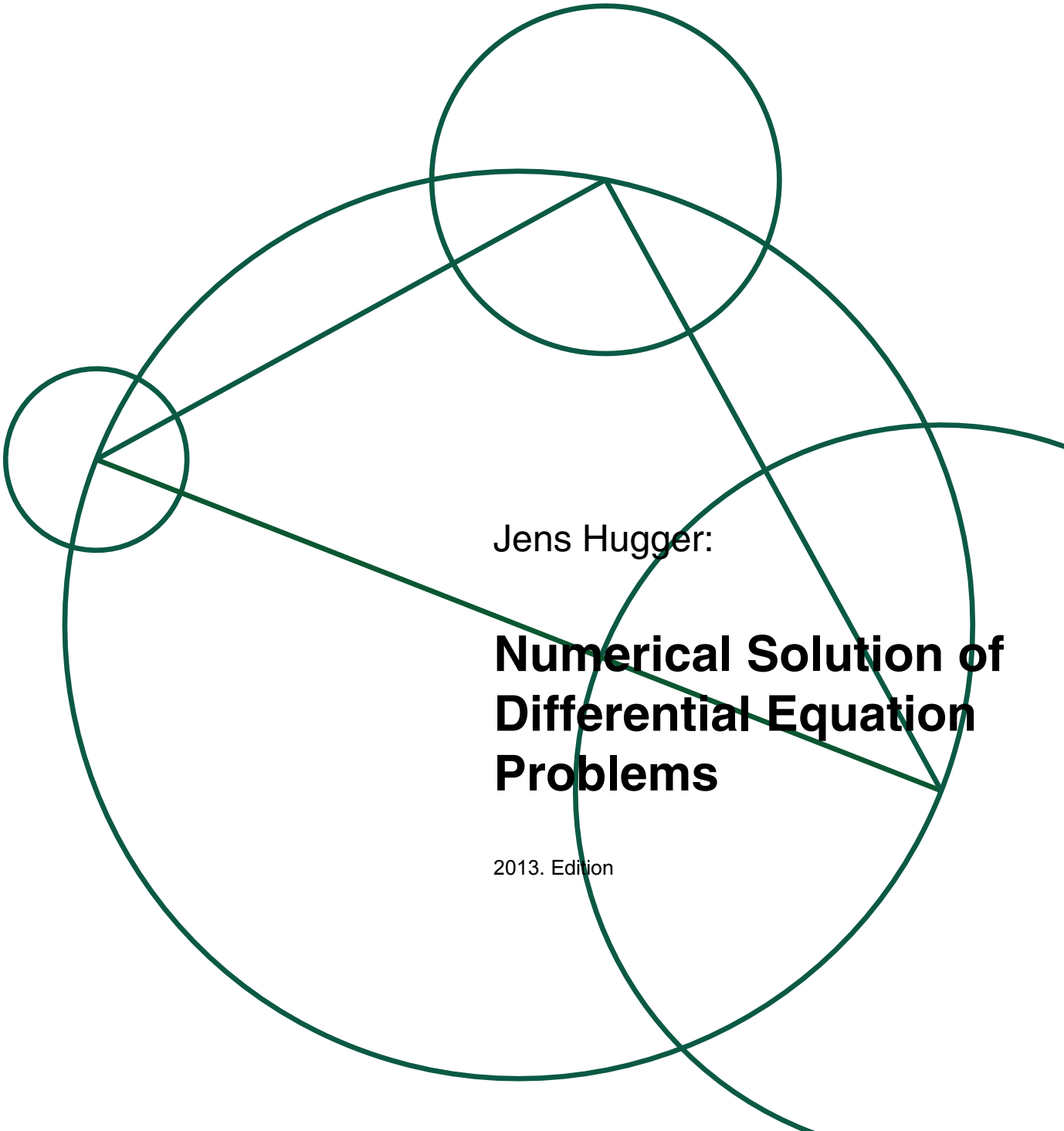
DEPARTMENT OF MATHEMATICAL SCIENCES

UNIVERSITY OF COPENHAGEN



Jens Hugger:

Numerical Solution of Differential Equation Problems

2013. Edition

Numerical solution of differential equation problems[1]

Jens Hugger
Institute for Mathematical Sciences
University of Copenhagen, Denmark
(E-mail: hugger@math.ku.dk)

January 2, 2013

[1]ISBN 87-91927-21-8

Contents

List of exercises

Color codes

All colors are used for graphs and other illustrative purposes as needed. Further
Red is used for defining new words.
Blue is used for stating results.
Green is used for example stuff.
Magenta is used for sections for more advanced readers.

Chapter 1

Differential equation problems

1.1 Introduction and general definitions

A differential equation as for example $u'(x) = \cos(x)$ for $0 < x < 3$ ■ is written as an equation involving some derivative of an unknown function u. There is also a domain of the differential equation (for the example $0 < x < 3$ ■). In reality, a differential equation is then an infinite number of equations, one for each x in the domain. The analytic or exact solution is the functional expression of u or for the example case $u(x) = \sin(x) + c$ where c is an arbitrary constant. This can be verified using Maple and the command `dsolve(diff(u(x),x)=cos(x));` ■ . Because of this non uniqueness which is inherent in differential equations we typically include some additional equations. For our example case, an appropriate additional equation would be $u(1) = 2$ which would allow us to determine c to be $2 - \sin(1)$ and hence recover the unique analytical solution $u(x) = \sin(x) + 2 - \sin(1)$. Here the appropriate Maple command is `dsolve(diff(u(x),x)=cos(x),u(1)=2);` ■ . The differential equation together with the additional equation(s) are denoted a differential equation problem.

Note that for our example, if the value of $u(1)$ is changed slightly, for example from 2 to 1.95 then also the values of u are only changing slightly in the entire domain ■ . This is an example of the continuous dependence on data that we shall require: A well-posed differential equation problem consists of at least one differential equation and at least one additional equation such that the system together have one and only one solution (existence and uniqueness) called the analytic or exact solution to distinguish it from the approximate numerical solutions that we shall consider later on. Further, this analytic solution must depend continuously on the data in the (vague) sense that if the equations are changed slightly then also the solution does

not change too much.

Example 1.1 Consider the differential equation problem

$$\begin{cases} u'(x) = u(x)\tan(x+2) \text{ for } -3 < x < 3 \\ u(-2) = 1 \end{cases} \tag{1.1}$$

The problem may be solved with Maple using the code

```
> ode:=diff(u(x),x)=u(x)*tan(x+2);
> icd:=u(-2)=1;
> sol:=dsolve({ode,icd});
> sol:=simplify(sol,trig);
> eval(u(x),sol);
> uex:=unapply(%,x);
> p1:=plot(uex(x),x=-3..3,y=0..10,color=red):
> p2:=plot(1,x=-3..3,y=0..10,color=blue):
> with(plots):
> display(p1,p2);
```

The solution is found to be $u(x) = |\sec(x+2)|$ where $\sec(x) = 1/\cos(x)$. But sec becomes infinite at $\pm\pi/2$ so the solution is not valid in the points $x = -\pi/2 - 2$ and $x = \pi/2 - 2$. Note that the domain of the differential equation is *not* included in the Maple dsolve command. The result is a function that solves the differential equation for some x-values. It is up to the user to determine which x-values if any should be excluded. ■

★ **For advanced readers 1.2** ★ In example 1.1 we wanted the solution in the interval $]-3,3[$ but we can only use intervals *not* containing the "jump" points. On the other hand, we can only use intervals containing the additional equation point -2. Once we "cross a jump point" all information from the additional equation is lost and we again have arbitrary constants in our solution. Hence (1.1) can be solved only in the interval $]-\pi/2-2, \pi/2-2[$, and (1.1) may be reformulated as

$$\text{Find } u: \; u'(x) = u(x)\tan(x+2) \; \forall x \in]-\pi/2-2, \pi/2-2[, \quad u(-2) = 1. \tag{1.2}$$

★

Exercise 1.3
Use Maple to redo the computations in example 1.1. Use `?diff`, `?dsolve`, `?simplify`, `?unapply`, `?plot` and `?plots[display]` to understand the functionality of these commands. With `ode` and `icd` defined as in the example, try also the command `dsolve[interactive]({ode,icd});`. Write a report giving the solution and a description of the functionality of the commands above. ■

Since this note is about differential equations, it is appropriate to mention some standard notation for these:

When u and u' are the only unknown functions in an equation, we denote it a first order differential equation. Examples are $u'(x)-3u(x) = e^x$, $u'(x)+\sin(u(x)) = \tan(x)$ and $e^{\cos(u'(x))} + 4u(x) = 7x$. The first is denoted a linear first order differential equation since it is linear in both u and u'. The second is denoted a quasilinear first order differential equation since it is linear in u' but not in u (quasilinear could be translated "almost linear" or "linear in the most important part (u')"). The last one is denoted a nonlinear first order differential equation since it is nonlinear in u'. The fact that the example is linear in u does not help us in the solution process and hence is not reflected in the notation ■ . Our general notation covering *any* first order differential equation is $F(x, u(x), u'(x)) = 0$. For the examples above we then have $F(x, u(x), u'(x)) = u'(x) - 3u(x) - e^x$, $F(x, u(x), u'(x)) = u'(x) + \sin(u(x)) - \tan(x)$ and $F(x, u(x), u'(x)) = e^{\cos(u'(x))} + 4u(x) - 7x$ respectively ■ .

If we are looking at more than one differential equation at the same time we denote it a system of differential equations as for example $u_1'(x)-3u_1(x) = e^x$ and $u_2'(x)+\sin(u_1(x)) = \tan(x)$ ■ . In a system of differential equations we then have more than one unknown function. In the example above we have 2, u_1 and u_2 or for short u, where u is the vector with the two components u_1 and u_2. A system as in the example is called a coupled system of differential equations if more than one of the unknown functions appear in at least one of the differential equations in the system. If a system is not coupled then it is an uncoupled system of differential equations. We use the same general notation for systems as we used for single differential equations, i.e. $F(x, u(x), u'(x)) = 0$. Only for systems, both u and F are now vectors. For the example above we then have $F_1(x, u_1(x), u_2(x), u_1'(x), u_2'(x)) = u_1'(x)) - 3u_1(x) - e^x$ and $F_2(x, u_1(x), u_2(x), u_1'(x), u_2'(x)) = u_2'(x) + \sin(u_1(x)) - \tan(x)$ where we have written out the components of the vectors ■ . To separate from systems, a single differential equation is also called a scalar differential equation.

All this notation can be used also for differential equations where both u, u' and u'' appear. Only they are called second order differential equations.

And we can go higher yet to L'th order differential equations, for any positive integer L. We can also generalize to x being a vector: If x is a scalar, i.e. a vector with one component, as above, we talk about ordinary differential equations. If instead x is a vector with at least two components, we talk about partial differential equations. For partial differential equations derivatives appear as partial derivatives as for example $\frac{\partial u(x_1,x_2)}{\partial x_2}$ ■ .

★ **For advanced readers 1.4** ★ For "notation freaks" all the notation gathered above is collected in the following definition which together with the following example 1.6 can be skipped on a first reading. If the reader is *not* a notation freak but prefer to learn by example, the definition (and example 1.6) should be skipped all together.

Definition 1.5 *A System of m Differential Equations in n unknowns, in r dimensions and of order L (n, r being positive integers and L being a non negative integer) is an equation of the form*

$$F(x, u(x), Du(x), D^2u(x), \ldots, D^Lu(x)) = 0, \quad \forall x \in \Omega \tag{1.3}$$

where the domain of definition Ω is a non degenerated subset of $\mathcal{R}^r$, $x \in \mathcal{R}^r$ is an r-vector (The reals can here and below when necessary be exchanged for the complex numbers), $u : \Omega \subseteq \mathcal{R}^r \to \mathcal{R}^n$ is a vector of n unknown functions (to be recovered), $F : \mathcal{R}^{r+n+rn+r^2n+\ldots+r^Ln} \to \mathcal{R}^m$ is a vector of m known functions assumed to depend non trivially on its last variable D^Lu. (D^iu is an array af all i'th (partial) derivatives of all component functions in u and the 0 on the right hand side in (1.3) is in $\mathcal{R}^m$).
For the special case $L = 0$ normally the terminology differential equations of order 0 is replaced by Functional equations.
If F is linear in $u, \ldots, D^Lu$, we call (1.3) a Linear System of Differential Equations. If F is linear in its last variable D^Lu, we call (1.3) a Quasi Linear System of Differential Equations. Otherwise, we call (1.3) a Nonlinear System of Differential Equations.
When $n = m = 1$, also called the Scalar Case, (1.3) is simply called a Differential Equation instead of a system of one differential equation in 1 unknown.
When $r = 1$ (1.3) is called a System of Ordinary Differential Equations (ODE's) and when $r \geq 2$ (1.3) is called a System of Partial Differential Equations (PDE's) in r dimensions (or an ordinary differential equation respectively a partial differential equation for $n = m = 1$).

Example 1.6
We shall here concentrate on the scalar case $n = m = 1$, in $r = 1$ to 4 dimensions and with orders $L = 1$ or 2, i.e. on scalar ordinary and partial differential equations (in up to 4 dimensions) of order 1 or 2, and in particular we focus on linear equations. In one dimension ($r = 1$) and for $L = 1$ this gives the general linear, first order, ordinary differential equation

$$a(x) + b(x)u(x) + c(x)u_x(x) = 0. \tag{1.4}$$

The corresponding general quasi linear, first order, ordinary differential equation takes the form

$$a(x, u(x)) + b(x, u(x))u_x(x) = 0. \tag{1.5}$$

In two dimensions ($r = 2$) and for $L = 2$ this gives the general linear, 2 dimensional, second order, partial differential equation

$$\begin{aligned} &a(x,y) + b(x,y)u(x,y) + c(x,y)u_x(x,y) + d(x,y)u_y(x,y) + \\ &e(x,y)u_{xx}(x,y) + f(x,y)u_{xy}(x,y) + g(x,y)u_{yy}(x,y) = 0. \end{aligned} \tag{1.6}$$

In all cases above (and below) $a, \ldots, g$ are known (nonlinear) functions. Equations of the form of (1.6) are classified into Elliptic, Parabolic and Hyperbolic type depending on whether $e(x,y)g(x,y) - f^2(x,y) > 0, = 0$ or < 0 respectively. Note that the type of an equation generally will vary with the point (x, y). This classification of differential equations are at times extended to cover also other types of equations (in higher dimensions etc.). The classification is based on the notion of Characteristic Curves of which there are 0, 1 and 2 families respectively. For details see for example [1]. ■

Exercise 1.7
Write a 3-5 page essay about characteristic curves and the classification of (1.6). ■

★

As seen above, to get uniqueness of the analytic solution, it is necessary to supplement the differential equations in a differential equation problem by additional equations.

For first order ordinary differential equations (whether scalar or systems) we consider only functional equations (also denoted zero'th order differential equations) like $u(1) = 4$ ■ . If an additional equation involves a point at the boundary of the domain for the differential equation (like end points of an interval ■) then the additional equation is denoted a boundary condition.

If one and only one point is involved in all the additional equations, then the additional equation(s) is/are also denoted initial condition(s). (Note that an additional equation can be both a boundary condition and an initial condition at the same time). To give the same child yet another name, any additional equation which is a zero'th order boundary condition is also called a Dirichlet boundary condition.

For second order ordinary differential equations we consider zero'th and first order differential equations for additional equations like $u(1) = 4$, $u'(1) = 0$ or $u'(1) + 4u(1) = 5$ ■ . Boundary, initial and Dirichlet boundary conditions are introduced the same way as for first order differential equations. A boundary condition consisting only of the value of the first derivative, like $u'(1) = 0$ above ■ , is denoted a Neumann boundary condition. For partial differential equations only the normal derivative, orthogonal to the boundary, may be involved in order to "deserve" the title "Neumann boundary condition". If both the first derivative and the function value participate, like $u'(1) + 4u(1) = 5$ ■ the additional condition is denoted a Robin boundary condition. Again for partial differential equations only the normal derivative, orthogonal to the boundary, and the function value may be involved in order to get this name. As an example of a more exotic boundary condition we could mention $\int_0^1 u(x)dx = 1$ where the additional equation involves "information" from an entire interval ■ .

★ **For advanced readers 1.8** ★ Again we give a complete definition that can be omitted at will. The same goes for the following example 1.10.

Definition 1.9 *Given a system of m differential equations in n unknowns, in r dimensions and of order $L > 0$ according to definition 1.5, additional differential equations, or functional equations (order $L = 0$), involving the same unknown functions u but of order at most $L - 1$ and with domain of definition in Ω', a lower dimensional hyper plane in Ω, like for example the boundary $\partial\Omega$ of Ω, are called Boundary conditions to the system of differential equations. If one of the components of x, say x_1, physically can be related to something "timelike", and if for some real number t_0, only $x \in \Omega|_{x_1=t_0}$ is involved in a condition, then this is often called an Initial Condition.*
If the order of a boundary condition is 0, the condition is functional and is called a Dirichlet Boundary Condition.
If the order of a boundary condition is 1, and only normal derivatives (no tangential derivatives and no functional values) are involved, the condition is called a Neumann Boundary Condition. If instead both normal derivatives and functional values are involved (but still no tangential derivatives), the condition is called a Robin Boundary Condition.

Example 1.10
Even though most of the additional equations met in the literature are boundary conditions of Dirichlet, Neumann or Robin type, we start with an example of a condition which is neither:

$$\int_{\Omega} u(x)dx = 0, \tag{1.7}$$

is an unusual but feasible additional equation for a differential equation problem with domain Ω. It is unusual since the condition involves all of Ω and not just a subset (like a point) of Ω. Thus having considered an exception to the rule, we shall from here on concentrate on the usual types of additional equations:

$$u(x) = d(x), \ \forall x \in \partial\Omega \tag{1.8}$$

is an example of a Dirichlet boundary condition.

$$\frac{\partial u(x)}{\partial n} + a(x)u(x) = b(x), \ \forall x \in \partial\Omega \tag{1.9}$$

where $\frac{\partial}{\partial n}$ stands for the normal derivative (which we shall here always take to be the outward such) is an example of a Neumann boundary condition if $a(x) \equiv 0$ and otherwise a Robin condition, and is feasible whenever the order of the system of differential equations L is at least 2. ■

★

Definition 1.11 *A Differential Equation Problem (DEP) consists of a system of one or more differential equations (see definition 1.5 or the preceding text) and one or more additional equations.*
A differential equation problem where all the additional conditions are boundary or initial conditions respectively (see definition 1.9 or the preceding text) is called a Boundary Value Problem (BVP) or an Initial Value Problem (IVP) respectively. If there are both boundary and *initial conditions, and only such conditions, also the notation Initial Boundary Value Problem (IBVP) is used.*

Exercise 1.12
Write a survey, where you give examples (other than the ones given in this note) of all the new notation, i.e. words, introduced so far in this section. ■

A DEP is usually only of interest if it is well-posed:

Definition 1.13 *A Well-posed Differential equation problem is one for which*

1. *there exists a solution*
2. *the solution is unique*
3. *the solution depends continuously on the data for the problem*

The unique solution is denoted the Analytic or Exact Solution and its functional expression is called its Closed Form Expression.

Below, we shall always assume that our DEP's are well-posed.

The third condition in definition 1.13 above requires a short explanation: The Data for the DEP is for example for the ODE $a(x) + b(x)u(x) + c(x)u'(x) = 0$ (see also (1.4)) the functions a, b and c and for the boundary condition $a + bu(0) + cu'(0) = 0$ similarly the constants a, b and c. In general for $F(x, u(x), \ldots, u^{(L)}(x)) = 0$ (see also (1.3)) the function F constitutes the data. Continuous dependence on data then signifies that if the data of the problem is changed a little then also the solution is changed only a little. There is no unique definition of just how much the solution may change when the data changes. Often terms like linear dependence on data and exponential dependence on data are used to signify that if F is changed to $F + \delta F$ and the solution in the process is changed from u to w, then $\max(w - u) \leq C \max(\delta F)$ or $\max(w - u) \leq C \max(\exp(\delta F))$ respectively. This is a very important property when numerical solution methods are involved, since in this case some change of the original problem (for example due to rounding errors when computing with a computer) will always take place. If we could not be assured that changes in the problem, however small, would lead to small changes in the solutions, then "why compute in the first place?"

Note that it is a very hard problem to establish well-posedness of a general differential equation problem. Normally it is done for one problem or somewhat better for one (small) class of problems at a time. We shall see an example of this in section 1.2 below. For more on the subject of this section see for example [1], [2] and [3].

Below, we shall work with classes of DEP's. A class of DEP's is made up of a DEP where part of the formulation is indeterminate data. A class of DEP's can then be made up for example of all the DEP's with data satisfying certain conditions.

Example 1.14

The class of linear, first order, ordinary differential equations with Dirichlet boundary conditions and 3 times continuously differentiable data consists of

all differential equations on the form $a(x)+b(x)u(x)+c(x)u'(x) = 0$ (see also (1.4)) with boundary conditions of the form $u(x_0) = d$ (see also (1.8)) and where further a, b and c are 3 times continuously differentiable functions, i.e. belong to $\mathcal{C}^3(\Omega)$, x_0 is some point on the boundary of the domain of the differential equation and d is some constant. ■

Exercise 1.15
Write a 2-5 page survey about well-posedness of DEP's and classes of DEP's. ■

1.2 A first order model problem

Among the classes of DEP's where general well-posedness results are known is the quasi linear, scalar, one dimensional, first order, initial value problem (1.10):

Theorem 1.16 *Let I be an open real interval whose closure $\bar{I}$ contains the point x_0. The initial value problem*

$$\text{(1.10)} \quad \textit{Find } u \in \mathcal{C}^1(\bar{I}) : \ u'(x) = f(x, u(x)) \ \forall x \in I, \ u(x_0) = u^*,$$

where $f : S = \bar{I} \times \mathcal{R} \to \mathcal{R}$, $f \in \mathcal{C}(S)$ and f is globally Lipschitz continuous in its second variable, i.e.

$$\text{(1.11)} \quad \exists L > 0 : \ |f(x, u_2) - f(x, u_1)| \leq L|u_2 - u_1| \ \forall x \in I, \ \forall u_1, u_2 \in \mathcal{R}$$

is well-posed.

Example 1.17
Consider the differential equation $u'(x) = \cos(x)$. With the notation of theorem 1.16, $f(x, u(x)) = \cos(x)$ which is obviously Lipschitz continuous in its second variable since $|f(x, u_2) - f(x, u_1)| = |\cos(x) - \cos(x)| = 0 \leq L|u_2 - u_1|$ for any positive L. ■

Example 1.18
Consider the differential equation $u'(x) = \lambda u(x)$. With the notation of theorem 1.16, $f(x, u(x)) = \lambda u(x)$ which is obviously Lipschitz continuous in its second variable since $|f(x, u_2) - f(x, u_1)| = |\lambda u_2 - \lambda u_1| = |\lambda| \cdot |u_2 - u_1| \leq L|u_2 - u_1|$ for $L = \lambda$. ■

Exercise 1.19
Consider the differential equation $u'(x) = 3u(x) + a,\ 0 < x < 5,\ u(0) = b$.
a) Show that this problem is well-posed.
b) Use Maple's `dsolve` command to find the solution to this problem.
Obviously $u(x)$ depends on both a and b. To emphasize this write it instead as $u_{(a,b)}(x)$.
c) Now write up $u_{(A+\delta,B+\kappa)}(x) - u_{(A,B)}(x)$ and verify that for any fixed x, $u_{(a,b)}$ depends linearly on the (change δ and κ in) data a and b.
d) Plot with Maple $u_{(A+\delta,B+\kappa)}(x) - u_{(A,B)}(x)$ for $\delta = \kappa = 0.0001$ in the entire domain of the DEP, i.e. $0 < x < 5$ and note that a small initial perturbation of 0.0001 in the data, giving an initial difference in the solution also of 0.0001 (show this), at $x = 5$ has been magnified to more than 400 (read this of the Maple graph). ■

★ **For advanced readers 1.20** ★ Even for this simple class of problems, complications easily arise as evidenced by the problem considered in example 1.1:

Example 1.21
Returning to example 1.1 and rewriting (1.2) in the form of (1.10), the problem there can be written as

$$\begin{aligned} &\text{Find } u \in \mathcal{C}^1(\,]-\pi/2-2, \pi/2-2[\,) : \\ &u'(x) = u(x)\tan(x+2)\ \forall x \in]-\pi/2-2, \pi/2-2[, \\ &u(-2) = 1. \end{aligned} \tag{1.12}$$

With the notation of theorem 1.16 we have $I =]-\pi/2-2+\epsilon, \pi/2-2-\epsilon[$, $f(x, u(x)) = u(x)\tan(x+2)$, $x_0 = -2$ and $u^* = 1$ where we have reduced the interval by a small non negative ϵ for reasons to be clear immediately: To show well-posedness we start with the Lipschitz continuity taking the form

$$|u_2\tan(x+2) - u_1\tan(x+2)| \le \underbrace{\max_{x\in I} |\tan(x+2)|}_{L} |u_2 - u_1|. \tag{1.13}$$

Obviously, L is finite only for $\epsilon > 0$. Also f is continuous on the slab $S = \bar{I} \times \mathcal{R}$ only for $\epsilon > 0$. On the other hand *any* $\epsilon > 0$ will do. Hence the result from theorem 1.16 is that the problem is well-posed in any closed subset of $]-\pi/2-2, \pi/2-2[$. Since $]-\pi/2-2, \pi/2-2[= \cup_{\epsilon>0}[-\pi/2-2+\epsilon, \pi/2-2-\epsilon]$ we say that the problem is well-posed in the open interval $]-\pi/2-2, \pi/2-2[$. But since $L \to \infty$ as $\epsilon \to 0$ the well-posedness is *not* uniform and we talk about non uniform well-posedness in the open interval $]-\pi/2-2, \pi/2-2[$. ■

For a proof of theorem 1.16, see for example [3]. Here just a comment on the continuous dependence on data which for theorem 1.16 is linear and given the following form:

$$\exists C > 0 : \ |u(t) - z_\epsilon(t)| < C\epsilon, \ \forall t \in I, \ \forall \epsilon \text{ sufficiently small.} \tag{1.14}$$

Here z_ϵ is the solution to the following problem which is a perturbed data version of (1.10):

$$\text{Find } z_\epsilon \in \mathcal{C}^1(I) : \ z'_\epsilon(t) = f(t, z_\epsilon(t)) + \delta_\epsilon(t) \ \forall t \in I, \ z_\epsilon(t_0) = u^* + \delta_{\epsilon,0}. \tag{1.15}$$

Here δ_ϵ and $\delta_{\epsilon,0}$ are selected so that (1.15) has existence and uniqueness of solution for all ϵ sufficiently small and $|\delta_\epsilon(t)| < \epsilon \ \forall t \in I$ and $|\delta_{\epsilon,0}| < \epsilon$.

Note that C in (1.14) must be independent of ϵ, δ_ϵ and $\delta_{\epsilon,0}$.

Note also that not all possible perturbations of data are taken into consideration. For example, no non linear perturbations in the derivative term like $z'_\epsilon + (z'_\epsilon)^2$ is taken into consideration.

Continuous dependence on the data is in this case also denoted Stability in the sense of Liapunov or just Stability. Note that the word "stability" is dangerous, since it is used in many connections to describe only vaguely similar notions. For a proof that Lipschitz continuity in the second variable of f is sufficient to guarantee continuous dependence on the data see for example [4] §11.1 p.471-472.

We shall use (1.10) quite extensively for examples. For all examples we shall let t_0 be the left endpoint of I.

Exercise 1.22
Write up a proof of Liapunov stability for (1.10) given (1.11). ■

★

Chapter 2

Polynomial interpolation

It turns out that there are only very few cases where it is actually possible to derive the closed form expression for the exact solution for a well-posed differential equation problem. Still a theoretical approach very often can give important information about the structure of the solution. We shall not follow that line of thought any further here however, but instead concentrate on Numerical Methods for Approximating the (unknown and sought for) Exact Solution. Such a method is basically a prescription for replacing the DEP by one (or more for nonlinear problems) systems of linear algebraic equations that can be solved on a computer using software written in a standard programming language. All numerical methods involve in one way or another the notion of polynomial interpolation of functions:

★ For advanced readers 2.1 ★ Interpolation is a special case of approximation, so we start with that issue.

2.1 Approximation and error

Example 2.2
When solving a differential equation numerically we most often look for an approximation to the value of the solution function in certain nodal points. Sometimes it is practical to find a function approximating the nodal point values, i.e. having more or less the same values in the nodal points. This process is illustrated in figure 2.1. ■

Example 2.3
If we do not know the primitive of a function f, it can be impossible to compute an integral like $\int_0^3 f(x)\mathrm{d}x$. Instead we can approximate the integrand

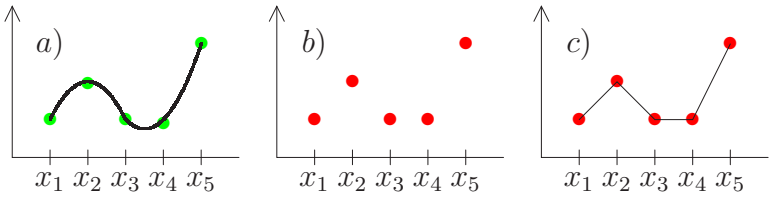


Figure 2.1: Nodal points x_1, x_2, x_3, x_4 and x_5. (a) Exact (unknown) solution to some differential equation and the exact (unknown) nodal point values ●. (b) (Known) approximate nodal point values ●. (c) (Known) linear spline (piecewise linear, continuous interpolant in the nodal points) using approximate nodal point values.

f by a polynomial p (or some other function whose primitive is known) and then approximate the integral by the computable $\int_0^3 p(x)\mathrm{d}x$. For example we could approximate $I = \int_0^3 \sin(x)\mathrm{d}x = 1.989992497$ by $I_n = \int_0^3 p_n(x)\mathrm{d}x$ where p_n is the n'th order Taylor polynomial for $\sin(x)$ around $x = 0$. Here $I_5 = 1.125000000$ while $I_{20} = 1.989992497$. In Maple the commands are

```
evalf(int(sin(x),x=0..3));
evalf(int(convert(taylor(sin(x),x=0,5),polynom),x=0..3));
evalf(int(convert(taylor(sin(x),x=0,20),polynom),x=0..3));
```

■

Relevant problems in connection to the examples above are:

1. How do we find a good approximation to a function, i.e. how do we find the linear spline in example 2.2 and how do we find the polynomial p in example 2.3? (The Taylor polynomial around 0 is not always a good selection as is obvious from exercise 2.4

2. How big is the error, i.e. the difference between the function being approximated and the approximated function in example 2.2 and the correct integral and its approximation in example 2.3, and how do we control it? The control involves many issues as for example the following: Given 2 functions, which one is the best approximation to a third function? Given one function, how do we select another function which is a better approximation to a given function than the first one?

Exercise 2.4
Try to approximate $\int_0^3 \mathrm{erf}(x)\mathrm{d}x$ with $\int_0^3 T^n_{x=0}(\mathrm{erf})(x)\mathrm{d}x$ where $T^n_{x=0}(\mathrm{erf})$ is the n'th order Taylor polynomial for the error function (erf) around $x = 0$: Use Maple to compute the exact result and the approximations for $n = 1, 5, 10, 20$.

The maple command for the error function is `erf`. Plot erf and $T^n_{x=0}(\text{erf})$ together for $n = 1, 5, 10, 20$ and report what is "going wrong". To impress your teacher, you might plot separately for each n and make a movie out af the plots (include some more n-values to get a smoother video. Hint: Use the command `plots[display](...,insequence=true)`). ■

To work towards an answer to the questions above, we now define a general approximation problem in mathematical terms. The approximation problem has 3 parts: We want to approximate something say v with something else say $\tilde{v}$ and then we want to measure the error between v and $\tilde{v}$.

Start considering v: We would like methods that can approximate *many* different functions or data sets. *All* is probably too much to ask so we start selecting a function space $\mathcal{V}$ (normally $\dim\mathcal{V} = \infty$). The idea is, that the function that we intend to approximate belongs to $\mathcal{V}$, i.e. $v \in \mathcal{V}$.

Example 2.5
$\mathcal{V} = \mathcal{C}^\infty([a,b])$, $-\infty < a < b < \infty$. This is the space of infinitely often differentiable functions. Data sets (sets of tuples of nodal points and nodal point values) can be considered point values for a $\mathcal{C}^\infty$ function. ■

Example 2.6
$\mathcal{V} = \mathcal{L}^2((a,b)\times(c,d))$, $-\infty < a < b < \infty$, $-\infty < c < d < \infty$. This is the space of (measurable) functions of 2 variables $f : (x,y) \to \mathcal{R}$ whose square is integrable over the box $(a,b)\times(c,d)$, i.e. $\int_a^b \int_c^d f^2(x,y)\mathrm{d}x\mathrm{d}y < \infty$. ■

Now consider $\tilde{v}$: Select a second function space $\tilde{\mathcal{V}}$. (Normally $\tilde{\mathcal{V}} \subset \mathcal{V}$ and $\dim\tilde{\mathcal{V}} < \infty$). The idea is, that the function that we intend to approximate *with* belongs to $\tilde{\mathcal{V}}$, i.e. $\tilde{v} \in \tilde{\mathcal{V}}$.

Example 2.7
$\tilde{\mathcal{V}} = \mathcal{P}_k([a,b])$, $-\infty < a < b < \infty$. This is the space of polynomials of degree at most k. ■

Example 2.8
$\tilde{\mathcal{V}} = \mathcal{S}^{k,\ell}_\Delta([a,b])$, $-\infty < a < b < \infty$. This is the space of functions in $\mathcal{C}^{\ell-1}([a,b])$ i.e. of global degree of smoothness $\ell - 1$ (for $\ell = 0$ no global smoothness is enforced) that are polynomials of degree at most k i.e. belong to $\mathcal{P}_k$ in each subinterval determined by the subdivision $\Delta = \{\Delta_i\}_{i=1}^n$ of $[a,b]$, i.e. $\exists\{x_i\}_{i=0}^n : a = x_0 < x_1 < \ldots < x_n = b$, $\Delta_i = (x_{i-1}, x_i)$, $i = 1, \ldots, n$. For short we say that $\tilde{\mathcal{V}} = \mathcal{S}^{k,\ell}_\Delta([a,b])$ is the space of functions that are globally $\mathcal{C}^{\ell-1}$ and locally $\mathcal{P}_k$. The most widely used special cases are $\ell = k$ and

$\ell = 1$: $\mathcal{S}_\Delta^{k,k}([a,b])$ is called the space of splines of degree k with respect to the subdivision Δ and $\mathcal{S}_\Delta^{k,1}([a,b])$ is called the space of piecewise Lagrange interpolants of degree k with respect to the subdivision Δ. For examples of $\mathcal{S}_\Delta^{1,1}([a,b])$, $\mathcal{S}_\Delta^{2,2}([a,b])$ and $\mathcal{S}_\Delta^{3,3}([a,b])$ see figure 2.2. The spaces $\mathcal{S}_\Delta^{k,1}([a,b])$ and $\mathcal{S}_\Delta^{k,k}([a,b])$ will be discussed in details in section 2.3 below. ■

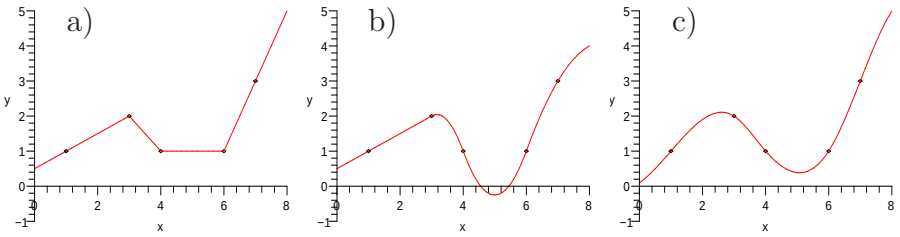


Figure 2.2: (a) Linear spline, (b) Quadratic spline and (c) Cubic spline all using the same set of nodal points and nodal point values.

Finally consider the error between v and $\tilde{v}$: We saw in example 2.3 that the error may be something quite different from just a normal difference between v and $\tilde{v}$. Hence we need some more general way to measure the distance between two functions $v \in \mathcal{V}$ and $\tilde{v} \in \tilde{\mathcal{V}}$. The mathematical notion that we need is a distance measure $d : \mathcal{V} \times \tilde{\mathcal{V}} \to \mathcal{R}_{0,+} := \{r \in \mathcal{R} : r \geq 0\}$. d is a function that for any choice of v in $\mathcal{V}$ and any choice of $\tilde{v}$ in $\tilde{\mathcal{V}}$ returns a real, non negative number $d(v, \tilde{v})$ which is our error. Note that we do not allow an error which is simply $v - \tilde{v}$ which is a function and not a real number. If we used such an error it would be hard to compare two errors $v - \tilde{v}_1$ and $v - \tilde{v}_2$. With our choice, the error is a real number, and we can easily find out which of $\tilde{v}_1$ and $\tilde{v}_2$ is the best approximation to v. It is the one with the smallest $d(v, \tilde{v})$.

Example 2.9
$d(v, \tilde{v}) = |v(0) - \tilde{v}(0)| + |v(1) - \tilde{v}(1)|$. In this case we emphasize only the difference in the two (nodal) points 0 and 1. We do not care what so ever about how the functions behave in other points. ■

Example 2.10
$d(v, \tilde{v}) = \|v - \tilde{v}\|$ where $\|\cdot\|$ is a norm as for example the $\mathcal{L}^\infty$ norm (pronounced L infinity) $\|v - \tilde{v}\|_\infty = \max_{0 \leq x \leq 1} |v(x) - \tilde{v}(x)|$ or the $\mathcal{L}^2$ norm (pronounced L two) $\|v - \tilde{v}\|_2 = \sqrt{\int_0^1 (v - \tilde{v})^2(x) \mathrm{d}x}$. In this case we emphasize the difference in all points in the interval $[0, 1]$ but in different ways. ■

Definition 2.11 *A Norm $\|\cdot\|$ on a vector space $\mathcal{V}$ is a function $\mathcal{V} \to \mathcal{R}_{0,+}$. Instead of the standard notation, where we would denote the function by say n and the function value in say v by $n(v)$, we denote the function $\|\cdot\|$ and the function value in v by $\|v\|$.*
For a function $\mathcal{V} \to \mathcal{R}_{0,+}$ to be a norm it must *satisfy the following conditions:*

$$
\begin{aligned}
& v \in \mathcal{V} \setminus \{0\} \Rightarrow \|v\| > 0 \\
& \lambda \in \mathcal{R} \text{ and } v \in \mathcal{V} \Rightarrow \|\lambda v\| = |\lambda| \cdot \|v\| \\
& v, w \in \mathcal{V} \Rightarrow \|v + w\| \leq \|v\| + \|w\| \quad \textit{(triangle inequality)}
\end{aligned} \tag{2.1}
$$

For $\mathcal{V} = \mathcal{R}^n$ we define (for $v = (v_1, \ldots, v_n)^T \in \mathcal{R}^n$)

$$
\|v\|_{\ell^1} = \sum_{j=1}^{n} |v_j|, \quad \|v\|_{\ell^2} = \sqrt{\sum_{j=1}^{n} v_j^2}, \quad \|v\|_{\ell^\infty} = \max_{j=1\ldots n} |v_j|. \tag{2.2}
$$

For $\mathcal{V}$ being a vector space of functions defined, measurable, integrable, square integrable and continuous on a closed interval I, i.e. all of the following definitions give finite numbers as results, then

$$
\|v\|_{\mathcal{L}^1} = \int_I |v(s)|ds, \quad \|v\|_{\mathcal{L}^2} = \sqrt{\int_I v^2(s)ds}, \quad \|v\|_{\mathcal{L}^\infty} = \max_{s \in I} |v(s)|. \tag{2.3}
$$

It can be proven that all of the functions in (2.2) and (2.3) are indeed norms. We shall do so only for the $\|\cdot\|_{\ell^1}$ case checking the 3 conditions in (2.1) in order of appearance:

1. If $v \neq 0$ then at least one of the components, say v_k, is non zero. But then $|v_k| > 0$ and hence $\|v\|_{\ell^1} = \sum_{j=1}^{n} |v_j| > |v_k| > 0$.
2. Obviously $\|\lambda v\|_{\ell^1} = \sum_{j=1}^{n} |\lambda v_j| = |\lambda| \sum_{j=1}^{n} |v_j| = |\lambda| \|v\|_{\ell^1}$.
3. $\|v+w\|_{\ell^1} = \sum_{j=1}^{n} |v_j+w_j| \leq \sum_{j=1}^{n} (|v_j|+|w_j|) = \sum_{j=1}^{n} |v_j| + \sum_{j=1}^{n} |w_j| = \|v\|_{\ell^1} + \|w\|_{\ell^1}$.

Example 2.12 Let us find the distance measured in the norms defined above between the functions $f : x \to x^2$ and $g : x \to x^3$ on the interval from 0 to 5. For the ℓ^1- ℓ^2- and ℓ^∞-norms we define $v = (v(0), v(1), v(2), v(3), v(4), v(5))^T$ where $v = f - g$, i.e. the vector of nodal point values of the difference between the two functions in the nodal points $0, 1, 2, 3, 4, 5$. This is easily computed with Maple using the commands

```
f:=x->x^2; g:=x->x^3; h:=x->abs(f(x)-g(x));
for i from 0 to 5 do x[i]:=i end do; i:=evaln(i);
hseq:=seq(abs(f(x[i])-g(x[i])),i=0..5);
l1:=sum(hseq[i],i=1..6);
l2:=sqrt(sum((hseq[i])^2,i=1..6));
linf:=max(hseq);
L1:=evalf(int(h(x),x=0..5));
L2:=sqrt(evalf(int(h(x)^2,x=0..5)));
Linf:=maximize(h(x),x=0..5);
```

and gives the results

$$\|f-g\|_{\ell^1} = 170, \quad \|f-g\|_{\ell^2} = 12644, \quad \|f-g\|_{\ell^\infty} = 100,$$
$$\|f-g\|_{L^1} = 114.75, \quad \|f-g\|_{L^2} = 6577.380952, \quad \|f-g\|_{L^\infty} = 100.$$

Note that the norm of a function is not at all uniquely defined. The size depends heavily on the type of norm used. ■

Definition 2.13 *Given function spaces $\mathcal{V}$ and $\tilde{\mathcal{V}}$, a distance measure $d : \mathcal{V} \times \tilde{\mathcal{V}} \to \mathcal{R}_{0,+}$ and $v \in \mathcal{V}$, the solution $\tilde{v}$ to the best approximation problem*

$$\text{find } \tilde{v} \in \tilde{\mathcal{V}} : \ d(v,\tilde{v}) = \inf_{w \in \tilde{\mathcal{V}}} d(v,w) \tag{2.4}$$

is called the best approximation in $\tilde{\mathcal{V}}$ to $v \in \mathcal{V}$ with the distance measure d. $e = v - \tilde{v}$ is called the best error function. $|e| = d(v,\tilde{v})$ is called the minimal error.

Best approximation problems with $d(v,\tilde{v}) = \|v-\tilde{v}\|_{\ell^1}$ are called interpolation problems (see a complete definition below) and if $d(v,\tilde{v}) = \|v-\tilde{v}\|_{\ell^2}$ they are called least squares problems. Maple can handle these and many other best approximation problems through the `CurveFitting` package, containing among other the commands `PolynomialInterpolation` and `LeastSquares`. For example

```
xydata:=[[0,0],[1,1],[2,5],[3,2],[4,7],[5,0]];
with(CurveFitting):
q:=LeastSquares(xydata,x,curve=a+b*x+c*x^2+d*x^3);
q1:=plot(q,x=0..5,color=red):
p0:=plots[pointplot](xydata,symbol=diamond,color=blue):
plots[display](p0,q1);
```

Here we approximate the data set with a cubic curve which is termed Cubic Least Squares.

Example 2.14
$\mathcal{V} = \mathcal{C}^0([0,1])$, $\tilde{\mathcal{V}} = \mathcal{P}_1([0,1])$, $d(v,\tilde{v}) = |v(0) - \tilde{v}(0)| + |v(1) - \tilde{v}(1)|$.
The unique solution $\tilde{v}$ to the best approximation problem with this selection of $\mathcal{V}$, $\tilde{\mathcal{V}}$ and d is called the linear interpolant to v in 0 and 1. Note that the search for a best approximation from (2.4) is simplified by the fact that $\inf_{w\in\tilde{\mathcal{V}}} d(v,w) = 0$ so that (2.4) is replaced by the equation system

$$\text{find } \tilde{v} \in \tilde{\mathcal{V}} : \ d(v,\tilde{v}) = 0. \tag{2.5}$$

This is what distinguishes the interpolants in the class of approximants. ■

Definition 2.15 *Given function spaces $\mathcal{V}$ and $\tilde{\mathcal{V}}$, and a distance measure $d : \mathcal{V} \times \tilde{\mathcal{V}} \to \mathcal{R}_{0,+}$ and $v \in \mathcal{V}$, such that $\inf_{w\in\tilde{\mathcal{V}}} d(v,w) = 0$ has a unique solution, the solution $\tilde{v}$ to the interpolation problem*

$$\textit{find } \tilde{v} \in \tilde{\mathcal{V}} : \ d(v,\tilde{v}) = 0 \tag{2.6}$$

is called the interpolant in $\tilde{\mathcal{V}}$ to $v \in \mathcal{V}$ with the distance measure d.
$e = v - \tilde{v}$ is called the interpolation error function. The interpolation error $|e| = d(v,\tilde{v})$ is zero.

Exercise 2.16
a) Describe in words, the properties of the linear interpolant to a function v in 0 and 1.
b) Find the expression for the linear interpolant to the function $x \to x^2$ in 0 and 1.
c) The quadratic interpolant to a function v in 0, $\frac{1}{2}$ and 1 can be described as the unique solution to the approximation problem given by the selection $\mathcal{V} = \mathcal{C}^\infty([0,1])$, $\tilde{\mathcal{V}} = \mathcal{P}_2([0,1])$, $d(v,\tilde{v}) = |v(0)-\tilde{v}(0)|+|v(\frac{1}{2})-\tilde{v}(\frac{1}{2})|+|v(1)-\tilde{v}(1)|$. Do (a) and (b) for the quadratic interpolant. ■

There is a large amount of literature on best approximation theory. We shall not go into details here but refer to [5] for an introductory treatment. In practice, it is often neither possible nor really required to find the best approximation. Instead the real problem can be formulated as follows: Given $\mathcal{V}$ and a positive, real valued tolerance T, find $\tilde{\mathcal{V}}$ so that $|e| \leq T$. Normally, $\tilde{\mathcal{V}}$ is restricted to be one of a sequence of function spaces $\{\tilde{\mathcal{V}}_k\}_{k=0}^{\infty}$ with a corresponding sequence of best approximations $\{\tilde{v}_k\}_{k=0}^{\infty}$ such that

- $\tilde{\mathcal{V}}_0 \subset \tilde{\mathcal{V}}_1 \subset \ldots \subset \mathcal{V}$.

- $0 < \dim\tilde{\mathcal{V}}_0 < \dim\tilde{\mathcal{V}}_1 < \ldots < \infty$. The sequence $\{\tilde{\mathcal{V}}_k\}_{k=0}^{\infty}$ is said to be increasing.
- $\lim_{k\to\infty} |e_k| = \lim_{k\to\infty} d(v, \tilde{v}_k) = 0,\ \forall v \in \mathcal{V}$. The sequence $\{\tilde{\mathcal{V}}_k\}_{k=0}^{\infty}$ is said to be dense in $\mathcal{V}$.

Definition 2.17 *The tolerance T approximation problem:*
Given a function space $\mathcal{V}$, an increasing sequence of function spaces $\{\tilde{\mathcal{V}}_k\}_{k=0}^{\infty}$ which is dense in $\mathcal{V}$, a distance measure $d : \mathcal{V} \times \tilde{\mathcal{V}}_k \to \mathcal{R}_{0,+}$ defined for all $k = 1, 2, \ldots$ and a tolerance $T \in \mathcal{R}_+$, find the smallest integer k such that $|e_k| = d(v, \tilde{v}_k) \leq T\ \forall v \in \mathcal{V}$, where $\{\tilde{v}_k\}_{k=0}^{\infty}$ is the sequence of best approximations.

Again, finding the exact smallest integer k may be hard or impossible, so often the goal is simply set at finding just one k (but not necessarily the smallest such) so that $|e_k| \leq T$. Of course, the larger the k the more computational work, since the sequence of spaces $\{\tilde{\mathcal{V}}_k\}_{k=0}^{\infty}$ is increasing with k. An issue of general interest is thus how fast the error goes towards zero as k increases in the tolerance T approximation problem. The speed is measured using the following $\mathcal{O}$-notation: ★

Definition 2.18 *$\mathcal{O}$-notation and order of convergence for sequences:*
Given two infinite sequences of numbers $\{a_k\}_{k=1}^{\infty}$ and $\{b_k\}_{k=1}^{\infty}$ we write

$$a_k = \mathcal{O}_{k\to\infty}(b_k) \tag{2.7}$$

and say that a_k is "big oh" of b_k if there exists constants k_0 and C such that $|a_k| \leq C|b_k|$ when $k \geq k_0$.
In mathematical analysis $\mathcal{O}_{k\to\infty}(\{b_k\}_{k=1}^{\infty})$ is defined as the collection of all sequences $\{a_k\}_{k=1}^{\infty}$ for which there exists constants k_0 and C such that $|a_k| \leq C|b_k|$ when $k \geq k_0$. Hence the following more elaborate notation may be used: $\{a_k\}_{k=1}^{\infty} \in \mathcal{O}_{k\to\infty}(\{b_k\}_{k=1}^{\infty})$. In these notes however, we shall stick to the first (and simpler) notation.
If $\lim_{k\to\infty} a_k = \lim_{k\to\infty} b_k = 0$ and $a_k = \mathcal{O}_{k\to\infty}(b_k^q)$ then we say that $\{a_k\}_{k=1}^{\infty}$ is convergent of order q with respect to $\{b_k\}_{k=1}^{\infty}$.

Example 2.19
Let $a_k = \frac{k+2}{k^2}$ and $b_k = \frac{1}{k}$, $k \geq 1$. Clearly $a_k = \mathcal{O}_{k\to\infty}(b_k)$ since

$$\left|\frac{a_k}{b_k}\right| = \left|\frac{\frac{k+2}{k^2}}{\frac{1}{k}}\right| = \frac{k+2}{k} = 1 + \frac{2}{k} \leq 3\ \forall k \geq 1. \tag{2.8}$$

Likewise $\frac{1}{k} + e^{-k} = \mathcal{O}_{k\to\infty}(\frac{1}{k})$ since

$$\left|\frac{\frac{1}{k} + e^{-k}}{\frac{1}{k}}\right| = 1 + ke^{-k} \leq 2 \ \forall k \geq 1. \tag{2.9}$$

The last inequality can be verified with Maple using the Maple command `plot(k*exp(-k),k=0..10);` i.e. e^{-k} goes to zero faster than k goes to infinity. It is a special case of the more general result, that an exponential beats any polynomial, i.e. e^{-k} goes to zero faster than k^p goes to infinity for any positive number p. ■

Definition 2.20 *$\mathcal{O}$-notation and order of convergence for functions:*
Given two functions f and g defined on the same domain Ω (open set) containing the point x_0 we write

$$f = \mathcal{O}_{x\to x_0}(g) \tag{2.10}$$

and say that f is "big oh" of g in x_0 if there exists constants δ and C such that $|f(x)| \leq C|g(x)|$ when $|x - x_0| \leq \delta$ if $|x_0| < \infty$ or $|x| \geq \delta$ if $x_0 = \infty$.
A more precise but also more elaborate notation would be $f \in \mathcal{O}_{x\to x_0}(g)$ defining $\mathcal{O}_{x\to x_0}(g)$ as the collection of all functions f for which there exists constants δ and C such that $|f(x)| \leq C|g(x)|$ when $|x - x_0| \leq \delta$ if $|x_0| < \infty$ or $|x| \geq \delta$ if $x_0 = \infty$. In these notes however, we shall stick to the first (and simpler) notation.
If $\lim_{x\to x_0} f(x) = \lim_{x\to x_0} g(x) = 0$ and $f = \mathcal{O}_{x\to x_0}(g^q)$ then we say that f is convergent of order q with respect to g.

Example 2.21
$\sqrt{x^2+3} = \mathcal{O}_{x\to\infty}(x)$ since

$$\left|\frac{\sqrt{x^2+3}}{x}\right| = \sqrt{1 + \frac{3}{x^2}} \leq 2 \text{ whenever } |x| \geq 1. \tag{2.11}$$

Likewise $\sin x - x + \frac{x^3}{6} = \mathcal{O}_{x\to 0}(x^5)$ which can be seen from the Taylor expansion of $\sin x$ around $x = 0$ with Lagrange remainder: $\sin x = x - \frac{x^3}{6} + \frac{\cos\xi}{120}x^5$ for some $\xi \in]0, x[$ so that $|\sin x - x + \frac{x^3}{6}| \leq \frac{|x|^5}{120}$. This result will normally for clarity be written in the form $\sin x = x - \frac{x^3}{6} + \mathcal{O}_{x\to 0}(x^5)$. Since $\lim_{x\to 0}\left(\sin x - x + \frac{x^3}{6}\right) = \lim_{x\to 0} x = 0$ we say that $\sin x - x + \frac{x^3}{6}$ is convergent of order 5 with respect to x. ■

Exercise 2.22
a) Show by Taylor expansion that $e^x = 1 + x + \mathcal{O}_{x\to 0}(x^2)$.
b) Show also that $\frac{1}{k} + \frac{k}{1+k} - 1 = \mathcal{O}_{k\to\infty}(\frac{1}{k^2})$. *Hint:* Replace k by $1/x$ and use Taylor expansion. ■

Other than Taylor expansion the order of convergence can also be found in another (graphical) way. For example $\sin x - x + \frac{x^3}{6} = \mathcal{O}_{x\to 0}(x^5)$ means that $\sin x - x + \frac{x^3}{6} = c_1x^5 + c_2x^6 + c_3x^7 + \ldots$ for some constants c_1, c_2, c_3, ... As x approaches 0 the term c_1x^5 starts dominating no matter how big c_2, c_3 and so on are because the terms with higher orders of x go towards zero faster than x^5. Hence asymptotically, as $x \to 0$ we can count on $\sin x - x + \frac{x^3}{6} = c_1x^5$ as an increasingly better and better approximation. Taking the logarithm on both sides we have $\ln\left(\sin x - x + \frac{x^3}{6}\right) = \ln c_1 + 5\ln x$. Plotting this, i.e. plotting $\sin x - x + \frac{x^3}{6}$ against x in a double logarithmic coordinate system, we get a straight line with slope 5. In general, if f is convergent of order q with respect to g, we plot f against g in double logarithmic scale (in Maple there is a `loglogplot` command) and recover q as the slope of the resulting line. Because of the approximations we are making (in throwing away the terms $c_2x^6 + c_3x^7 + \ldots$) the line appears to show up only for sufficiently small values of x. The x interval where the curve is considered straight, is called the asymptotic tail. In reality, the curve only converges to be exactly straight as $x \to 0$. Note, that it is generally not possible up front to know how big the asymptotic tail is. This also depends on the tolerance the user applies in determining whether the curve is a straight line. For an example see figure 2.3 below, where f is denoted $e(h)$ and g is denoted h and the order of convergence is $q = 3$.

★ **For advanced readers 2.23** ★ Now we are prepared to define the speed of convergence of the tolerance T approximation problem.

Definition 2.24 *The convergence problem:*
Given a function space $\mathcal{V}$, an increasing sequence of function spaces $\{\tilde{\mathcal{V}}_k\}_{k=0}^{\infty}$ which is dense in $\mathcal{V}$ and a distance measure $d : \mathcal{V} \times \tilde{\mathcal{V}}_k \to \mathcal{R}_{0,+}$ defined for all $k = 1, 2, \ldots$, find the largest $q \in \mathcal{R}$ such that $|e_k| = \mathcal{O}_{k\to\infty}(h_k^q)$, (i.e. $\exists x_0 \in \mathcal{R} : \frac{|e_k|}{h_k^q} \leq x_0 \forall k = 1, 2, \ldots$ sufficiently large), $\forall v \in \mathcal{V}$, where h_k is a "typical parameter" for the $\tilde{\mathcal{V}}_k$ problem and $h_k \to 0$ as $k \to \infty$.
q is called the (theoretical) order of convergence of the tolerance T approximation problem.

Example 2.25
Typical selections for h_k are $h_k = \frac{1}{k}$ or $h_k = \frac{1}{\dim \tilde{\mathcal{V}}_k}$, but also other values may be reasonable under certain circumstances. In any case, h_k is selected by the user so that it "makes sense" for the given situation. ■

Even though the error function e_k above is only defined in terms of h_k for the discrete sequence $k = 1, 2, \ldots$, in reality e_k often can be defined for any real value of h_k. Hence it makes sense to drop the index k and consider e a function of h. Assuming that e is smooth enough to have a Taylor expansion of any order that we require, the condition $|e_k| = \mathcal{O}_{k\to\infty}(h_k^q)$ can be changed into $|e(h)| = \mathcal{O}_{h\to 0}(h^q) \Leftrightarrow e(h) = x_0 h^q + x_1 h^{q+1} + \ldots$ for some sequence of (unknown) constants $x_0, x_1, \ldots$ where $x_0 \neq 0$. Note that no matter how the constants are valued, the terms with lower exponent will dominate the ones with higher exponent in absolute value for sufficiently small values of h, as long as the coefficient of the term with lower exponent is not zero. For sufficiently small values of h, $e(h)$ can then be approximated well with $x_0 h^q + x_1 h^{q+1} + \ldots + x_n h^{q+n}$, and better the larger n. To compute q from this expression, we need $n+2$ equations since apart from q there are $n+1$ other unknowns $x_0, \ldots, x_n$. To get these $n+2$ equations we use computed values of $e(h_1), \ldots, e(h_{n+2})$ say $\tilde{e}_{k_1}, \ldots, \tilde{e}_{k_{n+2}}$ and solve the (approximating) equalities $\tilde{e}_{k_1} = x_0 h_{k_1}^q + x_1 h_{k_1}^{q+1} + \ldots + x_n h_{k_1}^{q+n}, \ldots, \tilde{e}_{k_{n+2}} = x_0 h_{k_{n+2}}^q + x_1 h_{k_{n+2}}^{q+1} + \ldots + x_n h_{k_{n+2}}^{q+n}$. For example for $n = 0$ we need to solve the system $\tilde{e}_{k_1} = x_0 h_{k_1}^q$ and $\tilde{e}_{k_2} = x_0 h_{k_2}^q$ where $\tilde{e}_{k_1}$, $\tilde{e}_{k_2}$, h_{k_1} and h_{k_2} are known values while x_0 and q must be recovered. q is recovered by dividing the first equation by the second and taking logarithms. We get $q = \ln(\tilde{e}_{k_1}/\tilde{e}_{k_2})/\ln(h_{k_1}/h_{k_2})$. Note that typically, we are not interested in a full solution, but only on the extraction of the value of q from these equations. Hence we eliminate $x_0, \ldots, x_n$ and solve for q. The q that we recover is of course not the exact theoretical order of convergence since several approximations are involved. First of all, the computation of the values $\tilde{e}(h_{k_i})$ for $i = 1, \ldots, n+2$ may involve imprecisions and secondly the "cut off" of the Taylor expansion involves an error which is bigger the smaller n we choose. To distinguish the recovered q that we shall denote $\tilde{q}_{k_1,\ldots,k_{n+2}}$ from the theoretical order of convergence, we shall denote it the practical order of convergence. The $k_1, \ldots, k_{n+2}$ refer to the $\tilde{\mathcal{V}}_{k_1}, \ldots, \tilde{\mathcal{V}}_{k_{n+2}}$ used in the computations. Hopefully $\tilde{q}_{k_1,\ldots,k_{n+2}} \simeq q$, but it is wrong to claim that q has been found, if it is really some $\tilde{q}_{k_1,\ldots,k_{n+2}}$ that has been recovered. In short, q comes from theoretical investigations, while $\tilde{q}_{k_1,\ldots,k_{n+2}}$ comes from computations resulting in graphs like the one shown in figure 2.3. Note how different the graphs in figure 2.3 look. For the human eye, it is much easier to spot a polynomial dependence with double logarithmic axes than with linear axes. Also note how much easier it is to spot the asymptotic tail and

estimate the order of convergence from the double logarithmic plot, even with the significant reduction in the extension of the linear axes.

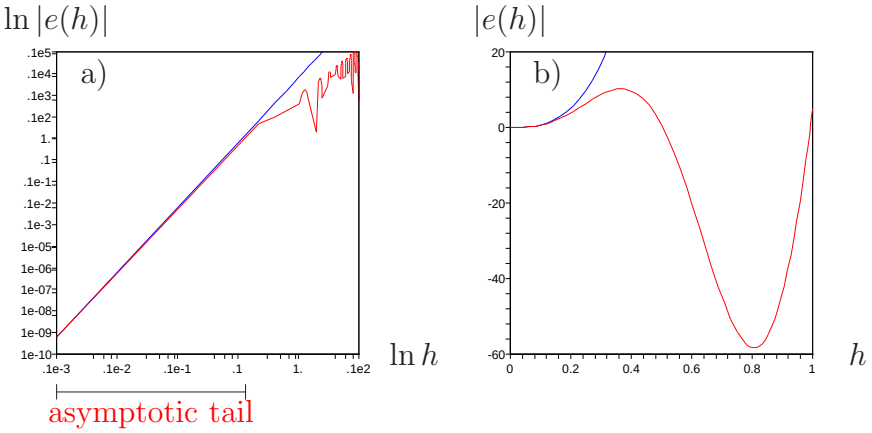


Figure 2.3: Error function $e(h)$ with cubic order of convergence (red, bottommost curve) and for reference the function $(5 + 200\pi)h^3$ (blue, topmost curve) with (a) double logarithmic axes, (b) Linear axes.

When plotting the computed error $|\tilde{e}_k|$ as a function of h_k in double logarithmic coordinates, so that a function $x_0 h^q$ shows up as a straight line with slope q, normally the straight line appears to show up only for sufficiently small values of $\max_{i=1,\ldots,n} h_{k_i}$. The h interval where the curve is considered straight, is called the asymptotic tail of the method. In reality, the curve only converges to be exactly straight as $\max_{i=1,\ldots,n+2} h_{k_i} \to 0$. This means that $\tilde{q}_{k_1,\ldots,k_{n+2}} \to q$ only as we consider smaller and smaller values of $\max_{i=1,\ldots,n+2} h_{k_i}$ for the computation of $\tilde{q}_{k_1,\ldots,k_{n+2}}$. Note, that it is generally not possible up front to know how big the asymptotic tail is. This also depends on the tolerance the user applies in determining whether the curve is a straight line.

Exercise 2.26
Recreate the plots in figure 2.3 using Maple and the commands `plot` and `loglogplot` from the `plots` package.
Hint: The construction `p1:=loglogplot(...): p2:=loglogplot(...): display(p1,p2,...);` may be used. The error function is given by $e_k = 5h_k^3 + 100h_k^2 \sin(2\pi h_k)$. Show that the theoretical order of convergence is $q = 3$ and derive the formula to find the practical order of convergence $\tilde{q}_{h_1,h_2}$ ($n = 0$ above) and the corresponding constant x_0. Find the value of $\tilde{q}_{h_1,h_2}$ for $h_1 = 10^{-1}$ and $h_2 = 2h_1$. Do the same for $h_1 = 10^{-i}$ for $i = 2, 3, \ldots, 10$.

For the practical computations needed for $\tilde{q}_{h_1,h_2}$ use the exact values $e(h_1)$ and $e(h_2)$. This constitutes an ideal situation, where the error is coming only from the fact that $\tilde{q}_{h_1,h_2}$ is measuring the order not at $h = 0$ but somewhere depending on h_1 and h_2, or to put it in other terms: It comes from the fact that we are only using the first and leading term in the taylor expansion of the error. ■

★

2.2 The well-determined polynomial interpolation problem

We shall now study the most common examples of interpolation problems as defined in generality in definition 2.15.

Theorem 2.27 *Let $x_0, x_1, \ldots, x_n$ be distinct real numbers (nodal points). Then for arbitrary real values $y_0, y_1, \ldots, y_n$ (nodal point values) there exists a unique polynomial p_n of degree at most n ($\exists! p_n \in \mathcal{P}_n$) such that*

$$p_n(x_i) = y_i, \text{ for } i = 0, 1, \ldots, n.$$

p_n is called the (Lagrange) interpolating polynomial for the data set $\{(x_0, y_0), \ldots, (x_n, y_n)\}$.

