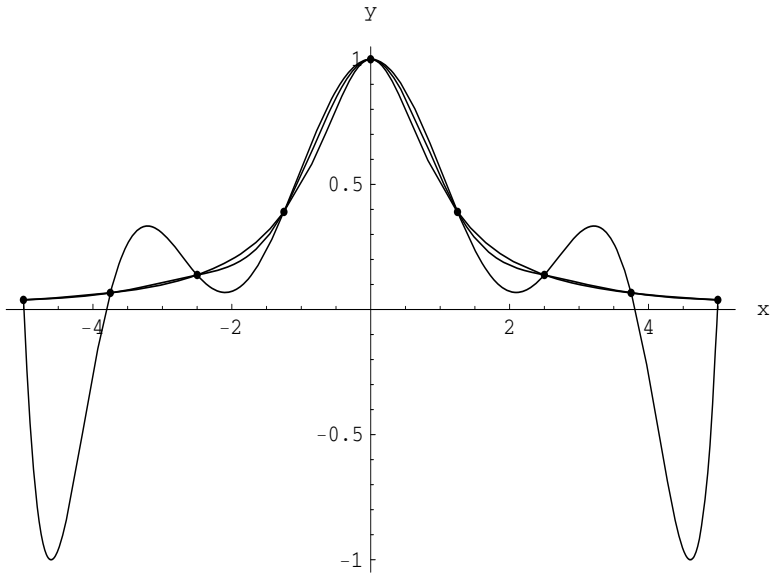
NUMERICAL METHODS AND ALGORITHMS

Milan Kubíček, Drahoslava Janovská, Miroslava Dubcová



Translation from the Czech
Drahoslava Janovská, Pavel Pokorný, Miroslava Dubcová
Original: NUMERICKÉ METODY A ALGORITMY,
Milan Kubíček, Miroslava Dubcová, Drahoslava Janovská,
VŠCHT Praha 2005.

Contents

Chapter 1

Numerical algorithms of linear algebra

The methods of the linear algebra count among the most important areas used at the solution of technical problems: the understanding of numerical methods of linear algebra is important for the understanding of full problems of numerical methods.

In the numerical algebra we encounter two basic variants of problems. The solution of systems of linear equations and the algebraic eigenvalue problem.

The whole range of technical problems leads to the solution of systems of linear equations. The first step in numerical solution of many problems of linear algebra is a choice of an appropriate algorithm

At first we inform readers about the most important knowledge of the numerical linear algebra.

1.1 Fundamental terms in matrix theory

Let us denote $\mathbb{R}^{m\times n}$ the linear space of all real $m\times n$ matrices. Let to $\mathbf{A}\in\mathcal{R}^{m\times n}$, $A=(a_{ij})$, $i=1,\ldots,m$; $j=1,\ldots,n$. If $m\neq n$ then this matrix $\mathbf{A}$ is called a rectangular matrix , if $m=n$ then the matrix is called a square matrix. We denote furthermore the zero matrix $\mathbf{O}$, i.e. the matrix, all of whose element are zero. The square identity matrix is denoted by the symbol $\mathbf{E}$, i.e. $\mathbf{E}=(e_{ij})$, $i,j=1,\ldots,n$, $e_{ii}=1$, $e_{ij}=0$ for $i\neq j$.

Let a matrix has relatively few non-zero elements, then this matrix is called the sparse matrix. A matrix with "few" zero elements is called the dense matrix. Some sparse matrices have a special structure, which we can utilize for algorithmization.

Let $\mathbf{A}=(a_{ij})\in\mathbb{R}^{n\times n}$. We say the matrix $\mathbf{A}$ is

- the diagonal, if $a_{ij}=0$ for $j\neq i$; we write $\mathbf{A}=\mathrm{diag}(a_{11},a_{22},\ldots,a_{nn})$
- the upper triangular matrix, if $a_{ij}=0$ for $i>j$
- strictly upper triangular matrix, if $a_{ij}=0$ pro $i\geq j$
- lower triangular matrix , if $a_{ij}=0$ for $i<j$
- strictly lower triangular matrix, if $a_{ij}=0$ for $i\leq j$
- upper bidiagonal matrix, if $a_{ij}=0$ for $j\neq i,i+1$
- lower bidiagonal matrix, if $a_{ij}=0$ for $j\neq i,i-1$
- tridiagonal matrix, if $a_{ij}=0$ for i, j, such that $|j-i|>1$

- band matrix , if $a_{ij} \neq 0$ for only $i - m_l \leq j \leq i + m_k$, where m_l and m_k are two natural numbers; the number $m_l + m_k + 1$ is called bandwidth of the matrix $\mathbf{A}$
- upper Hessenberg matrix, if $a_{ij} = 0$ for i, j such that $i > j + 1$; accordingly we define lower Hessenberg matrix
- permutation matrix, if the columns of a matrix $\mathbf{A}$ are permutations of the columns of the identity matrix $\mathbf{E}$ (every row and column of permutation matrix involves one and only one unity, others elements are zero.)
- block diagonal matrix, if the matrix is a block matrix in which the blocks off the diagonal are the zero matrices, and the diagonal matrices are square matrices, we write
$$\mathbf{A} = \text{diag } (\mathbf{A}_{11}, \mathbf{A}_{22}, \ldots, \mathbf{A}_{nn})$$
- the block tridiagonal matrix, if the matrix is special block matrix having square matrices (blocks) in the lower diagonal, main diagonal and upper diagonal, with all other blocks being zero matrices.

If a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ has the elements a_{ij}, $i = 1, \ldots, m$, $j = 1, \ldots, n$, then the matrix $\mathbf{A}^T \in \mathbb{R}^{n\times m}$ with the elements a_{ji}, $j = 1, \ldots, n$, $i = 1, \ldots, m$ we call transposed matrix of matrix $\mathbf{A}$. If $\mathbf{A}^T = \mathbf{A}$, we say that $\mathbf{A}$ is symmetric. If $\mathbf{A}^T\mathbf{A} = \mathbf{A}\mathbf{A}^T$, we say that $\mathbf{A}$ is normal. If $\mathbf{Q}^T\mathbf{Q} = \mathbf{Q}\mathbf{Q}^T = \mathbf{E}$ we say that $\mathbf{Q}$ is orthogonal.

If a matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\det \mathbf{A} \neq 0$, we say that $\mathbf{A}$ is regular (nonsingular) . In such case a matrix $\mathbf{A}$ has an inverse matrix $\mathbf{A}^{-1}$, i. e. matrix, for that hold $\mathbf{A}^{-1}\mathbf{A} = \mathbf{A}\mathbf{A}^{-1} = \mathbf{E}$.

We say that a square matrix $\mathbf{A}$ is

- diagonally dominant matrix if
$$|a_{ii}| \geq \sum_{j=1, j\neq i}^{n} |a_{ij}|, \quad i = 1, \ldots, n; \tag{1.1.1}$$
- strictly diagonally dominant if
$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|, \quad i = 1, \ldots, n. \tag{1.1.2}$$

A square matrix $\mathbf{A}$ is called reducible if exists a permutation matrix $\mathbf{P}$ such that a matrix $\mathbf{P}^T\mathbf{A}\mathbf{P}$ is block upper triangular :

$$\mathbf{P}^T\mathbf{A}\mathbf{P} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{0} & \mathbf{A}_{22} \end{pmatrix}.$$

A square matrix that is not reducible is said to be irreducible..

We say that a matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is

- positive definite if $\boldsymbol{x}^T\mathbf{A}\boldsymbol{x} > 0$ for all nonzero $\boldsymbol{x} \in \mathbb{R}^n$;
- positive semidefinite if $\boldsymbol{x}^T\mathbf{A}\boldsymbol{x} \geq 0$ for all $\boldsymbol{x} \in \mathbb{R}^n$;
- negative definite if $\boldsymbol{x}^T\mathbf{A}\boldsymbol{x} < 0$ for all nonzero $\boldsymbol{x} \in \mathbb{R}^n$;
- negative semidefinite if $\boldsymbol{x}^T\mathbf{A}\boldsymbol{x} \leq 0$ for all $\boldsymbol{x} \in \mathbb{R}^n$;
- indefinite if exist nonzero vectors x and y that such $\boldsymbol{x}^T\mathbf{A}\boldsymbol{x}$ is negative and $\boldsymbol{y}^T\mathbf{A}\boldsymbol{y}$ is positive.

1.2 Direct methods for solving systems of linear equations

Numerical methods for solving systems of linear equations divide into two categories: direct methods and iterative methods. Direct methods would give exact solution of problem after finite number of elementary algebraic operations without rounding error. Such direct method is for example Cramer rule for calculation of solution of systems with a nonsingular matrix. This rule is not too practical because calculation of many determinants is needed.

Iterative methods find a solution $\mathbf{x}$ of a given system of linear equations as a limit of a sequence of "approximate" solutions $\mathbf{x}_k$.

Literature dealing with solution of linear equations is very extensive. A reader will find detailed description of algorithms (direct and iterative methods) in books [8], [3], [24]. We limit ourself to only selected problems and methods which are important in chemical engineering.

1.2.1 Conditioning of a system of linear equations

We do not compute accurately because of rounding errors, which affect results of calculations. If, measurements than the elements of the matrix and the right-hand side of the system are usually not accurate numbers. In this section we will study how the solution process is affected by small changes in the problem. Usually this involves concept known as "conditioning".

Example 1.2.1 *Consider two systems of two equations in two unknowns:*

	$u+v$	$=$	2	$u+v$	$=$	2
	$u+1.0001v$	$=$	2.0001	$u+1.0001v$	$=$	2.0002
Solution:	$u=1,$		$v=1$	$u=0,$		$v=2$

This example demonstrates that the change of the fifth digit of one right-hand side gives totally different solution.

Analyze now a solution of a system of linear equations $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ where $\mathbf{A} \in \mathbb{R}^{n\times n}$ is nonsingular, $\boldsymbol{b} \in \mathbb{R}^n$. We will examine how small changes in elements of $\mathbf{A}$ and $\boldsymbol{b}$ affect the solution $\boldsymbol{x} = \mathbf{A}^{-1}\boldsymbol{b}$. The sensitivity of a system is obviously measured by a condition number of the matrix $\mathbf{A}$. Let us mention a vector norm. The vector norm on $\mathbb{R}^n$ is a function $||\cdot|| : \mathbf{R}^n \to \mathbb{R}^1$ with the following properties:

$$||\boldsymbol{x}|| \geq 0, \quad \boldsymbol{x} \in \mathbb{R}^n, \quad ||\boldsymbol{x}|| = 0 \iff \boldsymbol{x} = 0\ , \tag{1.2.1}$$

$$||\boldsymbol{x}+\boldsymbol{y}|| \leq ||\boldsymbol{x}|| + ||\boldsymbol{y}||, \quad \boldsymbol{x},\, \boldsymbol{y} \in \mathbb{R}^n\ , \tag{1.2.2}$$

$$||\alpha\boldsymbol{x}|| = |\alpha| \cdot ||\boldsymbol{x}||, \quad \alpha \in \mathbf{R}^1,\, \boldsymbol{x} \in \mathbb{R}^n\ . \tag{1.2.3}$$

Hereafter we will use particularly following norms:

$$||\boldsymbol{x}||_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}, \quad p \geq 1, \tag{1.2.4}$$

especially

$$||\boldsymbol{x}||_1 = |x_1| + |x_2| + \ldots + |x_n| \tag{1.2.5}$$

$$||\boldsymbol{x}||_2 = \sqrt{\sum_{i=1}^{n}(x_i)^2} \quad \text{(Euclidean norm)} \tag{1.2.6}$$

$$||\boldsymbol{x}||_\infty = \max_{1\le i\le n}|x_i|. \tag{1.2.7}$$

We introduce analogously a matrix norm, for matrices $\mathbf{A}\in\mathbb{R}^{n\times n}$, by

$$||\mathbf{A}||_p = \max_{\boldsymbol{x}\neq\mathbf{0}}\frac{||\mathbf{A}\boldsymbol{x}||_p}{||\boldsymbol{x}||_p}. \tag{1.2.8}$$

Hereafter we will us particularly the following norms:

$$||\mathbf{A}||_1 = \max_{\boldsymbol{x}\neq\mathbf{0}}\frac{||\mathbf{A}\boldsymbol{x}||_1}{||\boldsymbol{x}||_1} = \max_{j=1,\dots,n}\sum_{i=1}^{n}|a_{ij}| \quad \text{(column norm)}, \tag{1.2.9}$$

$$||\mathbf{A}||_\infty = \max_{i=1,\dots,n}\sum_{j=1}^{n}|a_{ij}| \quad \text{(row norm)}. \tag{1.2.10}$$

Remark that from the equality (1.2.8) follows

$$||\mathbf{A}\boldsymbol{x}|| \le ||\mathbf{A}||\cdot||\boldsymbol{x}||. \tag{1.2.11}$$

The equality holds at least for one non-zero $\boldsymbol{x}$.

Examine now how the solution $\boldsymbol{x}$ of a system $\mathbf{A}\boldsymbol{x}=\boldsymbol{b}$, $\mathbf{A}\in\mathbb{R}^{n\times n}$, $\boldsymbol{b}\in\mathbb{R}^n$, changes as the matrix $\mathbf{A}$ and the right-hand-side vector $\boldsymbol{b}$ change. Let us consider the following system:

$$(\mathbf{A}+\varepsilon\mathbf{F})\boldsymbol{x}(\varepsilon) = \boldsymbol{b}+\varepsilon\boldsymbol{g}, \tag{1.2.12}$$

where $\mathbf{F}\in\mathbb{R}^{n\times n}$, $\boldsymbol{g}\in\mathbb{R}^n$, ε is a "small" parameter. If $\mathbf{A}$ is nonsingular the map $\boldsymbol{x}(\varepsilon)$ is differentiable in the neighbourhood of zero

$$\begin{aligned}
\boldsymbol{x}(\varepsilon) &= (\mathbf{A}+\varepsilon\mathbf{F})^{-1}(\boldsymbol{b}+\varepsilon\boldsymbol{g})\\
\dot{\boldsymbol{x}}(\varepsilon) &= \left(\frac{\mathrm{d}}{\mathrm{d}\varepsilon}(\mathbf{A}+\varepsilon\mathbf{F})^{-1}\right)(\boldsymbol{b}+\varepsilon\boldsymbol{g})+(\mathbf{A}+\varepsilon\mathbf{F})^{-1}\boldsymbol{g}\\
\dot{\boldsymbol{x}}(0) &= -\mathbf{A}^{-1}\mathbf{F}\mathbf{A}^{-1}\boldsymbol{b}+\mathbf{A}^{-1}\boldsymbol{g}\\
\dot{\boldsymbol{x}}(0) &= \mathbf{A}^{-1}(\boldsymbol{g}-\mathbf{F}\boldsymbol{x}(0))
\end{aligned}$$

and thus the Taylor expansion of $\boldsymbol{x}(\varepsilon)$ at the point 0 gives

$$\boldsymbol{x}(\varepsilon) = \boldsymbol{x}(0)+\varepsilon\dot{\boldsymbol{x}}(0)+\mathcal{O}(\varepsilon^2).$$

The symbol $\mathcal{O}(\varepsilon^2)$ characterizes the remainder of the Taylor expansion of the map $\boldsymbol{x}(\varepsilon)$ at the point 0. If $R(\varepsilon)$ is this remainder then the equality $R(\varepsilon)=\mathcal{O}(\varepsilon^2)$ means that there exist constants $\alpha>0$ and $A>0$ such that $|R(\varepsilon)|\le A\varepsilon^2$ for all $|\varepsilon|<\alpha$.
We obtain estimation, see [8]:

$$\frac{||\boldsymbol{x}(\varepsilon)-\boldsymbol{x}(0)||}{||\boldsymbol{x}(0)||} \le |\varepsilon|||\mathbf{A}^{-1}||\left(\frac{||\boldsymbol{g}||}{||\boldsymbol{x}(0)||}+||\mathbf{F}||\right)+\mathcal{O}(\varepsilon^2). \tag{1.2.13}$$

For a square nonsingular matrix $\mathbf{A}$ the condition number is defined by $\kappa(\mathbf{A})=||\mathbf{A}||\cdot||\mathbf{A}^{-1}||$. If $\mathbf{A}$ is singular, then $\kappa(\mathbf{A})=+\infty$. According to (1.2.11) $||\boldsymbol{b}||=||\mathbf{A}\boldsymbol{x}(0)||\le||\mathbf{A}||\cdot||\boldsymbol{x}(0)||$ and thus we can rewrite inequality (1.2.13) in the form

$$\frac{||\boldsymbol{x}(\varepsilon)-\boldsymbol{x}(0)||}{||\boldsymbol{x}(0)||} \le \kappa(\mathbf{A})(r_{\mathbf{A}}+r_{\boldsymbol{b}})+\mathcal{O}(\varepsilon^2),$$

where $r_{\mathbf{A}} = |\varepsilon|\dfrac{||\mathbf{F}||}{||\mathbf{A}||}$ and $r_{\boldsymbol{b}} = |\varepsilon|\dfrac{||\boldsymbol{g}||}{||\boldsymbol{b}||}$ are relative errors in $\mathbf{A}$ and $\boldsymbol{b}$.

The relative error of the solution $\boldsymbol{x}(0)$ is thus approximately equal to $\kappa(\mathbf{A})$ multiple of sum of relative errors of $\mathbf{A}$ and $\boldsymbol{b}$. In this sense the number $\kappa(\mathbf{A})$ measures the sensitivity of the problem $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$. Remark that the condition number depends on the used norm. E.g., if $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of a matrix $\mathbf{A}^T\mathbf{A}$ then the spectral norm is defined as $||\mathbf{A}||_* = \max_{i=1,\ldots,n}\sqrt{\lambda_i}$. Since the inverse matrix has reciprocal eigenvalues it holds for regular matrix $\mathbf{A}$

$$\kappa(\mathbf{A}) = \frac{\max\limits_{i=1,\ldots,n}\sqrt{\lambda_i}}{\min\limits_{i=1,\ldots,n}\sqrt{\lambda_i}}\,.$$

The reader will find enough details e.g. in [8], [3], [14].

It is possible to show that for every matrix $\mathbf{A}$ hold

$$\kappa(\mathbf{A}) \geq 1, \tag{1.2.14}$$
$$\kappa(\mathbf{A}) = \kappa(\mathbf{A}^{-1}), \tag{1.2.15}$$
$$\kappa(\alpha\mathbf{A}) = \kappa(\mathbf{A}), \quad \alpha \in \mathbb{R}, \quad \alpha \neq 0. \tag{1.2.16}$$

If $\kappa(\mathbf{A})$ is "low" we say the system is well-conditioned ; it is better conditioned if the condition number $\kappa(\mathbf{A})$ is nearer to 1. If $\kappa(\mathbf{A})$ is "high" we say the system is ill-conditioned. There are methods for decreasing the condition number, e.g. balancing matrix, see [29], [30].

Example 1.2.2 *Let us compute the condition number of the matrix of the system from example 1.2.1:*

$$\mathbf{A} = \begin{pmatrix} 1 & 1 \\ 1 & 1.0001 \end{pmatrix}, \qquad \mathbf{A}^{-1} = 10^4 \begin{pmatrix} 1.0001 & -1 \\ -1 & 1 \end{pmatrix},$$

$$\|\mathbf{A}\|_\infty = 2.0001\,, \quad \|\mathbf{A}^{-1}\|_\infty = 10^4 \cdot 2.0001\,, \quad \kappa(\mathbf{A}) = \|\mathbf{A}\|_\infty \cdot \|\mathbf{A}^{-1}\|_\infty = 40004.001\,.$$

The condition number is relatively high, this indicates that the systems is ill-conditioned. The value of $\det \mathbf{A} = 0.0001$ could indicate it, too. However, unlike (1.2.16) is $\det(\alpha\mathbf{A}) = \alpha^n \det \mathbf{A}$. Generally, there is no relation between $\det(\mathbf{A})$ and condition number $\kappa\mathbf{A}$, this illustrates the following example.

Example 1.2.3 *Let*

$$\mathbf{B}_n = \begin{pmatrix} 1 & -1 & \ldots & -1 \\ 0 & 1 & \ldots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 \end{pmatrix} \in \mathbb{R}^{n\times n} \Rightarrow \mathbf{B}_n^{-1} = \begin{pmatrix} 1 & 1 & 2 & 4 & 8 & \ldots & 2^{n-2} \\ 0 & 1 & 1 & 2 & 4 & \ldots & 2^{n-3} \\ \cdot & & \cdot & \cdot & & & \cdot \\ \cdot & & & \cdot & \cdot & & \cdot \\ \cdot & & & & \cdot & \cdot & \cdot \\ 0 & & & & & 1 & 1 \\ 0 & & & & & & 1 \end{pmatrix}.$$

We can easily get

$$\det(\mathbf{B}_n) = 1, \quad ||\mathbf{B}_n||_\infty = \max_{1\leq i\leq n}\sum_{j=1}^{n}|b_{ij}| = n,$$

$$||\mathbf{B}_n^{-1}||_\infty = \max_{1\leq i\leq n}\sum_{j=1}^{n}|\widehat{b}_{ij}| = 2 + \sum_{j=1}^{n-2}2^i = 2^{n-1}.$$

Finally $\kappa(\mathbf{B}_n) = n \cdot 2^{n-1}$, *depends on* n *while* $\det(\mathbf{B}_n)$ *not.*

Example 1.2.4 *Let*

$$\mathbf{D}_n = \mathrm{diag}(10, \ldots, 10) \in \mathbb{R}^{n\times n} \quad \Rightarrow \quad \mathbf{D}_n^{-1} = \mathrm{diag}\,(0.1; 0.1; \ldots; 0.1).$$

Then

$$\det(\mathbf{D}_n) = 10^n, \quad ||\mathbf{D}_n||_\infty = 10, \quad ||\mathbf{D}_n^{-1}||_\infty = 0.1.$$

Therefore

$$\kappa(\mathbf{D}_n) = 1.$$

For calculation of the condition number of a given matrix $\mathbf{A}$, we would have to know the inverse matrix $\mathbf{A}^{-1}$. But the most affective algorithm for calculation of inverse matrix is three times laborious than solving problem by elimination (see subsection 1.2.2). For our purposes the estimation of condition number $\kappa(\mathbf{A})$ is sufficient. Let us reason the vector norm (1.2.7) and the corresponding matrix norm (1.2.10). The calculation of $||\mathbf{A}||$ is easy. The basic idea of estimation of $||\mathbf{A}^{-1}||$ consists in following relations: if $\mathbf{A}\boldsymbol{y} = \boldsymbol{d}$ then from inequality (1.2.11) results

$$||\boldsymbol{y}|| = ||\mathbf{A}^{-1}\boldsymbol{d}|| \le ||\mathbf{A}^{-1}|| \cdot ||\boldsymbol{d}||, \quad \text{i. e.} \quad ||\mathbf{A}^{-1}|| \ge \frac{||\boldsymbol{y}||}{||\boldsymbol{d}||},$$

and for condition number we can obtain an estimation

$$\kappa(\mathbf{A}) = ||A|| \cdot ||A^{-1}|| \ge ||\mathbf{A}|| \frac{||\boldsymbol{y}||}{||\boldsymbol{d}||}. \tag{1.2.17}$$

The large number on the right-side (1.2.17) indicates the probable ill-conditioning of the matrix $\mathbf{A}$. The reader will find enough details e.g. in [8], [14].

1.2.2 Gaussian elimination

Elimination methods are based on an addition of a multiple of some row to the given row so that the matrix of the system is simplified. Let us illustrate the whole method by the following algorithm.

Let us consider the systems of equations

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \vdots \qquad\qquad & \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n\,. \end{aligned} \tag{1.2.18}$$

Let us suppose $a_{11} \neq 0$. We divide first equation by this coefficient:

$$\frac{a_{12}}{a_{11}} \to a_{12},\ \frac{a_{13}}{a_{11}} \to a_{13},\ \cdots,\ \frac{a_{1n}}{a_{11}} \to a_{1n},\ \frac{b_1}{a_{11}} \to b_1,\ 1 \to a_{11}\,.$$

Let us eliminate the unknown x_1 from 2-nd, 3-rd to n-th row of the system (1.2.18). We will subtract the relevant multiple of the new first equation from 2-nd, 3-rd to n-th equations:

$$\begin{aligned} &a_{i2} - a_{i1}a_{12} \to a_{i2},\ a_{i3} - a_{i1}a_{13} \to a_{i3}, \ldots,\ a_{in} - a_{i1}a_{1n} \to a_{in}, \\ &b_i - a_{i1}b_1 \to b_i,\ 0 \to a_{i1} \qquad\qquad i = 2, 3, \ldots, n\,. \end{aligned}$$

Let us go over to the second equation: let us suppose now the new element $a_{22} \neq 0$. Again we will divide second equation by this element:

$$\frac{a_{23}}{a_{22}} \to a_{23},\ \frac{a_{24}}{a_{22}} \to a_{24},\ \cdots,\ \frac{a_{2n}}{a_{22}} \to a_{2n},\ \frac{b_2}{a_{22}} \to b_2,\ 1 \to a_{22}\,.$$

Similarly to the previous step we will eliminate the unknown x_2 from 3-rd, 4-th to n-th equation:

$$\begin{aligned} &a_{i3} - a_{i2}a_{23} \to a_{i3},\ a_{i4} - a_{i2}a_{23} \to a_{i4}, \ldots,\ a_{in} - a_{i2}a_{2n} \to a_{in}, \\ &b_i - a_{i2}b_2 \to b_i,\ 0 \to a_{i2} \qquad\qquad i = 3, 4, \ldots, n\,. \end{aligned}$$

Thus we go until $(n-1)$-th equation, which we divide by the element $a_{n-1,n-1} \neq 0$. Then we eliminate $(n-1)$-th unknown from n-th equation , so the matrix of the final system has the form (the elements below the main diagonal are zero):

$$\mathbf{A} = \begin{pmatrix} 1 & a_{12} & a_{13} & \ldots & & a_{1n} \\ & 1 & a_{23} & \ldots & & a_{2n} \\ & & 1 & \ldots & & a_{3n} \\ & & & \ddots & & \vdots \\ & & & & 1 & a_{n-1,n} \\ & & & & & a_{nn} \end{pmatrix}.$$

Reducing the original matrix $\mathbf{A}$ into the upper triangular form is called forward elimination.

From the last equation we compute x_n easily:

$$\frac{b_n}{a_{nn}} \to x_n\,.$$

From the $(n-1)$-st equation we obtain x_{n-1}:

$$b_{n-1} - a_{n-1,n}x_n \to x_{n-1}.$$

The same way we obtain all x_i. The process of progressively solving for the unknowns $x_{n-1}, x_{n-2}, \ldots, x_1$ is called backward elimination step of Gaussian elimination. Number of multiplications and divisions necessary to find the solutions of the system of n equations using previous algorithm is $\frac{1}{3}n(n^2 + 3n - 1)$. Previous algorithm is not universal. During computation we suppose that the diagonal elements of matrix $\mathbf{A}$ are not zero. If these elements are near to zero we can decrease an accuracy of calculation. Thus the Gaussian elimination is to be modified so that we choose to transfer (e. g. by interchanging rows) to the diagonal a proper element with the largest absolute value (so-called pivot).

The algorithm with pivoting is relatively complicated. There is Gaussian elimination without backward solution step (so-called Gauss-Jordan elimination). In this method we eliminate coefficients not only below leader ("diagonal") elements but also above this element. In the method without pivoting the identity matrix results after forward step of the method, in the method with pivoting is the permutation matrix results.

Gaussian elimination can be also used to computation of determinants. The determinant of matrix $\mathbf{A}$ is equal to the product of leader elements (before division of row by this element) in the Gaussian elimination without pivoting and is equal to the product of leader elements times determinant of permutation matrix in case of Gauss-Jordan elimination.

1.2.3 Systems with tridiagonal matrix

Frequently, we meet with systems of equations with tridiagonal matrix (see 1.1). These systems arise by the construction of difference analogy for boundary value problem of ordinary differential equations (see chapter 5).

If we write the systems of equations in the matrix form

$$\begin{pmatrix} d_1 & h_1 & & & & \\ s_2 & d_2 & h_2 & & & \\ & s_3 & d_3 & h_3 & & 0 \\ & & & \ddots & & \\ & 0 & & s_{n-1} & d_{n-1} & h_{n-1} \\ & & & & s_n & d_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix} \tag{1.2.19}$$

The tridiagonal matrix algorithm (TDMA), also known as the Thomas algorithm, is a simplified form of Gaussian elimination that can be used to solve tridiagonal systems of equations. Zero elements are not saved in the memory and also the operations "$0-x.0 \to 0$" are not realized. This leads to significant computer memory and time savings. Analogously we can derive algorithms for five-diagonal systems or for tridiagonal systems with few nonzero elements out of diagonal etc.

1.3 Iterative methods for linear systems

In contrast to the direct methods of Section 1.2 are the iterative methods. For an arbitrary starting vector $\boldsymbol{x}^{(0)}$, these methods generate a sequence of approximate solutions $\{\boldsymbol{x}^{(k)}\}$, $k > 0$, that converges to the solution of the given problem. The quality of an iterative method depends on how quickly the iterates $\boldsymbol{x}^{(k)}$ converge to the solution.

In this Section, we present only a particular class of iterative methods, so called point iterative methods (only one entry of the vector is corrected per one iteration) and block iterative methods (a group of entries of the vector is corrected in one iteration).

More details about iterative methods for solution of linear systems can be found for example in [8], [24], [3].

1.3.1 Point iterative methods

The simplest point iterative scheme for the solution of the system of linear equations

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b}, \quad \mathbf{A} = (a_{ij})_{i,j=1,\dots,n}\,, \quad \mathbf{A} \in \mathbb{R}^{n\times n}, \quad \boldsymbol{b} \in \mathbb{R}^n, \tag{1.3.1}$$

is the Jacobi method. It is defined for matrices $\mathbf{A}$ that have nonzero diagonal elements.

One iteration step that produces an improved approximation $\boldsymbol{x}^{(k+1)}$ from the previous $\boldsymbol{x}^{(k)}$ represents a solution of $i-$th equation in (1.3.1) with respect to x_i. All other components x_j are transferred to the right-hand side of the $i-$th equation:

$$\underset{\substack{\uparrow \\ (k+1)-\text{st} \\ \text{iteration}}}{x_i^{(k+1)}} = \frac{1}{a_{ii}}\,(b_i - \sum_{j=1, j\neq i}^{n} a_{ij} \underset{\substack{\uparrow \\ k-\text{th} \\ \text{iteration}}}{x_j^{(k)}})\,, \qquad i = 1, \dots, n\,. \tag{1.3.2}$$

Jacobi method has a disadvantage: it requires us to keep all the components of $\boldsymbol{x}^{(k)}$ until the calculation of $\boldsymbol{x}^{(k+1)}$ is complete. A much more natural idea is to start using each component of the new vector $\boldsymbol{x}^{(k+1)}$ as soon as it is corrected. At the moment we are computing $x_i^{(k+1)}$ we can use already updated components

$$x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_{i-1}^{(k+1)}.$$

We obtain the iteration procedure called the Gauss–Seidel method:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right), \quad i = 1, \ldots, n\,. \tag{1.3.3}$$

For the Gauss–Seidel method, the latest approximations to the components of $\boldsymbol{x}$ are used in the update of subsequent components. It is convenient to overwrite the old components of $\boldsymbol{x}^{(k)}$ with those of $\boldsymbol{x}^{(k+1)}$ as soon as they are computed.

Let us remark that in application of Jacobi method we don't need to care about the order of corrected components in the vector $\boldsymbol{x}^{(k)}$. On the contrary, in the Gauss–Seidel method the order of unknowns is fixed (here $i = 1, \ldots, n$).

The convergence rate of the Gauss–Seidel method often can be improved by introducing a relaxation parameter ω. The method is called SOR (successive overrelaxation) method. It is defined by

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) + (1-\omega) x_i^{(k)}, \quad i = 1, \ldots, n\,. \tag{1.3.4}$$

Let x_i^{GS} denote the $(k+1)-$st iteration of the Gauss–Seidel method, i.e. x_i^{GS} is the right-hand side of (1.3.3). Then the equation (1.3.4) for the $(k+1)-$st iteration of the SOR method can be written in the form

$$x_i^{(k+1)} = x_i^{(k)} + \omega(x_i^{GS} - x_i^{(k)}), \quad i = 1, \ldots, n\,.$$

Let us try to rewrite the methods (1.3.2), (1.3.3) and (1.3.4) in a matrix form.

A general technique to derive an iterative method is based on a splitting of the matrix $\mathbf{A}$, $\mathbf{A} = \mathbf{B} - (\mathbf{B} - \mathbf{A})$, being $\mathbf{B}$ a suitable nonsingular matrix (called the preconditioner of $\mathbf{A}$). Then $\mathbf{B}\boldsymbol{x} = (\mathbf{B} - \mathbf{A})\boldsymbol{x} + \boldsymbol{b}$. Correspondingly, we define the following general iterative method

$$\boldsymbol{x}^{(k+1)} = \mathbf{B}^{-1} \left((\mathbf{B} - \mathbf{A})\boldsymbol{x}^{(k)} + \boldsymbol{b} \right), \qquad k \geq 0. \tag{1.3.5}$$

The matrix $\mathbf{G} = \mathbf{B}^{-1}(\mathbf{B} - \mathbf{A}) = \mathbf{E} - \mathbf{B}^{-1}\mathbf{A}$ is called the iteration matrix associated with (1.3.5).

Let $\mathbf{A} = (a_{ij}) \in \mathbb{R}^{n \times n}$ be written in the form

$$\mathbf{A} = \mathbf{D} - \mathbf{L} - \mathbf{U}\,,$$

where $\mathbf{D} = \mathrm{diag}(a_{11}, \ldots, a_{nn})$ is the diagonal of $\mathbf{A}$, $\mathbf{L} = (l_{ij})$ is the strictly lower triangular matrix whose non null entries are $l_{ij} = -a_{ij},\ i = 2, \ldots, n,\ j = 1, \ldots, i-1$, and $\mathbf{U} = (u_{ij})$ is the strictly upper triangular matrix whose non null entries are $u_{ij} = -a_{ij},\ i = 1, \ldots, n-1,\ j = i+1, \ldots, n$. Let $\mathbf{A}$ have nonzero diagonal entries. On the contrary, the matrix $\mathbf{L}+\mathbf{U}$ has all diagonal entries equal to zero.

If $\mathbf{B}$ in (1.3.5) is taken to be the diagonal $\mathbf{D}$ of $\mathbf{A}$ then the corresponding iteration procedure

$$\boldsymbol{x}^{(k+1)} \;=\; \mathbf{D}^{-1}\left((\mathbf{L}+\mathbf{U})\boldsymbol{x}^{(k)} \;+\; \boldsymbol{b}\right). \tag{1.3.6}$$

is the Jacobi method.

The Gauss–Seidel method corresponds to the choice $\mathbf{B} = \mathbf{D} - \mathbf{L}$, i.e. for ordering $i = 1, \ldots, n$ we obtain

$$\mathbf{D}\boldsymbol{x}^{(k+1)} \;=\; \boldsymbol{b} + \mathbf{L}\boldsymbol{x}^{(k+1)} + \mathbf{U}\boldsymbol{x}^{(k)}\,,$$

therefore

$$\boldsymbol{x}^{(k+1)} \;=\; (\mathbf{D}-\mathbf{L})^{-1}\left(\mathbf{U}\boldsymbol{x}^{(k)} \;+\; \boldsymbol{b}\right). \tag{1.3.7}$$

Now, let $\mathbf{B} = \mathbf{D} - \omega\mathbf{L}$, where $\omega \in \mathbb{R}$ is the relaxation parameter. Since $\mathbf{L} = \mathbf{D} - \mathbf{U} - \mathbf{A}$ we have

$$\mathbf{D} - \omega\mathbf{L} = (1-\omega)\mathbf{D} + \omega\mathbf{U} + \omega\mathbf{A}\,.$$

In matrix terms, the SOR step is given by

$$\boldsymbol{x}^{(k+1)} = (\mathbf{D}-\omega\mathbf{L})^{-1}((1-\omega)\mathbf{D} + \omega\mathbf{U})\boldsymbol{x}^{(k)} + \omega(\mathbf{D}-\omega\mathbf{L})^{-1}\boldsymbol{b}\,. \tag{1.3.8}$$

The relaxation parameter ω in the SOR method has to satisfy $\omega \in (0,2)$. For a few typical problems, the optimal value of the relaxation parameter is known, see Section **xxx7.1.1.1xxx**. In more complicated problems, however, it may be necessary to perform a sophisticated eigenvalue analysis in order to determine an appropriate ω. Let us remark that for $\omega = 1$ the SOR reduces to the Gauss–Seidel method. The methods of this Section are treated for example in [24], [3], [8]. We will discuss the application of these methods for solving of the elliptic partial differential equations in Section **xxx7.1.1.1xxx**.

