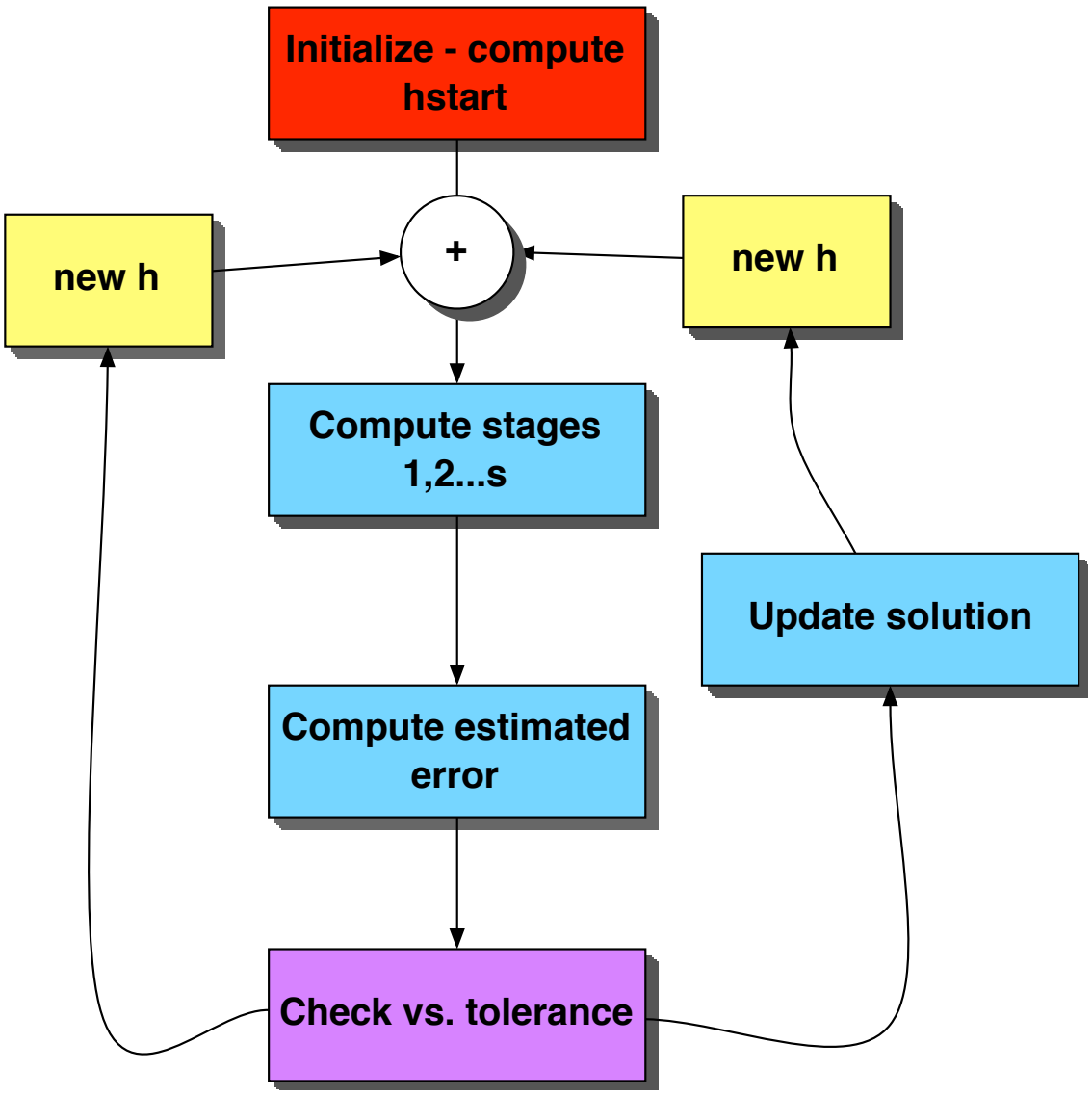


Figure 5.4: Layout of a steps for implementing a one-step method.

Butcher scheme

The method is characterized by its layout and the coefficients $a_{i,j}$, b_i and c_i and may appear complicated. In order to make it easier to distinguish between methods and different choices of the coefficients we introduce the Butcher tableau for the three-stage Runge-Kutta method.

c_1	0		
c_2	$a_{2,1}$	0	
c_3	$a_{3,1}$	$a_{3,2}$	0
$\boldsymbol{b}$	b_1	b_2	b_3
$\boldsymbol{d}$	d_1	d_2	d_3

Table 5.4: The Butcher Tableau.

If we want the method to be explicit, we need that

$$c_1 = 0$$

For the method to be consistent we will need to satisfy the conditions

$$c_2 = a_{2,1} \quad , \quad c_3 = a_{3,1} + a_{3,2}$$

This leaves six free parameters to be determined for the definition of our method.

Order relations

Earlier we have derived the relations that have to be satisfied for the method to have accuracy up to a given order. For the first orders we have

$$\begin{aligned} \mathcal{O}(h): &\quad (b_1 + b_2 + b_3) = 1 \\ \mathcal{O}(h^2): &\quad (b_2 c_2 + b_3 c_3) = \tfrac{1}{2} \\ \mathcal{O}(h^3): &\quad (b_2 c_2^2 + b_3 c_3^2) = \tfrac{1}{3} \\ \text{and} &\quad b_3 c_2 a_{3,2} = \tfrac{1}{6} \end{aligned}$$

This means that with the number of free coefficients that are available we can satisfy these four conditions. There are two free coefficients that we can choose for other purposes. We say that we have a two parameter family of methods of order three.

Stepsize control

We may consider comparing our result from the third order method with a second order method for estimation of the local truncation error of the second order method. This is done by selecting a new vector $\tilde{\mathbf{b}}$ for an alternative computation of the second order, two-stage method. If we choose $\tilde{b}_3 = 0$ we get

$$\tilde{y}_{n+1} = y_n + h(\tilde{b}_1 f(t_n, \xi_1) + \tilde{b}_2 f(t_n + c_2 h, \xi_2))$$

The estimate for the local truncation error can then be derived as the difference between second and third order solutions

$$e_{n+1} = \tilde{y}_{n+1} - y_{n+1} = h\sum_{i=1}^{3}(\tilde{b}_i - b_i)f(t_i, \xi_i) = h\sum_{i=1}^{3} d_i f(t_i, \xi_i)$$

Continuous extensions

Together with the discrete computation step by step we may find a continuous extension to the method that will be used for computing an approximate solution at any point between the mesh-points. To find this extension we consider the requirement for order two at a point $t_\theta = t_n + \theta h$. Using comparison with the Taylor expansion at that point evaluated from the point t_n we obtain the conditions for order of the method

$$y_{n,\theta} = y_n + h(p_1 f(t_n, \xi_1) + p_2 f(t_n + c_2 h, \xi_2) + p_3 f(t_n + c_3 h, \xi_3))$$

The unknown coefficents p_i can be found from satisfying the conditions

$$\begin{aligned}
&\mathcal{O}(h): && (p_1 + p_2 + p_3) = \theta \\
&\mathcal{O}(h^2): && (p_2 c_2 + p_3 c_3) = \tfrac{\theta^2}{2} \\
&\mathcal{O}(h^3): && (p_2 c_2^2 + p_3 c_3^2) = \tfrac{\theta^3}{3} \\
&\text{and} && p_3 c_2 a_{3,2} = \tfrac{\theta^3}{6}
\end{aligned}$$

With only three free parameters we can only satisfy three equations and must be content with order two. The system of equations is a linear system with polynomial right hand sides. The solution must be polynomials in θ. In matrix vector form the system looks like this

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & c_1 & c_2 \\ 0 & c_1^2 & c_2^2 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} \theta & 0 & 0 \\ 0 & \frac{\theta^2}{2} & 0 \\ 0 & 0 & \frac{\theta^3}{6} \end{bmatrix}$$

The elements in the vector $\mathbf{p}$ are polynomials of order three.

Discontinuities

The typical situation with discontinuous right hand sides in the differential equation is that a condition is given where the system changes when this condition is met. The simplest of these cases is when the solution meets a certain value y_D and the function

$$\phi(y, t) = y(t) - y_D$$

has a change of sign. For example this case is typical for a control system where y is a control variable that meets a condition set by the controller. The condition to satisfy is then a zero of the function

$$\phi(y^*, t^*) = y^*(t^*) - y_D = 0$$

In order to determine the value t^* of transition we may use the continuous extension. Doing so the problem is to find a value of θ that fulfills the requirement stated in the equation

$$\phi(y_{n,\theta}, t + \theta * h) = y_{n,\theta} - y_D = 0$$

The unknown in this equation is θ and the equation is a polynomial in θ making the problem one of finding the zero of a polynomial that lies in the interval $[0, 1]$.

The more general case of a discontinuity given by the condition

$$\phi(y^*, t^*) = 0$$

may be solved using the same technique but the problem is more general and certain assumptions must be satisfied by the function for the solution to be unique. Under this condition we can treat the situation quite the same way as the case above if we use a general root-finding procedure for determining the value of θ. In the literature this is treated under the name $event - handling$.

Stability

The stability properties of the method is treated in Chapter 6. Here the conclusion is that the stability region of all second or third order methods are those shown in Figure 6.3.

Implementation

The results above have to be given a specific form when the solver is implemented. This means that several choices have to be made. First of all the method has to be defined and its coefficients determined.

We approach this issue by looking at what may be obviously very cost effective choices. Take a Butcher scheme like this:

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

The Trapez method

This has the property that the second evaluation of the right hand side is at the endpoint of the interval, and function evaluations takes place only at the meshpoints. This is like switching between Euler and Trapezodal rule, in a manner similar to a Predictor-Corrector method.

Another Butcher scheme can be derived from choosing $c_1 = 0$, $c_2 = \frac{1}{2}$, $c_3 = 1$

0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	0	0
1	$\frac{1}{2}$	$\frac{1}{2}$	0
b	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$
d	d_1	d_2	d_3

The RK23 scheme

Optimal choices

Among the possible choices for the method defining coefficients it is possible to select some which give special properties for the method. One possibility is to find the method of order two with the smallest error constant. The error for a second order method has two terms, since the two conditions for order three are not satisfied. We can find the residual in these two equations and make the sum of the residuals as small as possible. Thereby we get an optimal method with respect to minimizing the local error. We may assume that all the coefficients are positive and the function to minimize is

$$TE = b_3 c_2 a_{3,2} + (b_2 c_2^2 + b_3 c_3^2) - \tfrac{1}{2}$$

Subject to the constraints

$$\begin{aligned} \mathcal{O}(h): &\quad (b_2 + b_3) = 1 - b_1 \\ \mathcal{O}(h^2): &\quad (b_2 c_2 + b_3 c_3) = \tfrac{1}{2} \end{aligned}$$

Where the order one constraint is including a choice for b_1. We will find the optimum for the case where $c_3 = 1$, this reduces the problem to finding the minimum for the

function

$$TE = b_3 c_2 a_{3,2} + (b_2 c_2^2 + b_3) - \tfrac{1}{2}$$

subject to the constraint

$$\begin{aligned} (b_2 + b_3) &= 1 - b_1 \\ (b_2 c_2 + b_3) &= \tfrac{1}{2} \end{aligned}$$

Chapter 6

Absolute stability

We have seen that sometimes our methods fail to give the kind of results that we are expecting. The errors may grow unbounded over the solution interval. Our aim in this chapter is to investigate why this happens and how to analyse those properties of our methods that guarantee stable solutions.

For the purpose of gaining insight in the nature of the problem for some of the methods we have considered earlier we start by looking at a model problem called the *Test Equation* defined as the linear problem

$$y' = \lambda y, \quad y_0 = \alpha \tag{6.1}$$

We know that the solution is given by $y(t) = \alpha \exp(\lambda t)$. The nature of this problem is shown in Figure 6.1. The solution is increasing (exponentially) for positive values

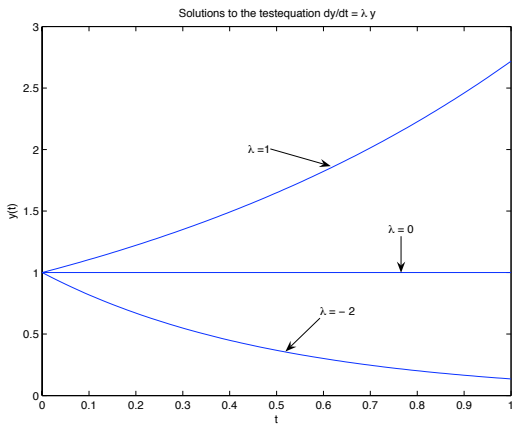


Figure 6.1: Characteristics of solutions to the test problem.

of λ it is constant for $\lambda = 0$ and goes asymptotically to zero for negative values of λ. We want our numerical solutions to behave just like the exact solution.

6.1 Stability for one-step methods

As a representative for one-step methods we first consider an explicit three-stage Runge-Kutta method defined from

$$\begin{array}{c|ccc} c_1 & 0 & & \\ c_2 & a_{2,1} & 0 & \\ c_3 & a_{3,1} & a_{3,2} & 0 \\ \hline & b_1 & b_2 & b_3 \end{array}$$

The coefficients for the method for order three are repeated here

$$\begin{aligned} \mathcal{O}(h): &\quad (b_1+b_2+b_3)=1 \\ \mathcal{O}(h^2): &\quad (b_2c_2+b_3c_3)=\tfrac{1}{2} \\ \mathcal{O}(h^3): &\quad (b_2c_2^2+b_3c_3^2)=\tfrac{1}{3} \\ \text{and} &\quad b_3c_2a_{3,2}=\tfrac{1}{6} \end{aligned} \tag{6.2}$$

This method is now applied to the test equation above and we get the following result for the three stages and the numerical solution y_{n+1} when the right hand side function is given as $f(y,t)=\lambda y$

$$\begin{aligned} k_1 &= \lambda y_n \\ k_2 &= \lambda(y_n + a_{2,1}hk_1) = \lambda(1+c_2h\lambda)y_n \\ k_3 &= \lambda(y_n + a_{3,1}hk_1 + a_{3,2}hk_2) \\ &= \lambda(1 + a_{3,1}h\lambda + a_{3,2}h\lambda(1+c_2h\lambda)y_n \\ &= \lambda(1 + c_3h\lambda + a_{3,2}c_2h^2\lambda^2)y_n \\ y_{n+1} &= y_n + h(b_1k_1 + b_2k_2 + b_3k_3) \\ &= y_n(1 + h\lambda(b_1+b_2+b_3) + h^2\lambda^2(b_2c_2+b_3c_3) + h^3\lambda^3b_3a_{3,2}c_2) \end{aligned}$$

where it has been taking into account that $a_{2,1}=c_2$ and $a_{3,1}+a_{3,2}=c_3$. It is then possible to make use of the conditions for order three and we obtain the result

$$y_{n+1} = y_n(1+\lambda h+\tfrac{1}{2}\lambda^2h^2+\tfrac{1}{6}\lambda^3h^3) = y_nP_3(h\lambda) \tag{6.3}$$

Here $P_3(h\lambda)$ is a third order polynomial. This polynomial is fundamental for all methods of order three when applied to the test equation and when defined form a one-step method with three stages independent of the choice of coefficients because the conditions given in (6.2) are fulfilled. In a similar way we can find expressions for orders $1,2,3,4$, below is a list of the corresponding polynomials. For convenience we have introduced the substitution $\hat{h}=h\lambda$.

$$\begin{aligned} P_1(\hat{h}) &= 1+\hat{h} \\ P_2(\hat{h}) &= 1+\hat{h}+\tfrac{1}{2}\hat{h}^2 \\ P_3(\hat{h}) &= 1+\hat{h}+\tfrac{1}{2}\hat{h}^2+\tfrac{1}{6}\hat{h}^3 \\ P_4(\hat{h}) &= 1+\hat{h}+\tfrac{1}{2}\hat{h}^2+\tfrac{1}{6}\hat{h}^3+\tfrac{1}{24}\hat{h}^4 \end{aligned}$$

Common for all the methods is that if we solve the test equation, our numerical result can be writen as

$$y_n = (P_\nu(\hat{h}))^n y_0 \tag{6.4}$$

where $P_\nu(\hat{h})$ is a polynomial which turns out to be a truncated series expansion of

$$y(nh) = e^{nh\lambda} y_0. \tag{6.5}$$

This result may be interpreted that the polynomials are approximations to the exponential function $\exp(\hat{h})$ of accuracy corresponding to the order of the method. Each term in the polynomial corresponds to the same order term in the series expansion of the exponential function. This confirms what could be expected from the evaluation of method order in the previous chapter.

Example of solution strategy for linear systems

Consider the linear system of coupled ODE's

$$\mathbf{y}' = \mathbf{A}\mathbf{y}, \quad \mathbf{y}_0 = \mathbf{a}. \tag{6.6}$$

This system is characterized by the eigenvalues of the matrix $\mathbf{A}$ of dimension mm Let the eigenvalues be $\lambda_s, s = 1, 2, ...m$ and assume they are distinct. We can then find a similarity transformation that diagonalise the matrix $\mathbf{SAS^{-1}} = \mathbf{D} = diag[\lambda_s]$. We apply the same transformation to the ODE-system $\mathbf{z} = \mathbf{SyS^{-1}}$ and get

$$\begin{aligned} \mathbf{Sy'S}^{-1} &= \mathbf{SAS}^{-1}\mathbf{SyS}^{-1} \\ \mathbf{z}' &= \mathbf{Dz} \end{aligned}$$

in this form the equations are decoupled and can be solved independently for principal solution components. Each solution component is of the form

$$z_s = \alpha_s e^{\lambda_s t}$$

We point out that the eigenvalues in general will be complex $\lambda_s \in \mathbb{C}$ and if they are real they will behave like shown in fig.(6.1) If there is an imaginary part this results in oscillatory behaviour, see Figure 6.2. If the eigenvalues are strictly in the left half plane the system is stable in the sense that all solutions tend to zero as t tends to infinity. The envelope of the oscillatory case is the exponentially decaying solution shown in the figure.

If we apply the general Runge-Kutta method we obtain (6.4) a difference equation for the numerical solution. We shall call $P_\nu(\hat{h})$ the *Stability function* for our method. Clearly $y_n \to 0$ as $n \to \infty$ if and only if

$$\mid P_\nu(\hat{h}) \mid < 1 \tag{6.7}$$

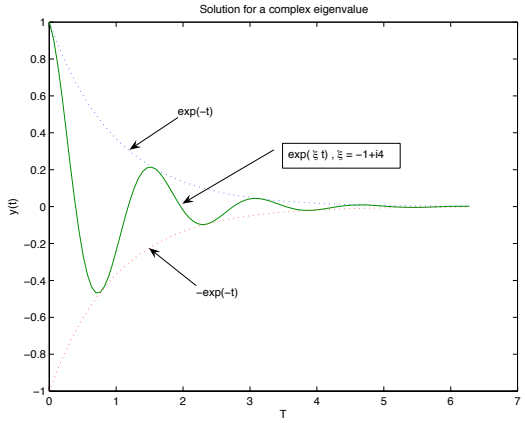


Figure 6.2: Test equation behaviour.

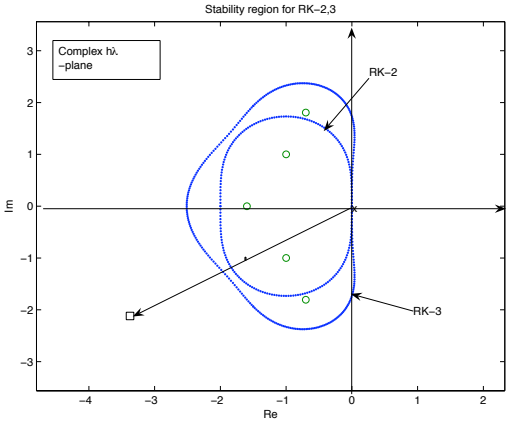


Figure 6.3: Stability regions for Runge-Kutta method of order two and three.

and the method is absolutely stable for those values of $\hat{h} \in \mathbb{C}$ for which this root condition holds. The region of the complex $\hat{h}$ -plane where this holds is called the *Stability region* $\mathbb{S}$, below is an example of such a region. This is the typical behavior of methods where the number of stages and the order are equal. in the Figure 6.3 is shown the stability regions for orders $q = 1, 2, 3, 4, 5$ and the size of the region is seen to increase with increasing order. For all the explicit methods we obtain stability properties with bounded stability regions. In order to get unbounded regions the methods have to be implicit, the first example being the Trapezoidal rule.

In Figure 6.3 a point $P = (-3.5, -2i)$ has been shown in the plot. This point represent the choice where $\lambda = -3.5 - 2i$ is complex valued. Stability is governed by

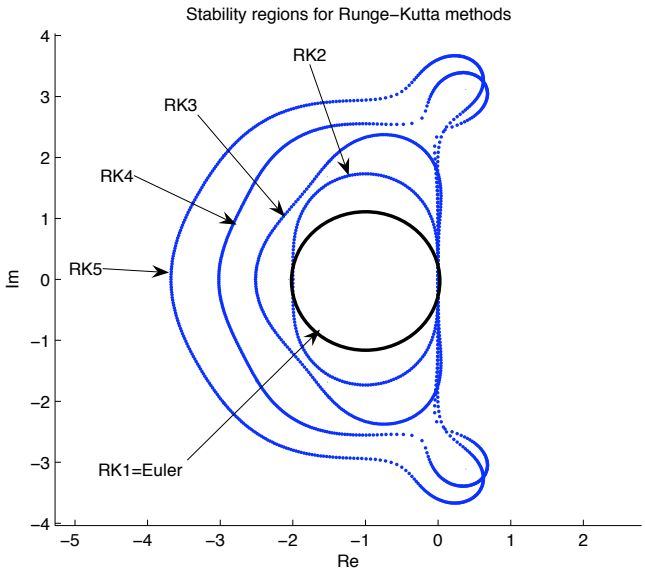


Figure 6.4: Stability region for Fehlberg Runge-Kutta methods of order 1-5.

the scaled value $h\lambda$ and therefore if the step size is set to $h = 1$ then the method is unstable. However, if we choose h to have a size that ensures that $h\lambda$ is just inside one of the stability regions, then the method can be shown to be stable. This is typical for explicit methods, and suggests that for unstable methods, we can test if it is related to an unstable choice of the step size, we can simply try and decrease the step size to turn the method stable.