Proof:
Uniqueness: Assume that there exists two interpolating polynomials, p_n and q_n.
Then $(p_n - q_n)(x_i) = 0$, for $i = 0, \ldots, n$, and $\deg(p_n - q_n) \leq n$ so $p_n - q_n \equiv 0$. (A nonzero polynomial of degree at most n may have at most n roots).
Existence: Induction over n:
$n = 0$: Select $p_0(x) = y_0$. Then $p_0 \in \mathcal{P}_0$ and $p_0(x_0) = y_0$.
Assume existence for $n - 1$, show existence for n: Select $p_n = p_{n-1} + c_n(x - x_0) \cdot \ldots \cdot (x - x_{n-1})$.
Then $p_n \in \mathcal{P}_n$ and $p_n(x_i) = p_{n-1}(x_i) = y_i$ for $i = 0, \ldots, n-1$ (by the assumtion on p_{n-1}). $p_n(x_n) = p_{n-1}(x_n) + c_n \prod_{i=0}^{n-1}(x_n - x_i) = y_n$ as long as we select $c_n = (y_n - p_{n-1}(x_n))/\prod_{i=0}^{n-1}(x_n - x_i)$. (Note that the x_i's are distinct, so that we are not dividing by zero). ■

Example 2.28 Take the nodal points $0, 1, 2, 3, 4, 5$ and the nodal point values $0, 1, 5, 2, 7, 0$ and pass a polynomial of degree at most 5 through them. This is easily done with Maple:

```
with(CurveFitting);
xydata:=[[0,0],[1,1],[2,5],[3,2],[4,7],[5,0]];
p:=PolynomialInterpolation(xydata,x);
```

Note that the CurveFitting package contains many interesting approximation commands. Apart from Lagrange interpolation, there are least squares, splines and more. To plot the data set and the interpolating polynomial

```
with(plots);
p1:=plot(p,x=0..5,color=red):
p0:=pointplot(xydata,symbol=diamond,color=blue):
display(p0,p1);
```

gives

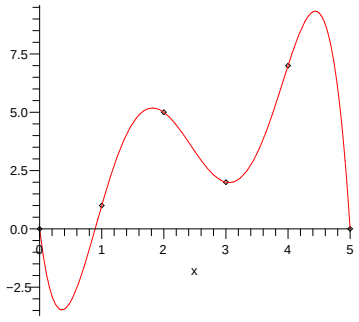


Figure 2.4: Lagrange interpolant $-\frac{1}{2}x^5 + \frac{145}{24}x^4 - \frac{305}{12}x^3 + \frac{1031}{24}x^2 - \frac{265}{12}x$ of a data set $\{(0,0),(1,1),(2,5),(3,2),(4,7),(5,0)\}$.

An option to the `PolynomialInterpolation` command is the form of the interpolating polynomial. The command then takes the form

```
PolynomialInterpolation(xydata,x,form=option);
```

where `option` is any of `Newton`, `Lagrange`, `monomial` or `power`. The significance of this selection will be clarified in the following subsections. ■

Exercise 2.29
Do all the Maple computations of example 2.28 including trying all 4 options to the `PolynomialInterpolation` command. What differences do you note using the 4 options. ■

Exercise 2.30
Extend example 2.14 to give the best approximation formulation of the interpolation problem implied by theorem 2.27, i.e. find the function spaces and distance measure in definition 2.13 for the best approximation problem in the case of theorem 2.27, i.e. for the degree n interpolating polynomial. ■

Example 2.31
Take the nodal points $x_0, \ldots, x_7$ distinct as required in theorem 2.27. Say all nodal point values are identical, i.e. $y_0 = \ldots = y_7 = 7$. Then $p_7(x) \equiv 7 \in \mathcal{P}_0 \subset \mathcal{P}_7$. In general $\exists! p_n \in \mathcal{P}_n$, but p_n may belong to $\mathcal{P}_\ell$ also for some $\ell < n$. ■

Note that we did not require any ordering (like $x_0 < \ldots < x_n$) in theorem 2.27. This was no mistake. Obviously, the Lagrange interpolating polynomial is independent of the order in which we name the nodal points. (Also for this reason, we talk about a data set and not a data vector (a set has no ordering whereas a vector does)).

In the common case where the data set is of the form $\{(x_i, f(x_i)\}_{i=0}^n$ for some function f, then we denote p_n the (Lagrange) interpolating polynomial to f in the nodal points $x_0, \ldots, x_n$. Even if no function f is given a priori, obviously it is always possible to find a function f so that $f(x_i) = y_i$ for $i = 0, \ldots, n$. Hence, the interpolating polynomial to a function f is as general a construction as the interpolating polynomial to a data set, and we shall use the two constructions interchangeably as we see fit.

In Lagrange interpolation, we interpolate a function in a set of distinct nodal points. In some cases it may be interesting to interpolate not only function values but also values of derivatives. That this is not as easy as theorem (2.27) is indicated by the following example:

Example 2.32
We want to interpolate a function f so that the interpolating polynomial q satisfies $q(0) = f(0) = 0$, $q'(\frac{1}{2}) = f'(\frac{1}{2}) = 2$, $q(1) = f(1) = 1$. We have 3 conditions so we need $q \in \mathcal{P}_2$, i.e. $q(x) = a + bx + cx^2$. Plugging in the conditions we get $0 = q(0) = a$, $2 = q'(\frac{1}{2}) = b + c$ and $1 = q(1) = b + c$ which is not possible. ■

To get existence (and uniqueness) we restrict to Hermite interpolation conditions: If $q^{(k)}(x_i)$ is prescribed, then so must be $q^{(j)}(x_i)$ for $j = 0, \ldots, k-1$.

Theorem 2.33 *Let $x_0, x_1, \ldots, x_n$ be distinct real numbers (nodal points) and let k_i Hermite interpolation conditions $q^{(j)}(x_i) = y_i^j$ for $j = 0, \ldots, k_i - 1$ be prescribed in x_i for $i = 0, \ldots, n$ for arbitrary real values y_i^j, $j = 0, \ldots, k_i - 1$, $i = 0, \ldots, n$ (nodal point Hermite interpolation values). Define $m = (\sum_{i=0}^{n} k_i) - 1$.*
Then there exists a unique polynomial q_m of degree at most m $(\exists! q_m \in \mathcal{P}_m)$ satisfying the Hermite interpolation conditions, i.e.

$$q_m^{(j)}(x_i) = y_i^j, \text{ for } j = 0, \ldots, k_i - 1, \ i = 0, \ldots, n.$$

q_m is called the Hermite interpolating polynomial for the data set $\{(x_0, y_0^0), \ldots, (x_0, y_0^{k_0-1}), \ldots, (x_n, y_n^0), \ldots, (x_n, y_n^{k_n-1})\}$.

Proof:
The proof will not be given, but is based on the following **proof idea:** There is a unique polynomial in $\mathcal{P}_m$ having zeros of multiplicity k_i in x_i, $i = 0, \ldots, n$ since he total number of zeros counted with multiplicity is $m+1$. Then there is also a unique polynomial in $\mathcal{P}_m$ having arbitrary values. ■

We shall concentrate on Lagrange interpolation below. Whenever Hermite interpolation is in play it will be underlined. Instead the notation interpolation will be used exclusively for Lagrange interpolation. (Hence the parantheses around Lagrange in theorem 2.27).

It is important to realize that the Lagrange interpolating polynomial p_n is unique. This does not stop us however from writing it down in various forms. We shall consider here 3 ways of writing the interpolating polynomial.

2.2.1 The Newton form of the interpolating polynomial

The first form of the interpolating polynomial that we shall consider here is called the Newton form of the interpolating polynomial and comes directly from the proof of theorem 2.27 by induction. Using the notation from theorem 2.27 we start defining

$$\omega_k(x) = \prod_{i=0}^{k-1}(x - x_i) \in \mathcal{P}_k, \ k = 1, 2, \ldots \qquad \omega_0 \equiv 1 \in \mathcal{P}_0. \tag{2.12}$$

Theorem 2.34 *There exists constants c_i, $i = 0, \ldots, n$ such that the Lagrange interpolating polynomial p_n for the data set $\{(x_0, y_0), \ldots, (x_n, y_n)\}$*

(see theorem 2.27) can be written in the form

$$p_n(x) = \sum_{k=0}^{n} c_k \omega_k(x), \ n = 0, 1, 2, \ldots \quad \textit{(Newton form)} \tag{2.13}$$

Proof:
Induction using the proof of theorem 2.27. ■

Exercise 2.35
Do the induction proof leading to (2.13). ■

Let us consider in more details the coefficients c_i. It turns out, that the coefficient c_i, $i = 0, \ldots, n$ in the Newton form of the interpolating polynomial depends on $(x_0, y_0), \ldots, (x_i, y_i)$ only and is the coefficient of x^i in the interpolating polynomial p_i in this data set. This is shown as follows:

$$p_n(x) = c_0 + c_1 \underbrace{\omega_1(x)}_{(x-x_0)} + c_2 \underbrace{\omega_2(x)}_{(x-x_0)(x-x_1)} + \ldots + c_n \underbrace{\omega_n(x)}_{(x-x_0)(x-x_1)\ldots(x-x_{n-1})} \tag{2.14}$$

Note from (2.14) that c_n is the coefficient of x^n in p_n. By exchanging n for i, also c_i is the coefficient of x^i in p_i. Here c_i is the $(i+1)$'st coefficient in p_i and might be different from the $(i+1)$st coefficient in p_n. We shall see next that it really is not. To get expressions for $c_0, \ldots, c_n$ in p_n just plug in the various nodal points in (2.14):

$$\begin{aligned} y_0 &= p_n(x_0) = c_0 \\ y_1 &= p_n(x_1) = c_0 + c_1(x_1 - x_0) \\ &\vdots \\ y_n &= p_n(x_n) = c_0 + c_1(x_n - x_0) + \ldots + c_n(x_n - x_0)\ldots(x_n - x_{n-1}). \end{aligned} \tag{2.15}$$

For notational reasons we rewrite (2.15) in the form of interpolation to a function f. Recall that we do not lose any generality in doing this. For compactness of notation we rewrite it in matrix form as follows

$$\begin{bmatrix} 1 & & & \\ 1 & (x_1 - x_0) & & 0 \\ \vdots & \vdots & \ddots & \\ 1 & (x_n - x_0) & \cdots & (x_n - x_0)\cdot\ldots\cdot(x_n - x_{n-1}) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} y_0 = f(x_0) \\ y_1 = f(x_1) \\ \vdots \\ y_n = f(x_n) \end{bmatrix} \tag{2.16}$$

with a lower triangular matrix with zero entries everywhere above the diagonal. Such a linear system is easily solved by forward substitution where only the upper $i+1$ equations are utilized to recover c_i for $i = 0, \ldots, n$. These

upper $i+1$ equations are exactly the same as we would get if considering the equations for $c_0, \ldots, c_i$ in p_i. The additional nodal points and nodal point values $(x_{i+1}, y_{i+1}), \ldots, (x_n, y_n)$ play absolutely no role. Hence c_i in (2.13) depends on $(x_0, y_0), \ldots, (x_i, y_i)$ only, or on f and $x_0, \ldots, x_i$, and is the same as c_i in p_i. This proves our claim (in blue) above.

It turns out that the c_i's can be expressed using the following notation clearly exposing the dependence of c_i on f and $x_0, \ldots, x_i$ only:

Definition 2.36 *Divided differences:*

$$f[x_i] = f(x_i), \text{ for } i = 0, 1, \ldots \tag{2.17}$$

$$f[x_i, x_{i+1}, \ldots, x_j] = \frac{f[x_{i+1}, \ldots, x_j] - f[x_i, \ldots, x_{j-1}]}{x_j - x_i}, \quad \text{for } i = 0, 1, \ldots, \ j > i. \tag{2.18}$$

We show in theorem 2.37 below that $c_i = f[x_0, \ldots, x_i]$ so that (2.13) can be rewritten

$$p_n(x) = \sum_{k=0}^{n} f[x_0, \ldots, x_k]\omega_k(x), \ n = 0, 1, 2, \ldots \tag{2.19}$$

c_i is denoted the i'th divided difference.

Theorem 2.37 *Let c_i, $i = 0, \ldots, n$ be the divided differences given by* (2.16)*. Then*

$$\begin{aligned} c_0 &= f[x_0] = f(x_0) \\ c_1 &= f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \\ c_i &= f[x_0, \ldots, x_i] = \frac{f[x_1, \ldots, x_i] - f[x_0, \ldots, x_{i-1}]}{x_i - x_0} \ \text{for } i = 1, \ldots, n. \end{aligned} \tag{2.20}$$

Proof:
The proof is by induction on i:
The expressions for c_0 and c_1 follow directly from (2.16).
Assume that the theorem holds for $i-1$. For the general expression for c_i consider on top of the usual p_i interpolating f in $x_0, \ldots, x_i$ also $\tilde{p}_i \in \mathcal{P}_i$ interpolating f in $x_1, \ldots, x_{i+1}$. Then $p_i = \tilde{p}_{i-1} + \frac{x - x_i}{x_i - x_0}(\tilde{p}_{i-1} - p_{i-1})$ is evident since $rhs \in \mathcal{P}_i$ and $rhs(x_k) = f(x_k)$ for $k = 0, \ldots, i$. Now the general expression for c_i is obtained by comparing coefficients of the highest order term in x on the left and right hand sides ($c_i = \frac{\tilde{c}_{i-1} - c_{i-1}}{x_i - x_0}$). Obviously $\tilde{c}_{i-1} =$

$f[x_1, \ldots, x_i]$ by a simple change of notation. Plugging in, we get the required result. ■

With theorem 2.37 we can now construct an arbitrary divided difference using the following table construction where each new entry is computed from the one in the same row and previous column and from the one in the next row and previous column (1 back and 0 and 1 down) divided by the difference between the "extreme" x_i's.

$$\begin{array}{llll} x_0 & f[x_0] & f[x_0,x_1] & f[x_0,x_1,x_2] \quad f[x_0,x_1,x_2,x_3] \\ x_1 & f[x_1] & f[x_1,x_2] & f[x_1,x_2,x_3] \\ x_2 & f[x_2] & f[x_2,x_3] & \\ x_3 & f[x_3] & & \end{array} \tag{2.21}$$

The c_i's are appearing in the top row and each new divided difference requires adding another subdiagonal to the table as illustrated by the red entries in the table which are needed to compute the blue entry (c_3) having already all the black entries. It is important to note, that the black entries do not have to be recomputed just because a new nodal point (x_3) is added. This way of behavior of the coefficients in the Newton form allows a hierarchical construction of the interpolating polynomials. Say that we have an interpolating polynomial p_n corresponding to the data set $\{(x_0, y_0), \ldots, (x_n, y_n)\}$ and then add another nodal point and nodal point value (x_{n+1}, y_{n+1}). Then $p_{n+1} = p_n + c_{n+1}\omega_{n+1}$ so that only c_{n+1} must be found.

When p_n is interpolating some function f it is natural to be interested in the difference $(f - p_n)(x^*)$ in a point x^* which is *not* a nodal point. We may use the procedure above to get an expression for this interpolation error: Taking x_{n+1} above to be x^* and let f be a continuous function passing through the data set $\{(x_0, y_0), \ldots, (x_n, y_n)\}$. Then letting $y_{n+1} = f(x^*)$ we have $f(x^*) - p_n(x^*) = c_{n+1}\omega_{n+1}(x^*)$ where as above $c_{n+1} = f[x_0, \ldots, x_n, x^*]$ is the coefficient of x^{n+1} in p_{n+1}, the interpolating polynomial in the data set $\{(x_0, y_0), \ldots, (x_n, y_n), (x^*, f(x^*))\}$. Comparing to (2.21), we just need to add another subdiagonal to get the interpolation error in some point.

We recapitulate 2 properties of divided differences from above and add two new ones in the following theorem:

Theorem 2.38 *Divided differences have the following properties:*

1. $f[x_0, \ldots, x_n]$ *is independent of the ordering of the nodal points* $\{x_i\}_{i=0}^n$.
2. *If* $t \notin \{x_i\}_{i=0}^n$ *then the Interpolation error in* t *is given by* $f(t) - p_n(t) = f[x_0, \ldots, x_n, t]\omega_{n+1}(t)$.

3. *If* $f \in \mathcal{C}^n[a,b]$ *and* $\{x_i\}_{i=0}^n$ *are distinct points in* $[a,b]$ *then* $\exists \xi \in]a,b[$: $f[x_0,\ldots,x_n] = \frac{1}{n!}f^{(n)}(\xi)$.

4. *Divided differences can be defined also for Hermite interpolation: Say that the interpolation data set* $\{(x_0,y_0),\ldots,(x_n,y_n)\}$ *is a renaming of a set of hermite interpolation data* $\{(z_0,y_0^0),\ldots,(z_0,y_0^{k_0-1}),\ldots,(z_r,y_r^0),\ldots,(z_r,y_r^{k_r-1})\}$ *signifying that the nodal points are no longer necessarily distinct. Then*

$$f[x_0,\ldots,x_i] = \begin{cases} \frac{1}{x_i-x_0}\left(f[x_1,\ldots,x_i] - f[x_0,\ldots,x_{i-1}]\right) & \text{if } x_i \neq x_0 \\ \\ \frac{1}{i!}f^{(i)}(x_0) & \text{if } x_i = x_0 \end{cases}$$

Proof:
1: $f[x_0,\ldots,x_n] = c_n$ is the coefficient of the highest order term in p_n which is independent of the ordering of the nodal points as we saw above.
2: Let $\tilde{p}_{n+1}$ be interpolating polynomial for f in the nodal points $x_0,\ldots,x_n,t$. Then $\tilde{p}_{n+1}(x) = p_n(x) + f[x_0,\ldots,x_n,t]\omega_{n+1}(x)$. Taking $x = t$ and using $\tilde{p}_{n+1}(t) = f(t)$ we get the result.
3: By (2) we have $f[x_0,\ldots,x_n]\omega_n(x_n) = f(x_n) - p_{n-1}(x_n)$. In theorem 2.47 below we then show that $f(x_n) - p_{n-1}(x_n) = \frac{1}{n!}f^{(n)}(\xi)\omega_n(x_n)$ hence proving our property.
4: The proof is omitted but is based on an induction over i. ■

Since most of what we deal with here require computer evaluation, it is highly relevant to consider the cost of such a computer evaluation. A detailed investigation of the cost is highly complicated since it would involve not only the mathematical operations taking place, but also the internal data movements taking place in the computer. To simplify matters, we define a cost function that only takes into consideration the number of multiplications and divisions required. This is a very coarse measure, but in practice it tends to give a useful picture of the situation. For large computations the number of additions and subtractions are in many cases similar to the number of multiplications and divisions, so omitting them only amounts to omitting a factor 2. Neglecting all the data management in the computer is in some cases problematic, but often, if the programming is done with care, the problem is minimal. In any case we shall stick to the following definition:

Definition 2.39 *The cost function:*
Given some operation M *to be performed on a computer. The cost* $C(M)$ *is defined as the number of multiplications and divisions in the operation.*

Example 2.40
$C(\omega_n) = n - 1$. $C(p_n) = C(\sum_{k=0}^{n} c_k\omega_k) = \sum_{k=0}^{n}(1 + (k-1)) = \frac{1}{2}(n^2 + n) = \mathcal{O}_{n\to\infty}(n^2)$. (Since $C(p_n)/n^2 = \frac{1}{2}(1 + \frac{1}{n}) \to \frac{1}{2}$ as $n \to \infty$). ■

It turns out that if we are smart, the cost of computing p_n can be reduced significantly by using Horners algorithm for nested multiplication: Consider (2.14) and rewrite it by putting common factors outside as follows:

$$p_n(x)=c_0+(x-x_0)[c_1+(x-x_1)[c_2+(x-x_2)[c_3+\ldots+(x-x_{n-1})[c_n]\ldots]]]. \tag{2.22}$$

From this expression it is clear that we in reality can compute p_n at the cost n, i.e.

$$C_{\text{Horner}}(p_n) = n = \mathcal{O}_{n\to\infty}(n).$$

On top of this of course comes the cost of computing the coefficients c_i. But they only have to be computed once and can then be reused for each evaluation of p_n with a different x. For this reason it is safe to neglect the cost of computing the divided differences.

Now, why so much fuss about whether the cost is $\frac{1}{2}(n^2 + n)$ or n. Well consider the following table. While, for $n = 1$ or $n = 10$ the difference is

Table 2.1: The difference in growth between $\mathcal{O}(n)$ and $\mathcal{O}(n^2)$.

n	1	10	100	1000	10.000	100.000
$\frac{1}{2}(n^2 + n)$	1	55	5050	500.500	50.005.000	5.000.050.000

insignificant, instead doing 100.000 multiplications on a PC is doable whereas doing 5 billion is impossible. (Unless you are very, very patient). Note, that it is not implied here, that anybody should have any interest in $p_{100.000}$.

The main advantage of the Newton form of the interpolating polynomial lies in the fact seen above that it can be computed at linear cost, i.e. $C(p_n) = \mathcal{O}_{n\to\infty}(n)$ (when using Horners algorithm) and in the hierarchical construction, allowing us to add points one at a time without having to redo the previous work.

Example 2.41 Let us return to example 2.28, interpolating the data set from there. When using the option `form=Newton` i.e.

```
PolynomialInterpolation(xydata,x,form=Newton)
```

the Horner form corresponding to (2.22) is returned by Maple. In the case of example (2.28) this gives the output

$$((((-1/2*x+73/24)*(x-3)-5/3)*(x-2)+3/2)*(x-1)+1)*x.$$

Let us emphasize one final time, that this is the same polynomial as the one given in figure 2.4, only it is presented in a different form. ■

Exercise 2.42
a) Rewrite the polynomial from example 2.41 so that it precisely mimics (2.22), i.e. expand also the innermost term as $c_4 + (x - x_4)c_5$.
b) Now read of the coefficients c_0, c_1, c_2, c_3, c_4, c_5.
c) Recompute the coefficients c_i, $i = 0, \ldots, 5$ using their expressions as divided differences from theorem 2.37 and using definition 2.36 as basis for a recursive procedure. *Hint:* You do not need the function expression for f since all the function values that you need are stored in `xydata[i,2]`. All you need to convert definition 2.36 into a working recursion is the Maple procedure provided below. Note the +1's in `xydata` caused by the fact that x_i and $f(x_i)$ are stored in `xydata[i+1,1]` and `xydata[i+1,2]` respectively. A really cool way to print out the results is provided below the procedure: ■

```
dd:=proc(i::nonnegint,j::nonnegint)
  if i=j then
    xydata[i+1,2]
  else
    (dd(i+1,j)-dd(i,j-1))/(xydata[j+1,1]-xydata[i+1,1])
  end if;
end proc:

for i from 0 to 5 do
  printf("%s%d%s%a\n","c[",i,"]=",dd(0,i))
end do;
```

2.2.2 The Lagrange form of the interpolating polynomial

The Lagrange form of the interpolating polynomial is the one preferred for theoretical work since it is expressed using the concept af a basis.

Definition 2.43 *For $k = 0, \ldots, n$, the k'th characteristic polynomial or the k'th cardinal function for the distinct real nodal points $x_0, \ldots, x_n$ is given by*

$$\ell_k \in \mathcal{P}_n, \ \ell_k(x_i) = \delta_{ki}, i = 0, \ldots, n \tag{2.23}$$

where δ_{ki} is the Kronecker delta i.e takes the value 1 if $k = i$ and 0 if $k \neq i$.

Note that x_k is normally not a maximum (neither local nor global) for ℓ_k. See figure 2.5 and exercise 2.45).

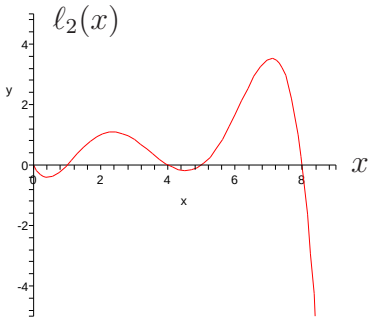


Figure 2.5: Example of the second cardinal function ℓ_2 for the nodal points $x_0 = 0$, $x_1 = 1$, $x_2 = 2$, $x_3 = 4$, $x_4 = 5$ and $x_5 = 8$.

Theorem 2.44 *For $k = 0, \ldots, n$ and for distinct real nodal points $x_0, \ldots, x_n$, the k'th cardinal function is uniquely determined and given by*

$$\begin{aligned} \ell_k(x) &= \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - x_i}{x_k - x_i} \\ &= \frac{(x - x_0) \cdot \ldots \cdot (x - x_{k-1})(x - x_{k+1}) \cdot \ldots \cdot (x - x_n)}{(x_k - x_0) \cdot \ldots \cdot (x_k - x_{k-1})(x_k - x_{k+1}) \cdot \ldots \cdot (x_k - x_n)}. \end{aligned} \tag{2.24}$$

Further $\{\ell_k\}_{k=0}^n$ is a basis for $\mathcal{P}_n$ and the interpolating polynomial p_n (see theorem 2.27) can be written in the Lagrange form of the interpolating polynomial for the data set $\{(x_0, y_0), \ldots, (x_n, y_n)\}$

$$p_n(x) = \sum_{k=0}^{n} y_k \ell_k(x). \tag{2.25}$$

Proof:
For uniqueness just apply theorem 2.27.
For the expression for ℓ_k just check the conditions in (2.23).
For the basis claim: there are $n+1$ ℓ_k's and also $\dim\mathcal{P}_n = n+1$ so the dimensions fit. Further $\ell_k \in \mathcal{P}_n$ for $k = 0, \ldots, n$ so $\{\ell_k\}_{k=0}^n$ span a subspace of $\mathcal{P}_n$ which must be all of $\mathcal{P}_n$ if only we can show that the ℓ_k's are linearly independent: Assume $0 \equiv \sum_{k=0}^n c_k \ell_k(x)$. This implies that $0 = \sum_{k=0}^n c_k \ell_k(x_i) = c_i,\ i = 0, \ldots, n$ by the Kronecker delta property of ℓ_k.
For the expression for p_n just check the conditions ($p_n \in \mathcal{P}_n$ and $p_n(x_i) = y_i$ for $i = 0, \ldots, n$). ■

Exercise 2.45
Using Maple, program ℓ_0, ℓ_1, ℓ_2, ℓ_3, ℓ_4 and ℓ_5, for the example of figure 2.5 and plot them.
Use this to program the Lagrange form of the interpolating polynomial with the data values y_0, y_1, y_2, y_3, y_4, y_5 as input variables.
Finally compute and plot the Lagrange interpolating polynomials with the following data values:
a) $y_0 = 1$, $y_1 = 0$, $y_2 = 0$, $y_3 = 0$, $y_4 = 0$, $y_5 = 0$.
b) $y_0 = 0$, $y_1 = 1$, $y_2 = 2$, $y_3 = 4$, $y_4 = 5$, $y_5 = 8$.
c) $y_0 = 1$, $y_1 = 7$, $y_2 = -4$, $y_3 = 3$, $y_4 = 0$, $y_5 = 5$. ■

Note the non-hierarchical approach of the Lagrange form of the interpolating polynomial: If another nodal point is added, all the characteristic polynomials change, and we must start from scratch. The Lagrange form is hierarchical in another way however. Consider the situation where we have one set of nodal points but a number of different sets of nodal point values (for example an experiment measuring seismic activity at various places in a given location (like Yellowstone) but at many different times). Here the characteristic functions remain the same since they are independent of the nodal point values.

The cost of evaluating the Lagrange form of the interpolating polynomial is $2(n-1)$ multiplications and 1 division for each of the characteristic polynomials. Then there is another multiplication by a nodal point value y_k for a total of $2n$. This is done $n+1$ times for a total cost of $C_{\text{Lagrange}}(p_n) = 2(n^2+n) = \mathcal{O}_{n\to\infty}(n^2)$, i.e. an order more than for the Newton form using Horners algorithm. In the case of one set of nodal points but a number, say m, of different sets of nodal point values considered above $C_{\text{Lagrange}}(p_n) = (2n-1)(n+1) + (n+1)m = 2n^2 + n - 1 + (n+1)m$. Newtons form must be redone for each new set of nodal point values giving

a total cost of nm. Hence for $m > n$ the orders of Lagrange and Newton become the same.

In conclusion the Lagrange form has advantages for theoretical work because of the basis concept. Further it is cost effective in a situation with many different nodal point value sets for each nodal point set.

2.2.3 The Vandermonde form of the interpolating polynomial

The Vandermonde form of the interpolating polynomial is the classical *high-school form* where the polynomial is written as a linear combination of powers of the unknown, i.e.

$$p_n(x) = \sum_{k=0}^{n} a_k x^k = a_0 + a_1 x + \ldots + a_n x^n. \tag{2.26}$$

To express the coefficients a_i in terms of the nodal points and nodal point values in the data set $\{(x_0, y_0), \ldots, (x_n, y_n)\}$ we must solve the linear equation system

$$y_i = p_n(x_i) = \sum_{k=0}^{n} a_k x_i^k = a_0 + a_1 x_i + \ldots + a_n x_i^n, \ i = 0, \ldots, n, \tag{2.27}$$

or written in matrix form

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix} \tag{2.28}$$

The matrix in (2.28) is called a Vandermonde matrix. A matrix of this form is often ill-conditioned for large values of n meaning that small numerical inaccuracies may result in large inaccuracies in the coefficients a_k. In practice, the a_k's are computed by first computing the Newton form of the interpolating polynomial using Horners algorithm and then computing the a_k's from the c_k's. This has a cost of $C_{\text{Vandermonde}}(p_n) = \mathcal{O}_{n\to\infty}(n^2)$ and hence is computationally much more expensive than the Newton form itself. In conclusion, the vandermonde form can be recommended only for small problems and is not used for "serious numerical work".

2.2.4 Error in polynomial interpolation

When it comes to passing a polynomial through the points of a data set, polynomial interpolation by definition is flawless. Instead, when considering the interpolating polynomial as the approximation to a function f also passing through the points of the data set, then there may be errors in the points that are not nodal points. (We considered this situation briefly in section 2.2.1). These errors may be significant (in the worst points) as shown in the following classical example.

Example 2.46 Runge's example
Take $f(x) = \frac{1}{x^2+1}$, $x \in [-5, 5]$ and consider the nodal points $x_0 = -5$ and $x_n = 5$ with $x_1, \ldots, x_{n-1}$ equidistributed between x_0 and x_n. f and p_n are shown in figure 2.6 for various values of n. It should be noted, that the oscillations in p_n with increasing n observed here for Lagrange interpolation also happens with Hermite interpolation. Hermite interpolation is *not* a way to avoid oscillations. ■

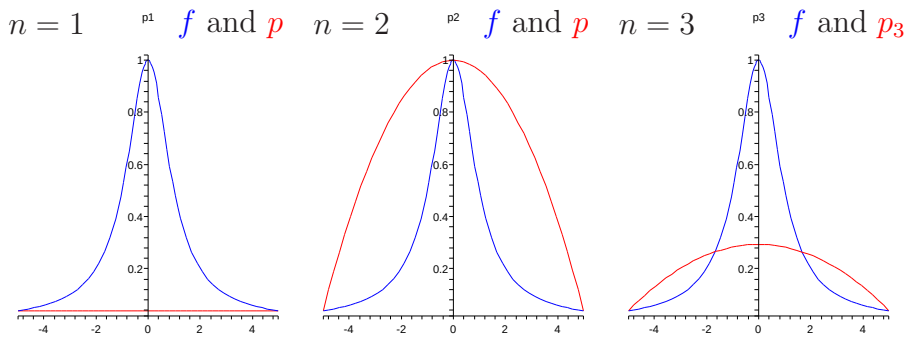


Figure 2.6: Runge's example of the function $f = \frac{1}{x^2+1}$ (blue, bellshaped graph) and a number of its interpolating polynomials p_n, $n = 1, 2, 3$ (red graphs) with equidistant nodal points always including $x_0 = -5$ and $x_n = 5$. The red graphs are *not* converging to the blue graph as n is increasing. See continuation below.

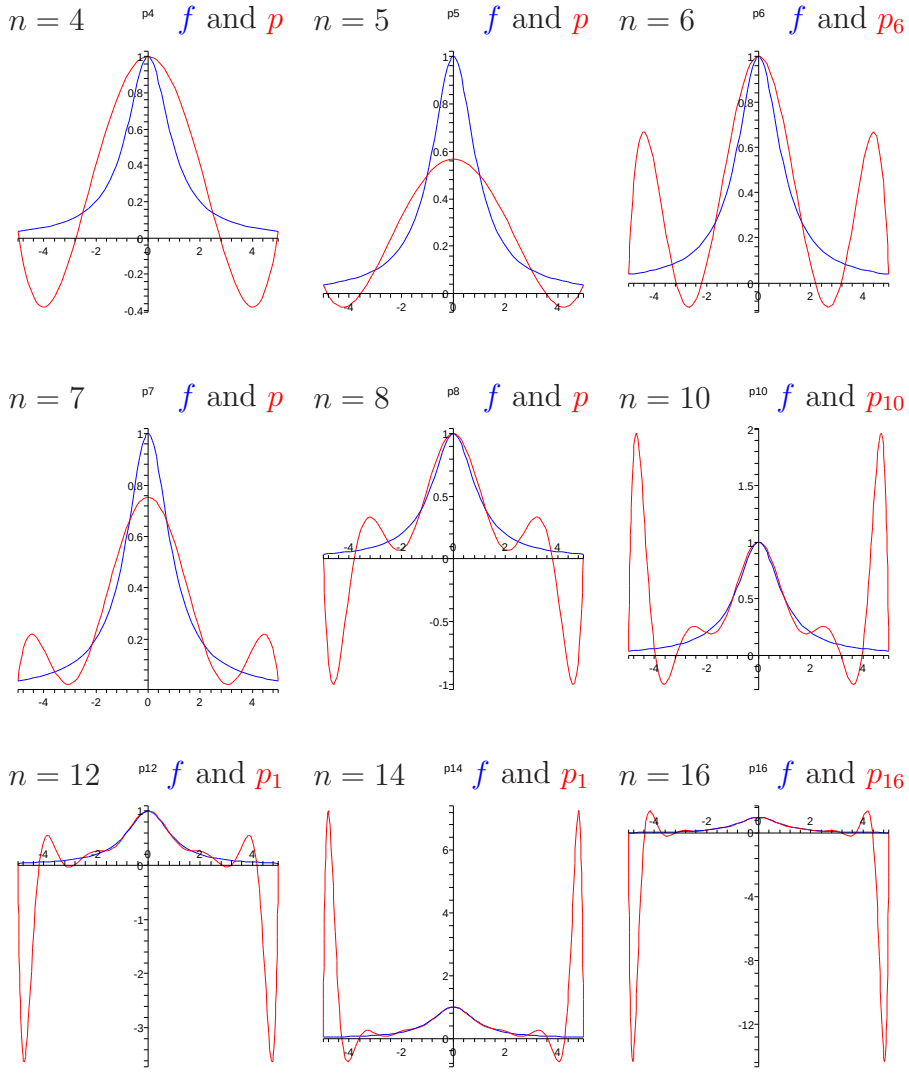


Figure 2.6: Runge's example of the function $f = \frac{1}{x^2+1}$ (blue, bellshaped graph) and a number of its interpolating polynomials p_n, $n = 4, 5, 6, 7, 8, 10, 12, 14, 16$ (red oscillating graphs) with equidistant nodal points always including $x_0 = -5$ and $x_n = 5$. The red graphs are *not* converging to the blue graph as n is increasing. See the cases $n = 1, 2, 3$ above.

In an attempt to understand the situation better we have the following theorem:

Theorem 2.47 *Let $f \in \mathcal{C}^{n+1}[a,b]$, p_n be the interpolating polynomial to f in the distinct nodal points $x_0, \ldots, x_n$ where $a \leq x_i \leq b$ for $i = 0, \ldots, n$ and let ω_{n+1} be defined by (2.12). Then*

$$(2.29)\quad \forall x \in [a,b]\ \exists \xi_x \in]a,b[:\ f(x) - p_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi_x)\omega_{n+1}(x).$$

Proof:
1) If $x = x_i$ for some $i = 0, \ldots, n$ then (2.29) takes the form $0 = 0$ which is true.
2) If x is distinct from x_i, $i = 0, \ldots, n$ then define $\phi(t) = f(t) - p_n(t) - \lambda\omega_{n+1}(t)$ for $\lambda = \frac{f(x)-p_n(x)}{\omega_{n+1}(x)}$ so that $\phi(x) = 0$. But $\phi(x_i) = 0$, $i = 0, \ldots, n$ so $\phi \in \mathcal{C}^{n+1}[a,b]$ has at least $n+2$ roots, which according to Rolle's theorem implies that ϕ' has at least $n+1$ roots. Reapplying Rolle's theorem n times we see that $\phi^{(n+1)}$ has at least 1 root. Call it ξ_x. Then $0 = \phi^{(n+1)}(\xi_x) = f^{(n+1)}(\xi_x) - \lambda(n+1)!$ Plug in λ and we get the required result. ■

Now consider the 3 terms on the right hand side in (2.29): $\frac{1}{(n+1)!}$ is small for large values of n which is good. $f^{(n+1)}(\xi_x)$ is somewhat out of our control. The best we can do generally is the upper estimate $f^{(n+1)}(\xi_x) \leq \max_{a\leq x\leq b} |f^{(n+1)}(x)|$ which of course depends heavily on the function f and in many cases may be very big. $M_n = \max_{a\leq x\leq b} |\omega_{n+1}(x)|$ instead only depends on the set of nodal points $\{x_i\}_{i=0}^n$ and hence something more specific can be said. In conclusion, theorem 2.47 then gives us the following upper bound for the interpolation error:

$$(2.30)\forall x \in [a,b]\ |f(x) - p_n(x)| \leq \frac{1}{(n+1)!} \max_{a\leq x\leq b} |f^{(n+1)}(x)| \max_{a\leq x\leq b} |\omega_{n+1}(x)|.$$

Maple offers 3 ways to find the maximum of a function:

- The Maple `maximize` is a symbolic command finding zero's of the derivatives of the function, resulting in the global maximum of the function. Unfortunately, often `maximize` is unable to perform the necessary computations and returns unevaluated.

- The command `Optimization[Maximize]` is a numerical command meaning that in practice it works in more cases than `maximize`. Unfortunately `Optimization[Maximize]` only returns a local maximum and hence must be wrapped in an additional layer (to be provided by the user) which finds a starting point for `Optimization[Maximize]` which is close to the global maximum.

- The third option is the "brute force" option. Simply put a lot of points evenly over the interval $[a, b]$, compute the function values in each point and find the biggest one. This method can be made more sophisticated by iteration: Once a subinterval is found for the global maximum, subdivide this subinterval by putting a lot of points evenly over the subinterval and find a new smaller subinterval where the global maximum is situated. Keep on iterating until the width of the subinterval with the global maximum is smaller than a given tolerance like 10^{-10} or until the difference between the function values in the endpoints of the subinterval is below the tolerance.

Exercise 2.48
a. Check for Runge's example the three terms on the right in (2.30) for n from 1 to 20 and give upper bounds for the error for these values of n. (Use maple). For the maximizations use the brute force method without iteration, selecting the 1000 test points $-5 + 5\frac{i-1}{999}$, $i = 1, \ldots, 1000$.
b. Compare your upper bounds for the error from (a) to the true errors you can read of figure 2.6. Is your bound "tight", i.e. are the upper bounds close to the true errors? ■

Consider the following result on how small M_n can get by choosing the nodal points optimally. We consider only one special case of $[a, b] = [-1, 1]$:

Theorem 2.49 *If $[a, b] = [-1, 1]$ then $M_n \geq \frac{1}{2^n}$ and the minimal value is attained when the set of nodal points $\{x_i\}_{i=0}^n$ are the roots of the Chebyshev polynomial $T_{n+1} \in \mathcal{P}_{n+1}$.*

Proof:
Self study about Chebyshev polynomials. ■

But minimizing M_n is not nearly enough to assure convergence. We have the following (unfortunate) result

Theorem 2.50 *For any selection of nodal points $\{\{x_i^{(n)}\}_{i=0}^n\}_{n=0}^\infty$ such that $a \leq x_0^{(n)} < x_1^{(n)} < \ldots < x_n^{(n)} \leq b$ for any $n = 0, 1, \ldots$*

$$(2.31) \quad \exists f \in \mathcal{C}^0([a,b]) : p_n \underset{n\to\infty}{\overset{u}{\not\to}} f \text{ i.e. } \max_{x\in[a,b]} |f(x) - p_n(x)| \underset{n\to\infty}{\not\to} 0.$$

Here $\overset{u}{\not\to}$ means "does not converge uniformly to". Uniform convergence would mean that
$\forall \epsilon > 0 \ \exists N_\epsilon > 0 : |f(x) - p_n(x)| < \epsilon \ \forall x \in [a, b], \ \forall n > N_\epsilon$.

Proof:
Omitted. The result dates back to 1914 and is not elementary. ■

In short, theorem 2.50 says that it is necessary that the nodal points depend on f to get convergence as $n \to \infty$. The following (fortunate) result says, that it is not only necessary but it is also sufficient that the nodal points depend on f to get convergence as $n \to \infty$ as long as the nodal points are selected in the right way for the given f:

Theorem 2.51 *For any $f \in \mathcal{C}^0([a,b])$ there exists a selection of nodal points $\{\{x_i^{(n)}\}_{i=0}^n\}_{n=0}^\infty$ such that*

$$\max_{x\in[a,b]} |f(x) - p_n(x)| \underset{n\to\infty}{\longrightarrow} 0. \tag{2.32}$$

Proof:
Omitted. The result is obtained by a combination of the Weirstrass approximation theorem and the Chebyshev alternation theorem and depends on the notion of Bernstein polynomials. For more details see for example [5]. ■

One reason that we omit the proof of theorem 2.51 is that the proof is not useful for numeric work. Although the proof is constructive, actually giving formulas for the relevant Bernstein polynomials, the convergence obtained is so slow that it is not useful for practical purposes, only to show the convergence in theory.

We finish the error section by considering how close the interpolating polynomial is to the best approximation in the sense of definition 2.13 taking $\mathcal{V} = \mathcal{C}^0([a,b])$, $\tilde{\mathcal{V}} = \mathcal{P}_n([a,b])$ and $d = \|\cdot\|_\infty$ (see example 2.10). We have the following result:

Theorem 2.52 *With the notation of theorem 2.27, if $f \in \mathcal{C}^0([a,b])$ and $\|f\|_\infty = \max_{x\in[a,b]} |f(x)|$ we have*

$$\min_{q\in\mathcal{P}_n([a,b])} \|f-q\|_\infty \le \|f-p_n\|_\infty \le (1+\Lambda_n) \min_{q\in\mathcal{P}_n([a,b])} \|f-q\|_\infty \tag{2.33}$$

where the Lebesgue constant $\Lambda_n = \max_{x\in[a,b]} \sum_{k=0}^n |\ell_k(x)|$. ($\ell_k$ is the k'th characteristic polynomial given by definition 2.43 and theorem 2.44). Also

$$\Lambda_n > \frac{2}{\pi}\ln(n+1) - C \text{ for } n = 0,1,\ldots \text{ and some } C > 0 \tag{2.34}$$

and hence

$$\Lambda_n \underset{n\to\infty}{\longrightarrow} \infty.$$

Hence there is no guarantee that p_n gets close to the the best approximation in $\|\cdot\|_\infty$ as n increases.

$$\Lambda_n \simeq \frac{2^{n+1}}{e \cdot n \cdot \ln(n)} \text{ for equidistributed nodes } (e = 2.7183\ldots) \tag{2.35}$$

and hence

$$\Lambda_n \underset{n\to\infty}{\longrightarrow} \infty \text{ exponentially.}$$

Hence, equidistributed nodes are among the worst we may choose. This partially explains the Runge example.

Proof:
Omitted. For more details see for example [4]. ■

The Lebesgue constant is closely connected to the notion of stability. Stability is used in many different contexts, but the most common usage is the following: Given some proces P with an input i and an output $P(i)$. Given 2 possible inputs i_1 and i_2 that are close in some sense, P is said to be stable if also $P(i_1)$ and $P(i_2)$ are close.

In the context of interpolation consider 2 functions $f \in \mathcal{C}^0([a,b])$ and $g \in \mathcal{C}^0([a,b])$ with interpolating polynomials $p_n \in \mathcal{P}_n([a,b])$ and $q_n \in \mathcal{P}_n([a,b])$ respectively in the common nodal points $x_0, \ldots, x_n$ all situated in the interval $[a,b]$. Then for $\|f\|_\infty = \max_{x\in[a,b]} |f(x)|$

$$\begin{aligned} \|p_n - q_n\|_\infty &= \max_{x\in[a,b]} |\sum_{k=0}^{n} (f(x_k) - g(x_k))\ell_k(x)| \\ &\leq \max_{k=0,\ldots,n} |f(x_k) - g(x_k)| \cdot \Lambda_n, \end{aligned} \tag{2.36}$$

where the middle term is the Lagrange form of the interpolating polynomials. Hence the Lebesgue constant Λ_n is a condition number (stability measure) for interpolation in the sense that small differences in all nodal point values of the functions leads to small differences in the $\mathcal{L}^\infty$ norm for the interpolating polynomials if and only if the Lebesgue constant is not too big.

We conclude that Minimizing Λ_n gives the best stability and also an error closest to the best possible in the $\mathcal{L}^\infty$ norm. But because of (2.34), minimizing Λ_n will not by itself give a convergence result.

2.3 Piecewise polynomial interpolation

We shall discuss two different types of piecewise polynomial interpolation, both originating from the spaces $\mathcal{S}_\Delta^{k,\ell}([a,b])$ introduced in example 2.8 on page 21.

2.3.1 Piecewise Lagrange interpolation and $\mathcal{S}_{\Delta}^{k,1}([a,b])$

Recall the space of piecewise Lagrange interpolants of degree k, $\mathcal{S}_{\Delta}^{k,1}([a,b])$ introduced in example 2.8 as the space of globally continuous functions ($\mathcal{C}^0([a,b])$) that locally are polynomials of degree at most k i.e. belong to $\mathcal{P}_k(\Delta_i)$ for $i = 1,\ldots,n$ where the Δ_i are the subintervals in a subdivision $\Delta = \{\Delta_i\}_{i=1}^n$ of $[a,b]$, i.e. $\exists\{x_i\}_{i=0}^n : a = x_0 < x_1 < \ldots < x_n = b$, $\Delta_i = (x_{i-1}, x_i)$, $i = 1,\ldots,n$.

Let $k \geq 1$ and select $k+1$ distinct interpolation nodes $\{x_{ij}\}_{j=0}^k$ in each subinterval Δ_i, $i = 1,\ldots,n$ such that $x_{i0} = x_{i-1}$ and $x_{ik} = x_i$, i.e. the end points of the subintervals are always nodal points. Denote also $h_i = |\Delta_i| = x_i - x_{i-1}$ for $i = 1,\ldots,n$ and $h = \max_{i=1,\ldots,n} h_i$. See figure 2.7 for the notation.

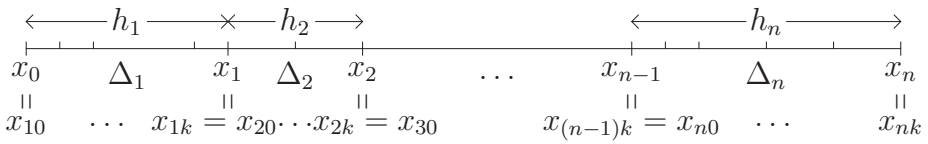


Figure 2.7: Notation for piecewise Lagrange interpolation.

Let $\{\{y_{ij}\}_{j=0}^k\}_{i=1}^n$ be nodal point values, such the y_{ij} is assigned in the nodal point x_{ij} for $i = 1,\ldots,n$ and $j = 0,\ldots,k$. By theorem 2.27 there exists a unique interpolating polynomial $p_{\Delta_i k} \in \mathcal{P}_k(\Delta_i)$ in each subinterval Δ_i, for $i = 1,\ldots,n$. To enforce the global continuity we only need to require $y_{ik} = y_{(i+1)0}$ for $i = 1,\ldots,n-1$, which is automatically satisfied if we consider piecewise Lagrange interpolation of a continuous function f, i.e. $y_{ij} = f(x_{ij})$ for $i = 1,\ldots,n$ and $j = 0,\ldots,k$. The resulting unique piecewise Lagrange interpolant of degree k is denoted $s_{\Delta}^{k,1}$. We sum up this result in the following theorem:

Theorem 2.53 *Let $\Delta = \{\Delta_i\}_{i=1}^n$ be a subdivision of $[a,b]$ such that $\exists\{x_i\}_{i=0}^n$: $a = x_0 < x_1 < \ldots < x_n = b$, $\Delta_i = (x_{i-1}, x_i)$, $i = 1,\ldots,n$.*
Let $k \geq 1$ and let $\{x_{ij}\}_{j=0}^k$ be distinct interpolation nodes in Δ_i, $i = 1,\ldots,n$ such that $x_{i0} = x_{i-1}$ and $x_{ik} = x_i$.
Let $f \in \mathcal{C}^0([a,b])$ and let $y_{ij} = f(x_{ij})$ for $j = 0,\ldots,k$ and $i = 1,\ldots,n$.
Then there exists a unique piecewise Lagrange interpolant of degree k, $s_{\Delta}^{k,1} \in \mathcal{S}_{\Delta}^{k,1}([a,b])$ interpolating f in the data set $\{\{(x_{ij}, y_{ij}\}_{j=0}^k\}_{i=1}^n$.

The following result on the interpolation error for piecewise Lagrange interpolants have big advantages over the results for Lagrange and Hermite interpolants:

Theorem 2.54 *With $s_\Delta^{k,1}$ given by theorem 2.53 and $f \in \mathcal{C}^{k+1}([a,b])$ we have*

$$\|f - s_\Delta^{k,1}\|_{\mathcal{L}^\infty(a,b)} \leq \frac{1}{(k+1)!}\|f^{(k+1)}\|_{\mathcal{L}^\infty(a,b)} h^{k+1} \underset{h\to 0}{\longrightarrow} 0, \tag{2.37}$$

where $\|f\|_{\mathcal{L}^\infty(a,b)}$ is an alternative (more explicit) way to denote $\|f\|_\infty$, i.e. $\max_{x\in(a,b)} |f(x)|$.

Proof:
From (2.29) we have

$$f(x) - s_\Delta^{k,1}(x) = \frac{1}{(k+1)!} f^{(k+1)}(\xi_x) \underbrace{\omega_{k+1}(x)}_{=\prod_{j=0}^{k}(x-x_{ij})}, \text{ for } x \in \Delta_i$$

$$\Downarrow$$

$$\|f(x) - s_\Delta^{k,1}(x)\|_{\mathcal{L}^\infty(\Delta_i)} \leq \frac{1}{(k+1)!}\|f^{(k+1)}\|_{\mathcal{L}^\infty(\Delta_i)} \underbrace{\max_{x\in\Delta_i} \prod_{j=0}^{k} \underbrace{|x - x_{ij}|}_{\leq h_i}}_{\leq h_i^{k+1} \leq h^{k+1}}$$

from which the result is evident. ■

Similar results can be obtained for other norms like the $\mathcal{L}^2$ norm also. So if you insist on a predetermined distribution of the nodal points (like in a uniform subdivision) then piecewise Lagrange interpolation will converge as the maximal distance between nodal points converge to zero, whereas Lagrange and Hermite interpolation is not guaranteed to do so.

Exercise 2.55
Use Maples Spline command in the CurveFitting package to redo Runges example using piecewise linear Lagrange interpolants instead of the interpolating polynomials of figure 2.6, for $1, 2, 3, \ldots, 20$ subintervals. Note that the Spline command gives a piecewise Lagrange interpolant *only* in the linear case. In general you need to use the PolynomialInterpolation command in the CurveFitting package multiple times, which is more complicated. Compare the results to the results with the lagrange interpolants: Is it better or worse? Can you see the "effect" of theorem 2.54? ■

2.3.2 Spline interpolation and $\mathcal{S}_\Delta^{k,k}([a,b])$

Let us increase the complexity by turning to the space of splines of degree k, $\mathcal{S}_\Delta^{k,k}([a,b])$ introduced in example 2.8 as the space of $k-1$ times globally

continuous differentiable functions ($\mathcal{C}^{k-1}([a,b])$) that locally are polynomials of degree at most k i.e. belong to $\mathcal{P}_k(\Delta_i)$ for $i = 1,\ldots,n$ where the Δ_i are the subintervals in a subdivision $\Delta = \{\Delta_i\}_{i=1}^n$ of $[a,b]$, i.e. $\exists\{x_i\}_{i=0}^n : a = x_0 < x_1 < \ldots < x_n = b$, $\Delta_i = (x_{i-1}, x_i)$, $i = 1,\ldots,n$. We shall denote by s_Δ^k any function in $\mathcal{S}_\Delta^{k,k}([a,b])$ interpolating a data set $\{(x_0,y_0),\ldots,(x_n,y_n)\}$ for instance based on the values of a function $f \in \mathcal{C}^0([a,b])$ in the endpoints of the subintervals, i.e. $s_\Delta^k(x_i) = y_i = f(x_i)$ for $i = 0,\ldots,n$. Here we only consider the two most commonly used examples, namely the linear and the cubic spline.

Example 2.56 The linear spline s_Δ^1

Note that the space of linear splines and the space of piecewise Lagrange polynomials of degree 1 are both $\mathcal{S}_\Delta^{1,1}([a,b])$. Also the unique linear spline s_Δ^1 is identical to $s_\Delta^{1,1}$ (see theorem 2.53), i.e. the globally continuous function ($\mathcal{C}^0([a,b])$) that locally is a polynomial of degree at most 1 which interpolates in the endpoints of each subinterval of Δ. For an example see figure 2.8. ■

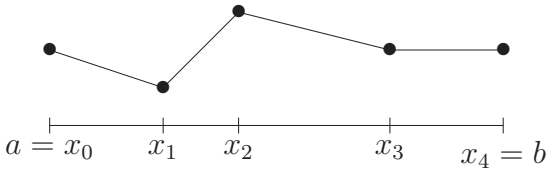


Figure 2.8: Example of linear spline interpolation.

Example 2.57 The cubic splines s_Δ^3

Note the plural. s_Δ^3 is not unique as we shall see now. s_Δ^3 is a function which is globally twice continuously differentiable ($s_\Delta^3 \in \mathcal{C}^2([a,b])$) and locally a polynomial of degree at most 3 ($s_{\Delta_i}^3 := s_\Delta^3|_{\Delta_i} \in \mathcal{P}_3(\Delta_i)$ for $i = 1,\ldots,n$). Finally it takes on specific values in all nodal points. In order to understand existence and uniqueness properties we shall consider separately the conditions for global smoothness ($\mathcal{C}^2([a,b])$) and local form $\mathcal{P}_3(\Delta_i)$ for $i = 1,\ldots,n$ starting with the local form: In order for a cubic polynomial to exist and be uniquely determined, by theorem 2.27 and 2.33 it suffices with 4 conditions of either Lagrange or Hermite type in the interval. Hence we need a total of 4 Lagrange or Hermite conditions in each of the n intervals, i.e. $4n$ conditions to determine uniquely the local form. Turning to the global smoothness, since $\mathcal{P}_3 \subset \mathcal{C}^\infty$ we only need to worry about the smoothness in the internal endpoints of the subintervals i.e. in $x_1,\ldots,x_{n-1}$ (see figure 2.7). To have smoothness $\mathcal{C}^2$ in an internal end point x_i ($i = 1,\ldots,n-1$) we need the function value and the first and second derivatives in x_i to be the same

from both left (in Δ_i) and from right (in Δ_{i+1}), i.e. $s^3_{\Delta_i}(x_i) = s^3_{\Delta_{i+1}}(x_i)$, $(s^3_{\Delta_i})'(x_i) = (s^3_{\Delta_{i+1}})'(x_i)$ and $(s^3_{\Delta_i})''(x_i) = (s^3_{\Delta_{i+1}})''(x_i)$. Finally we need s^3_Δ to satisfy the interpolation conditions $s^3_\Delta(x_i) = y_i = f(x_i)$ for $i = 0, \ldots, n$. Some of these conditions are overlapping. To get a clear picture consider figure 2.9. Here a $\bullet$ indicates a function value given for s^3_Δ, valid for both the left and right subinterval in the case of internal nodes. A $\circ$ and a $\bigcirc$ indicates a first and second derivative respectively that must be the same both in the left and right subinterval. Instead no value is given to the derivatives. Now

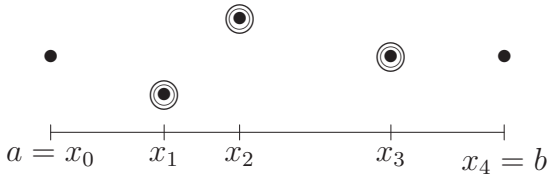


Figure 2.9: Example of cubic spline interpolation conditions.

let us count the conditions: There is one "value" condition in each of the two end points x_0 and x_n. Further there are 2 value conditions in each of the $n-1$ internal nodes $x_1, \ldots, x_{n-1}$ (one in the left subinterval and one in the right). Finally there are 2 "derivative" conditions in each of the $n-1$ internal nodes (one on the first derivative and one on the second). All together we have 4 conditions in each of the $n-1$ internal nodal points plus the 2 end point value conditions for a total of $4n-2$ conditions. Here there are 2 important things to note:
1. We have $4n-2$ conditions but we need $4n$ conditions to determine s^3_Δ uniquely. So we miss two conditions in order to make s^3_Δ unique. The following two special cases are the most common:

- The natural cubic spline $s^{3,n}_\Delta$ is obtained by further imposing $(s^{3,n}_\Delta)''(x_0) = 0$ and $(s^{3,n}_\Delta)''(x_n) = 0$ on top of the usual cubic spline conditions. Look at it in the following way: "How do we do the least damage in the interpolation". Well, the answer is, that if you do not know anything, forcing the second derivative will be less obvious than forcing the first derivative or even the value into some (wrong) value. We could of course force an even higher derivative but since $s^{3,n}_\Delta$ is a cubic polynomial locally, the third derivative is constant and can not be forced with two conditions in one interval. Hence we loose the special case $n = 1$. What value should the second derivative be forced to attain. Well, knowing nothing we choose the value 0 for no particular reason. But at least 0 is independent of scaling.

- The periodic cubic spline $s_\Delta^{3,p}$ is obtained by further imposing $(s_\Delta^{3,p})'(x_0) = (s_\Delta^{3,p})'(x_n)$ and $(s_\Delta^{3,p})''(x_0) = (s_\Delta^{3,p})''(x_n)$ on top of the usual cubic spline conditions. The periodic spline is used *only* when further $s_\Delta^{3,p}(x_0) = y_0 = y_n = s_\Delta^{3,p}(x_n)$ is satisfied by the data set. The periodic spline is deviced for the situation where the underlying function f being interpolated is known (or suspected) to be periodic with the period $b-a$ (or some integer fraction of this like $\frac{b-a}{2}$).

2. It is important to realize that the question of existence is not elementary. To guarantee existence, we need 4 conditions of Lagrange or Hermite type in each of the n subintervals. Instead we have (if we consider only the natural or periodic cubic splines) a total of $4n$ conditions (which is good) but many of them are cross interval conditions involving 2 intervals. For the natural spline clearly the conditions in x_0 and x_n are *not* of Hermite type, since the conditions on the first derivatives are missing. Whether the conditions in the internal nodes are of Hermite type depends on how we can interpret the cross interval conditions. We shall return to this issue below. ■

Exercise 2.58
We know that s_Δ^1 has 0 free parameters and s_Δ^3 has 2. Show that s_Δ^k has $k-1$ free parameters for any $k \geq 1$. ■

Exercise 2.59
Omitting the requirement that the spline has to attain the nodal point values in the nodal points, but maintaining the smoothness requirement ($\mathcal{C}^{k-1}$) and in particular that the values in the internal nodal points are the same from left and right, show that $\dim \mathcal{S}_\Delta^{k,k} = n + k$. ■

The natural spline $s_\Delta^{k,n}$ is defined for k odd (since by exercise 2.58 we then have an even number of free parameters) by the additional $k-1$ conditions

$$(s_\Delta^{k,n})^{(k-1)}(a) = (s_\Delta^{k,n})^{(k-1)}(b) = 0, \ldots, (s_\Delta^{k,n})^{(\frac{k+1}{2})}(a) = (s_\Delta^{k,n})^{(\frac{k+1}{2})}(b) = 0.$$

The periodic spline $s_\Delta^{k,p}$ is defined for any k by the additional $k-1$ conditions

$$(s_\Delta^{k,p})'(a) = (s_\Delta^{k,p})'(b), \ldots, (s_\Delta^{k,p})^{(k-1)}(a) = (s_\Delta^{k,p})^{(k-1)}(b)$$

and is used only when $s_\Delta^{k,p}(a) = y_0 = y_n = s_\Delta^{k,p}(b)$.

Example 2.57 continued. The natural cubic spline $s_\Delta^{3,n}$

Theorem 2.60 *If $f \in \mathcal{C}^2([a,b])$ then*

$$\int_a^b (s_\Delta^{3,n})''(x)^2\,dx \le \int_a^b f''(x)^2\,dx \tag{2.38}$$

i.e. The natural cubic spline has at most the same curvature as the interpolated function f.

