Numerical Solutions of Ordinary Differential Equations

(Inital Value Problems)

Doug Curran
University of Sunderland
United Kingdom

András Sövegjártó
Loránd Eötvös University
Hungary

László Szili
Loránd Eötvös University
Hungary

Mária Vicsek
Budapest University of Economic Sciences
Hungary

1997

Contents

Preface

This course is part of the TEMPUS project Nr. S_JEP-07318-94 *European Course in Modelling and Simulation* as it is one of the modules offered to the students.

The course is planned for both undergraduate and postgraduate students who have the proper background in basic calculus, matrix algebra and who are interested in the numerical solution of initial value problems of differential equations. This interest is usually aroused by the fact that almost all dynamic models describing time-dependent phenomena in economy and in the sciences can only be solved by numerical methods.

We shall use the computer algebra package Maple V Release 4 to illustrate the theory. It helps in understanding the material and provides the means for experimenting with the different methods.

Our aim was to construct the material in a self-contained way. For this purpose in Chapter 1 we summarise very briefly the basic concepts and results from the theory of ordinary differential equations and numerical methods.

In Chapter 2 we deal with the numerical solutions of initial value problems using single step methods. Linear multistep methods are analysed in Chapter 3. Chapter 4 treats stiff and delay systems of differential equations.

• Acknowledgements

We acknowledge and greatly appreciate the help of the TEMPUS project Nr. S_JEP-07318-94. The work of the third author has partly been supported by the grant No. MKM 184/1996 of the Hungarian Ministry for Education.

CHAPTER 1

Introduction

1.1. Some concepts and results from the theory of ordinary differential equations

Many problems of applied mathematics lead to differential equations. We begin our study by explaining what a differential equation is. As the two words *differential* and *equation* suggest, a *differential equation*—loosely speaking—is an equation containing derivatives of an unknown function. For example

$$\begin{aligned} \frac{dy}{dx} &= 2(y^2(x) + \sin x), & \frac{d^2y}{dx^2} &= e^{y(x)} + xy'(x), \\ y'(x) &= \sqrt{1 + y^2(x)}, & xy''(x) &= xy'(x) - x^3 + 2 \end{aligned} \tag{1.1}$$

are differential equations with respect to the unknown function $y(x)$. For the unknown two variable function $u(x, y)$ we can consider the following differential equations:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \quad \frac{\partial u}{\partial x} = \frac{\partial^2 u}{\partial y^2}. \tag{1.2}$$

If the unknown function depends only on one independent real variable, then the equation is called an *ordinary differential equation*. The equations in (1.1) are of this kind.

If the unknown function is a function of two or more independent real variables, then it is a *partial differential equation*. For instance, (1.2).

The *order* of a differential equation is the highest order of the derivative entering into the equation. For instance, the equations

$$x\frac{dy}{dx} = y^3(x) + 4\sin xy(x), \quad y'(x) = x^2 \cos y(x)$$

are differential equations of the first-order, and both the equations

$$x\frac{d^2y}{dx^2} = y^3(x) + 4\sin xy(x), \quad y''(x) = x^2 \cos y(x)$$

are differential equations of the second-order.

We shall study only ordinary differential equations.

• First-order differential equations

In the most general case an ordinary differential equation of the first-order contains an independent variable, an unknown function and its derivative and has the form $F(x, y(x), y'(x)) = 0$, where $F(x, y_0, y_1)$ is a given real valued function.

The results given here apply to the case where F is such that the equation $F(x, y_0, y_1) = 0$ can be solved for y_1 in the form $y_1 = f(x, y_0)$. Thus we shall consider differential equations of the form

$$\boxed{y'(x) = f(x, y(x)).} \tag{1.3}$$

Our goal is to find the unknown function $y(x)$ satisfying (1.3), i.e. we want to solve the following problem.

PROBLEM 1.1. *Let I be a fixed interval of the real line and consider the following rectangle or strip on the plane*

$$D := \{(u, v) \in \mathbb{R}^2 \mid u \in I, v \in [c, d], -\infty \le c < d \le +\infty\}.$$

Suppose that f is a real valued function defined on D. Find a differentiable function $y(x)$ defined on a real interval $J \subset I$ such that

$$\begin{aligned} &\text{(i)} \quad (x, y(x)) \in D \qquad (x \in J), \\ &\text{(ii)} \quad y'(x) = f(x, y(x)) \quad (x \in J). \end{aligned}$$

This problem is called an *ordinary differential equation of the first-order*, and is usually written in the form (1.3). If such an interval J and a function $y(x)$ exist, then $y(x)$ is called a *solution of the differential equation* (1.3) *on the interval* J. The graph of a solution of a differential equation is called an *integral curve* of the equation. If there are no such J and $y(x)$, we say that (1.3) has no solution.

In order to understand what is meant by a solution, we give both the equation and its solution, and we verify that it is a solution.

EXAMPLE 1.1. *Verify that the given function is a solution to the corresponding differential equation:*

$$\text{(a)} \ y(x) = \frac{\sin x}{x} \ (x > 0), \quad y'(x) + \frac{y(x)}{x} = \frac{\cos x}{x},$$

$$\text{(b)} \ y(x) = ce^{-2x} + \frac{1}{3}e^x \ (x \in \mathbb{R}), \quad y'(x) + 2y(x) = e^x,$$

where c is a real parameter,

$$\text{(c)} \ y(x) = -x + \frac{2x}{10 - x^2} \ (x \in (-3, 3)),$$
$$xy'(x) - xy^2(x) - (2x^2 + 1)y(x) - x^3 = 0.$$

SOLUTION. (a) Differentiating $y(x)$ we have

$$y'(x) = \frac{x\cos x - \sin x}{x^2} = -\frac{\sin x}{x^2} + \frac{\cos x}{x} = \\ = -\frac{y(x)}{x} + \frac{\cos x}{x} \quad (x > 0).$$

(b) Let c be a fixed number. Differentiating $y(x)$ we obtain

$$y'(x) = -2ce^{-2x} + \frac{1}{3}e^x = -2\big(ce^{-2x} + \frac{1}{3}e^x\big) + e^x = \\ = -2y(x) + e^x \qquad (x \in \mathbb{R}),$$

i.e. the function $y(x)$ is a solution of the given differential equation.

(c) In this case, we illustrate how to use Maple. First we define the given function $y(x)$

```
> y := x -> -x + 2*x/(10-x^2);
```

$$y := x \to -x + 2\,\frac{x}{10 - x^2}$$

Then we use the Maple's **diff** procedure to compute the derivative $y'(x)$, naming the resulting output `d1`

```
> d1 := diff(y(x), x);
```

$$d1 := -1 + \frac{2}{10 - x^2} + 4\,\frac{x^2}{(10 - x^2)^2}$$

Finally, we compute the left hand side of the given equation

```
> x*d1 - x*y(x)^2 - (2*x^2+1)*y(x) - x^3;
```

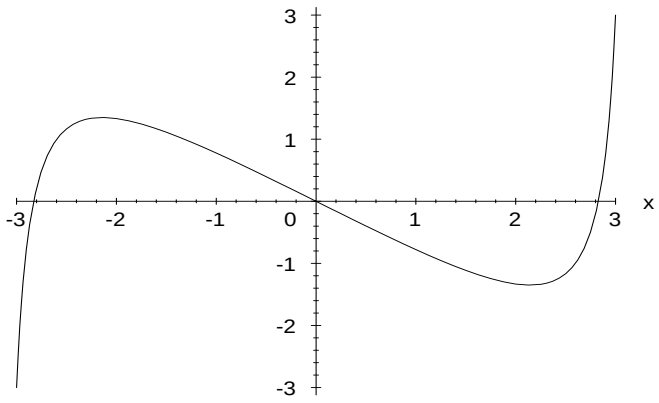
$$x\,(-1 + \frac{2}{10 - x^2} + 4\,\frac{x^2}{(10 - x^2)^2}) - x\,(-x + 2\,\frac{x}{10 - x^2})^2 - \\ (2\,x^2 + 1)\,(-x + 2\,\frac{x}{10 - x^2}) - x^3$$

In this case Maple does not simplify the result automatically. If we use the command **simplify** we obtain

```
> simplify(");
```

To graph $y(x)$ we can use the **plot** procedure. For example, entering

```
> plot(y(x), x=-3..3);
```



graphs $y(x)$ on the interval $[-3, 3]$. □

In geometrical language, (1.3) prescribes a slope $f(u, v)$ at each point $(u, v) \in D$. A solution $y(x)$ on J is a function whose graph has the slope $f(x, y(x))$ for each $x \in J$ (see Figure 1.1).

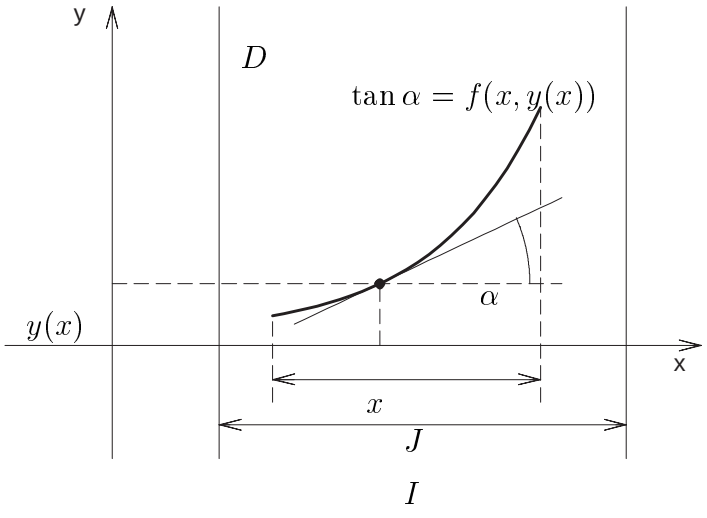


Figure 1.1. Geometrical interpretation of (1.3)

A set of short line segments representing the tangent lines can be constructed for a large number of points. This collection of line segments (or vectors) is known as the *direction field* of the differential equation and provides a great deal of information concerning the behavior of the family of solutions. The direction field associated with an equation can easily be studied using Maple's **DEplot** procedure, which is contained in the **DEtools** package.

EXAMPLE 1.2. *Draw the direction field of the differential equation*

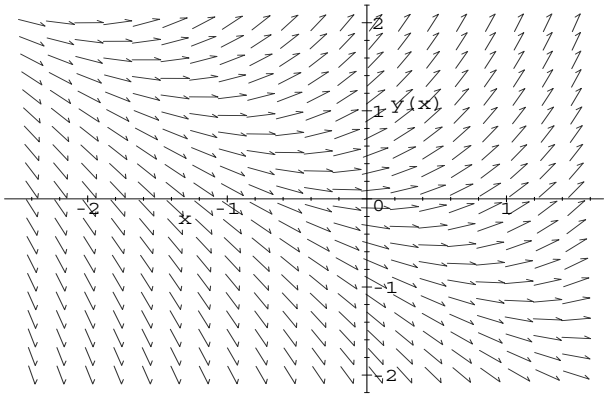
$$y'(x) = x + y(x). \tag{1.4}$$

SOLUTION. First we load the **DEtools** package:

```
> with(DEtools):
```

The next command draws the direction field

```
> DEplot({diff(y(x), x) = x + y(x)}, {y(x)},
          x=-2.4..1.5, y=-2..2, arrows=SMALL);
```



As we see the magnitude of the arrow at a point (u, v) is proportional to the magnitude of $f(u, v) := u + v$. □

Fortunately the **dsolve** function of Maple is able to find the solutions to many differential equations. The next examples illustrate how Maple can be used to solve first-order differential equations.

EXAMPLE 1.3. *Use Maple's* **dsolve** *procedure to find all solutions of the differential equation* (1.4). *Draw the graphs of some solutions.*

SOLUTION. The differential equation (1.4) can be defined as follows

```
> ex1 := diff(y(x), x) = x + y(x);
```

$$ex1 := \frac{\partial}{\partial x}\,\mathrm{y}(x) = x + \mathrm{y}(x)$$

Now, we invoke the **dsolve** function

```
> sol1 := dsolve(ex1, y(x));
```

$$sol1 := \mathrm{y}(x) = -x - 1 + e^{x}\,\_C1$$

Our differential equation has a family of possible solutions, parameterized by the "constant of integration". The **dsolve** function labels this constant $\_C1$.

We would like to check if the answer is correct. (The reader should do this in every case.) Substituting the obtained functions in the equation itself we have

```
> subs(", ex1);
```

$$\frac{\partial}{\partial x}\,(-x - 1 + e^{x}\,\_C1\,) = -1 + e^{x}\,\_C1$$

Evaluating the difference of the two sides of this equation we see that

```
> expand(lhs(") - rhs("));
```

$$0$$

Therefore, for any fixed real number c the function

$$y_c(x) = -x - 1 + ce^{x} \qquad (x \in \mathbb{R}) \tag{1.5}$$

and its restrictions to all the intervals $J \subset \mathbb{R}$ are solutions of (1.3). It can be shown that this differential equation has no other solution. We express this fact so that the *general solution* of (1.3) is (1.5).

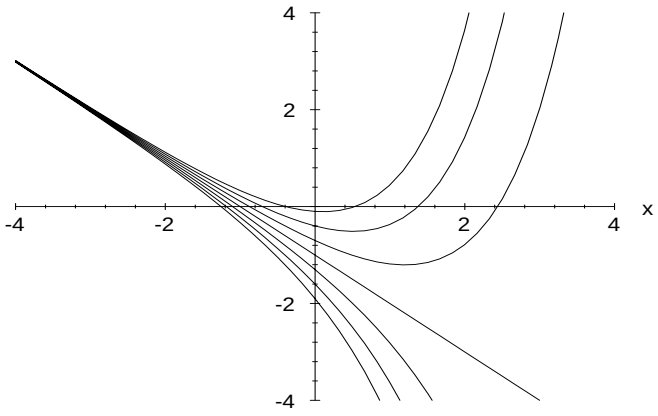
Let us observe that Maple returns an equation for the unknown function. In order to draw the graph of a solution we have to select the expression for the function.

```
> rhs(sol1);
```

$$-x - 1 + e^{x}\,\_C1$$

We use **seq** and **subs** to define the set of seven functions $y_c(x)$ ($c = -0.9, -0.6, \ldots, 0.6, 0.9$). These functions can be plotted for example on the interval $(-4, 4)$ in the following way

```
> plot({seq(subs(_C1=0.3*i, rhs(sol1)), i=-3..3)},
        x=-4..4, -4..4);
```



□

EXAMPLE 1.4. *Find the general solution of the differential equation*

$$y'(x) = -y^2(x) \tag{1.6}$$

on the upper half plane using Maple and check the result.

SOLUTION. Similarly to the previous example we obtain

```
> ex2 := diff(y(x), x) = -y(x)^2;
```

$$ex2 := \frac{\partial}{\partial x}\,\mathrm{y}(x) = -\mathrm{y}(x)^2$$

```
> sol2 := dsolve(ex2, y(x), explicit=true);
```

$$sol2 := \mathrm{y}(x) = \frac{1}{x + \_C1}$$

The `explicit=true` optional equation forces the solution to be returned explicitly in terms of the dependent variable. The default value is `explicit=false`. In this case Maple is content with giving the solution in an implicit form.

Now we check the result in the following way:

```
> subs(", ex2):
  expand(lhs(") - rhs("));
```

$$0$$

Thus, the general solution of our differential equation on the upper half plane is

$$y(x) = \frac{1}{x+c},$$

where c is an arbitrary real number. This means that for any fixed real number c the functions

$$y_c(x) = \frac{1}{x+c} \qquad (x \in (-c, +\infty)),$$

$$y_c(x) = \frac{1}{x+c} \qquad (x \in (-\infty, -c)),$$

and their restrictions to an appropriate interval $J \subset \mathbb{R}$ and only these are solutions of (1.6). □

The above examples show that the problem (1.3) may have many solutions. However, in many cases there exists only one solution passing through a point and existing on a maximal interval, as illustrated in the following examples.

EXAMPLE 1.5. *Find the solution $y(x)$ of (1.4) passing through the point $(1, 2)$, i.e. satisfying the condition $y(1) = 2$.*

SOLUTION. From EXAMPLE 1.3 we know that the general solution of (1.4) is

$$y(x) = ce^x - x - 1 \qquad (x \in \mathbb{R}),$$

where, c is an arbitrary real parameter. We have to find a real number c for which

$$2 = y(1) = ce^2 - 1 - 1.$$

This number c can be obtained in the following way

```
> c := solve(subs(x=1, y(1)=2, sol1), _C1);
```

$$c := \frac{4}{e}$$

thus the asked solution is

```
> simplify(subs(_C1=c, sol1));
```

$$\mathrm{y}(x) = -x - 1 + 4\,e^{(x-1)}$$

which is defined on the whole real line. It is easy to prove that this is the unique solution of (1.4) on $\mathbb{R}$ satisfying the condition $y(1) = 2$. □

EXAMPLE 1.6. *Find the solution $y(x)$ of (1.6) passing through the point $(1, 2)$, i.e. satisfying the condition $y(1) = 2$.*

SOLUTION. We can solve this problem similarly to the previous example, but fortunately in the **dsolve** procedure we can immediately specify an initial condition.

The differential equation (1.6) is defined in our Maple's variable `ex2`. Now we give the initial condition

```
> in_cond := y(1) = 2;
```

$$in\_cond \;:=\; \mathrm{y}(1) = 2$$

For the solution the **dsolve** function have to use in the following form:

```
> dsolve(ex2, in_cond, y(x));
```

$$\mathrm{y}(x) = \frac{1}{x - \frac{1}{2}}$$

It is clear that the maximal interval on which this solution may be defined is the interval $(1/2, +\infty)$ and there isn't any other solution of our problem on this interval. □

Therefore, in order to be able to talk about uniqueness of solutions of (1.3), one is led to the problem of finding a solution passing through a given point of the strip D.

Suppose that (τ, ξ) is a given point in D. Then an *initial value problem* associated with (1.3) and this point is defined in the following way.

PROBLEM 1.2. *Find a solution $y(x)$ of (1.3) satisfying the condition $y(\tau) = \xi$.*

This problem is denoted by

$$\boxed{y'(x) = f(x, y(x)), \qquad y(\tau) = \xi.} \tag{1.7}$$

• Basic questions for initial value problems

1. The first question to be answered is under what conditions on f can we say that the problem (1.7) has at least one solution. (*The problem of existence of solution.*)

The following theorem (see, e.g. [**CL**], Theorem 1.2) lays down a sufficient condition for a solution to exist.

THEOREM 1.1 (Cauchy–Peano existence theorem). *If f is a continuous function on the strip D then there exists a solution of the initial value problem* (1.7).

2. The second question is *the problem of uniqueness of solutions.* For example we know that the initial value problem

$$y'(x) = x + y(x), \qquad y(1) = 2$$

has a unique solution on the whole real line. EXAMPLE 1.6 shows that the initial value problem

$$y'(x) = -y^2(x), \qquad y(1) = 2$$

has a unique solution on the interval $(1/2, +\infty)$ and this is the maximal interval in which a solution exists.

Not all problems possess a unique solution. The following example shows that something more than the continuity of f in (1.7) is required in order to guarantee that a solution passing through a given point be a unique solution on a maximal interval.

EXAMPLE 1.7. *Let (τ, ξ) be a fixed point of the plane. Find all solutions of the initial value problem*

$$y'(x) = 2\sqrt{|y(x)|}, \qquad y(\tau) = \xi. \tag{1.8}$$

SOLUTION. Assume that $\xi > 0$ and consider the differential equation

```
> ex3:=diff(y(x), x)=2*sqrt(abs(y(x)));
```

$$ex3 := \frac{\partial}{\partial x}\,\mathrm{y}(x) = 2\,\sqrt{|\mathrm{y}(x)|}$$

The **dsolve** function gives the general solution of the above equation in implicit form

```
> sol3 := dsolve(ex3, y(x));
```

$$sol3 := -\frac{\mathrm{y}(x)}{\sqrt{|\mathrm{y}(x)|}} + x = \_C1$$

We can solve this equation on the upper half-plane ($S^+ := \mathbb{R} \times \mathbb{R}^+$) in the following way:

```
> assume(y(x)>0);
> sol3u := factor(dsolve(ex3, y(x), explicit=true));
```

$$sol3u := \mathrm{y}(x\tilde{}) = (-x\tilde{} + \_C1)^2$$

The appended tilde ˜ indicates that the variable x carries assumption. Therefore on S^+ we obtain the following solutions

$$y_{c_1}(x) := (x - c_1)^2 \qquad (x \in (c_1, +\infty)),$$

where c_1 is an arbitrary real number. The integral curve passes through the point (τ, ξ) if

```
> c1 := solve(subs(x=tau, y(tau)=xi, sol3u), _C1)[2];
```

$$c1 := \tau - \sqrt{\xi}$$

On the lower half-plane ($S^- := \mathbb{R} \times \mathbb{R}^-$) we have

```
> assume(y(x)<0);
> sol3l := factor(dsolve(ex3, y(x), explicit=true));
```

$$sol3l := \mathrm{y}(x\tilde{}) = -(x\tilde{} - \_C1)^2$$

Thus the general solution on S^- can be written in the following form

$$y_{c_2}(x) := -(x - c_2)^2 \qquad (x \in (-\infty, c_2)),$$

where c_2 is an arbitrary real constant.

It is clear that the zero function is also a solution of the differential equation $y'(x) = 2\sqrt{|y(x)|}$.

These results mean that the initial value problem (1.8) has infinitely many solutions: for example if $\xi > 0$ then for every fixed number $c \le \tau - \sqrt{\xi}$ the function

$$y_c(x) := \begin{cases} (x - \tau + \sqrt{\xi})^2 & \text{if } x > \tau - \sqrt{\xi}, \\ 0 & \text{if } c < x \le \tau - \sqrt{\xi}, \\ -(x - c)^2 & \text{if } x < c \end{cases}$$

is a solution on the whole real line (see Figure 1.2).

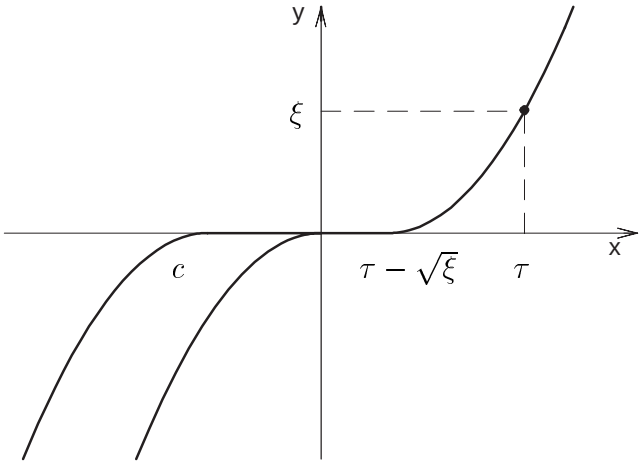


Figure 1.2. Different solutions of (1.8)

We can solve the problem for $\xi \leq 0$ in a similar way. □

A simple condition which permits one to imply uniqueness is the Lipschitz condition.

DEFINITION 1.1. *Suppose f is defined on a strip D of the plane. If there exists a constant $L > 0$ such that for every (u, v_1) and (u, v_2) in D*

$$|f(u, v_1) - f(u, v_2)| \leq L|v_1 - v_2|$$

then f is said to satisfy a ***Lipschitz condition*** *(with respect to the second variable of f) in D. The constant L is called the* ***Lipschitz constant****.*

The following fundamental existence and uniqueness theorem for the initial value problem given in (1.7) states that the problem (1.7) has exactly one solution, provided f satisfies a Lipschitz condition (see, e.g. [**He**], Theorem 3.1).

THEOREM 1.2 (Picard–Lindelöf theorem). *Let f be a continuous function defined on the strip $D := \{(u, v) \mid a \leq u \leq b,\ v \in \mathbb{R}\}$, where a, b are finite real numbers. Suppose that f satisfies a Lipschitz condition on D. Then for every $\tau \in [a, b]$ and every $\xi \in \mathbb{R}$ there exists exactly one function $y(x)$ such that*

(i) *$y(x)$ is differentiable for $x \in [a, b]$,*
(ii) *$y'(x) = f(x, y(x))$ for $x \in [a, b]$,*
(iii) *$y(\tau) = \xi$.*

From the mean value theorem it easily follows that the Lipschitz condition is satisfied if the partial derivative $\partial f/\partial y$ exists on the strip D and it is continuous and bounded there.

3. *Explicit representation of solutions.* It turnes out that given an *arbitrary* differential equation, constructing a closed-form solution is nearly always impossible. For example, the simple differential equation

$$y'(x) = x + x^2 + y^2(x)$$

does not possess a closed-form solution in terms of elementary functions, although its solutions can be expressed in a complicated way in terms of Bessel functions of fractional order. We remark that Kamke's book [**Ka**] contains those special types of differential equations that may be solved in terms of elementary functions using a finite number of arithmetic operations. Consequently, although mathematicians were first concerned with finding closed-form solutions to differential equations, after realizing that these type of solutions were usually impossible to construct, mathematicians have since (frequently) turned their attention to addressing properties of the solution without actually finding them and seeking algorithms to approximate the solution.

• Systems of ordinary differential equations

Many problems in practice are modelled with more than one equation and involve more than one unknown functions. For example, if we want to determine the population of two interacting populations such as foxes and rabbits, we would have two functions to represent the quantities of two populations where these populations depend on one independent variable that represents time.

More precisely, one can formulate the following problem.

PROBLEM 1.3. *Suppose n is a positive integer and $f_1, f_2, \ldots, f_n$ are n real valued continuous functions defined on some strip D of the real $(n+1)$-dimensional Euclidean space. Find n differentiable functions $y_1(x), y_2(x), \ldots, y_n(x)$ defined on a real interval J such that*

$$\begin{aligned} &\text{(i)} \quad (x, y_1(x), \ldots, y_n(x)) \in D \qquad (x \in J), \\ &\text{(ii)} \quad y_i'(x) = f_i(x, y_1(x), \ldots, y_n(x)) \quad (x \in J). \end{aligned}$$

This problem is called a *system of n ordinary differential equations of the first-order*, and is denoted by

$$y_i'(x) = f_i(x, y_1(x) \ldots, y_n(x)) \qquad (i = 1, 2, \ldots, n). \tag{1.9}$$

Correspondingly, if such an interval J and functions $y_1(x), \ldots, y_n(x)$ exist, then the set of functions $(y_1(x), \ldots, y_n(x))$ is called a *solution of the system* (1.9) *on the interval J.*

Let $(\tau, \xi_1, \ldots, \xi_n) \in D$. The *initial value problem* consists of finding a solution $(y_1(x), \ldots, y_n(x))$ of (1.9) on an interval J containing τ such that $y_i(\tau) = \xi_i$ $(i = 1, 2, \ldots, n)$.

Such initial value problems can be written analogously to (1.7) in vector form

$$\boxed{\mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y}(x)), \qquad \mathbf{y}(\tau) = \boldsymbol{\xi},} \tag{1.10}$$

where

$$\mathbf{y}(x) := \begin{bmatrix} y_1(x) \\ y_2(x) \\ \vdots \\ y_n(x) \end{bmatrix}, \qquad \mathbf{f} := \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}, \qquad \boldsymbol{\xi} := \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{bmatrix}$$

and $(\tau, \boldsymbol{\xi}) \in D$ is a given point.

A system of n differentiable functions

$$y_i(x, c_1, c_2, \ldots, c_n), \quad (i = 1, 2, \ldots, n) \tag{1.11}$$

of the independent variable x, and n arbitrary constants $c_1, c_2, \ldots, c_n$ is said to be *the general solution* of the system (1.9) if

(i) for any values of $c_1, c_2, \ldots, c_n$ the system of functions (1.11) are solutions of (1.9),

(ii) the solution of any initial value problem related to (1.9) can be obtained from (1.11) by appropriately choosing $c_1, c_2, \ldots, c_n$.

It turns out that the basic questions and results for the case $n = 1$ can be carried over successfully to the system (1.10). In terms of the definitions introduced above, the theorems 1.1 and 1.2 are valid for the vector equation (1.10) if, in their statements y, f are replaced by the vectors $\mathbf{y}, \mathbf{f}$ and the magnitude is understood for vectors as:

$$\|\mathbf{y}\| := \left(\sum_{i=1}^{n} |y_i|^2 \right)^{1/2} \qquad (\mathbf{y} \in \mathbb{R}^n).$$

A particularly interesting system is the *linear system*

$$\begin{aligned} y_1'(x) &= a_{11}(x)y_1(x) + \cdots + a_{1n}(x)y_n(x) + h_1(x), \\ y_2'(x) &= a_{21}(x)y_1(x) + \cdots + a_{2n}(x)y_n(x) + h_2(x), \\ &\vdots \\ y_n'(x) &= a_{n1}(x)y_1(x) + \cdots + a_{nn}(x)y_n(x) + h_n(x), \end{aligned} \tag{1.12}$$

where the functions a_{ij} and h_i are real valued continuous functions on some bounded interval $I \subset \mathbb{R}$.

Using the notations

$$\mathbf{y}(x) := \begin{bmatrix} y_1(x) \\ y_2(x) \\ \vdots \\ y_n(x) \end{bmatrix}, \mathbf{A}(x) := \begin{bmatrix} a_{11}(x) & \dots & a_{1n}(x) \\ a_{21}(x) & \dots & a_{2n}(x) \\ \vdots & & \\ a_{n1}(x) & \dots & a_{nn}(x) \end{bmatrix}, \mathbf{h}(x) := \begin{bmatrix} h_1(x) \\ h_2(x) \\ \vdots \\ h_n(x) \end{bmatrix}$$

the system (1.12) can be written in vector form:

$$\boxed{\mathbf{y}'(x) = \mathbf{A}(x)\mathbf{y}(x) + \mathbf{h}(x).} \tag{1.13}$$

From the above mentioned results it follows that for every initial value $(\tau, \boldsymbol{\xi}) \in I \times \mathbb{R}^n$ the equation (1.13) has exactly one solution $\mathbf{y}(x)$ on the whole interval I satisfying the condition $\mathbf{y}(\tau) = \boldsymbol{\xi}$.

Let $\mathbf{A}(x)$ be independent of x, i.e. we have a *linear system with constant coefficients*. In this case there is a general method (see, e.g. [**CL**]) to compute the solution of the corresponding initial value problem.

Fortunately many systems of linear differential equations with constant coefficient can also be solved with the Maple's **dsolve** procedure. We illustrate this in the following examples.

EXAMPLE 1.8. *Use Maple's* **dsolve** *procedure to find the general solution of the system*

$$\begin{aligned} y_1'(x) &= y_1(x) - y_2(x), \\ y_2'(x) &= y_2(x) - 4y_1(x). \end{aligned} \tag{1.14}$$

SOLUTION. First we define the system in Maple

```
> sys1 := diff(y1(x), x) = y1(x) - y2(x),
          diff(y2(x), x) = y2(x) - 4*y1(x);
```

$$sys1 := \frac{\partial}{\partial x}\,\mathrm{y1}(x) = \mathrm{y1}(x) - \mathrm{y2}(x),$$
$$\frac{\partial}{\partial x}\,\mathrm{y2}(x) = \mathrm{y2}(x) - 4\,\mathrm{y1}(x)$$

For systems we have to use the **dsolve** function in this form:

```
> dsolve({sys1}, {y1(x), y2(x)});
```

$$\begin{aligned} \{\mathrm{y1}(x) &= \frac{1}{2}\,\_C1\, e^{(-x)} + \frac{1}{2}\,\_C1\, e^{(3\,x)} - \frac{1}{4}\,\_C2\, e^{(3\,x)} + \frac{1}{4}\,\_C2\, e^{(-x)}, \\ \mathrm{y2}(x) &= -\_C1\, e^{(3\,x)} + \_C1\, e^{(-x)} + \frac{1}{2}\,\_C2\, e^{(-x)} + \frac{1}{2}\,\_C2\, e^{(3\,x)}\} \end{aligned}$$

By using the **collect** procedure we can collect together those terms containing e^{-x} and e^{3x}:

```
> collect(", {exp(-x), exp(3*x)});
```

$$\begin{aligned}\{\mathrm{y1}(x) &= (\frac{1}{2}\_C1 - \frac{1}{4}\_C2)\, e^{(3\,x)} + (\frac{1}{2}\_C1 + \frac{1}{4}\_C2)\, e^{(-x)},\\ \mathrm{y2}(x) &= (-\_C1 + \frac{1}{2}\_C2)\, e^{(3\,x)} + (\_C1 + \frac{1}{2}\_C2)\, e^{(-x)}\}\end{aligned}$$

We leave it to the reader to verify that this is the general solution of (1.14), indeed. □

EXAMPLE 1.9. *Use Maple's* **dsolve** *procedure to find the solution of the initial value problem*

$$\begin{aligned} y_1'(x) &= -5y_1(x) + 3y_2(x) + e^{-2x}, \qquad & y_1(0) &= 1,\\ y_2'(x) &= 2y_1(x) - 10y_2(x) + 1, & y_2(0) &= 3, \end{aligned} \tag{1.15}$$

SOLUTION. Similarly to the previous example we have

```
> sys2 := diff(y1(x),x) = -5*y1(x)+3*y2(x)+exp(-2*x),
          diff(y2(x),x) = 2*y1(x)-10*y2(x)+1;
```

$$\begin{aligned} sys1 := &\frac{\partial}{\partial x}\,\mathrm{y1}(x) = -5\,\mathrm{y1}(x) + 3\,\mathrm{y2}(x) + e^{(-2\,x)},\\ &\frac{\partial}{\partial x}\,\mathrm{y2}(x) = 2\,\mathrm{y1}(x) - 10\,\mathrm{y2}(x) + 1 \end{aligned}$$

```
> init_cond := y1(0)=1, y2(0)=3;
```

$$init\_cond := \mathrm{y1}(0) = 1,\ \mathrm{y2}(0) = 3$$

```
> funcs := {y1(x), y2(x)};
```

$$funcs := \{\mathrm{y1}(x),\ \mathrm{y2}(x)\}$$

```
> dsolve({sys2, init_cond}, funcs);
```

$$\begin{aligned}\{\mathrm{y2}(x) &= \frac{1552}{693}\, e^{(-11\,x)} + \frac{15}{28}\, e^{(-4\,x)} + \frac{5}{44} + \frac{1}{9}\, e^{(-2\,x)},\\ \mathrm{y1}(x) &= \frac{45}{28}\, e^{(-4\,x)} - \frac{776}{693}\, e^{(-11\,x)} + \frac{3}{44} + \frac{4}{9}\, e^{(-2\,x)}\}\end{aligned}$$

It is easy to verify that the obtained function is the unique solution of the initial value problem (1.15). □

• Higher-order differential equations

Many physical situations exist that need to be modeled by higher-order differential equations.

More precisely, one can formulate the following problem.

PROBLEM 1.4. *Suppose f is a real valued continuous function defined on some strip D of the real $(n+1)$-dimensional Euclidean space. Find a function $y(x)$ defined on a real interval J possessing n derivatives there such that*

$$\begin{aligned} &\text{(i)} \quad (x, y(x), y'(x), \ldots, y^{(n-1)}(x)) \in D \qquad (x \in J), \\ &\text{(ii)} \quad y^{(n)}(x) = f(x, y(x), y'(x), \ldots, y^{(n-1)}(x)) \quad (x \in J). \end{aligned}$$

This problem is called the *nth-order differential equation* associated with f, and is denoted by

$$y^{(n)}(x) = f(x, y(x), y'(x), \ldots, y^{(n-1)}(x)). \tag{1.16}$$

If such an interval J and a function $y(x)$ exist, then $y(x)$ is said to be a *solution of* (1.16) *on the interval* J.

Let $(\tau, \xi_1, \ldots, \xi_n) \in D$. The *initial value problem* consists of finding a solution $y(x)$ of (1.16) on an interval J containing τ such that

$$y(\tau) = \xi_1, \ y'(\tau) = \xi_2, \ \ldots, \ y^{(n-1)}(\tau) = \xi_n. \tag{1.17}$$

The equation (1.16) can always be transformed into an equivalent system of first-order differential equations. For this purpose one usually sets

$$\begin{aligned} z_1(x) &:= y(x), \\ z_2(x) &:= y'(x), \\ &\vdots \\ z_n(x) &:= y^{(n-1)}(x), \end{aligned}$$

so that equation (1.16) becomes

$$\mathbf{z}'(x) = \begin{bmatrix} z_1'(x) \\ z_2'(x) \\ \vdots \\ z_{n-1}'(x) \\ z_n'(x) \end{bmatrix} = \begin{bmatrix} z_2(x) \\ z_3(x) \\ \vdots \\ z_n(x) \\ f(x, z_1(x), \ldots, z_n(x)) \end{bmatrix} \tag{1.18}$$

This system is called the *system associated with the nth-order equation* (1.16).

From the above mentioned fact it follows that the theory of equation (1.16) can be reduced to the theory of a system of n first-order differential equations. It is thus clear that all statements about the system (1.18) carry over directly to the statements about the nth-order equation (1.16).

The general solution of the nth-order differential equation (1.16) is the set of all of its solutions defined by a formula $y(x, c_1, c_2, \ldots, c_n)$ containing n arbitrary constants $c_1, c_2, \ldots, c_n$ such that, if the initial conditions (1.17) are given, values $\bar{c}_1, \bar{c}_2, \ldots, \bar{c}_n$ can be found such that $y(x, \bar{c}_1, \bar{c}_2, \ldots, \bar{c}_n)$ is a solution of equation (1.16) satisfying these initial conditions.

A particularly interesting higher-order differential equation is the *nth-order linear differential equation*:

$$y^{(n)}(x) + a_{n-1}(x)y^{(n-1)}(x) + \cdots + a_1(x)y'(x) + a_0(x)y(x) = h(x),$$

where the functions a_i $(i = 0, 1, \ldots, n-1)$ and h are real valued continuous functions on some interval $I \subset \mathbb{R}$.

From the above mentioned results it follows that if the functions $a_0, a_1, \ldots, a_{n-1}, h$ are continuous on an interval $I \subset \mathbb{R}$ then for every initial value $\tau \in I$ and $\xi_i \in \mathbb{R}$ $(i = 1, 2, \ldots, n)$ the initial value problem

$$y^{(n)}(x) + a_{n-1}(x)y^{(n-1)}(x) + \cdots + a_1(x)y'(x) + a_0(x)y(x) = h(x),$$
$$y(\tau) = \xi_1,\ y'(\tau) = \xi_2,\ \ldots,\ y^{(n-1)}(\tau) = \xi_{n-1}.$$

has a unique solution on the interval I.

We can use the Maple's **dsolve** procedure to compute solutions of many higher-order differential equations. We illustrate the possibilities on the following examples.

EXAMPLE 1.10. *Find the general solution of the equation*

$$xy^{(5)}(x) - y^{(4)}(x) = 0. \tag{1.19}$$

SOLUTION. We use the **diff** function with `$` to represent the higher-order derivatives of $y(x)$.

```
> dsolve(x*diff(y(x), x$5)-diff(y(x), x$4) = 0, y(x));
```

$$\mathrm{y}(x) = \_C1 + \_C2\, x + \_C3\, x^2 + \_C4\, x^3 + \_C5\, x^5$$

Let us observe that the general solution of this fifth-order equation contains five independent parameters $\_C1, \ldots, \_C5$. □

EXAMPLE 1.11. *Solve the following initial value problem*

$$2y''(x) = \frac{y'(x)}{x} + \frac{x^2}{y'(x)} \qquad y(1) = \frac{\sqrt{2}}{5},\ y'(1) = \frac{\sqrt{2}}{2}$$

SOLUTION. In Maple we can define a differential equation with the procedure **D**, too.

```
> eq := 2*(D@@2)(y)(x) = D(y)(x)/x+ x^2/D(y)(x);
```

$$eq := 2\,(D^{(2)})(y)(x) = \frac{\mathrm{D}(y)(x)}{x} + \frac{x^2}{\mathrm{D}(y)(x)}$$

```
> init_cond := y(1) = sqrt(2)/5, D(y)(1)=sqrt(2)/2;
```

$$init\_cond := \mathrm{y}(1) = \frac{1}{5}\sqrt{2},\ \mathrm{D}(y)(1) = \frac{1}{2}\sqrt{2}$$

```
> simplify(dsolve({eq, init_cond}, y(x)));
```

$$\mathrm{y}(x) = \frac{1}{5}\,\frac{\sqrt{2}\,x^4}{\sqrt{x^3}}$$

It is easy to show that this function is the unique solution on the interval $(0, +\infty)$. □

Besides initial value problems, *boundary value problems* also frequently occur in practice. Here, the desired solution $y(x)$ of the differential equation (1.16) has to satisfy a boundary condition of the form

$$r(y(a), y(b)) = 0,$$

where $a \neq b$ are two different point of the interval I and r is a given two variable function.

We will discuss methods for the solution of initial value problems only, since the class of the methods for boundary value problems is different.

We only give a single example to illustrate that Maple's **dsolve** procedure can be used to solve boundary value problems, too.

EXAMPLE 1.12. *Find the solution of the boundary value problem*

$$y''(x) - 9y(x) = e^{2x}, \qquad y(0) = 1,\ y(1) = 2.$$

SOLUTION. As in the previous examples we have

```
> eq1 := (D@@2)(y)(x) - 9*y(x)=exp(2*x);
```

$$eq1 := (D^{(2)})(y)(x) - 9\,\mathrm{y}(x) = e^{(2\,x)}$$

```
> bound_cond := y(0)=1, y(1)=2;
```

$$bound\_cond := \mathrm{y}(0) = 1,\ \mathrm{y}(1) = 2$$

```
> dsolve({eq1,bound_cond}, y(x));
```

$$\mathrm{y}(x) = -\frac{1}{5}\,e^{(2\,x)} + \frac{1}{5}\,\frac{(6\,e^{(-3)} - e^2 - 10)\,e^{(3\,x)}}{-e^3 + e^{(-3)}} - \\ -\frac{1}{5}\,\frac{(-e^2 + 6\,e^3 - 10)\,e^{(-3\,x)}}{-e^3 + e^{(-3)}}$$

We leave it to the reader to verify that this function is the unique solution on the whole real line. □

1.2. Necessity and a classification of the approximate solutions of initial value problems

As we have already mentioned, explicit solutions of initial value problems can only be found in relatively few cases. In Section 1.1 we have seen some of these problems.

Even with quite simple differential equations it may happen that their solutions cannot be expressed in closed form. For example, the differential equation

$$y'(x) = x^2 + y^2(x) \tag{1.20}$$

does not possess a closed form solution in terms of elementary functions.

