Numerical integration of systems arising from the method of lines

Einar Magerøy

February 19, 1997

Department of Mathematical Sciences
The Norwegian University of Science and Technology

Abstract

This report surveys the method of lines as a method for solving partial differential equations. This method involves discretising of all independent variables except one, and integration of the resulting system of ordinary differential equations in the remaining variable. An introduction to space discretisation is given, and different Runge–Kutta methods and linear multi step methods are considered with emphasis on their linear stability properties.

Preface

This report is a result of my final studies at the Department of Mathematical Sciences, Norwegian University of Technology and Science (NTNU) in Trondheim. The writing has been done at the Department of Mathematics, University of Auckland.

The topic of this report developed through a wish to combine two interesting fields of numerical analysis, the solution of ordinary and partial differential equations. All the different aspects of the method of lines covers such a vast area that this report is by no means, a complete survey of the method. But I hope through this work to have collected some of the aspects important for applying the method of lines to partial differential equations.

I would like to thank John C. Butcher who willingly accepted to supervise me here at the University of Auckland. He has given me support and advice with my work here and also been very helpful with practical details during my stay. I would also like to thank Anne Kværnø, who has been my superviser at NTNU, and Syvert P. Nørsett who introduced me to John C. Butcher, and thus made this stay in Auckland possible. Also thanks to the "Runge–Kutta Club" for our weekly workshop and lunch sessions.

Auckland, February 19, 1997
Einar Magerøy

Contents

Chapter 1

The method of lines

1.1 Introduction

The method of lines (MOL) is a technique for solving partial differential equations (PDEs). Usually the method is used on equations involving a time variable t and one or more space variables $x_1, ..., x_m$. They are solved by converting them to a system of ordinary differential equations (ODEs) in t.

The part of the equations involving the space variables is discretised, giving us a system of ODEs approximating the PDE. This system can then be integrated directly with a standard ODE code.

It might be more efficient to solve a PDE by a method specially constructed to suit the problem, but the MOL usually enables us to solve quite general and complicated PDEs relatively easily and with acceptable efficiency. The MOL is also attractive since we can use the theoretical knowledge from ODEs to solve PDEs, and powerful ODE solvers are readily available.

Following [8], the connection between partial and ordinary differential equations was already known to Lagrange in 1759. He studied the propagation of sound by modelling the medium as a sequence of mass points, using known results in elasticity. Thus, he obtained a system of ordinary differential equations

$$\begin{aligned} y_1^{''} &= K^2(-2y_1 + y_2), \\ y_2^{''} &= K^2(y_1 - 2y_2 + y_3), \\ &\vdots \\ y_n^{''} &= K^2(y_{n-1} - 2y_n). \end{aligned} \tag{1.1}$$

Lagrange observed that by modelling d'Alemberts equation

$$\frac{\partial^2 u}{\partial t^2} = a^2 \frac{\partial^2 u}{\partial x^2},$$

with a finite number of mass points attached to a light string , the same equations (1.1) are found.

In [8] it is also noted that Fourier in 1807, motivated by the problem of heat conduction, modelled a rod as a sequence of molecules with temperature y_i. For the total gain of energy to the i-th molecule he obtained the equation

$$y_i^{'} = K^2(y_{i-1} - 2y_i + y_{i+1}), \quad i = 1, \ldots, N. \tag{1.2}$$

By taking N larger and larger in 1.2 he obtained the heat equation

$$\frac{\partial u}{\partial t} = a^2 \frac{\partial^2 u}{\partial x^2},$$

thus using an "inverse" of the method of lines.

But the MOL as a numerical method was first applied by Rothe [15] in 1930. He used the method on the parabolic equation

$$u_{xx} = R(x,t)u_t + S(x,t,u), \tag{1.3}$$

with $R > 0$, $0 \leq x \leq 1$, $0 \leq t \leq T$. He discretised the time variables in (1.3) and approximated the equation by the scheme

$$u_{n+1}^{''} = R(x, t_{n+1}) \frac{u_{n+1} - u_n}{h} + S(x, t_{n+1}, u_n), \quad t_n = nh. \tag{1.4}$$

In (1.4) the integration of the ODEs is along lines parallel to the x-axis, and this is called a *transversal* scheme. Later development has been most concentrated on *longitudinal* schemes, in which the lines are parallel to the t-axis. That is, discretising the space variables leaving the time variables continuous. Longitudinal schemes lead to initial value problems, while transversal schemes lead to boundary value problems.

Until the 1960s the method was most popular in Russia (the Soviet Union). In 1965 Liskovets wrote a review paper on the MOL [13]. He mentions that the MOL can be extended to a much wider area than parabolic equations. The first examples were the MOL applied to elliptic problems in mathematical physics (1939), and the MOL with longitudinal schemes on hyperbolic equations (1954). Neither is the MOL restricted to classical problems (parabolic, elliptic or hyperbolic) or linear problems, and the method is not restricted to rectangular Cartesian coordinates. Liskovets mentions that the method has been used in oblique Cartesian, polar, cylindrical and spherical coordinates.

The method of lines is applicable to a wide range of problems in many different areas. Schiesser [16] has an appendix on problems in physics, fluid dynamics, mathematics, reactor models, automatic control and more.

The method of lines consists of mainly two tasks, discretisation of the space variables and the time-integration of the system of ODEs. The emphasis in this report will be on numerical methods for the time-integration. In chapter 1 we will look at an example of the method of lines and some background material. Chapter 2 gives an introduction to different methods for semi discretisation of PDEs. Different numerical schemes for the time integration of large initial value problems are considered in chapter 3. Some numerical experiments are found in chapter 4. Explicit Runge–Kutta methods with extended real stability intervals are compared with backward differentiation formulas. Explicit and implicit linear multistep methods are compared on a problem not regarded as stiff, and a Rosenbrock method is compared with the backward differentiation formula on a non linear problem.

1.2 Converting a parabolic PDE to a system of ODEs

Consider the heat equation on a light string

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, \tag{1.5}$$

with $0 < x < 1$, $t \geq 0$, the initial condition $u(x,0) = u_0(x)$ and the Dirichlet boundary conditions $u(0,t) = g_0(t)$, $u(1,t) = g_1(t)$. This models the development of the temperature u in a string with initial temperature given by $u_0(x)$, when the temperature at the ends g_0, g_1 is known.

If the temperature in the ends are unknown but we know the heat flux out of the string, we have the Neumann boundary conditions $u_x(0,t) = \Gamma_0(t)$ and $u_x(1,t) = \Gamma_1(t)$.

We use the approximation

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{1}{h^2}\left(u(x-h,t) - 2u(x,t) + u(x+h,t)\right) \tag{1.6}$$

and keep $x = x_0$ fixed. Thus we obtain the ordinary differential equation

$$\frac{du(x_0,t)}{dt} \approx \frac{1}{h^2}(u(x_0-h,t) - 2u(x_0,t) + u(x_0+h,t)). \tag{1.7}$$

We discretise the space coordinate by subdividing the interval $0 < x < 1$ in $N+1$ equal subintervals of length $h = 1/(N+1)$. With the time coordinate still continuous, we can write down equation (1.7) at every point $x_i = ih$, $i = 1...N$. This gives us a discretised version of equation (1.5), as a system of N ordinary differential equations

$$\frac{d}{dt}\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_{N-1} \\ V_N \end{bmatrix} = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix}\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_{N-1} \\ V_N \end{bmatrix} + \frac{1}{h^2}\begin{bmatrix} g_0(t) \\ 0 \\ \vdots \\ 0 \\ g_1(t) \end{bmatrix}, \tag{1.8}$$

where $V_i(t) = V(x_i,t) \approx U(x_i,t)$ and $V_i(0) = g(x_i)$ for $i = 1, \ldots, N$.

The approximate solution of (1.5) can now be found by applying established numerical integrators to (1.8).

However, the coefficient matrix in (1.8) has eigenvalues[1]

$$\lambda_j = \frac{1}{h^2}\left(-2 + 2\cos\frac{j\pi}{N+1}\right) \quad j = 1\ldots, N.$$

Thus $\lambda_j \in [-4/(h^2), 0]$ and for small values of h, the eigenvalues will be spread over a large interval on the negative real axis, making (1.8) stiff. Systems derived from the method of lines are often stiff, and combined with the often very large size of the systems this will afflict what integrators are suitable to use.

With the Neumann boundary conditions we do not have the values of V_0 and V_{N+1}. But by approximating the boundary condition $u_x(0,t) = \Gamma_0(t)$ with the one-sided approximation

$$\frac{-3V_0 + 4V_1 - V_2}{2h} = \Gamma_0(t), \tag{1.9}$$

we can solve for V_0 and substitute into

$$\frac{V_0 - 2V_1 + V_2}{h^2} = \frac{2}{3h^2}(V_2 - V_1) - \frac{2}{3h}\Gamma_0(t).$$

Proceeding in the same manner for the boundary at $x = 1$ we find the new system of ODEs

$$\frac{d}{dt}\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_{N-1} \\ V_N \end{bmatrix} = \frac{1}{h^2}\begin{bmatrix} -\frac{2}{3} & \frac{2}{3} & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & \frac{2}{3} & -\frac{2}{3} \end{bmatrix}\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_{N-1} \\ V_N \end{bmatrix} - \frac{2}{3h}\begin{bmatrix} \Gamma_0(t) \\ 0 \\ \vdots \\ 0 \\ \Gamma_1(t) \end{bmatrix}.$$

V_0 can be obtained from (1.9), and similar for V_{N+1}.

1.3 Notation and some useful results

Throughout this report it is assumed that $y : \mathbb{R} \mapsto \mathbb{R}^m$ and $F : \mathbb{R}^m \mapsto \mathbb{R}^m$ in the ordinary differential equation

$$\dot{y} = F(y).$$

We will use h for the mesh size in space discretisation and τ for the step size in the time integration.

The direct product (or Kronecker product) of $A \in \mathbb{C}^{m\times m}$ and $B \in \mathbb{C}^{n\times n}$ is defined by

$$(A \otimes B) = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1m}B \\ a_{21}B & a_{22}B & \cdots & a_{2m}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \ldots & a_{mm}B \end{bmatrix}$$

[1] See section 1.3

where $(A \otimes B) \in \mathbb{C}^{mn \times mn}$. If $A, C \in \mathbb{C}^{m \times m}$ and $B, D \in \mathbb{C}^{n \times n}$ then

$$(A \otimes B)(C \otimes D) = (AC \otimes BD) \tag{1.10}$$
$$(A \otimes B)^{-1} = (A^{-1} \otimes B^{-1}). \tag{1.11}$$

We define the divided differences recursively by

$$G[n] = F_n = F(x_n),$$
$$G[n, n-1, \ldots, n-i] = \frac{G[n, \ldots, n-i+1] - G[n-1, \ldots, n-i]}{x_n - x_{n-i}}.$$

The interpolation polynomial I_q through the points (x_{n-j}, F_{n-j}), $j = 1, \ldots, q$ is then given by

$$I_q = G[n] + (x - x_n)G[n, n-1] + \cdots + (x - x_n) \cdots (x - x_{n-q+1})G[n, \cdots, n-q].$$

The Gershgorin Circle theorem is useful for obtaining qualitative information about the eigenvalues of a matrix.

Theorem 1.1 (The Gershgorin Circle Theorem) *Let $\sigma(A) = \{\lambda_1, \ldots, \lambda_n\}$ be the set of eigenvalues of $A \in \mathbb{C}^{n \times n}$. Then*

$$\lambda(A) \subseteq \bigcup_{i=1}^{n} D_i,$$

where

$$D_i = \{z \in \mathbb{C}; |z - a_{ii}| \leq \sum_{j \neq i} |a_{ij}|\}.$$

Proof. For any eigenvalue λ with eigenvector $x = (x_1, \ldots, x_n)$ we have

$$(\lambda I - A)x = 0$$
$$(\lambda - a_{ii})x_i = \sum_{j \neq i} a_{ij} x_j$$
$$|\lambda - a_{ii}| \leq \sum_{j \neq i} |a_{ij}| \frac{|x_j|}{|x_i|}$$

Let $x_i = \max_j |x_j|$ and we obtain $|\lambda - a_{ii}| \leq \sum_{j \neq i} |a_{ij}|$.

$\square$

Following [18] we provide a theorem about the eigenvalues of a tridiagonal matrix.

Theorem 1.2 *The tridiagonal $N \times N$ matrix*

$$A = \begin{bmatrix} a & b & & & \\ c & a & b & & \\ & \ddots & \ddots & \ddots & \\ & & c & a & b \\ & & & c & a \end{bmatrix}$$

with $b, c \in \mathbb{R}$ has eigenvalues

$$\lambda_s = a + 2\sqrt{bc}\cos\left(\frac{s\pi}{N+1}\right) \qquad s = 1, \ldots, N.$$

Proof. Suppose $c < 0, b > 0$ and let D be a diagonal matrix where $[D]_{ii} = (|b/c|)^{(i-1)/2}$ for $k = 1, \ldots, N$. Using the similarity transformation

$$DAD^{-1} = \mathrm{diag}\{a\} + \sqrt{|bc|}\ \mathrm{tridiag}\{-1, 0, 1\},$$

and that tridiag$\{-1, 0, 1\}$ has eigenvectors $v = (i\sin\theta, \ldots, i^m \sin m\theta, \ldots, i^N \sin N\theta)^T$ corresponding to the eigenvalues $2i\cos\theta$, $\theta = s\pi/(N+1)$ and $s = 1, \ldots, N$ the result follows.

Suppose $b > 0, c > 0$. Then

$$DAD^{-1} = \mathrm{diag}\{a\} + \sqrt{bc}\ \mathrm{tridiag}\{1, 0, 1\},$$

where tridiag$\{1, 0, 1\}$ has eigenvalues $2\cos\theta$. The cases $b < 0, c < 0$ and $b < 0, c > 0$ will change the sign of the eigenvalues, but the result is still valid since $\theta(s)$ is symmetrically distributed around $\pi/2$.

□

Chapter 2

Space discretisation

Most often the equations considered with the method of lines are on the form

$$\frac{\partial u(t, x_1, \ldots, x_m)}{\partial t} = \mathcal{L}(u), \tag{2.1}$$

with the initial conditions

$$u(0, x_1, \ldots, x_m) = u_0(x_1, \ldots, x_m),$$

and Neumann, Dirichlet or mixed boundary conditions. Here $u = [u_1, \ldots, u_n]^T$ and $\mathcal{L}$ is a differential operator in $x_1, \ldots, x_m$. If the t-derivatives are of higher order, say k, the method of lines gives us ODEs of order k. These can be solved by introducing new variables and reducing the kth-order ODEs to first order ODEs.

We will look at finite differences and finite elements as methods for semi discretising of the partial differential equation (2.1).

2.1 Finite differences

Applying the method of lines to (2.1), we replace the continuous variables x_j by discrete values $x_j^i, i = 0, \ldots, N+1$. Using finite differences for the space discretisation we replace the differential operator $\mathcal{L}$ on $x_1, \ldots, x_m$by a difference operator $\mathbf{L}_i$ on the grid defined by x_j^i. To each point of this grid we can now associate an ordinary differential equation for each element in u, and the system of PDEs has been converted to n systems of ODEs (n the number of elements in u). Each system will be of dimension N^m. To derive the difference operator $\mathbf{L}_i$ we need algebraic approximations to derivatives.

2.1.1 Functions of one variable

We want to find an approximation $\widehat{f^{(n)}}(x_0) \approx f^{(n)}(x_0)$, by a linear combination of the function $f(x)$ evaluated in m neighbouring points $x_i, i = -k, \ldots, l$ where

$m = l + k + 1$. Let $f_0 = f(x_0)$, with $d_i = x_i - x_0$ we obtain

$$\begin{aligned} \sum_{i=-k}^{l} a_i f(x_0 + d_i) &= \sum_{i=-k}^{l} a_i \sum_{j=0}^{\infty} \frac{d_i^j}{j!} f^{(j)}(x_0) \\ &= \sum_{i=-k}^{l} a_i \left(f_0 + d_i f_o' + \frac{d_i^2}{2} f_0'' + \cdots + \frac{d_i^n}{n!} f_0^{(n)} + \cdots \right) \end{aligned} \tag{2.2}$$

We impose the condition that the coefficient corresponding to $f_0^{(n)}$ must be equal to one, and as many as possible of the other coefficients corresponding to $f_0^{(j)}$, $j = 0, 1, \ldots$, $j \neq n$ must be equal to zero. With m points available for the approximation, we have m parameters we can use to match terms from the Taylor expansion. Thus we obtain

$$\sum_{i=-k}^{l} a_i \frac{d_i^j}{j!} = \begin{cases} 0, & 0 < j \leq n, \\ 1, & j = n, \\ 0, & n < j \leq m-1. \end{cases} \tag{2.3}$$

By inserting the coefficients defined by (2.3) into (2.2) we obtain

$$f^{(n)}(x_0) = \widehat{f}^n(x_0) + R(x_0),$$

where

$$\widehat{f}^n(x_0) = \sum_{i=-k}^{l} a_i f(x_0 + d_i), \quad R(x_0) = \sum_{i=-k}^{l} a_i \sum_{j=0}^{\infty} \frac{d_i^j}{j!} f^{(j)}(x_0).$$

If f is analytic we obtain from the mean value theorem that the error in our approxiamtion is given by the first non zero term in R for some x. With $d_i = ih$ we need n parameters to get an approximation of $\mathcal{O}(1)$, and therefore our approximation is $\mathcal{O}(h^{m-n})$. If our points are symmetrically distributed around x_0 we will find that $a_{-i} = -a_i$ in approximations where n is odd, and that $a_{-i} = a_i$ where n is even. So for even n the first term in $R(x)$ disappears, and we obtain an one order higher approximation. Thus we find

$$f^{(n)}(x_0) = \sum_{i=-k}^{l} a_i f(x_0 + ih) + \mathcal{O}(h^{\Delta}), \tag{2.4}$$

where $\Delta = m - n$ for one sided approximations and centred approximations for uneven n, and $\Delta = m - n + 1$ for centred approximations for even n.