1.3.2 Block iterative methods

Block versions of the Jacobi, Gauss–Seidel, and SOR iterations are easily defined. In one iteration, the block relaxation methods update more than one components of the solution vector. To illustrate, we consider an example of the matrix $\mathbf{A} \in \mathbb{R}^{6\times 6}$:

$$\mathbf{A} \;=\; \left(\begin{array}{cc|c|ccc} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \\ \hline a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \end{array}\right), \quad \boldsymbol{x} \;=\; \left(\begin{array}{c} x_1 \\ x_2 \\ \hline x_3 \\ \hline x_4 \\ x_5 \\ x_6 \end{array}\right), \quad \boldsymbol{b} \;=\; \left(\begin{array}{c} b_1 \\ b_2 \\ \hline b_3 \\ \hline b_4 \\ b_5 \\ b_6 \end{array}\right).$$

We divide the matrix $\mathbf{A}$ into blocks. Correspondingly, we divide also the vectors $\boldsymbol{x}$ a $\boldsymbol{b}$:

$$\mathbf{A} \;=\; \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{pmatrix}, \quad \boldsymbol{x} \;=\; \begin{pmatrix} \boldsymbol{\xi}_1 \\ \boldsymbol{\xi}_2 \\ \boldsymbol{\xi}_3 \end{pmatrix}, \quad \boldsymbol{b} \;=\; \begin{pmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \\ \boldsymbol{\beta}_3 \end{pmatrix},$$

where $\mathbf{A}_{ij}$, $i,j = 1,2,3$, are blocks of the original matrix, $\boldsymbol{\xi}_i$ a $\boldsymbol{\beta}_i$ are vectors of the corresponding size. For example

$$\mathbf{A}_{13} = \begin{pmatrix} a_{14} & a_{15} & a_{16} \\ a_{24} & a_{25} & a_{26} \end{pmatrix}, \quad \boldsymbol{\xi}_3 = \begin{pmatrix} x_4 \\ x_5 \\ x_6 \end{pmatrix}, \quad \boldsymbol{\beta}_1 \;=\; \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}.$$

Similarly as in the point iterative methods, we rewrite the matrix $\mathbf{A}$ as

$$\mathbf{A} = \mathbf{D} - \mathbf{L} - \mathbf{U},$$

where in this case

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \mathbf{A}_{22} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \mathbf{A}_{33} \end{pmatrix}, \ \mathbf{L} = -\begin{pmatrix} \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{A}_{21} & \mathbf{O} & \mathbf{O} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{O} \end{pmatrix}, \ \mathbf{U} = -\begin{pmatrix} \mathbf{O} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ \mathbf{O} & \mathbf{O} & \mathbf{A}_{23} \\ \mathbf{O} & \mathbf{O} & \mathbf{O} \end{pmatrix}.$$

Now we can define the block Jacobi iterative method as the method that in one iteration updates some of the vectors $\boldsymbol{\xi}_i$:

$$\mathbf{A}_{ii}\boldsymbol{\xi}_i^{(k+1)} = (\mathbf{L}+\mathbf{U})\boldsymbol{\xi}_i^{(k)} + \boldsymbol{\beta}_i, \quad \text{i.e.}$$

$$\boldsymbol{\xi}_i^{(k+1)} = \mathbf{A}_{ii}^{-1}\left((\mathbf{L}+\mathbf{U})\boldsymbol{\xi}_i^{(k)}\right) + \mathbf{A}_{ii}^{-1}\boldsymbol{\beta}_i, \quad i = 1,2,3.$$

Similarly as in (1.3.6), we obtain:

$$\boldsymbol{x}^{(k+1)} = \mathbf{D}^{-1}\left((\mathbf{L}+\mathbf{U})\boldsymbol{x}^{(k)} + \boldsymbol{b}\right). \tag{1.3.9}$$

The only difference is that the matrices $\mathbf{D}$, $\mathbf{L}$ and $\mathbf{U}$ represent the block analog of the matrices in (1.3.6).

The block versions of the Gauss–Seidel and SOR methods are defined likewise.

We will investigate the block iterative methods and their convergence properties in Section **xxx7.1xxx** within the context of the solution of the partial differential equations by making use of finite differences. Namely, the matrices arising from finite difference approximations of partial derivatives are often block tridiagonal matrices.

1.4 Eigenvalues and eigenvectors of a matrix

The eigenvalue problem is a problem of considerable theoretical interest and wide-ranging applications. For example, this problem is crucial in solving systems of differential equations and in the stability analysis of nonlinear systems. Eigenvalues and eigenvectors are also used in estimates for the convergence rate of iterative methods (not only in Linear Algebra).

Let a real or complex $n \times n$ matrix $\mathbf{A}$ be given. A number $\lambda \in \mathbb{C}$ is called an eigenvalue of the matrix $\mathbf{A}$ if there exists a non-zero vector $\boldsymbol{x}$ such that

$$\mathbf{A}\boldsymbol{x} = \lambda\boldsymbol{x}. \tag{1.4.1}$$

Every such vector $\boldsymbol{x}$ is called an eigenvector of $\mathbf{A}$ associated with the eigenvalue λ. The set of all eigenvalues is called the spectrum of $\mathbf{A}$.

Algebraically, the eigenvectors are just those vectors such that multiplication by $\mathbf{A}$ has a very simple form – the same as multiplication by a scalar (the eigenvalue). Geometrically it means that the linear transformation $\boldsymbol{y} = \mathbf{A}\boldsymbol{x}$ doesn't change the direction of the vector $\boldsymbol{x} \neq \mathbf{0}$; in general only its length is changed.

If $\boldsymbol{x} \neq \mathbf{0}$ is an eigenvector associated with the eigenvalue λ of $\mathbf{A}$ then any nonzero scalar multiple of $\boldsymbol{x}$ is also an eigenvector. If the eigenvalue λ has a nonzero imaginary part then also the associated eigenvector is, in general, complex–valued (not real).

The equation (1.4.1) can be written in the form

$$\begin{array}{rcl} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &=& \lambda x_1\,, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &=& \lambda x_2\,, \\ \vdots & & \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &=& \lambda x_n\,, \end{array} \tag{1.4.2}$$

i.e. as a homogeneous system of linear equations with the matrix

$$\begin{pmatrix} a_{11}-\lambda & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22}-\lambda & \dots & a_{2n} \\ \vdots & & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn}-\lambda \end{pmatrix} = \mathbf{A} - \lambda\mathbf{E}. \tag{1.4.3}$$

The system (1.4.2) has a nonzero solution $\boldsymbol{x}$ if and only if the determinant of the matrix (1.4.3) is equal to zero, i.e.

$$\det(\mathbf{A} - \lambda\mathbf{E}) = 0\,. \tag{1.4.4}$$

The equation (1.4.4) is called the characteristic equation of the matrix $\mathbf{A}$ and

$$P(\lambda) = \det(\mathbf{A} - \lambda\mathbf{E}) \tag{1.4.5}$$

is the characteristic polynomial of the matrix $\mathbf{A}$.

Recall that a $k-$by$-k$ principal submatrix of $n \times n$ matrix $\mathbf{A}$ is one lying in the same set of k rows and columns and that a $k-$by$-k$ principal minor is the determinant of such a principal submatrix. There are $\binom{n}{k}$ different $k-$by$-k$ principal minors of $\mathbf{A}$.

The characteristic polynomial $P(\lambda)$ has degree n,

$$P(\lambda) = (-1)^n \left(\lambda^n + p_1\lambda^{n-1} + p_2\lambda^{n-2} + \ldots + p_n\right), \tag{1.4.6}$$

where $p_k\,,\ k = 1, \ldots, n\,,$ is (up to the sign $(-1)^k$) equal to the sum of all $k-$by$-k$ principal minors of $\mathbf{A}$, in particular

$$-p_1 = a_{11} + a_{22} + \ldots + a_{nn} = \text{trace of } \mathbf{A}\,, \tag{1.4.7}$$

$$p_n = (-1)^n \det \mathbf{A}\,. \tag{1.4.8}$$

The roots $\lambda_1, \lambda_2, \ldots, \lambda_n$ of the characteristic equation (1.4.4) are the eigenvalues of the matrix $\mathbf{A}$. Due to the well-known relations between the zeroes of a polynomial and its coefficients we obtain

$$\lambda_1 + \lambda_2 + \ldots + \lambda_n = -p_1 = a_{11} + a_{22} + \ldots + a_{nn} \tag{1.4.9}$$

and

$$\lambda_1\lambda_2\cdots\lambda_n = (-1)^n p_n = \det\mathbf{A}. \tag{1.4.10}$$

Let us remark that if we compare the solution of a system of linear equations $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ and the computation of eigenvalues of the matrix $\mathbf{A}$, $\mathbf{A}\boldsymbol{x} = \lambda\boldsymbol{x}$, there is a substantial difference: If we solve a system of equations with a real matrix then the solution is also real, but the eigenvalues of a real matrix might have a nonzero imaginary part.

If $\lambda_1, \ldots, \lambda_k$ are the distinct zeros of the characteristic polynomial $P(\lambda)$, then $P(\lambda)$ can be represented in the form

$$P(\lambda) = (-1)^n(\lambda - \lambda_1)^{\alpha_1}(\lambda - \lambda_2)^{\alpha_2}\cdots(\lambda - \lambda_k)^{\alpha_k}\,.$$

The integer α_i is called the (algebraic) multiplicity of the eigenvalue $\lambda_i\,,\ i = 1, \ldots, k\,.$

Example 1.4.1 *Determine all the eigenvalues, and an associated eigenvector for each, of the matrix*

$$\mathbf{A} = \begin{pmatrix} 2 & 1 \\ 3 & 0 \end{pmatrix}. \tag{1.4.11}$$

The matrix $\mathbf{A}$ *has the characteristic polynomial*

$$\det(\mathbf{A} - \lambda\mathbf{E}) = \begin{vmatrix} 2-\lambda & 1 \\ 3 & -\lambda \end{vmatrix} = \lambda^2 - 2\lambda - 3.$$

The zeros of this quadratic polynomial, i.e. the eigenvalues are $\lambda_1 = 3$, $\lambda_2 = -1$. *The (algebraic) multiplicity of both these eigenvalues is equal to* 1 .
Let us find the associated eigenvectors:

$$\mathbf{A}\boldsymbol{x}_1 = \lambda_1\boldsymbol{x}_1 \implies \mathbf{A}\boldsymbol{x}_1 = 3\boldsymbol{x}_1 \implies (\mathbf{A} - 3\mathbf{E})\boldsymbol{x}_1 = \mathbf{0}.$$

The matrix $\mathbf{A} - 3\,\mathbf{E}$ *is singular,*

$$\mathbf{A} - 3\,\mathbf{E} = \begin{pmatrix} -1 & 1 \\ 3 & -3 \end{pmatrix}.$$

The eigenvector associated to the eigenvalue $\lambda_1 = 3$ *is for example the vector* $\boldsymbol{x}_1 = (1,1)^T$. *Analogously, the associated eigenvector to the eigenvalue* $\lambda_2 = -1$ *is for example the vector* $\boldsymbol{x}_2 = (1,-3)^T$.

Example 1.4.2 *Let us find the eigenvalues and the associated eigenvectors of the matrices*

$$\mathbf{A} = \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix}, \qquad \mathbf{B} = \begin{pmatrix} 3 & 1 \\ 0 & 3 \end{pmatrix}, \tag{1.4.12}$$

The matrix $\mathbf{A}$ *has the characteristic equation* $(3-\lambda)^2 = 0$, *i.e.* $\mathbf{A}$ *has the only eigenvalue* $\lambda_{\mathbf{A}} = 3$ *with the (algebraic) multiplicity* 2. *For the associated eigenvector we obtain*

$$(\mathbf{A} - 3\,\mathbf{E})\boldsymbol{x} = \mathbf{0}\,, \quad \textit{i.e.} \quad \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}\boldsymbol{x} = \mathbf{0}.$$

The solution of this equation consists of all two–dimensional space and as the associated eigenvectors may serve any two linearly independent vectors, for example $\boldsymbol{x}_1 = (1,0)^T$, $\boldsymbol{x}_2 = (0,1)^T$.

The matrix $\mathbf{B}$ *has also the only eigenvalue* $\lambda_{\mathbf{B}} = 3$ *with the (algebraic) multiplicity* 2, *but in this case we find only one (up to a nonzero scalar multiple) associated eigenvector* $\boldsymbol{x} = (1,0)^T$.

Example 1.4.3 *Consider the matrix*

$$\mathbf{A} = \begin{pmatrix} 2 & 1 \\ -5 & 0 \end{pmatrix}. \tag{1.4.13}$$

It has the characteristic polynomial

$$det\ (\mathbf{A} - \lambda\mathbf{E}) = \begin{vmatrix} 2-\lambda & 1 \\ -5 & -\lambda \end{vmatrix} = \lambda^2 - 2\lambda + 5.$$

The roots of this quadratic polynomial are $\lambda_1 = 1 + 2i$, $\lambda_2 = 1 - 2i$, *i.e. in this case, the matrix* $\mathbf{A}$ *has a complex conjugate pair of eigenvalues. The multiplicity of both eigenvalues*

is 1.
Let us calculate the associated eigenvectors:

$$\mathbf{A}\boldsymbol{x}_1 = \lambda_1 \boldsymbol{x}_1 \implies \mathbf{A}\boldsymbol{x}_1 = (1+2i)\boldsymbol{x}_1 \implies (\mathbf{A} - (1+2i)\mathbf{E})\boldsymbol{x}_1 = \mathbf{0}.$$

Since

$$\mathbf{A} - (1+2i)\,\mathbf{E} = \begin{pmatrix} 1-2i & 1 \\ -5 & -1-2i \end{pmatrix},$$

the eigenvector $\boldsymbol{x}_1$ *associated to the eigenvalue* $\lambda_1 = 1+2i$ *is (up to a nonzero multiple) the vector* $\boldsymbol{x}_1 = (-1, 1-2i)^T$. *Similarly, the eigenvector associated to* $\lambda_2 = 1-2i$ *is the vector* $\boldsymbol{x}_2 = (-1, 1+2i)^T$, *that is the entries of the associated eigenvectors are also complex conjugate.*

We will present two methods for finding coefficients of the characteristic polynomial of the higher degree.

The first one, Krylov method, is based on the Cayley–Hamilton theorem. This theorem is often paraphrased as "every square matrix satisfies its own characteristic equation", but this must be understood carefully: The scalar polynomial $P(t)$ is first computed as $P(t) = \det(\mathbf{A} - t\mathbf{E})$, and one forms the matrix $P(\mathbf{A})$ from the characteristic polynomial. Then

$$P(\mathbf{A}) = \mathbf{0}, \tag{1.4.14}$$

or equivalently

$$\mathbf{A}^n + p_1\mathbf{A}^{n-1} + \cdots + p_{n-1}\mathbf{A} + p_n\mathbf{E} = \mathbf{0}. \tag{1.4.15}$$

Our aim is to find the coefficients $p_1, p_2, \ldots, p_n$ in (1.4.6). We multiply the equation (1.4.15) from the right by a nonzero vector $\boldsymbol{x}^{(0)}$:

$$\mathbf{A}^n\boldsymbol{x}^{(0)} + p_1\mathbf{A}^{n-1}\boldsymbol{x}^{(0)} + \cdots + p_n\boldsymbol{x}^{(0)} = \mathbf{0}. \tag{1.4.16}$$

We obtain a system of equations

$$\begin{pmatrix} x_1^{(n-1)} & x_1^{(n-2)} & \ldots & x_1^{(0)} \\ x_2^{(n-1)} & x_2^{(n-2)} & \ldots & x_2^{(0)} \\ \vdots & \vdots & \vdots & \vdots \\ x_n^{(n-1)} & x_n^{(n-2)} & \ldots & x_n^{(0)} \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix} = \begin{pmatrix} -x_1^{(n)} \\ -x_2^{(n)} \\ \vdots \\ -x_n^{(n)} \end{pmatrix}, \tag{1.4.17}$$

where we denoted $\boldsymbol{x}^{(k)} = \mathbf{A}^k\boldsymbol{x}^{(0)}$, $\boldsymbol{x}^{(k)} = (x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)})$, $k = 0, 1, \ldots, n$. Since $\mathbf{A}^k\boldsymbol{x}^{(0)} = \mathbf{A}\mathbf{A}^{k-1}\boldsymbol{x}^{(0)} = \mathbf{A}\boldsymbol{x}^{(k-1)}$, it is not necessary to compute the powers of the matrix $\mathbf{A}$; to obtain the vector $\boldsymbol{x}^{(k)}$ we have only to multiply the vector $\boldsymbol{x}^{(0)}$ $k-$times by the matrix $\mathbf{A}$ from the left. If the system (1.4.17) for $p_1, \ldots, p_n$ is uniquely solvable then this solution gives the desired coefficients of the characteristic polynomial. If the system (1.4.17) is not uniquely solvable we have to choose a different vector $\boldsymbol{x}^{(0)}$. The system (1.4.17) can be solved for example using the Gaussian elimination.

Example 1.4.4 *Using the Krylov method, compute the characteristic polynomial of the matrix (1.4.11).*
Let us set $\boldsymbol{x}^{(0)} = (1, 0)^T$. *Then*

$$\mathbf{A}\boldsymbol{x}^{(0)} = \boldsymbol{x}^{(1)} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}, \quad \mathbf{A}\boldsymbol{x}^{(1)} = \boldsymbol{x}^{(2)} = \begin{pmatrix} 7 \\ 6 \end{pmatrix}.$$

The system (1.4.17) has the form

$$\begin{pmatrix} 2 & 1 \\ 3 & 0 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \begin{pmatrix} -7 \\ -6 \end{pmatrix},$$

i.e. the coefficients of the characteristic polynomial are $p_1 = -2$, $p_2 = -3$, the same result as in the Example 1.4.1.

Let us briefly discuss the second method for computing the coefficients of the characteristic polynomial, so called interpolation method. For a given $n \times n$ matrix $\mathbf{A}$,

$$P(\lambda) \,=\, \det(\mathbf{A} \,-\, \lambda \mathbf{E})$$

is the characteristic polynomial of $\mathbf{A}$. Its degree is n. If we choose $n+1$ different values λ:

$$\lambda^{(0)}, \lambda^{(1)}, \ldots, \lambda^{(n)}$$

and compute

$$P(\lambda^{(i)}) = \det\,(\mathbf{A} - \lambda^{(i)}\mathbf{E}), \quad i = 0, 1, 2, \ldots, n\,,$$

we can construct explicitly the Lagrange interpolation polynomial (see the part Lagrange interpolation). In view of the uniqueness of polynomial interpolation this must be also the characteristic polynomial. The coefficient by λ^n should be equal to $(-1)^n$, if it is not the case then the other coefficients are also not correct.

Example 1.4.5 *We compute the characteristic polynomial of the matrix (1.4.11) using the interpolating method.*
Let us set $\lambda^{(0)} = 0$, $\lambda^{(1)} = 1$, $\lambda^{(2)} = 2$. The corresponding determinants are

$$P(0) = \begin{vmatrix} 2 & 1 \\ 3 & 0 \end{vmatrix} = -3, \quad P(1) = \begin{vmatrix} 1 & 1 \\ 3 & -1 \end{vmatrix} = -4, \quad P(2) = \begin{vmatrix} 0 & 1 \\ 3 & -2 \end{vmatrix} = -3.$$

Then the Lagrange interpolation polynomial is

$$L_2(\lambda) = -3\frac{(\lambda-1)(\lambda-2)}{(-1)\cdot(-2)} - 4\frac{\lambda(\lambda-2)}{1\cdot(-1)} - 3\frac{\lambda(\lambda-1)}{2\cdot 1} = \lambda^2 - 2\lambda - 3\,,$$

i.e. the result is again the same as in Example 1.4.1.

1.4.1 Location of eigenvalues

It is natural to ask whether one can say anything useful about the eigenvalues of a given matrix. For example in some differential equations problems involving the long–term stability of an oscillating system, one is sometimes interested in showing that the eigenvalues $\{\lambda_i\}$ of a matrix all lie in the left half–plane, that is, that $\Re(\lambda_i) < 0$. Sometimes in statistic or numerical analysis one needs to show that a Hermitian matrix is positive definite, that is, that all $\lambda_i > 0$. We give a simple estimate for eigenvalues. It may serve, e.g. to locate the eigenvalues of a matrix and to study their sensitivity with respect to small perturbations.

We know that the characteristic polynomial has exactly n in general complex–valued roots. We are able to choose the largest one in absolute value. If λ_k are the eigenvalues of the $n \times n$ matrix $\mathbf{A}$, then

$$\rho(\mathbf{A}) = \max_{1 \le k \le n} |\lambda_k|$$

is called the spectral radius of $\mathbf{A}$.

The following theorem (often called the Gershgorin disc theorem) gives a simple estimate of the spectral radius.

Let $\mathbf{A} = (a_{jk})$ be a given $n \times n$ matrix. The union of all discs

$$K_j = \{\, \mu \in \mathbb{C}, \quad |\mu - a_{jj}| \leq \sum_{k=1, k\neq j}^{n} |a_{jk}| \,\} \tag{1.4.18}$$

contains all eigenvalues of the matrix $\mathbf{A}$.

Let us notice that the sets $K_j = (S_j, r_j)$ are discs centered at $S_j = a_{jj}$ with radius $r_j = \sum_{k=1, k\neq j}^{n} |a_{jk}|$. Moreover, since $\mathbf{A}$ and $\mathbf{A}^T$ have the same eigenvalues (they have the same characteristic equation), to obtain more information about the location of the eigenvalues we can apply (1.4.18) to $\mathbf{A}$ as well as to $\mathbf{A}^T$.

Example 1.4.6 *Matrices (1.4.11) and (1.4.12) have the spectral radius $\rho(\mathbf{A}) = 3$. The Gershgorin discs for the matrix (1.4.13) are:*

$$K_1 \equiv (S_1, r_1) \equiv (2, 1), \quad K_2 \equiv (S_2, r_2) \equiv (0, 5).$$

Therefore, two eigenvalues of the matrix (1.4.13) are located in the union $K_1 \cup K_2$. The spectral radius of this matrix is $\rho(\lambda) = \sqrt{5}$ and eigenvalues $\lambda_{1,2} = 1 \pm 2i$ lie in the disc K_2.

1.4.2 Methods for determining the eigenvalues and eigenvectors

Throughout this section, we will suppose that the matrix $\mathbf{A}$ is $n \times n$, where $n > 5$. For $n \leq 5$, it has no sense to use complicated numerical methods, since the eigenvalues can be computed exactly as roots of the characteristic polynomial.

Let $\boldsymbol{x} \neq \mathbf{0}$ be an eigenvector associated to the eigenvalue λ, i.e. $\mathbf{A}\boldsymbol{x} = \lambda\boldsymbol{x}$. Let $\mathbf{T}$ be an arbitrary nonsingular $n \times n$ matrix and let $\boldsymbol{y} = \mathbf{T}^{-1}\boldsymbol{x}$, i.e. $\boldsymbol{x} = \mathbf{T}\boldsymbol{y}$. Then

$$\mathbf{T}^{-1}\mathbf{A}\mathbf{T}\boldsymbol{y} = \mathbf{T}^{-1}\mathbf{A}\boldsymbol{x} = \lambda\mathbf{T}^{-1}\boldsymbol{x} = \lambda\boldsymbol{y}, \quad \boldsymbol{y} \neq \mathbf{0},$$

which implies that $\boldsymbol{y}$ is an eigenvector of the transformed matrix $\mathbf{B} = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}$ associated with the same eigenvalue λ. Such transformations are called similarity transformations and $\mathbf{B}$ is said to be similar to $\mathbf{A}$, $\mathbf{B} \sim \mathbf{A}$. Similarity of matrices is an equivalence relation. Similar matrices have not only the same eigenvalues, but also the same characteristic polynomial. Moreover, the multiplicity of the eigenvalues remains the same.On the other hand, let us notice that having the same eigenvalues is a necessary but not sufficient condition for similarity.

The most common methods for determining the eigenvalues and eigenvectors of a dense matrix $\mathbf{A}$ proceed as follows. By means of a finite number of similarity transformations one first transforms the matrix $\mathbf{A}$ into a matrix $\mathbf{B}$ of simpler form,

$$\mathbf{B} = \mathbf{A}_m = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}, \qquad \mathbf{T} = \mathbf{T}_1 \cdot \mathbf{T}_2 \cdots \mathbf{T}_m,$$

and then determines the eigenvalues λ and eigenvectors $\boldsymbol{y}$ of the matrix $\mathbf{B}$, $\mathbf{B}\boldsymbol{y} = \lambda\boldsymbol{y}$. For $\boldsymbol{x} = \mathbf{T}\boldsymbol{y}$, since $\mathbf{B} = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}$, we then have $\mathbf{A}\boldsymbol{x} = \lambda\boldsymbol{x}$, i.e., to the eigenvalue λ of $\mathbf{A}$ there belongs the eigenvector $\boldsymbol{x}$.

The matrix $\mathbf{B}$ is chosen in such a way that

- the determination of the eigenvalues and eigenvectors of $\mathbf{B}$ is as simple as possible (i.e., requires as few operations as possible);
- the eigenvalue problem for $\mathbf{B}$ is not worse conditioned than that for $\mathbf{A}$.

For symmetric matrices, we have the following very useful result of Schur:
If $\mathbf{A} \in \mathbb{R}^{n\times n}$ is symmetric , then there exists a real orthogonal matrix $\mathbf{Q}$ such that

$$\mathbf{Q}^T\mathbf{A}\mathbf{Q} = \mathbf{\Lambda} = \text{diag}\,(\lambda_1, \lambda_2, \ldots, \lambda_n), \tag{1.4.19}$$

the diagonal entries of the matrix $\mathbf{\Lambda}$ are eigenvalues $\lambda_i,\ i = 1, \ldots, n$, of the matrix $\mathbf{A}$, and these eigenvalues are real. The $i-$th column $\boldsymbol{x}_i$ of $\mathbf{Q}$ is an eigenvector belonging to the eigenvalue λ_i: $\mathbf{A}\boldsymbol{x}_i = \lambda_i\boldsymbol{x}_i$. $\mathbf{A}$ thus has n linearly independent pairwise orthogonal eigenvectors.

Since $\mathbf{Q}$ is orthogonal, we have $\mathbf{Q}^{-1} = \mathbf{Q}^T$. The theorem says that any real symmetric matrix $\mathbf{A}$ is orthogonally similar to a diagonal matrix. The matrix $\mathbf{A}$ is said to be diagonalizable by an orthogonal similarity transformation.

On this idea, Jacobi method, one of the earliest matrix algorithms for computation of eigenvalues of a symmetric matrix, is based. This technique is of current interest because it is amenable to parallel computation and because under certain circumstances it has superior accuracy.

The method of Jacobi employs similarity transformations with the special unitary matrices, so called (Givens or Jacobi) plane rotation matrices. For a given real symmetric matrix $\mathbf{A}$, it produces an infinite sequence of matrices $\mathbf{A}_k$, $k = 0, 1, 2, \ldots$, converging to a diagonal matrix

$$\mathbf{D} = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix},$$

where the $\lambda_j\ ,\ j = 1, \ldots, n$ are just the eigenvalues of $\mathbf{A}$.

As a result of Jacobi method we obtain the similarity transformation

$$\mathbf{D} = \mathbf{V}^T\mathbf{A}\mathbf{V},$$

where $\mathbf{D}$ is diagonal and $\mathbf{V}$ is orthogonal. More precisely, the diagonal matrix $\mathbf{D}$ is the limit of the sequence of matrices $\mathbf{A}_k$ for $k \to \infty$ and $\mathbf{V}$ is the product of all rotation matrices that were used for the diagonalization. The $k-$th column of the matrix $\mathbf{V}$ is the eigenvector associated to the eigenvalue λ_k. Therefore although Jacobi mathod does not, in general, terminate after finitely many plane rotations, it produces a diagonal matrix as well as an orthonormal set of eigenvectors.

Jacobi method is designed to solve the complete eigenvalue problem of the real symmetric matrix. It might work also in nonsymmetric cases, but then the sequence of, in general complex, matrices $\mathbf{A}_k$ converges to an upper triangular matrix.

The mostly used iterative method for computing eigenvalues of a general matrix is the QR method. The algorithm is based on the QR decomposition of a sequence of matrices.

Let $\mathbf{A}$ be a given real matrix. Let us set $\mathbf{A}_0 = \mathbf{A}$ and find the QR decomposition of the matrix $\mathbf{A}_0$, i.e., we decompose the matrix $\mathbf{A}_0$ into a product of an orthogonal matrix $\mathbf{Q}_0$ and an upper triangular matrix $\mathbf{R}_0$, $\mathbf{A}_0 = \mathbf{Q}_0\mathbf{R}_0$ Now we interchange the order of the

multiplication and set $\mathbf{A}_1 = \mathbf{R}_0\mathbf{Q}_0$. The matrix $\mathbf{A}_1$ is orthogonally similar to the matrix $\mathbf{A}_0$:

$$\mathbf{Q}_0^T\mathbf{A}_0\mathbf{Q}_0 = \mathbf{Q}_0^T(\mathbf{Q}_0\mathbf{R}_0)\mathbf{Q}_0 = \mathbf{A}_1,$$

i.e. the matrices $\mathbf{A}_0$ and $\mathbf{A}_1$ have the same eigenvalues. Note that a QR decomposition always exists and that it can be computed in a numerically stable way using e.g. Householder matrices.

In general, the $k-$th iterative step of the QR method can be described as follows:

$$\text{a)} \quad \text{we decompose the matrix } \mathbf{A}_k: \qquad \mathbf{A}_k = \mathbf{Q}_k\mathbf{R}_k, \quad \mathbf{Q}_k^{\mathrm{T}}\mathbf{Q}_k = \mathbf{E}\,, \tag{1.4.20}$$

$$\text{b)} \quad \text{the next approximation } \mathbf{A}_{k+1}: \qquad \mathbf{A}_{k+1} = \mathbf{R}_k\mathbf{Q}_k. \tag{1.4.21}$$

If all the eigenvalues of $\mathbf{A}$ are real and have distinct absolute values, the matrices $\mathbf{A}_k$ converge to an upper triangular matrix as $k \to \infty$. Moreover, the diagonal elements of $\mathbf{A}_k$ converge to the eigenvalues in their natural order, $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$.

If $\mathbf{A}$ has any nonreal eigenvalues or if two eigenvalues have the same modulus the QR method has to be modified, see e.g. [29], [8].

We just described the original form of the QR method. It has some drawbacks, e.g. the convergence is very slow if some quotients $|\lambda_j/\lambda_k|$ of $\mathbf{A}$ are close to 1. The method will be improved substantially if

- one applies the QR method only to reduced matrices, namely, matrices of Hessenberg form, or in case of symmetric matrices, to symmetric tridiagonal matrices. A general matrix, therefore, must first be reduced to one of these forms by means of suitable Givens rotation matrices or Householder reflection matrices;
- one applies so-called shift strategy;
- one implements the double implicit shift strategy if the matrix has nonreal eigenvalues. We will not discuss this technique here.

Let us turn our attention to the QR method with shifts. Let the number α_k is "closed" to an eigenvalue of the given matrix $\mathbf{A}$. We replace the equations (1.4.20) and (1.4.21) by:

$$\mathbf{A}_k - \alpha_k\mathbf{E} = \mathbf{Q}_k\mathbf{R}_k, \tag{1.4.22}$$

$$\mathbf{A}_{k+1} = \mathbf{R}_k\mathbf{Q}_k + \alpha_k\mathbf{E}. \tag{1.4.23}$$

The number α_k is referred to as a shift (of the spectrum) of the matrix $\mathbf{A}_k$. The matrix $\mathbf{A}_{k+1}$ is again orthoganally similar to the matrix $\mathbf{A}_k$:

$$\mathbf{Q}_k^T\mathbf{A}_k\mathbf{Q}_k = \mathbf{Q}_k^T(\mathbf{Q}_k\mathbf{R}_k + \alpha_k\mathbf{E})\mathbf{Q}_k = \mathbf{A}_{k+1}.$$

Let us suppose that our matrix is tridiagonal or Hessenberg matrix. In practical implementation, the shift parameter α_k is chosen as the (n,n) entry of the matrix $\mathbf{A}_k$. If we shift by this quantity during each iteration then the convergence of the $(n,n-1)$ entry to zero, i.e., the convergence of the (n,n) entry to the smallest eigenvalue λ_n, is quadratic, for symmetric tridiagonal matrix even cubic.

A lot of different methods for computation of eigenvalues and eigenvectors can be find in literature. We can recommend for example [8], [3], [29].

* * *

For additional references and a detailed description of eigenvalues, eigenvectors and of efficient methods for their computation, see e.g. [3], [8], [24], [26], [29], [14].

Chapter 2

Interpolation, numerical differentiation and integration

In this section, we will study the approximation of values, derivatives and integrals of the functions, which are not given analytically but only by their function values at given points. Occasionally, the values of their derivatives at these points are also prescribed. Here, we will be interested only in the case when we have "just enough" information to approximate the function. The case when we have more information than needed, but not quite precise, is postponed to the Section "Experimental data".

As a classical example of the approximation of function values may serve the Taylor polynomial. It exploits the values of derivatives at a given point. If the function f has n derivatives at $x_0 \in (a, b)$, then

$$T_n(x) = f(x_0) + \frac{f'(x_o)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n.$$

It is well known that the approximation error is in this case given by formula

$$R_n(x) = \frac{f^{(n+1)}(c)}{(n+1)!}(x - x_0)^{n+1}, \tag{2.0.1}$$

where c is a number (not further specified) between the points x_0 a x.

2.1 Formulation of the interpolation problem

Interpolation is a basic tool for the approximation of a given function. It is frequently used to interpolate function values gathered from tables. If we need to determine the approximate value of the function $f(x)$ at the point x, which is not contained in the table, we construct a function $\phi(x)$, which has the same values as $f(x)$ at the table points. At the other points of a given interval (a, b) from the domain of the function f, the function ϕ approximates f with a certain precision.

The construction of such a function ϕ is called the interpolation problem. Interpolation is used also in such cases when we know the analytical formula for a given f, but this formula is too complicated or time consuming and we have to compute a large number of values. Then we choose only some points $x_0, x_1, \ldots, x_n$ at which we compute the formula

exactly. The values of f at the rest of points we just interpolate from values at points $x_0, x_1, \ldots, x_n$.

Let us consider a family (finite or infinite) of real functions ϕ_i of a single variable defined on an interval (a, b). Let us suppose that every finite system of these functions is linearly independent. As the "coordinate functions" ϕ_i, we usually choose a sequence of powers of x: $1, x, x^2, x^3, \ldots$, a sequence of trigonometric functions: $1, \sin x, \cos x, \sin 2x, \cos 2x, \ldots$, a sequence of exponential functions: $1, e^{\alpha_1 x}, e^{\alpha_2 x}, \ldots$, etc.

Let us take the first $n+1$ functions of the sequence $\{\phi_i\}$ and let us form all possible linear combinations

$$\phi(x) = a_0\phi_0(x) + a_1\phi_1(x) + \ldots + a_n\phi_n(x). \tag{2.1.1}$$

In the interval (a, b), let as choose $m+1$ knots $x_0, x_1, \ldots, x_m$, $x_i \neq x_j$ for $i \neq j$. For values of the functions f and ϕ at these points, we require $f(x_j) = \phi(x_j)$, i.e.,

$$f(x_j) = a_0\phi_0(x_j) + a_1\phi_1(x_j) + \ldots + a_n\phi_n(x_j), \quad j = 0, 1, \ldots, m. \tag{2.1.2}$$

In other words, we want to determine constants $a_0, a_1, \ldots, a_n$ so that (2.1.2) holds. In some cases, if both f and ϕ are differentiable, we can require the derivatives to be identical at these knots, too.

The given function can be also interpolated using so called splines. For example in the case of cubic splines we approximate the given function by the function ϕ, which is assumed to be twice continuosly differentiable in $\langle x_0, x_m\rangle$ and to coincide with some cubic polynomial on every subinterval $\langle x_i, x_{i+1}\rangle$, $i = 0, \ldots, m-1$.

The equation (2.1.2) represents $m+1$ equations for $n+1$ unknowns $a_0, a_1, \ldots, a_n$. If we want to compute the coefficients a_i for an arbitrary function f, the rank of the matrix of the system has to be equal to $m+1$, otherwise the equations would be linearly dependent, i.e., $n \geq m$. Since we are looking for a unique solution of the system (2.1.2), $n = m$. Therefore, let us suppose that $m = n$ and that the determinant

$$\Delta = \begin{vmatrix} \phi_0(x_0) & \phi_1(x_0) & \ldots & \phi_n(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & \ldots & \phi_n(x_1) \\ \vdots & & & \\ \phi_0(x_n) & \phi_1(x_n) & \ldots & \phi_n(x_n) \end{vmatrix} \tag{2.1.3}$$

doesn't vanish. Then there exists a unique solution of the system (2.1.2) for arbitrary values $f(x_j)$.