The Trapezoidal rule

If we apply the method to the test equation we obtain the following result

$$y_{n+1} = y_n + \frac{\hat{h}}{2}(y_n + y_{n+1})$$

$$y_{n+1} = \frac{1 + \frac{\hat{h}}{2}}{1 - \frac{\hat{h}}{2}} y_n$$

The general expression for this type of result is

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

where

$$\mathbf{R}(\hat{h}) = \frac{1 + \frac{\hat{h}}{2}}{1 - \frac{\hat{h}}{2}}$$

Comparing this result to the expression for advancing one step along the exact solution we see that this can be interpreted as a rational approximation to the exponential function

$$\mathbf{R} = \frac{1 + \frac{\hat{h}}{2}}{1 - \frac{\hat{h}}{2}} \approx \exp(\hat{h})$$

The condition for absolute stability is $\mid \mathbf{R}(\hat{h}) \mid \leq 1$ and the region where this is satisfied is the complex left half plane , the imaginary axis is the stability boundary.

In a later section we will construct methods that have unlimited stability regions like the Trapezoidal rule.

6.2 Stability for multistep methods

The analysis for multistep methods follows the idea of applying the method to the linear test equation. For the k-step linear multistep method we get

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h \sum_{j=0}^{k} \beta_j \lambda y_{n+j}, \tag{6.8}$$

This can be rewritten in the form of a homogenous difference equation of order k

$$\sum_{j=0}^{k} \alpha_j y_{n+j} - h \sum_{j=0}^{k} \beta_j \lambda y_{n+j} = 0, \tag{6.9}$$

The solution of this difference equation is found in a similar way to the one we applied for the zero-stability case. Here we get the *characteristic polynomium*

$$\pi(r, \hat{h}) = \sum_{j=0}^{k} \alpha_j r^{n+j} - \hat{h} \sum_{j=0}^{k} \beta_j r^{n+j} = 0 \tag{6.10}$$

We rewrite this using the two characteristic polynomials from the definition of the multistep method

$$\pi(r, \hat{h}) = \rho(r) - \hat{h}\sigma(r) = 0 \tag{6.11}$$

This polynomium of degree k is called the *stability polynomium* for the linear multistep method. For different values of $\hat{h}$ we have different coefficients and this leads to different roots of the polynomial. Thus, the roots are dependent on $\hat{h}$. The solution y_n to the difference equation is composed from a linear combination of components that are each of the form $y_n = r_s^n$ where r_s is one of the k roots of the stability polynomium. We find the solution

$$y_n = \sum_{s=1}^{k} \delta_s r_s^n \tag{6.12}$$

The values δ_s are determined from satisfying k initial conditions.

The stability criterion can be found by requiring that the solution to the difference equation remains bounded for all n as $n \to \infty$ This requires all the roots to satisfy the condition

$$\mid r_s \mid \leq 1 \quad for \quad s = 1, 2, \ldots, k \tag{6.13}$$

The points in the $\hat{h}$ complex plane where $\mid r_s \mid = 1$ form the boundary for the stability region. This condition is equivalent to requiring that

$$r_s = e^{i\theta}, \quad for \quad 0 \leq \theta \leq 2\pi. \tag{6.14}$$

Since we know from the requirement of the method to be *zero-stable* that $\rho(1) = 0$ there is at least one point $(0, 0)$ where the method is stable. This corresponds to the value $\theta = 0$. The other points can be found from (6.11) where we can introduce (6.14) and get

$$\hat{h} = \frac{\rho(e^{i\theta})}{\sigma(e^{i\theta})}, \quad 0 \leq \theta \leq 2\pi. \tag{6.15}$$

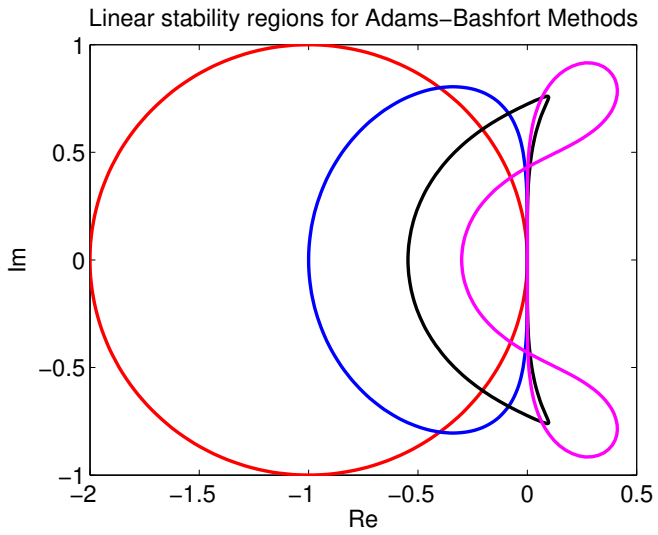


Figure 6.5: Stability regions for Adams-Bashforth methods of order 1, 2, 3 and 4.

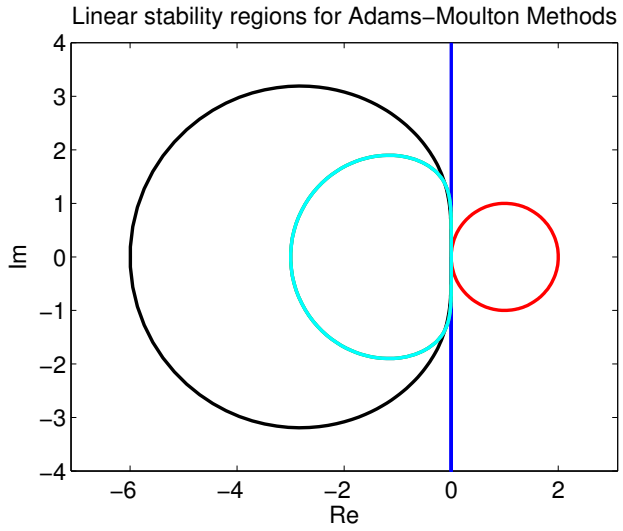


Figure 6.6: Stability regions for Backward Euler of order 1 together with Adams-Moulton methods of order 2, 3 and 4.

This formula will produce closed curves defining the boundaries of the stability regions. The linear stability regions for some explicit and implicit Adams methods are shown in Figures 6.5 and 6.6. The second order implicit Adams-Moulton method is the Trapezoidal rule and the stability boundary is the imaginary axis as shown in the Figure 6.6.

Example. We consider the Adams-Bashforth method for $k = 3$ as an example for the process of generating the stability region. First we find the coefficients of the method from the backward difference form.

$$y_{n+1} = y_n + h\left(f_n + \frac{1}{2}\Delta f_n + \frac{5}{12}\Delta^2 f_n\right)$$

We introduce the explicit expressions for the backward differences and get

$$y_{n+1} = y_n + h(\frac{23}{12}f_n - \frac{4}{3}f_{n-1} + \frac{1}{12}f_{n-2})$$

We now identify the stability polynomial from the coefficients in this formula and find

$$\pi(r, \hat{h}) = \rho(r) - \hat{h}\sigma(r) = r^3 - r^2 - \hat{h}\left(\frac{23}{12}r^2 - \frac{4}{3}r + \frac{1}{12}\right)$$

The explicit expression for the stability boundary curve is now

$$\hat{h} = \frac{e^{3i\theta} - e^{2i\theta}}{(\frac{23}{12}e^{2i\theta} - \frac{4}{3}e^{i\theta} + \frac{1}{12})}$$

The result is shown as the middle stability boundary on Figure (6.5).

6.3 A-Stability

The property that the stability region is unbounded means that stability does not impose severe restrictions on the choice of stepsize. Since this is important in connection with implementations of the methods for a general class of problems it has had a lot of attention in the past. The property that a method has unbounded stability region is used in the following

Definition 6.3.16. A-stability: A method is A-stable if the stability region contains all of the left half plane.

A slightly weaker property is stated by using the wedge shaped region shown in Figure (6.7) This type is taylored towards methods like the BDF-methods where a loop of the stability boundary cut into the left half plane.

Definition 6.3.17. A(α)-stability: A method is A(α)-stable if the stability region contains the wedge with angle α to the negative real axis in the left half plane.

To illustrate this property we show in Figure 6.7 where the angle α is indicated. It is a consequence of the definitions that the property A-stability is equivalent to A(π)-stability.

In certain connections, for example when Differential Algebraic equations are solved,

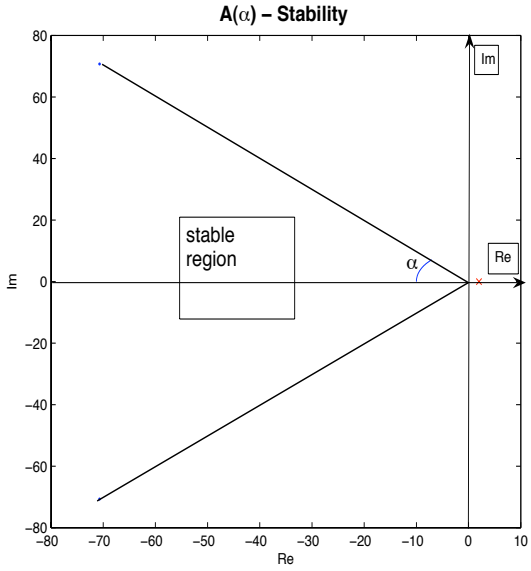


Figure 6.7: A(α)-stability region.

we make use of the following stability property.

Definition 6.3.18. L-stability: A method is L-stable if the stability region contains the entire left half plane, and the stability function is such that

$$| \mathbf{R}(\hat{h}) | \to 0 \quad \text{when} \quad | \hat{h} | \to \infty.$$

6.4 Stability Matrix

The dynamics of a differential equation system can be characterized by examining the local properties of the solution [44]. This can be done by considering a local linearization as described in the following.

To examine the linear stability of a given scalar problem

$$\dot{x} = f(x) \tag{6.16}$$

we can linearize about some fix-point x and then investigate the behavior of a perturbation δx to the equations. For this scalar problem the linearization is found using

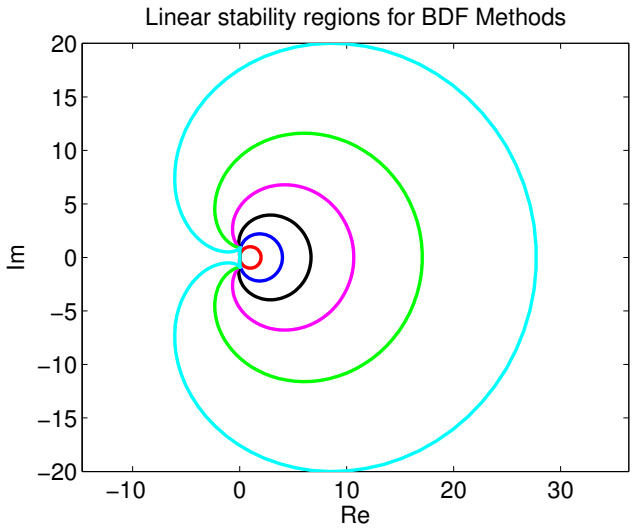


Figure 6.8: Stability regions for BDF methods of order 1 to 6.

a Taylor expansion of $f(x)$ as

$$f(x+\delta x) = f(x) + \delta x f'(x) + \delta x^2 f''(x) + \cdots \tag{6.17}$$

Assume that the solution of (6.16) is subject to a small perturbation δx and insert the Taylor expansion for $f(x)$ in the scalar problem

$$\dot{x} + \delta\dot{x} = f(x) + \delta x f'(x) + \delta x^2 f''(x) + \cdots \tag{6.18}$$

from which we find the evolution equation for the small perturbation

$$\delta\dot{x} = f'(x)\delta x + \cdots \tag{6.19}$$

If we assume that the perturbations are sufficiently small, we can neglect terms of order $\mathcal{O}(\delta x^2)$ and stability can be examined by considering the locally linear equation

$$\delta\dot{x} = f'(x)\delta x = \lambda\delta x \tag{6.20}$$

Thus, we can infer that the local stability will be governed by the derivative $f'(x)$.

As an example, let us consider the initial value problem stated as

$$\dot{x} = \lambda(x(t) - \cos(t)) - \sin(t), \quad x(0) = 1 \tag{6.21}$$

where λ is assumed to be real-valued.

By a local linearization, we find that small perturbations should satisfy the equation

$$(\delta x)' = \lambda\delta x \tag{6.22}$$

Thus, for an 4-step Adams-Bashforth (AB4) method it is possible to show that for absolute stability we need to satisfy the condition

$$-0.3 \leq h\lambda \leq 0$$

Since the step size h is usually selected to be positive (forward integration) the method can only be absolutely stable for some value $\lambda \leq 0$. For example, if $\lambda = -0.1$ the AB4 will only be stable for step sizes in the range $-3 \leq h \leq 0$.

In the same manner, we can examine the linear stability of a linear system

$$\begin{aligned} \dot{x} &= f(x, y) \\ \dot{y} &= g(x, y) \end{aligned} \tag{6.23}$$

We linearize the system about some fix point (x, y) using the Taylor expressions

$$f(x + \delta x, y + \delta y) = f(x, y) + f_x(x, y)\delta x + f_y(x, y)\delta y + f_{xy}\delta x\delta y + \cdots \tag{6.24}$$

$$g(x + \delta x, y + \delta y) = g(x, y) + g_x(x, y)\delta x + g_y(x, y)\delta y + g_{xy}\delta x\delta y + \cdots \tag{6.25}$$

Insert into linear system and we find

$$\begin{aligned} \dot{x} + \delta\dot{x} &= f_x(x, y)\delta x + f_y(x, y)\delta y + f_{xy}\delta x\delta y + \cdots \\ \dot{y} + \delta\dot{y} &= g_x(x, y)\delta x + g_y(x, y)\delta y + g_{xy}\delta x\delta y + \cdots \end{aligned} \tag{6.26}$$

from we we can deduce the linear evolution equation for the perturbations

$$\frac{d}{dt}\begin{bmatrix} \delta x \\ \delta y \end{bmatrix} = \begin{bmatrix} f_x(x, y) & f_y(x, y) \\ g_x(x, y) & g_y(x, y) \end{bmatrix}\begin{bmatrix} \delta x \\ \delta y \end{bmatrix} \tag{6.27}$$

where the 2×2 matris is called the *linear stability matrix* or *Jacobian matrix* of the linear system (6.23). If the Jacobian Matrix is diagonalizable it can be written in the form

$$J = V^{-1}\Lambda V \tag{6.28}$$

where V contains the linearly independent eigenvectors in the columns and Λ is a diagonal matrix holding the corresponding set of eigenvalues. Using this matrix decomposition, it is possible to transform the coupled equation system (6.27) into a decoupled equation system of the form

$$\frac{d}{dt}\mathbf{q} = \Lambda\mathbf{q}, \quad \mathbf{q} = V\begin{bmatrix} \delta x \\ \delta y \end{bmatrix} \tag{6.29}$$

By examining the eigenvalues of the Jacobian matrix (also referred to as the linear stability matrix) J for a system we can obtain information about the linear stability of linear systems. Furthermore, the analysis has a local nature and for both linear and nonlinear problems the linear stability matrix is usually not constant as the solution changes. The latter implies that the stability properties may change over time. A known system with this property is the Van Der Pol Equation with μ sufficiently large.

Chapter 7

Control of Error and Convergence

Development of scientific software packages has been carried out over the last couple of decades and several so called ODE-solvers have been published in the domain of open software. The database included in the NETLIB environment has very well reputed codes. We refer to such packages as DASSL, LSODE and the Matlab ODESUITE for more information, see the website *www.netlib.org*.

Here we give an introduction to the basic techniques and algorithms behind the more successful examples. In particular we will refer to the work carried out in the 'GODESS' [19] research program for results and analysis of these techniques. An extensive testing of existing software and the use of modern control theory has led to major improvements over previous techniques for step-size and error control [39].

As always the careful analysis combined with inspired use of new theory is the basis for improvement and we shall here try to give a distilled outline of the development starting with some experimental data for different applications. A key example in the process is the test example of Van der Pol presented in detail in the applications section 9.2. This problem is covering many of the important aspects of a typical system of ODEs involving a parameter that may make the problem change from non-stiff to stiff even very stiff, it is non-linear autonomous and the eigenvalues of the Jacobian matrix varies along the solution. This problem has been used extensively in the literature for testing purposes.

We start by considering two different methods for solving the problem one is an explicit Runge-Kutta method of order four and the other is a three stage ESDIRK method of order three. The problem we use for testing is the Van der Pol equation with two values of the parameter μ.

$$y'' = \mu(1 - y^2)y' - 1 \tag{7.1}$$

The solution starts with a transient and continues oscillating periodically. The final phase plane solution is a closed curve in the plane (y, y') and is a limit cycle. The shape of the limit cycle depends on the parameter μ as illustrated in figure 7.1.

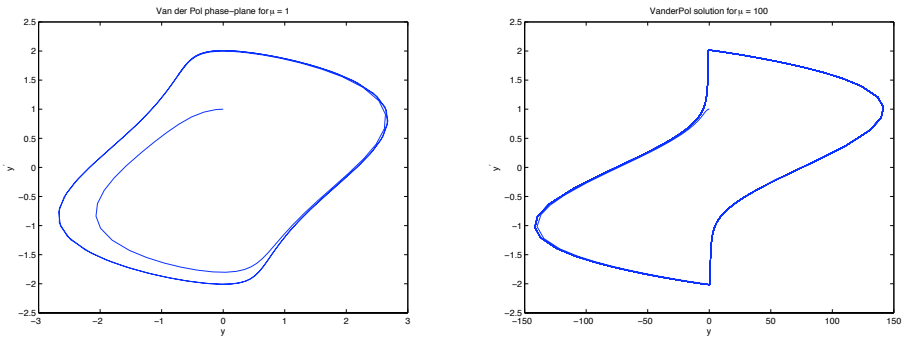


Figure 7.1: Phase plane solutions to the Van der Pol solution for a) $\mu = 1$ and b) $\mu = 100$

For $\mu = 1$ the solution is drawn as a function of time together with the sequence of step sizes used as determined by the control of of the estimated local error from (5.4). A fourth-order Runge-Kutta method was used to solve the ODE in time.

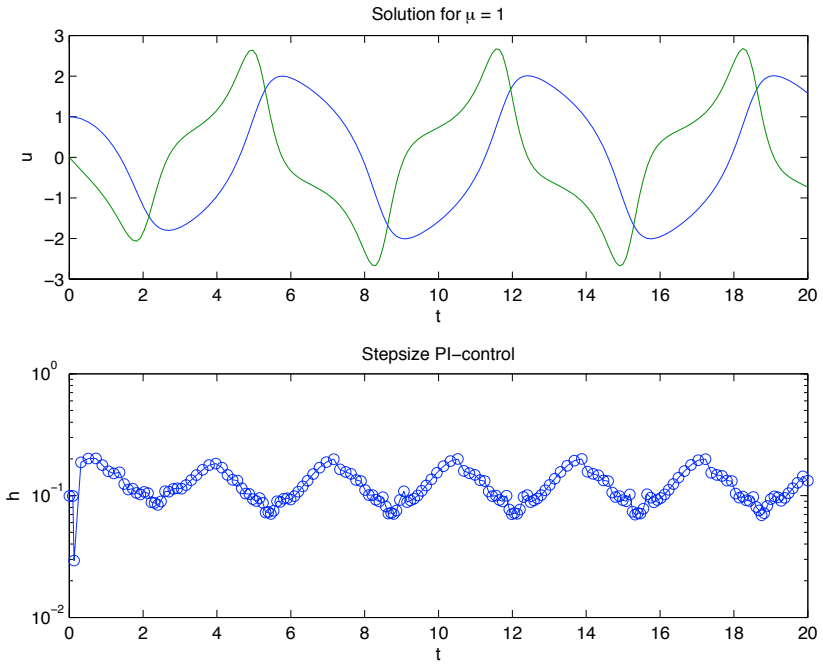


Figure 7.2: Development of solution in time and step sizes for Van der Pol solution using asymptotic control for $\mu = 1$.

7.1 Asymptotic control

We repeat the development for the traditional step size control, later referred to as *asymptotic control*. In terms of control theory this is equivalent to *Proportional control*. Assume that we have a method where the LTE (Local Truncation Error) can be estimated and the order q is known. The estimate can be written

$$LTE \approx \Psi(y_n)h^q$$

and is only strictly valid in the asymptotic limit where $h \to 0$.

Let us further assume that h is the step size used to take one step and for computing the estimate of the LTE. We want to find the step size $\hat{h}$ that should have been used to give a LTE equal to a given tolerance τ. For this step size it must be required that

$$\tau = \Psi(y_n)\hat{h}^q$$

If we take the ratio between these two equations we obtain

$$\frac{\tau}{LTE} \cong \frac{\Psi(y_n)\hat{h}^q}{\Psi(y_n)h^q} = \frac{\hat{h}^q}{h^q}$$

This leads to an equation for the estimated optimal step size for obtaining the prescribed local accuracy.

$$\hat{h} = h\sqrt[q]{\frac{\tau}{LTE}} \tag{7.2}$$

If we use this step-size to compute the next step and the solution is smooth in the neighborhood we should be able to satisfy the tolerance requirement on this step. However we will expect that it may fail quite often also the strategy is like balancing on a knifes edge. It is better to be a little conservative and take a smaller step f.ex. by multiplying by a factor α smaller that one. This leads to the strategy

$$\hat{h} = \alpha h\sqrt[q]{\frac{\tau}{LTE}} \tag{7.3}$$

This modification using for example $\alpha = 0.8$ is used in many codes such as the Matlab odeXX series of ODE solvers. Some further modifications are commonly used in practical codes, a minimum step-size is required and sometimes a maximum . These are such practical issues that improve the robustness of the implementation.