Example 2.1 *We consider the case when the points x_i are equally spaced, and we can write $d_i = ih$. We will find an approximation of $f'(x_0)$ by using the five points $f(x_0 + ih), i = -2, \ldots, 2$.*

We use conditions 2.3

$$\sum_{i=-2}^{2} a_i \frac{(ih)^j}{j!} = \begin{cases} 0, & j = 0, \\ 1, & j = 1, \\ 0, & j \leq 4, \end{cases}$$

which we can write as

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ -2h & -h & 0 & h & 2h \\ 2h^2 & \frac{1}{2}h^2 & 0 & \frac{1}{2}h^2 & 2h^2 \\ -\frac{4}{3}h^3 & -\frac{1}{6}h^3 & 0 & \frac{1}{6}h^3 & \frac{4}{3}h^3 \\ \frac{2}{3}h^4 & \frac{1}{24}h^4 & 0 & \frac{1}{24}h^4 & \frac{2}{3}h^4 \end{pmatrix} \begin{pmatrix} a_{-2} \\ a_{-1} \\ a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Solving this for $a_{-2}, \ldots, a_2$ we obtain the approximation

$$f'(x_0) \approx \frac{f(x_0 - 2h) - 8f(x_0 - h) + 8f(x_0 + h) - f(x_0 + 2h)}{12h}. \tag{2.5}$$

Tables of finite difference formulas on equally spaced one dimensional grids can be found in [7].

2.1.2 Functions of several variables

We can proceed in a similar manner to the one dimensional case. We define the index vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, where the components are non-negative integers and define

$$|\alpha| = \alpha_1 + \alpha_2 + \cdots + \alpha_n.$$

We use the notation

$$D^\alpha = \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \cdots x_n^{\alpha_n}}.$$

Suppose $x = (x_1, x_2, \ldots, x_n)$. We want to find an approximation $\mathcal{D}^\alpha f(\bar{x}) \approx D^\alpha f(\bar{x})$, where $\bar{x} = (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)$, by a linear combination of the the function $f(x)$ evaluated in $\bar{x}$ ant the m neighbouring points $\bar{x} + \xi_i$, $i = 1, \ldots, m$ where $\xi_i = (\xi_{1i}, \xi_{2i}, \ldots, \xi_{ni})$. As with the one dimensional case we write down the Taylor expansion of our linear combination

$$\sum_{i=1}^{m} a_i f(\bar{x} + \xi_i) = \sum_{i=1}^{m} a_i \sum_{j=0}^{\infty} \left[\frac{1}{j!} (\xi_i \nabla)^j f(x) \right]_{x=\bar{x}}, \tag{2.6}$$

where

$$(\xi_i \nabla)^j = \left(\xi_{1i} \frac{\partial}{\partial x_1} + \xi_{2i} \frac{\partial}{\partial x_2} + \cdots + \xi_{ni} \frac{\partial}{\partial x_n} \right)^j.$$

By setting the coefficient corresponding to $D^\alpha f(x)$ equal to one and as many as possible of the other coefficients equal to zero, we obtain

$$D^\alpha f(\bar{x}) = \mathcal{D}^\alpha f(\bar{x}) + R(\bar{x}),$$

where

$$\mathcal{D}^\alpha f(\bar{x}) = \sum_{i=1}^{m} a_i f(\bar{x} + \xi_i), \quad R(\bar{x}) = \sum_{i=1}^{m} a_i \sum_{j=m}^{\infty} \left[\frac{1}{j!} (\xi_i \nabla)^j f(x) \right]_{x=\bar{x}}.$$

As before, if f is analytic the error in the approximation is given by the first non zero term of R for som x.

Example 2.2 *With $\alpha = (1,1)$ we want to find an approximation for $D^\alpha f(\bar{x}) = f_{x_1 x_2}(\bar{x}_1, \bar{x}_2)$. Equation (2.6) now reads*

$$\mathcal{D}(\bar{x}) = \sum_{i=1}^{m} a_i \left(f + \xi_{1i} f_{x_1} + \xi_{2i} f_{x_2} + \frac{\xi_{1i}^2}{2} f_{x_1 x_1} + \xi_{1i}\xi_{2i} f_{x_1 x_2} + \frac{\xi_{2i}^2}{2} f_{x_2 x_2} + \right.$$
$$\left. \frac{\xi_{1i}^3}{3!} f_{3x_1} + \frac{\xi_{1i}^2 \xi_{2i}}{2} f_{2x_1 x_2} + \frac{\xi_{1i}\xi_{2i}^2}{2} f_{x_1 2x_2} + \frac{\xi_{2i}^3}{3!} f_{3x_2} + \cdots \right). \quad (2.7)$$

Setting the coefficients corresponding to $f_{x_1 x_2}$ equal to one and the others to zero we obtain

$$\begin{array}{llll}
\sum a_i = 0 & & & \\
\sum a_i \xi_{1i} = 0 & \sum a_i \xi_{2i} = 0 & & \\
\sum a_i \xi_{1i}^2/2 = 0 & \sum a_i \xi_{1i}\xi_{2i} = 1 & \sum a_i \xi_{2i}^2/2 = 0 & \\
\sum a_i \xi_{1i}^3/3! = 0 & \sum a_i \xi_{1i}^2 \xi_{2i}/2 = 0 & \sum a_i \xi_{1i}\xi_{2i}^2/2 = 0 & \sum a_i \xi_{2i}^3/3! = 0
\end{array}$$

If we assume the same mesh size h in the x_1 and x_2 directions, some of the equations are linearly dependent. Choosing the points numbered $1, \ldots, 5$ to approximate f_{x_1,x_2} in the node marked i (see Figure 2.1) we obtain

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ -1 & 1 & 0 & -1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ h^2 & -h^2 & 0 & -h^2 & h^2 \\ -1 & -1 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

From this we find

$$D^\alpha f_i \approx \frac{1}{4h^2}(f_{i-N-1} - f_{i-N+1} - f_{i+N-1} + f_{i+N+1}),$$

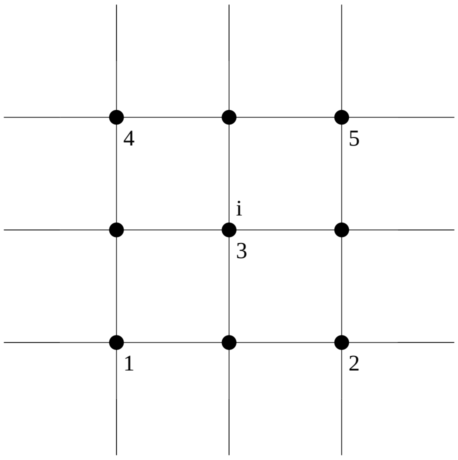

Figure 2.1: The nodes chosen to approximate $D^\alpha f_i$.

where $f_i = f(x_i)$. Note that in this case the same result is found by applying the one dimensional approximation for f' first in the x_1 direction, and then in the x_2 direction. With the notation

$$\widehat{f'_{x_1}} f_i = \frac{1}{2h}(f_{i+1} - f_{i-1}) \approx \frac{\partial}{\partial x_1} f(x_{1i}, x_{2i}),$$

we obtain

$$\begin{aligned} D^\alpha f_i &\approx \widehat{f'_{x_2}}\widehat{f'_{x_1}} f_i \\ &= \widehat{f'_{x_2}}\left(\frac{1}{2h}(f_{i+1} - f_{i-1})\right) \\ &= \frac{1}{4h^2}(f_{i-N-1} - f_{i-N+1} - f_{i+N-1} + f_{i+N+1}). \end{aligned}$$

We have in this example only considered a Cartesian grid. Finite difference formulas can of course be found for other grid types like for example hexagonal grids, grids in polar coordinates or different types of grids in three or more dimensions. The procedure is the same as with a Cartesian grid. We write down equation (2.6) for the points we want to use in our approximation. By imposing the condition that the coefficient corresponding to our derivative is equal to one, and that as many as possible of the others are equal to zero we obtain our approximation.

2.1.3 Handling points outside the boundary

If we use finite differences requiring use of more than just the closest neighbouring points, we will need values of points outside the boundary there the equation is defined. This problem can be handled in several ways.

Imposing periodicity

We impose periodicity on the solution in each of the space variables, letting one period equal the range of the interval where the variable is defined. This can be regarded as mapping of the interval of one variable onto a circle. In the one dimensional case this is equivalent to solving the equation on a cylinder instead of a strip in the $x-t$ plane.

A prerequisite for this method is that the boundary conditions are equal in both ends of an interval. This gives us $u(x_0) = u(x_{N+1})$, $u(x_1) = u(x_{N+2})$ and so on. The band structure of the Jacobian will be destroyed.

Example 2.3 *Consider the equation*

$$u_t = u_x, \quad 0 \le x \le 1, \quad t > 0, \tag{2.8}$$

with $u(0,t) = u(1,t) = g(t)$. We use the five point difference formula 2.5 and obtain the discretised equations

$$\dot{U}_i = \frac{1}{12h}(U_{i-2} - 8U_{i-1} + 8U_{i+1} - U_{i+2}).$$

Imposing periodicity on the solution we can use $U_{-1} = U_{N+1}$ and we obtain the system

$$\dot{U} = AU + G, \tag{2.9}$$

where

$$A = \frac{1}{12h}\begin{bmatrix} 0 & 8 & -1 & & & & 1 \\ -8 & 0 & 8 & -1 & & & \\ 1 & -8 & 0 & 8 & -1 & & \\ & & & \ddots & & & \\ & & 1 & -8 & 0 & 8 & -1 \\ & & & 1 & -8 & 0 & 8 \\ 1 & & & & 1 & -8 & 0 \end{bmatrix}, \quad G = \begin{bmatrix} -8g \\ g \\ 0 \\ \vdots \\ 0 \\ -g \\ 8g \end{bmatrix}.$$

Fictitious points

Another approach is to extrapolate the solution to points outside the boundary (fictitious points), and use these extrapolations to eliminate the fictitious points from the difference formula. If the difference formula is of order p we use $p+1$ points in the interpolation.

Example 2.4 *Again we consider equation (2.8) with the same discretisation. We find the polynomial interpolating $U_0, \dots, U_4$ ($U_{N-3}, \dots, U_{N+1}$) ant extrapolate the values U_{-1} (U_{N+2}). Thus*

$$\begin{aligned} U_{-1} &= 5U_0 - 10U_1 + 10U_2 - 5U_3 + U_4, \\ U_{N+2} &= 5U_{N-3} - 10U_{N-2} + 10U_{N-1} - 5U_{N+1} + U_{N+2}, \end{aligned}$$

and substitute this in the difference equations for U_1 and U_N. Thus we obtain equation (2.9) with

$$A = \frac{1}{12h}\begin{bmatrix} -10 & 18 & -6 & 4 & & & & \\ -8 & 0 & 8 & -1 & & & & \\ 1 & -8 & 0 & 8 & -1 & & & \\ & & & \ddots & & & & \\ & & 1 & -8 & 0 & 8 & -1 \\ & & & 1 & -8 & 0 & 8 \\ & & & 4 & -6 & 18 & -10 \end{bmatrix}, \quad G = \begin{bmatrix} -3g_0 \\ g_0 \\ 0 \\ \vdots \\ 0 \\ -g_0 \\ 3g_0 \end{bmatrix}.$$

2.1.4 Boundary conditions involving derivatives

Now consider boundary conditions of the mixed type

$$\alpha(x,t)u + \beta(x,t)\frac{\partial u}{\partial \eta} = \gamma(x,t)$$

on the boundary $\partial\Omega$, where $\alpha(x,t) > 0$, $\beta(x,t) > 0$ for x on $\partial\Omega$. Here $\partial/\partial\eta$ refers to the derivative normal to the boundary $\partial\Omega$. This means that we do not have a given value of u on the boundary, which we will need in the finite differences for the space variables.

For simplicity we consider the one dimensional case, where the boundary conditions at the left end are given by $\alpha(t)u + \beta(t)u_x(0,t) = \gamma(t)$. We can now use a second order one sided approximation in the boundary condition to obtain

$$\alpha U_0 + \beta\frac{-3U_0 + 4U_1 - U_2}{2h} = \gamma.$$

Solving for U_0 we obtain

$$U_0 = \frac{4\beta U_1 - \beta U_2 - 2h\gamma}{3\beta - 2h\alpha},$$

which we can insert into our difference formulas.

Another approach is to introduce a fictitious point U_{-1} and use a second order central approximation in the boundary condition

$$\alpha U_0 + \beta\frac{U_1 - U_{-1}}{2h} = \gamma.$$

We solve this for U_{-1} and obtain

$$U_{-1} = U_1 + 2h\frac{\alpha}{\beta}U_0 - 2h\gamma.$$

We can now introduce a new equation for $\dot{U}_0$ there the value for U_{-1} can be used in the finite differences. This gives us one more equation in our system.

2.1.5 Graded and Adaptive grids

A *graded* grid is a non-uniform fixed grid. We use *a priori* information about the solution to put more grid points, where the solution is expected to change rapidly. This is useful if we know that rapid changes in the solution is confined to a constant region of the space where the equation is defined.

The easiest implementation is obtained if the mesh sizes of two adjoining areas differ with a factor of two. For some of the finite differences in the area with the smallest mesh size we will need solution values between the mesh points in the area with the larger mesh size. These values can then be obtained by interpolation.

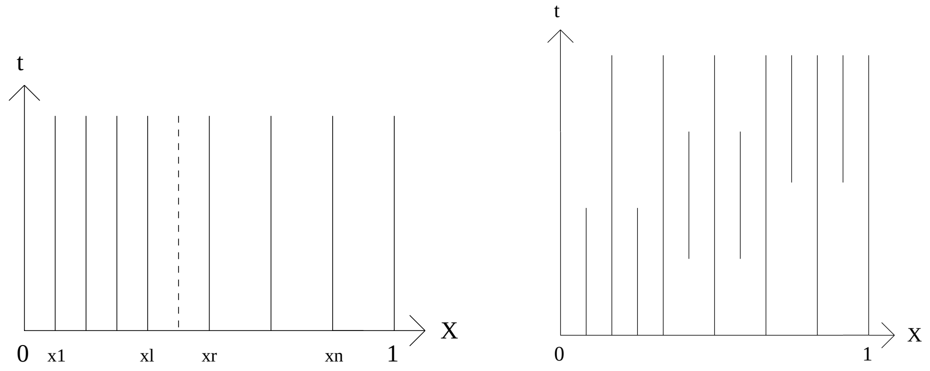

Figure 2.2: Graded grid and adaptive grid.

An *adaptive* grid develops in time. We start with a fixed grid, and introduce more grid points between the fixed grid points when necessary. If the 2nd derivative in one space variable in a point rises above a certain threshold, we insert an extra grid point in between its neighbouring points. If desirable the fixed grid can be graded. Adaptive grids are useful when regions of rapid change in the solution moves in time.

Adaptive grids are more complicated to implement than fixed grids, but they are effective in the sense that fewer grid points (and fewer ODEs) are necessary for a certain accuracy in the solution. They will will cope better with the development of sharp fronts (shocks).

2.1.6 Examples of space discretisation with finite differences

Elliptic PDEs

Consider the Laplace equation in two dimensions

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \tag{2.10}$$

where $0 \leq x \leq 1$, $0 \leq y \leq 1$ and $u(x,0) = u(x,1) = 0$ and $u(0,y) = f_0(y)$, $u(1,y) = f_1(y)$.

We discretise one of the variables. Following the same procedure as before we get

$$\frac{d^2U_j}{dx^2} = -\frac{U_{j-1} - 2U_j + U_{j+1}}{h^2}, \quad j = 1, ..., N, \tag{2.11}$$

where the boundary conditions are given with $U_0 = U_{N+1} = 0$.

This is a system of second order ODEs. With $dU_j/dx = V_j$ we can write 2.11 as

$$\frac{d}{dx}W = AW, \tag{2.12}$$

where

$$W = \begin{bmatrix} U_1 \\ \vdots \\ U_N \\ V_1 \\ \vdots \\ V_N \end{bmatrix}, \quad A = \begin{bmatrix} 0 & I \\ A_1 & 0 \end{bmatrix},$$

with

$$A_1 = \frac{1}{h^2} \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & -2 & 1 \end{bmatrix}.$$

This system can be solved by a shooting technique. That is, choose initial values for $V_1, ..., V_N$ and integrate the system to obtain the values for $U_1, ..., U_N$ at $x = 1$. Information from this solution can be used for another integration of the system with new initial values for $V_1, ..., V_N$. Continue this process until the boundary conditions for $U_1, ..., U_N$ is satisfactory fulfilled at $x = 1$.

Graded grid

We use the graded grid in Figure 2.2 and consider the effects it will have on

$$\dot{U}_i = \frac{U_{i-1} - 2U_i + U_{i+1}}{h^2}, \quad i = 1, \ldots, n.$$

For $\dot{U}_l$ we lack the value $U_{l+h} = U^*$ (on the dotted line in the figure). We obtain this value from interpolation between U_{l-1}, U_l, U_r and U_{r+1}

$$U^* = -\frac{1}{5}U_{l-1} + \frac{3}{4}U_l + \frac{2}{3}U_r - \frac{1}{20}U_{r+1}.$$

Thus we find $\dot{U} = 1/h^2 AU$ with

$$A = \begin{bmatrix} A_1 & & \\ & C & \\ & & A_2 \end{bmatrix} U,$$

where $C = (a_{l,l-1}, \ldots, a_{l,l+2}) = (-1/5, 3/4, 1/2, -1/20)$.

$$A_1 = \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & -2 & 1 \end{bmatrix}$$

and $A_2 = \frac{1}{4} A_1$.