Proof:
Let $g = f - s_\Delta^{3,n}$ and note that $g(x_i) = 0$ for $i = 0,\dots,n$ since $s_\Delta^{3,n}$ interpolates f in all the nodal points. Then

$$\int_a^b f''(x)^2\mathrm{d}x = \int_a^b (s_\Delta^{3,n})''(x)^2\mathrm{d}x + \underbrace{\int_a^b g''(x)^2\mathrm{d}x}_{\ge 0} + 2\underbrace{\int_a^b (s_\Delta^{3,n})''(x)g''(x)\mathrm{d}x}_{=A}$$

where

$$\begin{aligned} A &= \sum_{i=1}^n \int_{x_{i-1}}^{x_i} (s_\Delta^{3,n})''(x)g''(x)\mathrm{d}x \\ &= \sum_{i=1}^n \Bigg(\underbrace{[(s_\Delta^{3,n})''(x)g'(x)]_{x_{i-1}}^{x_i}}_{\substack{=0 \text{ (telescoping sum with first and last term } = 0 \\ \text{(natural spline property) and } s''g' \text{ continuous)}}} - \underbrace{\int_{x_{i-1}}^{x_i} \underbrace{(s_\Delta^{3,n})'''(x)}_{=c_i \text{ (const.)}}\, g'(x)\mathrm{d}x}_{=c_i[g(x)]_{x_{i-1}}^{x_i}=0} \Bigg) \end{aligned}$$

proving the theorem. ∎

Now we return to the important question of existence and uniqueness. We shall only prove the following special case:

Theorem 2.61 *The natural cubic spline exists and is unique.*

Proof:
To simplify notation we shall in the proof use the notation s for $s_\Delta^{3,n}$ and s_i for $s_\Delta^{3,n}|_{\Delta_i}$ where i is always any integer between 1 and n when referring to a subinterval and between 0 and n when referring to a nodal point.
The proof is constructive and starts from $s_i'' \in \mathcal{P}_1(\Delta_i)$, $s_i''(x_{i-1}) = z_{i-1}$ and $s_i''(x_i) = z_i$.
Here the z_i are just place holders (unknowns apart from $z_0 = z_n = 0$ coming from the natural spline conditions) that allows us an appropriate way of insuring $s'' \in \mathcal{C}^0$ and the explicit expression $s_i''(x) = \frac{z_i}{h_i}(x - x_{i-1}) + \frac{z_{i-1}}{h_i}(x_i - x)$.
Now integrate the expression for s_i twice to get $s_i(x) = \frac{z_i}{6h_i}(x - x_{i-1})^3 + \frac{z_{i-1}}{6h_i}(x_i - x)^3 + C_i(x - x_{i-1}) + D_i(x_i - x)$ where the last two terms makes up

the "arbitrary linear function" coming from the two times integration.
Enforcing the interpolation conditions $s_i(x_{i-1}) = y_{i-1}$ and $s_i(x_i) = y_i$, at the same time insuring $s \in \mathcal{C}^0$, C_i and D_i can be eliminated using the known nodal point values y_{i-1} and y_i. We have
$y_{i-1} = \frac{z_{i-1}h_i^2}{6} + D_i h_i, \quad y_i = \frac{z_i h_i^2}{6} + C_i h_i \quad \Leftrightarrow \quad D_i = \frac{y_{i-1}}{h_i} - \frac{z_{i-1}h_i}{6}, \quad C_i = \frac{y_i}{h_i} - \frac{z_i h_i}{6}$. Now we only miss using the condition $s' \in \mathcal{C}^0$. For this we differentiate above and get
$s_i'(x) = \frac{z_i}{2h_i}(x - x_{i-1})^2 - \frac{z_{i-1}}{2h_i}(x_i - x)^2 + C_i - D_i \Rightarrow s_i'(x_{i-1}) = -\frac{z_{i-1}h_i}{2} + C_i - D_i,\ s_i'(x_i) = \frac{z_i h_i}{2} + C_i - D_i$.
Because of the smoothness of s_i, continuity of s' is equivalent to the internal nodal point conditions $s_i'(x_i) = s_{i+1}'(x_i)$, for $i = 1, \ldots, n-1$. $s_{i+1}'(x_i)$ we get from the expressions above by considering $s_i'(x_{i-1})$ and changing i to $i+1$. Hence $s_i'(x_i) = \frac{z_i h_i}{2} + C_i - D_i = s_{i+1}'(x_i) = -\frac{z_i h_{i+1}}{2} + C_{i+1} - D_{i+1}$, for $i = 1, \ldots, n-1$. Inserting the expressions for C_i and D_i from above, multiplicating by 6 and collecting coefficients of the z_i's we get $h_i z_{i-1} + 2(h_i + h_{i+1})z_i + h_{i+1}z_{i+1} = 6(\frac{y_{i-1}}{h_i} - \frac{y_i}{h_i} - \frac{y_i}{h_{i+1}} + \frac{y_{i+1}}{h_{i+1}}) =: 6v_i$, for $i = 1, \ldots, n-1$ or in matrix form

$$\begin{bmatrix} 2(h_1+h_2) & h_2 & & & & 0 \\ h_2 & 2(h_2+h_3) & h_3 & & & \\ & h_3 & 2(h_3+h_4) & h_4 & & \\ & & \ddots & \ddots & \ddots & \\ 0 & & & h_{n-1} & 2(h_{n-1}+h_n) & \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_{n-1} \end{bmatrix} = \begin{bmatrix} 6v_1 - h_1 z_0 \\ 6v_2 \\ 6v_3 \\ \vdots \\ 6v_{n-1} - h_n z_n \end{bmatrix}$$

Note that $z_0 = z_n = 0$. They have only been included on the left for completeness.
The coefficient matrix above is symmetric, tridiagonal and diagonally dominant (the absolute value of any diagonal element is bigger than the sum of the absolute values of the off diagonal elements in the same row). The diagonal dominance implies (see for example [5] §4.3) that there exist unique z_i's that can be found for example with Gaussian elimination without pivoting. The reference also provide an algorithm for the solution for the special case of tridiagonal matrices that we are facing here.
This proves the result. ■■

Example 2.62 The tension spline s_Δ^τ
As a final example we consider the so called tension spline (which is by the way not a spline) defined as follows:

1. $s_\Delta^\tau \in \mathcal{C}^2([a,b])$,
2. $s_\Delta^\tau(x_i) = y_i$ for $i = 0, \ldots, n$,

3. $(s_\Delta^\tau)^{(4)} - \tau^2 (s_\Delta^\tau)'' \equiv 0$ on Δ_i for $i = 1, \ldots, n$.

The first two conditions are consistent with a cubic spline while the third condition is different from the condition $s \in \mathcal{P}_3(\Delta_i)$ of a cubic spline. This is only apparent however, for the special case of the no tension tension spline $\tau = 0$ where the third condition turns into $(s_{\Delta_i}^{\tau=0})^{(4)} \equiv 0 \Leftrightarrow s_\Delta^{\tau=0}|_{\Delta_i} \in \mathcal{P}_3(\Delta_i)$ for $i = 1, \ldots, n$. This is the missing condition of the cubic spline so that $s_\Delta^{\tau=0}$ is a cubic spline. Turning to the very high tension tension spline $\tau \simeq \infty$ the third condition becomes (approximately) $(s_\Delta^{\tau\simeq\infty})'' \equiv 0 \Leftrightarrow s_\Delta^{\tau\simeq\infty}|_{\Delta_i} \in \mathcal{P}_1(\Delta_i)$ for $i = 1, \ldots, n$. Hence $s_\Delta^{\tau\simeq\infty}$ is almost a linear spline (apart from the stricter requirements of global smoothness.
Tension splines is a simplified example of Computer Assisted Design (CAD), where the designer of (in the following case) a car has some predefined design specifications like the length of the passenger cabin, the size of the trunk and the engine room defined by a data set. By varying the tension of a tension spline interpolating the data set he can vary the design from a "volvo like" car to a "General Motors like" one as shown in figure 2.10. In figure 2.10 is shown a linear and a natural cubic spline fitting a given data set. To see actual tension splines (that are indistinguishable from the ones shown here, see [5] §6.4 where the tension splines with tensions $\tau = 10$ and $\tau = 0.1$ are shown. ■

Exercise 2.63
Interpolation is useful also in 2 or more dimensions. Go to the library, find some literature about higher dimensional interpolation and write an essay about the topic. ■

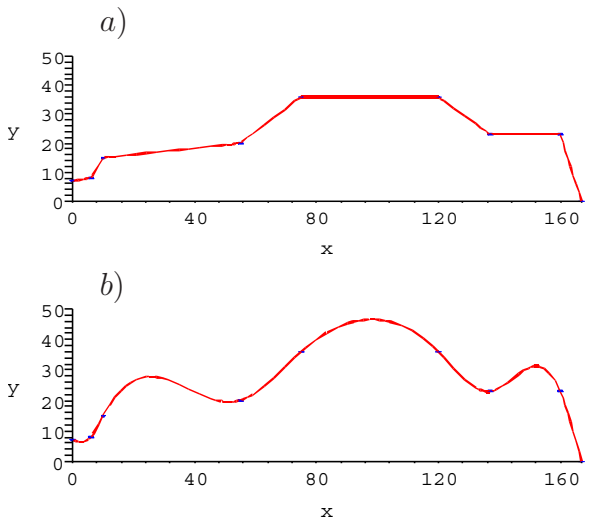


Figure 2.10: Example of car design using the tension spline. "From Volvo to General Motors": (a) Linear spline simulating a tension spline with high tension $\tau \simeq \infty$. (Actually $\tau = 10$ is sufficiently high to be indistinguishable in the resolution shown). (b) Natural cubic spline simulating a tension spline with low tension $\tau = 0$. ($\tau = 0.1$ is sufficiently small to be indistinguishable in the resolution shown). The data set is shown with blue dots.

Chapter 3

Numerical methods for DEP's

3.1 Numerical methods of type FDM, CM and FEM for differential equation problems

There are basically 3 types of methods for the numerical solution of DEP's: Finite Difference Methods (FDM's), Collocation Methods (CM's) and Finite Element Methods (FEM's). For an elementary introduction see for example [4] ch. 10-13. For many standard linear DEP's, the three methods are mathematically equivalent, in the sense that given a numerical method of one type, there exist methods of the other two types leading to exactly the same equation systems and hence to the same solutions (up to rounding errors). (See for example [6], [7] and [8]). As soon as the problems become a little bit non standard (read: real life like) the methods start showing their differences however, and hence it is important to be familiar with the strengths and weaknesses of them all.

To avoid drowning in notation, let us consider the main ideas of the 3 methods for the following simple one dimensional BVP (assumed to be well-posed):

$$\text{(3.1)}\quad \text{Find } u \in \mathcal{C}^2([0,1]): \quad -u''(x) = f(x)\ \forall x \in]0,1[, \quad u(0) = u'(1) = 0,$$

for some appropriate, known data function f.

To solve (3.1) we basically have to look through the infinite dimensional space $\mathcal{C}^2([0,1])$ until we find a function that satisfies the differential equation and the boundary conditions. Since software based on the standard programming languages like C, C++, Fortran, Pascal, Java, Lisp etc. is unable to make complete searches in infinite dimensional environments, the *job* of

any numerical method for DEP's is to replace the infinite seach for a solution with a finite one, which can be completed in finite time by software written in any standard language. This replacement process is called a Discretization, and the various types of numerical methods for DEP's are distinguished by the different general approaches they take to the discretization process.

Definition 3.1 *A* numerical method for DEP's *is a* discretization *of the DEP, i.e. a process which in a finite number of operations on a computer leads to an approximation to the solution to the DEP.*
There are 3 main types of discretizations of DEP's in practical use today, namely finite difference methods (FDM's), collocation methods (CM's) *and* finite element methods (FEM's).

Let us consider the approaches for the 3 types of numerical methods for solving DEP's examplified by the BVP (3.1).

3.1.1 Finite Difference Methods — FDM's

The main idea of all FDM's is to discretize by considering the differential equation only in a finite number of points instead of in the whole interval $(0, 1)$, i.e.

- consider the equations in (3.1) only for x in a finite set of Nodal Points $0 = x_1 < \ldots < x_n = 1$, i.e.

$$-u''(x_i) = f(x_i),\ i = 1, \ldots, n,\ u(x_1) = u'(x_n) = 0.$$

  (The Nodal Point Instances of (3.1)).

Note that 0 and 1 are nodal points to ensure that the boundary conditions are nodal point instances of (3.1). Note also that while we in chapter 2 used $n+1$ nodal points for approximation, here we use only n nodal points in the discretized problem. The most common example is to consider a uniform subdivision taking $x_i = (i-1)h$ for $h = 1/(n-1)$, so that the nodal point instances become

$$-u''(\frac{i-1}{n-1}) = f(\frac{i-1}{n-1}),\ i = 1, \ldots, n,\ u(0) = u'(1) = 0.$$

Since this does not really simplify the notation at this point, we shall stick to the x_i's for now. To simplify notation, instead we shall use the short hand notation $g(x_i) = g_i$ for any nodal point x_i and any function g. Hence, we can rewrite the nodal point instances as

$$-u_i'' = f_i,\ i = 1, \ldots, n,\ u_1 = u_n' = 0.$$

Instead of trying to recover the solution u to (3.1) in all points x in the domain $[0,1]$ we shall be satisfied finding u in the nodalpoints $x_1,\ldots,x_n$, i.e. $u_1,\ldots,u_n$. Hence we

- consider $u_i := u(x_i)$ for $i = 1,\ldots,n$ the unknowns in the nodal point instances of (3.1).

The mathematical definition of $u'(x_i)$ and $u''(x_i)$ contain limit processes involving points in a neighborhood of x_i:

$$u'(x_i) = \lim_{h\to 0} \frac{u(x_i+h)-u(x_i)}{h},$$

$$u''(x_i) = \lim_{h\to 0} \frac{u(x_i-h)-2u(x_i)+u(x_i+h)}{h^2}.$$

Hence $u'(x_i)$ and $u''(x_i)$ depend on u in a neighborhood of x_i, which is no good since we allow only values $u(x_i)$ as unknowns. Instead we need to replace the derivatives with some approximations that only depend on the unknowns $u_1,\ldots,u_n$. The most immediate choice might be simply "stopping" the limiting processes before h reaches zero and approximate as follows

$$u'(x_i) \simeq \frac{u(x_{i+1})-u(x_i)}{x_{i+1}-x_i},$$

$$u''(x_i) \simeq \frac{u(x_{i-1})-2u(x_i)+u(x_{i+1})}{((x_{i+1}-x_{i-1})/2)^2}.$$

As we shall see later on, many other replacements are possible however. Any approximation to a derivative, depending only on the nodal point values, are denoted a difference operator and written as δ with a super script denoting the order of the derivative being approximated (1 is typically omitted though) and a subscript distinguishing the particular approximation. For example

$$\delta_- u_{i+1} = \frac{u(x_{i+1})-u(x_i)}{x_{i+1}-x_i},$$

$$\delta_0^2 u_i = \frac{u(x_{i-1})-2u(x_i)+u(x_{i+1})}{((x_{i+1}-x_{i-1})/2)^2}.$$

Then the next step in the finite difference method is to

- replace the derivatives in all the nodal point instances of (3.1) generated above with difference operators, which for the examples of difference operators given above results in

$$-\delta_0^2 u_i \simeq f_i,\ i = 1,\ldots,n,\ u_1 = 0, \delta_- u_n \simeq 0.$$

Note the "approximately equal signs" $\simeq$ because the difference operators are not equal to but only approximating the derivatives. In order to get back to equalities that we can hand over to the computer to solve, at this point we have to give up even the recovery of the solution values in the nodal points. Instead we shall be satisfied recovering approximations $\tilde{U}_1, \ldots, \tilde{U}_n$ to $u_1, \ldots, u_n$. Hence

- replace the $\simeq$ by $=$ and define $\tilde{U}_i$ for $i = 1, \ldots, n$ as the solutions to the resulting equations

$$-\delta_0^2 \tilde{U}_i = f_i, \ i = 1, \ldots, n, \ \tilde{U}_1 = \delta_- \tilde{U}_n = 0.$$

A problem here is that the definition of $-\delta_0^2 \tilde{U}_0$ involves the undefined $\tilde{U}_0$ (which should be some approximation to u in the undefined nodal point x_0). Likewise $-\delta_0^2 \tilde{U}_n$ involves the undefined $\tilde{U}_{n+1}$ and implicitly the undefined nodal point x_{n+1}. Instead all the other $-\delta_0^2 \tilde{U}_i$, for $i = 2, \ldots, n-1$ only involve known $\tilde{U}_i$'s. Undefined variables and points like these are popularly denoted ghost values and ghost points. The finite difference method solution to this problem is simply to

- discard all equations that involve ghost variables, resulting in

$$\text{(3.2)} \quad \text{Find } \tilde{\mathbf{U}} \in \mathcal{R}^n : \ -\delta_0^2 \tilde{U}_i = f_i, \ i = 2, \ldots, n-1, \ \tilde{U}_1 = 0, \ \delta_- \tilde{U}_n = 0.$$

The fact that we have n equations for the n unknowns $\tilde{U}_i$, $i = 1, \ldots, n$ is good news. Still some essential questions are left unanswered: Does (3.2) have a unique solution? and if so are $\tilde{U}_i$ a good approximation to u_i for all $i = 1, \ldots, n$? We shall return to these questions in detail below.

Omitting most details, the general idea in the finite difference methods can now be summed up as follows

Definition 3.2 *A* finite difference method for a DEP *is constructed as follows*

- *Consider the DEP only in a finite set of nodal points, containing in particular all points where there are defined additional conditions (boundary or initial conditions).*
- *Replace all derivatives with finite difference operators (at the same time replacing many $=$ by $\simeq$.*
- *Replace the $\simeq$ by $=$ again, at the "cost" of at same time also replacing the exact solution values in the nodal points u_i by approximations $\tilde{U}_i$.*

- *Discard all equations containing ghost values or ghost nodes.*

Defining the i'th Step Length $h_i = x_{i+1} - x_i$ and letting $f(x_i) = f_i$ for $i = 1, \ldots, n$, (3.2) can, after multiplication by $((h_i + h_{i-1})/2)^2$ for $i = 2, \ldots, n-1$, be written as the following n dimensional, linear equation system:

(3.3) Find $\tilde{\mathbf{U}} = (\tilde{U}_1, \ldots, \tilde{U}_n)^T \in \mathcal{R}^n$:

$$\begin{bmatrix} 1 & & & & & & 0 \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & -1 & 2 & -1 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & -1 & 2 & -1 \\ 0 & & & & & -1 & 1 \end{bmatrix} \begin{bmatrix} \tilde{U}_1 \\ \vdots \\ \tilde{U}_n \end{bmatrix} = \begin{bmatrix} 0 \\ ((h_2 + h_1)/2)^2 f_2 \\ ((h_3 + h_2)/2)^2 f_3 \\ \vdots \\ ((h_{n-1} + h_{n-2})/2)^2 f_{n-1} \\ 0 \end{bmatrix}.$$

Here the first equation comes from the left boundary condition $\tilde{U}_1 = 0$ while the last one comes from right boundary condition $\delta_- U_n = 0 \Leftrightarrow \tilde{U}_n - \tilde{U}_{n-1} = 0$. The reason for the multiplication by $((h_i + h_{i-1})/2)^2$ is that it makes all elements in the matrix "of the order of one". Such a Well scaled matrix is "good" for numerical solution methods like Gaussian elimination

Problem (3.3) (or equivalently (3.2)) defines a particular FDM. There are many possible FDM's for a given DEP. Various FDM's are distinguished mainly by which difference operators are used for the various derivatives, but also direct changes in the differential equations may be involved in more complicated cases. To actually use a given FDM, the step lengths $\{h_i\}_{i=1}^{n-1}$ must be selected. The selection of step lengths, as well as the selection of the difference operators distinguishing the particular FDM is done in order to obtain an acceptable Error Vector $\mathbf{e}$, with components $e_i = |u_i - \tilde{U}_i|$, $i = 1, \ldots, n$ and error $|e| = \|e\|$ for some vector norm $\|\cdot\|$ like $\|e\|_1 = \sum_{i=1}^n e_i$ or $\|e\|_2 = \sqrt{\sum_{i=1}^n e_i^2}$. There will be much more about this issue below. Here only a few introductory remarks: Generally the size of the error measured in any reasonable vector norm will decrease with increasing precision of the difference operators and with decreasing step lengths. Often (generally whenever computationally feasible) a Uniform Step length $h = h_i = \frac{1}{n-1}$, $i = 1, \ldots, n-1$ is used to simplify programming and to optimize the observed order of convergence of the method. $\{x_i\}_{i=1}^n = \{(i-1)h\}_{i=1}^n$ then determines a Uniform subdivision of or Uniform Grid on the Computational

Domain $(0,1)$, simplifying (3.3) to

(3.4) Find $\tilde{\mathbf{U}} = (\tilde{U}_1, \ldots, \tilde{U}_n)^T \in \mathcal{R}^n$:

$$\begin{bmatrix} 1 & & & & & & 0 \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & -1 & 2 & -1 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & -1 & 2 & -1 \\ 0 & & & & & -1 & 1 \end{bmatrix} \begin{bmatrix} \tilde{U}_1 \\ \vdots \\ \tilde{U}_n \end{bmatrix} = \begin{bmatrix} 0 \\ h^2 f_2 \\ h^2 f_3 \\ \vdots \\ h^2 f_{n-1} \\ 0 \end{bmatrix}.$$

Here is a piece of Maple code for programming and plotting (3.4) in the special case $f(x) = x$:

```
> restart;
> with(LinearAlgebra): with(plots):
> f:=x->x;
> exsol:=x->x/2-x^3/6;
> explot:=plot(exsol(x),x=0..1,color=blue);
> FDM:=proc(n,f) local x,i,h,A,B,U,points;
    x:=Vector(n);
    h:=1/(n-1);
    for i to n do x[i]:=(i-1)*h end do;
    A:=Matrix(n);
    for i from 2 to n-1 do A[i,i]:=2; A[i,i-1]:=-1; A[i,i+1]:=-1 end do;
    A[1,1]:=1; A[n,n]:=1; A[n,n-1]:=-1;
    # print(A);
    B:=Vector(n);
    for i from 2 to n-1 do B[i]:=h^2*f(x[i]) end do;
    # print(B);
    U:=Vector(n);
    U:=LinearSolve(A,B);
    points:=zip((a,b)->[a,b],x,U);
    pointplot(points,symbol=point);
  end proc;
> display(FDM(500,f),FDM(100,f),FDM(10,f),FDM(4,f),explot);
```

Exercise 3.3
Program (3.4) for general n and for uniform step lengths and with $f(x) = x$. The output from the program should be a graph of the numerical solution,

i.e. the points $\{(x_i, \tilde{U}_i)\}_{i=1}^n$ connected with line pieces (a linear spline). The graph should also show the analytic solution $u(x) = \frac{x}{2} - \frac{x^3}{6}$.
Run your program for $n = 2, 4, 8, 16, \ldots$ until you see no further progress. (This is referred to as "convergence in the eye ball norm").
At what uniform step length h did you decide that you had convergence in the eye ball norm?
For all h values you have computed up to convergence in the eye ball norm, compute the error in the discrete ℓ^∞ and ℓ^1 norms $\|e\|_{\ell^\infty} = \max_{i=1,\ldots,n} |u(x_i) - \tilde{U}_i|$ and $\|e\|_{\ell^1} = \sum_{i=1}^n |u(x_i) - \tilde{U}_i|$ respectively. Comment on what you see, for example on which of the two norms are "closer" to the eye ball norm? ■

For more general problems in more than one dimension, the finite difference approach is entirely equivalent. Derivatives are replaced by finite differences, with approximate solutions defined over a grid of nodal points. To get optimal orders of the difference operators the grid should be a tensor product of uniform subdivisions. For example in 2 space dimensions, the set of nodal points should be of the form (ih, jk), $i = 0, \ldots, n$, $j = 0, \ldots, m$, corresponding to uniform step lengths h and k in the 1st and 2nd coordinate directions respectively ■ . The issues of error are also similar in one and more dimensions, only harder in more dimensions.

A general advantage of the finite difference approach to discretization of DEP's is that it is easy to understand since it involves no more than the replacement of derivatives with difference operators. This advantage can also be a pitfall, since it turns out that there is more to the notion of error than just the precision of the difference operators and the size of the step lengths. Once the difference operators have been selected, it is also easy to implement an FDM either by programming the discretized problem (like (3.2)) or by directly programming the equation system (like (3.3)).

A general disadvantage of the finite difference approach is the difficulties arising when the computations take place over non rectangular domains in two or more dimensions or if non uniform step lengths are required. If for example a lot is happening to the solution to a given problem at a particular region of time and space, then it might be beneficial to use small step lengths there and bigger step lengths in other regions where little is happening. Unfortunately, this often leads to problems with the precision of the difference operators. If the step lengths have to be as small as required by the region of "highest demand" everywhere in the computational domain, this soon leads to problems with excessive computation times. This difficulty in using Non Uniform Subdivisions is probably the most serious drawback for the finite difference technology. It is possible to remedy the problem to some extent, but only at the expense of loosing the general advantages of ease of under-

standing and implementing mentioned above. For more on FDM's see for example [4], [9] and [10].

Exercise 3.4
Construct an FDM for (1.10) and write out the equation system corresponding to (3.3). Consider the case $I = (0, 4)$, $f(x, u(x)) = \cos(x)$, $x_0 = 0$ and $u^* = 0$. Program your method, solve and compare the numerical solutions graphically to the exact solution $u(t) = \sin(t)$ like in exercise 3.3. ■

3.1.2 Collocation Methods — CM's

The main idea in the CM's is to replace the exact solution function $u \in \mathcal{C}^2(0,1)$ in (3.1) with a function $\tilde{U}$ from a finite dimensional space $\mathcal{S} \subset \mathcal{C}^2(0,1)$. Then, $\tilde{U}$ not being the exact solution u, it is not possible for $\tilde{U}$ to satisfy (3.1) in all points, but it can satisfy it in a finite number of points. The procedure goes as follows:

- replace $\mathcal{C}^2([0,1])$ in (3.1) by a Discrete Solution Space (or Trial Space) $\mathcal{S} = \text{span}\{\phi_1, \ldots, \phi_n\} \subset \mathcal{C}^2([0,1])$, where $\mathcal{S}$ is of finite dimension n, and $\phi_j(0) = \phi_j'(1) = 0$, for $j = 1, \ldots, n$.
- replace u by $\tilde{U} = \sum_{j=1}^n c_j \phi_j$ with the n unknowns $\{c_j\}_{j=1}^n$ in (3.1), i.e.

$$-\tilde{U}''(x) = f(x), \ \forall x \in]0,1[\Leftrightarrow -\sum_{j=1}^{n} c_j \phi_j''(x) = f(x), \ \forall x \in]0,1[. \tag{3.5}$$

(3.5) normally has no solution since we have an infinite number of equations (one for each $x \in]0,1[$) while we have only n unknowns. With the n unknowns we need n equations. The solution chosen is to

- satisfy (3.5) only in a set of Nodal Points $0 \leq x_1 < \ldots < x_n \leq 1$, i.e.

$$\text{Find } \mathbf{c} \in \mathcal{R}^n : \ -\sum_{j=1}^{n} c_j \phi_j''(x_i) = f(x_i), \ i = 1, \ldots, n, \tag{3.6}$$

or in matrix form

$$\text{Find } \mathbf{c} \in \mathcal{R}^n : \ \mathbf{Bc} = \mathbf{f}, \tag{3.7}$$

where $B_{ij} = -\phi_j''(x_i)$ and $f_i = f(x_i)$, $i, j = 1, \ldots, n$.

The various CM's are distinguished by how the discrete solution space $\mathcal{S}$ is chosen. One possibility is to take $\mathcal{S} = \mathcal{P}^0_{n+1}[0,1]$, the space of polynomials p of degree at most $n+1$ with the super index 0 indicating that the two boundary conditions $p(0) = p'(1) = 0$ are enforced, hence reducing the dimension to n. As basis functions for $\mathcal{P}^0_{n+1}[0,1]$ can for example be used the following Lagrange like basis functions (Hermite cardinal like functions) where we have introduced the additional nodal point $x_0 = 0$ and hence have $x_1 > 0$. Also $x_n = 1$:

$$\phi_j(x) = \left(\prod_{k=0,k\neq j}^{n-1} \frac{x-x_k}{x_j-x_k}\right)\left(\frac{x-x_n}{x_j-x_n}\right)^2, \ j = 1,\ldots,n-1, \tag{3.8}$$

$$\phi_n(x) = \left(\prod_{k=0}^{n-1} \frac{x-x_k}{x_n-x_k}\right)\left(1 - \left(\prod_{k=0}^{n-1} \frac{x-x_k}{x_n-x_k}\right)'\Big|_{x=x_n}(x-x_n)\right),$$

This corresponds to $n+1$ nodes $0 = x_0 < x_1 < \ldots < x_n = 1$.

Exercise 3.5

Construct a CM for (3.1) and write out the equation system corresponding to (3.7) in matrix form as in (3.3). Write also the expression for the solution $\tilde{U}$. Use the basis functions from (3.8), in which case the matrix system is the hard part and the expression for $\tilde{U}$ the easy part. ■

Exercise 3.6

Program the method from exercise 3.5 for general n and for uniform step lengths and with $f(x) = x$. The output from the program should be a graph of the numerical solution, i.e. the points $\{(x_i, \tilde{U}_i)\}_{i=1}^n$ connected with line pieces (a linear spline). The graph should also show the analytic solution $u(x) = \frac{x}{2} - \frac{x^3}{6}$.

Run your program for $n = 2, 4, 8, 16, \ldots$ until you see no further progress. (This is referred to as "convergence in the eye ball norm").

At what uniform step length h did you decide that you had convergence in the eye ball norm?

For all h values you have computed up to convergence in the eye ball norm, compute the error in the discrete ℓ^∞ and ℓ^1 norms $\|e\|_{\ell^\infty} = \max_{i=1,\ldots,n} |u(x_i) - \tilde{U}_i|$ and $\|e\|_{\ell^1} = \sum_{i=1}^n |u(x_i) - \tilde{U}_i|$ respectively. Comment on what you see, for example on which of the two norms are "closer" to the eye ball norm? ■

Instead of using a polynomial space for $\mathcal{S}$, the computational domain can be divided into small pieces called Elements corresponding to the subdivision of the FDM's determined by the nodal points. In each element, the CM

is using a local function space, normally polynomials of degree no greater than p for some integer p which is normally 3 or 4, but in the so called Spectral Collocation Methods (SCM's) may be much higher (like 8 or 16). In between the elements, the local function spaces are "glued" together by some global smoothness/Regularity conditions (like splines). Recall that to get $\mathcal{S} \subset \mathcal{C}^2(0,1)$ we need a global smoothness of at least $\mathcal{C}^2$ which for splines is provided only for the cubic or higher degree ones. Finally some or all of the boundary conditions are enforced. Hence $\mathcal{S}$ is described by 3 properties:

- Global Smoothness. Example: $v \in \mathcal{C}^0[0,1]$ ■ .
- Local form. Example: $v|_{(x_i,x_{i+1})}$ is a linear polynomial for $i = 0, \ldots, n$. Note that we have added two nodes to make up for the dimensions "lost" because of the boundary conditions below. This would normally be accomplished by taking $0 = x_0 < x_1 < \ldots < x_n < x_{n+1} = 1$, but it is also possible to use a set of nodal points completely different from the one introduced above for (3.6) ■ .
- boundary conditions. Example: $v(0) = 0$, $v'(1) = 0$ ■ .

Example: A Lagrange Basis for $\mathcal{S}$, given by

$$\phi_j \in \mathcal{S} \text{ and } \phi_j(x_i) = \delta_{ij} = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{else} \end{cases}, \text{ for } i, j = 1, \ldots, n \tag{3.9}$$

is indicated in figure 3.1 for the example selections above ■ . Note that this

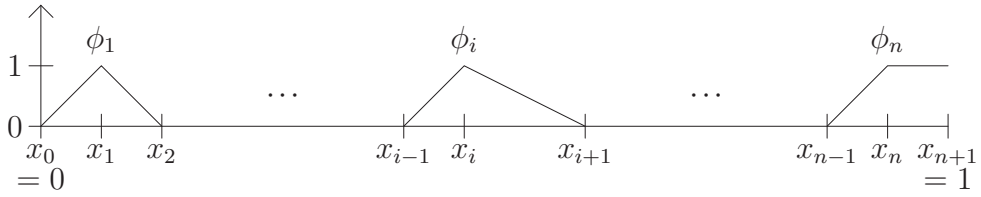


Figure 3.1: Lagrange basis for the CM example above

example is for illustrative purposes only. In reality global smoothness $\mathcal{C}^0$ is too little for $\mathcal{S}$ to be a subset of $C^2(0,1)$ but instead splines of degree at least 3 could be used. Denoting by $\mathcal{S}^{k,k}_{\Delta(n)}$ the space of splines of degree k relative to n distinct nodes, then it is well known that $\dim(\mathcal{S}^{k,k}_{\Delta(n)}) = n + k - 1$. (See exercise 2.59 on page 54 just replacing n by $n-1$ since there are $n+1$ nodal points in exercise 2.59 and only n here). Letting $\sigma^{k,k}_{\Delta(n)}$ be $\mathcal{S}^{k,k}_{\Delta(n)}$ with the two boundary conditions $u(0) = 0$ and $u'(1) = 0$ enforced, we have $\dim(\sigma^{k,k}_{\Delta(n)}) = n + k - 3$ and hence $\dim(\sigma^{k,k}_{\Delta(n+3-k)}) = n = \dim(\mathcal{S})$. Hence for $k = 3$ (cubic splines) we can take as the n nodes, the nodal points $x_1, \ldots, x_n$

introduced above for (3.6). For $k > 3$ we would need only $n + 3 - k < n$ nodes so that the set of nodes for the spline space can not be identical to the set of nodal points for the CM from (3.6). Taking $k = 3$ this leads to the following example, which is more realistic than the one above:

- Global Smoothness. Example: $v \in \mathcal{C}^2[0,1]$ ■ .
- Local form. Example: $v|_{(x_i,x_{i+1})}$ is a cubic polynomial for $i = 1, \ldots, n-1$. Here we take $0 = x_1 < \ldots < x_n = 1$ in order to be able to enforce the boundary conditions below ■ .
- boundary conditions. Example: $v(x_1) = v(0) = 0$, $v'(x_n) = v'(1) = 0$ ■ .

The basis giving the simplest form to (3.6) is the Cardinal Basis given by

$$\phi_j \in \sigma^{3,3}_{\Delta(n)} \text{ and } \phi_j''(x_i) = \delta_{ij} = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{else} \end{cases}, \text{ for } i, j = 1, \ldots, n. \tag{3.10}$$

Using the cardinal basis, the matrix $\mathbf{B}$ in (3.7) becomes simply minus the identity matrix. On the other hand, the construction of the cardinal basis is somewhat complicated.

The traditional way to recover basis functions for spline spaces is to use B-splines. For the basic theory of B-splines see for example [5]. Here we shall only give the most central results for the special case of cubic splines and uniform subdivisions $x_1, \ldots, x_n$ of $(0,1)$ with $x_i = \frac{i-1}{n-1}$, $i = 1, \ldots, n$: $\{B^3_{-3}, \ldots, B^3_{n-2}\}$ is a basis for $\mathcal{S}^{k,k}_{\Delta(n)}$. Here B^3_i is a B-spline with support in the interval (x_{i+1}, x_{i+5}). Note that the support of the basis functions stretches over the interval (x_{-2}, x_{n+3}). The additional points are defined with the same formula as the regular nodal points i.e. $x_i = \frac{i-1}{n-1}$, $i = -2, \ldots, n+3$. B^3_i can be computed with Maple using the commands

```
with(CurveFitting):
BSpline(4,x,[x_{i+1},x_{i+2},x_{i+3},x_{i+4},x_{i+5},]);
```

and using Maple notation a basis for $\mathcal{S}^{k,k}_{\Delta(n)}$ is then
$\{\text{BSpline}(4, x, [x_{-2}, \ldots, x_2]), \ldots, \text{BSpline}(4, x, [x_{n-1}, \ldots, x_{n+3}]\}$.
To recover a basis for $\sigma^{k,k}_{\Delta(n)}$, a small investigation shows that all but the first 3 basis functions above take the value zero in 0. The first 3 basis functions B^3_{-3}, B^3_{-2}, and B^3_{-1} take the values $B^3_{-3}(0) = \frac{1}{6}$, $B^3_{-2}(0) = \frac{2}{3}$ and $B^3_{-1} = \frac{1}{6}$. Hence, the condition $u(0) = 0$ may be enforced by replacing B^3_{-3}, B^3_{-2}, and B^3_{-1} by the 2 linearly independent functions $B^3_{-3} - B^3_{-1}$ and $2(B^3_{-3} + B^3_{-1}) - B^3_{-2}$ both taking the value zero at 0. Correspondingly B^3_{n-3} and $B^3_{n-4} +$

B^3_{n-2} are 2 linearly independent functions constructed from B^3_{n-4}, B^3_{n-3} and B^3_{n-2} having first derivative equal 0 in 1, and all other basis functions satisfy this property "from birth". Hence a basis for $\sigma^{k,k}_{\Delta(n)}$ is $\{B^3_{-3} - B^3_{-1}, B^3_{-3} + B^3_{-1} - B^3_{-2}, B^3_0, \ldots, B^3_{n-5}, B^3_{n-3}, B^3_{n-4} + B^3_{n-2}\}$. Here is a sequence of Maple commands, which for general n (here selected to $n = 4$ in the third command) define the necessary nodal points, define and plot the $\mathcal{S}^{3,3}_{\Delta(n)}$ basis functions denoted $f_1, \ldots, f_{n+2}$ and investigate the relevant function values in 0 and derivatives in 1, define and plot the $\sigma^{3,3}_{\Delta(n)}$ basis functions denoted $\phi_1, \ldots, \phi_n$ and finally compute the matrix B from (3.7):

```
> restart:
> with(CurveFitting):
> n:=4:
> for i from -2 to n+3 do x[i]:=(i-1)/(n-1) end do:
> for i from -3 to n-2 do
    f[i+4]:=BSpline(4,x,knots=[seq(x[j],j=i+1..i+5)])
  end do:
> fseq:=seq(f[i],i=1..n+2):
> plot([fseq],x=-2..3);
> for i from 1 to n+2 do funf[i]:=unapply(f[i],x) end do:
> funf[1](0);funf[2](0);funf[3](0);
> D(funf[n])(1);D(funf[n+1])(1);D(funf[n+2])(1);
> phi[1]:=funf[1]-funf[3]:phi[2]:=2*(funf[1]+funf[3])-funf[2]:
> for i from 3 to n-2 do phi[i]:=funf[i+1] end do:
> phi[n-1]:=funf[n]+funf[n+2]:phi[n]:=funf[n+1]:
> phiseq:=seq(phi[i](x),i=1..n):
> plot([phiseq],x=-2..3);
> plot([phiseq],x=0..1);
> g:=(i,j)->-(D@@2)(phi[j])(x[i]): Matrix(n,g);
```

Exercise 3.7

Construct a CM for (3.1) and write out the equation system corresponding to (3.7) in matrix form as in (3.3). Write also the expression for the solution $\tilde{U}$. Use cubic splines as basis functions, in which case the matrix system is the hard part and the expression for $\tilde{U}$ the easy part. ■

Exercise 3.8

Program the method from exercise 3.7 for general n and for uniform step lengths and with $f(x) = x$. The output from the program should be a graph of the numerical solution, i.e. the points $\{(x_i, \tilde{U}_i)\}_{i=1}^n$ connected with line

pieces (a linear spline). The graph should also show the analytic solution $u(x) = \frac{x}{2} - \frac{x^3}{6}$.
Note that the matrix part has been done above. It only remains to copy the commands into Maple, implement the right hand side, the matrix solution and the plots and then try it all out.
Run your program for $n = 2, 4, 8, 16, \ldots$ until you see no further progress. (This is referred to as "convergence in the eye ball norm").
At what uniform step length h did you decide that you had convergence in the eye ball norm?
For all h values you have computed up to convergence in the eye ball norm, compute the error in the discrete ℓ^∞ and ℓ^1 norms $\|e\|_{\ell^\infty} = \max_{i=1,\ldots,n} |u(x_i) - \tilde{U}_i|$ and $\|e\|_{\ell^1} = \sum_{i=1}^n |u(x_i) - \tilde{U}_i|$ respectively. Comment on what you see, for example on which of the two norms are "closer" to the eye ball norm? ■

Exercise 3.9
Construct a CM for (3.1) and write out the equation system corresponding to (3.7) in matrix form as in (3.3). Write also the expression for the solution $\tilde{U}$. Use the cardinal basis from (3.10) as basis functions, in which case the matrix system is the easy part and the expression for $\tilde{U}$ the hard part. ■

Exercise 3.10
Program the method from exercise 3.9 for general n and for uniform step lengths and with $f(x) = x$. The output from the program should be a graph of the numerical solution, i.e. the points $\{(x_i, \tilde{U}_i)\}_{i=1}^n$ connected with line pieces (a linear spline). The graph should also show the analytic solution $u(x) = \frac{x}{2} - \frac{x^3}{6}$.
Run your program for $n = 2, 4, 8, 16, \ldots$ until you see no further progress. (This is referred to as "convergence in the eye ball norm").
At what uniform step length h did you decide that you had convergence in the eye ball norm?
For all h values you have computed up to convergence in the eye ball norm, compute the error in the discrete ℓ^∞ and ℓ^1 norms $\|e\|_{\ell^\infty} = \max_{i=1,\ldots,n} |u(x_i) - \tilde{U}_i|$ and $\|e\|_{\ell^1} = \sum_{i=1}^n |u(x_i) - \tilde{U}_i|$ respectively. Comment on what you see, for example on which of the two norms are "closer" to the eye ball norm? ■

Where the error for FDM's is the vector norm of an error vector, the error for CM's is the function norm of an error function $e = u - \tilde{U}$, i.e. $|e| = \|e\| = \|u - \tilde{U}\|$. Examples of $|e|$ are $|e| = \|e\|_1 = \int_0^1 |(u - \tilde{U})(x)| \mathrm{d}x$ or $|e| = \|e\|_2 = \sqrt{\int_0^1 (u - \tilde{U})^2(x) \mathrm{d}x}$.

Convergence is obtained in CM's by increasing the dimension of $\mathcal{S}$ either increasing the local polynomial degree p or reducing the size of the elements,

i.e. increasing n (or both). Another possibility for improving results is to keep the dimension of $\mathcal{S}$ fixed, but try to locate the nodes of the (spline) space $\mathcal{S}$ in a more optimal way. This is often done iteratively, moving the points around with small steps and keeping track of the error in the process. This relocation or r-method does *not* lead to convergence, but if the number of points is big enough, the error may be reduced significantly in the relocation process so that the optimal error is below a predefined tolerance.

The general theory for collocation methods for problems posed in more than one dimension is the same as for the one dimensional case. Only does the finite dimensional space consist of functions defined over a multidimensional domain. There is not the same requirement as for the finite difference methods to make uniform subdivisions. As long as it is possible to construct bases for the finite dimensional space with the selected subdivision things are OK.

The obvious advantage of the collocation approach is the ease of adapting the method to non rectangular domains and non uniform subdivisions.

The disadvantage of the collocation approach is that it is somewhat harder than the finite difference approach to understand. The understanding hinges on the understanding of function approximation, and convergence depends to some extent on the theory of projections (considering the differential equation only in a discrete set of points is some sort of projection). The practical implementation of a CM is also somewhat more complicated than that of a FDM, since it is necessary to implement the bases of the local function spaces and their derivatives. For more on CM's see for example [11].

Exercise 3.11
Construct a CM for (1.10) and write out the equation system corresponding to (3.3). Consider the case $I = (0, 4)$, $f(t, u(t)) = \cos(t)$, $t_0 = 0$ and $u^* = 0$. Program your method, solve and compare the numerical solutions graphically to the exact solution $u(t) = \sin(t)$ and to the numerical solutions found in exercise 3.4. For basis use the following modification of (3.8):

$$\phi_j(x) = \prod_{k=0, k\neq j}^{n} \frac{x - x_k}{x_j - x_k}, \ j = 1, \ldots, n. \tag{3.11}$$

Here $x_0 = 0$ and $x_n = 4$. ■

3.1.3 Finite Element Methods — FEM's

The finite element approach is closely related to the collocation approach described above, only is the projection different. Instead of considering the

differential equation for the approximate solution $\tilde{U}$ in n points, the whole equation is projected onto $\mathcal{S}$ with an $\mathcal{L}^2$-projection. The first two bullets in the CM description at the beginning of section 3.1.2 remain unchanged, only the third one is changed:

- replace $\mathcal{C}^2([0,1])$ in (3.1) by a Discrete Solution Space (or Trial Space) $\mathcal{S} = \text{span}\{\phi_1, \ldots, \phi_n\} \subset \mathcal{C}^2([0,1])$, where $\mathcal{S}$ is of finite dimension n, and $\phi_j(0) = \phi_j'(1) = 0$, for $j = 1, \ldots, n$.
- replace u by $\tilde{U} = \sum_{j=1}^n c_j \phi_j$ with the n unknowns $\{c_j\}_{j=1}^n$ in (3.1), i.e.

$$-\tilde{U}''(x) = f(x), \ \forall x \in]0,1[\Leftrightarrow -\sum_{j=1}^n c_j \phi_j''(x) = f(x), \ \forall x \in]0,1[. \tag{3.12}$$

(3.12) normally has no solution since we have an infinite number of equations (one for each $x \in]0,1[$) while we have only n unknowns. With the n unknowns we need n equations. The solution chosen is to

- satisfy only the projection of the equation onto $\mathcal{S}$:

$$\text{Find } \mathbf{c} \in \mathcal{R}^n : \ (-\sum_{j=1}^n c_j \phi_j'', \phi_i) = (f, \phi_i), \ i = 1, \ldots, n. \tag{3.13}$$

  Here $(\cdot,\cdot)$ is some sort of projection, normally the $\mathcal{L}^2$-inner product, i.e. $(f,g) = \int_0^1 f(x)g(x)\mathrm{d}x$.

The finite element approach can also be seen as an approximation to a so called Variational Formulation of the DEP (3.1), derived by multiplying the differential equation in (3.1) by a Test Function $v \in \mathcal{C}^2(0,1)$ satisfying $v(0) = 0$ (like u), and integrating on both sides over the domain $(0,1)$:

$$\text{Find } u \in \mathcal{C}^2(0,1) : \ \int_0^1 -u''v dx = \int_0^1 fv dx, \quad \forall v \in \mathcal{C}^2(0,1), \text{ with } v(0) = 0. \tag{3.14}$$

(This is called a variational formulation since v can vary freely in $\mathcal{C}^2(0,1)$ (apart from $v(0) = 0$)).
Now do partial integration to obtain symmetry in the order of derivatives on the test function v and the Solution or Trial Function u:

$$\text{Find } u \in \mathcal{V} : \ \int_0^1 u'v' dx = \int_0^1 fv dx \ \forall v \in \mathcal{V}. \tag{3.15}$$

($[-u'v]_0^1 = 0$ since $v(0) = u'(1) = 0$). Also, we have replaced the space $\mathcal{C}^2(0,1)$ by the bigger space $\mathcal{V}$ ($\mathcal{C}^2(0,1) \subset \mathcal{V} \subseteq \mathcal{H}^1(0,1)$) containing square integrable functions (in $\mathcal{L}^2(0,1)$) with (weak) first derivatives existing and lying in $\mathcal{L}^2(0,1)$. This is the Sobolev space $\mathcal{H}^1(0,1)$ in which the integrals in (3.15) make sense as long as also $f \in \mathcal{L}^2(0,1)$. (3.15) is like (3.14) a variational formulation, and since the bigger spaces generally make the convergence results easier to recover, we shall initiate our discretization process from there. The finite element discretization is similar to the collocation discretization, only starting from (3.15) instead of (3.1) and *not* enforcing both boundary conditions:

- Replace $\mathcal{V}$ in (3.15) by a finite dimensional Solution (or Trial) Space $\mathcal{S} = \text{span}\{\phi_1, \ldots, \phi_n\} \subset \mathcal{V}$, where $\phi_j(0) = 0$, for $j = 1, \ldots, n$.
- Replace u by $\tilde{U} = \sum_{j=1}^n c_j \phi_j$ with the n unknowns $\{c_j\}_{j=1}^n$ in (3.15) and let v be any of the basis functions $\phi_1, \ldots, \phi_n$, i.e.

$$\text{Find } \mathbf{c} \in \mathcal{R}^n : \int_0^1 \sum_{j=1}^n c_j \phi_j' \phi_i' dx = \int_0^1 f \phi_i dx \ \forall i = 1, \ldots, n, \tag{3.16}$$

  or in matrix form

$$\text{Find } \mathbf{c} \in \mathcal{R}^n : \ \mathbf{Bc} = \mathbf{q}, \tag{3.17}$$

  where $B_{ij} = \int_0^1 \phi_j' \phi_i' dx$ and $q_i = \int_0^1 f \phi_i dx$, $i, j = 1, \ldots, n$.

$u(0) = 0$ is called an Essential or Strong boundary condition. It is enforced explicitly in $\mathcal{V}$ and is hence *always* satisfied.
$u'(1) = 0$ is called an Natural or Weak boundary condition. It is only implicitly part of the variational formulation, since it is used to remove the boundary terms after the partial integration. It may or may not be satisfied exactly. Unlike the collocation method, the basis functions ϕ_j are not required to satisfy $\phi_j'(1) = 0$ for the FEM.

The discrete space $\mathcal{S}$ is constructed for the FEM's as for the CM's. Only is the degree of global smoothness needed for $\mathcal{V}$ generally much lower ($\mathcal{C}^0(0,1)$) than $\mathcal{C}^2(0,1)$. We repeat the 3 points determining the discrete space $\mathcal{S}$ and a typical example here:

- Global Smoothness. Example: $v \in \mathcal{C}^0[0,1]$ ■ .
- Local form. Example: $v|_{(x_i, x_{i+1})}$ is a linear polynomial for $i = 0, \ldots, n-1$. Note that we have introduced $n+1$ nodes and n elements to get $\dim(\mathcal{S}) = n$ (see figure 3.2) ■ .

- boundary conditions. Example: $v(0) = 0$ ■ .

Example: A Lagrange Basis for $\mathcal{S}$, given by

$$\phi_j \in \mathcal{S} \text{ and } \phi_j(x_i) = \delta_{ij} = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{else} \end{cases}, \text{ for } i, j = 1, \ldots, n \tag{3.18}$$

is indicated in figure 3.2 for the example selections above. To compute the

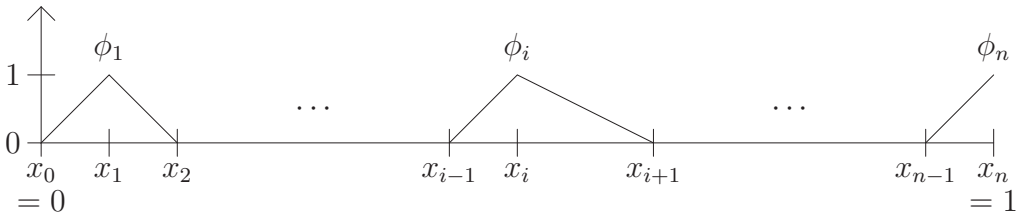


Figure 3.2: Lagrange basis for the FEM example above

integrals in B_{ij} and q_i in (3.17) we take advantage of the simple structure of the ϕ_i's: First of all, the support of ϕ_i covers at most 2 elements. Hence also the support of $\phi_i'\phi_j'$ and of $f\phi_i$ covers at most 2 elements. Since ϕ_i' is piecewise constant, B_{ij} is the sum of at most 2 integrals of constants; one for each of the intervals in the support. q_i is likewise the sum of at most 2 integrals over the intervals in the support of ϕ_i. Here the integrants may be more complex however, depending on the complexity of f. As a general rule of thumb, B_{ij} is easy to compute, while q_i may be hard. Luckily there are only n q_i's against n^2 B_{ij}'s ■ .

Convergence is obtained in FEM's like in CM's by either increasing the polynomial degree p of the local polynomial spaces over each element (the **p**-version of the FEM), by reducing the size h of the elements and hence increasing the number of elements (the **h**-version of the FEM), or possibly by doing both (the **hp**-version of the FEM). Whether this actually does lead to convergence depends on fairly heavy theoretical results from the area of "Interpolation theory in Banach spaces". As for the CM's also relocation of the nodes (the **r**-version of the FEM) is used to reduce the error below an a priori given acceptable treshold (the Tolerance). Also for the generalization to higher dimensional problems, obviously the comments for the collocation methods go through also for finite element methods.

It is considered an FEM advantage (among mathematicians) that the FEM is resting on a solid foundation of advanced mathematics like variational formulation of DEP's, Banach and Hilbert space theory, especially in Sobolev spaces, and interpolation theory. This allows existence, uniqueness and convergence results to be found relatively easily, without worries about stability and consistency like for the FDM's (see below). Practically, the

FEM's have the same advantages as the CM's: In contrast to the FDM's, the FEM's are (theoretically) no harder to compute with in non rectangular domains and if non uniform meshes are required than they are in rectangular domains with uniform meshes.

The disadvantages are also the same as for the CM's, and even more so, since we for the FEM's have to compute also projections involving (numerical) integration. Instead, because of the lower demands for global smoothness, generally the finite element bases are easier to construct and program. For more on FEM's see for example [12], [13], [14] and [15].

Exercise 3.12

Construct an FEM for (3.1) and write out the equation system corresponding to (3.3). Use the Lagrange basis from (3.18). ■

Exercise 3.13

Program the method from exercise 3.12 for general n and for uniform step lengths and with $f(x) = x$. The output from the program should be a graph of the numerical solution, i.e. the points $\{(x_i, \tilde{U}_i)\}_{i=1}^n$ connected with line pieces (a linear spline). The graph should also show the analytic solution $u(x) = \frac{x}{2} - \frac{x^3}{6}$.
Run your program for $n = 2, 4, 8, 16, \ldots$ until you see no further progress. (This is referred to as "convergence in the eye ball norm").
At what uniform step length h did you decide that you had convergence in the eye ball norm?
For all h values you have computed up to convergence in the eye ball norm, compute the error in the discrete ℓ^∞ and ℓ^1 norms $\|e\|_{\ell^\infty} = \max_{i=1,\ldots,n} |u(x_i) - \tilde{U}_i|$ and $\|e\|_{\ell^1} = \sum_{i=1}^n |u(x_i) - \tilde{U}_i|$ respectively and additionally in the continuous $\mathcal{L}^2$ norm $\|e\|_{\mathcal{L}^2} = \sqrt{\int_0^1 (u(x) - \tilde{U}(x))^2 \mathrm{d}x}$. Comment on what you see, for example on which of the three norms are "closer" to the eye ball norm? ■

Exercise 3.14

Construct an FEM for (1.10) and write out the equation system corresponding to (3.3). Consider the case $I = (0, 4)$, $f(t, u(t)) = \cos(t)$, $t_0 = 0$ and $u^* = 0$. Program your method, solve and compare the numerical solutions graphically to the exact solution $u(t) = \sin(t)$ and to the numerical solutions found in exercises 3.4 and 3.11. Use the Lagrange basis from exercises 3.12 and 3.13. ■

3.2 Construction of difference operators for FDM's

From here on, we shall concentrate on the finite difference methods, only bringing in the collocation and finite element methods on a few occations. The main point distinguishing the various FDM's is the selection of the difference operators that we use to replace the derivatives in the DEP. An important point is how well the difference operators approximate the derivatives in question. Consider the following setting:

- We want to approximate $f^{(k)}(z)$, the k'th derivative (for some positive integer k) of some function f in some point z where f is sufficiently smooth ($f \in \mathcal{C}^k(]z-\epsilon, z+\epsilon[$ for some $\epsilon > 0$). Example: $f'(z)$ ■ .
- For the approximation we have at our disposal a set of nodal points $x_1, \ldots, x_n$, where $x_1 < x_2 < \ldots < x_n$ and $z \in [x_1, x_n]$. To simplify matters, we shall only consider uniform subdivisions where $x_{i+1} = x_i + h$ for $i = 1, \ldots, n-1$. h is then simply denoted the step size. Our approximation $\delta_h^k f(z) = g(f_1, \ldots, f_n)$ is some function (say g) of (some of) the nodal point values of f. Example: $\delta_h f(z) = \frac{1}{h}(f(x_i+h) - f(x_i))$, where $z \in [x_i, x_i + h]$ ■ .
- We shall assume that our approximation is consistent, meaning that $\lim_{h\to 0} \delta_h^k f(z) = f^{(k)}(z)$. Example: $\lim_{h\to 0} \frac{1}{h}(f(x_i+h) - f(x_i)) = f'(z)$ ■ .

There is an important point to be made here: Until now, we have considered $\{x_1, \ldots, x_n\}$ a fixed set of n different nodal points. The introduction of the term $\lim_{h\to 0} \delta_h^k f(z)$ makes sense only when instead considering an infinite sequence of sets of more and more nodal points designed such that the uniform step size h goes to 0 as the number of nodal points goes to infinity. See for example figure 3.3. Hence let $\{n^{(k)}\}_{k=1}^{\infty}$ be a divergent strictly increasing sequence of integers and select for each $k = 1, 2, \ldots$ a set of $n^{(k)}$ different nodal points $\{x_1^{(k)}, \ldots, x_{n_k}^{(k)}\}$ and let $h^{(k)}$ be the corresponding uniform step size. Obviously, such a change of notation also implies the change from for example $\tilde{U}_i$ to $\tilde{U}_i^{(k)}$. Here comes the point: Since these sequences are always understood, we shall allow ourselves to simplify notation by omitting the upper index (k) so that we end up with the standard notation used until now. For example this means that x_i above is a different nodal point for $h = 0.1$ than for $h = 0.001$. Even worse, the definition $\delta_h f(z) = \frac{1}{h}(f(x_i+h) - f(x_i))$, where $z \in [x_i, x_i + h]$ implies that also i is changing with h, since we must take the i so that z lies in the element $[x_i, x_i + h]$. Actually $i \to \infty$ as $h \to 0$

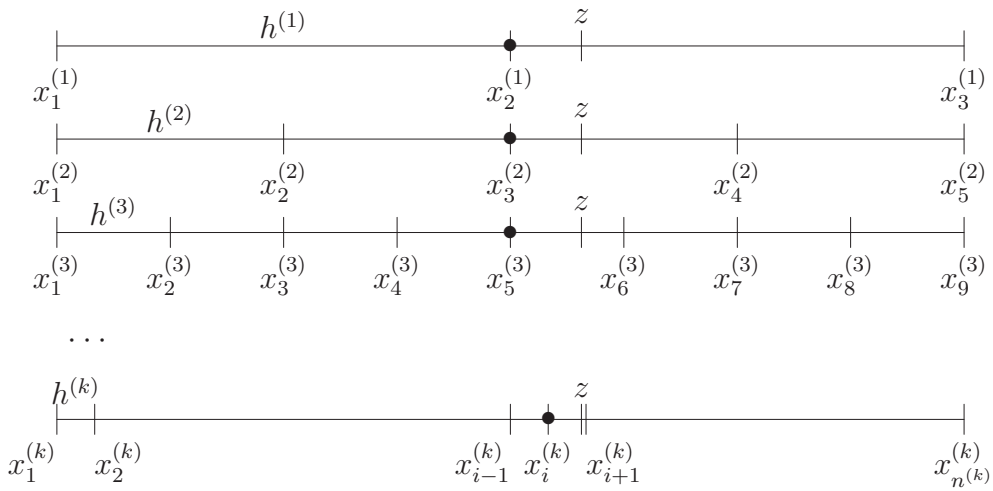


Figure 3.3: A sequence of subdivisions obtained by "halving", and a sequence of nodal points $\{x_i^{(k)}\}_{k=1}^{\infty}$ (denoted by •'s) converging towards a point z and picked so that $z \in [x_i^{(k)}, x_{i+1}^{(k)}[$.

■ . Fortunately this turns out to be a small problem in most cases, but occasionally, it is necessary to use the "full" notation to clarify matters. In any case, it is wise to keep in mind, that both x_i and i itself depend on h.

Exercise 3.15
Consider the interval $[3, 4]$ and the point $\pi \simeq 3.14159$. Let $x_1^{(1)} = 3$, $x_2^{(1)} = 3.5$ and $x_3^{(1)} = 4$ and construct a sequence of subdivisions by halving, i.e. $x_1^{(2)} = 3$, $x_2^{(2)} = 3.25$, $x_3^{(2)} = 3.5$, $x_4^{(2)} = 3.75$ and $x_5^{(2)} = 4$. Let Maple continue the process. Let also Maple find $x_i^{(k)}$ for $k = 1, 2, \ldots$ such that $\pi \in [x_i^{(k)}, x_{i+1}^{(k)}[$. Compute $e_k = |x_i^{(k)} - \pi|$ for $k = 1, 2, \ldots$, and find the order of convergence of e_k with respect to the step length $\left(\frac{1}{2}\right)^k$. (See below exercise 2.22 on page 28).
■

Returning to the difference operators, the assumption of consistency also implies that we consider the difference operator a function of h. Instead of just being satisfied with consistency it is interesting to know how fast the convergence is:

- We measure the "goodness" of the difference operator by the consistency order $q \in \mathcal{R}$ of δ_h^k defined by $\sup_f |f^{(k)}(z) - \delta_h^k f(z)| = \mathcal{O}_{h\to 0}(h^q)$ where the supremum is taken over alle k times differentiable functions f. If we only consider one particular f, we shall denote the resulting q the observed consistency order of δ_h^k for the instance f.

Consensus:

- The bigger q the better (more precise) the method.
- The smaller n the better (less expensive) the method.
- Bigger q requires bigger n (the *no free lunch* concept).

Reality:

- $\mathcal{O}_{h\to 0}(h^q)$ means an expression which for small h takes the form Ch^q i.e. for any given h, small q and small C may be better than big q and big C.

So the consensus should be taken with a grain of salt, and only considered a rule of thumb.

Example 3.16 How fast does it go?
Consider the standard approximation to $f'(z)$ namely $\delta_h f(z) = \delta_{+h} f(z) = \frac{1}{h}(f(z+h) - f(z))$ for the example $f(x) = (x-1)^4 e^x$. Taking $z = 1$ and noting that $f'(1) = -3$ we easily compute with Maple

```
> Digits:=20;
> df:=(x,h)->(f(x+h)-f(x))/h:
> evalf(df(0,10^(-1)));
                      -2.7489736065056759337
> evalf(df(0,10^(-10)));
                         -2.9999999998
> evalf(df(0,10^(-11)));
                          -3.000000000
```

The computations suffer from subtractive cancellation (see example 3.24) so Maple only shows a limited number of digits.
There are better ways than δ_+ above however. Take the same z and f but $\delta_h f(z) = \delta_{0h} f(z) = \frac{1}{2h}(f(z+h) - f(z-h))$. Maple then gives

```
> Digits:=20;
> d0f:=(x,h)->(f(x+h)-f(x-h))/(2*h):
> evalf(d0f(0,010^(-1)));
                      -2.9983491219850800221
> evalf(d0f(0,10^(-6)));
                        -2.99999999999981
> evalf(d0f(0,10^(-7)));
                         -3.0000000000000
```

Note that we may take h 1000 times bigger for the last δ having the same tolerance.
To compare δ_{+h} and δ_{0h} better, we use Maple to investigate the order of consistency of the 2 difference operators: In general we expect that for h sufficiently small $f^{(k)}(z) = \delta_h^k f(z) + Ch^q$ for some constants C and q. But then $|f^{(k)}(z) - \delta_h^k f(z)| = |C|h^q \Rightarrow \log_{10}|f^{(k)}(z) - \delta_h^k f(z)| = \log_{10}|C| + q\log_{10} h$. Thus plotting $\log_{10} h$ against $\log_{10}|f^{(k)}(z) - \delta_h^k f(z)|$ should for sufficiently small h give a straight line with slope q: The following Maple commands, building on the definitions of f, df and $d0f$ above give the plot in figure 3.4 showing $q = 1$ for δ_{+h} and $q = 2$ for δ_{0h}.

```
> Df0:=D(f)(0):
> m:=12:
> for i from 1 to m do
    h:=10^(-i):
    dfh:=df(0,h):d0fh:=d0f(0,h):
    q[i]:=[log10(h),evalf(log10(abs(Df0-dfh)))]:
    p[i]:=[log10(h),evalf(log10(abs(Df0-d0fh)))]:
  end do:
> qlist:=[seq(q[i],i=1..m)]:plist:=[seq(p[i],i=1..m)]:
> with(plots):
> qplot:=pointplot(qlist,connect=true,
                   scaling=constrained,color=red):
> pplot:=pointplot(plist,connect=true,
                   scaling=constrained,color=blue):
> display([qplot,pplot]);
```

3.2.1 Taylor Series Methods

The most popular method of construction for difference operators is based on Taylor expansion and linear difference operators $\delta_h^k f(z) = \sum_{j=1}^n c_j f(x_j)$. We determine the c_j, $j = 1, \ldots, n$ to maximize the order of consistency of the difference operator. The methods go as follows:

(a) Let $f^{(k)}(z)$ be the derivative to approximate. Select the number of nodal points n and the nodal points $x_1, \ldots, x_n$ themselves.

(b) Let $\delta_h^k f(z) = \sum_{j=1}^n c_j f(x_j)$ be the finite difference approximation of $f^{(k)}(z)$, where the c_j's for now are unknowns (we only consider linear difference operators).

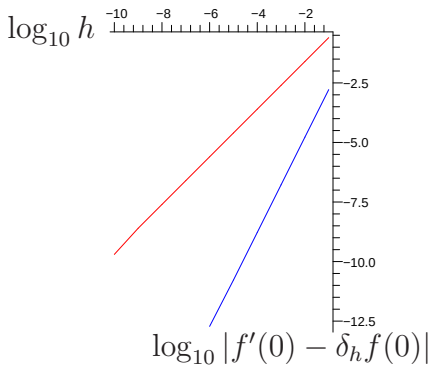


Figure 3.4: Experimental orders of consistency for δ_{+h} (slope of topmost red curve) and δ_{0h} (slope of bottommost blue curve) for example 3.16

(c) The goal is to determine the c_j's so that

$$\sum_{j=1}^{n} c_j f(x_j) - f^{(k)}(z) = \mathcal{O}_{h\to 0}(h^q)$$

for as big an order of consistency q as possible, where h is some typical step length parameter, like the step size for uniform subdivisions or the maximal step size for nonuniform subdivisions. To do this make Taylor series expansions of $f(x_j)$ around z, $\forall j = 1, \ldots, n$:

$$\sum_{j=1}^{n} c_j \left(\sum_{i=0}^{\infty} \frac{1}{i!} f^{(i)}(z)(x_j - z)^i \right) - f^{(k)}(z) = \mathcal{O}_{h\to 0}(h^q) \tag{3.19}$$

In practice the Taylor series can be truncated at $i = q + 1$ with a remainder term to verify the order q or even at $i = q$ to verify the order *at least* q. The problem is that q typically is not known. Often though, the user has some idea of what value q takes. In practice you may have to choose between including too few terms at the first attempt and hence having to add further terms in a second attempt, or adding too many terms in the first attempt and hence doing some unnecessary work.