In Maple the usual elementary functions like the exponential function, the natural logarithm, the trigonometric functions are present and the program can solve many ordinary differential equations exactly in explicit or in implicit form. Maple also includes all the commonly used special functions of applied mathematics, for example orthogonal polynomials, Bessel functions, Gamma function, etc. The complete list of the available built-in mathematical functions can be obtained by the command **?inifcns** (help about **ini**tially known **fun**ctio**ns**).

Maple's **dsolve** procedure tries to find the solutions of a differential equation in terms of built-in mathematical functions. For example, the general solution of the differential equation (1.20) can be expressed in terms of Bessel functions and Maple can find it.

```
> dsolve(diff(y(x), x) = x^2 + y(x)^2, y(x));
```

$$\mathrm{y}(x) = -\frac{x\,(\_C1\ \mathrm{BesselY}(\frac{-3}{4},\ \frac{1}{2}\,x^2) + \mathrm{BesselJ}(\frac{-3}{4},\ \frac{1}{2}\,x^2))}{\_C1\ \mathrm{BesselY}(\frac{1}{4},\ \frac{1}{2}\,x^2) + \mathrm{BesselJ}(\frac{1}{4},\ \frac{1}{2}\,x^2)}$$

If we cannot construct an expression in a closed form for the solution of an initial value problem, then we can give an approximation of the exact solution. The methods providing approximations can be divided into two groups according to the form of the representation of the solution.

- *Analytic methods* which give an approximate solution of a differential equation in the form of an analytic expression, like a polynomial.
- *Discrete methods* or *numerical methods* which give an approximation of the exact solution only at discrete points of its domain.

We shall investigate numerical methods in later chapters. For the first group of methods we mention the *power series method* (see, e.g. [**AB**], Chapter 6) and the *method of successive approximations* or *Picard's method*. This method was first used to prove the existence and uniqueness of the solution of initial value problems (see Theorem 1.2).

Consider the initial value problem (1.7) and assume that the condition of the Theorem 1.2 are satisfied. The method of successive approximations consists in that the solution of (1.7) is obtained as the limit of a sequence of functions $y_n(x)$, which are found by the recursive formula

$$\begin{aligned} y_0(x) &= \xi, \\ y_{n+1}(x) &= \xi + \int_\tau^x f(s, y_{n-1}(s)ds \quad (n = 0, 1, 2, \dots). \end{aligned} \tag{1.21}$$

It is proved (see, e.g. [**CL**]) that in a certain interval, which contains the point τ the sequence (1.21) converges uniformly to the unique solution of the initial value problem (1.7). This method gives a global approximation, i.e. gives an approximation on a suitable *interval*.

The next example illustrates how Maple can be used to calculate the sequence of the successive approximations.

EXAMPLE 1.13. *Use Picard's method to find the first few terms of* (1.21) *for the initial value problem*

$$y'(x) = x^2 + y^2(x), \quad y(0) = 0.$$

SOLUTION. First we define the right-hand side of the differential equation

```
> f := (x, y) -> x^2 + y^2;
```

$$f := (x,\, y) \to x^2 + y^2$$

The initial condition is

```
> tau := 0;
```

$$\tau := 0$$

```
> xi := 0;
```

$$\xi := 0$$

A simple program which shall compute the recursive sequence (1.21) is this

```
> y := proc(n)
       local ww;
       option remember;
        ww := unapply(y(n-1), x);
          xi + int(f(s, ww(s)), s=tau..x);
       end:
  y(0) := xi:
```

Note that in defining the recursively defined function `y`, we take advantage of the option `remember`. This instructs Maple to remember the values of `y` computed, and thus, when computing `y(n)`, Maple need not recompute `y(n-1)`.

We obtain – for example – the first four terms of (1.21) in the following way:

```
> for i from 0 to 3 do y(i) od;
```

$$0$$

$$\frac{1}{3}\,x^3$$

$$\frac{1}{3}\,x^3+\frac{1}{63}\,x^7$$

$$\frac{1}{3}\,x^3+\frac{1}{59535}\,x^{15}+\frac{2}{2079}\,x^{11}+\frac{1}{63}\,x^7$$

The question how close these functions are to the exact solution will not be treated here (cf. [**CL**]). □

1.3. Basic problems of approximate methods

As we mentioned in the previous sections it is often impossible to find the exact solution of an initial value problem using standard techniques. In fact, there are very few problems for which an exact solution can be determined with elementary functions using a finite number of arithmetic operations. Therefore, we need numerical methods to approximate the solutions.

Similar problems arise in other topics of mathematics, too. For example, there are formulas for solving quadratic, cubic and quartic polynomial equations but no such formula exists for polynomial equations of degree greater than four or even for a single equation such as

$$x = \tan x.$$

Let us consider another example from calculus. Suppose we want to compute the definite integral

$$\int_a^b f(x)dx.$$

The Fundamental Theorem of Calculus states that if $F(x)$ is any antiderivative of $f(x)$, then

$$\int_a^b f(x)dx = F(b) - F(a).$$

Therefore, to evaluate a definite integral by means of this theorem it is necessary to find an antiderivative of the function $f(x)$. If an antiderivative cannot be found, then numerical methods may be used to approximate the integral to any degree of accuracy.

Even if an exact solution of a given problem can be found it may be of more theoretical than practical use. For example, it is well known that there exists a complicated explicit formula for the roots of a third degree polynomial equation which is not used in practice.

The objective of *numerical analysis* is to construct and analyze numerical methods and algorithms for the solution of problems in science and technology.

The goal of the rest of this section is to present some general problems which arise in the process of a numerical solution. For further reading on general principles and problems of numerical analysis we suggest the Chapters 1 and 2 of the book of G. Dahlquist and Å. Björk [**DB**].

• Numerical algorithm

A *numerical algorithm* consists of a sequence of arithmetic and logical operations which produces an approximate solution to within any prescribed accuracy. An algorithm can be described loosely or in great detail. A comprehensive description is obtained when an algorithm is formulated using a programming language.

For a given problem one can consider different algorithms. These may give approximate answers which have widely varying accuracy.

• Sources of error

Numerical results are influenced by many types of errors. Some sources of error are difficult to influence; others can be reduced or even eliminated. Errors propagate from their source to quantities computed later, sometimes with a considerable amplification or damping.

We distinguish the following types of errors.

A. Errors in given input data. The input data can be the result of measurements. Rounding errors occur, for example whenever an irrational number is rounded to a fixed number of decimals.

B. Round-off errors during the computations. The limited word-length in a calculating device leads to a loss of information.

C. Truncation errors. Such errors result from replacing a desired mathematical operation by a realizable computation. These errors are committed when a limiting process is truncated. For example, if an infinite series is broken off after a finite number of terms.

D. Simplification in the mathematical model. In most of the applications of mathematics, one makes idealizations. For a calculation in economics, for example, one might assume that the rate of interest is constant over a given period of time.

E. "Human" errors and machine errors. When one uses computers, one can expect errors in the program itself, typing errors in entering data, operator errors and (more seldom) machine errors.

The effects of the errors are usually difficult to estimate. If we do not proceed carefully it may well happen that the computed approximations have very little to do with the desired solution, or may even be meaningless.

Errors of type A. and D. are usually to be considered uncontrollable in the numerical treatment. Errors of type B. and C. are controllable.

Whereas round-off error and its properties are somewhat independent of application area, truncation error can only be analyzed in the context in which it occurs.

We illustrate the effect of these errors by the next simple example. Recall from calculus that the derivative of a function $f(x)$ at a point x_0 is defined by the formula

$$\lim_{h\to 0} \frac{f(x_0+h)-f(x_0)}{h} := f'(x_0).$$

We can approximate the derivative $f'(x_0)$ by

$$Df(h) := \frac{f(x_0+h)-f(x_0)}{h} \approx f'(x_0)$$

for a given h. The truncation error associated with using the "realizable" arithmetic formula $Df(h)$ for approximating an unrealizable limiting operation, of the derivative is

$$f'(x_0) - Df(h).$$

Such an error would occur even if $f(x)$ and $Df(h)$ could be evaluated exactly. This error, then, is of mathematical origin. It is clear that if the magnitude of h is "large", $Df(h)$ is inaccurate because h is not sufficiently close to the limit. For reasons to be discussed later, as h becomes small, inaccuracies due to round-off error dominate. The error in computing the difference $f(x_0+h)-f(x_0)$ is large relative to the actual value of this difference.

We can demonstrate errors in numerical approximation of the derivative of the function $f(x) = \exp x$ at the point $x_0 = 1$ using Maple. First we calculate the exact value $f'(x_0)$

```
> exact_value := D(exp)(1);
```

$$exact\_value := e$$

Now we define an approximation `d10f` of $Df(h)$ as a function of h

```
> d10f := h -> (evalf(exp(1+h)) - evalf(exp(1)))/h;
```

$$d10f := h \to \frac{\mathrm{evalf}(e^{(1+h)}) - \mathrm{evalf}(e)}{h}$$

Therefore `d10f(h)` gives $Df(h)$ calculated with 10-digits precision (see below).

The errors for $h = 0.5 * 10^{-i}$ $(i = 0, 1, \ldots, 10)$ can be calculated in the following way

```
> error := h -> exact_value - d10f(h);
```

$$error := h \to exact\_value - \mathrm{d10f}(h)$$

```
> for i from 0 to 10 do evalf(error(0.5*10^(-i))) od;
```

$$
\begin{gathered}
-.808532656 \\
-.069103972 \\
-.006807172 \\
-.000680172 \\
-.000078172 \\
-.000118172 \\
-.001718172 \\
-.001718172 \\
-.081718172 \\
-3.281718172 \\
2.718281828
\end{gathered}
$$

The obtained results suggest that there is an optimal value of h with which the derivative of $\exp(x)$ at $x_0 = 1$ can be approximated by the finite difference formula $Df(h)$.

• Round-off errors

Round-off errors have their origins in computer operations regardless of problem area. Here we describe these origins and examine some frequently encountered settings.

Regardless of its source, error is usually quantified in two different but related ways. Let x denote an exact value and x^* its computer approximation. Then the value $|x - x^*|$ is known as the *absolute error*. In many cases, the absolute error does not properly reflect its influence. For instance, an error of 0.01 m in measuring the distance to the moon would seem negligible, but it might be disastrous in designing a piston to fit into cylinder of a car motor. In view of the effect of scale, the concept of *relative error*, that is $|x - x^*|/|x|$, is helpful. Any number $\delta(x^*)$ satisfying the inequality

$$|x - x^*| \leq \delta(x^*)$$

is called an *absolute error bound* for the error of x^* as an approximation of x. Similarly, any number $\mathrm{Rel}\,(x^*)$ satisfying

$$\frac{|x - x^*|}{|x|} \leq \mathrm{Rel}\,(x^*)$$

is called a *relative error bound*. If $\delta(x^*)$ is a small number compared to $|x^*|$, then $\delta(x^*)/|x^*|$ is a good approximation for $\mathrm{Rel}\,(x^*)$, i.e.

$$\mathrm{Rel}\,(x^*) \approx \frac{\delta(x^*)}{|x^*|}.$$

Toward understanding the source and magnitude of round-off error we recall that computers store numbers in *floating-point representation*, i.e. in the form

$$x = f \cdot 10^E,$$

where $x \neq 0$ is any real number,

$$\frac{1}{10} \leq |f| \leq 1$$

and E is an integer. For example,

$$113.25 = 0.11325 \cdot 10^3,$$
$$-0.0546712 = -0.546712 \cdot 10^{-1}.$$

In floating-point arithmetic, it is correct to suppose that given any two floating point numbers, the arithmetical operation is performed perfectly, and then the result is rounded, as necessary, so that the result fits into a floating point computer word. Roughly speaking, significant figures are preserved in addition, multiplication and division. But subtraction can induce a special phenomenon known as *subtractive cancellation*. This arises when two nearly equal floating point numbers are subtracted from one another. We illustrate this phenomenon by the help of Maple.

Let us consider the positive number

```
> A := ((sqrt(5)-1)/2)^100;
```

$$A := (\frac{1}{2}\sqrt{5} - \frac{1}{2})^{100}$$

Expanding we have

```
> A := expand(A);
```

$$A := -\frac{354224848179261915075}{2}\sqrt{5} + \frac{792070839848372253127}{2}$$

First we show that the number A is the difference of two nearly equal numbers. Extracting components of A with the procedure **op** and then evaluating these terms – for example – in 50-digits floating-point arithmetic we get

```
> op(A)[1];
```

$$-\frac{354224848179261915075}{2}\sqrt{5}$$

```
> evalf(",50);
```

$$-.396035419924186126563499999999999999999999873748666\,10^{21}$$

```
> op(A)[2];
```

$$\frac{792070839848372253127}{2}$$

```
> evalf(", 50);
```

$$.396035419924186126563500000000000000000000000000000\,10^{21}$$

The true value of A lies in the interval $[0, 10^{-20}]$ which follows from

$$0 < A = \left(\frac{\sqrt{5}-1}{2}\right)^{100} = \left[\left(\frac{\sqrt{5}-1}{2}\right)^{5}\right]^{20} =$$

$$= \left(\frac{5\sqrt{5}-11}{2}\right)^{20} < \left(\frac{1}{10}\right)^{20} = 10^{-20}.$$

Now let us investigate what happens if we perform the operations in floating-point arithmetic. The precision of floating-point arithmetic can be defined by setting different values to the Maple variable `Digits`, whose default value is equal to ten. There are several functions that make Maple compute in floating-point arithmetic, the most important being **evalf** (**eval**uate using **f**loating-point arithmetic). This procedure approximates its first argument, the number of digits used is equal to the value of its second argument. If there is no second argument to **evalf**, then Maple takes the value of `Digits` as the number of digits to be used in floating-point arithmetic.

An approximate value of A in 10-digits floating-point arithmetic is

```
> evalf(A);
```

$$-.1\,10^{12}$$

which is very inaccurate because the true value of A lies in the interval $[0, 10^{-20}]$.

We can obtain better result if we increase the precision of the floating-point arithmetic:

```
> for n from 1 to 6 do evalf(A, 10*n) od;
```

$$-.1\,10^{12}$$

$$-10.$$

$$0$$

$$-.1\,10^{-18}$$

$$.126251334\,10^{-20}$$

$$.12625133380638429 42\,10^{-20}$$

In numerical methods subtractive cancellation is to be avoided if at all possible. Such loss of accuracy can often be avoided by a reformulation of the expression into a mathematically equivalent form, for example

$$\sqrt{1+x}-\sqrt{1-x}=\frac{2x}{\sqrt{1+x}+\sqrt{1-x}}\quad (x\in(0,1)),$$
$$\ln b-\ln a=\ln\frac{a}{b}\quad (a,b>0).$$

If it is difficult to find a suitable reformulation of an expression of the form $f(x+\varepsilon)-f(x)$, then subtractive cancellation can be avoided by using the Taylor expansion.

• Problem of stability

In most situations the effect of errors of a numerical method does not significantly affect the final results. However, in certain cases it can lead to a serious loss of accuracy so that computed results are very different from those obtained. The term of *instability* is used to describe this phenomenon. There are two fundamental types of instability in numerical analysis – inherent and induced. The first of these is a fault of the problem, the second one is that of the method of solution.

A *problem* is said to be *inherently unstable* (or *ill-conditioned*) if small changes in the input data of the problem cause large changes in its solution.

We illustrate this phenomenon in the following example.

EXAMPLE 1.14. *Show that the initial value problem*

$$y''(x)-10y'(x)-11y(x)=0,\qquad y(0)=1,\ y'(0)=-1$$

is ill-conditioned.

SOLUTION. We can solve this problem with Maple

```
> eq1 := (D@@2)(y)(x)-10*(D)(y)(x)-11*y(x)=0;
```

$$eq1 := (D^{(2)})(y)(x) - 10\,\mathrm{D}(y)(x) - 11\,y(x) = 0$$

```
> in_val1 := y(0)=1, (D)(y)(0)=-1;
```

$$in\_val1 := \mathrm{y}(0) = 1,\ \mathrm{D}(y)(0) = -1$$

```
> sol1 := dsolve({eq1, in_val1}, y(x));
```

$$sol1 := \mathrm{y}(x) = \frac{1}{e^x}$$

Now suppose that the initial conditions are replaced by

$$y(0) = 1 + \delta, \qquad y'(0) = -1 + \varepsilon$$

for some small numbers δ and ε. The particular solution satisfying these conditions is

```
> in_val2 := y(0)=1+delta, (D)(y)(0)=-1+epsilon;
```

$$in\_val2 := \mathrm{y}(0) = 1 + \delta,\ \mathrm{D}(y)(0) = -1 + \varepsilon$$

```
> sol2 := dsolve({eq1, in_val2}, y(x));
```

$$sol2 := \mathrm{y}(x) = \frac{\frac{11}{12}\,\delta - \frac{1}{12}\,\varepsilon + 1 + (\frac{1}{12}\,\delta + \frac{1}{12}\,\varepsilon)\,e^{(11\,x)}\,e^x}{e^x}$$

and therefore the change in the solution is

```
> collect(rhs(sol1) - rhs(sol2), exp(x));
```

$$-(\frac{1}{12}\,\delta + \frac{1}{12}\,\varepsilon)\,e^{(11\,x)} + \frac{-\frac{11}{12}\,\delta + \frac{1}{12}\,\varepsilon}{e^x}$$

The term $(\delta+\varepsilon)e^{11x}/12$ is large compared with e^{-x} for $x > 0$, indicating that this problem is ill-conditioned. □

If the problem is ill-conditioned then any numerical results, irrespective of the method used to obtain them, will be highly inaccurate and may be worthless. Nevertheless, it may happen that the original ill-conditioned problem can be transformed into a well-conditioned one with the same (or approximately same) solution.

We now consider a different type of instability which is the consequence of the method of solution rather than the problem itself.

A *method* is said to suffer from *induced instability* (or *numerical instability*) if small errors present at one stage of the method adversely affect the calculations in subsequent stages to such an extent that the final results are totally inaccurate.

This is illustrated by the following example.

EXAMPLE 1.15. *Let us consider the definite integrals*

$$E_n := \int_0^1 \frac{x^n}{x+5} dx \qquad (n = 0, 1, 2, \dots).$$

(a) *Show that the sequence* E_n $(n \in \mathbb{N})$ *satisfies the recursion formula*

$$\begin{aligned} E_0 &= \ln 6 - \ln 5 \\ E_n &= \frac{1}{n} - 5E_{n-1} \quad (n = 1, 2, 3, \dots), \end{aligned} \tag{1.22}$$

and monotone decreasingly tends to zero.

(b) *Compute some terms of these sequence with* (1.22) *using—for example—*16*-digits floating-point arithmetic. Let us observe the effect produced by round-off errors.*

SOLUTION. (a) It is clear that

$$E_0 = \int_0^1 \frac{1}{x+5} dx = [\ln(x+5)]_0^1 = \ln 6 - \ln 5.$$

The recursion formula follows from

$$E_n + 5E_{n-1} = \int_0^1 \frac{x^n + 5x^{n-1}}{x+5} dx = \int_0^1 x^{n-1} dx = \frac{1}{n}.$$

Since $x^{n+1} < x^n$ on the interval $(0, 1)$ thus

$$E_{n+1} = \int_0^1 \frac{x^{n+1}}{x+5} dx < \int_0^1 \frac{x^n}{x+5} dx = E_n,$$

i.e. the sequence E_n $(n \in \mathbb{N})$ is monotone decreasing. Finally from the inequalities

$$0 < E_n = \int_0^1 \frac{x^n}{x+5} dx < \int_0^1 x^n dx = \frac{1}{n} \quad (n \in \mathbb{N})$$

we obtain that

$$\lim_{n \to \infty} E_n = 0.$$

(b) Below we use the formula (1.22) to compute E_n, using 16 decimals throughout. The precision of floating-point arithmetic can be

defined by setting a value to the Maple variable `Digits`, whose default value is equal to ten.

```
> Digits := 16:
```

We can define the recursion formula (1.22) in the following way

```
> a := evalf(ln(6)) - evalf(ln(5));
```

$$a := .182321556793955$$

```
> E := proc(n)
      option remember;
        if n=0 then a
         else 1/n-5*E(n-1)
        fi
      end:
```

Using this simple program we get

```
> for n from 0 to 11 do
        sprintf(‘E(%f)=%f‘, 2*n+1, E(2*n+1))od;
```

$$\begin{aligned} E(1) &= .088392 \\ E(3) &= .043138 \\ E(5) &= .028468 \\ E(7) &= .021232 \\ E(9) &= .016926 \\ E(11) &= .014071 \\ E(13) &= .012039 \\ E(15) &= .010509 \\ E(17) &= .009056 \\ E(19) &= .001265 \\ E(21) &= -.170732 \\ E(23) &= -4.452110 \end{aligned}$$

It is absurd that $E_{21}, E_{23} < 0$! The reason for the absurd result is that the round-off error ε in E_0, whose magnitude can be as high as $5 \cdot 10^{-16}$ is multiplied by -5 in the calculation of E_1, which then has an error of -5ε. That error produces an error in E_2 of 25ε, etc. Therefore the

error in the calculated value of E_{23}, caused by the inaccuracy in E_0, is $5^{23} \cdot 5 \cdot 10^{-16} \approx 5.96$.

If we use more decimal places of accuracy, the absurd result will show up at a later stage.

We propose the reader to redo the calculations with other values of `Digits` and observe the changes. In practical problems this possibility of changing accuracy may be enough to get rid of this problem. □

The induced instability can be avoided either by modifying the existing method or by using a more suitable algorithm.

1.4. Exercises

1. Verify that the given function is a solution to the corresponding differential equation:

   (a) $y(x) = \sin(1 + \log(1 + x^2)), \quad (x \in \mathbb{R})$,
   $2x\sqrt{1 - y^2(x)} = y'(x)(1 + x^2)$,

   (b) $y(x) = (x - 4)e^{x+1} + 2$,
   $xy''(x) = y'(x) \log \dfrac{y'(x)}{x}$.

2. Show that the system of functions

   $$y_1(x) = -\frac{1}{x^2} \quad (x > 0), \quad y_2(x) = -x \log x \quad (x > 0)$$

   is a solution of the system of differential equations

   $$y_1'(x) = 2xy_1^2(x), \quad y_2'(x) = \frac{y_2(x)}{x} - 1.$$

3. Show that $y(x) = 2 + c\sqrt{1 - x^2}$ $(x \in (-1, 1))$, where c an arbitrary real parameter, is the general solution of the differential equation $(1 - x^2)y'(x) + xy(x) = 2x$.

4. Show that $y(x) = c_1x + c_2$ $(x \in \mathbb{R})$, where c_1 and c_2 arbitrary real parameters, is the general solution of the differential equation $y''(x) = 0$.

5. Find the coinciding solutions of the two equations:

   $$y'(x) = y^2(x) + 2x - x^4,$$
   $$y'(x) = -y^2(x) - y(x) + 2x + x^2 + x^4.$$

6. Determine domains in which the given equations have unique solutions:

   (a) $y'(x) = \sqrt{1 - y^2(x)}$,

   (b) $y'(x) = \dfrac{y(x) + 1}{x - y(x)}$,

(c) $y'(x) = \sqrt{x^2 - y(x)} - x$.

7. Sketch the direction field associated with the equations

   (a) $y'(x) = e^{-x} - 2y(x)$,
   (b) $y'(x) = \sin(x + y(x))$,
   (c) $y'(x) = y(x) - x^2 + 2x - 2$

   using the Maple's **DEplot** procedure. (It is contained in the **DEtools** package, which can be loaded by entering the command `with(DEtools)`.)

8. Solve the following differential equations with Maple. Try to find the solutions in their simplest form, and check if Maple finds all solution.

   (a) $3y^2(x)y'(x) + 16x = 12xy^3(x)$,
   (b) $y'(x) = 2\dfrac{y(x)}{x} - \left(\dfrac{y(x)}{x}\right)^2$,
   (c) $xy'(x) - y(x) = x\tan\left(\dfrac{y(x)}{x}\right)$.

9. Write the differential equation

   $$3y'''(x) + 4xy''(x) + \sin y(x) = f(x)$$

   as a system of first-order differential equations.

10. Find the general solution of the following equations

    (a) $y''(x) + 4y'(x) + 13y(x) = x\cos^2 3x$,
    (b) $y'(x) = 2x - y(x)$,
    (c) $y'(x) = xy^2(x)$.

11. Solve the following initial value problems with Maple and check the results

    (a) $y'(x) = y(x), \quad y(0) = 1$,
    (b) $y'(x) = 2x - y(x), \quad y(0) = 1$,
    (c) $y'(x) = xy^2(x), \quad y(0) = 1$.

12. Define

    $$a := 7,555,555,555,555,555 - 7,555,555,555,555,554.$$

    Compute this difference with Maple in integer and different digits floating-point arithmetic and compare the results.

13. Define

$$a(n) := 10^n - (10^n - 1),$$

with $n = 1, 2, \ldots$, and use Maple to evaluate $a(n)$ as a floating-point variable. Explain what is happening when the computed solution differs from 1.

14. (a) Find a root of the equation $ax^2 + bx + c = 0$ using the formula

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

with values $b = 1$ and $a = c = 10^{-n}$. Assess the accuracy of the results by seeing how closely the equation is satisfied for $n = 2, \ldots, 10$. Show that going to high precision does not count. (b) Think of a clever way to avoid subtractive cancellation, and improve the results.

15. Use Taylor expansion to avoid subtractive cancellation in the expression $e^x - e^{-x}$, when x is close to 0.

16. Derive absolute and relative error bounds for arithmetic operations on inexact data.

17. Assume that the real valued function f is continuously differentiable in a neighborhood of the point $x := (x_1, \ldots, x_n) \in \mathbb{R}^n$. Let x_i^* be an approximation of x_i $(i = 1, \ldots, n)$ and $x^* := (x_1^*, \ldots, x_n^*)$. Denote by $\delta(x_i^*)$ an absolute error bound for the error of x_i^* $(i = 1, \ldots, n)$. Explain why the number

$$\sum_{i=1}^{n} \left| \frac{\partial f}{\partial x_i}(x^*) \right| \delta(x_i^*)$$

may be considered as an upper bound for the absolute error $|f(x) - f(x^*)|$.

18. Derive error-bound formulas and absolute error bounds for the following expressions:

(a) $x + y + x^2 - xy + y^2$,

(b) $\sqrt{x^2 - y^2}$,

(c) $2 \cos x \cdot \cos y + \sin x \cdot \sin y$,

(d) $\dfrac{\sin x}{\cos y} + z^2 - 1$,

(e) $(x - y)e^{2x+y^2}$.

19. The following (decimal) numbers are approximated within a relative error bound 1%. In each case, give the smallest intervals that can be assured to contain the exact values.

    (a) 111.1,
    (b) 0.01111,
    (c) 43.1234,
    (d) 0.0431234.

20. The exact value of the number

$$\sum_{k=1}^{\infty} \frac{1}{k^2}$$

    is $\pi^2/6$. Approximate this number by summing for 1 to n, and alternatively, from n to 1. Let k be a floating-point number. Which direction of summation gives the more accurate answer?

Bibliography

[AB] Abel, M.L. and Braselton, J.P., *Differential Equations with Maple V.* AP Professional, Boston, 1994.

[BG] Birkhoff, G. and Gian-Carlo, Rota, *Ordinary Differential Equations.* John Wiley and Sons, Inc., New York, 1989.

[BD] Boyce, W.E. and DiPrima R.C., *Elementary Differential Equations and Boundary Value Problems.* John Wiley and Sons, Inc., New York, (5th ed.), 1992.

[Bu] Butcher, J.C. *The Numerical Analysis of Ordinary Differential Equations.* John Wiley and Sons, Inc., New York, 1987.

[CL] Coddington, E.A. and Levinson, N., *Theory of Ordinary Differential Equations.* McGraw-Hill Book Company, Inc., New York, 1955.

[Co] Collatz, L., *The Numerical Treatment of Differential Equations.* Springer-Verlag, Berlin-Heidelberg-New York, 1966.

[CB] Conte, S.D. and de Boor, C., *Elementary Numerical Analysis: An Algorithmic Aproach.* McGraw-Hill Kögakusha, Tokyo, (3rd ed.). 1980.

[DB] Dahlquist, G. and Björk, Å., *Numerical Methods.* Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1994.

[EW] Eldén, L. and Wittmeyer-Koch, L., *Numerical Analysis, An Introduction.* Academic Press, Inc., Boston, 1990.

[Ge] Gear, C.W., *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1971.

[H1] Hairer, E., Nørsett, S.P. and Wanner, G., *Solving Ordinary Differential Equations I. Nonstiff Problems.* Springer Verlag, Berlin, (2nd ed.), 1991.

[H2] Hairer, E., Wanner, G., *Solving Ordinary Differential Equations II. Stiff Problems and Differential-algebraic Equations.* Springer Verlag, Berlin, 1991.

[HW] Hall, G. and Watt, J.M. (Eds.), *Modern Numerical Methods for Ordinary Differential Equations.* Clarendon Press, Oxford. 1976.

[Ha] Hamming, R.W., *Numerical Methods for Scientists and Engineers.* McGraw-Hill Book Company, Inc., New York, (2nd ed.) 1973.

[Hr] Hartman, Ph., *Ordinary Differential Equations.* John Wiley and Sons, Inc., New York, 1964.

[HH] Hämmerlin, G. and Hoffmann, K-H., *Numerical Mathematics.* Springer-Verlag, New York Inc., 1991.

[Hc] Heck, A., *Introduction to Maple.* Springer-Verlag, New York, 1993.

[He] Henrici, P., *Discrete Variable Methods in Ordinary Differential Equations.* John Wiley and Sons, Inc., New York, 1962.

[HJ] Higham, N. J., *Accuracy and Stability of Numrical Algorithms*, SIAM, Philadelphia, 1996.

[Hi] Hildebrand, F.B., *Introduction to Numerical Analysis.* McGraw-Hill Book Company, New York, 1974.

[IK] Isaacson, E. and Keller, H.B., *Analysis of Numerical Methods.* John Wiley and Sons, Inc., New York, 1966.

[Is] Iserles, A., *A First Course in the Numerical Analysis of Differential Equations.* Cambridge Text in Applied Mathematics. Cambridge Univ. Press., 1996.

[Ka] Kamke, E., *Differentialgleichungen, Lösungsmethoden und Lösungen, Vol. 1.* Leipzig, 1959.

[KM] Kopchenova, N.V. and Maron, I.A., *Computational Mathematics, Worked Examples and Problems with Elements of Theory.* Mir Publishers, Moscow, 1975.

[La] Lambert, J.D., *Numerical Methods for Ordinary Differential Systems.* John Wiley and Sons, Ltd., Chichester, 1991.

[PT] Press, W.H., Teukolsky, S.A., Vatterling, W.T. and Flannery, B.P., *Numerical Recipes in C. The Art of Scientific Computing.* Second Ed., Cambridge Univ. Press, 1992.

[RR] Ralston, A. and Rabinowitz, P., *A First Course in Numerical Analysis.* McGraw-Hill Book Company, New York, 1978.

[Sc] Schwarz, H.R., *Numerical Analysis, A Comprehensive Introduction.* John Wiley and Sons, Ltd., Chichester, 1989.

[SG] Shampine, L.F. and Gordon, M.K., *Computer Solution of Ordinary Differential Equations.* W.H. Freeman, San Francisco, 1975.

[St] Stetter, H.J., *Analysis of Discretization Methods for Ordinary Differential Equations.* Springer Tracts in Natural Philosophy. Vol. 23, Springer Verlag, Berlin, 1973.

[SB] Stoer, J. and Bulirsch, R., *Introduction to Numerical Analysis.* Springer Verlag, Berlin, 1980

[YS] Yakowitz, S. and Szidarovszky, F., *An Introduction to Numerical Computations.* Macmillan Publ. Comp., New York, 1986.

CHAPTER 2

Single Step Methods

2.1. Basic concepts

We consider the first-order scalar initial value problem

$$y'(x) = f(x, y(x)), \qquad y(a) = \alpha, \tag{2.1}$$

where a and α are prescribed real values. We shall suppose that the problem (2.1) has a unique solution on the bounded interval $I := [a, b] \subset \mathbb{R}$ (see Theorem 1.2). Denote by $y(x)$ $(x \in I)$ the exact solution of (2.1).

If we cannot construct an expression in a closed-form for the solution $y(x)$ $(x \in I)$, then we can give an approximation of the exact solution. As we have already mentioned in Section 1.2, there are two fundamental types of approximate methods—*analytic* and *discrete*.

In the next part of this book we study only *discrete methods* which are also called *numerical methods*. These are based on the following idea: fix a positive integer N and try to determine an approximate value y_k of the exact value $y(x_k)$ for some discrete abscissae x_k in the interval I, where

$$x_0 := a < x_1 < x_2 < \cdots < x_{N-1} < x_N := b.$$

The points x_k $(k = 0, 1, \ldots, N)$ are also called *mesh points*. They are often equidistant, i.e.

$$x_k := a + kh \qquad (k = 0, 1, \ldots, N),$$

where

$$h := \frac{b - a}{N}$$

is the *step size* of the method. Thus, starting with the given initial values x_0, $y_0 := \alpha$, we can successively compute $y_1, y_2, \ldots, y_N$ which are the approximations of $y(x_1), y(x_2), \ldots, y(x_N)$.

The discrete methods can be divided into two groups according to the form of the representation of the solution:

- *single step methods* determine the approximation y_{k+1} at the abscissa $x_{k+1} = x_k + h$ solely on the basis of the approximation point (x_k, y_k); whereas

- *multistep methods* use the information at more than one previous support abscissae to determine the approximation at the next point.

In this chapter we investigate the first group of numerical methods and the second one will be discussed in the next chapter.

In general, any *single step method* can be written in the form:

$$\boxed{\begin{aligned} x_0 &:= a, \qquad y_0 := \alpha \\ x_{k+1} &:= x_k + h, \qquad h := (b-a)/N, \\ y_{k+1} &:= y_k + h\Phi(x_k, y_k, h) \\ &(k = 0, 1, \ldots, N-1), \end{aligned}} \tag{2.2}$$

where Φ (the *increment function*) is a real valued function defined on $I \times \mathbb{R} \times \mathbb{R}^+$. Then $\Phi(x_k, y_k, h)$ describes how the new approximate value y_{k+1} is computed from (x_k, y_k) and the step size h. Therefore, starting with the initial values a and α of the initial value problem (2.1), one now obtains approximate values y_k for the exact quantities $y(x_k)$ $(k = 0, 1, 2, \ldots, N)$.

REMARK. Therefore to define a single step method it is necessary to prescribe the values x_0, y_0, h and the increment function Φ. Then y_k can be computed recursively. The single step method (2.2) is also called a *difference equation* for the unknown values y_k. In some cases (for example if Φ is linear) y_k may be simply expressed explicitly by k and then we say that we *solve the difference equation* (see Section 2.8). □

For different choices of the function Φ different single step methods may be obtained.

Single step methods (2.2) can be directly generalized to systems of differential equations and therefore also to higher-order differential equations. The methods and results for initial value problems for systems of ordinary differential equations of first-order are essentially independent of the number of unknown functions. In the following we therefore limit ourselves for simplicity and clarity to the case of only one ordinary differential equation of first-order for a single unknown function.

Before continuing, we would like to introduce the *"big O"* concept, which will be used in the following chapters. The function $f(h)$ is said to be $O(h^p)$ (read "big oh of h^p") at $h = 0$ if there are positive numbers K and h_0 such that $|f(h)| \leq Kh^p$ for all h with $|h| \leq h_0$. We usually

write just

$$\boxed{f(h) = O(h^p),}$$

with $h = 0$ understood.

For example $\sin h$ is $O(h)$ at $h = 0$ because $|\sin h| \leq |h|$ for every number h. From the Taylor' series expansion we have

$$|\cos h - 1| \leq \frac{1}{2}h^2 \quad (|h| \leq 1),$$

and thus $\cos h - 1$ is $O(h^2)$ at $h = 0$. Similarly, $\sin h - h$ is $O(h^3)$ at $h = 0$. It is customary to express statements such as these in the form

$$\sin h = O(h),$$
$$\cos h = 1 + O(h^2),$$
$$\sin h = h + O(h^3).$$

2.2. Euler's method

A first numerical method for the solution of the initial value problem (2.1) is suggested by the following simple observation. Since $f(x, y(x))$ is just the slope, $y'(x)$, of the desired exact solution of (2.1), one has for $h \neq 0$ approximately (see Figure 2.1)

$$\frac{y(x+h) - y(x)}{h} \approx y'(x) = f(x, y(x))$$

or

$$y(x+h) \approx y(x) + hf(x, y(x)).$$

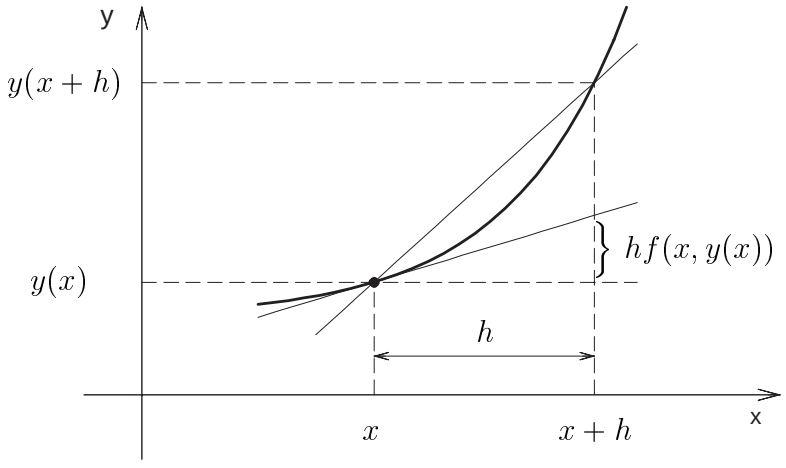


Figure 2.1. Approximation of the derivative

Once a step size $h = (b-a)/N$ has been chosen, starting with the given initial values x_0, $y_0 := y(a) = \alpha$, one thus obtains at equidistant points

$$x_k := x_0 + kh \qquad (k = 0, 1, \ldots, N)$$

approximations y_k $(k = 0, 1, \ldots, N)$ to the values $y(x_k)$ of the exact solution $y(x)$ $(x \in I)$ as follows:

$$\boxed{\begin{aligned} x_0 &:= a, \qquad y_0 := \alpha \\ x_{k+1} &:= x_k + h, \qquad h := (b-a)/N, \\ y_{k+1} &:= y_k + hf(x_k, y_k) \\ &(k = 0, 1, \ldots, N-1). \end{aligned}} \tag{2.3}$$

This is the oldest and the simplest method for the numerical solution of initial value problems. It was proposed by Euler in 1768 and is called *Euler's method* or the *polygon method of Euler*.

Euler's method has a geometric interpretation. We start at the point (x_0, y_0), and approximate the solution curve by the tangent at the point (x_0, y_0). We compute the slope $y'(x)$ of the tangent directly from the differential equation. We continue along this tangent until we reach $x_1 = x_0 + h$. The corresponding y-value is y_1. Through the point (x_1, y_1), there is a solution curve (which, however, does not correspond to the given initial value). Similarly, we approximate this curve by a tangent through the point (x_1, y_1) and continue along this tangent until we reach x_2, and so on.

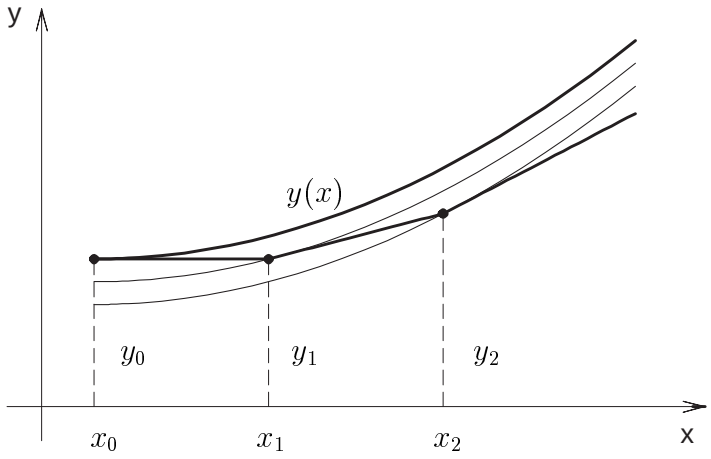


Figure 2.2 Euler's method

We want to emphasize that Euler's method is hardly ever used in practice, as there are more accurate and more efficient (but, at the same time, more complicated) methods. Euler's method is simple, and that is why we use it for introducing the basic concepts in numerical solution of initial value problems.

We will illustrate Euler's method by numerically solving the initial value problem

$$\boxed{y'(x) = -2xy^2(x), \qquad y(0) = 1.} \tag{2.4}$$

We will also show that how Maple can be used to perform the calculations encountered when solving a differential equation.

This problem has a closed-form solution, and we will use it to check the accuracy of the numerical solution.

EXAMPLE 2.1. *Use the Maple's* **dsolve** *procedure to find the exact solution of the initial value problem* (2.4).

SOLUTION. We define `eq1` to be the equation and `in_cond` to be the initial condition in (2.4).

```
> eq1 := diff(y(x), x) = -2*x*y(x)^2;
```

$$eq1 := \frac{\partial}{\partial x}\,\mathrm{y}(x) = -2\,x\,\mathrm{y}(x)^2$$

```
> in_cond := y(0) = 1;
```

$$in\_cond := \mathrm{y}(0) = 1$$

The **dsolve** procedure can exactly solve this problem:

```
> dsolve({eq1, in_cond}, y(x));
```

$$\mathrm{y}(x) = \frac{1}{x^2+1}$$

This answer can easily be checked. First, by using the **subs** procedure we substitute the obtained function in the equation itself we have

```
> subs(", ode);
```

$$\frac{\partial}{\partial x}\,\frac{1}{x^2+1} = -2\,\frac{x}{(x^2+1)^2}$$

Evaluating the difference of the two sides of this equation we get

```
> expand( lhs(") - rhs(") );
```

$$0$$

Therefore, the function

$$y(x) = \frac{1}{x^2+1} \qquad (x \in \mathbb{R})$$

is the exact solution of (2.4). It is easy to prove that this problem has no other solution on the whole real line. □

EXAMPLE 2.2. *Write a short Maple program to compute the approximate values y_k of the solution of (2.4) at the points $x_k := 0.1k$ $(k = 0, 1, \ldots, 6)$ by means of Euler's method with the step size $h = 0.001$.*

SOLUTION. First we define the right hand side of the given differential equation, the initial values and the step size

```
> f := (x,y) -> -2*x*y^2:
> a := 0:  y0 := 1:  h := 0.001:
```

A Maple program of the algorithm (2.3) is

```
> y:=proc(k)
      option remember;
      if k=0 then y0
        else y(k-1)+h*f(a+(k-1)*h,y(k-1))
      fi
     end:
```

Note that in defining the recursively defined function `y`, we take advantage of the option `remember`. This instructs Maple to remember the values of `y` computed, and thus, when computing `y(n)`, Maple need not recompute `y(n-1)`.