2 and 3 dimensional problems

We consider the heat equation on the unit square

$$u_t = \Delta u = u_{xx} + u_{yy}. \tag{2.13}$$

For simplicity we assume $u = 0$ on the boundary. We discretise the rectangle with a uniform mesh with mesh size $h = 1/(N+1)$ and number the nodes in *natural* order (see Figure 2.3).

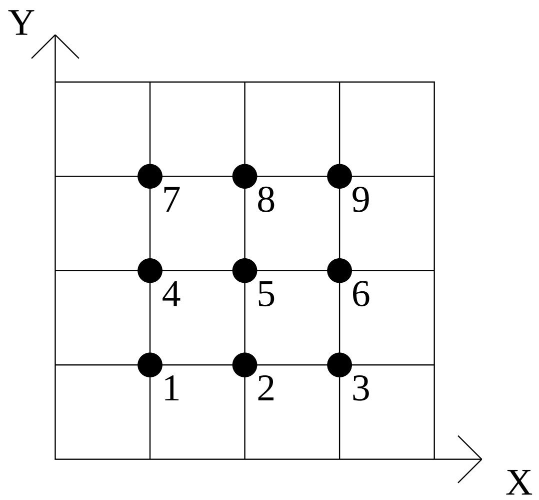

Figure 2.3: Natural ordering of nodes.

We denote $V_i(t) \approx u(x_i, y_i, t),\ i = 1, \ldots, N^2$. We can express x_i and y_i in terms of the mesh size h and the index i.

$$\begin{aligned} x_i &= (((i-1) \texttt{ mod } N) + 1)h, \\ y_i &= (((i-1) \texttt{ div } N) + 1)h. \end{aligned}$$

In each node we use the second order central approximation for u_{xx} and u_{yy}. For each inner node (a node with no neighbours on the boundary) we obtain

$$\frac{d}{dt}V_i = \frac{1}{h^2}(V_{i-N} + V_{i-1} + V_{i+1} + V_{i+N} - 4V_i).$$

This gives the equation

$$\frac{d}{dt}V = \frac{1}{h^2}AV, \tag{2.14}$$

where

$$A = \begin{bmatrix} B & I & & & \\ I & B & I & & \\ & & \ddots & & \\ & & I & B & I \\ & & & I & B \end{bmatrix}, \quad B = \begin{bmatrix} -4 & 1 & & & \\ 1 & -4 & 1 & & \\ & & \ddots & & \\ & & 1 & -4 & 1 \\ & & & 1 & -4 \end{bmatrix},$$

with $A \in \mathbb{R}^{N^2 \times N^2}$, $B \in \mathbb{R}^{N \times N}$ and I is the identity matrix.

With the same example on the unit cube in 3 dimensions, we obtain for each inner node

$$\frac{d}{dt}V_i = \frac{1}{h^2}(V_{i-N^2} + V_{i-N} + V_{i-1} + V_{i+1} + V_{i+N} + V_{i+N^2} - 6V_i),$$

giving us the equation

$$\frac{d}{dt}V = \frac{1}{h^2}CV, \tag{2.15}$$

where

$$C = \begin{bmatrix} A & I & & & \\ I & A & I & & \\ & & \ddots & & \\ & & I & A & I \\ & & & I & A \end{bmatrix},$$

with $C \in \mathbb{R}^{N^3 \times N^3}$ and $A \in \mathbb{R}^{N^2 \times N^2}$ is as in 2.14 but with -6 on the diagonal.

We denote $V_i(t) \approx u(x_i, y_i, z_i, t),\ i = 1, \ldots, N^3$, where x_i, y_i and z_i are now given by

$$\begin{aligned} x_i &= (((i-1) \texttt{ mod } N) + 1)h, \\ y_i &= (((i-1) \texttt{ div } N) + 1)h, \\ z_i &= (((i-1) \texttt{ div } N^2) + 1)h. \end{aligned}$$

2.2 Finite elements

Another approach for semi-discretisation of a partial differential equation is to use the finite element method[1] for the semi discretisation of the PDE (2.1).

[1] See for example [11]

The discretisation process starts with a reformulation of the problem as an equivalent variational problem. Then the variational problem is discretised using finite elements, and the solution is approximated by a linear combination of basis functions (orthogonal, often piecewise polynomial functions).

With the finite element method, as compared to finite differences, it is easier to cope with complicated geometry, general boundary conditions and variable or non-linear material properties.

2.2.1 Variational formulation of a model problem

We define a region $\Omega \in \mathbb{R}^n$ and the Hilbert space

$$L_2(\Omega) = \{v : v \text{ is defined on } \Omega;\ \int_\Omega v^2\, d\Omega < \infty\}.$$

And we introduce

$$H_0^1(\Omega) = \{v : v, v^{'} \in L_2(\Omega);\ v = 0 \text{ on } \partial\Omega\},$$

and the scalar product

$$(v, w) = \int_\Omega vw\, d\Omega.$$

Let $V = H_0^1(\Omega)$. We also define $a : V \times V \to \mathbb{R}$

$$a(v, w) = \int_\Omega \nabla v \nabla w\, d\Omega.$$

We can now find a variational formulation of a model problem

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega \times I \\ u &= 0 \quad \text{on } \partial\Omega \times I \\ u(\cdot, 0) &= u_0 \end{aligned} \tag{2.16}$$

Let $v \in V$ and multiply the equation for a given t by v, and integrate over Ω:

$$\int_\Omega -\Delta uv\, d\Omega = \int_\Omega fv\, d\Omega.$$

Using Green's formula and the fact that $u = v = 0$ on $\partial\Omega$, we obtain the variational formulation of 2.16: Find $u \in V$ such that

$$a(u, v) = (f, v), \quad \forall\, v \in V,\ t \in I. \tag{2.17}$$

Let V_h be a finite-dimensional subspace of V with orthogonal basis functions $\{\varphi_1, \ldots, \varphi_M\}$. A discretisation of 2.17 is then: Find $u_h \in V_h$ such that

$$a(u_h, v) = (f, v), \quad \forall\, v \in V_h,\ t \in I. \tag{2.18}$$

Since $u_h \in V_h$ we can write

$$u_h = \sum_{i=1}^{M} \xi_i \varphi_i, \quad t \in I.$$

Inserting this in 2.18 and using $v = \varphi_j$ we find

$$a(\varphi_i, \varphi_j)\xi_i = (f, \varphi_j), \quad i, j = 1, \ldots, M.$$

2.2.2 Example with semi discretisation by finite elements

Consider the heat equation with source

$$\begin{aligned} u_t - \Delta u &= f \quad \text{in } \Omega \times I \\ u &= 0 \quad \text{on } \partial\Omega \times I \\ u(x, y, 0) &= u_0(x, y), \end{aligned} \tag{2.19}$$

where Ω is the unit square.

We multiply the equation for a given t by $v \in V$, and integrate over Ω. Carrying on as with the model problem in section 2.2.1 , we obtain the discretised variational equation

$$(\dot{u}_h, v) + a(u_h, v) = (f, v), \quad \forall\, v \in V_h,\ t \in I. \tag{2.20}$$

We write u_h as a linear combination of the basis functions and let $v = \varphi_j$ in (2.20). Thus we obtain

$$\begin{aligned} B\dot{\xi}(t) + A\xi(t) &= F(t) \quad t \in I, \\ B\xi(0) &= U^0, \end{aligned} \tag{2.21}$$

where $[B]_{ij} = (\varphi_i, \varphi j)$, $[A]_{ij} = a(\varphi_i, \varphi j)$, $F_i(t) = (f(t), \varphi_i)$ and $U_i^0 = (u_0, \varphi_i)$. Often B is called the *mass* matrix and A the *stiffness* matrix.

We make a uniform triangularisation of Ω with T_h as the set of triangles, and define the piecewise planar basis functions (see Figure 2.4) by

$$\varphi_i(\text{vertex}_j) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}.$$

Thus the support of φ_i consists of the triangles where node i is a vertex.

The components of the stiffness matrix A is in practice computed by summing the contributions from the triangles in T_h

$$a(\varphi_i, \varphi_j) = \sum_k a_k(\varphi_i, \varphi_j) = \sum_k \int_k \nabla\varphi_i \nabla\varphi_j \, dA = \sum_k \int \nabla\psi_i \nabla\psi_j \, dA.$$

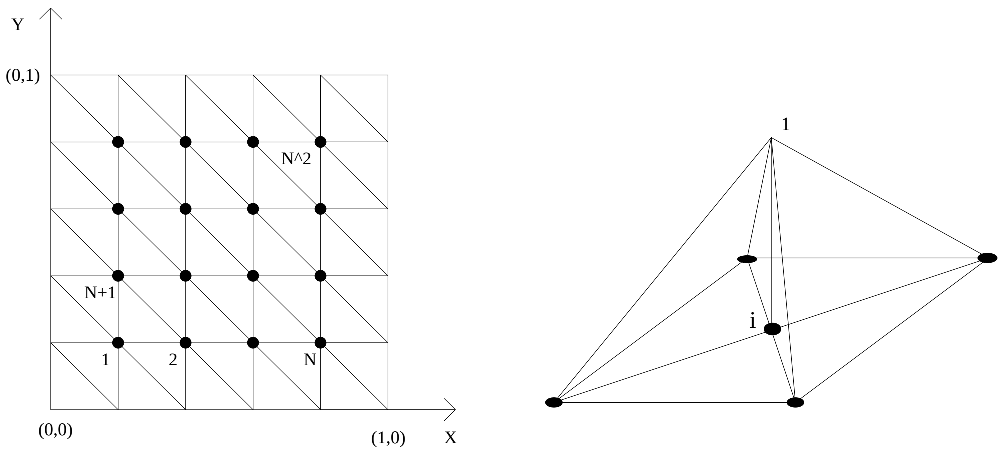

Figure 2.4: The triangularisation T_h of Ω and the basis function φ_i.

We have here denoted the restriction of φ_i on the triangle k as ψ_i. The same procedure can be followed for the mass matrix B. We note that $a_k(\varphi_i, \varphi_j) = a(\psi_i, \psi_j) = 0$ unless node i and j are both vertices of the triangle k.

We consider the triangle $\{(0,0),(h,0),(0,h)\}$ with nodes 1,2 and 3 as vertices. The restrictions of $\varphi_1, \varphi_2, \varphi_3$ on the triangle will then be

$$\psi_1 = \frac{1}{h}(h - x - y), \quad \psi_2 = \frac{1}{h}x, \quad \psi_3 = \frac{1}{h}y. \tag{2.22}$$

Computing (ψ_i, ψ_j) and $a(\psi_i, \psi_j)$ for $i, j = 1..3$ we obtain the *element* mass and stiffness matrices

$$B_k = \frac{h^2}{12}\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix}, \quad A_k = \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}.$$

These matrices are the same for all triangles k where all vertices are nodes (and not boundary points), except from their indexes which are in accordance with the node numbers of their vertices.

Now we can compute B and A from B_k and A_k. To find row i in A find all triangles with node i as a vertex and sum the contributions from row i in A_k. From this we obtain

$$B = \frac{h^2}{12}\begin{bmatrix} B_1 & I & & & \\ I & B_1 & I & & \\ & & \ddots & & \\ & & I & B_1 & I \\ & & & I & B_1 \end{bmatrix}, \quad B_1 = \begin{bmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & & \ddots & & \\ & & 1 & 4 & 1 \\ & & & 1 & 4 \end{bmatrix}$$

and

$$A = \begin{bmatrix} A_1 & -I & & & \\ -I & A_1 & -I & & \\ & & \ddots & & \\ & & -I & A_1 & -I \\ & & & -I & A_1 \end{bmatrix}, \quad A_1 = \begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & & \ddots & & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{bmatrix},$$

where A and B are $N^2 \times N^2$ matrices, and A_1 and B_1 are $N \times N$ matrices.

Chapter 3

Integration

The different discretisation schemes in the last chapter will produce a system of ordinary differential equations of the form

$$\begin{aligned} &B(t)\dot{y}(t) = F(t, y), \quad t \in [t_0, T], \\ &y(t_0) = y_0. \end{aligned} \tag{3.1}$$

For some spatial discretisation schemes $B(t) = I$ and (3.1) is reduced to

$$\begin{aligned} &\dot{y}(t) = F(t, y), \quad t \in [t_0, T], \\ &y(t_0) = y_0. \end{aligned} \tag{3.2}$$

For schemes where $B(t) \neq I$ we have two possibilities

1. $B(t)$ is nonsingular and well conditioned for all t.
   In this case, the numerical methods for solving (3.2) can easily be extended to solve equation (3.1)

2. $B(t)$ is singular or ill conditioned.
   Equation (3.1) can then best be solved with numerical methods for differential algebraic equations (DAEs).

The problems associated with the numerical solution of DAEs will not be considered in this report, and we will therefore assume that $B(t) = I$ or that if $B(t) \neq I$ it is non singular and well conditioned for all t. In the latter case it is possible to multiply by B^{-1} to obtain a system of the form (3.2). But semi discretisation of PDEs usually leads to systems with a banded structure which can be exploited in the integration. Multiplying by B^{-1} will destroy this structure and it is therefore better to adapt the numerical method to the form (3.1).

3.1 Stiffness

Consider the linear problem

$$\dot{y}(t) = Ay(t) + \varphi(t), \tag{3.3}$$

where $y, \varphi \in \mathbb{R}^m$, and $A \in \mathbb{R}^{m\times m}$ is a constant matrix with eigenvalues $\lambda_i \in \mathbb{C}$, $i = 1, \ldots, m$, with corresponding eigenvectors $\chi_i \in \mathbb{C}^m$.

The general solution of (3.3) is

$$y(t) = \sum_{i=1}^{m} c_i e^{\lambda_i t}\chi_i + \psi(t), \tag{3.4}$$

where c_i are arbitrary constants and $\psi(x)$ is a particular integral. Let us assume that Re $\lambda_i < 0 \Rightarrow y(t) \to \psi(t)$ as $t \to \infty$. We can then call the homogeneous solution for the transient solution and the particular solution for the steady state solution.

If $|\text{Re } \lambda_i|$ is large, then the corresponding part of the solution $c_i e^{\lambda_i t}\chi_i$ will decay quickly as t increases and is called a fast transient. If $|\text{Re } \lambda_i|$ is small the corresponding part of the solution will decay slowly and is called a slow transient.

To reach the steady state solution by numerical integration of (3.3) we must integrate until the slowest transient is negligible. If we use an explicit method we might observe that the step size is restricted by stability and not by accuracy. The stability properties of a numerical method[1] is dependent on the size of $\tau\lambda_i$. Thus, to ensure stability a large $\max|\text{Re } \lambda_i|$ implies small step length, and if $\min|\text{Re } \lambda_i|$ is small we will have to integrate for a long time after the fast transient has died, with an excessively small step length. So by using a method with finite region of stability, we can be forced to integrate an increasingly smooth solution with intolerable small step size. The problem is then said to be *stiff*. But if the steady state solution itself varies rapidly, we will expect small step lengths and the problem can hardly be regarded as stiff.

Obviously the eigenvalues of the Jacobian are important for the concept of stiffness. But the dimension of the system, the smoothness of the solution and the interval of integration are also important. Many different "definitions" of stiffness has been made and two examples are cited here.

Definition 3.1 (van der Houwen[2]) *An equation of the form $\frac{d}{dt}y = F(y)$ is called stiff at (t_0, y_0) with respect to the interval $[t_0, t_1]$ when, in the neighbourhood of y_0, the eigenvalues λ of the Jacobian matrix $J(y)$ of the system are situated in two sets of points C_0 and C_1, such that for each two points $\lambda_0 \in C_0$ and $\lambda_1 \in C_1$*

$$\exp(\text{Re } \lambda_1(t_1 - t_0)) \ll \exp(\text{Re } \lambda_0(t_1 - t_0)).$$

[1] See sections 3.2.1 and 3.3.1

[2] cf. [19], p. 10.

Definition 3.2 (Lambert[3]) *If a numerical method with a finite region of absolute stability, applied to a system with any initial conditions, is forced to use in a certain interval of integration a step length which is excessively small in relation to the smoothness of the exact solution in that interval, then the system is said to be stiff in that interval.*

Van der Houwen's definition is more stringent than $\max \mathrm{Re}\,|\lambda| \gg \min \mathrm{Re}\,|\lambda|$ and with this definition a semi discretisation of a parabolic PDE will not be regarded as stiff, since the eigenvalues are not clustered around two points. Lambert's definition relates to what happens in practice when you try to integrate a stiff system with an explicit integrator.

From a geometrical point of view, stiffness is connected to the behaviour of neighbouring solution curves compared with the solution itself. In a interval where a system is stiff, neighbouring solution curves will approach the solution curve at a rate that is large compared to the rate the solution varies in the interval. Thus $F(t, y) = Ay + \varphi$ is ill conditioned and a small change in y gives a large change in $F(t, y)$.

The above statements on stiffness can be extended to non-linear problems by piecewise linearising the problem. Thus if we "freeze" the Jacobian at a point and use information from the frozen Jacobian to describe stiffness locally in the non-linear problem. This will not always work, and can only be used as an indication of the behaviour of the non-linear system.