From the equation (2.1.2), it follows by Cramer's rule that

$$\phi(x) = f(x_0)\Phi_0(x) + f(x_1)\Phi_1(x) + \ldots + f(x_n)\Phi_n(x), \tag{2.1.4}$$

where functions $\Phi_i(x)$ are linear combinations (dependent on the interpolation knots) of the functions $\phi_i(x)$.

2.2 Lagrange interpolation polynomial

Let us choose as the system of functions ϕ_i in (2.1.1) the sequence $1, x, x^2, \ldots, x^n$. Then

$$\phi(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_n x^n. \tag{2.2.1}$$

If we suppose that $x_i \neq x_j$ for $i \neq j$, then in (2.1.3) the determinant $\Delta \neq 0$. The solution (2.1.4) of the interpolating problem can be written in the form

$$L_n(x) = \sum_{i=0}^{n} f(x_i) \frac{\omega_n(x)}{(x - x_i)\omega_n'(x_i)}, \tag{2.2.2}$$

where we set

$$L_n(x) = \phi(x), \quad \omega_n(x) = (x - x_0)(x - x_1) \cdots (x - x_n)$$

and $\omega_n'(x_i)$ is the first derivative of ω_n with respect to x evaluated at the point x_i, i.e.,

$$\omega_n'(x_i) = (x_i - x_0) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n).$$

The interpolation polynomial (2.2.2) is called the Lagrange interpolation polynomial. It can be expressed also in the form

$$L_n(x) = \sum_{i=0}^{n} f(x_i) l_i(x), \quad \text{where} \quad l_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}.$$

Note that the polynomials $l_i(x)$ satisfy

$$l_i(x_k) = \begin{cases} 0 & k \neq i \\ 1 & k = i \end{cases}.$$

It implies that $L_n(x_k) = f(x_k)$.

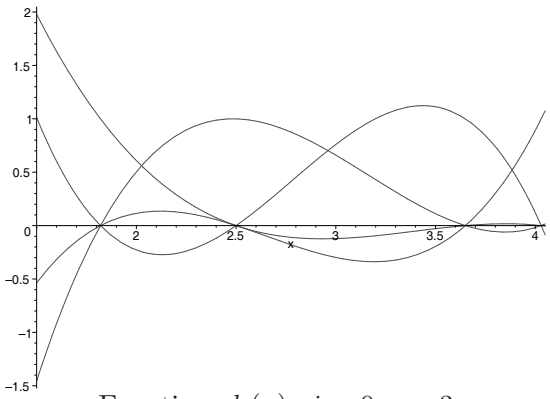
Let $E(x) = f(x) - L_n(x)$ be the error of the approximation of the function f by the Lagrange interpolation polynomial. Since $L_n(x_j) = f(x_j)$ in knots x_j, we have $E(x_j) = 0$ for $j = 0, 1, \ldots, n$. We wish to ask how well the interpolating polynomial reproduces f for arguments different from the knots. The error $E(x)$, where $x \neq x_j$, $j = 0, 1, \ldots, n$, can become arbitrarily large unless some restrictions are imposed on f. For example if the function f has n continuous derivatives on (a, b) and for all $x \in (a, b)$ and an $(n+1)-$st derivative $f^{(n+1)}(x)$ exists, then

$$E(x) = \omega_n(x) \cdot \frac{f^{(n+1)}(\xi_x)}{(n+1)!}, \tag{2.2.3}$$

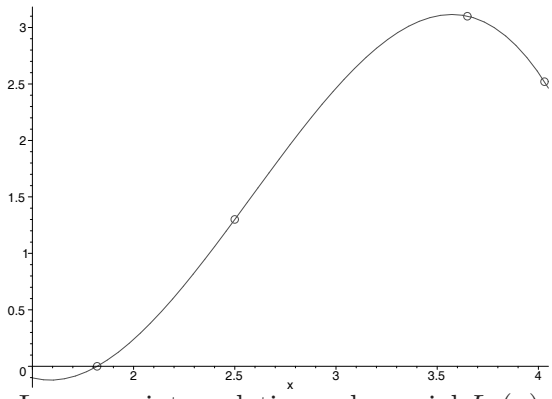
where $a < \xi_x < b$, ξ_x depends on x and on the knots. If the derivative $f^{(n+1)}(x)$ doesn't change too much on (a, b), the error depends substantially on the behaviour of the polynomial $\omega_n(x)$. This polynomial doesn't depend on the interpolated function, only on the knots. By a suitable choice of the knots x_j, $j = 0, 1, \ldots, n$, the error of the approximation can decrease. Note that $\omega_n(x_j) = 0$, $j = 0, 1, \ldots, n$.

Formula (2.2.3) has the similar form as formula (2.0.1) for the error of the approximation of the function f by Taylor polynomial. And similarly as for Taylor polynomial, formula (2.2.3) is not very useful in practical error estimates.

The error of the approximation depends on a number of knots and on their position in the interval (a, b). The error is smallest in the middle of the interval and it increases near the points a, b (this is not quite correct, because the error is equal to zero in the knots). This implies the strategy for the choice of the knots: if we want to approximate the function value at the point x, we choose (if it is possible) the knots to the right and to the left from the point x in such a way that x lies closely to the center of the interval (a, b).



Functions $l_i(x)$, $i = 0, \ldots, 3$



Lagrange interpolation polynomial $L_3(x)$

Figure 2.1: Example 2.2.1

Note that the approximation $\phi(x)$ of the function $f(x)$ at the point x, which lies outside the interval (a, b) containing the interpolation knots, is sometimes called extrapolation. In this case, the value of $|\omega_n(x)|$ grows very fast and therefore the error is large.

While theoretically important, Lagrange interpolation is not suitable for actual calculations, particularly for large numbers n of knots. On the other hand it is useful in situations in which we interpolate many different functions at the same knots.

Example 2.2.1 *We will interpolate the data in the following table of values by the Lagrange polynomial $L_3(x)$,*

x	$1,82$	$2,50$	$3,65$	$4,03$
y	$0,00$	$1,30$	$3,10$	$2,52$

The Lagrange polynomial $L_3(x)$ has (after some computations) the form

$$L_3(x) = 9.420183 - 14.10596x + 6.41469x^2 - 0.82862x^3.$$

Functions $l_i(x)$ and polynomial $L_3(x)$ are depicted on Fig.2.1.

Example 2.2.2 *Using the Lagrange interpolation polynomial $L_8(x)$ at knots $x_j = -5 + 5j/4$, $j = 0, \ldots, 8$, we approximate so called function of Runge*

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

on $\langle -5, 5\rangle$. We also compute the maximal value of the error $|f(x) - L_8(x)|$ on the interval $\langle -5, 5\rangle$. The tableau of the values:

x	-5	-3.75	-2.5	-1.25	0	$1.,25$	2.5	3.75	5
$f(x)$	0.0385	0.0664	0.1379	0.3902	1	0.3902	0.1379	0.0664	0.0385

The values in the table are rounded while the results of the following computations are not.

$$\begin{aligned} L_8(x) \quad = \quad & 0.14 \cdot 10^{-3}\, x^8 + 0.22 \cdot 10^{-11}\, x^7 - 0.006581\, x^6 + 0.13 \cdot 10^{-9}\, x^5 + \\ & + 0.098197\, x^4 + 0.52 \cdot 10^{-9}\, x^3 - 0.528162\, x^2 - 0.1 \cdot 10^{-8}\, x + 1, \end{aligned}$$

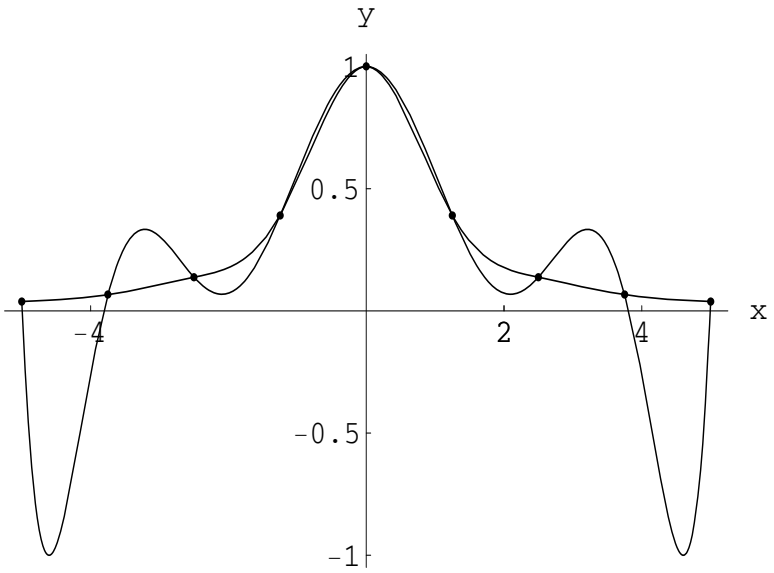


Figure 2.2: Example 2.2.2

$$\max_{x\in\langle -5,5\rangle} |f(x) - L_8(x)| = 1.045241279.$$

Functions $f(x)$ and $L_8(x)$ are depicted on Fig. 2.2. One can see that near the end points of the interval $\langle -5, 5\rangle$ the approximation is worse.

The data in the table of values of the function $f(x)$ are symmetric with respect to the axis y, which implies that also coefficients of odd powers of x should be equal to zero. They are really substantially smaller than coefficients of the even powers. Due to the rounding errors, they are not precisely zero. We would obtain practically the same result by using the coordinate functions 1, x^2, x^4, x^6, x^8 and the table values for $x \geq 0$.

In the previous example, we used the equidistant knots. In some cases, we can obtain better results for nonequidistant knots. In Example 2.2.2, let us choose the knots as Chebyshev points, i.e., as the zeros of Chebyshev polynomials, cf. Example 2.2.3.

Example 2.2.3 *Let us approximate Runge's function from the previous example by making use of Lagrange interpolation polynomial $L_8(x)$ at knots*

$$-4.92404;\ -4.33013;\ -3.21394;\ -1.7101;\ 0;\ 1.7101;\ 3.21394;\ 4.33013;\ 4.92404\,.$$

The table of values:

x	-4.924	-4.330	-3.213	-1.710	0	1.710	3.214	4.330	4.924
$f(x)$	0.0396	0.0506	0.0883	0.2548	1	0.2548	0.0883	0.0506	0.0396

The values in the table are rounded while the results of the following computations are not.

$$\begin{aligned} L_8(x) \;=\; & 4,5\cdot 10^{-5}\,x^8 + 2,7\cdot 10^{-20}\,x^7 - 0.00258\,x^6 - 3,5\cdot 10^{-18}\,x^5 + \\ & +0,05016\,x^4 - 5,6\cdot 10^{-17}\,x^3 - 0,38054\,x^2 + 2,4\cdot 10^{-16}\,x + 1, \end{aligned}$$

$$\max_{x\in\langle -5,5\rangle} |f(x) - L_8(x)| = 0,119846.$$

The functions $f(x)$ and $L_8(x)$ are depicted on Fig. 2.3.

The result will be even better if we approximate the Runge's function by splines.

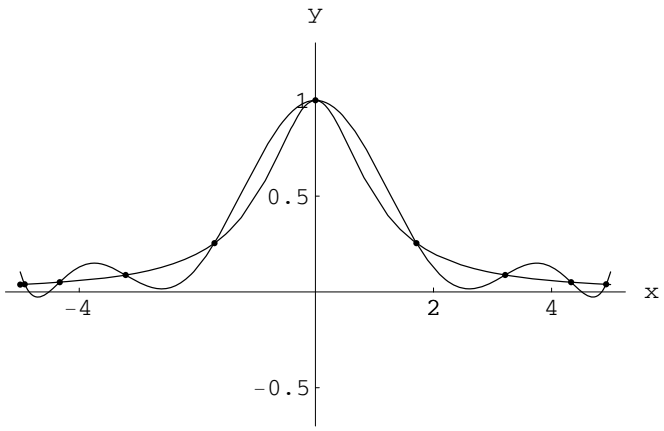


Figure 2.3: Example 2.2.3

2.3 Hermite interpolation polynomial

If not only values of the function f are given, but also its first derivatives (in general, the values of all derivatives up to the order k) at the knots we can approximate the function values using so called Hermite interpolation polynomial.

Consider the real numbers x_i, $f(x_i)$, $f'(x_i)$, $i = 0, \ldots, m$, $x_0 < x_1 < \ldots < x_m$. The Hermite interpolation problem for these data consists of determining a polynomial H whose degree does not exceed $2m + 1$, and which satisfies

$$H(x_i) = f(x_i), \quad H'(x_i) = f'(x_i), \quad i = 0, 1, \ldots, m. \tag{2.3.1}$$

There are exactly $2(m+1)$ conditions (2.3.1) for the $2(m+1)$ coefficients of the interpolating polynomial. It can be shown that for $x_0 < x_1 < \ldots < x_m$ there exists precisely one polynomial H, degree of $H \leq 2m + 1$, which satisfies these conditions.

Analogously as for the Lagrange interpolation polynomial, the Hermite interpolation polynomial can be given explicitly. At first, we define for $i = 0, 1, \ldots, m$

$$\phi_i(x) = \prod_{j=0, j\neq i}^{m} \left(\frac{x - x_j}{x_i - x_j} \right)^2 = \frac{\omega_m^2(x)}{(x - x_i)^2 (\omega_m'(x_i))^2}, \tag{2.3.2}$$

$$\psi_i(x) = (x - x_i)\phi_i(x). \tag{2.3.3}$$

Then the Hermite interpolation polynomial has the form

$$H(x) = \sum_{i=0}^{m} f(x_i) \left(\phi_i(x) - \phi_i'(x_i)\psi_i(x) \right) + \sum_{i=0}^{m} f'(x_i)\psi_i(x) \quad .$$

2.4 Interpolation by spline functions

The error of the polynomial interpolation depends strongly on the length of the interval $\langle a, b \rangle$, which contains the nodes. If we reduce this length, we will get a better approximation. Let us split the interval $\langle a, b \rangle$ into subintervals $\langle x_i, x_{i+1} \rangle$, $a = x_0 < x_1 < \cdots < x_n = b$. On each subinterval, we approximate $f(x)$ by a polynomial. The approximations over all subintervals form an interpolant on $\langle a, b \rangle$ called a spline. A key issue is how smoothly the polynomials connect at the knots.

The simplest continuous spline is one that is piecewise linear, that is, $S(x)$ is a broken-line function. If $S(x)$ is required to interpolate $f(x)$ at the knots x_i and x_{i+1}, then $S(x)$ is Lagrange interpolation polynomial $L_1(x)$ on each $\langle x_i, x_{i+1}\rangle$, $0 \le i \le n-1$:

$$L_1(x) = S(x) = f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i). \tag{2.4.1}$$

The linear interpolation spline (2.4.1) is very easy to evaluate once the proper subinterval has been located. The disadvantage of this interpolation is that the derivatives of the interpolant are discontinuous at the knots.

The higher the degree, the more accurate the approximation, but the greater the possibility of unwanted oscillations. A good compromise seems to be the use of cubic polynomials.

Let $a = x_0 < x_1 < \ldots < x_n = b$ be the dividing of the interval $\langle a, b\rangle$. Then the cubic spline S on this division is a real function $S : \langle a, b\rangle \to \mathbb{R}$, which has the following properties:

- $S \in \mathbb{C}^2(\langle a, b\rangle)$, i.e. S is two times continuously differentiable on $\langle a, b\rangle$;
- the restriction of S on each subinterval $\langle x_i, x_{i+1}\rangle$, $i = 0, 1, \ldots, n-1$, is a polynomial of the third degree.

Thus, the cubic spline consists of the polynomials of the third degree that match in the interior knots together with their first and second derivatives.

Let us construct such a smooth cubic spline. On each subinterval $\langle x_i, x_{i+1}\rangle$, $i = 0, 1, \ldots, n-1$

$$S(x) = \alpha_i + \beta_i(x - x_i) + \gamma_i(x - x_i)^2 + \delta_i(x - x_i)^3. \tag{2.4.2}$$

On the whole interval $\langle a, b\rangle = \langle x_0, x_n\rangle$, we have to determine $4n$ parameters α_i, β_i, γ_i, δ_i, $i = 0, \ldots, n-1$. The interpolation conditions require for $0 \le i \le n-1$

$$S_+(x_i) = \lim_{x \to x_i^+} S(x) = f(x_i), \quad S_-(x_{i+1}) = \lim_{x \to x_{i+1}^-} S(x) = f(x_{i+1}), \tag{2.4.3}$$

so we have $2n$ conditions. Since the first derivative S' and the second derivative S'' have to be continuous in the interior knots $\langle a, b\rangle$, we obtain

$$S'_-(x_i) = S'_+(x_i), \quad S''_-(x_i) = S''_+(x_i) \quad \text{pro} \quad i = 1, 2, \ldots, n-1. \tag{2.4.4}$$

These equations represent $2(n-1)$ conditions. Altogether we have $2n + 2(n-1) = 4n - 2$ equations for the determination of the parameters α_i, β_i, γ_i a δ_i. For the full evaluation of the all coefficients, two conditions are left. We say that the system has two degrees of freedom. These two conditions are chosen as co called end conditions, for example

- 1.type - the complete cubic spline:
$$S'(x_0) = f'(x_0), \quad S'(x_n) = f'(x_n);$$
- 2.type - the natural cubic spline:
$$S''(x_0) = S''(x_n) = 0;$$
- 3.type (no specific name):
$$S''(x_0) = f''(x_0), \quad S''(x_n) = f''(x_n);$$

- 4.type - the periodic cubic spline with the period $x_n - x_0$:

$$S'(x_0) = S'(x_n), \quad S''(x_0) = S''(x_n).$$

Each of these conditions guarantee the uniqueness of the cubic spline S.

Example 2.4.1 *By the cubic spline $S(x)$, we will interpolate the function $f(x) = \frac{1}{x}$ on the interval $\langle 0.5, 3\rangle$. The data of the problem:*

x	$f(x)$	$f'(x)$	$f''(x)$
0.5	2	-0.25	16
1	1	–	–
3	1/3	$-1/9$	2/27

We obtain

- *for end conditions of the first type:*

$$S(x) = \begin{cases} 5.88889 - 12.4444\,x + 11.1111\,x^2 - 3.55556\,x^3 & x \in \langle 0.5, 1\rangle \\ 2.41667 - 2.02778\,x + 0.694444\,x^2 - 0.0833333\,x^3 & x \in \langle 1, 3\rangle \end{cases}.$$

- *for end conditions of the second type:*

$$S(x) = \begin{cases} 3. - 1.66667\,x - 1.\,x^2 + 0.666667\,x^3 & x \in \langle 0.5, 1\rangle \\ 3.83333 - 4.16667\,x + 1.5\,x^2 - 0.166667\,x^3 & x \in \langle 1, 3\rangle \end{cases}.$$

- *for end conditions of the third type:*

$$S(x) = \begin{cases} 7. - 16.6049\,x + 15.8148\,x^2 - 5.20988\,x^3 & x \in \langle 0.5, 1\rangle \\ 1.81481 - 1.04938\,x + 0.259259\,x^2 - 0.0246914\,x^3 & x \in \langle 1, 3\rangle \end{cases}.$$

- *for end conditions of the fourth type:*

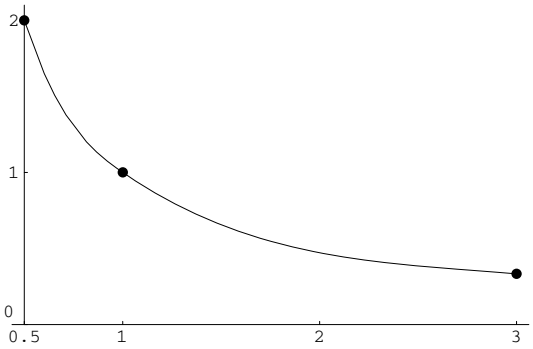
$$S(x) = \begin{cases} 2. + 2.33333\,x - 6.\,x^2 + 2.66667\,x^3 & x \in \langle 0.5, 1\rangle \\ 5.33333 - 7.66667\,x + 4.\,x^2 - 0.666667\,x^3 & x \in \langle 1, 3\rangle \end{cases}.$$

The resulting splines are depicted on the Fig. 2.4

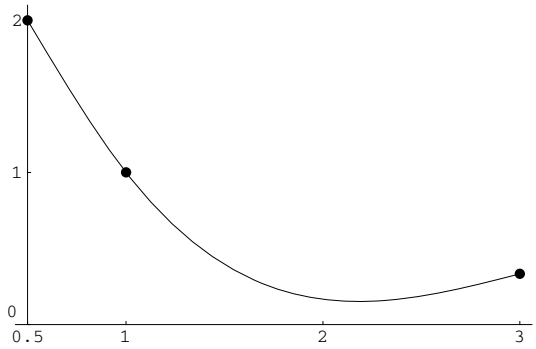
When we construct the periodic spline, it is natural to expect periodical data. We don't recommend to use this type of the spline if the interpolated function is not periodical, see Example 2.4.1 and Fig. 2.4.

The cubic splines are suitable namely for a graphical processing of data. They don't oscillate much and they represent curves, which pass "smoothly" through the given points.

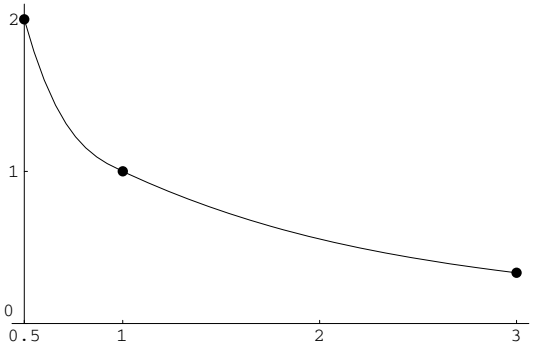
In the section 2.2, we stated that the interpolation polynomials need not converge to the interpolated function f, though we refine the division of the interval $\langle a, b\rangle$ more and more (see Example 2.2.2). The approximation by cubic splines is much better. Namely, let the approximated function be $f \in \mathbb{C}^4(\langle a, b\rangle)$, $a = x_0 < x_1 < \ldots < x_n = b$ is the division of the interval $\langle a, b\rangle$ to subintervals of the length $h_i = x_{i+1} - x_i$, $i = 0, 1, \ldots, n-1$, such that $\frac{\max h_i}{\min h_i} \leq K$. Let us denote $h = \max_{0 \leq i \leq n-1} h_i$.



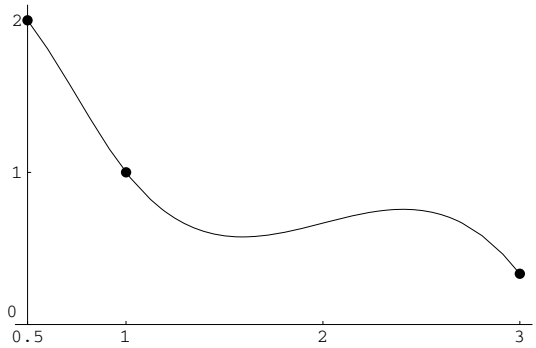
The complete cubic spline



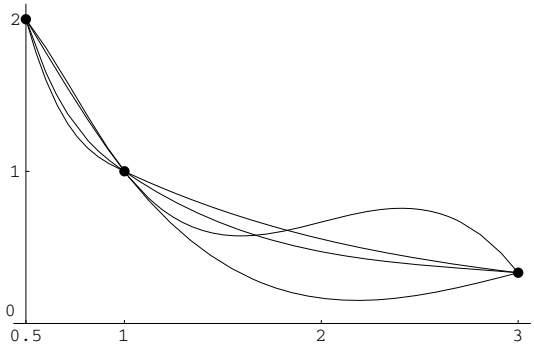
The natural cubic spline



The cubic spline of the 3rd type



The periodic cubic spline



The common graph of the previous cubic splines

Figure 2.4: Example 2.4.1

If $S(x)$ is the complete cubic spline (end conditions of the first type), which interpolates the function f on $\langle a, b\rangle$, then the following estimate is valid for all $x \in \langle a, b\rangle$ and $k = 0, 1, 2, 3$,

$$|f^{(k)}(x) - S^{(k)}(x)| \leq CKh^{4-k}, \tag{2.4.5}$$

where C is a constant independent on x and on the division of the interval $\langle a, b\rangle$. Consequently, for $h \to 0$ we obtain a good approximation of the function and its derivatives up to the third order on the whole interval $\langle a, b\rangle$.

A similar statement is valid also for the cubic spline with the end conditions of the third type. The natural cubic spline (end conditions of the second type) doesn't demand any information about derivatives of the interpolated function, but the error of the approximation near the ends of the interval $\langle a, b\rangle$ is not better then $\mathcal{O}(h^2)$ and, in general, it doesn't approximate the second derivative at all.

On Fig. 2.5, one can see a comparison of Runge's function and its approximation by cubic splines. Compare also with Fig. 2.2 and Fig. 2.3.

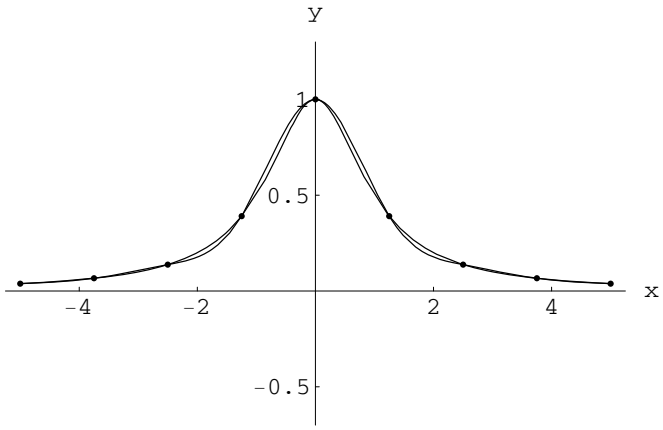


Figure 2.5: The approximation of Runge's function by cubic splines.

More theory about the approximation by splines can be found in [4].

2.5 Difference formulas

In calculus we often compute derivative of a given function analytically according to simple rules. This becomes laborious for more complicated functions and for derivatives of higher order. These rules cannot be used at all for differentiating functions given by a table of values, which is a typical situation when the function is a result of numerical integration. In this case we replace the given function $f(x)$ by its interpolation polynomial $\phi(x)$ and we consider the derivative of this polynomial as the approximation of the derivative of the given function.

Let us approximate the function $f(x)$ by the Lagrange interpolation polynomial (2.2.2). The first derivative in x is formally

$$L_n'(x) = \sum_{i=0}^{n} \frac{f(x_i)}{\omega_n'(x_i)} \left(\frac{\omega_n'(x)}{x - x_i} - \frac{\omega_n(x)}{(x - x_i)^2} \right). \tag{2.5.1}$$

This result is not good for practical computation. In most cases we want to evaluate the derivative at a given node of an equidistant grid. This is best done by difference formulas as shown in the next section.

2.5.1 Difference formulas for equidistant grid

We want to approximate the derivative of a function given by a table of values in an equidistant grid. To find the derivative at a point x_j with the distance between the nodes $h = x_{i+1} - x_i$, $i = 0, 1, \ldots, n-1$, it is easy to find that in (2.5.1) the second term in parenthesis vanishes and the first term after dividing by $\omega_n'(x_i)$ is in the form C_i/h, thus the difference formula for the first derivative at x_j can be written as

$$L_n'(x_j) = \frac{1}{h} \sum_{i=0}^{n} C_{ji} f_i \,, \tag{2.5.2}$$

where we denote $f_i = f(x_i)$.

To estimate the error of this approximation we must use some additional information about the function. If the function has $n+1$ continuous derivatives in the interval $[x_0, x_n]$ and $f^{(n+2)}(x)$ exists for all $x \in (x_0, x_n)$ then the error can be estimated by differentiating (2.2.3). The derivative is evaluated at x_j, $j = 0, \ldots, n$, considering $\omega(x_j) = 0$, $j = 0, \ldots, n$:

$$E'(x_j) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\,\omega_n'(x_j), \qquad j = 0, \ldots, n\,,$$

where ξ depends on x_j.

Difference formulas for the first derivative and for $n = 1, 2, 3, 4, 5, 6$ are listed in Table 2.1, the derivative is evaluated in the node whose number is underscored. The meaning of Table 2.1 is best illustrated by the following example.

For $n = 4$ (five nodes) the first derivative in the second node from the left x_1 is

$$f'(x_1) = f_1' = \frac{1}{12h}\,(-3f_0 - 10f_1 + 18f_2 - 6f_3 + f_4) - \frac{h^4}{20} f^{(5)}(\xi)$$

according to formula 11 in Table 2.1.

Comparing difference formulas in Table 2.1, we see that the simplest formulas are those for even n in middle nodes. Also their error coefficients are the lowest. Thus these formulas are used most often.

Similarly the formulas for the second derivative are prepared. It can be shown that

$$L_n''(x_j) = \frac{1}{h^2} \sum_{i=0}^{n} D_{ji} f_i\,. \tag{2.5.3}$$

Table 2.2 shows the coefficients along with the error estimates for the second derivative approximation using 3, 4 and 5 nodes in a similar way to Table 2.1. E.g.

$$f''(x_1) = f_1'' = \frac{1}{h^2}\,(f_0 - 2f_1 + f_2 + 0 \cdot f_3) - \frac{1}{12} h^2 f^{(4)}(\xi)$$

according to formula 5 in Table 2.2. Table 2.3 shows the formulas for the third and the fourth derivative using 4 and 5 nodes. Only the leading term in the error estimate is given i.e. the term with the lowest power of the step size h.

2.5.2 Method of unknown coefficients

When preparing difference formulas for non-equidistant grid, it is better to use the method of unknown coefficients instead of the Lagrange polynomial. The formula can be written as

$$f^{(k)}(\overline{x}) = \sum_{i=0}^{n} C_i f(x_i) + R(f)\,. \tag{2.5.4}$$

We choose the coefficients C_i so that $R(f) = 0$ for $f = 1; x; x^2; \ldots; x^n$. This gives a system of linear algebraic equations

$$\begin{array}{lllll}
C_0 & + C_1 & + \ldots + C_n & = 0 & \\
C_0 x_0 & + C_1 x_1 & + \ldots + C_n x_n & = 0 & \left(= \dfrac{d^k}{dx^k}(x) \mid_{x=\overline{x}} \right) \\
\vdots & & & & \\
C_0 x_0^{k-1} & & + \ldots + C_n x_n^{k-1} & = 0 & \\
C_0 x_0^{k} & & + \ldots + C_n x_n^{k} & = k! & \left(= \dfrac{d^k}{dx^k}(x^k) \mid_{x=\overline{x}} \right) \\
C_0 x_0^{k+1} & & + \ldots + C_n x_n^{k+1} & = (k+1)!\,\overline{x} & \\
\vdots & & & & \\
C_0 x_0^{n} & & + \ldots + C_n x_n^{n} & = n(n-1)\cdots(n-k+1)\overline{x}^{n-k}\,. &
\end{array} \tag{2.5.5}$$

Solving (2.5.5) gives the unknown coefficients C_i in formula (2.5.4).

As the formula is independent of a shift along x we can choose this shift so that one node (say, x_0) is zero. This simplifies the system of equations (2.5.5).

Example 2.5.1 *Let us choose the nodes $x_0 = 0$, $x_1 = h$, $x_2 = 2h$, i.e. $n = 2$. We want to derive formula 2 in Table 2.2 using the method of unknown coefficients. Using (2.5.4) we have ($\overline{x} = h$)*

$$f''(\overline{x}) = \sum_{i=0}^{2} C_i\ f(x_i)\ +\ R(f).$$

For $f = 1, x, x^2$ we get the equation

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & h & 2h \\ 0 & h^2 & 4h^2 \end{pmatrix} \quad \begin{pmatrix} C_0 \\ C_1 \\ C_2 \end{pmatrix} \quad = \quad \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix} \tag{2.5.6}$$

for unknown coefficients C_0, C_1, C_2 and this gives the coefficients of the formula $C_0 = \dfrac{1}{h^2}$, $C_1 = -\dfrac{2}{h^2}$, $C_2 = \dfrac{1}{h^2}$. As the right hand side of (2.5.6) does not depend on $\overline{x}$, the coefficients in formula 1 and 3 in Table 2.2 are the same as those in formula 2.

2.5.3 Richardson extrapolation

The error estimates of difference formulas using equidistant grid are in the form

$$R = Ch^n + \mathcal{O}(h^{n+1})\,. \tag{2.5.7}$$

The symbol $\mathcal{O}(h^p)$ is used to express how a given quantity goes to zero for $h \to 0_+$. More precisely, if $R(h) = \mathcal{O}(h^p)$, then

$$\lim_{h \to 0_+} \frac{R(h)}{h^p} = K \neq 0. \tag{2.5.8}$$

For small h we can write $R(h) \doteq Kh^p$ or $R(h) = Kh^p + \mathcal{O}(h^{p+1})$. When investigating the asymptotic behavior of the error R for $h \to 0_+$, then the term $\mathcal{O}(h^{n+1})$ in (2.5.7) goes to zero much faster and it can be neglected.

If we know the order n then after computing two results Q_1 and Q_2 with two different values h_1 and h_2 of the step h we can estimate the correct value Q with an error smaller than that of Q_1 or Q_2. Using the step size h_1 we find

$$Q_1 = Q + Ch_1^n \tag{2.5.9}$$

and with h_2 we find

$$Q_2 = Q + Ch_2^n \,. \tag{2.5.10}$$

We can consider (2.5.9) and (2.5.10) as a system of two equations for two unknowns C and Q, assuming $h_1 \neq h_2$. Solving this system gives the value for Q denoted as Q_{12}

$$Q_{12} = \frac{\left(\frac{h_1}{h_2}\right)^n Q_2 - Q_1}{\left(\frac{h_1}{h_2}\right)^n - 1} \,. \tag{2.5.11}$$

We often use $h_1/h_2 = 2$. Then

$$Q_{12} = \frac{4}{3}Q_2 - \frac{1}{3}Q_1 \,, \quad \text{for } n = 2 \,, \tag{2.5.12}$$

$$Q_{12} = \frac{16}{15}Q_2 - \frac{1}{15}Q_1 \,, \quad \text{for } n = 4 \,. \tag{2.5.13}$$

Here Q_{12} represents the estimate of the correct value of Q based on values Q_1 and Q_2 (its error is $\mathcal{O}(h^{n+1})$, that was neglected in (2.5.9) and (2.5.10)). The value Q_{12} can be used for the aposteriori error estimate: the error of Q_i is approximately $|Q_i - Q_{12}|$. This allows the adaptive step size control to achieve the desired error.