(d) Find $c_1, \ldots, c_n$ that makes q as big as possible (for a general $f \in \mathcal{C}^{q+1}$) and define $\delta_h^k f(z) = \sum_{j=1}^{n} c_j f(x_j) = \mathbf{c}^T \mathbf{f}$ with $\mathbf{c} = (c_1, \ldots, c_n)^T$ and $\mathbf{f} = (f(x_1), \ldots, f(x_n))^T$.
This is done by cancelling the lowest order terms: Pick c_j, $j = 1, \ldots, n$

such that first of all all terms involving negative or zero powers of h cancel out. Then cancel out terms with higher and higher powers of h until all c_j's are determined. This is not as simple as it sounds since the c_j typically depend on h. Instead the coefficients of the lower order derivatives of f, $f(z), f'(z), \ldots, f^{(k)}(z)$ can be canceled out. In this process we do not need to take into consideration lower derivatives multiplied with high powers of h. Those we can leave alone. This is all somewhat confusing, but some examples below should clarify the situation.

Theorem 3.17 *The Taylor series methods result in the linear difference operators with the highest possible consistency order among all linear difference operators with the same nodal points.*

Proof:
Obvious by construction. ■

Now let us consider some examples of this construction. The examples will all be one dimensional. Standard procedure in higher dimensions is to combine one dimensional operators, but a direct derivation can also be made using Taylor expansion in the required number of dimensions. Until now, we have denoted the difference operators by δ_h^k. To simplify notation, often the h is discarded and the dependence on h is understood without appearing explicitly.

Example 3.18 δ_{+h} or δ_+ the First Order Forward Difference Operator of Consistency Order One
• *First Order* refers to the fact that $\delta_+ f(z)$ will be treated as an approximation to $f'(z)$.
• *Consistency Order One* refers to the fact that $q = 1$ (as we shall verify below).
• $\delta_+ f(z)$ is defined as the Taylor series method with $n = 2$, $x_1 = z$ and $x_2 = z + h$ for some real parameter $h > 0$, i.e. $\delta_+ f(z) = c_1 f(z) + c_2 f(z + h)$ so (3.19) takes the form

$$c_1 f(z) + c_2 \left(f(z) + hf'(z) + \frac{h^2}{2} f''(\xi_z) \right) - f'(z) = \mathcal{O}_{h\to 0}(h^q) \tag{3.20}$$

$$\Updownarrow$$

$$(c_1 + c_2) f(z) + (c_2 h - 1) f'(z) + c_2 \frac{h^2}{2} f''(\xi_z) = \mathcal{O}_{h\to 0}(h^q)$$

We stop the Taylor series at $k = 2$ with a remainder term. How high it is necessary to go is to some extent up to a (qualified) guess.

To maximize q first take $c_1 + c_2 = 0$ and $c_2 = \frac{1}{h} \Rightarrow c_1 = -\frac{1}{h}$. This cancels out the $f(z)$ and $f'(z)$ terms and leaves us with no more c_j's to determine. The lowest order surviving term is $\frac{1}{2}hf''(z)$ giving $q = 1$. Note that for a particular f and z combination, it is possible that $q \geq 2$ (if $f''(z) = 0$). This does not change the consistency order of δ_+ but may change the observed consistency order for the instance $f(z)$. For example if $f(x) = \sin(x)$ and $z = \pi/2$ then $f''(z) = -\sin(\pi/2) = 0$ giving second order observed consistency for this particular instance of f and z. But taking $f(x) = \cos(x)$, $f''(\pi/2) = -\cos(\pi/2) = -1$ giving only first order observed consistency for this particular instance of f and z. To find a fixed order of consistency for a given difference operator, we take the smallest among all f's by choosing the supremum of $|f'(z) - \delta_+ f(z)|$ over all differentiable f, which by Taylor expansion as seen above is 1 ■ . Consistency orders are verified theoretically, while instances of observed consistency errors typically arise from numerical experiment and should always be at least as big as the consistency orders (but may be bigger without "destroying" the theory). In conclusion

$$\delta_{+h} f(z) = \delta_+ f(z) = \frac{1}{h}\left(f(z+h) - f(z)\right) = f'(z) + \mathcal{O}_{h\to 0}(h). \tag{3.21}$$

■

We shall simply state the formulas corresponding to (3.21) for 3 other often used difference operators that may be derived using the Taylor series method:

Example 3.19 δ_- the First Order Backward Difference Operator of Consistency Order One

$$\delta_{-h} f(z) = \delta_- f(z) = \frac{1}{h}\left(f(z) - f(z-h)\right) = f'(z) + \mathcal{O}_{h\to 0}(h). \tag{3.22}$$ ■

Example 3.20 δ_0 the First Order Central Difference Operator of Consistency Order Two

$$\delta_{0h} f(z) = \delta_0 f(z) = \frac{1}{2h}\left(f(z+h) - f(z-h)\right) = f'(z) + \mathcal{O}_{h\to 0}(h^2). \tag{3.23}$$ ■

Example 3.21 δ_0^2 the Second Order Central Difference Operator of Consistency Order Two

$$\begin{aligned}\delta_{0h}^2 f(z) = \delta_0^2 f(z) = \frac{1}{h^2}\left(f(z+h) - 2f(z) + f(z-h)\right) \\ = f''(z) + \mathcal{O}_{h\to 0}(h^2).\end{aligned} \tag{3.24}$$ ■

Exercise 3.22
Derive the consistency orders of examples 3.19, 3.20 and 3.21 above, i.e. derive the last equality sign in each case using Taylor expansion to the appropriate order. ■

Exercise 3.23 Non Uniform Subdivision
Consider the central difference operator of example 3.20 above. Change it to

$$\delta_0^* f(z) = \frac{1}{(1+a)h}(f(z+h) - f(z-ah)).$$

Investigate the order of consistency as a function of a. ■

Example 3.24 Warning – Look out for subtractive cancellation
Consider a computer with 15 significant digits, $h = 10^{-20}$, $f(z) = 1$, $f(z+h) = 1 + 10^{-16}$. Then $\delta_+ f(z) = 10^4$ but the computer instead gets 0 as follows:

$$\delta_+ f(z) = (\underbrace{\underbrace{1+10^{-16}}_{\text{stored as 1}}}_{\text{stored as 0}} - 1)/10^{-20} = 0 \text{ (0/anything=0)}$$

Rule of thumb: $h = 10^{-d+r}$ where d is the number of significant digits for the computer and $r \in [1, d[$: If f is flat, r is big to avoid subtractive cancellation. If request for precision is low, r is big to avoid over spenditure of computer ressources to get unnecessarily precise results.
Precision depends in practice on the step size h in the fashion sketched in figure 3.5, so h should neither be too small nor too big. ■

The result of theorem 3.17 on page 82 could be improved if we did not preselect the nodal points in item (a) of the construction process at the beginning of this subsection, but left them to be decided, replacing the fourth item (d) just above theorem 3.17 with

(d') Find $c_1, \ldots, c_n$ and $x_1, \ldots, x_n$ that makes q as big as possible (for a general $f \in \mathcal{C}^{q+1}$) and define $\delta f(z) = \sum_{j=1}^n c_j f(x_j) = \mathbf{c}^T \mathbf{f}$ with $\mathbf{c} = (c_1, \ldots, c_n)^T$ and $\mathbf{f} = (f(x_1), \ldots, f(x_n))^T$.

A similar approach is used in numerical integration in the socalled Gaussian quadrature formulas. Instead it is basically not used within numerical differentiation. The reason is, that very often a certain derivative must be

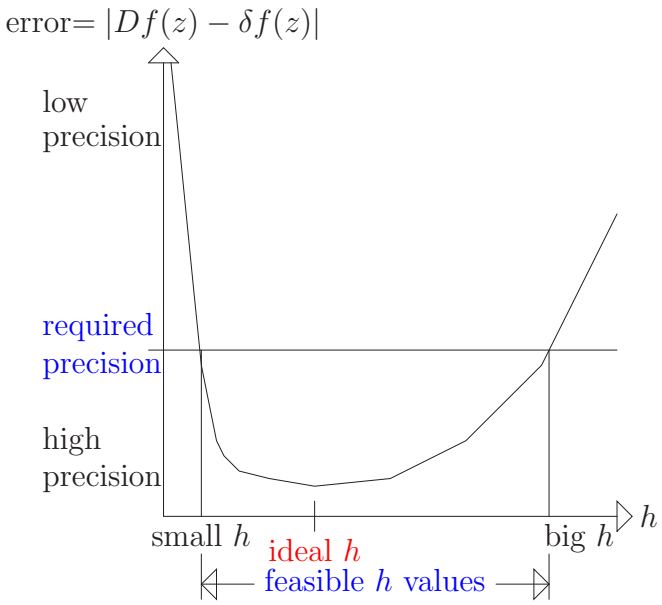


Figure 3.5: Practically observed dependence of precision of difference operators on step size h

approximated not just in one but in several points. There is a great computational advantage in being able to reuse function evaluations from derivative to derivative, but this requires that the function evaluations are needed in the same points. This requires some control over the location of the nodal points which does not exist for methods optimizing the position of the nodal points.

3.2.2 Taylor expansion in Maple

Maple has the command `taylor` for computing Taylor expansions. `taylor` takes 3 arguments: The function to be expanded, the point to expand around and the order that the function should be expanded to. An example is

```
> taylor(x^4,x=3,3);
```

resulting in $81 + 108(x-3) + 54(x-3)^2 + \mathcal{O}\left((x-3)^3\right)$. When it is required to do further manipulations, the $\mathcal{O}$-term is generally a nuisance. It can be omitted by "converting to a polynomial":

```
> convert(taylor(x^4,x=3,3),polynom);
```

results in $81+108(x-3)+54(x-3)^2$ which can be used directly in a function definition as in

```
> p:=unapply(convert(taylor(x^4,x=3,3),polynom),x);
```

resulting in $p := x \to -243 + 108x + 54(x-3)^2$.

Unfortunately, reality is often slightly more complicated than this simple example. Consider the expansion $f(z+h) = f(z)+hf'(z)+\frac{h^2}{2}f''(z)+\mathcal{O}(h^3)$ from example 3.18. The correct way to read this is "find the Taylor expansion of the function $f : x \to f(x)$ around z and take the value of this expansion in $z+h$". The Maple command for this is

```
> subs(x=z+h,taylor(f(x),x=z,3));
```

resulting in the required expression. There are other ways to get the correct result as for example

```
> taylor(f(x+h),x=z,3);
> subs(z+h=z,%);
> subs(x-z=h,%);
```

but these are typically harder to explain, i.e. relate to what is happening mathematically.

If f is a scalar function of 2 variables as in $f : (x,y) \to f(x,y)$ then in general Maple provides the `mtaylor` command for multivariable Taylor expansion. For this course however, the second variable is typically depending on the first as in equation (1.10) where f is in reality a function of one variable $f : x \to f(x,u(x))$. In this case it is recommended to use Taylor expansion in one variable, i.e. the Maple command `taylor`.

3.2.3 Polynomial Interpolation Methods

(a) Select the interpolation nodal points $\{x_j\}_{j=1}^n$. This then results in the data set $\{(x_j, f(x_j))\}_{j=1}^n$.

(b) By a standard interpolation result (see theorem 2.44 on page 41 and theorem 2.47 on page 46 and change $n+1$ to n) $f(z) = \sum_{j=1}^n f(x_j)\ell_j(z) + \frac{1}{n!}f^{(n)}(\xi_z)\omega_n(z)$ where ℓ_j is the j'th cardinal function for the points $x_1,\ldots,x_n$ and $\omega_n(z) = \prod_{j=1}^n(z-x_j)$.
But then $Df(z) = \sum_{j=1}^n f(x_j)D\ell_j(z) + \frac{1}{n!}D\left(f^{(n)}(\xi_z)\omega_n(z)\right)$.

(c) Take $\delta f(z) = \sum_{j=1}^n f(x_j)D\ell_j(z)$ and find q so that $\frac{1}{n!}D\left(f^{(n)}(\xi_z)\omega_n(z)\right) = \mathcal{O}_{h\to 0}(h^q)$ where $h = \max_{i,j=1,\ldots,n}|x_i - x_j|$ corresponding to the situation where we make polynomial interpolation of degree $n-1$ in an

interval of length h. Eventually, parts of the $\frac{1}{n!}D\left(f^{(n)}(\xi_z)\omega_n(z)\right)$ term can be included in $\delta f(z)$ when advantageous.

Example 3.25 $D = \frac{d}{dx}$, data set $\{(x_1-h, f(x_1-h)), (x_1, f(x_1)), (x_1+h, f(x_1+h)\}$, $z = x_1$
The Maple commands

```
> with(CurveFitting):
> xydata:=[[x1-h,f0],[x1,f1],[x1+h,f2]]:
> PolynomialInterpolation(xydata,x):
> diff(%,x):
> subs(x=x1,%):
> simplify(%);
```

results in $\frac{f2-f0}{2h}$, i.e. the wellknown δ_0 derived in example 3.20 by the Taylor series method. ■

Example 3.26 $D = \frac{d}{dx}$, $z = x_k$ for some $k = 1, \ldots, n$
$D\left(f^{(n)}(\xi_z)\omega_n(z)\right)\Big|_{z=x_k} = \omega_n(x_k)\frac{d}{dx}f^{(n)}(\xi_z)|_{z=x_k} + (\omega_n)'(x_k)f^{(n)}(\xi_{x_k}) =$
$(\omega_n)'(x_k)f^{(n)}(\xi_{x_k}) = \mathcal{O}_{h\to 0}(h^{n-1})$ since $\omega_n(x_k) = 0$, $(\omega_n)'(x_k) =$
$\sum_{i=1}^{n}\prod_{j=1,j\neq i}^{n}(x_k - x_j) = \prod_{j=1,j\neq k}^{n}(x_k - x_j)$ and $|x_k - x_j| \leq h$. ■

Similar methods exist for higher derivatives.

The Taylor series and polynomial interpolation methods will in many cases lead to the same difference operators. The main difference between the two approaches is that for the polynomial interpolation methods the formulas are given directly in the form of derivatives of the known cardinal functions, whereas a system of equations must be solved for the Taylor series methods. On the other hand the Taylor series methods provide the optimality result of theorem 3.17 on page 82 while a similar result is not obvious for the interpolation methods.

3.2.4 Compact Finite Difference Methods

In (d') just below figure 3.5 on page 85 was proposed a possibility to increase the consistency order of linear difference operators. The proposal was rejected on the base of lack of possibility to reuse function evaluations when several derivatives had to be evaluated. Another possibility lies in the socalled Compact Finite Difference Methods which can be formulated as follows:

(a) Assume that we want to approximate a certain derivative Df in a set of nodal points $(x_1, \ldots, x_n)$. Define the vectors of derivatives $\mathbf{Df} = (Df(x_1), \ldots, Df(x_n))^T$ and approximating linear difference operators $\boldsymbol{\delta}\mathbf{f} = (\delta f(x_1), \ldots, \delta f(x_n))^T$ defined by $\mathbf{B}\boldsymbol{\delta}\mathbf{f} = \mathbf{C}\mathbf{f}$ where $\mathbf{B}$ and $\mathbf{C}$ are constant $n \times n$ matrices and $\mathbf{f} = (f(x_1), \ldots, f(x_n))^T$.

(b) Find the $n \times n$ matrices $\mathbf{B}$ and $\mathbf{C}$ that makes the consistency order q as big as possible where $\|\mathbf{B}(\mathbf{Df} - \boldsymbol{\delta}\mathbf{f})\| = \mathcal{O}_{h\to 0}(h^q)$ for some vector norm $\|\cdot\|$.

Now the difference formulas can be recovered for all points at once, and this turns out to give some advantages in certain cases. If $\mathbf{B}$ and $\mathbf{C}$ are completely arbitrary matrices, the problem becomes very hard and usually some structure is imposed on the matrices (like symmetric, tri-diagonal or penta-diagonal) For more on this subject see for example [4] §10.10.2.

Exercise 3.27
Write a small essay on compact finite difference methods. ■

3.2.5 Richardson extrapolation

It is possible to derive iterative hierarchical methods that given an approximation method increases the consistency order, at the cost of some additional function evaluations. Let $\phi(h)$ be the numerical method depending on a parameter h for approximating something denoted L not depending on h. Assume that $\phi(h)$ has consistency order 1, i.e. that

$$\phi(h) = L + \sum_{j=1}^{\infty} a_j h^j \tag{3.25}$$

for some sequence of constants $\{a_j\}_{j=1}^{\infty}$ not depending on h. To give an idea we present the following example:

Example 3.28 $f \in \mathcal{C}^\infty$, $f'(x) = \delta_{+h} f(x) + \mathcal{O}_{h\to 0}(h)$

$$\begin{aligned} \underbrace{\frac{f(x+h) - f(x)}{h}}_{\delta_{+h}f(x)=\phi(h)} &= \underbrace{f'(x)}_{L} + \sum_{j=2}^{\infty} \frac{1}{j!} f^{(j)}(x) h^{j-1} \\ &= f'(x) + \sum_{j=1}^{\infty} \underbrace{\frac{1}{(j+1)!} f^{(j+1)}(x)}_{a_j} h^j. \end{aligned} \tag{3.26}$$

■

Now consider the same method ϕ but with a different parameter value (step length) αh:

$$\phi(\alpha h) = L + \sum_{j=1}^{\infty} a_j \alpha^j h^j. \tag{3.27}$$

Subtracting (3.27) from (3.25) we get

$$\phi(h) - \phi(\alpha h) = \sum_{j=1}^{\infty} (1 - \alpha^j) a_j h^j. \tag{3.28}$$

Note that the unknown L has disappeared, and hence that we may isolate and recover $a_1 h$:

$$a_1 h = \frac{\phi(h) - \phi(\alpha h)}{1 - \alpha} - \sum_{j=2}^{\infty} \frac{1 - \alpha^j}{1 - \alpha} a_j h^j. \tag{3.29}$$

Inserting in (3.25) we get

$$\begin{aligned} L &= \phi(h) - \frac{\phi(h) - \phi(\alpha h)}{1 - \alpha} - \sum_{j=2}^{\infty} \left(1 - \frac{1 - \alpha^j}{1 - \alpha}\right) a_j h^j \\ &= -\frac{\alpha}{1 - \alpha}\phi(h) + \frac{1}{1 - \alpha}\phi(\alpha h) - \sum_{j=2}^{\infty} \frac{\alpha^j - \alpha}{1 - \alpha} a_j h^j. \end{aligned} \tag{3.30}$$

Hence $\Phi(h) = -\frac{\alpha}{1-\alpha}\phi(h) + \frac{1}{1-\alpha}\phi(\alpha h)$ is our new numerical method for approximating L. Note that Φ has order of consistency 2. Note also that the construction can be iterated, now starting from Φ instead of from ϕ.

How do we select α? To make sure that the new method Φ works better than the original ϕ, we would like all the coefficients of the powers of h to decrease (or at least not increase) from ϕ to Φ, i.e. $\left|\frac{\alpha^j - \alpha}{1-\alpha} a_j\right| \leq |a_j| \Leftrightarrow \left|\frac{\alpha^j - \alpha}{1-\alpha}\right| \leq 1$ for all $j = 2, 3, \ldots$. But it is possible to verify that $\left|\frac{\alpha^j - \alpha}{1-\alpha}\right| \leq 1 \Leftrightarrow \alpha \in [0, \kappa_j]$ where $\kappa_2 = 1$ and $\kappa_j \downarrow \frac{1}{2}$ for $j \to \infty$. This means that we should take $\alpha \leq \frac{1}{2}$. The most common choice is $\alpha = \frac{1}{2}$ because this gives the highest possibility of being able to reuse some function evaluations: Consider example 3.28: If we need to approximate $f'(x)$ to second order consistency with $\alpha = 1/2$, we need to use $f(x)$, $f(x + \frac{1}{2}h)$ and $f(x + h)$. If we further need to approximate $f'(x + \frac{1}{2}h)$ then we need to use $f(x + \frac{1}{2}h)$, $f(x + h)$ and $f(x + \frac{3}{2}h)$. Note that only 1 new function value is needed. Taking $\alpha = \frac{1}{2}$ in (3.30) we get

$$L = 2\phi(\frac{h}{2}) - \phi(h) + \mathcal{O}_{h \to 0}(h^2). \tag{3.31}$$

Exercise 3.29 Richardson extrapolation
Take $\phi(h) = \delta_{+h}\sin(1)$ and $\psi(h) = 2\phi(\frac{h}{2}) - \phi(h)$. Show using Maple that $|\cos(1) - \phi(h)| = \mathcal{O}(h)$ and that $|\cos(1) - \psi(h)| = \mathcal{O}(h^2)$. Hint: Use many digits, for example `Digits:=40 and compute` $|\cos(1) - \phi(h)|/h$ `and` $|\cos(1) - \psi(h)|/h^2$ `for` $h = 10^{-n}$, $n = 1, 2, \ldots$. ■

Another situation often ocurring is the one where only even powers of h appears, i.e.

$$\psi(h) = L + \sum_{j=1}^{\infty} a_{2j} h^{2j}. \tag{3.32}$$

Example 3.30 $f \in \mathcal{C}^\infty$, $f'(x) = \delta_{0h} f(x) + \mathcal{O}_{h\to 0}(h^2)$

$$\begin{aligned} \underbrace{\frac{f(x+h) - f(x-h)}{2h}}_{\delta_{0h}f(x)=\psi(h)} &= \underbrace{f'(x)}_{L} + \sum_{k=2}^{\infty} \frac{1}{k!} f^{(k)}(x) \frac{h^k - (-h)^k}{2h} \\ &= f'(x) + \sum_{j=1}^{\infty} \underbrace{\frac{1}{(2j+1)!} f^{(2j+1)}(x)}_{a_{2j}} h^{2j}, \end{aligned} \tag{3.33}$$

where we have taken $\{k\}_{k=2}^{\infty} = \{2j\}_{j=1}^{\infty} \cup \{2j+1\}_{j=1}^{\infty}$, noting that the first part on the right hand side gives no contribution since $h^{2j} - (-h)^{2j} = 0$. ■

Taking $\tilde{h} = h^2$, $\tilde{a}_j = a_{2j}$ and $\phi(\tilde{h}) = \phi(h^2) = \psi(h)$, (3.32) transforms into (3.25) (with $\tilde{h}$ replacing h and $\tilde{a}_j$ replacing a_j).

Choosing again to put the new nodes in the middle between the ones used for the original method, this corresponds to considering for the improved method a linear combination of $\psi(\frac{h}{2})$ and $\psi(h)$. But $\psi(\beta h) = \phi(\beta^2 h^2) = \phi(\beta^2 \tilde{h})$, so taking $\beta = \frac{1}{2}$ corresponds to taking $\alpha = \beta^2 = \frac{1}{4}$ in (3.30) resulting in the method

$$L = \frac{4}{3}\phi(\frac{\tilde{h}}{4}) - \frac{1}{3}\phi(\tilde{h}) + \mathcal{O}_{\tilde{h}\to 0}(\tilde{h}^2) = \frac{4}{3}\psi(\frac{h}{2}) - \frac{1}{3}\psi(h) + \mathcal{O}_{h\to 0}(h^4). \tag{3.34}$$

As above, also this construction may be iterated, increasing by 2 the order of consistency with each iteration. This iteration is described in the following theorem, where $D(n,k)$ indicates the n'th halving of h and the k'th iteration (each iteration increasing the order by 2).

Theorem 3.31 *Richardson Extrapolation*
If ψ is given by (3.32) and we define

$$D(n,0) = \psi(h/2^n), \quad n = 0,1,\ldots \tag{3.35}$$

$$D(n,k) = \frac{4^k}{4^k-1}D(n,k-1) - \frac{1}{4^k-1}D(n-1,k-1), \tag{3.36}$$
$$k = 1,2,\ldots,\ n = k,k+1,\ldots$$

Then there exist constants $\{A_{j,k+1}\}$ for all the index values indicated, such that

$$D(n,k) = L + \sum_{j=k+1}^{\infty} A_{j,k+1}\left(\frac{h}{2^n}\right)^{2j} \quad k = 0,1,2,\ldots,\ n = k,k+1,\ldots \tag{3.37}$$

Proof: (Induction over k)
$k = 0:\ D(n,0) = L + \sum_{j=1}^{\infty} a_{2j}\left(\frac{h}{2^n}\right)^{2j}$ by (3.32) and (3.35). Take $A_{j,1} = a_{2j}$. Then (3.37) holds.
Assume that (3.37) holds for $k-1$. (3.36) and (3.37) then gives

$$\begin{aligned} &D(n,k) \\ &= \frac{4^k}{4^k-1}\left(L + \sum_{j=k}^{\infty} A_{j,k}\left(\frac{h}{2^n}\right)^{2j}\right) - \frac{1}{4^k-1}\left(L + \sum_{j=k}^{\infty} A_{j,k}\left(\frac{h}{2^{n-1}}\right)^{2j}\right) \\ &= L + \sum_{j=k}^{\infty} \frac{4^k - 4^j}{4^k-1} A_{j,k}\left(\frac{h}{2^n}\right)^{2j} = L + \sum_{j=k+1}^{\infty} A_{j,k+1}\left(\frac{h}{2^n}\right)^{2j} \end{aligned}$$

for $A_{j,k+1} = \frac{4^k-4^j}{4^k-1}A_{j,k}$. ■

Computation:
To compute the iterations, the following triangle may be useful:

(3.38)

	$\mathcal{O}(h^2)$		$\mathcal{O}(h^4)$		$\mathcal{O}(h^6)$	
h	$D(0,0)$	$\searrow$				
$\frac{h}{2}$	$D(1,0)$	$\rightarrow$	$D(1,1)$	$\searrow$		
$\frac{h}{4}$	$D(2,0)$	$\rightarrow$	$D(2,1)$	$\rightarrow$	$D(2,2)$	
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\ddots$

3.3 Classification and notation for FDM's

Recall from section 3.1.1 that an FDM is constructed by replacing all derivatives in some equation from the DEP (either the differential equation taken in

some point or a boundary condition) by difference operators. This approach leads to the optimal representation for the FDM. If instead the equations resulting from this replacement process are scaled, i.e. multiplied through by something other than the constant 1, then the resulting equations are *not* the optimal representation. For example

$$\tilde{U}_1 = u^*,\ \frac{1}{h}(\tilde{U}_{i+1} - \tilde{U}_i) = f(x_i, \tilde{U}_i), i = 1, \ldots, n-1$$

is an optimal representation for the DEP

$$u(t_0) = u^*,\ u'(t) = f(t, u(t)),\ t > t_0,$$

replacing $u'(x_i)$ by $\delta_{+h}u(x_i)$ whereas

$$\tilde{U}_1 = u^*,\ \tilde{U}_{i+1} - \tilde{U}_i = hf(x_i, \tilde{U}_i), i = 1, \ldots, n-1,$$

where we have multiplied all but the first equation through by h, is a non optimal representation ■ .

Note that scaling does not change the solution. The reason that we are concerned with the optimal representation is that we are interested in the error in the finite difference solution $u(x_i) - \tilde{U}_i$. It turns out, that this error can be computed more easily through the difference between the equations in the optimal representation of the FDM and the equations in the DEP that these are approximations of. For simplicity of notation, we shall write $\text{DEP}_i - \text{FDM}_i,\ i = 1, \ldots, n$ for these differences and return with more explanation later. Obviously, for this difference it is significant which representation for the FDM we use: For example say $\text{DEP}_i - \text{FDM}_i = 7$ then clearly $\text{DEP}_i - 17 \cdot \text{FDM}_i \neq 7$ ■ .

Another point to be made here is that an FDM should be considered a method working on a class of DEP's. (Recall the discussion about classes of DEP's at the end of section 1.1). For example, for the FDM above, it is a method working for all continuous functions f of 2 variables that is Lipschitz continuous in the second variable (to make the DEP well-posed) and for all real numbers u^* ■ . In the following we will distinguish between classes of DEP's and single DEP's lying in such a class, that we shall denote a representative for the class, where the data is specified (but possibly unknown to the reader). Classes of DEP's are primarily used when we talk theory of FDM's while representatives of a class is used when we compute with an FDM.

★ **For advanced readers 3.32** ★ Optimal representative for an FDM for a class of DEP's Here we present some further specification of the notion of optimal representations for FDM's. It may be skipped at the first reading or all together, if the reader is satisfied with the exposition above.

Definition 3.33 *An FDM for a class of DEP's is a prescription (algorithm) that given a DEP in the class, i.e. given the data, results in a sequence of values $\tilde{U}_1, \ldots, \tilde{U}_n$ serving as approximations to solution values $u(x_1), \ldots, u(x_n)$ for the DEP in predetermined nodal points $x_1, \ldots, x_n$.*
If $\tilde{U}_1, \ldots, \tilde{U}_n$ are given implicitly (and uniquely) by an n dimensional system of equations

$$\mathbf{\Psi}(\tilde{U}_1, \ldots, \tilde{U}_n) = 0, \tag{3.39}$$

where $\mathbf{\Psi}$ depends on the data, then such a system is called a Representation for the FDM for the class of DEP's.
Given a representation for an FDM for a class of DEP's, $\mathbf{\Psi}(\tilde{U}_1, \ldots, \tilde{U}_n) = 0$, we shall identify the FDM for the class of DEP's with the Representation Class consisting of all non singular scalings of the representation, $\boldsymbol{diag}(b_1, \ldots, b_n)\mathbf{\Psi}(\tilde{U}_1, \ldots, \tilde{U}_n) = 0$, for any non zero scalars $b_1, \ldots, b_n$.

As $\tilde{U}_i$ is our approximation to $u(x_i)$ for $i = 1, \ldots, n$, for the various i, some of the Ψ_i in (3.43) are approximations to the differential equation(s) of the DEP evaluated in various points and others to approximations to the additional (boundary) conditions. For example for (3.2) on page 62, equation i for $i = 2, \ldots, n-1$ corresponds to an approximation of the differential equation in the nodal point x_i, while the first and last equation for $i = 1$ and $i = n$ correspond to approximations to the boundary conditions in x_i ■ . Below, we shall meet examples where the situation is more complicated and where equation i is an approximation to some equation of the DEP that is not necessarily "located" in the nodal point x_i. Actually, we shall meet equations located in points that are not even nodal points. As a matter of fact, it may not even be clear up front at exactly what points the equations are located. In any case however, the Ψ_i's are connected to Approximation Points z_i, $i = 1, \ldots, n$ and we denote the approximated equations of the class of DEP's by

$$F_i(z_i, u(z_i), u'(z_i), u''(z_i), \ldots, u^{(L)}(z_i)) = 0, \text{ for } i = 1, \ldots, n, \tag{3.40}$$

where $\boldsymbol{F} = (F_1, \ldots, F_n)$ is denoted the data and L is the order of the highest derivative occurring in the DEP. Hence we have complicated our lives by introducing a possibly unknown set of approximation points $\{z_i\}_{i=1}^n$. To complicate our lives even further, the Ψ_i's are only *representations* for the FDM. The complete FDM is represented by $b_i\Psi_i = 0$, $i = 1, \ldots, n$ for arbitrary constants b_i. Now why is this at all relevant? The answer lies in the error:

Our measure of "goodness" of an FDM is that the errors $u(x_i) - \tilde{U}_i$ between the exact solutions to the DEP in the nodal points and the (exact)

solutions to the FDM are small for all $i = 1, \ldots, n$. This would mean that our approximation is good, since we use $\tilde{U}_i$ as the approximation of $u(x_i)$ for $i = 1, \ldots, n$. It turns out to be hard to evaluate these errors directly, but also that we can compute the error in two steps, the first of which involves the error between the DEP and the FDM. But how do we measure the error between a differential equation problem and a finite difference method? The answer turns out to be that the correct measure is $\sup_v |F_i(z_i, v(z_i), \ldots, v^{(L)}(z_i)) - b_i\Psi_i(v(x_1), \ldots, v(x_n))|$ for $i = 1, \ldots, n$. This error involves the worse function v among all functions in the solution space of the DEP, for example $\mathcal{C}^2([0,1])$ for (3.1) on page 59 ■, which is plugged into not only the DEP (F_i) but also the FDM (Ψ_i). More importantly the error could depend on the selection of both b_i and z_i. The error we "need" in order to connect the error between the DEP and the FDM to the error between the solutions to the DEP and the FDM turns out to be the smallest possible one. Hence we need to distinguish the z_i's and the b_i's giving the smallest error. This is done in definition 3.34 below where these optimal choices are called y_i and a_i respectively.

For now, and for the rest of this note, we shall only consider sufficiently smooth classes of DEP's where the data is so smooth that both the data and the corresponding solutions to (3.40) belong to $\mathcal{C}^{q_i}$ where q_i is introduced in the following definition. The aim of the definition is to introduce notation allowing us to distinguish between the many representations of an FDM.

Definition 3.34 *Let h be a typical (step length) parameter for an FDM with a representation* (3.39) *consisting of approximations to the equations in* (3.40)*, let $\mathcal{V}_{\boldsymbol{F}}$ be the solution space for the DEP with data $\boldsymbol{F}$, let $\Psi_i = 0$, $i = 1, \ldots, n$ be any representation in the representation class generated by* (3.39) *with data $\boldsymbol{F}$ and let $\{z_i\}_{i=1}^n$ be any set of approximation points for* (3.40)*. Let for any $i \in \{1, \ldots, n\}$, $q_i(b_i, z_i)$ be the largest real numbers such that*

$$\sup_{\boldsymbol{F}} \sup_{v \in \mathcal{V}_{\boldsymbol{F}}} |F_i(z_i, v(z_i), v'(z_i), v''(z_i), \ldots, v^{(L)}(z_i)) - b_i\Psi_i(v(x_1), \ldots, v(x_n))| = \mathcal{O}_{h\to 0}(h^{q_i(b_i,z_i)}), \tag{3.41}$$

where the supremum over $\boldsymbol{F}$ is over all valid data which is so smooth that $F_i \in \mathcal{C}^{q_i(b_i,z_i)}$ and also $\mathcal{V}_{\boldsymbol{F}} \subseteq \mathcal{C}^{q_i(b_i,z_i)}$. Let for $i = 1, \ldots, n$, a_i and y_i be selections of b_i and z_i, maximizing $q_i(b_i, z_i)$ over all real numbers b_i and over all approximation points z_i and let $q_i(a_i, y_i) = q_i$.
$\Phi_i = a_i\Psi_i = 0$, $i = 1, \ldots, n$, i.e. (3.39) *is then denoted an Optimal Representation for the FDM for the class of DEP's with respect to* (3.40) *and $y_1, \ldots, y_n$ are denoted Optimal Approximation Points for* (3.39) *with respect to* (3.40)*. Finally, for $i = 1, \ldots, n$, q_i is denoted the i'th local order of consistency for the FDM for the class of DEP's and $q = \min_{i=1,\ldots,n} |q_i|$ is denoted the global order of consistency. (See also definition 3.40).*

The optimal representation

$$\Phi_i(\tilde{U}_1, \ldots, \tilde{U}_n) = 0, \text{ for } i = 1, \ldots, n. \tag{3.42}$$

for an FDM is important for the derivation of the theoretical properties (most importantly consistence and convergence) of the FDM. For computations with the FDM, any representation will do, because the solution is independent of which representation is selected.

The optimal approximation points are generally not unique but may be chosen within $\mathcal{O}(h^q)$-neighborhoods of certain points where q depends on the consistency orders of the difference operators involved. Often the optimal approximation points y_i may be chosen to be nodal points, and often the y_i's are also repeated like $y_1 = x_1$, $y_{i+1} = x_i$ for $i = 1, \ldots, n-1$. The repeated points will then typically correspond to one or more boundary conditions *and* a differential equation. The optimal approximation points are not important in themselves. Their importance lie entirely in the role they play for the determination of the optimal representation for the FDM for the DEP. The representation $\{\Phi_i\}_{i=1}^n$ most closely approximating the equations $\{F_i\}_{i=1}^n$ from the DEP is clearly optimal. Normally it is a question of selecting plus or minus the appropriate powers of h to multiply the given representation with, in order to transform the equations into approximations to the differential equations. Once having found an optimal representation, q_i can be recovered from an arbitrary v, as for example the exact solution to the DEP u, as long as "occational zero's", like $u''(x_i) = 0$, are *not* taken into consideration. ★

We shall use the following generic form for a representation of a general FDM corresponding to a general class of DEP's

$$\Psi_i(\tilde{U}_1, \ldots, \tilde{U}_n) = 0, \text{ for } i = 1, \ldots, n. \tag{3.43}$$

The Ψ_i depend on the data but may also depend on other known items as for example the nodal points. See for example equation (3.2) on page 62 for a practical example. There $\Psi_1(\tilde{U}_1, \ldots, \tilde{U}_n) = \tilde{U}_1 = 0$, $\Psi_i(\tilde{U}_1, \ldots, \tilde{U}_n) = -\delta_0^2 \tilde{U}_i - f_i = -\frac{1}{h^2}(\tilde{U}_{i-1} - 2\tilde{U}_i + \tilde{U}_{i+1}) - f_i = 0$, $i = 2, \ldots, n-1$, $\Psi_n(\tilde{U}_1, \ldots, \tilde{U}_n) = \delta_- \tilde{U}_n = \frac{1}{h}(\tilde{U}_n - \tilde{U}_{n-1}) = 0$ ■ .

Note that the n equations in an FDM are each approximations to some equation in the class of DEP's being approximated. For the example above $\Psi_1(\tilde{U}_1, \ldots, \tilde{U}_n) = 0$ approximates the left boundary condition $u(0) = 0$ while $\Psi_n(\tilde{U}_1, \ldots, \tilde{U}_n) = 0$ approximates the right boundary condition $u'(1) = 0$. For $i = 2, \ldots, n-1$, $\Psi_i(\tilde{U}_1, \ldots, \tilde{U}_n) = 0$ finally approximates the differential equation in the point x_i, i.e. $-u''(x_i) = f(x_i)$ ■ . In general it is not this

simple, that the i'th equation in the FDM is an approximation of an equation from the DEP in the i'th nodal point x_i. In a given case, the point may not even be a nodal point. Generally we shall denote these optimal approximation points by $y_1, \ldots, y_n$ and we shall write the equations from the DEP being approximated by the equations $\Psi_i = 0$ of the FDM as

$$F_i(y_i, u(y_i), u'(y_i), u''(y_i), \ldots, u^{(L)}(y_i)) = 0, \text{ for } i = 1, \ldots, n, \tag{3.44}$$

with L being the order of the differential equation in the DEP.

The Ψ_i's in (3.43) may be explicitly or implicitly given functions. So far we have only met explicitly given functions, but in section 4.5 we shall encounter also implicitly given Ψ_i's. For example $\Psi_1(\tilde{U}_1, \ldots, \tilde{U}_n) = \tilde{U}_1$ expresses Ψ_1 explicitly while $\tan(\Psi_1(\tilde{U}_1, \ldots, \tilde{U}_n)) = 0$ expresses Ψ_1 implicitly, even though it is fairly easy to make an explicit representation in this case simply by taking the inverse tangent on both sides ■. If (3.43) for all valid data can be written in the form

$$\tilde{U}_i = \psi_i(\tilde{U}_1, \ldots, \tilde{U}_{i-1}), \text{ for } i = 1, \ldots, n, \tag{3.45}$$

for explicitly given ψ_i's (possibly after a renumbering of the nodal points), then the FDM is said to be Explicit: For an explicit FDM the unknowns $\{\tilde{U}_i\}_{i=1}^n$ can be recovered one by one with only a function evaluation. Otherwise the FDM is called Implicit: For an implicit FDM a system of equations must be solved in order to recover the unknowns $\{\tilde{U}_i\}_{i=1}^n$. For example $\tilde{U}_i = 3\tilde{U}_{i-1} + \tilde{U}_{i-2}^2$ could be part of an explicit method while $\tilde{U}_i^2 = 3\tilde{U}_{i-1} + \tilde{U}_{i-2}$ normally would be part of an implicit method since the recovery of $\tilde{U}_i$ requires the solution of a nonlinear equation, although a simple one ■.

If (3.43) can be written in the form

$$\Psi_i(\tilde{U}_{j_i - s}, \ldots, \tilde{U}_{j_i}) = 0, \text{ for some } j_i \in \{1, \ldots, n\} \text{ and } i = 1, \ldots, n, \tag{3.46}$$

(possibly after a renumbering of the nodal points), then the FDM is said to be an s Step Method or a Multi Step Method for $s \geq 2$: For an s step method, at most $s + 1$ consequtive unknowns are involved in each equation. For example $\tilde{U}_3 + 3\tilde{U}_4 = 3$ could be part of a 1 step method while $\tilde{U}_3 + 3\tilde{U}_5 = 3$ would be part of a multi step method with at least 2 steps ■. Since $\tilde{U}_i$ is our approximation to the exact solution in the nodal point x_i, we simply count how many steps we need to go from the first nodal point involved in an equation to the last one.

3.3.1 Euler, Crank-Nicolson and Heun methods for $u' = f(x,u)$ – Optimal representations

In this section we give examples based on (1.10) on page 16 to clarify the optimality issues introduced above. We shall use a Uniform Step Length for our FDM's meaning that the nodal points are equidistributed i.e. $x_{i+1} - x_i = h$ for all $i = 1, \ldots, n-1$ and we shall take $I = (x_0, x_0 + (n-1)h)$ and $x_1 = x_0$. (Here x_0 is the position of the boundary condition while x_1 is the first of the nodal points for the FDM). (1.10) then takes the form

(3.47) Find $u \in \mathcal{C}^1(I)$: $u'(x) = f(x, u(x))\ \forall x \in I = (x_1, x_1 + (n-1)h)$, $u(x_1) = u^*$.

We shall use the notation of § 2.1 on page 19ff and § 3.1.1 on page 60ff except that here f is a function of 2 variables and we need to distinguish $f_i = f(x_i, u_i)$ from $\tilde{f}_i = f(x_i, \tilde{U}_i)$.

The class of DEP's that we shall consider is then (3.47) with any value of u^* and any bi-variate function f (including bi-variate functions that actually depend on only one of the variables) that is sufficiently smooth to allow the Taylor expansions of f and the solutions u needed to obtain the highest possible order of consistency. (If this is too vague, just assume f to be infinitely smooth). Let us consider some FDM's for this class of DEP's, that are so much in use that they have been given names.

Example 3.35 The Forward Euler Method

The forward Euler method is constructed by replacing the first derivative u_i' in (3.47) by $\delta_{+h} u_i$:

$$\tilde{U}_1 = u^*,\ \delta_{+h}\tilde{U}_{i-1} = \tilde{f}_{i-1} \Leftrightarrow \tilde{U}_i = \tilde{U}_{i-1} + h\tilde{f}_{i-1},\ i = 2, \ldots, n. \tag{3.48}$$

This is obviously an explicit one step method. The optimal representation is the first equation and the equations on the left side of the biimplication. Instead the first equation and the equations on the right side of the biimplication makes up a well scaled system adapted for practical computations. ■

Example 3.36 The Backward Euler Method

The backward Euler method is constructed by replaceing the first derivative u_i' in (3.47) by $\delta_{-h} u_i$:

$$\tilde{U}_1 = u^*,\ \delta_{-h}\tilde{U}_i = \tilde{f}_i \Leftrightarrow \tilde{U}_i = \tilde{U}_{i-1} + h\tilde{f}_i,\ i = 2, \ldots, n. \tag{3.49}$$

This is obviously an implicit one step method unless f does not depend on or is linear in its second variable ($\tilde{U}_i$). The optimal representation is the first equation and the equations on the left side of the biimplication. Instead the first equation and the equations on the right side of the biimplication makes up a well scaled system adapted for practical computations. ■

Note that FDM's generally can be expressed in many mathematically equivalent but notationally different ways. For example for (3.49) we have $\delta_{-h}\tilde{U}_i = \delta_{+h}\tilde{U}_{i-1}$ and the last equation may alternatively be expressed as $\tilde{U}_{i+1} = \tilde{U}_i + h\tilde{f}_{i+1},\ i = 1, \ldots, n-1$. With the notation we have chosen, i denotes the number of the equation in the equation system.

Example 3.37 The Crank-Nicolson Method

$$(3.50)\ \tilde{U}_1 = u^*,$$
$$\frac{1}{2}\left(\delta_{+h}\tilde{U}_{i-1} + \delta_{-h}\tilde{U}_i\right) = \frac{1}{2}\left(\tilde{f}_{i-1} + \tilde{f}_i\right) \Leftrightarrow \tilde{U}_i = \tilde{U}_{i-1} + \frac{h}{2}\left(\tilde{f}_{i-1} + \tilde{f}_i\right), \quad i = 2, \ldots, n.$$

Being the average of the forward and backward Euler methods this is obviously an implicit one step method unless f does not depend on or is linear in its second variable. The optimal representation is the first equation and the equations on the left side of the biimplication. Instead the first equation and the equations on the right side of the biimplication makes up a well scaled system adapted for practical computations. ■

Example 3.38 The Heun Method

$$(3.51)\ \tilde{U}_1 = u^*,$$
$$\frac{1}{2}\left(\delta_{+h}\tilde{U}_{i-1} + \delta_{-h}\tilde{U}_i\right) = \frac{1}{2}\left(\tilde{f}_{i-1} + \phi_i\right) \Leftrightarrow \tilde{U}_i = \tilde{U}_{i-1} + \frac{h}{2}\left(\tilde{f}_{i-1} + \phi_i\right), \quad i = 2, \ldots, n,$$

where $\phi_i = f(x_i, \tilde{U}_{i-1} + h\tilde{f}_{i-1})$. This is obviously an explicit one step method and to understand its origin, consider it an "explicitification" of the Crank-Nicolson method where the implicit part $\tilde{f}_i$ has been changed according to $\tilde{f}_i = f(x_i, \tilde{U}_i) \simeq f(x_i, u_i) = f(x_i, u_{i-1} + hu'_{i-1} + \ldots) = f(x_i, u_{i-1} + hf(x_{i-1}, u_{i-1}) + \ldots) \simeq f(x_i, \tilde{U}_{i-1} + hf(x_{i-1}, \tilde{U}_{i-1})) = \phi_i$. (Here we have used Taylor expansion and (1.10)).
The optimal representation is the first equation and the equations on the left

side of the biimplication. Instead the first equation and the equations on the right side of the biimplication makes up a well scaled system adapted for practical computations. ■

In order to handle more complex methods like the Heun method, we brush up on differentiation and the chain rule by stating the following results, where we shall use on several occations that $u'(x) = f(x, u(x))$:

$$f'(x, u(x)) = \partial_1 f + \partial_2 f \cdot u' = \partial_1 f + f\partial_2 f. \tag{3.52}$$

$$\begin{aligned} f''(x, u(x)) &= \partial_{11} f + 2\partial_{12} f \cdot u' + \partial_{22} f \cdot u' + \partial_2 f \cdot u'' \\ &= \partial_{11} f + 2f\partial_{12} f + f\partial_{22} f + f'\partial_2 f. \end{aligned} \tag{3.53}$$

Also defining $\phi_x(h) = f(x + h, u(x) + hf(x, u(x))$ the following derivatives are relevant

$$\phi'(0) = \partial_1 f + f\partial_2 f = f'(x, u(x)) \tag{3.54}$$

$$\begin{aligned} \phi''(0) &= \partial_{11} f + 2f\partial_{12} f + f^2\partial_{22} f \\ &= f''(x, u(x)) - f'(x, u(x))\partial_2 f(x, u(x)) \end{aligned} \tag{3.55}$$

The details of the derivation of these equalities are left to the reader as needed.

★ For advanced readers 3.39 ★ Optimal representative for the Euler, Crank-Nicolson and Heun methods. For those who have studied section 3.32 we give here some more details in the derivation of the optimal representations of the methods presented above:

The Forward Euler Method revisited

The details of the optimality according to definition 3.34 are as follows: The first equation in (3.48) is an approximation of the boundary condition (actually it is exact) so that $y_1 = x_1$, $\Phi_1 = \tilde{U}_1 - u^*$ and $F_1 = u(x_1) - u^*$ (compare to (3.42) and (3.40)). The remaining equations are approximations of the differential equation in various points and we have $F_i = u'(y_i) - f(y_i, u(y_i))$ for $i = 2, \ldots, n$. To find the optimal approximation points y_i and the optimal representation Φ_i, we shall take $\Phi_i^{c_i, r_i} = c_i h^{r_i} \left(\delta_{+h} \tilde{U}_{i-1} - \tilde{f}_{i-1} \right)$ for some constants c_i and r_i, covering all representations in the representation class

generated by (3.48). We then see by Taylor expansion that

$$
\begin{aligned}
(3.56)\ & \Phi_i^{c_i,r_i}(v(x_1),\dots,v(x_n)) - F_i(y_i, v(u_i), v'(y_i)) \\
&= c_i h^{r_i}\left(\frac{v(x_i)-v(x_{i-1})}{h} - f(x_{i-1}, v(x_{i-1}))\right) \\
&\quad -(v'(y_i) - f(y_i, v(y_i))) \\
&= c_i h^{r_i-1}\Bigg(v(y_i) + (x_i-y_i)v'(y_i) + \frac{(x_i-y_i)^2}{2}v''(y_i) + \dots \\
&\qquad\qquad -v(y_i) - (x_{i-1}-y_i)v'(y_i) - \frac{(x_{i-1}-y_i)^2}{2}v''(y_i) - \dots\Bigg) \\
&\quad -v'(y_i) + f(y_i, v(y_i)) \\
&\quad -c_i h^{r_i}\Big(f(y_i, v(y_i)) + (x_{i-1}-y_i)\big(\partial_1 f(y_i, v(y_i)) \\
&\qquad\qquad +v'(y_i)\partial_2 f(y_i, v(y_i))\big) + \dots\Big) \\
&= c_i h^{r_i}\Big(v'(y_i) + (\frac{x_i+x_{i-1}}{2} - y_i)v''(y_i) + \dots\Big) \\
&\quad -v'(y_i) + f(y_i, v(y_i)) \\
&\quad -c_i h^{r_i}\Big(f(y_i, v(y_i)) + (x_{i-1}-y_i)\big(\partial_1 f(y_i, v(y_i)) \\
&\qquad\qquad +v'(y_i)\partial_2 f(y_i, v(y_i))\big) + \dots\Big).
\end{aligned}
$$

We need to select c_i, r_i and y_i so that this has the highest possible order for all smooth v and an arbitrary f. Since v is not necessarily a solution to the differential equation, we can not depend on $-v'(y_i) + f(y_i, v(y_i))$ being zero and hence this term is generally of order 0 ($\mathcal{O}_{h\to 0}(h^0)$). We can do better than that only by taking $c_i = 1$ and $r_i = 0$ resulting in

$$
\begin{aligned}
(3.57)\quad & \Phi_i^{1,0}(v(x_1),\dots,v(x_n)) - F_i(y_i, v(u_i), v'(y_i)) \\
&= (\frac{x_i+x_{i-1}}{2} - y_i)v''(y_i) \\
&\quad -(x_{i-1}-y_i)\big(\partial_1 f(y_i, v(y_i)) + v'(y_i)\partial_2 f(y_i, v(y_i))\big) + \dots \\
&= \mathcal{O}_{h\to 0}(h^q) \\
&\quad \text{if } y_i = \frac{x_i+x_{i-1}}{2} + \mathcal{O}_{h\to 0}(h^q) \wedge y_i = x_{i-1} + \mathcal{O}_{h\to 0}(h^q).
\end{aligned}
$$

But $\frac{x_i+x_{i-1}}{2} = x_{i-1} + \frac{h}{2} = x_{i-1} + \mathcal{O}_{h\to 0}(h)$ so that the largest obtainable q is $q = 1$. In conclusion we arrive at the optimal selections $\Phi_i = \delta_{+h}\tilde{U}_{i-1} - \tilde{f}_{i-1}$ and $y_i = x_{i-1}$ for $i = 2,\dots,n$ corresponding to the left hand side of the biimplication in (3.48). Instead the right hand side is part of a non optimal

representation of the forward Euler method. Obviously all selections of y_i within $\mathcal{O}_{h\to 0}(h)$ neighborhoods of x_{i-1} would work as well, so that we might select $y_i = x_i$ or $y_i = x_{\max\{n,i+5\}}$ whereas $y_i = x_n$ would *not* be be optimal.

Having done the algebra once, it is normally easy to recognize the optimal representation for an FDM for a DEP. We just need to select the representation giving the best approximation to the differential equations and boundary conditions being approximated. The difference operators then eliminate the differential operators up to the order of consistency of the difference operators.

The Backward Euler Method revisited

A short version of the details about the optimality according to definition 3.34 are as follows: The first equation in (3.49) is an approximation of the boundary condition (actually it is exact) so that $y_1 = x_1$, $\Phi_1 = \tilde{U}_1 - u^*$ and $F_1 = u(x_1) - u^*$ (compare to (3.42) and (3.40)). Also $y_i = x_i$, $\Phi_i = \delta_{-h}\tilde{U}_i - \tilde{f}_i$ and $F_i = u'(x_i) - f(x_i, u(x_i))$ for $i = 2, \ldots, n$ may be verified as above. Hence, the left hand side of the biimplication in (3.49) together with the initial condition gives an optimal representation of the backward Euler method, while the right hand side is part of a non optimal representation. Again $y_i = x_i$ is by no means a unique choice. y_i may be changed within a $\mathcal{O}_{h\to 0}(h)$ neighborhood without losing the optimality.

The Crank-Nicolson Method revisited

A short version of the details about the optimality according to definition 3.34 are as follows: The first equation in (3.50) is an approximation of the boundary condition (actually it is exact) so that $y_1 = x_1$, $\Phi_1 = \tilde{U}_1 - u^*$ and $F_1 = u(x_1) - u^*$ (compare to (3.42) and (3.40)). Also $y_i = x_{i-\frac{1}{2}} := x_i - \frac{h}{2}$ for $i = 2, \ldots, n$ is an optimal choice. Then $\Phi_i = \delta_{+h}\tilde{U}_{i-1} - \frac{1}{2}\left(\tilde{f}_{i-1} + \tilde{f}_i\right)$ and $F_i = u'(x_{i-\frac{1}{2}}) - f(x_{i-\frac{1}{2}}, u(x_{i-\frac{1}{2}}))$ for $i = 2, \ldots, n$. Again, the left hand side of the biimplication in (3.50) together with the initial condition gives an optimal representation of the Crank Nicolson method, while the right hand side is part of a non optimal representation.

The Heun Method revisited

A short version of the details about the optimality according to definition 3.34 are as follows: The first equation in (3.51) is an approximation of the boundary condition (actually it is exact) so that $y_1 = x_1$, $\Phi_1 = \tilde{U}_1 - u^*$ and $F_1 = u(x_1) - u^*$ (compare to (3.42) and (3.40)). Treating Heun's method the same way as Crank-Nicolson, we take $y_i = x_{i-\frac{1}{2}} := x_i - \frac{h}{2}$,

$\Phi_i = \delta_{+h}\tilde{U}_{i-1} - \frac{1}{2}\left(\tilde{f}_{i-1} + \phi_i\right)$ and $F_i = u'(x_{i-\frac{1}{2}}) - f(x_{i-\frac{1}{2}}, u(x_{i-\frac{1}{2}}))$ for $i = 2, \ldots, n$. Again, the left hand side of the biimplication in (3.51) together with the initial condition gives an optimal representation of the Heun method, while the right hand side is part of a non optimal representation. ★

3.4 Error analysis for FDM's: Consistency, convergence and stability

Obviously (see also §3.1.1), when we approximate a DEP with an FDM, we do not get exact results, i.e. $u(x_i)$ is not necessarily the same value as $\tilde{U}_i$. To be able to distinguish between different FDM's we are interested in assessing the n-dimensional Error vector $\mathbf{e} = \mathbf{u} - \tilde{\mathbf{U}}$ where $\mathbf{u} = (u(x_1), \ldots, u(x_n))^T$. If we could compute $\mathbf{e}$ then we would also know $\mathbf{u}$, i.e. the exact solution in the nodal points. This is not possible. Instead we may compute an approximation to $\mathbf{e}$. Methods (depending on $\tilde{\mathbf{U}}$) for doing this is developed in the A Posteriori Error Analysis which we shall not consider here. Instead we shall concentrate on the A Priori Error Analysis where we investigate the error based on knowledge about the DEP. We subdivide into the Aymptotic Analysis dealing with the asymptotic behavior of the error, $\lim_{h\to 0} \|\mathbf{e}\|$, where h is a typical (step length) parameter for the FDM and the Non Asymptotic Analysis dealing with the error for finite values of h.

For the a priori error analysis, we need to introduce some concepts based on (3.42) and (3.40):

Definition 3.40 *Let*

$$\mathbf{FDM}(\tilde{\mathbf{U}}) = (\Phi_1(\tilde{U}_1, \ldots, \tilde{U}_n), \ldots, \Phi_n(\tilde{U}_1, \ldots, \tilde{U}_n))^T = \mathbf{0} \tag{3.58}$$

be an optimal representation for the n equations in an FDM and let

$$\begin{aligned}\mathbf{DEP}(u) = (&F_1(y_1, u(y_1), u'(y_1), u''(y_1), \ldots, u^{(L)}(y_1)), \ldots, \\ &F_n(y_n, u(y_n), u'(y_n), u''(y_n), \ldots, u^{(L)}(y_n)))^T = \mathbf{0}\end{aligned} \tag{3.59}$$

be the equations in the class of DEP's that are being approximated by the FDM.
Define the Vector of Local Truncation Errors for the FDM wrt. the class of DEP's by

$$\boldsymbol{\tau} = \sup_{\boldsymbol{F}} \Big[\mathbf{FDM}(u) - \mathbf{DEP}(u)\Big] = \sup_{\boldsymbol{F}} \Big[\mathbf{FDM}(u)\Big]. \tag{3.60}$$

Note the replacement of the approximate solution values $\tilde{U}_j$ by the exact solution values $u(x_j)$ for $j = 1, \ldots, n$ in FDM.
Define the Global Truncation Error for the FDM wrt. the class of DEP's by

$$\begin{aligned} \tau &= \max_{i=1,\ldots,n} |\tau_i| = \|\boldsymbol{\tau}\|_\infty = \sup_{\boldsymbol{F}} \|\mathbf{FDM}(u) - \mathbf{DEP}(u)\|_\infty \\ &= \sup_{\boldsymbol{F}} \|\mathbf{FDM}(u)\|_\infty. \end{aligned} \tag{3.61}$$

Define the Global Order of Consistency for the FDM wrt. the class of DEP's as the biggest number q such that $\tau = \mathcal{O}_{h\to 0}(h^q)$.
We say that the FDM is Consistent with the class of DEP's iff $\tau \to 0$ as $h \to 0$ which happens in particular if $q > 0$. Otherwise it is inconsistent.
Define the Global Order of Convergence for the FDM towards the class of DEP's, measured in the ℓ^∞-norm as the biggest number $p \geq 0$ such that $\sup_{\boldsymbol{F}} \|\mathbf{e}\|_\infty = O_{h\to 0}(h^p)$, where $\mathbf{e} = \mathbf{u} - \tilde{\mathbf{U}}$ with $\mathbf{u}_i = u(x_i)$ and $\tilde{\mathbf{U}}_i = \tilde{U}_i$ for $i = 1, \ldots, n$.
We say that the FDM is Convergent to the class of DEP's in the ℓ^∞-norm iff $\sup_{\boldsymbol{F}} \|\mathbf{e}\|_\infty \to 0$ as $h \to 0$ (in particular if $p > 0$). Otherwise it is non convergent.
Convergence is defined similarly for other measures than the ℓ^∞-norm. Note that the order of convergence may depend on the choice of measure.

The $\sup_{\boldsymbol{F}}$ occurring in various places in definition 3.40 is over all smooth data of the class of DEP's being considered. For example, for $u'(x) = f(x, u(x))$ the right hand side function f is data, and we only consider functions that are sufficiently smooth to allow the necessary Taylor expansions to compute the order of consistency. Typically we omit the supremum from the notation, implicitly assuming (1) that the data is sufficiently smooth that we do not need to worry about it and (2) that f does not have any special properties apart from those expressed by the DEP (like $u'(x) = f(x, u(x))$). That is, we can not make difficult terms "disappear" from our Taylor expansions by assuming that f is selected so that they are zero.

★ For advanced readers 3.41 ★ Local consistency and convergence. Not only global consistency and convergence but also local consistency and convergence may be defined along the lines of definition 3.40. For completeness, we shall give the definition here, but emphasize, that the local definitions will not be used in this note, and hence the following definition can be omitted at the readers discretion:

For the local definitions it is important to recall the discussion in connection to figure 3.3 on page 78. For example local consistency in a point

z involves a sequence of local truncations errors τ_i where i changes (goes to ∞) as $h \to 0$ in order that $x_i \to z$ ■ .

Definition 3.42 *Let z be any point in the domain of definition of u and let $\{x_{i_k}^{(k)}\}_{k=1}^{\infty}$ be any sequence of nodal points converging to z as $k \to \infty$.*
Define the Local Order of Consistency for the FDM wrt. the class of DEP's in the point z as the biggest number $q_z \geq 0$ such that $\{\tau_{i_k}^{(k)}\}_{k=1}^{\infty} = \mathcal{O}_{k\to\infty}((h^{(k)})^{q_z})$, i.e. $\{\tau_{i_k}^{(k)}/(h^{(k)})^{q_z}\}_{k=1}^{\infty}$ is a bounded sequence. We say that the FDM is locally consistent with the class of DEP's in the point z if $\tau_{i_k}^{(k)} \to 0$ as $k \to \infty$. Otherwise, the FDM is locally inconsistent with the class of DEP's in z.
Define the Local Order of Convergence for the FDM towards the class of DEP's, in the point z as the biggest number $p_z \geq 0$ such that $\{\sup_{\boldsymbol{F}} |u(z) - \tilde{U}_{i_k}^{(k)}|\}_{k=1}^{\infty} = \mathcal{O}_{k\to\infty}((h^{(k)})^{p_z})$, i.e. $\{\sup_{\boldsymbol{F}} |u(z) - \tilde{U}_{i_k}^{(k)}|/(h^{(k)})^{p_z}\}_{k=1}^{\infty}$ is a bounded sequence. We say that the solution of the FDM is converging to the solution of the DEP in z if $|u(z) - \tilde{U}_{i_k}^{(k)}| \to 0$ as $k \to \infty$. Otherwise, the solution of the FDM is not converging to the solution of the DEP in z.

★

Note that while error and convergence measures the difference between the exact and the approximated solution, truncation error and consistency measures the difference between the DEP and the optimal representation of the FDM (allthough of course $\mathbf{DEP}(u) = 0$ by definition). Obviously these notions are different, but they are connected by the notion of zero stability:

Definition 3.43 *Let ϵ be any positive real number, and $\delta_{\epsilon,1}, \ldots, \delta_{\epsilon,n}$ any real numbers satisfying $|\delta_{\epsilon,i}| < \epsilon$, for $i = 1, \ldots, n$. The problem*

$$\Phi_i(\tilde{Z}_{\epsilon,1}, \ldots, \tilde{Z}_{\epsilon,n}) = \delta_{\epsilon,i}, \text{ for } i = 1, \ldots, n, \tag{3.62}$$

is called an ϵ-Perturbation of (3.42) or (3.58).
The representation (3.42) of the FDM is Zero Stable in the ℓ^∞ norm (with respect to ϵ-perturbations) if for all ϵ-perturbations (3.62) (fixed ϵ but arbitrary δ's) $\exists h_0 > 0$, $\exists C > 0$ (independent of h, ϵ and $\delta_{\epsilon,i}$ for $i = 1, \ldots, n$) such that $|\tilde{U}_i - \tilde{Z}_{\epsilon,i}| < C\epsilon$ $\forall i = 1, \ldots, n$, $\forall h \in]0, h_0]$, i.e. for sets of nodal points, large enough to give sufficiently small step lengths.
More generally, zero stability in a sequence of distance measures $\{d_k\}_{k=1}^{\infty}$ requires $d_k(\tilde{\mathbf{U}} - \tilde{\mathbf{Z}}) < C\epsilon$ for all k sufficiently big.

Note that zero stability is nothing but continuous dependence on data for the FDM solutions and hence the discrete version of Liapunov stability.

Recalling (3.60), it is clear that the exact solution vector $(u_1, \ldots, u_n)^T$ is nothing but the solution to the perturbed problem (3.62), taking $\delta_{\epsilon,i} = \tau_i$. This is the key to the proof af the following theorem which is the central convergence theorem of these notes:

Theorem 3.44 *The Convergence Theorem of Lax*
An FDM which is consistent of order q with a class of DEP's and has an optimal representation which is zero stable in a sequence of distance measures is also convergent of order q in that sequence of distance measures to the solution to the DEP.
Or in short form as a help for memorizing:
for consistent methods, zero stability implies convergence.