3.2 Linear multi step methods

A linear k-step method with constant step size τ can be written as

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = \tau \sum_{j=0}^{k} \beta_j F_{n+j}, \tag{3.5}$$

where α_j and β_j are constants and

$$\alpha_k = 1, \quad |\alpha_0| + |\beta_0| \neq 0. \tag{3.6}$$

We define the *first* and *second characteristic polynomial* associated with equation (3.5) by

$$\rho(\xi) \equiv \sum_{j=0}^{k} \alpha_j \xi^j, \quad \sigma(\xi) \equiv \sum_{j=0}^{k} \beta_j \xi^j. \tag{3.7}$$

We can use (3.7) to write (3.5) in the alternative form

$$\rho(E) y_n = h\sigma(E) F_n, \tag{3.8}$$

[3] cf. [12], p. 220.

where E is the forward shift operator. The method is explicit if $\deg(\sigma) < k$ ($\beta_k = 0$), and implicit if $\deg(\sigma) = k$ ($\beta_k \neq 0$).

Among the general class (3.5) of linear multi step methods there are several sub-classes, which can be characterised by the characteristic polynomials (3.7).

$\rho(\xi) = \xi^k - \xi^{k-1}$
: Methods of Adams type. If these methods have maximum order with $\deg(\sigma) < k$ they are called Adams-Bashforth methods, and with $\deg(\sigma) = k$ they are called Adams-Moulton methods.

$\rho(\xi) = \xi^k - \xi^{k-2}$
: Methods of this class with maximum order are called Nyström methods if $\deg(\sigma) < k$, and Generalised Milne-Simpson methods if $\deg(\sigma) = k$.

$\sigma(\xi) = \beta^k \xi^k$
: With maximum order these implicit methods are called backward differentiation formulas (BDFs).

Definition 3.3 *The linear differential operator ℓ associated with the linear multi step method (3.5) is defined by*

$$\ell[z(t);\tau] = \sum_{j=0}^{k} [\alpha_j z(t+j\tau) - \tau\beta_j z'(t+j\tau)],$$

where $z(t) \in [a,b]$ is an arbitrary function.

If we choose $z(t)$ to be as smooth as required and expand $z(t+j\tau)$ and $z^{'}(t+j\tau)$ about t we obtain

$$\ell[z(t);\tau] = C_0 z(t) + C_1 \tau z'(t) + \ldots + C_n \tau^n z^{(n)}(t) + \ldots \tag{3.9}$$

where

$$\begin{aligned} C_0 &= \textstyle\sum_{j=0}^{k} \alpha_j \equiv \rho(1) \\ C_1 &= \textstyle\sum_{j=0}^{k} (j\alpha_j - \beta_j) \equiv \rho'(1) - \sigma(1) \\ C_n &= \textstyle\sum_{j=0}^{k} \left[\frac{1}{n!} j^n \alpha_j - \frac{1}{(n-1)!} j^{n-1} \beta_j\right] \quad n = 2,3,\ldots \end{aligned} \tag{3.10}$$

Definition 3.4 *The linear multi step method (3.5) has* order p *and* error constant C_{p+1} *if $C_0 = C_1 = \ldots = C_p = 0$, $C_{p+1} \neq 0$, where the Cs are given in (3.10).*

One drawback with linear multi step methods is that it is not easy to change the step size during the integration. Equation (3.5) requires equidistant points. So to change step size we must use interpolation to obtain starting values with the new step size, or we must construct methods accepting variable step sizes.

3.2.1 Linear stability

We consider the linear system

$$\dot{y} = Ay, \tag{3.11}$$

where the eigenvalues $\lambda_1, \ldots, \lambda_s$ of A are assumed distinct. A is then diagonalisable and there exists a Q such that

$$Q^{-1}AQ = \Lambda,$$

where $\Lambda = \mathrm{diag}\{\lambda_1, \ldots, \lambda_s\}$. With the transformation $y = Q\eta$ we obtain the decoupled system

$$\dot{\eta} = \Lambda\eta.$$

To examine the behaviour of numerical methods on the linear system (3.11) it is therefore sufficient to consider the scalar test equation,

$$\dot{y} = \lambda y, \quad \lambda \in \mathbb{C}. \tag{3.12}$$

If $\mathrm{Re}(\lambda) < 0$ then $\|y(t)\| \to 0$ as $t \to \infty$. We want to find under what conditions the numerical solution has the same behaviour.

We apply the linear multi step method (3.5) to the test equation (3.12) and obtain the difference system

$$\sum_{j=0}^{k} (\alpha_j - \tau\beta_j\lambda) y_{n+j} = 0. \tag{3.13}$$

We set $\mu = \tau\lambda$, $y_j = \xi^j$ and divide (3.13) by ξ^n. We obtain the characteristic equation

$$\sum_{j=0}^{k} (\alpha_j - \mu\beta_j)\xi^j = \rho(\xi) - \mu\sigma(\xi) = 0, \tag{3.14}$$

and we find that the general solution of (3.13) is

$$y_n = \sum_{s=1}^{k} \kappa_s \xi_s^n,$$

where κ_s are arbitrary complex constants and $\xi_s(\mu)$, $s = 1, \ldots, k$ are the roots of the *stability polynomial* in (3.14)

$$\pi(\xi, \mu) = \rho(\xi) - \mu\sigma(\xi). \tag{3.15}$$

Clearly, to get the behaviour $\|y_n\| \to 0$ as $n \to \infty$ of the numerical solution, we must demand that all the roots $\xi_s(\mu)$, of the stability polynomial $\pi(\xi, \mu)$ satisfies $|\xi_s(\mu)| < 1$, $s = 1, \ldots, k$.

Definition 3.5 *The linear multi step method (3.5) has a stability region S in the complex plane, where*

$$S = \left\{ \mu \in \mathbb{C};\ \begin{array}{l} |\xi_s(\mu)| \leq 1 \textit{ for all single roots of (3.15)} \\ |\xi_s(\mu)| < 1 \textit{ for all multiple roots} \end{array} \right\}.$$

If $S \supset \mathbb{C}^- = \{\mu \in \mathbb{C}; \mathrm{Re}\ \mu \leq 0\}$, the method is called A-stable.
If $S \supset \mathbb{C}(\alpha) = \{\mu \in \mathbb{C}; |arg(-\mu)| \leq \alpha\, \mu \neq 0\}$ the method is $A(\alpha)$ stable.
If $S \supset \mathbb{C}_s\{\mu \in \mathbb{C}; \mathrm{Re}\ \mu \leq D\} \cup \{\mu \in \mathbb{C}; \mathrm{Im}\ \mu < c\}$ the method is called stiffly stable.

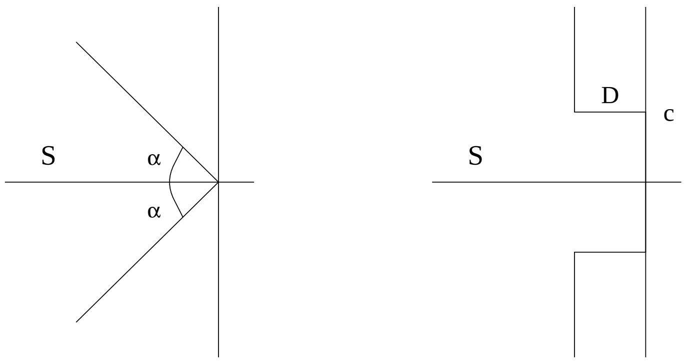

Figure 3.1: $A(\alpha)$ stability and stiff stability.

All numerical solutions of the linear multi step method (3.5) applied to a linear equation $\dot{y} = Ay$, where A is diagonalisable are bounded if $\mu = \tau\lambda \in S$ for all eigenvalues λ of A.

Theorem 3.6 *If a linear multi step method is A-stable then*

$$\mathrm{Re}\ \left(\frac{\rho(\xi)}{\sigma(\xi)}\right) > 0 \quad \textit{for} \quad |\xi| > 1. \tag{3.16}$$

Proof. If a method is A-stable then Re $\mu \leq 0 \Rightarrow |\xi| \leq 1$ for all roots ξ of $\pi(\xi, \mu)$. This is equivalent to the statement $|\xi| > 1 \Rightarrow$ Re $\mu > 0$ and we obtain 3.16 since $\mu = \rho(\xi)/\sigma(\xi)$. □

For irreducible methods ($\sigma(\xi) \neq 0$) we also have that (3.16) implies A-stability[4].

The boundary of S can be found by the *boundary locus technique.* Let ∂S be the curve in the complex plane where one of the roots of $\pi(\xi, \mu)$ has modulus 1. Since the roots of a polynomial are given as a continuous function of its coefficients the boundary of S must consist of the parts of ∂S where no roots have modulus greater than 1. The curve ∂S is found from solving for μ in $\pi(exp(i\theta), \mu) = 0$, and thus we obtain

$$\mu(\theta) = \frac{\rho(e^{i\theta})}{\sigma(e^{i\theta})}.$$

[4]See complete proof in [9] p. 265.

The linear stability analysis is only valid in the context of linear constant coefficient systems. However, the results are often used on non-linear systems. But they can only be used as an indication of the behaviour in the non-linear case, and does not always make sense.

3.2.2 Implicit linear multi step methods

For implicit methods, we must for each time step solve the implicit equation

$$g(y_{n+k}) \equiv \alpha_k y_{n+k} - \tau\beta_k F_{n+k} + \sum_{j=0}^{k-1} (\alpha_j y_{n+j} - \tau\beta_j F_{n+j}) = 0, \tag{3.17}$$

for y_{n+k}. If $B(t) \neq I$ in (3.1) this will give us the modification

$$g(y_{n+k}) \equiv \alpha_k B(t_{n+k}) y_{n+k} - \tau\beta_k F_{n+k} + \sum_{j=0}^{k-1} (\alpha_j B(t_{n+j}) y_{n+j} - \tau\beta_j F_{n+j}) = 0.$$

With a Newton-Raphson iteration scheme we obtain

$$\begin{aligned} J_g^{(l)} \delta^l &= -g_{n+k}^l, \\ y_{n+k}^{l+1} &= y_{n+k}^l + \delta^l, \end{aligned} \tag{3.18}$$

where $J_g^{(l)} = g'(y_{n+k}^{(l)}) = \alpha_k B - \tau\beta_k B J_F^{(l)}$ is the $m \times m$ Jacobian-matrix of g. Since the problems derived from the MOL will be large, the evaluation of J_F and the LU-factorisation of J_g will by far be the most expensive computations during the integration. Therefore simplified Newton methods where J_g is held constant as long as efficient are often used to save work on evaluation and factorisation of J_g.

Backward differentiation formulas

If we let p_n be the polynomial that interpolates the equally spaced points (t_{n-i}, y_{n-i}) for $i = 1, \ldots, k$ and determine y_n from the collocation condition $p_n'(t_n) = F(t_n, y_n)$ we obtain

$$\frac{1}{\tau} \sum_{i=0}^{k} \alpha_i y_{n-i} = F(t_n, y_n),$$

where τ is the step size and coefficients α_i are dependent on k only. The backward differentiation method is given by

$$\sum_{j=1}^{k} \frac{1}{j} \nabla^j y_{n+1} = \tau F_{n+1}, \tag{3.19}$$

where $\nabla y_n = y_n - y_{n-1}$.

For each time step (3.17) now becomes

$$g(y_{n+k}) \equiv \alpha_k y_{n+k} - \tau F_{n+k} + \sum_{j=0}^{k-1} \alpha_j y_{n+j} = 0. \tag{3.20}$$

With $\rho = \sum_{j=1}^{k} 1/j \xi^{k-j}(\xi - 1)^j$ and $\sigma = \xi^k$ we find that the root locus curves are given by

$$\mu(\theta) = \frac{\rho(e^{i\theta})}{\sigma(e^{i\theta})} = \sum_{j=1}^{k} \frac{1}{j}(1 - \frac{1}{e^{i\theta}})^j. \tag{3.21}$$

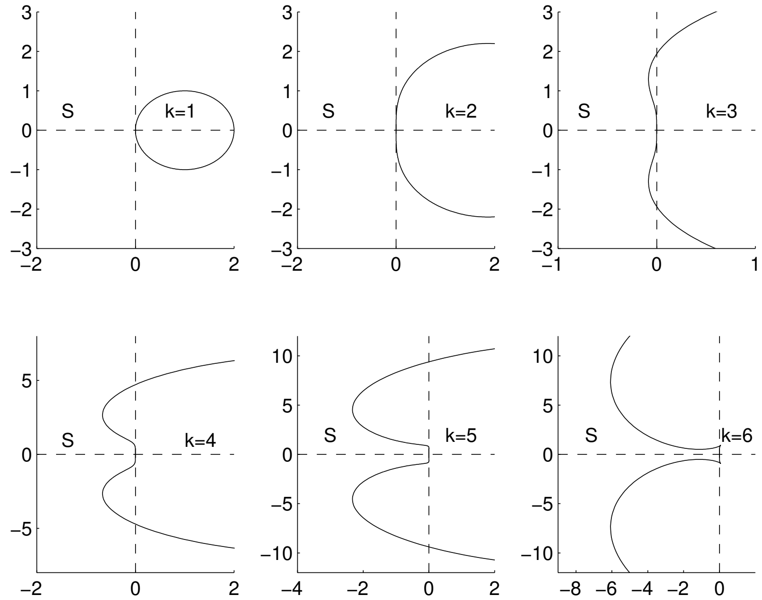

Figure 3.2: Stability regions of BDF methods.

Among the linear multi step methods the BDFs are the only with infinite stability regions (see Figure 3.2), a necessary property for methods that shall be used on stiff problems. But they are only A-stable for order 1 and 2, for order 3 to 6 the stability regions decreases. The characteristics of the stability regions are given in Table 3.1. Because of the restricted stability regions it is possible that a limitation on order may be experienced on certain problems.

Table 3.1: Stability properties of the BDF-methods.

k	1	2	3	4	5	6
α	90°	90°	86°	73°	52°	18°
D	0	0	0.08	0.67	2.33	6.08

Following [10] we will now consider three different variable step size BDF methods. To construct a BDF scheme accepting variable step sizes we drop the requirement that the points (t_n, y_n) must be equally spaced and by writing p_n in Lagrangian form we obtain

$$\frac{1}{\tau_n}\sum_{i=0}^{k}\alpha_{n,i}y_{n-i} = F(t_n, y_n), \tag{3.22}$$

where

$$\alpha_{n,i} = \tau_n l'_{n,i}(t_n) \quad \text{and} \quad l_{n,i}(t) = \prod_{j=0, j\neq i}^{k}\frac{t - t_{n-j}}{t_{n-i} - t_{n-j}}.$$

(3.22) is called the *variable coefficient* implementation of the BDF since $\alpha_{n,i}$ depends on $\tau_{n-1}, \ldots, \tau_{n-k+1}$. The Jacobian in the Newton iterations now becomes

$$J_g = \frac{\alpha_{n,0}}{\tau_n}I - J_F.$$

Since $\alpha_{n,0}$ will change for the $k-1$ first steps after a change of step size the simplified Newton iterations tends to be less effective with this implementation.

Another approach is to use the *fixed coefficient* (FC) implementation (3.19). When the step size is changed we use the values $(t_n, y_n), (t_n - \tau_n, y_{n-1}), (t_n - 2\tau_n, \tilde{y}_{n-2}), \ldots, (t_n - k\tau_n, \tilde{y}_{n-k})$, where $\tilde{y}_{n-2}, \ldots, \tilde{y}_{n-k}$ are obtained from interpolation between the past evenly spaced values $(t_{n-1}, y_{n-1}), (t_{n-1} - \tau_{n-1}, y_{n-2}), (t_{n-1} - 2\tau_{n-1}, \bar{y}_{n-3}), \ldots, (t_{n-1} - k\tau_{n-1}, \bar{y}_{n-k})$ used on the previous step. If $\tau_n = \tau_{n-1}$ it is not necessary to interpolate. We can write the method

$$\frac{1}{\tau_n}\left(\alpha_0 y_n + \alpha_1 y_{n-1} + \sum_{i=2}^{k}\alpha_i\tilde{y}_{n-i}\right) = F(t_n, y_n). \tag{3.23}$$

The Jacobian in the Newton iterations now becomes

$$J_g = \frac{\alpha_0}{\tau_n}I - J_F. \tag{3.24}$$

A re-evaluation of J_g is now only necessary when we change the step size and we can use the simplified Newton iterations more effectively. However, the fixed coefficient implementation is less stable than the variable coefficient implementation. To ensure stability the step size and order should not change for at least k steps after a step size change.

The *fixed leading coefficient* (FLC) implementation is also obtained by using the evenly spaced values $(t_n, y_n), (t_n - \tau_n, y_{n-1}), (t_n - 2\tau_n, \tilde{y}_{n-2}), \ldots, (t_n - k\tau_n, \tilde{y}_{n-k})$, but now they are interpolated from the computed values (t_{n-i}, y_{n-i}), $i = 1, \ldots, k+1$. This method can be written as

$$\frac{1}{\tau_n}\left(\alpha_0 y_n + \sum_{i=1}^{k}\hat{\alpha}_{n,i}\tilde{y}_{n-i}\right) = F(t_n, y_n). \tag{3.25}$$

$\hat{\alpha}_{n,i}$ is now dependent on past step sizes, but α_0 remains fixed and in the Newton iterations we obtain the same Jacobian (3.24) as with fixed coefficient implementation. In a way this method combines the best from the variable and fixed coefficient implementations. Since this methods uses "true" values to interpolate the necessary past values it does not suffer from the potential instability, and the Jacobian in the Newton iterations has the same form as in fixed coefficients enabling efficient use of simplified Newton iterations.