7.2 PI-control of step size

In order to optimize the step size-control strategy a Matlab implementation of the method has been implemented under the name *gerk.m*. As a special feature the gerk

code has the possibility to choose from a number of step size-control strategies. These have all been forged in the same template that is derived from the PI-controllers that have been developed by Søderlind [19]. The basic formula for estimating the step size for the current step from data gathered in previous steps is the following.

$$h_{n+1} = h_n \left(\frac{\tau}{e_n}\right)^{\beta_1} \left(\frac{\tau}{e_{n-1}}\right)^{\beta_2} \left(\frac{h_n}{h_{n-1}}\right)^{-\alpha_2} \tag{7.4}$$

For the purpose of relating this control to the traditional step-control strategy and others from the literature we give a couple of examples on typical choices of parameters. For further discussion on the properties of the PI-control strategies we refer

Type of control	α_2	β_1	β_2
Asymptotic step control	0	1/3	0
Watts step control	0	1/3	1/3
Choice used by Gustavsson	1	1/3	1/3
Second order PI	1/2	1/6	1/6

Table 7.1: Strategies for control

to [39]. Results from testing the four strategies is found in the example below.

By observation directly the variation in step size used by the two strategies is very similar. The secret is in the number of rejected steps as shown in the table.

Van der Pol test		
	$\mu = 1$	$\mu = 100$
	steps / failed steps	steps / failed steps
Asymptotic	550 / 105	9965 / 2510
PI-control	545 / 25	9343 / 1529

Table 7.2: Efficiency of step control strategies.

The efficiency of the control strategy is seen for the Van der Pol problem in the table where the total number of steps and the number of failed steps are given for two values of μ and the two step size strategies. The gain in efficiency is obtained by the smaller number of failed steps for the PI-controller and overall this reduces the total number of steps by 20 % and this does not cost anything extra per step.

The following experiment is showing how the relation between the error tolerance and the total work are related. The example is again the Van der Pol equation and the figure below shows the total number of steps versus the tolerance over a range of seven orders of magnitude. It is seen that the log-log plot gives a straight line with good approximation and that is a very satisfactory behavior of the code. In other words we get what we pay for in the way of accuracy versus work.

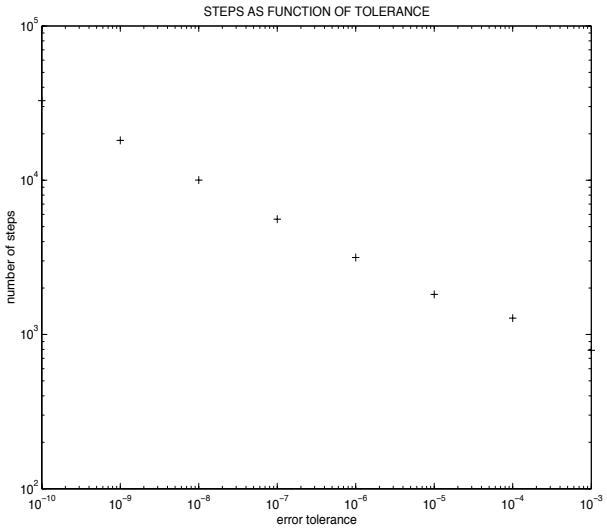


Figure 7.3: Steps versus tolerance test, log-log plot, $\mu = 200$

7.3 Measuring local errors

As shown in section 5.4 it is possible to estimate the local error at the new step for a one-step method using an expression of the form

$$e_{n+1} = x_{n+1} - \hat{x}_{n+1} = h \sum_{j=1}^{s} d_j f(t_j, x_j) \tag{7.5}$$

The local error estimation can be used for controlling the local errors in an adaptive time step algorithm. Generally, we want to impose a condition on either or both the absolute and relative local error, e.g.

$$|e_i| \leq abs_i, \quad \frac{|e_i|}{|x_i|} \leq rel_i$$

which can be rewritten into

$$\frac{|e_i|}{abs_i} \leq 1, \quad \frac{|e_i|}{rel_i|x_i|} \leq 1$$

Using the latter expressions for either the absolute or relative error committed in one step, we can collect the two into a single expression of the form

$$||e||_\infty = \frac{|e_i|}{abs_i + rel_i|x_i|} \leq 1 \tag{7.6}$$

that can be used in the control algorithm. Furthermore, we can use a 2-norm measure of the form

$$||e||_2 = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\frac{|e_i|}{abs_i + rel_i|x_i|}\right)^2} \tag{7.7}$$

The 2-norm can be expensive to compute, but can often produce more reliable results since all contributions over the set of components is averaged. In special cases, the infinity norm is chosen, but this will depend on the actual application.

7.4 Implementation of one-step methods

The practical use of any solution method will depend on the implementation into a program using some available programming environment. Here we will describe how the implementation in the form of a MATLAB-function can be made to form an efficient tool. This implementation makes full use of the vector notation that is behind the basic variables in MATLAB.

Special versions can be made extending the code to cope with systems where state changes takes place or to be used in computing Poincare sections.

MATLAB- code:

We suggest that you make your Matlab solver as a function with the following header. This makes room for most thinkable situations where a system of ODEs is solved over an interval. (*This is used as a reference implementation, not necessarily the most efficient in existence).*

```
function  [tout,yout] = rksolver( func, jac, t0, t1, y, aeps, reps, parm)
% func : name of the function for computing the right hand side.
% jac  : name of the function for computing the Jacobian matrix.
% t0   : start time.
% t1   : end time.
% y    : solution vector
% aeps : absolute error tolerance.
% reps : relative error tolerance.
% parm : a parameter that may beused in func and jac.
```

The functions:
The two functions func and jac may have the following headers:

function `dy = func( t, y, parm)`

```
% dy :  output value, this is the right hand side .
% t :  actual time.
% y :  actual solution vector.
% parm:  the parameter value.
% Here follows the statements that define the derivative.
```

function `jy = jac( t, y, parm)`

```
%jy :  output value, this is the jacobian matrix .
% t :  actual time.
% y :  actual solution vector.
% parm:  the parameter value.
% Here follows the statements defining the matrix elements.
```

The stages
Stagewise the Matlab code will be based on iteration, either direct successive fix-point iterations or using a Newton-Raphson process. First we look at the fix-point version.

```
% iteration in the inner stages
 for j = 2:nstage
  nres = 1;
  Y = y+h*fy*A(j,:)';
  ri=0;
  while nres > keps  & ri< minit
      nresg = nres;
      fy(:,j) = feval(func, t+C(j)*h,Y, param);
              % compute the stage - residual
              R = Y - (y+h*fy*A(j,:)');
              del=-M\R;
              Y= Y+del;
      ri = ri+1;
      fy(:,j) = feval(func, t+C(j)*h, Y,param);
      %dis = norm(del);  % displacement norm as an alternative possibility
      nres= norm(R);     % residual norm the normal choice
      crate= nres/nresg;
      if crate >1 & ri>1
          disp(sprintf('stepsize %13.6f at %16.12f diverged after %i ...
          iterations \nat iteration nr.%i rate= %13.6f \n',h,t,ri,j,crate));
      end
  end    %while
 end     %for j
```

Error control:
The error control is based on an estimate of the local truncation error. Here the 2-norm is used and there is a choice between error-per-step or error-per-unit-step is preferred. Error-per-unit-step is preferred for non-stiff problems while error-per-step is preferred for stiff systems.

```
   % Estimate the error
   if epus,
     % error per unit step
     r = norm(fy*Berr'./(afact+rfact.*abs(y)),2)/sqrtnst;
   else
     % error per step
     r = norm(h*fy*Berr'./(afact+rfact.*abs(y)),2)/sqrtnst;
   end
   % Collect statistics
   if nargout > 2
     sind = sind + 1;
     stat(sind,:) = [t log10(h) log10(r) r<=1];
     if rem(sind,100) == 0
       stat = [stat; zeros(100,4)];
     end
   end
     oldr = r;
   if r <= 1
% Error acceptable, update solution and calculate new stepsize
```

The update:
After an accepted step the solution is updated. If the error is larger than the tolerance the stepsize is reduced and the step is repeated. If the stepsize is too small a message is printed.

```
while t < tfinal
   if abs(h) < hsmall,
     disp(sprintf('Using a stepsize less than 1E-8*(tfinal-t0) at t = %g',t));
   end
   if t + h > tfinal,
      h = tfinal - t;
      return
   end
   % Compute the Jacobian matrix
   JA = feval(jac, t, y, param);
   % compute the iteration matrix
   M =  imi-h*gam*JA ;
   % Compute the slopes
   if (t == t0) | ~reuse
     % Can't reuse last stage
     fy(:,1) = feval(func,t,y,param);
   else
     % After a successful step the last stage equals the new point,
     % and after a rejected step we are still at the same point
     if ~prevrej
       fy(:,1) = fy(:,nstage);
     end
   end
```

The control:
The next stepsize is found using a strategy based on a PI-controller.

```
stepch = 'true';
% New stepsize  using PI-control. ki and kp are needed  (default 1/p).
s1 =  sat((0.95/r)^ki,100,0.01);
s2 =  sat((oldr/r)^kp,100,0.01);
%sh= (h/hgl)^(-ks)       may be added ;
sh = 1.0;    % a default choice
hstate = min([hmax h*incmax hstate*s1*s2*sh*tolfact]);
stepch = hstate > h*1.05;
if hstate == 0
   hstate = hsmall;
end;
if stepch
    hgl = h;
    h = hstate;
% if the previous step was rejected
if prevrej
    h=hgl;
end
```

Starting stepsize:
A starting stepsize must be provided or we may choose to let the code generate one automatically. This algorithm is based on gathering information about the solution at the initial point and estimating the error for a test step using Euler's method.

```
% Compute a "geometric average" of the tolerance vector. Bias the
% value slightly to the smallest tolerance values.
   tolexp = log10(etol);
   tolmin = min(tolexp);
   tolsum = sum(tolexp);
   tol = 10^(0.5*(tolsum/n+tolmin));
% Step a. Evaluate function at initial point
   f = feval(func,t0,y0,p);
% Step b and c. Determine stepsize based on initial point
   den = ( 1.0/max([abs(t0) abs(tfinal)]) )^order + norm(f,2)^order;
   h = ( tol/den )^(1/order);
% Step d. Do one Euler step using the calculated stepsize.
   f = feval(func,t0+h,y0+f*(h*sign(tfinal-t0)),p);
% Repeat step b and c, and choose the smallest stepsize
   den = ( 1.0/max([abs(t0) abs(tfinal)]) )^order + norm(f,2)^order;
   h = min([h ( tol/den )^(1/order)]);
   h = h*sign(tfinal-t0);
```

Chapter 8

Special problems for ODE-solvers

Solving standard type ODE-systems is a subject covered very well in textbooks on Numerical Analysis, on the other hand applications often give rise to problems that are not only of standard type and we have reserved this chapter to three such types of problems, *stiff systems, systems with changes of state* and Poincaré sections.

8.1 Stiff systems

Many practical applications from engineering lead to what has been known as *stiff systems* the origin for the name comes from mechanical engineering where such systems occur frequently. The nature of the problem can be explained from the effect of the stability properties of the method on the choice of stepsize. *If for a given problem the stepsize has to be selected based on stability requirements rather than accuracy then we call the system* ***stiff*** .

This rather loosely stated concept may be understood better by looking at a specific example. Let us consider the following.

The Robinson problem.
A source for generating interesting problems has been defined by looking at

$$y' = g'(t, y) - \lambda(y - g(t, y)), \quad y(t_0) = g(t_0, y_0) + \delta. \tag{8.1}$$

This equation will for $\delta = 0$ have the solution $g(t)$, this is what we call the smooth solution. If we solve this problem using one of our favorite explicit methods we will find solutions of the form shown below, depending on the value of λ the transient part will get steep for large values and less steep for smaller values. The steeper the transient the stiffer the problem so λ determines the stiffness of the problem.

If we try a solution with automatic stepsize-control using a explicit RK-4 (ode45)

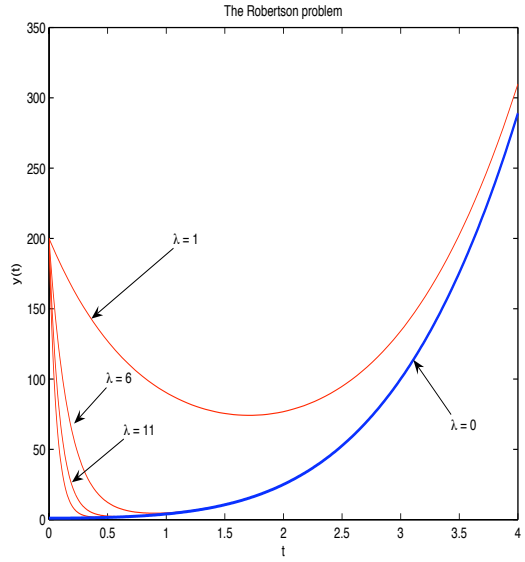


Figure 8.1: The Robinson problem for different values of λ.

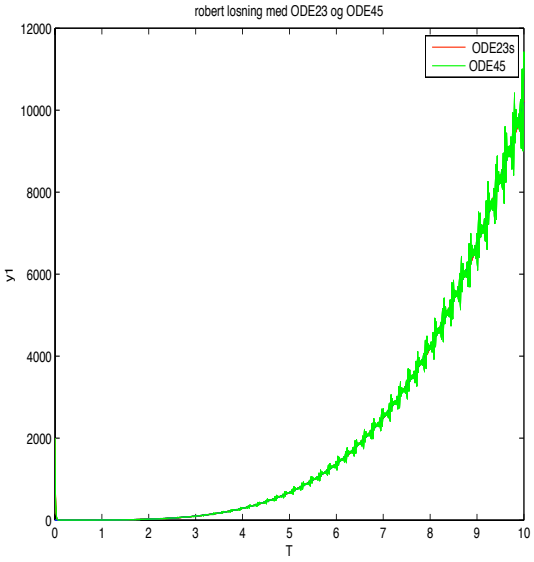


Figure 8.2: The Robinson problem, oscillations.

method we find the following solution picture

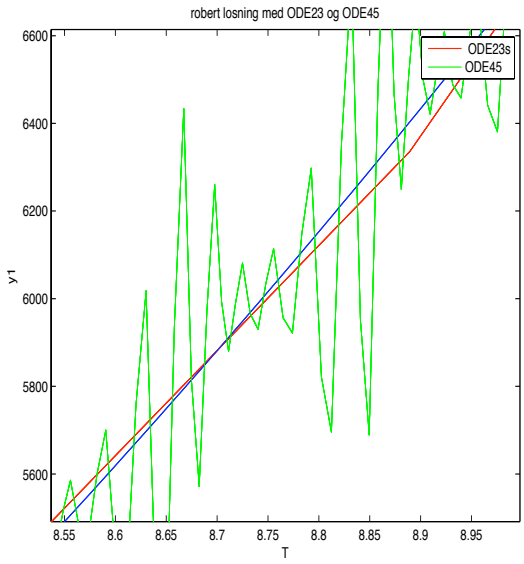


Figure 8.3: The Robinson problem, closeup.

We now compare two solutions using an A-stable method (ode23s) for comparison, we get the stepsize-history-plot

This indicates that the A-stable method gives the solution with a stepsize of the order of 10^{-1} while the explicit method uses very small steps to generate an erroneous solution. The reason for this behaviour is, that the explicit method used is trying to find stepsizes that will meet the accuracy requirement. This can be met and the step is tried, leading to an unstable step. The instability gives a large increase in the estimated error and the step fails, lleading to a reduction of the stepsize, the steps will oscillate around the smooth solution with an amplitude that is of magnitude the tolerance used for error control.

Opposed to this the A-stable solution has no problem satisfying the accuracy and stability is not putting any limitations on the choice of stepsize resulting in a very smooth stepsize sequence. This underlines the power of A-stable methods for the stiff type of problem.

The concept of absolute stability gives an understanding of why systems like the Robinson problem give rise to erroneous behavior of some methods. We can always choose very small stepsizes to get a stable solution but in most cases this approach is very inefficient.

Our interest is in developing methods that are suitable for solving stiff problems and

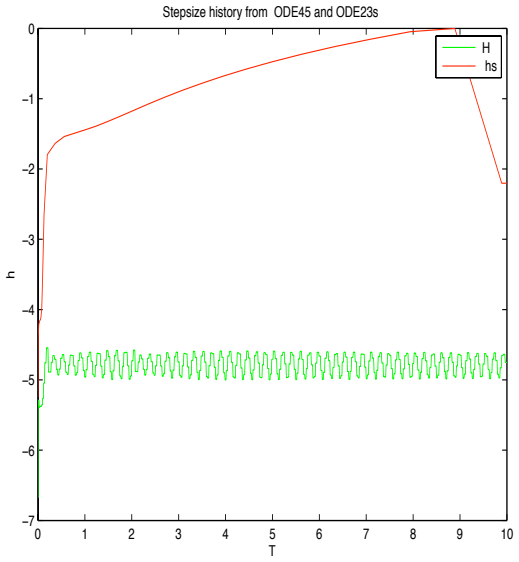


Figure 8.4: The Robinson problem, stepsize history.

the family of ESDIRK methods will be analysed in detail since they seem to fit our basic needs.

8.2 Semi-implicit Runge-Kutta methods

Among the one-step methods that were presented in the previous chapter we have experienced that explicit methods have severe stability restrictions that make them less suited for stiff problems. By generalizing the definition of these methods we obtain families of methods that have stability properties that make them better candidates for our stiff solvers.

As a simple example of a semi-implicit method we illustrate the ideas by using the following Butcher tableau of a two-stage SDIRK method. Here SDIRK means a Semi-Diagonally-Implicit-Runge-Kutta method, these methods have been introduced by Nørsett and Thomsen [42] in 1984.

γ	γ	
1	$1-\gamma$	γ
y_{n+1}	b_1	b_2
$\tilde{y}_{n+1}$	$\tilde{b}_1$	$\tilde{b}_2$

SDIRK - two stage, order 2.
First we look at the order conditions that will give order 3. These are derived directly from the Butcher tableau.

$$\begin{aligned} b_1 + b_2 &= 1 \\ b_1\gamma + b_2 &= \frac{1}{2} \\ b_1\gamma^2 + b_2 &= \frac{1}{3} \\ b_1\gamma^2 + b_2(1-\gamma)\gamma + b_2\gamma &= \frac{1}{6} \end{aligned}$$

From the number of free coefficients we see that only three conditions can be satisfied and the highest order is 2. Solving the first two equations for b_1 and b_2 we find.

$$b_1 = \frac{1}{2(1-\gamma)} \text{ and } b_2 = \frac{1-2\gamma}{2(1-\gamma)} \tag{8.2}$$

This gives a one parameter family of methods with γ as a free parameter. In order to find the stability properties of the methods we look at the test equation.

$$y' = \lambda y, \quad y(0) = y_0 \tag{8.3}$$

Using the method we get the following stages

$$\begin{aligned} k_1 &= \lambda(y_n + h\gamma k_1) \\ k_1 &= \frac{\lambda y_n}{1 - h\gamma\lambda} \\ k_2 &= \lambda(y_n + h((1-\gamma)k_1 + \gamma k_2) \\ k_2 &= \frac{\lambda y_n(1 + (1-2\gamma)h\lambda)}{(1-h\gamma\lambda)^2} \end{aligned} \tag{8.4}$$

From these equations we can find the solution y_{n+1} and derive the *Stability function*

$$y_{n+1} = R(h\lambda)y_n = (y_n + b_1hk_1 + b_2hk_2) = (1 + b_1h\frac{\lambda y_n}{1-h\gamma\lambda} + b_2h\frac{\lambda y_n(1+(1-2\gamma)h\lambda)}{(1-h\gamma\lambda)^2})y_n$$

This may be simplified by introducing $z = h\lambda$ and we get

$$R(z) = \frac{P(z)}{Q(z)} = \frac{1 + (1-\gamma)z + (\frac{1}{2} - \gamma^2)z^2}{(1-\gamma z)^2} \tag{8.5}$$

The stability properties are analyzed by looking for those regions in the complex plane where:

$$z \in \mathbb{C}, \text{where } \mid R(z) \mid < 1.$$

A very attractive method results if we require that the second order term in the numerator is zero. This will provide L-stability. The result is

$$\gamma = \frac{\sqrt{(2)}}{2}.$$

Exercise. Explain why we obtain L-stability in this case.

We can find the linear stability region for the method using the condition for the stability function being less than or equal to size unity. The relative stability region or *order star* is the set

$$\{h\lambda \in \mathbb{C} || R(h\lambda)| \leq |e^{h\lambda}|\} \tag{8.6}$$

The order star compares the growth of the iteration $y_{i+1} = R(h\lambda)y_i$ to the exact iteration found from the exact solution to the test equation determined as $y_{i+1} = e^{h\lambda}y_i$. The order star for the SDIRK-2 method with stability function (8.5) is illustrated in Figure 8.5. The dark blue areas are the regions where the condition (8.6) is fullfilled.

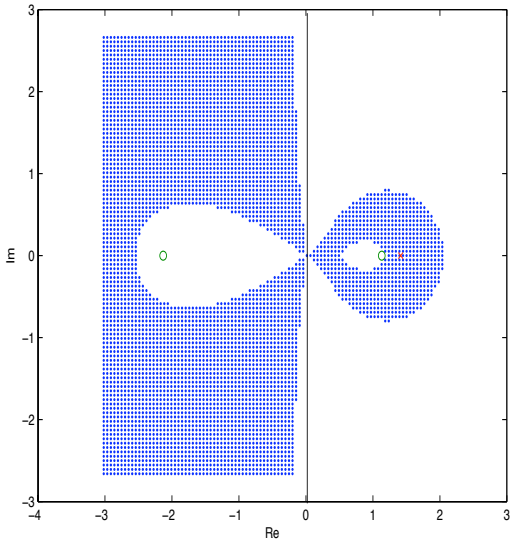


Figure 8.5: Order Star for SDIRK-2.