Table 2.1: Difference formulas for $h\,f'$ (equidistant grid)

Multiplier	Coefficients at x_0	x_1	x_2	x_3	x_4	x_5	x_6	Error		Formula #
1	$\underline{-1}$	1						$-\frac{1}{2}$	$hf''(\xi)$	1
	-1	$\underline{1}$						$\frac{1}{2}$		2
$\frac{1}{2}$	$\underline{-3}$	4	-1					$\frac{1}{3}$		3
	-1	$\underline{0}$	1					$-\frac{1}{6}$	$h^2f'''(\xi)$	4
	1	-4	$\underline{3}$					$\frac{1}{3}$		5
$\frac{1}{6}$	$\underline{-11}$	18	-9	2				$-\frac{1}{4}$		6
	-2	$\underline{-3}$	6	-1				$\frac{1}{12}$	$h^3f^{(4)}(\xi)$	7
	1	-6	$\underline{3}$	2				$-\frac{1}{12}$		8
	-2	9	-18	$\underline{11}$				$\frac{1}{4}$		9
$\frac{1}{12}$	$\underline{-25}$	48	-36	16	-3			$\frac{1}{5}$		10
	-3	$\underline{-10}$	18	-6	1			$-\frac{1}{20}$		11
	1	-8	$\underline{0}$	8	-1			$\frac{1}{30}$	$h^4f^{(5)}(\xi)$	12
	-1	6	-18	$\underline{10}$	3			$-\frac{1}{20}$		13
	3	-16	36	-48	$\underline{25}$			$\frac{1}{5}$		14
$\frac{1}{60}$	$\underline{-137}$	300	-300	200	-75	12		$-\frac{1}{6}$		15
	-12	$\underline{-65}$	120	-60	20	-3		$\frac{1}{30}$		16
	3	-30	$\underline{-20}$	60	-15	2		$-\frac{1}{60}$	$h^5f^{(6)}(\xi)$	17
	-2	15	-60	$\underline{20}$	30	-3		$\frac{1}{60}$		18
	3	-20	60	-120	$\underline{65}$	12		$-\frac{1}{30}$		19
	-12	75	-200	300	-300	$\underline{137}$		$\frac{1}{6}$		20
$\frac{1}{60}$	$\underline{-147}$	360	-450	400	-225	72	-10	$\frac{1}{7}$		21
	-10	$\underline{-77}$	150	-100	50	-15	2	$-\frac{1}{42}$		22
	2	-24	$\underline{-35}$	80	-30	8	-1	$\frac{1}{105}$		23
	-1	9	-45	$\underline{0}$	45	-9	1	$-\frac{1}{140}$	$h^6f^{(7)}(\xi)$	24
	1	-8	30	-80	$\underline{35}$	24	-2	$\frac{1}{105}$		25
	-2	15	-50	100	-150	$\underline{77}$	10	$-\frac{1}{42}$		26
	10	-72	225	-400	450	-360	$\underline{147}$	$\frac{1}{7}$		27

Table 2.2: Difference formulas for $h^2\, f''$ (equidistant grid)

Multiplier	Coefficients at x_0	x_1	x_2	x_3	x_4	Error		Formula #
1	$\underline{1}$	-2	1			-1	$hf'''(\xi)$	1
	1	$\underline{-2}$	1			$-\frac{1}{12}$	$h^2 f^{(4)}(\xi)$	2
	1	-2	$\underline{1}$			1	$hf'''(\xi)$	3
1	$\underline{2}$	-5	4	-1		$\frac{11}{12}$		4
	1	$\underline{-2}$	1	0		$-\frac{1}{12}$	$h^2 f^{(4)}(\xi)$	5
	0	1	$\underline{-2}$	1		$-\frac{1}{12}$		6
	-1	4	-5	$\underline{2}$		$\frac{11}{12}$		7
$\frac{1}{12}$	$\underline{35}$	-104	114	-56	11	$-\frac{5}{6}$		8
	11	$\underline{-20}$	6	4	-1	$\frac{1}{12}$	$h^3 f^{(5)}(\xi)$	9
	-1	4	6	$\underline{-20}$	11	$-\frac{1}{12}$		11
	11	-56	114	-104	$\underline{35}$	$\frac{5}{6}$		12
	-1	16	$\underline{-30}$	16	-1	$\frac{1}{180}$	$h^4 f^{(6)}(\xi)$	10

Table 2.3: Difference formulas for $h^3\, f'''$ and $h^4\, f^{(4)}$ (equidistant grid)

Multiplier	Coefficients at x_0	x_1	x_2	x_3	x_4	Error		Formula #
1	$\underline{-1}$	3	-3	1		$-\frac{3}{2}$		1
	-1	$\underline{3}$	-3	1		$-\frac{1}{2}$	$hf^{(4)}(\xi)$	2
	-1	3	$\underline{-3}$	1		$\frac{1}{2}$		3
	-1	3	-3	$\underline{1}$		$\frac{3}{2}$		4
$\frac{1}{2}$	$\underline{-5}$	18	-24	14	-3	$\frac{7}{4}$		5
	-3	$\underline{10}$	-12	6	-1	$\frac{1}{4}$		6
	-1	2	$\underline{0}$	-2	1	$-\frac{1}{4}$	$h^2 f^{(5)}(\xi)$	7
	1	-6	12	$\underline{-10}$	3	$\frac{1}{4}$		8
	3	-14	24	-18	$\underline{5}$	$\frac{7}{4}$		9
1	$\underline{1}$	-4	6	-4	1	-2		10
	1	$\underline{-4}$	6	-4	1	-1		11
	1	-4	6	$\underline{-4}$	1	1	$hf^{(5)}(\xi)$	12
	1	-4	6	-4	$\underline{1}$	2		13
	1	-4	$\underline{6}$	-4	1	$-\frac{1}{6}$	$h^2 f^{(6)}(\xi)$	14

2.6 Quadrature formulas

If we know the anti-derivative $F(x)$ to a given function $f(x)$ on some interval $[c,d]$ then we can compute the definite integral using the Newton's formula

$$\int_c^d f(x)dx = F(d) - F(c). \tag{2.6.1}$$

Usually it is not possible to find the anti-derivative in an analytic form (using elementary functions). Then we must evaluate the definite integral approximately. We approximate the given function $f(x)$ by the interpolation function $\phi(x)$, then

$$\int_c^d f(x)dx \doteq \int_c^d \phi(x)dx. \tag{2.6.2}$$

The interpolation polynomial $\phi(x)$ can be written in the form (2.1.4). Assume we can compute the integrals

$$\int_c^d \Phi_i(x)dx = c_i, \quad i = 0, 1, \ldots, n \tag{2.6.3}$$

analytically. The coefficients c_i do not depend on the choice of the function $f(x)$, they can be evaluated beforehand and then used for integrals (2.6.2) for any $f(x)$. Putting (2.6.3) into (2.6.2), the formula for numerical integration can be written as

$$\int_c^d f(x)dx \approx c_0 f(x_0) + c_1 f(x_1) + \cdots + c_n f(x_n)\,. \tag{2.6.4}$$

2.6.1 Equidistant nodes - Newton-Cotes formulas

Choosing polynomials for $\phi_i(x)$ we get the Lagrange interpolation polynomial $\phi(x)$. For practical computation it is convenient to choose the equidistant grid

$$x_i = a + ih, \quad i = 0, 1, 2, \ldots, n\,. \tag{2.6.5}$$

To evaluate (2.6.2), the relative position of the nodes (2.6.5) and the interval $[c,d]$ can be arbitrary. To get a small approximation error it turns out that two cases are convenient:

- closed formulas: $c = x_0$, $d = x_n$;
- open formulas: $c = x_0 - h$, $d = x_n + h$.

The coefficients $c_0, c_1, \ldots, c_n$ in (2.6.4) are given in Table 2.6.1 for various n for closed formulas and for $d - c = 1$. The reader is invited to derive the coefficients for open formulas using the method of unknown coefficients (see below).

For $n = 1$ the closed Newton-Cotes formula is called the trapezoidal rule:

$$\int_c^d f(x)dx = \frac{d-c}{2}(f(c) + f(d)) - \frac{(d-c)^3}{12} f''(\xi) \tag{2.6.6}$$

and for $n = 2$ it is called the Simpson's rule:

$$\int_c^d f(x)dx = \frac{d-c}{6}\left(f(c) + 4f\left(\frac{c+d}{2}\right) + f(d)\right) - \left(\frac{d-c}{2}\right)^5 \frac{f^{IV}(\xi)}{90}. \tag{2.6.7}$$

Table 2.4: Coefficients of closed Newton-Cotes formulas

n	$i=$	0	1	2	3	4	5	6
1	$2c_i=$	1	1					
2	$6c_i=$	1	4	1				
3	$8c_i=$	1	3	3	1			
4	$90c_i=$	7	32	12	32	7		
5	$288c_i=$	19	75	50	50	75	19	
6	$840c_i=$	41	216	27	272	27	216	41

If we divide the interval $[c,d]$ into m equal parts of length $h=(d-c)/m$ and denoting

$$x_0=c,\ x_1=c+h,\ \ldots\ ,x_m=d,$$

we can use the trapezoidal rule to each part $[x_i,x_{i+1}]$ and we can sum up the integrals

$$\begin{aligned}\int_c^d f(x)dx &= \frac{h}{2}\left(f(x_0)+2f(x_1)+2f(x_2)+\ldots+2f(x_{m-1})+f(x_m)\right)- \\ &\quad -\frac{h^3}{12}\left(f''(\xi_1)+f''(\xi_2)+\ldots+f''(\xi_m)\right), \end{aligned} \tag{2.6.8}$$

where $\xi_i\in(x_{i-1},x_i)$. The expression in the second bracket is equal to $mf''(\xi)$, where $\xi\in(c,d)$.

Thus the formula (2.6.8), which is called the generalized trapezoidal rule, can be written as

$$\begin{aligned}\int_c^d f(x)dx &= \frac{d-c}{2m}(f(c)+2f(c+h)+2f(c+2h)+\ldots \\ &\quad \ldots+2f(c+(m-1)h)+f(d))-\frac{(d-c)^3}{12m^2}f''(\xi)\end{aligned} \tag{2.6.9}$$

and the error of the generalized trapezoidal rule is $\mathcal{O}(h^2)$.

Similarly, dividing the interval $[c,d]$ into $2m$ equal parts, we get the generalized Simpson's rule

$$\begin{aligned}\int_c^d f(x)dx &= \frac{d-c}{6m}(f(c)+4f(c+h)+2f(c+2h)+4f(c+3h)+ \\ &\quad +2f(c+4h)+\ldots+4f(c+(2m-1)h)+f(d))- \\ &\quad -\left(\frac{d-c}{2}\right)^5\frac{f^{IV}(\xi)}{90m^4}.\end{aligned} \tag{2.6.10}$$

Here the error is $\mathcal{O}(h^4)$.

2.6.2 Method of unknown coefficients

To find the quadrature formula based on the integration of the Lagrange interpolation polynomial we can use the method of unknown coefficients. Consider the integral (2.6.4) where $x_0, x_1, x_2, \ldots, x_n$ is a chosen grid of nodes (not necessarily equidistant). Requiring (2.6.4) is exact for $f = 1, x, x^2, \ldots, x^n$, we get a system of linear algebraic equations for unknown coefficients c_i:

$$\begin{array}{ccccccccc} c_0 & + & c_1 & + & c_2 & +\cdots+ & c_n & = & \mu_0 \\ c_0 x_0 & + & c_1 x_1 & + & c_2 x_2 & +\cdots+ & c_n x_n & = & \mu_1 \\ \vdots & & & & & & & & \\ c_0 x_0^n & + & c_1 x_1^n & + & c_2 x_2^n & +\cdots+ & c_n x_n^n & = & \mu_n \end{array}, \tag{2.6.11}$$

where

$$\mu_j = \int_c^d x^j dx. \tag{2.6.12}$$

Similarly as for difference formulas we can shift the grid so that e.g $x_0 = 0$, to get a simpler system.

Example 2.6.1 *Using the method of unknown coefficients compute $\int_1^2 f(x)dx$, where the function f is given at the points $x_0 = 0$, $x_1 = 1$, $x_2 = 3$. According to (2.6.4) we have*

$$\int_1^2 f(x)\, dx \approx c_0\, f(0) + c_1\, f(1) + c_2\, f(3)\,.$$

Requiring this to be exact for $f = 1, x, x^2$, we get a system of equations for the unknown coefficients c_0, c_1, c_2

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 3 \\ 0 & 1 & 9 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{3}{2} \\ \frac{7}{3} \end{pmatrix}.$$

The solution is $c_0 = -\dfrac{4}{18}$, $c_1 = \dfrac{13}{12}$, $c_2 = \dfrac{5}{36}$, and thus

$$\int_1^2 f(x)\, dx \approx -\frac{4}{18}\, f(0) + \frac{13}{12}\, f(1) + \frac{5}{36}\, f(3)\,.$$

Note that the nodes $x_0, x_1, \ldots, x_n$ and the limits of the integral c, d are fixed. Under this assumption the condition (2.6.4) is exact for polynomials of order up to n. If, on the other hand, we leave the nodes unfixed, and we require (2.6.11) to hold exactly for as many polynomials $f(x) = x^k$ as possible we get the Gauss quadrature formulas. They use non-equidistant nodes and the order of approximation is higher. See more in [25] etc.

* * *

Further reading: [1], [4], [3], [25], [26].

Chapter 3

Numerical solution of nonlinear algebraic equations

Numerical solution of nonlinear algebraic equations is an important problem in numerical analysis. Nonlinear equations appear in various engineering applications, such as

- complicated chemical equilibrium,
- counter-current separation devices such as distillation and absorption columns,
- stationary simulation of a system of devices,
- replacement of parabolic or elliptic equations using finite differences,
- finding stationary states of dynamical models described by ordinary differential equations.

3.1 Equation with one unknown

3.1.1 General iteration method

To solve the equation

$$f(x) = 0 \tag{3.1.1}$$

several iteration methods have been developed. The main idea of these methods is as follows: Assume we know a sufficiently small interval containing a single root $x = x^*$ of the equation (3.1.1). We choose an initial approximation x_0 (close to the root x^*) in this interval and we build a series of points $x_1, x_2, \ldots, x_n, \ldots$ according to the recurrent rule

$$x_k = \phi_k(x_0, x_1, \ldots, x_{k-1}). \tag{3.1.2}$$

The recurrent rule (3.1.2) is constructed in such a way that (under certain assumptions) the series $\{x_n\}$ converges to x^*. Various choices of the function ϕ_k (depending on the function f) give various iteration methods.

The function $\phi(x)$ is often designed so that the wanted solution x^* is also a solution of an equation

$$x = \phi(x), \tag{3.1.3}$$

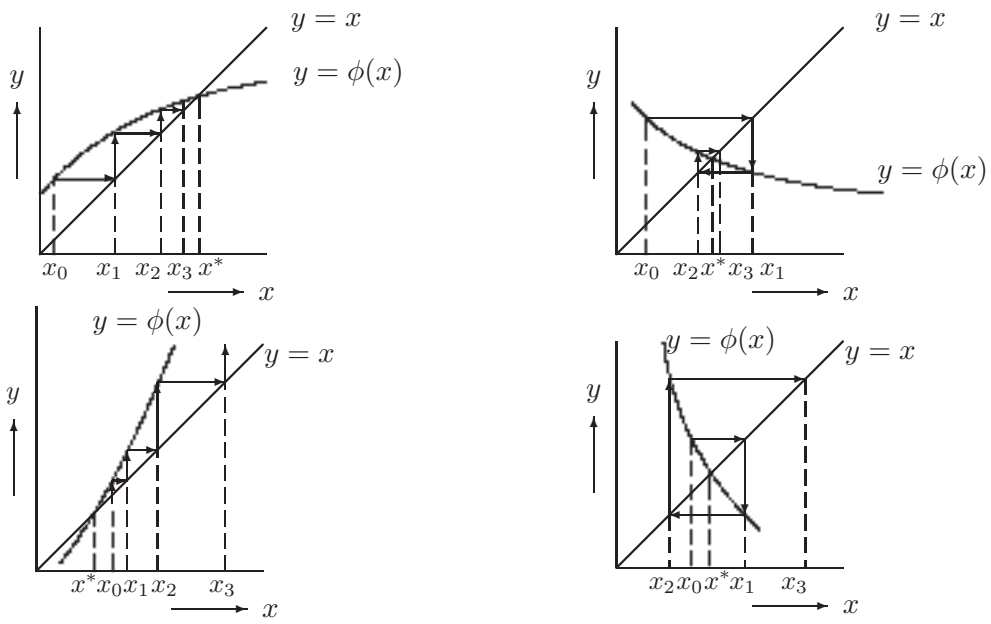


Figure 3.1: Course of iteration (3.1.4) for various values of ϕ'

where the series $\{x_k\}$ is constructed according to the rule

$$x_k = \phi(x_{k-1}), \qquad k = 1, 2, \dots \ . \tag{3.1.4}$$

Here, the function ϕ does not depend on the increasing index k, methods of this type are called stationary methods.

Often, the function $\phi(x)$ is differentiable. If

$$|\phi'(x^*)| \le K < 1 \tag{3.1.5}$$

and if ϕ' is continuous then $|\phi'(x)| < 1$ also in some neighborhood of the root x^* and the successive approximations (3.1.4) converge, provided x_0 is close to x^*. The smaller the constant K the faster the convergence.

Four different cases are shown in Fig. 3.1 where the derivative ϕ' has values in intervals $(0, 1)$, $(-1, 0)$, $(1, \infty)$, $(-\infty, -1)$ respectively. The series converges to the root in the first two cases only.

If we want a solution x^* with the accuracy ϵ, then we stop the iteration when

$$\frac{K}{1-K}|x_k - x_{k-1}| < \epsilon. \tag{3.1.6}$$

The order of iteration is a measure of the rate of convergence of (3.1.4). We say that the iteration (3.1.4) is of order m if

$$\phi'(x^*) = \phi''(x^*) = \cdots = \phi^{(m-1)}(x^*) = 0, \quad \phi^{(m)}(x^*) \neq 0. \tag{3.1.7}$$

If the function $\phi(x)$ has m continuous derivatives in a neighborhood of x^*, then using the Taylor expansion we get

$$\phi(x) - x^* = \phi(x) - \phi(x^*) = (x - x^*)\phi'(x^*) + \frac{1}{2!}(x - x^*)^2\phi''(x^*) + \cdots$$

$$\cdots + \frac{1}{(m-1)!}(x-x^*)^{m-1}\phi^{(m-1)}(x^*) + \frac{1}{m!}(x-x^*)^m\phi^{(m)}(\xi).$$

If the iteration is of order m we have

$$\phi(x) - x^* = \frac{1}{m!}(x-x^*)^m\phi^{(m)}(\xi),$$

or for $x = x_{k-1}$:

$$x_k - x^* = \frac{1}{m!}(x_{k-1}-x^*)^m\phi^{(m)}(\xi_k).$$

If $M_m = \max|\phi^{(m)}(x)|$ in a neighborhood of x^*, we get

$$|x_k - x^*| \le \frac{M_m}{m!}|x_{k-1}-x^*|^m. \tag{3.1.8}$$

If

$$|x_0 - x^*| < 1 \quad \text{and} \quad \frac{M_m}{m!}|x_0 - x^*| = \omega < 1,$$

then for $m > 1$ after simplification we get

$$|x_k - x^*| \le \omega^{\frac{m^k-1}{m-1}}, \tag{3.1.9}$$

which represents a fast convergence of x_k to x^*.

3.1.2 Bisection method and secant method

Before we explain simple iteration methods we discuss methods to locate the solution in a small interval. If the function $f(x)$ in (3.1.1) is continuous then it is sufficient to find two points x' and x'' such that $f(x')f(x'') < 0$, i.e. the function f has different signs at these two points. Then, due to the continuity of f, there is at least one root between x' and x''. If there is exactly one root and not more in a given interval, we call the interval a separation interval.

The simplest method to decrease an interval $[x', x'']$ containing the root is the bisection method. Let us denote x the center of the interval $[x', x'']$ i.e. $x = (x' + x'')/2$. Then either $f(x') \cdot f(x) < 0$ or $f(x) \cdot f(x'') < 0$. In the former case we decrease the interval to $[x', x]$, in the latter case the new interval will be $[x, x'']$. After n bisection steps the size of the interval is

$$|x' - x''| = 2^{-n}r, \tag{3.1.10}$$

where r is the size of the original interval. After 10 bisection steps the interval shrinks 1024 times.

This method converges slowly but it is reliable and it is good when we have not enough information about the precise location of the root.

If $x = x^*$ is a root of the equation $f(x) = 0$, i.e. $f(x^*) = 0$, and the function $\psi(x)$ is continuous in some neighborhood of x^*, then the equation

$$x = \phi(x) \tag{3.1.11}$$

where $\phi(x) = x - \psi(x)f(x)$ has also the root x^*. We can choose the function ψ in such a way so that the iteration $x_k = \phi(x_{k-1})$ for (3.1.11) converges. Let us start with one classical method of this type, the method of regula falsi (the secant method). Suppose f, f' and f'' are continuous and f' and f'' are non-vanishing in some neighborhood of x^*. Thus x^* is

a simple root of $f(x) = 0$. Let $f(x_0)f''(x_0) > 0$ for some x_0 from this neighborhood. Then we choose the function ψ to be

$$\psi(x) = \frac{x - x_0}{f(x) - f(x_0)}. \tag{3.1.12}$$

For the initial approximation we take some point x_1 from the given neighborhood satisfying $f(x_1)f(x_0) < 0$. Successive approximations are computed by

$$x_k = \frac{x_0 f(x_{k-1}) - x_{k-1} f(x_0)}{f(x_{k-1}) - f(x_0)}, \quad k = 2, 3, \dots \ . \tag{3.1.13}$$

Differentiating $\phi(x) = x - \psi(x)f(x)$ and using the Taylor expansion we get after simplification

$$\phi'(x^*) = \frac{1}{2}(x_0 - x^*)^2 \frac{f''(\xi)}{f(x_0)}. \tag{3.1.14}$$

If x_0 is sufficiently close to x^*, then

$$|\phi'(x)| \le K < 1$$

in some neighborhood of x^*. Choosing x_1 in this neighborhood the series (3.1.13) converges to x^*. As $\phi'(x^*) \neq 0$ according to (3.1.14), the method regula falsi is of order one.

3.1.3 Newton method

One of the most frequently used methods for solving nonlinear algebraic equations is the Newton method. We get this method when we put

$$\psi(x) = \frac{1}{f'(x)} \tag{3.1.15}$$

in (3.1.11). Thus we solve the iteration equation

$$x = x - \frac{f(x)}{f'(x)} = \phi(x), \tag{3.1.16}$$

which has the same root x^* as the equation (3.1.1). Let there is a unique root x^* in the interval $[a, b]$ and let the function f has continuous non-vanishing derivatives f' and f'' in this interval. Then

$$\phi'(x) = 1 - \frac{(f'(x))^2 - f(x)f''(x)}{(f'(x))^2},$$

and thus $\phi'(x^*) = 0$, as $f(x^*) = 0$. Because $|\phi'(x^*)| < 1$, there exists such a neighborhood of x^* that successive approximations

$$x_k = x_{k-1} - \frac{f(x_{k-1})}{f'(x_{k-1})}, \quad k = 1, 2, \dots \tag{3.1.17}$$

converge to x^*, if we choose x_0 in this neighborhood. The Newtond method is sometimes called the method of tangents due to its geometrical meaning, see Mathematics I. Under the above assumptions the convergence of $\{x_k\}$ to the solution x^* is monotone (i.e. from the side of x^*, where x_0 was chosen so that $f(x_0)f''(x_0) > 0$) - show it yourself by a graph.

As $\phi'(x^*) = 0$ and $\phi''(x^*)$ is non-vanishing (in a general case) the Newton method is an iteration method of order 2. Denoting

$$m = \min_{[a,b]} |f'(x)|, \qquad M = \max_{[a,b]} |f''(x)|,$$

and assuming $x_0 \in [a,b]$, $x^* \in [a,b]$ and assuming f' a f'' do not change the sign in the interval $[a,b]$ then using the Taylor expansion and simplification we get the estimate

$$|x_{k+1} - x^*| \leq \frac{M}{2m}|x_k - x^*|^2. \tag{3.1.18}$$

This estimate shows a fast convergence of the Newton method; for iterations close to x^* the number of valid decimal places to the right of the decimal point approximately doubles in each step.

The iteration (3.1.17) is sometimes (especially far away from the solution) replaced by the iteration

$$x_k = x_{k-1} - \alpha \frac{f(x_{k-1})}{f'(x_{k-1})}, \quad k = 1, 2, \dots, \tag{3.1.19}$$

where $0 < \alpha \leq 1$. This is to prevent divergence for bad initial approximation.

To evaluate the derivative f' we can use the analytic expression or the difference formula when the analytic differentiation is complicated or impossible. Then we approximate

$$f'(x_k) \doteq \frac{f(x_k + h) - f(x_k)}{h} \tag{3.1.20}$$

for a suitable small h. Then we evaluate the function f twice in each iteration.

3.2 Numerical solution of systems of nonlinear equations

A frequent problem in engineering is to find n unknowns $x_1, x_2, \dots, x_n$, satisfying nonlinear equations

$$\begin{aligned} f_1(x_1, x_2, \dots, x_n) &= 0 \\ f_2(x_1, x_2, \dots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, x_2, \dots, x_n) &= 0. \end{aligned} \tag{3.2.1}$$

For solving systems of nonlinear equations many iteration methods have been developed of the type

$$\boldsymbol{x}_{k+1} = \boldsymbol{\Phi}(\boldsymbol{x}_k), \quad k = 0, 1, \dots \quad , \tag{3.2.2}$$

where $\boldsymbol{x}_k$ is the k-th approximation of the vector of unknowns $\boldsymbol{x} = (x_1, x_2, \dots, x_n)^T$. Among the most frequently used methods are the Newton method and the generalized secant method.

3.2.1 Newton method

Let us denote $\boldsymbol{f} = (f_1, \dots, f_n)^T$ and the Jacobi matrix of the functions f_i

$$\mathbf{J}(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \dots & & \\ \vdots & & & \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \dots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}. \tag{3.2.3}$$

For the Newton method $\mathbf{\Phi}$ in (3.2.2) is chosen to be

$$\mathbf{\Phi}(\boldsymbol{x}) = \boldsymbol{x} - \lambda \mathbf{J}^{-1}(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x}), \tag{3.2.4}$$

i.e.

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \lambda_k \mathbf{J}^{-1}(\boldsymbol{x}_k)\boldsymbol{f}(\boldsymbol{x}_k). \tag{3.2.5}$$

After multiplying by the matrix $\mathbf{J}(\boldsymbol{x}_k)$ we have the form of the Newton method which is used in practical computation

$$\begin{aligned} \mathbf{J}(\boldsymbol{x}_k)\triangle\boldsymbol{x}_k &= -\boldsymbol{f}(\boldsymbol{x}_k) && (3.2.6)\\ \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \lambda_k \triangle\boldsymbol{x}_k. && (3.2.7) \end{aligned}$$

Here (3.2.6) is a system of n linear equations for n unknowns (the corrections) $\triangle\boldsymbol{x}_k$. This linear problem can be solved by method of linear algebra, see chapter 1.

The damping coefficient λ_k can be set to 1; then it is desirable to test, whether the residuum decreases, i.e.

$$\sum_{i=1}^{n} f_i^2(\boldsymbol{x}_{k+1}) < \sum_{i=1}^{n} f_i^2(\boldsymbol{x}_k).$$

If this condition fails, λ_k has to be decreased.

The Newton method for a system of equations is of order 2, similarly as for a single equation. If $J(x)$ is continuous in a neighborhood of $\boldsymbol{x}^*$ and if $J(\boldsymbol{x}^*)$ is non-singular, then the method converges, assuming the initial approximation $\boldsymbol{x}_0$ was chosen sufficiently close to $\boldsymbol{x}^*$. The following stop criterion is usually used: if $\|\Delta\boldsymbol{x}_k\| < \varepsilon$ then $\boldsymbol{x}_{k+1}$ approximates the solution $\boldsymbol{x}^*$ with an error better than ε.

Often a modified Newton method is used, where the Jacobi matrix is evaluated in $\boldsymbol{x}_0$ only, it is inverted, and then the iterations are computed according to

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \mathbf{J}^{-1}(\boldsymbol{x}_0)\boldsymbol{f}(\boldsymbol{x}_k), \quad k = 0, 1, \dots \quad . \tag{3.2.8}$$

We can combine the original and the modified Newton method to use the original method when far away from the solution and to use the modified method when close to the solution, where the modified method has almost the same rate of convergence.

Let us illustrate the Newton method in the following example.

Example 3.2.1 *Find the solution of the system of equations*

$$\begin{aligned} f_1(\boldsymbol{x}) &= 16x_1^4 + 16x_2^4 + x_3^4 - 16 = 0\\ f_2(\boldsymbol{x}) &= x_1^2 + x_2^2 + x_3^2 - 3 = 0\\ f_3(\boldsymbol{x}) &= x_1^3 - x_2 = 0. \end{aligned} \tag{3.2.9}$$

We take $\boldsymbol{x}_0 = (1, 1, 1)$ *for the initial approximation. Then*

$$\boldsymbol{f}(\boldsymbol{x}_0) = \begin{pmatrix} 17 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{J}(\boldsymbol{x}_0) = \begin{pmatrix} 64 & 64 & 4 \\ 2 & 2 & 2 \\ 3 & -1 & 0 \end{pmatrix},$$

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 - \mathbf{J}^{-1}(\boldsymbol{x}_0)\boldsymbol{f}(\boldsymbol{x}_0) = \left(\frac{223}{240}, \frac{63}{80}, \frac{79}{60}\right).$$

A few iterations are listed in Tab. 3.1. The fourth iteration is valid in 6 decimal digits.

Table 3.1: Newton method for the system 3.2.9

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$f_1(\boldsymbol{x}_k)$	$f_2(\boldsymbol{x}_k)$	$f_3(\boldsymbol{x}_k)$
0	1	1	1	17	0	0
1	0.929167	0.787500	1.283333	4.791917	0.130451	0.014697
2	0.887075	0.693176	1.320865	0.645310	0.012077	0.004864
3	0.878244	0.677195	1.330610	0.001845	0.000428	0.000207
4	0.877966	0.676757	1.330855	0.000015	0.000000	0.000000
5	0.877966	0.676757	1.330855	0.000000	0.000000	0.000000

Similarly as for the Newton method for a single equation, we can evaluate the Jacobi matrix using the difference formulas. Thus

$$\frac{\partial f_i(\boldsymbol{x})}{\partial x_j} \approx \frac{f_i(\boldsymbol{x}+h\mathbf{e}_j)-f_i(\boldsymbol{x})}{h} \tag{3.2.10}$$

or

$$\frac{\partial f_i(\boldsymbol{x})}{\partial x_j} \approx \frac{f_i(\boldsymbol{x}+h\mathbf{e}_j)-f_i(\boldsymbol{x}-h\mathbf{e}_j)}{2h}, \tag{3.2.11}$$

where $\mathbf{e}_j$ is a unit vector with its j-th coordinate equal to 1 and h is a suitable small number. When choosing the value for h we must consider that decreasing h increases the accuracy of the difference formulas (3.2.10), (3.2.11) (see chapter 2), but number of valid decimal digits in derivatives decreases (due to subtracting similar values).

Roughly speaking, for example, when working with 6 decimal digits, after taking $h = 0.01$, we cannot expect more than 4 valid decimal digits assuming that the number of valid digits does not decrease in evaluation of f_i.

For one iteration step of the difference version of the Newton method the left hand side vector in (3.2.6) must be evaluated $(n+1)$ times when using (3.2.10), or $(2n+1)$ times when using (3.2.11). The time of computation may increase by this significantly, especially for large n.

Higher order iteration methods can be derived but they are much more complicated and seldom used.

Chapter 4

Numerical solution of ordinary differential equations - initial value problem

Numerical integration of ordinary differential equations is a frequent task of numerical analysis in chemical engineering problems. Numerical integration of differential equations is used if the equations are nonlinear or if we have a large system of linear equations with constant coefficients, where the analytical solution can be found, but it is in the form of long and complicated expressions containing exponential functions. Numerical integration of linear equations with non-constant coefficients is also more efficient than the analytical solution; in the case of internal diffusion in porous catalyst with a chemical reaction of the 1. order the analytical solution contains Bessel functions. The solution can be evaluated more conveniently when we use numerical integration of the original equations than to evaluate Bessel functions.

Many problems in chemical engineering involve solution of ordinary differential equations. These are dynamical problems in isotropic media and stationary problems with a single space variable. The former include batch reactor, differential distillation, non-stationary regime of a distillation column etc. The latter include tubular reactors and heat exchangers.

In some chemical engineering problems dynamic balance must be solved with accumulation that terms differ by several orders of magnitude. This corresponds to physical processes where some dependent variables relax very fast while others approach the stationary state slowly. This type of problems is called "stiff" and it is difficult to solve. Stiff problems often arise in reactor engineering (radical reactions, complex reactions with some of them very fast) and in system engineering (dynamic regime of a distillation column with a mixture containing one very volatile component or one component with very low concentration).

Problems in dynamics of counter-current separation devices or systems of interacting devices lead to systems of hundreds of ordinary differential equations. Solution of such problems often requires special algorithms.

We shall not discuss differential-algebraic equations (DAE) that can be expressed in the form

$$\boldsymbol{F}(\boldsymbol{y}', \boldsymbol{y}) = \boldsymbol{0}\,,$$

that cannot be solved in $\boldsymbol{y}'$. These equations appear in several chemical engineering problems and they are difficult to solve. The reader is invited to check the specialized literature

[2], [10], [11].

4.1 Euler method and the method of Taylor expansion

Consider a single differential equation

$$y' = f(x, y) \tag{4.1.1}$$

with the initial condition

$$y(a) = c. \tag{4.1.2}$$

We want to find the solution $y(x)$ at discrete points (nodes) $a = x_0 < x_1 < x_2 < \ldots < x_N = b$ i.e. we want to find numbers $y_0 = c$, $y_1, \ldots, y_N$, approximating the values $y(x_0), y(x_1), \ldots, y(x_N)$ of the exact solution at the nodes $x_0, \ldots, x_N$. We often consider the equidistant grid, i.e. $x_{n+1} - x_n = h$; $n = 0, 1, \ldots, N-1$. The number $h = \frac{x_N - x_0}{N}$ is called the step size. The approximation y_n of the exact solution $y(x_n)$ at x_n is computed from the values of the approximate solution evaluated at previous nodes. If y_{n+1} is evaluated from k values $y_n, y_{n-1}, \ldots, y_{n+1-k}$, the method is called a k-step method. If we replace the derivative y' at $x = x_n$ by the difference formula using two points x_n and x_{n+1} we get the Euler method

$$y_{n+1} = y_n + hf(x_n, y_n)\,, \qquad n = 0, 1, 2, \ldots, N-1\,, \tag{4.1.3}$$

with

$$y_0 = c\,.$$

The computation using the Euler method is very easy. We can illustrate it by the following simple example. Solve the equation $y' = y\,;\, y(0) = 1$ using the Euler method. The recurrent relation (4.1.3) is

$$y_{n+1} = (1+h)y_n\,, \qquad y_0 = 1\,,$$

i.e.

$$y_n = (1+h)^n\,.$$

For a given $x = nh$ we have $n = \dfrac{x}{h}$, and thus

$$y_n = (1+h)^{\frac{x}{h}} = [(1+h)^{\frac{1}{h}}]^x.$$

For $h \to 0_+$ the approximate solution y_n converges to the exact solution e^x.

Denoting $y(x)$ the exact solution, the difference

$$e_n = y_n - y(x_n) \tag{4.1.4}$$

is called the global approximation error or the global discretization error and y_n is called the theoretical approximation of the solution. Another type of error comes from the fact that we cannot compute the value y_n exactly (on infinite number of decimal places). Denoting $\tilde{y}_n$ the values that are computed instead of y_n, the difference

$$r_n = \tilde{y}_n - y_n \tag{4.1.5}$$

is called the round-off error. Then the total error is given by the triangle inequality

$$|\tilde{y}_n - y(x_n)| \le |e_n| + |r_n|\,. \tag{4.1.6}$$

The values $\tilde{y}_n$ are called the numerical approximation. In the following we deal with the theoretical approximation only, though the round-off errors are also important, because they may be larger than the approximation error in some cases. We also skip the derivation of the error estimates because it is out of the scope of this text.

If the function $f(x, y)$ satisfies the Lipschitz condition in y, i.e. if there is a constant $L > 0$ such that

$$|f(x, y) - f(x, y^*)| \leq L|y - y^*| \tag{4.1.7}$$

is true for $x \in [a, b]$ and any y and y^* and if the exact solution $y(x)$ of the equation (4.1.1) is twice differentiable in the interval $[a, b]$, and denoting

$$N(x) = \frac{1}{2} \max_{t \in [a,x]} |y''(t)| \,, \tag{4.1.8}$$

then the global approximation error of the Euler method can be estimated by

$$|e_n| \leq hN(x_n)E_L(x_n - a) \,. \tag{4.1.9}$$

Here

$$E_L(x) = \begin{cases} \dfrac{\mathrm{e}^{Lx} - 1}{L} & \text{if } L > 0 \\ x & \text{if } L = 0 \end{cases} \tag{4.1.10}$$

.

Assuming the function f has the first partial derivatives in $\Omega = [a, b] \times (-\infty, \infty)$ continuous then we can estimate $N(x)$ by

$$2N(x) \leq N = \max_{(x,y) \in \Omega} |f_x(x, y) + f_y(x, y)f(x, y)| \,, \tag{4.1.11}$$

where the index x and y denotes the partial derivative with respect to x and y, respectively

The estimates (4.1.9) are usually very pessimistic, which can be illustrated by the following example:

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

The exact solution is $y(x) = \mathrm{e}^x$. Equation (4.1.7) gives $L = 1$. The estimate $N(x)$ can be done from the exact solution, i.e.

$$2N(x) = \mathrm{e}^x \,.$$

According to (4.1.9) we have

$$|e_n| \leq \frac{1}{2} h \mathrm{e}^{x_n} (\mathrm{e}^{x_n} - 1) \,. \tag{4.1.12}$$

Table 4.1 compares this theoretical estimate with the real global approximation error for $h = 2^{-6}$.

Table 4.1: Global approximation error e_n and its theoretical estimate (4.1.12), $h = 2^{-6}$

x_n	1	2	3	4	5
y_n	2.69735	7.27567	19.62499	52.93537	142.7850
e_n	-0.02093	-0.11339	-0.46055	-1.66278	-5.6282
estimate (4.1.12)	0.03649	0.36882	2.99487	22.86218	170.9223

The estimate (4.1.9) shows that the error of the Euler method for a given x is proportional to the first power of the step size h, i.e. $\mathcal{O}(h)$ (see 2.5.8). We say the Euler method

is of the first order. Thus the Richardson extrapolation can be used for an a posteriori error estimate (see (2.5.11)).