Proof:
Consider a particular ϵ-perturbation in (3.62), taking $\delta_{\epsilon,i} = \tau_i$ for $i = 1, \ldots, n$ which is allowed since (3.42) is consistent and which allows $\epsilon = \tau = \mathcal{O}(h^q)$. But for this particular perturbation $\tilde{Z}_{\epsilon,i} = u(x_i)$ for $i = 1, \ldots, n$. The zero stability of the FDM then shows by definition 3.43 that for sufficiently small h there exists a positive constant C such that

$$|u(x_i) - \tilde{U}_i| < C\tau = \mathcal{O}_{h\to 0}(h^q) \text{ for } i = 1, \ldots, n, \tag{3.63}$$

where the equality comes directly from consistency. Using this result on a sequence of points $\{x_{i_k}^{(k)}\}_{k=1}^{\infty}$ converging to any point $x \in I$ we have pointwise convergence in x. ■

Note that zero stability is uniform in the sense that it applies with the same C and ϵ to *all* nodal points $x_i^{(k)}$ whenever $h^{(k)} < h_0$ i.e. in the box $I \times]0, h_0]$. Hence in (3.63) we have also proven a sort of semi uniform convergence. (Full uniform convergence, i.e. small error in all nodal points below a certain h_0, would also require uniform consistency. For normal convergence, the h_0 will depend on the sequence of nodal points that we are following towards convergence). The semi uniform convergence that follows from zero stability again implies zero stability: Just note that the τ's can really be anything. Renaming them to ϵ's (and renaming the $u(x_i)$ to $\tilde{Z}_{\epsilon,i}$) we recover the zero stability. Hence zero stability is not only sufficient but also necessary to obtain (semi) uniform convergence. We shall here not get into further details about uniform convergence. We shall settle for the non uniform convergence defined above.

Showing consistency of an FDM is normally little more than showing consistency of the difference operators replacing the differential operators. The rest of the way to convergence is then covered by the notion of zero stability. We postpone convergence and zero stability to the following sections and consider here only consistency.

3.4.1 Euler, Crank-Nicolson and Heun methods for $u' = f(x,u)$ – Consistency

Let us consider the issue of consistency for the 4 FDM's for (3.47) from examples 3.35–3.38 on page 97:

Example 3.45 Forward Euler
$\boldsymbol{\tau} = (0, \{[\delta_{+h}u(x_i) - f(x_i, u(x_i))] - [u'(x_i) - f(x_i, u(x_i))]\}_{i=1}^{n-1})^T = \mathcal{O}_{h\to 0}(h)$ by (3.21). ■

Example 3.46 Backward Euler
$\boldsymbol{\tau} = (0, \{[\delta_{-h}u(x_i) - f(x_i, u(x_i))] - [u'(x_i) - f(x_i, u(x_i))]\}_{i=2}^{n})^T = \mathcal{O}_{h\to 0}(h)$ by (3.22). ■

Example 3.47 Crank Nicolson
$\boldsymbol{\tau} = (0, \{[\delta_{+h}u_i - \frac{f(x_i,u(x_i))+f(x_{i+1},u(x_{i+1}))}{2}] - [u'(x_{i+\frac{1}{2}}) - f(x_{i+\frac{1}{2}}, u(x_{i+\frac{1}{2}}))]\}_{i=1}^{n-1})^T$
$= \mathcal{O}_{h\to 0}(h^2)$ by exercise 3.49. ■

Example 3.48 Heun's method
$\boldsymbol{\tau} = (0, \{[\delta_{+h}u_i - \frac{f(x_i,u(x_i))+f(x_{i+1},u(x_i)+hf(x_i,u(x_i)))}{2}] -$
$[u'(x_{i+\frac{1}{2}}) - f(x_{i+\frac{1}{2}}, u(x_{i+\frac{1}{2}}))]\}_{i=1}^{n-1})^T = \mathcal{O}_{h\to 0}(h^2)$ by exercise 3.50. ■

Note the importance of using optimal representations $\boldsymbol{\Phi}(\tilde{U}_1, \ldots, \tilde{U}_n) = \mathbf{0}$ for the calculation of the orders of consistency. Otherwise, considering any representation $\boldsymbol{\Psi}(\tilde{U}_1, \ldots, \tilde{U}_n) = \mathbf{0}$ obtained by multiplying the optimal representation with some power of h, we could get any orders of $\boldsymbol{\Psi}(u_1, \ldots, u_n)$.

Exercise 3.49
Use Taylor expansion to prove the consistency result for the Crank Nicolson method from example 3.47. ■

Exercise 3.50
Use Taylor expansion to prove the consistency result for Heun's method from example 3.48. ■

Exercise 3.51
What difference would it make for the order of consistency, to consider the Crank Nicolson method an approximation in x_i instead of $x_{i+\frac{1}{2}}$. ■

Chapter 4

FDM's for $u' = f(x, u)$

4.1 Convergence and stability of explicit, one step FDM's for $u' = f(x, u)$

We shall here consider only uniform step length, explicit, one step FDM's for (3.47), repeated here again for reference:

$$\text{Find } u \in \mathcal{C}^1(I): \ u'(x) = f(x, u(x)) \ \forall x \in I = (x_1, x_1 + (n-1)h), \\ u(x_1) = u^*.$$

Further, we shall consider only FDM's that can be written in the form

$$\tilde{U}_1 = \tilde{U}^*, \ \tilde{U}_i = \tilde{U}_{i-1} + h\phi(\tilde{U}_{i-1}), \ i = 2, \ldots, n, \tag{4.1}$$

where the nodal points are distributed uniformly, i.e. $x_i = x_1 + (i-1)h$, $i = 1, \ldots, n$ and $I = (x_1, x_n)$. Here, apart from $\tilde{U}_i$, the Increment Function ϕ may also depend on known factors such as f, h, $x_1, \ldots, x_n$ etc.

We shall assume that (4.1) is consistent of global order q. To use definition 3.40 on page 102 we first need to put (4.1) in the form of an optimal representation, i.e. in the form of an approximation to (1.10). This is easily done rewriting (4.1) into

$$\tilde{U}_1 = \tilde{U}^*, \ \delta_{+h}\tilde{U}_{i-1} = \phi(\tilde{U}_{i-1}), \ i = 2, \ldots, n. \tag{4.2}$$

The first equation is obviously an optimal approximation of the initial condition in x_1. Regarding the other equations, the one with index i is an optimal approximation of the differential equation in (3.47) in the point x_{i-1}. Hence consistence of global order q by definition 3.40 on page 102 turns into the

conditions:

$$
\begin{aligned}
(4.3) \qquad \tau_1 &= (u(x_1) - \tilde{U}^*) - (u(x_1) - u^*) = u^* - \tilde{U}^* = \mathcal{O}_{h\to 0}(h^q), \\
\tau_i &= (\delta_+ u(x_{i-1}) - \phi(u(x_{i-1}))) - (u'(x_{i-1}) - f(x_{i-1}, u(x_{i-1}))) \\
&= \delta_+ u(x_{i-1}) - \phi(u(x_{i-1}))) = \mathcal{O}_{h\to 0}(h^q),\ i = 2, \ldots, n.
\end{aligned}
$$

The observations about what equations in (4.2) corresponds to what equations in (3.47) are of course superfluous since the left hand side of the differential equation is zero in any point. This was also noted (for the advanced readers) already in section 3.3 below definition 3.34 on page 94.

The ϵ-perturbation of the optimal representation (4.2) (see definition (3.43) on page 104) is

$$
(4.4) \qquad \tilde{Z}_{\epsilon,1} = \tilde{U}^* + \delta_{\epsilon,1},\ \delta_{+h}\tilde{Z}_{\epsilon,i-1} = \phi(\tilde{Z}_{\epsilon,i-1}) + \delta_{\epsilon,i},\ i = 2, \ldots, n.
$$

Here $|\delta_{\epsilon,i}| < \epsilon$ for $i = 1, \ldots, n$.

The discrete result corresponding to theorem 1.16 on page 16 is

Theorem 4.1 *If ϕ is globally Lipschitz continuous, say with constant Λ independent of h and $x_1, \ldots, x_n$, then the optimal representation (4.2) for the explicit, one step FDM's for (3.47), is Zero Stable.*

★ For advanced readers 4.2 ★

Proof:

Summing over $i = 2, \ldots, j$ where $j \le n$ in (4.2) and (4.4) we get, noting the telescoping sum on the left hand side,

$$
\begin{aligned}
&\frac{1}{h}(\tilde{U}_j - \tilde{U}_1) = \sum_{i=2}^{j} \phi(\tilde{U}_{i-1}) \text{ and } \frac{1}{h}(\tilde{Z}_{\epsilon,j} - \tilde{Z}_{\epsilon,1}) = \sum_{i=2}^{j} \Big(\phi(\tilde{Z}_{\epsilon,i-1}) + \delta_{\epsilon,i}\Big) \\
&\Downarrow \\
&\frac{1}{h}(\tilde{Z}_{\epsilon,j} - \tilde{U}_j) = \frac{1}{h}\delta_{\epsilon,1} + \sum_{i=2}^{j} \delta_{\epsilon,i} + \sum_{i=2}^{j} \Big(\phi(\tilde{Z}_{\epsilon,i-1}) - \phi(\tilde{U}_{i-1})\Big) \\
&\Downarrow \\
&|\tilde{Z}_{\epsilon,j} - \tilde{U}_j| \le |\delta_{\epsilon,1}| + h\sum_{i=2}^{j} |\delta_{\epsilon,i}| + h\Lambda \sum_{i=2}^{j} |\tilde{Z}_{\epsilon,i-1} - \tilde{U}_{i-1}| \\
&\Downarrow \\
&|\tilde{Z}_{\epsilon,j} - \tilde{U}_j| \le (|\delta_{\epsilon,1}| + h\sum_{i=2}^{j} |\delta_{\epsilon,i}|)e^{(j-1)h\Lambda} < (1 + (j-1)h)\epsilon e^{(j-1)h\Lambda} < C\epsilon,
\end{aligned}
$$

where $C = (1 + |I|)e^{|I|\Lambda}$ and where the first inequality on the last line comes from the Discrete Gronwall lemma in the form

$$\phi_1 \leq |c_1|,\ \phi_j \leq \sum_{i=1}^{j} |c_i| + \sum_{i=2}^{j} |b_i|\phi_{i-1},\ \text{for } j \geq 2 \tag{4.5}$$

$$\Rightarrow \phi_j \leq (\sum_{i=1}^{j} |c_i|)e^{\sum_{i=2}^{j}|b_i|},\ \text{for } j \geq 2,$$

taking $\phi_j = |\tilde{Z}_{\epsilon,j} - \tilde{U}_j|$, $c_1 = \delta_{\epsilon,1}$, $c_i = h\delta_{\epsilon,i}$, $b_i = h\Lambda$, $i = 2, \ldots, j$. (For a proof of Gronwall, see for example [16]). ■

Considering zero-stability for non optimal representations, each equation in (4.2) may be multiplicated with a different constant and power of h in the form $c_i h^{r_i}$. For zero-stability, the constants are irrelevant, but multiplying with powers of h would be equivalent to multiplicating each $\delta_{\epsilon,i}$ by the inverse power of h in (4.4). This corresponds to the consideration in (4.4) of perturbations of the form

$$\delta_{\epsilon,i} = h^{-r_i}\hat{\delta}_{\epsilon,i},\ \text{with } \hat{\delta}_{\epsilon,i} < \hat{\epsilon} \text{ for } i = 1, \ldots, n. \tag{4.6}$$

In the proof of theorem 4.1 we would then get

$$\begin{aligned} |\tilde{Z}_{\epsilon,j} - \tilde{U}_j| &< (h^{-r_1} + |I| \max\{h^{-r_2}, \ldots, h^{-r_n}\})e^{|I|\Lambda}\hat{\epsilon} \\ &\leq (h_0^{-r_1} + |I| \max\{h_0^{-r_2}, \ldots, h_0^{-r_n}\})e^{|I|\Lambda}\hat{\epsilon} \\ &\leq h_0^{\max\{-r_1,\ldots,-r_n\}}(1 + |I|)e^{|I|\Lambda}\hat{\epsilon}, \end{aligned} \tag{4.7}$$

for any $h_0 > h > 0$ as long as $-r_i \geq 0$, for all $i = 1, \ldots, n$. Instead if any of the r_i were to become positive, like in the representation (4.1) where $r_i = 1$ for $i = 2, \ldots, n$, we would not have a finite bound for all $h \in]0, h_0[$ for any $h_0 > 0$ and hence not have zero-stability. Hence also for zero-stability it is important to consider an optimal representation for the FDM. ★

Theorem 4.1 can easily be extended to a convergence result:

Theorem 4.3 *Consider the FDM with representation* (4.1) *and optimal representation* (4.2) *assumed to have global order of consistency* $q \geq 0$.
ϕ globally Lipschitz continuous $\Rightarrow$ (4.2) *is Zero Stable $\Rightarrow$ the FDM is convergent of order q in all points of I.*

Proof:
The first implication is simply theorem 4.1 and the second implication simply theorem 3.44 on page 105. ■

Now consider the 4 FDM's for (3.47) from examples 3.35–3.38 on page 97:

Example 4.4 Forward Euler
Here $\phi(\tilde{U}_i) = f(x_i, \tilde{U}_i)$ so that if f is Lipschits in its second variable then forward Euler is Zero stable and hence with the consistency result of example 3.45 on page 106 forward Euler is convergent of order 1. ■

Example 4.5 Backward Euler
This method is implicit and can not be treated with the method of this section. It will be considered below when the general theory for multi step methods is developed. ■

Example 4.6 Crank Nicolson
This method is implicit and can not be treated with the method of this section. It will be considered below when the general theory for multi step methods is developed. ■

Example 4.7 Heun's method
Here $\phi(\tilde{U}_i) = \frac{1}{2}(f(x_i, \tilde{U}_i) + f(x_{i+1}, \tilde{U}_i + hf(x_i, \tilde{U}_i))$ so that if f is Lipschits continuous in its second variable with Lipschitz constant L then ϕ is Lipschits continuous with Lipschitz constant $\Lambda = L + hL^2/2$ so that Heun's method is Zero stable and hence with the consistency result of example 3.48 on page 106 Heun's method is convergent of order 2. ■

Exercise 4.8
Verify the Lipschitz constant of Heun's method stated in example 4.7. ■

Exercise 4.9
Write out the Crank-Nicolson method for (1.10) in the case $I = (0, 4)$, $f(t, u(t)) = \cos(t)$, $t_0 = 0$ and $u^* = 0$. Program, solve and compare the numerical solutions graphically to the exact solution $u(t) = \sin(t)$. Do you believe that Crank-Nicolson converges? ■

Exercise 4.10
Repeat exercise 4.9 for

1. Forward Euler
2. Backward Euler
3. Heun

Do you believe that Backward Euler converges? ■

4.2 Non asymptotic error analysis – Absolute stability for FDM's for $u' = \lambda u$

For a numerical method that has to be evaluated on a computer, "convergence is not everything". Convergence tells us that if we select h small enough, then the error is small. Unfortunately, we do not get a value for when h is small enough. Even more unfortunate, the behavior of a numerical method may depend dramatically on the value of h. As an exampe of this, consider the following

Example 4.11 Forward Euler for $u'(t) = -5u(t),\ t > 0,\ u(0) = 1$
The exact solution to the DEP is $u(t) = e^{-5t}$ and the forward Euler method takes the form $\tilde{U}_1 = 1,\ \tilde{U}_i = \tilde{U}_{i-1} - 5h\tilde{U}_{i-1} = (1-5h)\tilde{U}_{i-1},\ i = 2, 3, \ldots$ where the nodal points are $x_i = (i-1)h,\ i = 1, 2, \ldots$. The solution to this difference equation is easily found by iteration: $\tilde{U}_i = (1 - 5h)\tilde{U}_{i-1} = (1 - 5h)^2\tilde{U}_{i-2} = \ldots = (1 - 5h)^{i-1}\tilde{U}_1 = (1 - 5h)^{i-1}$.
We note that

- $\tilde{U}_i$ oscillates with i (changes sign from i even to i odd) with increasing amplitude for $h > \frac{2}{5}$
- $\tilde{U}_i$ oscillates with i with constant amplitude for $h = \frac{2}{5}$
- $\tilde{U}_i$ oscillates with i with decreasing amplitude for $\frac{1}{5} < h < \frac{2}{5}$
- $\tilde{U}_i$ is identicaly 0 (for $i \geq 2$) for $h = \frac{1}{5}$
- $\tilde{U}_i$ goes exponentially towards 0 for $h < \frac{1}{5}$

In figure 4.1 on page 112 is shown various plots generated with Maple code similar to the one shown in figure 4.2 on page 113. It is clear that $h \geq \frac{2}{5}$ is directly unacceptable since the general form of the numerical solution graphs are not similar at all to the form of the exact solution graph and also the errors are very big. For $\frac{1}{5} \leq h < \frac{2}{5}$ the oscillations are still misleading, but if large errors are allowed this case could be acceptable. Not until $h < \frac{1}{5}$ can we say that the numerical solution has found a stable form that is reminiscent of that taken by the exact solution, and which is maintained until convergence.
■

To capture at least part of the situation described in the example above we introduce yet another stability notion: Absolute stability. Since this is normally done using complex numbers, we shall start giving a short brush up on complex numbers and the notation used in that connection.

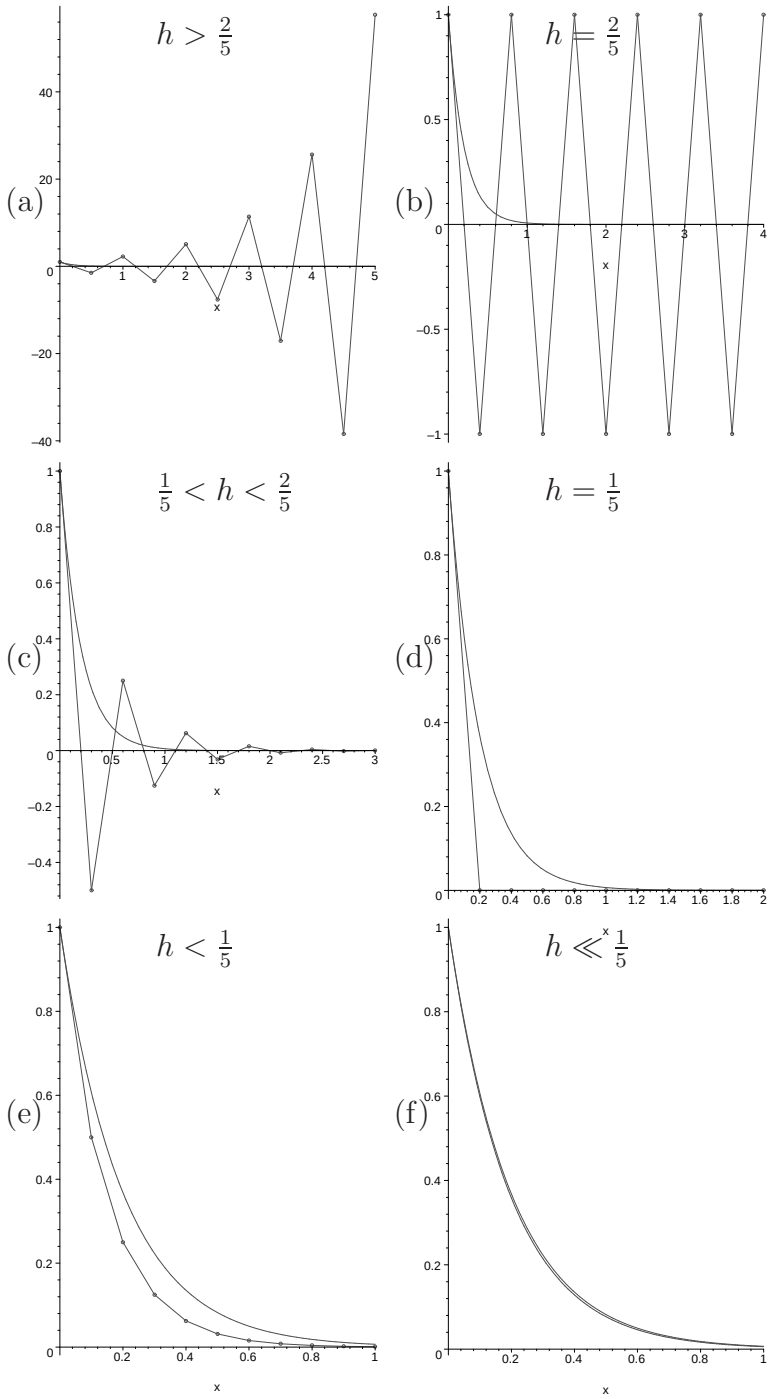


Figure 4.1: Exact ($u(t)$) and numerical ($\{\tilde{U}_i\}_{i=1}^{10}$) solutions for example 4.11 with a: $h = 0.5$, b: $h = 0.4$, c: $h = 0.3$, d: $h = 0.2$, e: $h = 0.1$, f: $h = 0.01$. (For (f), $\{\tilde{U}_i\}_{i=1}^{100}$ are plotted)

```
h:=0.5: n:=10:
l := [seq([i*h,(1-5*h)^i],i=0..n)]:
p1:=plot(l, x=0..n*h, style=line):
p1x:=plot(l, x=0..n*h, style=point,symbol=circle):
p2:=plot(exp(-5*x),x=0..n*h):
with(plots):
display([p1,p1x,p2]);
```

Figure 4.2: Maple code for plot (a) in figure 4.1

Example 4.12 Complex numbers

The space of complex numbers, $\mathbb{C}$, consists of all tuples (x, y) so that x and y are real numbers, i.e. $x, y \in \mathbb{R}$. Hence any complex number z can be identified with some real tuple (x, y), i.e. $z = (x, y)$. Another popular notation for the same is $z = x+iy$ where i is called the Imaginary unit. In many connections i behaves as if it was really $\sqrt{-1}$, but of course this similarity must be treated with care since $\sqrt{-1}$ is not defined at all. Instead it is solid enough to define $i^2 = -1$.

If $z = (x, y)$ or $z = x + iy$, x is denoted the real part of z or $\mathcal{R}e(z)$ and y is denoted the imaginary part of z or $\mathcal{I}m(z)$ so that we may also write $z = (\mathcal{R}e(z), \mathcal{I}m(z))$ or $z = \mathcal{R}e(z) + i\mathcal{I}m(z)$.

Graphically complex numbers are typically plotted in the Complex plane which is a standard (x, y) coordinate system, with the real part x on the horisontal axis, which is then also denoted the real axis, and the imaginary part y on the vertical axis, which is then also denoted the imaginary axis. Interpreting the tuple (x, y) as a vector going from $(0, 0)$ to (x, y) and having length $\sqrt{x^2 + y^2}$ we define the length of a complex number $z = (x, y)$ as $|z| = \sqrt{x^2 + y^2}$. Below, we shall need the negative complex half plane $\mathbb{C}_- = \{z \in \mathbb{C} : \mathcal{R}e(z) < 0\}$.

A few computational rules are convenient to know: First we introduce for any complex number $z = x + iy$ the complex conjugate of z denoted $\bar{z}$ and defined by $\bar{z} = x - iy$. Now $|\bar{z}| = |z| = \sqrt{z\bar{z}}$ and $\left|\frac{1}{z}\right| = \frac{1}{|z|}$. The first is obvious knowing the multiplication rule $(x+iy)(a+ib) = (xa-yb)+i(xb+ya)$ while the last requires a little computation: $\frac{1}{z} = \frac{z\bar{z}}{|z|^2 z} = \frac{\bar{z}}{|z|^2} \Rightarrow \left|\frac{1}{z}\right| = \left|\frac{\bar{z}}{|z|^2}\right| = \frac{1}{|z|}$. For the last equality sign we need the scaling rule $|az| = |a||z|$ where z is any complex number and a is any real number.

Finally define for a complex number $z = x + iy$ that $e^z = e^{|z|}(\cos x + i \sin y)$.

■

Definition 4.13 *A numerical method with solution $\tilde{U}_i(h,\lambda)$ for $i = 1, 2, \ldots$ for the DEP $u'(t) = \lambda u(t),\ t > 0,\ u(0) = 1$ (with solution $u(t) = e^{\lambda t}$) for some growth parameter $\lambda \in \mathbb{C}$ and uniform step length parameter h is Absolutely Stable iff $\lim_{i\to\infty} |\tilde{U}_i(h,\lambda)| = 0$. ($h$ and λ are fixed and we study what happens as the nodal points move towards ∞).*
The Region of Absolute Stability is $\mathcal{A} = \{z = h\lambda \in \mathbb{C} :\ \lim_{i\to\infty} |\tilde{U}_i| = 0\}$ and the numerical method is called $\mathcal{A}$-stable if $\mathcal{A} \supset \mathbb{C}_- = \{z \in \mathbb{C} : \mathcal{R}e(z) < 0\}$.
Of practical importance is also $\mathcal{A}_\lambda = \{h > 0 :\ \lim_{i\to\infty} |\tilde{U}_i(h,\lambda)| = 0\}$.

Note that while absolute stability, for $\mathcal{R}e(\lambda) < 0$ gives some indication of reasonability of the numerical solution, the same is not the case for $\mathcal{R}e(\lambda) \geq 0$ where the exact solution is actually not converging to 0.

As examples, we consider the 4 methods from examples 3.35–3.38 on page 97.

Example 4.11 (Continued) Forward Euler for $u'(t) = \lambda u(t),\ t > 0,\ u(0) = 1$
As above, it is easily seen that $\tilde{U}_i = (1 + h\lambda)^{i-1},\ i \geq 1$, i.e. $\lim_{i\to\infty} |\tilde{U}_i| = 0 \Leftrightarrow |1 + h\lambda| < 1$. If we start considering only real λ then $|1 + h\lambda| < 1 \Leftrightarrow 1 + h\lambda < 1 \wedge 1 + h\lambda > -1 \Leftrightarrow h\lambda \in]-2, 0[$ i.e. $\mathcal{R}e(\mathcal{A}) =]-2, 0[$. For complex λ we instead have $|1 + h\lambda| < 1 \Leftrightarrow \mathcal{A} = \{h\lambda = -1 + z :\ |z| < 1\} = B_{(-1,0)}(1)$ (the open ball of radius 1 centered in $(-1, 0)$). Clearly, forward Euler is *not* $\mathcal{A}$-stable. Instead

$$\mathcal{A}_\lambda = \begin{cases}]0, -\frac{2\mathcal{R}e(\lambda)}{|\lambda|^2}[& \mathcal{R}e(\lambda) < 0 \\ \emptyset & \mathcal{R}e(\lambda) \geq 0 \end{cases}$$

can be shown, i.e. for $\lambda < 0,\ \mathcal{A}_\lambda =]0, -\frac{2}{\lambda}[$. ■

Example 4.14 Backward Euler for $u'(t) = \lambda u(t),\ t > 0,\ u(0) = 1$
As above, it is easily seen that $\tilde{U}_i = \left(\frac{1}{1-h\lambda}\right)^{i-1},\ i \geq 1$, i.e. $\lim_{i\to\infty} |\tilde{U}_i| = 0 \Leftrightarrow |1 - h\lambda| > 1 \Leftrightarrow \mathcal{A} = \{h\lambda \neq 1 + z :\ |z| \leq 1\} = \mathbb{C} \setminus \bar{B}_{(1,0)}(1)$ (the closed ball of radius 1 centered in $(1, 0)$). Clearly, backward Euler *is* $\mathcal{A}$-stable and

$$\mathcal{A}_\lambda = \begin{cases}]0, \infty[& \mathcal{R}e(\lambda) < 0 \\ ]h_{\min}, \infty[& \mathcal{R}e(\lambda) \geq 0,\ |\lambda| > 0 \\ \emptyset & \lambda = 0 \end{cases}$$

for some positive $h_{\min}$ depending on λ. Note that for $\lambda > 0$, $\mathcal{A}_\lambda =]\frac{2}{\lambda}, \infty[$ so that in this case the numerical Backward Euler solution will actually for sufficiently *large* step lengths converge to zero even though the exact solution goes to ∞. ■

Example 4.15 Crank-Nicolson for $u'(t) = \lambda u(t),\ t > 0,\ u(0) = 1$

As above, it is easily seen that $\tilde{U}_i = \left(\frac{1+h\lambda/2}{1-h\lambda/2}\right)^{i-1}$, $i \geq 1$, i.e. $\lim_{i\to\infty} |\tilde{U}_i| = 0 \Leftrightarrow |1 + h\lambda/2| < |1 - h\lambda/2| \Leftrightarrow \mathcal{A} = \mathbb{C}_-$ i.e. Crank Nicolson *is* $\mathcal{A}$-stable and

$$\mathcal{A}_\lambda = \begin{cases}]0, \infty[& \mathcal{R}e(\lambda) < 0 \\ \emptyset & \mathcal{R}e(\lambda) \geq 0 \end{cases}.$$

Hence the numerical Crank Nicolson solution will actually converge to zero when the exact solution does so, will converge to ∞ when the exact solution does so and will converge to a number of unit length when the exact solution does so. ■

Example 4.16 Heun's method for $u'(t) = \lambda u(t),\ t > 0,\ u(0) = 1$
As above, it is easily seen that $\tilde{U}_i = (1 + h\lambda + h^2\lambda^2/2)^{i-1}$, $i \geq 1$, i.e. $\lim_{i\to\infty} |\tilde{U}_i| = 0 \Leftrightarrow |1 + h\lambda + h^2\lambda^2/2| < 1 \Leftrightarrow \mathcal{A} = D \subset \mathbb{C}_-$ where $D \supset B_{(-1,0)}(1)$ has the same restriction to the real axes as $B_{(-1,0)}(1)$. Clearly, Heun's method is *not* $\mathcal{A}$-stable. Instead

$$\mathcal{A}_\lambda = \begin{cases}]0, h_{\max}[& \mathcal{R}e(\lambda) < 0 \\ \emptyset & \mathcal{R}e(\lambda) \geq 0 \end{cases}$$

for some $h_{\max} > -\frac{2\mathcal{R}e(\lambda)}{|\lambda|^2}[$ can be shown. For a plot of D see for example [4] Figure 11.3. ■

Note, that we have only developed the notion of absolute stability for one very particular instance of the DEP (1.10). While it turns out that parallels can be drawn to many other cases, this should be done with the utmost care, based on understanding and knowledge and not guesses. Also note that absolute stability has nothing to do with convergence and zero stability.

Exercise 4.17
Verify numerically the $\mathcal{A}_\lambda$ given in examples 4.11–4.16 for $\lambda = -5$. ■

4.3 Convergence and stability for linear, constant coefficient, multi step FDM's for $u' = f(x, u)$

As the order of convergence q in the explicit, one step methods for (3.47) is increased, generally the complexity of the function ϕ also increases making

it more costly to evaluate. Still there are advantages to this approach which we shall consider below in section 4.5 on page 143.

In this section we shall consider a different approach to increasing the order of convergence, where the complexity of ϕ is kept low, requiring only one evaluation of f for each new value of $\tilde{U}$. Instead the number of previous steps $\tilde{U}_{i-1}, \tilde{U}_{i-2}, \ldots, \tilde{U}_{i-s}$ taken into consideration when evaluating the next value $\tilde{U}_i$ is increased. The disadvantage is that it turns out that uniform step lengths are quite essential to reach the higher order of convergence. Hence these methods are unsuited for an adaptive approach, where the step lengths are adapted to the situation and may differ from step to step. For situations where adaptivity is not needed we shall then consider Uniform Step Length, Linear, Constant Coefficient, s Step FDM's for (3.47), repeated here again for reference:

$$\begin{aligned}&\text{Find } u \in \mathcal{C}^1(I): \ u'(x) = f(x,u(x)) \ \forall x \in I = (x_1, x_1 + (n-1)h),\\ &u(x_1) = u^*,\end{aligned}$$

that can be written in the optimal form

$$\begin{aligned}(4.8)\quad &\tilde{U}_1 = \tilde{U}_1^*, \ldots, \tilde{U}_s = \tilde{U}_s^*, \ \sum_{j=0}^{s} \left(\frac{a_j}{h} \tilde{U}_{i-j} + b_j \tilde{f}_{i-j} \right) = 0, \ i = s+1, \ldots, n,\\ &a_0 = -1, \ (a_s, b_s) \neq (0,0),\end{aligned}$$

where the nodal points are distributed uniformly according to $x_i = x_1 + (i-1)h$, $i = 1, \ldots, n$, $I = (x_1, x_n)$ and $s \geq 1$. Note that as above $\tilde{f}_i = f(x_i, \tilde{U}_i)$. Note also that the notation Constant Coefficients is used because the constants $\{a_j\}_{j=0}^s$ and $\{b_j\}_{j=0}^s$ do not depend on the nodal points $\{x_i\}_{i=1}^n$. Clearly (4.8) is explicit if $b_0 = 0$ and otherwise implicit. Note finally that we impose $a_0 \neq 0$ to insure existence of solution at least for the explicit case where we then may write (4.8) in the non optimal form

$$\begin{aligned}(4.9)\qquad &\tilde{U}_1 = \tilde{U}_1^*, \ldots, \tilde{U}_s = \tilde{U}_s^*,\\ &\tilde{U}_i = \sum_{j=1}^{s} \left(\frac{a_j}{-a_0} \tilde{U}_{i-j} + h \frac{b_j}{-a_0} \tilde{f}_{i-j} \right), \ i = s+1, \ldots, n.\end{aligned}$$

Taking $a_0 = -1$ is only to simplify the fractions in (4.9). For the implicit case, existence is of course more tricky, depending on the nonlinearity of f, but in any case $a_0 \neq 0$ helps but could be replaced by the more general $(a_0, b_0) \neq (0,0)$. The conditions $(a_s, b_s) \neq (0,0)$ are simply imposed in order to insure that the method is s step and not p step for some $p < s$.

For example Forward and Backward Euler and Crank-Nicolson (but not Heun's) methods as introduced in examples 3.35–3.38 on page 97 are all examples of linear, constant coefficient, multi step FDM's for (3.47) with $s = 1$, i.e. single step methods. Recall, that we still need to show convergence of Backward Euler and Crank-Nicolson. This will be done in the current section ■ .

To be able to use (4.8) we need s initial values $\tilde{U}_1^*, \ldots \tilde{U}_s^*$ whereas (3.47) provides only one value u^*. The rest of the values, for $s > 1$, must be found by other methods: Considering for example explicit methods, with $\tilde{U}_1^*$ given (by u^*), $\tilde{U}_2^*$ must be found by a method which is at most 1 step. Then $\tilde{U}_3^*$ can be found by a method which is at most 2 step since now both $\tilde{U}_1^*$ and $\tilde{U}_2^*$ are given. Continuing like this eventually $\tilde{U}_s^*$ can be found by a method which is at most $(s-1)$ step. Apart from explicit methods, also implicit methods may be used and often explicit or implicit Runge-Kutta FDM's (see section 4.5 on page 143 below) are used for this process of Priming the Pumps. With the notation of section 3.3 (see (3.42)) the priming process can be expressed as

$$\tilde{U}_1 = \tilde{U}^*, \ \Phi_i(\tilde{U}_2, \ldots, \tilde{U}_s) = 0, \ \text{for } i = 2, \ldots, s, \tag{4.10}$$

for some Φ_i's approximating $u'(y_i) - f(y_i, u(y_i)$ in various points y_i, $i = 2, \ldots, s$.

To find the local truncation error defined in definition 3.40 on page 102 by (3.60) we shall assume that the representation given by (4.8) and (4.10) is optimal. We then introduce the "priming" operators

$$\Psi_i[v] = \Phi_i(v(x_2), \ldots, v(x_s)), \ \forall v \in \mathcal{C}^1(I), \ \text{for } i = 2, \ldots, s \tag{4.11}$$

and the linear differential operator

$$\mathcal{L}[v(x)] = \sum_{j=0}^{s} \left(\frac{a_j}{h} v(x - jh) + b_j v'(x - jh) \right), \ \forall v \in \mathcal{C}^1(I) \tag{4.12}$$

and get

$$\begin{aligned} \boldsymbol{\tau} &= (\tau_1, \ldots, \tau_n)^T \\ &= (u^* - \tilde{U}^*, \Psi_2[u], \ldots, \Psi_s[u], \mathcal{L}[u(x_{s+1})], \ldots, \mathcal{L}[u(x_n)])^T. \end{aligned} \tag{4.13}$$

It is clear from (4.13) that the priming method must be of as high an order of consistency as the multi step method used for the rest of the recovery process, to avoid getting truncation errors dominated by the priming process.

Example 4.18 Adams methods for (3.47)

So far, we have derived FDM's by replacing derivatives with finite differences. For the Adams methods the approach is slightly different: We eliminate the derivatives by integrating and then we simplify the integrals by interpolating the integrants:

$$(4.14)\quad u'(x) = f(x,u) \Rightarrow u(x_i) - u(x_{i-1}) = \int_{x_{i-1}}^{x_i} f(x,u(x))dx,\ i = 1,2,\ldots.$$

For interpolating f we shall use the Lagrange form of the interpolating polynomial so please recall section 2.2.2 on page 40 and in particular equations (2.24) and (2.25). Compared to that section we need to use a more complex notation here, where the dependence of the cardinal functions on the nodal points appear explicitly. What was called ℓ_k in section 2.2.2 will here be denoted $\ell_k^{\{x_0,\ldots,x_n\}}$: Let s be a positive integer, take $i \geq s+1$ and denote the interpolating polynomial to f in the s nodes $x_{i-1}, x_{i-2}, \ldots, x_{i-s}$ by $\Pi_{i,s}^{ex}(f) = \sum_{k=i-s}^{i-1} f_k \ell_k^{\{x_{i-1},\ldots,x_{i-s}\}} = \sum_{j=1}^{s} f_{i-j} \ell_{i-j}^{\{x_{i-1},\ldots,x_{i-s}\}} \in \mathcal{P}_{s-1}$ with the cardinal functions given by

$$(4.15)\quad \ell_{i-j}^{\{x_{i-1},\ldots,x_{i-s}\}}(x) = \prod_{k=1,k\neq j}^{s} \frac{x - x_{i-k}}{x_{i-j} - x_{i-k}},\ j = 1,\ldots,s,\ \text{for } s > 1,$$

$$\ell_{i-1}^{\{x_{i-1}\}} \equiv 1,\ \text{for } s = 1,$$

satisfying the cardinal property $\ell_{i-j}^{\{x_{i-1},\ldots,x_{i-s}\}}(x_{i-k}) = \delta_{j,k}$ for $j,k = 1,\ldots,s$. Let correspondingly $\Pi_{i,s}^{im}(f) = \sum_{j=0}^{s-1} f_{i-j} \ell_{i-j}^{\{x_i,\ldots,x_{i-(s-1)}\}} \in \mathcal{P}_{s-1}$ be the interpolating polynomial to f in the s nodes $x_i, x_{i-1}, \ldots, x_{i-(s-1)}$ with the cardinal functions given by

$$(4.16)\quad \ell_{i-j}^{\{x_i,\ldots,x_{i-(s-1)}\}}(x) = \prod_{k=0,k\neq j}^{s-1} \frac{x - x_{i-k}}{x_{i-j} - x_{i-k}},\ j = 0,\ldots,s-1,\ \text{for } s > 1,$$

$$\ell_i^{\{x_i\}} \equiv 1,\ \text{for } s = 1,$$

satisfying the cardinal property $\ell_{i-j}^{\{x_i,\ldots,x_{i-(s-1)}\}}(x_{i-k}) = \delta_{j,k}$ for $j,k = 0,\ldots,s-1$.

ex stands for explicit and im for implicit (see below). Let also

$$(4.17)\quad I_{i,s}^{ex}(f) = \int_{x_{i-1}}^{x_i} \Pi_{i,s}^{ex}(f)(x)dx = h\sum_{j=1}^{s} b_{j,s}^{ex} f_{i-j},$$

$$I_{i,s}^{im}(f) = \int_{x_{i-1}}^{x_i} \Pi_{i,s}^{im}(f)(x)dx = h\sum_{j=0}^{s-1} b_{j,s}^{im} f_{i-j},$$

where we change coordinates according to $x = x_1 + (i - 1 - r)h$ to get

$$(4.18)\quad b^{ex}_{1,1} = 1,\ b^{ex}_{j,s} = \frac{1}{h}\int_{x_{i-1}}^{x_i} \prod_{k=1,k\neq j}^{s} \frac{x - x_{i-k}}{x_{i-j} - x_{i-k}}dx = \int_0^1 \prod_{k=1,k\neq j}^{s} \frac{k-r}{k-j}dr,$$
$$j = 1, \ldots, s,\ s > 1$$

and with the same change of coordinates

$$(4.19)\, b^{im}_{0,1} = 1,\ b^{im}_{j,s} = \frac{1}{h}\int_{x_{i-1}}^{x_i} \prod_{k=0,k\neq j}^{s-1} \frac{x - x_{i-k}}{x_{i-j} - x_{i-k}}dx = \int_0^1 \prod_{k=0,k\neq j}^{s-1} \frac{k-r}{k-j}dr,$$
$$j = 0, \ldots, s-1,\ s > 1.$$

Note here that $b^{ex}_{j,s}$ and $b^{im}_{j,s}$ are independent of the index i.

Definition 4.19 *The s step Adams-Bashford method, AB(s) is defined for* $s \geq 1$ *by*

$$(4.20)\qquad \tilde{U}_i - \tilde{U}_{i-1} = I^{ex}_{i,s}(\tilde{f}) = h\sum_{j=1}^{s} b^{ex}_{j,s}\tilde{f}_{i-j},\ i = s+1, \ldots, n.$$

The $\max\{s-1, 1\}$ *step Adams-Moulton method, AM(s) is defined for* $s \geq 1$ *by*

$$(4.21)\quad \tilde{U}_i - \tilde{U}_{i-1} = I^{im}_{i,s}(\tilde{f}) = h\sum_{j=0}^{s-1} b^{im}_{j,s}\tilde{f}_{i-j},\ i = \max\{s, 2\}, \ldots, n.$$

Here, as always, $\tilde{f}_i = f(x_i, \tilde{U}_i)$ *and some other method is priming the pumps for* $s > 1$.
(The s in AB(s) and AM(s) is the number of nodes used in the interpolation of f. Alternatively, AB(s) is the s step explicit method and AM(s) is the implicit method corresponding to AB(s)).

Clearly AB(s) fits in the form (4.8) with $a_0 = -1$, $a_1 = 1$, $a_2 = \ldots = a_s = 0$, $b_0 = 0$ and $b_1 = b^{ex}_{1,s}, \ldots, b_s = b^{ex}_{s,s}$ and hence is an explicit, linear, constant coefficient, s step FDM for (3.47). Correspondingly AM(s) is an implicit, linear, constant coefficient, $\max\{s-1, 1\}$ step FDM for (3.47) with $a_0 = -1$, $a_1 = 1$, $a_2 = \ldots = a_{s-1} = 0$, $b_0 = b^{im}_{0,s}, \ldots, b_{s-1} = b^{im}_{s-1,s}$, $b_s = 0$. The

"missing" coefficients for the first 5 AB(s) and AM(s) methods are

(4.22)

Adams-Bashford AB(s)

s	b_1	b_2	b_3	b_4	b_5
1	1	0	0	0	0
2	$\frac{3}{2}$	$-\frac{1}{2}$	0	0	0
3	$\frac{23}{12}$	$-\frac{16}{12}$	$\frac{5}{12}$	0	0
4	$\frac{55}{24}$	$-\frac{59}{24}$	$\frac{37}{24}$	$-\frac{9}{24}$	0
5	$\frac{1901}{720}$	$-\frac{2774}{720}$	$\frac{2616}{720}$	$-\frac{1274}{720}$	$\frac{251}{720}$

Adams-Moulton AM(s)

s	b_0	b_1	b_2	b_3	b_4
1	1	0	0	0	0
2	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0
3	$\frac{5}{12}$	$\frac{8}{12}$	$-\frac{1}{12}$	0	0
4	$\frac{9}{24}$	$\frac{19}{24}$	$-\frac{5}{24}$	$\frac{1}{24}$	0
5	$\frac{251}{720}$	$\frac{646}{720}$	$-\frac{264}{720}$	$\frac{106}{720}$	$-\frac{19}{720}$

For example AB(2) has the expression

(4.23)
$$\frac{a_0}{h}\tilde{U}_i + \frac{a_1}{h}\tilde{U}_{i-1} + b_1\tilde{f}_{i-1} + b_2\tilde{f}_{i-2} = 0,\ i = 3, \ldots, n$$
$$\Updownarrow$$
$$\frac{-1}{h}\tilde{U}_i + \frac{1}{h}\tilde{U}_{i-1} + \frac{3}{2}\tilde{f}_{i-1} + \frac{-1}{2}\tilde{f}_{i-2} = 0,\ i = 3, \ldots, n$$
$$\Updownarrow$$
$$\frac{\tilde{U}_i - \tilde{U}_{i-1}}{h} = \frac{3}{2}\tilde{f}_{i-1} - \frac{1}{2}\tilde{f}_{i-2},\ i = 3, \ldots, n.$$

The AB(2) method does not have a "name", probably since it does not compare favorably to Heuns method. They ar both 2nd order but Heun is one step whereas BDF(2) is 2 step. Still the complexity of the function evaluation for Heun is not discouraging. Neither Heun nor AB(2) is $\mathcal{A}$-stable.

Using interpolation and numerical integration theory it can be shown that AM(s) and AB(s) are generally consistent of order s.

Theorem 4.20 *AB(s) and AM(s) are consistent of order at least s.*

Proof:
Let * indicate either *ex* or *im*. Then the i'th local truncation error $\tau_i^*(s)$ for

either AB(s) or AM(s) is given by

$$
\begin{aligned}
(4.24)\ \tau_i^*(s) &= \frac{u_i - u_{i-1}}{h} - \frac{1}{h}\int_{x_{i-1}}^{x_i} I_{i,s}^*(f)(t)dt \\
&= \left(u'_{i-1} + \frac{h}{2}u''_{i-1} + \frac{h^2}{6}u'''_{i-1} + \dots\right) - \frac{1}{h}\int_{x_{i-1}}^{x_i} (f + \mathcal{O}(h^s))\, dt \\
&= \left(u'_{i-1} + \frac{h}{2}u''_{i-1} + \frac{h^2}{6}u'''_{i-1} + \dots\right) - \frac{1}{h}(F_i - F_{i-1}) + \mathcal{O}(h^s) \\
&= \left(u'_{i-1} + \frac{h}{2}u''_{i-1} + \frac{h^2}{6}u'''_{i-1} + \dots\right) \\
&\quad - \left(F'_{i-1} + \frac{h}{2}F''_{i-1} + \frac{h^2}{6}F'''_{i-1} + \dots\right) + \mathcal{O}(h^s) \\
&= \mathcal{O}(h^s),\ i = s+1, \dots, n,
\end{aligned}
$$

where F_i is the primitive of f in x_i, i.e. $F'_i = f_i = u'_i$. ■

In the proof of theorem 4.20 we have not taken into consideration that better interpolation results may be obtained in special cases (compare to Newton-Cotes integration). To get the precise order of consistency for a specific method the following theorem 4.27 on page 123 may be applied. ■

Exercise 4.21
Prove that AB(1) is Forward Euler, AM(1) is Backward Euler and AM(2) is Crank-Nicolson, i.e. verify the relevant parts of (4.22) and write up the expression for the corresponding multi step methods. ■

Exercise 4.22
Derive the formulas for AB(2), AB(3), AB(4), AB(5), AM(3), AM(4) and AM(5), i.e. verify the relevant parts of (4.22). Then write up the expression for the corresponding multi step methods. ■

Example 4.23 Backward Differentiation Formula (BDF) methods for (3.47)
For the Adams methods we integrated the differential equation and approximated the right hand side integral by the integral of an interpolant. For the BDF methods we leave the differential equation and interpolate directly the left hand side by the derivative of the interpolating polynomial $\Pi_{i,s+1}^{im}(u) = \sum_{j=0}^{s} u_{i-j}\ell_{i-j}^{\{x_i,\dots,x_{i-s}\}} \in \mathcal{P}_s$ in the $s+1$ nodes $x_i, x_{i-1}, \dots, x_{i-s}$, with the cardinal functions given by

$$
(4.25)\qquad \ell_{i-j}^{\{x_i,\dots,x_{i-s}\}}(x) = \prod_{k=0,k\neq j}^{s} \frac{x - x_{i-k}}{x_{i-j} - x_{i-k}},\ j = 0, \dots, s,\ \text{for } s \geq 1,
$$

satisfying the cardinal property $\ell_{i-j}^{\{x_i,\ldots,x_{i-s}\}}(x_{i-k}) = \delta_{j,k}$ for $j,k = 0,\ldots,s$. Let also

$$a_{j,s} = h\frac{d}{dx}\left(\ell_{i-j}^{\{x_i,\ldots,x_{i-s}\}}\right)(x_i) = -\frac{d}{dr}\left(\prod_{k=0,k\neq j}^{s}\frac{k-r}{k-j}\right)\Bigg|_{r=0} \tag{4.26}$$

where we have changed coordinates acording to $x = x_1 + (i-r-1)h \Rightarrow r = i-1-\frac{1}{h}(x-x_1)$ to verify that $a_{j,s}$ is independent of i.

Definition 4.24 *The s step Backward Differentiation Formula method, BDF(s) is defined for $s \geq 1$ by*

$$\begin{aligned}(\Pi_{i,s+1}^{im})(\tilde{U})'(x_i) &= \sum_{j=0}^{s}\tilde{U}_{i-j}\frac{d}{dx}\left(\ell_{i-j}^{\{x_i,\ldots,x_{i-s}\}}\right)(x_i) \\ &= \sum_{j=0}^{s}\frac{a_{j,s}}{h}\tilde{U}_{i-j} = \tilde{f}_i,\ i = s+1,\ldots,n.\end{aligned} \tag{4.27}$$

Here, as always, $\tilde{f}_i = f(x_i,\tilde{U}_i)$ and some other method is priming the pumps for $s > 1$, providing $\tilde{U}_1,\ldots,\tilde{U}_s$.

BDF(s) is clearly an implicit, linear, constant coefficient, s step FDM for (1.10) with $a_0 = -1$, $a_1 = -\frac{a_{1,s}}{a_{0,s}},\ldots,a_s = -\frac{a_{s,s}}{a_{0,s}}$, $b_0 = \frac{1}{a_{0,s}}$ and $b_1 = \ldots = b_s = 0$. The coefficients for the first 6 BDF(s) methods are

(4.28)

s	a_1	a_2	a_3	a_4	a_5	a_6	b_0
1	1	0	0	0	0	0	1
2	$\frac{4}{3}$	$-\frac{1}{3}$	0	0	0	0	$\frac{2}{3}$
3	$\frac{18}{11}$	$-\frac{9}{11}$	$\frac{2}{11}$	0	0	0	$\frac{6}{11}$
4	$\frac{48}{25}$	$-\frac{36}{25}$	$\frac{16}{25}$	$-\frac{3}{25}$	0	0	$\frac{12}{25}$
5	$\frac{300}{137}$	$-\frac{300}{137}$	$\frac{200}{137}$	$-\frac{75}{137}$	$\frac{12}{137}$	0	$\frac{60}{137}$
6	$\frac{360}{147}$	$-\frac{450}{147}$	$\frac{400}{147}$	$-\frac{225}{147}$	$\frac{72}{147}$	$-\frac{10}{147}$	$\frac{20}{49}$

For example BDF(2) has the expression

$$\begin{aligned}&\frac{a_0}{h}\tilde{U}_i + \frac{a_1}{h}\tilde{U}_{i-1} + \frac{a_2}{h}\tilde{U}_{i-2} + b_0\tilde{f}_i = 0,\ i = 3,\ldots,n \\ &\Updownarrow \\ &\frac{-1}{h}\tilde{U}_i + \frac{4}{3h}\tilde{U}_{i-1} + \frac{-1}{3h}\tilde{U}_{i-2} + \frac{2}{3}\tilde{f}_i = 0,\ i = 3,\ldots,n \\ &\Updownarrow \\ &\frac{1}{2h}\left(3\tilde{U}_i - 4\tilde{U}_{i-1} + \tilde{U}_{i-2}\right) = \tilde{f}_i,\ i = 3,\ldots,n.\end{aligned} \tag{4.29}$$

The BDF(2) method is implicit, $\mathcal{A}$-stable and second order convergent like the Crank Nicolson method. Since BDF(2) is "harder to use" than CN since it is a 2 step method, it is not much in use and has never been given a proper "name".

The consistency result for BDF is similar to theorem 4.20 for Adams methods:

Theorem 4.25 *BDF(s) is consistent of order at least s.*

Proof:
The proof is similar in structure to that of theorem 4.20 for the Adams methods relying on interpolation results. The details will not be included here. ■

Comparing AM and BDF methods the typical situation is that for methods of the same order, the error is smaller (smaller constant) for the AM methods while the stability area is bigger for the BDF methods. ■

Exercise 4.26
Derive the formulas for BDF(1), BDF(2) and BDF(6), i.e. verify the relevant parts of (4.28) and write up the expression for the corresponding multi step methods. ■

After introducing the methods, we are now ready to consider the question of convergence. As for the explicit one step methods in sections 3.4 and 4.1 we start considering consistency:

Theorem 4.27 (4.8) *is consistent of order q with* (3.47) *iff the initial conditions are consistent of order q* ($\tau_1, \ldots, \tau_s = \mathcal{O}(h^q)$) *and*

$$\sum_{j=0}^{s} \left((-j)^k a_j + (-j)^{k-1} k b_j \right) \begin{cases} = 0, & k = 0, 1, \ldots, q. \\ \neq 0, & k = q+1. \end{cases} \tag{4.30}$$

(Here $(-0)^{-1} * 0 = 0$ *and* $0^0 = 1$*).*

Proof:

For $i = s+1, \ldots, n$

$$
\begin{aligned}
\tau_i &= \mathcal{L}[u(x_i)] = \sum_{j=0}^{s} \left(\frac{a_j}{h} u_{i-j} + b_j u'_{i-j} \right) \qquad (4.31)\\
&= \sum_{j=0}^{s} \left(\frac{a_j}{h} \sum_{k=0}^{\infty} \frac{(-jh)^k}{k!} u_i^{(k)} + b_j \underbrace{\sum_{k=0}^{\infty} \frac{(-jh)^k}{k!} u_i^{(k+1)}}_{=\sum_{k=1}^{\infty} \frac{(-jh)^{k-1}}{k!} k u_i^{(k)}} \right)\\
&= \sum_{k=0}^{\infty} \sum_{j=0}^{s} u_i^{(k)} \frac{h^{k-1}}{k!} \left((-j)^k a_j + (-j)^{k-1} k b_j \right).
\end{aligned}
$$

For the second line we have made a Taylor expansion around x_i of all terms. For the third line we have shiftet the summation by $k = r - 1$, and then changed r to k again for the "underbrace". Finally we have taken the consensus that the coefficient $(-j)^{k-1}k$ to b_j for $k = 0$ is 0. ■

Example 4.28 Consistency of order q then requires apart from consistency of the initial conditions of the same order also the satisfaction of the first $q+1$ of the conditions (4.30) of which we explicitly show the first 4 here:

$$
\begin{aligned}
k = 0: &\quad \sum_{j=0}^{s} a_j = 0 \Leftrightarrow \sum_{j=1}^{s} a_j = 1 \qquad (4.32)\\
k = 1: &\quad \sum_{j=0}^{s} (-j a_j + b_j) = 0\\
k = 2: &\quad \sum_{j=0}^{s} \left(j^2 a_j - 2 j b_j \right) = 0\\
k = 3: &\quad \sum_{j=0}^{s} \left(-j^3 a_j + 3 j^2 b_j \right) = 0
\end{aligned}
$$

■

For example it is easily checked that the consistency orders for Forward Euler ($a_0 = -1$, $a_1 = 1$, $b_0 = 0$, $b_1 = 1$), Backward Euler ($a_0 = -1$, $a_1 = 1$, $b_0 = 1$, $b_1 = 0$) and for Crank-Nicolson ($a_0 = -1$, $a_1 = 1$, $b_0 = b_1 = \frac{1}{2}$) found in examples 3.45–3.48 on page 106 are valid also using theorem 4.27 on page 123 ■ .

Exercise 4.29
Show that the orders of consistency for AB(s) and AM(s) for $s = 1, \ldots, 5$ are in all cases s, under the assumption that the priming the pumps process has this order of consistency. ■

Exercise 4.30
Show that the orders of consistency for BDF(s) for $s = 1, \ldots, 6$ are in all cases s, under the assumption that the priming the pumps process has this order of consistency. ■

Exercise 4.31
Write a small essay on backward differentiation formula methods. ■

The ϵ-perturbation of (4.8) (see definition (3.43) on page 104) is

$$
\begin{aligned}
&\tilde{Z}_{\epsilon,1} = \tilde{U}^* + \delta_{\epsilon,1}, \\
&\Phi_i(\tilde{Z}_{\epsilon,2}, \ldots, \tilde{Z}_{\epsilon,s}) = \delta_{\epsilon,i}, \ i = 2, \ldots, s, \\
&\sum_{j=0}^{s} \left(\frac{a_j}{h} \tilde{Z}_{\epsilon,i-j} + b_j \tilde{f}_{\epsilon,i-j} \right) = \delta_{\epsilon,i}, \ i = s+1, \ldots, n.
\end{aligned} \tag{4.33}
$$

Here $|\delta_{\epsilon,i}| < \epsilon$ for $i = 1, \ldots, n$ and $\tilde{f}_{\epsilon,i-j} = f(x_{i-j}, \tilde{Z}_{\epsilon,i-j})$.

The discrete result corresponding to theorem 4.1 on page 108 is

Theorem 4.32 *If the "priming the pumps process" is zero stable, i.e.* $\exists h_0 > 0, \ \exists C > 0 : \ |\tilde{U}_i - \tilde{Z}_{\epsilon,i}| < C\epsilon \ \forall i = 1, \ldots, s, \ \forall h \in]0, h_0]$, *if* f *is globally Lipschitz continuous in its second variable (see* (1.11)*) and if the optimal representation* (4.8) *for the FDM satisfies the Root Condition then the FDM represented by* (4.8) *is zero stable.*

To define the root condition and prove theorem 4.32 we need some preparations, and we shall postpone this until after the following convergence result corresponding to theorem 4.3 on page 109.

Theorem 4.33 *Consider the FDM represented by* (4.8) *which is assumed to have global order of consistency* $q \geq 0$, *a zero stable priming the pumps process and an* f *which is Lipschitz continuous in its second variable.*
The representation (4.8) *satisfies the root condition* $\Rightarrow$ *The FDM represented by* (4.8) *is Zero Stable* $\Rightarrow$ *The FDM represented by* (4.8) *is convergent of order* q *in all points of* I.

Proof:
The first implication is simply theorem 4.32 and the second implication simply theorem 3.44 on page 105. ■

Now let us return to the root condition that for the linear multi step methods is taking the place that Lipschitz continuity had for the explicit one step methods for (3.47).

Definition 4.34 *The First and Second Characteristic Polynomials for the linear multistep method* (4.8) *are*

$$\rho(r) = -\sum_{j=0}^{s} a_j r^{s-j} = -a_0 r^s - a_1 r^{s-1} - a_2 r^{s-2} - \ldots - a_{s-1} r - a_s \tag{4.34}$$

and

$$\sigma(r) = \sum_{j=0}^{s} b_j r^{s-j} = b_0 r^s + b_1 r^{s-1} + b_2 r^{s-2} + \ldots + b_{s-1} r + b_s \tag{4.35}$$

respectively.
The First Characteristic Roots $r_k,\ k = 1, \ldots, s' \leq s$ *are the roots of the first characteristic polynomial* ρ *and their multiplicities are denoted* $m_1, \ldots, m_{s'}$. $(m_1 + \ldots + m_{s'} = s$ *for* $a_s \neq 0)$.
(4.8) *is said to satisfy the Root Condition if*

$$|r_k| \leq 1,\ k = 1, \ldots, s',\ \textit{and if for any } k, |r_k| = 1,\ \textit{then } \rho'(r_k) \neq 0, \tag{4.36}$$

i.e. all roots lie in the complex closed unit disk, and the eventual roots on the unit circle (the boundary) are simple roots (of multiplicity one).
(4.8) *is said to satisfy the Strong Root Condition if (after possible renumbering of the roots)*

$$|r_k| < 1,\ k = 2, \ldots, s',\ \textit{and if } |r_1| \geq 1,\ \textit{then } r_1 = 1 \textit{ and } m_1 = 1, \tag{4.37}$$

i.e. all roots lie in the complex open unit disk, except eventually the simple real root 1.

By theorem 4.27 on page 123 (see also (4.32)) it is evident that 1 is a first characteristic root if (4.8) is consistent. Hence the special concern for this case in the strong root condition.

Note in (4.34) and (4.35) that the indices on the coefficients and the exponents of r in the characteristic polynomials "go in the opposite directions".

This is somewhat inconvenient notationally, so when working with the characteristic polynomials, it is common to "turn around" the indices and also change the sign in the first characteristic polynomial defining

$$\alpha_j = -a_{s-j},\ j = 0,\dots,s,\ \beta_j = b_{s-j},\ j = 0,\dots,s, \tag{4.38}$$

so that the characteristic polynomials turn into

$$\rho(r) = -\sum_{j=0}^{s} a_j r^{s-j} = \sum_{j=0}^{s} \alpha_j r^j, \qquad \sigma(r) = \sum_{j=0}^{s} b_j r^{s-j} = \sum_{j=0}^{s} \beta_j r^j. \tag{4.39}$$

Theorem 4.35 *All Adams methods and all BDF(s) methods for* $1 \le s \le 6$ *satisfy the root condition and the strong root condition.*

Proof:
Adams methods all have the first characteristic polynomial $\rho(r) = r^s - r^{s-1} = (r-1)r^{s-1}$, with simple root 1 and $(s-1)$-double root 0.
The BDF methods are checked one at a time using Maple and the coefficients from (4.28). They have the following first characteristic polynomials and roots:
BDF(1): $r - 1 = 0 \Rightarrow r = 1$.
BDF(2): $r^2 - \frac{4}{3}r + \frac{1}{3} = 0 \Rightarrow r = 1$ or $r = \frac{1}{3}$
BDF(3): $r^3 - \frac{18}{11}r^2 + \frac{9}{11}r - \frac{2}{11} = 0 \Rightarrow r = 1$ or $r = 0.3181... \pm i * 0.2838...$
BDF(4): $r^4 - \frac{48}{25}r^3 + \frac{36}{25}r^2 - \frac{16}{25}r + \frac{3}{25} = 0 \Rightarrow r = 1$ or $r = 0.3814...$ or $r = 0.2692 \pm i * 0.4920...$
BDF(5): $r^5 - \frac{300}{137}r^4 + \frac{300}{137}r^3 - \frac{200}{137}r^2 + \frac{75}{137}r - \frac{12}{137} = 0 \Rightarrow r = 1$ or $r = 0.2100 \pm i * 0.6768...$ or $r = 0.3848... \pm i * 0.1621...$
BDF(6): $r^6 - \frac{360}{147}r^5 + \frac{450}{147}r^4 - \frac{400}{147}r^3 + \frac{225}{147}r^2 - \frac{72}{147}r + \frac{10}{147} = 0 \Rightarrow r = 1$ or $r = 0.4061...$ or $r = 0.1452... \pm i * 0.8510...$ or $r = 0.3761... \pm i * 0.2884...$ ■

The BDF(s) methods for $s > 6$ are not zero stable and hence are not convergent and hence are not being used for computations.

The proof of theorem 4.32 on page 125 will be reserved for advanced readers. As part of the proof there is however the theory of difference equations which is for everybody.