It can be verified that if the order star does not cross the imaginary axis, then the method is A-stable. An extensive description of the theory and interesting discussion on the use of *Order Stars* is found in [24, 40].

ESDIRK-methods

A very interesting case is obtained when the first stage is explicit. This idea was introduced by Butcher and Chen [6] and have been studied by Kværnø [28]. The first stage of the computation for one step is then the function evaluation at the present point (t_n, y_n) The second stage is the following function evaluation.

$$\xi_1 = y_n + h\gamma f(\xi_1) \tag{8.7}$$

This equation has to be solved for ξ_1. This is done by iteration. Here we can make a choice.

Fix-point iteration
This puts a limit to the stepsize of the form $hL\gamma < 1$. Good for non-stiff ODE's.

Newton iteration
No limits for h except that we need a good starting guess. We need to compute the Jacobian matrix.

The equation may be rearranged to give

$$\begin{aligned} \xi_1 - (y_n + h\gamma f(\xi_1)) &= G(\xi_1) \\ G(\xi_1) &= 0 \end{aligned}$$

Define the *residual* for a value ξ^s

$$r(\xi^s) = \xi^s - y_n - h\gamma f(\xi^s)$$

We look for a zero of this residual.

The Jacobian matrix
For the equation we get

$$\frac{\partial G}{\partial \xi} = \begin{pmatrix} \frac{\partial g_1}{\partial \xi_1} & \frac{\partial g_1}{\partial \xi_2} & \frac{\partial g_1}{\partial \xi_3} \\ \frac{\partial g_2}{\partial \xi_1} & \frac{\partial g_2}{\partial \xi_2} & \frac{\partial g_2}{\partial \xi_3} \\ \frac{\partial g_3}{\partial \xi_1} & \frac{\partial g_3}{\partial \xi_2} & \frac{\partial g_3}{\partial \xi_3} \end{pmatrix}$$

In matrix notation we get what is called
The Iteration Matrix

$$\frac{\partial G}{\partial \xi} = I - h\gamma \frac{\partial f}{\partial y} = I - h\gamma \mathbf{J} = \mathbf{M} \tag{8.8}$$

The Newton process
For finding the solution to our equation $G(\xi_1) = 0$ we have the scheme

$$\begin{aligned} \mathbf{M}\delta^s &= -r(\xi^s) \\ \xi^{s+1} &= \xi^s + \delta^s \end{aligned}$$

This is what is usually called *exact Newton* assuming that the iteration matrix is evaluated in every iteration, but this is not necessary for convergence.

Every evaluation means that we compute new elements in the Jacobian $\mathbf{J}$. An approximation is to evaluate only once for each step, since the elements will normally vary very slowly we can get vast improvements in efficiency doing this.

SDIRK- schemes

The special feature of the SDIRK schemes, is that they are composed by stages that share the form:

$$\xi - \phi_n - h\gamma f(\xi) = 0 \tag{8.9}$$

This means that the iteration matrix is the same in all stages and we can make a solution to the system in the following way:

$$\mathbf{M} = \mathbf{LU}, \quad \mathbf{LU}\delta = -r(\xi)$$

Where $\mathbf{L}$ is lower triangular and $\mathbf{U}$ is upper triangular. The solution can then be determined in two steps as

$$\mathbf{L}z = -r, \quad \mathbf{U}\delta = z$$

The update after every iteration should give a contribution that in size is making enough difference to the solution or else the iteration process should be stopped. This gives rise to a *Stopping criterion* based on the relative norm of the form

$$\|\delta\| \leq \|\xi\|\epsilon$$

Here ϵ is a tolerance that must be chosen orders of magnitude smaller than the error tolerance. A number of alternatives are available with respect to he stopping criterion. The above will work in most cases but in [22] it has been shown that a more robust choice is by testing the norm of the residual.

$$\|\mathbf{r}\| \leq \|\xi\|\epsilon$$

With respect to implementation both choices can be used but the residual vector is in the same vector space as the solution and this is to be preferred.

evaluate	$\mathbf{M}$	and factorize it	$\mathbf{LU} = \mathbf{M}$
find initial guess	ξ^0	for the stage	
evaluate function	$f(\xi)$		(*)
evaluate the residual	$r(\xi)$		
solve the system	$\mathbf{LU}\delta = -\mathbf{r}(\xi)$		
update the unknowns	$\xi = \xi + \delta$		
is the	$\|\mathbf{r}\| < \epsilon\|\xi\|$		
if not continue from			(*)
else goto next stage.			

Table 8.1: Overall the solution is as follows.

A four-stage ESDIRK Method

The Butcher tableau of the four-stage ESDIRK method with error estimate e_n presented here is

$$\begin{array}{c|cccc} 0 & 0 & & & \\ c_2 & a_{21} & \gamma & & \\ c_3 & a_{31} & a_{32} & \gamma & \\ 1 & b_1 & b_2 & b_3 & \gamma \\ \hline \mathbf{y}_{n+1} & b_1 & b_2 & b_3 & \gamma \\ \hline \mathbf{e}_{n+1} & d_1 & d_2 & d_3 & d_4 \end{array} \tag{8.10}$$

In order to find coefficients that lead to good properties of the method we will make use of the following approach, first setup the order conditions for orders $1, 2, 3, 4$ expressed with the coefficients from (8.10) , next solve the set of equations for our stability test problem and find the stability function . Here we may attempt to find conditions for A- and L-stability from which we can find the γ values that are optimal for our purpose. The last step in the derivation is to solve the set of nonlinear equations from the order conditions for the unknown coefficients and we have a method. After reduction These equations reduce to

$$\begin{aligned} \tilde{b}_1 + \tilde{b}_2 + \tilde{b}_3 + \tilde{b}_4 &= 1 \\ \tilde{b}_2 c_2 + \tilde{b}_3 c_3 + \tilde{b}_4 &= \frac{1}{2} \\ \tilde{b}_2 c_2^2 + \tilde{b}_3 c_3^2 + \tilde{b}_4 &= \frac{1}{3} \\ \tilde{b}_2 c_2^3 + \tilde{b}_3 c_3^3 + \tilde{b}_4 &= \frac{1}{4} \\ \left(\tilde{b}_2\gamma + \tilde{b}_3 a_{32} + \tilde{b}_4 b_2\right) c_2 & \\ + \left(\tilde{b}_3\gamma + \tilde{b}_4 b_3\right) c_3 + \tilde{b}_4\gamma &= \frac{1}{6} \\ \left(\tilde{b}_2 c_2\gamma + \tilde{b}_3 c_3 a_{32} + \tilde{b}_4 b_2\right) c_2 & \\ + \left(\tilde{b}_3 c_3\gamma + \tilde{b}_4 b_3\right) c_3 + \tilde{b}_4\gamma &= \frac{1}{8} \\ \left(\tilde{b}_2\gamma + \tilde{b}_3 a_{32} + \tilde{b}_4 b_2\right) c_2^2 & \\ + \left(\tilde{b}_3\gamma + \tilde{b}_4 b_3\right) c_3^2 + \tilde{b}_4\gamma &= \frac{1}{12} \\ \left(\tilde{b}_2\gamma^2 + 2\tilde{b}_3 a_{32}\gamma + \tilde{b}_4(2b_2\gamma + b_3 a_{32})\right) c_2 & \\ + \left(\tilde{b}_3\gamma^2 + 2\tilde{b}_4 b_3\gamma\right) c_3 + \tilde{b}_4\gamma^2 &= \frac{1}{24} \end{aligned} \tag{8.11}$$

To assess the accuracy of the basic integrator, an estimate of the local error is needed. We use the strategy of embedded formulas asking the method with quadrature weights $\tilde{\mathbf{b}}$ to be of fourth order. The error estimate is obtained as the difference between the third and fourth order approximations to $\mathbf{x_{n+1}}$. Fourth order of the

embedded method requires

$$\begin{aligned}
\mathbf{b}^T\mathbf{C}^{k-1}\mathbf{e} &= \frac{1}{k}, \quad k = 1, 2, 3, 4 \\
\mathbf{b}^T\mathbf{ACe} &= \frac{1}{6} \\
\mathbf{b}^T\mathbf{CACe} &= \frac{1}{8} \\
\mathbf{b}^T\mathbf{AC}^2\mathbf{e} &= \frac{1}{12} \\
\mathbf{b}^T\mathbf{A}^2\mathbf{Ce} &= \frac{1}{24}
\end{aligned} \tag{8.12}$$

We have now established the necessary conditions to find the coefficients for defining the method. These equations together with the requirement for L-stability results in the unique value of $\gamma = 0.435866521508$ and the following table of method coefficients. A proof of uniqueness is found in ([1]). A discussion on a similar method of

γ	0.435866521508
c_2	0.871733043017
c_3	0.468238744852
a_{21}	0.435866521508
a_{31}	0.140737774725
a_{32}	-0.108365551381
b_1	0.102399400620
b_2	-0.376878452256
b_3	0.838612530127
d_1	0.054625497240
d_2	0.494208893626
d_3	-0.221934499735
d_4	-0.326899891131

Table 8.2: Coefficients for ESDIRK34.

order 3 with three stages is found in [45].

8.3 Systems with changes of state

In many applications of numerical simulation the systems may change state. Such cases are found in the simulation of multibody dynamics and control systems, for example when a thermostat makes some part of the system cut in and off. This means in the mathematical model that the equations for the dynamic system are changing in a discontinuous way across a solution point [2].

The direct application of a numerical method for the solution of such a system will lead to unwanted growth of errors as well as a wasted extra computational effort. All in all this is an unwanted situation.

By applying modern continuous extensions in combination with the solution method we may derive a strategy for passing the discontinuity points without loss of accuracy and at a very minimal extra cost in computational effort.

Discontinuous right hand sides

The dynamic system we will consider for illustration can be defined the following way

$$y' = f(t, y) \ , \ t \in [a, b] \ , \ y(a) = \eta \tag{8.13}$$

where the function $f(t, y)$ is given by

$$f(t,y) = \begin{cases} f_1(t,y) & \text{for} \quad \phi(t,y) < 0 \\ f_2(t,y) & \text{for} \quad \phi(t,y) \geq 0 \end{cases} \tag{8.14}$$

The functions $f_1(t, y)$ and $f_2(t, y)$ need not have the same value at the point where the solution crosses the curve $\phi(t, y) = 0$. This means that the solution will have a discontinuous derivative across this curve (see [31]). We illustrate the situation in the Figure 8.6.

The existence and continuity of the solution is guaranteed under very modest assumptions for the differential system, we refer to [31] for the details. If we assume that $\phi(t, y)$ is analytic in t and y we will obtain that the curve $\phi(t, y)$ is differentiable with respect to both t and y and the solution to

$$y_1'(t) = f_1(t, y_1(t)) \ , \ y_1(a) = \eta \tag{8.15}$$

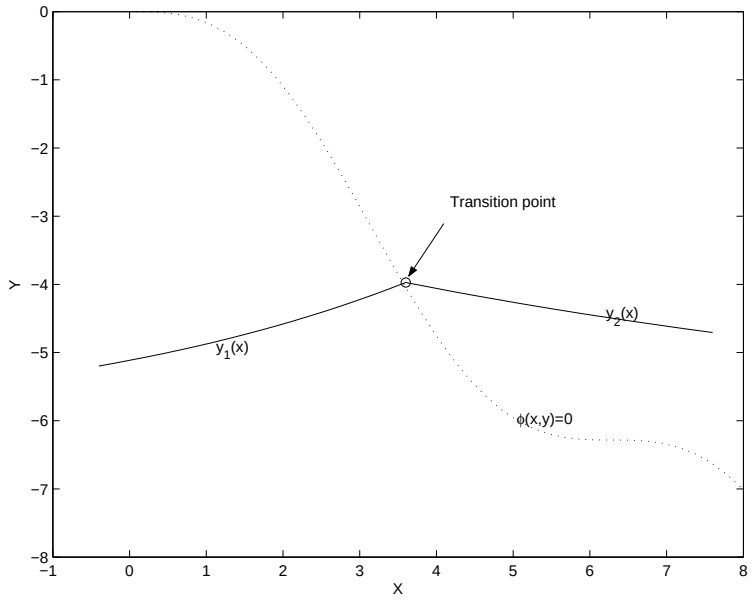


Figure 8.6: Discontinuity across curve of state change.

will cross the curve $\phi(t, y)$ at some point P defined by the condition

$$\phi(t_1, y_1(t_1)) = 0 \tag{8.16}$$

The differential equation (8.13) define a new initial value problem that may be rewritten as

$$y_2'(t) = f_2(t, y_2(t)) \ , \ y_2(t_1) = y_1(t_1). \tag{8.17}$$

The solution to (8.13) can now be found as the solution to (8.15) in combination with the solution to (8.17). The simple form of a discontinuous problem is found when we have a jump-discontinuity that satisfy the condition

$$\mid f_1(t, y) - f_2(t, y) \mid < C \tag{8.18}$$

We will consider in this report problems where this condition is assumed to be satisfied everywhere.

The numerical solution across a jump-discontinuity

Following the idea from [17] we consider the problem specified in the previous section

using either a one-step method like a Runge Kutta method or a multistep method. In the two domains specified by the regions where the function ϕ is either positive or negative the methods are solving IVP's in the usual manner and all we need to consider is the region close to the point where the solution crosses from one region to the other. The point P is called the transition point.

Multistep methods

When solving the system (8.13) using a multistep method we consider for simplicity a constant stepsize defined by

$$y_n \approx y(t_n)\ ,\ t_n = a + nh\ ,\ h = t_{n+1} - t_n\ ,\ n = 1, 2, \ldots, N. \tag{8.19}$$

We follow the treatise of multistep methods in [29] where the multistep method is defined as

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = \sum_{j=0}^{k} \beta_j y'_{n+j} \tag{8.20}$$

The accuracy of the formula is found by looking at the local truncation error given by the linear difference operator

$$\mathbf{L}[y(t_n); h] = \sum_{j=0}^{k} (\alpha_j y(t_n + jh) - h\beta_j y'(t_n + jh)) \tag{8.21}$$

The assumption that $y(t)$ has continuous derivatives of sufficiently high order leads to the result that for a method of order p we find that

$$\mathbf{L}[y(t_n); h] = C_p h^{p+1} y^{(p+1)} + O(h^{p+2}) \tag{8.22}$$

This is the traditional result that leads to convergence when $p \geq 1$. Now consider the situation shown in the figure below where the step is across a transition point. In this case we can derive the result for the truncation error by using Taylor expansions of the sum from (8.22) and we arrive to the result (after some derivation) using the conditions for order p that

$$\mathbf{L}[y(t_n); h] = h(1 - \delta - \beta_k)(y'(\xi^+) - y'(\xi^-)) + O(h^2) \tag{8.23}$$

According to the normal definition of order we conclude that in this case the order is $p = 0$ and the method is no longer convergent. We may simplify the expression for the local truncation error by using the bound from the jump-condition (8.18) and we obtain

$$\mid \mathbf{L}[y(t_n); h] \mid \leq h \mid (1 - \delta - \beta_k) \mid C \tag{8.24}$$

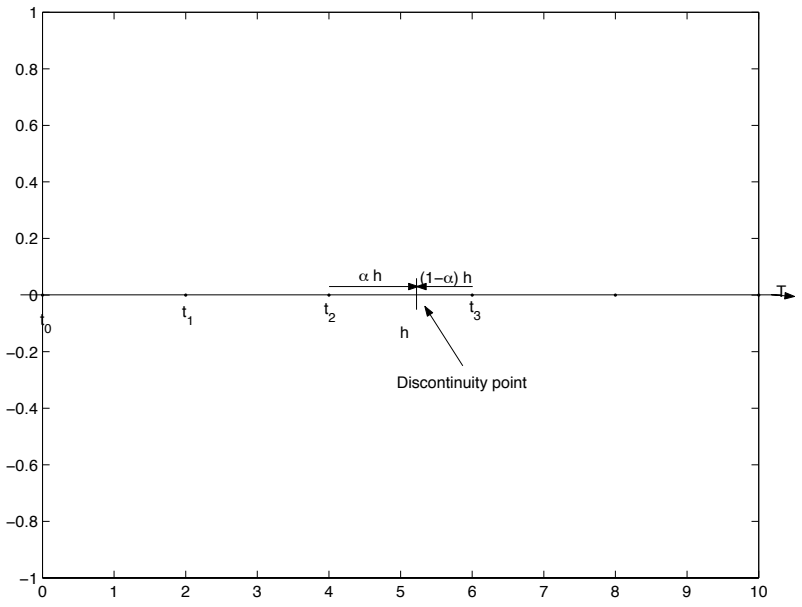


Figure 8.7: Integrating across a point of state change.

Basically we obtain that the local truncation error is proportional to h and to the size of the jump in first derivative across the discontinuity.

For discontinuities in higher order derivatives we may use the same type of derivation to obtain the result that if the jump is in the q'th derivative and bounded like (8.18) we find

$$\mathbf{L}[y(t_n); h] \approx h^q \hat{C} \tag{8.25}$$

We see that in cases where $q < p+1$ we may expect a decrease in the order observed. This result means that we will be able to predict the local behaviour of a given method across boundaries with discontinuities in derivatives of variable orders.

One-step methods

The general form of a onestep method is the following

$$y_{n+1} = y_n + h\Phi(t_n, y_n; h) \tag{8.26}$$

Again assuming smoothness of all derivatives up to the order $p+1$ will lead to a local truncation error of the form

$$T_{n+1} = \psi(t_n, y(t_n))h^{p+1} \tag{8.27}$$

The result of the analysis in the case with a discontinuity in the derivative will in this case lead to a similar result to the case with multistep methods and we find

$$T_{n+1} \approx \tilde{\psi}(t_n, \xi^+ - \xi^-)h \tag{8.28}$$

In the one-step case we find that the discontinuity may be in any of the mixed derivatives of the function $f(t, y)$ of orders lower than the order of the method. In principle though the two types of methods behave in a similar way. To get more details we refer to the reference [17].

Example of behavour of standard ODE solver when experiencing a slope discontinuity

As an illustration of the behaviour of a standard ODE solver, consider the computed results for the solution of the following ODE

$$y' = \begin{cases} y & for \quad 0 \leq t \leq 1 \\ -y & for \quad 1 < t \leq 2 \end{cases} \quad y(0) = 1 \ , \ t \in [0, 1] \tag{8.29}$$

The figure shows that the automatic stepsize control will cut down the stepsize to

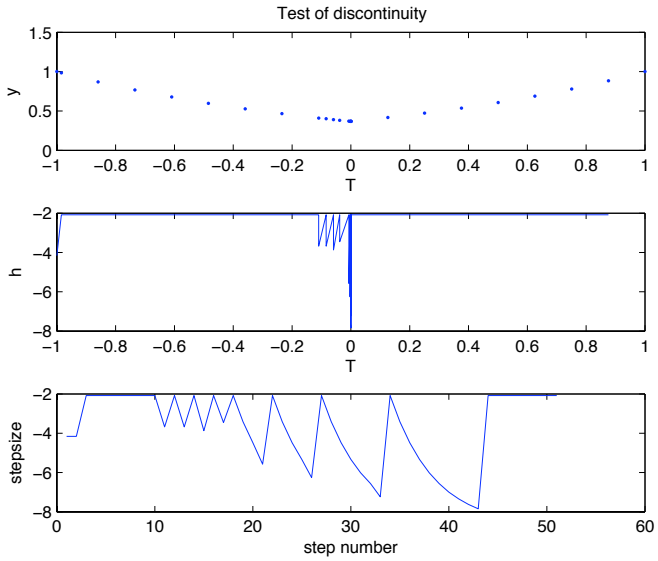


Figure 8.8: Solution and stepsize history with a discontinuity.

the smallest allowable value because the error estimator becomes unreliable due to the fact that the error behaves like order zero instead of order p. If we assume the correct order of the method the error will be estimated to the actual stepsize times

0	0	
1	$\frac{1}{2}$	$\frac{1}{2}$
y_{n+1}	$\frac{1}{2}$	$\frac{1}{2}$
$\tilde{y}_{n+1}(\theta)$	$\theta(1-\frac{\theta}{2})$	$\frac{\theta^2}{2}$

Table 8.3: Coefficients for the trapezoidal-method with continued extension.

the size of the jump. For the example this would mean that the step should be of the order of 10^{-4} compared to the value 10^{-6} observed from the result. The driver waste many unaccepted steps cutting down the stepsize before passing the transition point.

Continuous extension

A traditional method for the solution of ODE's is basically finding the approximate solution on a discrete set of points, the discretization is defined by the stepsize control. The transition points wil however not in general be at one of these points. In order to develop a method for passing the transition point we need to be able to find an approximate solution in a continuous way. The tool for doing that is the continuous extension, developed for Runge-Kutta methods ([42] and [36]) and the general interpolant for linear multistep methods ([49]).

The Trapezoidal method with continuous extension.

As a simple example of an implicit method we illustrate the ideas by using the Trapezoidal method, in this case we use the GERK-formulation by giving the Butcher tableau of the method. It is customary to use θ as the parameter for defining the interpolation point. This point will in connection with the discontinuity be defined by the position of the transition point. We wish to determine this point and the condition for this is the function $\phi(t, y)$ being zero.

$$\phi(t, y(\theta)) = 0 \ , \ 0 \leq \theta \leq 1. \tag{8.30}$$

This equation is a normal condition for a zero of the function with θ as the variable. Any convenient zero-finding method may be used for determining the solution, if $\phi(t, y)$ is a smooth function the most efficient method will be based on a Newton-Raphson method. This assumes that derivatives of the functions are available. In the following example we treat a system which passes a level, like in an application where a thermostat reaches a set-point. The system is the following.

$$y' = 1 - y \ , \ y(0) = 2 \ , \ y(t) = 1 + \exp(-t) \tag{8.31}$$

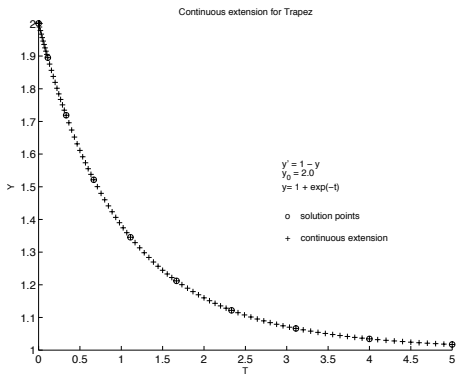


Figure 8.9: Solution and Continuous Extension.