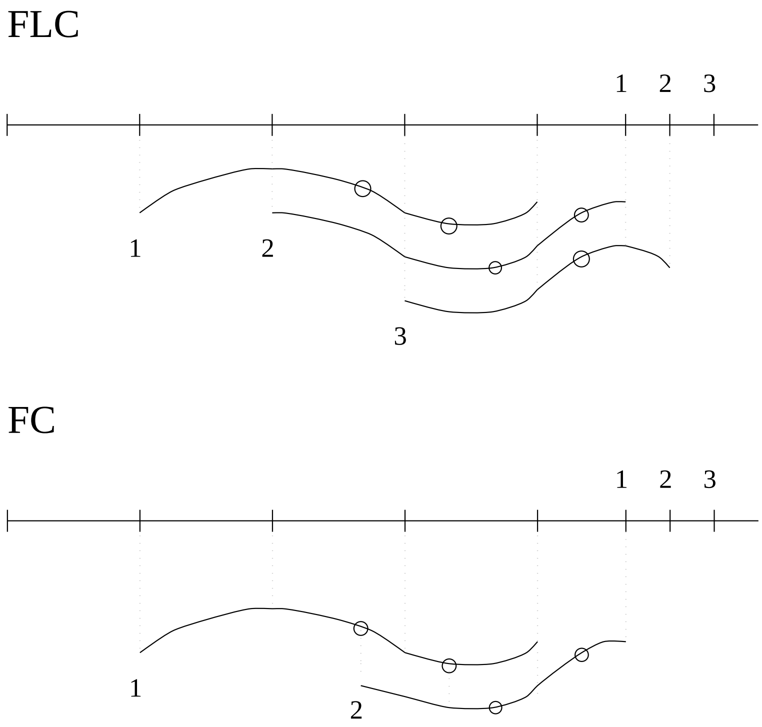

Figure 3.3: The interpolation to obtain evenly spaced past values in FLC and FC implementation of BDF.

Figure 3.3 illustrates the difference between the interpolation techniques used in FC and FLC implementation. To obtain the past values after a change in step size FLC interpolates between "true" past values (marked with dotted line in figure). The circles on the interpolation polynomials indicate the interpolated values used on that step. In the FC implementation the past values are obtained by interpolating the values used on the previous step. Therefore it is not necessary with any interpolation at the step marked 3.

3.3 One-step methods

With linear multi step methods we achieved higher order of accuracy by using information from earlier steps, but they retain the linearity with respect to y_{n+j} and F_{n+j}, $j = 1, \ldots, k$. This gives the local error an simple structure and we can estimate it easily, but because of the dependence of earlier steps it is difficult to change the step size.

RK-methods obtain higher accuracy by sacrificing the linearity, but they are on a one-step format. Thus it is easy to change step length, but the local error is complicated and making it more difficult to decide when to change the step size.

A general s-stage RK-method can be written on the form

$$\begin{aligned} y_{n+1} &= y_n + \tau \textstyle\sum_{i=1}^{s} b_i K_i, \\ K_i &= F(t_n + c_i\tau, y_n + \tau \textstyle\sum_{j=1}^{s} a_{ij} K_j), \end{aligned} \tag{3.26}$$

where a_{ij}, b_i and c_i characterises the method. Thus a method can be described by

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array},$$

where $[A]_{ij} = a_{ij}$, $[b]_i = b_i$ and $[c]_i = c_i$. We will always assume that the row-sum condition holds:

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

A non-autonomous system $\dot{y} = F(t, y)$ can always be converted to an autonomous system by setting $\dot{y}_{n+1} = 1$ and inserting y_{n+1} for t. We can therefore without loss of generality assume autonomous systems then working with systems of ODEs.

Alternatively the RK-method (3.26) can be written as

$$\begin{aligned} y_{n+1} &= y_n + \tau(b^T \otimes I_m) F(e \otimes t_n + e \otimes c\tau, Y), \\ Y &= e \otimes y_n + \tau(A \otimes I_m) F(e \otimes t_n + e \otimes c\tau, Y), \end{aligned} \tag{3.27}$$

where I_m is the m-dimensional identity matrix, $Y^T = (Y_1^T, \ldots, Y_s^T)^T$, $e = (1, \ldots, 1)^T$ has s elements and $F(Y) = (F(Y_1), \ldots, F(Y_s))^T$.

A RK-method is explicit if A is strictly lower triangular, otherwise the method is implicit.

3.3.1 Linear stability

We apply (3.27) to the test equation

$$y' = \lambda y,$$

setting $z = \tau\lambda$ we obtain

$$\begin{aligned} Y &= e \otimes y_n + zAY \\ y_{n+1} &= y_n + zb^T Y \end{aligned} . \tag{3.28}$$

Solving for Y we find $y_{n+1} = R(z)y_n$ where $R(z)$ is the stability polynomial given by

$$R(z) = 1 + zb^T(I - zA)^{-1}e. \tag{3.29}$$

Another approach to obtain the stability polynomial is to write (3.28) as the linear system

$$\begin{pmatrix} I - zA & 0 \\ -zb^T & 1 \end{pmatrix} \begin{pmatrix} Y \\ y_{n+1} \end{pmatrix} = y_n \begin{pmatrix} e \\ 1 \end{pmatrix}.$$

Applying Cramer's rule and solving for y_{n+1} we find that the denominator of $R(z)$ is $\det(I - zA)$. The numerator is obtained from

$$\det \begin{pmatrix} I - zA & e \\ -zb^T & 1 \end{pmatrix} = \det \begin{pmatrix} I - zA + zeb^T & 0 \\ -zb^T & 1 \end{pmatrix}.$$

Thus we obtain an alternative form for (3.29)

$$R(z) = \frac{\det(I - zA + zeb^T)}{\det(I - zA)}. \tag{3.30}$$

If we integrate $y' = y, y(0) = 1$ one single step of length z the exact solution is $\exp(z)$ and the numerical solution is $R(z)$. Using a RK-method is of order p we obtain

$$e^z - R(z) = Cz^{p+1} + \mathcal{O}(z^{p+2}) \text{ for } z \to 0. \tag{3.31}$$

Therefore we call the stability function $R(z)$ a rational approximation to $\exp(z)$ of order p and C is the error constant.

Analogous with linear multi step methods we make the following definition:

Definition 3.7 *The RK-method (3.26) has a stability region S in the complex plane, where*

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

If $S \supset \mathbb{C}^-$, the method is called A-stable.

As with linear multi step methods all numerical solutions of the RK-method (3.27) applied to a linear equation $\dot{y} = Ay$, where A is diagonalisable are bounded if $z = \tau\lambda \in S$ for all eigenvalues λ of A.

For an explicit method A is strictly lower triangular. Thus $I - zA$ is lower triangular with ones on the diagonal. This implies that $\det(I - zA) = 1$ and that $R(z)$ is a polynomial in z. For consistency $R(z)$ must be of the form $1 + z + \mathcal{O}(z^2)$. $R(z)$ can therefore not remain bounded as $z \to \infty$ for an explicit method, and all explicit RK-methods have finite regions of stability. For an implicit method $\det(I - zA)$ is also a polynomial in z and $R(z)$ becomes a rational function. Therefore all A-stable RK-methods are implicit.

From [9] we have

Theorem 3.8 *We define $E(y) = |Q(iy)|^2 - |P(iy)|^2$. A RK-method with stability polynomial $R(z) = P(z)/Q(z)$ is A-stable if and only if $R(z)$ is analytic for $Re(z) < 0$ and $E(y) \geq 0\ \forall y \in \mathbb{R}$.*

Proof. Applying the maximum principle to $\mathbb{C}^-$ we find that the maximum of $R(z)$ must be on the imaginary axis since $R(z)$ is analytic. For a method to be A-stable we must therefore have $|R(iy)| = |P(iy)|/|Q(iy)| \leq 1$ for all real y and thus $|P(iy)|^2 \leq |Q(iy)|^2$.

□

Since $R(z)$ approximates $\exp(z)$, it seems like a method where $S \equiv \mathbb{C}^-$ would be optimal. But for implicit RK-methods, $R(z)$ is a rational function and therefore

$$\lim_{z \to \infty} R(z) = \lim_{z = iy, y \to \infty} R(z).$$

On the imaginary axis $|R(z)| = 1$. This means that for z with large negative real parts $R(z)$ is close to one. Therefore components with eigenvalues where the real part is large and negative will decay slowly in the numerical solution, while in the exact solution they decay very quickly. To avoid this we must use a smaller step size such that $R(z) \ll 1$, or demand that $R(z)$ is much smaller than 1 as $z \to \infty$.

Definition 3.9 *A one-step method is called L-stable if it is A-stable and*

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

The following result connecting L-stability with the RK-matrix A is found in [9].

Theorem 3.10 *If an RK-method with non-singular A-matrix satisfies*

$$\begin{aligned} a_{sj} &= b_j, \quad j = 1, \dots, s \quad \textit{or} \\ a_{i1} &= b_1, \quad i = 1, \dots, s, \end{aligned}$$

then

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

Proof. We use the notation $e_i = (0, \dots, 0, 1, 0, \dots, 0)^T$, where the i th component is 1 and the vector has s components. From (3.29) we find that

$$\lim_{|z| \to \infty} = 1 - b^T A^{-1} e.$$

Using the first condition we obtain $A^T e_s = b \Rightarrow e_s^T = b^T A^{-1}$ and

$$\lim_{|z| \to \infty} R(z) = 1 - e_s^T e = 0.$$

From the second condition we obtain $Ae_1 = eb_1 \Rightarrow A^{-1}e = e_1/b_1$ and

$$\lim_{|z|\to\infty} R(z) = 1 - b^T e_1/b_1 = 0.$$

□

In some sense L-stable methods might be too stable. Their stability regions must necessarily contain parts of $\mathbb{C}^+$, and the methods might therefore in some cases dampening components that should be increasing. But for very stiff problems A-stable methods might be forced to take very short steps to ensure $R(z) \ll 1$, and thus we would be better off with an L-stable method.

3.3.2 Implicit RK-methods

A general implicit RK-method with a full RK-matrix A requires for each time step the solution of a (non)-linear implicit system of dimension ms

$$g(Y) \equiv Y - \tau(A \otimes I_m)F(Y) - e \otimes y_n = 0 \tag{3.32}$$

to find the stage values $Y = (Y_1, \dots, Y_s)^T$. This is analogous with (3.17) but the system is s times larger. If $B(t) \neq I$ in (3.1) we obtain the modification

$$g(Y) \equiv (B \otimes I_s)Y - \tau(A \otimes I_m)F(Y) - (B \otimes I_s)(e \otimes y_n) = 0.$$

Applying a Newton Raphson iteration scheme to (3.32) we obtain

$$\begin{aligned} J_g^{(l)}\delta^l &= -g_n^l, \\ Y^{l+1} &= Y^l + \delta^l, \end{aligned} \tag{3.33}$$

where $J_g^{(l)} = g^{'}(Y_n^{(l)}) = B \otimes I_s - \tau A \otimes J_F^{(l)}$.

Even if the Jacobian J is kept constant as long as acceptable rate of convergence is achieved, the general implicit RK-method suffer from higher computational cost per time step than linear multi step methods. This drawback gets worse as the dimension of the system increases. Since the systems connected with MOL are (very) large we are interested in implicit RK-methods, where the computational cost per time step is decreased.

Diagonally implicit RK-methods

One way to reduce costs is to construct methods where the RK-matrix A is lower triangular with non-zero elements on the diagonal. This decouples equation (3.32). Instead of solving a system of dimension ms, we solve s systems of dimension m. These methods are called diagonally implicit Runge-Kutta methods (DIRK-methods)[5].

[5] Also referred to as semi-implicit- or semi-explicit methods.

We can now write (3.32) as

$$g(Y_i) \equiv Y_i - \tau a_{ii} F(Y_i) - \tau \sum_{j=1}^{i-1} a_{ij} F(Y_j) - y_n = 0 \quad i = 1, \ldots, s \tag{3.34}$$

and solve these equations in natural order. With the iteration scheme (3.33) and keeping the Jacobian constant during iterations we find $J_{g,i} = g^{'}(Y_i) = I_s - \tau a_{ii} J_F$. If $a_{11} = \ldots = a_{ss} = \gamma$ we get additional savings, since we need only one LU-decomposition of the Jacobian $J_g = I_s - \tau\gamma J_F$ per time step. We call these methods singly diagonally implicit RK-methods (SDIRK-methods).

The restrictions on the A-matrix imposes constraints on the order and stability properties on these methods. The order for an s-stage DIRK or SDIRK method is at most $s+1$, and A-stable SDIRK-methods of this order exists only for $s = 1, 2, 3, 5$. L-stable SDIRK-methods of order $p = s$ are known for $s = 1, \ldots, 6, 8$.

Since the RK-matrix A is lower triangular it follows directly from (3.30) that the stability function for a DIRK-method is

$$R(z) = \frac{P(z)}{(1 - a_{11}z)(1 - a_{22}z)\ldots(1 - a_{ss}z)}, \tag{3.35}$$

where the numerator is a polynomial of degree s at most. If the method has order p we have

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

Multiplying with the denominator of (3.35) we obtain

$$(1 - a_{11}z)(1 - a_{22}z)\ldots(1 - a_{ss}z)e^z = P(z) + \mathcal{O}(z^{p+1}).$$

For an SDIRK method we have $(1 - a_{11}z)(1 - a_{22}z)\ldots(1 - a_{ss}z) = (1 - \gamma z)^s$ and we obtain $(1 - \gamma z)^s e^z = P(z) + \mathcal{O}(z^{p+1})$. Thus $P(z)$ is the truncated power series of $(1 - \gamma z)^s e^z$. To find $P(z)$ we use the following theorem from [5].

Theorem 3.11

$$(1 - \gamma z)^s e^z = (-1)^s \sum_{i=0}^{\infty} L_s^{(s-i)}(1/\gamma)(\gamma z)^i,$$

where

$$L_s(x) = \sum_{j=0}^{s} (-1)^j \binom{s}{j} \frac{x^j}{j!}$$

is the Laguerre polynomial of degree s and $L_s^{(n)}$ its nth derivative.

Proof. From

$$L_s(1/\gamma) = \sum_{j=0}^{s} \left(-\frac{1}{\gamma}\right)^j \binom{s}{j} \frac{1}{j!}$$

we find

$$L_s^{(s-i)}(1/\gamma) = (-1)^{s-i} \sum_{j=0}^{s} \binom{s}{j} \frac{1}{(j-s+i)!} \left(-\frac{1}{\gamma}\right)^{j-s+i}.$$

Thus

$$\begin{aligned}
(-1)^s \sum_{i=0}^{\infty} L_s^{(s-i)}(1/\gamma)(\gamma z)^i &= \sum_{i=0}^{\infty} (-1)^i (\gamma z)^i \sum_{j=0}^{s} \left(-\frac{1}{\gamma}\right)^{j-s+i} \binom{s}{j} \frac{1}{(j-s+i)!} \\
&= \sum_{j=0}^{s} \sum_{i=0}^{\infty} \binom{s}{j} (-\gamma z)^i \left(-\frac{1}{\gamma}\right)^{j-s+i} \frac{1}{(j-s+i)!} \\
&= \sum_{j=0}^{s} \sum_{k=j-s}^{\infty} \binom{s}{j} (-\gamma z)^{k-j+s} \left(-\frac{1}{\gamma}\right)^{k} \frac{1}{k!} \\
&= \sum_{j=0}^{s} \sum_{k=j-s}^{\infty} \binom{s}{s-j} (-\gamma z)^{s-j} \frac{z^k}{k!} \\
&= (1-\gamma z)^s e^z,
\end{aligned}$$

if we use the convention $1/n! = 0$ when n is a negative integer. □

From this theorem we obtain the stability function for an SDIRK-method

$$R(z) = \frac{(-1)^s \sum_{i=0}^{s} L_s^{(s-i)}(1/\gamma)(\gamma z)^i}{(1-\gamma z)^s}. \tag{3.36}$$

If we choose $1/\gamma$ such that $L_s(1/\gamma) = 0$ we obtain $\lim_{|z|\to\infty} R(z) = 0$ since the degree of the nominator becomes $s-1$ and the degree of the denominator is s, and if the method is A-stable it is also L-stable.

However, the SDIRK-methods become more difficult to construct as order increases, and because of the structure of the RK-matrix the first stages have low order.

Singly implicit RK-methods

We look at collocation methods where the characteristic polynomial for the RK-matrix A has a single s-fold zero γ. From [4] we have the following theorem.