Fig. 4.1 illustrates the behaviour of round-off error independence on h. The global approximation error is proportional to h while the round-off error is proportional to $1/h$ (the smaller the h the greater the number of arithmetic operations). As a result there is a certain "optimal" step size h_{opt} giving the least total error.

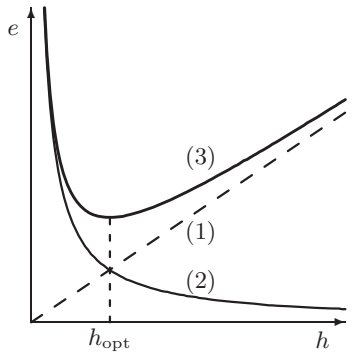


Figure 4.1: Global approximation error e_n (1), round-off error (2) and the total error (3) for the Euler method.

We do not want to use h_{opt} as the step size, because then the round-off error is of the same size as the approximation error and the Richardson extrapolation cannot be used for the estimate of the total approximation error. The only way how to estimate the round-off error is to repeat the computation with different precision (different number of digits used by the computer).

Advanced algorithms adjust the step size h automatically according to the local approximation error to get the final approximation with the required accuracy with a small number of operations (see 4.2).

For special cases the method of Taylor expansion can be used. If the function f in (4.1.1) has enough derivatives then we can write

$$\begin{aligned} y'' &= \frac{d\,f}{d\,x}(x, y(x)) = f_x(x, y) + f_y(x, y)y' = \\ &= f_x(x, y) + f_y(x, y)f(x, y)\,, \end{aligned} \tag{4.1.13}$$

where the index x or y denotes the partial derivative with respect to x or y resp. The third derivative is

$$y''' = f_{xx} + 2ff_{xy} + f_{yy}f^2 + f_x f_y + f f_y^2\,, \tag{4.1.14}$$

etc. The change in $y(x)$ can be found by the Taylor expansion

$$\begin{aligned} y(x_n + h) \doteq y_{n+1} = \\ = y_n + hy'(x_n) + \frac{h^2}{2}y''(x_n) + \ldots + \frac{h^p}{p!}y^{(p)}(x_n) + \mathcal{O}(h^{p+1})\,. \end{aligned} \tag{4.1.15}$$

The method (4.1.15) is called the method of Taylor expansion of order p. Its global error is of order p, i.e. $\mathcal{O}(h^p)$.

Example 4.1.1 *Use the method of Taylor expansion of the third order to solve the initial value problem*

$$y' = \frac{4}{x^2} - y^2 - \frac{y}{x}\,, \qquad y(1) = 0\,. \tag{4.1.16}$$

Table 4.2: Solution of 4.1.16 using Taylor expansion of order 3.

	x	1	1.2	1.4	1.6	1.8	2
$h = 0.2$		0	0.576000	0.835950	0.920226	0.920287	0.884745
$h = 0.1$	$y(x)$	0	0.581645	0.838338	0.919251	0.918141	0.882631
$h = 0.05$		0	0.582110	0.838443	0.919062	0.917872	0.882386
Richardson extrapolation (see 2.5.11) at $x = 2$: $p = 3\,, \quad h_1 = 0.1\,, \quad h_2 = 0.05\,,$ $y_1(2) = 0.882631\,, \quad y_2(2) = 0.882386 \quad \Rightarrow \quad y_{12}(2) = 0.882351$							
Exact solution: $y(x) = \dfrac{2(x^4-1)}{x(x^4+1)}\,, \qquad y(2) = 0.882353$							

Solution:
According to (4.1.15) for $n = 0, 1, 2, \ldots$ we have

$$y(x_n + h) \doteq y_{n+1} = y_n + hy'(x_n) + \frac{h^2}{2}y''(x_n) + \frac{h^3}{3!}y'''(x_n)\,.$$

Here

$$\begin{aligned}
x_0 = 1\,, \qquad & y_0 = 0\,, \\
y'(x_n) &= \frac{4}{x_n^2} - y_n^2 - \frac{y_n}{x_n}\,, \\
y''(x_n) &= -\frac{8}{x_n^3} - 2y_n y'(x_n) - \frac{y'(x_n)x_n - y_n}{x_n^2} = -\frac{12}{x_n^3} - \frac{6y_n}{x_n^2} + \frac{3y_n^2}{x_n} + 2y_n^3\,, \\
y'''(x_n) &= \frac{24}{x_n^4} - 2(y'(x_n))^2 - 2y_n y''(x_n) - \frac{y''(x_n)x_n^2 - 2(y'(x_n)x_n - y_n)}{x_n^3} = \\
&= \frac{12}{x_n^4} - \frac{42y_n}{x_n^3} + \frac{21y_n^2}{x_n^2} - \frac{12y_n^3}{x_n} - 6y_n^4\,.
\end{aligned}$$

Table 4.2 shows the computed values of the solution at the point $x_N = 2$ for various N (and thus for various $h = 1/N$).

It is obvious that this method is not suitable generally, because analytical differentiation may be very laborious for higher orders. This method can be used even for systems of differential equations, but the complexity of the derivation increases. Richardson extrapolation can be used as well as illustrated in Table 4.2.

4.2 Runge-Kutta methods

The analytical differentiation needed for the Taylor expansion as shown in the previous section is a principal obstacle for most practical problems. We show a method with similar properties (order of approximation) as the Taylor expansion method, but without the need of analytical differentiation. Let us write the increment in the form

$$y_{n+1} = y_n + h\Phi(x_n, y_n; h) \tag{4.2.1}$$

where $y_n \sim y(x_n)$. For the Euler method we had $\Phi(x,y;h) = f(x,y)$. Assume the increment function Φ in the form

$$\Phi(x,y;h) = a_1 f(x,y) + a_2 f\Big(x + p_1 h, y + p_2 h f(x,y)\Big) \tag{4.2.2}$$

where the constants a_1, a_2, p_1 and p_2 are to be found so that the method approximates the solution as good as possible. Put Φ from (4.2.2) into (4.2.1) and expand in powers of h (with $x = x_n$, $y = y_n$) :

$$y_{n+1} = y_n + h\Big\{(a_1 + a_2) f(x,y) + h a_2 \Big(p_1 f_x(x,y) + p_2 f_y(x,y) f(x,y)\Big) + \mathcal{O}(h^2)\Big\} \ . \tag{4.2.3}$$

We want the expansion (4.2.3) to agree with the Taylor expansion

$$y(x_n + h) = y(x_n) + h f(x,y) + \frac{1}{2} h^2 \Big(f_x(x,y) + f_y(x,y) f(x,y)\Big) + \mathcal{O}(h^3) \tag{4.2.4}$$

where y' was replaced by f and y'' was replaced by (4.1.13). Comparing the terms linear in h in (4.2.3) and (4.2.4) we get

$$a_1 + a_2 = 1. \tag{4.2.5}$$

The agreement of the terms quadratic in h (for any $f(x,y)$) requires

$$a_1 p_1 = \frac{1}{2} \quad , \quad a_2 p_2 = \frac{1}{2} \ . \tag{4.2.6}$$

It can be shown that the agreement of cubic terms in h cannot be achieved for general $f(x,y)$. We have three equations (4.2.5), (4.2.6) for four unknown parameters a_1, a_2, p_1, p_2. We can choose one of them, say $a_2 = \alpha$, then

$$a_1 = 1 - \alpha, \quad a_2 = \alpha, \quad p_1 = p_2 = \frac{1}{2\alpha} \tag{4.2.7}$$

where $\alpha \neq 0$ is a free parameter. Then the equation (4.2.1) using (4.2.2) has the form

$$y_{n+1} = y_n + (1-\alpha) h f(x_n, y_n) + \alpha h f\Big(x_n + \frac{h}{2\alpha}, y_n + \frac{h}{2\alpha} f(x_n, y_n)\Big) + \mathcal{O}(h^3) \ . \tag{4.2.8}$$

The result (4.2.8) can be conveniently written in successive equations

$$\begin{aligned} k_1 &= h f(x_n, y_n) \\ k_2 &= h f(x_n + \tfrac{h}{2\alpha}, y_n + \tfrac{1}{2\alpha} k_1) \\ y_{n+1} &= y_n + (1-\alpha) k_1 + \alpha k_2 \ . \end{aligned}$$

The cases $\alpha = \frac{1}{2}$ and $\alpha = 1$ are well known and they are called improved Euler method or Heun method :

$$\begin{aligned} k_1 &= h f(x_n, y_n) \\ k_2 &= h f(x_n + h, y_n + k_1) \\ y_{n+1} &= y_n + \tfrac{1}{2}(k_1 + k_2) \end{aligned} \tag{4.2.9}$$

and modified Euler method

$$\begin{aligned} k_1 &= h f(x_n, y_n) \\ k_2 &= h f(x_n + \tfrac{h}{2}, y_n + \tfrac{1}{2} k_1) \\ y_{n+1} &= y_n + k_2 \ . \end{aligned} \tag{4.2.10}$$

In some texts (4.2.9) is called modified Euler method. Both of these methods have the local error $\mathcal{O}(h^3)$, and the global error $\mathcal{O}(h^2)$. They belong to the family of Runge-Kutta methods as the simplest examples of them. More complicated and more accurate methods can be derived by a similar approach. We mention some representatives of them of order 3, 4, and 5. A general Runge-Kutta method can be written in successive equations (with $x = x_n$, $y = y_n$):

$$\begin{aligned} k_1 &= hf(x, y) \\ k_2 &= hf(x + \alpha_1 h, y + \beta_{11} k_1) \\ k_3 &= hf(x + \alpha_2 h, y + \beta_{21} k_1 + \beta_{22} k_2) \\ &\vdots \\ k_{j+1} &= hf(x + \alpha_j h, y + \beta_{j1} k_1 + \beta_{j2} k_2 + \cdots + \beta_{jj} k_j) \\ y_{n+1} &= y_n + \gamma_1 k_1 + \gamma_2 k_2 + \cdots + \gamma_{j+1} k_{j+1} \, . \end{aligned} \tag{4.2.11}$$

The method (4.2.11) can be written in the form of Table 4.3. This table also lists some Runge-Kutta methods and their order (global error).

If we want to get the order m with the Runge-Kutta method then for $m = 2, 3, 4$ we need $2, 3, 4$ evaluations of the right hand side of the differential equation. For $m = 5$ we need at least 6 evaluations and for $m > 4$ we need more than m evaluations. Thus the methods of order greater than 4 are seldom used, because their advantages become important only when very high accuracy is needed.

Sometimes the solution has a different character for different values of the independent variable x, and a different step size h should be used to get the desired accuracy. If we choose the step size to be the minimum of all the required step sizes, the accuracy is achieved, but in some parts we integrate unnecessarily accurate. This is not an effective approach. Single step methods (as Runge-Kutta e.g.) allow adaptive adjustment of the integration step size according to the character of the solution. A whole class of methods have been developed where the error in each step is estimated from the computed k_i, where the number of these k_i must be more than the minimal number of them. The first method of this kind was developed by Merson, others were found e.g. by Fehlberg. The Merson method is of order 4 and it uses 5 evaluations of the right hand side $f(x, y)$. It can be written as follows:

$$\begin{aligned} k_1 &= hf(x_0, y_0) & y_1 &= y_0 + \frac{k_1}{3} \\ k_2 &= hf(x_0 + \frac{h}{3}, y_1) & y_2 &= y_0 + \frac{k_1 + k_2}{6} \\ k_3 &= hf(x_0 + \frac{h}{3}, y_2) & y_3 &= y_0 + 0.125k_1 + 0.375k_3 \\ k_4 &= hf(x_0 + 0.5h, y_3) & y_4 &= y_0 + 0.5k_1 - 1.5k_3 + 2k_4 \\ k_5 &= hf(x_0 + h, y_4) & y_5 &= y_0 + \frac{k_1 + 4k_4 + k_5}{6} \, . \end{aligned} \tag{4.2.12}$$

For small h assuming $f(x, y)$ approximated by

$$f(x, y) = Ax + By + C \tag{4.2.13}$$

Merson derived that the error or y_4 is $\frac{-h^5 y^{(5)}}{120}$ and the error of y_5 is $\frac{-h^5 y^{(5)}}{720}$. Then we can estimate the error of y_5 by

$$E = \frac{1}{5}(y_4 - y_5) \, . \tag{4.2.14}$$

Table 4.3: Overview of Runge-Kutta methods

Scheme of Runge-Kutta methods

α_1	β_{11}				
α_2	β_{21}	β_{22}			
α_3	β_{31}	β_{32}	β_{33}		
$\vdots$					
α_j	β_{j1}	β_{j2}	$\dots$	β_{jj}	
	γ_1	γ_2	$\dots$	γ_j	γ_{j+1}

Euler

improved (4.2.8)			$\mathcal{O}(h^2)$	modified (4.2.9)			$\mathcal{O}(h^2)$
1	1			$\frac{1}{2}$	$\frac{1}{2}$		
	$\frac{1}{2}$	$\frac{1}{2}$			0	1	

Heun				$\mathcal{O}(h^3)$	Kutta				$\mathcal{O}(h^3)$
$\frac{1}{3}$	$\frac{1}{3}$				$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{2}{3}$	0	$\frac{2}{3}$			1	-1	2		
	$\frac{1}{4}$	0	$\frac{3}{4}$			$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$	

Runge-Kutta order 4

standard					$\mathcal{O}(h^4)$	three eighth				$\mathcal{O}(h^4)$
$\frac{1}{2}$	$\frac{1}{2}$					$\frac{1}{3}$	$\frac{1}{3}$			
$\frac{1}{2}$	0	$\frac{1}{2}$				$\frac{2}{3}$	$-\frac{1}{3}$	1		
1	0	0	1			1	1	-1	1	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$			$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$

Butcher order 5 $\mathcal{O}(h^5)$

$\frac{1}{4}$	$\frac{1}{4}$					
$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{8}$				
$\frac{1}{2}$	0	$-\frac{1}{2}$	1			
$\frac{3}{4}$	$\frac{3}{16}$	0	0	$\frac{9}{16}$		
1	$-\frac{3}{7}$	$\frac{2}{7}$	$\frac{12}{7}$	$-\frac{12}{7}$	$\frac{8}{7}$	
	$\frac{7}{90}$	0	$\frac{32}{90}$	$\frac{12}{90}$	$\frac{32}{90}$	$\frac{7}{90}$

If this estimate E is less than the desired error ε then the current step size is suitable. If not, we decrease the step size (by taking one half of it) and we recompute the last step. If $|E| < \frac{\varepsilon}{32}$ we can increase the step size (by taking its double). Instead of taking one half or the double of the step size, we can predict the optimal step size by

$$h_{new} = 0.8\, h_{old} \left(\frac{\varepsilon}{|E|}\right)^{0.2}. \tag{4.2.15}$$

The factor 0.8 is used to avoid the case when after prolongation we have to shorten the step size.

Each Runge-Kutta method can be used not just for a single differential equation but also for a system of differential equations of the first order. Then y, f, k_i become vectors. They can be used for equations of a higher order as well. Such a system can be converted into a system of the first order as illustrated by the following example. The equation

$$y'' = f(x, y, y')$$

is equivalent to the system

$$y' = z \qquad z' = f(x, y, z)\,.$$

There are special Runge-Kutta methods for equations of the 2. order. Their advantages are weak so they are seldom used.

4.3 Multi step methods

When using single-step methods as described in the previous section, we do not utilize the course of the solution found before. After each step we forget all the information. This is not effective. Multi step methods have been designed to utilize a few last points of the solution.

The solution is computed at an equidistant grid of points with the step size h. We denote $x_i = x_0 + ih\,, y_i \approx y(x_i)\,, f_i = f(x_i, y_i)\,$. A general linear multi-step method can be written as

$$\alpha_k y_{n+k} + \alpha_{k-1} y_{n+k-1} + \cdots + \alpha_0 y_n = h\Big(\beta_k f_{n+k} + \beta_{k-1} f_{n+k-1} + \cdots + \beta_0 f_n\Big) \tag{4.3.1}$$

assuming $\alpha_k \neq 0\,,\ \alpha_0^2 + \beta_0^2 > 0\,$. This is called a k-step method. Let us denote the polynomial

$$\varrho(\xi) = \alpha_k \xi^k + \cdots + \alpha_1 \xi + \alpha_0\,. \tag{4.3.2}$$

A necessary condition for the convergence (i.e. for $h \to 0_+$ we approach the exact solution) of the linear multi-step method (4.3.1) is: all the roots of the polynomial $\varrho(\xi)$ must be in absolute value less than 1, or equal to 1 but then they must be of multiplicity 1. This is called the stability condition of the method. Methods that fail this condition are useless.

Let us define the adjoint differential operator

$$\begin{aligned} L[y(x); h] \;=\; & \alpha_k y(x + kh) + \alpha_{k-1} y(x + (k-1)h) + \cdots + \alpha_0 y(x) - \\ & - h\Big(\beta_k y'(x + kh) + \beta_{k-1} y'(x + (k-1)h) + \cdots + \beta_0 y'(x)\Big)\,. \end{aligned} \tag{4.3.3}$$

Expanding $y(x + mh)$ and $y'(x + mh)$ by the Taylor polynomial around x we get

$$\begin{aligned} y(x + mh) &= y(x) + mhy'(x) + \frac{1}{2}m^2h^2y''(x) + \cdots + \frac{1}{i!}m^i h^i y^{(i)}(x) + \cdots \\ hy'(x + mh) &= hy'(x) + mh^2y''(x) + \frac{1}{2}m^2h^3y'''(x) + \cdots + \frac{1}{i!}m^i h^{i+1} y^{(i+1)}(x) + \cdots \end{aligned}$$

Put these expansions into (4.3.3) and we have

$$L[y(x);h] = C_0 y(x) + C_1 h y'(x) + \cdots + C_q h^q y^{(q)}(x) + \cdots \tag{4.3.4}$$

where the coefficients C_q satisfy:

$$\begin{array}{rcl} C_1 & = & \alpha_0 + \alpha_1 + \cdots + \alpha_k \\ C_1 & = & \alpha_1 + 2\alpha_2 + \cdots + k\ \alpha_k - (\beta_0 + \beta_1 + \cdots + \beta_k) \\ \vdots & & \\ C_q & = & \frac{1}{q!}(\alpha_1 + 2^q\alpha_2 + \cdots + k^q\alpha_k) - \frac{1}{(q-1)!}(\beta_1 + 2^{q-1}\beta_2 + \cdots k^{q-1}\beta_k). \end{array} \tag{4.3.5}$$

We say that the differential operator L is of order p if

$$C_0 = C_1 = \cdots = C_p = 0, \quad C_{p+1} \neq 0. \tag{4.3.6}$$

Thus

$$L[y(x);h] = \mathcal{O}(h^{p+1}) \tag{4.3.7}$$

and the local error is $\mathcal{O}(h^{p+1})$, the global error is $\mathcal{O}(h^p)$. The process of finding the coefficients α and β so that (4.3.6) is satisfied is called the method of unknown coefficients. A method of order p approximates exactly a solution which is a polynomial of order not more than p. A necessary condition for getting the exact solution as $h \to 0_+$ is that the order of the adjoint differential operator is at least 1, i.e. $C_0 = 0$ and $C_1 = 0$. For k odd, the order of a stable operator cannot be greater than $k+1$. For k even, the order of a stable operator cannot be greater than $k+2$. To get $p = k+2$ all the roots of $\varrho(\xi)$ must be on the unit circle (in absolute value equal to 1) and the formula is designed so that as many as possible of the constants $C_0, C_1, C_2, \ldots$ vanish.

4.4 Adams formulas

We present some special multi-step methods. Adams formulas have only two nonzero coefficients α_i in (4.3.1), namely the coefficients with the highest index. They split into two groups, explicit Adams-Bashforth formulas (with $\beta_k = 0$) and implicit Adams-Moulton formulas (with $\beta_k \neq 0$). Adams-Bashforth formulas are often written in the form

$$y_{p+1} - y_p = h \sum_{i=0}^{q} \beta_{qi} f_{p-i}\ . \tag{4.4.1}$$

The coefficients β_{qi} are listed in Table 4.4. For $q = 0$ we have the Euler method. For $q = 1$ we have

$$y_{p+1} = y_p + h\frac{(3f_p - f_{p-1})}{2}\ . \tag{4.4.2}$$

It is important that the wanted value y_{p+1} appears in (4.4.1) linearly and thus can be expressed explicitly. On the other hand the Adams-Moulton methods are implicit

$$y_p - y_{p-1} = h \sum_{i=0}^{q} \beta_{qi} f_{p-i}\ . \tag{4.4.3}$$

Table 4.4: Adams formulas

Adams-Bashforth						
i	0	1	2	3	4	5
β_{0i}	1					
$2\beta_{1i}$	3	-1				
$12\beta_{2i}$	23	-16	5			
$24\beta_{3i}$	55	-59	37	-9		
$720\beta_{4i}$	1901	-2774	2616	-1274	251	
$1440\beta_{5i}$	4227	-7673	9482	-6798	2627	-425
Adams-Moulton						
i	0	1	2	3	4	5
β_{0i}	1					
$2\beta_{1i}$	1	1				
$12\beta_{2i}$	5	8	-1			
$24\beta_{3i}$	9	19	-5	1		
$720\beta_{4i}$	251	646	-264	106	-19	
$1440\beta_{5i}$	475	1427	-798	482	-173	27

Here the wanted value y_p appears also in the nonlinear right hand side in f_p. To solve the nonlinear system of (algebraic) equations (4.4.3) with y and f being vectors, we must use some iteration method. Often a simple iteration

$$y_p^{\text{new}} - y_{p-1} = h\beta_{q0} f(x_p, y_p^{\text{old}}) + h\sum_{i=1}^{q} \beta_{qi} f_{p-i} \tag{4.4.4}$$

is used which converges for sufficiently small h.

The coefficients for Adams-Moulton methods are given in Table 4.4. For $q = 0$ we have

$$y_p = y_{p-1} + hf_p \ , \tag{4.4.5}$$

which can be called the "implicit Euler method". Pro $q = 1$ we get

$$y_p = y_{p-1} + h(f_p + f_{p-1})/2 \ , \tag{4.4.6}$$

which is called the trapezoidal rule (note the similarity with the formula for numerical evaluation of a definite integral with the same name).

The global error of the Adams-Bashforth formulas (4.4.1) is $\mathcal{O}(h^{q+1})$, for Adams-Moulton formulas (4.4.3) we get also $\mathcal{O}(h^{q+1})$. However, the order of the implicit methods is higher by one for the same number of the node points. However, we need to iterate, which is a disadvantage. A combination of an explicit and an implicit method gives the "predictor - corrector" method which is a compromise. The explicit method is used as a predictor to get the initial value of y_p to use in the iteration in the implicit method, which is compromise. When we combine the Adams-Bashforth and the Adams-Moulton method

of the 2nd order we get the final "predictor - corrector" method of the 2nd order

$$\begin{aligned} \bar{y} &= y_{p-1} + h(3f_{p-1} - f_{p-2})/2 \\ y_p &= y_{p-1} + h(f(x_p, \bar{y}) + f_{p-1})/2 \,. \end{aligned} \tag{4.4.7}$$

There are many predictor - corrector methods. Also besides Adams methods, there are other methods, as Nyström methods and Milne-Simpson methods to name a few. More details can be found in the original literature.

All the multi-step methods have one big disadvantage: it is not possible to start the computation just with knowledge of the initial condition. These methods require the knowledge of the solution (and its derivatives) in a few nodes, one of them being the point where the initial condition is given. To get this information various means are used, we mention here the two simplest ones: using the Taylor expansion when the function f is easy to differentiate and the Runge-Kutta method otherwise. It is important to use a method with the order not less than the order of the multi-step method used later. Using a high order of the multi-step method has no sense if the first few points are computed with a large error. Asymptotically (for $h \to 0$) the resulting method would have the order of the starting method, if it is lower than the order of the multi-step method used later. Using multi-step methods for systems of differential equations is formally the same, now y and f being vectors. The advantage of multi-step methods as compared to single-step methods is that the number of evaluations of the right hand side f is much lower for the same order of the method. The disadvantage is the need of starting values. Also it is difficult to adjust the step size h automatically so the effectiveness of these methods is reduced especially for cases when the solution changes its character considerably.

4.5 Numerical methods for stiff systems

Many physical problems lead to differential equations where the eigenvalues of the linearized system differ by several orders of magnitude, or they also change during integration. Such systems are called stiff. In the following we try to define stiff systems and we show their properties important for numerical integration. To start with, consider a system of linear differential equations with constant coefficients

$$\boldsymbol{y'} = \mathbf{A}\boldsymbol{y} \,, \tag{4.5.1}$$

where $y = (y_1, y_2, y_3)^T$ and the matrix $\mathbf{A}$ is

$$\mathbf{A} = \begin{pmatrix} -0.1 & -49.9 & 0 \\ 0 & -50 & 0 \\ 0 & 70 & -120 \end{pmatrix} \,. \tag{4.5.2}$$

The reader is invited to write the general solution of (4.5.1). For initial condition

$$y_1(0) = 2 \quad y_2(0) = 1 \quad y_3(0) = 2 \,. \tag{4.5.3}$$

we get

$$y_1(x) = \mathrm{e}^{-0.1x} + \mathrm{e}^{-50x} \,, \quad y_2(x) = \mathrm{e}^{-50x} \,, \quad y_3(x) = \mathrm{e}^{-50x} + \mathrm{e}^{-120x} \,. \tag{4.5.4}$$

The eigenvalues of the matrix $\mathbf{A}$ are

$$\lambda_1 = -120 \,, \quad \lambda_2 = -50 \,, \quad \lambda_3 = -0.1 \,. \tag{4.5.5}$$

The solutions y_1, y_2 and y_3 have quickly decreasing terms corresponding to the eigenvalues λ_1 and λ_2, which are negligible after a short period of x. After this short transient period, where the terms corresponding to λ_1 and λ_2 are not negligible, we could continue with numerical integration with a step size h determined by approximation of the term corresponding to λ_3. For a stable numerical integration most methods require that $|h\lambda_i|\,,\ i = 1, 2, \ldots$ be bounded by some small value roughly between 1 and 10 (here h is the integration step size and λ_i are the eigenvalues of the right hand side). As λ_1 is the largest in absolute value of the eigenvalues of the matrix $\mathbf{A}$, the stability of the method is given by the value $|120h|$. E.g. for the Euler method we need $|120h| < 2$, giving the largest possible step size being $h = 1/60$.

Let us derive this result for the system (4.5.1) with the matrix (4.5.2). The Euler method is

$$\boldsymbol{y}^{n+1} = \boldsymbol{y}^n + h\mathbf{A}\boldsymbol{y}^n = (\mathbf{E} + h\mathbf{A})\boldsymbol{y}^n\ . \tag{4.5.6}$$

As the eigenvalues of the matrix $\mathbf{A}$ are in the left complex half-plane then for $n \to \infty$ it should be that $\boldsymbol{y}^n \to \mathbf{0}$. This is governed by the eigenvalues of the matrix

$$(\mathbf{E} + h\mathbf{A}) = \begin{pmatrix} 1 - 0.1h & -49.9h & 0 \\ 0 & 1 - 50h & 0 \\ 0 & 70h & 1 - 120h \end{pmatrix}. \tag{4.5.7}$$

The eigenvalues of the matrix $(\mathbf{E} + h\mathbf{A})$ are $\lambda_1 = 1 - 0.1h,\ \lambda_2 = 1 - 50h,\ \lambda_3 = 1 - 120h$. To get $\boldsymbol{y}^n \to \mathbf{0}$ it is necessary that all the eigenvalues of the matrix $(\mathbf{E} + h\mathbf{A})$ lie inside the unit circle. This gives the condition $h < \frac{1}{60}$.

Although the term corresponding to λ_1 is negligible, the stability condition requires a very small integration step size h. As a result the integration is slow, often unnecessarily precise, without the possibility to integrate less precise. We say a system of differential equations is stiff if it is stable i.e. its eigenvalues have negative real parts and these differ by several orders of magnitude. If the system $\boldsymbol{y}' = \boldsymbol{f}(\boldsymbol{y})$ of ordinary differential equations is nonlinear, it is characterized by the eigenvalues the Jacobi matrix $\{\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\}$ of the right hand side. If in a linear system the matrix $\mathbf{A}$ depends on the independent variable x, i.e. $\mathbf{A} = \mathbf{A}(x)$, then the eigenvalues may differ with x similarly as in the nonlinear system.

Dahlquist defined the so called A-stability (absolute stability) this way. Consider the scalar equation

$$y' = \lambda y \tag{4.5.8}$$

with $\mathrm{Re}\,\lambda < 0$. We say a numerical integration method generating the sequence $y_n \doteq y(x_n)$ with the integration step size h is A-stable (absolutely stable) if in the recurrent relation describing the method used to solve (4.5.8)

$$y_{n+1} = P(h\lambda)y_n \tag{4.5.9}$$

the quantity P (depending on $h\lambda$) satisfies

$$|P(h\lambda)| < 1 \tag{4.5.10}$$

for arbitrarily large step size h, assuming $\mathrm{Re}\,\lambda < 0$. This definition means

$$|y_n| \to 0\ , \qquad n \to \infty \tag{4.5.11}$$

for any $h > 0$ assuming $\mathrm{Re}\,\lambda < 0$. There are modifications of this definition, e.g. a method is called L-stable if

$$|P(h\lambda)| \to 0\,, \qquad h \to \infty\,. \tag{4.5.12}$$

The problem of stiff systems has two sides: stability and accuracy. If we use a method that is not absolutely stable, i.e. the region of $h\lambda$ satisfying (4.5.10) does not cover the entire left complex half plane, eigenvalues with large negative part require a very small integration step size, so that the integration is not effective. If an absolutely stable method is used there are no problems with stability, but the term corresponding to the largest eigenvalues in absolute value may be approximated not very precisely for some values of the step size h.

4.5.1 Semi-implicit single-step methods

It is easy to show that none of the explicit Runge-Kutta methods presented in Table 4.3 is A-stable. E.g. consider the improved Euler method (4.2.9). For the differential equation (4.5.8) and the step size h we get

$$y_{n+1} = \Big[1 + h\lambda + \frac{1}{2}h^2\lambda^2\Big]y_n = P(h\lambda)y_n\,. \tag{4.5.13}$$

It is easy to show that for $h\lambda = -4$ we have $P(h\lambda) = 5$ and thus this method is not A-stable. Most of the A-stable methods are implicit, with the disadvantage to solve a system of nonlinear algebraic equations in each integration step using some iteration method. The Newton method (or a similar iteration method) can be used. The initial approximation is usually good enough to use 1 to 3 iterations in each step. We show an example of a semi-implicit Runge-Kutta method without the need of iteration.

Consider an autonomous system of differential equations

$$\boldsymbol{y}' = \boldsymbol{f}(\boldsymbol{y}).$$

The method can be described by this algorithm:

$$\begin{aligned} \boldsymbol{k}_1 &= h\Big(\mathbf{E} - ha_1\mathbf{J}(\boldsymbol{y}_n)\Big)^{-1}\boldsymbol{f}(\boldsymbol{y}_n) \\ \boldsymbol{k}_2 &= h\Big(\mathbf{E} - ha_2\mathbf{J}(\boldsymbol{y}_n + c_1\boldsymbol{k}_1)\Big)^{-1}\boldsymbol{f}(\boldsymbol{y}_n + b_1\boldsymbol{k}_1) \end{aligned} \tag{4.5.14}$$

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + w_1\boldsymbol{k}_1 + w_2\boldsymbol{k}_2. \tag{4.5.15}$$

Here $\mathbf{J}(\boldsymbol{y}) = \{\partial\boldsymbol{f}/\partial\boldsymbol{y}\}$ is the Jacobi matrix of the right hand side. The coefficients a_1, a_2, b_1, c_1, w_1 and w_2 are shown in Table 4.5. All these methods are A-stable as can be verified by applying them to the equation (4.5.8). Note that to find $\boldsymbol{k}_1$ and $\boldsymbol{k}_2$ the evaluation of the Jacobi matrix is needed (for the Rosenbrock method of order 3 two evaluations are needed) and also solving a system of linear algebraic equations (instead of computing the inverse matrix) is necessary. No iteration method is needed unlike the implicit methods.

There are many semi-implicit Runge-Kutta methods, here we showed only three of them. One of the first A-stable methods is the trapezoidal rule (4.4.6). Substituting into (4.5.8) we get

$$P(h\lambda) = \frac{1 + h\lambda/2}{1 - h\lambda/2}. \tag{4.5.16}$$

For $h\lambda$ from the left complex half-plane we have $|P(h\lambda)| < 1$ and thus the method is A-stable. However for $|h\lambda| \to \infty$ we have $|P(h\lambda)| \to 1$, and thus this method is not L-stable.

Table 4.5: Coefficients of semi-implicit Runge-Kutta methods

Method	Rosenbrock	Rosenbrock	Calahan
order	2.	3.	3.
a_1	$1-\sqrt{2}/2$	1.40824829	0.788675134
a_2	$1-\sqrt{2}/2$	0.59175171	0.788675134
b_1	$(\sqrt{2}-1)/2$	0.17378667	0.788675134
c_1	0	0.17378667	0
w_1	0	-0.41315432	0.75
w_2	1	1.41315432	0.25

Note that we have to use some iteration method to find y_p from (4.4.6) if the function f is nonlinear.

Another example of an A-stable method is the implicit Euler method as a special case of Adams-Moulton methods for $k = 0$ (see Table 4.3). This method is L-stable (verify it yourself) but its order in only 1 and thus it is not very effective. For solution of stiff problems free software is available, let us mention LSODE as an example.

* * *

For further study see [1], [5], [9], [10], [12], [16], [17], [26].

Chapter 5

Boundary value problem for ordinary differential equations

Nonlinear boundary value problems for ordinary differential equations often appear in chemical engineering. Examples being all models based on diffusion or heat conduction with exothermic chemical reaction, adsorption, ion exchange etc. Another important nonlinear boundary value problems are models including radiation.

The entire field of nonlinear boundary value problems is very large, often special properties of particular cases must be utilized. This chapter cannot cover all the cases, we try to show some typical approaches that can be used for a large number of boundary value problems. More interested readers can find detailed information in specialized literature.

Methods for nonlinear boundary value problems split into two main groups: difference methods and shooting methods. Besides, there are hybrid methods, e.g. multiple shooting method, collocation method, spline method etc.

5.1 Difference methods

We begin with a 2-point boundary value problem for one differential equation of the 2.nd order

$$y'' = f(x, y, y') \tag{5.1.1}$$

with linear boundary conditions

$$\alpha_0\ y(a) + \beta_0\ y'(a) \quad = \quad \gamma_0\,, \tag{5.1.2}$$

$$\alpha_1\ y(b) + \beta_1\ y'(b) \quad = \quad \gamma_1\,. \tag{5.1.3}$$

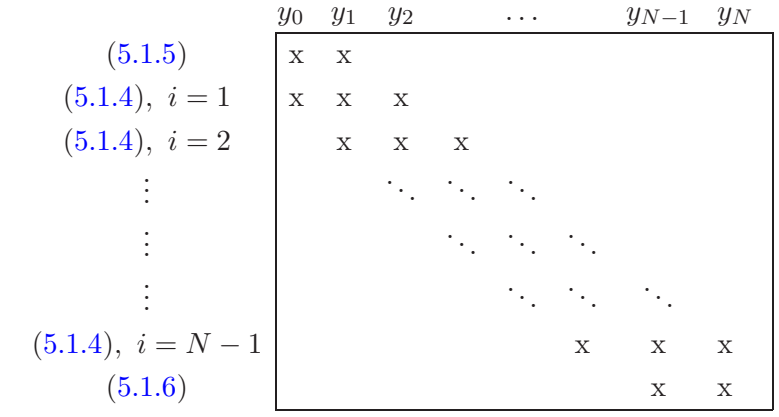
We divide the interval $[a, b]$ by an equidistant grid of points (nodes) $x_0 = a,\ x_1, \ldots, x_N = b$, $x_i = a + i\,h$, $h = (b-a)/N$. The values of the wanted solution $y(x)$ will be approximated by the values $y_i \sim y(x_i)$ in the nodes x_i. The differential equation (5.1.1) is replaced by the difference formula at x_i

$$\frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} = f\left(x_i, y_i, \frac{y_{i+1} - y_{i-1}}{2h}\right), \qquad i = 1, 2, \ldots, N-1. \tag{5.1.4}$$

Difference formulas with the error $\mathcal{O}(h^2)$ were used for both derivatives. Finally we must replace the boundary conditions (5.1.2), (5.1.3). We start with the simplest approximation

$$\alpha_0 y_0 + \beta_0 \frac{y_1 - y_0}{h} \quad = \quad \gamma_0\,, \tag{5.1.5}$$

Figure 5.1: Appearance of unknowns in (5.1.4), (5.1.5), (5.1.6)



$$\alpha_1 y_N + \beta_1 \frac{y_N - y_{N-1}}{h} \;=\; \gamma_1\,, \tag{5.1.6}$$

with the approximation error $\mathcal{O}(h)$. The equations (5.1.4), (5.1.5), (5.1.6) form a system of $N+1$ nonlinear algebraic equations for $N+1$ unknowns $y_0, y_1, \ldots, y_N$. This system can be solved using some method from chapter 3.2, usually using the Newton method. To get more precise results we choose the step-size h small, but then the number of equations $N+1$ is large. Fortunately not all equations contain all unknowns, the scheme of their appearance is 3-diagonal and thus also the Jacobi matrix used in the Newton method is 3-diagonal, i.e. it has zeroes besides 3 diagonals, see Fig. 5.1. A modified Gauss elimination is used to solved the system of linear algebraic equations in each step of the Newton method. This modified Gauss elimination uses only the nonzero elements on the three diagonals, the zero elements are not considered, they do not even have to be stored in memory. This method is called factorization.