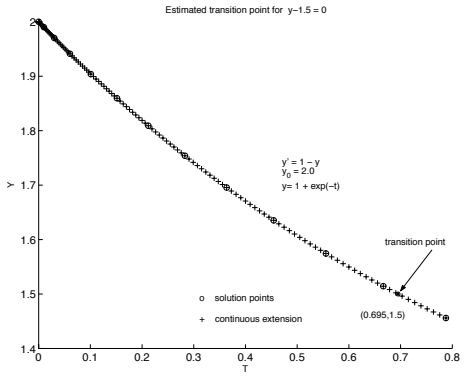


Figure 8.10: Solution and determination of a transition point.

The set-point is defined by the condition,

$$\phi(t, y) = y - 1.5 = 0. \tag{8.32}$$

The solution is shown in the figure and the transition point is marked. We have applied a constant stepsize to get to the setpoint and then the value θ is found from the equation that is derived from (8.16) leading to the equation.

$$\begin{aligned} y_n - 1.5 - h(\frac{\theta^2}{2}(f_{n+1} - f_n) + \theta f_n) = 0 \\ \theta = 0.084 \quad t = 0.6931. \end{aligned}$$

The stepsize strategy here is very different from the one leading to the results in figure (8.3) and no steps are wasted for the approach to the transition point. The solution may be restarted using the transition point as the initial value for a solution in the new state.

Implementations

The example in the previous section has shown that a quite general strategy may be applied to change from state to state if we apply the conditions (??) in connection with the continuous extension a discussion is found in [50].

This is straightforward in the scalar case with only one active condition as in the example. In the general case where we may have a system of ODE's and where change of state may happen between several states and guided by a number of conditions, the implementation must be done very carefully to give satisfactory performance.

In the DALI [27] a matrix of conditions are kept, rows representing the active states and columns containing the conditions for passing to another state. Thus $\phi_{i,j}(t, y)$

changing sign will mean that the system in state i will change to state j. Not all states are reachable from all other states and we define a state-transition matrix containing 'ones' where a change is possible and 'zeros' where there is no possible state transition. The next figure shows the situation for a system illustrated by a state diagram and the corresponding state transition matrix. The example is from a simulation of a glider in the starting process over a free flight to landing, the transition is one-way following the numbering of the states assuming that the landing leads back to the original state of start.

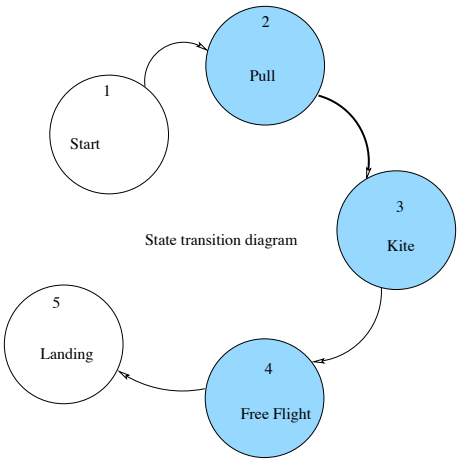


Figure 8.11: State transition diagram and matrix

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

We consider the simulation of a tank heated to a given temperature and controlled by thermostats to keep its temperature between given bounds. The system is shown in the figure below. The states of the system can be identified quite easily and the state

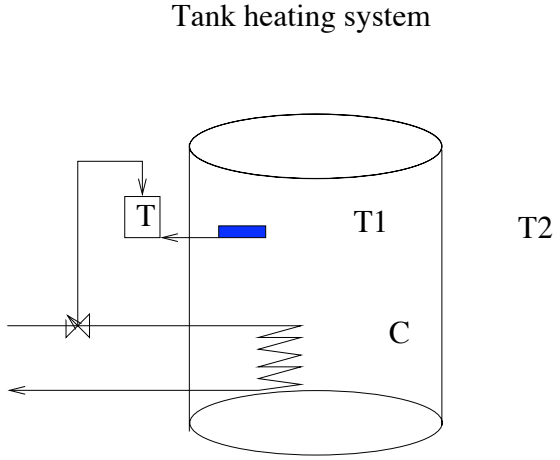


Figure 8.12: The tank with heater and thermostat.

diagram is shown in figure 8.13. When carrying out the simulation the model will change state using the continuous extension for determining the transitions between states and the solution will look like shown here. In the **Intersim** package for simulations a similar structure is applied but is derived from a language description of the system. The model is defined in a pseudo-programming language with a syntax that incorporates switch-conditions. From the model description the transition matrix is derived and the system refers to this to indirectly apply the transition functions in a Newton type iterative solution process.

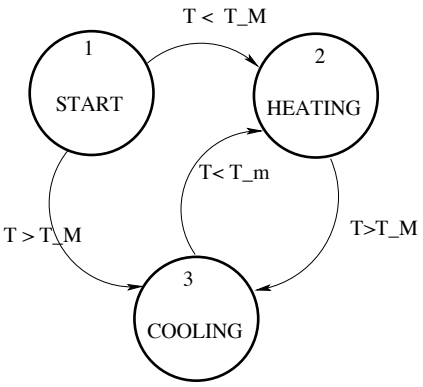


Figure 8.13: State diagram for the tank-heater.

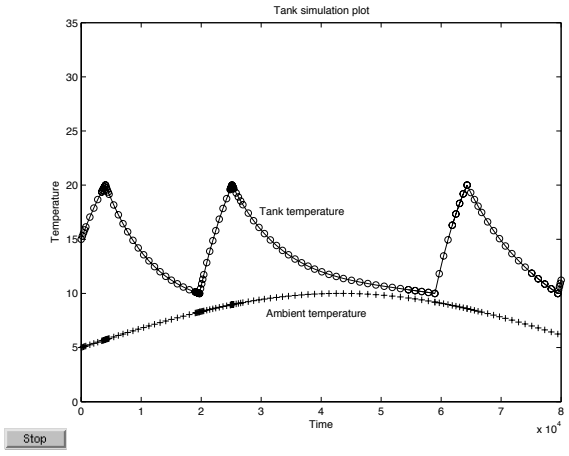


Figure 8.14: Simulation of the tank-heater.

8.4 The Poincaré section

Special attention can be given to problems where we look for periodic solutions. These appear often enough in technical applications so that it is worthwhile to treat them carefully. As an example we consider the following model equation called the *Lorenz equation*

$$\begin{aligned} y_1' &= -\sigma y_1 + \sigma y_2 \\ y_2' &= -y_1 y_3 + r y_1 - y_2 \\ y_3' &= y_1 y_2 - b y_3 \end{aligned} \tag{8.33}$$

This equation system has for the choice of parameters $\sigma =?$ the solution illustrated in figure 8.15. We could be interested in knowing more about the solution behaviour.

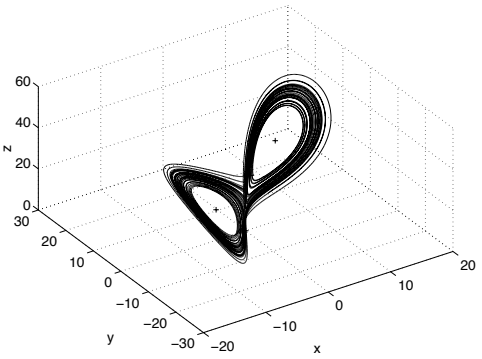


Figure 8.15: Lorenz solution.

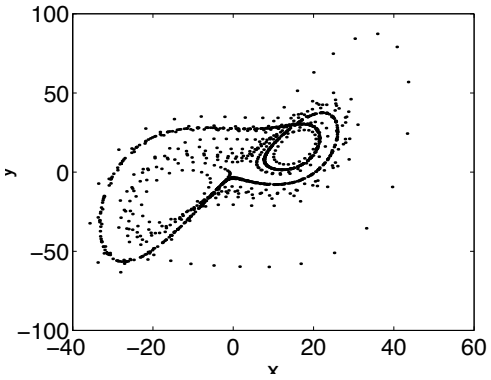


Figure 8.16: Lorenz solution.

For example, is the solution periodic? Does it have a special structure? We get different types of information from the code depending on what parameters we choose, what initial conditions we choose and so forth.

If we look for structure we should do something to prepare the investigation. Here the *Poincare sections* is a very good tool. The section is defined as the set of points where the solution crosses a plane or surface in space. If the solution is periodic it will pass the same intersection points repeatedly and this is an example of the kind of structure we are looking for. For the Lorenz system we can find the section where our solution intersects the xz-plane as illustrated in Figure 8.16.

Attractors

Another system is a forced oscillator *The Duffing equation.*

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= -0.15 y_2 + y_1 - y_1^3 - 0.3 cos(t) \end{aligned} \tag{8.34}$$

This system generates the Poincare-section called *Duffings forced oscillator* illustrated in Figure 8.17. In order to find points on the Poincare map we have to satisfy

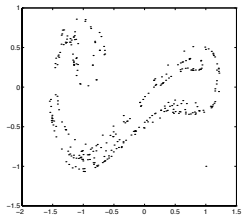


Figure 8.17: Poincare section of Duffings equation.

some condition

$$Gp(t, y(t_n + \theta h)) = 0 \tag{8.35}$$

where θ is the usual fraction of our step and the function $GP(t, Y)$ defines the plane or surface we wish the solution to cross and where points on the Poincare section are found.

Continuous extension

If we introduce the formula we found above for the continuous extension we get the following

$$GP(y_n + h\sum_{s=1}^{\nu} d_s(\theta) f(\xi_s)) = GP(t, \theta) = 0 \tag{8.36}$$

The efficiency of this application of the continuous extension is verified by the fact that **Matlab** has a built-in *event* functionality that appliesa similar technique to track down this kind of discrete points.

8.5 Differential Algebraic Equations

The theory of numerical solution of ordinary differential equations (ODE's) has been developed since the early part of this century beginning with Adams and Runge and Kutta. At the present time the theory is well understood and the development of software has reached a state where robust methods are available for a large variety of problems. The theory for Differential Algebraic Equations (DAE's) has not been stydied to the same extent, early attempts have been made by Gear and Petzold in the early 1970'es [16]. Not only are the problems harder to solve but the theory is also harder to understand.

The problems that lead to DAE's are found in many applications of which some are mentioned in later chapters of these lecture notes. The choice of sources for problems have been influenced by the students following this version the course [2]. For further treatment of this kind of problems readers are referred to [3].

Definitions

The problems considered are in the most general form a fully implicit equation of the form

$$F(y', y, t) = 0 \tag{8.37}$$

F and y are of dimension n and F is assumed to be sufficiently smooth. This is the non-autonomous form, an autonomous case is defined by

$$F(y', y) = 0 \tag{8.38}$$

Since the non-autonomous equation may be made autonomous by adding the equation $t' = 1$ we need not consider the non-autonomous form seperately. (this may be subject to debate since the non-autonomous case can have special features)

A special case arises when we can solve for the y'-variable since we can make the equation explicit and obtain a system of ODE's, the condition for this is that $\frac{\partial F}{\partial y'}$ is nonsingular. The case when this process does not work is the case we wish to consider in these notes it is the case when $\frac{\partial F}{\partial y'}$ is singular. We talk about systems of Differential Algebraic Equations or DAE's for short.

Semi-explicit DAE's

The simplest form of problem is the one where we can write the system in the form

$$\begin{aligned} y\prime &= f(y,z) \\ 0 &= g(y,z) \end{aligned}$$

and g_z has a bounded inverse in a neighbourhood of the solution.

Assuming we have a set of consistent initial values (y_o, z_o) it may be shown that in this case z can be found as a function of y. This implies local existence, uniqueness and regularity of the solution.

Index

Numerous examples exist where the conditions above do not hold. These cases have general interest and below we give a couple of examples from applications.

Example. We consider the problem defined by the system of three equations

$$\begin{aligned} y_1' &= y_3 \\ 0 &= y_2(1-y_2) \\ 0 &= y_1 y_2 + y_3(1-y_2) - t \end{aligned}$$

The second equation has two solutions $y_2 = 0 \text{and} y_2 = 1$ and we may get different situations depending on the initial conditions. t is a parameter of our choice.

case 1: if $y_2 = 0$ we get $y_3 = t$ from the last equation and we can solve the first equation for y.

case 2: Setting $y_2 = 1$ we get $y_1 = t$ from the last equation and $y_3 = 1$ comes out of the first one.

Example. Example 1.2: Implicit algebraic variable.

$$y' = f(y,z) \tag{8.39}$$
$$0 = g(y) \tag{8.40}$$

In this case we have that $g_z = 0$ and the condition of boundedness of the inverse does not hold. However, if the condition that $g_y f_z$ has a bounded inverse holds we can do the trick of differentiating the second equation leading to

$$0 = g_y(y)f(y,z) \tag{8.41}$$

and this will then be like the semi-explicit case where the conditions are satisfied.

We now introduce the definition of *Index*

Definition 8.5.19. For general DAE-systems we define the index along the solution path as the minimum number of differentiations of the systems that is required to reduce the system to a set of ODE's for the variable y.

Description.
The concept of index has been introduced in order to qualify the level of difficulty that is involved in the solution of the DAE. This must be understood in the way that methods that converge for one index may not be useful for higher index. Index is a concept that indicates the degree of difficulty.

Definition 8.5.20. The perturbation index is defined as beeing complementary to the differential index [10] Problem (1) has perturbation index m along a solution y if m is the smallest integer such that, for all functions $\hat{y}$ having a defect

$$f(y', y) = \delta(x) \tag{8.42}$$

there exists an estimate

$$\| \hat{y}(x) - y(x) \| \leq C(\| \hat{y}(0) - y(0) \| + \max \| \delta(\xi) \| + \ldots + \max \| \delta^{(m-1)}(\xi) \|) \tag{8.43}$$

whenever the expression on the right hand side is sufficiently small. C is a constant that depends only on the function f and the length of the interval. If we consider the ODE-case (1.1), the lemma by Gronwall [12, p. 62] gives the bound

$$\| \hat{y}(x) - y(x) \| \leq C(\| \hat{y}(0) - y(0) \| + \max_{0 \leq \xi \leq x} \| \int_0^{\xi} \delta(t)dt \|). \tag{8.44}$$

If we interpret this in order to find the perturbation index it is obviously zero.

Index reduction

A process called index reduction may be applied to a system for lowering the index from an initially high value down to f.ex. index one. This reduction is performed by successive differentiation and is often used in theoretical contexts. It is illustrated by an example.

Example.

We consider again Example 1.2, (8.39) and differentiate this equation two times whereby we obtain

$$0 = g_y f_x + g_{yy} f^2 + g_y f_z z' \tag{8.45}$$

$$z' = \frac{g_y f_x + g_{yy} f^2}{g_y f_z} \tag{8.46}$$

The two differentiations have reduced the system to one of index zero and we can solve the resulting ODE-system using well known methods.

If we want to find the perturbation index we may look at the equations for the perturbed system

$$\hat{y}' = f(\hat{y}, \hat{z}) + \delta(x) \tag{8.47}$$

$$0 = g(\hat{y}) + \theta(x) \tag{8.48}$$

Using differentiation on the second equation leads to

$$0 = g_y(\hat{y}) f(\hat{y}, \hat{z}) + g_y(\hat{y})\delta(x) + \theta'(x) \tag{8.49}$$

From the estimates and using Gronwalls lemma 8.44 like we did above we can now obtain

$$\| \hat{y}(x) - y(x) \| \leq C(\| \hat{y}(0) - y(0) \| + \int_0^x (\| \delta(\xi) \| + \| \theta'(\xi) \|) d\xi \tag{8.50}$$

$$\| \hat{z}(x) - z(x) \| \leq C(\| \hat{y}(0) - y(0) \| + max \| \delta(\xi) \| + max \| \theta'(\xi) \|) \tag{8.51}$$

All the maximum values are to be taken over $0 \leq \xi \leq x$.

Singular Perturbations

One important source for DAE problems comes from using singular perturbations [35]. Here we look at systems of the form

$$y' = f(y, z)$$
$$\epsilon z' = g(y, z), 0 < \epsilon \ll 1$$

Letting ϵ go to zero ($\epsilon \to 0$) and differentiating the second equation we obtain an index one problem in semi-explicit form. This system may be proven to have an ϵ - expansion where the expansion coefficients are solution to the system of DAE-s that we get in the limit when $\epsilon \to 0$. We will not here go into more detail around singular perturbations but refer to examples.

Example. The Van Der Pol equation

A very famous test problem for systems of ODE's is the Van Der Pol equation defined by the second order differential equation

$$y" = \mu(1 - y^2)y' - y$$

This equation may be treated in different ways, the most straigthforward is to split the equation into two by introducing a new variable for the derivative

$$y_1 = y, y_2 = y'$$
$$y_1{}' = y_2, y_2{}' = \mu(1 - {y_1}^2)y_2 - y_1$$

The system of two ODE's may be solved by any standard library solver but the outcome will depend on the solver and on the parameter μ. If we divide the second of the equations by μ we get an equation that has the character of a singular perturbation problem. Letting $\mu \to \infty$ we see that this corresponds to $\epsilon \to 0$ in the equation. Several other approaches may show other aspects of the nature of this problem for example [11] introduces the transformation $t = x/\mu$ after a scaling of y_2 by μ we get

$$y_1' = y_2$$
$$y_2' = \mu^2((1 - {y_1}^2)y_2 - y_1)$$

The introduction of $\epsilon = 1/\mu^2$ finally results in a problem in singular perturbation form

$$y_1' = y_2$$
$$\epsilon y_2' = (1 - {y_1}^2)y_2 - y_1$$

As explained previously this problem will approach a DAE if $\epsilon \to 0$ Using terminology from ODE's the stiffness of the problem increases as ϵ gets smaller giving rise to stability

problems for numerical ODE-solvers that are explicit while we expect methods that are A- or L-stable to perform better. The BDF-methods have been known to be very good to solve index-1 DAEs. They represent a family of methods that have $L(\alpha)$- stability properties.

The Van Der Pol equation is derived as a model of an electric oscillator circuit in Section 9.2, look there for further information.

8.6 General Linear Methods

In order to obtain the properties that are necessary for solving DAE problems of higher index than one we will introduce a new class of methods that are constructed as a combination of multistep- and Runge-Kutta methods. The general form corresponds to a Butcher tableau of the form

A	U
B	V

The form of the method is written in the following form:

$$Yi = y_n + \sum_{j=1}^{q} u_{i,j} y_{n-j} + h \sum_{j=1}^{s} a_{i,j} f(Y_j)$$
$$y_{n+1} = y_n + \sum_{j=1}^{q} v_{i,j} y_{n-j} + h \sum_{i=1}^{s} b_i f(Y_i)$$

The idea behind this construction is to introduce more degrees of freedom in the method in order to obtain better order and stability properties. For a thorough description of the methods we refer to the recent book by Butcher [5].

8.7 Collocation

There is a general interest into obtaining favorable order and stability properties of the methods. In the case of one-step methods the most favorable choices are by selecting among the implicit methods. The will provide high order and the possibility of getting A- or even L-stability. A good way to find implicit one step methods can be found by using the idea of *Collocation*. The idea behind Collocation is to require that the solution satisfies the differential equation at a discrete set of points carefully chosen in order to give the method certain properties. For a detailed description we refer to [23]. We define the solution

$$\begin{aligned} y' &= f(t,y), t \in [a,b] \\ y(t_n) &= y(a+nh) \\ t_{n+1} &= t_n + h \end{aligned}$$

Choose ν distinct collocation parameters $c_1, c_2, \ldots, c_\nu$
and define a polynomial of degree ν such that the collocation conditions are satisfied.

$$\begin{aligned} u(t_n) &= y_n \\ u'(t_n + c_j h) &= f(t_n + c_j h, u(t_n + c_j h) \end{aligned}$$

This is the basis of the collocation design idea:
select $y_{n+1} = u(t_{n+1})$
and use the polynomial to find conditions on the coefficients. The theory behind the construction is provided by the following

Theorem 8.7.21. Construct the Implicit Runge Kutta (IRK)-method
set

$$q(t) = \prod_{j=1}^{\nu}(t - c_j),\ q_l(t) := \frac{q(t)}{t - c_l}\ l = 1, 2, \ldots, \nu$$

let

$$\begin{aligned} a_{j,i} &:= \int_0^{c_j} \frac{q_i(\tau)}{q_i(c_i)} d\tau,\ for\ i, j = 1, 2, \ldots, \nu \\ b_j &:= \int_0^1 \frac{q_j(\tau)}{q_j(c_j)} d\tau,\ for\ j = 1, 2, \ldots, \nu \end{aligned}$$

This process leads directly to a scheme that has the Butcher tableau

The theorem is constructive and practically applicable for designing many methods, we exemplify the use by a simple example.

c	**A**
	b

Example. Example of an IRK-method.

An IRK-method. Choose :

$$\nu = 2 \; and \; \mathbf{c} = [0, 2/3]$$

We then get the polynomial:

$$q(t) = t(t - \frac{2}{3})$$

and

$$q_1(t) = t - \frac{2}{3} \; , \; q_2(t) = t$$

The resulting method then becomes:

0	0	0
$\frac{2}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
	$\frac{1}{4}$	$\frac{3}{4}$

Table 8.4: IRK

Theoretically the basis of finding the accuracy properties of a method of this construction is provided by the following theorem.

Theorem 8.7.22. Grøbner's lemma
Let **v** be a smoothly differentiable function that obeys the initial condition $v(t_0) = y_0$
Then:

$$v(t) - y(t) = \int_{t_0}^{t} \Phi(t, \tau, v(\tau)) d(\tau, v) d\tau \; , \; t \geq t_0 \tag{8.52}$$

where Φ is the matrix of partial derivatives of the solution of the ODE $w' = f(t, w), w(\tau) = v(\tau)$ with respect to $v(\tau)$.