We obtain the approximate values at the prescribed points in the following way:

```
> for k from 0 to 6 do y(100*k) od;
```

$$\begin{array}{c} 1 \\ .9901957693 \\ .9617138409 \\ .9176551110 \\ .8623085107 \\ .8002271764 \\ .7354898746 \end{array}$$

In this case Maple used 10 digits in floating-point arithmetic, so we had the result with round-off errors. The precision of the floating-point arithmetic can be defined by setting different values to the Maple variable `Digits`. If we want to obtain more accuracy we have to set this variable to a higher value at the beginning.

We remark that we can obtain results without round-off errors if we give `h=1/1000` instead of `h=0.001` since Maple will then use exact arithmetic. □

The above program may be modified to obtain a more complete and convenient procedure. Fortunately this is unnecessary because Maple contains such a procedure. Using Maple help

```
> ?dsolve[classical];
```

we can see that the program can solve a problem by means of Euler's method if we invoke the **dsolve** function with the options

`type=numeric` and `method=classical[foreuler]`

In this function the step size may be modified.

Let us consider how EXAMPLE 2.2 may be solved using the Euler's method of Maple.

EXAMPLE 2.3. *Compute the approximate values y_k of the solution of* (2.4) *at the points $x_k := 0.1k$ $(k = 0, 1, \ldots, 6)$ by means of the built-in Euler's method of Maple with the step sizes $h = 0.1,\ 0.01,\ 0.001$. Give also the errors*

$$E_k := y(x_k) - y_k \qquad (k = 0, 1, \ldots, 6).$$

SOLUTION. First we define the initial value problem (2.4):

```
> InValPr := {diff(y(x), x)=-2*x*y(x)^2, y(0)=1}:
```

Now, we invoke the **dsolve** function with the above option and with the step size 0.001, say.

```
> es0 := dsolve(InValPr, y(x), type=numeric,
          method=classical[foreuler], stepsize=0.001);
```

$$es0 := \mathbf{proc}(x\_classical) \ \ldots \ \mathbf{end}$$

The output shows that at this point the program only remembers the name of the procedure which will be executed if we invoke our function `es0`. The approximate value for example at the mesh point $x = 0.4$ may be obtained in this manner:

```
> es0(0.4);
```

$$[x = .4,\ \mathrm{y}(x) = .8623085097414066]$$

Note the form in which Maple gives the result. The *value* of the approximation may be selected from this answer in the following way:

```
> rhs(es0(0.4)[2]);
```

$$.8623085097414066$$

In this case the **dsolve** function used 16 digits in floating-point arithmetic. There are several functions that make Maple compute in floating-point arithmetic, the most important being **evalf**. This procedure approximates its first argument; the number of digits used is equal to the value of its second argument. For example

```
> evalf(", 10);
```

$$.8623085097$$

Our aim is to compute the approximate values at different mesh points with different step sizes. Thus we define the following Maple functions:

```
> x := k -> k*0.1:  # for the mesh points
> es1 := h -> dsolve(InValPr, y(x), type=numeric,
              method=classical[foreuler], stepsize=h):
```

The function `es1` has a new variable, the step size. Thus, `es1(h)(x)` gives an approximate value of the exact solution of our problem at the point `x` using the Euler's method with the step size `h`. For example

```
> es1(0.001)(x(4));
```

$$[x = .4,\ \mathrm{y}(x) = .8623085097414066]$$

Let us collect everything into a new Maple function which has two arguments. The first argument is the mesh point and the second one is the step size.

```
> EulerSol := (x, h) -> rhs(es1(h)(x)[2]):
```

Thus `EulerSol(x, h)` gives the same result as `es1(h)(x)` but in a different form:

```
> EulerSol(x(4), 0.001);
```

$$.8623085097414066$$

We can quickly make several numerical experiments using this function and not only for problem (2.4). We illustrate these possibilities by making a table which contains the mesh points, approximate values of the exact solution and the values obtained by means of Euler's method with step sizes $h = 0.1, 0.01, 0.001$.

From the previous example we know the exact solution

```
> ExactSol := x->1/(x^2+1);
```

$$ExactSol := x \to \frac{1}{x^2+1}$$

To create a table we use the **array** procedure with corresponding headings.

```
> mm := array(1..8, 1..5):
      mm[1,1]:=`x(k)`:  mm[1,2]:=`Exact sol.`:
        mm[1,3]:=` h=0.1 `:
      mm[1,4]:=` h=0.01 `:  mm[1,5]:=` h=0.001 `:
   for i from 2 to 8 do
        mm[i,1]:=0.1*(i-2):
        mm[i,2]:=evalf(ExactSol(x(i-2)),5):
      for j from 3 to 5 do
        mm[i,j]:=evalf(EulerSol(x(i-2),10^(-j+2)), 5)
      od:
   od:
> eval(mm);
```

$x(k)$	*Exact sol.*	$h = 0.1$	$h = 0.01$	$h = 0.001$
0	1.	1.	1.	1.
.1	.99010	1.	.99107	.99020
.2	.96154	.98000	.96330	.96171
.3	.91743	.94158	.91969	.91766
.4	.86207	.88839	.86448	.86231
.5	.80000	.82525	.80229	.80023
.6	.73529	.75715	.73727	.73549

Since we know the exact solution of (2.4), the accuracy of the numerical method can be checked very easily and quickly using Maple.

```
> err := (x, h) -> ExactSol(x) - EulerSol(x, h):
```

Therefore, `err(x, h)` gives the error at the point `x` if we use Euler's method with step size `h`. To observe the accuracy more conveniently we compile a table.

```
> tt := array(1..8,1..4):
     tt[1,1]:=`x(k)`:  tt[1,2]:=`h=0.1`:
     tt[1,3]:=`h=0.01`:  tt[1,4]:=`h=0.001`:
   for i from 2 to 8 do
       tt[i,1]:=0.1*(i-2);
     for j from 2 to 4 do
       tt[i,j]:=evalf(err(x(i-2),10^(-j+1)),5);
     od:
   od:
> eval(tt);
```

$$\begin{bmatrix} x(k) & h=0.1 & h=0.01 & h=0.001 \\ 0 & 0 & 0 & 0 \\ .1 & -.00990 & -.00097 & -.00010 \\ .2 & -.01846 & -.00176 & -.00017 \\ .3 & -.02415 & -.00226 & -.00023 \\ .4 & -.02632 & -.00241 & -.00024 \\ .5 & -.02525 & -.00229 & -.00023 \\ .6 & -.02186 & -.00198 & -.00020 \end{bmatrix}$$

□

• The evaluation of the results

1.) From the above table it can be seen that if the step size is fixed, the error of the numerical solution will grow as the number of steps is increasing. There are two main reasons for this phenomena. The first is the so-called *truncation* (or *discretization*) *error* which is related to the discretized equation that we solved instead of the exact equation. The second is the *round-off error* which is related to the finite representations of the numbers on the computer.

2.) It appears that with decreasing step size h the accuracy of the approximate solutions is increasing. This leads to the following issue: whether any desired degree of accuracy can be achieved for any problem by picking a small enough h. This suggests the definition of the *convergence* which will be made more precise when specific classes of methods are discussed. Since as h decreases the number of steps and hence the amount of calculation increases, we would expect the effect of round-off errors is to increase because there are more of them. Thus, in the definition of the convergence, we must require that the computations indicated in the method be performed exactly. In practice, this means that additional digits are carried in the computations as h decreases.

In general, we may ask how can we evaluate the error of a numerical method on the *whole interval uniformly*, if we have in hand the numerical results only at discrete points? The answer to this question is not simple. We want to stress that the basic problem of numerical analysis is the estimation of errors which occur in numerical processes.

3.) To calculate the errors in EXAMPLE 2.3, we had to know the exact solution at every grid point. But, *how can we estimate the error* of an approximate solution if we do not know the exact solution of the original problem? One possibility is as follows: to compute the approximate solution twice, first by a given step size and after this, with a smaller step size for which the set of grid points contains the grid points of the first calculation. Thus, we may compare the two computational results in the common grid points: their differences give us information on the error committed in the calculation.

Every discrete method for the solution of an initial value problem determines the approximate values y_k of the exact values $y(x_k)$ only at the mesh points x_k $(k = 0, 1, \ldots, N)$ of an interval $[a, b]$. But in practice we also need to approximate the exact solution at further points of the interval.

A natural way to treat this problem is the following: try to fit a smooth curve through the points (x_0, y_0), (x_1, y_1), $\ldots$, (x_N, y_N). There are more possibilities to do this, for example with polynomials or with spline functions. These formulas will be not reviewed here.

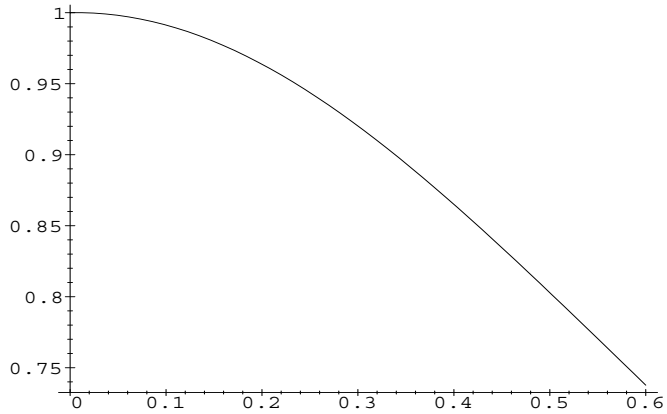
We only remark that fortunately Maple can help also in this type of case because it contains a built-in algorithm to fit smooth curves through discrete points using spline functions. The program automatically calls these algorithms when we invoke the **dsolve** procedure with the option `numeric=true`. To illustrate this let us consider the previous example. Entering

```
> es1(0.1)(0.235);
```

$$[x = .235,\ \mathrm{y}(x) = .9665544000000000]$$

we see that Maple is able to compute the approximate values not only at the mesh points. To graph the calculated approximate function we use the procedure **odeplot** contained in the **plots** package. We load the **plots** package and then graph the result given in `es1(0.1)`:

```
> with(plots):
> odeplot(es1(0.1), [x, y(x)], 0..0.6);
```



Euler's method is readily applied to systems of differential equations as well as to differential equations of high-orders. For illustration we consider the system of two equations of first-order (see Section 1.1)

$$y_1'(x) = f_1(x, y_1(x), y_2(x)),$$
$$y_2'(x) = f_2(x, y_1(x), y_2(x))$$

with the initial conditions

$$y_1(a) = \alpha_1, \quad y_2(a) = \alpha_2.$$

Here f_1 and f_2 are given real valued functions and (a, α_1, α_2) is an arbitrarily fixed point in the intersection of the domain of definition of f_1 and f_2.

Suppose that this problem has a unique solution $y_1(x), y_2(x)$ on the interval $[a, b]$. Denote by $y_1^{[k]}$ and $y_2^{[k]}$ an approximate value of $y_1(x_k)$ and $y_2(x_k)$ respectively. If the mesh points x_k are equidistant (i.e.

$x_k = x_0 + hk,\ h = (b-a)/N$), then Euler's method has the following form:

$$\boxed{\begin{aligned} x_0 &:= a, \quad y_1^{[0]} := \alpha_1,\ y_2^{[0]} := \alpha_2, \\ x_{k+1} &:= x_k + h, \quad h := (b-a)/N, \\ y_1^{[k+1]} &:= y_1^{[k]} + hf_1(x_k, y_1^{[k]}, y_2^{[k]}), \\ y_2^{[k+1]} &:= y_2^{[k]} + hf_2(x_k, y_1^{[k]}, y_2^{[k]}), \\ &(k = 0, 1, \ldots, N-1). \end{aligned}} \tag{2.5}$$

The next example illustrates that Maple can also solve systems of differential equations by means of Euler's method.

EXAMPLE 2.4. *Using Euler's method, find a numerical solution of the following system*

$$y_1'(x) = x \cdot (y_2(x) - y_1(x)),$$
$$y_2'(x) = x \cdot (y_2(x) + y_1(x))$$

with the initial conditions

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

on the interval $[0, 0.6]$ *with the step size* $h = 0.1$.

SOLUTION. As in the EXAMPLE 2.3 we have

```
> s1 := diff(y1(x),x)=x*(y2(x)-y1(x));
  s2 := diff(y2(x),x)=x*(y2(x)+y1(x));
```

$$s1 := \frac{\partial}{\partial x}\,\mathrm{y1}(x) = x\,(\mathrm{y2}(x) - \mathrm{y1}(x))$$

$$s2 := \frac{\partial}{\partial x}\,\mathrm{y2}(x) = x\,(\mathrm{y2}(x) + \mathrm{y1}(x))$$

```
> in_conds1 := y1(0) = 1, y2(0) = 1;
```

$$in\_conds1 := \mathrm{y1}(0) = 1,\ \mathrm{y2}(0) = 1$$

```
> num_sol1 := dsolve({s1,s2,in_conds1}, {y1(x),y2(x)},
          type=numeric, method=classical[foreuler],
          stepsize=0.1):
> for n from 0 to 6 do num_sol1(0.1*n) od;
```

$$[x = 0,\ \mathrm{y1}(x) = 1.,\ \mathrm{y2}(x) = 1.]$$

$$[x = .1,\ \mathrm{y1}(x) = 1.,\ \mathrm{y2}(x) = 1.]$$

$$[x = .2,\ \mathrm{y1}(x) = 1.,\ \mathrm{y2}(x) = 1.020000000000000]$$

$$[x = .3,\ \mathrm{y1}(x) = 1.000400000000000,\ \mathrm{y2}(x) = 1.060400000000000]$$

$$[x = .4,\ \mathrm{y1}(x) = 1.002200000000000,\ \mathrm{y2}(x) = 1.122224000000000]$$

$$[x = .5,\ \mathrm{y1}(x) = 1.007000960000000,\ \mathrm{y2}(x) = 1.207200960000000]$$

$$[x = .6,\ \mathrm{y1}(x) = 1.017010960000000,\ \mathrm{y2}(x) = 1.317911056000000]$$

□

Euler's method can also be used to solve higher-order differential equations, as illustrated in the following example.

EXAMPLE 2.5. *Use the Euler's method of Maple to solve the initial value problem*

$$y''(x) + \frac{y'(x)}{x} + y(x) = 0, \qquad y(1) = 0.77, \quad y'(1) = -0.44$$

on the interval $[1, 1.5]$ *with the step size* $h = 0.1$.

SOLUTION. Similarly to the previous example we get

```
> hoeq := (D@@2)(y)(x)+(D)(y)(x)/x+y(x)=0;
```

$$hoeq := (D^{(2)})(y)(x) + \frac{\mathrm{D}(y)(x)}{x} + \mathrm{y}(x) = 0$$

```
> in_conds2 := y(1) = 0.77, (D)(y)(1) = -0.44;
```

$$in\_conds2 := \mathrm{y}(1) = .77, \mathrm{D}(y)(1) = -.44$$

```
> num_sol2 := dsolve({hoeq, in_conds2}, y(x),
              type=numeric, method=classical[foreuler],
              stepsize=0.1):
> for n from 0 to 5 do num_sol2(1+0.1*n) od;
```

$$[x = 1., \mathrm{y}(x) = .77, \frac{\partial}{\partial x}\mathrm{y}(x) = -.44]$$

$$[x = 1.1, \mathrm{y}(x) = .7260000000000000, \frac{\partial}{\partial x}\mathrm{y}(x) = -.4730000000000000]$$

$$[x = 1.2, \mathrm{y}(x) = .6787000000000001, \frac{\partial}{\partial x}\mathrm{y}(x) = -.5025999999999999]$$

$$[x = 1.3, \mathrm{y}(x) = .6284400000000001, \frac{\partial}{\partial x}\mathrm{y}(x) = -.5285866666666666]$$

$$[x = 1.4, \mathrm{y}(x) = .5755813333333335, \frac{\partial}{\partial x}\mathrm{y}(x) = -.5507701538461538]$$

$$[x = 1.5, \mathrm{y}(x) = .5205043179487181, \frac{\partial}{\partial x}\mathrm{y}(x) = -.5689875619047619]$$

□

2.3. Convergence and consistency of single step methods

Numerical results are influenced by many types of errors (see Section 1.3), therefore we naturally are concerned with how close we can make the numerical solution to the exact solution. When we pick a method it may depend on one or more parameters, for example, the step size h for Euler's method. We would like to know how to pick these parameters to achieve any desired accuracy. It is possible that there is an error below which is not possible to go. At this point we loosely define the *concept of convergence* to mean that any desired degree of accuracy can be achieved by picking a small enough h. This definition will be made more precise.

The numerical solutions of initial value problems contain two main sources of error, *truncation* and *round-off error*.

We can ask the following: what is the accumulated error of a method for a given step size h after one step and after several steps? Does the numerical solution converge to the exact solution as $h \to 0$? How fast is the convergence of a method? The answers to these questions are very important because if we do not proceed carefully it may well happen that the computed approximations have very little to do with the desired solution functions, or may even be meaningless. In this section we deal with these problems.

• Truncation errors

Now we suppose that the computations indicated in the method be performed exactly, i.e. round-off errors are not taken into account.

We distinguish between *local* and *global truncation error*. First we give a general definition of the *local truncation error* which characterizes the error of a method committed in one step of calculation.

DEFINITION 2.1. *The* ***local truncation error*** *at a point is the difference between the value given by the method and the value of the solution of the differential equation which passes through the value at the beginning of the step.*

We reformulate it for a single step method.

DEFINITION 2.2. *The* ***local truncation error*** *or the* ***local discretization error*** e_{k+1} *at the point* x_{k+1} *is defined by the expression*

$$\begin{gathered} e_{k+1} := e(x_{k+1}, h) := \\ y(x_{k+1}) - y(x_k) - h\Phi(x_k, y(x_k), h) =: y(x_{k+1}) - \tilde{y}(x_{k+1}). \end{gathered} \tag{2.6}$$

Therefore, the quantity e_{k+1} indicates how well the exact solution $y(x)$ fulfills the formula (2.6).

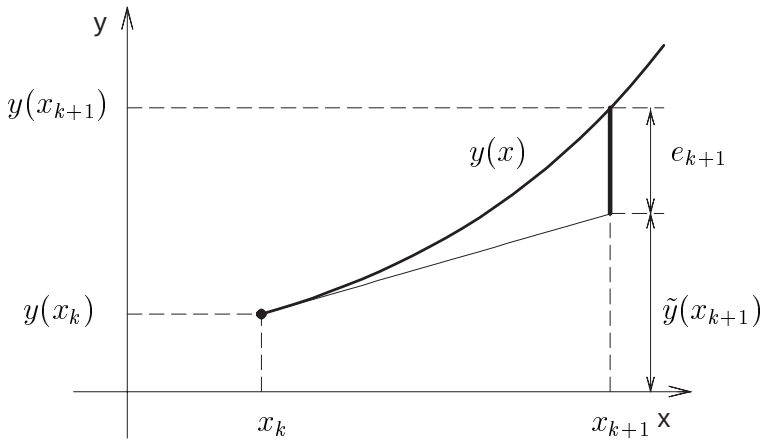


Figure 2.3. The local truncation error

Sometimes the local truncation error (2.6) is defined by the difference between the exact value $y(x_{k+1})$ and the computed approximation y_{k+1} after one step, if the exact value $y(x_k)$ at the point x_k is considered.

When we solve a problem with Euler's method numerically, at each step we usually cross over onto another member of the family of solutions, as displayed graphically in Figure 2.1. Thus, in practice, in

every step of calculation we have to solve a new initial value problem. The local truncation error in Euler's method is the deviation after each step between a solution curve and its tangent. It is easy to estimate the local truncation error of this method.

EXAMPLE 2.6. *Show that if the solution of the initial value problem* (2.1) *is twice continuously differentiable, then in Euler's method the local truncation error* e_{k+1} *at the mesh point* x_{k+1} *satisfies the inequality*

$$|e_{k+1}| \leq Ch^2 \quad (k = 0, 1, \ldots, N-1),$$

where $C > 0$ *is a suitable real number, i.e.* e_{k+1} *is* $O(h^2)$.

SOLUTION. According to the definition (2.6), the local truncation error of Euler's method is given by

$$e_{k+1} = y(x_{k+1}) - y(x_k) - hf(x_k, y(x_k)).$$

Replacing $y(x_{k+1})$ by the Taylor's series expansion with remainder term at the point x_k we get

$$e_{k+1} = y(x_k) + y'(x_k)h + \frac{y''(\xi)}{2}h^2 - y(x_k) - hy'(x_k),$$

where $\xi \in (x_k, x_{k+1})$ and $y'(x_k) = f(x_k, y(x_k))$. Therefore the local truncation error is

$$|e_{k+1}| = \frac{|y''(\xi)|}{2}h^2 \leq \frac{h^2}{2} \max_{x_0 < \xi < x_N} |y''(\xi)| = O(h^2).$$

□

After *several steps* the total error between the computed approximation and the exact solution is of interest.

DEFINITION 2.3. *The* ***global (truncation) error*** E_k *at the point* x_k *is given by the difference*

$$E_k := y(x_k) - y_k. \tag{2.7}$$

This quantity measures the error that accumulated after k steps (see Figure 2.4).

The global error may be estimated from above by the help of the local truncation error. Therefore, it plays the central role in the qualitative judgement of a single step method. The following theorem for the general single step method can be proved.

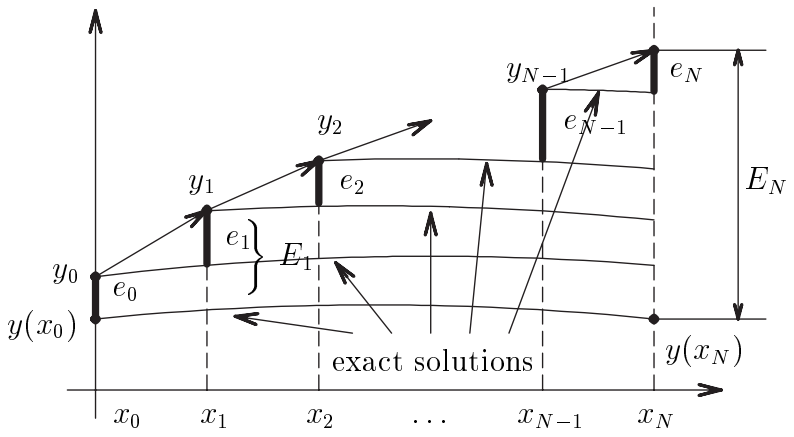


Figure 2.4. The global truncation error

THEOREM 2.1. *Suppose that the local truncation error satisfies*

$$\max_{0\le k\le N-1} |e_{k+1}| \le Ch^{p+1},$$

and suppose that the increment function Φ satisfies the Lipschitz condition with respect to its second variable with constant L. Then the global truncation error E_N at the fixed abscissa $x_N = x_0 + Nh$ is bounded by

$$|E_N| \le h^p \frac{C}{L}(e^{NLh} - 1). \tag{2.8}$$

The theorem remains valid if the method is applied with variable step sizes. In this case $h = \max_{0\le i\le N} h_i$.

• Round-off errors

In practice, when we apply a numerical method, for example Euler's method, round-off errors occur in the calculation because of the finite digits we used. So, in every step an additional term r_k is added to the Euler formula

$$y_{k+1} = y_k + hf(x_k, y_k) + r_k,$$

where r_k acts in the same way as an additional local truncation error. It may be proved that the total error is

$$t_k = O(\frac{|r|}{h} + h),$$

where $|r|$ is independent on h if the precision (the number of digits) is kept fixed in the calculation. Thus, the total error t_k will initially decrease as the step size h decreases and the truncation error decreases,

and then will increase as the round-off error becomes significant. This effect is illustrated in Figure 2.5.

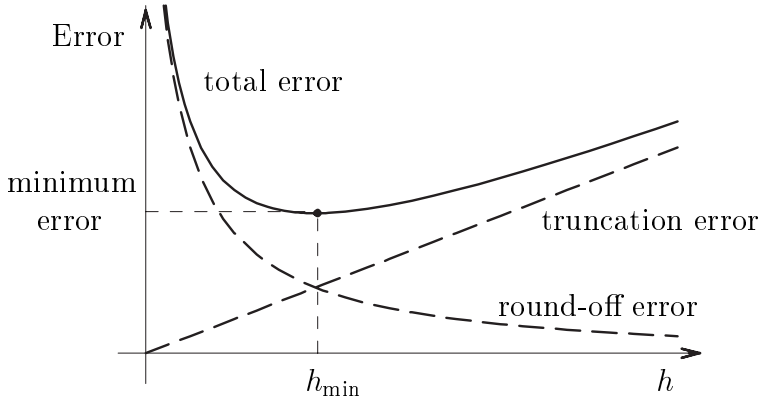


Figure 2.5. Total error as a function of h.

The situation is similar in the general single step methods. Therefore, we can conclude that if the step size h decreases we have to increase the decimal digits of precision to avoid the effect of round-off errors.

• Convergence

From the EXAMPLE 2.6 it follows that the local truncation error is $O(h^2)$ for Euler's method. Thus, the inequality (2.8) shows that the error bound decreases proportionally to the step size h. So, the value y_n *converges* to the exact value $y(x_n)$ at the fixed abscissa x_n as $h \to 0$, at least if rounding errors do not accour. The convergence is linear, with respect to the step size h, and we say that Euler's method is of *order one*. This implies the notion of the convergence.

For given x and h such that $(x - x_0)/h = n$ is an integer, we introduce the following notation for the numerical solution:

$$y_h(x) = y_n \qquad \text{if} \qquad x - x_0 = nh. \tag{2.9}$$

A method is expected to be "good" in the sense that the numerical solution $y_h(x)$ converges to the exact solution $y(x)$ as $h \to 0$. Furthermore, we expect rapid convergence.

DEFINITION 2.4. *A single step method* (2.2) *is said to be* ***convergent****, if, for all initial value problems satisfying the hypotheses stated in the existence theorem, we have*

$$\lim_{h \to 0} y_h(x) = y(x), \qquad x \in [x_0, b] \tag{2.10}$$

whenever the starting value satisfy

$$\lim_{h\to 0} y_0 = y(x_0).$$

A method which is not convergent is said to be ***divergent.***

DEFINITION 2.5. *The single step method* (2.2) *is* ***convergent of order p****, if to any problem* (2.1) *with f sufficiently differentiable, there exists a positive* h_0 *such that*

$$|y(x) - y_h(x)| \le Ch^p \qquad \text{for} \qquad h \le h_0 \tag{2.11}$$

whenever the starting value satisfy

$$|y(x_0) - y_0| \le C_0 h^p \qquad \text{for} \qquad h \le h_0.$$

We say also that the single step method is of *order p.*

Hence the convergence means that the numerical solution tends to the exact solution as the grid becomes increasingly fine. Therefore, convergence is related to the behavior of the *solution* of the difference equations.

We note that errors are permitted in the starting value y_0 since in practice we cannot represent $y(x_0)$ exactly in finite precision.

From the above definition it follows that a single step method is of order p if and only if its local truncation error e_k satisfies

$$\max_{0\le k\le N} |e_k| = O(h^{p+1})$$

so that for the global truncation error E_N we have

$$\max_{0\le k\le N} |E_N| = O(h^p).$$

We now turn to the question of what conditions a numerical method must satisfy if it is to be convergent.

As from the EXAMPLE 2.6 may be shown, for a reasonable single step method we have to require that the local truncation error tend to zero as $h \to 0$, which is equivalent to

$$\lim_{h\to 0} \Phi(x, y, h) = f(x, y), \quad \text{for all} \quad (x, y). \tag{2.12}$$

This implies the following

DEFINITION 2.6. *A single step method* (2.2) *is called* ***consistent*** *with the problem* (2.1) *if* (2.12) *holds.*

In other words, consistency means that the difference equation formally converges to the differential equation as $h \to 0$.

It can be proved (see e.g. [**He**] Theorem 2.1) that a single step method (2.2) is convergent if and only if it is consistent, whenever the increment function Φ is continuous with respect to its variables and

satisfies a Lipschitz condition with respect to its second variable. It is easy to see that the Euler's method is consistent and so convergent.

2.4. A first improvement of Euler's method

There are several ways to derive more accurate methods for the approximate solution of the initial value problem (2.1). In this section we shall show some of the ideas from which higher-order methods can be obtained.

• Taylor series method

We derived the Euler's method from the relation

$$y(x+h) \approx y(x) + y'(x)h,$$

which can be viewed as an approximation of $y(x+h)$ by the first two terms of the Taylor expansion of the function y about the fixed point x. It is well known that a more accurate approximation may be obtained in a small neighborhood of the point x if the function y has sufficiently many derivatives and we preserve some higher order terms of the expansion.

More precisely, suppose that the solution y of the initial value problem (2.1) is $(p+1)$ times continuously differentiable on the interval $[a,b]$ and consider a mesh point x_k. Then by Taylor formula we have

$$y(x_k+h) = y(x_k) + \frac{y'(x_k)}{1!}h + \frac{y''(x_k)}{2!}h^2 + \cdots + \frac{y^{(p)}(x_k)}{p!}h^p + R_p, \tag{2.13}$$

where the remainder term R_p has the following form:

$$R_p = \frac{y^{(p+1)}(\xi)}{(p+1)!}h^{p+1} \qquad (x_k < \xi < x_k + h). \tag{2.14}$$

The condition is satisfied with respect to the solution y if the two variable function f is p times continuously differentiable on the strip $[a,b] \times \mathbb{R}$. In this case we can compute the higher derivatives of the function y at the point x_k directly from the given differential equation $y'(x) = f(x, y(x))$.

In order to perform this first introduce the shorthand notations

$$\begin{aligned} f &:= f(x,y), \qquad & f_x &:= \frac{\partial f(x,y)}{\partial x}, \\ f_{xx} &:= \frac{\partial^2 f(x,y)}{\partial x^2}, \qquad & f_{xy} (\equiv f_{yx}) &:= \frac{\partial^2 f(x,y)}{\partial x \partial y} \end{aligned} \tag{2.15}$$

etc. all evaluated at the point $(x_k, y(x_k))$. By differentiating both sides of the differential equation $y'(x) = f(x, y(x))$ in accordance with the chain rule of elementary calculus, we find the relations

$$\begin{aligned} y'(x_k) &= f =: (D^0 f)(x_k, y(x_k)) \\ y''(x_k) &= f_x + f f_y =: (Df)(x_k, y(x_k)), \\ y'''(x_k) &= (f_{xx} + 2f f_{xy} + f^2 f_{yy}) + (f_x + f f_y) f_y =: (D^2 f)(x_k, y(x_k)) \\ &\vdots \end{aligned} \tag{2.16}$$

Therefore from (2.13) we have

$$y(x_k + h) = y(x_k) + \sum_{i=1}^{p} \frac{(D^{i-1} f)(x_k, y(x_k))}{i!} h^i + R_p.$$

If we replace $y(x_k)$ by the approximate value y_k and neglect the remainder term R_{p+1} then for a fixed positive integer p (≥ 1) and for equidistant mesh points we obtain the following *p-term Taylor series method*:

$$\boxed{\begin{aligned} x_0 &:= a, \quad y_0 := \alpha \\ x_{k+1} &:= x_k + h, \qquad h := (b-a)/N, \\ y_{k+1} &:= y_k + \frac{(D^0 f)(x_k, y_k)}{1!} h + \cdots + \frac{(D^{p-1} f)(x_k, y_k)}{p!} h^p \\ &(k = 0, 1, \ldots, N-1), \end{aligned}} \tag{2.17}$$

The increment function Φ of this method is

$$\Phi(x_k, y_k, h) = \sum_{i=1}^{p} \frac{(D^{i-1} f)(x_k, y(x_k))}{i!} h^{i-1}.$$

From (2.13) and (2.14) it follows that under the supposed condition the local truncation error of this procedure is

$$e_{k+1} = O(h^{p+1})$$

and thus *the order of p-term Taylor series method is p.*

We remark that the one term Taylor series method is the Euler's method.

Taylor series methods have the merit that they are self-starting and allow easy changes of step size. They have the disadvantage that they require successive derivatives of the two variable function f to be calculated. The coefficients $(D^j f)(x_k, y_k)$ may be obtained by successively

differentiation of the given differential equation. Since the resulting expression becomes complicated, hand computation of these derivatives is tedious. Fortunately Maple can help us because of its symbolic differentiation facility.

Let us illustrate this by the following example.

EXAMPLE 2.7. *Use the three term Taylor series method to obtain approximations y_k of the initial value problem* (2.4) *at the points* $x_k := 0.1k$ $(k = 0, 1, \ldots, 6)$ *using the step size* $h = 0.01$. *Give also the errors*

$$E_k = y(x_k) - y_k \qquad (k = 0, 1, ..., 6),$$

where $y(x)$ *denotes the exact solution of* (2.4).

SOLUTION. From the EXAMPLE 2.1 we know that the function

```
> ExactSol := x->1/(1+x^2);
```

$$ExactSol := x \to \frac{1}{1+x^2}$$

is the unique solution of (2.4) on the whole real line.

We define the functions $(D^j f)(x, y)$ $(j = 0, 1, 2, \ldots)$ – see (2.16) – in Maple as a three variable function `Df(x,y,j)` in the following way:

```
> f := (x,y) -> -2*x*y^2;
```

$$f := (x,\, y) \to -2\, x\, y^2$$

```
> tt :=
  proc(x,y,j)
    option remember;
    if j=0 then f(x,y)
       else subs(diff(y(x),x)=f(x,y), y(x)=y,
                 diff(tt(x,y(x),j-1),x))
    fi
  end:
> Df := (x,y,j) -> subs(t=x, s=y, tt(t,s,j)):
```

Let us test this function:

```
> for j from 0 to 2 do Df(x,y,j) od;
```

$$-2\,x\,y^2$$

$$-2\,y^2+8\,x^2\,y^3$$

$$24\,y^3\,x-48\,x^3\,y^4$$

(We can check the results by using hand calculations.)

We write a Maple program for the algorithm (2.17). First we give the initial values and the step size

```
> a := 0:  y0 := 1:  h:=0.01:
```

Now, we define the two variable function `TaSeM` in such a way that `TaSeM(k,p)` gives the approximate value of the solution of (2.4) at the point $x_k = a + kh$ by means p-term Taylor series method with the step size h.

```
> TaSeM :=
   proc(k,p)
    option remember;
     if k=0 then y0
       else TaSeM(k-1,p) +
              sum('(subs(x=a+(k-1)*h, y=TaSeM(k-1,p),
                Df(x,y,i-1))*h^i)/i!', 'i'=1..p)
     fi
   end:
```

The approximate values at the points 0.1, 0.2, . . . , 0.6 can be obtained in the following way

```
> for k from 0 to 6 do TaSeM(10*k, 3) od;
```

1
.9900989154
.9615383044
.9174310208
.8620688206
.7999999047
.7352940754

Since we know the exact solution of (2.4) thus we can calculate the (total) error of the applied method:

```
> for k from 0 to 6 do
      ExactSol(a+10*k*h) - TaSeM(10*k,3) od;
```

$$
\begin{gathered}
0 \\
.945\,10^{-7} \\
.1571\,10^{-6} \\
.1719\,10^{-6} \\
.1449\,10^{-6} \\
.953\,10^{-7} \\
.422\,10^{-7}
\end{gathered}
$$

This results illustrate that a better approximation can be attained by means a p-term ($p > 1$) Taylor series method than by the Euler's method (see EXAMPLE 2.3). □

The p-term Taylor series method can be approached in another way. The algorithm (2.17) gives an approximate value y_{k+1} at the point x_{k+1} by the following formula

$$y_{k+1} := y_k + c_1^{(k)} h + c_2^{(k)} h^2 + \cdots + c_p^{(k)} h^p,$$

where y_k is a given approximate value at the point $x_k = x_{k+1} - h$. Here the coefficients $c_i^{(k)}$ $(i = 1, 2, \ldots, p)$ may be calculated by successive differentiation of the given differential equation.

The basic idea of the new approach is to determine the coefficients $c_i^{(k)}$ $(i = 1, 2, \ldots, p)$ in another way. Let us denote by $y(x)$ the exact solution of (2.4) and consider the formula

$$y(x) = y(x_k) + c_1^{(k)}(x - x_k) + c_2^{(k)}(x - x_k)^2 + \cdots + c_p^{(k)}(x - x_k)^p + T_p,$$

where $c_i^{(k)}$ $(i = 1, 2, \ldots, p)$ are unknown numbers. Substitute this into the differential equation of (2.1) and take $x = x_k + h$. Now use the fact that the left- and right-hand sides are equal if and only if the coefficients of the same powers of h are equal. Then comparing the corresponding coefficients results in a set of nonlinear equation from which the unknown coefficients $c_i^{(k)}$ $(i = 1, 2, \ldots, p)$ can be determined recursively.

This approach has the disadvantage that the set of recursion formulae must be found for each new differential equation. However, this task can be done by the computer. Moreover, this method allows us to get a relatively simple control of step size and to control the number of terms which must be considered in order to keep the approximation errors within a prescribed bounds.

The **dsolve** function of Maple contains such a method if we invoke it with the optional equations

`type = numeric` and `method = taylorseries`

This method can be used for solutions with high accuracy. This method will usually take more time than other methods with low accuracy results, therefore it is suggested that it is better to use this method only when a very high degree of accuracy is desired.

EXAMPLE 2.8. *Applying the* **dsolve** *procedure of Maple with the option* `method=taylorseries`*, find an approximate value of the solution of the initial value problem* (2.4) *and compute the global truncation error at the point* 0.4.

SOLUTION. Let us consider the function

```
> ns := dsolve({D(y)(x)=-2*x*y(x)^2, y(0)=1}, y(x),
                type=numeric, method=taylorseries):
```

If `x` is a number then `ns(x)` gives an approximate value of the solution of (2.4), for example at `0.4` we have

```
> ns(0.4);
```

$$[x = .4,\ \mathrm{y}(x) = .8620689655297506]$$

The global truncation error is defined by $E_k := y(x_k) - y_k$. The exact solution of (2.4) is $y(x) = 1/(x^2+1)$ $(x \in \mathbb{R})$, thus for the global truncation error Maple gives

```
> 1/(1+0.4^2) - rhs(ns(0.4)[2]);
```

$$-.125092\,10^{-10}$$

□

• An extrapolation method

The idea of this powerful method is as follows. Suppose that we have calculated an approximate value y_k of the exact solution of the initial value problem (2.1) at the point $x_k \in [a,b]$. We solve (2.1) by means of Euler's method with the step size h and obtain the value

$$y_{k+1}^{(1)} = y_k + hf(x_k, y_k). \tag{2.18}$$

Then the same problem is solved with the step size $h/2$. A double step with the step size $h/2$ produce the values

$$\begin{aligned} y^{(2)}_{k+\frac{1}{2}} &= y_k + \frac{h}{2} f(x_k, y_k), \\ y^{(2)}_{k+1} &= y^{(2)}_{k+\frac{1}{2}} + \frac{h}{2} f(x_k + \frac{h}{2}, y^{(2)}_{k+\frac{1}{2}}). \end{aligned} \tag{2.19}$$

Therefore, we obtain two approximate values $y^{(1)}_{k+1}$ and $y^{(2)}_{k+1}$ at the point x_{k+1}. Eliminating c_1 from the asymptotic expansions

$$\begin{aligned} y^{(1)}_{k+1} &= y(x_{k+1}) + c_1 h + O(h^2), \\ y^{(2)}_{k+1} &= y(x_{k+1}) + c_1 \frac{h}{2} + O(h^2). \end{aligned}$$

we get

$$y(x_{k+1}) = 2y^{(2)}_{k+1} - y^{(1)}_{k+1} + O(h^2), \tag{2.20}$$

and we can define a new approximate value y_{k+1} of the exact solution at the point x_{k+1} by the formula

$$y_{k+1} := 2y^{(2)}_{k+1} - y^{(1)}_{k+1}. \tag{2.21}$$

Using (2.18) and (2.19) y_{k+1} can be written in more convenient form:

$$y_{k+1} = y_k + hf(x_k + \frac{h}{2}, y_k + \frac{h}{2} f(x_k, y_k)).$$

We formulate the result as an algorithm:

$$\boxed{\begin{aligned} x_0 &:= a,\ y_0 := \alpha, \\ x_{k+1} &:= x_k + h,\ h := (b-a)/N, \\ k_1 &:= f(x_k, y_k), \\ k_2 &:= f(x_k + \tfrac{1}{2}h, y_k + \tfrac{1}{2}hk_1), \\ y_{k+1} &:= y_k + hk_2, \\ &(k = 0, 1, \ldots, N-1). \end{aligned}} \tag{2.22}$$

This procedure is called the *modified Euler's method* or the *improved polynomial method*. Using the asymptotic expansion (2.20) it may be proved that *this method is of second order*.

The geometric interpretation of this method is given in Figure 2.6. Namely, a simple step requires the evaluation of the function f for two different pairs of values. The quantity k_1 is equal to the slope of the directional field at the point (x_k, y_k). It serves to determine the

auxiliary point $(x_k + h/2, y^{(2)}_{k+\frac{1}{2}})$ and the corresponding slope k_2. The approximation y_{k+1} is computed by means of this slope, so that the change in the directional field is taken into account.

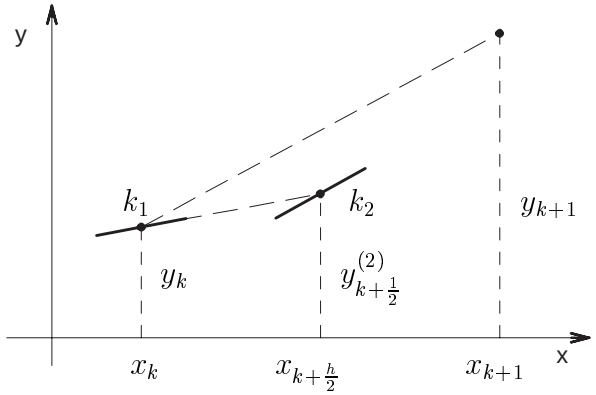


Figure 2.6. Modified Euler's method

Maple's **dsolve** function with the option

`method = classical[impoly]`

allows us to implement the modified Euler's method. More information can be obtained by the command

```
> ?dsolve[classical]
```

Only the available possibilities will be illustrated.

EXAMPLE 2.9. *Solve the initial value problem* (2.4) *on the interval* $[0, 1]$ *by the modified Euler's method of Maple. Choose the step sizes* $h = 0.1,\ 0.01$. *Verify the order of convergence of the method by means of the computed global truncation errors at the points* $x_k := 0.2k$ $(k = 0, 1, \ldots, 5)$.