Theorem 3.12 *For an s-stage implicit collocation method with $\sigma(A) = \{\gamma\}$, $L_s(c_1/\gamma) = \ldots = L_s(c_s/\gamma) = 0$ and*

$$\sum_{j=1}^{s} a_{ij} p(c_j) = \int_0^{c_i} p(c)\, dc,$$

p any polynomial of degree less than s, we have the similarity transformation

$$T^{-1}AT = M,$$

where

$$M = \begin{bmatrix} \gamma & & & & \\ -\gamma & \gamma & & & \\ & \ddots & \ddots & & \\ & & -\gamma & \gamma & \\ & & & -\gamma & \gamma \end{bmatrix}$$

and $[T]_{ij} = L_{j-1}(c_i/\gamma)$ *and* $[T^{-1}]_{ij} = c_j L_{i-1}(c_j/\gamma)/(\gamma s^2 L_{s-1}(c_j/\gamma)^2)$

Proof. Set $p(x) = L_k(x/\gamma)$, $k = 1, \dots, s$. Thus

$$\sum_{j=1}^{s} a_{ij} p(c_j) = \sum_{j=1}^{s} a_{ij} L_k(c_j/\gamma) = \int_0^{c_i} L_k(\frac{c}{\gamma})\, dc.$$

Using the property $L_n'(x) = L_{n-1}'(x) - L_{n-1}(x)$ and substituting $c = \gamma\xi$ we obtain

$$\gamma \int_0^{\xi_i} L_k'(\xi) - L_{k+1}'(\xi)\, d\xi = \gamma L_k(\xi_i) - \gamma L_{k+1}(\xi_i).$$

Setting $v_k = (L_{k-1}(\xi_1), \dots, L_{k-1}(\xi_s))^T$, $k = 1, \dots, s$ this reads

$$A v_k = \gamma v_k - \gamma v_{k+1}, \quad k = 1, \dots, s.$$

Since $T = [v_1, \dots, v_s]$ and $L_s(\xi_i) = 0$ we obtain

$$AT = TM.$$

□

We can now obtain computational savings similar to those of SDIRK-methods by exploiting the similarity transformation $T^{-1}AT = M$. We consider a simplified Newton scheme for solving (3.32)

$$(I_s \otimes I_m - \tau A \otimes J_F)\delta = -g(Y), \tag{3.37}$$

where $\delta = Y^{(l)} - Y^{(l-1)}$. We define the transformations

$$Y = (T^{-1} \otimes I_m)\hat{Y}, \quad g = (T^{-1} \otimes I_m)\hat{g},$$

and write $\hat{\delta} = \hat{Y}^{(l)} - \hat{Y}^{(l-1)}$. Inserting this into (3.37) and multiplying with $(T \otimes I_m)^{-1}$ we obtain

$$\begin{gathered} (I_s \otimes I_m - \tau (T \otimes I_m)^{-1}(A \otimes J_F)(T \otimes I_m))\hat{\delta} = -\hat{g} \\ \Updownarrow \\ (I_s \otimes I_m - \tau M \otimes J_F)\hat{\delta} = -\hat{g}, \end{gathered} \tag{3.38}$$

where

$$(I_s \otimes I_m - \tau M \otimes J_F) = \begin{bmatrix} I_m - \tau\gamma J_F & & & & \\ -\tau\gamma J_F & I_m - \tau\gamma J_F & & & \\ & & \ddots & & \\ & & -\tau\gamma J_F & I_m - \tau\gamma J_F & \\ & & & -\tau\gamma J_F & I_m - \tau\gamma J_F \end{bmatrix}.$$

In this transformed system (3.38) we need only the LU-decomposition of $I_m - \tau\gamma J_F$ for each time step. Before the transformation we needed the LU-decomposition of $I_s \otimes I_m - \tau A \otimes J_F$. Thus the computational cost can almost compare with SDIRK-methods. The difference is the overhead from the transformations. The restricted form of the A-matrix still imposes constraints on order and stability properties as with SDIRK-methods. But with SIRK-methods it is easier to construct higher order methods and they don't suffer from loss of stage order.

SIRK-methods with $\sigma(A) = \gamma$ have the same stability functions (3.36) as SDIRK-methods of the same order with $a_{ii} = \gamma$. An example of a stability region for a 3 stage SIRK method is shown in figure 3.4

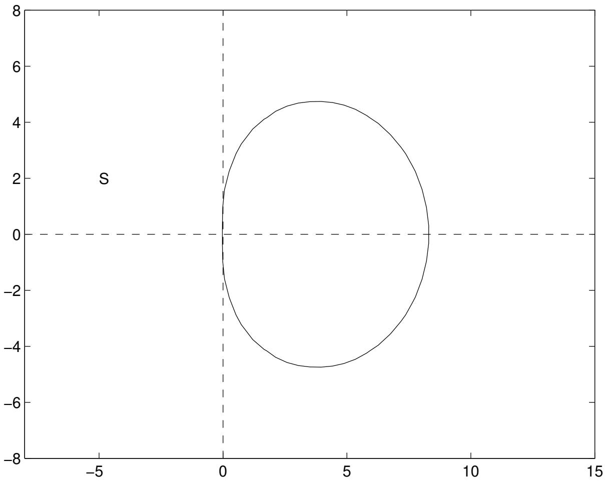

Figure 3.4: The stability region S for the 3 stage SIRK method with $\gamma = 0.43587$ (outside the closed curve).

From [2] we find values for γ giving A-stable and L-stable SIRK/SDIRK methods. These are given in table 3.2

Table 3.2: Values of γ giving A-stable or L-stable SIRK/SDIRK-methods.

s	A-stability and $p = s + 1$	L-stability and $p = s$
1	1/2	1
2	$(3 + \sqrt{3})/6$	$(2 \pm \sqrt{2})/2$
3	1.06858	0.43587
4	—	0.57282
5	0.47328	0.27805
6	—	.33414
7	—	—
8	—	0.23437

3.3.3 RK-methods with extended stability regions

If we consider large systems that are only "mildly" stiff it might be possible to use explicit methods with large stability regions, where conventional explicit methods can not compete with implicit ones due to the severe stability constraint on the step size. Since explicit methods avoid the cost of the Newton iterations involved in implicit methods, computational savings can be obtained as long the modulus of the eigenvalues are "moderate".

We will look at RK-methods with extended real stability regions, and a requirement is therefore that the eigenvalues of the Jacobian is situated along or close to, the negative real axis. Such problems typically arise from the semi discretisation of parabolic partial differential equations or diffusion dominated problems.

Conventional RK-methods try to obtain maximal order p with a minimum number of stages s. The idea of the RK-methods with extended stability regions is to use a large number of stages, where most of them are used to obtain larger stability regions and only a few to obtain the required order. The construction process consists of two segments,

1. For a given s and p, find a polynomial of the form $P_s(z) = 1 + \alpha_1 z + \alpha_2 z^2 + \ldots + \alpha_s z^s$, $\alpha_i = 1/(i!)$ for $i = 1, \ldots, p$ and $\alpha_{p+1}, \ldots, \alpha_s$ are free parameters used to maximise the region along the negative real axis where $|P_s(z)| \leq 1$.

2. Find RK-methods of order p with $P_s(z)$ as its stability polynomial.

First order stability polynomials

For $p = 1$ we must require $\alpha_1 = 1$, and the other coefficients are used to maximise the real stability boundary which we will denote $\beta(s)$. The polynomials that maximises the real stability boundary, $\beta(s)$, are called optimal. For first order methods the optimal polynomials are known.

We introduce the Chebyshev polynomials

$$\begin{aligned} T_s(x) &= \cos(s \arccos x), \quad -1 \leq x \leq 1 \quad \text{or} \\ T_s(x) &= 2xT_{s-1}(x) - T_{s-2}(x), \quad T_0(x) = 1, \quad T_1(x) = x. \end{aligned} \tag{3.39}$$

The recursive definition is valid for all complex x. These polynomials have the property that $|T(X)| \leq 1$ whenever $x \in [-1, 1]$.

Following [22] we introduce the shifted Chebyshev polynomials

$$R_s(z) = T_s(1 + z/s^2), \tag{3.40}$$

where $R_s(z) = 1 + z + O(z^2)$ and $|R_s(z)| \leq 1$ for $-2s^2 \leq z \leq 0$. For first order methods these are the unique, optimal polynomials[6], that is the polynomials which maximises the real stability boundary with $\beta(s) = 2s^2$.

In points where $R_s(z) = \pm 1$ ∂S touches the negative real axis, the width of the stability region is zero. We would like $R(z) < 1$ in a narrow strip along the axis. To introduce damping in these points we define

$$R_s(z) = \frac{1}{T_s(\omega_0)} T_s(\omega_0 + \omega_1 z), \quad \omega_0 = 1 + \frac{\epsilon}{s^2}, \quad \omega_1 = \frac{T_s(\omega_0)}{T_s'(\omega_0)}, \tag{3.41}$$

where ϵ is a small parameter (say $\epsilon = 0.05$). These polynomials oscillates approximately between $1-\epsilon$ and $-1+\epsilon$ and also satisfies $R_s(z) = 1 + z + O(z^2)$.

Theorem 3.13 *For the Chebyshev polynomials (3.39) we have*

$$T_s'(1) = s^2 \quad \textit{and} \quad T_s''(1) = \frac{s^2}{3}(s^2 - 1).$$

Proof. Suppose that the result is true for $s-1$ and $s-2$. Then, by using the recursive definition in (3.39)

$$\begin{aligned} T_s'(1) &= 2T_{s-1}(1) + 2T_{s-1}'(1) - T_{s-2}'(1) \\ &= 2 + 2(s-1)^2 - (s-2)^2 \\ &= s^2, \end{aligned}$$

and the result follows since $T_0'(1) = 0$ and $T_1'(1) = 1$.

Again suppose that the result is true for $s-1$ and $s-2$. Then

$$\begin{aligned} T_s''(1) &= 4T_{s-1}'(1) + 2T_{s-1}''(1) - T_{s-2}''(1) \\ &= 4(s-1)^2 + 2\frac{1}{3}(s-1)^2\left((s-1)^2 - 1\right) \\ &\quad -\frac{1}{3}(s-2)^2\left((s-2)^2 - 1\right) \\ &= \frac{1}{3}s^2(s^2 - 1), \end{aligned}$$

and the result follows since $T_0''(1) = 0$ and $T_1''(1) = 0$.

□

The stability boundary for (3.41) is defined by $\omega_0 + \omega_1 z = -1$. Using the above result we obtain

$$\begin{aligned} \beta(s) &= \frac{(\omega_0 + 1)T_s'(\omega_0)}{T_s(\omega_0)} \\ &\approx \left(2 + \frac{\epsilon}{s^2}\right)\left(s^2 + \frac{\epsilon}{3}(s^2 - 1)\right)\left(1 - \frac{\epsilon}{s^2}\right) \\ &\approx (2 - \frac{4}{3}\epsilon)s^2 \end{aligned}$$

For $\epsilon = 0.05$ we get 5 % damping with $\beta(s) \approx 1.93s^2$. The stability domain becomes a slightly shorter, but ∂S is a safe distance from the axis.

[6]See [19], theorem 4.2.1.

Example 3.1 *We compare the stability regions for an explicit 4 stage fourth order RK-method with stability polynomial R_4, and two first order methods with P_4 and P_4 with damping as stability polynomials.*

$$\begin{aligned} R_4(z) &= 1 + z + \frac{1}{2}z^2 + \frac{1}{6}z^3 + \frac{1}{24}z^4 \\ P_4(z) &= 1 + z + \frac{5}{32}z^2 + \frac{1}{128}z^3 + \frac{1}{8192}z^4 \end{aligned}$$

We can see from figure 3.5 that $\beta(4) = 32$ for P_4 and $\beta(4) \approx 31$ for P_4 damped.

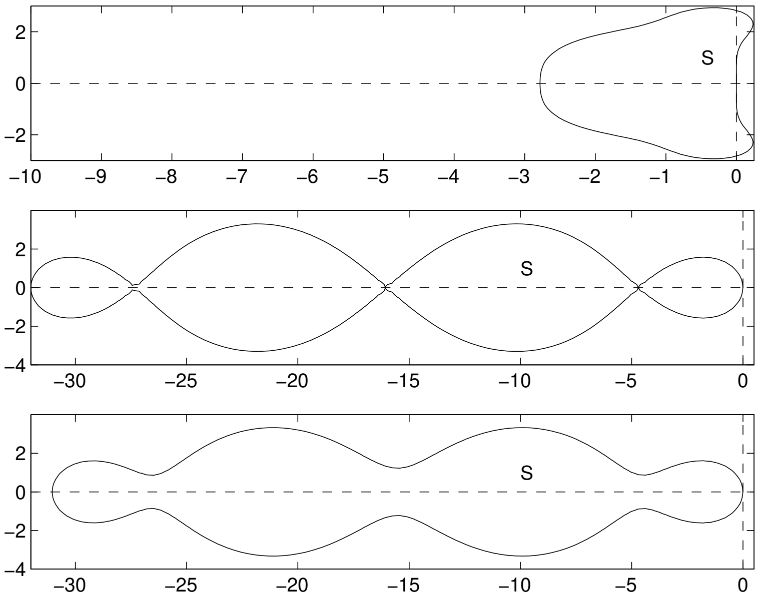

Figure 3.5: Stability regions for R_4, P_4 and P_4 with damping.

Second order stability polynomials

For $p = 2$ we must require $\alpha_1 = 1$ and $\alpha_2 = 1/2$. In [9] it is reported that analytical expressions for second order optimal polynomials was found in terms of elliptical integrals by Lebedev & Medovikov in 1994. But their practical computation is usually done numerically. Two approximate polynomials have been derived by Bakker and by van der Houwen and Sommeijer. The polynomial derived by Bakker is defined by

$$B_s(z) = \frac{2}{3} + \frac{1}{3s^2} + \left(\frac{1}{3} - \frac{1}{3s^2}\right) T_s\left(1 + \frac{3z}{s^2 - 1}\right), \tag{3.42}$$

and generates approximately 80 % of the optimal interval, with $\beta(s) \approx 2/3(s^2 - 1)$.

The polynomial derived by van der Houwen and Sommeijer

$$P_s(z) = \frac{2}{2-z} - \frac{z}{2-z} T_s\left(\cos\frac{\pi}{s} + \frac{z}{2}(1-\cos\frac{\pi}{s})\right), \tag{3.43}$$

is closer to the optimal one, with $\beta(s) \approx 0.81s^2$, but the Bakker polynomials seems to perform better because of smaller error constants.

In a similar manner to the first order stability polynomials we can introduce damping in the Bakker-polynomials (3.42)

$$B_s(z) = a_s + b_s T_s(\omega_0 + \omega_1 z), \tag{3.44}$$

where $\omega_0 = 1 + \epsilon/s^2$ and

$$\omega_1 = \frac{T_s'(\omega_0)}{T_s''(\omega_0)}, \quad a_s = 1 - b_s T_s(\omega_0), \quad b_s = \frac{T_s''(\omega_0)}{(T_s'(\omega_0))^2}.$$

For $\epsilon = 2/13$ we obtain approximately 5% damping with a reduction in $\beta(s)$ of about 2%.

The existence of optimal polynomials for arbitrarily high order has been proved, but a closed analytic expression is not known for $p > 1$. For $p = 2$ we have approximate polynomials, and for higher orders the coefficients of optimal polynomials can be found numerically.

In [20] van der Houwen reports numerical computations of the stability boundaries for the optimal stability polynomials for methods up to order $p = 4$. For these methods we have $\beta(s) = c_p(s)s^2$ as $s \to \infty$, with $c_2(s) \approx 0.814$, $c_3(s) \approx 0.489$, $c_4(s) \approx 0.341$. Since C_p decreases with order, a higher number of stages will be required with a method of order p to achieve the same stability regions as a lower order method. This seems to favour the development of low order methods.

Construction of RK-methods corresponding to $R_s(z)$

By applying an explicit RK-method to the test equation $\dot{y} = \lambda y$ we obtain $y_{n+1} = R(z)y_n$, $z = \tau\lambda$, where

$$R(z) = 1 + z\sum_j b_j + z^2 \sum_{j,k} b_j a_{jk} + z^3 \sum_{j,k,l} b_j a_{jk} a_{kl} + \cdots. \tag{3.45}$$

We can now construct RK-methods by comparing the coefficients of (3.45) with a chosen stability polynomial. Following [22] we consider three different techniques for constructing RK-methods from a given stability polynomial.

The *diagonal method*: all coefficients of A are 0 except the first sub-diagonal and $b_s = 1$. The method can be written as

$$\begin{aligned} Y_j &= y_n + a_{j,j-1} F(t_n + c_{j-1}\tau, Y_{j-1}), \\ y_{n+1} &= y_n + \tau F(t_n + \tau, Y_s). \end{aligned}$$

From (3.45) we obtain the stability function

$$R(z) = 1 + z + \sum_{i=2}^{s} \left(\prod_{j=2}^{i} a_{s+2-j,s+1-j} \right) z^i.$$

By comparing this with $P_s(z) = 1 + z + \alpha_2 z^2 + \ldots + \alpha_s z^s$ we obtain

$$a_{j,j-1} = \frac{\alpha_{s-j+2}}{\alpha_{s-j+1}} \quad j = 2, \ldots, s.$$

This method is simple to construct but suffers from internal instability, making the method unstable for $s \geq 12$. This restricts the usefulness of the method since the idea is to use a large number of stages.

The *factorised method*: This method can be interpreted as an RK-method that is factorised as a sequence of Euler steps and reads

$$\begin{aligned} Y_1 &= y_n \\ Y_j &= Y_{j-1} + a_{j,j-1}\tau F(t_n + c_{j-1}\tau, Y_{j-1}) \quad j = 2, \ldots, s, \\ y_{n+1} &= Y_s + b_s \tau F(t_n + c_s \tau, Y_s). \end{aligned}$$

The stability function is

$$R_s(z) = (1 + b_s) \prod_{j=2}^{s} (1 + a_{j,j-1} z).$$

Comparing this with a factorisation of a polynomial with real zeros thus defines the method. For examples the first order polynomials (3.41) can be factorised as

$$P_s(z) = \prod_{j=1}^{s} \left(1 + \frac{\frac{\omega_1}{\omega_0}}{1 - \frac{1}{\omega_0} \cos(\frac{2j-1}{2s}\pi)} z \right).$$

This method can not be used if the polynomial possesses complex zeros (f. ex. the Bakker polynomials (3.42)). This method also suffers from internal instability, but this can be overcome by a special ordering of the zeros associated with $a_{j,j-1}$. The ordering of the zeros is dependent on s and and is not very convenient in an automatic code, where s changes throughout the integration.