The accuracy of the method is given by the order that is obtained and we have the following result.

Theorem 8.7.23. Suppose that the polynomial q(t) satisfies the condition:

$$\int_0^1 q(\tau)\tau^j d\tau = 0 \; for \; j = 0, 1, \ldots, m-1 \tag{8.53}$$

for some $m \in \{0, 1 \ldots, \nu\}$
Then the collocation method is of order $m + \nu$.

Orthogonal collocation

For collocation to work to its maximum potential the choice of polynomials is very essential. The most attractive choice is among the families of Orthogonal polynomials. We will illustrate this by stating very shortly the important results without proofs.

Theorem 8.7.24. Let $c_1, c_2, \ldots, c_\nu$ be the zeros of the polynomial
Ê

$$\tilde{P}_\nu \in \{the \; set \; of \; Polynomials - of - degree \; \nu\}$$

that are orthogonal with the weight function $\omega(t) = 1 \; , 0 \leq t \leq 1$
Then the underlying collocation method has order 2ν.

Example. **The implicit Midpoint Rule** As an example of a method with this property we have the following: Here we have the polynomial: $\tilde{P}_1(t) = t - \frac{1}{2}$ and $c_1 = \frac{1}{2}$.

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

Table 8.5: IMPR

The order is 2. The stability function can be found from the test equation.

$$y_{n+1} = \frac{1 + \frac{1}{2}h\lambda}{1 - \frac{1}{2}h\lambda} y_n$$

This is an example of a *Padè - approximation* to the exponential function. This shows that the IMPR is equivalent to the Trapezoidal method with respect to order and stability. Since we chose $\nu = 1$ we get order $q = 2$.

The collocation idea may be extended to cover a wider range of methods, for details see [41].

Chapter 9

ODEs in applications

In this chapter, we consider some non-trivial applications based on coupled sets of Ordinary Differential Equations (ODEs) for the numerical methods considered in the early chapters. The applications will be presented and discussed with the purpose of illustrating some of the difficulties in real applications and the strengths of the numerical methods applied when these are properly implemented. All application examples have been solved using Matlab routines that are similar to those considered in the course.

9.1 Dynamics of a rolling Railway Wheel set

The fundamental guidance system of railways consists of a flanged wheel set of steel rolling on a track with two steel rails. A clearance between the outer edge of the flange of the wheel and the inner face of the rail is provided to prevent squeezing of the wheels between the rails. We want to calculate the motion of the wheel set relative to the car body for varying values of the speed, V, of the vehicle.

As long as the wheel set rolls along a straight track, the flanges ought not contact the rails. Disturbances in the track geometry will push the wheel set away from its centered position but owing to the stabilizing effect of the wheel profile, the wheel set will then oscillate horizontally around the center line of the track. Flange contact will only occur if the disturbances are sufficiently large. When it occurs, the restoring force suddenly grows very fast. The oscillations normally die out fast because of the dissipation in both the wheel/rail contact surface and the suspension of the car body.

There exists for each and every railway vehicle, however, a certain speed, V_c, denoted *the critical speed*, above which the oscillations no longer die out, and the wheel sets will exhibit sustained self-oscillations with an amplitude, which grows with the speed. The oscillations will excite the other vehicle structures and in particular the car body, which leads to a strong decrease of the comfort in passenger trains and to possible damage of sensitive freight articles, such as electronic equipment, in freight trains.

The calculation of the critical speed is therefore a very important issue for railway engineers. The basis for the calculation is a dynamical model of the moving vehicle.

A simple model of a wheel set running on a rail consists of two truncated rigid cones connected with a rigid axle, which run on two rigid rails with surfaces, which are circular arcs. The apex of each cone lies outside the track. The restoring effect of the wheel flange is modelled by a very stiff linear horizontal spring. The mass of the car body is assumed to be so large relative to the mass of the wheel set that its inertia will not be affected by the relative motion of the wheel set beneath it. The car body moves with the constant speed, V, along the track, which is assumed to be straight and horizontal. V is a control parameter in the problem. See Figure 9.2.

The angle λ between the tangent to the wheel profile and the center line of the wheel set measured in radians is called *the contact angle*. It is normally positive. The positive contact angle makes the wheel set kinematically self-centering. In our problem the contact angle is equal to half the top angle of the truncated cones.

The rolling contact between the rails and the wheel set produces strain velocities in the contact surface. The geometric sum of these strain velocities is called 'the creep' and the creep normalized by the speed of the vehicle is called *the creepage*. The contact surface itself is a product of the real elastic deformations of the contacting bodies under the load of the vehicle. Hertz [21] showed that under rather general conditions the contact surface is plane and elliptic. From now on we shall therefore refer to it as the contact ellipse. The creepage is a vector in the contact ellipse, which has a component in the running direction of the vehicle v_x and a component perpendicular to the running direction v_y. Due to the shape of the wheel set as two 'opposite' truncated cones that are rigidly connected with an axle, also an angular strain velocity is produced in the contact ellipse. It is called the spin creep ϕ (see Figure 9.1). The creepage produces tangential stresses, which act on the wheels as well as on the rails. The geometric sum of the stresses can be expressed as a creep force, and the spin creep produces a torque, M. The creep force has a component in the running direction of the vehicle T_x and a component perpendicular to the running direction T_y, both in the contact ellipse. The torque, M is perpendicular to the contact ellipse (see Figure 9.1). In addition the axle load acts as a vertical force. The axle load has a component in the contact ellipse and a component normal to the contact ellipse. The reactions act on the wheel set.

We shall describe the motion of the wheel set in a cartesian coordinate system (see Figure 9.2) moving forward with the speed, V, of the vehicle. It must not be confused with the notation on Figure 9.1, which is a contact plane system. The new system has a vertical axis, which is upwards - the z-axis. The y-axis - is longitudinal in the direction of travel of the wheel set, and the third one - the x-axis - is oriented such that the xyz-coordinate system forms a right-handed cartesian system. The creep force and spin torque acting on the wheel set must therefore be expressed in this coordinate system (see Figure 9.2).

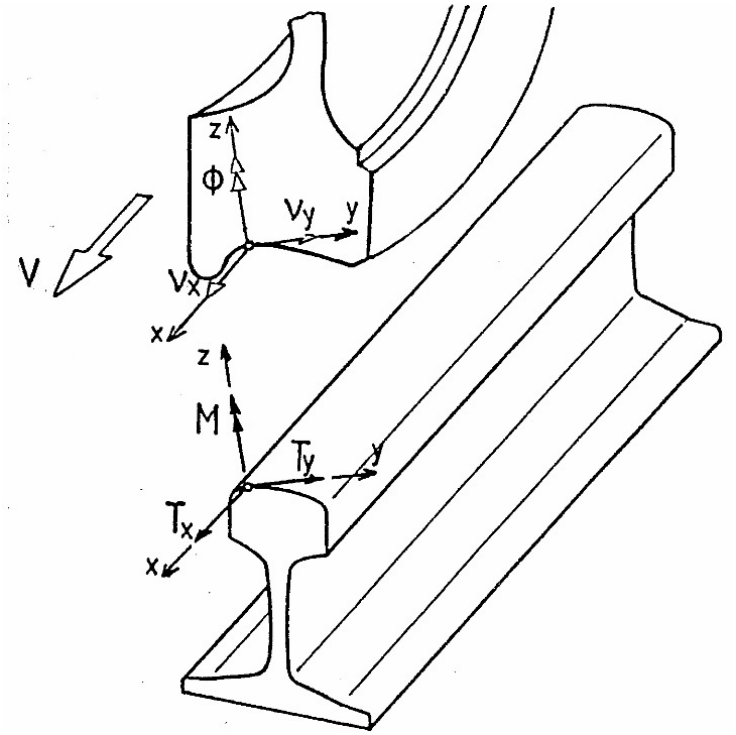


Figure 9.1: Sketch of contact plane systems for a wheel and a railway flange.

Now Newton's equations of motion are used to set up the mathematical model for the motion of the rolling wheel set in the moving coordinate system:

$$m\ddot{x} + 2k_1 x + 2F_x + F_T(x) = 0, \tag{9.1}$$

$$I\ddot{\phi} + 2aF_y = 0. \tag{9.2}$$

x denotes the lateral displacement of the wheel set and ϕ the yaw angle. m is the mass of the wheel set and I its moment of inertia around the z-axis. k_1 is the spring constant, and a is half the distance between the running circles on the wheels. $F_T(x)$ is the restoring force of the strong spring with a clearance that simulates the action of the wheel flanges.

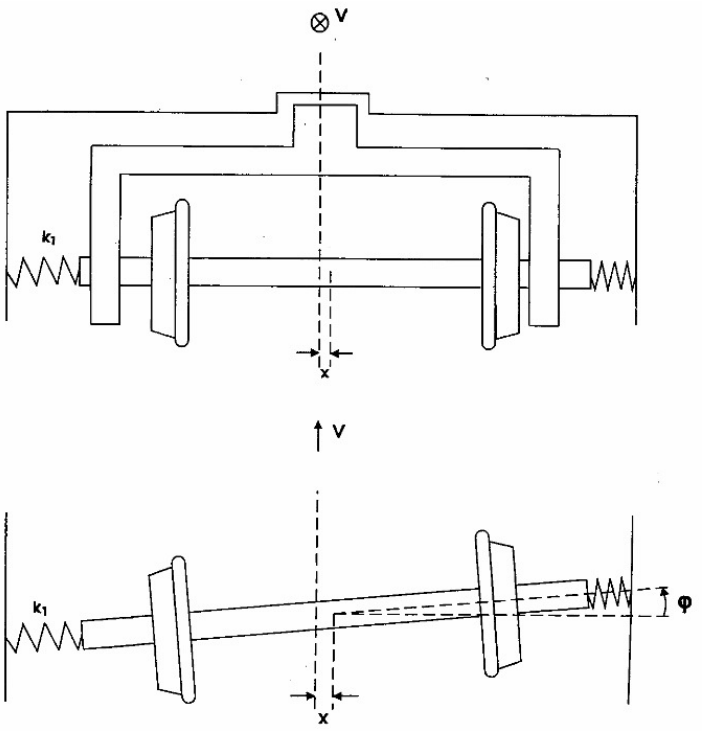


Figure 9.2: Sketch of wheel set and notations in the contact plane.

F_x and F_y are the lateral and longitudinal creep forces respectively. They depend nonlinearly on the displacements x and ϕ, and we shall apply the Vermeulen-Johnson theory [47] to find the dependence of the creep forces on the displacements. It is a simple relation that neglects the spin creep. It is permissible, because we assume that the contact angle, λ, is small. We furthermore assume that the vertical displacements are so small that their effects have no influence on the horizontal force balance, so they can be neglected. We are only interested in the horizontal motion.

The lateral and longitudinal components of the creepage are:

$$\xi_x = \frac{1}{V}\dot{x} - \phi, \tag{9.3}$$

$$\xi_y = \frac{a}{V}\dot{\phi} + \frac{\lambda x}{r_0}. \tag{9.4}$$

The total creep ξ_R is calculated from

$$\xi_R = \sqrt{(\xi_x/\Psi)^2 + (\xi_y/\Phi)^2}. \tag{9.5}$$

Ψ and Φ are weight constants, which depend on the length of the main axes of the contact ellipse. They are tabulated in Vermeulen-Johnson's theory. V is the speed of the vehicle, λ is the contact angle and r_0 is the rolling radius of the centered wheel set.

We now introduce a new variable:

$$u = (G\pi a_e b_e/\mu N)\xi_R, \tag{9.6}$$

and following Vermeulen-Johnson we finally obtain the nonlinear relation between the normalized total creepage u and the total creep force F_R:

$$F_R = \begin{cases} \mu N(u - u^2/3 + u^3/27), & , u < 3, \\ \mu N & , u \geq 3. \end{cases} \tag{9.7}$$

G is the shear modulus and $\pi a_e b_e$ is the area of the contact ellipse. μ is the coefficient of adhesion and N is the normal force on the contact ellipse. The creep force components F_x and F_y that enter the equations (1) and (2) are finally found from the two equations:

$$F_x = (\xi_x/\Psi)F_R/\xi_R, \tag{9.8}$$
$$F_y = (\xi_y/\Phi)F_R/\xi_R. \tag{9.9}$$

We shall use the system of equations to find the critical speed V_c of a simple wheel set model. For that purpose the *equilibrium solutions* of our dynamical problem must be investigated. An *equilibrium solution* of a dynamical problem is defined as the asymptotic solution to an initial value problem for the time, t $\to \infty$, in other words, the solution found after the transients have died out. Notice, that in contrast to linear dynamical problems, where the equilibrium solution is uniquely defined, nonlinear dynamical problems may have several co-existing equilibrium solutions. In parameter dependent problems such as our problem (the control parameter is V) the number of equilibrium solutions in general depends on the parameter. It is easy to show that our nonlinear dynamical problem has the trivial solution $(x, \phi) = (0, 0)$ for all values of V. This solution is the desired calm ride of the wheel set. We want to

find the value V_c, where another equilibrium solution - a time periodic oscillation - first appears. We are therefore looking for an additional equilibrium solution. It may be characterized simply by its amplitude, which is different from zero in contrast to the already found stationary solution $(x, \phi) = (0, 0)$. Since the flange force is not active for motions close to $(x, \phi) = (0, 0)$, we neglect it in our simple model.

The search for the equilibrium solution with an amplitude different from zero must be done numerically due to the complicated nonlinear form of the creep forces. We start with a solution of an initial value problem formulated by our dynamical problem together with four suitable initial conditions for a small value of V - for example $(x, \dot{x}, \phi, \dot{\phi}) = (\epsilon, 0, 0, 0)$ where ϵ is small - for example 0.0001. We find that the equilibrium solution is $(x, \phi) = (0, 0)$. We increase the parameter value, V and solve the initial value problem for stepwise growing values of V until the equilibrium solution $(x, \dot{x}, \phi, \dot{\phi})$ is different from $(0, 0, 0, 0)$. The last applied parameter value, V is then above the desired critical speed V_c. A better approximation of V_c can then be found by application of the Newton-Raphson scheme.

The method we apply is called *path following*; it means that a path of an equilibrium solution is followed in a parameter-state space in dependence on the parameter. The result can be plotted on a plane with the parameter on the x-axis and a suitable norm of the equilibrium solution on the y-axis. In our case such a norm is the amplitude of the lateral displacement of the equilibrium solution. The amplitude of the lateral displacement of the stationary solution equals zero - the x-axis - and the amplitude of the oscillating solution is different from zero, so the different solutions are easy to distinguish.

Such a plot is Figure 9.3. It is called a *bifurcation diagram*, because it shows how the two equilibrium solutions bifurcate from each other at the critical parameter value V_c. V_c is therefore called a *bifurcation point*. We find $V_c \cong 58$ m/s (≈ 209 km/h). In our example we have extended the calculations to higher values of V in spite of the fact that the results are unrealistic, when the restoring flange force is neglected. The reason is that we can demonstrate how the oscillating periodic solution bifurcates repeatedly into chaos for higher values of V.

The parameter values that were used for the calculation of the plot in Figure 9.3 are listed in the following table:

Additional information about wheel/rail mechanics and the dynamics of railway vehicles can be found in [46].

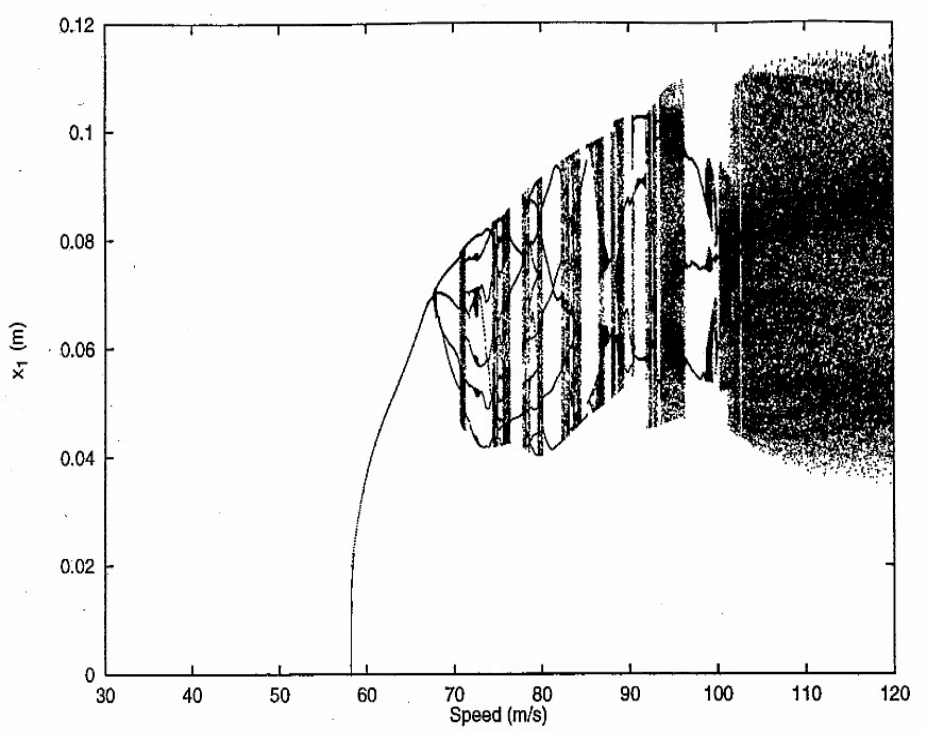


Figure 9.3: Bifurcation diagram for the dynamic railway problem.

λ	0.05
r_0	0.4572 m
μ	0.15
μN	10 kN
$G\pi a_e b_e$	656.3 kN
Ψ	0.54219
Φ	0.60252
k_1	18230 N/m
m	1022 kg
I	678 kg·m^2
a	0.716 m

9.2 The Van der Pol oscillator

The Van der Pole equation is a well known test-problem that has been used for benchmarking software for solving systems of ODE's. It has some properties that make is well suited for evaluation of the performance and efficiency of an implementation of a given method.

In the discussion on DAE's the Van der Pol equation was used as an example of a singular perturbed system. We will now explain the background for this equation as a model of an electrical oscillator circuit.

In the figure a simple electrical circuit is shown. It is a LRC-circuit with a negative resistance that could be a tunnel-diode or an old-fashioned electronic tube called a triode. Using simple circuit analysis we can derive the differential equation that models the electrical circuit. The equations for the various elements are

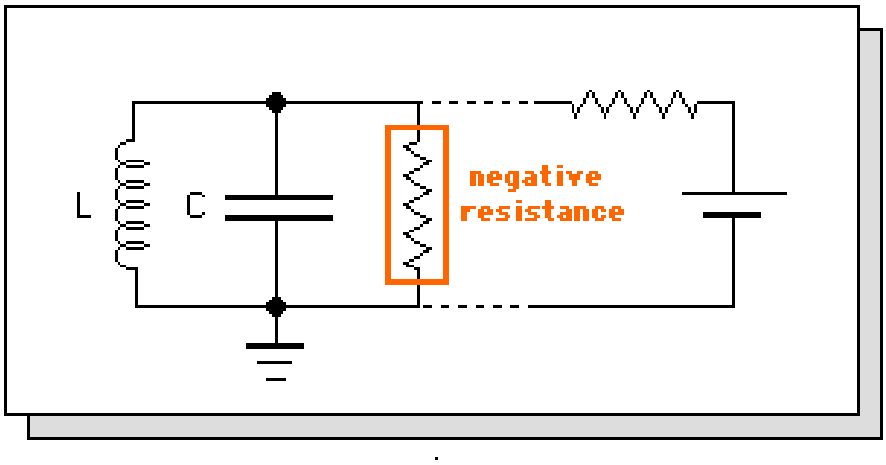


Figure 9.4: Oscillator circuit.

Charge on the capacitor C : $q = Cv$
Voltage over the inductor L : $v = -L\frac{di_L}{dt}$
Current through the negative resistor : $i_N = -\alpha v + \beta v^3$ The model suggested by this component is shown in the figure below.