SOLUTION. First we define `vv` as a function of the step size `h` so that `vv(h)` gives an approximate *solution* of (2.4) by means of the modified Euler's method.

```
> vv := h->dsolve({D(y)(x)=-2*x*y(x)^2, y(0)=1}, y(x),
        type=numeric, output=listprocedure,
        method=classical[impoly], stepsize=h):
```

Now we define the two variable function `ModEuM` in such a way that `ModEuM(mp, h)` gives an approximate *value* of the solution at the point `mp` using step size `h`:

```
> ModEuM := (mp, h)-> subs(vv(h), y(x))(mp):
```

We can quickly compute the numerical values at the mesh points using different step sizes. For example, if $h = 0.1$ then at the points $0, 0.2, 0.4, \ldots, 1.$ we have

```
> for k from 0 to 5 do ModEuM(k*0.2, 0.1) od;
```

$$
\begin{gathered}
1. \\
.9611762976119700 \\
.8611044498912499 \\
.7341796574958591 \\
.6089524203772536 \\
.4996377478773945
\end{gathered}
$$

For the global truncation error we obtain

```
> for k from 0 to 5 do
          1/((k*0.2)^2+1) - ModEuM(k*0.2, 0.1) od;
```

$$
\begin{gathered}
0 \\
.0003621639 \\
.0009645156 \\
.0011144601 \\
.0008036772 \\
.0003622521
\end{gathered}
$$

Now we compute only the global truncation errors at the required mesh points using step size $h = 0.01$.

```
> for k from 0 to 5 do
          1/((k*0.2)^2+1) - ModEuM(k*0.2, 0.01) od;
```

$$
\begin{gathered}
0 \\
.34337\,10^{-5} \\
.87766\,10^{-5} \\
.98154\,10^{-5} \\
.68547\,10^{-5} \\
.29068\,10^{-5}
\end{gathered}
$$

The obtained results suggest that the order of the modified Euler's method is two. □

• The idea of Runge

It was Runge who, in 1895, first pointed out a possibility of evading successive differentiations and of preserving at the same time the increased accuracy afforded by Taylor series. His idea can be formulated as follows.

Consider the initial value problem (2.1) and let us start from the geometrical interpretation of Euler's method. In Figure 2.1, we see that the error of this procedure is large due to the fact that we go along the direction of tangent at the point (x_k, y_k) for a *whole* step, while the solution curve starts to deviate from this direction by a considerable amount during the step.

We hope that we should be able to make a correction for the bending of the curve if we make a *weighted average* of the tangent direction at the points (x_k, y_k) and $(x_{k+1}, y_{k+1}^{(E)})$, where $y_{k+1}^{(E)}$ denotes the approximate value obtained by means of Euler's method at the point x_{k+1}. More generally, we can also average using a smaller step size (say ch, where $0 < c \leq 1$) to compute the approximate value y_{k+1} at the point x_{k+1}. If we introduce the parameters b_1, b_2 and d for the corresponding weights then we obtain the following class of methods:

$$\begin{aligned} x_0 &:= a, \quad y_0 := \alpha, \\ x_{k+1} &:= x_k + h, \quad h := (b-a)/N, \\ k_1 &:= f(x_k, y_k), \\ k_2 &:= f(x_k + ch, y_k + dhk_1), \\ y_{k+1} &:= y_k + h(b_1 k_1 + b_2 k_2), \\ &(k = 0, 1, \ldots, N-1). \end{aligned} \tag{2.23}$$

The central principle of the Runge approach is to choose the parameters b_1, b_2, c and d in such a way that the method (2.23) *has as high an order as possible.*

For the solution of this problem we have to consider the local truncation error of the method (2.23) at the point $x_{k+1} = x_k + h$. It is defined by

$$e_{k+1} = y(x_k + h) - y(x_k) - h(b_1 \tilde{k}_1 + b_2 \tilde{k}_2), \tag{2.24}$$

where

$$\tilde{k}_1 := f(x_k, y(x_k)), \qquad \tilde{k}_2 := f(x_k + ch, y(x_k) + dh\tilde{k}_1)$$

and $y(x)$ $(x \in I)$ denotes the exact solution of (2.1).

We expand the function $y(x)$ $(x \in I)$ into Taylor series about x_k and the two variable function f about $(x_k, y(x_k))$. Using notations (2.15) we have

$$\begin{aligned} y(x_k + h) &= y(x_k) + y'(x_k)h + \frac{y''(x_k)}{2}h^2 + O(h^3) = \\ &= y(x_k) + fh + \frac{f_x + ff_y}{2}h^2 + O(h^3). \end{aligned} \tag{2.25}$$

Expanding the function f in a Taylor series about the point $(x_k, y(x_k))$ gives

$$\tilde{k}_2 = f + (cf_x + dff_y)h + O(h^2). \tag{2.26}$$

Substituting (2.25) and (2.26) into (2.24), we obtain the following expansion for the local truncation error:

$$e_{k+1} = (1 - b_1 - b_2)fh + (\frac{1}{2} - b_2c)f_xh^2 + (\frac{1}{2} - b_2d)ff_xh^2 + O(h^3).$$

We see that order 2 can be achieved by choosing

$$b_1 + b_2 = 1, \qquad b_2c = \frac{1}{2}, \qquad b_2d = \frac{1}{2}$$

resulting in a family of solutions

$$b_1 = 1 - \lambda, \qquad b_2 = \lambda \qquad c = d = \frac{1}{2\lambda}, \tag{2.27}$$

where $\lambda \neq 0$ is a free parameter. A natural question arises: is it possible to get a method of order 3 or not? However, this is not the case because it may be shown that the coefficient of h^3 of the Taylor series for e_{k+1} contains a term that is independent of the four parameters. *Thus the maximal attainable order is* 2 *for the methods* (2.23).

By taking $\lambda = 1$, i.e.

$$b_1 = 0, \qquad b_2 = 1 \qquad c = d = \frac{1}{2},$$

we get the *modified Euler's method*. For $\lambda = 1/2$, i.e.

$$b_1 = b_2 = \frac{1}{2} \qquad c = d = 1,$$

we obtain the so called *improved Euler method* or *Heun's method*.

The **dsolve** function computes an approximate solution of an initial value problem by means of this method if we invoke it with the following options

`type = numeric` and `method = classical[heunform]`

EXAMPLE 2.10. *Solve the initial value problem* (2.4) *on the interval* $[0, 0.6]$ *by means of the built-in Heun's method of Maple with the step size* $h = 0.1$. *Print out the computed values at each step, and the error as calculated with respect to the exact solution.*

SOLUTION. Similarly to the previous example we get

```
> h := 0.1:

> ExactSol := x -> 1/(x^2+1):

> vv:= dsolve({D(y)(x)=-2*x*y(x)^2, y(0)=1}, y(x),
        type=numeric, output=listprocedure,
        method=classical[heunform], stepsize=h):

> HeunSol := mp -> subs(vv, y(x))(mp):
```

Therefore `HeunSol(mp)` represents an approximate value of the solution at the mesh point `mp`.

Now we define an error function and compile a table which contains the mesh points, the values of the exact solution, of the Heun's solution and of the errors.

```
> Err := t -> ExactSol(t) - HeunSol(t):

> mm:=array(1..8,1..4):
        mm[1,1]:=‘point‘:  mm[1,2]:=‘exact sol.‘:
        mm[1,3]:=‘Heun sol.‘:  mm[1,4]:=‘error‘:
      for i from 2 to 8 do
        mm[i,1]:=0.1*(i-2):
        mm[i,2]:=evalf(ExactSol(0.1*(i-2)),7):
        mm[i,3]:=evalf(HeunSol(0.1*(i-2)),7):
        mm[i,4]:=evalf(Err(0.1*(i-2)),7):
      od:

> eval(mm);
```

point	*exact sol.*	*Heun sol.*	*error*
0	1.	1.	0
.1	.9900990	.9900000	.0000990
.2	.9615385	.9613656	.0001729
.3	.9174312	.9172458	.0001854
.4	.8620690	.8619543	.0001147
.5	.8000000	.8000340	−.0000340
.6	.7352941	.7355270	−.0002329

Compare the errors of Heun's method and those of the Euler's method (see EXAMPLE 2.3). □

• An implicit method

Another single step method can be obtained with the aid of a definite integration of the differential equation $y'(x) = f(x, y(x))$ over the interval $[x_k, x_{k+1}]$. Thus we obtain the integral equation

$$y(x_{k+1}) - y(x_k) = \int_{x_k}^{x_{k+1}} f(x, y(x))dx,$$

where $y(x)$ is the unknown function. The value of the integral can be approximated by means of a quadrature formula. For example, using the simple *trapezoidal rule* we get

$$y_{k+1} = y_k + \frac{h}{2}\left(f(x_k, y_k) + f(x_{k+1}, y_{k+1})\right). \tag{2.28}$$

If (x_k, y_k) is known this is an implicit equation for the unknown value y_{k+1}. Therefore we call (2.28) an *implicit method*. Each step requires the solution of a nonlinear equation. (We remark that in the special case of a first order linear differential equation, (2.28) is also a linear equation from which an explicit recursion formula can be obtained.)

For a nonlinear differential equation the implicit equation (2.28) already has the convenient fixed point form. We can use the *Banach fixed point theorem* (see, e.g. [**Sc**]) which in this case states the following. If the function f satisfies a Lipschitz condition, where the Lipschitz

constant L satisfies $(hL/2) < 1$, then the sequence of successive approximation

$$\begin{aligned} y_{k+1}^{[0]} &:= y_k + hf(x_k, y_k), \\ y_{k+1}^{[n]} &= y_k + \frac{h}{2}[f(x_k, y_k + f(x_{k+1}, y_{k+1}^{[n]})] \qquad (n = 1, 2, 3, \cdots). \end{aligned} \tag{2.29}$$

converges to the unique solution of (2.28) which is denoted by y_{k+1}.

Since the value y_{k+1} is only an approximation of $y(x_{k+1})$ we may restrict the fixed point iteration (2.29) to a single step. Thus, by slightly changing the notation, we obtain *Heun's method*:

$$\begin{aligned} y_{k+1}^{[P]} &:= y_k + hf(x_k, y_k), \\ y_{k+1} &= y_k + \frac{h}{2}[f(x_k, y_k + f(x_{k+1}, y_{k+1}^{[P]})]. \end{aligned} \tag{2.30}$$

In this case the explicit first order Euler's method is used to determine a so-called *predicted value* $y_{k+1}^{(P)}$, which is subsequently corrected by means of the implicit method (2.28) to obtain y_{k+1}. The explicit method (2.30) is therefore called a *predictor-corrector method*. We formulate this as an algorithm:

$$\boxed{\begin{aligned} x_0 &:= a, \quad y_0 := \alpha, \\ x_{k+1} &:= x_k + h, \quad h := (b-a)/N, \\ k_1 &:= f(x_k, y_k), \\ k_2 &:= f(x_k + h, y_k + hk_1), \\ y_{k+1} &:= y_k + \tfrac{1}{2}h(k_1 + k_2), \\ &(k = 0, 1, \ldots, N-1). \end{aligned}} \tag{2.31}$$

In order to determine y_{k+1} the average of the two slope k_1 and k_2 of the direction field are used at the points (x_k, y_k) and $(x_{k+1}, y_{k+1}^{[P]})$. It may be proved that the *Heun's method is of second order*.

2.5. Runge–Kutta methods

In the previous section we explained the Runge's idea to obtain a numerical solution of initial value problem (2.1). His method was subsequently improved by K. Heun, in 1900, and W. Kutta, in 1901.

Kutta's proposal (somewhat more general than a similar one made by Heun) consists in considering the following class of single step methods:

$$\begin{aligned} x_0 &:= a, \quad y_0 := \alpha \\ x_{k+1} &:= x_k + h, \quad h := (b-a)/N, \\ k_1 &:= f(x_k + c_1 h, y_k + h \sum_{j=1}^{s} a_{1j} k_j) \\ &\dots \\ k_s &:= f(x_k + c_s h, y_k + h \sum_{j=1}^{s} a_{sj} k_j) \\ y_{k+1} &:= y_k + h \sum_{j=1}^{s} b_i k_i \\ &(k = 0, 1, \dots, N-1), \end{aligned} \tag{2.32}$$

where s is a given positive integer and a_{ij}, b_i, c_i $(i = 1, 2, \dots, s;\ j = 1, 2, \dots, s)$ are undetermined parameters. This procedure is called an *s-stage Runge–Kutta method* for the solution of initial value problem (2.1).

The increment function in this case is given by

$$\Phi := \sum_{i=1}^{s} b_i k_i,$$

where b_i $(i = 1, 2, \dots, s)$ are real numbers and the functions $k_i : I \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ $(i = 1, 2, \dots, s)$ satisfy the following equations

$$k_i(x, y, h) = f(x + c_i h, y + \sum_{j=1}^{s} a_{ij} k_j(x, y, h))$$
$$(i = 1, 2, \dots, s)$$

It is convenient to display the coefficients occurring in (2.32) in the following form, known as a *Butcher array*

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

As we have mentioned in the previous section, the idea behind the Runge approach is to choose the parameters in such a way that the method (2.32) has as high an order as possible. This means that we try to determine the above parameters in such a way that the Taylor expansion of the local truncation error

$$e(x,h) := y(x+h) - y(x) - h\Phi(x, y(x), h)$$

in powers of h starts with the largest possible power. That is for a given positive integer s and a fixed point $x \in [a,b]$ we must determine the constants a_{ij}, b_i, c_i $(i = 1,2,\ldots,s;\ j = 1,2,\ldots,s)$ (which do not depend on f) and the largest possible positive integer p such that

$$\left[\frac{\partial^i e(x,h)}{\partial h^i}\right]_{h=0} = y^{(i)}(x) - i\left[\frac{\partial^{i-1}\Phi(x,y(x),h)}{\partial h^{i-1}}\right]_{h=0} = 0 \qquad (i = 1,2,\ldots,p) \tag{2.33}$$

and

$$\left[\frac{\partial^{p+1} e(x,h)}{\partial h^{p+1}}\right]_{h=0} \neq 0.$$

More detailed calculations show that the p equalities in (2.33) are equivalent to a system of, in general, nonlinear implicit equations for the parameters a_{ij}, b_i, c_i. In many cases the number of the equations obtained is smaller than the number of unknowns. It may be shown that for each s, there will be a largest value of p for which these equations are solvable if the following (the *row-sum condition*) holds:

$$\boxed{c_i = \sum_{j=1}^{s} a_{ij}, \quad (i = 1,2,\ldots,s)} \tag{2.34}$$

which we shall always assume.

We see that the idea behind this approach is simple and natural. The actual derivation, however, is lengthy, and the corresponding calculations become rapidly more complicated as s is further increased. For example, if $s = 5$, Kutta obtains 16 equations in 15 unknowns, and it appears as yet uncertain whether these equations are dependent.

It may also be proved that the general s-stage Runge–Kutta method is consistent (it is a necessary condition for the convergence of a numerical method) if and only if

$$\boxed{\sum_{i=1}^{s} b_i = 1} \tag{2.35}$$

which we shall always assume.

If in (2.32) we have that $a_{ij} = 0$ for $j \geq i$ $(i = 1, 2, \ldots, s)$ then each of the k_i is given explicitly in terms of previously computed k_j $(j = 1, 2, \ldots, i-1)$, and the method is then an *explicit* or *classical Runge–Kutta method*. If this is not the case then the method is *implicit* and, in general, it is necessary to solve at each step of the computation an implicit system for the k_i. Runge–Kutta methods first appeared in 1895, and up to the 1960s only explicit methods were considered.

• Special cases

Now, we consider some special cases of *explicit* Runge–Kutta methods.

For *one-stage* rules ($s = 1$), in view of (2.32) and (2.34), the Runge–Kutta methods take the form

$$y_{k+1} = y_k + hb_1 f(x_k, y_k).$$

From (2.35) we see that the necessary and sufficient condition for these methods to be consistent is $b_1 = 1$. Therefore the consistent one-stage Runge–Kutta method coincides with Euler's method which was examined in Section 2.2. We remark that the *order of one-stage Runge–Kutta method is* 1.

As we have seen in the previous section the maximal attainable order is 2 for the *two-stage* explicit Runge–Kutta methods and there exists a singly infinite family of these methods of order 2.

We illustrate these types of methods solving the initial value problem in (2.1).

EXAMPLE 2.11. *Solve the initial value problem* (2.4) *by means of the two-stage explicit Runge–Kutta method built-in into Maple. Let the step size be—for example—* $h = 0.1$.

SOLUTION. Using Maple help

```
> ?dsolve[classical]
```

we can see that the program can solve an initial value problem by means of the two-stage classical Runge–Kutta method if we invoke the **dsolve** procedure with the options

`type = numeric` and `method = classical[rk2]`

First we define the step size, the exact solution of (2.4) and then invoke the **dsolve** procedure with the above options

```
> h := 0.1:

> ExactSol := x -> 1/(x^2+1):

> vv2 := dsolve({diff(y(x),x)=-2*x*(y(x))^2, y(0)=1},
      {y(x)}, type=numeric, output=listprocedure,
      method=classical[rk2], stepsize=0.1):
```

The value of the approximate solution at a mesh point can be obtained by the following function

```
> RK2_Sol := mp -> subs(vv2, y(x))(mp):
```

For example

```
> RK2_Sol(0.4);
```

$$.8619543198099594$$

We calculate only the errors at the points $0., 0.1, \ldots, 0.6$.

```
> RK2_Err := t -> ExactSol(t) - RK2_Sol(t):

> for k from 0 to 6 do RK2_Err(0.1*k) od;
```

$$0$$
$$.0000990099$$
$$.0001729071$$
$$.0001853854$$
$$.0001146457$$
$$-.0000340251$$
$$-.0002329011$$

Compare these errors with the errors obtained by Euler's method in EXAMPLE 2.3. □

By (2.32) and (2.34) the *three-stage* explicit Runge–Kutta methods can be written as

$$\begin{aligned} x_0 &:= a, \quad y_0 := \alpha \\ x_{k+1} &:= x_k + h, \quad h := (b-a)/N, \\ k_1 &:= f(x_k, y_k), \\ k_2 &:= f(x_k + hc_2, y_k + hc_2 k_1), \\ k_3 &:= f(x_k + hc_3, y_k + h[(c_3 - a_{32})k_1 + a_{32}k_2]) \\ y_{k+1} &:= y_k + h(b_1 k_1 + b_2 k_2 + b_3 k_3) \\ &(k = 0, 1, \ldots, N-1) \end{aligned} \tag{2.36}$$

or in a Butcher array form

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ c_2 & c_2 & 0 & 0 \\ c_3 & c_3 - a_{32} & a_{32} & 0 \\ \hline & b_1 & b_2 & b_3 \end{array}$$

EXAMPLE 2.12. *Derive the three-stage explicit Runge–Kutta methods.*

SOLUTION. In this case we can use Maple to obtain the system of equations (2.33).

The local truncation error of the method (2.36) at the fixed point x is defined by

$$e(x,h) := y(x+h) - y(x) - h(b_1 k_1 + b_2 k_2 + b_3 k_3), \tag{2.37}$$

where

$$\begin{aligned} k_1 := f(x, y(x)), \quad k_2 := f(x + hc_2, y(x) + hc_2 k_1), \\ k_3 := f(x + hc_3, y(x) + h[(c_3 - a_{32})k_1 + a_{32}k_2]) \end{aligned} \tag{2.38}$$

and $y(x)$ ($x \in [a,b]$) denotes the exact solution of (2.1).

Our aim is to give the Taylor's series expansion of $e(x,h)$ as a function of h about the point $h = 0$. To do this we use the Maple's **taylor** procedure, which gives the Taylor's *series* expansion of a function, with respect to its variable, about a given point, up to an also a given order. Then the **convert** function with the option `polynom` may be used to convert the series expansion to a polynomial.

Consider $y(x+h) - y(x)$ as a function of h (x is a fixed point). Then the third-degree Taylor's polynomial of this function about $h = 0$ can be obtained in the following way:

```
> tay_pol_3 :=
      convert(taylor(y(x+h)-y(x), h=0, 4), polynom);
```

$$tay\_pol\_3 := \mathrm{D}(y)(x)\,h + \frac{1}{2}\,(D^{(2)})(y)(x)\,h^2 + \frac{1}{6}\,(D^{(3)})(y)(x)\,h^3$$

Since the function y is a solution of (2.1) thus its derivatives satisfy the relations (2.16). We calculate the derivatives using the following procedure:

```
> tt :=
  proc(x,y,j)
   option remember;
    if j=0 then f(x,y)
      else subs(diff(y(x),x)=f(x,y), y(x)=y,
                diff(tt(x,y(x),j-1),x))
    fi
  end:
```

Now we define the functions $D^j f(x,y)$ (see (2.16))

```
> Df := (x, y, j) -> subs(t=x, s=y, tt(t,s,j)):
```

For $j = 0, 1, 2$ we have

```
> for j from 0 to 2 do simplify(Df(x, y, j)) od;
```

$$\mathrm{f}(x,\,y)$$

$$D_1(f)(x,\,y) + D_2(f)(x,\,y)\,\mathrm{f}(x,\,y)$$

$$D_{1,1}(f)(x,\,y) + 2\,D_{1,2}(f)(x,\,y)\,\mathrm{f}(x,\,y) + D_{2,2}(f)(x,\,y)\,\mathrm{f}(x,\,y)^2 + \\ + D_2(f)(x,\,y)\,D_1(f)(x,\,y) + D_2(f)(x,\,y)^2\,\mathrm{f}(x,\,y)$$

Substituting these formulas into the polynomial `tay_pol_3` we obtain

```
> convert(taylor(y(x+h)-y(x), h=0, 4), polynom):
> for j from 1 to 3 do
          subs( (D@@j)(y)(x)=Df(x, y, j-1), ") od:
> tay_1 := ";
```

$$tay\_1 := \mathrm{f}(x,\,y)\,h + \frac{1}{2}\,(D_1(f)(x,\,y) + D_2(f)(x,\,y)\,\mathrm{f}(x,\,y))\,h^2 + \\ \frac{1}{6}(D_{1,1}(f)(x,\,y) + D_{1,2}(f)(x,\,y)\,\mathrm{f}(x,\,y) + (D_{1,2}(f)(x,\,y) + \\ D_{2,2}(f)(x,\,y)\,\mathrm{f}(x,\,y))\,\mathrm{f}(x,\,y) + D_2(f)(x,\,y)\,(D_1(f)(x,\,y) + \\ D_2(f)(x,\,y)\,\mathrm{f}(x,\,y)))h^3$$

We define the shorthand notation introduced in (2.15)

```
> sn := {f(x,y)=f, D[1](f)(x,y)=fx, D[2](f)(x,y)=fy,
      D[1,1](f)(x,y)=fxx, D[2,2](f)(x,y)=fyy,
      D[1,2](f)(x,y)=fxy}:
```

Thus the expression `tay_1` can be written in the following form

```
> Tywsn:= subs(sn, tay_1);
```

$$Tywsn := f\,h + \frac{1}{2}\,(fx + fy\,f)\,h^2 + \frac{1}{6}\,(fxx + fxy\,f + (fxy + fyy\,f)\,f + fy\,(fx + fy\,f))\,h^3$$

Let us consider the remainder term of (2.37). First define the quantities

```
> k1:=f(x,y):
  k2:=f(x+h*c2,y+h*c2*k1):
  k3:=f(x+h*c3,y+h*((c3-a32)*k1+a32*k2)):
```

The third-degree Taylor's polynomial of the remainder term of (2.37) can be calculated in the following way

```
> tf := convert(taylor(h*(b1*k1+b2*k2+b3*k3), h=0, 4),
        polynom):
```

(We do not display the output.) Substituting `sn` into `tf` we get

```
> Tfwsn := (subs(sn, tf));
```

$$Tfwsn := (b1\ f + b2\ f + b3\ f)\,h + (b3\,(fx\ c3 + fy\ f\ c3) + b2\,(fx\ c2 + fy\ c2\ f))\,h^2 + \\ + (b3\,(\frac{1}{2}\,fxx\ c3^2 + fxy\ f\ c3^2 + \frac{1}{2}\,f^2\ c3^2\,fyy + fy^2\ a32\ c2\ f + fy\ a32\ fx\ c2) \\ + b2\,(\frac{1}{2}\,fxx\ c2^2 + fxy\ c2^2\ f + \frac{1}{2}\,c2^2\ f^2\,fyy))h^3$$

We save the Taylor's expansion of the local truncation error in the new variable

```
> lte := collect(Tywsn - Tfwsn, h):
```

Collecting the coefficients of h, h^2 in the local truncation error we get

```
> eq_h := factor(coeff(lte, h));
```

$$eq\_h := -f\,(-1 + b1 + b2 + b3)$$

```
> eq_h2 := factor(coeff(lte, h, 2));
```

$$eq\_h2 := -\frac{1}{2}\,(-1 + 2\,b3\,c3 + 2\,b2\,c2)\,(fx + fy\,f)$$

In order to simplify the coefficient of h^3 we introduce the notations `F` and `G` defined below

```
> expand(coeff(lte, h, 3)):
> collect(expand(algsubs(fxx+2*fxy*f+fyy*f^2=G,")),G):
> eq_h3:= collect(algsubs(fy*fx+fy^2*f=F,% "),{F,G});
```

$$eq\_h3 := (\frac{1}{6} - \frac{1}{2}\,b2\,c2^2 - \frac{1}{2}\,b3\,c3^2)\,G + (-b3\,a32\,c2 + \frac{1}{6})\,F$$

The obtained results mean that if the six parameters $b_1, b_2, b_3, c_1, c_2, c_3$, a_{32} satisfy the system of four nonlinear equations

$$\begin{aligned} b_1 + b_2 + b_3 &= 1, \\ b_2 c_2 + b_3 c_3 &= \frac{1}{2}, \\ b_2 c_2^2 + b_3 c_3^2 &= \frac{1}{3}, \\ a_{32} b_3 c_2 &= \frac{1}{6} \end{aligned} \tag{2.39}$$

then the coefficients of h, h^2, h^3 in the Taylor series expansion of (2.37) are zeros, i.e. the order of the corresponding methods is at least 3.

The obvious question arises of whether a method of order 4 is possible. However, this is not the case because the coefficient of h^4 of the Taylor series for $e(x, h)$ contains a term that is independent of the six parameters. Thus *the maximal attainable order for explicit three-stage Runge–Kutta methods is* 3. □

The above example shows that there exists a doubly infinite family of explicit three-stage Runge–Kutta methods. A well-known particular case is the *classical three-stage Runge–Kutta method* with Butcher array

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

It is clear that the order of this method is 3.

In a similar way it is possible to show that there exists a doubly infinite family of explicit four-stage Runge–Kutta methods of order 4, none of which has order greater than 4. The best known of these is the *classical four-stage Runge–Kutta method* which has Butcher array

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

Maple also contains the algorithms of the three- and four-stage classical Runge–Kutta methods. We illustrate these methods on the following example.

EXAMPLE 2.13. *Apply the classical three- and four-stage Runge–Kutta methods of Maple to the solution of the initial value problem* (2.4)*, over the interval* $[0, 0.6]$*. Let the step size be* $h = 0.1$*. Compare the computed values with the values of the exact solution at the mesh points* $0, 0.1, \ldots, 0.6$*.*

SOLUTION. First we define the step size and the exact solution

```
> h := 0.1:
> ExactSol := x -> 1/(x^2+1):
```

To obtain the approximate values of the exact solution by means of the three-stage classical Runge–Kutta method we have to use the option `method=classical[rk2]`.

```
> vv3 := dsolve({diff(y(x),x)=-2*x*(y(x))^2, y(0)=1},
      {y(x)}, type=numeric, output=listprocedure,
      method=classical[rk3], stepsize=0.1):
> RK3_Sol := mp -> subs(vv3, y(x))(mp):
```

The computed errors at the points $0, 0.1, \ldots, 0.6$ are

```
> RK3_Err := t -> ExactSol(t) - RK3_Sol(t):
```

```
> for k from 0 to 6 do RK3_Err(0.1*k) od;
```

$$
\begin{gathered}
0 \\
-.0000329901 \\
-.0000617933 \\
-.0000817271 \\
-.0000904622 \\
-.0000883845 \\
-.0000779894
\end{gathered}
$$

For the four-stage classical Runge–Kutta method we use the option `method=classical[rk4]` and then we have

```
> vv4 := dsolve({diff(y(x),x)=-2*x*(y(x))^2, y(0)=1},
      {y(x)}, type=numeric, output=listprocedure,
      method=classical[rk4], stepsize=0.1):
> RK4_Sol := mp -> subs(vv4, y(x))(mp):
> RK4_Err := t -> ExactSol(t) - RK4_Sol(t):
> for k from 0 to 6 do RK4_Err(0.1*k) od;
```

$$
\begin{gathered}
0 \\
.849\,10^{-7} \\
.3178\,10^{-6} \\
.5952\,10^{-6} \\
.7820\,10^{-6} \\
.7910\,10^{-6} \\
.6173\,10^{-6}
\end{gathered}
$$

These results show that the four-stage Runge–Kutta method is more accurate than the three-stage Runge–Kutta method. □

We have mentioned before that there exists a single explicit one-stage Runge–Kutta method of order 1, a singly-infinite family of two-stage methods of order 2, a doubly-infinite family of three-stage methods of order 3 and a doubly-infinite family of four-stage methods of order 4. In 1987, J.C. Butcher showed that there exist no p-stage explicit Runge–Kutta methods of order p for $p \geq 5$. The question of what order can be achieved by an explicit s-stage method is still an open one. For example the following is known: Maximal attainable orders of explicit $s = 5, 6, 7, 8, 9$-stage Runge–Kutta method s are $p = 4, 5, 6, 6, 7$ respectively.

• Implicit Runge–Kutta methods

The numerical solution of differential equations with certain properties, for example stiff equations, require special methods. The implicit Runge–Kutta methods belong to this class, that is characterized by the fact that the slopes $k_1, k_2, \ldots$ are defined by an implicit system of equations. *Heun's method* (2.30) is a special case of an implicit Runge–Kutta method. Much of our discussion of implicit methods will be left to Chapter 4, where the problem of stiffness is addressed.

2.6. Advanced methods

Up to this point we have not discussed how the step size h of the previous methods is to be chosen. Obviously, there is trade-off to be made: If the step size is too small, then computer time is needlessly wasted and accumulation of round-off errors can become a hazard. A large step size invites large truncation error. Therefore the practical use of a numerical method requires convenient techniques for estimating errors. These techniques are used in adaptive implementations of the methods for assessing the appropriateness of the step size being used in the light of the accuracy requirements being imposed.

We examine principles for *step size selection*. Techniques for automatic step size selection are based on estimating the local truncation error at each step and then choosing the step size to keep this estimated error within some tolerance bound.

• Error control

Ideally a numerical method should use the minimum step size to ensure that the global error $|y_k - y(x_k)|$ remains within a specified tolerance $\delta > 0$ for $k = 0, 1, ..., N$. This requirement is inconsistent with using a constant step size h. Now it is impossible, in general, to control the global error. However controlling the local truncation error will, under certain conditions on the initial value problem and numerical method, serve to control the global truncation error.

There are two commonly used techniques for error control with single step methods.

A possible procedure for obtaining error estimates was proposed by Richardson (1927) and called the *deferred approach to the limit* or the *Richardson extrapolation method*. It is an old technique, and one which is applicable to any numerical method. Richardson's extrapolation method consists in repeating the integration from x_k to x_{k+1} but with two half-size steps instead of a single full-size step. A comparison of the results obtained furnishes the error estimate. These types of estimates

work well in practice, and can be successfully used to monitor the step size.

To illustrate this technique of controlling the local truncation error suppose that we have used a Runge–Kutta method of order p to obtain the numerical solution y_{k+1} at x_{k+1}. Let us now compute a second numerical solution at x_{k+1} by applying the same method with step size $h/2$; denote the solution so obtained by $\tilde{y}_{k+1}$. Then using the asymptotic expansion of the local truncation error the following convenient indicator of local truncation error can be obtained

$$e_{k+1} \approx \frac{y_{k+1} - \tilde{y}_{k+1}}{2^{p+1} - 1}.$$

This formula can be used for automatic step size selection. The above idea has been developed by C.W. Gear, J. Stoer and R. Bulirsch.

An other modern approach is to devise special methods which are actually two methods built into one. One of the constituent methods has an order p, say, and the second has an order $p + 1$. The difference of the results computed by these methods provides an error estimate for the order p method.

To illustrate the technique of controlling the local truncation error consider Euler's method

$$y_{k+1} = y_k + hf(x_k, y_k),$$

which has local truncation error

$$e_{k+1} = y(x_{k+1}) - y(x_k) - hf(x_k, y(x_k)).$$

Now the modified Euler's method (2.22)

$$\tilde{y}_{k+1} = \tilde{y}_k + hf(x_k + \frac{1}{2}h, \tilde{y}_k + \frac{1}{2}hf(x_k, \tilde{y}_k))$$

has local truncation error $\tilde{e}_{k+1}$ of order $O(h^2)$. Suppose that

$$\tilde{y}_k \approx y_k \approx y(x_k).$$

Then

$$\begin{aligned} y(x_{k+1}) - y_{k+1} &= y(x_{k+1}) - y_k - hf(x_k, y_k) \\ &\approx y(x_{k+1}) - y(x_k) - hf(x_n, y(k_n)) \\ &= e_{k+1}. \end{aligned}$$

Thus

$$\begin{aligned} e_{k+1} &= y(x_{k+1}) - y_{k+1} \\ &= [y(x_{k+1}) - \tilde{y}_{k+1}] - [\tilde{y}_{k+1} - y_{k+1}] \\ &= \tilde{e}_{k+1} + [\tilde{y}_{k+1} - y_{k+1}]. \end{aligned}$$

But e_{k+1} is $O(h)$ while $\tilde{e}_{k+1}$ is $O(h^2)$, and so the most significant portion of e_{k+1} must be attributed to $[\tilde{y}_{k+1} - y_{k+1}]$. Consequently the local truncation error can be approximated by

$$e_{k+1} \approx [\tilde{y}_{k+1} - y_{k+1}] \tag{2.40}$$

and this formula can be used for the selection of the step size.

Methods of this type have been devised by E. Fehlberg and by a number of other authors.

• Runge–Kutta–Fehlberg method

Now suppose that two discrete methods are available, one with a local truncation error e_{k+1} of order $O(h^p)$ and the second has a local truncation error $\tilde{e}_{k+1}$ of order $O(h^{p+1})$. By a similar analysis to that above equation (2.40) will still apply. However since e_{k+1} is of order $O(h^p)$ a constant c exists such that

$$e_{k+1} \approx ch^p.$$

Thus

$$ch^p \approx [\tilde{y}_{k+1} - y_{k+1}].$$

Now since our intention is to vary the step size to control the local truncation error consider a stepsize qh where $q > 0$. Then

$$e_{k+1}(qh) \approx c(qh)^p = q^p(ch^p) \approx q^p(\tilde{y}_{k+1} - y_{k+1}).$$

To bound the truncation error by δ, choose q so that

$$q^p(\tilde{y}_{k+1} - y_{k+1}) \approx e_{k+1}(qh) \leq \delta.$$

Thus

$$q \leq \left(\frac{\delta}{|\tilde{y}_{k+1} - y_{k+1}|} \right)^{\frac{1}{p}} \tag{2.41}$$

One technique which utilizes (2.41) for error control is the *Runge–Kutta–Fehlberg method* (or RKF45 method) which consists of the order

four explicit method, in Butcher notation

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

to advance the solution and the order five explicit method

$$
\begin{array}{c|cccccc}
0 & & & & & & \\
\frac{1}{4} & \frac{1}{4} & & & & & \\
\frac{3}{8} & \frac{3}{32} & \frac{9}{32} & & & & \\
\frac{12}{13} & \frac{1932}{2197} & -\frac{7200}{2197} & \frac{7296}{2197} & & & \\
1 & \frac{439}{216} & -8 & \frac{3680}{513} & -\frac{845}{4104} & & \\
\frac{1}{2} & -\frac{8}{27} & 2 & -\frac{3544}{2565} & \frac{1859}{4104} & -\frac{11}{40} & \\
\hline
 & \frac{16}{135} & 0 & \frac{6565}{12825} & \frac{28561}{56430} & -\frac{9}{50} & \frac{2}{55}
\end{array}
$$

to estimate the error. This process is known as *embedding.* In practice to avoid too many step size changes q is chosen conservatively as

$$q = \left(\frac{\delta}{2|\tilde{y}_{k+1} - y_{k+1}|} \right)^{\frac{1}{4}} = .84 \left(\frac{\delta}{|\tilde{y}_{k+1} - y_{k+1}|} \right)^{\frac{1}{4}}.$$

Another possibility, as we have seen earlier, to estimate the error control is based on step-doubling (i.e. on Richardson extrapolation), however experience has shown that the above error control is roughly a factor of two more efficient than one based on step doubling . The

RKF45 method seem to be efficient and suitable for tolerances down to about 10^{-7}.

The Maple's **dsolve** procedure uses the above Runge–Kutta–Fehlberg method if we invoke it with the options

`type = numeric` and `method = rkf45`

(Note the default value of the `method` option is just `rkf45`.)

We refer to the on-line help system for immediate help on how to change the various parameters of this method.

EXAMPLE 2.14. *Compute the approximate value y_k of the solution of (2.4) at the point $x_k := 0.4$ by means of the built-in Runge–Kutta–Fehlberg method of Maple. Give also the errors*

$$E_k := y(x_k) - y_k \qquad (k = 0, 1, \ldots, 6).$$

SOLUTION. First we compute the exact solution of the initial value problem (2.4):

```
> eq := diff(y(x), x) = -2*x*y(x)^2;
```

$$eq := \frac{\partial}{\partial x}\,\mathrm{y}(x) = -2\,x\,\mathrm{y}(x)^2$$

```
> in_cond := y(0) = 1;
```

$$in\_cond := \mathrm{y}(0) = 1$$

```
> es:=dsolve({eq, in_cond}, y(x));
```

$$es := \mathrm{y}(x) = \frac{1}{x^2+1}$$

```
> Exact_Sol:= t -> subs(x=t, rhs(e)):
```

At the point $x = 0.4$ we have

```
> Exact_Sol(0.4);
```

$$.8620689655$$

Obtaining the solution by means of RKF45 method we get

```
> RKF45_Meth := dsolve({eq,in_cond},y(x),
      type=numeric, method=rkf45);
```

$$RKF45\_Meth := \mathbf{proc}(rkf45\_x) \ldots \mathbf{end}$$

```
> RKF45_Meth(0.4);
```

$$[x = .4,\ \mathrm{y}(x) = .8620689574901867]$$

```
> RKF45_Sol := x-> rhs(op(2,RKF45_Meth(x)));
```

$$RKF45\_Sol := x \to \mathrm{rhs}(\mathrm{op}(2,\ \mathrm{RKF45\_Meth}(x)))$$

```
> RKF45_Sol(0.4);
```

$$.8620689574901867$$

The errors can be computed in the following way:

```
> Error := x -> Exact_Sol(x) - RKF45_Sol(x):
> for k from 0 to 6 do Error(0.1*k) od;
```

$$0$$

$$.158\,10^{-7}$$

$$.164\,10^{-7}$$

$$.148\,10^{-7}$$

$$.127\,10^{-7}$$

$$.105\,10^{-7}$$

$$.85\,10^{-8}$$

□

• Gear method

C.W. Gear developed a *polynomial extrapolation* method which can be used for the numerical solution of stiff problems (see Chapter 4). The Maple's **dsolve** procedure uses this method if we invoke it with the options

`type = numeric` and `method = gear[polyextr]`

R. Bulirsch and J. Stoer worked out a *rational extrapolation* method to obtain high-accuracy solutions to initial value problems with minimal computational effort. Maple also contains this method in its options

`type = numeric` and `method = gear[bstoer]`

• Continuous Runge-Kutta methods: DVERK78

DVERK78 is another well-known high-order continuous Runge–Kutta method with step size control. This method is based on a 13 stage 7th order formula with 8th order error estimate by Fehlberg. It has been much used for high precision computation, for example in astronomy and it is preferable for tolerances between approximately 10^{-7} and 10^{-13}.

Invoking the **dsolve** function in Maple with the

`type = numeric` and `method = dverk78`

options causes **dsolve** to find a numerical solution with the seven-eight order continuous Runge–Kutta method.

2.7. Stability of single step methods

Any numerical method applied to the initial value problem (2.1) will introduce errors due to discretization and round-off. In most situations the effect of errors in a numerical method does not significantly affect the final results. However, in certain cases it can lead to a serious loss of accuracy. The terms *stability* and *instability* are used to describe this phenomenon. As we have mentioned in Section 1.3 there are two types of instability – *inherent* and *numerical*.

In this section we suppose that the conditions of the Theorem 1.2 are satisfied. From it follows that the initial value problem (2.1) has a unique solution on the interval $[a, b]$. We also suppose that the problem (2.1) is *well-conditioned*. Roughly speaking this means that small perturbations in the stated problem will only lead to small changes in the solutions.

There are several types of numerical stability. Here we consider only two fundamental types of them: *zero-stability* and *absolute stability*. We shall also investigate the stability properties of some of the single step methods.

• Zero-stability

We have already seen that for fixed positive values of h, the errors produced by a *convergent* method increase as x increase, when applied to the initial value problem (2.1). Even when the local error at each step is small, the global error may become large due to accumulation and amplification of the local errors.

We supposed that our problem (2.1) is well-conditioned. We can ask what conditions must be imposed on the method in order that the numerical solution displays a stability property analogous to that

displayed by the exact solution. This leads to the concept of *zero-stability*, which controls the manner in which error accumulate, but only in the *limit as* $h \to 0$.

DEFINITION 2.7. *We say that a single step method of class* (2.2) *is* ***zero stable*** *if, for sufficiently small stepsizes* h*, small perturbations in the starting values produce small perturbations in subsequent values.*

Therefore a single step method (2.2) is zero-stable if for each differential equation satisfies a Lipschitz condition there exist positive constants h_0 and K such that the difference between two different numerical solution y_k and y_k^* of (2.1) satisfies

$$|y_k - y_k^*| \leq K|y_0 - y_0^*|$$

for all $0 \leq h \leq h_0$ and $k = 0, 1, \ldots, N$.

Several comments can be made about the definition.

1. Zero-stability is concerned with what happens in the limit as $h \to 0$.

2. Well-posed is a property of differential equation, zero-stability is a property of numerical method, respectively.

3. Zero-stability does not require convergence, although we will show that the converse is true. The method $y_{k+1} = y_k$, $(k = 0, 1, ..., N-1)$ is zero-stable, but not convergent for any but the trivial problem $y' = 0$.