★ For advanced readers 4.36 ★ We now develop the theory necessary to complete the proof of theorem 4.32 on page 125. Subtracting the linear multi step iterations in (4.33) from those of (4.8) and changing to "α and β

notation", we get

$$\sum_{j=0}^{s}\Big(\frac{a_j}{h}(\tilde{U}_{i-j}-\tilde{Z}_{\epsilon,i-j})+b_j(\tilde{f}_{i-j}-\tilde{f}_{\epsilon,i-j})\Big)+\delta_{\epsilon,i}=0,$$
$$i=s+1,\ldots,n \quad (a_0=-1)$$

$$\Updownarrow$$

$$\sum_{j=0}^{s}\alpha_{s-j}(\tilde{U}_{i-j}-\tilde{Z}_{\epsilon,i-j})=h\delta_{\epsilon,i}+h\sum_{j=0}^{s}\beta_{s-j}(\tilde{f}_{i-j}-\tilde{f}_{\epsilon,i-j}),$$
$$i=s+1,\ldots,n \quad (\alpha_s=1)$$

$$\Updownarrow \quad (s-j\to j,\ i-s-1\to i)$$

$$\sum_{j=0}^{s}\alpha_{j}(\tilde{U}_{i+1+j}-\tilde{Z}_{\epsilon,i+1+j})=h\delta_{\epsilon,i+1+s}+h\sum_{j=0}^{s}\beta_{j}(\tilde{f}_{i+1+j}-\tilde{f}_{\epsilon,i+1+j}),$$
$$i=0,\ldots,n-1-s \quad (\alpha_s=1)$$

$$\Updownarrow$$

(4.40) $$\sum_{j=0}^{s}\alpha_j W_{i+j}=\phi_{i+s},\ i=0,\ldots,n-s-1 \quad (\alpha_s=1),\ \text{where}$$
$$W_{i+j}=\tilde{U}_{i+1+j}-\tilde{Z}_{\epsilon,i+1+j},\ j=0,\ldots,s,\ i=0,\ldots,n-s-1 \text{ and}$$
$$\phi_{i+s}=h\delta_{\epsilon,i+1+s}+h\sum_{j=0}^{s}\beta_j(\tilde{f}_{i+1+j}-\tilde{f}_{\epsilon,i+1+j}),\ i=0,\ldots,n-s-1.$$

To prove theorem 4.32 on page 125 we need to show that $|W_i|<C\epsilon,\ i=s,\ldots,n-1$. For this we shall first study the general theory for solutions to equations of the form (4.40). ★

4.3.1 Linear, constant coefficient, Homogeneous Difference Equations of order s

We consider equation systems of the form

(4.41) $$\sum_{j=0}^{s}\alpha_j W_{i+j}=0,\ i=0,1,\ldots \quad (\alpha_s=1,\ \alpha_0\neq 0)$$
$$W_0=\tilde{W}_0^*,\ldots,W_{s-1}=\tilde{W}_{s-1}^*,$$

where $\tilde{W}_0^*,\ldots,\tilde{W}_{s-1}^*$ are known constants. Such equation systems are denoted Linear, Constant Coefficient, Homogeneous Difference Equations of

order s. The linearity is with respect to the coefficients $\alpha_0,\ldots,\alpha_s$ and the constant coefficients refer to the fact that no α_j depends on i. (4.41) is solved iteratively, knowing the Initial Data $\tilde{W}_0^*,\ldots,\tilde{W}_{s-1}^*$.

For the proof of theorem 4.32 on page 125 but also for many other situations where difference equations occur we need to express the solution in closed form. To do this we start considering $\rho(r) = \sum_{j=0}^{s}\alpha_j r^j$ defined in (4.34) (see also (4.39)) and denoted the Characteristic Polynomial for (4.41) and the s sequences $\{\xi_{\ell(k,j)}^{(i)}\}_{i=0}^{\infty}$, $j = 0,\ldots,m_k-1$, $k = 1,\ldots,s'$ where $\xi_{\ell(k,j)}^{(i)} = i^j r_k^i$ denoted the s Characteristic Fundamental Solutions for (4.41). Here $\ell(k,j) = j + \sum_{\tilde{k}=1}^{k-1} m_{\tilde{k}} \in \{0,\ldots,s-1\}$ orders the fundamental solutions sequencially.

$$\begin{aligned} \mathcal{S} &= \text{span}\left\{\{\xi_\ell^{(i)}\}_{i=0}^{\infty}\right\}_{\ell=0}^{s-1} \\ &= \text{span}\left\{\{r_k^i\}_{i=0}^{\infty}, \{ir_k^i\}_{i=0}^{\infty},\ldots,\{i^{m_k-1}r_k^i\}_{i=0}^{\infty}\right\}_{k=1}^{s'} \end{aligned} \tag{4.42}$$

is denoted the Solution Space for (4.41). Any set of sequences spanning $\mathcal{S}$ is denoted a set of Fundamental Solutions for (4.41).

Theorem 4.37 *The unique solution to* (4.41) *belongs to the solution space* $\mathcal{S}$*, i.e. can for some constants* γ_ℓ*,* $\ell = 0,\ldots,s-1$ *(independent of* i*) be expressed by*

$$\{W_i\}_{i=0}^{\infty} = \{\sum_{k=1}^{s'}\sum_{j=0}^{m_k-1}\gamma_{\ell(k,j)}\xi_{\ell(k,j)}^{(i)}\}_{i=0}^{\infty} = \{\sum_{k=1}^{s'}\sum_{j=0}^{m_k-1}\gamma_{\ell(k,j)}i^j r_k^i\}_{i=0}^{\infty}, \tag{4.43}$$

and when the initial data is not *taken into consideration then any sequence in* $\mathcal{S}$ *solves* (4.41)*.*

Proof:
We first show that any function in $\mathcal{S}$ is a solution. By the linearity of the equation system, it is sufficient to show that all characteristic fundamental solutions are solutions. Consider first $W_i = r_k^i$, $i = 0,1,\ldots$. Inserting into (4.41) gives

$$\sum_{j=0}^{s}\alpha_j r_k^{i+j} = r_k^i\rho(r_k) = 0,\ i = 0,1,\ldots \tag{4.44}$$

Now consider the general case $W_i = i^m r_k^i$, $i = 0,1,\ldots$ where $0 \le m < m_k$

and where we here and below take $0^0 = 1$. Again inserting into (4.41) gives

$$
\begin{aligned}
(4.45)\qquad \sum_{j=0}^{s} \alpha_j (i+j)^m r_k^{i+j} &= \sum_{j=0}^{s} \alpha_j (i+j)^{m-1} \frac{dr_k^{i+j}}{dr_k} r_k \\
&= \sum_{j=0}^{s} \alpha_j (i+j)^{m-2} \frac{d}{dr_k} \left(\frac{dr_k^{i+j}}{dr_k} r_k \right) r_k \\
&= \ldots = \sum_{j=0}^{s} \alpha_j \underbrace{\frac{d}{dr_k} \left(\ldots \frac{d}{dr_k} \left(\frac{dr_k^{i+j}}{dr_k} r_k \right) r_k \ldots \right) r_k}_{m \text{ repetitions}} \\
&= \underbrace{\frac{d}{dr_k} \left(\ldots \frac{d}{dr_k} \left(\frac{d\left(r_k^i \rho(r_k)\right)}{dr_k} r_k \right) r_k \ldots \right) r_k}_{m \text{ repetitions}} = 0, \ i = 0, 1, \ldots,
\end{aligned}
$$

where the last zero comes from the fact that r_k is an m_k-multiple root of ρ and $m < m_k$.

The uniqueness of solution comes directly from (4.41) rewritten as $W_{i+s} = -\sum_{j=0}^{s-1} \alpha_j W_{i+j}$, $i = 0, 1, \ldots$ since $W_0, \ldots, W_{s-1}$ are given initial data.

To show that the solution to (4.41) lies in $\mathcal{S}$ we take a constructive approach. First we construct a basis $\{\{\psi_0^{(i)}\}_{i=0}^{\infty}, \ldots, \{\psi_{s-1}^{(i)}\}_{i=0}^{\infty}\}$ for $\mathcal{S}$ such that $\psi_t^{(i)} = \delta_{i,t}, \ i, t = 0, \ldots, s-1$ where $\delta_{i,t}$ is the Kronecker delta. We postpone the existence of such a basis until after noting that $W_i = \sum_{t=0}^{s-1} \tilde{W}_t^* \psi_t^{(i)}, \ i = 0, 1, \ldots$ satisfies the initial conditions $W_i = \tilde{W}_i^*$ for $i = 0, \ldots, s-1$ and, $\{W_i\}_{i=0}^{\infty}$ being a finite linear combination of basis functions for $\mathcal{S}$, satisfies also the iteration equation system, so that we have found a solution to (4.41) in $\mathcal{S}$ as claimed.

We are left with the problem of showing existence of the $\psi_t^{(i)}$'s, i.e. to show existence of an s by s matrix $\mathbf{G}$ with components $g_{k,j}, \ k, j = 0, \ldots, s-1$ independent of i such that $\{\psi_t^{(i)}\}_{i=0}^{\infty} = \sum_{\ell=0}^{s-1} g_{t,\ell} \{\xi_\ell^{(i)}\}_{i=0}^{\infty}$ and $\psi_t^{(i)} = \sum_{\ell=0}^{s-1} g_{t,\ell} \xi_\ell^{(i)} = \delta_{i,t}, \ i, t = 0, \ldots, s-1$. The latter condition is equivalent to the non singularity of the matrix

$$
\begin{aligned}
(4.46)\qquad \mathbf{F} &= \{\xi_\ell^{(i)}\}_{i,\ell=0}^{s-1} \\
&= \begin{bmatrix}
r_1^0 & r_1^1 & r_1^2 & \cdots & r_1^{s-1} \\
0 r_1^0 & 1 r_1^1 & 2 r_1^2 & \cdots & (s-1) r_1^{s-1} \\
0^2 r_1^0 & 1^2 r_1^1 & 2^2 r_1^2 & \cdots & (s-1)^2 r_1^{s-1} \\
\vdots & \vdots & \vdots & \cdots & \vdots \\
0^{m_{s'}-1} r_{s'}^0 & 1^{m_{s'}-1} r_{s'}^1 & 2^{m_{s'}-1} r_{s'}^2 & \cdots & (s-1)^{m_{s'}-1} r_{s'}^{s-1}
\end{bmatrix}.
\end{aligned}
$$

Non singularity of $\mathbf{F}$ is again equivalent to the fact that

$$\begin{aligned}(4.47)\qquad &\forall \mathbf{b} = (b_0, \ldots, b_{s-1})^T \exists \mathbf{c} = (c_0, \ldots, c_{s-1})^T : \mathbf{Fc} = \mathbf{b} \Leftrightarrow \\ &\sum_{i=0}^{s-1} \xi_\ell^{(i)} c_i = b_\ell,\ \ell = 0, \ldots, s-1 \Leftrightarrow \\ &\sum_{i=0}^{s-1} i^j r_k^i c_i = b_{\ell(k,j)},\ j = 0, \ldots, m_k - 1,\ k = 1, \ldots, s'.\end{aligned}$$

Defining $\rho^*(r) = \sum_{i=0}^{s-1} c_i r^i$, (4.47) is equivalent to

$$\begin{aligned}(4.48)\ \forall \mathbf{b} \exists \mathbf{c} : &\rho^*(r_k) = b_{\ell(k,0)}, (\rho^*)'(r_k) r_k = b_{\ell(k,1)}, (((\rho^*)'(r_k) r_k)' r_k = b_{\ell(k,2)}, \\ &\ldots, \underbrace{(\ldots(((\rho^*)'(r_k) r_k)' r_k)' \ldots r_k)' r_k}_{m_k - 1 \text{ repetitions}} = b_{\ell(k,m_k-1)},\ k = 1, \ldots, s',\end{aligned}$$

(compare to (4.45)), which is again, with a different $\mathbf{c}$, equivalent to

$$\begin{aligned}(4.49)\qquad &\forall \mathbf{b} \exists \mathbf{c} : \rho^*(r_k) = b_{\ell(k,0)}, (\rho^*)'(r_k) = b_{\ell(k,1)}, \ldots, \\ &(\rho^*)^{(m_k-1)}(r_k) = b_{\ell(k,m_k-1)},\ k = 1, \ldots, s'.\end{aligned}$$

Just rewrite the left hand side (lhs) in (4.48) in the form of the lhs of (4.49). This changes $b_{\ell(k,1)}$ to $\tilde{b}_{\ell(k,1)} = b_{\ell(k,1)}/r_k$, $b_{\ell(k,2)}$ to $\tilde{b}_{\ell(k,2)} = (b_{\ell(k,2)} - b_{\ell(k,1)})/r_k^2$ and so on. When $b_{\ell(k,j)}$ goes through all possibilities, so does $\tilde{b}_{\ell(k,j)}$ only noting that $r_k \neq 0$ since $\rho(0) = \sum_{j=0}^{s} \alpha_j 0^j = \alpha_0 \neq 0$.
That (4.49) is true is an immediate consequence of the Hermite interpolation theorem for polynomials of degree $s - 1$. (Theorem 2.33 on page 34). ■

(It can be remarked, that by picking $b_{\ell(k,j)} = -s^j r_k^s$ in (4.47) or (4.48) but not in (4.49), where $\mathbf{c}$ has changed, we see that $\rho^*(r) + r^s$ has the same roots with the same multiplicities as ρ and hence $\rho^*(r) = \rho(r) - r^s$ recalling that $\alpha_s = 1$).

From theorem 4.37 comes the following method for the solution of homogeneous difference equations of the form (4.41):

1. Read off the order s from the difference equation system.
2. Write down the degree s characteristic polynomial.
3. Find the characteristic roots and their multiplicity.
4. Construct the s characteristic fundamental solutions $\xi_0, \ldots, \xi_{s-1}$, each being infinite sequences: $\xi_k = \{\xi_k^{(i)}\}_{i=0}^{\infty}$, $k = 0, \ldots, s-1$.

5. Write down an arbitrary linear combination of the characteristic fundamental solutions $a_0\xi_0 \cdot \ldots \cdot a_{s-1}\xi_{s-1}$.

6. Consider the equation system consisting of the s equations $a_0\xi_0^{(i)} \cdot \ldots \cdot a_{s-1}\xi_{s-1}^{(i)} = \tilde{W}_i^*$, $i = 0, \ldots, s-1$. Solve this equation system for $a_0, \ldots, a_{s-1}$.

7. The solution to (4.41) is $W = a_0\xi_0 \cdot \ldots \cdot a_{s-1}\xi_{s-1}$ or $W_i = a_0\xi_0^{(i)} \cdot \ldots \cdot a_{s-1}\xi_{s-1}^{(i)}$, $i = 0, 1, \ldots$.

Exercise 4.38
Express the solution to the difference equation $W_{i+2} - W_i = 0$, $i = 0, 1, \ldots$ in terms of the initial conditions $W_0 = \tilde{W}_0^*$ and $W_1 = \tilde{W}_1^*$. ■

Exercise 4.39
Express the solution to the difference equation $W_{i+3} - 2W_{i+2} - 7W_{i+1} - 4W_i = 0$, $i = 0, 1, \ldots$ in terms of the initial conditions $W_0 = \tilde{W}_0^*$, $W_1 = \tilde{W}_1^*$ and $W_2 = \tilde{W}_2^*$. ■

Exercise 4.40
Express the solution to the difference equation $W_{i+2} - W_{i+1} - W_i = 0$, $i = 0, 1, \ldots$ in terms of the initial conditions $W_0 = 0$ and $W_1 = 1$. Hint: This is the difference equations generating the famous Fibonacci sequence so the correct result is $W = \left\{ \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^i - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2} \right)^i \right\}_{i=0}^{\infty}$. The first few numbers in the sequence are 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765 as can be computed with the Maple procedure `Fib:=proc(n::nonnegint) option remember; if n<2 then n else Fib(n-1)+Fib(n-2) end if; end proc;`. ■

It should be noted that the results obtained above do not easily generalize to the non constant coefficient case since there the characteristic polynomial has coefficients and hence also roots depending on i. We shall not consider this case here but refer to [4] section 11.4 for details. Instead we immediately turn to the inhomogeneous case.

4.3.2 Linear, constant coefficient, Inhomogeneous Difference Equations of order s

We consider equation systems of the form

$$\sum_{j=0}^{s} \alpha_j W_{i+j} = \phi_{i+s},\ i = 0, 1, \ldots \quad (\alpha_s = 1,\ \alpha_0 \neq 0) \tag{4.50}$$
$$W_0 = \tilde{W}_0^*, \ldots, W_{s-1} = \tilde{W}_{s-1}^*,$$

where $\tilde{W}_0^*, \ldots, \tilde{W}_{s-1}^*$ are known constants. The notation is differing from that of section 4.3.1 only by the fact that the right hand sides are non trivial. Hence the denotion non homogeneous difference equations.

The solution to an inhomogeneous difference equation can be expressed as the sum of the solution to the corresponding homogeneous difference equation and an addition that we shall call the inhomogeneous solution. Note that this is *not* the same as constructing the solution to the difference equation by taking any solution to the inhomogeneous difference equation, adding an arbitrary solution to the corresponding homogeneous difference equation and then at the end fitting the initial conditions using the arbitrary constants in the arbitrary solution to the homogeneous difference equation. (This latter is the standard approach for inhomogeneous differential equations, but is *not* used here for difference equations).

To get the appropriate setup, we need a new basis for the space $\mathcal{S} = \text{Span}\left\{\{\xi_\ell^{(i)}\}_{i=0}^{\infty}\right\}_{\ell=0}^{s-1}$ of fundamental solutions for the homogeneous difference equation introduced in section 4.3.1, equation (4.42). It was shown in the proof of theorem 4.37 on page 129 that there exists a Lagrange type basis for $\mathcal{S}$ with basis functions $\{\psi_\ell^{(i)}\}_{i=0}^{\infty}$, $\ell = 0, \ldots, s-1$ having the properties $\psi_\ell^{(i)} = \delta_{i,\ell}$ for $i, \ell = 0, \ldots, s-1$. For the practical construction of the $\psi_\ell^{(i)}$ we use the fact that when we have 2 bases for the same space $\mathcal{S}$, then one can be expressed as a linear combination of the other, i.e. there exist constants $g_{\ell,j}$, $j = 0, \ldots, s-1$ such that $\{\psi_\ell^{(i)}\}_{i=0}^{\infty} = \sum_{j=0}^{s-1} g_{\ell,j}\{\xi_j^{(i)}\}_{i=0}^{\infty}$ where the coefficients $g_{\ell,j}$ are chosen such that $\psi_\ell^{(i)} = \delta_{i,\ell}$ for $i, \ell = 0, \ldots, s-1$. For example $\psi_0^{(i)}$ is determined from the equation system $\sum_{j=0}^{s-1} g_{0,j}\xi_j^{(0)} = 1$ and $\sum_{j=0}^{s-1} g_{0,j}\xi_j^{(i)} = 0$ for $i = 1, \ldots, s-1$ ■ .

Theorem 4.41 *The unique solution to* (4.50) *is*

$$\{W_i\}_{i=0}^{\infty} = \{W_i^{hom}\}_{i=0}^{\infty} + \{W_i^{inhom}\}_{i=0}^{\infty} \tag{4.51}$$

where

$$\{W_i^{hom}\}_{i=0}^{\infty} = \{\sum_{\ell=0}^{s-1} \tilde{W}_\ell^* \psi_\ell^{(i)}\}_{i=0}^{\infty} \tag{4.52}$$

and

$$\{W_i^{inhom}\}_{i=0}^{\infty} = \underbrace{0,\ldots,0}_{s\ times}, \{\sum_{t=0}^{i-s} \phi_{t+s}\psi_{s-1}^{(i-t-1)}\}_{i=s}^{\infty}, \tag{4.53}$$

i.e. $W_i^{inhom} = 0$ *for* $i = 0,\ldots,s-1$ *and* $W_i^{inhom} = \sum_{t=0}^{i-s} \phi_{t+s}\psi_{s-1}^{(i-t-1)}$ *for* $i = s, s+1, \ldots$.

To allow a simplifying notation, we shall define $\phi_{i+s} = 0$ for $i < 0$ and $\psi_{s-1}^{(i)} = 0$ for $i < 0$. We then have

$$W_i^{inhom} = \sum_{t=s}^{i} \phi_t \psi_{s-1}^{(i-t+s-1)}, \text{ for } i = 0,1,\ldots \tag{4.54}$$

where $\sum_{t=s}^{i}$ means $t = s, s-1, \ldots, i$ for $i \le s$ and $t = s, s+1, \ldots, i$ for $i \ge s$ so that $W_0^{inhom} = \phi_s \psi_{s-1}^{(-1)} = 0$ $(i = 0)$ and $W_i^{inhom} = \phi_s \psi_{s-1}^{(i-1)} = \delta_{i-1,s-1} = 0$, for $i = 1,\ldots,s-1$, $W_i^{inhom} = \sum_{t=s}^{i} \phi_t \psi_{s-1}^{(i-t+s-1)} = \sum_{t=0}^{i-s} \phi_{t+s}\psi_{s-1}^{(i-t-1)}$, for $i \ge s$.

★ For advanced readers 4.42 ★

Proof: of theorem 4.41

The uniqueness is clear from (4.50) when it is rewritten in the form $W_{i+s} = (\phi_{i+s} - \sum_{j=0}^{s-1} \alpha_j W_{i+j})/\alpha_s$, $i = 0, 1, \ldots$. W_s is then uniquely determined from the initial data $W_0, \ldots, W_{s-1}$. W_{s+1} is similarly uniquely determined from $W_1, \ldots, W_s$ and so on.

Next we will demonstrate that W_i satisfy the initial conditions of (4.50), so let us take $i \in \{0,\ldots,s-1\}$. Then $W_i = \sum_{\ell=0}^{s-1} \tilde{W}_\ell^* \psi_\ell^{(i)} + 0$ since $W_i^{inhom} = 0$ for $i = 0,\ldots,s-1$. Since $\psi_\ell^{(i)} = \delta_{i,\ell}$, only the term $\ell = i$ survives in the first sum and gives $W_i = \tilde{W}_i^*$ as required.

Now we turn to the iteration equation system in (4.50):

Note that $\psi_t^{(i)} = \sum_{k=1}^{s'} \sum_{m=0}^{m_k-1} g_{t,\ell(k,m)} i^m r_k^i$ for some set of coefficients $g_{t,\ell(k,m)}$, $t, \ell = 0,\ldots,s-1$ and $\sum_{j=0}^{s} \alpha_j (i+j)^m r_k^{i+j} = 0$, $\forall i = 0, 1, \ldots$ and the relevant k and m-values by (4.45), so that

$$\sum_{j=0}^{s} \alpha_j \psi_\ell^{(i+j)} = 0, \ \forall \ell = 0,\ldots,s-1, \ \forall i = 0,1,\ldots. \tag{4.55}$$

Hence we have

$$(4.56)\ \sum_{j=0}^{s} \alpha_j W_{i+j} = \sum_{\ell=0}^{s-1} \tilde{W}_\ell^* \sum_{j=0}^{s} \alpha_j \psi_\ell^{(i+j)} + \sum_{j=0}^{s} \sum_{t-s=0}^{i+j-s} \alpha_j \phi_{(t-s)+s} \psi_{s-1}^{(i+j-(t-s)-1)}$$
$$= \sum_{j=0}^{s} \sum_{k=0}^{i+j-s} \phi_{k+s} \alpha_j \psi_{s-1}^{(i+j-k-1)} \text{ since the first sum is 0 by (4.55).}$$

We shall collect the terms in (4.56) as coefficients of the various ϕ_{k+s}:
$k = i$ requires $j = s$ so the only coefficient to ϕ_{i+s} is $\alpha_s \psi_{s-1}^{(s-1)} = 1$.
$k = i + a$ for some $a = 1, 2, \ldots$ requires $j \geq s + a$ but this is impossible, so there are no such terms.
$k = i - a$ for some $a = 1, 2, \ldots$ requires $j \geq s - a$ so we get the following coefficient to ϕ_{i-a+s}:

$$(4.57)\ \sum_{j=s-a}^{s} \alpha_j \psi_{s-1}^{(j+a-1)} = \sum_{j=0}^{s} \alpha_j \psi_{s-1}^{(j+a-1)} \text{ since } \psi_{s-1}^{(j+a-1)} = 0 \text{ for } j < s - a$$
$$= 0 \text{ by (4.55).}$$

Putting these results together we get $\sum_{j=0}^{s} \alpha_j W_{i+j} = \phi_{i+s}$ as required. ■★

Note that the solution (4.51) requires the expressions for the Lagrange type basis functions $\psi_\ell^{(i)}$.

Example 4.43 $W_{i+3} - W_i = \phi_{i+3}$, $i = 0, 1, \ldots$, $W_i = \tilde{W}_i^*$, $i = 0, 1, 2$
First we find the roots of the characteristic polynomial $\rho(r) = r^3 - 1$. In this case we get 3 simple roots r_1, r_2 and r_3 so that we can write up the 3 fundamental solutions $\{r_1^i\}_{i=0}^\infty$, $\{r_2^i\}_{i=0}^\infty$ and $\{r_3^i\}_{i=0}^\infty$. The fundamental solution space $\mathcal{S}$ is then made up of all sequences $\{W_i\}_{i=0}^\infty = \{ar_1^i + br_2^i + cr_3^i\}_{i=0}^\infty$ for any arbitrary a, b and c.
The solution to the homogeneous problem ($\phi_i = 0$ for all i) is uniquely determined as the sequence in $\mathcal{S}$ satisfying $W_i = \tilde{W}_i^*$ for $i = 0, 1, 2$. Hence a, b and c are determined by the equation system $a(x_0, x_1, x_2)r_1^i + b(x_0, x_1, x_2)r_2^i + c(x_0, x_1, x_2)r_3^i = x_i$, $i = 0, 1, 2$ with $x_i = \tilde{W}_i^*$ for $i = 0, 1, 2$.
The 3 Lagrange type basis functions $\psi_\ell^{(i)}$, $\ell = 0, 1, 2$ all lie in $\mathcal{S}$, i.e. $\psi_\ell^{(i)} = ar_1^i + br_2^i + cr_3^i$ for some a, b and c determined by the lagrange properties $\psi_\ell^{(i)} = \delta_{i,\ell}$ for $i, \ell = 0, 1, 2$. For example $\psi_0^{(i)}$ satisfies $\psi_0^{(0)} = 1$, $\psi_0^{(1)} = 0$, $\psi_0^{(2)} = 0$. But this is nothing but the equation system $a(x_0, x_1, x_2)r_1^i + b(x_0, x_1, x_2)r_2^i + c(x_0, x_1, x_2)r_3^i = x_i$, $i = 0, 1, 2$ with $(x_0, x_1, x_2) = (1, 0, 0)$, i.e. $\psi_0^{(i)} = a(1, 0, 0)r_1^i + b(1, 0, 0)r_2^i + c(1, 0, 0)r_3^i$. Similarly $\psi_1^{(i)} = a(0, 1, 0)r_1^i + b(0, 1, 0)r_2^i + c(0, 1, 0)r_3^i$ and $\psi_2^{(i)} = a(0, 0, 1)r_1^i + b(0, 0, 1)r_2^i + c(0, 0, 1)r_3^i$.

Having recovered the lagrange basis functions, the homogeneous, inhomogeneous and full solutions are easily recovered. The Maple commands are given below.

```
> charroots:=[solve(r^3-1=0,r)];
> for k from 0 to 2 do
     eq[k]:=a*charroots[1]^k+b*charroots[2]^k
            +c*charroots[3]^k=x[k]
  end do;
> assign(solve([eq[0],eq[1],eq[2]],[a,b,c]));
> a:=unapply(a,x[0],x[1],x[2]);b:=unapply(b,x[0],x[1],x[2]);
  c:=unapply(c,x[0],x[1],x[2]);
> Lagrangefct:=proc(i,x,y,z)
     a(x,y,z)*charroots[1]^i+b(x,y,z)*charroots[2]^i
     +c(x,y,z)*charroots[3]^i;
  end proc:
> homsoluLagrange:=proc(i)
     global W0,W1,W2;
     W0*Lagrangefct(i,1,0,0)+W1*Lagrangefct(i,0,1,0)
     +W2*Lagrangefct(i,0,0,1)
  end proc:
> inhomsoluLagrange:=proc(i)
     global p;
     if i in {0,1,2} then
        0
     else
        sum(p[t]*Lagrangefct(i-t+2,0,0,1),t=3..i)
     end if
  end proc:
> fullsolu:=proc(i)
     homsoluLagrange(i)+inhomsoluLagrange(i)
  end proc:
> for i from 0 to 8 do simplify(homsoluLagrange(i)) end do;
> for i from 0 to 8 do simplify(inhomsoluLagrange(i)) end do;
> for i from 0 to 8 do simplify(fullsolu(i)) end do;
> for i from 0 to 8 do simplify(fullsolu(i+3)-fullsolu(i))
  end do;
```

Exercise 4.44
Copy the Maple code above into a Maple worksheet. Run the code. Insert explanatory comments in the worksheet. ■

4.3.3 Return to the linear, constant coefficient multi step methods

★ **For advanced readers 4.45** ★ If $\alpha_0 = \ldots = \alpha_t = 0$ in (4.40) we "shift down" the indices by t ($j \to j - t$) so that (4.40) takes the form $\sum_{j=0}^{s-t} \alpha_{j+t} W_{i+j+t} = \phi_{i+s}$, $i = 0, \ldots, n - s - 1$. Letting $s^* = s - t$, $\alpha_j^* = \alpha_{j+t}$ and $W_{i+j}^* = W_{i+j+t}$ (4.40) then takes the form $\sum_{j=0}^{s^*} \alpha_j^* W_{i+j}^* = \phi_{i+s}$, $i = 0, \ldots, n - s - 1$ which can be treated exactly as the case $\alpha_0 \neq 0$. Without loss of generality we shall then only consider the case $\alpha_0 \neq 0$ here.

Proof of theorem 4.32 on page 125:
We are now able to express the solution to our original problem (4.40) in the form

$$W_i = \sum_{\ell=0}^{s-1} \tilde{W}_\ell^* \psi_\ell^{(i)} + \sum_{t=s}^{i} \phi_t \psi_{s-1}^{(i-t+s-1)}, \ i = 0, 1, \ldots \tag{4.58}$$

using the notation from section 4.3.1 and 4.3.2.

To bound W_i note that if the root condition holds then all characteristic fundamental solutions are uniformly bounded sequences: If $|r_k| = 1$ then $m_k = 1$ and $\{|r_k^i|\}_{i=0}^{\infty}$ is uniformly bounded by 1. If alternatively $|r_k| < 1$ then $\{|i^m r_k^i|\}_{i=0}^{\infty}$ is uniformly bounded for any $m = 0, \ldots, m_k - 1$ ($|r_k^i|$ goes towards zero faster when i goes towards infinity than $|i^{m_k - 1}|$ grows). Hence also all the other fundamental solutions are uniformly bounded sequences, so that there exists a positive real number M such that $|\psi_\ell^{(i)}| \leq M$, for $\ell = 0, \ldots, s - 1$ and $i = 0, 1, 2, \ldots$ and hence

$$|W_i| \leq M \left\{ s \max_{\ell=0,\ldots,s-1} |\tilde{W}_\ell^*| + \sum_{t=s}^{i} |\phi_t| \right\}, \ i = s, s + 1, \ldots \tag{4.59}$$

The first term is bounded by assumption ($\max_{\ell=0,\ldots,s-1} |\tilde{W}_\ell^*| < C\epsilon$). To bound the ϕ_t's we use the definition in (4.40), the Lipschitz continuity of f

(with constant L) and let $\beta = \max_{j=0,\ldots,s} |\beta_j|$ and $\epsilon = \max_{i=1,\ldots,n} |\delta_{\epsilon,i}|$. Then

$$
\begin{aligned}
|\phi_t| &= |h\delta_{\epsilon,t+1} + h\sum_{j=0}^{s} \beta_j \left(\tilde{f}_{t+j-s+1} - \tilde{f}_{\epsilon,t+j-s+1} \right)| \\
&\leq h\epsilon + h\beta L \sum_{j=0}^{s} |W_{t+j-s}|
\end{aligned} \tag{4.60}
$$

and hence

$$
|W_i| \leq M \left\{ sC\epsilon + |I|\epsilon + h\beta L \sum_{t=s}^{i} \sum_{j=0}^{s} |W_{t+j-s}| \right\}, \; i = s, \ldots, n-1. \tag{4.61}
$$

Here the double sum can be expanded to

$$
\begin{aligned}
&\sum_{t=s}^{i} \sum_{j=0}^{s} |W_{t+j-s}| \\
&= |W_0| + |W_1| + \ldots + |W_{s-1}| + |W_s| \\
&+ |W_1| + |W_2| + \ldots + |W_s| + |W_{s+1}| \\
&\vdots \\
&+ |W_{i-s-1}| + |W_{i-s}| + \ldots + |W_{i-2}| + |W_{i-1}| \\
&+ |W_{i-s}| + |W_{i-s+1}| + \ldots + |W_{i-1}| + |W_i| \\
&= \begin{cases}
|W_0| + 2|W_1| + 3|W_2| + \ldots + (i-s+1)|W_{i-s}| + \ldots \\
\quad + (i-s+1)|W_s| + \ldots + 3|W_{i-2}| + 2|W_{i-1}| + |W_i| \\
\quad \text{for } i - s \leq s \\
|W_0| + 2|W_1| + 3|W_2| + \ldots + (s+1)|W_s| + \ldots \\
\quad + (s+1)|W_{i-s}| + \ldots + 3|W_{i-2}| + 2|W_{i-1}| + |W_i| \\
\quad \text{for } i - s > s
\end{cases} \\
&\leq |W_0| + |W_i| + (s+1) \sum_{j=1}^{i-1} |W_j| \text{ for } i = s, \ldots, n-1.
\end{aligned} \tag{4.62}
$$

We may pick h so small that $Mh\beta L < 1$ and throw the W_i-term on the right hand side onto the left hand side, still maintaining a positive coefficient to get

$$
|W_i| \leq \frac{M}{1 - Mh\beta L} \left\{ (s + h\beta L)C\epsilon + |I|\epsilon + (s+1)h\beta L \sum_{j=2}^{i} |W_{j-1}| \right\}, \quad \text{for } i = s, \ldots, n-1 \tag{4.63}
$$

and by selecting M sufficiently large and h sufficiently small so that $\frac{M}{1-Mh\beta L} \geq 1$ we also have

$$|W_i| \leq C\epsilon \leq \frac{M}{1-Mh\beta L}\left\{(sC + h\beta LC + |I|)\epsilon + |I|\beta L\sum_{j=2}^{i}|W_{j-1}|\right\}, \quad \text{for } i = 1, \ldots, s-1. \tag{4.64}$$

Hence taking

$$\begin{aligned}
\phi_i &= |W_i| \text{ for } i = 1, \ldots, n-1,\\
c_1 &= \frac{M}{1-Mh\beta L}\left(C + h\beta LC + |I|\right)\epsilon,\\
c_j &= \frac{M}{1-Mh\beta L}C\epsilon \text{ for } j = 2, \ldots, s,\\
c_j &= 0 \text{ for } j = s+1, \ldots, n-1,\\
b_j &= \frac{M}{1-Mh\beta L}(s+1)h\beta L \text{ for } j = 2, \ldots, n-1,
\end{aligned}$$

we finally have

$$\phi_1 \leq |c_1|,\ \phi_i \leq \sum_{j=1}^{i}|c_j| + \sum_{j=2}^{i}|b_j|\phi_{j-1},\ \text{for } i = 2, \ldots, n-1 \tag{4.65}$$

which by the discrete Gronwall lemma (4.5) implies

$$\phi_i \leq \sum_{j=1}^{i}|c_j|e^{\sum_{j=2}^{i}|b_j|},\ \text{for } i = 2, \ldots, n-1 \tag{4.66}$$

or

$$|W_i| \leq \frac{M}{1-Mh\beta L}\left\{(s + h\beta L)C + |I|\right)\epsilon e^{\frac{M}{1-Mh\beta L}|I|\beta L(s+1)}, \quad \text{for } i = 2, \ldots, n-1, \tag{4.67}$$

to obtain $|W_i| < \tilde{C}\epsilon$ for $i = s, \ldots, n-1$, hence proving theorem 4.32 on page 125. ■★

There is a connection between the order of convergence q of a multi step method and the number of steps s:

Theorem 4.46 (The First Dahlquist barrier) *There are no zero stable, s-step, linear, constant coefficient multi step methods with order of convergence greater than $s+1$ if s is odd and $s+2$ if s is even.*

Exercise 4.47
Based on the first Dahlquist barrier, comment on the optimality of the order of convergence of all of the following methods: AB(1), ..., AB(4), AM(1), ..., AM(5) and BDF(1), ..., BDF(6). ■

4.4 Non asymptotic error analysis – Absolute stability for linear, constant coefficient, multi step FDM's for $u' = \lambda u$

Definition 4.13 on page 113 of absolute and $\mathcal{A}$-stability in section 4.2 applies also to the uniform step length, linear, constant coefficient, s step FDM's (4.8) considered in section 4.3 when applied to the special DEP $u' = \lambda u,\ t > 0,\ u(0) = 1$.

Since $f = \lambda u \neq 0$ the problem is non homogeneous, but because of the linearity of f with respect to u it can be treated as a homogeneous problem: (4.8) takes the form

$$
\begin{aligned}
&\tilde{U}_1 = \tilde{U}_1^*, \ldots, \tilde{U}_s = \tilde{U}_s^*, \\
&\sum_{j=0}^{s} \left(\frac{a_j}{h} \tilde{U}_{i-j} + b_j \tilde{f}_{i-j} \right) = \sum_{j=0}^{s} \left(\frac{a_j}{h} + \lambda b_j \right) \tilde{U}_{i-j} = 0, \\
&i = s+1, s+2, \ldots,\ a_0 = -1,\ (a_s, b_s) \neq (0,0),
\end{aligned}
\tag{4.68}
$$

where we do not stop i at n since we are dealing with absolute stability, i.e. we want to investigate what happens to $\tilde{U}_i$ as i goes to infinity. To bring this on the form of (4.41) we need some coordinate transformations:

First pick $i^* = i - (s+1)$ in order to let i^* run over the values $0, 1, \ldots$ appropriate for (4.41). (4.68) is then transformed into

$$
\begin{aligned}
&\tilde{U}_1 = \tilde{U}_1^*, \ldots, \tilde{U}_s = \tilde{U}_s^*,\ \sum_{j=0}^{s} (a_j + h\lambda b_j)\, \tilde{U}_{i^*+s+1-j} = 0, \\
&i^* = 0, 1, \ldots,\ a_0 = -1,\ (a_s, b_s) \neq (0,0).
\end{aligned}
\tag{4.69}
$$

Since we are interested in the product $h\lambda$ when dealing with absolute stability we have also multiplied the middle equation in (4.68) by h to get to (4.69).

Second pick $j^* = s - j$ in order to keep j^* summing over the values $0, \ldots, s$ while at the same time changing the index on $\tilde{U}$ from $i^* - j + cst$ to

$i^* + j^* + cst.$ (4.69) is then transformed into

$$\begin{aligned} (4.70) \qquad & \tilde{U}_1 = \tilde{U}_1^*, \ldots, \tilde{U}_s = \tilde{U}_s^*, \ \sum_{j^*=0}^{s} \left(a_{s-j^*} + h\lambda b_{s-j^*}\right) \tilde{U}_{i^*+j^*+1} = 0, \\ & i^* = 0, 1, \ldots, \ a_0 = -1, \ (a_s, b_s) \neq (0,0). \end{aligned}$$

Third pick $\gamma_{j^*} = a_{s-j^*} + h\lambda b_{s-j^*}$ and $W_{i^*+j^*} = \tilde{U}_{i^*+j^*+1}$. (4.70) is then transformed into

$$\begin{aligned} (4.71) \qquad & W_0 = \tilde{U}_1^*, \ldots, W_{s-1} = \tilde{U}_s^*, \ \sum_{j^*=0}^{s} \gamma_{j^*} W_{i^*+j^*} = 0, \\ & i^* = 0, 1, \ldots, \ a_0 = -1, \ (a_s, b_s) \neq (0,0). \end{aligned}$$

Finally we need to scale the system by dividing the sum-equation by $\gamma_s = -1 + h\lambda b_0$ which is non zero except for at most one particular value of h which we shall then disallow. Further we define $\alpha_{j^*} = \gamma_{j^*}/\gamma_s$ in order to obtain the condition $\alpha_s = 1$. For the condition $\alpha_0 \neq 0$ we have $\alpha_0 = \gamma_0/\gamma_s = \frac{a_s + h\lambda b_s}{\gamma_s}$ which since $(a_s, b_s) \neq (0,0)$ is non zero apart from at most another one particular value of h which we shall also disallow. (4.71) is then transformed into

$$\begin{aligned} (4.72) \qquad & W_0 = \tilde{U}_1^*, \ldots, W_{s-1} = \tilde{U}_s^*, \ \sum_{j^*=0}^{s} \alpha_{j^*} W_{i^*+j^*} = 0, \\ & i^* = 0, 1, \ldots, \ \alpha_0 = 1, \ \alpha_s \neq 0. \end{aligned}$$

(4.72) is apart from the * on the i and j now in the form of (4.41) and can be solved with the approach from section 4.3.1: The characteristic polynomial for (4.72) is

$$\begin{aligned} (4.73) \quad \pi(r) \ &:= \ \sum_{j=0}^{s} \alpha_j r^j = \frac{1}{\gamma_s} \sum_{j=0}^{s} \gamma_j r^j = \frac{1}{\gamma_s} \sum_{j=0}^{s} \left(a_{s-j} + h\lambda b_{s-j}\right) r^j \\ &= \ -\frac{1}{\gamma_s} \left(\rho(r) - h\lambda\sigma(r)\right), \end{aligned}$$

where ρ and σ are respectively the first and second characteristic polynomial for the multistep method. Denoting the roots of π by $r_k^\pi(h\lambda)$, $k = 1, \ldots, s^\pi$ with multiplicities $m_1^\pi, \ldots, m_{s^\pi}^\pi$, the solution to (4.72) takes by theorem 4.37 on page 129 the form

$$(4.74) \qquad \{W_i\}_{i=0}^{\infty} = \{\sum_{k=1}^{s^\pi} \sum_{j=0}^{m_k^\pi - 1} \gamma_{k,j}(h) i^j (r_k^\pi(h\lambda))^i\}_{i=0}^{\infty},$$

for some constants $\gamma_{k,j}(h)$ independent of i but depending on h.

Exercise 4.48
Find explicitly the solutions to AB(2), AB(3), AM(3) and AM(4) for $u' = \lambda u$, $t > 0$, $u(0) = 1$, with $\lambda = 5$. When more data is needed for the priming the pumps process then use the exact solution $u(t) = \exp(\lambda t)$ in the appropriate nodal points. To verify your solutions pick $n = 100$ and $h = 0.01$ and plot the numerical solution points together with the exact solution in the interval $[0, 1]$. ■

From (4.74) and definition 4.13 on page 113 it is clear that

Theorem 4.49 *A uniform step length, linear, constant coefficient, s step FDM given by* (4.8)*, applied to the special DEP* $u' = \lambda u,\ t > 0,\ u(0) = 0$*, is*

- *absolutely stable for given h and* $\lambda \Leftrightarrow |r_k^\pi(h\lambda)| < 1,\ k = 1, \ldots, s^\pi$
- $\mathcal{A}$*-stable* $\Leftrightarrow |r_k^\pi(\xi)| < 1,\ k = 1, \ldots, s^\pi\ \forall \xi :\ \mathcal{R}e(\xi) < 0$

That $\mathcal{A}$-stability is a fairly rare occurrence is expressed in the following result

Theorem 4.50 (The Second Dahlquist barrier)
There are no explicit, $\mathcal{A}$*-stable, linear, constant coefficient multi step methods.*
There are no $\mathcal{A}$*-stable, linear, constant coefficient, multi step methods with order of convergence greater than* 2*.*

Exercise 4.51
Which are the $\mathcal{A}$-stable Adams and BDF methods? ■

Exercise 4.52
Construct, using MATLAB or Maple or similar, the absolute stability regions for AB(2), AM(3), BDF(3) and BDF(5). Is BDF(3) $\mathcal{A}$-stable? ■

The s step Adams-Bashford AB(s) and Adams-Moulton AM($s+1$) methods have regions of absolute stability $\mathcal{A} \supset \{h\lambda \in]0, -|\delta|[\}$ where both δ and the entire region $\mathcal{A}$ decreases when s increases and is smaller for AB(s) than for AM($s + 1$). Instead Backward Differentiation Formula methods BDF(s) all have regions of absolute stability $\mathcal{A} \supset \{h\lambda \in]0, -\infty[\}$ while $\mathcal{A}$ also here decreases with increasing s. For plots of regions of absolute stability for various Adams-Bashford, Adams-Moulton and Backward Differentiation Formula methods see for example [4] Figures 11.5 and 11.6.

4.5 Convergence, stability and consistency of Runge-Kutta FDM's for $u' = f(t, u)$

As mentioned in the beginning of section 4.3, for the multi step methods, order of convergence is increased compared to the single step methods considered in section 4.1 by increasing the number of previous steps taken into consideration when evaluating the next value. The advantage is cheap function evaluations and the disadvantage is the unsuitability for adaptivity.

In this section we shall consider methods that maintain the one step form making them better suited for adaptivity, increasing order of convergence at the price of more complex ϕ functions and hence more costly function evaluations. Compared to section 4.1 we generalize by considering also implicit methods, but we also specialize, considering one particular family of ϕ functions.

Definition 4.53 *Let* $\mathbf{A} \in \mathcal{R}^{s\times s}$ *and* $\mathbf{b} \in \mathcal{R}^s$ *be given, let* $\mathbf{c} \in \mathcal{R}^s$ *satisfy*

$$c_j = \sum_{k=1}^{s} a_{jk}, \ j = 1, \ldots, s \tag{4.75}$$

and define the Butcher Array

$$B = \begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^T \end{array} \tag{4.76}$$

The s stage Runge-Kutta method (RK(s)) for (1.10) *with nodes* $x_1, \ldots, x_n$, *corresponding to the Butcher array* B *has the optimal representation*

$$\frac{\tilde{U}_i - \tilde{U}_{i-1}}{h_{i-1}} = \sum_{j=1}^{s} b_j \tilde{K}_{j,i-1}, \ i = 2, \ldots, n \tag{4.77}$$

$$\tilde{K}_{j,i-1} = f(x_{i-1} + c_j h_{i-1}, \tilde{U}_{i-1} + h_{i-1} \sum_{k=1}^{s} a_{jk} \tilde{K}_{k,i-1}), \quad j = 1, \ldots, s, \ i = 2, \ldots, n, \tag{4.78}$$

where the i'th step length $h_{i-1} = x_i - x_{i-1}$, $i = 2, \ldots, n$. *To start up the method* $\tilde{U}_1 = u^*$ *must be known.*

Note that RK(s)-methods are one step methods since only $\tilde{U}_i$ and $\tilde{U}_{i-1}$ are involved in each step in (4.77–4.78). Also the RK-methods are generally implicit (see (3.45)) since the $\tilde{K}_{j,i}$'s must be recovered by solving the $s \times s$

equation system (4.78) before being able to insert them into (4.77). If the matrix $\mathbf{A}$ can be put in lower triangular form ($a_{jk} = 0$ if $j < k$) after reordering of the j's, the equation system simplifies and can be solved one equation at the time so that the $K_{j,i}$'s can be recovered solving s single nonlinear equations. In this case the RK is said to be Semi-Implicit. Only if the matrix $\mathbf{A}$ can be put in lower triangular form with zeros on the diagonal ($a_{jk} = 0$ if $j \leq k$) after reordering of the j's, does the RK methods become explicit.

We shall here not go into any details about the RK-methods but simply state some results that can be found in the literature. We refer to [4] §11.8 for details and references to proofs.

The local truncation errors $\tau_i(h_{i-1}) = \frac{u_i - u_{i-1}}{h_{i-1}} - \sum_{j=1}^{s} b_j K_{j,i-1}$ ($K_{j,i-1}$ is $\tilde{K}_{j,i-1}$ with $\tilde{U}_i$ replaced by u_i, $i = 2, \ldots, n$) are generally hard to find (only $\tau_1 = u(t_0) - u^*$ is easy), but it is possible to obtain some results at least for the special case of uniform step lengths $h_i = h$, $i = 1, \ldots, n-1$:

Theorem 4.54 *For uniform step length RK(s) methods*

$$\begin{aligned} \text{(4.79)} \qquad \textit{Consistency} \quad &\Leftrightarrow \quad \tau(h) = \max_{i=1,\ldots,n} |\tau_i(h)| \to 0 \text{ as } h \to 0 \\ &\Leftrightarrow \quad \tau_1(h) \to 0 \text{ as } h \to 0 \text{ and } \sum_{j=1}^{s} b_j = 1. \end{aligned}$$

Lax' theorem: *For a consistent, uniform step length RK(s) method*

$$\begin{aligned} \text{(4.80)} \qquad & f \text{ Lipschitz continuous in its 2nd variable} \\ & \Rightarrow \textit{ Zero stability } \Rightarrow \textit{ Convergence} \end{aligned}$$

and the order of convergence is the same as the order of consistency and is at most s, the number of stages, for $s < 5$ and at most $s - 1$ for $s \geq 5$. More precisely the minimal number of stages to get an order from 1 to 8 is shown here:

(4.81)

order	1	2	3	4	5	6	7	8
min stages	1	2	3	4	6	7	9	11

Example 4.55 Consistent RK(1)'s
The Butcher array for consistent, 1 stage RK methods all take the form

$$\text{(4.82)} \qquad \begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^T \end{array} = \begin{array}{c|c} a & a \\ \hline & 1 \end{array}$$

for some real constant a, giving the following expression for the RK(1) methods:

$$\begin{aligned}(4.83)\qquad &\tilde{U}_1 = u^*,\ \tilde{U}_i = \tilde{U}_{i-1} + h_{i-1}\tilde{K}_{1,i-1},\ i = 2,\ldots,n\\ &\tilde{K}_{1,i-1} = f(x_{i-1} + ah_{i-1}, \tilde{U}_{i-1} + ah_{i-1}\tilde{K}_{1,i-1}),\ i = 2,\ldots,n.\end{aligned}$$

In general this is a semi-implicit method. Only for $a = 0$ we get an explicit method denoted $\mathrm{RK}_0^{ex}(1)$ which is easily seen to be the Forward Euler method. The most common semi-implicit RK(1) method is the one with $a = \frac{1}{2}$ denoted $\mathrm{RK}_{\frac{1}{2}}^{im}(1)$, the implicit midpoint rule. ■

Example 4.56 Consistent RK(2)'s
The Butcher array for consistent, 2 stage RK methods all take the form

$$(4.84)\qquad \begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^T \end{array} = \begin{array}{c|c} \begin{bmatrix} c+e \\ a+d \end{bmatrix} & \begin{bmatrix} c & e \\ a & d \end{bmatrix} \\ \hline & \begin{bmatrix} 1-b & b \end{bmatrix} \end{array}$$

for some real constants a, b, c, d, e, giving the following expression for the RK(2) methods:

$$\begin{aligned}(4.85)\quad &\tilde{U}_1 = u^*,\ \tilde{U}_i = \tilde{U}_{i-1} + h_{i-1}\left((1-b)\tilde{K}_{1,i-1} + b\tilde{K}_{2,i-1}\right),\\ &\tilde{K}_{1,i-1} = f\left(x_{i-1} + (c+e)h_{i-1}, \tilde{U}_{i-1} + h_{i-1}\left(c\tilde{K}_{1,i-1} + e\tilde{K}_{2,i-1}\right)\right),\\ &\tilde{K}_{2,i-1} = f\left(x_{i-1} + (a+d)h_{i-1}, \tilde{U}_{i-1} + h_{i-1}\left(a\tilde{K}_{1,i-1} + d\tilde{K}_{2,i-1}\right)\right),\\ &\qquad\qquad i = 2,\ldots,n.\end{aligned}$$

In general these are fully implicit methods, but for $e = 0$ the methods are semi-implicit and if further $c = d = 0$ the methods are explicit. The 2 most common explicit methods are the $\mathrm{RK}_{1,\frac{1}{2}}^{ex}(2)$ with $a = 1$ and $b = \frac{1}{2}$ which is Heuns method, and $\mathrm{RK}_{\frac{1}{2},1}^{ex}(2)$ with $a = \frac{1}{2}$ and $b = 1$ which is denoted the method of Runge and takes the following form

$$(4.86)\qquad \tilde{U}_1 = u^*,\ \tilde{U}_i = \tilde{U}_{i-1} + h_{i-1}f(x_{i-1} + \frac{1}{2}h_{i-1}, \tilde{U}_{i-1} + \frac{1}{2}h_{i-1}\tilde{f}_{i-1})),\quad i = 2,\ldots,n.$$

Of the implicit methods, probably the implicit trapezoidal rule taking $a = d = \frac{1}{2}$, $c = e = 0$ and $b = \frac{1}{2}$ is most common. It takes the form

$$\begin{aligned}(4.87)\quad &\tilde{U}_1 = u^*,\ \tilde{U}_i = \tilde{U}_{i-1} + \frac{1}{2}h_{i-1}\left(\tilde{f}_{i-1} + \tilde{K}_{2,i-1}\right),\\ &\tilde{K}_{2,i-1} = f\left(x_{i-1} + h_{i-1}, \tilde{U}_{i-1} + \frac{1}{2}h_{i-1}\left(\tilde{f}_{i-1} + \tilde{K}_{2,i-1}\right)\right),\\ &\qquad\qquad i = 2,\ldots,n.\end{aligned}$$

■

Example 4.57 Constructing a consistent, optimal order, explicit RK
Start RK from the exact solution u_{i-1} and let $\hat{U}_i$ be the result.
Select coefficients so that as many Taylor terms as possible of u_i and $\hat{U}_i$ coincide.
For example for the consistent, optimal order, 2 stage, explicit RK the details are as follows:
For a 2 stage RK,

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}. \tag{4.88}$$

The method is explicit only if $a_{11} = a_{12} = a_{22} = 0$ and for simplicity we shall then take $a_{21} = a$. Enforcing also (4.75) and the consistency condition (4.79), for simplicity taking $b_1 = b$, a consistent, explicit, 2 stage RK has the Butcher array

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^T \end{array} = \begin{array}{c|c} \begin{bmatrix} 0 \\ a \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ a & 0 \end{bmatrix} \\ \hline & \begin{bmatrix} b & 1-b \end{bmatrix} \end{array} \tag{4.89}$$

and hence when started in u_{i-1} takes the form

$$\begin{aligned} &\hat{U}_i = u_{i-1} + bhK_{1,i-1} + (1-b)hK_{2,i-1}, \\ &K_{1,i-1} = f(x_{i-1}, u_{i-1}) = f_{i-1}, \\ &K_{2,i-1} = f(x_{i-1} + ah, u_{i-1} + ahK_{1,i-1}), \\ &i = 2, \ldots, n \\ &\Updownarrow \\ &\hat{U}_i = u_{i-1} + bhf_{i-1} + (1-b)hf(x_{i-1} + ah, u_{i-1} + ahf_{i-1}) \\ &\quad = u_{i-1} + bhf_{i-1} + (1-b)h\big(f_{i-1} + ah\partial_1 f_{i-1} + ahf_{i-1}\partial_2 f_{i-1} \\ &\qquad\qquad + \mathcal{O}(h^2)\big) \\ &\quad = u_{i-1} + hf_{i-1} + h^2 a(1-b)\left(\partial_1 f_{i-1} + f_{i-1}\partial_2 f_{i-1}\right) + \mathcal{O}(h^3), \\ &i = 2, \ldots, n. \end{aligned} \tag{4.90}$$

We know from theorem 4.54 that the order of a 2 stage RK method may be at most 2, so we have not specified the third order term. (We could have done that and then verified the theorem on this particular point). A Taylor

expansion of u_i around x_{i-1} gives

$$
\begin{aligned}
(4.91) \qquad u_i &= u_{i-1} + hu'_{i-1} + \frac{h^2}{2}u''_{i-1} + \mathcal{O}(h^3) \\
&= u_{i-1} + hf_{i-1} + \frac{h^2}{2}\left(\partial_1 f_{i-1} + f_{i-1}\partial_2 f_{i-1}\right) + \mathcal{O}(h^3), \\
& i = 2, \ldots, n.
\end{aligned}
$$

Comparing terms of equal order in (4.90) and (4.91) we get the single condition $a(1-b) = \frac{1}{2} \Leftrightarrow b = 1 - \frac{1}{2a}$ so that there is an infinite number of consistent, 2nd order, 2 stage explicit Runge-Kutta methods that all have a Butcher array of the form

$$
(4.92) \qquad \begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^T \end{array} = \begin{array}{c|c} \begin{bmatrix} 0 \\ a \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ a & 0 \end{bmatrix} \\ \hline & \begin{bmatrix} 1 - \frac{1}{2a} & \frac{1}{2a} \end{bmatrix} \end{array} \quad \blacksquare
$$

Exercise 4.58
Explain why the method of construction outlined in example 4.57 works, and how the order of consistency is determined. (Hint: Start showing that $\tau_i = \frac{u_i - \hat{U}_i}{h_{i-1}}$, $i = 2, \ldots, n$). ■

Exercise 4.59
Extend the computations in example 4.57 to prove that there exist no 3rd order, 2 stage explicit Runge-Kutta methods. ■

Exercise 4.60
Show that the Classical Runge Kutta Method, which is the 4 stage, explicit RK with the following Butcher array

$$
(4.93) \qquad \begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^T \end{array} = \begin{array}{c|c} \begin{bmatrix} 0 \\ \frac{1}{2} \\ \frac{1}{2} \\ 1 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \\ \hline & \begin{bmatrix} \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{bmatrix} \end{array}
$$

is 4th order convergent. ■

Exercise 4.61
Write an essay on theorem 4.54 and its proof. ■

Chapter 5

Further issues for $u' = f(t, u)$

5.1 How to solve with an implicit method – Predictor-Corrector FDM's

Explicit FDM's generally have quite small absolute stability regions, meaning in practice that very small step lengths h must be used to avoid oscillatory numerical solutions. Here *very small* is in relation to what should be expected for a method of the given convergence order and with the chosen error tolerance. Implicit methods and in particular $\mathcal{A}$-stable methods generally allow larger step lengths only limited by the given convergence order and the error tolerance chosen. Instead implicit methods generally involve the solution of nonlinear equations while explicit methods only involve multiplications and additions (solution of linear equations) on top of the function evaluations needed by both types of method. This leads to the following Rule of Thumb:

- Explicit methods involve many, small, cheap steps (solving linear equations).
- Implicit methods involve few, big, expensive steps (solving nonlinear equations).

Just how expensive the implicit steps are, determines whether the implicit methods compare favorably to the explicit methods, so let us consider the price of nonlinear equation solvers:

5.1.1 Nonlinear equation solvers

There are many different types of nonlinear equation solvers. To mention just a few we have Newtons method, fixed point iteration, bisection.

All nonlinear equation solvers share the following general algorithm for solving the problem $f(x) = 0$. They have to be started of by an initial guess of a solution provided by the user or chosen at random by the computer, and then they go on with a sequence of "update" steps:

$x_0 =$ initial guess
for $k = 0, 1, 2, \ldots$
    $x_{k+1} =$ update of x_k
end

The update of x_k consists in solving a linear equation expressing x_{k+1} as a function of x_k. For Newtons method the update of x_k is for example given by the condition that x_{k+1} is the point where the tangent to f in the point x_k crosses the x-axis or $f(x_k) - f'(x_k)(x_{k+1} - x_k) = 0 \Leftrightarrow x_{k+1} = x_k - f(x_k)/f'(x_k)$. If the method is convergent then $f(\lim_{k\to\infty} x_k) = 0$ but in practice of course we stop after a finite number of steps, say at x_m. Hence solving a nonlinear equation on a computer in practice means providing an initial guess and then solving a finite sequence of linear problems. Since implicit FDM's means solving nonlinear equations and explicit FDM's means solving linear equations, this by the way means that

- An implicit method is, when it comes to the computational cost, equivalent to a finite sequence of two or more explicit methods.

To understand how long the sequence needs to be, i.e. how big m has to be, in order to get sufficiently precise solutions we need to recall some basic properties of nonlinear equation solvers:

Let us start defining that a nonlinear equation solver is convergent of order q if $|x_{k+1} - x| \leq C|x_k - x|^q$ for some C and k bigger than some treshold K where x is a zero of f, i.e. $f(x) = 0$. With the notation of definition 2.18 on page 26 this is the same as saying that the sequence $\{x_{k+1} - x\}_{k=0}^{\infty}$ is convergent of order q with respect to the sequence $\{x_k - x\}_{k=0}^{\infty}$. Newtons method for example can be shown to be quadratically convergent ($q = 2$). Note that $|x_{k+1} - x| \leq C|x_k - x|^q \Rightarrow |x_k - x| \leq C^k|x_0 - x|^{q^k}$. Since for $q > 1$, q^k grows much faster than k we may in these situations neglect the C^k and consider $|x_0 - x|^{q^k}$ as the upper bound for $|x_k - x|$. If for example $q = 2$ as for Newtons method and if we assume that $|x_0 - x| = 10^{-1}$, then the upper bounds for $|x_k - x|$, $k = 0, 1, 2, 3, 4$ become $10^{-1}, 10^{-2}, 10^{-4}, 10^{-8}, 10^{-16}$. We see the *doubling of the number of correct digits in each step* which is also observed in practice. This means that 4 steps with Newtons method is all that anyone should ever need.

Unfortunately an important part of the convergence result is that k has to be bigger than a treshold K, or rather only when $|x_k - x|$ is sufficiently small

do we start observing the doubling of correct digits in each step. Unless the initial guess is very good and $|x_0 - x|$ is very small, initially we tend to see a very irratic behavior of the nonlinear equation solvers, where the updates are almost randomly moving around on the x-axis. Once one of these "random guesses" happens to be close enough, then we get the doubling of correct digits in each step from then on. The interval around x where we see the doubling of correct digits in each step is called the Asymptotic Tail for the method.

This situation is also described as if the nonlinear equation solver is a *slow starter*. To find a solution to $f(x) = 0$ we start from an initial guess x_0. If $f(x_0)$ is far from zero it may take the method a large number of iterations to get close enough to a solution x for the order of convergence of the method to be apparent (to get in the asymptotic tail for the method) at which point the error decreases rapidly. For the quadratically convergent Newton method for example the number of correct digits generally double for each iteration, but only once the asymptotic tail has been reached.

5.1.2 Predictor corrector methods

For an implicit FDM we basically go from the approximating solution $\tilde{U}_i^{im}$ in x_i to the approximating solution $\tilde{U}_{i+1}^{im}$ in x_{i+1} by using $\tilde{U}_i^{im}$ as initial guess for the nonlinear solver. If the step length $h = x_{i+1} - x_i$ is big, then this guess may be very poor and the nonlinear solver will need a high number of iterations to get in the asymptotic tail where the fast convergence starts.

The million dollar question then is whether there is a faster way to get in the asymptotic tail. The best answer to this question is to use an explicit method to improve the initial guess $\tilde{U}_i^{im}$ to $\tilde{U}_{i+1}^{ex}$ and then use the implicit method starting from $\tilde{U}_{i+1}^{ex}$ to go to $\tilde{U}_{i+1}^{im}$. (Of course we know that an explicit method works well only when h is small but practice shows that when h is big an explicit method often still works better than first step of a nonlinear equation solver).

This idea is elaborated into the Predictor-multiEvaluator-and-Corrector methods $P(EC)^m E$ and $P(EC)^m$ also called Predictor-Multicorrector methods or simply Predictor-Corrector methods. These methods consist of 3 parts:

- The Predictor method which is an explicit multi step method of the form

$$\tilde{U}_i = \sum_{j=1}^{s^{ex}} \left(a_j^{ex} \tilde{U}_{i-j} + h b_j^{ex} f(x_{i-j}, \tilde{U}_{i-j}) \right), \ i = s^{ex} + 1, \ldots, n. \tag{5.1}$$

- The Evaluator method simply evaluating and assigning a function value

$$\tilde{f}_j^{(k)} = f(x_j, \tilde{U}_j^{(k)}), \ j = 1, \ldots, n. \tag{5.2}$$

- The Corrector method which is an implicit multi step method of the form

$$\tilde{U}_i - hb_0^{im} f(x_i, \tilde{U}_i) = \sum_{j=1}^{s^{im}} \left(a_j^{im} \tilde{U}_{i-j} + hb_j^{im} f(x_{i-j}, \tilde{U}_{i-j}) \right), \quad i = s^{im} + 1, \ldots, n. \tag{5.3}$$

The initial data is denoted $\tilde{U}_1^{(m)}, \ldots, \tilde{U}_{\max\{s^{ex}, s^{im}\}}^{(m)}$ and the i'th step in the algorithm for the entire $P(EC)^m E$ method is

(5.4)

For fixed $i > \max\{s^{ex}, s^{im}\}$
and given $\tilde{U}_1^{(m)}, \ldots, \tilde{U}_{i-1}^{(m)}$ and $\tilde{f}_1^{(m)}, \ldots, \tilde{f}_{i-1}^{(m)}$ do

$$[P] \ \tilde{U}_i^{(0)} = \sum_{j=1}^{s^{ex}} \left(a_j^{ex} \tilde{U}_{i-j}^{(m)} + hb_j^{ex} \tilde{f}_{i-j}^{(m)} \right)$$

for k from 0 to $m-1$ do

$$[E] \ \tilde{f}_i^{(k)} = f(x_i, \tilde{U}_i^{(k)})$$

$$[C] \ \tilde{U}_i^{(k+1)} = hb_0^{im} \tilde{f}_i^{(k)} + \sum_{j=1}^{s^{im}} \left(a_j^{im} \tilde{U}_{i-j}^{(m)} + hb_j^{im} \tilde{f}_{i-j}^{(m)} \right)$$

end do

$$[E] \ \tilde{f}_i^{(m)} = f(x_i, \tilde{U}_i^{(m)}).$$

The $P(EC)^m$ method is identical except that the last [E] step is omitted and $\tilde{f}_{i-j}^{(m)}$ is replaced by $\tilde{f}_{i-j}^{(m-1)}$ in the [P] step.