We now consider the equation for conservation of charge in the system.

$$i_L = i_C + i_N \ , \ \frac{di_L}{dt} = \frac{di_C}{dt} + \frac{di_N}{dt}$$

Using the models above this leads to the second order differential equation

$$\frac{d^2v}{dt^2} + \frac{d}{dt}(-\alpha v + \beta v^3) + \frac{v}{L} = 0$$

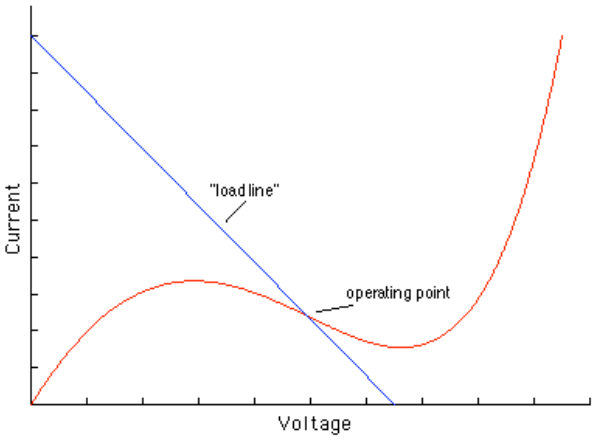


Figure 9.5: Negative resistor diagram.

$$\frac{d^2v}{dt^2} - \frac{\alpha}{C}\frac{dv}{dt} + 3\frac{\beta v^2}{C}\frac{dv}{dt} + \frac{1}{LC}v = 0$$

We now introduce $\omega^2 = \frac{1}{LC}$ and the scaling $t = k_t T$ and $v(t) = k_V V(t)$ and we get

$$\frac{k_V}{k_t^2}\frac{d^2V}{dT^2} - \frac{\alpha}{C}\frac{k_V}{k_t}\frac{dV}{dt} + 3\frac{k_V^3}{k_t}V^2\frac{dv}{dt} + \omega^2 k_V V = 0$$

After some rearrangements and the convenient choices of the scalings, $k_t^2\omega^2 = 1$ and $k_V^2 = \frac{\alpha}{3\beta}$ and introducing the parameter $\mu = \frac{\alpha}{\omega C}$ we can write the final model as the Van der Pol equation

$$\frac{d^2V}{dT^2} = \mu(1 - V^2)\frac{dV}{dt} - V$$

One of the interesting features of this problem is that for small values of the parameter μ the solution in the phase plane is almost a circle. For $\mu = 0$ it is a circle. When we increase μ the problem gets more and more stiff but the solution is periodic after a short transient. In electrical terms it means that there is an initial transient response and after that the voltage is a stable oscillation. The solution is on a closed curve in the phase-plane called the limit cycle. The solution has been computed by a fixed step Runge-Kutta fourth order method over a time interval corresponding to five periods of the oscillation. it is seen from the number of steps used that for the larger stepsizes an unstable solution appears (the scaling on the axes are well beyond the scale of the solution). The problem appears to be stiff and very small steps must be used in order to obtain a meaningful result. There are many ways of showing the solution , the next figure shows the solution as function of time from the same computation . The dependency of the parameter μ is shown in the next figure where a time-dependent parameter is chosen and the time history of the stepsize is presented together with the phase plane picture. This shows that the period increases with

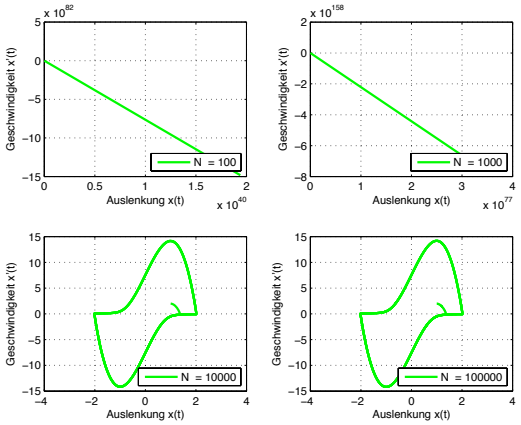


Figure 9.6: Van der Pol solution for $\mu = 10$.

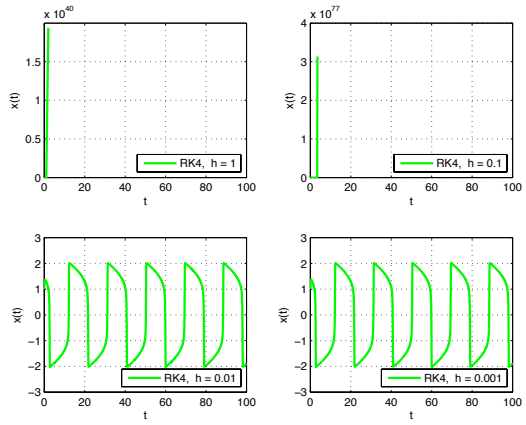


Figure 9.7: Van der Pol solution for $\mu = 10$.

increasing μ. In this test the formula for varying μ is $\mu = 0.01 \exp(t/3000)$ in this way the parameter is increasing from a small initial value to a very large number as time increases. At the same time the amplitude of the peaks in the first derivative increases while the amplitude of the solution remains close to 2. This is a solution computed with the highly sophisticated stepsize control found by using the ESDIRK method derived in Section 8.2. While this last example is not a practical application it is presented as a very challenging test-problem for testing the robustness of a stiff solver. This is not a periodic solution.

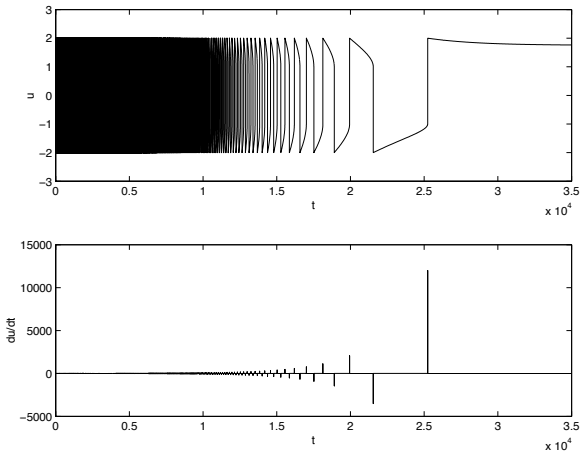


Figure 9.8: Van der Pol solution for variable μ.

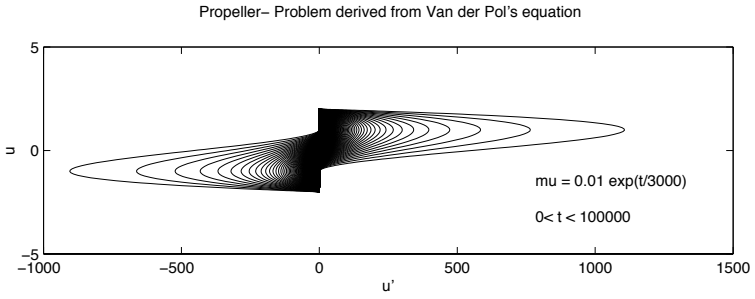


Figure 9.9: Phase plane solution for variable μ.

9.3 The golfstroke

Any golfplayer will search for the ultimate distance in a golfstroke, thereby minimizing the number of strokes needed to get the ball in the hole. The distance depends on a number of parameters, the choice of club, the choice of ball and the way the golfer hits the ball. The aim of this study is to simulate under certain assumptions the stroke. The modelling is done in three stages, first the swing of the club, secondly the impact of the club with the ball and thirdly the flight of the ball. Each stage is modelled seperately but transition between stages is very important in order to reflect the physics of the problem.

The basis of the modelling process is the Newtonian mechanics for the dynamic systems involved together with some models of impact that is based on measurements and analysis of the real world. The resulting systems of nonlinear differential equations may be solved numerically. The impact stage is relying on data that are available on public internet pages. The results can be compared with real measurements to validate the model.

Motivation

The industry producing sports equipment is a multi-million-dollar industry that relies to a large extent on research in the physics of their products. Golf equipment manufacturing is a very good example of this and modern golf depends a lot on research carried out at universities and research institutes like NASA. The modern golfball with dimples is a good example of aerodynamics being used to optimize the flight of the ball.

The Double Pendulum

As a model of how the player swings the driver a first approach is to use a double-pendulum model to describe the system of arm and club. In textbooks we find for small angles, the formulae:

$$\begin{aligned}(m_1+m_2)l_1^2\ddot{\theta_1}+(m_1+m_2)l_1g\theta_1+m_1l_1l_2\ddot{\theta_2}=0\\ m_2l_2^2\ddot{\theta_2}+m_2l_1l_2\ddot{\theta_1}+m_2l_2g\theta_2=0\end{aligned} \tag{9.10}$$

The model is limited to cases where we have no mass in the arms of the pendulum and large angles are not covered, we need to get a multibody-model based on the mechanics of a system with two arms and two masses as illustrated in Figure 9.10. The limitations of this model are not realistic for an actual golfstroke since first of all it is not limited to small angles and the two masses are unrealistic. We therefore need to look for a better model.

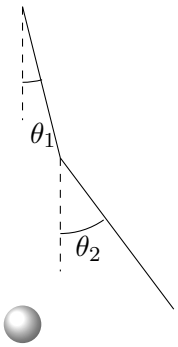


Figure 9.10: Illustration of a double pendulum and notation for angles.

The multibody system

If we adapt a two-body approach we obtain a set of dynamic equations that are valid for large movements of the arm and club. the basic assumption is that all movements take place in a vertical plane. Although this an ideal situation that is not a golfers movement it is useful for the kind of study we want to carry out. The two angles shown in Figure 9.10 are chosen as the state variables which lead to the two second order ODEs

$$\begin{aligned} \ddot{\theta}_1 &= (a_{22}f_1 - a_{12}f_2)/J \\ \ddot{\theta}_2 &= (a_{11}f_2 - a_{21}f_1)/J \end{aligned} \tag{9.11}$$

Where the expressions on the right hand side are given by

$$\begin{aligned} f_1 &= -M_k L_A L_G \dot{\theta}_1^2 sin(\theta_2 - \theta_1) - M_K L_G g \sin\theta_1 + M_1(t) \\ f_2 &= M_k L_A L_G \dot{\theta}_2^2 \sin(\theta_2 - \theta_1) - (\frac{1}{2}M_A + M_K)L_A g \sin\theta_1 + M_0(t) + M_1(t) \\ J &= (I_A + M_K L_A^2)(I_G + M_K L_G^2) - (M_k L_A L_G \cos(\theta_2 - \theta_1))^2 \end{aligned} \tag{9.12}$$

and the coefficients a_{11}, a_{12}, a_{21} and a_{22} can be derived from the geometry.

Trying the multibody model

The system has chaotic behaviour for long time intervals but we only need a very short interval of approx 0.4 seconds.

The initial condition can be found from analysing a Golf-player when he starts the swing.

The two moments that drives the club is :
M_1: The force from the shoulder movement.
M_2: The moment from the arms kept straight and hands.

Hitting the ball

When the clubhaed actually hits the ball a model for the impact is used but first the exact time of impact must be found.

The stopping criterium

When the clubhead is within the ball radius the ball is hit. Given the angles we get the condition:

$$\sqrt{l_1 \cos\theta_1^2 + (l_2 \cos\theta_2 + l_z)^2} - 0.0205 < 0. \tag{9.13}$$

The picture on the front page of these lecture notes shows another example of a

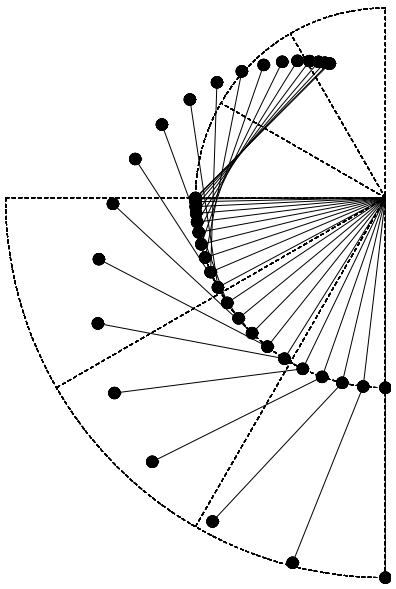


Figure 9.11: Golfer swings the driver.

swing of the club seen from the front of a left-handed player for a case where a constant momentum is applied through the shoulder part and a linearly increasing moment is applied by the hands. This is a close to natural swing simulation.

Forces acting on a spinning golf ball

We consider a spinning golf ball moving in two dimensions under the influence of gravitation $\mathbf{F_g}$, air resistance $\mathbf{F_c}$, and Magnus lifting force $\mathbf{F_m}$ due to spinning of the ball as illustrated in Figure 9.12. The gravity force F_g is well known [7], and given

by

$$\mathbf{F}_g = -mg \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{9.14}$$

Air Resistance

The air through which the ball is moving will, because of friction due to air resistance, change the motion of the ball from what it would be in a vacuum. In vacuum, the trajectory has the shape like parabola (see for example [7]). Taking account the air resistance, ball falls more steeply than it rises. Horizontal friction due to air is acting during the whole flight opposite to the horizontal motion of the ball, this means that it is decreasing the horizontal part of the velocity. While the ball is rising, the vertical component of the frictional force has the same direction than the force caused by gravity. When the ball is going down, the horizontal part of the velocity is less than during the rise, and the vertical part of the frictional force is pointing upwards. Because the ball does not go as far horizontally during the fall as during the rise, the fall is steeper. Air resistance $\mathbf{F}_c$ can be presented as

$$\mathbf{F}_c = \kappa\, v^2 \begin{bmatrix} \cos(\alpha) \\ \sin(\alpha) \end{bmatrix} \text{ such that } \kappa = -\frac{1}{2}\rho A c, \tag{9.15}$$

where c is a shape constant (for ball $0.3 - 0.34$), ρ is the density of the air, $\mathbf{v}$ is the velocity of the ball, and A is the cross-sectional area of the ball measured in the plane perpendicular to its motion.

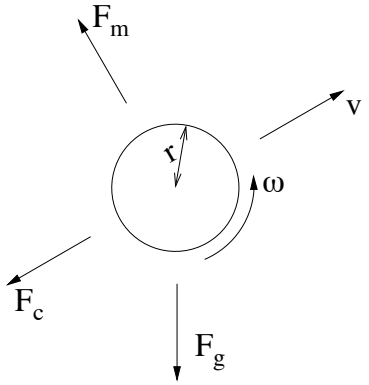


Figure 9.12: Illustration of forces acting on a spinning ball.

Magnus Effect

For a ball, which is spinning about an axis perpendicular to its direction of motion, the speed of the ball is different on opposite sides of the ball. The lower side of the

ball has a larger speed relative to the air than the upper side. This results in a force acting upwards. This force is known as the Magnus force [26] which can be written

$$\mathbf{F}_m = \pi\rho r^3 \mathbf{v}_\omega \times \mathbf{v}, \tag{9.16}$$

where $\mathbf{v}_\omega$ is the angular velocity. In numerical examples, we use, to maintain the model in two dimensions, the following approximation introduced by Davies [30]

$$\mathbf{F}_m = \kappa_1 \begin{bmatrix} -\dot{y} \\ \dot{x} \end{bmatrix} \Big(1 - \exp(-\kappa_2\omega_0)\Big), \tag{9.17}$$

where ω_0 is initial spin frequency (angular speed), $\kappa_1 = 9.05 \cdot 10^{-3}$, and $\kappa_2 = 0.00248$. By Newton's second law of motion, the total force $\mathbf{F}$ acting on the ball is

$$\mathbf{F} = \mathbf{F}_g + \mathbf{F}_c + \mathbf{F}_m = m_b \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix}. \tag{9.18}$$

Finally, the system of ordinary differential equations for the flight of the golf ball becomes

$$\begin{aligned} \ddot{x} &= -\frac{\kappa_1\big(1 - \exp(-\kappa_2\omega_0)\big)}{m_b}\dot{y} + \frac{\kappa}{m_b}\Big(\dot{x}^2 + \dot{y}^2\Big)\cos\arctan\Big(\frac{\dot{y}}{\dot{x}}\Big), \\ \ddot{y} &= -g + \frac{\kappa_1\big(1 - \exp(-\kappa_2\omega_0)\big)}{m_b}\dot{x} + \frac{\kappa}{m_b}\Big(\dot{x}^2 + \dot{y}^2\Big)\sin\arctan\Big(\frac{\dot{y}}{\dot{x}}\Big), \\ \dot{x}(0) &= v_0\cos\alpha_0, \\ \dot{y}(0) &= v_0\sin\alpha_0. \end{aligned} \tag{9.19}$$

If the weather is windy, the velocity of the wind can be applied to the velocity of the ball $\mathbf{v} = (\dot{x}, \dot{y})$. There are also some effects to study more to develop even more realistic models. Those are, for example, temperature and humidity which affect on air density ρ as well as the quality of grass which varies the trajectory. There is also reason why golf balls are dimpled. Although, there is less surface friction with a smooth ball than with a dimpled one, dimples increase the lifting force [25], and so the ball flies further.

Numerical examples

Above we have modelled the set of two coupled differential equations for $x(t)$ and $y(t)$ that includes gravity, air resistance, and the Magnus force (9.19). While choosing parameters, we have to take account some restrictions because of standards on golf ball size and weight. In Table 9.1, are the values of parameters used in numerical examples unless other mentioned.

The equation system 9.19 is solved by using Matlab standard function `ode45()` as fourth order Runge-Kutta solver. The values for $\dot{x}(t)$, $\dot{y}(t)$, $x(t)$, and $y(t)$ are calculated, and then the trajectories are plotted.

Initial speed	$\mathbf{v}_0$	$60\frac{\mathrm{m}}{\mathrm{s}}$
Initial angle due to club head	α_0	$43^\circ = \frac{43\pi}{180}$ rad
Acceleration caused by gravitation	g	$9.81\frac{\mathrm{m}}{\mathrm{s}^2}$
Mass of the golf ball	m_b	0.04593 kg
Density of the air	ρ	$1.0\frac{\mathrm{kg}}{\mathrm{m}^3}$
Initial rotation	ω_0	$50\frac{1}{\mathrm{s}}$
Radius of the ball	r	0.021335 m
Shape coefficient for ball	c	0.30

Table 9.1: Parameters used in numerical examples.

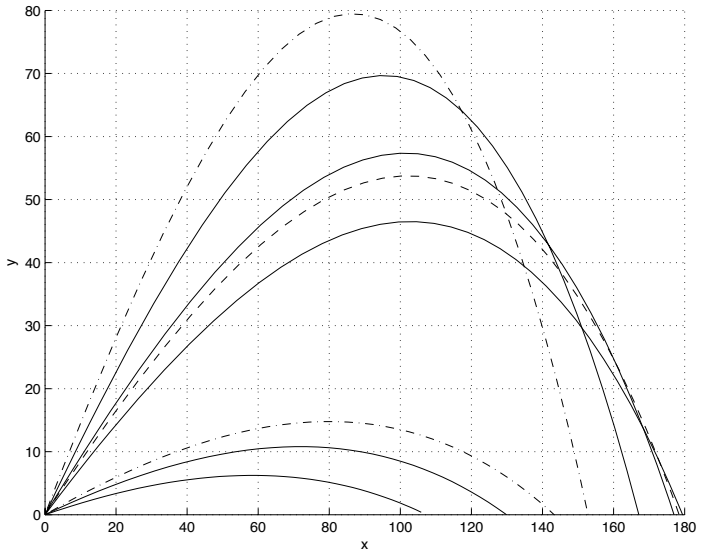


Figure 9.13: Trajectories with constant initial speed and different launch angles. The lower launch angle leads to lower shot. Optimal length of the ball flight is achieved with initial angle, 37°.

To hit a lower shot, a less angled club is chosen [18]. This derives a lower launch angle and less angular speed. We chose to compute trajectories with constant initial speed $\mathbf{v}_0 = 60$ m/s for launch angles 11°, 15°, 18°, 37°, 41°, 43°, 50° and 56°. Figureă9.13 shows a plot of the trajectory of a golf ball as a function of the launch angle, α_0. The maximum range is obtained for a launch angle 37°.

For given initial conditions, there always exists an optimal angular speed that results in the furthest range [13]. There are also higher and lower angular rotations that result in less range than the optimal one.

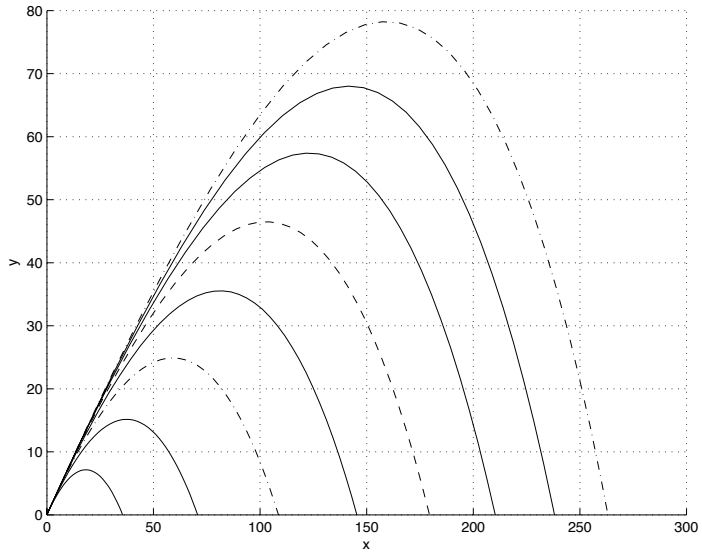


Figure 9.14: Trajectories with constant launch angle and different initial speed. The larger initial speed achieves the ball fly further.

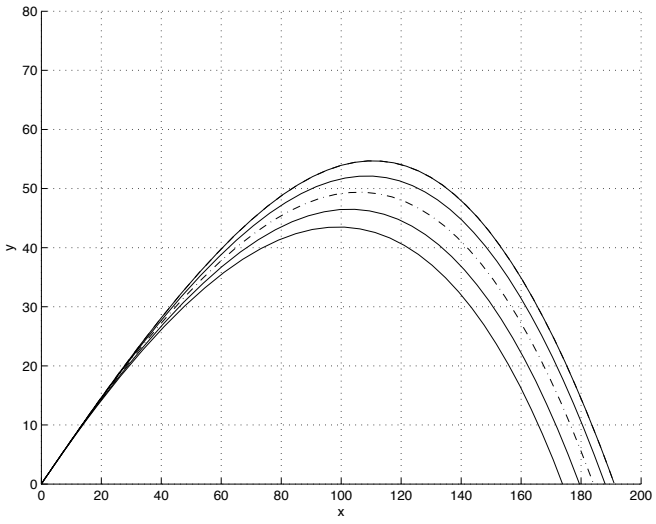


Figure 9.15: Trajectories with constant launch angle and initial speed as a function of the initial spin rate. Higher spinning rate achieves the ball fly further.

Increasing the speed of the club in hitting increases also the initial speed of the ball. This makes the ball fly longer. In Figure 9.14, it is plotted how far the ball will go given an initial speed. Launch angle is constant, 37°, and initial speeds used are 20 m/s, 30 m/s, 40 m/s, 50 m/s, 60 m/s, 70 m/s, 80 m/s, and 90 m/s.

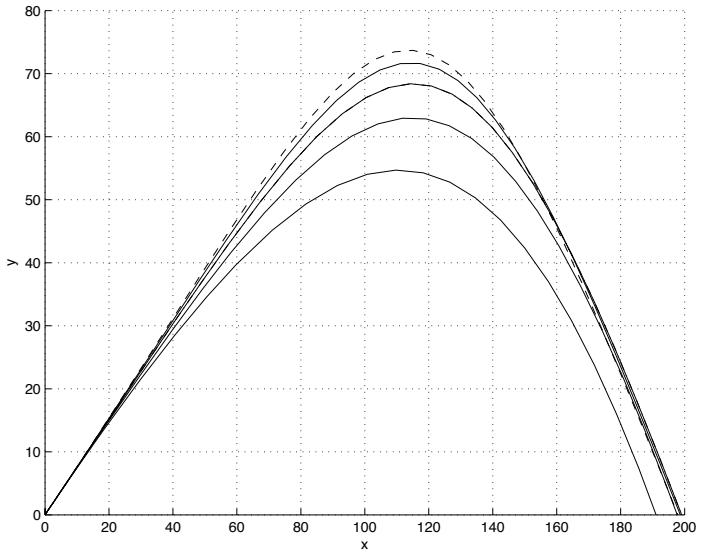


Figure 9.16: Trajectories with constant launch angle and initial speed as a function of the initial spin rate. Too high spin frequency decreases range.