4. Computers can calculate only with finite precision, so that inevitably round-off errors arise. When we perform calculations by the method using two different rounding procedures—for example using two different computers—this could result two numerical results which being infinitely far apart, no matter how fine the precision. Thus, if the method zero-unstable, the sequence y_k is essentially uncomputable.

Stability is nearly automatic for single step methods as the following theorem shows. (In Chapter 3 we shall see that the situation changes in the case of multistep methods.)

THEOREM 2.2. *If* $\Phi(x, y, h)$ *satisfies a Lipschitz condition, then the single step method given by* (2.2) *is zero-stable.*

It can be seen that for all of the methods discussed earlier, Φ will also satisfy a Lipschitz condition for $0 \leq h \leq h_0$, therefore they are zero-stable.

EXAMPLE 2.15. *Show that the modified Euler's method* (2.22) *for the numerical solution of the initial value problem* (2.1) *is zero-stable.*

SOLUTION. The increment function is

$$\Phi(x, y, h) = f(x + \frac{h}{2}, y + \frac{h}{2} f(x, y)),$$

which is continuous in x and y because f is. Since f satisfies a Lipschitz condition with respect to its second variable (see Theorem 1.2) thus we have

$$|\Phi(x, y, h) - \Phi(x, y^*, h)| =$$

$$= |f(x + \frac{h}{2}, y + \frac{h}{2} f(x, y)) - f(x + \frac{h}{2}, y^* + \frac{h}{2} f(x, y^*)| \leq$$

$$\leq L|y + \frac{h}{2} f(x, y) - y^* - \frac{h}{2} f(x, y^*)| \leq$$

$$\leq L|y - y^*| + L\frac{h}{2}|f(x, y) - f(x, y^*)| \leq$$

$$\leq L(1 + \frac{Lh}{2})|y - y^*|.$$

Thus Φ satisfies a Lipschitz condition in y for $0 \leq h \leq h_0$ and by the last theorem the modified Euler's method is zero-stable. □

• Absolute stability (Linear stability theory)

The concept of zero-stability, and also convergence are concerned with the limiting process as $h \to 0$. In practice, we must compute with a finite number of steps, i.e. with finite, nonzero step size h. In particular we want to know if the errors we introduced at each step (truncation and round-off) have a small or large effect on the answer. What is needed is a stability theory which applies when h takes a fixed non-zero value.

We attempt to define *absolute stability* as follows: A method is *absolute stable* for a given step size h and a given differential equation if the change due to a perturbation of size δ in one of the mesh value y_k is no larger than δ in all subsequent values y_m $m > n$.

But, unfortunately, this definition is strongly dependent on the problem, so we utilize the idea of a *test equation*. The simplest such test equation is the linear scalar differential equation

$$y'(x) = \lambda y(x), \qquad \lambda \in \mathbb{C}, \tag{2.42}$$

with the initial condition $y(x_0) = y_0$, which is simple enough to be analyzed theoretically but still so general that it can present some difficulty for a numerical method.

REMARK. We can draw conclusions about how a method works on the system $dy/dx = Ay$, where A is a constant, diagonalizable matrix, by checking its behavior on test equations with $\lambda =$ an eigenvalue of A. □

Equation (2.42) has as its solution

$$y(x) = y_0 e^{\lambda(x-x_0)},$$

which at $x_k = x_0 + kh$ becomes

$$y(x_k) = y_0 e^{\lambda kh} = y_0 (e^{\lambda h})^k. \tag{2.43}$$

A single step method when applied to (2.42) will lead to a first order difference equation

$$y_{k+1} = R(\lambda h) y_k. \tag{2.44}$$

where R is a function determined by the coefficients in the method. The function R is called the *stability function* of the method. For example in the case of explicit Runge–Kutta methods the stability function R is a polynomial and for implicit Runge–Kutta methods R is a rational function.

It is clear that (2.44) has a solution of the form

$$y_n = c_1 (R(\lambda h))^n, \tag{2.45}$$

where c_1 is a constant to be determined from the initial condition.

Let us first consider the simpler case of a real λ. For $\lambda > 0$ we have $\lambda h > 0$ and $R(\lambda h) > 1$. This means that the approximations, y_k, are computed qualitatively in a correct way. The process of natural and applied sciences that are described by differential equations usually contain exponentially decreasing components. This is the case for $\lambda < 0$. Now y_k decreases like the exact solution $y(x_k)$ if and only if the condition $|R(\lambda h)| < 1$ is satisfied.

If $R(\lambda h)$ is a polynomial in λh, this condition cannot be satisfied for all negative values of λh. Systems of differential equations often have solutions consisting of components that decay exponentially but in an oscillating way. Such components correspond to complex values λ. Now the solution $y(x)$ is complex, and again we have the equation $y(x_{k+1}) = e^{\lambda h} y(x_k)$. The complex multiplier $e^{\lambda h}$ has, in the only case of real interest $Re(\lambda) < 0$, a modulus less than one. The necessary and sufficient condition for the numerical approximations y_k to decrease in absolute value like $y(x_k)$ is therefore given by $|R(\lambda h)| < 1$. This gives a motivation for the next definition

DEFINITION 2.8. *A single step method* (2.2) *is said to be* ***absolutely stable*** *for given* λh, *if for that* λh, $|R(\lambda h)| < 1$, *when the method is*

applied for the test initial value problem (2.42), *and to be* ***absolutely unstable*** *for that* λh *otherwise. The set*

$$B := \{z \in \mathbb{C} : |R(z)| < 1\}$$

is called the ***region of absolute stability.*** *The intersection of* B *with the real axis is called the* ***interval of absolute stability.***

Consequently, the step size $h > 0$ must be chosen in such a way that for $Re(\lambda) < 0$ we have $\lambda h \in B$. In the case of systems of differential equations the step size h must be chosen in such a way that for all decay constants λ_i with $Re(\lambda_i) < 0$ the conditions $\lambda_i h \in B$ are all simultaneous satisfied. If the condition $\lambda h \in B$ is violated the method produces meaningless results, that is the method is unstable.

EXAMPLE 2.16. *Find and sketch the region of absolute stability for*
a) *Euler's method,*
b) *trapezoidal method* (2.30).

SOLUTION. (a) If Euler's method is used for the test equation (2.42), we get

$$y_{k+1} = y_k + h\lambda y_k = (1 + \lambda h)y_k =: R(\lambda h)y_k.$$

Thus the stability function of Euler's method has the form

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

Consequently, Euler's method is absolutely stable in the region

$$|1 + z| < 1,$$

which is a unit circle in the complex z-plane centered at $(-1, 0)$.

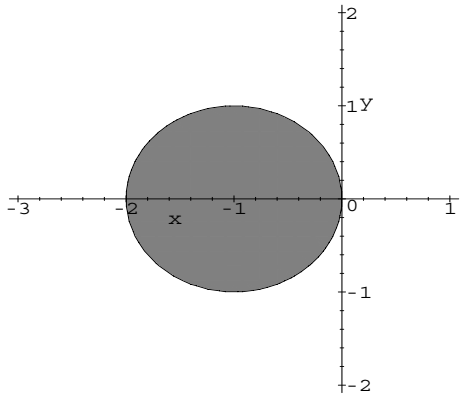
We can draw the region of absolute stability using Maple. First we define the stability function R:

```
> z := x + I*y:
> R := (x, y) -> simplify(evalc(abs(1+z)));
> R(x,y);
```

$$\sqrt{1 + 2\,x + x^2 + y^2}$$

Now we use the Maple's **implicitplot** procedure—which is in the **plots** package to draw the region of absolute stability:

```
> with(plots):
> implicitplot(R(x,y)=1, x=-3..1, y=-2..2,
        filled=true, scaling=CONSTRAINED);
```



(b) The trapezoidal method (2.30) applied to the test equation (2.42) leads us to the explicit computational scheme

$$y_{k+1} = y_k + \frac{h}{2}(\lambda y_k + \lambda y_{k+1}).$$

Thus

$$y_{k+1} = \frac{1 + \frac{1}{2}\lambda h}{1 - \frac{1}{2}\lambda h} y_k =: R(\lambda h) y_k.$$

The stability function of trapezoidal method is

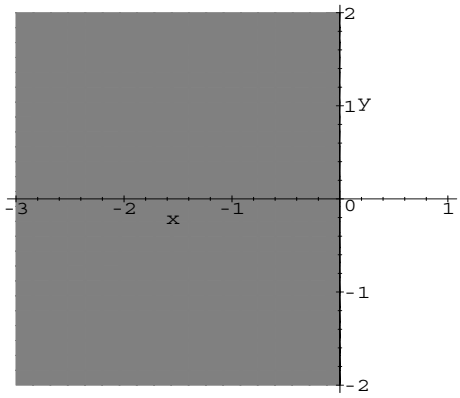
$$R(z) = \frac{1+z}{1-z}.$$

We draw the region of absolute stability as below

```
> z := x + I*y:
> R := (x, y) -> simplify(evalc(abs(1+z+z^2/2))):
> R(x,y);
```

$$\frac{1}{2}\sqrt{4 + 8\,x + 8\,x^2 + 4\,x^3 + 4\,x\,y^2 + x^4 + 2\,x^2\,y^2 + y^4}$$

```
> with(plots):
> implicitplot(R(x,y)=1, x=-3..1, y=-2..2,
      filled=true, scaling=CONSTRAINED);
```



Experiments with different regions of the plot suggests that the region of absolute stability of the trapezoidal method is the open left half complex plane. It is easy to prove that

$$|R(z)| < 1 \quad \text{iff} \quad Re(z) < 0.$$

Thus for the trapezoidal method there is no bound on the step size which has to be taken into account in order to obtain a stable integration. The problem of the choice of h will be treated in connection with systems of stiff differential equations. □

EXAMPLE 2.17. *Find and sketch the absolute stability region for the following second-order Runge–Kutta method:*

$$y_{k+1} = y_k + \frac{1}{2}(k_1 + k_2), \tag{2.46}$$

where

$$k_1 = f(x_k, y_k),$$
$$k_2 = f(x_k + h, y_h + hk_1).$$

SOLUTION. The right-hand side of the equation (2.42) is $f(x, y) = \lambda y$. Applying the above method for (2.42) we have

$$y_{k+1} = y_k + \frac{h}{2}(\lambda y_k + \lambda(y_k + \lambda h y_k)) = \left(1 + \lambda h + \frac{\lambda^2 h^2}{2}\right) y_k.$$

Thus the stability function of (2.46) has the form

$$R(z) = 1 + z + \frac{z^2}{2}$$

and the region of absolute stability of the method (2.46) is given by

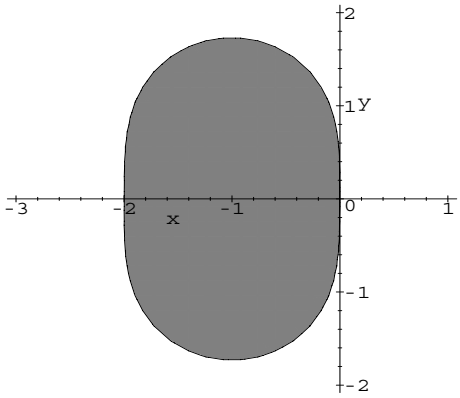
$$B = \{z \in \mathbb{C} \ : \ \left|1 + z + \frac{z^2}{2}\right| < 1\}$$

which is shown below:

```
> z := x + I*y:
> R := (x, y) -> simplify(evalc(abs(1+z+z^2/2))):
> R(x,y);
```

$$\frac{1}{2}\sqrt{4 + 8\,x + 8\,x^2 + 4\,x^3 + 4\,x\,y^2 + x^4 + 2\,x^2\,y^2 + y^4}$$

```
> with(plots):
> implicitplot(R(x,y)=1, x=-3..1, y=-2..2,
      filled=true, scaling=CONSTRAINED);
```



□

2.8. Exercises

1. Use the Taylor's series expansion to find the order p in $O(h^p)$ at $h = 0$ for the function $e^h - \cos h$.
2. Give the solution of the following difference equation

$$y_0 := 1, \quad y_{k+1} = -3y_k \ (k = 0, 1, \ldots).$$

3. Show that the solution of the difference equation $y_0 = 0,\ y_1 = 1,\ \ y_{k+1} = y_k + y_{k-1}\ (k = 0, 1, \dots)$ is

$$y_k = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^k - \left(\frac{1-\sqrt{5}}{2}\right)^k\right) \quad (k = 0, 1, 2, \dots).$$

Try to solve the above difference equation using Maple.

4. Obtain the local truncation error for

   (a) Heun's method,

   (b) the modified Euler's method.

5. Examine the consistency of the methods of Q4.

6. Determine an upper bound for the global truncation error of Euler's method in solving the initial value problem

$$y'(x) = -1000y(x), \qquad y(0) = 1.$$

7. Use the second order Taylor series method to find a numerical solution of the differential equation

$$y'(x) = 1 + x\sin xy(x), \qquad y(0) = 0, \quad 0 \le x \le 1$$

using a stepsize $h = 0.1$.

8. Use the fourth order Taylor series method to find a numerical solution of the differential equation

$$y'(x) = -y(x) + x + 1, \qquad y(0) = 2, \quad 0 \le x \le 2$$

using a stepsize $h = 0.2$.

9. By choosing a suitable value for λ in (2.27) obtain the method

$$y_{k+1} = y_k + \frac{h}{4}\left[f(x_k, y_k) + 3f(x_k + \tfrac{2}{3}h, y_k + \tfrac{2}{3}f(x_k, y_k))\right]$$

and the mid-point method

$$y_{k+1} = y_k + hf\big(x_k + \tfrac{1}{2}h, y_k + \tfrac{1}{2}hf(x_k, y_k)\big).$$

10. Use the methods of Q9 together with the modified Euler's method

$$y_{k+1} = y_k + \frac{h}{2}\left[f(x_k, y_k) + f\big(x_k + h, y_k + hf(x_k, y_k)\big)\right]$$

to find a numerical solution of

$$y'(x) = -y(x) + x^2 + 1, \qquad y(0) = 1, \quad 0 \le x \le 1$$

using a stepsize $h = 0.1$. Which method is most accurate?

11. Explain why all the above methods would give the same solution for the differential equation

$$y'(x) = -y(x) + x + 1, \qquad y(0) = 1.$$

12. Use Maple's **dsolve** procedure to find the exact solution of the following initial value problem

$$y'(x) = x + xy(x), \qquad y(0) = 1.$$

Solve this problem also by means of a) Euler's method, b) four-stage Runge–Kutta method, c) Heun's method, d) RKF45 method.
Experiment with different step sizes h. Compare the results with the exact solution. Compare also the answers obtained by the above methods.

13. Write Maple programs to solve the system of ordinary differential equations

$$\mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y}(x)), \qquad \mathbf{y}(x_0) = \mathbf{y}_0$$

by a) Huen's method and b) the classical 4th order Runge–Kutta method. Test your programs on the systems

i) $y_1'(x) = y_1(x) - y_1(x)y_2(x), \quad y_1(0) = 0.5,$
$y_2'(x) = -y_2(x) + y_1(x)y_2(x), \quad y_2(0) = 0.5;$

ii) $y_1'(x) = y_2(x), \quad y_1(0) = 0,$
$y_2'(x) = -y_1(x) + \varepsilon(1 - y_1^2(x))y_2(x), \quad y_2(0) = 0.5;$

where ε is a parameter. (Try $\varepsilon = 0.5, 1, 5, 10$).

iii) $y_1'(x) = -(y_2(x) + y_3(x)), \quad y_1(0) = 1,$
$y_2'(x) = y_1(x) + 0.2y_2(x), \quad y_2(0) = 1;$
$y_3'(x) = 0.2 - 8y_3(x) + y_1(x)y_3(x), \quad y_3(0) = 1.$

14. Examine the zero-stability of the three-stage classical Runge–Kutta method.

15. Find and sketch the region of absolute stability for the following methods

(a) $y_{k+1} = y_k + hf(x_k, y_{k+1})$ (backward Euler's method),

(b)

$$y_{k+1} = y_k + \frac{h}{4}\left[f(x_k, y_k) + 3f(x_k + \tfrac{2}{3}h, y_k + \tfrac{2}{3}f(x_k, y_k))\right]$$

(c) three-stage classical Runge–Kutta method.

Bibliography

[AB] Abel, M.L. and Braselton, J.P., *Differential Equations with Maple V.* AP Professional, Boston, 1994.

[BeCo] Bellmann, R. and Cooke, K.L., *Differential-Difference Equations*, Academic Press, 1963.

[BG] Birkhoff, G. and Gian-Carlo, Rota, *Ordinary Differential Equations.* John Wiley and Sons, Inc., New York, 1989.

[BD] Boyce, W.E. and DiPrima R.C., *Elementary Differential Equations and Boundary Value Problems.* John Wiley and Sons, Inc., New York, (5th ed.), 1992.

[Bu] Butcher, J.C. *The Numerical Analysis of Ordinary Differential Equations.* John Wiley and Sons, Inc., New York, 1987.

[CL] Coddington, E.A. and Levinson, N., *Theory of Ordinary Differential Equations.* McGraw-Hill Book Company, Inc., New York, 1955.

[Co] Collatz, L., *The Numerical Treatment of Differential Equations.* Springer-Verlag, Berlin-Heidelberg-New York, 1966.

[CB] Conte, S.D. and de Boor, C., *Elementary Numerical Analysis: An Algorithmic Aproach.* McGraw-Hill Kögakusha, Tokyo, (3rd ed.). 1980.

[DB] Dahlquist, G. and Björk, Å., *Numerical Methods.* Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1994.

[Dri] Driver, R.D., *Ordinary and delay differential equations*, Applied Math. Sciences 20, Springer Verlag, 1977.

[EW] Eldén, L. and Wittmeyer-Koch, L., *Numerical Analysis, An Introduction.* Academic Press, Inc., Boston, 1990.

[Ge1] Gear, C.W., *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1971.

[Ge2] Gear, C.W., *The automatic integration of ordinary differential equations.* Comm. ACM **14** (Mar. 1971), pp. 176-179.

[Ge3] Gear, C.W., *The automatic integration of stiff ordinary differential equations*, Information Processing 68, A.J.H. Morrell, Ed., North Holland, Amsterdam, 1969, pp. 187-193.

[Ge4] Gear, C.W., *The numerical integration of ordinary differential equations*, Math. Comp. **21,** 2 (Apr. 1967), pp. 146-156.

[H1] Hairer, E., Nørsett, S.P. and Wanner, G., *Solving Ordinary Differential Equations I. Nonstiff Problems.* Springer Verlag, Berlin, (2nd ed.), 1991.

[H2] Hairer, E., Wanner, G., *Solving Ordinary Differential Equations II. Stiff Problems and Differential-algebraic Equations.* Springer Verlag, Berlin, 1991.

[HW] Hall, G. and Watt, J.M. (Eds.), *Modern Numerical Methods for Ordinary Differential Equations.* Clarendon Press, Oxford. 1976.

[Ha] Hamming, R.W., *Numerical Methods for Scientists and Engineers.* McGraw-Hill Book Company, Inc., New York, (2nd ed.) 1973.

[Hr] Hartman, Ph., *Ordinary Differential Equations.* John Wiley and Sons, Inc., New York, 1964.

[HH] Hämmerlin, G. and Hoffmann, K-H., *Numerical Mathematics.* Springer-Verlag, New York Inc., 1991.

[Hc] Heck, A., *Introduction to Maple.* Springer-Verlag, New York, 1993.

[He] Henrici, P., *Discrete Variable Methods in Ordinary Differential Equations.* John Wiley and Sons, Inc., New York, 1962.

[HJ] Higham, N. J., *Accuracy and Stability of Numrical Algorithms*, SIAM, Philadelphia, 1996.

[Hi] Hildebrand, F.B., *Introduction to Numerical Analysis.* McGraw-Hill Book Company, New York, 1974.

[Hi1] Hindmarsh, A.C., *LSODE and LSODI, two new initial value ordinary differential equation solvers*, ACM-SIGNUM Newsletter 15, 1980, pp. 10-11.

[Hi2] Hindmarsh, A.C., *GEAR: ordinary differential equation system solver*, UCID-30001, Rev. 2, LLL, Livermore, Calif. 1972.

[Hi3] Hindmarsh, A.C., *ODEPACK, a Systemized Collection of ODE Solvers*, In: Scientific Computing, R.S. Stepleman et al. (eds.) North-Holland, Amsterdam, 1983.

[IK] Isaacson, E. and Keller, H.B., *Analysis of Numerical Methods.* John Wiley and Sons, Inc., New York, 1966.

[Is] Iserles, A., *A First Course in the Numerical Analysis of Differential Equations.* Cambridge Text in Applied Mathematics. Cambridge Univ. Press., 1996.

[Ka] Kamke, E., *Differentialgleichungen, Lösungsmethoden und Lösungen, Vol. 1.* Leipzig, 1959.

[KM] Kopchenova, N.V. and Maron, I.A., *Computational Mathematics, Worked Examples and Problems with Elements of Theory.* Mir Publishers, Moscow, 1975.

[La] Lambert, J.D., *Numerical Methods for Ordinary Differential Systems.* John Wiley and Sons, Ltd., Chichester, 1991.

[No] Nordsieck, A., *On numerical integration of ordinary differential equations*, Math. Comp. **16,** 1 (Jan. 1962), pp. 22-49.

[PT] Press, W.H., Teukolsky, S.A., Vatterling, W.T. and Flannery, B.P., *Numerical Recipes in C. The Art of Scientific Computing.* Second Ed., Cambridge Univ. Press, 1992.

[RR] Ralston, A. and Rabinowitz, P., *A First Course in Numerical Analysis.* McGraw-Hill Book Company, New York, 1978.

[Sc] Schwarz, H.R., *Numerical Analysis, A Comprehensive Introduction.* John Wiley and Sons, Ltd., Chichester, 1989.

[SG] Shampine, L.F. and Gordon, M.K., *Computer Solution of Ordinary Differential Equations.* W.H. Freeman, San Francisco, 1975.

[St] Stetter, H.J., *Analysis of Discretization Methods for Ordinary Differential Equations.* Springer Tracts in Natural Philosophy. Vol. 23, Springer Verlag, Berlin, 1973.

[SB] Stoer, J. and Bulirsch, R., *Introduction to Numerical Analysis.* Springer Verlag, Berlin, 1980

[YS] Yakowitz, S. and Szidarovszky, F., *An Introduction to Numerical Computations.* Macmillan Publ. Comp., New York, 1986.

CHAPTER 3

Linear Multistep Methods

3.1. Basic concepts

Consider the first-order scalar initial value problem

$$y'(x) = f(x, y(x)), \qquad y(a) = \alpha, \tag{3.1}$$

where a and α are given real values and suppose that it has a unique solution $y(x)$ on the bounded interval $I := [a, b] \subset \mathbb{R}$.

In this chapter our aim is to determine approximate values y_k of the exact value $y(x_k)$ using *multistep methods* at the equidistant *mesh points* of the interval I

$$x_k := a + kh \qquad (k = 0, 1, 2, \ldots, N),$$

where $h := (b - a)/N$ is the *step size*.

As we have seen in Chapter 2 a single step method determines an approximation of the exact solution of (3.1) at a mesh point solely on the basis of the approximation at the *previous* mesh point. In contrast multistep methods utilize the approximation at *more then one previous* mesh points to calculate the approximation at the next point.

Here we study only *linear multistep methods* (these are also called *m-step linear methods*) which have the following general form:

$$\sum_{j=0}^{m} \alpha_j y_{k+j} = h \sum_{j=0}^{m} \beta_j f(x_{k+j}, y_{k+j}), \quad (k = 0, 1, 2, \ldots, N - m), \tag{3.2}$$

where $m < N$ is a given positive integer, α_j and β_j $(j = 0, 1, \ldots, m;$ $\alpha_m \neq 0)$ are given real numbers (the *parameters of the method*). Thus the approximate value y_{k+m} depends on m previous values $y_k, y_{k+1}, \ldots,$ y_{k+m-1}. In order to generate the sequence of approximations $(y_k,\ k \in \mathbb{N})$ it is first necessary to obtain starting values $y_0, y_1, \ldots, y_{m-1}$. They may be determined, for example, by a single step method. Using the shorthand notations

$$f_i := f(x_i, y_i) \quad (i = 0, 1, 2, \ldots)$$

the general m-step linear methods can be written in the form

$$\boxed{\begin{array}{c} y_0 := \alpha, y_1, \ldots, y_{m-1}, \\ \displaystyle\sum_{j=0}^{m} \alpha_j y_{k+j} = h \sum_{j=0}^{m} \beta_j f_{k+j}, \\ (k = 0, 1, 2, \ldots, N - m). \end{array}} \tag{3.3}$$

If $\beta_m = 0$ the method is said to be *explicit* because y_{k+1} can be expressed explicitly. If $\beta_m \neq 0$ then the method is *implicit* and leads to a nonlinear equation for y_{m+k}.

REMARK. To define a linear multistep method it is necessary to prescribe the parameters m, h, α_j, β_j $(j = 0, 1, \ldots, m)$ and the starting values $y_0, y_1, \ldots, y_{m-1}$. Then y_k $(k = m, m+1, \ldots, N)$ can be computed using (3.3) which is also called a *difference equation* for the unknown values y_k $(k = m, m + 1, \ldots, N)$. □

It is clear that the following single step methods

$$\begin{aligned} y_{k+1} &= y_k + h f_k && \text{(explicit Eulers' method)} \\ y_{k+1} &= y_k + h f_{k+1} && \text{(implicit Eulers' method)} \\ y_{k+1} &= y_k + \frac{h}{2}(f_k + f_{k+1}) && \text{(trapezoidal method)} \end{aligned}$$

are special cases of the class of 1-step linear methods.

Linear multistep methods (3.3) can be directly generalized to systems of differential equations and therefore also to higher-order differential equations. The methods and results for initial value problems for systems of ordinary differential equations of first-order are essentially independent of the number of unknown functions. In the following we therefore limit ourselves for simplicity and clarity to the case of only one ordinary differential equation of first-order for a single unknown function.

3.2. Polynomial interpolation

To introduce linear multistep methods we need some fundamental facts from the theory of polynomial interpolation.

Let $x_1, x_2, \ldots, x_n$ denote distinct real numbers (they are called *nodal points*) and let $y_1, y_2, \ldots, y_n$ be arbitrary real numbers. The points (x_j, y_j) $(j = 1, 2, \ldots, n)$ can be imagined to be data values to be connected by a curve.

It is easy to prove that there exists a uniquely determined *interpolation polynomial*

$$P_{n-1}(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

of degree at most $(n-1)$ satisfying the *interpolation conditions*

$$P_{n-1}(x_j) = y_j, \qquad (j = 1, 2, \ldots, n).$$

This polynomial can be written in the *Lagrangian form*:

$$P_{n-1}(x) = \sum_{j=1}^{n} y_j l_j(x), \tag{3.4}$$

where

$$l_j(x) = \frac{(x - x_1) \cdots (x - x_{j-1})(x - x_{j+1}) \cdots (x - x_n)}{(x_j - x_1) \cdots (x_j - x_{j-1})(x_j - x_{j+1}) \cdots (x_j - x_n)} \qquad (j = 1, 2, \ldots, n) \tag{3.5}$$

are *Lagrange's fundamental polynomials* with respect to the nodal points $x_1, \ldots, x_n$.

Fortunately the Maple's **interp** procedure helps us to compute interpolation polynomials.

EXAMPLE 3.1. *Using the* **interp** *function of Maple find the interpolating polynomial of degree* ≤ 4 *for the data*

$$(-2, -9), \quad (-1, -1), \quad (0, 1), \quad (1, 3), \quad (2, 11).$$

Also, check the result.

SOLUTION. First we define the data in Maple

```
> points := [-2, -1, 0, 1, 2]:
  values := [-9, -1, 1, 3, 11]:
```

$$points := [-2,\ -1,\ 0,\ 1,\ 2]$$

$$values := [-9,\ -1,\ 1,\ 3,\ 11]$$

Now we invoke the **interp** procedure to obtain the required interpolation polynomial:

```
> ip := interp(points, values, x);
```

$$ip := x^3 + x + 1$$

The value of this expression, for example, at the point $x = -2$ can be obtained using the substitution procedure **subs**

```
> subs(x=-2, ip);
```

$$-9$$

In order to compute the value of the above polynomial at any point in a more convenient way we define the following *function*

```
> Int_Pol := proc(t)
              local z;
                subs(z=t, interp(points, values, z));
             end:
```

The asked polynomial is

```
> Int_Pol(x);
```

$$x^3 + x + 1$$

and, for example, at $x = -1$ we get

```
> Int_Pol(-1);
```

$$-1$$

The values of the polynomial `Int_Pol` at the `points` may be obtained using the Maple's **map** procedure

```
> points;
  map(Int_Pol, points);
```

$$[-2,\ -1,\ 0,\ 1,\ 2]$$

$$[-9,\ -1,\ 1,\ 3,\ 11]$$

The output shows that the polynomial `Int_Pol` satisfies the required interpolation conditions.

Let us remark that the resulting polynomial is only of the degree 3. □

EXAMPLE 3.2. *Find Lagrange's fundamental polynomials with respect to the nodal points* $-2, -1, 0, 1, 2$.

SOLUTION. We define the nodal points in the Maple's variable `points` and take the `values` as parameters denoted by `y(1),...,y(5)`.

```
> points := [-2, -1, 0, 1, 2];
  values := [y(1), y(2), y(3), y(4), y(5)];
```

$$points := [-2, -1, 0, 1, 2]$$

$$values := [\mathrm{y}(1), \mathrm{y}(2), \mathrm{y}(3), \mathrm{y}(4), \mathrm{y}(5)]$$

Now we construct the interpolation polynomial collecting its terms with respect to `y(1),...,y(5)`.

```
> pp := collect(interp(points, values, x),
        {y(1), y(2), y(3), y(4), y(5)});
```

$$\begin{aligned} pp := {} & (-\frac{1}{24}\,x^2 - \frac{1}{12}\,x^3 + \frac{1}{12}\,x + \frac{1}{24}\,x^4)\,\mathrm{y}(1) \\ & + (-\frac{2}{3}\,x + \frac{1}{6}\,x^3 + \frac{2}{3}\,x^2 - \frac{1}{6}\,x^4)\,\mathrm{y}(2) \\ & + (-\frac{5}{4}\,x^2 + 1 + \frac{1}{4}\,x^4)\,\mathrm{y}(3) \\ & + (\frac{2}{3}\,x - \frac{1}{6}\,x^4 + \frac{2}{3}\,x^2 - \frac{1}{6}\,x^3)\,\mathrm{y}(4) \\ & + (-\frac{1}{12}\,x + \frac{1}{12}\,x^3 - \frac{1}{24}\,x^2 + \frac{1}{24}\,x^4)\,\mathrm{y}(5) \end{aligned}$$

The Lagrange's fundamental polynomial with respect to the point $x_1 = -2$ can be obtained using the `coeff` procedure

```
> coeff(pp, y(1));
```

$$-\frac{1}{24}\,x^2 - \frac{1}{12}\,x^3 + \frac{1}{12}\,x + \frac{1}{24}\,x^4$$

Finally we get

```
> for i from 1 to 5 do
        sprintf(`l(%f)=`, i), coeff(pp, y(i)) od;
```

$$l(1) = -\frac{1}{24}\,x^2 - \frac{1}{12}\,x^3 + \frac{1}{12}\,x + \frac{1}{24}\,x^4$$

$$l(2) = -\frac{2}{3}\,x + \frac{1}{6}\,x^3 + \frac{2}{3}\,x^2 - \frac{1}{6}\,x^4$$

$$l(3) = -\frac{5}{4}\,x^2 + 1 + \frac{1}{4}\,x^4$$

$$l(4) = \frac{2}{3}\,x - \frac{1}{6}\,x^4 + \frac{2}{3}\,x^2 - \frac{1}{6}\,x^3$$

$$l(5) = -\frac{1}{12}\,x + \frac{1}{12}\,x^3 - \frac{1}{24}\,x^2 + \frac{1}{24}\,x^4$$

□

3.3. Classical linear multistep methods

In this section we introduce the historically oldest variants of the linear multistep methods.

• Adams–Bashforth methods

In 1883, Adams and Bashforth gave an improvement of the Euler's method. Their idea to obtain a numerical solution of initial value problem (3.1) is based on integration of an interpolating polynomial. Let us take a fixed number k and consider the equation

$$y(x_{k+1}) - y(x_k) = \int_{x_k}^{x_{k+1}} f(t, y(t))dt. \tag{3.6}$$

which is obtained by integrating the differential equation in (3.1) over the interval $[x_k, x_{k+1}]$. In general no primitive for the right-hand side of (3.6) can be found, because $y(x)$ is an unknown function. Therefore the value of the integral has to be approximated.

Fix the positive integer m and suppose that $k \geq m$. Denote by $P_{m-1,k}(x)$ the uniquely determined polynomial of degree $\leq m-1$ for the following m support points: $(x_k, f(x_k, y(x_k)))$, $(x_{k-1}, f(x_{k-1}, y(x_{k-1})))$, $\cdots (x_{k+1-m}, f(x_{k+1-m}, y(x_{k+1-m})))$. Now the integrand in (3.6) can be replaced by the interpolating polynomial

$$P_{m-1,k}(x) = \sum_{j=1}^{m} f(x_{k-m+j}, y(x_{k-m+j}))l_j(x),$$

where $l_j(x)$ $(j = 1, 2, \ldots, m)$ are Lagrange's fundamental polynomials with respect to the nodal points $x_{k+1-m}, \ldots, x_k$. Thus, we obtain the following approximate formula (see Figure 3.1)

$$\begin{aligned} y(x_{k+1}) - y(x_k) &\approx \int_{x_k}^{x_{k+1}} P_{m-1,k}(t)dt = \\ &= \sum_{j=1}^{m} f(x_{k-m+j}, y(x_{k-m+j})) \int_{x_k}^{x_{k+1}} l_j(t)dt. \end{aligned} \tag{3.7}$$

Observe that the coefficients of $f(x_{k-m+j}, y(x_{k-m+j}))$ in (3.7) do not depend on f. The importance of this fact is obvious. Therefore, replacing in the above formula all exact values $y(x_i)$ by approximate values y_i, $f(x_i, y(x_i))$ by $f_i := f(x_i, y_i)$ and $\approx$ by an equality sign, we obtain the *m-step Adams–Bashforth method*

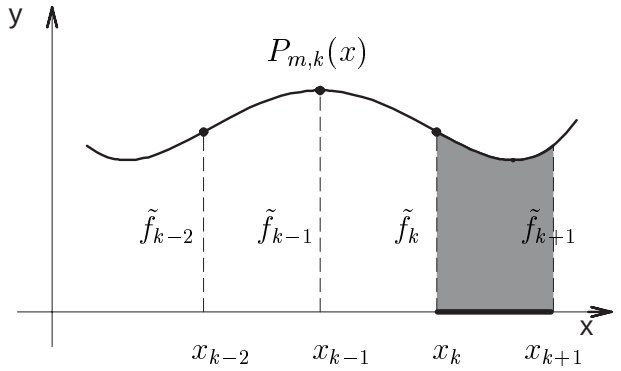
$$\boxed{\begin{array}{c} y_0 := \alpha, y_1, \ldots, y_{m-1}, \\ y_{k+1} = y_k + h\left(\beta_{m1} f_{k+1-m} + \beta_{m2} f_{k+2-m} + \cdots + \beta_{mm} f_k\right), \\ (k = m-1, m, m+1, \cdots), \end{array}} \tag{3.8}$$

where the *coefficients* β_{mj} $(j = 1, 2, \ldots, m)$ of the method are given by

$$h\beta_{mj} = \int_{x_k}^{x_{k+1}} l_j(t)dt$$

and they only depend on the mesh points.

These methods have the feature that they are *explicit*. The reader may wish to check that the 1-step Adams–Bashforth method is the well known explicit *Euler's method*.



$$\tilde{f}_{k-j} := f(x_{k-j}, y(x_{k-j}))$$

Figure 3.1 Approximation of the integral

In order to be able to use an m-step method (for $m > 1$), besides the initial condition $y_0 := \alpha$, $(m-1)$ more initial values $y_1, y_2, \cdots, y_{m-1}$ are necessary. They must be determined by other methods, e.g. with the aid of a single step method. The Runge–Kutta procedures are quite suitable and possibly a smaller step size $h_0 \leq h$ is adequate.

We recall that (see Section 2.3) the *local truncation error* of a method at the point x_{k+1} is the difference between the true solution

and the solution provided by one step of the method starting from the exact values, i.e. in the present case

$$e(x_{k+1},h) := y(x_{k+1}) - y(x_k) - h\sum_{j=1}^{m}\beta_{mj}f(x_{k-m+j},y(x_{k-m+j})).$$

A simple calculation is enough to see that

$$e(x_{k+1},h) = O(h^{m+1})$$

and this means that *each type of m-step Adams–Bashforth methods has the order* $p = m$.

The order can simply be increased whereby the property that each integration step requires only one function evaluation is maintained. This minimal amount of effort, explains why Adams–Bashforth methods are often used.

The next example illustrates how the coefficients of Adams–Bashforth methods can be determined.

EXAMPLE 3.3. *Find the coefficients of the m-step Adams–Bashforth method for* $m = 2, 3, 4, 5$.

SOLUTION. Consider the case $m = 2$ and introduce the notation

$$\tilde{f}_{k-1} := f(x_{k-1},y(x_{k-1})), \quad \tilde{f}_k := f(x_k,y(x_k)).$$

The interpolation polynomial through the points $(x_{k-1},\tilde{f}_{k-1})$, $(x_k,\tilde{f}_k)$ can be written in the form

$$P_{1,k}(x) = \frac{x-x_{k-1}}{x_k-x_{k-1}}\tilde{f}_k + \frac{x-x_k}{x_{k-1}-x_k}\tilde{f}_{k-1}.$$

Thus (3.7) gives

$$y(x_{k+1}) - y(x_k) \approx \int_{x_k}^{x_{k+1}} P_{1,k}(t)dt = h\left(-\frac{1}{2}\tilde{f}_{k-1} + \frac{3}{2}\tilde{f}_k\right). \tag{3.9}$$

If in (3.9) we replace $y(x_i)$ by the approximate values y_i, $\tilde{f}_i$ by $f_i := f(x_i,y_i)$ and $\approx$ by an equality sign, then for $k = 1, 2, \cdots$ we obtain

$$y_{k+1} = y_k + \frac{h}{2}(-f_{k-1} + 3f_k),$$

i.e. $\beta_{21} = -\frac{1}{2}$ and $\beta_{22} = \frac{3}{2}$.

In order to obtain a numerical method for the solution of the initial value problem (3.1) we have to provide two starting values y_0 and y_1. Taking $y_0 := \alpha$ and $y_1 := y_0 + hf_0$, where $f_0 := f(a,y_0)$, we get the two-step Adams–Bashforth method

$$
\begin{aligned}
x_0 &:= a, \quad y_0 := \alpha, \\
x_{k+1} &:= x_k + h, \quad h := (b-a)/N, \\
f_0 &:= f(x_0, y_0), \\
y_1 &:= y_0 + h f_0, \\
y_{k+1} &:= y_k + h(-\frac{1}{2} f_{k-1} + \frac{3}{2} f_k), \\
f_{k+1} &:= f(x_{k+1}, y_{k+1}), \\
&(k = 1, 2, \ldots, N-1).
\end{aligned}
\tag{3.10}
$$

In order to derive the coefficients in the case of $m = 3, 4, 5$ let us investigate the right-hand side of (3.7) using Maple.

First we define a function which gives the array of the fundamental points

$$(x_{k+1-m}, \cdots, x_{k-1}, x_k)$$

of the interpolation. (In the following commands `x` denotes the mesh point x_k.)

```
> fpoints := proc(m)
              local i, vv:
                vv:=array(1..m):
                   for i to m do vv[i]:=x-(m-i)*h od:
              eval(vv)
             end:
```

For example if $m = 4$ we have

```
> fpoints(4);
```

$$[x - 3\,h,\ x - 2\,h,\ x - h,\ x]$$

Using the notation `f(i)` for $f(x_{k+i}, y(x_{k+i}))$ we define the array of the function values in a similar way

```
> fvalues := proc(m)
              local i, vv:
                vv:=array(1..m):
                     for i to m do vv[i]:=f(i-m) od:
              eval(vv)
            end:
```

For $m = 4$ we obtain

```
> fvalues(4);
```

$$[\mathrm{f}(-3),\ \mathrm{f}(-2),\ \mathrm{f}(-1),\ \mathrm{f}(0)]$$

For $m = 4$ the integral in (3.7) can be computed in the following way

```
> ip := interp(fpoints(4), fvalues(4), z):
  int(ip, z=x..x+h):
  simplify(");
```

$$\frac{1}{24}\,h\,(-9\,\mathrm{f}(-3) + 55\,\mathrm{f}(0) - 59\,\mathrm{f}(-1) + 37\,\mathrm{f}(-2))$$

Now we determine the integral in (3.7) for $m = 1, 2, 3, 4, 5$.

```
> for i to 5 do simplify(
  int(interp(fpoints(i),fvalues(i),z),z=x..x+h)) od;
```

$$\mathrm{f}(0)\,h$$

$$\frac{1}{2}\,h\,(3\,\mathrm{f}(0) - \mathrm{f}(-1))$$

$$\frac{1}{12}\,h\,(23\,\mathrm{f}(0) - 16\,\mathrm{f}(-1) + 5\,\mathrm{f}(-2))$$

$$\frac{1}{24}\,h\,(-9\,\mathrm{f}(-3) + 55\,\mathrm{f}(0) - 59\,\mathrm{f}(-1) + 37\,\mathrm{f}(-2))$$

$$-\frac{h}{720}\,(-2616\,\mathrm{f}(-2) + 2774\,\mathrm{f}(-1) - 251\,\mathrm{f}(-4) - 1901\,\mathrm{f}(0) + 1274\,\mathrm{f}(-3))$$

From the results one can see that the coefficients of `f(i)` do not depend on `x`, i.e. on the index k. The importance of this fact is obvious. In order to use a method of such type we have to store only those coefficients which are collected in Table 3.1. □

m	β_{m1}	β_{m2}	β_{m3}	β_{m4}	β_{m5}
1	1				
2	$-\frac{1}{2}$	$\frac{3}{2}$			
3	$\frac{5}{12}$	$-\frac{16}{12}$	$\frac{23}{12}$		
4	$-\frac{9}{24}$	$\frac{37}{24}$	$-\frac{59}{24}$	$\frac{55}{24}$	
5	$\frac{251}{720}$	$-\frac{1274}{720}$	$\frac{2616}{720}$	$-\frac{2774}{720}$	$\frac{1901}{720}$

Table 3.1 Coefficients for Adams–Bashforth methods

Maple includes an Adams–Bashforth method procedure as illustrated in the following example.