If boundary conditions (5.1.2), (5.1.3) contain derivatives, i.e. $\beta_0 \neq 0$ or $\beta_1 \neq 0$, then approximations (5.1.5), (5.1.6) with the error $\mathcal{O}(h)$ spoil the order of approximation (5.1.4) with the error $\mathcal{O}(h^2)$. When we use differential formula with the error $\mathcal{O}(h^2)$ for boundary conditions too, we have

$$\alpha_0 y_0 + \beta_0 \frac{-3y_0 + 4y_1 - y_2}{2h} \;=\; \gamma_0\,, \quad \alpha_1 y_N + \beta_1 \frac{3y_N - 4y_{N-1} + y_{N-2}}{2h} \;=\; \gamma_1\,. \tag{5.1.7}$$

This approximation, however, changes the 3-diagonal scheme by two new appearances, one in the first row, the other in the last one. The corresponding matrix (in the Newton method) can still be transformed to a 3-diagonal matrix by adding an appropriate multiple of the 2-nd row to the 1.st row and similarly by adding an appropriate multiple of the N-the row to the $N+1$-st row.

As central difference formulas have lower error, a method of fictitious nodes is used for the approximation of the boundary condition. Then the boundary condition at $x = a$ is approximated by

$$\alpha_0 y_0 + \beta_0 \frac{y_1 - y_{-1}}{2h} \;=\; \gamma_0 \tag{5.1.8}$$

and the approximation (5.1.4) of the differential equation is considered also for $i = 0$. The new unknown y_{-1} can be expressed from (5.1.8) and the appearance scheme is again 3-diagonal.

If the equation (5.1.1) contains no first derivative, i.e. we have the equation

$$y'' = f(x, y) \tag{5.1.9}$$

and if the boundary conditions are

$$y(a) = \gamma_0\,, \qquad y(b) = \gamma_1\,, \tag{5.1.10}$$

we can use the 4-th order approximation instead of the 2-nd order approximation used above, namely

$$y_{i+1} - 2y_i + y_{i-1} = \frac{h^2}{12}\,(f_{i-1} + 10f_i + f_{i+1})\,. \tag{5.1.11}$$

Here $f_i = f(x_i, y_i)$. If we want to get the 4-th order approximation even for the equation containing the first derivative, we have to use a difference formula using more nodes. When we approximate the second derivative according to formula 10 in Table 2.2 and we approximate the first derivative according to formula 12 in Table 2.1, we get

$$\begin{aligned} \frac{-2y_{i-2} + 32y_{i-1} - 60y_i + 32y_{i+1} - 2y_{i+2}}{24h^2} \quad &= \quad f\left(x_i, y_i, \frac{y_{i-2} - 8y_{i-1} + 8y_{i+1} - y_{i+2}}{12h}\right)\,, \\ i = 2, 3, \ldots, N-2\,. & \end{aligned}$$

For $i = 1$ and $i = N-1$ we use the non-symmetric formulas and we approximate the boundary conditions by formulas of order high enough. The scheme of appearance is no more 3-diagonal and the computation time increases.

5.1.1 Difference approximation for systems of differential equations of the first order

Consider a system of differential equations of the first order

$$y_j' = f_j(x, y_1, \ldots, y_n)\,, \qquad j = 1, \ldots, n \tag{5.1.12}$$

with 2-point boundary condition

$$g_i(y_1(a), \ldots, y_n(a), y_1(b), \ldots, y_n(b)) = 0\,, \quad i = 1, \ldots, n\,. \tag{5.1.13}$$

After approximating the equations (5.1.12) in the equidistant grid of nodes $x_0 = a$, $x_1, \ldots, x_N = b$ we get

$$\begin{aligned} \frac{y_j^{i+1} - y_j^i}{h} \quad &= \quad f_j\left(\frac{x_{i+1} + x_i}{2}, \frac{y_1^{i+1} + y_1^i}{2}, \ldots, \frac{y_n^{i+1} - y_n^i}{2}\right)\,, \\ & i = 0, 1, \ldots, N-1\,; \quad j = 1, 2, \ldots, n\,; \end{aligned} \tag{5.1.14}$$

and after approximating the boundary condition (5.1.13) we have

$$g_i(y_1^0, \ldots, y_n^0, y_1^N, \ldots, y_n^N) = 0\,, \quad i = 1, \ldots, n\,. \tag{5.1.15}$$

Here we denote $y_j^i \sim y_j(x_i) = y_j(a + ih)$, $h = (b-a)/N$. We get the system of $n \cdot (N+1)$ nonlinear equations (5.1.14),(5.1.15) for $n \cdot (N+1)$ unknowns

$$(y_1^0, y_2^0, \ldots, y_n^0, y_1^1, \ldots y_n^1, \ldots, y_1^N, \ldots, y_n^N)\,. \tag{5.1.16}$$

The equations (5.1.14), (5.1.15) can be ordered as follows:

1) All the boundary conditions (5.1.15) depending on values in $x = a$ only, i.e. depending on $y_1^0, \ldots, y_n^0$.

2) Equations (5.1.14) for $i = 0$, i.e. n equations for $j = 1, 2, \ldots, n$.

3) Equation (5.1.14) for $i = 1$. (5.1.17)

. . . . . .

N+1) Equation (5.1.14) for $i = N - 1$.

N+2) Remaining boundary conditions (5.1.15), i.e. those depending on values at $x = b$, i.e. on $y_1^N, \ldots, y_n^N$.

The scheme of appearance of such an ordered system of nonlinear equations (after ordering the unknowns according to (5.1.16)) has almost a multi-diagonal band structure, see Fig. 5.2. Boundary conditions (5.1.13) with no equations containing both $y(a)$ and $y(b)$ are called separated boundary conditions. The scheme of appearance has a multi-diagonal band structure, see Fig. 5.3.

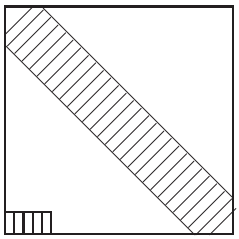


Figure 5.2: Scheme of appearance for (5.1.17)

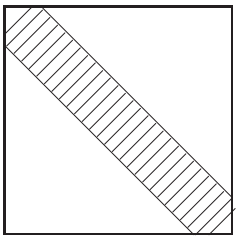


Figure 5.3: Scheme of appearance for separated boundary conditions

5.2 Conversion to initial value problem

The methods we are going to describe in this section are called shooting methods. Let us remind the difference between an initial value problem and an boundary value problem. In initial value problem the initial conditions specified in one value of the independent variable x contain enough information to start the numerical integration. In the boundary value problem, however, this information is divided into two (or more) pieces, each of them specified in different x. The main idea of the shooting method is to choose the remaining information in one x value so that we can start the integration (to shoot) and to observe, how the boundary condition in the other x value is satisfied (how the target is hit). Let us explain it more precisely. Consider the system of differential equations

$$\frac{\mathrm{d}y_i}{\mathrm{d}x} = f_i(x, y_1, \ldots, y_n) \,, \qquad i = 1, 2, \ldots, n \tag{5.2.1}$$

with 2-point boundary conditions

$$g_i\Big(y_1(a), \ldots, y_n(a), y_1(b), \ldots, y_n(b)\Big) = 0 \,, \qquad i = 1, 2, \ldots, n \,. \tag{5.2.2}$$

The problem (5.2.1), (5.2.2) can be written in a vector form

$$\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}x} = \boldsymbol{f}(x, \boldsymbol{y}) \,, \quad \boldsymbol{g}(\boldsymbol{y}(a), \boldsymbol{y}(b)) = \boldsymbol{0} \,.$$

Assume $\boldsymbol{f}$ and $\boldsymbol{g}$ have continuous derivatives according to all the arguments. If the appearance scheme of (5.2.2), (n equations in $2n$ unknowns) is in the form

a)
$$\begin{matrix} \times & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \times & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \times & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \times & 0 & 0 & 0 & 0 & 0 \end{matrix}$$
resp. b)
$$\begin{matrix} \times & \times & \times & \times & \times & 0 & 0 & 0 & 0 & 0 \\ \times & \times & \times & \times & \times & 0 & 0 & 0 & 0 & 0 \\ \times & \times & \times & \times & \times & 0 & 0 & 0 & 0 & 0 \\ \times & \times & \times & \times & \times & 0 & 0 & 0 & 0 & 0 \\ \times & \times & \times & \times & \times & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

(here $n = 5$), then it is an initial value problem (a Cauchy problem) in $x = a$ in a standard form or a Cauchy problem where the initial condition can be found by solving n equations (5.2.2) in n unknowns. After solving this system we again have all the n conditions in $x = a$ necessary to start the integration.

Now, suppose that the complete initial conditions cannot be found from (5.2.2). Instead, consider some other initial condition

$$y_1(a) = \eta_1, \ldots, y_n(a) = \eta_n \tag{5.2.3}$$

and suppose the Cauchy problem (5.2.1) with this initial condition has a unique solution for any $\boldsymbol{\eta} = (\eta_1, \eta_2, \ldots, \eta_n)$ in some domain $M \subset R^n$. Then the solution of (5.2.1), (5.2.3) for any fixed $x \in [a, b]$ defines in this domain M a unique vector-valued function depending on n variables - the components of the vector $\boldsymbol{\eta}$:

$$\boldsymbol{y}(x) = \boldsymbol{w}(x, \boldsymbol{\eta}) \; . \tag{5.2.4}$$

For $x = b$ we have $\boldsymbol{y}(b) = \boldsymbol{w}(b, \boldsymbol{\eta})$. Substituting into boundary condition (5.2.2) we have

$$\boldsymbol{g}(\boldsymbol{\eta}, \boldsymbol{w}(b, \boldsymbol{\eta})) = 0 \; , \tag{5.2.5}$$

or

$$\boldsymbol{G}(\boldsymbol{\eta}) = \boldsymbol{0} \; . \tag{5.2.6}$$

Now (5.2.6) is a system of n nonlinear algebraic equations for n unknowns $\eta_1, \eta_2, \ldots, \eta_n$ (of course it is not written by elementary functions if the system of differential equations (5.2.1) cannot be solved analytically).

We have the following result: If for any $\boldsymbol{\eta}$ there exists a solution of the Cauchy problem (5.2.1) with the initial condition (5.2.3) on the interval $[a, b]$ then the number of solutions of the boundary value problem is the same as the number of solutions of the equation (5.2.6) in the corresponding domain. If the equation (5.2.6) has no solution, then the boundary value problem (5.2.1), (5.2.2) has no solution either.

The main task is to find $\boldsymbol{\eta}$ satisfying $\boldsymbol{G}(\boldsymbol{\eta}) = \boldsymbol{0}$. In other words we want to find an initial condition for (5.2.1) in $x = a$ satisfying the boundary condition (5.2.2). This can be achieved by various methods for nonlinear algebraic equations.

Boundary conditions have been formulated in a rather general way, including also so-called mixed boundary conditions, meaning values of $\boldsymbol{y}$ both in $x = a$ and in $x = b$ appear in the function g_i. Many practical problems involve separated boundary conditions, meaning values of $\boldsymbol{y}$ either in $x = a$ or in $x = b$ appear in each function g_i. Then in the appearance scheme for (5.2.2) in each row either the first n entries are zeroes or the last n entries are zeroes which may (for $n = 5$) look like this

$$\begin{matrix} \times & \times & \times & \times & \times & 0 & 0 & 0 & 0 & 0 \\ \times & \times & 0 & 0 & \times & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & 0 & 0 & \times & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \times & \times & 0 \end{matrix} \; . \tag{5.2.7}$$

Let us introduce the term problem of order p in the point $x = a$ (or $x = b$ resp.). We say that a boundary value problem with separated boundary conditions is of order p in $x = a$ (or in $x = b$ resp.) if $p = n - r$ where r is the number of functions g_i in (5.2.2) depending on $\boldsymbol{y}(a)$ (or on $\boldsymbol{y}(b)$ resp.). E.g. the problem described by the scheme (5.2.7) is of order 3 in $x = a$ and it is of order 2 in $x = b$. It is obvious that if a given problem with separated boundary conditions is of order p in $x = a$ then it is of order $(n - p)$ in $x = b$.

In simple words in a point where the problem is of order p we must choose p initial conditions and to compute the remaining $n - p$ ones from the boundary conditions. The problem can be converted into an initial value problem either in $x = a$ or in $x = b$ and it is convenient to choose $x = a$ or $x = b$ according to where the order is lower.

5.2.1 Problem of order 1

To start with consider the differential equation (5.1.1) written as a system of differential equations of the first order

$$\begin{array}{rcl} y_1' &=& y_2, \\ y_2' &=& f(x, y_1, y_2). \end{array} \tag{5.2.8}$$

Boundary conditions (5.1.2), (5.1.3) are then

$$\begin{array}{rcccl} \alpha_0 y_1(a) &+& \beta_0 y_2(a) &=& \gamma_0, \\ \alpha_1 y_1(b) &+& \beta_1 y_2(b) &=& \gamma_1. \end{array} \tag{5.2.9}$$

The appearance scheme for (5.2.9) is for nonzero α_i, β_i, $i = 0, 1$, in the form

$$\begin{array}{cccc} \times & \times & 0 & 0 \\ 0 & 0 & \times & \times \end{array} .$$

Thus it is a problem with separated boundary conditions. As this is a problem of order 1 in $x = a$ (and also in $x = b$) we must choose one condition in $x = a$ (or in $x = b$). Assuming $\beta_0 \neq 0$ we choose the initial condition

$$y_1(a) = \eta_1 \tag{5.2.10}$$

and we compute

$$y_2(a) = \eta_2 = \frac{\gamma_0 - \alpha_0 \eta_1}{\beta_0} \tag{5.2.11}$$

from the first equation (5.2.9). When integrating (5.2.8) with the initial conditions (5.2.10) and (5.2.11) we get $y_1(b) = y_1(b, \eta_1)$ and $y_2(b) = y_2(b, \eta_1)$, dependent on the choice of η_1. These values must satisfy the boundary conditions (5.2.9). The first of them is automatically satisfied by the choice of (5.2.11), the second one can be written as

$$\alpha_1 y_1(b, \eta_1) + \beta_1 y_2(b, \eta_1) - \gamma_1 = \varphi(\eta_1) = 0 \, . \tag{5.2.12}$$

Now, after choosing η_1, we can compute the value of $\varphi(\eta_1)$ according to (5.2.12) using some method for numerical integration of initial value problem. To solve the equation $\varphi(\eta_1) = 0$ we use some method from chapter 3. Efficient methods use derivatives, an example being the Newton's method or the Richmond's method. The derivative can be found using some difference formula, but this is not very precise, since the numerical integration itself introduces certain error. A better choice is to consider variation

$$\Omega_1 = \frac{\partial y_1}{\partial y_1(a)} = \frac{\partial y_1}{\partial \eta_1} \, , \qquad \Omega_2 = \frac{\partial y_2}{\partial y_1(a)} = \frac{\partial y_2}{\partial \eta_1} \, . \tag{5.2.13}$$

The equations for Ω_1 and Ω_2 can be derived by differentiating (5.2.8) with respect to η_1 and interchanging the differentiation with respect to x and η_1

$$\begin{aligned} \Omega_1' &= \Omega_2 \,, \\ \Omega_2' &= \tfrac{\partial f}{\partial y_1}\Omega_1 + \tfrac{\partial f}{\partial y_2}\Omega_2 \end{aligned} \tag{5.2.14}$$

with the initial conditions

$$\Omega_1(a) = 1 \,, \qquad \Omega_2(a) = -\frac{\alpha_0}{\beta_0} \tag{5.2.15}$$

derived from (5.2.10) and (5.2.11). From (5.2.12) we have

$$\frac{\mathrm{d}\varphi(\eta_1)}{\mathrm{d}\eta_1} = \alpha_1\Omega_1(b) + \beta_1\Omega_2(b) \,. \tag{5.2.16}$$

Then the Newton's method can be written as

$$\eta_1^{k+1} = \eta_1^k - \frac{\varphi(\eta_1^k)}{\varphi'(\eta_1^k)} = \eta_1^k - \frac{\alpha_1 y_1(b) + \beta_1 y_2(b) - \gamma_1}{\alpha_1\Omega_1(b) + \beta_1\Omega_2(b)} \,, \tag{5.2.17}$$

where $y_1(b), y_2(b), \Omega_1(b), \Omega_2(b)$ are evaluated for $\eta_1 = \eta_1^k$.

The following example illustrates this method.

Example 5.2.1 *Consider the equation describing non-isothermal inner diffusion in a slab catalyst with the concentration* $y \in [0,1]$

$$y'' = \Phi^2 \; y \; \exp\left(\frac{\gamma\beta(1-y)}{1+\beta(1-y)}\right) \tag{5.2.18}$$

with boundary conditions

$$y'(0) = 0 \,, \qquad y(1) = 1. \tag{5.2.19}$$

Introducing $y_1 = y$, $y_2 = y'$ *the equation (5.2.18) can be written as*

$$y_1' = y_2 \,, \qquad y_2' = \Phi^2 \; y_1 \; \exp\left(\frac{\gamma\beta(1-y_1)}{1+\beta(1-y_1)}\right) \,. \tag{5.2.20}$$

We choose

$$y_1(0) = \eta_1 \tag{5.2.21}$$

and from (5.2.19) using $y_2 = y'$ *we have*

$$y_2(0) = 0 \,. \tag{5.2.22}$$

The function φ *is then defined by the expression*

$$\varphi(\eta_1) = y_1(1) - 1 \,. \tag{5.2.23}$$

The variational equations corresponding to (5.2.20) are

$$\begin{aligned} \Omega_1' &= \Omega_2 \,, \\ \Omega_2' &= \Phi^2 \exp\left(\frac{\gamma\beta(1-y_1)}{1+\beta(1-y_1)}\right) \cdot \left(1 - \frac{\gamma\beta y_1}{(1+\beta(1-y_1))^2}\right) \; \Omega_1 \end{aligned} \tag{5.2.24}$$

and the initial conditions are

$$\Omega_1(0) = 1 \,, \qquad \Omega_2(0) = 0 \,. \tag{5.2.25}$$

The numerical integration of the initial value problem (5.2.20), (5.2.24) with initial conditions (5.2.21), (5.2.22) and (5.2.25) was done using the Merson modification of the Runge-Kutta method. The results are shown in Table 5.1. The convergence is very fast.

Table 5.1: Newton method for Example 5.2.1 ($\gamma = 20$; $\beta = 0.1$; $\Phi = 1$)

$y(0) = \eta_1$	$y(1)$	$y'(1)$	$\Omega_1(1)$	$\varphi(\eta_1)$	$\varphi'(\eta_1)$
1.00000	1.45949	0.84223	0.53898	0.45949	0.53898
0.14747	0.58712	1.00124	2.68144	−0.41288	2.68144
0.30145	0.89906	1.21398	1.53643	−0.10094	1.53643
0.36715	0.99073	1.23051	1.26792	−0.00927	1.26792
0.37446	0.99991	1.23081	1.24276	−0.00009	1.24276
0.37453	1.00000	1.23081	1.24251	0.00000	1.24251
0.50000	1.13356	1.20276	0.91577	0.13356	0.91577
0.35416	0.97396	1.22931	1.31470	−0.02604	1.31470
0.37396	0.99929	1.23080	1.24444	−0.00071	1.24444
0.37453	1.00000	1.23081	1.24251	0.00000	1.24251
0.10000	0.44534	0.83312	3.32963	−0.55466	3.32963
0.26658	0.84243	1.19239	1.71764	−0.15757	1.71764
0.35832	0.97940	1.22979	1.29940	−0.02060	1.29940
0.37417	0.99955	1.23080	1.24373	−0.00045	1.24373
0.37453	1.00000	1.23081	1.24251	0.00000	1.24251

5.2.2 Problem of higher order

Boundary conditions (5.2.2) for the system of equations (5.2.1) are for the problem with separated boundaries in the form of

$$g_i(y_1(a), \ldots, y_n(a)) = 0\,, \qquad i = 1, 2, \ldots, r \tag{5.2.26}$$

$$g_i(y_1(b), \ldots, y_n(b)) = 0\,, \qquad i = r+1, \ldots, n\,. \tag{5.2.27}$$

Problem (5.2.1), (5.2.26), (5.2.27) is thus of order $n - r$ in $x = a$ and of order r in $x = b$. After choosing $n - r$ "missing" values of initial conditions in $x = a$

$$y_1(a) = \eta_1, \quad y_2(a) = \eta_2, \ldots, y_{n-r}(a) = \eta_{n-r}\,, \tag{5.2.28}$$

it is possible to solve r values

$$y_{n-r+1}(a) = \eta_{n-r+1}, \ldots, y_n(a) = \eta_n \tag{5.2.29}$$

from (5.2.26), possibly after a suitable rearrangement of $(y_1, ..., y_n)$. As a result we have n conditions (5.2.28) and (5.2.29) in $x = a$, this presenting a Cauchy (initial value) problem. After integrating this initial value problem in the interval $[a, b]$ we get the values $y_1(b), \ldots, y_n(b)$, dependent on the chosen initial conditions (5.2.28). These values must also satisfy the conditions (5.2.27) (so far unused)

$$g_i\Big(y_1(b, \eta_1, \ldots, \eta_{n-r}), \ldots, y_n(b, \eta_1, \ldots, \eta_{n-r})\Big) = 0\,, \qquad i = r+1, \ldots, n\,. \tag{5.2.30}$$

The equations (5.2.30) can be written as

$$G_i(\eta_1, \ldots, \eta_{n-r}) = 0\,, \qquad i = 1, \ldots, n-r\,. \tag{5.2.31}$$

To solve this system we can use some method from chapter 3. So we are able to evaluate $G_1, \dots, G_{n-r}$ for given $\eta_1, \dots, \eta_{n-r}$. Without the knowledge of derivatives of G_i Warner scheme can be applied (see section **??**). To do this we have to evaluate the functions G_i for $n-r+1$ different values $\eta_1^k, \dots, \eta_{n-r}^k$, $k = 1, 2, \dots, n-r+1$, meaning we have to solve the initial value problem (5.2.1), (5.2.28), (5.2.29) with $(n-r+1)$ different initial conditions (5.2.28) (thus $(n-r+1)$ times).

The system (5.2.31) can also be solved by some method from chapter 3 that uses derivatives if the derivatives of the functions G_i are known. Let us try to derive the Newton's method for system (5.2.31), thus for the boundary value problem of order $n-r$ in $x = a$. To find the Jacobi matrix we must compute the partial derivatives $\frac{\partial G_i}{\partial \eta_j}, i, j = 1, 2, \dots, n-r$. Considering (5.2.30) we have

$$\frac{\partial G_i}{\eta_j} = \sum_{k=1}^{n} \frac{\partial g_{i+r}}{\partial y_k(b)} \frac{\partial y_k(b)}{\partial \eta_j} , \qquad i, j = 1, 2, \dots, n-r . \tag{5.2.32}$$

After differentiating the system (5.2.1) with respect to η_j and denoting

$$\Omega_{kj} = \frac{\partial y_k}{\partial \eta_j} , \qquad k = 1, 2, \dots, n , \quad j = 1, 2, \dots, n-r , \tag{5.2.33}$$

and changing the order of differentiation we get a system of variational differential equations

$$\frac{\mathrm{d}\Omega_{kj}}{\mathrm{d}x} = \sum_{m=1}^{n} \frac{\partial f_k}{\partial y_m} \Omega_{mj} , \qquad k = 1, 2, \dots, n , \quad j = 1, 2, \dots, n-r . \tag{5.2.34}$$

In view of the initial condition (5.2.28) the variational variables Ω_{kj} satisfy the initial conditions

$$\Omega_{kj}(a) = \begin{cases} 0 & \text{pro} \quad k \neq j \\ 1 & \text{pro} \quad k = j \end{cases} \qquad k, j = 1, 2, \dots, n-r . \tag{5.2.35}$$

The remaining initial conditions can be found from the conditions (5.2.26) assuming the system of r equations (5.2.26) is solvable in r variables $y_{n-r+1}(a), y_{n-r+2}(a), \dots \dots, y_n(a)$, thus

$$y_k(a) = \Phi_k\Big(y_1(a), y_2(a), \dots, y_{n-r}(a)\Big) , \qquad k = n-r+1, \dots, n . \tag{5.2.36}$$

Then

$$\frac{\partial y_k(a)}{\partial \eta_j} = \Omega_{kj}(a) = \frac{\partial \Phi_k(\eta_1, \dots, \eta_{n-r})}{\partial \eta_j}, \quad k = n-r+1, \dots, n, \; j = 1, 2, \dots, n-r . \tag{5.2.37}$$

Even in case the equations (5.2.36) cannot be solved explicitly, we still can get (5.2.37) as a solution of some system of linear algebraic equations using the Implicit function theorem. The relations (5.2.35) and (5.2.37) present a complete set of $n(n-r)$ initial conditions for $n(n-r)$ functions Ω_{kj} and $n(n-r)$ differential equations (5.2.34).

To conclude we integrate the system of equations (5.2.1) with initial conditions (5.2.28) and

$$y_k(a) = \Phi_k(\eta_1, \eta_2, \dots, \eta_{n-r}) , \qquad k = n-r+1, \dots, n , \tag{5.2.38}$$

and the system of equations (5.2.34) with initial conditions (5.2.35) and (5.2.37) simultaneously, this is an initial value problem of $n + n(n-r)$ differential equations with the same

number of initial conditions. For chosen $\eta_1, \eta_2, \ldots, \eta_{n-r}$ we have in $x = b$

$$\begin{array}{l} y_1(b), \quad y_2(b), \ldots, y_n(b) \\ \Omega_{11}(b), \quad \Omega_{12}(b), \ldots, \Omega_{1,n-r}(b) \\ \vdots \\ \Omega_{n1}(b), \quad \Omega_{n2}(b), \ldots, \Omega_{n,n-r}(b). \end{array}$$

We can evaluate G_i from (5.2.31) and (5.2.30) and we can find the Jacobi matrix of the functions G_i from (5.2.32), where $\partial y_k(b)/\partial \eta_j$ is replaced by $\Omega_{kj}(b)$. We have all we need for the Newton's method.

This shooting method for boundary value problems is a reliable algorithm. The method is widely applicable if initial value problem can be integrated. In some problems the numerical integration can be done from one side only or it cannot be integrated from either side. For these problems the shooting method must be modified (the multiple shooting method) or it cannot be applied at all.

The following example illustrates the use of variational equations once again.

Example 5.2.2 *The stationary regime of a homogeneous exothermic reaction of the first order in a tube non-isothermal non-adiabatic flow-through system can be described by the equations* ($' = \frac{d}{dx}$)*:*

$$\frac{1}{\mathrm{Pe}}\,\theta'' - \theta' - \beta(\theta - \theta_c) + \mathrm{B}\,\mathrm{Da}\,(1-y)\,\exp\left(\frac{\theta}{1+\varepsilon\theta}\right) = 0\,, \tag{5.2.39}$$

$$\frac{1}{\mathrm{Pe}}\,y'' - y' + \mathrm{Da}\,(1-y)\,\exp\left(\frac{\theta}{1+\varepsilon\theta}\right) = 0 \tag{5.2.40}$$

with boundary conditions

$$x = 0: \qquad \theta' = \mathrm{Pe}\,\theta\,; \quad y' = \mathrm{Pe}\,y \tag{5.2.41}$$

$$x = 1: \qquad \theta' = 0\,; \qquad y' = 0 \quad . \tag{5.2.42}$$

Here y is the dimensionless conversion, θ is the dimensionless temperature, Pe *is the Peclet criterion, x is the dimensionless space coordinate,* B *is the dimensionless adiabatic thermal increase,* Da *is the Damköhler criterion, ε is the dimensionless activation energy, β is the dimensionless thermal throughput coefficient, θ_c is the dimensionless cooling medium temperature.*

We convert the problem to the initial value problem in $x = 1$ (this is better from the numerical point of view, for higher Pe it is not possible to convert it to the initial value problem in $x = 0$ at all due to instability of the integration of the corresponding initial value problem) and we use the Newton's method. Thus we choose

$$\theta(1) = \eta_1; \qquad y(1) = \eta_2 \tag{5.2.43}$$

and the conditions (5.2.42) give the remaining two initial values necessary for the integration. Let us denote the variation variables

$$\Omega_{11} = \frac{\partial\theta}{\partial\eta_1}\,; \quad \Omega_{12} = \frac{\partial\theta}{\partial\eta_2}\,; \quad \Omega_{21} = \frac{\partial y}{\partial\eta_1}\,; \quad \Omega_{22} = \frac{\partial y}{\partial\eta_2}\,. \tag{5.2.44}$$

For these functions we get the equations

$$\frac{1}{\mathrm{Pe}}\Omega_{11}'' - \Omega_{11}' - \beta\Omega_{11} + \mathrm{B\ Da}\ \exp\left(\frac{\theta}{1+\varepsilon\theta}\right)\cdot\left(-\Omega_{21} + \frac{1-y}{(1+\varepsilon\theta)^2}\Omega_{11}\right) = 0 \quad (5.2.45)$$

$$\frac{1}{\mathrm{Pe}}\Omega_{12}'' - \Omega_{12}' - \beta\Omega_{12} + \mathrm{B\ Da}\ \exp\left(\frac{\theta}{1+\varepsilon\theta}\right)\cdot\left(-\Omega_{22} + \frac{1-y}{(1+\varepsilon\theta)^2}\Omega_{12}\right) = 0 \quad (5.2.46)$$

$$\frac{1}{\mathrm{Pe}}\Omega_{21}'' - \Omega_{21}' + \mathrm{Da}\ \exp\left(\frac{\theta}{1+\varepsilon\theta}\right)\cdot\left(-\Omega_{21} + \frac{1-y}{(1+\varepsilon\theta)^2}\Omega_{11}\right) = 0 \quad (5.2.47)$$

$$\frac{1}{\mathrm{Pe}}\Omega_{22}'' - \Omega_{22}' + \mathrm{Da}\ \exp\left(\frac{\theta}{1+\varepsilon\theta}\right)\cdot\left(-\Omega_{22} + \frac{1-y}{(1+\varepsilon\theta)^2}\Omega_{12}\right) = 0\,. \quad (5.2.48)$$

The equations (5.2.45) and (5.2.47) come from differentiation of (5.2.39) and (5.2.40) with respect to η_1, the equations (5.2.46) and (5.2.48) come from differentiation with respect to η_2. We let the equations of the second order and we do not convert them into a system of 1-st order equations for clear arrangement. The initial conditions for (5.2.45) - (5.2.48) are

$$\Omega_{11}(1) = 1\,;\quad \Omega_{12}(1) = 0\,;\quad \Omega_{21}(1) = 0\,;\quad \Omega_{22}(1) = 1\,; \quad (5.2.49)$$

$$\Omega_{11}'(1) = \Omega_{12}'(1) = \Omega_{21}'(1) = \Omega_{22}'(1) = 0\,. \quad (5.2.50)$$

To satisfy the boundary conditions (5.2.41) we must solve

$$G_1(\eta_1,\eta_2) = \mathrm{Pe}\ \theta(0) - \theta'(0) = 0 \quad (5.2.51)$$

$$G_2(\eta_1,\eta_2) = \mathrm{Pe}\ y(0) - y'(0) = 0\,. \quad (5.2.52)$$

Partial derivatives for the Jacobi matrix are

$$\begin{aligned}
&\frac{\partial G_1}{\partial\eta_1} = \mathrm{Pe}\ \Omega_{11}(0) - \Omega_{11}'(0) = a_{11}\,, \quad \frac{\partial G_1}{\partial\eta_2} = \mathrm{Pe}\ \Omega_{12}(0) - \Omega_{12}'(0) = a_{12}\,,\\
&\frac{\partial G_2}{\partial\eta_1} = \mathrm{Pe}\ \Omega_{21}(0) - \Omega_{21}'(0) = a_{21}\,, \quad \frac{\partial G_2}{\partial\eta_2} = \mathrm{Pe}\ \Omega_{22}(0) - \Omega_{22}'(0) = a_{22}\,.
\end{aligned} \quad (5.2.53)$$

For a given $\boldsymbol{\eta} = (\eta_1,\eta_2)$ we can integrate the equations (5.2.39), (5.2.40), (5.2.45)-(5.2.48) with the initial conditions (5.2.42), (5.2.43), (5.2.49), (5.2.50) from $x = 1$ to $x = 0$. In this way we get the values of all the functions y, θ, Ω_{ij} along with their derivatives in $x = 0$. Then we can evaluate G_1 and G_2 using (5.2.51), (5.2.51) and the Jacobi matrix using (5.2.53). Table 5.2 gives the results of the Newton's method for one initial approximation $\eta = (0;0)$ for the following parameter values

$$\mathrm{Pe} = 2\,;\quad \beta = 2\,;\quad \theta_c = 0\,;\quad \mathrm{B} = 12\,;\quad \mathrm{Da} = 0.12\,;\quad \varepsilon = 0\,. \quad (5.2.54)$$

Table 5.3. shows the iterations for four other initial approximations $\boldsymbol{\eta}$. These two tables show that we have found five different solutions of the boundary value problem (5.2.39), (5.2.42). The solutions $\theta(x)$ and $y(x)$ are plotted in Fig. 5.4. The solution from Table 5.2 is denoted `e`*, other solutions are denoted* `a`, `b`, `c`, `d` *in agreement with Table 5.3. This example illustrates that a boundary value problem (especially a nonlinear one) can have more than one solution. On the other hand, such a problem can have no solution.*

* * *

For further study the reader is invited to check the following literature [5], [16], [17], [22], [27].