Note that the implicit method is implemented by a simple update procedure, m times inserting the latest guess for $\tilde{U}_{i+1}$. This is completely equivalent to the update part in the nonlinear equation solver in section 5.1.1.

We provide the following results without proof:

Theorem 5.1 *Let the predictor [P] and the corrector [C] have order of convergence q^{ex} and q^{im} respectively.*

- *$P(EC)^m E$ and $P(EC)^m$ have always the same order of convergence.*
- *$P(EC)^m E$ has order of convergence q^{im} if $q^{ex} \geq q^{im}$.*

- *$P(EC)^m E$ has order of convergence q^{im} if $q^{ex} < q^{im}$ and $m \geq q^{im} - q^{ex}$.*
- *$P(EC)^m E$ has order of convergence $q^{ex} + m < q^{im}$ if $q^{ex} < q^{im}$ and $m < q^{im} - q^{ex}$.*

Example 5.2 Adams-Bashford-Moulton ABM(q)
The predictor-corrector methods where the Adams-Bashford method of order q, AB(q), is used as the predictor and the Adams-Moulton method of order q, AM(q), is used as the corrector are denoted Adams-Bashford-Moulton methods of order q, ABM(q). (For the orders of AB(s) and AM(s) see theorem 4.20). ■

Exercise 5.3
Implement ABM(1) and test $P(EC)^m E$ and $P(EC)^m$ for $m = 1, 2, 3, \ldots$ on the test problem $u'(x) = u(x)$ for $x > 0$, $u(0) = 1$ with solution $u(x) = e^x$. Compare the results. ■

Exercise 5.4
Implement AB(p) and AM(q) for $p, q = 1, 2, 3$ for the test problem $u'(x) = u(x)$ for $x > 0$, $u(0) = 1$ with solution $u(x) = e^x$. Verify numerically theorem 5.1 for these cases. ■

Exercise 5.5
Write an essay on theorem 5.1 and its proof. ■

5.2 Adaptive FDM's for $u' = f(t, u)$

The term adaptivity has been mentioned a couple of times so far. Now it is time to go into more details. The most general question to consider is
Q1: What is adaptivity?
If the derivative u' of the exact solution u to (1.10) on page 16 is varying a lot in a certain subset of the computational domain I, then we will generally need small steplengths there to get the total error small. The Rule of Thumb is that

- the Local Errors in each step (being the difference between the exact solution to (1.10) on page 16 and the numerical solution after the step presuming that we start from the exact solution and advance with the chosen FDM) should be about the same in all steps in order to get the smallest Total Error at the end (being the difference $u_n - \tilde{U}_n$ starting with $\tilde{U}_1$ and advancing with the chosen FDM).

But maybe there are big subsets of I where u' is almost constant so that we could take large steps there and still keep the local error small. Adaptive Methods automatically and optimally in some sense adapt the steplength to the situation, to keep the local truncation errors almost constant in all steps. (If the optimality is missing, the method is often denoted a Feed Back Method).

Q2: What methods can be made adaptive?

Until now, we have always assumed constant step length h. If h is not constant, the order of multistep methods will generally drop, since the terms in the Taylor expansion do not cancel. See exercise 3.23 on page 84 for an example concerning δ_0, but the situation is entirely similar for δ_0^2 as seen in the following example:

Example 5.6 Order of consistency for δ_0^2 with non uniform step lengths

$$
\begin{aligned}
(5.5)\quad & u_{i+1} = u_i + h_i u_i' + \frac{h_i^2}{2}u_i'' + \frac{h_i^3}{6}u_i''' + \ldots, \quad \text{for } h_i = x_{i+1} - x_i \\
& u_{i-1} = u_i - h_{i-1}u_i' + \frac{h_{i-1}^2}{2}u_i'' - \frac{h_{i-1}^3}{6}u_i''' + - \ldots, \quad \text{for } h_{i-1} = x_i - x_{i-1} \\
& \Downarrow \text{ (requiring the coeff. of } u_i' \text{ to be 0)} \\
& h_{i-1}u_{i+1} + h_i u_{i-1} - (h_{i-1} + h_i)u_i \\
& = \left(\frac{h_{i-1}h_i^2}{2} + \frac{h_i h_{i-1}^2}{2}\right)u_i'' + \left(\frac{h_{i-1}h_i^3}{6} - \frac{h_i h_{i-1}^3}{6}\right)u'''(\xi) \\
& \Leftrightarrow u_i'' = \frac{2}{h_{i-1}h_i(h_{i-1} + h_i)} \cdot \\
& \qquad (h_{i-1}u_{i+1} + h_i u_{i-1} - (h_{i-1} + h_i)u_i) + \mathcal{O}(h_{i-1} - h_i),
\end{aligned}
$$

so only when $h_{i-1} = h_i$ (uniform step length) do we get the well known truncation error $\mathcal{O}(h^2)$ of order 2. ■

The conclusion is, that in general one step methods (like Runge-Kutta) are to be preferred as adaptive methods since here only one step length occur in each step. It should be emphasized, that when doing adaptivity with a one step method, still all theoretical results about the asymptotical behavior of the method become invalid. A method that converges of order p when uniform step lengths are used will no longer (necessarily) have order of convergence p when used for adaptivity with nonuniform step lengths, even though it is still denoted a p'th order method, tacitly understanding "if the step length had been uniform". The important fact is that the one step method will generally still behave as a p'th order convergent method whereas a multi step method will degrade in performance. It should be realized, that we have only given a weak indication of the reason for this above.

Q3: How do we do adaptivity?
We need to estimate the error e in each step. If $|e| \geq \tau$ for some given Tolerance τ, then the step length h is halved and the step is repeated. If instead $|e| < \frac{\tau}{C}$, where often $C = 2^{p+1}$ for a p'th order method, then the step length h is doubled in the following step. Otherwise h is kept unchanged for the following step.
Q4: How do we find the error?
First the bad news: In section 4.2 on page 111 we have already seen one implication of the fact that "p'th order of convergence" is an asymptotical property meaning that only for step lengths below a certain treshold h_0 can we with some right assume $e(x) \simeq C(x)h^p$. ($C(x)$ will vary over the computational domain I). The interval $]0, h_0[$ where this approximation is acceptable is called the Asymptotic Tail for the method and the notion of absolute stability is basically concerned with recovering h_0. It should be clear from the discussion in section 4.2 that h_0 will generally be unknown except for very special and simple cases. Still, to do adaptivity we make the assumption $e(x) = C(x)h^p$.

★ **For advanced readers 5.7** ★ To complicate matters even further, the e in the assumption is not the usual error $e(x_i) = e_i = u_i - \tilde{U}_i$ with $\tilde{U}_i$ being the solution to the FDM started at x_1 with the initial value $\tilde{U}_1 = u^*$. Instead we shall use $e(x_i) = e_i = u_i - \bar{U}_i$ with $\bar{U}_i$ being the solution to the FDM started at x_{i-1} with the unknown value u_{i-1}. Below, after presenting the first approach, we shall explain why this is the right approach. ★

With this somewhat troublesome assumption made, the rest is easy. We shall show two possibilities of estimating the error:

Adaptivity 1: Use a single one step, p'th order method with a number of different step lengths and the nodal points $x_1, x_2, \ldots$:

$$e_i^h := u_i - \tilde{U}_i = Ch^q \text{ (start from } \tilde{U}_{i-1} \text{ at } x_{i-1} \text{ and step } h \text{ to } x_i^*) \tag{5.6}$$
$$e_i^{2h} := u_i - \hat{U}_i = C(2h)^q \text{ (start from } \tilde{U}_{i-2} \text{ at } x_{i-2} \text{ and step } 2h \text{ to } x_i^*) \tag{5.7}$$
$$e_i^{3h} := u_i - \check{U}_i = C(3h)^q \text{ (start from } \tilde{U}_{i-3} \text{ at } x_{i-3} \text{ and step } 3h \text{ to } x_i^*) \tag{5.8}$$

Here $h = x_{i-1} - x_{i-2}$ and $x_i^* = x_{i-1} + h$ is our "candidate for x_i" to be accepted or rejected according to Q3 above. Also q is the "observed order of convergence". If we are in the asymptotic tail of the method, then q should be very close to the "theoretical order of convergence" p, but if not, q may differ significantly from p. Further, the second equality in each of (5.6), (5.7) and (5.8) is our assumption, meaning that they do not really hold exactly, but only as approximations, and for the computations we pretend that they hold. Finally, with this method we are only allowed to change step length

after two steps with the same step length, so that when we consider a change of step length then $x_{i-2} = x_{i-1} - h$ and $x_{i-3} = x_{i-1} - 2h$.

★ **For advanced readers 5.7 (Continued)** We would have liked to start in (5.6), (5.7) and (5.8) from u_{i-1}, u_{i-2} and u_{i-3} respectively, but since this would render $\tilde{U}_i$, $\hat{U}_i$ and $\check{U}_i$ non computable (not knowing the value to start from), instead we start from the best approximations $\tilde{U}_{i-1}$, $\tilde{U}_{i-2}$ and $\tilde{U}_{i-3}$ respectively. If the e in for example (5.7) had been the usual one, we should have started the method in x_1 and run it with double step lengths all the way. But doing this we would likely never hit x_i if only for the different step sizes from step to step. For this reason we need to adapt the different interpretation af e for the adaptivity computation. ★

Note that $\tilde{U}_i$, $\hat{U}_i$ and $\check{U}_i$ are computable and hence can be considered known, whereas u_i, C and q are our 3 unknowns. Solving, we get $\frac{\tilde{U}_i - \hat{U}_i}{\tilde{U}_i - \check{U}_i} = \frac{2^q-1}{3^q-1}$. Here

q	1	2	3	4
$\frac{2^q-1}{3^q-1}$	0.5	0.375	0.2692	0.1875

If $\frac{2^q-1}{3^q-1}$ is very different from $\frac{2^p-1}{3^p-1}$ it is a good indication that we are not yet in the asymptotic tail so h should be halved and the computations repeated.

If a priori we are sure that we *are* in the asymptotic tail then we can avoid (5.8) and the deliberations above and go straight to the evaluation of the error based on (5.6) and (5.7) giving $u_i - \tilde{U}_i = Ch^q = \frac{\tilde{U}_i - \hat{U}_i}{2^q-1}$. In this case we can also avoid the requirement above, that we only consider changing the step after two steps with the same step length.

Having estimated the error we now adjust the step length as explained in the answer to Q3 above, and either repeat the step or go to the next step.

The main problem with this approach to adaptivity is that it is quite expensive, requiring 3 or at least 2 function evaluations for each step. Adaptivity then at least doubles the price of the method. ■

Adaptivity 2: This method can not be used to investigate whether we are in the asymptotic tail of the method or not. This must be verified in other ways. On the other hand this method has the advantage that is is almost free computationally. The method goes as follows: Use two different RK methods with the same number of stages s and the same values of $K_{j,i}$ but with order p and $p+1$ respectively:
We need 2 RK methods with Butcher arrays with the same $\mathbf{A}$-matrix and $\mathbf{c}$-vector but with different $\mathbf{b}$-vectors giving orders of convergence p and $p+1$ respectively. This is not always possible, but there are cases where it is. The

most popular example is the Runge Kutta Fehlberg method RKF45 using 6 stage RK methods of order 4 (RK(4)) and 5 (RK(5)). RK(5) has the following Butcher array:

$$(5.9)\quad \begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^T \end{array} = \begin{array}{c|c} \begin{bmatrix} 0 \\ \frac{1}{4} \\ \frac{3}{8} \\ \frac{12}{13} \\ 1 \\ \frac{1}{2} \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ \frac{3}{32} & \frac{9}{32} & 0 & 0 & 0 & 0 \\ \frac{1932}{2197} & -\frac{7200}{2197} & \frac{7296}{2197} & 0 & 0 & 0 \\ \frac{439}{216} & -8 & \frac{3680}{513} & -\frac{845}{4104} & 0 & 0 \\ -\frac{8}{27} & 2 & -\frac{3544}{2565} & \frac{1859}{4104} & -\frac{11}{40} & 0 \end{bmatrix} \\ \hline & \begin{bmatrix} \frac{16}{135} & 0 & \frac{6656}{12825} & \frac{28561}{56430} & -\frac{9}{50} & \frac{2}{55} \end{bmatrix} \end{array}$$

RK(4) has the same $\mathbf{c}$ and $\mathbf{A}$ and a different $\mathbf{b}$. Since we only need the difference between RK(5) and RK(4) for the adaptivity, we shall give $\mathbf{b}^T_{RK(5)} - \mathbf{b}^T_{RK(4)} = \begin{bmatrix} \frac{1}{360} & 0 & -\frac{128}{4275} & -\frac{2197}{75240} & \frac{1}{50} & \frac{2}{55} \end{bmatrix}$. Details can be found for example in [4] §11.8 that we also referred to in section 4.5 for details about the RK methods in general ■ .

Assuming that we *are* in the asymptotic tail, (5.6) and (5.7) are now replaced with

$$(5.10)\quad \text{RK(p+1): } e_i := u_i - \tilde{U}_i = Ch^{p+1}$$
(start from $\tilde{U}_{i-1}$ at x_{i-1} and step h to x_i)

$$(5.11)\quad \text{RK(p): } \hat{e}_i := u_i - \hat{U}_i = \hat{C}h^{p}$$
(start from $\tilde{U}_{i-1}$ at x_{i-1} and step h to x_i)

★ **For advanced readers 5.7 (Continued)** We would have liked to start from u_{i-1} but since this would render $\tilde{U}_i$ and $\hat{U}_i$ non computable (not knowing the value to start from), instead we start from the best approximation $\tilde{U}_{i-1}$. ★

Now $u_i - \hat{U}_i = (u_i - \tilde{U}_i) + (\tilde{U}_i - \hat{U}_i) = \hat{C}h^p$. Since the first paranthesis after the first equality sign is $\mathcal{O}(h^{p+1})$ and the sum of the two parantheses is $\mathcal{O}(h^p)$, the second paranthesis must be $\hat{C}h^p + \mathcal{O}(h^{p+1})$ so that $\hat{E}_i = \tilde{U}_i - \hat{U}_i$ can serve as our estimate of the error. Then for the $(i-1)$'st step we first compute the relevant $\tilde{K}_{j,i-1}$ (depending on $\tilde{U}_{i-1}$) and then we compute $\tilde{U}_i$ using RK(p+1):

$$\tilde{U}_i = \tilde{U}_{i-1} + h_{i-1} \sum_{j=1}^{s} b_j^{RK(p+1)} \tilde{K}_{j,i-1}.$$

Then to compute the error, note that $\hat{U}_i$ is computed using RK(p) but starting in $\tilde{U}_{i-1}$ computed with RK(p+1):

$$\hat{U}_i = \tilde{U}_{i-1} + h_{i-1} \sum_{j=1}^{s} b_j^{RK(p)} \tilde{K}_{j,i-1}.$$

Subtracting the two our approximation to the error is simply

$$\hat{E}_i = \tilde{U}_i - \hat{U}_i = h_{i-1} \sum_{j=1}^{s} (b_j^{RK(p+1)} - b_j^{RK(p)}) \tilde{K}_{j,i-1}.$$

Note that $\hat{E}_i$ is an approximation to $\hat{e}_i$, the error in the p'th order method. So in reality we should accept or reject $\hat{U}_i$ when $\hat{E}_i < \tau$ or $\hat{E}_i > \tau$ respectively. But we know that $\tilde{U}_i$ is more accurate than $\hat{U}_i$, so why not accept or reject this one instead. (We may reject some $\tilde{U}_i$'s that are really good enough, but the price of computing an approximation to e_i is much bigger (requiring a $p + 2$ order method, so we accept this situation).

Note finally that no additional function evaluations are needed when using adaptivity method 2, only some extra algebra. Hence this implementation of adaptivity is very inexpensive, although it implicates a certain overhead in programming effort. ■

Exercise 5.8
Implement RKF45 and test it on the test problem $u'(x) = u(x)$ for $x > 0$, $u(0) = 1$ with solution $u(x) = e^x$. Start in $x = 0$ with step length $h = 1.0$ and use tolerance $T = 0.0001$. Double the step length if $|e| < T/16$ and halve the step length if $|e| \geq T$. Step until you reach $x = 5$ or 100 steps, whatever comes first. Plot the exact and the numerical solution together. ■

5.3 Systems $\boldsymbol{u}' = \boldsymbol{f}(t, \boldsymbol{u})$ and absolute stability. Stiff systems

All the theory presented above for the scalar ODE (1.10) can be repeated for systems of ODE's of the form

$$\boldsymbol{u}' = \boldsymbol{f}(t, \boldsymbol{u}),\ t > t_0,\ \boldsymbol{u}(t_0) = \boldsymbol{u}_0, \tag{5.12}$$

which is well-posed if $\boldsymbol{f}$ is Lipschitz continuous in its second (vector) argument. A little bit of "flexibility" may be required at times as we demonstrate here: Consider the following system of linear ODE's with constant coefficients

$$\boldsymbol{u}'(t) = \boldsymbol{A}\boldsymbol{u}(t),\ t > t_0,\ \boldsymbol{u}(t_0) = \boldsymbol{u}_0, \tag{5.13}$$

where $\boldsymbol{u} \in \mathcal{R}^n$ and where we shall assume that $\boldsymbol{A} \in \mathcal{R}^{n\times n}$ has n distinct eigenvalues $\lambda_1, \ldots, \lambda_n$. As a consequence $\boldsymbol{A}$ also has n linearly independent eigenvectors $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n$ (see for instance [17], Chapter 6, Lemma 3.1, p.276).

In the scalar case $n = 1$, (5.13) has the solution

$$u(t) = e^{A(t-t_0)}u_0 \tag{5.14}$$

and the natural generalisation would then be to expect a solution to the general case in the form

$$\boldsymbol{u}(t) = e^{\boldsymbol{A}(t-t_0)}\boldsymbol{u}_0. \tag{5.15}$$

For the right hand side to even make sense, we need to define the exponential of a matrix. We will do this below, and also show that (5.15) actually *is* the solution. Note that in the scalar case $e^{A(t-t_0)}u_0 = u_0e^{A(t-t_0)}$. This commutability does not hold in general in the matrix case.

In the following we shall identify an n-vector with a one column matrix and its transpose with a one row matrix. For example $\boldsymbol{u} \in \mathcal{R}^n = \mathcal{R}^{n\times 1}$ and $\boldsymbol{u}^T \in \mathcal{R}^n = \mathcal{R}^{1\times n}$. Then we have

$$\boldsymbol{A}\boldsymbol{q}_j = \lambda_j\boldsymbol{q}_j, \text{ for } j = 1, \ldots, n \text{ and } \boldsymbol{Q} = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n] \text{ is invertible}$$

$$\Downarrow$$

$$\boldsymbol{A}\boldsymbol{Q} = \boldsymbol{Q}\boldsymbol{\Lambda}, \text{ where } \boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix}$$

$$\Downarrow$$

$$\boldsymbol{A} = \boldsymbol{A}\boldsymbol{Q}\boldsymbol{Q}^{-1} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1}. \tag{5.16}$$

Let $\boldsymbol{z}(t) = \boldsymbol{Q}^{-1}\boldsymbol{u}(t)$ so that

$$\boldsymbol{z}'(t) = \boldsymbol{Q}^{-1}\boldsymbol{u}'(t) = \boldsymbol{Q}^{-1}\boldsymbol{A}\boldsymbol{u}(t) = \boldsymbol{\Lambda}\boldsymbol{Q}^{-1}\boldsymbol{u}(t) = \boldsymbol{\Lambda}\boldsymbol{z}(t), \text{ for } t > t_0 \text{ and}$$
$$\boldsymbol{z}(t_0) = \boldsymbol{Q}^{-1}\boldsymbol{u}(t_0) = \boldsymbol{Q}^{-1}\boldsymbol{u}_0$$

$$\Updownarrow$$

$$z_j'(t) = \lambda_j z_j(t), \text{ for } t > t_0, \; z_j(t_0) = (\boldsymbol{Q}^{-1}\boldsymbol{u}_0)_j, \; j = 1, \ldots, n$$

$$\Updownarrow$$

$$z_j(t) = e^{\lambda_j(t-t_0)}(\boldsymbol{Q}^{-1}\boldsymbol{u}_0)_j$$

$$\Updownarrow$$

$$\boldsymbol{z}(t) = \boldsymbol{D}(t)\boldsymbol{Q}^{-1}\boldsymbol{u}_0 \text{ where } \boldsymbol{D}(t) = \begin{bmatrix} e^{\lambda_1(t-t_0)} & & 0 \\ & \ddots & \\ 0 & & e^{\lambda_n(t-t_0)} \end{bmatrix}$$

$$\Updownarrow$$

$$\boldsymbol{u}(t) = \boldsymbol{Q}\boldsymbol{z}(t) = \boldsymbol{Q}\boldsymbol{D}(t)\boldsymbol{Q}^{-1}\boldsymbol{u}_0. \tag{5.17}$$

To investigate the connection between (5.15) and (5.17) we define the exponential of a matrix by generalizing the series expansion holding for the scalar case to the matrix case, i.e.

$$e^{\boldsymbol{A}} = \sum_{k=0}^{\infty} \frac{\boldsymbol{A}^k}{k!}. \tag{5.18}$$

Then by (5.16)

$$\begin{aligned} e^{\boldsymbol{A}(t-t_0)} &= \sum_{k=0}^{\infty} \frac{\boldsymbol{A}^k (t-t_0)^k}{k!} \\ &= \sum_{k=0}^{\infty} \frac{\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1} \cdot \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1} \cdot \ldots \cdot \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1} (t-t_0)^k}{k!} \\ &= \sum_{k=0}^{\infty} \frac{\boldsymbol{Q}\boldsymbol{\Lambda}^k\boldsymbol{Q}^{-1} (t-t_0)^k}{k!} \\ &= \boldsymbol{Q} \left(\sum_{k=0}^{\infty} \frac{\boldsymbol{\Lambda}^k (t-t_0)^k}{k!} \right) \boldsymbol{Q}^{-1} \\ &= \boldsymbol{Q} \begin{bmatrix} \sum_{k=0}^{\infty} \frac{(\lambda_1 (t-t_0))^k}{k!} & & 0 \\ & \ddots & \\ 0 & & \sum_{k=0}^{\infty} \frac{(\lambda_n (t-t_0))^k}{k!} \end{bmatrix} \boldsymbol{Q}^{-1} \\ &= \boldsymbol{Q}\boldsymbol{D}(t)\boldsymbol{Q}^{-1}. \end{aligned} \tag{5.19}$$

(5.17) and (5.19) now verifies (5.15).

Let us conclude this subject with a few comments on the matrix exponential: To compute the exponential of a matrix we have the series definition (5.18). A more convenient expression comes from (5.19) when the eigenvalues and eigenvectors are known. Then

$$e^{\boldsymbol{A}} = \sum_{k=0}^{\infty} \frac{\boldsymbol{A}^k}{k!} = \boldsymbol{Q}\boldsymbol{D}_0\boldsymbol{Q}^{-1} \text{ where } \boldsymbol{D}_0 = \begin{bmatrix} e^{\lambda_1} & & 0 \\ & \ddots & \\ 0 & & e^{\lambda_n} \end{bmatrix}. \tag{5.20}$$

The matrix exponential has some but not all of the properties of the scalar exponential function. We leave some of these properties to the reader in the following exercise.

Exercise 5.9

Show the following properties of the exponential of a matrix for appropriate matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ and scalars s and t:

1. $e^{s\boldsymbol{A}}e^{t\boldsymbol{A}} = e^{(s+t)\boldsymbol{A}}$
2. $e^{\boldsymbol{A}}e^{-\boldsymbol{A}} = e^{0\boldsymbol{A}} = \boldsymbol{I}$
3. $(e^{\boldsymbol{A}})^k = e^{k\boldsymbol{A}}$ for $k = \{0, \pm 1, \pm 2, \ldots\}$
4. $\boldsymbol{AB} = \boldsymbol{BA} \Rightarrow e^{\boldsymbol{A}+\boldsymbol{B}} = e^{\boldsymbol{A}}e^{\boldsymbol{B}}$.

■

For the theory presented above for the scalar ODE (1.10) only when generalizing the notion of absolute stability some care must be taken:

Example 5.10 Absolute stability for systems of ODE's

$$\mathbf{u}' = \mathbf{A}\mathbf{u},\ t > 0,\ \mathbf{u}(0) = \mathbf{1}, \tag{5.21}$$

where $\mathbf{A} \in \mathcal{R}^{n\times n}$ has n distinct eigenvalues $\lambda_1, \ldots, \lambda_n$ i.e. $\exists \mathbf{Q}:\ \mathbf{\Lambda} = \mathbf{Q}^{-1}\mathbf{A}\mathbf{Q}$ where $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_n)$. Let $\mathbf{z} = \mathbf{Q}^{-1}\mathbf{u}$ so that $\mathbf{z}'(t) = \mathbf{\Lambda}\mathbf{z}(t)$, i.e. $\mathbf{z}(t) = \sum_{j=1}^{n} c_j e^{\lambda_j t}\mathbf{e}_j$ where the j'th unit vector $\mathbf{e}_j$ is an eigenvector for $\mathbf{\Lambda}$ corresponding to λ_j. Hence $\mathbf{z} \to \mathbf{0}$ as $t \to \infty$ iff $\mathcal{R}e(\lambda_j) < 0,\ j = 1, \ldots, n$. ($\mathcal{R}e(z)$ is the real part of z). ■

The general idea is to diagonalize and hence split the system of dependent ODE's into a system of independent ODE's and then treat each independent scalar ODE by the methods introduced above.

Example 5.11 Stiff systems of ODE's

$$\mathbf{u}' = \mathbf{A}\mathbf{u} + \mathbf{g}(t),\ t > 0,\ \mathbf{u}(0) = \mathbf{1}, \tag{5.22}$$

with the same $\mathbf{A}$ as in example 5.10 having $\mathcal{R}e(\lambda_j) < 0,\ j = 1, \ldots, n$. The soluton to (5.22) can be written in the following form

$$\mathbf{u}(t) = \sum_{j=1}^{n} C_j e^{\lambda_j t}\mathbf{q}_j + \mathbf{G}(t) \to \mathbf{G}(\infty) \text{ as } t \to \infty \tag{5.23}$$

where $\mathbf{q}_j = \mathbf{Q}\mathbf{e}_j$ is an eigenvector for $\mathbf{A}$ corresponding to λ_j and $\mathbf{G}$ is an arbitrary solution to the ODE without considering the boundary conditions. $\sum_{j=1}^{n} C_j e^{\lambda_j t}\mathbf{q}_j$ is called the Transient Solution since it dies out as $t \to \infty$. $\mathbf{G}$ is called the Steady State Solution since it may stay on as $t \to \infty$.
If the region of absolute stability is bounded, there will be a condition on h uniformly for all t, since absolute stability is about what happens as $t \to \infty$ with h fixed, making h smaller the larger the biggest absolute eigenvalue

$|\lambda_{\max}|$ is. But the bigger $|\lambda_{\max}|$ is, the faster the transient dies out. Hence a fast dying transient initially forces a small step length which must be maintained at all times, also when the transient is long time gone. The system of ODE's is said to be Stiff when in this way small step lengths are necessary also in regions where the solution is very smooth. The solution to the problem is generally to use implicit methods with unbounded regions of absolute stability. ■

Note that we have given only a very rudimentary and incomplete introduction to the notion of stiff systems of ODE's. A thorough treatment would require an entire course all by itself.

Exercise 5.12
Write an essay on stiff problems and implement some methods for the solution of test cases. ■

Chapter 6

Second order problems

6.1 Linear, scalar, one dimensional, second order, boundary value problem

So far, we have been considering the first order quasi linear DEP (1.10). In this and the following sections we change focus to second order DEP's. Since these are somewhat more complicated than the first order DEP's we restrict to the linear case. As we have done until now, we still consider only one dimensional equations, i.e. ODE's. In general second order ODE's require two additional conditions in order to have uniqueness of solution. The main theoretical result corresponding to theorem 1.16 on page 16 is the following:

Definition 6.1 *The general expression for a Linear, Scalar, One Dimensional, Second Order, Boundary Value Problem can be written as*

$$\begin{aligned} (6.1)\quad & \text{Find } u \in \mathcal{C}^2(\bar{I}): \ a(t)u''(t) + b(t)u'(t) + c(t)u(t) + d(t) = 0 \ \forall t \in I, \\ & b_1 u'(t_1) + c_1 u(t_1) + d_1 = 0 \text{ and } b_2 u'(t_2) + c_2 u(t_2) + d_2 = 0, \end{aligned}$$

where a, b, c and d are known functions, b_1, b_2, c_1, c_2, d_1 and d_2 are known constants and I is a known real interval containing the two points t_1 and t_2. t_1 and t_2 may or may not be identical. If $t_1 \neq t_2$ we denote (6.1) a Two Point Boundary Value Problem. If instead $t_1 = t_2$ (6.1) is an Initial Value Problem or a One Point Boundary Value Problem. In any case the generic name Boundary Value Problem is used for all situations.

Theorem 6.2 *If in (6.1) $a, b, c, d \in \mathcal{C}^0(\bar{I})$, $a(t) \neq 0$, $\forall t \in I$, if $(c_1, c_2) \neq (0, 0)$ and if, when in the special case $t_1 = t_2$, then the two boundary conditions are consistent and independent i.e. can be transformed into equivalent conditions $u(t_1) = \tilde{d}_1$ and $u'(t_1) = \tilde{d}_2$ then (6.1) is well-posed.*

For a proof, see for example [3] ch. 2 and 6. The relevant Lipschitz condition is automatically satisfied by the linearity of the system. IVP's are generally transformed into systems of first order ODE's and solved with the methods discussed up until now, so we shall concentrate on two point BVP's without completely dismissing the IVP's. Here we shall consider only two special cases, which are also examples that we shall follow through this and the next couple of sections:

Example 6.3 The Poisson problem in 1 dimension
Consider the special case of (6.1) which takes the form

$$\text{(6.2)} \qquad \text{Find } u \in \mathcal{C}^2([0,1]): \ -u''(t) = f(t) \ \forall t \in]0,1[, \ u(0) = u(1) = 0,$$

for a known function $f \in \mathcal{C}^0([0,1])$. Note that we have met this problem before in (3.1) on page 59 except for the different boundary conditions. By the fundamental theorem of calculus, we may integrate the ODE in (6.2) twice. Taking $F(s) = \int_0^s f(t)dt$ this results in $u(t) = \alpha t + \beta - \int_0^t F(s)ds$ for two arbitrary constants α and β which can be determined by the boundary conditions $u(0) = u(1) = 0$ to be $\beta = 0$ and $\alpha = \int_0^1 F(s)ds$. Integrating by parts $\int_0^t F(s)ds = [sF(s)]_0^t - \int_0^t sf(s)ds = \int_0^t (t-s)f(s)ds$ so that $u(t) = t\int_0^1 (1-s)f(s)ds - \int_0^t (t-s)f(s)ds$ i.e.

$$\text{(6.3)} \quad u(t) = \int_0^1 G(t,s)f(s)ds, \ G(t,s) = \begin{cases} s(1-t) & \text{if } 0 \le s \le t, \\ t(1-s) & \text{if } t \le s \le 1. \end{cases}$$

G is called the Greens Function for (6.2) and various plots of G are shown in figure 6.1 below. With (6.3) we have existence and uniqueness of solution for the Poisson problem (6.2) but (6.3) also imply other important properties: Since $G(t,s) \ge 0$ it is clear that we have the following

$$\text{(6.4)} \quad \text{Monotonicity Property: } f(t) \ge 0 \ \forall t \in [0,1] \Rightarrow u(t) \ge 0 \ \forall t \in [0,1].$$

Also $|u(t)| \le \|f\|_\infty \int_0^1 G(t,s)ds = \frac{1}{2}t(1-t)\|f\|_\infty \le \frac{1}{8}\|f\|_\infty$ so we have the following

$$\text{(6.5)} \qquad \text{Maximum Principle: } \|u\|_\infty \le \frac{1}{8}\|f\|_\infty.$$

Note, that the maximum principle is just continuous dependence on the data so that we have well-posedness with respect to perturbations in f. ■

The second special case example that is met many places in real life problems (and in the literature), is the following:

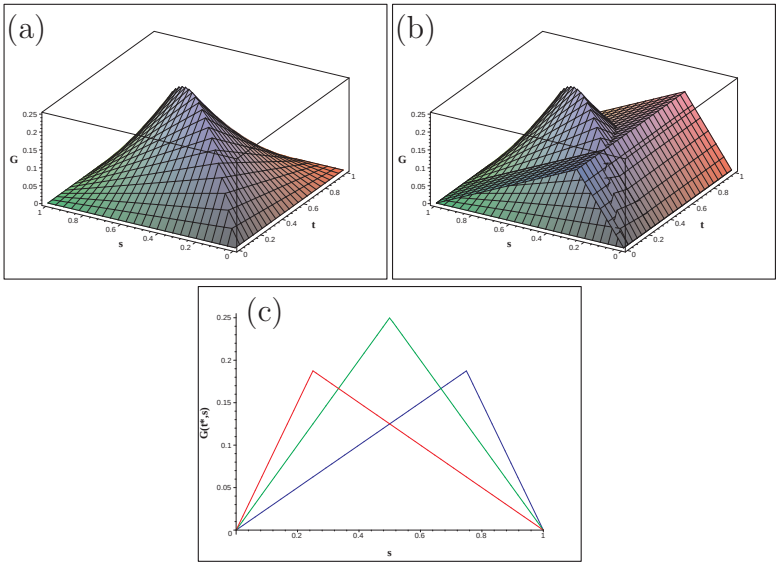


Figure 6.1: Greens function $G(t,s)$ for the Poisson problem in one dimension: (a) 3D-plot of $G(t,s)$, (b) 3D plots of $G(t,s)$ and $G(0.25,s)$ and (c) 2D-plot of $G(0.25,s)$ (red, leftmost curve) $G(0.5,s)$ (green, center curve) and $G(0.75,s)$ (blue, rightmost curve).

Example 6.4 The Divergence or self adjoint form problem in 1 dimension
Here we consider the special case of (6.1) where $b = a'$. Without loss of generality we restrict to $I =]0,1[$, and for simplicity we also take $t_1 = 0$, $t_2 = 1$:

$$\text{Find } u \in \mathcal{C}^2([0,1]): \ -(a(t)u'(t))' - c(t)u(t) = d(t) \ \forall t \in]0,1[, \tag{6.6}$$
$$b_1u'(0) + c_1u(0) + d_1 = 0 \text{ and } b_2u'(1) + c_2u(1) + d_2 = 0.$$

We shall not discuss this example in details but just mention, that many practical problems are naturally formulated in this way. ■

6.2 Difference operators for FDM's, revisited

In section 3.2 starting on page 77 we presented the Taylor series methods for construction of difference operators, and used them to construct a few standard difference operators. These were sufficient until now, but when considering second order ODE's we need a somewhat deeper knowledge.

For the first order ODE's considered so far, we have presented both first and second order consistent methods. Instead for second derivatives we have

only proposed the second order difference approximation δ_0^2 in example 3.21 on page 83. We start answering the question about why exactly this difference operator was presented as the standard for approximating second derivatives.

Example 6.5 What consistency order to use for second order ODE's
$\delta_0^2 f$ uses 3 nodal points, so let us try to obtain a first order consistent method with just 2 points, which we without loss of generality can take to be $z+h$ and $z+c$. Using the Taylor series method (with c_1, c_2 and c depending on h) we have

$$\begin{aligned} & c_1 f(z+h) + c_2 f(z+c) = f''(z) + \mathcal{O}(h) \Leftrightarrow \\ & c_1(f(z) + hf'(z) + \frac{h^2}{2}f''(z) + \ldots) + c_2(f(z) + cf'(z) \\ & + \frac{c^2}{2}f''(z) + \ldots) - f''(z) = \mathcal{O}(h) \end{aligned} \tag{6.7}$$

from which it is clear that we need to take $c_1 + c_2 = \mathcal{O}(h)$, $c_1 h + c_2 c = \mathcal{O}(h)$ and $c_1\frac{h^2}{2} + c_2\frac{c^2}{2} - 1 = \mathcal{O}(h)$ to get rid of the $f(z)$, the $f'(z)$ and the $f''(z)$-term respectively To simplify the solution process, we shall settle for linear approximations to these equations in the form $c_1 + c_2 = \alpha h$, $c_1 h + c_2 c = \beta h$ and $c_1\frac{h^2}{2} + c_2\frac{c^2}{2} - 1 = \gamma h$. Eliminating c_2 from the last two equations using the first, leads to $c_1(h-c) = -\alpha hc + \beta h$ and $\frac{1}{2}c_1(h-c)(h+c) = -\alpha h\frac{c^2}{2} + 1 + \gamma h$. Now eliminating c_1 from the last equation using the first leads to $c = \frac{2(1+\gamma h)}{\beta h - \alpha h^2} = \mathcal{O}(h^{-1})$. Plugging this into the first equation gives $c_1 = \frac{\beta h - \alpha hc}{h-c} = \frac{\alpha h + (\alpha\gamma - \frac{\beta^2}{2})h^2 + \frac{\alpha\beta}{2}h^3}{1+\gamma h - \frac{\beta}{2}h^2 + \frac{\alpha}{2}h^3} = \mathcal{O}(h)$ and finally $c_2 = -c_1 + \alpha h = \frac{\frac{\beta^2}{2}h^2 + \frac{\alpha^2}{2}h^4}{1+\gamma h - \frac{\beta}{2}h^2 + \frac{\alpha}{2}h^3} = \mathcal{O}(h^2)$. Hence we get the formula

$$\begin{aligned} & \frac{1}{1+\gamma h - \frac{\beta}{2}h^2 + \frac{\alpha}{2}h^3}\left\{\left(\alpha h + (\alpha\gamma - \frac{\beta^2}{2})h^2 + \frac{\alpha\beta}{2}h^3\right) f(z+h)\right. \\ & \left. + \left(\frac{\beta^2}{2}h^2 + \frac{\alpha^2}{2}h^4\right) f(z + \frac{2(1+\gamma h)}{\beta h - \alpha h^2})\right\} = f''(z) + \mathcal{O}(h). \end{aligned} \tag{6.8}$$

This formula has many disadvantages: First of all it is unsymmetric and further the second nodal point is ackwardly moving away towards $\pm\infty$ (depending on the sign of β) as $h \to 0$. Generally, no reuse of function evaluations will be possible with this type of formula. For this reason, 2 point formulas are almost never used for approximating second derivatives, and while first order 3 point formulas can be constructed like for example

$$c_1 f(z-h) - \frac{2}{h^2}f(z) + \left(\frac{2}{h^2} - c_1\right) f(z+h) = f''(z) + \mathcal{O}(h) \tag{6.9}$$

this approach makes little sense since we already know that taking $c_1 = \frac{1}{h^2}$ we obtain the symmetric second order $\delta_0^2 f$. Hence we really have to bend over to get first order approximations to second derivatives and such are basically never used. For this reason and to limit the number of function evaluations, we shall concentrate on second order approximations to second order derivatives. ■

To avoid degradation of the truncation errors for (6.1) we shall then also for lower order derivatives and for boundary conditions (see below in example 6.8 on page 168) be using second order difference operators. This can be done at no extra cost as long as the nodes $z-h$, z and $z+h$ can be used as for example in $\delta_0 f$.

Note that the choice of (at least) second order consistent, (a least) 3 point difference approximations to the second derivative in (6.1) has as a consequence that the FDM's for (6.1) become (at least) 2 step with the problems this implies for adaptivity (see section 5.2 on page 152 and in particular the subsection "Q2"). Also, since there will be (at least) 3 unknown $\tilde{U}$ values in each equation for the FDM (typically $\tilde{U}_{i-1}$, $\tilde{U}_i$ and $\tilde{U}_{i+1}$), except for the boundary conditions, in general the FDM's for two point BVP's will be implicit. The only way to make explicit methods would be to have what corresponds to a "priming the pumps" process as described in section 4.3 on page 115 for the multistep methods, i.e. to start out with $\tilde{U}_1$ and $\tilde{U}_2$ given. This is exactly what happens for initial value problems like

$$\text{(6.10)} \quad \text{Find } u \in \mathcal{C}^2([0,1]): \ -u''(t) = f(t) \ \forall t \in]0,1[, \ u(0) = u'(0) = 0,$$

for which a typical FDM would be

$$\text{(6.11)} \quad -\delta_0^2 \tilde{U}_i = -\frac{\tilde{U}_{i-1} - 2\tilde{U}_i + \tilde{U}_{i+1}}{h^2} = f_i, \ i = 2, \ldots, n-1, \ \tilde{U}_1 = \tilde{U}_2 = 0,$$

where $\tilde{U}_2 = 0$ comes from the (only first order) approximation $\delta_+ \tilde{U}_1 = 0$ of the initial condition $u'(0) = 0$ (and using also $\tilde{U}_1 = 0$). $\tilde{U}_3$ is then recovered from $\tilde{U}_1$ and $\tilde{U}_2$, $\tilde{U}_4$ from $\tilde{U}_2$ and $\tilde{U}_3$ and so on, following the *arrow of time* as was done for all the explicit and implicit first order DEP's met so far.

For the corresponding two point BVP with $u(0) = u(1) = 0$, instead we have $\tilde{U}_1 = \tilde{U}_n = 0$ and the situation changes:
Taking $i = 2$, we get an equation involving $\tilde{U}_1$, $\tilde{U}_2$ and $\tilde{U}_3$ of which only $\tilde{U}_1$ is known. We can hence *not* recover $\tilde{U}_2$ and follow the arrow of time. Instead

we may write all the equations in matrix form as

$$(6.12)\qquad \begin{bmatrix} 2 & -1 & & & \mathbf{0} \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ \mathbf{0} & & & -1 & 2 \end{bmatrix} \begin{bmatrix} \tilde{U}_2 \\ \tilde{U}_3 \\ \vdots \\ \tilde{U}_{n-2} \\ \tilde{U}_{n-1} \end{bmatrix} = h^2 \begin{bmatrix} f_2 \\ f_3 \\ \vdots \\ f_{n-2} \\ f_{n-1} \end{bmatrix}.$$

When this equation system is solved, all of the $\tilde{U}_i$ are recovered at the same time, so instead of following the arrow of time, where $\tilde{U}_i$ depends only on $\tilde{U}_j$ for $j < i$, in the present case of two point BVP's, all the $\tilde{U}_i$'s are depending on each other. To distinguish from the implicit methods following the arrow of time, we may refer to those as Locally Implicit methods while the methods for the two point BVP's may be called Globally Implicit methods.
The divergence form (6.6) presents yet another problem:

Example 6.6 How to approximate terms like $(a(z)u'(z))'$
The simple minded solution is to expand the derivative and do

$$\begin{aligned}(6.13)\quad (a(z)u'(z))' &= a'(z)u'(z) + a(z)u''(z) \\ &= \delta_0 a(z)\delta_0 u(z) + a(z)\delta_0^2 u(z) + \mathcal{O}(h^2) \\ &= \frac{a(z+h)-a(z-h)}{2h}\frac{u(z+h)-u(z-h)}{2h} \\ &\quad + a(z)\frac{u(z+h)-2u(z)+u(z-h)}{h^2} + \mathcal{O}(h^2).\end{aligned}$$

Instead it turns out that we may get a much simpler solution by using the "half points" first introduced for the Crank-Nicolson method for (1.10) in example 3.37 on page 98. Defining the First Order Central Half Point Difference Operator of Consistency Order Two by

$$(6.14)\qquad \delta_{0,\frac{1}{2}} f(z) = \frac{f(z+\frac{h}{2}) - f(z-\frac{h}{2})}{h} = f'(z) + \mathcal{O}(h^2)$$

with the order easily verifiable by Taylor expansion, we get

$$\begin{aligned}(6.15)\quad \delta_{0,\frac{1}{2}}\left(a(z)\delta_{0,\frac{1}{2}}u(z)\right) &= \delta_{0,\frac{1}{2}}\left(a(z)\frac{u(z+\frac{h}{2})-u(z-\frac{h}{2})}{h}\right) \\ &= \frac{a(z+\frac{h}{2})\frac{u(z+h)-u(z)}{h} - a(z-\frac{h}{2})\frac{u(z)-u(z-h)}{h}}{h} \\ &= \frac{a(z+\frac{h}{2})u(z+h) - (a(z+\frac{h}{2})+a(z-\frac{h}{2}))u(z) + a(z-\frac{h}{2})u(z-h)}{h^2} \\ &= (a(z)u(z)')' + \mathcal{O}(h^2)\end{aligned}$$

where the last equality comes from the usual Taylor expansions. Note that the function u is still only evaluated in the "usual" nodal points (which is what we want), while only a is evaluated in the half points, and one of the values can be reused if also $(a(z+h)u(z+h)')'$ needs to be evaluated. ■

Exercise 6.7
Verify the order of consistency in (6.14) and (6.15). ■

Finally there is the question about how to evaluate boundary conditions involving derivatives. We shall consider only one example explaining a general approach:

Example 6.8 How to approximate boundary conditions like $a(1)u'(1)$
The problem with boundary conditions is that the symmetric second order formulas that we are using will reach outside the domain. The simple minded approach would be to use the Taylor method to construct a method based on the points 1, $1-h$ and $1-2h$. (Since we can not use symmetric points, it is not possible to construct a second order approximation based on just 2 points). We easily arrive at

$$a(1)\frac{3u(1)-4u(1-h)+u(1-2h)}{2h} = a(1)u'(1)+\mathcal{O}(h^2). \tag{6.16}$$

Again as in example 6.6 it is possible to achieve a simpler result using a more involved approach. The first step in this approach is to use a symmetric half point formula in the form of an average, forming

$$\begin{aligned}&\frac{1}{2}\left(a(1-\frac{h}{2})\delta_{0,\frac{1}{2}}(u(1-\frac{h}{2}))+a(1+\frac{h}{2})\delta_{0,\frac{1}{2}}(u(1+\frac{h}{2}))\right)\\&= a(1)u'(1)+\mathcal{O}(h^2),\end{aligned} \tag{6.17}$$

where the right hand is easily verified by Taylor expansion of the left hand side and using (6.14). The problem is that the left hand side involves the Ghost Nodes $1+\frac{h}{2}$ and $1+h$ that lie outside the computational domain $[0,1]$. Now if the ODE inside the domain is the divergence form problem from (6.6), i.e. $(a(t)u'(t))'+c(t)u(t)+d(t)=0$, then we have by (6.15) (extending also the ODE outside the domain) that

$$\begin{aligned}&\frac{1}{h}\left(a(1+\frac{h}{2})\delta_{0,\frac{1}{2}}(u(1+\frac{h}{2}))-a(1-\frac{h}{2})\delta_{0,\frac{1}{2}}(u(1-\frac{h}{2}))\right)\\&+c(1)u(1)+d(1)=\mathcal{O}(h^2).\end{aligned} \tag{6.18}$$

Eliminating the ghost nodes in (6.17) using (6.18) we arrive at

$$a(1-\frac{h}{2})\delta_{0,\frac{1}{2}}(u(1-\frac{h}{2}))-\frac{h}{2}(c(1)u(1)+d(1)) \tag{6.19}$$
$$=a(1)u'(1)+\mathcal{O}(h^2). \quad \blacksquare$$

Exercise 6.9
Verify the order of consistency in (6.17), (6.18) and (6.19). ■

6.3 Convergence for FDM's for $u''=f$

FDM's for (6.1) (or for (6.2) or (6.6)) can be constructed in the usual way as described in section 3.1.1 on page 60: Just replace all derivatives with finite differences, taking care to use difference operators with the same order of consistency everywhere (or being aware that otherwise, the lowest order is determinating for the error), and using the simplifying methods of section 6.2 or similar or more advanced methods when necessary.

Example 6.10 Second order consistent FDM for (6.2)
The standard FDM for the 1 dimensional Poisson problem is the following globally implicit, 2 step method:

$$-\delta_0^2\tilde{U}_i=-\frac{\tilde{U}_{i-1}-2\tilde{U}_i+\tilde{U}_{i+1}}{h^2}=f_i,\ i=2,\ldots,n-1,\ \tilde{U}_1=\tilde{U}_n=0, \tag{6.20}$$

for $n\geq 2$ uniformly distributed nodes $x_i=(i-1)h$, $i=1,\ldots,n$ where $h=\frac{1}{n-1}$ and with $f_i=f(x_i)$, $\forall i=1,\ldots,n$.

Theorem 6.11 (6.20) *is well-posed and has the unique solution*

$$\tilde{U}_i=\sum_{j=1}^{n}G_j(x_i)f(x_j),\ \forall i=1,\ldots,n. \tag{6.21}$$

where the Discrete Greens Function is given by

$$G_j(t)=hG(t,x_j),\ j=1,\ldots,n \tag{6.22}$$

with G being the Greens function introduced in (6.3).

Proof
First we show continuous dependence on the data f and existence and uniqueness of solution:
(6.20) can also be written in the form of the tridiagonal equation system $\mathbf{A}^*\tilde{\mathbf{U}}^* = h^2\mathbf{f}^*$ where $\mathbf{A}^* = \text{tridiag}(-1, 2, -1) \in \mathcal{R}^{(n-2)\times(n-2)}$,
$\tilde{\mathbf{U}}^* = (\tilde{U}_2, \ldots, \tilde{U}_{n-1})^T$ and $\mathbf{f}^* = (f_2, \ldots, f_{n-1})^T$ (see (6.12)). Solving this equation system only $\tilde{U}_2, \ldots, \tilde{U}_{n-1}$ are recovered, while $\tilde{U}_1 = \tilde{U}_n = 0$ is understood.
$\mathbf{A}^*$ is Diagonally Dominant since $2 \geq |-1| + |-1|$.
$\mathbf{A}^*$ is also Positive Definite since for any $\mathbf{v}^* = (v_2, \ldots, v_{n-1})^T \neq \mathbf{0}$ we have ${\mathbf{v}^*}^T\mathbf{A}^*\mathbf{v}^* = v_2^2 + v_{n-1}^2 + \sum_{i=2}^{n-2}(v_{i+1} - v_i)^2 > 0$.
Hence $\mathbf{A}^*$ is nonsingular and we have a unique solution $U_1, \ldots, U_n$ to (6.20) depending continuously on f since $\tilde{\mathbf{U}}^* = h^2{\mathbf{A}^*}^{-1}\mathbf{f}^*$.
Secondly we verify that (6.21) *defines a solution to* (6.20)*:*
Recall from (6.3) and figure 6.1c that $G(t, x_j)$ is piecewise linear, with peak in x_j. Then

$$
\begin{aligned}
(6.23) \qquad -\delta_0^2 G_j(x_i) &= -\frac{1}{h}\left(G(x_{i-1}, x_j) - 2G(x_i, x_j) + G(x_{i+1}, x_j)\right) \\
&= \begin{cases} 0 \text{ for } i = 2, \ldots, j-1,\ j+1, \ldots, n-1 \\ \quad \text{since here } G \text{ is a line} \\ 1 \text{ for } i = j \text{ by (6.3)} \end{cases} \\
&= \delta_{i,j},\ i = 2, \ldots, n-1,\ j = 1, \ldots, n
\end{aligned}
$$

where $\delta_{i,j}$ is the Kronecker delta. Here the 1 in the braces comes from the following algebra: $G(x_{i-1}, x_i) - 2G(x_i, x_i) + G(x_{i+1}, x_i) = x_{i-1}(1 - x_i) - 2x_i(1-x_i) + x_i(1-x_{i+1}) = (x_i - h)(1-x_i) - 2x_i(1-x_i) + x_i(1-x_i-h) = -h$.
From (6.23) we get $-\delta_0^2\left(\sum_{j=1}^n G_j(x_i)f(x_j)\right) = f(x_i)$, $\forall i = 2, \ldots, n-1$, and since $G(0, s) = G(1, s) = 0$, for all $s \in [0, 1]$ we finally have $\sum_{j=1}^n G_j(x_1)f(x_j) = 0$ and $\sum_{j=1}^n G_j(x_n)f(x_j) = 0$. ■

Exercise 6.12
Verify that the matrix $\mathbf{A}^*$ in the proof of theorem 6.11 is positive definite as claimed. ■

Theorem 6.13 *For $f \in \mathcal{C}^2(0, 1)$,* (6.20) *is second order consistent with* (6.2)

and

$$\begin{aligned} &\tau_1 = \tau_n = 0, \\ &|\tau_i| = \left| -\left(\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} + f_i \right) \right| \leq \frac{h^2}{12} \|f''\|_\infty = \mathcal{O}(h^2), \\ &i = 2, \ldots, n-1, \\ &\Rightarrow \ \tau = \mathcal{O}(h^2). \end{aligned} \tag{6.24}$$

Proof
Taylor expansion. ■

To show convergence, we first introduce a notion that turns out to be the zero stability that we need.

Theorem 6.14 *Let* $\tilde{\mathbf{U}} = (\tilde{U}_1, \ldots, \tilde{U}_n)^T$ *and* $\mathbf{f} = (f_1, \ldots, f_n)^T$ *be the solution and right hand sides in* (6.20) *respectively and define the Discrete* $\mathcal{L}^\infty$*-norm by*

$$\|\mathbf{f}\|_{h,\infty} = \max_{i=1,\ldots,n} |f_i| \ \forall \mathbf{f} = (f_1, \ldots, f_n)^T. \tag{6.25}$$

Then the following Discrete Maximum Principle holds:

$$\|\tilde{\mathbf{U}}\|_{h,\infty} \leq \frac{1}{8} \|\mathbf{f}\|_{h,\infty} \tag{6.26}$$

Proof
By (6.21) we have $|\tilde{U}_i| \leq h \sum_{j=1}^n G(x_i, x_j) |f(x_j)| \leq \|\mathbf{f}\|_{h,\infty} h \sum_{j=1}^n G(x_i, x_j) \leq \frac{1}{8} \|\mathbf{f}\|_{h,\infty}$, $\forall i = 1, \ldots, n$. The proof of the last inequality is left as exercise 6.16 below. ■

Theorem 6.15 *For* $f \in \mathcal{C}^2(0,1)$*, the solution* $\tilde{\mathbf{U}} = (\tilde{U}_1, \ldots, \tilde{U}_n)^T$ *to* (6.20) *is second order convergent towards the exact solution vector* $\mathbf{u} = (u_1, \ldots, u_n)^T$ *to* (6.2) *and*

$$|u_i - \tilde{U}_i| \leq \|\mathbf{u} - \tilde{\mathbf{U}}\|_{h,\infty} \leq \frac{h^2}{96} \|f''\|_\infty = \mathcal{O}(h^2), \ i = 1, \ldots, n. \tag{6.27}$$

Proof
(6.26) implies zero stability of (6.20) with respect to ϵ-perturbations (see definition (3.43) on page 104) of the form

$$\begin{aligned} &-\delta_0^2 \tilde{Z}_{\epsilon,i} = f_i + \delta_{\epsilon,i}, \ |\delta_{\epsilon,i}| < \epsilon, \ i = 2, \ldots, n-1, \\ &\tilde{Z}_{\epsilon,1} = \tilde{Z}_{\epsilon,n} = 0, \ (\delta_{\epsilon,1} = \delta_{\epsilon,n} = 0), \end{aligned} \tag{6.28}$$

since by linearity and (6.26)

$$|\tilde{U}_i - \tilde{Z}_{\epsilon,i}| \leq \frac{1}{8}\epsilon, \ i = 1, \ldots, n. \tag{6.29}$$

By the convergence theorem of Lax (theorem 3.44 on page 105) taking $\delta_{\epsilon,i} = \tau_i$ and using also (6.24) we get (6.27). ■

Exercise 6.16
Fill in the details of the following verification of $h\sum_{j=1}^{n} G(x_i, x_j) \leq \frac{1}{8}$:

$$\begin{aligned}
&h\sum_{j=1}^{n} G(x_i, x_j) \\
&= h\left(\sum_{j=1}^{i} x_j(1-x_i) + \sum_{j=i+1}^{n} x_i(1-x_j)\right) \\
&= h^3\left(\sum_{j=1}^{i}(j-1)(n-i) + \sum_{j=i+1}^{n}(i-1)(n-j)\right) \\
&= \frac{1}{2}h^3\left((n-i)i(i-1) + (i-1)(n-i)(n-i-1)\right) \\
&= \frac{1}{2(n-1)^3}\left(-(n-1)i^2 + (n^2-1)i - (n^2-n)\right) \\
&\leq \frac{1}{2(n-1)^3}\left(-(n-1)\left(\frac{n+1}{2}\right)^2 + (n^2-1)\frac{n+1}{2} - (n^2-n)\right) \\
&= \frac{1}{8}
\end{aligned} \tag{6.30}$$

For the inequality, take the derivative with respect to i and see that it is zero at $i = \frac{n+1}{2}$ and note that this determines a maximum because of the negative coefficient of i^2. The last equality just requires a little algebra. ■

■

The key to the convergence result in theorem 6.15 is the discrete maximum principle, which again hinges on knowledge of the discrete Greens function. For the general problem (6.1) it is hard to find the discrete Greens function, and the approach hence becomes more difficult. An approach which does not depend directly on the discrete Greens function is to note that since the DEP is linear and hence can be formulated as

$$\mathbf{A}\tilde{\mathbf{U}} = \mathbf{f} \text{ and } \mathbf{A}\mathbf{u} = \mathbf{f} + \boldsymbol{\tau} \ \Rightarrow \ \mathbf{A}\mathbf{e} = \boldsymbol{\tau}, \tag{6.31}$$

for some $n \times n$ matrix $\mathbf{A}$ and some n vector $\mathbf{f}$, where $\tilde{\mathbf{U}} = (\tilde{U}_1, \ldots, \tilde{U}_n)^T$, $\mathbf{u} = (u_1, \ldots, u_n)^T$, $\mathbf{e} = (u_1 - \tilde{U}_1, \ldots, u_n - \tilde{U}_n)^T$ and $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_n)^T$. From (6.31) we get 0-stability and also convergence (given consistency) if only we can prove that $\mathbf{A}$ is non singular with an inverse that is uniformly bounded in the step length h (as $h \to 0$). Often it is possible to make some progress for specific examples either with the Greens function approach or with the matrix approach. Just as often though, the preferred solution is to transform the finite difference problem into an equivalent collocation or finite element problem (having the same identical system of equations) and use the approaches described in sections 6.4 and 6.5 below for those methods.

6.4 Convergence for CM's for $u'' = f$

Example 6.17 Convergence of CM's for the one dimensional Poisson problem (6.2)
In section 3.1.2 on page 66 we described how to construct collocation methods to find numerical approximations $\tilde{U} = \sum_{j=1}^{n} c_j \phi_j \in \mathcal{S} \subset \mathcal{C}^2(0,1)$ to the exact solution u to (6.2) of the form

$$\text{Find } (c_1, \ldots, c_n)^T \in \mathcal{R}^n : \; -\sum_{j=1}^{n} c_j \phi_j''(x_i) = f(x_i), \; i = 1, \ldots, n. \tag{6.32}$$

Here we shall concentrate on convergence results for such methods. We shall restrict to the case of $\mathcal{S} = \mathcal{P}^0_{n+1}[0,1]$ (polynomials p of degree at most $n+1$ with the boundary conditions $p(0) = p(1) = 0$ enforced) with dimension n. As basis functions for $\mathcal{P}^0_{n+1}[0,1]$ will be used the following Lagrange basis functions (cardinal functions)

$$\phi_j(t) = \prod_{k=0, k\neq j}^{n+1} \frac{t - x_k}{x_j - x_k}, \; j = 1, \ldots, n, \tag{6.33}$$

using the $n+2$ Gauss-Lobatto nodes satisfying $0 = x_0 < x_1 < \ldots < x_n < x_{n+1} = 1$ and having been selected such that there exist non zero Gauss-Lobatto weights $w_0, \ldots, w_{n+1}$ such that $\int_0^1 p(t)dt = \sum_{j=0}^{n+1} w_j p(x_j), \; \forall p \in \mathcal{P}_{2n+1}(0,1)$. Incidentally, taking $p \equiv 1$, we see that $\sum_{j=1}^{n} w_j = 1$. Note also that $\phi_j(x_i) = \delta_{i,j}$ so that $\tilde{U}_i = \tilde{U}(x_i) = c_i$, $i = 1, \ldots, n$.