EXAMPLE 3.4. *Solve the initial value problem*

$$y'(x) = -2xy^2(x), \quad y(0) = 1 \tag{3.11}$$

by means of the built-in Adams–Bashforth method of Maple. Take the step size $h = 0.1$*, and print the difference between* y_k *and the exact value* $y(x_k)$ *at the mesh points* $x_k = kh$*,* $k = 0, 1, \ldots, 6$*.*

SOLUTION. Invoking the **dsolve** procedure with the options

`type = numeric` and `method = classical[adambash]`

and using 16-digits floating-point arithmetic we get

```
> Digits := 16:
> AB_1 := dsolve({diff(y(x),x)=-2*x*(y(x))^2, y(0)=1},
        {y(x)}, type=numeric,
        method=classical[adambash], stepsize=0.1);
```

$$AB\_1 := \mathbf{proc}(x\_classical) \ \ldots \ \mathbf{end}$$

```
> AB_1(0.4);
```

$$[x = .4,\ \mathrm{y}(x) = .8623885930937653]$$

Now we define the function `AB_Sol` which gives the *value* of the approximate solution at an arbitrary mesh point.

```
> AB_Sol := x -> rhs(op(2, AB_1(x))):  AB_Sol(0.4);
```

$$.8623885930937653$$

The exact solution of (3.11) is

```
> es := dsolve({diff(y(x),x)=-2*x*(y(x))^2, y(0)=1},
            y(x));
```

$$es := \mathrm{y}(x) = \frac{1}{x^2+1}$$

In order to compute the value of this expression at different points we need the following function

```
> Exact_Sol := proc(t)
                  subs(x=t, rhs(es));
               end:
```

```
> Exact_Sol(x);
```

$$\frac{1}{x^2+1}$$

```
> Exact_Sol(0.4);
```

$$.8620689655172414$$

Therefore the differences between the approximate and the exact values at the required mesh points are

```
> Error := x -> Exact_Sol(x) - AB_Sol(x):
```

```
> for k from 0 to 6 do Error(0.1*k) od;
```

$$\begin{gathered} 0 \\ .10050941\,10^{-8} \\ .35444935\,10^{-8} \\ .63116676\,10^{-8} \\ .78879904\,10^{-8} \\ .74913027\,10^{-8} \\ .52147567\,10^{-8} \end{gathered}$$

□

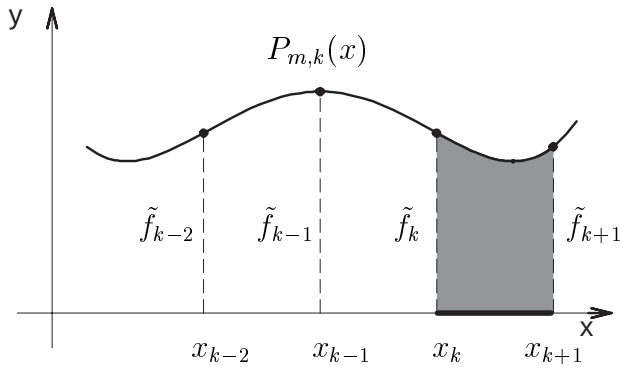
• **Adams–Moulton methods**

In 1926, Moulton modified the Adams–Bashforth methods as follows. To obtain a better approximation of the integral in (3.6), the value $f(x_{k+1}, y(x_{k+1}))$ at the new abscissa x_{k+1} is also used in addition to the values $f(x, y(x))$ at the abscissae

$$x_{k+1-m}, \cdots, x_{k-2}, x_{k-1}, x_k.$$

As $(m+1)$ support ordinates are available, we can construct an interpolation polynomial $P_{m,k}(x)$ of degree $\leq m$ (see Figure 3.2). Therefore we get

$$\begin{aligned} y(x_{k+1}) - y(x_k) &\approx \int_{x_k}^{x_{k+1}} P_{m,k}(t)dt = \\ &= \sum_{j=1}^{m+1} f(x_{k-m+j}, y(x_{k-m+j})) \int_{x_k}^{x_{k+1}} l_j(t)dt. \end{aligned} \tag{3.12}$$



$$\tilde{f}_{k-j} := f(x_{k-j}, y(x_{k-j}))$$

Figure 3.2 Approximation of the integral

Replacing in (3.12) all exact values $y(x_i)$ by approximate values y_i, $f(x_i, y(x_i))$ by $f_i := f(x_i, y_i)$ and $\approx$ by equality sign, we obtain the *m-step Adams–Moulton method*

$$\boxed{\begin{array}{c} y_0 := \alpha, y_1, y_2, \ldots, y_{m-1}, \\ y_{k+1} = y_k + h(\beta_{m1} f_{k+1-m} + \cdots + \beta_{m,m+1} f(x_k, y_{k+1}), \\ (k = m-1, m, m+1, \cdots), \end{array}} \tag{3.13}$$

where the *coefficients* β_{mj} $(j = 1, 2, \ldots, m+1)$ of the method are given by

$$h\beta_{mj} = \int_{x_k}^{x_{k+1}} l_j(t)dt$$

and they only depend on the mesh points.

In order to be able to use an m-step method (for $m > 1$), besides the initial condition $y_0 := \alpha$, $(m-1)$ more initial values $y_1, y_2, \cdots, y_{m-1}$ are necessary. They must be determined by other methods, e.g. with the aid of a single step method. The Runge–Kutta procedures are quite suitable and possibly a smaller step size $h_0 \leq h$ is adequate.

These methods have the feature that they are *implicit* in the sense that right side of (3.13) will depend on the unknown y_{k+1}. In order to use an implicit method we have to solve a nonlinear equation. We shall see later how we can do it.

It may be shown that the local truncation error of the method (3.13) satisfies

$$e(x_{k+1}, h) = O(h^{m+2}),$$

therefore *the order of the m-step Adams–Moulton method is $p = m+1$.*

It is interesting to compare a m-step Adams–Bashforth (explicit) method to the $(m-1)$-step Adams–Moulton (implicit) method. Both methods require m evaluation of f per step and both have the terms $O(h^m)$ in their local truncation error, as we have seen above. In general, the local truncation errors are smaller for the Adams–Moulton methods. This leads to greater stability for the implicit methods and smaller rounding errors. The price for this higher accuracy is that in general a nonlinear equation has to be solved at each step.

The next example illustrates how the coefficients of Adams–Bashforth methods can be determined.

EXAMPLE 3.5. *Compute the coefficients of the m-step Adams–Moulton method for $m = 0, 1, 2, 3, 4$.*

SOLUTION. If $m = 0$ then (3.6) yields

$$y(x_{k+1}) - y(x_k) = \int_{x_k}^{x_{k+1}} f(x, y(x))dx \approx hf(x_{k+1}, y(x_{k+1})).$$

Replacing $y(x_i)$ by its approximate value y_i and $\approx$ by $=$, we obtain

$$\boxed{y_{k+1} = y_k + hf(x_{k+1}, y_{k+1})}$$

which is the *implicit Euler's method.*

In the case of $m = 1$ we have two points $(x_k, \tilde{f}_k)$ and $(x_{k+1}, \tilde{f}_{k+1})$, where we used notation $\tilde{f}_i := f(x_i, y(x_i))$. It is easy to see that the

minimal degree interpolating polynomial through these points has the form

$$P_{1,k}(x) = \frac{x - x_{k+1}}{x_k - x_{k+1}} \tilde{f}_k + \frac{x - x_k}{x_{k+1} - x_k} \tilde{f}_{k+1}.$$

Integrating gives

$$y(x_{k+1}) - y(x_k) = \int_{x_k}^{x_{k+1}} f(x, y(x))dx \approx$$
$$\approx \int_{x_k}^{x_{k+1}} P_{1,k}(x)dx = \frac{h}{2}(\tilde{f}_k + \tilde{f}_{k+1}).$$

This leads to the following numerical method

$$\boxed{y_{k+1} = y_k + \frac{h}{2}\left(f(x_k, y_k) + f(x_{k+1}, y_{k+1})\right).} \tag{3.14}$$

This is the well-known *trapezoidal method* which is also an *implicit* procedure.

Now, suppose that m is an arbitrary nonnegative integer and consider again the general approximate formula

$$\int_{x_k}^{x_{k+1}} f(x, y(x))dx \approx \int_{x_k}^{x_{k+1}} P_{m,k}(x)dx.$$

The integral on the right-hand side can be computed using Maple.

```
> fpoints := proc(m)
              local i, vv:
                vv:=array(1..m+1):
                 for i to m+1 do vv[i]:=x-(m-i)*h od:
              eval(vv)
             end:
```

```
> fpoints(4);
```

$$[x - 3\,h,\ x - 2\,h,\ x - h,\ x,\ x + h]$$

```
> fvalues := proc(m)
              local i, vv:
                vv:=array(1..m+1):
                     for i to m+1 do vv[i]:=f(i-m) od:
              eval(vv)
            end:
```

```
> fvalues(4);
```

$$[\mathrm{f}(-3),\ \mathrm{f}(-2),\ \mathrm{f}(-1),\ \mathrm{f}(0),\ \mathrm{f}(1)]$$

```
> for i from 0 to 4 do simplify(
  int(interp(fpoints(i),fvalues(i),z),z=x..x+h)) od;
```

$$\mathrm{f}(1)\,h$$

$$\frac{1}{2}\,h\,(\mathrm{f}(1)+\mathrm{f}(0))$$

$$-\frac{1}{12}\,h\,(-5\,\mathrm{f}(1)-8\,\mathrm{f}(0)+\mathrm{f}(-1))$$

$$\frac{1}{24}\,h\,(\mathrm{f}(-2)+19\,\mathrm{f}(0)+9\,\mathrm{f}(1)-5\,\mathrm{f}(-1))$$

$$\frac{1}{720}\,h\,(251\,\mathrm{f}(1)-264\,\mathrm{f}(-1)+646\,\mathrm{f}(0)-19\,\mathrm{f}(-3)+106\,\mathrm{f}(-2))$$

We collect the coefficients in Table 3.2. □

m	β_{m1}	β_{m2}	β_{m3}	β_{m4}	β_{m5}
0	1				
1	$\frac{1}{2}$	$\frac{1}{2}$			
2	$-\frac{1}{12}$	$\frac{8}{12}$	$\frac{5}{12}$		
3	$\frac{1}{24}$	$-\frac{5}{24}$	$\frac{19}{24}$	$\frac{9}{24}$	
4	$-\frac{19}{720}$	$\frac{106}{720}$	$-\frac{264}{720}$	$\frac{646}{720}$	$\frac{251}{720}$

Table 3.2 Coefficients for Adams–Moulton methods

• Predictor-corrector methods

As we have mentioned above, in order to use an implicit method we have to solve a nonlinear equation of the form

$$x = F(x), \tag{3.15}$$

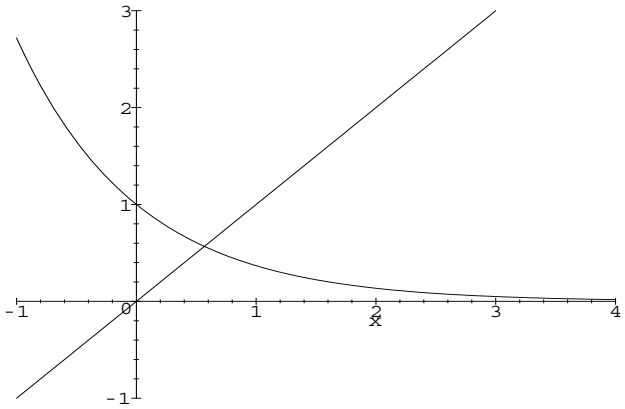
where F is a real valued function of one real variable. An approximate solution of (3.15) may be found by means of *fixed point iteration*, i.e. we take some initial value $x^{[0]}$, insert it into the right-hand side of (3.15) to get an updated value of the solution, insert this updated value back into the right-hand side, and continue iterating

$$x^{[i+1]} = F(x^{[i]}) \qquad (i = 0, 1, 2, \ldots).$$

EXAMPLE 3.6. *Use the fixed point iteration to find an approximate solution of the real root of the equation* $x = e^{-x}$. *Perform the computation using Maple.*

SOLUTION. It is easy to see that our equation has exactly one real root.

```
> plot({x, exp(-x)}, x=-1..4, -1..3);
```



Using the graph we choose the following initial value

```
> x0 := 0.5:
```

The sequence of iteration can be defined as follows

```
> ItSeq := proc(i)
            option remember:
            if i=0 then x0
             else exp(-ItSeq(i-1))
            fi
          end:
```

Now, we give the results for $i = 0, 10, 20, \ldots, 60$.

```
> for i from 0 to 6 do ItSeq(10*i) od;
```

.5
.5669072129
.5671424776
.5671432876
.5671432904
.5671432904
.5671432904

It can be shown that this method converges to the unique root. □

Let us consider the implicit trapezoidal method

$$y_{k+1} = y_k + \frac{h}{2}(f(x_k, y_k) + f(x_{k+1}, y_{k+1})) \tag{3.16}$$

for the solution of the initial value problem (3.1). Suppose that we know the approximate values $y_0, y_1, \ldots, y_k$. The next approximation y_{k+1} can be computed from the implicit equation (3.16). We normally do this by the fixed point iteration

$$\begin{aligned} &y_{k+1}^{[0]}, \\ &y_{k+1}^{[i+1]} = y_k + \frac{h}{2}(f(x_k, y_k) + f(x_{k+1}, y_{k+1}^{[i]})), \\ &\qquad (i = 0, 1, 2, \ldots). \end{aligned} \tag{3.17}$$

It can be shown that the sequence will converge to the unique solution of (3.16) provided that f satisfies the Lipschitz condition and h is small enough. Although (3.17) will converge for arbitrary $y_{k+1}^{[0]}$, each iteration calls for one evaluation of the function f, and computation can obviously be minimised if we can provide as good a guess as possible for $y_{k+1}^{[0]}$. This is conveniently done by using a separate explicit method to provide the initial guess, $y_{k+1}^{[0]}$. We call this explicit method the *predictor* and the implicit method (3.16) the *corrector*; the two together comprise a *predictor-corrector pair*.

Let the predictor be the two-step Adams–Bashforth method

$$y_{k+1} = y_k + \frac{h}{2}(3f(x_k, y_k) - f(x_{k-1}, y_{k-1})). \tag{3.18}$$

There are various ways, or modes, in which the pair (3.18), (3.16) can be implemented.

Firstly, we could use the predictor to give the first guess $y_{k+1}^{[0]}$, then allow the iteration (3.17) to proceed until we achieve convergence. This is called the mode of *correcting to convergence*. This method is rarely

used since the inherent accuracy of the implicit formula does not warrant more than a couple of iterations.

A much more acceptable procedure is to fix in advance just how many iterations of the corrector are to be permitted at each step. In practice, only a fixed number, say M, of iterations are carried out. Normally this number is small, usually 1 or 2. A useful mnemonic for describing modes of this sort can be constructed by using P and C to indicate one application of the predictor or the corrector respectively, and E to indicate one evaluation of the function f.

Suppose we apply the predictor to evaluate $y_{k+1}^{[0]}$, evaluate $f_{k+1}^{[0]} = f(x_{k+1}, y_{k+1}^{[0]})$, and then apply (3.16) just once to obtain $y_{k+1}^{[1]}$. The mode is then described as PEC. If we call the iteration a second time to obtain $y_{k+1}^{[2]}$, which obviously involves the further evaluation $f_{k+1}^{[1]} = f(x_{k+1}, y_{k+1}^{[1]})$, then the mode is described as $\mathrm{P(EC)^2}$. There is one further decision we have to make. At the end of $\mathrm{P(EC)^2}$ step we have a value $y_{k+1}^{[2]}$ for y_{k+1} and a value $f_{k+1}^{[1]} = f(x_{k+1}, y_{k+1}^{[1]})$. We may choose to update the value of f by making a further evaluation $f_{k+1}^{[2]} = f(x_{k+1}, y_{k+1}^{[2]})$; the mode would then be described as $\mathrm{P(EC)^2E}$. There is some evidence that final E is superior, so the strategy usually recommended is either PECE or $\mathrm{P(EC)^2E}$.

By varying the predictor and the corrector different predictor-corrector methods can be derived. Almost all modern predictor-corrector codes for the solutions of initial value problems use Adams–Bashforth methods as predictors and Adams–Moulton methods as correctors. Such methods are consequently sometimes called *Adams–Bashforth–Moulton methods.*

In our case the algorithms of the corresponding Adams–Bashforth–Moulton methods in PEC mode and in PECE mode can be formulated as follows

$$
\begin{aligned}
& x_0 := a, \qquad y_0 := \alpha, \\
& x_{k+1} := x_k + k, \qquad h := (b-a)/N, \\
& y_1 := y_0 + hf(x_0, y_0) \\
P: \quad & y_{k+1}^{[0]} := y_k^{[1]} + \frac{h}{2}(3f(x_k, y_k^{[0]}) - f(x_{k-1}, y_{k-1}^{[0]}), \\
E: \quad & f_{k+1}^{[0]} := f(x_{k+1}, y_{k+1}^{[0]}), \\
C: \quad & y_{k+1}^{[1]} := y_k^{[1]} + \frac{h}{2}(f(x_k, y_k^{[0]}) + f_{k+1}^{[0]}).
\end{aligned}
\tag{3.19}
$$

$$
\begin{array}{ll}
 & x_0 := a, \qquad y_0 := \alpha, \\
 & x_{k+1} := x_k + k, \qquad h := (b-a)/N, \\
 & y_1 := y_0 + hf(x_0, y_0), \\
P: & y_{k+1}^{[0]} := y_k^{[1]} + \frac{h}{2}(3f(x_k, y_k^{[1]}) - f(x_{k-1}, y_{k-1}^{[1]}), \\
E: & f_{k+1}^{[0]} := f(x_{k+1}, y_{k+1}^{[0]}), \\
C: & y_{k+1}^{[1]} := y_k^{[1]} + \frac{h}{2}(f(x_k, y_k^{[1]}) + f_{k+1}^{[0]}), \\
E: & f_{k+1}^{[1]} := f(x_{k+1}, y_{k+1}^{[1]}).
\end{array}
\tag{3.20}
$$

EXAMPLE 3.7. *Compare the PECE and the PEC methods above, used to find approximations to* $y(0.1k)$ $(k = 0, 1, \cdots, 6)$ *for the initial value problem*

$$y'(x) = -2xy^2(x), \qquad y(0) = 1 \tag{3.21}$$

using a step size $h = 0.01$.

SOLUTION. First we define the parameters and the exact solution of the initial value problem (3.21)

```
> a := 0:  y0 := 1:  h := 0.01:
> f := (x,y) -> -2*x*y(x)^2;
```

$$f := (x,\, y) \to -2\, x\, y(x)^2$$

```
> ExactSol := x ->1/(x^2+1);
```

$$ExactSol := x \to \frac{1}{x^2+1}$$

The PECE method (3.20) can be coded in Maple in the following way

```
> PECE :=
   proc(k)
    local yk0, fk0;
    option remember;
     if k=0 then y0;
      elif k=1 then y0+h*f(a,y0)
       else yk0 := PECE(k-1)+
        0.5*h*(3*f(a+(k-1)*h,PECE(k-1))-
        f(a+(k-2)*h,PECE(k-2))):
```

```
        fk0 := f(a+k*h,yk0):
        PECE(k-1)+0.5*h*(f(a+(k-1)*h,PECE(k-1))+fk0)
    fi
  end:
```

The results follow

```
> for i from 0 to 6 do PECE(10*i) od;
```

$$
\begin{array}{c}
1 \\
.9901980130 \\
.9616344576 \\
.9175221568 \\
.8621530233 \\
.8000756972 \\
.7353606287
\end{array}
$$

```
> for i from 0 to 6 do ExactSol(0.1*i)-PECE(10*i) od;
```

$$
\begin{array}{c}
0 \\
-.0000990031 \\
-.0000959961 \\
-.0000909641 \\
-.0000840578 \\
-.0000756972 \\
-.0000665111
\end{array}
$$

Now we give a code of the PEC algorithm (3.19)

```
> PEC :=
   proc(k)
    local yk0, y1;
    option remember;
     if k=0 then ff(0) := f(a,y0):  y0;
      elif k=1 then y1:=y0+h*ff(0):
           ff(1):=f(a+h,y1):  y1
        else yk0:=PEC(k-1)+0.5*h*(3*ff(k-1)-ff(k-2)):
           ff(k):=f(a+k*h,yk0);
         PEC(k-1)+0.5*h*(ff(k-1)+ff(k))
     fi
   end:
```

In this case the results are

```
> for i from 0 to 6 do ExactSol(0.1*i)-PEC(10*i) od;
```

$$
\begin{gathered}
0 \\
-.0000990087 \\
-.0000960431 \\
-.0000911008 \\
-.0000843117 \\
-.0000760594 \\
-.0000669443
\end{gathered}
$$

□

Maple also contains an Adams–Bashforth–Moulton method. Using Maple help

```
> ?dsolve[classical]
```

we can see that Maple can solve a problem by means of an Adams–Bashforth–Moulton method if we invoke the **dsolve** function with the option

`type = numeric` and `method = classical[abmoulton]`

In this function the step size and the number of corrections may be modified.

EXAMPLE 3.8. *Solve the initial value problem* (3.21) *by means of the built-in Adams–Bashforth–Moulton method of Maple. Take the step size* $h = 0.1$*, and obtain the difference between* y_k *and the exact value* $y(x_k)$ *at the mesh points* $x_k = kh$*,* $k = 0, 1, \ldots, 6$*.*

SOLUTION. Using 16-digits floating-point arithmetic we get

```
> Digits := 16:
> ABM_1 := dsolve({diff(y(x),x)=-2*x*(y(x))^2, y(0)=1},
        {y(x)}, type=numeric,
        method=classical[abmoulton], stepsize=0.1):
> ABM_1(0.4);
```

$$[x = .4,\ \mathrm{y}(x) = .8620271439244532]$$

```
> ABM_Sol := x -> rhs(op(2, ABM_1(x))):
  ABM_Sol(0.4);
```

$$.8620271439244532$$

```
> es := dsolve({diff(y(x),x)=-2*x*(y(x))^2, y(0)=1},
          y(x)):
> Exact_Sol := proc(t)
                 subs(x=t, rhs(es));
               end:
> Exact_Sol(x);
```

$$\frac{1}{x^2+1}$$

```
> Exact_Sol(0.4);
```

$$.8620689655172414$$

```
> Error := x -> Exact_Sol(x) - ABM_Sol(x):
> for k from 0 to 6 do Error(0.1*k) od;
```

$$\begin{gathered} 0 \\ .10050941\,10^{-8} \\ .35444935\,10^{-8} \\ .63116676\,10^{-8} \\ .78879904\,10^{-8} \\ .74913027\,10^{-8} \\ .52147567\,10^{-8} \end{gathered}$$

□

• Backward differentiation methods

The multistep methods introduced above, are all based on numerical integration. Now we consider multistep formulas based on numerical differentiation.

Assume that the approximations $y_k, y_{k+1}, ..., y_{k+m-1}$ to the exact solution of (3.1) at the equidistant points $x_k, x_{k+1}, \ldots, x_{k+m-1}$ are known.

To derive a formula for y_{k+m} let us consider the minimal degree polynomial $P_{m,k}(x)$ which interpolates the data

$$\{(x_i, y_i) : \ i = k, k+1, \ldots, k+m\}. \tag{3.22}$$

The unknown value y_{k+m} will be determined in such a way that the polynomial $P_{m,k}(x)$ satisfies the differential equation (3.1) at least one mesh point:

$$P'_{m,k}(x_{k+m-r}) = f(x_{k+m-r}, y_{k+m-r}). \tag{3.23}$$

For $r = 1$ we get *explicit* formulas, which are equivalent to *the explicit Euler method* and *the mid-point rule* if $m = 1$ and $m = 2$, respectively.

We obtain much more interesting formulas when (3.23) is taken at $r = 0$. Expanding the polynomial $P'_{m,k}$ in terms of y_{k+j} ($j = 0, 1, \ldots, m$) we get the *implicit* (*m-step backward differentiation*) formulas

$$\boxed{\begin{array}{c} y_0 := \alpha, y_1, y_2, \ldots, y_{m-1}, \\ \displaystyle\sum_{j=0}^{m} \alpha_{mj} y_{k+j} = h f(x_{k+m}, y_{k+m}), \\ (k = 0, 1, 2, \ldots, N-m), \end{array}} \tag{3.24}$$

where α_{mj} ($j = 0, 1, \ldots, m$) are constants, independent on h, k and f.

The coefficients α_{mj} can be determined using Maple. By slightly changing the notation of indices, first we define the data (3.22) as a function of m.

```
> fpoints := proc(m)
              local i, vv:
                vv:=array(1..m+1):
                  for i to m+1 do vv[i]:=x-(m-i)*h od:
              eval(vv)
             end:
```

```
> fpoints(4);
```

$$[x - 3\,h,\ x - 2\,h,\ x - h,\ x,\ x + h]$$

```
> yvalues := proc(m)
              local i, vv:
                vv:=array(1..m+1):
                     for i to m+1 do vv[i]:=y(i-m) od:
```

```
    eval(vv)
  end:
```

```
> yvalues(4);
```

$$[\mathrm{y}(-3),\ \mathrm{y}(-2),\ \mathrm{y}(-1),\ \mathrm{y}(0),\ \mathrm{y}(1)]$$

Now we define the interpolation polynomials $P_{m,k}(x)$ as a function of m.

```
> ip := m -> interp(fpoints(m), yvalues(m), z):
```

The left-hand side of the equation in (3.24) for $m = 1, 2, 3, 4, 5$ can be obtained in the following way:

```
> bdf := m -> h*simplify(subs(z=x+h, diff(ip(m), z))):
```

```
> for m to 5 do bdf(m) od;
```

$$\mathrm{y}(1) - \mathrm{y}(0)$$

$$\frac{3}{2}\,\mathrm{y}(1) - 2\,\mathrm{y}(0) + \frac{1}{2}\,\mathrm{y}(-1)$$

$$\frac{11}{6}\,\mathrm{y}(1) - \frac{1}{3}\,\mathrm{y}(-2) - 3\,\mathrm{y}(0) + \frac{3}{2}\,\mathrm{y}(-1)$$

$$3\,\mathrm{y}(-1) + \frac{25}{12}\,\mathrm{y}(1) + \frac{1}{4}\,\mathrm{y}(-3) - \frac{4}{3}\,\mathrm{y}(-2) - 4\,\mathrm{y}(0)$$

$$-5\,\mathrm{y}(0) - \frac{1}{5}\,\mathrm{y}(-4) + \frac{5}{4}\,\mathrm{y}(-3) + 5\,\mathrm{y}(-1) + \frac{137}{60}\,\mathrm{y}(1) - \frac{10}{3}\,\mathrm{y}(-2)$$

$$-\frac{6}{5}\,\mathrm{y}(-4) - 6\,\mathrm{y}(0) + \frac{15}{2}\,\mathrm{y}(-1) + \frac{15}{4}\,\mathrm{y}(-3) + \frac{1}{6}\,\mathrm{y}(-5) - \frac{20}{3}\,\mathrm{y}(-2) + \frac{49}{20}\,\mathrm{y}(1)$$

We collect the obtained coefficients in Table 3.3.

It may be shown that the order of the m-step backward differentiation method is m. The important feature of these methods is the size of their regions of absolute stability. These properties are significant in the context of stiffness.

m	α_{m0}	α_{m1}	α_{m2}	α_{m3}	α_{m4}	α_{m5}	α_{m6}
1	-1	1					
2	$\frac{1}{2}$	-2	$\frac{3}{2}$				
3	$-\frac{1}{3}$	$\frac{3}{2}$	-3	$\frac{11}{6}$			
4	$\frac{1}{4}$	$-\frac{4}{3}$	3	-4	$\frac{25}{12}$		
5	$-\frac{1}{5}$	$\frac{5}{4}$	$-\frac{10}{3}$	5	-5	$\frac{137}{60}$	
6	$\frac{1}{6}$	$-\frac{6}{5}$	$\frac{15}{4}$	$-\frac{20}{3}$	$\frac{15}{2}$	-6	$\frac{49}{20}$

Table 3.3 Coefficients for backward differentiation methods

3.4. General linear multistep methods

All of the numerical methods for the solution of the initial value problem (3.1) studied in the previous section can be written in the general form

$$\boxed{\begin{gathered} y_0 := \alpha, y_1, \ldots, y_{m-1}, \\ x_k := a + kh, \ h := (b-a)/N, \\ \sum_{j=0}^{m} \alpha_j y_{k+j} = h \sum_{j=0}^{m} \beta_j f(x_{k+j}, y_{k+j}), \\ (k = 0, 1, 2, \ldots, N-m), \end{gathered}} \tag{3.25}$$

where m is a given positive integer, α_j and β_j $(j = 0, 1, \ldots, m)$ are given constants, independent on h, k and the underlying differential equation. We also assume that $\alpha_m = 1$. Therefore if values $y_k, y_{k+1}, \ldots, y_{k+m-1}$ are known at the points $x_k, x_{k+1}, \ldots, x_{k+m-1}$ then the new approximate value y_{k+m} at the point x_{k+m} can be calculated from the equation

$$y_{k+m} + \sum_{j=0}^{m-1} \alpha_j y_{k+j} = h\beta_m f(x_{k+m}, y_{k+m}) + h \sum_{j=0}^{m-1} \beta_j f(x_{k+j}, y_{k+j}). \tag{3.26}$$

This class of methods is referred to as *m-step linear methods* because the numerical approximations y_i as well as the values $f(x_i, y_i)$ appear linearly in the formula. The starting values $y_1, y_2, \ldots, y_{m-1}$ may be determined by a single step method. When $\beta_m = 0$ (for example, as in the Adams–Bashforth methods) the methods are called *explicit*, otherwise *implicit* (e.g. Adams–Bashforth–Moulton methods). For an explicit method, the sequence (y_k) can be computed directly, provided the necessary additional starting values have been obtained, whereas for an implicit method it is necessary to solve a nonlinear equation at each step.

When we solve an initial value problem by means of (3.25) then it is necessary to analyze:

- what happens when the step size tends to zero (*convergence*),
- how well the difference equation in (3.25) approximates the differential equation (3.1) (*truncation errors*),
- how sensitive the difference equation is to perturbations in the data (*stability*).

• Convergence

When we use a numerical method for the solution of the initial value problem (3.1) the main question is: whether any desired degree of accuracy can be achieved by picking a small enough step size. This suggests the definition of convergence, which expresses the property that by using a sufficiently small step size and accurate computation the numerical solution can be made arbitrarily close to the true solution.

If we apply the multistep method (3.25) with step size h to the problem (3.1) we obtain a sequence (y_i). For given x and h such that $(x - x_0)/h = n$ is an integer, we introduce the following notations for the numerical solution:

$$y_h(x) = y_n \qquad \text{if} \qquad x - x_0 = nh. \tag{3.27}$$

A method is expected to be "good" in the sense that the numerical solution $y_h(x)$ converges to the exact solution $y(x)$ as $h \to 0$. Furthermore, we expect rapid convergence.

Definition 3.1. *The linear multistep method* (3.25) *is said to be* ***convergent*** *if, for all initial value problems* (3.1) *satisfying the hypotheses of Theorem 1.2, we have*

$$\lim_{h \to 0} y_h(x) = y(x), \qquad x \in [a, b] \tag{3.28}$$

whenever the starting values satisfy

$$\lim_{h\to 0} y_h(a+ih) = y(a) \qquad (i = 0, 1, ..., m-1).$$

A method which is not convergent is said to be ***divergent.***

DEFINITION 3.2. *Method* (3.25) *is* ***convergent of order p*** *if, to any problem* (3.1) *with sufficiently differentiable* f*, there exists a positive real number* h_0 *such that*

$$|y(x) - y_h(x)| \le Ch^p \qquad \text{for} \qquad h \le h_0 \tag{3.29}$$

whenever the starting values satisfy

$$|y(a+ih) - y_h(a+ih)| \le C_0 h^p \quad \text{for} \quad h \le h_0, \quad (i = 0, 1, ..., m-1).$$

We also say that the linear multistep method is of *order* p.

It turns out that the problem of convergence in the cases of multistep methods is more complicated than for single-step cases.

• Truncation errors

Suppose that the computations indicated in the method are performed exactly, i.e. round-off errors are not taken into account.

Let us denote by $y(x)$ the exact solution of (3.1) and by y_i the approximate value of $y(x_i)$ obtained by the method (3.25).

As we have seen in Section 2.3 the local truncation error is a good measure for accuracy in the case of single step methods. Reformulating Definition 2.1 for linear multistep methods we get

DEFINITION 3.3. *Let* $u(x)$ *be the exact solution of the initial value problem*

$$u'(x) = f(x, u(x)), \quad u(x_{k+m-1}) = y_{k+m-1}.$$

Then the ***local truncation error*** *of the multistep method* (3.25) *at* x_{k+m} *is defined by*

$$e_{k+m} := e(x_{k+m}, h) := u(x_{k+m}) - \tilde{y}_{k+m}, \tag{3.30}$$

where $\tilde{y}_{k+m}$ *is the numerical solution obtained from* (3.25) *using the exact starting values* $y_i = u(x_i)$ *for* $i = k, k+1, ..., k+m-1$.

Using (3.26) the local truncation error e_{k+m} can be written in the form

$$e_{k+m} = u(x_{k+m}) - \tilde{y}_{k+m} =$$

$$u(x_k + mh) + \sum_{j=0}^{m-1} \alpha_j u(x_k + jh) - h \sum_{j=0}^{m} \beta_j f(x_k + jh, u(x_k + jh)). \tag{3.31}$$

The difference between the exact and the numerical solution is the *accumulated* or *global error*.

DEFINITION 3.4. *The* ***global truncation error*** *of a linear multistep method* (3.25) *at* x_{k+m} *denoted by* E_{k+m} *is defined by*

$$E_{k+m} = y(x_{k+m}) - y_{k+m},$$

where $y(x)$ *is the exact solution of* (3.1) *and* y_{k+m} *is the approximate value of* $y(x_{k+m})$ *obtained by the method* (3.25).

The local truncation error and the starting errors accumulate to produce the global truncation error, but this accumulation process is very complicated, and we cannot hope to obtain any usable general expression for the global truncation error. In this respect the situation here is different from the single step methods. Figure 2.4 illustrates the relationships between the local and the global truncation errors.

• Consistency

We now turn to the question of what conditions a numerical method must satisfy if it is to be convergent.

We know that the local truncation error indicates how well the exact solution of the initial value problem (3.1) satisfies the recurrence formula (3.25). A first thought on the appropriate level of accuracy that might be needed for convergence is that we should ask that the local truncation error $e_{k+m} \to 0$ as $h \to 0$. Further thought shows that this is not going to be enough. The appropriate level of accuracy is to demand that $e_{k+m}/h \to 0$ as $h \to 0$.

DEFINITION 3.5. *A method of class* (3.25) *is said to be* ***consistent*** *if*

$$\max_{0 \le k \le N} \left| \frac{e_k}{h} \right| \to 0 \quad as \quad h \to 0.$$

It is ***consistent of order p*** *if*

$$\max_{0 \le k \le N} \left| \frac{e_k}{h} \right| = O(h^p).$$

It may be proved (see, e.g. [**SB**], Theorem 7.2.11.4) that consistency is necessary for convergence, i.e. *a linear multistep method* (3.25) *which is convergent is also consistent*. This result shows that it is important to determine the consistency of a linear multistep method.

It is appropriate at this stage to introduce the *first* and the *second characteristic polynomials* ϱ and σ associated with the method (3.25),

defined by

$$\varrho(\zeta) = \sum_{j=0}^{m} \alpha_j \zeta^j, \quad \sigma(\zeta) = \sum_{j=0}^{m} \beta_j \zeta^j \qquad (\zeta \in \mathbb{C}). \tag{3.32}$$

Dahlquist was the first to observe the fundamental role of these polynomials in the theory of multistep methods.

Fortunately for linear multistep methods a simple sufficient condition may be given to ensure that the method is consistent. To obtain such a condition one has to examine the behavior of the expression of the local truncation error (3.31). Using the fact that the function $u(x)$ satisfies the differential equation $u'(x) = f(x, u(x))$ we get

$$e_{k+m} = u(x_k + mh) + \sum_{j=0}^{m-1} \alpha_j u(x_k + jh) - h \sum_{j=0}^{m} \beta_j u'(x_k + jh).$$

Assume that $u(x)$ is sufficiently smooth (this is the case if f has sufficiently many continuous partial derivatives). Expanding $u(x + ih)$ and its derivative $u'(x + ih)$ in Taylor series about x_k and collecting terms in powers of h gives

$$e_{k+m} = c_0 u(x) + c_1 h u'(x) + \ldots + c_p h^p u^{(p)}(x) + \ldots, \tag{3.33}$$

where the constants c_i are independent of $u(x_k)$ and h. Using our initial assumption $\alpha_m = 1$ the following formulae for the constants c_i are easily established:

$$\begin{aligned} c_0 &= \sum_{j=0}^{m} = \varrho(1), \\ c_1 &= \sum_{j=0}^{m} (j\alpha_j - \beta_j) = \varrho'(1) - \sigma(1), \\ c_i &= \frac{1}{i!}\Big(\sum_{j=0}^{m} \alpha_j j^i - i \sum_{j=0}^{m} \beta_j j^{i-1}\Big), \quad (i = 2, 3, \ldots). \end{aligned} \tag{3.34}$$

According to Definition 3.5 the following simple sufficient condition for the consistency may be obtained: *If the characteristic polynomials ϱ and σ associated with the method* (3.25) *satisfy the condition*

$$\boxed{\varrho(1) = 0 \quad \textit{and} \quad \varrho'(1) - \sigma(1) = 0} \tag{3.35}$$

then method (3.25) *is consistent*. In general, it has order p if

$$c_0 = c_1 = \cdots = c_p = 0 \quad \text{and} \quad c_{p+1} \neq 0.$$

EXAMPLE 3.9. *Examine the consistency of the two-step Adams-Bashforth method* (3.10).

SOLUTION. The method (3.10) can be written in the form

$$y_{k+2} - y_{k+1} = h\big(-\frac{1}{2}f_k + \frac{3}{2}f_{k+1}\big),$$

therefore its characteristic polynomials are

$$\varrho(\zeta) = \zeta^2 - \zeta, \qquad \sigma(\zeta) = \frac{3}{2}\zeta - \frac{1}{2}.$$

Since $\varrho'(\zeta) = 2\zeta - 1$ thus

$$\varrho(1) = 0, \qquad \varrho'(1) = 1 = \sigma(1)$$

and hence the method is consistent. □

EXAMPLE 3.10. *Determine the order of consistency of the trapezoidal method* (3.14).

SOLUTION. The trapezoidal method has the following form

$$y_{k+1} - y_k = \frac{h}{2}(f_k + f_{k+1}).$$

The characteristic polynomials are

$$\varrho(\zeta) = \zeta - 1 \qquad \sigma(\zeta) = \frac{1}{2}\zeta + \frac{1}{2}.$$

Since $\varrho'(\zeta) = 1$, thus

$$\varrho(1) = 0, \qquad \varrho'(1) = 1 = \sigma(1),$$

and the method is consistent.

Let us denote by $u(x)$ the solution of the initial value problem

$$u'(x) = f(x, u(x)), \quad u(x_k) = y_k.$$

By (3.31) we get

$$e_{k+1} = u(x_k + h) - u(x_k) - \frac{h}{2}\left(u'(x_k) + u'(x_k + h)\right).$$

Through Taylor expansion in h one finds

$$e_{k+1} = \Big(u(x_k) + hu'(x_k) + \frac{h^2}{2}u''(x_k) + \frac{h^3}{6}u'''(x_k) + \cdots\Big) - u(x_k) - \\ -\frac{h}{2}\Big(u'(x_k) + u'(x_k) + hu''(x_k) + \frac{h^2}{2}u'''(x_k) + \cdots\Big),$$

thus

$$\frac{e_{k+1}}{h} = -\frac{h^2}{12}u'''(x_k) + O(h^3).$$

Hence the method is consistent of order two. □

Unfortunately the consistency is only a necessary but not a sufficient condition for the convergence, as the following example illustrates.

EXAMPLE 3.11. *The method*

$$y_{k+2} + y_{k+1} - 2y_k =$$
$$\frac{h}{4}\big[f(x_{k+2}, y_{k+2}) + 8f(x_{k+1}, y_{k+1}) + 3f(x_k, y_k)\big]$$

is consistent but it is divergent.

3.5. Stability of linear multistep methods

In this section we suppose that the conditions of the Theorem 1.2 are satisfied. From this it follows that the initial value problem (3.1) has a unique solution on the interval $[a, b]$. We also suppose that the problem (3.1) is *well-conditioned*. Roughly speaking this means that small perturbations in the stated problem will only lead to small changes in the solutions.

We consider several types of numerical stability.

• Zero-stability

As we have seen in the preceding section convergence implies consistency, the converse is not true. It can happen that the difference system produced by applying the numerical method (3.25) to the initial value problem (3.1) suffers an in-built instability which persist even in the limit as $h \to 0$ and prevents convergence. This leads to the concept of *zero-stability*, which controls the manner in which errors accumulate, but only in the *limit as* $h \to 0$.

DEFINITION 3.6. *We say that a linear multistep method of class* (3.25) *is* ***zero stable*** *if, for sufficiently small stepsizes h, small perturbations in the starting values produce small perturbations in subsequent values.*

We recall that zero-stability is nearly automatic for single step methods (see Theorem 2.2). The following example shows that the situation changes in the case of multistep methods.