The model with air resistance and without the Magnus force (the initial spin frequency $\omega_0 = 0\ \mathrm{s}^{-1}$) was compared to the one which takes into account both air resistance and the Magnus force ($\omega_0 \neq 0\ \mathrm{s}^{-1}$). In both cases, the same initial speed $\mathbf{v}_0 = 60$ m/s and launch angle 37° were used, but initial spin frequency ω_0 varies from $0\ \mathrm{s}^{-1}$ to $200\ \mathrm{s}^{-1}$ with stepsize $50\ \mathrm{s}^{-1}$. The Magnus force produces lift on the ball, and for these initial spin frequencies, it affects the trajectory of the ball so that the ball flies longer. In Figure 9.15, we can see those results.

Too much lift decreases the range. Trajectories, in which the initial spin frequency ω_0 varies from $200\ \mathrm{s}^{-1}$ to $1000\ \mathrm{s}^{-1}$ with stepsize $200\ \mathrm{s}^{-1}$, are plotted in Figure 9.16. For initial condition used, spin frequency of around 600 radians per second seems to produce the furthest range.

Magnus force can deform golf ball trajectory into strange forms. For example, if the Magnus force has into y-direction a component that is larger than gravity, the ball rises to a steeper angle than its initial angle. This kind of situation is presented as solid line in Figure 9.17 using initial speed $\mathbf{v}_0 = 65$ m/s, spin frequency 1500 radians per second, and 64° as launch angle. Comparison with same initial conditions, except launch angle 10°, which is corresponding to a driver, makes sense to choose different kind of clubs for different distances.

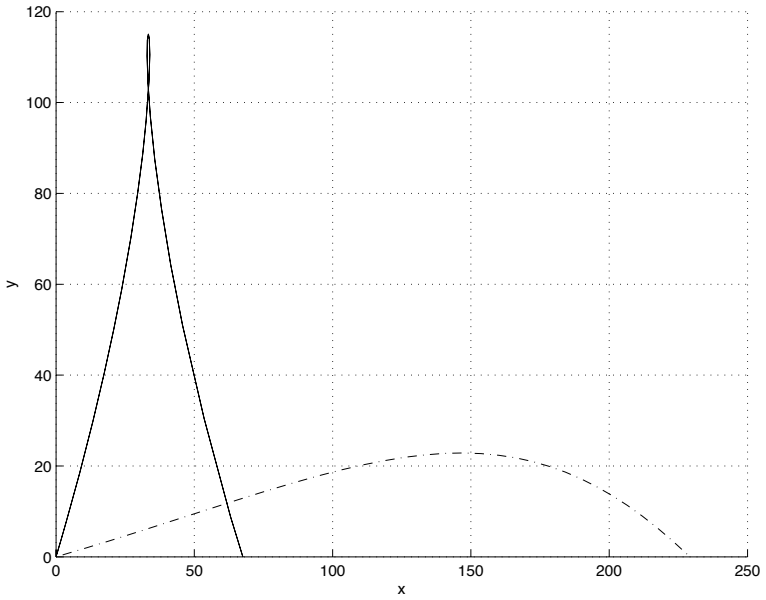


Figure 9.17: If initial spin frequency and launch angle are very large, ball rises to a steeper angle than its launch angle (solid line). Smaller launch angle makes the ball fly longer (dash-dot line).

Conclusions

The lower the launch angle the lower the flight path. For given initial conditions, there exists an optimal launch angle that results in the furthest range.

Adding initial speed makes the ball fly further. To do this, it is necessary to increase the speed of the club head before impact, or change the properties of the ball. With smaller and lighter balls, it is possible to make longer swings. That is the reason why there are minimum limits to weight and diameter of legal golf balls.

Higher spinning rate increases the lifting effect. For rather small initial spin frequencies the ball fly further. Too high spin frequency will decrease the range. So, there exists an optimal value of initial spin frequency, keeping other initial conditions fixed.

In practice, it is difficult to reproduce the same impact between ball and club, which also leads to different angular velocities. This might be one reason that makes golf an interesting game.

Appendix A

Prerequisites

Some mathematical tools are necessary for the student to feel comfortable with all the theory that is developed in these notes. Here we list some of the basic mathematics that is used throughout the book. For further help please look in [23] where the Appendix: *Bluffer's guide to useful mathematics* will prove helpful.

A.1 Taylor series

We will assume throughout that our solutions are infinitely smooth, this means that all derivatives exist and are bounded in the domain of the solution. Under this assumption we can write down the Taylor expansion of the solution around a point $(t_n, y(t_n))$:

$$\begin{aligned} y(t_n + h) &= y(t_n) + hy'(t_n) + \tfrac{1}{2}h^2 y''(t_n) + \cdots + \tfrac{1}{q!}h^q y^{(q)}(t_n) + \dots \\ &= y_n + hf(t_n, y_n) + \textstyle\sum_{q=2}^{\infty} \frac{1}{q!}h^q y^{(q)}(t_n) \end{aligned}$$

The first terms of the Taylor expansion of $y(t)$ around $t = t_n$ is

$$\begin{aligned} \hat{y}(t_n + h) &= \hat{y}(t_n) + h\hat{y}'(t_n) + \tfrac{1}{2}h^2\hat{y}''(\tau) \\ &= y_n + hf(t_n, y_n) + \tfrac{1}{2}h^2\hat{y}''(\tau) \ , \end{aligned}$$

We thus can find the truncation error as the difference between the full expansion and the truncated series.

A.2 Backward differences

We often wish to use tables of points or function values. Many tools exist to generate information from the tables and finite differences is a convenient way of deriving such information. We will here only use the *Backward differences* that are defined as follows.

Given the table

$$t_n\ , \quad n = 0, 1, 2, \ldots, N, \quad y(t_n) = y_n. \tag{A.1}$$

For convenience we assume that the points are equidistant, $t_n - t_{n-1} = h$. The table of Backward differences is then defined as

$$\left(\begin{array}{cccccccc}
n & y_n & \nabla_n & \nabla_n^2 & \nabla_n^3 & \nabla_n^4 & \nabla_n^5 & \nabla_n^6 \\
\hline
1 & y_1 & & & & & & \\
 & & y_2 - y_1 & & & & & \\
2 & y_2 & & \nabla_2 - \nabla_1 & & & & \\
 & & y_3 - y_2 & & \nabla_3^2 - \nabla_2^2 & & & \\
3 & y_3 & & \nabla_3 - \nabla_2 & & \nabla_4^3 - \nabla_3^3 & & \\
 & & y_4 - y_3 & & \nabla_4^2 - \nabla_3^2 & & \nabla_5^4 - \nabla_4^4 & \\
4 & y_4 & & \nabla_4 - \nabla_3 & & \nabla_5^3 - \nabla_4^3 & & \\
 & & y_5 - y_4 & & \nabla_5^2 - \nabla_4^2 & & & \\
5 & y_5 & & \nabla_5 - \nabla_4 & & & & \\
 & & y_6 - y_5 & & & & & \\
6 & y_6 & & & & & &
\end{array}\right)$$

The table of differences may be generated by the recursive use of the definition

$$\nabla_n^j = \nabla_n^{j-1} - \nabla_{n-1}^{j-1}\ , \quad \nabla_n^0 y_n = y_n \tag{A.2}$$

Using this definition we can find all the entries in the above table of Backward differences expressed in table entries. For example we get

$$\nabla^2 y_4 = \nabla y_4 - \nabla y_3 = y_4 - 2y_3 + y_2 \tag{A.3}$$

A.3 Interpolation

Interpolation is a tool that makes it possible to find values of functions that obtain given values on a discrete set of points

$$t_n\ , \quad n = 0, 1, 2, \ldots, N, \quad y(t_n) = y_n. \quad t = t_n + rh,\ 0 \le r \le 1. \tag{A.4}$$

The way we define the interpolating function $p(t)$ depends on the application but the one most often used is as a polynomial. In this case we can use for example the *Newton backward divided difference interpolation formula* on a discrete set of nodes with equal spacing

$$p(t_n + rh) = y_n + r\nabla y_n + \frac{r(r+1)}{2}\nabla^2 y_n + \ldots \tag{A.5}$$

In a slightly more compact form we can write the polynomial of degree q which interpolates $y(t)$ about $y(t_n)$ as

$$p_q(t_n + rh) = \sum_{i=0}^{q} (-1)^i \left(\begin{array}{c} -r \\ i \end{array}\right) \nabla^i y_n \tag{A.6}$$

with the *Binomial coefficient* defined as

$$\left(\begin{array}{c} n \\ k \end{array} \right) \equiv \frac{n!}{k!(n-k)!} = \frac{n(n-1)(n-2)\cdots(n-k+1)}{k!} \tag{A.7}$$

where k is a nonnegative integer and $n \in \mathbb{R}$.

A.4 Numerical Integration

Here we will just list some of the possible methods used in the text.

Rectangular formula

$$\int_a^b f(x)dx = h\sum_{i=0}^{n} f_i + h\frac{(b-a)}{2}f'(\eta) \tag{A.8}$$

Trapezoidal formula

$$\int_a^b f(x)dx = \frac{h}{2}(f_0 + f_n) + \sum_{i=1}^{n-1} f_i + h^2\frac{(b-a)}{12}f''(\eta) \tag{A.9}$$

Newton-Cote's formula

Substituting the interpolating Polynomial $P(x)$ for the function $f(x)$ one obtain

$$\int_a^b f(x)dx = \int_a^b P(x)dx + R_T$$

where the R_T term is the truncation error. This results in an approximation to the integral of the form

$$\int_a^b f(x)dx = h\sum_{i=0}^{n} A_i f_i + R_T \tag{A.10}$$

The polynomial degree determines the coefficients A_i and the truncation error. Table A.1 gives the coefficients for the first five together with the remainder term. The table entries are $a_i = A_i D$.

A.5 Nonlinear equations

The general form of a nonlinear system of equations is

$$\mathbf{G}(\mathbf{x}) = \mathbf{0} \tag{A.11}$$

where $x \in \mathbf{R}^n$.

N	D	a_0	a_1	a_2	a_3	a_4	a_5	R_T
1	2	1	1					$C_1(b-a)^3 f''(\xi)$
2	6	1	4	1				$C_2(b-a)^5 f^{IV}(\xi)$
3	8	1	3	3	1			$C_3(b-a)^5 f^{IV}(\xi)$
4	90	7	32	12	32	7		$C_4(b-a)^7 f^{VI}(\xi)$
5	288	19	75	50	50	75	19	$C_5(b-a)^7 f^{VI}(\xi)$

Table A.1: Table of coefficients for Newton-Cote formulaes.

Fixed point iteration

The simplest form of iteration is defined by transforming the system into the form

$$\mathbf{x} = \phi(\mathbf{x}) \tag{A.12}$$

This leads to the iteration

$$\mathbf{x}_{s+1} = \phi(\mathbf{x}_s) \tag{A.13}$$

For convergence of this iteration we require that the norm of the Jacobian of the function $\phi(\mathbf{x})$ matrix is bounded

$$\mathbf{J} = \frac{\partial \phi}{\partial x}$$

$$\frac{\partial \phi}{\partial x} = \begin{pmatrix} \frac{\partial \phi_1}{\partial x_1} & \frac{\partial \phi_1}{\partial x_2} & \frac{\partial \phi_1}{\partial x_3} \\ \frac{\partial \phi_2}{\partial x_1} & \frac{\partial \phi_2}{\partial x_2} & \frac{\partial \phi_2}{\partial x_3} \\ \frac{\partial \phi_3}{\partial x_1} & \frac{\partial \phi_3}{\partial x_2} & \frac{\partial \phi_3}{\partial x_3} \end{pmatrix}$$

$$\| \mathbf{J}(\mathbf{x}) \| < 1 \quad \text{for} \quad x \quad \text{near the solution.}$$

The requirement often leads to restrictions that are not desirable and in such cases we recommend the Newton Raphson iterative method.

Newton Raphson

The alternative to fix-point is to apply Newton-Raphson iteration, the advantage is that with a good initial guess it will always converge.

$$\mathbf{J} = \frac{\partial G}{\partial x}$$

$$\frac{\partial G}{\partial x} = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \frac{\partial g_1}{\partial x_3} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \frac{\partial g_2}{\partial x_3} \\ \frac{\partial g_3}{\partial x_1} & \frac{\partial g_3}{\partial x_2} & \frac{\partial g_3}{\partial x_3} \end{pmatrix}$$

Given an initial vector $\mathbf{x}_0$ we solve the linear systems

$$\mathbf{J}\delta^s = -\mathbf{r}(\mathbf{x}^s)$$
$$\mathbf{x}^{s+1} = \mathbf{x}^s + \delta^s$$

here the residual vector $bfr(\mathbf{x}^s)$ is introduced for the value of our function $\mathbf{G}(\mathbf{x}^s)$. The iterative process is stopped by a *Stopping criterion* based on the relative norm of the form

$$\|\delta\| \leq \|\mathbf{x}\|\epsilon$$

Here ϵ is a tolerance that must be chosen according to the precision wanted. The norm used depends on the type of problem and in some cases it is appropriate to use a component based relative scaling where each component in the δ-vector is scaled by the approximate solution component.

$$\max_j \frac{\mid \delta_j \mid}{\mid x_j \mid} \leq \epsilon$$

A.6 Polynomials

Polynomials are used in interpolation and in stability analysis. Some of the properties of polynomials and their roots are as follows. The definition of a polynomial of degree k is

$$P_k(x) = \sum_{j=0}^{k} \alpha_j x^j \tag{A.14}$$

The roots or zeros of the polynomials are those values $x = x_i$, $i = 1, 2, \ldots, k$ that satisfy the equation

$$P(x_i) = 0 \tag{A.15}$$

The roots are distinct if $x_i \neq x_j \forall i, j \in 1 \ldots k$ and in general complex numbers. In the most common case we do not know that the roots are distinct, some may have multiplicity m say. This has the consequence that

$$P_k(x_i) = P_k'(x_i) = P_k''(x_i) = \cdots = P_k^{[m]}(x_i) = 0. \tag{A.16}$$

Factorization of a polynomial can be done according to the formula

$$P_k(x) = \alpha_k(x - x_1)(x - x_2)\ldots(x - x_k) \tag{A.17}$$

If a root is of multiplicity m the factor is repeated since a number of the factors in (A.17) is occuring m times.

Bibliography

[1] Roger Alexander. Design and implementation of DIRK integrators for stiff systems. *Applied Numerical Mathematics*, 46:1–17, 2003.

[2] C. Bendtsen and Per G. Thomsen. Numerical solution of differential algebraic equations. 1998.

[3] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations.* Number 14 in Classics in applied mathematics. Society for Industrial and Applied Mathematics - SIAM, 1996.

[4] J.C. Butcher. An algebraic theory of integration methods. *Math.Comp*, 26:79–106, 1972.

[5] J.C. Butcher. *Numerical Methods for Ordinary Differential Equations.* John Wiley & Sons, 2008.

[6] J.C. Butcher and D: Chen. A new type of singly-implicit runge-kutta methods. *Appl. Numer. Meth.*, 34:179–188, 2000.

[7] Ohanian H. C. *Principles of Physics.* Norton, New York, 1994.

[8] G. Dahlquist. *Stability and error bounds in the numerical integration of ordinary differential equations.* PhD thesis, 1959.

[9] J. R. Dormand. *Numerical methods for differential equations: a computational approach.* CRC Press LLC, 1996.

[10] C. Lubich E. Hairer and M. Roche. *The Numerical Solution of Differential Algebraic Equations by Runge-Kutta Methods.* Lecture Notes in Mathematics. Springer-Verlag, 1989.

[11] E.Hairer and G.Wanner. *Solving Ordinary Differential Equations II*, volume 14 of *Springer series in computational Mathematics.* Springer Verlag, second revised edition, 1996.

[12] E.Hairer, S.P.Nørsett, and G.Wanner. *Solving Ordinary Differential Equations I*, volume 1 of *Springer Series in Computational Mathematics*. Springer-Verlag, 1980.

[13] H. Erlichson. Maximum projectile range with drag and lift, with particular application to golf. *American Journal of Physics*, 54(4):357–362, 1983.

[14] Leonhard Euler. *Mechanica*. 1736.

[15] F.Bashforth and J.C.Adams. *An attempt to test the theories of capillary action by comparing the theoretical and measured forms of drops of fluid.* Cambridge University Press, 1883.

[16] C.W Gear. *Numerical initial value problems in ordinary differential equations*. Prentice Hall., 1971.

[17] C.W. Gear and O. Østerby. Solving ordinary differential equations with discontinuities. *ACM Trans. Math. Softw.*, 10(1):23–44, 1984.

[18] Golf physics: Golf club; lie angle and shaft length. Available as http://kingfish.coastal.edu/physics/projects/2000_Spring/golf/GolfClub.html.

[19] Kjell Gustafsson. *Control of Error and Convergence in ODE Solvers*. PhD thesis, Department of Automatic Control, Lund Institute of Technology, 1992.

[20] P. Henrici. *Discrete Variable Methods in Ordinary Differential Equations*. J.Wiley & sons, 1962.

[21] H. Hertz. Über die Berührung zweier fester elastischer Körper. *Journal für die reine und angewandte Mathematik*, 92:156–171, 1882.

[22] N. Houbak, S.P. Nørsett, and P.G. Thomsen. Displacement or residual test in the application of implicit methods for stiff problems. *IMA Journal of Numerical Analysis*, 5:297 – 305, 1985.

[23] A. Iserles. *Afirst course in the Numerical Analysis of Differential Equations*. Cambridge University Press, 1996.

[24] A. Iserles and S.P. Nørsett. *Order Stars*, volume 2 of *Applied Mathematics and Mathematical Computation*. Chapman & Hall, 1991.

[25] T. P. Jorgensen. *The Physics of Golf*. American Institute of Physics, 1994.

[26] Sengupta T. K. and Talla S. B. Robin–magnus effect: A continuing saga. *Current Science*, 86(7):1033–1036, April 2004. Also available as http://www.ias.ac.in/currsci/apr102004/1033.pdf.

[27] K.F.Askjær. Dali , en differential-algebraisk ligningsløser. thesis, 1986.

[28] A. Kværnø. Singly diagonally implicit runge kutta methods with an explicit first stage. *BIT*, 44:489–502, 2004.

[29] J.D. Lambert. *Computational Methods in Ordinary Differential Equations*. J.Wiley & sons, 1972.

[30] Davies J. M. The aerodynamics of golf balls. *Journal of Applied Physics*, 20(9):821–828, 1949.

[31] R. Mannshardt. One step methods of any order for ode's with discontinuous right hand sides. *Numerische Mathematik*, 31(1), 1978.

[32] J. S. Michael, K. B. Rooney, and R. Smith. The metabolic demands of kayaking: A review. *Journal of Sports Science and Medicine*, 7:1–7, 2008.

[33] F.R. Moulton. *New Methods in exterior ballistics*. University of Chicago, 1926.

[34] Isaac Newton. *Principia Mathematica*. Royal Society, 1687.

[35] R.E. O'Malley. *Introduction to Singular Perturbations*. Academic Press, 1974.

[36] B. Owren and M Zennaro. Derivation of Efficient Continuous Explicit Runge-Kutta Methods. *SIAM J. Sci. Stat. Comput.*, 13: 1488–1501, 1992.

[37] C. Runge. Uber die numerische auflösung von differentialgleichungen. *Math. Ann.*, 1895.

[38] L.F. Shampine and S. Thompson. *Numerial methods for ODE's*. SIAM.

[39] G. Söderlind. The automatic control of numerical integration. *CWI Quaterly*, 11(1):55–74, 1998.

[40] M. Sofroniou. Order stars and linear stability theory. *J. Symbolic Computation*, 21:101–131, 1996.

[41] S.P.Nørsett. C-polynomials for rational approximation to the exponential function. *Numer. Math*, 25:39–56, 1975.

[42] S.P.Nørsett and P.G.Thomsen. Imbedded sdirk-methods of basic order three. *BIT*, 24:634–646, 1984.

[43] E. Sprigings, P. McNair, G. Mawston, C. Sumner, and M. Boocock. A method for personalising the blade size for competitors in flatwater kayaking. *Journal of Sports Engineering*, 9:147–153, 2006.

[44] M. Tabor. *Chaos and integrability in nonlinear dynamics - An introduction*. John Wiley & Sons, 1988.

[45] Per G. Thomsen. A generalised runge kutta method of order three. 2002.

[46] H. True. *Dynamics of Railway Vehicles and Rail/Wheel Contact*, pages 75–128. CISM Courses and Lectures - No. 497. SpringerWienNewYork, 2007.

[47] P.J. Vermeulen and K.L. Johnson. Contact of nonspherical elastic bodies transmitting tangential forces. *Journal of Applied Mechanics*, 31:338–340, 1964.

[48] Gottfried Wilhelm von Leibniz. *Acta Eruditorum*. 1686.

[49] S.P.Nørsett W.A.Enright, K.R.Jackson and P.G.Thomsen. Interpolants for Runge-Kutta formulas. *ACM TOMS*, 12(3):193–218, 1986.

[50] S.P.Nørsett W.A.Enright, K.R.Jackson and P.G.Thomsen. Effective solution of discontinuous ivp's using a runge kutta formula pair with interpolants. *Applied Mathematics and Computation*, 27, 1988.

[51] W.E.Milne. Numerical integration of ordinary differential equations. *Amer. Math. Monthly*, vol.33:455–460, 1926.