The *RKC-method*: The method is constructed such that all stability polynomials belonging to intermediate stages are defined by the three-term Chebyshev recursion, and such that at the sth stage the chosen stability polynomial P_s results. An s-stage RKC method can be written as

$$\begin{aligned} Y_0 &= y_n \\ Y_1 &= Y_0 + \tau_n \tilde{\mu}_1 F_0 \\ Y_j &= \mu_j Y_{j-1} + \nu_j Y_{j-2} + (1 - \nu_j - \mu_j) Y_0 \\ &\quad + \tau_n \tilde{\mu}_j F_{j-1} + \tilde{\gamma}_j \tau_n F_0 \qquad \text{for } j = 2, \ldots, s \\ y_{n+1} &= Y_s, \end{aligned}$$

where $F_j = F(t_n + c_j\tau_n, Y_j)$, $c_0 = 0$, $c_1 = \tilde{\mu}_1$ and $c_j = \mu_j c_{j-1} + \nu_j c_{j-2} + \tilde{\mu}_j + \tilde{\gamma}_j$. The RKC-method can be seen as an explicit s-stage RK-method by writing Y_j in the standard form (3.26) and thus obtaining the coefficients a_{ji} expressed by $\mu_j, \nu_j, \tilde{\mu}_j$ and $\tilde{\gamma}_j$.

We define the stability polynomial recursively by

$$\begin{aligned} &P_0(z) = 1, \quad P_1(z) = 1 + \tilde{\mu}_1 z, \\ &P_j(z) = (1 - \mu_j - \nu_j) + \tilde{\gamma}_j z + (\mu_j + \tilde{\mu}_j z)P_{j-1}(z) + \nu_j P_{j-2}(z). \end{aligned} \tag{3.46}$$

By adapting the recursion (3.46) to the three-term recursion of some chosen shifted Chebyshev polynomials the coefficients $\mu_j, \nu_j, \tilde{\mu}_j, \tilde{\gamma}_j$ are defined. This is done for first and second order in [23]. Since $|P_j(z)| \leq 1$, whenever $P_s(z)| \leq 1$, the RKC method is internally stable.

If $B(t) \neq I$ in (3.1) the above methods can be modified by applying them to the system

$$\dot{y} = B^{-1}F(y)$$

The resulting formulas is then pre-multiplied by B, thus avoiding destruction of banded structures.

3.3.4 Rosenbrock methods

With implicit RK-methods we must solve a non-linear system to obtain the stage values. In SDIRK-/SIRK-methods we achieve reduced computational cost by solving several smaller non-linear systems. These systems are solved with a Newton type iteration scheme where we use the Jacobian. Another approach to obtain computationally cheap implicit methods might therefore be to develop methods where the Jacobian is used in the method and replacing the non-linear systems by linear systems.

A Rosenbrock-method is given by

$$y_{n+1} = y_n + \sum_{i=1}^{s} b_i K_i, \tag{3.47}$$

$$K_i = \tau f\left(y_n + \sum_{j=1}^{i-1} \alpha_{ij} K_j\right) + \tau J \sum_{j=1}^{i} \beta_{ij} K_j, \tag{3.48}$$

where b_i, α_{ij} and β_{ij} are coefficients characterising the method and $J = f'(y_n)$.

If $\alpha_{ij} = a_{ij}$, $\beta_{ii} = a_{ii}$ and $\beta_{ij} = 0$ for $j = 1, \ldots, i-1$ the Rosenbrock-method (3.47, 3.48) can be regarded as a linearisation of a DIRK-method where

$$K_i = \tau f\left(y_n + \sum_{j=1}^{i-1} a_{ij} K_j + a_{ii} K_i\right),$$

are linearised to obtain

$$K_i = \tau f\left(y_n + \sum_{j=1}^{i-1} a_{ij} K_j\right) + \tau J a_{ii} K_i.$$

For each time step we must solve the linear equations

$$g(K_i) \equiv (I - \tau\beta_{ii} J)K_i - \tau f\left(y_n + \sum_{j=1}^{i-1} a_{ij} K_j\right) \\ -\tau J \sum_{j=1}^{i-1} \beta_{ij} k_j = 0 \quad i = 1, \ldots, s, \tag{3.49}$$

to find the stage values K_i. If $\beta_{11} = \ldots = \beta_{ss} = \beta$ we can use the same LU-decomposition of $I - \tau\beta J$ for all the stage values in each time step.

If the system we want to solve is on the implicit form (3.1), where B is non singular, we can pre multiply the system by B^{-1} and apply method (3.47), (3.48). The resulting formula is then multiplied with M and we obtain the following modification of (3.48)

$$MK_i = \tau f\left(y_n + \sum_{j=1}^{i-1} \alpha_{ij} K_j\right) + \tau J \sum_{j=1}^{i} \beta_{ij} K_j.$$

Thus we avoid the inversion of M and band structures of M and J will be preserved.

In some circumstances the matrix $I - \tau\beta J$ involved in the solution for the stage values, may be ill-conditioned. This makes a fundamental difference between other implicit methods and the Rosenbrock method. A similar matrix is also involved in the Newton iterations in conventional implicit methods, but there it is used to obtain a new iterate approximating the stage values. In this case ill-conditioning will only cause fewer valid digits in the new iterate and therefore a slowing down of the convergence rate. With a Rosenbrock method the results from the linear system will enter the solution at that step directly. Thus, inaccurate results from the linear systems lead to inaccurate solution values. However, this will usually not be very important if we do not demand too stringent accuracy.

By applying the Rosenbrock method (3.47) to the test equation $y' = \lambda y$ we obtain the stability function

$$R(z) = 1 + zb^T(I - zB)^{-1}e, \tag{3.50}$$

where $b^T = (b_1, \ldots, b_s)$ and $[B]_{ij} = \alpha_{ij} + \beta_{ij}$, with $\alpha_{sj} = 0$. The coefficient matrix B is lower triangular and (3.50) is equal to the stability function of a DIRK-method with RK-matrix B.

Chapter 4

Numerical experiments

The codes used in these numerical experiments are

VODE A code based on linear multi step methods. BDF methods are implemented for stiff problems and AM methods are implemented for non–stiff problems, with the fixed leading coefficient scheme. VODE can be obtained from Netlib [26].

RKC A code based on second order explicit RK-methods with extended real stability intervals. The Runge-Kutta-Chebyshev scheme is used for the implementation. RKC can be obtained from Netlib [26].

ROS4 A code based on Rosenbrock methods of fourth order. The methods implemented are discussed in chapter IV.7 of [9], and ROS4 can be obtained from the authors[1].

In this chapter we will compare RKC with VODE(BDF) on a parabolic problem in one and two dimensions. We will compare the explicit AM methods with the implicit BDF methods in VODE on the advection–diffusion equation, and we will use ROS4 and VODE(BDF) on a non linear system of partial differential equations.

4.1 A parabolic problem

We consider the heat equation

$$u_t = \mu \Delta u \tag{4.1}$$

on $[0, 1]$ in one dimension, and on the unit square in two dimensions, with $\mu = 0.01$. We semi discretise the equation in the same manner as in section 1.2 for the one dimensional case and as in section 2.1.6 in the two dimensional

[1]See "Appendix Fortran Codes" in [9].

case. We integrate from $t = 0$ to $t = 10.0$ with relative and absolute tolerance $1/(N+1)^2$. In one dimension we use the initial values $u(x, 0) = \sin 2\pi x$. In two dimensions we use $u(x, y, 0) = -2(x(x-1) + y(y-1))$ except on the boundary, where $u(x, y, 0) = 0$. This gives a discontinuity in the initial conditions on the boundary, but this will soon be smoothened out by the diffusion.

We compare the RKC code and the VODE with BDF methods. The solution of the one dimensional problem is presented in figure 4.1, and the initial condition for the two dimensional problem is given in figure 4.2. Some of the run characteristics are presented in the figures 4.3, 4.4 and the tables 4.1, 4.2.

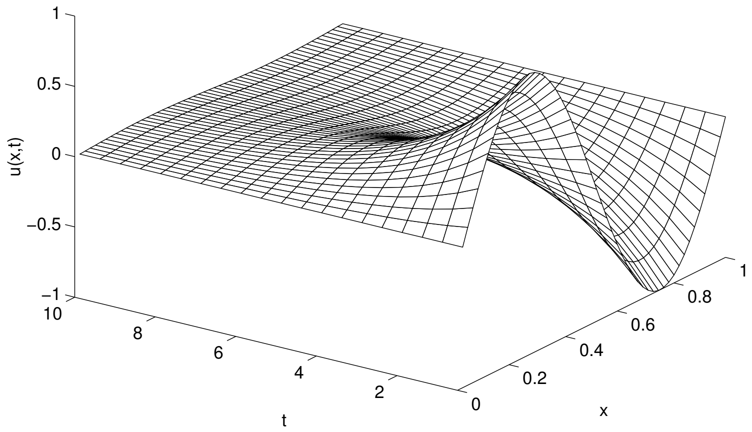

Figure 4.1: Numerical solution of 4.1 in one space dimension , with N=50.

Table 4.1: VODE and RKC on 1D heat equation.

N		Steps	f-s	J-s	LU-s
30	RKC	16	97		
	VODE	10	14	1	7
60	RKC	20	183		
	VODE	11	16	1	8
100	RKC	23	302		
	VODE	11	16	1	8

From theorem 1.2 we find that the eigenvalues of this problem are in the interval $[-4(N+1)^2, 0]$ for the one dimensional case and in $[-8(N+1)^2, 0]$ for the two dimensional case. (In two dimensions the number of equations are N^2.)

We can clearly see that in one dimension RKC uses many stages (many function evaluations) and still many steps (small step sizes). The extended stability region of RKC is not large enough for the code to be effective on this problem.

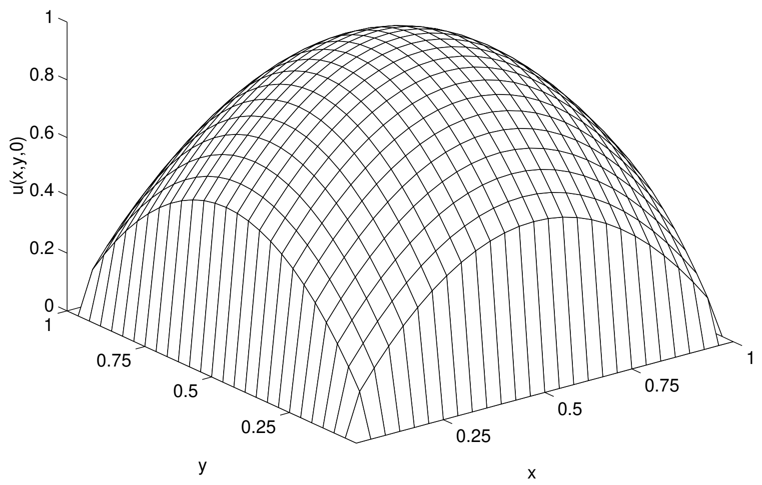

Figure 4.2: The initial condition of 4.1 in two space dimensions.

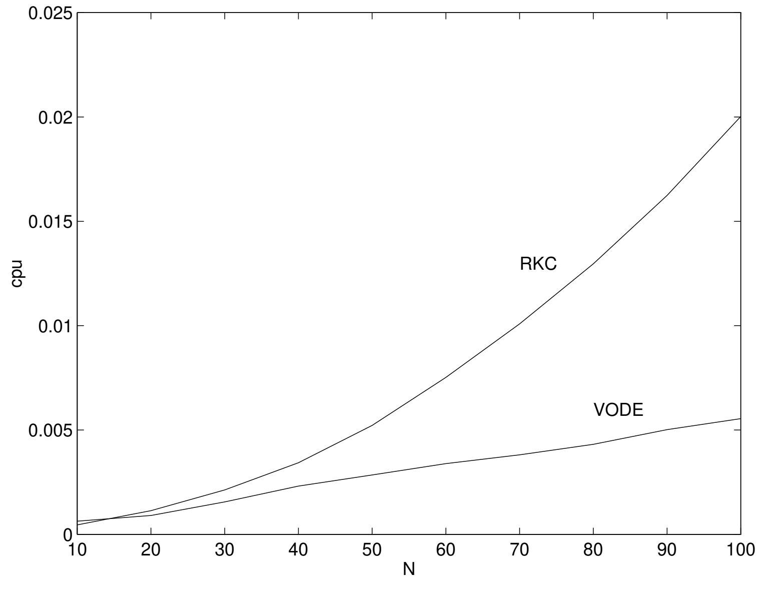

Figure 4.3: VODE versus RKC on (4.1) in one dimension.

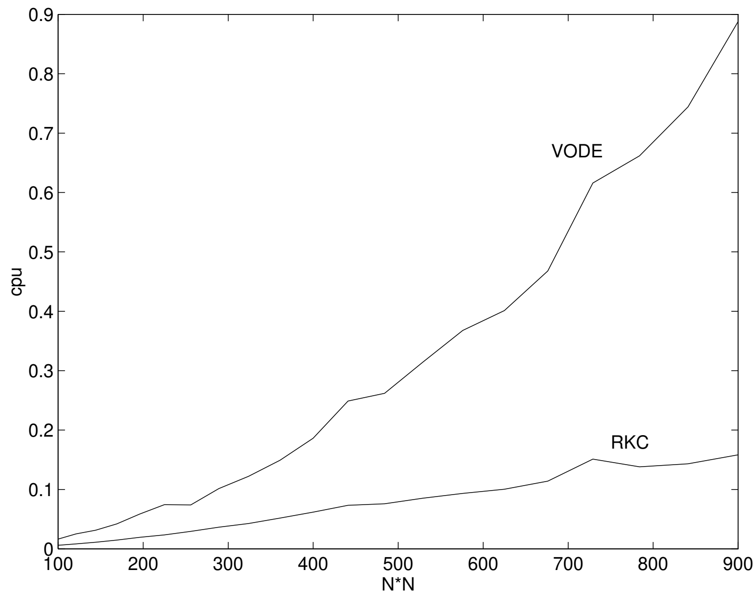

Figure 4.4: VODE versus RKC on (4.1) in two dimensions.

Table 4.2: VODE and RKC on 2D heat equation.

N		Steps	f-s	J-s	LU-s
100	RKC	13	49		
	VODE	10	35	1	6
400	RKC	29	135		
	VODE	16	62	1	8
900	RKC	23	156		
	VODE	19	85	1	9

In addition the tri diagonal structure of the Jacobian makes the linear algebra involved in the BDF methods very simple, and RKC can not compete with VODE.

However, in two dimensions RKC is clearly the fastest code. Now the eigenvalues increases linearly with the number of equations, while in one dimension the eigenvalues increased quadratically with the number of equations. The Jacobian has now a banded structure with band width $2N+1$. Thus this problem requires more work for VODE and less for RKC.

We note that for the RKC method the stability criterion on the step size is

$$2s^2 \geq \tau\lambda \Rightarrow s \geq \sqrt{\tau 2(N+1)^2}.$$

Thus the number of stages increases linearly with the number of equations in one dimension and increases as the square root of the number of equations in the two dimensional case. This is confirmed from the number of function evaluations in table 4.1 and 4.2.

4.2 Advection–diffusion problems

We consider the advection-diffusion equation in one space dimension

$$u_t + \nu u_x = \mu u_{xx}, \tag{4.2}$$

with $\nu, \mu \geq 0$, the initial condition $u(x,0) = \exp(-(x-0.25)^2/0.01)$ and the boundary conditions $u(0,t) = u(1,t) = 0$. We will look at the two problems $\nu = 1$, $\mu = 0$ and $\nu = 1$, $\mu = 10^{-4}$. The problems are integrated from $t = 0$ to $t = 1.0$ with relative and absolute tolerance $1/(N+1)^2$. Figure 4.5 shows an example of a solution of equation (4.2) for some t, when $\nu = 1$ and $\mu = 0.03$. Some of the run characteristics are presented in the figures 4.8, 4.7, and the tables 4.3, 4.4.

We discretise with second order central approximations and obtain

$$\dot{U} = \begin{bmatrix} a & b & & & \\ c & a & b & & \\ & \ddots & \ddots & \ddots & \\ & & c & a & b \\ & & & c & a \end{bmatrix} U, \tag{4.3}$$

where

$$a = -2\mu h^{-2}, \quad c = \mu h^{-2} - 0.5\nu h^{-1}, \quad \text{and} \quad b = \mu h^{-2} - 0.5\nu h^{-1}.$$

By theorem 1.2, the eigenvalues of the Jacobian are

$$\lambda_s = -\frac{2\mu}{h^2} + 2\sqrt{\frac{\mu^2}{h^4} - \frac{\nu^2}{4h^2}} \cos(s\pi h) \quad s = 1, \ldots, N. \tag{4.4}$$

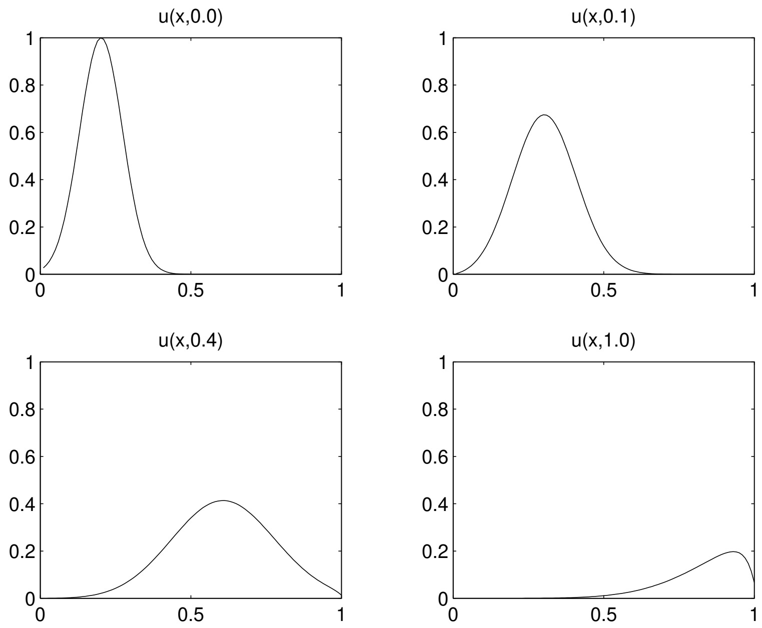

Figure 4.5: Solution of (4.2) with $\nu = 1$ and $\mu = 0.03$.