Consider the two step method

$$y_0 = s_0(h), \qquad y_1 = s_1(h),$$
$$y_{k+2} - 3y_{k+1} + 2y_k = h(f_{k+1} - 2f_k),$$

applied to the problem

$$y'(x) = 2x, \qquad y(0) = 0,$$

whose exact solution is $y(x) = x^2$. Now

$$\varrho(\zeta) = \zeta^2 - 3\zeta + 2, \qquad \sigma(\zeta) = \zeta - 2, \qquad \varrho'(\zeta) = 2\zeta - 3,$$

thus

$$\varrho(1) = 0, \qquad \varrho'(1) = \sigma(1) = -1$$

and hence the method is consistent provided we choose starting values such that $s_0(h) \to 0$, $s_1(h) \to 0$. Now

$$\begin{aligned} y_{k+2} - 3y_{k+1} + 2y_k &= 2h(x_{k+1} - 2x_k) = \\ &= 2h^2(1-k). \end{aligned}$$

The roots of $\varrho(\zeta)$ are $\zeta = 1$ and $\zeta = 2$. Hence the general solution of the difference equation is

$$y_k = A + B2^k + k(k-1)h^2.$$

The particular solution satisfying the initial conditions is

$$y_k = [2s_0(h) - s_1(h)] + [s_1(h) - s_0(h)]2^k + k(k-1)h^2.$$

Now for the starting values

$$s_0(h) = 0, \qquad s_1(h) = 0,$$

$$y_k = k(k-1)h^2 = x^2 - \frac{x^2}{k} \to x^2 \qquad \text{as} \qquad h \to 0 \qquad (k \to \infty).$$

However for the exact starting values

$$y_k = -h^2 + h^2 2^k + k(k-1)h^2 = -\frac{x^2}{k^2} + \frac{x^2}{k^2}2^k + x^2 - \frac{x^2}{k}.$$

But

$$\frac{2^k}{k^2} \to \infty \qquad \text{as} \qquad k \to \infty \ \ (h \to 0),$$

hence the method is not convergent for these starting values. Thus we see that the method is sensitive to small perturbations in the starting values i.e. it is *unstable* in the sense that the resulting perturbations are unbounded, even in the limit as $h \to 0$.

It is clear that the reason for the instability is the root $+2$ of the characteristic polynomial $\varrho(\zeta)$.

The definition of the zero-stability given above is a desirable one rather then a convenient one. We want a practical technique for testing for stability.

To obtain such a condition consider the equation

$$y'(x) = 0, \qquad y(0) = 0, \qquad (\text{soln. } y(x) = 0).$$

Then (3.25) reduces to

$$\sum_{j=0}^{m} \alpha_j y_{k+j} = 0. \tag{3.36}$$

If

$$\varrho(\zeta) = \sum_{j=0}^{m} \alpha_j \zeta^j$$

has distinct roots $\zeta_1, \zeta_2, ..., \zeta_m$, then the solution of the difference equation (3.36) is

$$y_k = A_1(\zeta_1)^k + A_2(\zeta_2)^k + \cdots + A_m(\zeta_m)^k.$$

If the method is consistent then, since $\sum_{j=0}^{m} \alpha_j = 0$, one root, say $\zeta_1 = 1$. This root corresponds to the exact solution of the differential equation. Hence for stability we require

$$|\zeta_i| \leq 1, \qquad 2 \leq i \leq m.$$

If $\varrho(\zeta)$ has a root ζ_j, of multiplicity r, then

$$(\zeta_j)^k,\ k(\zeta_j)^k,\ k^2(\zeta_j)^k, \ldots, k^{r-1}(\zeta_j)^k$$

are solutions of (3.36). Hence for a multiple root, with multiplicity $r > 1$, stability requires that

$$|\zeta_j| < 1.$$

DEFINITION 3.7. *We say that a linear multistep method* (3.25) *satisfies the* ***root condition*** *if the roots of the first characteristic polynomial* $\varrho(\zeta)$ *all lie within or on the unit circle in the complex plane, and are simple if they lie on the unit circle.*

We have shown that a necessary condition for zero-stability is that the method satisfies the root condition. We can strengthen this to:

THEOREM 3.1. *The linear multistep method* (3.25) *is zero-stable if and only if it satisfies the root condition.*

We can now state the fundamental theorem concerning convergence:

THEOREM 3.2. *A method of class* (3.25) *is convergent if and only if it is both consistent and zero stable.*

A more accurate result can be stated (see, e.g. [**H1**], Chapter III, Theorem 4.5).

THEOREM 3.3. *If the multistep method* (3.25) *is zero-stable and the order of its consistency is p then it is convergent of order p.*

• Absolute stability

The concept of zero-stability, and also convergence are concerned with the limiting process as $h \to 0$. In practice, we must compute with a finite number of steps, i.e. with finite, nonzero step size h. In particular we want to know if the errors we introduced at each step (truncation and round-off) have a small or large effect on the answer. What is needed is a stability theory which applies when h takes a fixed non-zero value.

We have already considered the notion of absolute stability for single step methods (see Section 2.7). This concept can be generalized to linear multistep methods.

To illustrate the problem of absolute stability let us consider the *mid-point method*

$$y_{k+2} = y_k + 2hf_{k+1}$$

applied to the *test equation*

$$y'(x) = \lambda y(x). \tag{3.37}$$

Substituting into the differential equation gives

$$y_{k+2} = y_k + 2h\lambda y_{k+1}.$$

Thus

$$y_{k+2} - 2\lambda y_{k+1} - y_k = 0.$$

Let $y_k = \zeta^k$. Thus

$$\zeta^k(\zeta^2 - 2h\lambda\zeta - 1) = 0.$$

Solving

$$\zeta_{1,2} = \frac{2h\lambda \pm \sqrt{4h^2\lambda^2 + 4}}{2} = h\lambda \pm \sqrt{h^2\lambda^2 + 1}.$$

Thus there are two roots

$$\begin{aligned}\zeta_1 &= h\lambda + \sqrt{h^2\lambda^2+1}\\ &= h\lambda + \big(1 + \frac{1}{2}h^2\lambda^2 - \frac{1}{4}h^4\lambda^4 + \cdots\big)\\ &= 1 + h\lambda + \frac{1}{2}h^2\lambda^2 - \frac{1}{4}h^4\lambda^4 + \cdots\\ &= e^{h\lambda} + O(h^3)\end{aligned}$$

and

$$\begin{aligned}\zeta_2 &= h\lambda - \sqrt{h^2\lambda^2+1}\\ &= h\lambda - \big(1 + \frac{1}{2}h^2\lambda^2 - \frac{1}{4}h^4\lambda^4 + \cdots\big)\\ &= -1 + h\lambda - \frac{1}{2}h^2\lambda^2 + \frac{1}{4}h^4\lambda^4 + \cdots\\ &= -e^{-h\lambda} + O(h^3).\end{aligned}$$

Therefore the solution of the difference equation is

$$\begin{aligned}y_k &= A\zeta_1^k + B\zeta_2^k = Ae^{kh\lambda} + B(-1)^k e^{-kh\lambda}\\ &= Ae\lambda x_k + B(-1)^k e^{-\lambda x_k}.\end{aligned}$$

Thus if $\lambda < 0$ the difference equation has a spurious solution corresponding to ζ_2 which will eventually dominate the desired solution leading to instability. This is illustrated in the following example.

EXAMPLE 3.12. *Solve the initial value problem*

$$y'(x) = -y(x), \quad y(0) = 1 \tag{3.38}$$

using the mid-point method with $h = 0.25$. Let the starting value y_1 be an Euler's step. Compare the computed one with the exact solution.

SOLUTION. First we find the exact solution of (3.38)

```
> In_Val_Pr := {diff(y(x), x) = -y(x), y(0) = 1};
```

$$In\_Val\_Pr := \{\frac{\partial}{\partial x}\,\mathrm{y}(x) = -\mathrm{y}(x),\ \mathrm{y}(0) = 1\}$$

```
> es := dsolve(In_Val_Pr, y(x)):
> Exact_Sol := proc(t)
                 subs(x=t, rhs(es));
               end:
```

```
> Exact_Sol(x);
```

$$\frac{1}{e^x}$$

Now we define the parameters of the initial value problem

```
> f := (x, y) -> -y;
```

$$f := (x,\, y) \to -y$$

```
> a := 0:  y0 := 1:  h := 0.25:
  y1 := y0 + h*f(a,y0);
```

$$y1 := .75$$

A Maple program of the mid-point method is

```
> mpm := proc(k)
          option remember;
           if k=0 then y0
             elif k=1 then y1
          else mpm(k-2)+2*h*f(a+(k-1)*h,mpm(k-1))
          fi
        end:
> x := k -> k*h:  # for the mesh points
> mpm(1);
```

$$.75$$

We define the error function in the following way

```
> Error := k -> Exact_Sol(x(k)) - mpm(k):
```

Finally we create a table with appropriate headings.

```
> mm:=array(1..10, 1..4):
          mm[1,1]:=‘Mesh points‘:mm[1,2]:=‘Exact sol.‘:
          mm[1,3]:=‘Approx.sol.‘:mm[1,4]:=‘Error‘:
        for i from 2 to 10 do
          mm[i,1]:=x(i-1):
          mm[i,2]:=evalf(Exact_Sol(x(i-1)), 5):
          mm[i,3]:=evalf(mpm(i-1), 5):
          mm[i,4]:=evalf(Error(i-1), 5):
```

```
      od:
eval(mm);
```

Mesh points	*Exact sol.*	*Approx. sol.*	*Error*
.25	.77882	.75	.02882
.50	.60654	.6250	−.01846
.75	.47237	.43750	.03487
1.00	.36788	.40624	−.03836
1.25	.28651	.23438	.05213
1.50	.22313	.28905	−.06592
1.75	.17377	.08985	.08392
2.00	.13533	.24412	−.10879
2.25	.10540	−.03221	.13761

□

If $\lambda > 0$ then the spurious solution decays as x increases and hence the instability does not manifest itself.

Now consider the linear multistep method

$$\sum_{j=0}^{m} \alpha_j y_{k+j} = h \sum_{j=0}^{m} \beta_j f_{k+j} \tag{3.39}$$

applied to the test equation

$$y'(x) = \lambda y(x).$$

On substitution into the differential equation (3.39) we obtain

$$\sum_{j=0}^{m} \alpha_j y_{k+j} = h\lambda \sum_{j=0}^{m} \beta_j y_{k+j}.$$

Thus

$$\sum_{j=0}^{m} \alpha_j y_{k+j} - h\lambda \sum_{j=0}^{m} \beta_j y_{k+j} = 0.$$

Let $y_k = \zeta^k$, then

$$\sum_{j=0}^{m} \alpha_j \zeta^j - h\lambda \sum_{j=0}^{m} \beta_j \zeta^j = 0.$$

Hence

$$\pi(\zeta; h\lambda) := \varrho(\zeta) - h\lambda\sigma(\zeta) = 0. \tag{3.40}$$

$\pi(\zeta; h\lambda)$ is called the *stability polynomial* of the linear multistep method (3.39). Now one of the roots $\zeta_1(h\lambda)$ of $\pi(\zeta; h\lambda)$ will correspond to the true solution, the other roots will lead to spurious solutions whose magnitude will have to be controlled to obtain stability. For a pth order method we can show that

$$\zeta_1(h\lambda) = e^{h\lambda} + O(h^{p+1}). \tag{3.41}$$

DEFINITION 3.8. *A linear multistep method is said to be* ***absolutely stable*** *for a given $h\lambda$ if all the roots of $\pi(\zeta; h\lambda)$ lie within the unit circle. A region $\mathcal{R}_\lambda$ of the complex plane is said to be a* ***region of absolute stability*** *if the linear multistep method is absolutely stable for all $h\lambda$ in $\mathcal{R}_\lambda$.*

The most convenient method for finding regions of absolute stability is the *boundary locus technique*. The region $\mathcal{R}_\lambda$ of the complex $(h\lambda)$-plane is defined by the requirement that for all $h\lambda \in \mathcal{R}_\lambda$ all of the roots of $\pi(\zeta, h\lambda)$ have modulus less than 1. Let the contour $\partial\mathcal{R}_\lambda$ in the complex $(h\lambda)$-plane be defined by the requirement that for all $h\lambda \in \mathcal{R}_\lambda$ *one* of the roots of $\pi(\zeta, h\lambda)$ has modulus 1, that is, is of the form $\zeta = e^{i\theta}$. Since the roots of a polynomial are continuous function of its coefficients it follows that the boundary of $\mathcal{R}_\lambda$ must consist of $\partial\mathcal{R}_\lambda$ (or of part of $\partial\mathcal{R}_\lambda$; some parts of $\partial\mathcal{R}_\lambda$ could, for example, correspond to $\pi(\zeta, h\lambda)$ having one root of modulus 1, some of the remaining roots having modulus less than 1 and some having modulus greater than 1). Thus, for all $h\lambda \in \partial\mathcal{R}_\lambda$, the identity

$$\pi(e^{i\theta}, h\lambda) = \varrho(e^{i\theta}) - h\lambda\sigma(e^{i\theta}) = 0$$

must hold. This equation is readily solved for $h\lambda$, and we have that the locus of $\partial\mathcal{R}_\lambda$ is given by

$$(h\lambda) = (h\lambda)(\theta) = \frac{\varrho(e^{i\theta})}{\sigma(e^{i\theta})}. \tag{3.42}$$

In most cases we simply use (3.42) to plot $(h\lambda)(\theta)$ for a range of $\theta \in [0, 2\pi]$.

EXAMPLE 3.13. *Find and sketch the region of absolute stability of m-step Adams–Moulton method for $m = 1, 2, 3, 4$.*

SOLUTION. If $m = 1$ then we have the trapezoidal rule

$$y_{k+1} - y_k = \frac{h}{2}(f_k + f_{k+1}).$$

From Example 2.16 of Section 2.7 we know that the region of absolute stability of this method is the whole left half-plane.

We investigate the cases $m = 2, 3, 4$ with Maple. The coefficients of these methods are in the Table 3.2. Therefore the corresponding characteristic polynomials can be defined in the following way

```
> rho2 := x -> x^2 - x;
```

$$\rho 2 := x \to x^2 - x$$

```
> sigma2 := x-> (-1/12)+(8/12)*x+(5/12)*x^2;
```

$$\sigma 2 := x \to -\frac{1}{12} + \frac{2}{3}\,x + \frac{5}{12}\,x^2$$

```
> rho3 := x -> x^3 - x^2;
```

$$\rho 3 := x \to x^3 - x^2$$

```
> sigma3 := x ->
          (1/24)-(5/24)*x+(19/24)*x^2+(9/24)*x^3;
```

$$\sigma 3 := x \to \frac{1}{24} - \frac{5}{24}\,x + \frac{19}{24}\,x^2 + \frac{3}{8}\,x^3$$

```
> rho4 := x -> x^4 - x^3;
```

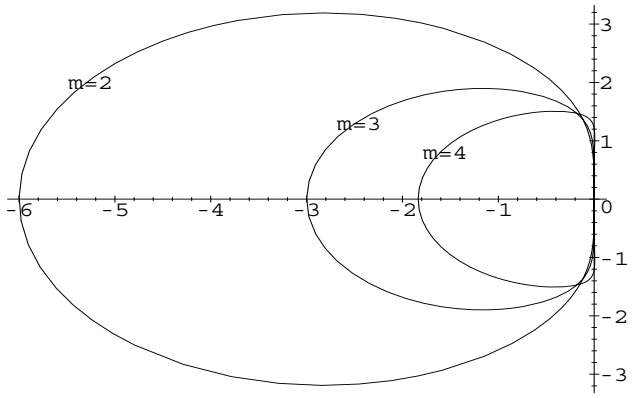
$$\rho 4 := x \to x^4 - x^3$$

```
> sigma4 := x -> -(19/720)+(106/720)*x-
        (264/720)*x^2+(646/720)*x^3+ (251/720)*x^4;
```

$$\sigma 4 := x \to -\frac{19}{720} + \frac{53}{360}\,x - \frac{11}{30}\,x^2 + \frac{323}{360}\,x^3 + \frac{251}{720}\,x^4$$

The locus of the boundary $\partial\mathcal{R}_\lambda$ can be drawn using the **complexplot** procedure which is in the **plots** package.

```
> with(plots):
> t := I*theta:
> am2 := complexplot(rho2(exp(t))/sigma2(exp(t)),
          theta=0..2*Pi):
> am3 := complexplot(rho3(exp(t))/sigma3(exp(t)),
          theta=0..2*Pi):
> am4 := complexplot(rho4(exp(t))/sigma4(exp(t)),
          theta=0..2*Pi):
> pr := textplot({[-5,2, `m=2`],
          [-2.2, 1.3, `m=3`],
          [-1.3, 0.8, `m=4`]}, align=LEFT):
> display({am2, am3, am4, pr});
```



For the Adams–Moulton methods with $m = 2, 3, 4$, $\partial\mathcal{R}_\lambda$ is a simple closed contour. To see that the *interiors* of the regions bounded by $\partial\mathcal{R}_\lambda$ are indeed the regions of absolute stability, all we need do is observe that, from (3.41), all linear multistep methods are necessarily absolute unstable for small positive values of $Re\,(h\lambda)$. □

• Strong stability

Now from (3.41) if $h\lambda$ is real, positive and small then $|\zeta_1(h\lambda)| > 1$ and such $h\lambda$ are outside $\mathcal{R}_\lambda$. If $h\lambda$ is real, negative and small then $|\zeta_1(h\lambda)| < 1$. However it is possible that $\pi(\zeta; h\lambda)$ possesses another root $\zeta_j(h\lambda)$ such that $|\zeta_j(0)| = 1$ and $|\zeta_j(h\lambda)| > 1$ for $h\lambda$ real, negative and small. Such a method will have no interval of absolute stability, at least in the neighborhood of the origin. An example of such a method is *Simpson's method*

$$y_{k+2} = y_k + \frac{h}{3}(f_{k+2} + 4f_{k+1} + f_k),$$

for which

$$\pi(\zeta; h\lambda) = (1 - \frac{h\lambda}{3})\zeta^2 - \frac{4h\lambda}{3}\zeta - (1 + \frac{h\lambda}{3}),$$

with roots

$$\zeta_1(h\lambda) = 1 + h\lambda + O(h^2),$$
$$\zeta_2(h\lambda) = -1 + \frac{1}{3}h\lambda + O(h^2),$$

so that if $Re(h\lambda)$ is small and positive $|\zeta_1(h\lambda)| > 1$ whereas if $Re(h\lambda)$ is small and negative $|\zeta_2(h\lambda)| > 1$. Thus the method has no region of absolute stability in the neighborhood of the origin. More precisely we can show that $\mathcal{R}_\lambda$ is empty.

If we wish to avoid the possibility of an empty $\mathcal{R}_\lambda$ we choose a method satisfying the *strong root condition.*

DEFINITION 3.9. *A method is said to satisfy the* ***strong root condition*** *if the characteristic polynomial has a simple root at $\varrho = 1$ and all the remaining roots lie strictly within the unit circle.*

DEFINITION 3.10. *A method of class* (3.25) *is said to be* ***strongly stable*** *if it is consistent and satisfies the strong root condition.*

A class of methods which are strongly stable are the Adams methods previously encountered for which the characteristic polynomial $\varrho(\zeta)$ has the form

$$\varrho(\zeta) = \zeta^k - \zeta^{k-1}.$$

3.6. Advanced methods

The unpredictable behavior of solutions of differential equations forces the numerical integration to proceed with step sizes which, in general, must vary from point to point if a prescribed error bound is to

be maintained. Multistep methods which use equidistant mesh points and a fixed order, therefore, are not very suitable in practice.

Predictor-corrector methods possess many advantages, notably the facility for monitoring the local truncation error cheaply and efficiently. However, there is a balancing disadvantage, shared by all multistep methods, namely the difficulties encountered in implementing a change of step size.

A program embodying a multistep method will have to use techniques for starting, changing step, and changing order as necessary. The choice of which class of methods to use depends on the problem. Often little is known about the problem to be integrated, so the Adams' methods whose extraneous eigenvalues are zero is usually the best choice. (Other methods for special problems will be discussed in Chapter 4.)

Suppose that we have used an mth order Adams–Bashforth–Moulton method to compute y_k, but before going on to compute y_{k+1} we want to change the step size from h to αh. In order to apply the method to compute an approximation to the exact solution at $x_k + \alpha h$, we need back data at x_k, which we have, and at $x_k - \alpha h$,$x_k - 2\alpha h$, ..., $x_k - (m-1)\alpha h$, which we do not have. Many different ways of tackling this problem have been proposed. The available techniques can be categorized into two different groups. The first, known as *interpolatory techniques*, use polynomial interpolation of the existing back data in order to approximate the missing back data; there are several ways of doing this. In the second group, the Adams–Bashforth–Moulton methods themselves are replaced by Adams–Bashforth–Moulton-like methods which assume that the data is unevenly spaced, and whose coefficients therefore vary as the step size varies. Stepchanging techniques based on such methods are usually known as *variable step techniques*.

3.7. Exercises

1. Write Maple programs to solve the system of ordinary differential equations

$$\mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y}(x)), \qquad \mathbf{y}(x_0) = \mathbf{y}_0$$

by the two-step Adams–Bashforth method.

2. Use the Euler's method as a predictor and trapezoidal method as a corrector in a) PECE and b) PEC modes and c) correcting to convergence to solve the initial value problem

$$y'(x) = xe^{-y(x)}, \quad y(1) = 0, \quad 1 \le x \le 2.$$

3. Examine the consistency of the methods

   a) $y_{k+1} = 4y_k - 3y_{k-1} - 2hf_{k-1}$,
   b) $y_{k+1} = -\frac{3}{2}y_k + 3y_{k-1} - \frac{1}{2}y_{k-2} + 3hf_k$.

4. Examine the zero stability of the methods of Q3.

5. Show that Simpson's method satisfies the root condition but not the strong root condition.

6. Show that the solution of the difference equation for Simpson's method applied to the test equation

$$y'(x) = \lambda y(x)$$

   can be written as

$$y_k = Ae^{\lambda x_k} + Be^{-\frac{1}{2}\lambda x_k}.$$

   What implications does this have for the stability of the method?

7. Use the boundary locus method to obtain regions of absolute stability for the following methods

   a) $y_{k+1} = y_k + hf_{k+1}$ backward Euler's method,
   b) $y_{k+1} = y_{k-1} + 2hf_k$ mid-point method.

8. Show that the method

$$y_{k+2} = y_k + \frac{h}{2}(f_{k+1} + 3f_{k+2})$$

   is not strongly stable.
   Show also that when the method is applied to the test equation $y'(x) = \lambda y(x)$ the roots of the difference equation are

$$\zeta_1 = 1 + h\lambda + O(h^2),$$
$$\zeta_2 = -1 - \frac{1}{2}h\lambda + O(h^2).$$

   Is $\mathcal{R}_\lambda$ empty?

Bibliography

[AB] Abel, M.L. and Braselton, J.P., *Differential Equations with Maple V.* AP Professional, Boston, 1994.

[BeCo] Bellmann, R. and Cooke, K.L., *Differential-Difference Equations*, Academic Press, 1963.

[BG] Birkhoff, G. and Gian-Carlo, Rota, *Ordinary Differential Equations.* John Wiley and Sons, Inc., New York, 1989.

[BD] Boyce, W.E. and DiPrima R.C., *Elementary Differential Equations and Boundary Value Problems.* John Wiley and Sons, Inc., New York, (5th ed.), 1992.

[Bu] Butcher, J.C. *The Numerical Analysis of Ordinary Differential Equations.* John Wiley and Sons, Inc., New York, 1987.

[CL] Coddington, E.A. and Levinson, N., *Theory of Ordinary Differential Equations.* McGraw-Hill Book Company, Inc., New York, 1955.

[Co] Collatz, L., *The Numerical Treatment of Differential Equations.* Springer-Verlag, Berlin-Heidelberg-New York, 1966.

[CB] Conte, S.D. and de Boor, C., *Elementary Numerical Analysis: An Algorithmic Aproach.* McGraw-Hill Kögakusha, Tokyo, (3rd ed.). 1980.

[DB] Dahlquist, G. and Björk, Å., *Numerical Methods.* Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1994.

[Dri] Driver, R.D., *Ordinary and delay differential equations*, Applied Math. Sciences 20, Springer Verlag, 1977.

[EW] Eldén, L. and Wittmeyer-Koch, L., *Numerical Analysis, An Introduction.* Academic Press, Inc., Boston, 1990.

[Ge1] Gear, C.W., *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1971.

[Ge2] Gear, C.W., *The automatic integration of ordinary differential equations.* Comm. ACM **14** (Mar. 1971), pp. 176-179.

[Ge3] Gear, C.W., *The automatic integration of stiff ordinary differential equations*, Information Processing 68, A.J.H. Morrell, Ed., North Holland, Amsterdam, 1969, pp. 187-193.

[Ge4] Gear, C.W., *The numerical integration of ordinary differential equations*, Math. Comp. **21,** 2 (Apr. 1967), pp. 146-156.

[H1] Hairer, E., Nørsett, S.P. and Wanner, G., *Solving Ordinary Differential Equations I. Nonstiff Problems.* Springer Verlag, Berlin, (2nd ed.), 1991.

[H2] Hairer, E., Wanner, G., *Solving Ordinary Differential Equations II. Stiff Problems and Differential-algebraic Equations.* Springer Verlag, Berlin, 1991.

[HW] Hall, G. and Watt, J.M. (Eds.), *Modern Numerical Methods for Ordinary Differential Equations.* Clarendon Press, Oxford. 1976.

[Ha] Hamming, R.W., *Numerical Methods for Scientists and Engineers.* McGraw-Hill Book Company, Inc., New York, (2nd ed.) 1973.

[Hr] Hartman, Ph., *Ordinary Differential Equations.* John Wiley and Sons, Inc., New York, 1964.

[HH] Hämmerlin, G. and Hoffmann, K-H., *Numerical Mathematics.* Springer-Verlag, New York Inc., 1991.

[Hc] Heck, A., *Introduction to Maple.* Springer-Verlag, New York, 1993.

[He] Henrici, P., *Discrete Variable Methods in Ordinary Differential Equations.* John Wiley and Sons, Inc., New York, 1962.

[HJ] Higham, N. J., *Accuracy and Stability of Numrical Algorithms*, SIAM, Philadelphia, 1996.

[Hi] Hildebrand, F.B., *Introduction to Numerical Analysis.* McGraw-Hill Book Company, New York, 1974.

[Hi1] Hindmarsh, A.C., *LSODE and LSODI, two new initial value ordinary differential equation solvers*, ACM-SIGNUM Newsletter 15, 1980, pp. 10-11.

[Hi2] Hindmarsh, A.C., *GEAR: ordinary differential equation system solver*, UCID-30001, Rev. 2, LLL, Livermore, Calif. 1972.

[Hi3] Hindmarsh, A.C., *ODEPACK, a Systemized Collection of ODE Solvers*, In: Scientific Computing, R.S. Stepleman et al. (eds.) North-Holland, Amsterdam, 1983.

[IK] Isaacson, E. and Keller, H.B., *Analysis of Numerical Methods.* John Wiley and Sons, Inc., New York, 1966.

[Is] Iserles, A., *A First Course in the Numerical Analysis of Differential Equations.* Cambridge Text in Applied Mathematics. Cambridge Univ. Press., 1996.

[Ka] Kamke, E., *Differentialgleichungen, Lösungsmethoden und Lösungen, Vol. 1.* Leipzig, 1959.

[KM] Kopchenova, N.V. and Maron, I.A., *Computational Mathematics, Worked Examples and Problems with Elements of Theory.* Mir Publishers, Moscow, 1975.

[La] Lambert, J.D., *Numerical Methods for Ordinary Differential Systems.* John Wiley and Sons, Ltd., Chichester, 1991.

[Lo] Lorentz, H.W., *Nonlinear Dynamical Economics and Chaotic Motion.* Springer Verlag, Berlin-Heidelberg, 2nd Ed., 1993.

[No] Nordsieck, A., *On numerical integration of ordinary differential equations*, Math. Comp. **16,** 1 (Jan. 1962), pp. 22-49.

[PT] Press, W.H., Teukolsky, S.A., Vatterling, W.T. and Flannery, B.P., *Numerical Recipes in C. The Art of Scientific Computing.* Second Ed., Cambridge Univ. Press, 1992.

[RR] Ralston, A. and Rabinowitz, P., *A First Course in Numerical Analysis.* McGraw-Hill Book Company, New York, 1978.

[Sc] Schwarz, H.R., *Numerical Analysis, A Comprehensive Introduction.* John Wiley and Sons, Ltd., Chichester, 1989.

[SG] Shampine, L.F. and Gordon, M.K., *Computer Solution of Ordinary Differential Equations.* W.H. Freeman, San Francisco, 1975.

[St] Stetter, H.J., *Analysis of Discretization Methods for Ordinary Differential Equations.* Springer Tracts in Natural Philosophy. Vol. 23, Springer Verlag, Berlin, 1973.

[SB] Stoer, J. and Bulirsch, R., *Introduction to Numerical Analysis.* Springer Verlag, Berlin, 1980

[YS] Yakowitz, S. and Szidarovszky, F., *An Introduction to Numerical Computations.* Macmillan Publ. Comp., New York, 1986.

CHAPTER 4

Stiff and Delay Systems of Differential Equations

4.1. Stiffness and stability

EXAMPLE 4.1. *Let us solve the following system of differential equations [Is].*

$$x'(t) = -100x(t) + y(t),$$
$$y'(t) = -0.1y(t)$$

with the initial conditions

$$x(0) = 1, \qquad y(0) = 1.$$

We determine the exact solution with Maple:

```
> sys := diff(x(t), t) = -100*x(t) + y(t),
         diff(y(t), t) = -(1/10)*y(t);
```

$$sys := \frac{\partial}{\partial t}\,\mathrm{x}(t) = -100\,\mathrm{x}(t) + \mathrm{y}(t),\ \frac{\partial}{\partial t}\,\mathrm{y}(t) = -\frac{1}{10}\,\mathrm{y}(t)$$

```
> init_cond := x(0)=1, y(0)=1;
```

$$init\_cond := \mathrm{x}(0) = 1,\ \mathrm{y}(0) = 1$$

```
> funcs := {x(t), y(t)};
```

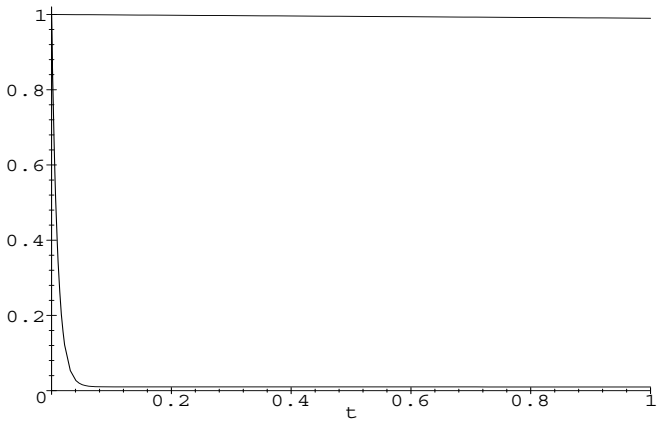
$$funcs := \{\mathrm{x}(t),\ \mathrm{y}(t)\}$$

```
> Sol := dsolve({sys, init_cond}, funcs);
```

$$Sol := \{\mathrm{x}(t) = \frac{989}{999}\,e^{(-100\,t)} + \frac{10}{999}\,e^{(-1/10\,t)},\ \mathrm{y}(t) = e^{(-1/10\,t)}\}$$

Let us display the components of the solution vector:

```
> assign(Sol);
> with(plots):
> P1 := plot(x(t), t=0..1):
> P2 := plot(y(t), t=0..1):
> display({P1, P2});
```



From the result it can be seen that the function e^{-100t} in the solution decays exceedingly fast while the decay of the other term $e^{-0.1t}$ is a thousandfold more sedate. Thus, the solution vector tends to the zero vector as $t \to \infty$.

Now let us try to solve the problem numerically with the explicit Euler's method with the step size $h = 0.025$:

```
> s1 := diff(x(t), x)= -100*x(t) + y(t);
  s2 := diff(y(t), t)= -0.1*y(t);
```

$$s1 := \frac{\partial}{\partial t}\,\mathrm{x}(t) = -100\,\mathrm{x}(t) + \mathrm{y}(t)$$

$$s2 := \frac{\partial}{\partial t}\,\mathrm{y}(t) = -.1\,\mathrm{y}(t)$$

```
> in_cond1 := x(0)=1, y(0)=1;
```

$$in\_cond1 := \mathrm{x}(0) = 1,\ \mathrm{y}(0) = 1$$

```
> num_sol1 := dsolve({s1, s2, in_cond1}, {x(t), y(t)},
              type=numeric, method=classical[foreuler],
              stepsize=0.025):
> for n from 0 to 5 do num_sol1(0.3*n) od;
```

$$[t = 0, \mathrm{x}(t) = 1., \mathrm{y}(t) = 1.]$$

$$[t = .3, \mathrm{x}(t) = 128.4572895542000, \mathrm{y}(t) = .9704090817588188]$$

$$[t = .6, \mathrm{x}(t) = 16665.61199080038, \mathrm{y}(t) = .9416937859599939]$$

$$[t = .9, \mathrm{x}(t) = .2162300910623603\,10^{7}, \mathrm{y}(t) = .9138282021314234]$$

$$[t = 1.2, \mathrm{x}(t) = .2805506233929891\,10^{9}, \mathrm{y}(t) = .8867871865156667]$$

$$[t = 1.5, \mathrm{x}(t) = .3640041597702512\,10^{11}, \mathrm{y}(t) = .8605463393821544]$$

We see that at the step size $h = 0.025$ the Euler's method gives a very bad result, it has a frightful numerical instability. If we choose the step size less then 0.02 we get appropriate results. The situation is quite similar if we apply another explicit numerical methods, discussed earlier. With any of the method, the presence of the e^{-100t} term would require a stepsize $h \ll 2/100$ (as we see below) for the method to be stable. This is so even though the term e^{-100t} is completely neglibile in determining the values of $x(t)$ and $y(t)$ as soon as one is away from the origin. This behaviour in the numerical solution is referred to as *stiffness*. Systems such as this where there is mismatch between the requirements of accuracy and stability, at least for methods with a finite region of absolute stability, are referred to as *stiff*.

DEFINITION 4.1. *A system of differential equations is called* ***stiff system*** *if there are either quickly or slowly varying components of the solution vector.*

We try to give an explanation in the case of Euler's method: If the abowe problem is integrated by the Euler's method, the numerical solution can be represented in "closed form" as follows

$$x_i = C_1(1 - 100h)^i + C_2(1 - 0.1h)^i$$

$$y_i = \qquad\quad + C_3(1 - 0.1h)^i.$$

Evidently, the approximations converge to zero as $i \to \infty$ only if the steplength h is chosen small enough to have

$$|1 - 100h| < 1 \quad \text{and} \quad |1 - 0.1h| < 1. \tag{4.1}$$

The influence of the component e^{-100t} in the exact solution is negligibly small in comparison with $e^{-0.1t}$. Unfortunately, this is not true for the numerical solution. In view of (4.1), indeed, the steplength $h > 0$ must be chosen so that

$$h < \frac{2}{100}.$$

Appropriate numerical methods for stiff differential equations can be derived from *implicit methods.* As an example, we consider the *implicit Euler method,*

$$y_{n+1} = y_n + hf(x_{n+1}, y_{n+1}), \qquad n = 1, 2, ... \tag{4.2}$$

Let us consider the following linear system of differential equations as model problem

$$y' = \mathbf{A}y, \qquad y(0) = y_0, \tag{4.3}$$

where $y(t)$ is the solution vector and $\mathbf{A}$ is a constant $n \times n$ matrix.

If the Euler method is applied to (4.3) with constant steplength, it will produce a sequence y_i of approximation vector for the solution $y(t_i)$ which satisfy a recurrence formula

$$y_{n+1} = g(h\mathbf{A})y_n. \tag{4.4}$$

The function $g(z)$ depends only on the method employed and is usually a rational function in which it is permissible to substitute a matrix for the argument. For the implicit Euler method the formula (4.4) gives

$$y_{n+1} = y_n + h\mathbf{A}y_{n+1}, \quad \text{i.e.} \quad y_{n+1} = (1 - h\mathbf{A})^{-1}y_n,$$

whence

$$g(z) = \frac{1}{1 - z}.$$

Let $\lambda_i \quad i = 1, .., n$ be the eigenvalues of the matrix $\mathbf{A}$. The differential equation (4.3) is stable if all λ_i are negative, or more generally if their real parts are negative. In this case, the solution $y(t)$ of (4.3) converges to zero as $t \to \infty$, while the discrete solution $\{y_n\}$ by virtue of (4.4), converges to zero as $i \to \infty$ only for those stepsizes $h > 0$ for which

$$|g(h\lambda_i)| < 1$$

for all eigenvalues λ_i of $\mathbf{A}$. Now the region of absolute stability of the implicit Euler method is

$$|1 - h\lambda| > 1$$

implying no restricition on the stepsize h for $Re(\lambda) < 0$. Thus although the system is stiff the implicit Euler method can provide an acceptable

solution with a stepsize which is governed by the requirement of accuracy rather than stability.

The more negative λ_i, the shorter the characteristic time. Let the step size h the same for all components of the solution for a numerical method. The step size may be controlled by the most negative eigenvalue λ_i which corresponds to the fastest decay and dies first in the true solution.

DEFINITION 4.2. *If the matrix* $\mathbf{A}$ *has eigenvalues of very different magnitudes, the system* (4.3) *is called* ***stiff****. The quotient of the largest and the smallest (in modulus) eigenvalues of a linear system (and for a general nonlinear system, the eigenvalues of the Jacobian matrix) is refferred to as the* ***stiffness ratio****.*

The stiffness ratio of the Example 4.1. is 10^3. Stiff systems frequently occur in practice, this is typical of certain physical processes, chemical engineering problems, economics etc.. There may be large negative eigenvalues (strong damping of some components) or large imaginary eigenvalues (rapid oscillations). It leads to a strict restriction for the step size. The integration with large step sizes causes instability, therefore, we need to keep

$$|\lambda_{max} h| < C, \quad C = constant$$

For linear differential systems, methods which have a region of absolute stability including the whole left hand plane impose no stability related restriction on the stepsize. Such methods are said to be A-stable.

DEFINITION 4.3. *A numerical method is said to be* ***A-stable*** *if*

$$R_A \supseteq \{h\lambda \mid Re(h\lambda) < 0\} \tag{4.5}$$

where R_A *is the stability domain of the method.*

A slightly less restrictive type of stability is A(α) stability defined as follows:

DEFINITION 4.4. *A numerical method is said to be* ***A(α) stable*** $\alpha \in (0, \frac{\pi}{2})$ *if*

$$R_A \supseteq \{h\lambda \mid \ -\alpha < \pi - arg(h\lambda) < \alpha\} \tag{4.6}$$

The concept of A(α) stability is more adequate, which means that the stability region should include the sector $|arg(h\lambda) - \pi| < \alpha$. The special case of the left half-plane, $\alpha = \pi/2$, is the A-stability.

In Chapter 3 was shown that the trapezoidal method is precisely A-stable. Dahlquist proved that if a multistep method is A-stable, then

$k \leq 2$, i.e. an A-stable multistep method may be at most a two-step method. It can be shown that the trapezoidal method has the smallest truncation error among all A-stable linear multistep methods.

The methods which have been most successful for stiff problems are all implicit.

An alternative way of slackening the requirements of A-stability is to assume that all the eigenvalues which produce the fastest transients lie to the left of a line $Re(h\lambda) = -a$, where $a > 0$. This leads to the definition of stiff stability:

DEFINITION 4.5. *A numerical method is said to be* ***stiffly stable*** *if*

$$R_A \supseteq R_1 \cup R_2$$

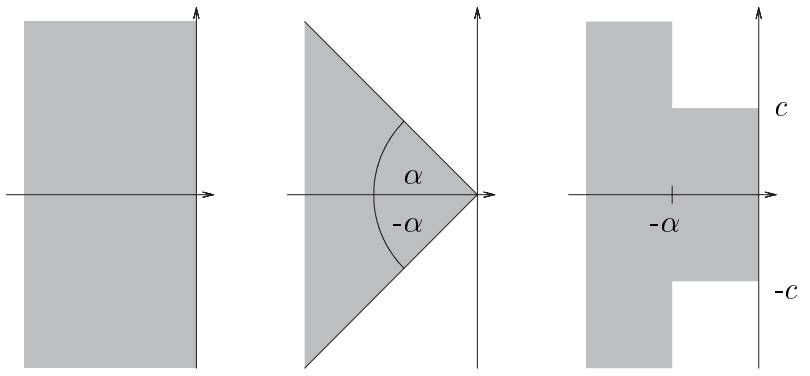
where

$$R_1 = \{h\lambda \mid Re(h\lambda) < -a\}$$
$$R_2 = \{h\lambda \mid -a < Re(h\lambda) < 0,\ -c \leq Im(h\lambda) \leq c\}$$

and a and c are positive real numbers.

The rationale for this definition is as follows. $e^{h\lambda}$ is the change in a component in one step due to an eigenvalue λ. If $h\lambda = u + iv$, then the change in magnitude is e^u. If $u < -a < 0$, then the component is reduced by at least e^{-a} in one step. We are not interested in the accuracy of components that are very small, so for some a we are willing to ignore all components in R_1.

These concepts are illustrated in the diagram below:



A-stable $A(\alpha)$-stable Stiffly-stable

4.2. Advanced methods for stiff systems

• Gear method

Considering [**Ge1**–**Ge4**], in this section we describe the technique used in the Gear's numerical integration program. The method may be either a form of the Adams methods, or a method for stiff equations. The order is chosen to try to maximize the step size. Since the amount of work per step is relatively independent of order, this chois will tend to minimize the amount of work.

The Adams-Bashforth pth order predictor equation for the differential equation $y' = f(t, y)$ is

$$y_n^{[0]} = y_{n-1} + \beta_1 h y'_{n-1} + ... + \beta_p h y'_{n-p}, \tag{4.7}$$

where $y = y(t_i)$ and $t_i = ih$ (h being the step size), $y'_i = f(t_i, y_i)$, and where β_i are given, for example in Chapter 3. The approximation $y_n^{[0]}$ is used in this predictor corrector scheme as the first approximation in the Adams-Moulton corrector formula of pth order given by

$$y_n^{[m+1]} = y_{n-1} + \beta_0^* h f(t_n, y_n^{[m]}) + \beta_1^* h y'_{n-1} + ... + \beta_{p-1}^* h y'_{n-p+1}. \tag{4.8}$$

The coefficients β_i^* can also be found in Chapter 3. If the corrector equation (4.8) is iterated until it converges to y_n (as is guaranteed for small enough h and smooth functions f) the truncation error introduced in the nth step of the integration will be

$$C_{p+1}^A h^{p+1} y^{(p+1)}(t_n) + O(h^{(p+2)}),$$

where $y^{(k)}$ is the kth derivative of y.

The method for stiff equations is similar. It uses a pth order predictor formula of the form

$$y_n^{[0]} = \alpha_1 y_{n-1} + ... + \alpha_p y_{n-p} + \eta_1 h y'_{n-1} \tag{4.9}$$

and a corrector

$$y_n^{[m+1]} = \alpha_1^* y_{n-1} + ... + \alpha_p^* y_{n-p} + \eta_0^* h f(t_n, y_n^{[m]}). \tag{4.10}$$

The truncation error when (4.10) is iterated to convergence is

$$C_{p+1}^S h^{p+1} y^{(p+1)}(t_n) + O(h^{p+2}),$$

where $C_{p+1}^S = 1/(p+1)$. The α_i^* and the η_0^* are given in [**He**].