Theorem 6.18 *Let $\tilde{U}$ and f be the solution generated from and right hand sides in* (6.32) *respectively and let $\|\cdot\|_p$ denote the $\mathcal{L}^p$ norm over the interval $(0,1)$. Then the following Maximum Principle holds:*

$$\exists C > 0 : \; \|\tilde{U}\|_2 \leq C \|f\|_\infty \tag{6.34}$$

Proof
For any $i = 1, \ldots, n$ we have

$$
\begin{aligned}
(6.35) \quad -w_i\tilde{U}''(x_i) &= -\sum_{j=0}^{n+1} w_j\tilde{U}''(x_j)\phi_i(x_j) \text{ (since } \phi_i(x_j) = \delta_{ij}) \\
&= -\int_0^1 \tilde{U}''(t)\phi_i(t)dt \text{ (since } \tilde{U}'' \in \mathcal{P}_{n-1}(0,1)) \\
&= \int_0^1 \tilde{U}'(t)\phi_i'(t)dt \text{ (since } \phi_i(0) = \phi_i(1) = 0).
\end{aligned}
$$

It can be shown that Poincaré's inequality holds, saying that

$$
\begin{aligned}
(6.36) \quad & \exists C_p > 0 : \ \|v\|_2 \le C_p\|v'\|_2, \\
& \forall v \in \{v \in \mathcal{L}^2(0,1) : \ v' \in \mathcal{L}^2(0,1) \text{ and } v(0) = v(1) = 0\}.
\end{aligned}
$$

Then using (6.36), (6.35) and (6.32) in that order we get

$$
\begin{aligned}
(6.37) \quad & \frac{1}{C_p^2}\|\tilde{U}\|_2^2 \le \|\tilde{U}'\|_2^2 = \sum_{i=1}^{n} \tilde{U}_i \int_0^1 \tilde{U}'(t)\phi_i'(t)dt = \sum_{i=1}^{n} \tilde{U}_i(-w_i\tilde{U}''(x_i)) \\
& = \sum_{i=1}^{n} \tilde{U}_i(w_i f(x_i)) = \sum_{j=1}^{n} w_j f(x_j)\tilde{U}(x_j) = \sum_{j=0}^{n+1} w_j f(x_j)\tilde{U}(x_j)
\end{aligned}
$$

since $\tilde{U}(x_0) = \tilde{U}(x_n) = 0$.
Using again the Gauss-Lobatto property $\sum_{j=0}^{n+1} w_j\tilde{U}(x_j) = \int_0^1 \tilde{U}\mathrm{d}t \le \int_0^1 |\tilde{U}|\mathrm{d}t$ $= \|\tilde{U}\|_1$ and the equivalence of norms on finite dimensional spaces ($\tilde{U} \in \mathcal{P}_{n+1}^0(0,1)$ with dimension $n < \infty$) that we shall take for granted, we get

$$
(6.38) \quad \sum_{j=0}^{n+1} w_j f(x_j)\tilde{U}(x_j) \le C\|f\|_\infty\|\tilde{U}\|_1 \le C\|f\|_\infty\|\tilde{U}\|_2.
$$

Putting (6.37) and (6.38) together, we get (6.34). ■

In section 3.4 on page 102 we defined the notions of truncation errors, consistency, convergence, ϵ-perturbation and zero stability for finite difference methods. For collocation methods we shall use the same notions but with the definitions slightly adjusted to fit the framework of collocation methods. Starting with the local truncation errors, for FDM's we replaced in the FDM equations the unknowns $\tilde{U}_i$ approximating the exact solution in the nodal points $u(x_i)$ by the exact values. For the CM's we similarly replace the unknowns c_j, that are approximations to the exact nodal point solution values

$u(x_i)$ as long as we use the Lagrange basis, by $u(x_i)$ which corresponds to replacing the function $\tilde{U}$ by Iu, the interpolant of u in the nodal points. For the other notions, the main difference lies in the fact that we for the CM's are recovering functions, while we for the FDM's recovered vectors. Hence norms for CM's will generally be function norms where for the FDM's they were vector norms:

Definition 6.19 *The Local Truncation Errors for the CM* (6.32) *for the Poisson DEP* (6.2) *are defined by*

$$\begin{aligned} \tau_i &= (-(Iu)''(x_i) - f(x_i)) - (-u''(x_i) - f(x_i)) \\ &= -(Iu)''(x_i) - f(x_i), \ i = 1, \ldots, n, \end{aligned} \tag{6.39}$$

where $Iu \in \mathcal{P}^0_{n+1}[0,1]$ is the interpolant of the exact solution u in the Gauss-Lobatto nodal points $x_0, \ldots, x_{n+1}$.
The Global Truncation Error for the CM (6.32) *for the DEP* (6.2) *is defined by*

$$\begin{aligned} \tau &= \max\{|\tau_1|, \ldots, |\tau_n|\} \\ &= \max\{|(Iu-u)''(x_1)|, \ldots, |(Iu-u)''(x_n)|\}. \end{aligned} \tag{6.40}$$

Consistency and Convergence is defined for CM's as for FDM's (see definition 3.40 on page 102, only do we normally use function norms (of $\mathcal{L}^p$-type) for the CM's where we mainly used vector norms (of ℓ^p-type) for the FDM's. Let ϵ be any positive real number, and $\delta_\epsilon \in \mathcal{C}[0,1]$ any continuous function satisfying $\|\delta_\epsilon\|_\infty < \epsilon$. Let as usual $\delta_{\epsilon,i} = \delta_\epsilon(x_i)$, $i = 1, \ldots, n$ so that also $|\delta_{\epsilon,i}| < \epsilon$, for $i = 1, \ldots, n$. The problem

$$\begin{aligned} &\text{Find } \tilde{Z}_\epsilon = \sum_{j=1}^{n} Z_{\epsilon,j}\phi_j \in \mathcal{P}^0_{n+1}[0,1] : \\ &-\sum_{j=1}^{n} Z_{\epsilon,j}\phi_j''(x_i) = f(x_i) + \delta_{\epsilon,i}, \ i = 1, \ldots, n, \end{aligned} \tag{6.41}$$

is called an ϵ-Perturbation of (6.32).
The CM (6.32) *is Zero Stable (in the norm $\|\cdot\|$ and with respect to ϵ-perturbations) if for all ϵ-perturbations* (6.41) *(fixed ϵ but arbitrary δ's) $\exists h_0 > 0$, $\exists C > 0$ (independent of h, ϵ and $\delta_{\epsilon,i}$ for $i = 1, \ldots, n$) such that $\|\tilde{U} - \tilde{Z}_\epsilon\| < C\epsilon$ $\forall h \in]0, h_0]$, i.e. for sets of nodal point, large enough to give sufficiently small step lengths.*

Theorem 6.20 *For $f \in \mathcal{C}[0,1]$, the solution $\tilde{U}$ to (6.32) is convergent in $\mathcal{L}^2$ norm towards the exact solution u to (6.2).*
Let $H^s(0,1) = \{v \in \mathcal{C}^{s-1}(0,1) : \ v^{(s)} \in \mathcal{L}^2(0,1)\}$ be the Sobolev space of order s with functions with weak derivatives up to order s in $\mathcal{L}^2(0,1)$. Let $\|\cdot\|_{H^s}$ be the norm on $\mathcal{H}^s(0,1)$.
If $f \in \mathcal{H}^s(0,1)$ then

$$\|u' - \tilde{U}'\|_2 \leq C \left(\frac{1}{n}\right)^s \left(\|f\|_{H^s} + \|u\|_{H^{s+1}}\right). \tag{6.42}$$

Proof
We start proving the convergence for continuous f:
The maximum principle (6.34) implies zero stability of (6.32) in the $\|\cdot\|_2$ norm and with respect to ϵ-perturbations according to definition 6.19, since by linearity and (6.34)

$$\|\tilde{U} - \tilde{Z}_\epsilon\|_2 \leq C\epsilon. \tag{6.43}$$

Taking $\delta_{\epsilon,i} = \tau_i$, $i = 1, \ldots, n$ and using (6.39) we get $\tilde{Z}_\epsilon = Iu$ so that $\|\tilde{U} - Iu\|_2 \leq C\tau \leq C\|(Iu - u)''\|_\infty$ and

$$\begin{aligned} \|\tilde{U} - u\|_2 &\leq \|\tilde{U} - Iu\|_2 + \|Iu - u\|_2 \\ &\leq C\|(Iu - u)''\|_\infty + \|Iu - u\|_2 \to 0 \text{ for } n \to \infty. \end{aligned} \tag{6.44}$$

From standard interpolation results, the convergence would be expected to be of order n^{-s} if $u \in \mathcal{C}^{s+1}[0,1]$.
The proof of (6.42) is somewhat more technical:
We start recalling (3.15) obtained by multiplying the ODE by a test function v, doing partial integration and using in this case $v(0) = v(1) = 0$ to eliminate the boundary terms. Using also (6.37) with constants c_i different from the $\tilde{U}_i$'s, corresponding to a generic $v = \sum_{i=1}^n c_i\phi_i \in \mathcal{P}^0_{n+1}[0,1]$ we get

$$\int_0^1 u'(t)v'(t)dt = \int_0^1 f(t)v(t)dt, \quad \int_0^1 \tilde{U}'(t)v'(t)dt = \sum_{j=1}^n w_j f(x_j)v(x_j) \tag{6.45}$$

$$\Downarrow$$

$$\begin{aligned} \int_0^1 (u - \tilde{U})'(t)v'(t)dt &= \int_0^1 f(t)v(t)dt - \sum_{j=1}^n w_j f(x_j)v(x_j) \\ &\leq C\left(\frac{1}{n}\right)^s \|f\|_{H^s}\|v\|_2, \end{aligned} \tag{6.46}$$

where the last inequality is a standard interpolation result which can be found for example in [4] §10.4 equation (10.36). Now we have

$$\begin{aligned}(6.47)\qquad &\int_0^1 ((u-\tilde{U})'(t))^2 dt \\ &= \int_0^1 (u-\tilde{U})'(t)(u-Iu)'(t)dt + \int_0^1 (u-\tilde{U})'(t)(Iu-\tilde{U})'(t)dt.\end{aligned}$$

For the first integral on the right hand side we shall need the inequality

$$(6.48)\qquad ab \leq \delta a^2 + \frac{1}{4\delta}b^2,\ \forall \delta > 0.$$

The proof is elementary: $0 \leq (\sqrt{2\delta}a - \frac{1}{\sqrt{2\delta}}b)^2 = -2ab + 2\delta a^2 + \frac{1}{2\delta}b^2$. Then taking $\delta = 1$

$$(6.49)\qquad \int_0^1 (u-\tilde{U})'(t)(u-Iu)'(t)dt \leq \frac{1}{4}\|(u-\tilde{U})'\|_2^2 + \|(u-Iu)'\|_2^2.$$

The last integral on the right hand side of (6.47) we handle with (6.46), recalling from definition 6.19 on page 175 that $Iu \in \mathcal{P}^0_{n+1}[0,1]$ is the interpolant of the exact solution u and hence can be used as v. We also use the triangle inequality, the Poincaré inequality (6.36) and (6.48) with $\delta = \frac{1}{2}$ and 1 for the two terms respectively:

$$\begin{aligned}(6.50)\qquad &\int_0^1 (u-\tilde{U})'(t)(Iu-\tilde{U})'(t)dt \leq C\left(\frac{1}{n}\right)^s \|f\|_{H^s}\|Iu-\tilde{U}\|_2. \\ &\leq C\left(\frac{1}{n}\right)^s \|f\|_{H^s}\|Iu-u\|_2 + C\left(\frac{1}{n}\right)^s \|f\|_{H^s}\|u-\tilde{U}\|_2. \\ &\leq C\left(\frac{1}{n}\right)^s \|f\|_{H^s}\|Iu-u\|_2 + CC_p\left(\frac{1}{n}\right)^s \|f\|_{H^s}\|(u-\tilde{U})'\|_2. \\ &\leq \frac{1}{2}C^2\left(\frac{1}{n}\right)^{2s} \|f\|_{H^s}^2 + \frac{1}{2}\|Iu-u\|_2^2 + C^2C_p^2\left(\frac{1}{n}\right)^{2s} \|f\|_{H^s}^2 \\ &\quad + \frac{1}{4}\|(u-\tilde{U})'\|_2^2.\end{aligned}$$

Collecting terms from (6.47), (6.49) and (6.50) we get

$$\begin{aligned}(6.51)\qquad &\|(u-\tilde{U})'\|_2^2 \\ &\leq C^2(1+2C_p^2)\left(\frac{1}{n}\right)^{2s} \|f\|_{H^s}^2 + 2\|(Iu-u)'\|_2^2 + \|Iu-u\|_2^2.\end{aligned}$$

Here the last two term are handled by standard interpolation results (see for example [4] §10.3 equation (10.22)) giving

$$(6.52)\quad \begin{aligned} &2\|(Iu-u)'\|_2^2 + \|Iu-u\|_2^2 \\ &\quad \le C_1\left(\frac{1}{n}\right)^{2s}\|u\|_{H^{s+1}}^2 + C_2\left(\frac{1}{n}\right)^{2s+2}\|u\|_{H^{s+1}}^2 \le C_3\left(\frac{1}{n}\right)^{2s}\|u\|_{H^{s+1}}^2. \end{aligned}$$

Hence we have

$$(6.53)\qquad \|(u-\tilde{U})'\|_2 \le C\left(\frac{1}{n}\right)^{s}\sqrt{\|f\|_{H^s}^2 + \|u\|_{H^{s+1}}^2}.$$

To get (6.42) we only need to use another algebraic inequality in the same family as (6.48)

$$(6.54)\qquad \sqrt{a^2+b^2} \le |a| + |b|.$$

The proof is again simple: $\sqrt{a^2+b^2} \le |a|+|b| \Leftrightarrow a^2+b^2 \le a^2+b^2+2|a||b| \Leftrightarrow 0 \le 2|a||b|$. ■

The first part of the previous proof is interesting since it uses the same approach to convergence that we have used for FDM's. Instead in the second part of the proof we rely heavily on Banach space theory using Cauchy-Schwartz, the triangle inequality and not the least Poincaré's inequality. ■

6.5 Convergence for FEM's for (6.1)

In the convergence proof for (6.42) (see theorem 6.20 on page 176) we used a variational form of the DEP obtained by multiplying the differential equation with a test function and doing partial integration to balance the number of derivatives on the trial and test functions. The boundary term arising in this partial integration was $[u'(x)v(x)]_{x=0}^{x=1} = 0$. It was a significant practical advantage that this boundary term vanished because of the boundary conditions on u and v. This advantage becomes even greater for finite element methods where the variational formulation is used not only for the convergence proofs but also for the construction of the numerical solution itself. For problem (6.1) one variational formulation (after partial integration but before eliminating boundary terms) is

$$(6.55)\quad \begin{aligned} &\int_I \left(-a(t)u'(t)v'(t) + (b(t)-a'(t))u'(t)v(t) + c(t)u(t)v(t) + d(t)v(t)\right)dt \\ &\quad + \left[a(t)u'(t)v(t)\right]_I = 0, \\ &\quad b_1u'(t_1) + c_1u(t_1) + d_1 = 0 \text{ and } b_2u'(t_2) + c_2u(t_2) + d_2 = 0. \end{aligned}$$

In order to make the boundary term vanish, we need first of all $t_1 \neq t_2$, i.e. boundary terms in two different points and $I = (t_1, t_2)$. Also Robin boundary conditions are problematic whereas Dirichlet and Neumann conditions are fine. We shall then assume that we have one boundary condition of the form $u(t^1) = e_1$ and another one of the form $u(t^2) = e_2$ or $u'(t^2) = e_2$ where t^1 is either t_1 or t_2 and t^2 is the other one ($t^2 \in \{t_1, t_2\},\ t^2 \neq t^1$). Since the test function v satisfies the same boundary conditions as the trial function u, the boundary term in (6.55) disappears if $e_1 = e_2 = 0$. This does not mean that we can handle only homogeneous boundary conditions with the finite element method, but it means that we need to insert another step in the finite element method. Before constructing the variational form we start Homogenizing the Boundary Conditions: In general we homogenize by "guessing" a function $\hat{u}$ so that $w = u - \hat{u}$ has homogeneous boundary conditions. Then if L is the differential operator in question, i.e. $Lu = 0$ is the differential equation, then $Lw = Lu - L\hat{u} = -L\hat{u} = \tilde{d}$. Let us consider two practical examples:

Example 6.21 Homogenizing the boundary conditions of (6.1) for the case $I = (t_1, t_2)$, $u(t_1) = e_1$, $u(t_2) = e_2$

Let

$$w(t) = u(t) - e_1 \frac{t_2 - t}{t_2 - t_1} - e_2 \frac{t - t_1}{t_2 - t_1}. \tag{6.56}$$

Then

$$w(t_1) = w(t_2) = 0 \tag{6.57}$$

and

$$\begin{aligned} &a(t)w''(t) + b(t)w'(t) + c(t)w(t) + d(t) \\ &\quad = a(t)u''(t) + b(t)u'(t) + c(t)u(t) + d(t) + \tilde{d}(t) = \tilde{d}(t), \end{aligned} \tag{6.58}$$

where

$$\tilde{d}(t) = b(t)e_1 \frac{1}{t_2 - t_1} - c(t)e_1 \frac{t_2 - t}{t_2 - t_1} - b(t)e_2 \frac{1}{t_2 - t_1} - c(t)e_2 \frac{t - t_1}{t_2 - t_1}. \tag{6.59}$$

Hence, we have homogenized the problem at the expense of replacing the function d with the function $d - \tilde{d}$.

The idea is now to work with the w-problem, find a finite element approximation $\tilde{W}$ and transform back to an approximation to u only at the very end defining

$$\tilde{U}(t) = \tilde{W}(t) + e_1 \frac{t_2 - t}{t_2 - t_1} + e_2 \frac{t - t_1}{t_2 - t_1}. \tag{6.60}$$

■

Example 6.22 Homogenizing the boundary conditions of (6.1) for the case $I = (t_1, t_2)$, $u(t_1) = e_1$, $u'(t_2) = e_2$
The Neumann condition is handled the same way as above. Let

$$w(t) = u(t) - e_1 - e_2(t - t_1). \tag{6.61}$$

Then

$$w(t_1) = w'(t_2) = 0 \tag{6.62}$$

and

$$a(t)w''(t) + b(t)w'(t) + c(t)w(t) + d(t) - \tilde{d}(t) = 0, \tag{6.63}$$

where

$$\tilde{d}(t) = -b(t)e_2 - c(t)e_1 - c(t)e_2(t - t_1). \tag{6.64}$$

Hence, we have again homogenized the problem at the expense of replacing the function d with the function $d - \tilde{d}$.
The idea is now again to work with the w-problem, find a finite element approximation $\tilde{W}$ and transform back to u only at the very end defining

$$\tilde{U}(t) = \tilde{W}(t) + e_1 + e_2(t - t_1). \tag{6.65}$$ ■

Returning to the boundary terms in (6.55) with the homogenized w replacing the original trial function u, i.e. $a(t_2)w'(t_2)v(t_2) - a(t_1)w'(t_1)v(t_1)$, we eliminate these terms by first using any homogeneous Neumann boundary condition on the trial function w. Dirichlet boundary conditions for w instead can not be used and must be enforced at some other place. Since there is at most one Neumann condition on w we still have one or two terms left corresponding exactly to the points where there are Dirichlet boundary conditions on the trial function. This or these terms we kill by enforcing homogeneous Dirichlet boundary conditions on the test function v. All homogeneous Dirichlet boundary conditions on the trial functions are hence countered with homogeneous Dirichlet boundary conditions also on the test functions, whereas Neumann boundary conditions on the trial functions are not countered by any conditions on the test functions.

After homogenisation we shall consider the following variational problem

$$
\begin{aligned}
(6.66) \qquad & \text{Find } w \in \mathcal{V}: \\
& \int_{t_1}^{t_2} \Big(-a(t)w'(t)v'(t) + (b(t) - a'(t))w'(t)v(t) \\
& \qquad + c(t)w(t)v(t) + (d(t) - \tilde{d}(t))v(t) \Big) dt = 0 \ \forall v \in \mathcal{V},
\end{aligned}
$$

where $\mathcal{V}$ is the subspace of the Sobolev space $\mathcal{H}^1(t_1, t_2)$, which is a Hilbert space so that also $\mathcal{V}$ is a Hilbert space, where the homogeneous Dirichlet conditions in t_1 and/or t_2 are enforced both on the trial and test functions. The eventual Neumann condition is not enforced on $\mathcal{V}$. It has already been used to eliminate the boundary term from the partial integration, and this Weak Enforcement turns out to be enough. Instead, the weak enforcement of the Dirichlet boundary conditions on the test functions turns out to be insufficient, and these must additionally be Strongly Enforced on $\mathcal{V}$ like the Dirichlet boundary conditions on the trial functions. (Recall that these have *not* been used to eliminate the boundary term from the partial integration). It is not obvious why this is so, but we can give some idea here:
Basically, we want (6.66) to be a Generalization of (6.1). By the derivation of (6.66) (and since $\mathcal{V} \supset \mathcal{C}^2(0,1)$) we know that if u satisfies (6.1) then the homogenized $w = u - \hat{u}$ satisfies (6.66), so if (6.1) has a solution, so does (6.66).
Instead, the opposite is generally not true, that is, if (6.66) has a solution, then (6.1) does not necessarily possess one.
The best we can hope for is that if w satisfies (6.66) and if w happens to be in $\mathcal{C}^2$, then $u = w + \hat{u}$ satisfies (6.1) (as long as $\hat{u}$ has been chosen in $\mathcal{C}^2$. Before indicating the proof of this, note that the terminology "generalization" is used because there still may be solutions to (6.66) in $\mathcal{V}$ but outside $\mathcal{C}^2$, so even if (6.1) does not have a solution, (6.66) may have one.
Returning to the proof, we start with a partial integration in (6.66) which is possible since $w \in \mathcal{C}^2$ and results in

$$
\begin{aligned}
(6.67) \qquad & \int_{t_1}^{t_2} \Big(a(t)w''(t) + b(t)w'(t) + c(t)w(t) + (d(t) - \tilde{d}(t)) \Big) v(t) dt \\
& = [aw'v]_{t_1}^{t_2} \ \forall v \in \mathcal{V} \\
& \Updownarrow \\
& \int_{t_1}^{t_2} \Big(a(t)w''(t) + b(t)w'(t) + c(t)w(t) + (d(t) - \tilde{d}(t)) \Big) v(t) dt \\
& = 0 \ \forall v \in \mathcal{V}^0,
\end{aligned}
$$

where $\mathcal{V}^0 = \{\mathcal{V} : \ v(t_1) = v(t_2) = 0\}$. Now there is only left to establish that there are "enough" functions in $\mathcal{V}^0$ to ensure, that the expression in the big parantheses under the integral in the last line of (6.67) must be zero almost everywhere. (To show this requires a good knowledge of Banach space theory, and we shall omit the details here). Continuity then implies that it is zero everywhere and adding $\hat{u}$ we recover the differential equation in (6.1). To recover the boundary conditions we shall for simplicity (but without losing generality) just consider the case $w(t_1) = a(t_2)w'(t_2) = 0$.
First take $v \equiv t - t_1 \in \mathcal{V}$ in the first line in (6.67). Then since we have already established that the differential equation in (6.1) is satisfied and the right hand side in the first line in (6.67) hence is zero for any $v \in \mathcal{V}$, we get

$$[aw'(t - t_1)]_{t_1}^{t_2} = 0 \Leftrightarrow a(t_2)w'(t_2) = 0, \tag{6.68}$$

hence recovering the second boundary condition.
Finally for the first boundary condition, $w(t_1) = 0$ is strongly enforced in $\mathcal{V}$ and hence is true.
Note that if we had enforced two homogeneous boundary conditions strongly in $\mathcal{V}$, then we would not have been able to take $v \equiv t - t_1$ and recover the second boundary condition. Hence it is important to follow the rules about boundary conditions on $\mathcal{V}$ given above.

Now we introduce the following bilinear and linear forms:

$$B(u, v) = \int_{t_1}^{t_2} (-a(t)u'(t)v'(t) + (b(t) - a'(t))u'(t)v(t) + c(t)u(t)v(t))\, dt \tag{6.69}$$

$$L(v) = \int_{t_1}^{t_2} (\tilde{d}(t) - d(t))v(t)dt$$

so that (6.66) can be rewritten in the shorter form

$$\text{Find } u \in \mathcal{V} : \ B(u, v) = L(v) \ \forall v \in \mathcal{V}. \tag{6.70}$$

Selecting the finite element space $\mathcal{S}$ as a finite dimensional subspace of $\mathcal{V}$, the discrete problem can be written simply as

$$\text{Find } \tilde{U} \in \mathcal{S} : \ B(\tilde{U}, v) = L(v) \ \forall v \in \mathcal{S}. \tag{6.71}$$

Let us at this point note the important Galerkin Orthogonality obtained by subtracting (6.71) from (6.70) taking test functions only in $\mathcal{S}$:

$$B(u - \tilde{U}, v) = 0 \ \forall v \in \mathcal{S}. \tag{6.72}$$

Galerkin orthogonality implies that the finite element error is orthogonal (measured with the bilinear form B) to $\mathcal{S}$.

Below we shall see that conditions for well-posedness and convergence hinges on (or at least are facilitated by) conditions of boundedness and ellipticity of the bilinear form B and boundedness of the linear form L.

Definition 6.23 *Let B and L be given by* (6.69) *and let $\|\cdot\|$ be a norm on $\mathcal{V}$ from* (6.70).
B is said to be Bounded if

$$\exists \overline{C} > 0: \ B(u,v) \leq \overline{C}\|u\|\|v\|, \ \forall u, v \in \mathcal{V}. \tag{6.73}$$

B is said to be Elliptic if

$$\exists \underline{C} > 0: \ B(u,u) > \underline{C}\|u\|^2, \ \forall u \in \mathcal{V} \setminus \{0\}. \tag{6.74}$$

L is said to be Bounded if

$$\exists C > 0: \ L(v) \leq C\|v\|, \ \forall v \in \mathcal{V}. \tag{6.75}$$

We have the following result:

Theorem 6.24 *Let B and L be given by* (6.69).

- *B is bounded if a, $b - a'$ and $c \in \mathcal{L}^\infty(t_1, t_2)$.*
- *L is bounded if $\tilde{d} - d \in \mathcal{L}^2(t_1, t_2)$.*
- *B is elliptic if $-a(t) \geq a_0 > 0$, $c(t) \geq c_0 > 0$ and $b(t) - a'(t) = 0$ $\forall t \in [t_1, t_2]$.*
  If $u(t_1) = u(t_2) = 0$ then the last two conditions can be replaced by $-\frac{1}{2}(b(t) - a'(t))' + c(t) \geq 0$ $\forall t \in [t_1, t_2]$ without losing ellipticity.

Proof:
The proofs of all but the last statement are simple once knowing the Sobolev norms. A norm on $\mathcal{H}^1$ and hence on $\mathcal{V}$ is $\|\cdot\|_{H^1}$ defined by $\|v\|_{H^1} = \sqrt{\|v'\|_2^2 + \|v\|_2^2}$, where $\|\cdot\|_2$ is the $\mathcal{L}^2$-norm.
For the boundedness of B we use (6.69), the Cauchy-Schwarz inequality and the definition of $\|\cdot\|_{H^1}$ to get

$$\begin{aligned} B(u,v) &\leq \|a\|_\infty \|u'\|_2 \|v'\|_2 + \|b - a'\|_\infty \|u'\|_2 \|v\|_2 + \|c\|_\infty \|u\|_2 \|v\|_2 \\ &\leq \{\|a\|_\infty + \|b - a'\|_\infty + \|c\|_\infty\} \|u\|_{H^1} \|v\|_{H^1}. \end{aligned} \tag{6.76}$$

For the boundedness of L we again use (6.69), the Cauchy-Schwarz inequality and the definition of $\|\cdot\|_{H^1}$ to get

$$L(v) \leq \|\tilde{d} - d\|_2 \|v\|_2 \leq \|\tilde{d} - d\|_2 \|v\|_{H^1}. \tag{6.77}$$

For the ellipticity of B in the first case we use (6.69) and the definition of $\|\cdot\|_{H^1}$ to get

$$B(u,u) \geq \min\{a_0, c_0\}\|u\|^2_{H^1}. \tag{6.78}$$

For the ellipticity of B in the second case we use (6.69) and the fact that $u'u = \frac{1}{2}(u^2)'$ to get

$$\begin{aligned} B(u,u) &= \int_{t_1}^{t_2} \left(-au'^2 + (b-a')\frac{1}{2}(u^2)' + cu^2\right) dt \\ &= \int_{t_1}^{t_2} \left(-au'^2 + (c - \frac{1}{2}(b-a')')u^2\right) dt + [\frac{1}{2}(b-a')u^2]_{t_1}^{t_2} \\ &\geq a_0\|u'\|_2^2 \geq C\|u\|^2_{H^1}. \end{aligned} \tag{6.79}$$

Here the first inequality comes from the fact that the boundary term disappears since $u(t_1) = u(t_2) = 0$. The second inequality is an application of the Poincaré inequality (6.36), giving $a_0\|u'\|_2^2 = \frac{a_0}{C_p^2+1}\|u'\|_2^2 + \frac{C_p^2 a_0}{C_p^2+1}\|u'\|_2^2 \geq \frac{a_0}{C_p^2+1}\|u'\|_2^2 + \frac{a_0}{C_p^2+1}\|u\|_2^2 = \frac{a_0}{C_p^2+1}\|u\|^2_{H^1}$. ■

Theorem 6.25 *Let $\mathcal{V}$ and $\mathcal{S}$ be Hilbert spaces, or just reflexive Banach spaces, let B be a bilinear, bounded (constant $\overline{C}$), elliptic (constant $\underline{C}$) form and let L be a linear, bounded form.*
Then (6.70) *and* (6.71) *are well-posed (Lax-Milgram Lemma), stable ($\|u\| < \frac{\|L\|}{\underline{C}}$ and $\|\tilde{U}\| < \frac{\|L\|}{\underline{C}}$) and the solution to* (6.71) *converges towards the solution to* (6.70) *as $\mathcal{S} \to \mathcal{V}$ since $\|u - \tilde{U}\| \leq \frac{\overline{C}}{\underline{C}} \min_{w\in\mathcal{S}} \|u - w\|$ (Céa's Lemma).*
For the Order of Convergence, if $u \in \overline{\mathcal{H}}^s$, $r \leq s$ and $\mathcal{S}$ is the space of globally continuous, piecewise polynomials of degree at most k, with two homogeneous Dirichlet boundary conditions enforced and with uniform subdivisions with element length h, then

$$\begin{aligned} &\|u - \tilde{U}\|_{H^r} \leq Ch^{\ell-r}\|u\|_{H^\ell}, \\ &\textit{where } \ell = \min\{k, s\} \textit{ is the Regularity Treshold.} \end{aligned} \tag{6.80}$$

Proof:
The proof of existence and uniqueness in the Hilbert space case is a fairly simple consequence of the Riesz representation theorem and the contraction mapping theorem. The extension to reflexive Banach spaces is more technical. The continuous dependence on data is realized as follows: Let $B_\epsilon(u_\epsilon, v) = L_\epsilon(v)$, $\forall v \in \mathcal{V}$ be a perturbation of (6.70) with $\|B - B_\epsilon\| < \epsilon$ and $\|L - L_\epsilon\| < \epsilon$. Then $\underline{C}\|u - u_\epsilon\|^2 < B(u - u_\epsilon, u - u_\epsilon) = (L - L_\epsilon)(u - u_\epsilon) + (B_\epsilon -$

$B)(u_\epsilon, u-u_\epsilon) \leq \|L-L_\epsilon\|\|u-u_\epsilon\|+\|B-B_\epsilon\|\|u_\epsilon\|\|u-u_\epsilon\| \Rightarrow \|u-u_\epsilon\| < \frac{1+\|u_\epsilon\|}{\underline{C}}\epsilon$ and similarly for (6.71).
The stability result is obtained as the continuous dependence on data using $\underline{C}\|u\|^2 < B(u,u) = L(u) \leq \|L\|\|u\|$ and similarly for (6.71).
Céa's lemma follows from Galerkin orthogonality as follows: $\underline{C}\|u-\tilde{U}\|^2 < B(u-\tilde{U}, u-\tilde{U}) = B(u-\tilde{U}, u-w), \forall w \in \mathcal{S}$ because Galerkin orthogonality implies $B(u-\tilde{U}, \tilde{U}) = B(u-\tilde{U}, w) = 0$. But $B(u-\tilde{U}, u-w) \leq \overline{C}\|u-\tilde{U}\|\|u-w\|$.
We shall not prove the regularity treshold but only note that the implication is that the order of convergence increases with the polynomial degree k until we reach the point where the exact solution is no longer smooth enough ($u \notin \mathcal{H}^k$). After this point an increase in k will have no influence on the order of convergence. ■

For the finite element method, we have avoided any discussion of consistency, ϵ-perturbations and zero stability. It is possible (but not necessary, and never used in practice) to introduce these notions as follows:
The Local Truncation Errors for the FEM (6.71) are defined by

$$B(Iu, \phi_i) = L(\phi_i) + \tau_i \int_{t_1}^{t_2} \phi_i(t)dt, \ i = 1, \ldots, n, \tag{6.81}$$

where Iu is the $\mathcal{S}$-interpolant of u.
The ϵ-perturbations of (6.71) would then be

$$B(\tilde{Z}_\epsilon, \phi_i) = L(\phi_i) + \delta_{\epsilon,i} \int_{t_1}^{t_2} \phi_i(t)dt, \ i = 1, \ldots, n, \tag{6.82}$$

where $\delta_{\epsilon,i} < \epsilon$ for $i = 1, \ldots, n$.
Letting $\tilde{U} - \tilde{Z}_\epsilon = \sum_{i=1}^n d_i\phi_i$ we then have
$\underline{C}^2\|\tilde{U} - \tilde{Z}_\epsilon\|_{H^1}^2 < B(\tilde{U} - \tilde{Z}_\epsilon, \tilde{U} - \tilde{Z}_\epsilon) = -\int_{t_1}^{t_2} \sum_{i=1}^n \delta_{\epsilon,i} d_i \phi_i(t)dt \leq \epsilon\|1\|_2\|\tilde{U} - \tilde{Z}_\epsilon\|_{H^1}$,
giving the Zero Stability

$$\|\tilde{U} - \tilde{Z}_\epsilon\|_{H^1} < \frac{\sqrt{t_2 - t_1}}{\underline{C}^2}\epsilon. \tag{6.83}$$

Then we would get convergence as for the collocation methods, taking $\delta_{\epsilon,i} = \tau_i$ so that $\tilde{Z}_\epsilon = Iu$. The problem is that it is hard to get an expression for the order of consistency in this setting since the truncation errors are hard to handle. Hence this approach is not used in practice.

Let us end up with some comments on the numerical problem: Let $\mathcal{S} = \text{span}\{\phi_1, \ldots, \phi_n\}$ and denote the matrix form of (6.71) (see (3.16) and (3.17))

$$\mathbf{Ac} = \mathbf{q}, \text{ where } A_{ij} = B(\phi_j, \phi_i), \ q_i = L(\phi_i), \ i, j = 1, \ldots, n. \tag{6.84}$$

A is denoted the Stiffness matrix and **q** the Load Vector. The notation comes from the origins of the finite element method as a method for computing stresses and strains in airplane wings and makes often little sense for the wide range of problems being solved with FEM's today.

Theorem 6.26 *Let B be the bilinear, elliptic form of* (6.71) *and* **A** *the matrix of* (6.84)*. Then*

- **A** *is positive definite.*
- **A** *is symmetric* $\Leftrightarrow$ B *is symmetric.*

Proof:
That **A** is positive definite is clear from the following argument:

$$\begin{aligned} \mathbf{v}^T\mathbf{A}\mathbf{v} &= \sum_{i=1}^{n}\sum_{j=1}^{n} v_i A_{ij} v_j = \sum_{i=1}^{n}\sum_{j=1}^{n} v_i B(\phi_j,\phi_i) v_j \\ &= B(\sum_{j=1}^{n} v_j\phi_j, \sum_{i=1}^{n} v_i\phi_i) = B(v,v) > \underline{C}\|v\|^2 \geq 0, \end{aligned} \tag{6.85}$$

with equality only if $\|v\| = 0 \Leftrightarrow \mathbf{v} = \mathbf{0}$.
A symmetric $\Leftrightarrow A_{ij} = A_{ji} \Leftrightarrow B(\phi_j,\phi_i) = B(\phi_i,\phi_j) \Leftrightarrow B$ symmetric. ■

Note that a symmetric, positive definite matrix problem can be solved with Gaussian elimination without pivoting.

Finally a note on the construction of basis functions: Considering the basis functions of figure 3.2 on page 75 they are normally implemented with linear mappings from reference basis functions

$$\psi_1 = \begin{cases} 1-y & \text{for } 0<y<1 \\ 0 & \text{else} \end{cases}, \qquad \psi_2 = \begin{cases} y & \text{for } 0<y<1 \\ 0 & \text{else} \end{cases}. \tag{6.86}$$

This is a general principle used whenever possible. This way it turns out, that all integrals in (6.84) can be expressed as linear mappings of integrals of products of the reference basis functions over $(0,1)$.

When going to polynomial degrees $k > 1$, two types of basis functions are in use: The Lagrange Bases satisfying $\phi_i^{(k)}(x_j) = \delta_{ij}$ and the Hierarchical Bases where $\{\phi_i^{(k)}\}_{i=1}^{n_k}$ is constructed by adding new Bubble Basis Functions to $\{\phi_i^{(k-1)}\}_{i=1}^{n_{k-1}}$. For example, the reference Lagrange basis for $k=2$ consists of $\psi_1(y) = 2(y-\frac{1}{2})(y-1)$, $\psi_2(y) = -4y(y-1)$ and $\psi_3(y) = -2y(y-\frac{1}{2})$ while the reference hierarchical basis for $k=2$ consists of $\psi_1(y) = 1-y$, $\psi_2(y) = -4y(y-1)$ and $\psi_3(y) = y$ ■.

Exercise 6.27
Plot the reference Lagrange and Hierarchical bases given just above. Construct cubic reference Lagrange and Hierarchical bases ($k = 3$). ■

6.6 Non asymptotic analysis of (6.1): The Convection-Diffusion problem

Let us reconsider the general problem (6.1) only put in divergence form like in (6.6) (but without taking $b = a'$) or like in the construction of the variational form (6.55) and with a particular selection of boundary conditions:

$$(6.87)\quad \underbrace{(a(t)u'(t))'}_{1} + \underbrace{(b(t) - a'(t))u'(t)}_{2} + \underbrace{c(t)u(t)}_{3} + d(t) = 0,\ 0 < t < 1,$$
$$u(0) = 0,\ u(1) = 1.$$

The first term (1) is denoted the Diffusion or Viscosity Term, the second term (2) the Convection, Advection or Transport Term and the third term (3) the Absorbtion or Reaction Term. The notation comes from physics and may hold little significance for particular applications. In order to expose the non asymptotic problems for (6.1) we focus on the convection and diffusion terms and consider the following simple example:

Example 6.28 $-\epsilon u'' + \beta u' = 0,\ u(0) = 0,\ u(1) = 1$
We consider the problem

$$(6.88)\ -\epsilon u''(t) + \beta u'(t) = 0,\ 0 < t < 1,\ u(0) = 0,\ u(1) = 1,\ \epsilon > 0,\ \beta \geq 0.$$

We have taken $\beta \geq 0$ only in order to simplify notation. The following works fine also for negative β's. Homogenizing by taking $w(t) = u(t) - t$ so that $-\epsilon w''(t) + \beta w'(t) = -\beta$, $w(0) = w(1) = 0$ and using theorem 6.25 on page 184, definition 6.23 on page 183 and theorem 6.24 on page 183 we easily verify that this is a well-posed problem with convergent finite element approximations as the approximation space $\mathcal{S}$ converges to the "variational" space $\mathcal{V}$. Actually, the problem is so simple that we can find the exact solution

$$(6.89)\ u(t) = \frac{e^{\frac{\beta t}{\epsilon}} - 1}{e^{\frac{\beta}{\epsilon}} - 1} \in \mathcal{C}^\infty[0,1],\ \text{for } \epsilon, \beta > 0,\ u(t) = t,\ \text{for } \epsilon > 0,\ \beta = 0.$$

The solution is plotted in figure 6.2 for various values of ϵ and β. Fixing $\epsilon > 0$ we see that $\lim_{\beta\to 0} u(t) = t$, and that $u(t) = t$ is also the solution to (6.88) for $\beta = 0$ so that there is a continuous behavior in this limit. More specifically

$$\epsilon \gg \beta \Rightarrow u(t) = t + \mathcal{O}(\frac{\beta}{\epsilon}t^2). \text{ (Verified with Taylor expansion).} \tag{6.90}$$

Fixing instead β we see that

$$\epsilon \ll \beta \Rightarrow u(t) \simeq \begin{cases} \frac{e^{\frac{\beta t}{\epsilon}}}{e^{\frac{\beta}{\epsilon}}} = e^{\frac{\beta}{\epsilon}(t-1)} & \text{for } t \text{ “}\neq\text{”} 0 \\ & (t \text{ not very close to } 0) \\ \frac{e^{\frac{\beta t}{\epsilon}}-1}{e^{\frac{\beta}{\epsilon}}} \simeq 0 & \text{for } t \text{ “}=\text{”} 0 \\ & (t \text{ very close to } 0) \end{cases} \tag{6.91}$$

$$\simeq 0 \text{ for } t \text{ “}\neq\text{”} 1 \ (t \text{ not very close to } 1) \text{ while } u(1) = 1,$$

and $\lim_{\epsilon\to 0} u(t) = \begin{cases} 0 & \text{for } t < 1 \\ 1 & \text{for } t = 1 \end{cases}$, which is *not* a solution to (6.88) for $\epsilon = 0$ since it is not a $\mathcal{C}^2[0,1]$-function. Actually, (6.88) *has* no solution (in the classical sense, i.e. in $\mathcal{C}^2[0,1]$) for $\epsilon = 0$. Instead for $\epsilon > 0$ but very small, the exponential growth towards $t = 1$ is clear also from figure 6.2. Without making a clear definition, we say that in a small region of size δt at the boundary of the computational domain where there is a big change (relative to δt) in the solution values or some functional or derivative of these there is a Boundary Layer of Width δt. If the layer is situated in the interior of the computational domain, we talk about an Internal Layer of width δt. Since “small” regions and “big” changes are relative notions, often the width of the layer is given as $\mathcal{O}(\delta t)$, focusing on the manner in which the size of the layer decreases with relevant problem parameters. If the “activity” in the layer is “focused” around a certain point t^*, we may say that there is a Layer at t^*. In our case, for $\epsilon \ll \beta$ we have a boundary layer at $t = 1$. To determine its width, note that for $e^{\frac{\beta}{\epsilon}(t-1)}$ to decrease from the value 1 at $t = 1$ to some small positive value $\delta < 1$, t must decrease to $1 - \delta t$ where $\delta t = |\ln(\delta)|\frac{\epsilon}{\beta}$. Hence we say that the width of the boundary layer at $t = 1$ is $\mathcal{O}(\epsilon/\beta)$. In practice, layers are often observed in numerical solutions in the form of oscillations which do not decrease with decreasing step lengths fast enough to show the expected rate of convergence. The oscillations can be avoided by selecting very small step lengths or subdivisions h. This is called Resolving the Layer, but in practice it may be problematic if ϵ/β is very small. A ratio of just 10^{-6} would require 1 million steps, and values much less than this is not unrealistic in practical situations. Hence we are in

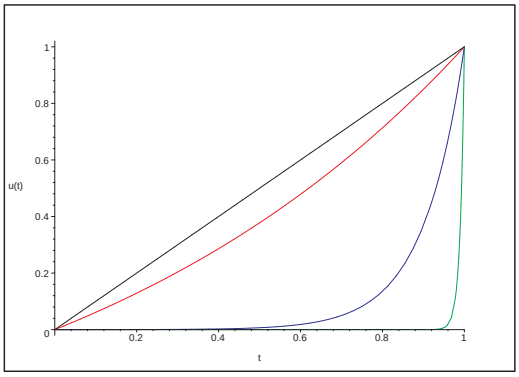


Figure 6.2: Solution to convection-diffusion problem (6.88) for $\epsilon > 0$, $\beta = 0$ (black, top curve—), $\epsilon = 1$, $\beta = 1$ (red, second from the top curve—), $\epsilon = 0.1$, $\beta = 1$ (blue, third from the top curve—) and $\epsilon = 0.01$, $\beta = 1$ (green, bottom curve—).

a situation where the non asymptotical behavior of the solution is far from the asymptotical behavior where we know that we have convergence. To understand the reason for the oscillations in the numerical solutions and how to avoid them (apart from the unrealistic possibility of resolving the layer) we first study how a couple of standard methods work on (6.88).

Solving (6.88) with linear finite elements:

We shall use the following variational formulation as basis for our FEM:

$$\text{(6.92)} \qquad \text{Find } u \in \mathcal{S}^{1*} : \int_0^1 (\epsilon u'(t)v'(t) + \beta u'(t)v(t))dt = 0, \ \forall v \in \mathcal{S}^{1*},$$

where $\mathcal{S}^{1*}$ is the space of continuous, piecewise linear functions p with $p(0) = 0$ and $p(1) = 1$ relative to a uniform subdivision $0 = x_0 < x_1 < \ldots, x_n < x_{n+1} = 1$, where $h = x_{i+1} - x_i$ for all $i = 0, \ldots, n$. To simplify the exposition, we add also the ghost node $x_{n+2} = 1 + h$. We take the usual basis $\{\phi_j\}_{j=1}^{n+1}$ for $\mathcal{S}^{1*}$ shown in figure 3.2 on page 75, except that we have added one node with respect to that figure, so that $\tilde{U} = \sum_{i=1}^{n+1} \tilde{U}_i \phi_i$ where $\tilde{U}_{n+1} = 1$ is given by the right boundary condition and v can be taken to be any of ϕ_j for $j = 1, \ldots, n$. We also let $\tilde{U}_0 = 0$ be given by the left boundary condition.

To evaluate (6.92) it is important to realize a fact that simplifies matters significantly. Note that

$$\text{(6.93)} \quad \phi_i(t) = \begin{cases} \frac{1}{h}(t - x_{i-1}) & \text{for } t \in [x_{i-1}, x_i] \\ \frac{1}{h}(x_{i+1} - t) & \text{for } t \in [x_i, x_{i+1}] \\ 0 & \text{else} \end{cases} \qquad \text{for } i = 1, \ldots, n+1,$$

and

$$\phi_i'(t) = \begin{cases} \frac{1}{h} & \text{for } t \in [x_{i-1}, x_i] \\ -\frac{1}{h} & \text{for } t \in [x_i, x_{i+1}] \\ 0 & \text{else} \end{cases} \qquad \text{for } i = 1, \ldots, n+1, \tag{6.94}$$

have support in only two elements each. This means that major parts of the integral over $(0,1)$ in (6.92) is void. This fact simplifies the practical computations significantly and *must* be utilized when programming finite element methods. Also

$$\int_0^1 (\phi_i'(t))^2 dt = \frac{2}{h}, \ \int_0^1 \phi_i'(t)\phi_{i\pm1}'(t)dt = -\frac{1}{h}, \ \int_0^1 \phi_i(t)\phi_i'(t)dt = 0, \tag{6.95}$$
$$\int_0^1 \phi_i(t)\phi_{i\pm1}'(t)dt = \pm\frac{1}{2}, \quad \text{for } i = 1, \ldots, n.$$

Returning to (6.92) it takes the form

(6.96) For $i = 1, \ldots, n$:

$$\begin{aligned} 0 &= \int_0^1 \sum_{j=1}^{n+1} \tilde{U}_j \phi_j'(t)(\epsilon\phi_i'(t) + \beta\phi_i(t))dt \\ &= \int_0^1 (\tilde{U}_{i-1}\phi_{i-1}'(t) + \tilde{U}_i\phi_i'(t) + \tilde{U}_{i+1}\phi_{i+1}'(t))(\epsilon\phi_i'(t) + \beta\phi_i(t))dt \\ &= \epsilon\left(-\frac{1}{h}\tilde{U}_{i-1} + \frac{2}{h}\tilde{U}_i - \frac{1}{h}\tilde{U}_{i+1}\right) + \beta\left(-\frac{1}{2}\tilde{U}_{i-1} + \frac{1}{2}\tilde{U}_{i+1}\right) \\ \Leftrightarrow \quad & \text{(multiplying with } -\tfrac{h}{\epsilon}\text{)} \end{aligned}$$

$$0 = (1+Pe)\tilde{U}_{i-1} - 2\tilde{U}_i + (1-Pe)\tilde{U}_{i+1}, \tag{6.97}$$

where we have defined the Local Pechlet Number $Pe = \frac{\beta h}{2\epsilon}$. Shifting the index from i to $i-1$, extending towards ∞ and enforcing the boundary conditions we end up with the following linear homogeneous difference equation,

$$\frac{1+Pe}{1-Pe}\tilde{U}_i - \frac{2}{1-Pe}\tilde{U}_{i+1} + \tilde{U}_{i+2} = 0, \ \text{for } i = 0, 1, 2, \ldots, \tag{6.98}$$
$$\tilde{U}_0 = 0, \ \tilde{U}_{n+1} = 1,$$

that we can solve with the method of section 4.3.1. The characteristic polynomial is

$$\rho(r) = \frac{1+Pe}{1-Pe} - \frac{2}{1-Pe}r + r^2 = (r-1)\left(r - \frac{1+Pe}{1-Pe}\right), \tag{6.99}$$

and hence the solution to (6.98) is of the form

$$\tilde{U}_i = C_1 + C_2 \left(\frac{1+Pe}{1-Pe}\right)^i, \ i = 0, \ldots, n+1, \tag{6.100}$$

where $\tilde{U}_0 = 0$ and $\tilde{U}_{n+1} = 1$ determines C_1 and C_2 to be

$$C_2 = \frac{1}{\left(\frac{1+Pe}{1-Pe}\right)^{n+1} - 1} = -C_1, \tag{6.101}$$

so that

$$\tilde{U}_i = \frac{\left(\frac{1+Pe}{1-Pe}\right)^i - 1}{\left(\frac{1+Pe}{1-Pe}\right)^{n+1} - 1}, \ i = 0, \ldots, n+1. \tag{6.102}$$

It is clear from (6.102) and figure 6.3a that $\tilde{U}_i$ changes sign (oscillates) with i if $Pe > 1$ and that the amplitude is increasing with i. Instead it turns out that the amplitude is strictly decreasing with Pe. This is indicated in figure 6.3b where we show the exact solution $u(t)$ given by (6.89) with the Global Pechlet Number $Pe_{gl} = \frac{\beta}{2\epsilon} = 50$ together with numerical solutions with 10, 20, 30 and 40 elements in the subdivisions. Note that $Pe = hPe_{gl} < 1 \Leftrightarrow h < \frac{1}{Pe_{gl}}$. The numerical solutions for more than 40 elements begin resembling the exact solution a lot in the scale chosen in figure 6.3b. Only do we see small oscillations (and negative solution values) for less that 50 elements, while more than 50 ($Pe < 1$) give no oscillations and no negative solution values ■ .

Solving (6.88) with centered finite differences:

Note that (6.96) (after multiplying with $\frac{1}{h}$) is exactly the 2nd order centered difference sceme for (6.88) which hence has the same identical behavior as the linear finite element solution. The only difference is that the lines joining the oscillating points are part of the finite element solution while they are not part of the finite difference solution. Hence the oscillation problem is not something inherent in the finite element method but occurs also in the finite difference method and actually in all numerical methods for (6.88) if no special care is taken ■ .

Remedy for the oscillatory numerical solutions to (6.88) with finite differences:

We shall show how to avoid the oscillations based in the finite difference formulation since it is somewhat simpler here, even though the approach is entirely equivalent for the finite element (and collocation) methods.

For the example shown in figure 6.3b we only needed 50 elements to resolve the boundary layer at $t = 1$. If $Pe_{gl} = 10^6$ we need 1.000.000 which is

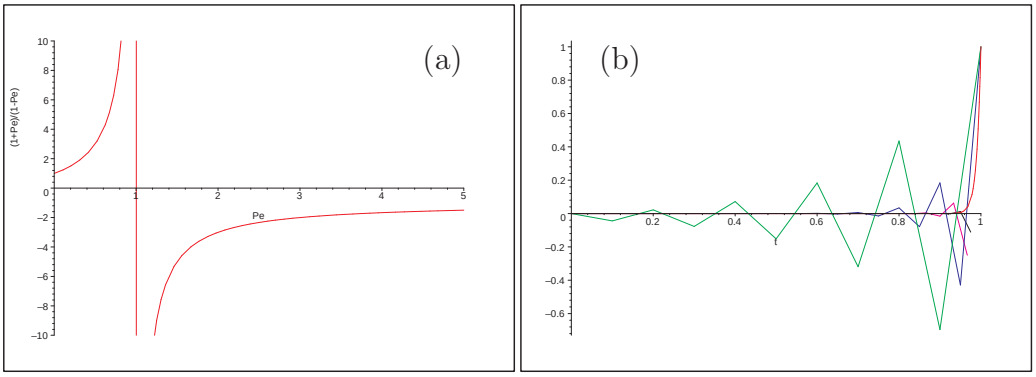


Figure 6.3: (a) Plot of $\frac{1+Pe}{1-Pe}$ as a function of Pe. (b) Plot of $u(t)$ (red, non oscillating curve—) and $\tilde{U}(t)$ as a function of t for $Pe_{gl} = 50$ and $Pe = 5$ i.e. $h = \frac{1}{n+1} = \frac{1}{10} = 0.1$ (green, most oscillating curve—), $Pe = 2.5$ i.e. $h = \frac{1}{n+1} = \frac{1}{20} = 0.05$ (blue, second most oscillating curve—), $Pe = 1.67$ i.e. $h = \frac{1}{n+1} = \frac{1}{30} = 0.025$ (magenta, third most oscillating curve—) and $Pe = 1.25$ i.e. $h = \frac{1}{n+1} = \frac{1}{40} = 0.0125$ (black, fourth most oscillating curve—). (In the latter two plots, the last line up to 1 has been omitted to avoid cluttering the graph close to $t = 1$.

too much for casual use. So the solution we are seeking is *not* that of resolving the layer. Instead in (6.96) we simply replace δ_0 by δ_- when approximating the first derivative. (If $\beta < 0$ we would have taken δ_+ instead). This approach is called Upwinding and the resulting method is called the Upwinding Finite Difference Sceme:

(6.103) For $i = 1, \ldots, n$:

$$\begin{aligned} 0 &= -\epsilon \frac{\hat{U}_{i-1} - 2\hat{U}_i + \hat{U}_{i+1}}{h^2} + \beta \frac{\hat{U}_i - \hat{U}_{i-1}}{h} \\ &= -\left(\epsilon + \frac{\beta h}{2}\right) \frac{\hat{U}_{i-1} - 2\hat{U}_i + \hat{U}_{i+1}}{h^2} + \beta \frac{\hat{U}_{i+1} - \hat{U}_{i-1}}{2h} \\ &= -\epsilon\,(1 + Pe)\,\delta_0^2 \hat{U}_i + \beta \delta_0 \hat{U}_i. \end{aligned}$$

The latter form is the 2nd order centered finite difference sceme for the differential equation $-\epsilon\,(1 + Pe)\,u'' + \beta u' = 0$, where the diffusion has been augmented by $-\epsilon Pe$ with respect to (6.88) and for this reason the upwinding method is also denoted the Artificial Diffusion or Numerical Viscosity Sceme. Taking $\hat{U}_0 = 0$ and $\hat{U}_{n+1} = 1$, the solution to (6.103) is still in the form of

(6.102) only with a new Pechlet number $\overline{Pe} = \frac{\beta h}{2\left(\epsilon + \frac{\beta h}{2}\right)}$

$$\hat{U}_i = \frac{\left(\frac{1+\overline{Pe}}{1-\overline{Pe}}\right)^i - 1}{\left(\frac{1+\overline{Pe}}{1-\overline{Pe}}\right)^{n+1} - 1} = \frac{\left(1+\frac{\beta h}{\epsilon}\right)^i - 1}{\left(1+\frac{\beta h}{\epsilon}\right)^{n+1} - 1}, \; i = 0, \ldots, n+1. \tag{6.104}$$

It is clear from (6.104) and the plots in figure 6.4 that $\hat{\mathbf{U}}$ does not oscillate. After enjoying the lack of oscillations in $\hat{\mathbf{U}}$, it might be noted that the clut-

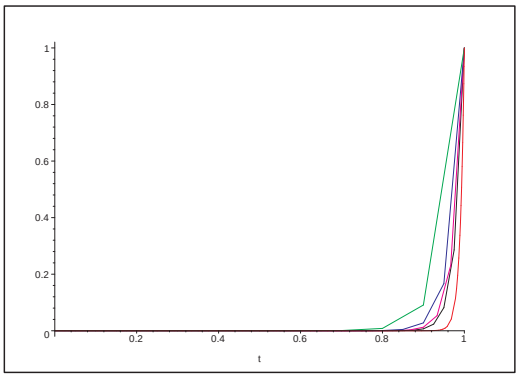


Figure 6.4: Plot of $u(t)$ (red, rightmost curve—) and $\hat{U}(t)$ as a function of t for $Pe_{gl} = 50$ and $Pe = 5$ i.e. $h = \frac{1}{n+1} = \frac{1}{10} = 0.1$ (green, leftmost curve—), $Pe = 2.5$ i.e. $h = \frac{1}{n+1} = \frac{1}{20} = 0.05$ (blue, second rightmost curve—), $Pe = 1.67$ i.e. $h = \frac{1}{n+1} = \frac{1}{30} = 0.025$ (magenta, third rightmost curve—) and $Pe = 1.25$ i.e. $h = \frac{1}{n+1} = \frac{1}{40} = 0.0125$ (black, fourth rightmost curve—).

tering near $t = 1$ is not quite as bad for the artificial diffusion method shown in figure 6.4 as for the simple method shown in figure 6.3b. For example, it has been possible for the artificial diffusion method to show all graphs all the way to 1, whereas we omitted the last line for the last two graphs for the simple method. This situation is explained by the fact that the simple method is second order convergent, whereas the artificial viscosity method is only linearly convergent since $-\epsilon\,(1+Pe)\,\delta_0^2 u_i = u_i'' + \mathcal{O}(h)$ (or alternatively $\delta_- u_i = u_i' + \mathcal{O}(h)$). There are ways to improve the rate of convergence while still avoiding the oscillations by adding smaller amounts of artificial diffusion than $-\frac{\beta h}{2}$, but we shall not get into the matter here ■ .

Remedy for the oscillatory numerical solutions to (6.88) with finite elements: In direct analogy with the difference approach, the oscillations can be avoided in the finite element method simply by replacing ϵ by $\epsilon(1 + \frac{\beta h}{2})$ ■ . ■

Chapter 7

Higher dimensions

7.1 Higher dimensions – FDM's for Elliptic PDE's

The general rule of thumb for finite difference methods for higher dimensional problems is: Just add indices! Instead of having a one dimensional computational domain we now have a multi dimensional domain and we need one new index for each new dimension. Most of the novelties are apparent already in two dimensions. The jump to even higher dimensions gives problems with for example geometrical representation and imagination of the constructions. Further, higher dimensions give practical problems with the computing time, which is expressed in various ways like the term Dimensional Death: To compute with for example 100 nodal points in each direction a one dimensional implicit method will give a system of 100 equations with 100 unknowns to solve, which is a piece of cake, while a similar 2 dimensional problem will give 10.000 equations, a 3 dimensional 1.000.000 and a 4 dimensional 100.000.000 equations with the same number of unknowns which is out of reach even for super computers.

All this means that in more than two dimensions, one must be more careful to avoid errors from "not fully comprehending geometrically what one is doing" and more thorough to get everything optimized for fast computing times. Otherwise there are no major difficulties arising with respect to what is found in 2 dimensions. In order to simplify the presentation, we shall hence restrict ourselves to the 2 dimensional case when doing examples in more than one dimension.

In one dimension, we used t for a generic point in the computational domain I and for the FDM's we defined a set of nodal points $x_1, \ldots, x_n \in I$. In two dimensions we shall use (x, y) for a generic point in the computa-

tional domain Ω and for the FDM's a two dimensional set of nodal points $(x_1, y_1), \ldots, (x_N, y_N)$. Like we in one dimension mainly considered uniform subdivisions where $x_i = x_1 + (i-1)h$, $i = 1, \ldots, n$, we shall in two dimensions generally restrict to Uniform, Cartesian Subdivisions $\{(x_i, y_j)\}_{i=1,\ldots,n_x, j=1,\ldots,n_y}$ where $x_i = x_1 + (i-1)h_x$, $i = 1, \ldots, n_x$ and $y_j = y_1 + (j-1)h_y$, $j = 1, \ldots, n_y$, i.e. $N = n_x n_y$. Often, we shall even be able to take $h_x = h_y$ and use Uniform, Square Subdivisions as indicated in figure 7.1. Obviously, with uniform

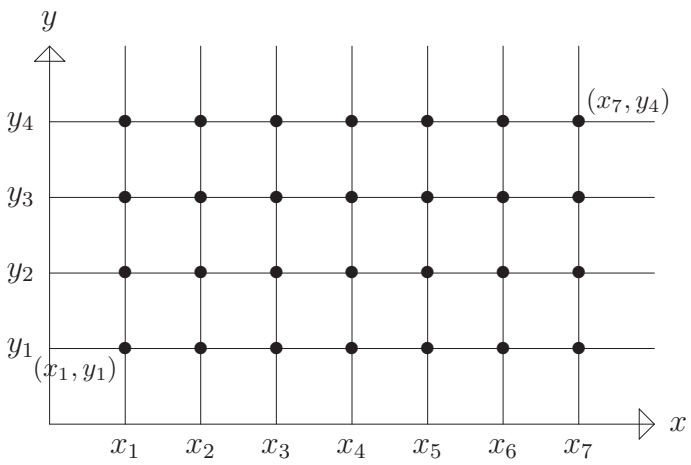


Figure 7.1: A uniform Cartesian subdivision

Cartesian subdivisions, the computational domain Ω had better be rectangular. When this is not the case, it spells problems. It is possible to do something, but it falls outside the scope of this text to consider the details. Instead we can refer to section 3.1.3 on finite element methods, where it is noted that the problems with non rectangular domains are greatly diminished.

We start considering the prototypical 2 dimensional elliptic PDE (see (1.6) on page 12).

Example 7.1 The Poisson problem in 2 dimensions

$$\text{Find } u \in \mathcal{C}^2(\bar{\Omega}): \; -\Delta u(x,y) = f(x,y) \; \forall (x,y) \in \Omega, \; u = 0 \text{ on } \partial\Omega, \tag{7.1}$$

where

$$\Delta u(x,y) = u_{xx}(x,y) + u_{yy}(x,y) = \frac{\partial^2 u}{\partial x^2}(x,y) + \frac{\partial^2 u}{\partial y^2}(x,y) \tag{7.2}$$

and Ω is some 2 dimensional, rectangular domain. The standard finite difference method for this problem is based on the central second order finite

difference operators:

$$\delta_{0,x}^2 \tilde{U}_{i,j} = \frac{1}{h_x^2}\left(\tilde{U}_{i-1,j} - 2\tilde{U}_{i,j} + \tilde{U}_{i+1,j}\right), \tag{7.3}$$
$$\delta_{0,y}^2 \tilde{U}_{i,j} = \frac{1}{h_y^2}\left(\tilde{U}_{i,j-1} - 2\tilde{U}_{i,j} + \tilde{U}_{i,j+1}\right).$$

Here $\tilde{U}_{i,j}$ serves as our approximation to $u_{i,j} = u(x_i, y_j)$ and we take $f_{i,j} = f(x_i, y_j)$, $i = 1, \ldots, n_x$, $j = 1, \ldots, n_y$. We then approximate (7.1) by

$$-(\delta_{0,x}^2 + \delta_{0,y}^2)\tilde{U}_{i,j} = f_{i,j}, \ i = 2, \ldots, n_x - 1, \ j = 2, \ldots, n_y - 1, \tag{7.4}$$
$$\tilde{U}_{1,j} = \tilde{U}_{n_x,j} = \tilde{U}_{i,1} = \tilde{U}_{i,n_y} = 0, \ i = 1, \ldots, n_x, \ j = 1, \ldots, n_y.$$

■

The geometrical picture of the index combination participating in the numerical approximation of the PDE in a generic point (x_i, y_j) is denoted the stencil of the numerical method. For example the stencil of the central second order finite difference approximation to the Poisson problem described in example 7.1 above is shown in figure 7.2.

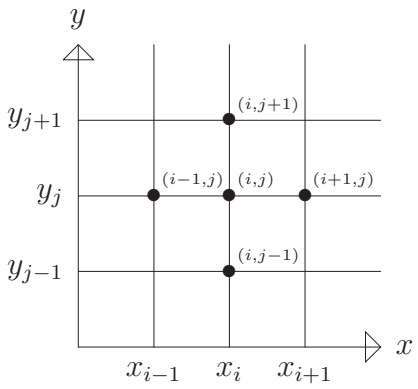


Figure 7.2: The 5 point stencil for the standard 2nd order central FDM for the poisson problem in 2D.

■ To be continued ...

Bibliography

[1] Fritz John. *Partial Differential Equations, 4th ed.* Springer, Applied Mathematical Sciences Vol. 1, 1982.

[2] Jeffery M. Cooper. *Introduction to partial differential equations with MATLAB.* Birkhäuser, 1998.

[3] Garrett Birkhoff and Gian-Carlo Rota. *Ordinary differential equations.* Blaisdell Publishing Company, 1969.

[4] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical Mathematics.* Springer, TAM 37, 2000.

[5] David Kincaid and Ward Cheney. *Numerical Analysis, 2nd (3rd) ed.* Brooks/Cole, 1996 (2002).

[6] Ivo Babuska. The connection between the finite difference like methods and the methods based on initial value problems for ode. In A. K. Aziz, editor, *Numerical Solution of Boundary Value Problems for ODE's*, pages 149–176. Academic Press, 1975.

[7] Ivo Babuska and V. Majer. The factorization method for two point boundary value problems for ode's and its relation to the finite element method. In Anthony Miller, editor, *Proceedings of the Centre for Mathematical Analysis, Australian National University. Contributions of Mathematical Analysis to the Numerical Solution of Partial Differential Equations*, pages 71–92, 1984.

[8] Marinela Lentini, Michael R. Osborne, and Robert D. Russell. The close relationships between methods for solving two-point boundary value problems. *SIAM Journal of Numerical Analysis*, 22(2):280–309, April 1985.

[9] J. W. Thomas. *Numerical Partial Differential Equations.* TAM 22 and 33. Springer, 1995 and 1999.

[10] J. D. Lambert. *Numerical methods for ordinary differential systems.* Wiley, 1991.

[11] Daniele Funaro. *Spectral elements for transport-dominated equations.* Springer, Lecture Notes in Computational Science and Engineering 1, 1997.

[12] Claes Johnson. *Numerical Solutions of partial differential equations by the finite element method.* Cambridge University Press, 1987.

[13] Philippe G. Ciarlet. *The Finite Element Method for Elliptic Problems.* North Holland, 1980.

[14] Susanne C. Brenner and L. Ridgway Scott. *The Mathematical Theory of Finite Element Methods, 2nd ed.* Springer, TAM 15, 2002.

[15] P. G. Ciarlet and J. L. Lions (Editors). *Handbook of numerical analysis, Volume II, Finite element methods (Part 1).* North Holland, 1991.

[16] A. Quarteroni and A. Valli. *Numerical approximation of partial differential equations.* Springer, 1994.

[17] G. W. Stewart. *Introduction to Matrix Computations.* Academic Press, 1973.

Index