If $\mu > 0$ we can write (4.4) as

$$\lambda_s = -\frac{2\mu}{h^2} + \frac{2\mu}{h^2}\sqrt{1 - \left(\frac{\nu h}{2\mu}\right)^2} cos(s\pi h), \quad s = 1, \ldots, N. \tag{4.5}$$

From (4.5) we can see that $\nu h/\mu \leq 2$ gives real eigenvalues, and that $\nu h/\mu > 2$ gives complex eigenvalues.

If $\nu h/\mu > 2$ then all the eigenvalues will be situated within the sector $\{x+iy; x < 0, |y/x| \leq \tan(\alpha)\}$, where

$$\tan(\alpha) = \sqrt{\left(\frac{\nu h}{2\mu}\right)^2 - 1}.$$

The eigenvalues of (4.3) are shown in figure (4.6) for $N = 200$ and $N = 800$. Note that only the eigenvalues with maximum modulus and some of those with smaller modulus have been drawn. We can see that the width of the sector where the eigenvalues are situated decreases as N increases, and then $\nu/(\mu(N+1)) < 2$ the eigenvalues become real. Thus the stiffness of the system increases for increasing N, but from the stability point of view this is an advantage for the BDF methods. (refer to the stability regions of the BDF methods in figure 3.2).

The computation times for AM and BDF on (4.2) with $\nu = 1$ and $\mu = 10^{-4}$ for different N are given in figure 4.7. We can see that BDF methods get better compared to AM methods as N increases.

With $\nu = 1$ and $\mu = 0$ all the eigenvalues are pure imaginary. Thus the problem is not regarded as stiff, and we might expect that an explicit method might be

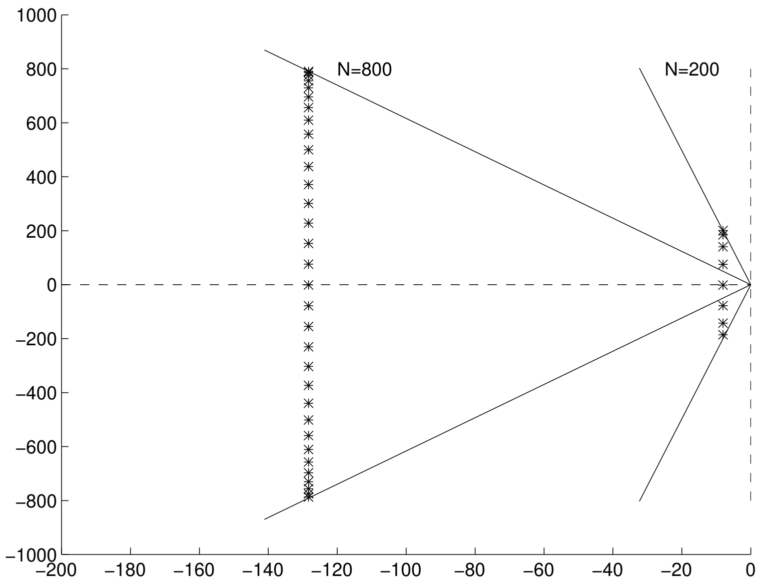

Figure 4.6: Eigenvalues for (4.3) with $\nu = 1$ and $\mu = 10^{-4}$.

more efficient than an implicit method. From figure 4.8 we can see that AM is slightly faster than BDF, but the difference is very small. This might be because the test problem is linear. Thus the Newton iterations in BDF will (in theory) converge in after one iteration. From table 4.3 and 4.4 it is confirmed that this happens for most of the steps. Therefore it is not much more work involved in BDF than in AM.

Table 4.3: BDF and AM on (4.2) with $\nu = 1$ and $\mu = 0$.

N		Steps	f-s	J-s	LU-s
200	AM	413	826		
	BDF	564	585	10	34
400	AM	902	1808		
	BDF	1049	1096	18	56
700	AM	1844	3694		
	BDF	1746	1826	30	89

4.3 Medical Akzo Nobel Problem

This problem is collected from the test site [25]. The problem is a reaction diffusion system in one space dimension

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} - kuv, \tag{4.6}$$

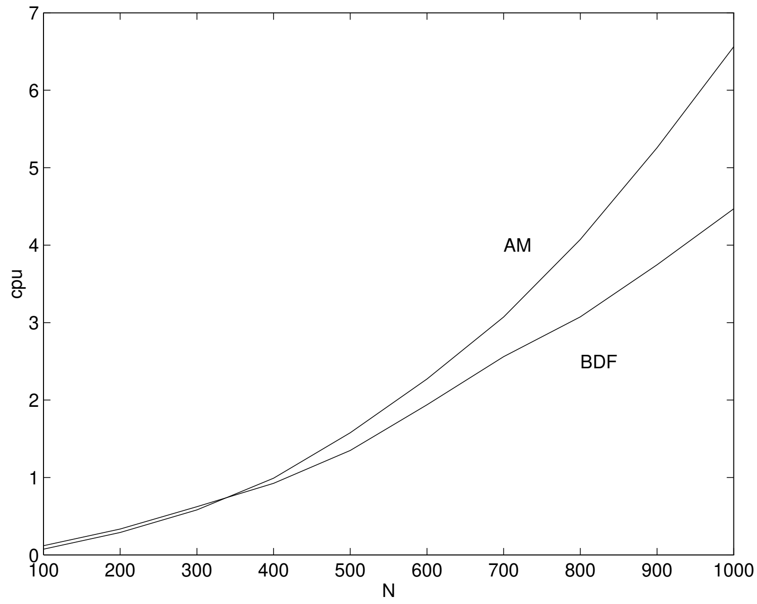

Figure 4.7: Computation times with $\nu = 1$ and $\mu = 10^{-4}$.

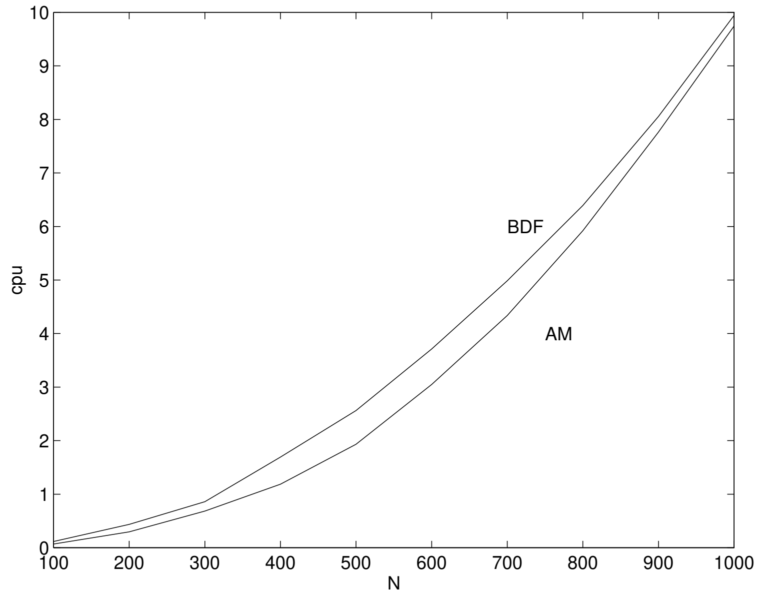

Figure 4.8: Computation times with $\nu = 1$ and $\mu = 0$.

Table 4.4: BDF and AM on (4.2) with $\nu = 1$ and $\mu = 10^{-4}$.

N		Steps	f-s	J-s	LU-s
200	AM	418	681		
	BDF	395	409	7	26
400	AM	949	1077		
	BDF	527	562	9	34
700	AM	1648	1869		
	BDF	791	843	14	48

$$\frac{\partial v}{\partial t} = -kuv, \tag{4.7}$$

considered in the strip $\Omega = \{(x,t) : 0 < x < \infty, 0 < t < T\}$ for some T, with the initial conditions $u(x,0) = 0, v(x,0) = v_0, x > 0$ and the boundary condition $u(0,t) = \phi(t)$ for $0 < t < T$.

With the transformation

$$\zeta = \frac{x}{x+c}$$

$c > 0$, Ω is transformed into the finite region $\{(\zeta,t), 0 < \zeta < 1, 0 < t < T\}$. Inserting this transformation into (4.6),(4.7) and using second order approximations for the space derivatives we obtain the system

$$\dot{y} = F(y), \tag{4.8}$$

where F is given by

$$f_{2j-1} = \alpha_j \frac{y_{2j+1} - y_{2j-3}}{2\Delta\zeta} + \beta_j \frac{y_{2j-3} - 2y_{2j-1} + y_{2j+1}}{(\Delta\zeta)^2} - ky_{2j-1}y_{2j} \tag{4.9}$$

$$f_{2j} = -ky_{2j}y_{2j-1}, \tag{4.10}$$

with

$$\alpha_j = \frac{2(j\Delta\zeta - 1)^3}{c^2}, \quad \beta_j = \frac{(j\Delta\zeta - 1)^4}{c^2}.$$

Here $y = (u_1, v_1, u_2, v_2, \ldots, u_N, v_N)^T$. The problem is solved with $T = 20, k = 100, v_0 = 1, c = 4$, relative and absolute tolerance $1/(N+1)^2$ and

$$\phi(t) = \begin{cases} 2 & \text{for } t \in (0,5], \\ 0 & \text{for } t \in (5,20]. \end{cases}$$

We have compared the performance of VODE based on the BDF formulas and the Rosenbrock procedure ROS4. The solution can be found in the description of problem 7 at [25]. Some of the run characteristics are presented in figure 4.9 and in table 4.5.

ROS4 seems to perform comparable with VODE(BDF), but as N increases VODE becomes slightly better. One reason might be that as N increases, the

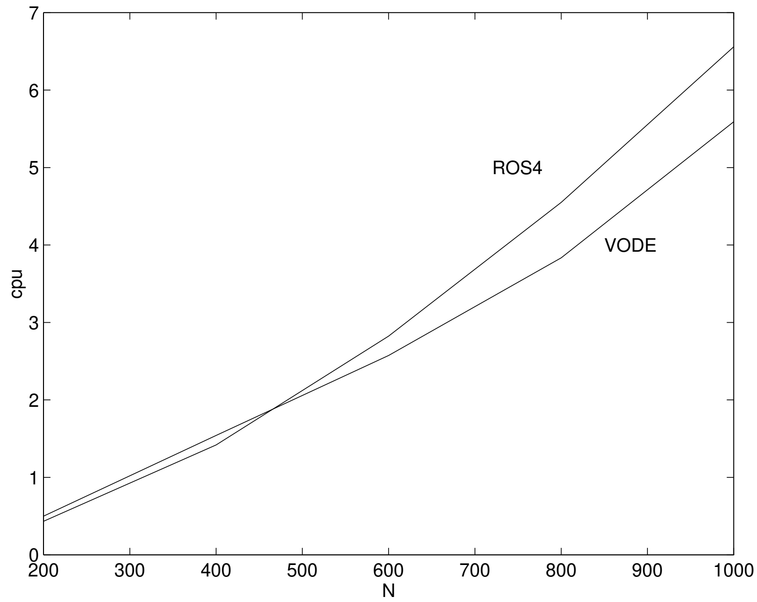

Figure 4.9: Computation times for the Medical Akzo Nobel problem.

Table 4.5: ROS4 and VODE(BDF) on the Medical Akzo Nobel problem.

N		Steps	f-s	J-s	LU-s
200	VODE	499	706	9	67
	ROS4	204	609	201	
600	VODE	735	983	13	104
	ROS4	387	1157	383	
1000	VODE	933	1200	17	133
	ROS4	521	1558	516	

tolerance gets more stringent. The linear systems involved in the integration might then become ill conditioned. This is worst for the Rosenbrock method since evaluations of the linear systems enters the solution directly (see section 3.3.4).

Chapter 5

Conclusions

In this report some of the aspects of the method of lines have been considered. To give a complete survey over the technique would be outside the scope of this report, and a unified treatment of the method of lines might also be difficult to write. One reason is that it is difficult to obtain general results, valid for all partial differential equations and all types of semi discretisation. There is probably no "best way" to apply the method of lines, but there are some aspects that are important to consider.

Then an integrator for the semi discretised system must be chosen it is important to consider the system at hand. For parabolic or diffusion dominated problems the system will have real eigenvalues or eigenvalues close to the negative real axis, thus it is natural to consider an implicit method with extended real stability interval.

We have seen from section 4.1 that one dimensional problems are too stiff for RKC, but in two dimensions in seems like a good choice. And for higher dimensions we might expect RKC to perform even better if we use the same number of equations, since the stiffness of the system decreases with increasing dimensions.

From section 4.2 we have also observed that not all systems arising from the method of lines are stiff. In this case an explicit integrator solves the system quite well. It might even be experienced for some problems that a "normal" explicit integrator is most efficient to use.

In section 4.3 we observed that a Rosenbrock method seems to perform comparable with a Backward differentiation method on this non linear problem, but for more stringent accuracy BDF methods seems preferable.

SDIRK and SIRK methods have not been tested here. But they have very good stability properties even for higher order methods, and it might therefore be worthwhile to consider these methods more closely.

Even though the method of lines are appliccable to a wide range of problems it is difficult to construct "general" MOL solver for PDEs. The problems at hand must be considered regarding both the space discretisation and the choice of integrator for the semi discretised system.

Bibliography

[1] Bui, T.D. *A Note on the Rosenbrock Procedure*. Math. Comp. 33 (1979) no. 147, 971-975.

[2] Burrage, K. *A special family of Runge–Kutta methods for solving stiff differential equations*. BIT 18 (1978), 22-41.

[3] Butcher, J.C. *On the implementation of implicit Runge–Kutta methods*. BIT 16 (1976), 237-240.

[4] Butcher, J.C. *A Transformed Implicit Runge–Kutta Method*. ACM 26 (1979) no. 4, 731-738.

[5] Butcher, J.C. *The Numerical Analysis of Ordinary Differential Equations*. Wiley, New York 1987.

[6] Butcher, J.C. *A history of Runge–Kutta methods*. Appl. Numer. Math. 20 (1996), 247-260.

[7] Fornberg, B. *Generation of Finite Difference Formulas on Arbitrarily Spaced Grids*. Math. Comp. 51 (1988) no. 184, 699-706.

[8] Hairer, E., Nørsett, S.P., Wanner, G. *Solving Ordinary Differential Equations I; Nonstiff Problems*. Springer-Verlag, Berlin 1993.

[9] Hairer, E., Wanner G. *Solving Ordinary Differential Equations II; Stiff and differential-algebraic problems*. Springer–Verlag, Berlin 1996.

[10] Jackson, K.R. *The numerical solution of stiff IVPs for ODEs*. Appl. Numer. Math. 20 (1996), 5-20.

[11] Johnson, C. *Numerical solutions of partial differential equations by the finite element method*. Cambridge University Press, Cambridge 1987 (Or Studentlitteratur, Lund 1987).

[12] Lambert, J.D. *Numerical Methods for Ordinary Differential Systems; The Initial Value Problem*. Wiley, Chichester 1991.

[13] Liskovets, O.A. *The Method of Lines*. Differential equations 1 (1965), 1308-1323.

[14] Mitchell, A.R., Griffiths, D.F. *The Finite Difference Method in Partial Differential Equations.* Wiley, 1980.

[15] Rothe, E. *Zweidimensionale parabolische Randwertaufgaben als Grenzfall eindimensionaler Randwertaufgaben.* Math. Ann. 102 (1930), 650-670.

[16] Schiesser, W.E. *The numerical method of lines; Integration of partial differential equations.* Academic Press Inc., California 1991.

[17] Shampine, L.F. *Implementation of Rosenbrock Methods.* ACM Trans. Math. Software 8 (1982) no 2, 93-113.

[18] Shampine, L.F. *ODE solvers and the Method of Lines.* Num. Meth. for PDE 10 (1994), 739-755.

[19] van der Houwen, P.J. *Construction of integration formulas for initial value problems.* North-Holland, Amsterdam 1977.

[20] van der Houwen, P.J. *The development of Runge–Kutta methods for partial differential equations.* Appl. Numer. Math. 20 (1996), 261-272.

[21] van der Houwen P.J., Sommeijer B.P. *On the internal stability of explicit m-stage Runge–Kutta methods for large values of m.* ZAMM, 60 (1980) 479-485.

[22] Verwer, J.G. *Explicit Runge–Kutta Methods for Parabolic Partial Differential Equations.* (paper prepared for the symposium One Hundred Years of Runge–Kutta methods, held at CWI December 8, 1995)

[23] Verwer, J.G. Hundsdorfer, W. H. Sommeijer, B. P. *Convergence Properties of the Runge–Kutta-Chebyshev Method.* Numer. Math. 57 (1990), 157-178.

[24] Verwer, J.G. Scholz, S. Louter-Nool, M. *A Class of Runge–Kutta–Rosenbrock Methods for Solving Stiff Differential Equations.* ZAMM 63 (1983), 13-20.

[25] `http://www.cwi.nl/cwi/projects/IVPtestset.shtml` *Test set for IVP solvers.* CWI.

[26] `http://www.netlib.org/` Netlib.