Table 5.2: Newton method for Example 5.2.2

	iteration					
	0	1	2	3	4	5
η_1	0.0000	0.7395	1.0299	1.0932	1.0963	1.0963
η_2	0.0000	0.1570	0.2206	0.2340	0.2346	0.2346
$\theta(0)$	−0.9236	0.1165	0.4170	0.4732	0.4759	0.4759
$\theta'(0)$	1.6624	1.0066	0.9499	0.9516	0.9518	0.9518
$y(0)$	−0.0568	0.0496	0.0866	0.0936	0.0940	0.0940
$y'(0)$	0.0680	0.1416	0.1790	0.1857	0.1880	0.1880
G_1	−3.5150	−0.7736	−0.1160	−0.0051	0.0000	0.0000
G_2	−0.1816	−0.0424	−0.0057	−0.0002	0.0000	0.0000
$\Omega_{11}(0)$	1.5021	0.8118	0.5151	0.4503	0.4471	
$\Omega'_{11}(0)$	−1.1431	0.0906	0.5023	0.5789	0.5825	
$\Omega_{12}(0)$	0.7947	1.5142	1.8810	1.9658	1.9700	
$\Omega'_{12}(0)$	−1.2645	−2.1345	−2.4099	−2.4557	−2.4578	
$\Omega_{21}(0)$	−0.0621	−0.1043	−0.1215	−0.1251	−0.1253	
$\Omega'_{21}(0)$	0.0838	0.1339	0.1438	0.1447	0.1447	
$\Omega_{22}(0)$	1.0473	1.0881	1.1075	1.1118	1.1120	
$\Omega'_{22}(0)$	−0.0424	−0.0540	−0.0434	−0.0391	−0.0389	
a_{11}	4.1474	1.5330	0.5279	0.3218	0.3118	
a_{12}	2.8539	5.1630	6.1718	6.3873	6.3977	
a_{21}	−0.2081	−0.3425	−0.3868	−0.3950	−0.3953	
a_{22}	2.1370	2.2303	2.2583	2.2627	2.2628	
$\triangle\eta_1$	0.7395	0.2904	0.0633	0.0031	0.0000	
$\triangle\eta_2$	0.1570	0.0636	0.0134	0.0006	0.0000	

Table 5.3: Newton method for Example 5.2.2

a		b		c		d	
η_1	η_2	η_1	η_2	η_1	η_2	η_1	η_2
2.0000	0.0000	4.0000	0.7500	2.9000	0.9800	3.6000	0.9500
2.1644	0.4378	3.1441	0.6149	3.2155	0.9815	3.6781	0.9396
4.4148	0.8706	3.1447	0.6189	3.2114	0.9853	3.6919	0.9374
4.1768	0.8817	3.1448	0.6189	3.2132	0.9848	3.6926	0.9373
4.1098	0.8949			3.2133	0.9848	3.6926	0.9373
4.0792	0.8971						
4.0775	0.8973						
4.0774	0.8973						

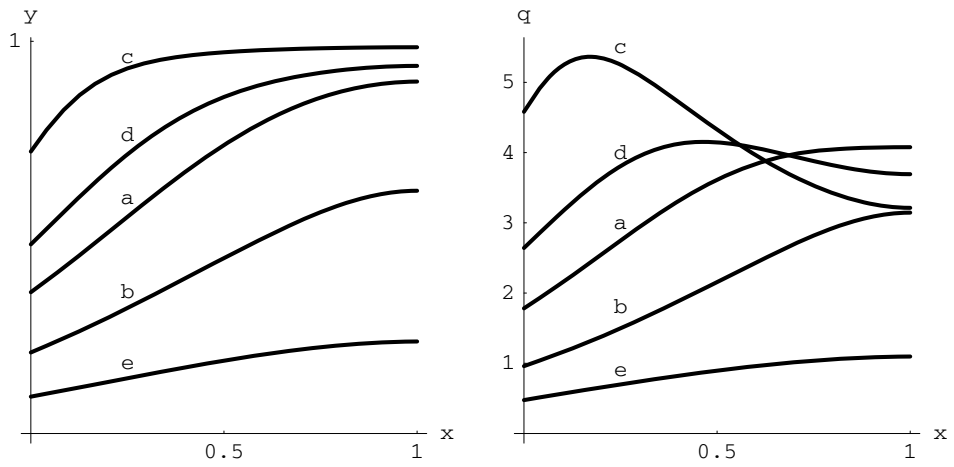


Figure 5.4: Five different solutions of the boundary value problem from Example 5.2.2

Chapter 6

Parabolic partial differential equations

Parabolic PDE (partial differential equations) belong to problems often encountered in chemical engineering. Non-stationary heat conduction or mass transport by diffusion lead to parabolic equations. Many problems are described by linear parabolic equations (simple problems in diffusion and heat conduction) which can be solved by classical analysis. The solution is in the form of an infinite series of special functions (e.g. Bessel and Hankel functions) and these functions must be evaluated which may be expensive. Thus even linear equations are often solved numerically. Many problems involve nonlinear parabolic equations (heat and mass exchange with exothermic reaction, adsorption, non-stationary heat exchange with radiation etc.). Nonlinear parabolic equations must always be solved numerically. The aim of this chapter is to give introduction to numerical analysis used in parabolic equations - the implicit and explicit difference schemes.

6.1 Canonical form of second order equations with two independent variables

Consider a quasilinear equation of the second order with two independent variables x and y in a given domain $D \subset R^2$:

$$A\frac{\partial^2 u}{\partial x^2} + 2B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + F\Big(x, y, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}\Big) = 0, \tag{6.1.1}$$

where the coefficients A, B and C are functions of x and y and have continuous derivatives up to order at least 2. Suppose that at least one of them is always nonzero. Corresponding to equation (6.1.1) we can write the quadratic form

$$At_1^2 + 2Bt_1t_2 + Ct_2^2\,. \tag{6.1.2}$$

Depending on the values of $A,\ B$ and C we distinguish three types of equation (6.1.1), see Tab.6.1.
We can introduce two new independent variables (X, Y) instead of (x, y) by the functions

$$X = X(x, y)\,, \qquad Y = Y(x, y)\,, \tag{6.1.3}$$

Table 6.1: Types of equations

Type	Condition
hyperbolic	$B^2 - AC > 0$
parabolic	$B^2 - AC = 0$
elliptic	$B^2 - AC < 0$

which are assumed to be twice continuously differentiable and to have nonzero Jacobian

$$\frac{D(X,Y)}{D(x,y)} = \begin{vmatrix} \dfrac{\partial X}{\partial x} & \dfrac{\partial X}{\partial y} \\ \dfrac{\partial Y}{\partial x} & \dfrac{\partial Y}{\partial y} \end{vmatrix} \neq 0 \tag{6.1.4}$$

in the domain D considered.

Putting (6.1.3) into (6.1.1), equation (6.1.1) changes to

$$\bar{A}\frac{\partial^2 u}{\partial X^2} + 2\bar{B}\frac{\partial^2 u}{\partial X \partial Y} + \bar{C}\frac{\partial^2 u}{\partial Y^2} + \bar{F}(X, Y, u, \frac{\partial u}{\partial X}, \frac{\partial u}{\partial Y}) = 0\,, \tag{6.1.5}$$

where

$$\begin{aligned} \bar{A}(X,Y) &= A\left(\frac{\partial X}{\partial x}\right)^2 + 2B\frac{\partial X}{\partial x}\frac{\partial X}{\partial y} + C\left(\frac{\partial X}{\partial y}\right)^2, \\ \bar{C}(X,Y) &= A\left(\frac{\partial Y}{\partial x}\right)^2 + 2B\frac{\partial Y}{\partial x}\frac{\partial Y}{\partial y} + C\left(\frac{\partial Y}{\partial y}\right)^2, \\ \bar{B}(X,Y) &= A\frac{\partial X}{\partial x}\frac{\partial Y}{\partial x} + B\left(\frac{\partial X}{\partial x}\frac{\partial Y}{\partial y} + \frac{\partial X}{\partial y}\frac{\partial Y}{\partial x}\right) + C\frac{\partial X}{\partial y}\frac{\partial Y}{\partial y}\,. \end{aligned} \tag{6.1.6}$$

It is easy to show that

$$\bar{B}^2 - \bar{A}\bar{C} = (B^2 - AC)\left(\frac{\partial X}{\partial x}\frac{\partial Y}{\partial y} - \frac{\partial X}{\partial y}\frac{\partial Y}{\partial x}\right)^2 \tag{6.1.7}$$

thus transformation (6.1.3) does not change the type of equation (6.1.1). Transformation (6.1.3) can be chosen so that exactly one of the following three conditions holds

$$\bar{A} = 0 \quad \wedge \quad \bar{C} = 0\,, \tag{6.8a}$$

$$\bar{A} = 0 \quad \wedge \quad \bar{B} = 0 \;\text{ or }\; \bar{B} = 0 \quad \wedge \quad \bar{C} = 0\,, \tag{6.8b}$$

$$\bar{A} = \bar{C} \quad \wedge \quad \bar{B} = 0\,. \tag{6.8c}$$

In each of these three cases (which differ in the sign of the expression $(B^2 - AC)$) equation (6.1.5) can be written in simple (canonical) form:

1. $(B^2 - AC) > 0$ *hyperbolic equation*
   The canonical form is

$$\frac{\partial^2 u}{\partial X \partial Y} = F_1\left(X, Y, u, \frac{\partial u}{\partial X}, \frac{\partial u}{\partial Y}\right). \tag{6.1.9}$$

Often another form is used as the canonical one, namely

$$\frac{\partial^2 u}{\partial \xi^2} - \frac{\partial^2 u}{\partial \eta^2} = F_2\left(\xi, \eta, u, \frac{\partial u}{\partial \xi}, \frac{\partial u}{\partial \eta}\right); \tag{6.1.10}$$

this equation can be derived from (6.1.9) by the transformation

$$X = \xi + \eta\,, \qquad Y = \xi - \eta\,.$$

These types of equations appear seldom in chemical engineering so we will not consider them in this text.

2. $(B^2 - AC) = 0$ *parabolic equation*
   The canonical form is
   $$\frac{\partial^2 u}{\partial Y^2} = F_3\left(X, Y, u, \frac{\partial u}{\partial X}, \frac{\partial u}{\partial Y}\right). \tag{6.1.11}$$

3. $(B^2 - AC) < 0$ *elliptic equation*
   The canonical form is
   $$\frac{\partial^2 u}{\partial X^2} + \frac{\partial^2 u}{\partial Y^2} = F_4\left(X, Y, u, \frac{\partial u}{\partial X}, \frac{\partial u}{\partial Y}\right). \tag{6.1.12}$$

6.2 Numerical solution of parabolic equations with two independent variables

Numerical solution of parabolic equations in two dimensions (or in one spatial coordinate x and one time coordinate t) is thoroughly treated in literature (as opposed to higher dimensional cases). As chemical engineering problems often lead to equations in time and one spatial coordinate, one section is devoted to this problem. Let us start with the linear equation. Later we will see that almost all the conclusion for the linear equation can be used for the nonlinear one as well.

6.2.1 Grid methods for linear problems

Let us start with the linear parabolic equation with constant coefficients

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}\,. \tag{6.2.1}$$

A more general equation (describing heat conduction or mass diffusion)

$$\frac{\partial u}{\partial \tau} = \sigma \frac{\partial^2 u}{\partial x^2} \tag{6.2.2}$$

can be converted to (6.2.1) by the substitution $t = \sigma\tau$.

The solution of equation (6.2.1) is often searched for on a rectangle $D = [0,1] \times [0,T]$ shown in Fig. 6.1.
The solution $u(x,t)$ must satisfy the initial condition (the function $\varphi(x)$ is given)

$$u(x,0) = \varphi(x)\,, \qquad 0 < x < 1\,, \tag{6.2.3}$$

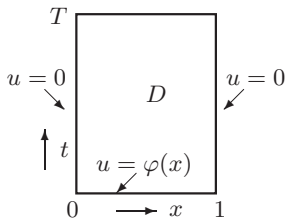


Figure 6.1: The rectangle D where the solution of the parabolic equation (6.2.1) is defined.

and a boundary condition, e.g.

$$u(0,t) = u(1,t) = 0\,. \tag{6.2.4}$$

Other problems may contain other boundary conditions, e.g.

$$x = 0: \qquad \frac{\partial u(0,t)}{\partial x} = 0\,, \tag{6.2.5}$$

$$x = 1: \qquad u(1,t) = 1 \tag{6.2.6}$$

or other.

6.2.1.1 Simple explicit formula

The most common approach to equation (6.2.1) is the difference method also called the grid method. There is a wide range of difference methods, let us start with the simplest one. Let us divide the interval $[0,1]$ in x into n subintervals by equidistant grid points

$$x_0 = 0,\ x_1 = h,\ x_2 = 2h,\ \ldots,\ x_{n-1} = 1-h,\ x_n = 1\,,$$

where $h = 1/n$ and $x_i = ih$, $i = 0, 1, \ldots, n$. Similarly the interval $[0,T]$ in t is divided into r equal parts by the grid points

$$t_0 = 0,\ t_1 = k,\ \ldots,\ t_r = T,$$

where the time step is $k = T/r$ and $t_j = jk$, $j = 0, 1, \ldots, r$. The set of nodes - the intersections of the lines $x = ih$, $i = 0, 1, \ldots, n$, and the lines $t = jk$, $j = 0, 1, \ldots, r$, forms a rectangular grid denoted by $D^{(h)}$ (see Fig.6.2). On this grid we can approximate the derivatives of the function u by the difference formulas (see chapter 2.5) for $i = 1, \ldots, n-1$, $j = 0, \ldots, r-1$:

$$\left.\frac{\partial u}{\partial t}\right|_{(x_i,t_j)} = \frac{u_i^{j+1} - u_i^j}{k} + \mathcal{O}(k)\,, \tag{6.2.7}$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{(x_i,t_j)} = \frac{u_{i-1}^j - 2u_i^j + u_{i+1}^j}{h^2} + \mathcal{O}(h^2)\,, \tag{6.2.8}$$

where we denote $u(ih, jk) = u(x_i, t_j) \doteq u_i^j$.

Consider the equation (6.2.1) in one node $(x_i, t_j) \in D^{(h)}$ and the approximation using (6.2.7) and (6.2.8):

$$\frac{u_i^{j+1} - u_i^j}{k} = \frac{u_{i-1}^j - 2u_i^j + u_{i+1}^j}{h^2} + \mathcal{O}(k + h^2)\,. \tag{6.2.9}$$

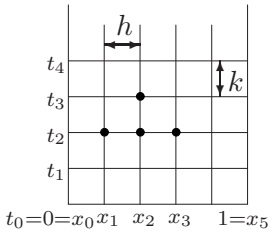


Figure 6.2: The grid $D^{(h)}$, n = 5 and the approximation (6.2.9) for $i = 2, j = 2$

This is illustrated in Fig.6.2. Neglecting $\mathcal{O}(k + h^2) = \mathcal{O}(k) + \mathcal{O}(h^2)$, which is called the approximation error and using the initial condition (6.2.3) and the boundary conditions (6.2.4) we get the following difference problem:

$$u_i^{j+1} = \frac{k}{h^2}\left(u_{i-1}^j + u_{i+1}^j\right) + \left(1 - \frac{2k}{h^2}\right)u_i^j\,, \qquad \begin{matrix} i = 1, 2, \ldots, n-1 \\ j = 0, 1, \ldots, r-1\,, \end{matrix} \tag{6.2.10}$$

$$u_i^0 = \varphi(ih)\,, \qquad i = 1, 2, \ldots, n-1\,, \tag{6.2.11}$$

$$u_0^j = 0\,, \qquad u_n^j = 0\,, \qquad j = 0, 1, \ldots, r\,. \tag{6.2.12}$$

If $u(x_i, t_j)$ is the solution of (6.2.1) with the initial condition (6.2.3) and the boundary condition (6.2.4), then the error of the solution computed by (6.2.10), (6.2.11) and (6.2.12) is

$$\varepsilon_i^j = u(x_i, t_j) - u_i^j\,. \tag{6.2.13}$$

Similarly as for ordinary differential equations (ODE) we require that making the grid finer, i.e. $h \to 0$, $k \to 0$, results in $\varepsilon_i^j \to 0$ in $D^{(h)}$. If this is the case we say that the solution of (6.2.10), (6.2.11) and (6.2.12) converges to the exact solution of (6.2.1), (6.2.3) and (6.2.4). It is obvious that if the numerical solution does not converge to the exact solution then the difference method is useless. The difference approximation (6.2.10) is called the explicit three point difference scheme. This name tells that the value u_i^{j+1} is computed explicitly from the values $u_{i-1}^j, u_i^j, u_{i+1}^j$. The relations (6.2.10), (6.2.11) and (6.2.12) are iterated. The vector $\boldsymbol{u}^j = (u_0^j, u_1^j, \ldots, u_n^j)$ is called the j-th profile. In (6.2.10) the j-th profile is called the old (the known) profile, and the $j + 1$-st profile is called the new profile. To sum up, the new profile is computed point-wise from the old profile.

6.2.1.2 Stability of the difference scheme

We denote by u_i^j the exact solution of the difference problem (6.2.10), (6.2.11) and (6.2.12) and we denote by $\widetilde{u}_i^j$ the numerically computed solution. These differ due to round-off errors introduced in each arithmetical operation done on a digital computer. We want this round-off error not to grow too much in the course of computation. We want the errors

$$\varrho_i^j = u_i^j - \widetilde{u}_i^j \tag{6.2.14}$$

to go to zero or at least to stay bounded for increasing j. This requirement presents the stability condition of the difference scheme. The total error of the numerical solution can be estimated by

$$|\varepsilon_i^j| + |\varrho_i^j|\,, \tag{6.2.15}$$

where $|\varrho_i^j|$ is small and negligible compared to the error of the method $|\varepsilon_i^j|$ for stable schemes. Unstable schemes are useless for practical computation because we can never compute with infinite number of decimal digits.

Let us explain the problem of stability for the scheme (6.2.10) in more detail. It is easy to rewrite (6.2.10), (6.2.11) and (6.2.12) using profiles as

$$\begin{aligned} \boldsymbol{u}^{j+1} &= \mathbf{A}_1 \boldsymbol{u}^j \,, \\ \boldsymbol{u}^0 &= \Big[0, \varphi(h), \varphi(2h), \ldots, \varphi((n-1)h), 0\Big]^T \,, \end{aligned} \tag{6.2.16}$$

where the matrix $\mathbf{A}_1$ is three-diagonal

$$\mathbf{A}_1 = \begin{pmatrix} 0 & 0 & & & & & & \\ \alpha & (1-2\alpha) & \alpha & & & & & \\ & \alpha & (1-2\alpha) & \alpha & & 0 & & \\ & & \ddots & \ddots & \ddots & & & \\ & 0 & & \alpha & (1-2\alpha) & \alpha & & \\ & & & & \alpha & (1-2\alpha) & \alpha & \\ & & & & & 0 & 0 & \end{pmatrix} , \tag{6.2.17}$$

where

$$\alpha = \frac{k}{h^2} \,. \tag{6.2.18}$$

After denoting $\bar{\boldsymbol{u}}^j = (u_1^j, u_2^j, \ldots, u_{n-1}^j)$, and using $u_0^j = u_n^j = 0$, we can rewrite (6.2.16) as

$$\bar{\boldsymbol{u}}^{j+1} = \mathbf{A}\bar{\boldsymbol{u}}^j \,, \qquad \bar{\boldsymbol{u}}^0 = \Big[\varphi(h), \varphi(2h), \ldots, \varphi((n-1)h)\Big]^T \,, \tag{6.2.19}$$

where the matrix $\mathbf{A}$ is of type $(n-1) \times (n-1)$:

$$\mathbf{A} = \begin{pmatrix} (1-2\alpha) & \alpha & & & & \\ \alpha & (1-2\alpha) & \alpha & & 0 & \\ & \ddots & \ddots & \ddots & & \\ 0 & & \alpha & (1-2\alpha) & \alpha & \\ & & & \alpha & (1-2\alpha) & \end{pmatrix} . \tag{6.2.20}$$

Consider now a small deviation of the initial condition (introduced by the round-off error) $\bar{\boldsymbol{\varrho}}^0$:

$$\bar{\boldsymbol{\varrho}}^0 = \bar{\boldsymbol{u}}^0 - \bar{\boldsymbol{u}}'^0 \,. \tag{6.2.21}$$

Here the prime does not mean derivative, it just denotes another profile. Equation (6.2.19) with the initial condition $\bar{\boldsymbol{u}}'^0$ becomes

$$\bar{\boldsymbol{u}}'^{j+1} = \mathbf{A}\bar{\boldsymbol{u}}'^j \,, \quad \bar{\boldsymbol{u}}'^0 = \bar{\boldsymbol{u}}^0 - \bar{\boldsymbol{\varrho}}^0 \,. \tag{6.2.22}$$

The error $\bar{\boldsymbol{\varrho}}^j = \bar{\boldsymbol{u}}^j - \bar{\boldsymbol{u}}'^j$ evolves as

$$\bar{\boldsymbol{\varrho}}^{j+1} = \mathbf{A}\bar{\boldsymbol{\varrho}}^j \,, \tag{6.2.23}$$

giving

$$\bar{\boldsymbol{\varrho}}^j = \mathbf{A}^j \bar{\boldsymbol{\varrho}}^0 \,. \tag{6.2.24}$$

The norm of the effect of the initial deviation $\bar{\boldsymbol{\varrho}}^0$ can be estimated by

$$\|\bar{\boldsymbol{\varrho}}^j\| \le \|\mathbf{A}\|^j \|\bar{\boldsymbol{\varrho}}^0\| \,, \tag{6.2.25}$$

where the norms can be defined (see 1.2.1)

$$\|\boldsymbol{v}\| = \max_i |v_i| \,, \tag{6.2.26}$$

$$\|\mathbf{A}\| = \max_i \sum_{s=1}^{n-1} |a_{is}| \,. \tag{6.2.27}$$

The estimate (6.2.25) gives:
If

$$\|\mathbf{A}\| \leq 1 \,, \tag{6.2.28}$$

then the deviation $\bar{\boldsymbol{\varrho}}^0$ of initial condition does not grow in the course of computation.

Similarly, a deviation $\bar{\boldsymbol{\varrho}}$ in the j-th profile (instead of in the first one) can be treated by considering this j-th profile as the initial condition and the conclusions are the same. In a real computation round-off errors appear in each profile. Thanks to the linearity of (6.2.19) the total error stays bounded if (6.2.28) remains valid.

It is easy to see that if the elements in the main diagonal of the matrix $\mathbf{A}$ are non-negative i.e. if

$$\alpha \leq \frac{1}{2} \,, \tag{6.2.29}$$

then due to (6.2.27) we have $||\mathbf{A}|| = 1$. Thus (6.2.29) is a sufficient condition for the stability of method (6.2.10).

Let us see whether this condition is also necessary. The necessary condition requires that for the least norm of the matrix $\mathbf{A}$ the non-equality (6.2.28) holds. As for any matrix norm it holds $\varrho(\mathbf{A}) = \max |\lambda_i| \leq \|\mathbf{A}\|$, where λ_i are the eigenvalues of the matrix $\mathbf{A}$, the necessary and sufficient condition is

$$|\lambda_i| \leq 1 \quad i = 1, 2, \ldots, n-1 \,. \tag{6.2.30}$$

The matrix (6.2.20) has eigenvalues

$$\lambda_i = 1 - 4\alpha \sin^2 \frac{i\pi}{2n} \,, \quad i = 1, \ldots, n-1 \,,$$

then the condition (6.2.30) is equivalent to the condition (6.2.29).

If the original equation (6.2.1) has non-constant coefficients (as functions of x) then the rows of the matrix $\mathbf{A}$ differ. Then the eigenvalues cannot be expressed analytically, they must be found numerically (see chapter 1), which is expensive for large n. Then it is better to use the sufficient stability condition (6.2.28) where $\|\mathbf{A}\|$ is defined according to (6.2.27).

Sometimes the stability is estimated by the Fourier (von Neumann) method. We describe this method briefly for the explicit differential scheme (6.2.10). This method ignores boundary conditions which is no problem in our case, since the boundary conditions specify zero values of u at the boundaries. In cases where the boundary conditions influence the stability this method has a limited validity. On the other hand the method using the spectral radius of the matrix is still valid, although sometimes difficult to apply.

Assume that the solution of the differential equations can be written as $u_i^j = z_j y_i$, and let us choose one harmonics $e^{\mathrm{i}\beta x}$ from its Fourier representation. Here i is the imaginary unit; to avoid confusion we denote the imaginary unit by i written in ordinary font while we denote the index by i written in mathematical font. The solution of the difference equation is assumed in the form $e^{\omega t} e^{\mathrm{i}\beta x}$ and we want to find conditions for the expression in t not to grow (ω may be complex).

We put

$$u_i^j = e^{\omega jk} e^{\mathrm{i}\beta ih} \tag{6.2.31}$$

into (6.2.10), $\alpha = k/h^2$:

$$e^{\omega(j+1)k} e^{\mathrm{i}\beta ih} = (1-2\alpha) e^{\omega jk} e^{\mathrm{i}\beta ih} + \alpha (e^{\omega jk} e^{\mathrm{i}\beta(i-1)h} + e^{\omega jk} e^{\mathrm{i}\beta(i+1)h}) .$$

After simplification we get

$$e^{\omega k} = 1 - 4\alpha \sin^2(\frac{1}{2}\beta h) ,$$

and the condition $|e^{\omega k}| \le 1$ gives $\alpha \le \frac{1}{2}$.

Table 6.2: Difference scheme (6.2.10), error propagation for $\alpha = \frac{1}{2}$

$u_i^{j+1} = \frac{1}{2}(u_{i-1}^j + u_{i+1}^j)$									
$j=4$	$\varepsilon/16$	0	$\varepsilon/4$	0	$3\varepsilon/8$	0	$\varepsilon/4$	0	$\varepsilon/16$
$j=3$	0	$\varepsilon/8$	0	$3\varepsilon/8$	0	$3\varepsilon/8$	0	$\varepsilon/8$	0
$j=2$	0	0	$\varepsilon/4$	0	$\varepsilon/2$	0	$\varepsilon/4$	0	0
$j=1$	0	0	0	$\varepsilon/2$	0	$\varepsilon/2$	0	0	0
$j=0$	0	0	0	0	ε	0	0	0	0

The error propagation is illustrated in Tables 6.2 and 6.3. The initial error in a single node is denoted by ε. The first case is for $\alpha = \frac{1}{2}$ and the deviation is damped. In the other case $\alpha = 10$ and the error grows quickly.

Table 6.3: Difference scheme (6.2.10), error propagation for $\alpha = 10$,

$u_i^{j+1} = -19u_i^j + 10(u_{i-1}^j + u_{i+1}^j)$							
$j=3$	1000ε	-5700ε	13830ε	-18259ε	13830ε	-5700ε	1000ε
$j=2$	0	100ε	-380ε	561ε	-380ε	100ε	0
$j=1$	0	0	10ε	-19ε	10ε	0	0
$j=0$	0	0	0	ε	0	0	0

Note that for the stability of the difference scheme it is necessary that the original differential equations are stable in a certain sense, i.e. a small change in the initial condition results in a small deviation in the exact solution. To show an example where this is not the case, consider the diffusion equation in backward time

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

which we get from (6.2.1) by changing t to $-t$. Now the method (6.2.10) is unstable for any $\alpha > 0$ and a similar result holds for further methods.

As an illustration we give an example of a stable and of an unstable scheme for equation (6.2.1) with boundary conditions (6.2.4) and with the initial condition (6.2.3) where

$$\varphi(x) = \begin{cases} 2x & \text{for } 0 \le x \le \frac{1}{2} \\ 2(1-x) & \text{for } \frac{1}{2} \le x \le 1. \end{cases} \tag{6.2.32}$$

Analytic solution can be found in the form

Table 6.4: Exact solution of (6.2.1),(6.2.3),(6.2.4) and (6.2.32)

t	x=0.3	x = 0.5	x=0.7
0.005	0.5966	0.8404	0.5966
0.01	0.5799	0.7743	0.5799
0.02	0.5334	0.6809	0.5334
0.10	0.2444	0.3021	0.2444

Table 6.5: Solution of (6.2.1),(6.2.3), (6.2.4) and (6.2.32) by explicit method for $h = 0.1$

			$x = 0.3$	$x = 0.5$	$x = 0.7$
$\alpha = 0.1$	$t = 0.01$	$(j = 10)$	0.5822	0.7867	0.5822
$k = 0.001$	$t = 0.02$	$(j = 20)$	0.5373	0.6891	0.5373
$\alpha = 0.5$	$t = 0.01$	$(j = 2)$	0.6000	0.8000	0.6000
$k = 0.005$	$t = 0.02$	$(j = 4)$	0.5500	0.7000	0.5500
	$t = 0.1$	$(j = 20)$	0.2484	0.3071	0.2484

$$u = \frac{8}{\pi^2} \sum_{n=1}^{\infty} \frac{1}{n^2} \left(\sin \frac{n\pi}{2} \right) \left(\sin n\pi x \right) e^{(-n^2\pi^2 t)} \tag{6.2.33}$$

and the values of this solution are given in Table 6.4. We use the difference scheme (6.2.10) for $h = 0.1$ and α equal to 0.1 and 0.5. The results are summarized in Table 6.5. Compare the achieved accuracy. Note that for $x = 0.5$ the agreement is worse because the initial condition (6.2.32) has at this point non-continuous derivative. The solution is symmetric in x around $x = 0.5$.

Figs. 6.3 and 6.4 show the agreement of numerical ($h = 0.1$) and of the analytic solution for $\alpha < 0.5$ and for $\alpha > 0.5$, i.e. for stable and for unstable scheme.

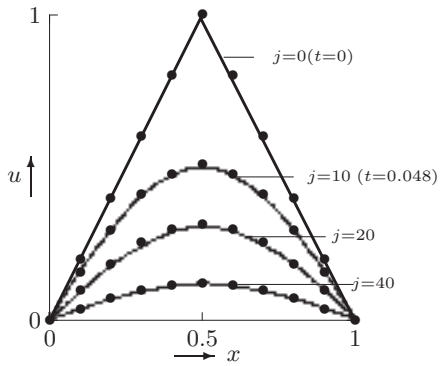


Figure 6.3: Numerical (•) and exact (——) solution for $\alpha = 0.48$, $h = 0.1$

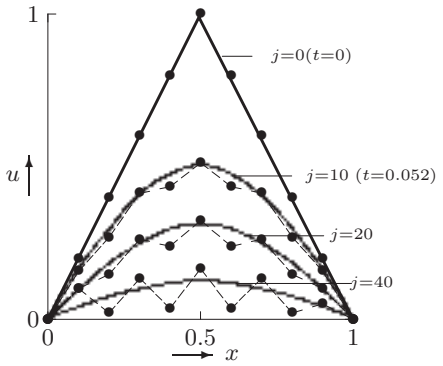


Figure 6.4: Numerical (− • −) and exact (——) solution for $\alpha = 0.52$, $h = 0.1$

6.2.1.3 Simple implicit formula

Let us discuss a more complicated form of the difference formula with a parameter w

$$\frac{u_i^{j+1}-u_i^j}{k}=w\frac{u_{i-1}^{j+1}-2u_i^{j+1}+u_{i+1}^{j+1}}{h^2}+(1-w)\frac{u_{i-1}^{j}-2u_i^{j}+u_{i+1}^{j}}{h^2}. \tag{6.2.34}$$

It is easy to see that for $w=0$ this equation simplifies to (6.2.9) and this special case represents the above discussed simple explicit scheme. For $w=1$ we have the opposite case - a simple implicit difference scheme

$$-\alpha u_{i-1}^{j+1}+(1+2\alpha)u_i^{j+1}-\alpha u_{i+1}^{j+1}=u_i^j\,. \tag{6.2.35}$$

For $w=\frac{1}{2}$ we get an "averaged" scheme called Crank-Nicolson:

$$-\frac{\alpha}{2}u_{i-1}^{j+1}+(1+\alpha)u_i^{j+1}-\frac{\alpha}{2}u_{i+1}^{j+1}=\frac{\alpha}{2}u_{i-1}^{j}+(1-\alpha)u_i^{j}+\frac{\alpha}{2}u_{i+1}^{j}. \tag{6.2.36}$$

Biographical note: Phyllis Nicolson (21 September 1917 - 6 October 1968) was a British mathematician most known for her work on the Crank-Nicolson scheme together with John Crank. John Crank (6 February 1916 - 3 October 2006) was a British mathematical physicist, best known for his work on the numerical solution of partial differential equations.

Similarly as for the explicit scheme it can be shown that approximation error is $\mathcal{O}(k+h^2)$ for (6.2.35) thus method (6.2.35) is of similar accuracy as the explicit formula. For the Crank-Nicolson scheme (6.2.36) it can be shown that the error is $\mathcal{O}(k^2+h^2)$, this method is more precise than the methods (6.2.10) and (6.2.35). This can be explained by the fact that the time derivative is approximated in the point $(t_{j+1}+t_j)/2$ corresponding to a central three-point difference formula (see Tab. 2.1) with the error $\mathcal{O}((\frac{k}{2})^2)$. The Crank-Nicolson scheme (6.2.36) is stable for any $\alpha=k/h^2$. The formula (6.2.34) is stable for any α if $w\in[\frac{1}{2},1]$. If $w<\frac{1}{2}$ then this method is stable if

$$\alpha\le\frac{1}{2(1-2w)}\,. \tag{6.2.37}$$

For $w=0$ this gives again condition (6.2.29).

Unlike the explicit method, where each value of the new profile $\boldsymbol{u}^{j+1}$ was computed explicitly one after another from the old profile $\boldsymbol{u}^j$, the equation (6.2.34) for $w\neq 0$ represents a system of linear equations for unknowns $u_i^{j+1}, i=1,2,\ldots,n-1$, which must be solved simultaneously.

The matrix of this system is three-diagonal for a general w (for $w=0$ the matrix is diagonal and the method is explicit). The system may be solved by factorization (see chapter 1.2.3) when computing a new profile from the old one. Let us compare the Crank-Nicolson method with the explicit method using the equations (6.2.1), (6.2.3), (6.2.4) and (6.2.32) for $h=0.1$. The results are given in Table 6.6. It is easy to see that the error of the results of the explicit method for $\alpha=0.1$ are similar to those of the Crank-Nicolson method with the step $k=0.01$ (where $\alpha=1$). The explicit method requires to compute ten times more profiles, although the computation was easier because it was not necessary to solve a system of linear equations with a three-diagonal matrix. When we compare the number of arithmetic operations then the Crank-Nicolson method is more efficient.

	Explicit method (6.2.10)		Crank-Nicolson method	Analytic solution (6.2.33)
	$\alpha = \frac{1}{10}$ $k = 0.001$	$\alpha = \frac{1}{2}$ $k = 0.005$	$k = 0.01$	
$t = 0.01$	0.7867	0.8000	0.7691	0.7743
$t = 0.02$	0.6891	0.7000	0.6921	0.6809
$t = 0.10$	0.3056	0.3071	0.3069	0.3021

Table 6.6: Comparison of the explicit and the Crank-Nicolson methods. Values in the point $x = 0.5$ are shown ($h = 0.1$)

6.2.1.4 Multi-step methods

So far, we have considered two-profile methods that contain $\boldsymbol{u}^j$ and $\boldsymbol{u}^{j+1}$ only. We have noted that the discretization in t has the greatest contribution to the error, namely $\mathcal{O}(k)$, or $\mathcal{O}(k^2)$ in special methods. This means we must use a small time step k and this requires a long computation time. Another possibility is (similarly to Adams formulas, see chapter 4.4), to approximate the derivative $\frac{\partial u}{\partial t}$ using more than two points. To start such a computation we must know more than just one profile (given by the initial condition). To prepare these profiles another method must be used. One disadvantage of multi-step methods is that it is not easy to adapt the step size k according to how complicated the solution is. Another disadvantage, namely the need of more computer memory to store extra profiles becomes less important with modern hardware. One important advantage of multi-step methods is that we can use a greater step size k because the approximation of $\frac{\partial u}{\partial t}$ is more precise. We show a few multi-step methods for the equation (6.2.1), using the approximation from table 2.1 and 2.2.

A non-central approximation of $\frac{\partial u}{\partial t}$ gives a three-profile implicit formula

$$\frac{3u_i^{j+1} - 4u_i^j + u_i^{j-1}}{2k} = \frac{u_{i-1}^{j+1} - 2u_i^{j+1} + u_{i+1}^{j+1}}{h^2}\,. \tag{6.2.38}$$

This can be rewritten to

$$-2\alpha u_{i-1}^{j+1} + (3 + 4\alpha)u_i^{j+1} - 2\alpha u_{i+1}^{j+1} = 4u_i^j - u_i^{j-1}\,. \tag{6.2.39}$$

Similarly a four-profile implicit formula is

$$-6\alpha u_{i-1}^{j+1} + (11 + 12\alpha)u_i^{j+1} - 6\alpha u_{i+1}^{j+1} = 18u_i^j - 9u_i^{j-1} + 2u_i^{j-2} \tag{6.2.40}$$

and finally a five-profile implicit formula is

$$-12\alpha u_{i-1}^{j+1} + (25 + 24\alpha)u_i^{j+1} - 12\alpha u_{i+1}^{j+1} = 48u_i^j - 36u_i^{j-1} + 16u_i^{j-2} - 3u_i^{j-3}. \tag{6.2.41}$$

Formulas (6.2.39), (6.2.40) and (6.2.41) have the error $\mathcal{O}(k^2+h^2)$, $\mathcal{O}(k^3+h^2)$ and $O(k^4+h^2)$ resp. From the computational point of view these formulas are not much more difficult than a simple implicit formula (6.2.35); the right-hand-side of the system of linear algebraic equations with a three-diagonal matrix contain a few more terms. To start we must prepare three initial profiles (besides the initial condition) using another method with a sufficiently small error.

There exist another multi-step formulas where the approximation of $\frac{\partial^2 u}{\partial x^2}$ is computed from more profiles with appropriate weights with total sum being one. On the other hand, explicit multi-step methods are seldom used, because the stability condition requires a small step size in t, so that the high accuracy of the approximation in t cannot be used (by taking a large step size).

6.2.1.5 Boundary conditions

We have considered boundary conditions of the first kind, i.e. boundary conditions specifying the value of the solution, e.g. for equation (6.2.1) the boundary condition was (6.2.4). Often the boundary conditions specify the derivative of the unknown function (for example the boundary between a heat conducting medium and an insulator is described by $\frac{\partial u}{\partial n} = 0$ where n means the normal i.e. perpendicular direction). This type of boundary condition is called the boundary condition of the second kind. The most often case, however, is a linear combination of the function value and its derivative at the boundary. i.e. $C_1 u + C_2 \frac{\partial u}{\partial n} = C_3$. This type of boundary condition is called the boundary condition of the third kind. Nonlinear boundary condition are discussed below.