The predictor corrector equations can be expressed in matrix form. In the case of Adams methods, we are going to define the vector $\mathbf{y_n} = [y_n, h y'_n, h y'_{n-1}, ..., h y'_{n-p+1}]^T$, where T is the transpose operator. Similarly define

$$\mathbf{y_n}^{[m]} = [y_n^{[m]}, h y_n'^{[m]}, h y'_{n-1}, ..., h y'_{n-p+1}]^T,$$

where $hy_n^{'[m]}$ is a symbol for a quantity to be defined below. For $m \geq 1$ we define it as $hf(t_n, y_n^{[m-1]})$. Then we note from (4.8) that

$$y_n^{[m+1]} = y_n^{[m]} + \beta_0^*[hf(t_n, y_n^{[m]}) - hy_n^{'[m]}] \tag{4.11}$$

for $m \geq 1$. Subtracting (4.7) from (4.8) and defining $\beta_p^* = 0$, we obtain

$$\begin{aligned} y_n^{[1]} =& y_n^{[0]} + \beta_0^*[hf(t_n, y_n^{[0]}) - \\ & - \{\frac{\beta_1 - \beta_1^*}{\beta_0^*} hy_{n_1}^{'} + ... + \frac{\beta_p - \beta_p^*}{\beta_0^*} hy_{n-p}^{'}\}]. \end{aligned} \tag{4.12}$$

We set $\delta_i = (\beta_i - \beta_i^*)/\beta_0^*$ and define

$$hy_n^{'[0]} = \delta_1 hy_{n-1}^{'} + ... + \delta_p hy_{n-p}^{'}. \tag{4.13}$$

Now (4.12) is equivalent to (4.11) when $m = 0$. We note that (4.7) and (4.13) may be written as

$$\mathbf{y_n}^{[0]} = \mathbf{B}\mathbf{y_{n-1}}, \tag{4.14}$$

where

$$\mathbf{B} = \begin{bmatrix} 1 & \beta_1 & \beta_2 & \dots & \beta_{p-1} & \beta_p \\ 0 & \delta_1 & \delta_2 & \dots & \delta_{p-1} & \delta_p \\ 0 & 1 & 0 & \dots & 0 & 0 \\ . & . & . & & . & . \\ . & . & & . & . & . \\ . & . & & . & . & . \\ 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix}.$$

Noting that

$$\begin{aligned} hy_n^{'[m+1]} &= hf(t_n, y_n^{[m]}) \\ &= hy_n^{'[m]} + [hf(t_n, y_n^{[m]}) - hy_n^{'[m]}], \end{aligned}$$

we see that (4.11) may be written as

$$\mathbf{y_n}^{[m+1]} = \mathbf{y_n}^{[m]} + \mathbf{c}F(\mathbf{y_n}^{[m]}), \tag{4.15}$$

where $\mathbf{c} = [\beta_0^*, 1, 0, ..., 0]^T$ and

$$F(\mathbf{y_n}^{[m]}) = hf(t_n, y_n^{[m]}) - hy_n^{'[m]}. \tag{4.16}$$

After M iteration we accept the result by setting

$$\mathbf{y_n} = \mathbf{y_n}^{[M]}.$$

For *stiff methods* we define the vector

$$\mathbf{y_n} = [y_n, hy_n^{'}, y_{n-1}, ..., y_{n-p}]^T$$

and perform similar operations. If we define the coefficients $\gamma_i = (\alpha_i - \alpha_i^*)/\eta_0^*$ and $\delta_1 = \eta_1/\eta_0^*$ we also arrive at eqs. (4.14) and (4.15) where the matrix $\mathbf{B}$ is now given by

$$\mathbf{B} = \begin{bmatrix} \alpha_1 & \beta_1 & \alpha_2 & \dots & \alpha_{p-1} & \alpha_p \\ \gamma_1 & \delta_1 & \gamma_2 & \dots & \gamma_{p-1} & \gamma_p \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ . & . & & . & . & . \\ . & . & & . & . & . \\ 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix}.$$

and $\mathbf{c}$ is as before.

We now note that the predictor formulas (4.7) and (4.9) are equivalent to fitting a pth degree polynomial through the known information carried in $\mathbf{y_{n-1}}$. Instead of saving the information in this form, we will make a linear transformation $\mathbf{Q}$ and save $\mathbf{z_{n-1}} = \mathbf{Q y_{n-1}}$. The transformation $\mathbf{Q}$ is chosen so that the $p+1$ components of $\mathbf{z_{n-1}}$ are the function value y_{n-1} and the first p derivatives of the polinomial used in the prediction process. If the kth derivative saved in $\mathbf{z_{n-1}}$ is scaled by $h^k/k!$ the matrix $\mathbf{Q}$ is independent of h.

Thus we have $\mathbf{z_n} = [y_n, hy_n^{'}, ..., h^p y_n^{(p)}/p!]^T$, where $y_n^{(k)}$ is the kth derivative of the approximating polynomial. By applying the transformation $\mathbf{Q}$ to eqs. (4.14) and (4.15) we get

$$\mathbf{z_n}^{[0]} = \mathbf{Q y_n}^{[0]} = \mathbf{Q B Q^{-1} z_{n-1}}, \tag{4.17}$$

$$\mathbf{z_n}^{[m+1]} = \mathbf{Q y_n}^{[m+1]} = \mathbf{z_n}^{[m]} + \mathbf{l} F(\mathbf{Q^{-1} z_n}^{[m]}), \tag{4.18}$$

where $\mathbf{l} = \mathbf{Qc}$. Since both $\mathbf{z_n}$ and $\mathbf{y_n}$ have y_n and $hy_n^{'}$ as their first two components and F depends on these only, $F(\mathbf{Q^{-1} z_n}) = F(\mathbf{z_n})$. The matrix $\mathbf{Q}$ depends on whether the Adams or stiff methods are used; hence $\mathbf{l}$ depends on the method. The $\mathbf{l}$ for the Adams methods are given in [**Ge4**]; those for stiff methods are given [**Ge3**]. (This formulation of the Adams method is essentially the same as the Nordsieck method [**No**].) The matrix $\mathbf{QBQ^{-1}}$ provides a pth order approximations to $\mathbf{z_n}^{[0]}$ in terms of $\mathbf{z_{n-1}}$; hence it is the Pascal triangle matrix

$$\begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 & 1 \\ & 1 & 2 & 3 & \dots & p-1 & p \\ & & 1 & 3 & \dots & . & . \\ & & & 1 & \dots & . & . \\ & & & & & . & . \\ & 0 & & & & 1 & p \\ & & & & & & 1 \end{bmatrix}$$

for either method. (The entries in the columns of this matrix are the binomial coefficients.)

A complication occurs in the case of stiff equations: Iteration (4.10) does not converge unless h is very small. To overcome this problem we note that the effect of any number of iterations of (4.18) is to compute

$$\mathbf{z_n} = \mathbf{z_n}^{[0]} + \mathbf{l}b, \tag{4.19}$$

where b is scalar. If h is small enough, the iteration will converge to a solution of

$$F(\mathbf{z_n}) = 0. \tag{4.20}$$

If we attempt to solve (4.20) directly by Newton's method using (4.19) and starting with a first approximation of $\mathbf{z_n}^{[0]}$, we get successive approximations given by

$$\mathbf{z_n}^{[m+1} = \mathbf{z_n}^{[m]} + \mathbf{l}[-(\partial F/\partial \mathbf{z})\mathbf{l}]^{-1} F(\mathbf{z_n}^{[m]}),$$

where

$$[-(\partial F/\partial \mathbf{z})\mathbf{l}]^{-1} = [l_1 - hl_0 \partial f/\partial y]^{-1} = W.$$

(In the case of system of equations, W is a matrix.) Therefore, if stiff methods are used, the program multiplies the value of F by W before performing the correction. If $f(t, y)$ is linear in y, the corrector will converge in one iteration. If it is nonlinear, several iterations may be needed. This process will converge for sufficiently small h since Newton's method is convergent in some neighborhood of the solution, and since, as h tends to 0, $y_n^{[0]}$ tends to y_n. For most functions $f(t, y)$ that occur, large values of h still permit rapid convergence.

EXAMPLE 4.2. *Let us solve the Curtiss-Hirschfelder equation*

$$y' = -50(y - cos(t)), \qquad 0 \le t \le 10, \qquad y(0) = 1.$$

The equation has significance as a good test case for computational algorithm because it is a moderately stiff problem.

SOLUTION. Let us find the exact solution by the Maple.

```
> eq := diff(y(t), t) = -50*(y(t) - cos(t));
```

$$eq := \frac{\partial}{\partial t}\,\mathrm{y}(t) = -50\,\mathrm{y}(t) + 50\cos(t)$$

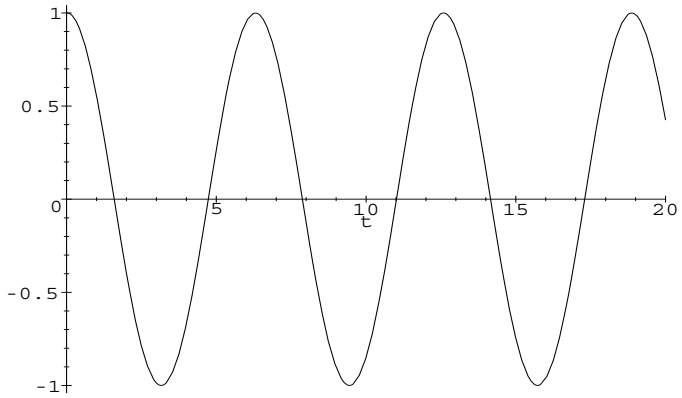
```
> init_cond := y(0) = 1;
```

$$init\_cond := \mathrm{y}(0) = 1$$

```
> Sol := dsolve({eq, init_cond}, y(t));
```

$$Sol := \mathrm{y}(t) = \frac{2500}{2501}\cos(t) + \frac{50}{2501}\sin(t) + \frac{1}{2501}\,e^{(-50\,t)}$$

```
> assign(Sol):
> plot(y(t), t=0..20);
```



The solution is a smooth solution in the vicinity of $y \approx \cos t$ and all other solutions reach this one after a rapid "transient phase".

Now, we solve the problem by means of the Euler's method

```
> num_sol := dsolve({eq, init_cond}, y(t),
            type=numeric, method=classical[foreuler],
            stepsize=0.05):
> for n from 0 to 5 do num_sol(2*n) od;
```

$$[t = 0,\ \mathrm{y}(t) = 1.]$$

$$[t = 2,\ \mathrm{y}(t) = -1106.564450996085]$$

$$[t = 4,\ \mathrm{y}(t) = -.1223125006062327\,10^{11}]$$

$$[t = 6,\ \mathrm{y}(t) = -.1352449966133859\,10^{18}]$$

$$[t = 8,\ \mathrm{y}(t) = -.1495448872298088\,10^{25}]$$

$$[t = 10,\ \mathrm{y}(t) = -.1653567514995976\,10^{32}]$$

We observe that whenever the step size is a little too large ($h > 0.04$), the numerical solution goes too far beyond the equilibrium and violent oscillations occur. □

Criteria for error control

The average user would like to be able to specify an error parameter which would cause the program to bound the error over the whole interval.

If, for a single equation, $\partial f/\partial y = 0$, the total is the sum of the truncation error in each step. If these errors are bounded by ε (which is the optimum choice for a fixed order method), the total error is bounded by $N\varepsilon$ in N steps. This leads us to consider bounding the error in a step of length h by $h\varepsilon$. Then the total error would be bounded by ε per unit interval.

Gear made some tests, using both ε and $h\varepsilon$ as the error bound in each step. The latter was unsatisfactory for stiff problems for the following reasons: Usually the object of integrating a stiff problem is to continue until the system is in equilibrium. Toward the end of the integration, very large steps may be taken. An error control of $h\varepsilon$ per step would then allow large errors to be made. However, the best choice for these problems is to make large errors initially, as they are later damped out. Therefore, the program controls an estimate of the error per step to be less than ε. The decision whether this is a relative or an absolute error control depends on the solution. If it is growing, a relative control is used; if it is decaying, an absolute control is used.

Starting, step, and order changing

Starting is achieved by setting the order to one the first call. For this order, $\mathbf{z_0}$ is $[y_0, hy_0']^T$. Since y_0 is given and hy_0' can be computed from $hf(t_0, y_0)$, there is no starting problem.

A change of step size requires only that components of $\mathbf{z}$ be scaled by powers of the change. A decrease in order corresponds to discarding a component of $\mathbf{z}$. When the order is increased from p to $p+1$, the backward difference of the last component of $\mathbf{z}$ divided by $p+1$ gives an estimate of $h^{p+1}y^{(p+1)}/(p+1)!$ to be appended to $\mathbf{z}$.

Invoking the **dsolve** function with the `type=numeric` option and `method=mgear` or `method=mgear[choices]` causes a numerical solution to be found by way of a Gear multi-step method. The chices of the mgear method are `adamspc, msteppart, mstepnum`. The first choice corresponds to an Adams predictor-corrector method. `msteppart` is a multi-step method suitable for stiff systems, and which evaluates the Jacobian matrix of the system at each step. `mstepnum` is essentially the

same as the above, however the Jacobian is computed using numerical differencing of the derivatives.

It can be asked about the existence of stiffly stable methods of order greater than two. It is shown in [**Ge3**], that the k-step methods presented by Gear are stiffly stable for $k \leq 6$ for some a and c and of order k, with $\sigma(\zeta) = \zeta^k$. The result is obtained by first computing $\varrho(\zeta)$ from $\sigma(\zeta)$ so as to get an order k method. The coefficients of the polynomials $\varrho(\zeta)$ are in the Table 3.3. (See also [**Ge1**] §8.1.1.) The locus in the z-plane for which a root of $\varrho(\zeta) + z\sigma(\zeta) = 0$ has magnitude one can then be plotted by plotting $z = -\varrho(e^{i\theta})/\sigma(e^{i\theta})$, where $\theta \in [0, 2\pi]$. These loci for $k = 1, 2, 3$ and $k = 4, 5, 6$ can be plotted in the following way

```
> sigma := (k, x) -> x^k;
```

$$\sigma := (k,\, x) \to x^k$$

```
> rho1 := x -> x-1;
```

$$\rho 1 := x \to x - 1$$

```
> rho2:= x-> (3/2)*x^2-2*x+(1/2);
```

$$\rho 2 := x \to \frac{3}{2}\, x^2 - 2\, x + \frac{1}{2}$$

```
> rho3 := x-> (11/6)*x^3-3*x^2+(3/2)*x-1/3;
```

$$\rho 3 := x \to \frac{11}{6}\, x^3 - 3\, x^2 + \frac{3}{2}\, x - \frac{1}{3}$$

```
> rho4 := x -> (25/12)*x^4-4*x^3+3*x^2-(4/3)*x+(1/4);
```

$$\rho 4 := x \to \frac{25}{12}\, x^4 - 4\, x^3 + 3\, x^2 - \frac{4}{3}\, x + \frac{1}{4}$$

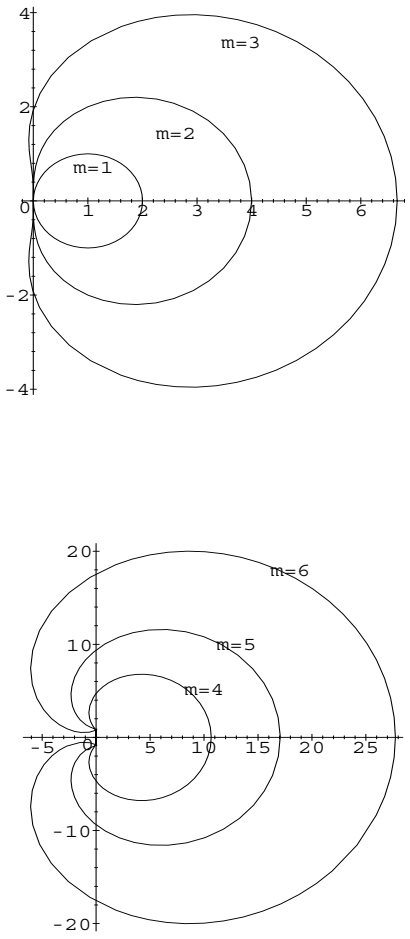
```
> rho5 := x-> (137/60)*x^5-5*x^4+5*x^3-
             (10/3)*x^2+(5/4)*x-(1/5);
```

$$\rho 5 := x \to \frac{137}{60}\, x^5 - 5\, x^4 + 5\, x^3 - \frac{10}{3}\, x^2 + \frac{5}{4}\, x - \frac{1}{5}$$

```
> rho6 := x-> (49/20)*x^6-6*x^5+(15/2)*x^4-
             (20/3)*x^3+(15/4)*x^2-(6/5)*x+(1/6);
```

$$\rho 6 := x \to \frac{49}{20}\,x^6 - 6\,x^5 + \frac{15}{2}\,x^4 - \frac{20}{3}\,x^3 + \frac{15}{4}\,x^2 - \frac{6}{5}\,x + \frac{1}{6}$$

```
> with(plots):
> t := I*theta:
> am1 := complexplot(rho1(exp(t))/sigma(1, exp(t)),
                     theta=0..2*Pi):
> am2 := complexplot(rho2(exp(t))/sigma(2, exp(t)),
                     theta=0..2*Pi):
> am3 := complexplot(rho3(exp(t))/sigma(3, exp(t)),
                     theta=-Pi..Pi):
> am4 := complexplot(rho4(exp(t))/sigma(4, exp(t)),
                     theta=0..2*Pi):
> am5 := complexplot(rho5(exp(t))/sigma(5, exp(t)),
                     theta=0..2*Pi):
> am6 := complexplot(rho6(exp(t))/sigma(6, exp(t)),
                     theta=0..2*Pi):
> pr1 := textplot({[1.5,1.7, `m=1`], [3,2, `m=2`],
                   [4.2,3.8, `m=3`]}, align=LEFT):
> pr2 := textplot({[12,5.2, `m=4`], [15,10, `m=5`],
                   [20,18, `m=6`]}, align=LEFT):
> display({am1, am2, am3, pr1},scaling=CONSTRAINED);
  display({am4, am5, am6, pr2},scaling=CONSTRAINED);
```



All roots for z outside of the closed locus are less then one in magnitude. Thus, the absolute stability region is the *exterior* of the closed curves. For $k = 7 - 15$, these methods are not stiffly stable.

• LSODE the Livermore Stiff ODE solver

LSODE is the "Livermore Solver" of Hindmarsh [**Hi1**]. The code is based on the Nordsieck representation of the fixed step size backward differentiation formalae methods. It emerged from a long development starting with Gear's DIFSUB program in 1971. Its exemplary user interface and ease of application has been a model for much subsequent ODE software. The method allows us to choose between analytically

supplied Jacobian or numerically computed finite difference approximations as well as between full or banded linear algebra. Maple also contains this method in its option

`type = numeric` and `method = lsode`

The various choices of `lsode` method we refer to the on-line help system for immediate help. Another reference on the `lsode` procedure is [**Hi3**].

EXAMPLE 4.3. *Let us solve the two-dimensional differential equation system of Van der Pol*

$$x'(t) = y(t)$$

$$y'(t) = \mu(1 - x^2)y - x$$

where μ is a positive constant.

We choose the initial condition as follows: $x(0) = -3$, $y(0) = 2$ and $x(0) = 0$, $y(0) = 1/2$, and let the parameter $\mu = 0.2, 1, 5, 10$ etc.

Plot the phase trajectories in the plane xy and plot $x(t)$ with respect to t.

It is possible to show, that the problem does have a unique limit cycle, it has a stable periodic solution whose period and amplitude depend on the parameter μ.

We investigate the solutions by Maple.

```
> with(plots):
> vderpol := diff(x(t), t) = y(t),
          diff(y(t), t) = -x(t)+epsilon*(1-x(t)^2)*y(t);
```

$$vderpol := \frac{\partial}{\partial t}\,\mathrm{x}(t) = \mathrm{y}(t), \quad \frac{\partial}{\partial t}\,\mathrm{y}(t) = -\mathrm{x}(t) + \varepsilon\,(1 - \mathrm{x}(t)^2)\,\mathrm{y}(t)$$

```
> ics := x(0)=0, y(0)=1/2;
```

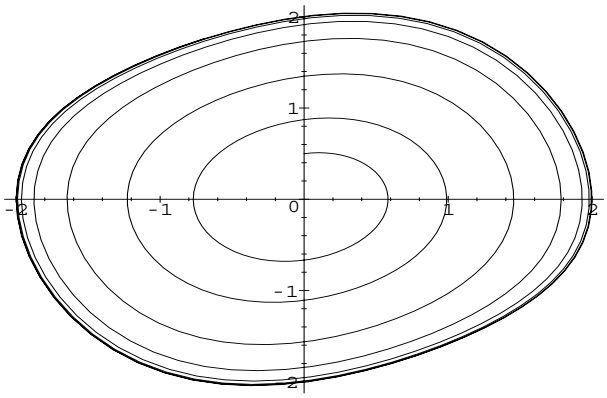
$$ics := \mathrm{x}(0) = 0,\ \mathrm{y}(0) = \frac{1}{2}$$

```
> epsilon := 0.2:
> f0 := dsolve({vderpol, ics}, {x(t), y(t)},
        type=numeric, output=listprocedure):
> fx0:  = subs(f0, x(t)):  fy0 := subs(f0, y(t)):
> fx0(15); fy0(15);
```

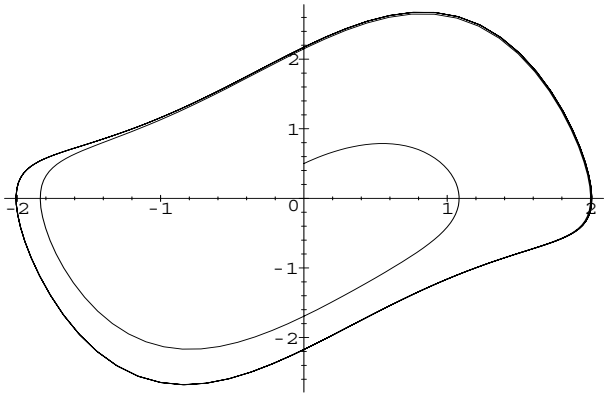
$$.9945526048666897$$

$$-1.036824196439655$$

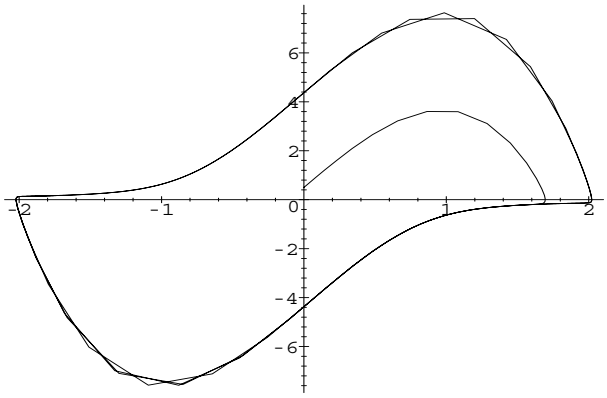
```
> odeplot(f0, [x(t), y(t)], 0..50, numpoints=500);
```



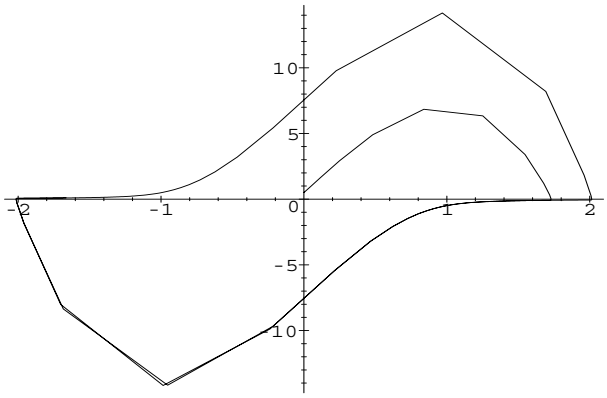

```
> epsilon := 1:
> F1 := dsolve({vderpol, ics}, {x(t),y(t)},
        type=numeric, method=mgear[adamspc]);
> odeplot(F1, [x(t), y(t)], 0..30, numpoints=500);
```



```
> epsilon := 5:
> F2 := dsolve({vderpol, ics}, {x(t), y(t)},
        type=numeric, method=lsode[backfunc]):
> odeplot(F2, [x(t), y(t)], 0..30, numpoints=500);
```




```
> epsilon := 10:
> F3 := dsolve({vderpol, ics}, {x(t), y(t)},
        type=numeric, method=lsode[adamsfull]):
> odeplot(F3, [x(t), y(t)], 0..30, numpoints=500);
```



We see that this equation is easily integrate for moderate values of μ. But if we choose $\mu > 100$, the problem might become difficult. It turns out that the period of the solution increases with μ.

4.3. Delay differential equations

Delay differential equations are equations with "retarded arguments" such as

$$y'(t) = f(t, y(t), y(t-\tau)) \tag{4.21}$$

or of even more general form

$$y'(t) = f(t, y(t), y(t-\tau_1), y(t-\tau_2)). \tag{4.22}$$

Here the derivative of the solutions depends also on its values at previous points.

Retarded arguments are present in many models of applied mathematics. They can also be the source of interesting mathematical phenomena such as instabilities, limit cycles, periodic behaviour.

• The existence of the solution

For equations of the type (4.21) or (4.22), where the delay values $t - \tau$ are bounded away from t by a positive constant, the question of existence is an easy matter: suppose that the solution is known, say

$$y(t) = \Phi(t) \qquad \text{for} \qquad t_0 - \tau \le t \le t_0. \tag{4.23}$$

Then $y(t-\tau)$ is a known function of t for $t_0 \le t \le t_0 + \tau$ and (4.21) becomes an ordinary differential equation, which can be treated by known existence theories. We then know $y(t)$ for $t_0 \le t \le t_0 + \tau$ and can compute the solution for $t_0 + \tau \le t \le t_0 + 2\tau$ and so on. This, so-called "method of steps" then yiels existence and uniqueness result for all t. For more details see [**BeCo**] and [**Dri**].

EXAMPLE 4.4. *Let us consider the equation*

$$y'(t) = -y(t-1), \qquad y(t) = 1 \qquad \textit{for} \qquad -1 \le t \le 0. \tag{4.24}$$

Procceeding as describe above, we obtain

$$y(t) = 1 - t \qquad \text{for} \qquad 0 \le t \le 1$$

$$y(t) = 1 - t + \frac{(t-1)^2}{2!} \qquad \text{for} \qquad 1 \le t \le 2$$

$$y(t) = 1 - t + \frac{(t-1)^2}{2!} - \frac{(t-1)^3}{3!} \quad \text{for} \quad 2 \le t \le 3, \quad \text{etc.}$$

It can be observed, that despite the fact that the differential equation and the initial function are C^∞, the solution has discontinuities in its derivatives. This results from the fact that the initial function does not satisfy the differential equation. With every time step τ, however, these discontinuities are smoothed out more and more.

Our next example clearly illustrates the fact that the solutions of a delay equation depend on the entire history between $t_0 - \tau$ and t_0, and not only on the initial value:

EXAMPLE 4.5. *Let us solve the equation*

$$y'(t) = -1.4y(t-1) \tag{4.25}$$

supposing that the solution is known on the intervals

(a) $\Phi(t) = 0.8 \quad for \quad -1 \le t \le 0,$

(b) $\Phi(t) = 0.8 + t \quad for \quad -1 \le t \le 0,$

(c) $\Phi(t) = 0.8 + 2t \quad for \quad -1 \le t \le 0.$

• Application of numerical methods

If we apply the Runge-Kutta method (Ch.2., formulas (2.32)) to a delay equation (4.21) we obtain

$$g_i^{(n)} = y_n + h\sum_j a_{ij} f(t_n + c_j h, g_j^{(n)}, y(t_n + c_j h - \tau))$$

$$y_{n+1} = y_n + h\sum_j b_j f(t_n + c_j h, g_j^{(n)}, y(t_n + c_j h - \tau)).$$

But which values should we give to $y(t_n + c_j h - \tau)$? If the delay is constant and satisfies $\tau = kh$ for some integer k, the most natural idea is to use the back-values of the old solution

$$g_i^{(n)} = y_n + h\sum_j a_{ij} f(t_n + c_j h, g_j^{(n)}, \gamma_j^{(n)}) \tag{4.26}$$

$$y_{n+1} = y_n + h\sum_j b_j f(t_n + c_j h, g_j^{(n)}, \gamma_j^{(n)}) \tag{4.27}$$

where

$$\gamma_j^{(n)} = \begin{cases} \Phi(t_n + c_j h - \tau) & \text{if} \quad n < k \\ g_j^{(n-k)} & \text{if} \quad n \ge k. \end{cases}$$

This can be interpreted as solving successively

$$y'(t) = f(t, y(t), \Phi(t-\tau))$$

for the interval $[t_0, t_0 + \tau]$, then

$$y^{'}(t) = f(t, y(t), z(t))$$
$$z^{'}(t) = f(t - \tau, z(t), \Phi(t - 2\tau))$$

for the interval $[t_0 + \tau, t_0 + 2\tau]$, then

$$y^{'}(t) = f(t, y(t), z(t))$$
$$z^{'}(t) = f(t - \tau, z(t), \nu(t))$$
$$\nu^{'}(t) = f(t - 2\tau, \nu(t), \Phi(t - 3\tau))$$

for the interval $[t_0+2\tau, t_0+3\tau]$, and so on. This is the perfect numerical analog of the "method of steps" mentioned above.

It can be proved, that if a_{ij}, b_j, c_i are the coefficients of a p-th order Runge-Kutta method, then (4.26), (4.27) is convergent of order p.

Unfortunately, this method does not allow us to change the step size arbitrarily, and an application to variable delay equations is not straightforward. If complete flexibility is desired, we need a *global approximation* to the solution. There is no use in having approximations only at a sequence of points. Therefore, choice methods for these problems are multistep methods of Adams or BDF type or continuous Runge-Kutta methods.

4.4. Exercises

1. Obtain a numerical solution of the system of differential equations

$$\begin{aligned} x^{'} &= -2000x + 999.75y + 1000.25, & x(0) &= 0 \\ y^{'} &= x - y, & y(0) &= -2 \end{aligned}$$

by

: a) the classical 4th order Runge-Kutta method,

: b) direct application of the Trapezoidal method.

Use a variety of stepsizes. Solve the equations exactly. Explain what happens.

2. Let us examine the so-called simplified "Brusselator" model which have important applications to the interpretation of biological phenomena

$$x^{'}(t) = A + x^2 y - (B + 1)x$$

$$y^{'}(t) = Bx - x^2 y.$$

The system has one critical point: $x = A,\ y = B/A$. For $B > A^2 + 1$ it has a limit cycle which, by numerical calculations, is seen to be unique. Plot the phase-trajectory with different A and $B : A = 1,\ B = 3$, etc.

3. Let us consider the equations of the spherical pendulum in spherical coordinates:

$$\phi'' = -2\frac{cos\psi}{sin\psi}\phi'\psi'$$

$$\psi'' = sin\psi cos\psi(\phi')^2 - sin\psi.$$

Plot the solution curve in $0 \le t \le 20$ and $0 \le t \le 100$.

4. Let us solve the "full Brussellator" model and plot the two-dimensional projections of the solutions.

$$x'(t) = 1 + x^2y - (z+1)x$$

$$y'(t) = xz - x^2y$$

$$z'(t) = -xz + \alpha.$$

The system posses a critical point at $x = 1,\ y = z = \alpha$. The condition for stability is $\alpha < 1.21922$. Thus when α increases beyond this value, there arises a limit cycle which exists for all values of α up to approximately 1.5. When α continues to grow, the limit cycle "explodes" and $x \to 0$ while y and $z \to \infty$. So the system has a completely different behavior from the simplified model.

5. A famous chemical reaction with a limit cycle in three dimensions is the "Oregonator" reaction, with a periodic solution describing the Belusov-Zhabotinskii reaction.

$$x'(t) = 77.27(y + x(1 - 8.375 \times 10^{-6}x - y))$$

$$y'(t) = \frac{1}{77.27}(z - (1+x)y)$$

$$z'(t) = 0.161(x - z)$$

$$x(0) = 1,\ y(0) = 2,\ z(0) = 3, \quad t = 30, 60, 90, ..., 360.$$

This is an example of a stiff differential equation whose solutions change rapidly over many orders of magnitude. It is a challenging example for numerical codes.

6. The chemical reaction of Robertson

$$x^{'}(t) = -0.04x + 10^4 yz$$

$$y^{'}(t) = 0.04x - 10^4 yz - 3 \times 10^7 y^2$$

$$z^{'}(t) = \qquad\qquad 3 \times 10^7 y^2$$

$$x(0) = 1,\ y(0) = 0,\ z(0) = 0,$$

one of the most prominent examples of the stiff literature. Hindmarsh discovered that many codes fail is t becomes very large (10^{11} for example). The reason is that whenewer the numerical solution of y accidentally becomes negative, it then tends to $-\infty$ and then run ends by overflow. Therefore let us try to choose $t_{out} = 1, 10, 10^2, 10^3, ..., 10^{11}$.

7. Let us consider the famous Lorenz model which was established for the weather prediction:

$$x^{'}(t) = -\sigma x + \sigma y$$

$$y^{'}(t) = -xz + rx - y$$

$$z^{'}(t) = xy - bz$$

where $\sigma,\ r$ and b are positive constants. Plot the solution curve in the planes xy and xy, if $b = 8/3,\ \sigma = 10$ and $r = 28$ and with the initial value $x = -8,\ y = 8,\ z = r - 1$. The solution curve looks pretty chaotic.

8. An example from population dynamics: Let $y(t)$ represent the population of a certain species, whose development as a function of time is to be studied. If we assume the growth rate to depend on the population of the *preceding* generation, we get a delay differential equation

$$y^{'}(t) = (a - y(t-1))y(t).$$

All solutions with initial value $y(0) > 0$ tend asymptotically to a as $t \to \infty$. This equation has an equilibrium point at $y(t) = a$. This point is locally stable if $0 < a \leq \pi/2$. It has two real solutions iff $a < 1/e = 0.368$, which makes monotonic solutions possible; otherwise they are oscillatory. For $a > \pi/2$ the equilibrium solution is unstable and gives rise to a periodic limit cycle. Let us integrate the problem for $0 \leq t \leq 10$ with $a = 0.35,\ 0.5,\ 1.,\ 1.4,\ 1.6$, and with initial values $y(t) = 0,\ -1 \leq t < 0,\ y(0) = 0.1$.

9. Predator-prey model. Consider, for example, foxes and rabbits in a closed forest. We will denote by $x(t)$ and $y(t)$ the population of the prey and predator, respectively, at time t. Besides some assumptions, we are led to the dynamical system consists of the two-dimensional differential equation system

$$x^{'}(t) = ax - bxy$$

$$y^{'}(t) = -cy + dxy$$

where $a,\ b,\ c,\ d > 0$. The prey are the only food source available to the predator. Thus, if $x = 0$, the predator population decreases exponentially at the rate c. If $y = 0$, the prey population grows exponentially to infinity at the rate a. b and d are the measures of the effect of the interaction between the two species. This equations are known as the Lotka-Volterra equations.

Discuss the solution of the system

$$x^{'} = x - 0.5xy$$

$$y^{'} = -0.75y + 0.25xy$$

for x and y positive. Plot the phase trajectories with different initial values. Plot the graph of the prey and predator populations with time.

10. Quasiperiodic motion on a two-dimensional torus. The underlying three-dimensional system consists of the differential equations

$$x^{'} = (a-b)x - cy + x(z + d(1.0 - z^2))$$

$$y^{'} = cx + (a-b)y + y(z + d(1.0 - z^2))$$

$$z^{'} = az - (x^2 + y^2 + z^2).$$

The parameter values are $a = 2.105$, $b = 3.0$, $c = 0.25$, and $d = 0.2$. Let the time step be in the numerical simulation 0.1 time units. Show the trajectory after 5000 and 20000 iterations. In the last case the torus will more densely be covered by the trajectory.

For other economical model we refer to [**Lo**].

Bibliography

[AB] Abel, M.L. and Braselton, J.P., *Differential Equations with Maple V.* AP Professional, Boston, 1994.

[BeCo] Bellmann, R. and Cooke, K.L., *Differential-Difference Equations*, Academic Press, 1963.

[BG] Birkhoff, G. and Gian-Carlo, Rota, *Ordinary Differential Equations.* John Wiley and Sons, Inc., New York, 1989.

[BD] Boyce, W.E. and DiPrima R.C., *Elementary Differential Equations and Boundary Value Problems.* John Wiley and Sons, Inc., New York, (5th ed.), 1992.

[Bu] Butcher, J.C. *The Numerical Analysis of Ordinary Differential Equations.* John Wiley and Sons, Inc., New York, 1987.

[CL] Coddington, E.A. and Levinson, N., *Theory of Ordinary Differential Equations.* McGraw-Hill Book Company, Inc., New York, 1955.

[Co] Collatz, L., *The Numerical Treatment of Differential Equations.* Springer-Verlag, Berlin-Heidelberg-New York, 1966.

[CB] Conte, S.D. and de Boor, C., *Elementary Numerical Analysis: An Algorithmic Aproach.* McGraw-Hill Kögakusha, Tokyo, (3rd ed.). 1980.

[DB] Dahlquist, G. and Björk, Å., *Numerical Methods.* Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1994.

[Dri] Driver, R.D., *Ordinary and delay differential equations*, Applied Math. Sciences 20, Springer Verlag, 1977.

[EW] Eldén, L. and Wittmeyer-Koch, L., *Numerical Analysis, An Introduction.* Academic Press, Inc., Boston, 1990.

[Ge1] Gear, C.W., *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1971.

[Ge2] Gear, C.W., *The automatic integration of ordinary differential equations.* Comm. ACM **14** (Mar. 1971), pp. 176-179.

[Ge3] Gear, C.W., *The automatic integration of stiff ordinary differential equations*, Information Processing 68, A.J.H. Morrell, Ed., North Holland, Amsterdam, 1969, pp. 187-193.

[Ge4] Gear, C.W., *The numerical integration of ordinary differential equations*, Math. Comp. **21,** 2 (Apr. 1967), pp. 146-156.

[H1] Hairer, E., Nørsett, S.P. and Wanner, G., *Solving Ordinary Differential Equations I. Nonstiff Problems.* Springer Verlag, Berlin, (2nd ed.), 1991.

[H2] Hairer, E., Wanner, G., *Solving Ordinary Differential Equations II. Stiff Problems and Differential-algebraic Equations.* Springer Verlag, Berlin, 1991.

[HW] Hall, G. and Watt, J.M. (Eds.), *Modern Numerical Methods for Ordinary Differential Equations.* Clarendon Press, Oxford. 1976.

[Ha] Hamming, R.W., *Numerical Methods for Scientists and Engineers.* McGraw-Hill Book Company, Inc., New York, (2nd ed.) 1973.

[Hr] Hartman, Ph., *Ordinary Differential Equations.* John Wiley and Sons, Inc., New York, 1964.

[HH] Hämmerlin, G. and Hoffmann, K-H., *Numerical Mathematics.* Springer-Verlag, New York Inc., 1991.

[Hc] Heck, A., *Introduction to Maple.* Springer-Verlag, New York, 1993.

[He] Henrici, P., *Discrete Variable Methods in Ordinary Differential Equations.* John Wiley and Sons, Inc., New York, 1962.

[HJ] Higham, N. J., *Accuracy and Stability of Numrical Algorithms*, SIAM, Philadelphia, 1996.

[Hi] Hildebrand, F.B., *Introduction to Numerical Analysis.* McGraw-Hill Book Company, New York, 1974.

[Hi1] Hindmarsh, A.C., *LSODE and LSODI, two new initial value ordinary differential equation solvers*, ACM-SIGNUM Newsletter 15, 1980, pp. 10-11.

[Hi2] Hindmarsh, A.C., *GEAR: ordinary differential equation system solver*, UCID-30001, Rev. 2, LLL, Livermore, Calif. 1972.

[Hi3] Hindmarsh, A.C., *ODEPACK, a Systemized Collection of ODE Solvers*, In: Scientific Computing, R.S. Stepleman et al. (eds.) North-Holland, Amsterdam, 1983.

[IK] Isaacson, E. and Keller, H.B., *Analysis of Numerical Methods.* John Wiley and Sons, Inc., New York, 1966.

[Is] Iserles, A., *A First Course in the Numerical Analysis of Differential Equations.* Cambridge Text in Applied Mathematics. Cambridge Univ. Press., 1996.

[Ka] Kamke, E., *Differentialgleichungen, Lösungsmethoden und Lösungen, Vol. 1.* Leipzig, 1959.

[KM] Kopchenova, N.V. and Maron, I.A., *Computational Mathematics, Worked Examples and Problems with Elements of Theory.* Mir Publishers, Moscow, 1975.

[La] Lambert, J.D., *Numerical Methods for Ordinary Differential Systems.* John Wiley and Sons, Ltd., Chichester, 1991.

[Lo] Lorentz, H.W., *Nonlinear Dynamical Economics and Chaotic Motion.* Springer Verlag, Berlin-Heidelberg, 2nd Ed., 1993.

[No] Nordsieck, A., *On numerical integration of ordinary differential equations*, Math. Comp. **16,** 1 (Jan. 1962), pp. 22-49.

[PT] Press, W.H., Teukolsky, S.A., Vatterling, W.T. and Flannery, B.P., *Numerical Recipes in C. The Art of Scientific Computing.* Second Ed., Cambridge Univ. Press, 1992.

[RR] Ralston, A. and Rabinowitz, P., *A First Course in Numerical Analysis.* McGraw-Hill Book Company, New York, 1978.

[Sc] Schwarz, H.R., *Numerical Analysis, A Comprehensive Introduction.* John Wiley and Sons, Ltd., Chichester, 1989.

[SG] Shampine, L.F. and Gordon, M.K., *Computer Solution of Ordinary Differential Equations.* W.H. Freeman, San Francisco, 1975.

[St] Stetter, H.J., *Analysis of Discretization Methods for Ordinary Differential Equations.* Springer Tracts in Natural Philosophy. Vol. 23, Springer Verlag, Berlin, 1973.

[SB] Stoer, J. and Bulirsch, R., *Introduction to Numerical Analysis.* Springer Verlag, Berlin, 1980

[YS] Yakowitz, S. and Szidarovszky, F., *An Introduction to Numerical Computations.* Macmillan Publ. Comp., New York, 1986.