Consider a general linear boundary condition

$$C_1 u + C_2 \frac{\partial u}{\partial x} = C_3 \tag{6.2.42}$$

for the equation (6.2.1) in $x = 0$. Assume $C_2 \neq 0$, i.e. (6.2.42) is not a condition of the first kind. The simplest approximation of (6.2.42) is to replace the derivative $\frac{\partial u}{\partial x}$ by a suitable difference formula (see chapter 5, boundary value problem for ordinary differential equation). Replacing

$$\left.\frac{\partial u}{\partial x}\right|_{\substack{x=0 \\ t=(j+1)k}} = \frac{u_1^{j+1} - u_0^{j+1}}{h} + \mathcal{O}(h)\,, \tag{6.2.43}$$

and putting into (6.2.42) we get a linear equation for u_0^{j+1} and u_1^{j+1} (upper indexes can be chosen arbitrarily because (6.2.42) holds for all t)

$$\left(C_1 - \frac{C_2}{h}\right) u_0^{j+1} + \frac{C_2}{h} u_1^{j+1} = C_3\,. \tag{6.2.44}$$

Using (6.2.44) for the explicit formula (6.2.10) is simple: u_0^{j+1} is evaluated by (6.2.44) based on u_1^{j+1} (computed from $u_0^j,\ u_1^j,\ u_2^j$). Put together we get

$$u_0^{j+1} = \frac{C_3 h}{C_1 h - C_2} - \frac{C_2}{C_1 h - C_2} u_1^{j+1} = \delta + \gamma_0 u_0^j + \gamma_1 u_1^j + \gamma_2 u_2^j\,, \tag{6.2.45}$$

where

$$\delta = \frac{C_3 h}{C_1 h - C_2}\,, \quad \gamma_0 = \gamma_2 = -\frac{\alpha C_2}{C_1 h - C_2}\,, \quad \gamma_1 = -(1 - 2\alpha)\frac{C_2}{C_1 h - C_2}\,.$$

The first row of the "transformation" matrix $\mathbf{A}_1$ (see (6.2.17)) changes to

$$(\gamma_0, \gamma_1, \gamma_2, 0, \ldots).$$

It is easy to see that

$$|\gamma_0| + |\gamma_1| + |\gamma_2| = \left|\frac{C_2}{C_1 h - C_2}\right|$$

for $\alpha = k/h^2 \leq \frac{1}{2}$ (which must be satisfied for stability reasons). From $h = \sqrt{k/\alpha}$ it follows that for constant α we have $|\gamma_0| + |\gamma_1| + |\gamma_2| = 1 + \mathcal{O}(\sqrt{k})$, which is a sufficient stability condition. Thus the method (6.2.10) with the boundary condition (6.2.44) is stable for $\alpha \leq \frac{1}{2}$. This is a non-trivial result. Replacement of boundary condition can change the stability. When investigating stability it is always necessary to consider the replacement of boundary conditions as well.

The replacement (6.2.43) has one big disadvantage both for explicit and for implicit scheme. The error is by one order worse than the error of the equation, thus it is better to use a more precise replacement for $\frac{\partial u}{\partial x}$. There are two possibilities:

1. To use a non-central three-point difference

$$\left.\frac{\partial u}{\partial x}\right|_{\substack{x=0\\ t=(j+1)k}} = \frac{-3u_0^{j+1} + 4u_1^{j+1} - u_2^{j+1}}{2h} + \mathcal{O}(h^2)\,, \tag{6.2.46}$$

   This is no complication for explicit formula. For the implicit formula the resulting system must be converted to a three-diagonal one.

2. To use a central three-point difference

$$\left.\frac{\partial u}{\partial x}\right|_{\substack{x=0\\ t=(j+1)k}} = \frac{u_1^{j+1} - u_{-1}^{j+1}}{2h} + \mathcal{O}(h^2) \tag{6.2.47}$$

   by introducing a fictitious node with index -1. This increases the number of unknowns and we must find one equation for this new unknown. This can be done by approximating equation (6.2.1) by the implicit formula (6.2.35) for $i = 0$. The unknown u_{-1}^{j+1} can be expressed from this equation as a function of u_0^j, u_0^{j+1} and u_1^{j+1} and we put the result into the approximation (6.2.47). For the implicit method (6.2.35) we get again a system of linear equations with a three-diagonal matrix. This second approach is better because the replacement (6.2.47) has a smaller error than the replacement (6.2.46), although they are of the same order (see chapter 2).

For the implicit or the explicit method the replacement of the boundary condition is easy. For more complex methods it is usually not obvious how to approximate the boundary condition to get the highest accuracy of the resulting replacement. The implicit replacement of the boundary condition usually gives good results.

In some problems the boundary conditions depend on time, e.g.

$$u(0,t) = \sin \omega t$$

is periodic in time t. This type of boundary conditions presents no big complication. We can use the same methods as for time independent boundary conditions. The resulting formula contains time dependent term.

Sometimes we have a linear parabolic equation with a nonlinear boundary condition, e.g. equation (6.2.1) with boundary conditions

$$\psi_0\left(u(0,t), \frac{\partial u(0,t)}{\partial x}, t\right) = 0\,, \qquad \psi_1\left(u(1,t), \frac{\partial u(1,t)}{\partial x}, t\right) = 0 \tag{6.2.48}$$

instead of (6.2.4).

This is the case of heat conduction with radiation, or diffusion with surface chemical reaction etc. Let us illustrate this by an example. Consider heat conduction in an insulated bar described by equation (6.2.1). One end of the bar is kept at a constant temperature and the other end of the bar receives heat by radiation from a source of constant temperature and looses heat by its own radiation. The boundary conditions are

$$x = 0: \quad u = U_0\,, \qquad x = 1: \quad s(1-u^4) - \frac{\partial u}{\partial x} = 0\,, \tag{6.2.49}$$

and the initial condition is: for $t = 0$ and $x \in [0,1]$ $u = U_0$. Here the temperature is related to the thermodynamic temperature of the radiation source. The dimensionless parameter s contains the fourth power of the source temperature, the Stephan-Boltzmann constant, heat conductivity, the length of the bar and the configuration factor. The partial differential equation can be discretized by the Crank-Nicolson method and the boundary condition (6.2.49) can be replaced by the implicit method by introducing a fictitious profile $n+1$:

$$s\Big(1-(u_n^{j+1})^4\Big) - \frac{u_{n+1}^{j+1} - u_{n-1}^{j+1}}{2h} = 0\,. \tag{6.2.50}$$

We have again a system of n equations for n unknowns $u_1^{j+1}, \ldots, u_n^{j+1}$ with a three-diagonal appearance. The first $n-1$ equations are linear and the last equation is nonlinear in the form

$$a u_{n-1}^{j+1} + b u_n^{j+1} = c - d(u_n^{j+1})^4\,. \tag{6.2.51}$$

The last equation comes from putting (6.2.50) into the Crank-Nicolson replacement for $i = n$, the constant c contains u_{n-1}^j, u_n^j. The right-hand-side of the last "linear" equation of the system with a three-diagonal matrix depends on the "parameter" u_n^{j+1}.

The first phase of the factorization and vanishing the bottom diagonal gives the last equation in the form

$$b' u_n^{j+1} = c' - d'(u_n^{j+1})^4\,. \tag{6.2.52}$$

This is an algebraic equation for one unknown u_n^{j+1}. This equation can be solved by some method in chapter 3.1 (we have a good initial approximation u_n^j). Only after solving the equation (6.2.52) the second phase of the factorization is done.

Exercise: How can we solve the same PDE with the non-linear boundary condition (6.2.49) on both ends of the bar?

6.2.1.6 Methods with higher accuracy

This section is devoted to algorithms that increase the order of the difference approximation and that allow higher step sizes h and k for the same accuracy. This can be achieved by two ways. The first way is to tune certain parameters in the difference formula so that the order is higher. This way has a big disadvantage that the difference formula is prepared to fit the given PDE and cannot be used for other equations. We do not discuss this type of methods here. The other way uses more nodes for the approximations of derivatives.

Exercise: Find the minimal number of nodes to approximate
$\frac{\partial^2 u}{\partial x^2}$, $\frac{\partial u}{\partial x}$, $\frac{\partial u}{\partial t}$, $\frac{\partial^2 u}{\partial x \partial t}$, $(\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2})$, etc.

To avoid problems with having more unknowns than equations we use non-symmetric difference formulas near boundaries. This is illustrated in Fig. 6.5 where the second derivative in the nodes 2, 3, 4 is approximated by a symmetric formula with 5 nodes and in the nodes 1, 5 by a non-symmetric formula again with 5 nodes. We consider a difference

approximation of equation (6.2.1) where the derivative $\frac{\partial^2 u}{\partial x^2}$ is approximated using 5 points. The case with more nodes is similar. The explicit approximation can be

$$\frac{u_i^{j+1} - u_i^j}{k} = \frac{-u_{i-2}^j + 16u_{i-1}^j - 30u_i^j + 16u_{i+1}^j - u_{i+2}^j}{12h^2} + \mathcal{O}(k + h^4)\,. \tag{6.2.53}$$

A necessary and sufficient stability condition is now more restrictive in the time step k, namely $\alpha \leq \frac{3}{8}$. On the other hand the spatial step size h can be larger so the restriction in k is not necessarily worse than in the classical explicit method (6.2.10).

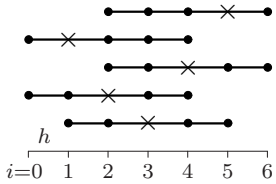


Figure 6.5: Non-symmetric approximations

The reader is invited to write the implicit formula of type (6.2.53), similarly as the non-symmetric approximation for one node near the boundary (use chapter 2.5).

Formula (6.2.53) and similar ones have one disadvantage - the approximation in the t direction is much worse than in the x direction. One way to remove this disadvantage is to use the Crank-Nicolson approximation, namely

$$\begin{aligned}\frac{u_i^{j+1} - u_i^j}{k} \;=\; & \frac{1}{2}\left(\frac{-u_{i-2}^{j+1} + 16u_{i-1}^{j+1} - 30u_i^{j+1} + 16u_{i+1}^{j+1} - u_{i+2}^{j+1}}{12h^2} + \right. \\ & \left. + \frac{-u_{i-2}^{j} + 16u_{i-1}^{j} - 30u_i^{j} + 16u_{i+1}^{j} - u_{i+2}^{j}}{12h^2}\right) + \mathcal{O}(k^2 + h^4)\,.\end{aligned} \tag{6.2.54}$$

The implicit approximation means that we must solve a system of linear equations with a five-diagonal matrix, this can be solved by an algorithm similar to factorization of a three-diagonal matrix.

The other way how to increase the accuracy in the t direction is to use more than two profiles, i.e. to use a multi-step method, see chapter 6.2.1.4.

6.2.2 Grid methods for nonlinear problems

A nonlinear problem can be formulated in general as

$$F\left(t, x, u, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}, \frac{\partial u}{\partial t}\right) = 0\,. \tag{6.2.55}$$

In chemical engineering we usually solve problems linear both in $\frac{\partial u}{\partial t}$ and in $\frac{\partial^2 u}{\partial x^2}$. These problems are called quasi-linear, e.g.

$$\frac{\partial u}{\partial t} = a\Big(t, x, u, \frac{\partial u}{\partial x}\Big)\frac{\partial^2 u}{\partial x^2} + b\Big(t, x, u, \frac{\partial u}{\partial x}\Big)\frac{\partial u}{\partial x} + c\Big(t, x, u, \frac{\partial u}{\partial x}\Big) \tag{6.2.56}$$

(the last two terms could be written as a single term, but b and c are often independent of $\frac{\partial u}{\partial x}$, so this form is more convenient).

Some authors use the term quasi-linear for systems with coefficients that do not depend on first derivatives; the terminology is not uniform. It is appropriate to say that unlike linear equations, there is no general approach to nonlinear parabolic equations. Each nonlinear equation (or a system of them) is usually a unique problem for numerical solution. Thus we discuss algorithms that often work in engineering applications, they are not however reliable recipes for all problems.

6.2.2.1 Simple explicit method

If we replace all spatial derivatives and nonlinear coefficients in the old profile in equation (6.2.56) we get the approximation

$$\begin{aligned}\frac{u_i^{j+1}-u_i^j}{k} &= a\left(t_j, x_i, u_i^j, \frac{u_{i+1}^j - u_{i-1}^j}{2h}\right)\frac{u_{i-1}^j - 2u_i^j + u_{i+1}^j}{h^2} + \\ &+b\left(t_j, x_i, u_i^j, \frac{u_{i+1}^j - u_{i-1}^j}{2h}\right)\frac{u_{i+1}^j - u_{i-1}^j}{2h} + \\ &+c\left(t_j, x_i, u_i^j, \frac{u_{i+1}^j - u_{i-1}^j}{2h}\right), \quad i = 1, 2, ..., n-1\,,\end{aligned} \tag{6.2.57}$$

which is from the computational point of view similar to the explicit method (6.2.10). From the known values of $u_0^j, u_1^j, \ldots, u_n^j$ it is possible to compute the right hand side of the approximation (6.2.57) and then we can get easily u_i^{j+1} for $i = 1, 2, \ldots, n-1$. The problem of approximation of the boundary condition is equivalent to that for linear equation.

Similarly as in the linear case, the steps h and k in the approximation (6.2.57) cannot be chosen arbitrarily because for some combinations of h and k the replacement (6.2.57) is unstable. Unlike the linear case it is not possible to get simple analytic condition of stability. The stability of nonlinear problems must be tested experimentally. This is done by computing a few steps for various values of the step k, the instability can be seen clearly. Also, the condition of stability may vary with time t. For equation (6.2.57) the necessary condition of stability (as the lower order terms have no significant influence on stability) is

$$\frac{k\, a\left(t_j, x_i, u_i^j, \frac{u_{i+1}^j - u_{i-1}^j}{2h}\right)}{h^2} < \frac{1}{2}\,. \tag{6.2.58}$$

In (6.2.58) the boundary conditions of the first kind are considered; the boundary conditions with derivatives may change the condition substantially. The estimate (6.2.58) shows that the acceptable step size k may indeed vary with time t and this must be taken into account.

Next, we use the explicit method (6.2.57) for a problem with a known analytic solution. Consider the partial differential equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{u}{2}\frac{\partial u}{\partial x} + c\,u^2 - e^{-2\pi^2 t}\left(c\sin^2 \pi x + \frac{\pi}{4}\sin 2\pi x\right) \tag{6.2.59}$$

with the boundary condition

$$u(0,t) = u(1,t) = 0 \tag{6.2.60}$$

and the initial condition

$$u(x,0) = \sin \pi x\,. \tag{6.2.61}$$

It is easy to check that the analytic solution (for any c) is

$$u(x,t) = e^{-\pi^2 t}\sin \pi x\,. \tag{6.2.62}$$

Table 6.7 shows results computed by the explicit method (for $c = 1$).

Table 6.7: Results for explicit method (6.2.57) and equation (6.2.59), values $u(0.5;t)$ for various values of h and k

t	$h=0.1$			$h=0.05$		Exact solution
	$k=0.005$	$k=0.002$	$k=0.001$	$k=0.001$	$k=0.0005$	(equation (6.2.62))
0.01	0.9045	0.9059	0.9063	0.9058	0.9060	0.9060
0.05	0.6053	0.6100	0.6115	0.6096	0.6104	0.6105
0.2	0.1341	0.1384	0.1399	0.1381	0.1388	0.1389
0.4	0.0180	0.0192	0.0196	0.0191	0.0193	0.0193

6.2.2.2 Method of simple linearization

The explicit method is easy to use, but it has a strong stability restriction which is here a greater disadvantage than for linear equations, because the evaluation of nonlinear functions is usually expensive. We often split nonlinear terms into two parts: a linear part, considered on the new profile and a nonlinear part (or a remaining part), considered on the old profile. E.g. u^2 can be split into $u^{j+1}u^j$, similarly u^3 can be split into $u^{j+1}(u^j)^2$, or $(\frac{\partial u}{\partial x})^2$ can be split into $(\frac{\partial u}{\partial x})^{j+1}(\frac{\partial u}{\partial x})^j$ etc. Here superscript 2 or 3 means power, while superscript j or $j+1$ denotes discretized time. This trick is called linearization. Thus equation (6.2.56) can be approximated by

$$\begin{aligned}\frac{u_i^{j+1}-u_i^j}{k} &= a\left(t_j,x_i,u_i^j,\frac{u_{i+1}^j-u_{i-1}^j}{2h}\right)\frac{u_{i-1}^{j+1}-2u_i^{j+1}+u_{i+1}^{j+1}}{h^2}+\\ &+b\left(t_j,x_j,u_i^j,\frac{u_{i+1}^j-u_{i-1}^j}{2h}\right)\frac{u_{i+1}^{j+1}-u_{i-1}^{j+1}}{2h}+\\ &+c\left(t_j,x_i,u_i^j,\frac{u_{i+1}^j-u_{i-1}^j}{2h}\right).\end{aligned} \tag{6.2.63}$$

The coefficients a,b,c are evaluated in the old profile j and the derivatives $\frac{\partial^2 u}{\partial x^2}$ and $\frac{\partial u}{\partial x}$ are approximated in the new profile $j+1$. The difference scheme (6.2.63) is actually an implicit scheme and it gives a system of linear equations for unknowns $u_0^{j+1},u_1^{j+1},\ldots,u_n^{j+1}$ (including boundary condition replacement). This is a three-diagonal system and it can be solved by factorization. Approximation (6.2.63) is implicit for spatial derivatives. Alternatively $\frac{\partial^2 u}{\partial x^2}$ and $\frac{\partial u}{\partial x}$ could be approximated by the average of the values in the old and in the new profile similarly to the Crank-Nicolson method. Each equation can usually be linearized by various ways, the experience and intuition is important.

6.2.2.3 Extrapolation techniques

Let us try to replace equation (6.2.56) in pure implicit way, i.e.

$$\begin{aligned}\frac{u_i^{j+1}-u_i^j}{k} &= a_i^{j+1}\frac{u_{i-1}^{j+1}-2u_i^{j+1}+u_{i+1}^{j+1}}{h^2}+b_i^{j+1}\frac{u_{i+1}^{j+1}-u_{i-1}^{j+1}}{2h}+c_i^{j+1},\\ &i=1,2,...,n-1.\end{aligned} \tag{6.2.64}$$

The coefficients a,b,c are functions of the unknowns $\boldsymbol{u}^{j+1}$, e.g.

$$a_i^{j+1}=a\left(t_{j+1},x_i,u_i^{j+1},\frac{u_{i+1}^{j+1}-u_{i-1}^{j+1}}{2h}\right). \tag{6.2.65}$$

System (6.2.64) can be solved as a set of nonlinear equations, which will be discussed later. Here we try to predict the values of $a_i^{j+1}, b_i^{j+1}, c_i^{j+1}$ based on the knowledge of a few last profiles. Assuming $u(x,t), a, b, c$ are sufficiently smooth functions we can extrapolate the values of $a_i^{j+1}, b_i^{j+1}, c_i^{j+1}$ linearly for small time step k from the known profiles j and $(j-1)$ according to

$$a_i^{j+1} \approx 2a_i^j - a_i^{j-1} \tag{6.2.66}$$

(and similarly for b and c). We can extrapolate from more than just two profiles, e.g. quadratic extrapolation gives

$$a_i^{j+1} = a_i^{j-2} - 3a_i^{j-1} + 3a_i^j \,. \tag{6.2.67}$$

Approximation (6.2.64) is implicit, thus the stability restriction is not so severe (if any) as for explicit one. The error introduced by extrapolation is much smaller than the error of linearization as discussed in the previous section. So what is the disadvantage of this approach? It is a multi-step method, meaning the first one or two steps must be computed by another method, e.g. by actual solving the nonlinear equations (6.2.64).

6.2.2.4 Predictor - corrector technique

In the last section we discussed the prediction of the coefficients a, b, c in the profile $(j+1)$. There is another way: to predict the values of $\bar{\boldsymbol{u}}^{j+1}$ using the explicit method (6.2.57), where $u_i^{j+1} = \bar{u}_i^{j+1}$, $i = 1, 2, \ldots, n-1$. This predicted $\bar{\boldsymbol{u}}_i^{j+1}$ can be substituted into the coefficients a, b, c in equation (6.2.64), e.g.

$$\bar{a}_i^{j+1} = a\left(t_{j+1}, x_i, \bar{u}_i^{j+1}, \frac{\bar{u}_{i+1}^{j+1} - \bar{u}_{i-1}^{j+1}}{2h}\right). \tag{6.2.68}$$

Then (6.2.64) becomes

$$\begin{aligned} \frac{u_i^{j+1} - u_i^j}{k} \quad &= \quad \bar{a}_i^{j+1} \frac{u_{i-1}^{j+1} - 2u_i^{j+1} + u_{i+1}^{j+1}}{h^2} + \bar{b}_i^{j+1} \frac{u_{i+1}^{j+1} - u_{i-1}^{j+1}}{2h} + \bar{c}_i^{j+1} \,, \\ &\quad i = 1, 2, ..., n-1 \,, \end{aligned} \tag{6.2.69}$$

which is a system in linear equations (including boundary conditions) with a three diagonal matrix; the solution being similar as in the linear case.

What advantages and disadvantages has this method as compared to extrapolation methods (which can be regarded as a special case of predictor - corrector methods)? It is not necessary to start with a different method i.e. the computation can start with the knowledge of the initial condition alone. Sometimes the memory requirements are weaker. As opposed to the linear extrapolation this prediction is usually better (even though they both are of order $\mathcal{O}(k)$). On the other hand the computation time can grow. Using a large step size k (from the point of view of stability of the explicit method) is no problem because the implicit method (6.2.69) eliminates this influence.

It is clear that when using the Crank-Nicolson method instead of (6.2.69) we must evaluate $\bar{a}_i^{j+1/2}, \bar{b}_i^{j+1/2}, \bar{c}_i^{j+1/2}$, which can be done using an explicit method with the step size $k' = k/2$. When using this predictor - corrector method we can compare $\bar{u}_i^{j+1}$ and u_i^{j+1} (predicted and computed values) in each profile. We want these values to be close. If they differ much we can substitute u_i^{j+1} for $\bar{u}_i^{j+1}$ and repeat the computation according to (6.2.69). This means we repeat the corrector step, similarly as for ordinary differential

equations (see 4.3). It would be too difficult to prove the convergence of this method for general a, b, c and arbitrary boundary conditions. The experience tells us that this approach usually converges for sufficiently small k.

6.2.2.5 Newton's method

Consider the system (6.2.64) including the boundary value replacement as a system of nonlinear equations

$$a_i^{j+1}\frac{u_{i-1}^{j+1}-2u_i^{j+1}+u_{i+1}^{j+1}}{h^2}+b_i^{j+1}\frac{u_{i+1}^{j+1}-u_{i-1}^{j+1}}{2h}+c_i^{j+1}-\frac{u_i^{j+1}}{k}+\frac{u_i^j}{k}=0\,, \tag{6.2.70}$$

thus

$$f_i(u_{i-1}^{j+1},u_i^{j+1},u_{i+1}^{j+1})=0\,,\qquad i=1,2,\ldots,n-1\,, \tag{6.2.71}$$

and possible boundary conditions

$$u_0^{j+1}=u_n^{j+1}=0\,, \tag{6.2.72}$$

that allow to eliminate u_0^{j+1} and u_n^{j+1} from equation (6.2.70). After choosing the initial approximation $u_1^{j+1,0}, u_2^{j+1,0}, \ldots, u_{n-1}^{j+1,0}$, the next approximation can be computed by the iteration

$$\boldsymbol{\Gamma}(\boldsymbol{u}^{j+1,s})\Delta\boldsymbol{u}^{j+1,s}=-\boldsymbol{f}(\boldsymbol{u}^{j+1,s})\,, \tag{6.2.73}$$

$$\boldsymbol{u}^{j+1,s+1}=\boldsymbol{u}^{j+1,s}+\Delta\boldsymbol{u}^{j+1,s}\,, \tag{6.2.74}$$

where

$$\boldsymbol{\Gamma}=\begin{pmatrix}\frac{\partial f_1}{\partial u_1^{j+1}} & \frac{\partial f_1}{\partial u_2^{j+1}} & \cdots & \frac{\partial f_1}{\partial u_{n-1}^{j+1}}\\ \vdots & & & \vdots\\ \frac{\partial f_{n-1}}{\partial u_1^{j+1}} & \frac{\partial f_{n-1}}{\partial u_2^{j+1}} & \cdots & \frac{\partial f_{n-1}}{\partial u_{n-1}^{j+1}}\end{pmatrix},\quad \boldsymbol{u}^{j+1}=\begin{pmatrix}u_1^{j+1}\\ u_2^{j+1}\\ \vdots\\ u_{n-1}^{j+1}\end{pmatrix},\quad \boldsymbol{f}=\begin{pmatrix}f_1\\ f_2\\ \vdots\\ f_{n-1}\end{pmatrix}.$$

From (6.2.71) we can see that the Jacobi matrix $\boldsymbol{\Gamma}$ is three diagonal. The Newton's method converges almost always in a few iterations because we have a very good initial approximation u_i^j, $i=1,2,\ldots,n-1$. The disadvantage is the need to evaluate the Jacobi matrix.

Up to now we considered one nonlinear partial differential equation. In most cases we have a system of partial differential equations and then the Jacobi matrix for the Newton's method is no longer three diagonal, it still has a band structure. We are going to show how appropriate linearization (sometimes called quasi-linearization) can be used to take the advantage of a three diagonal matrix.

Consider a system of two equations

$$\frac{\partial u_m}{\partial t}=\frac{\partial^2 u_m}{\partial x^2}+f_m(u_1,u_2)\,,\quad m=1,\,2\,.$$

Using the Crank-Nicolson method we get for $m=1,\,2$

$$\frac{u_{m,i}^{j+1}-u_{m,i}^j}{k}=\frac{1}{2}\Big(\frac{u_{m,i-1}^{j+1}-2u_{m,i}^{j+1}+u_{m,i+1}^{j+1}}{h^2}+\frac{u_{m,i-1}^{j}-2u_{m,i}^{j}+u_{m,i+1}^{j}}{h^2}\Big)+f_{m,i}^{j+\frac{1}{2}}\,. \tag{6.2.75}$$

If we replace the nonlinear term by the Taylor expansion

$$f_{m,i}^{j+\frac{1}{2}} \doteq f_m(\boldsymbol{u}_i^j) + \frac{\partial f_m(\boldsymbol{u}_i^j)}{\partial u_1}\,\frac{u_{1,i}^{j+1} - u_{1,i}^j}{2} + \frac{\partial f_m(\boldsymbol{u}_i^j)}{\partial u_2}\,\frac{u_{2,i}^{j+1} - u_{2,i}^j}{2}\,, \quad m = 1, 2\,,$$

we get actually the Newton's method (written in a different way) and the Jacobi matrix will have a band structure with five diagonals (with appropriate ordering of the unknowns and the equations). Doing only a partial linearization

$$\begin{aligned} f_{1,i}^{j+\frac{1}{2}} &\doteq f_1(\boldsymbol{u}_i^j) + \frac{\partial f_1(\boldsymbol{u}_i^j)}{\partial u_1}\,\frac{u_{1,i}^{j+1} - u_{1,i}^j}{2} \\ f_{2,i}^{j+\frac{1}{2}} &\doteq f_2(\boldsymbol{u}_i^j) + \frac{\partial f_2(\boldsymbol{u}_i^j)}{\partial u_2}\,\frac{u_{2,i}^{j+1} - u_{2,i}^j}{2}\,, \end{aligned} \tag{6.2.76}$$

the system of equations (6.2.75) splits into two independent subsystems, each one with a three diagonal matrix. The algorithm can be further improved by using $u_{1,i}^{j+1}$ for the computation of $f_{2,i}^{j+1/2}$ and to alternate the order of (6.2.76).

6.2.3 Method of lines

The method of lines is sometimes called the differential difference method. This name reflects the fact that we replace partial derivatives in one direction by difference formulas while we preserve them in the other direction and consider them as ordinary derivatives. We explain the method using a simple quasi-linear equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + R(u) \tag{6.2.77}$$

with boundary conditions of the first kind

$$u(0,t) = u(1,t) = 0\,, \qquad t > 0\,, \tag{6.2.78}$$

and the initial condition

$$u(x,0) = \varphi(x)\,, \qquad x \in (0,1)\,. \tag{6.2.79}$$

We replace the spatial derivative using a difference formula

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x=x_i} \approx \frac{u(x_{i-1},t) - 2u(x_i,t) + u(x_{i+1},t)}{h^2}\,, \quad i = 1, 2, \ldots, n-1\,, \tag{6.2.80}$$

where $x_i = ih,\ i = 0, 1, 2, \ldots, n$. We denote

$$u(x_i,t) \doteq u_i(t)\,. \tag{6.2.81}$$

Along vertical lines (see Fig. 6.6) we get differential equations

$$\frac{\mathrm{d}u_i(t)}{\mathrm{d}t} = \frac{u_{i-1}(t) - 2u_i(t) + u_{i+1}(t)}{h^2} + R\Big(u_i(t)\Big)\,, \quad i = 1, 2, \ldots, n-1\,, \tag{6.2.82}$$

by substituting into equation (6.2.77). To satisfy boundary equations (6.2.78), it is easy to see that it must be

$$u_0(t) = 0\,, \qquad u_n(t) = 0\,. \tag{6.2.83}$$

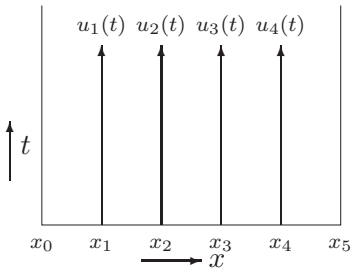


Figure 6.6: Method of lines

Initial condition (6.2.79) gives initial condition for ordinary differential equations (6.2.82):

$$u_i(0) = \varphi(x_i) = \varphi(ih)\,, \qquad i = 1, 2, \ldots, n-1\,. \tag{6.2.84}$$

Method of lines is easy even for more complicated problems. E.g. the equation

$$\frac{\partial u}{\partial t} = F\left(x, t, u, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}\right) \tag{6.2.85}$$

can be transformed into a system of ordinary differential equations (without considering boundary conditions)

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = F\left(x_i, t, u_i, \frac{u_{i+1} - u_{i-1}}{2h}, \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2}\right), \quad i = 1, 2, \ldots, n-1\,. \tag{6.2.86}$$

There is no principal difference between system (6.2.82) and system (6.2.86). The method of lines is a general approach both for linear and for nonlinear parabolic equations in two variables. A system of ordinary differential equations was discussed in chapter 4. Not all numerical methods for ordinary differential equations are appropriate for solution of systems (6.2.82) or (6.2.86), but most of them can be used. The system (6.2.82) has two important properties that must be considered when choosing the integration method:

1. It is a large system. The number of ordinary differential equations may be several hundreds or thousands.

2. It is not necessary to take an extremely precise method for the numerical integration because even a precise solution of this system suffers the error of discretization of the spatial derivative. A method with a similar accuracy to that of the spatial discretization is appropriate.

Having a large number of equations it seems that complicated single step methods (Runge-Kutta methods of a high order) are not good. Using the Euler's method we get the simple explicit formula (6.2.10). The reader is invited to check this. To integrate this system of ordinary differential equations we often use the Runge-Kutta method of order 2 or 3 or a multi step method or a predictor - corrector method. Then the starting profiles must be computed using Runge-Kutta methods.

Using an explicit integration method brings the problem of stability. We cannot use an arbitrarily long integration step for the Runge-Kutta method. The stability condition must be investigated for each combination of PDE, spatial derivative approximation and integration method separately. Thus it is better to use some implicit method, but this requires iteration or to solve a system of liner algebraic equations for linear PDE.

Treatment of boundary conditions for the method of lines is similar to that of difference methods. We can again introduce a fictitious profile or we can use non-symmetric difference formulas for derivatives in the boundary conditions.

The method of lines with a single step integration is a good starting method for multi profile methods.

The number of nodes in the spatial coordinate is given by the desired accuracy. For problems where the solution in different regions of x differs considerably (e.g. for the wave or front solution, where u changes significantly in a very small interval of x) with an equidistant grid we must choose the step size so small to approximate this sharp transition well. Then small changes of u in the rest of the interval are approximated too precisely and the total number of nodes is too high. For such problems methods with adaptive regulation of non-equidistant spatial grid have been developed (see [6]).

6.3 Numerical solution of parabolic equations with three independent variables

As compared to problems solved above, here we have one more spatial coordinate, so we solve parabolic equations in two spatial and one temporal coordinates. The strategies are similar to those discussed above, numerical realization is more difficult, memory requirements are higher and the computation time is usually much longer.

A typical and the simplest linear parabolic equation in three dimensions is the equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\,, \tag{6.3.1}$$

describing non-stationary heat conduction in a plane plate or non-stationary diffusion in a plane. Assume the initial condition

$$u(x,y,0) = \varphi(x,y)\,, \quad x \in [0,1]\,, \quad y \in [0,1] \tag{6.3.2}$$

and the boundary conditions

$$\begin{aligned} &u(x,0,t) = 1\,, \quad u(x,1,t) = 0\,, && x \in [0,1]\,, \quad t > 0\,,\\ &u(0,y,t) = u(1,y,t) = 0\,, && y \in [0,1]\,, \quad t > 0\,. \end{aligned} \tag{6.3.3}$$

This describes warming up a square plate with the initial temperature $\varphi(x,y)$, by keeping three sides at the zero temperature and one side at the unit temperature. In the region $0 \le x,y \le 1, t \ge 0$ we define a grid of nodes $x_i = ih;\ y_j = jh;\ t_m = mk$, where $i,j = 0,1,\ldots,n;\ m = 0,1,\ldots$. This grid is given by the step h in the two spatial coordinates x and y and by the temporal step k. Again we define

$$\alpha = \frac{k}{h^2}\,. \tag{6.3.4}$$

We denote the value of the numerical solution at a grid point

$$u_{i,j}^m \approx u(x_i, y_j, t_m) = u(ih, jh, mk)\,. \tag{6.3.5}$$

To keep the formulas simple we define central difference operators of the second order δ_x^2 and δ_y^2 by

$$\delta_x^2 u_{i,j} = u_{i+1,j} - 2u_{i,j} + u_{i-1,j}\,, \quad \delta_y^2 u_{i,j} = u_{i,j+1} - 2u_{i,j} + u_{i,j-1}\,. \tag{6.3.6}$$

The simple explicit formula then becomes

$$\boldsymbol{u}^{m+1} = (1 + \alpha(\delta_x^2 + \delta_y^2))\boldsymbol{u}^m + \mathcal{O}(k^2 + kh^2)\,, \tag{6.3.7}$$

or in details

$$u_{i,j}^{m+1} = u_{i,j}^m + \alpha(u_{i-1,j}^m - 2u_{i,j}^m + u_{i+1,j}^m + u_{i,j-1}^m - 2u_{i,j}^m + u_{i,j+1}^m)\,. \tag{6.3.8}$$

The order of this method is clearly $\mathcal{O}(k+h^2)$ and each point in the new profile is computed from five points in the old profile. It is possible to derive a similar formula

$$\boldsymbol{u}^{m+1} = (1 + \alpha\delta_x^2)(1 + \alpha\delta_y^2)\boldsymbol{u}^m + \mathcal{O}(k^2 + kh^2)\,, \tag{6.3.9}$$

that uses 9 points in the old profile and that has the same order as formula (6.3.7). The reader is invited to rewrite (6.3.9) in the form similar to (6.3.8).

Equation (6.3.8) can be written by the scheme

$$\begin{array}{ccccc} & & \alpha & & \\ & & | & & \\ \alpha & — & (1-4\alpha) & — & \alpha \\ & & | & & \\ & & \alpha & & \end{array}$$

and similarly equation (6.3.9) by the scheme

$$\begin{array}{ccccccc} \alpha^2 & — & \alpha(1-2\alpha) & — & \alpha^2 \\ | & & | & & | \\ \alpha(1-2\alpha) & — & (1-2\alpha)^2 & — & \alpha(1-2\alpha) \\ | & & | & & | \\ \alpha^2 & — & \alpha(1-2\alpha) & — & \alpha^2 \end{array}$$

Formula (6.3.9) differs from (6.3.8) by including $\alpha^2\delta_x^2\delta_y^2\boldsymbol{u}^m$. These formulas are illustrated in Fig. 6.7. They both are of order $\mathcal{O}(k+h^2)$; the stability condition of the 5 point formula (6.3.8) is

$$\alpha \le \frac{1}{4}\,, \tag{6.3.10}$$

while the 9 point formula (6.3.9) is stable for

$$\alpha \le \frac{1}{2}\,. \tag{6.3.11}$$

If we take $\alpha = \frac{1}{6}$, the order increases to $\mathcal{O}(k^2 + h^4)$ and this formula is appropriate for preparing precise starting profiles for multi profile methods (this is true for equation (6.3.1) only). Strict stability conditions (6.3.10) and (6.3.11) require small temporal step size k resulting in a long computation time which in turn limits the usability of explicit methods (6.3.7) and (6.3.9) for numerical solution of three dimensional problems. For four dimensional problems the stability restrictions are even stronger. On the other hand, a big advantage of explicit methods is their generality and ease of use (evaluation of recurrent formulas).

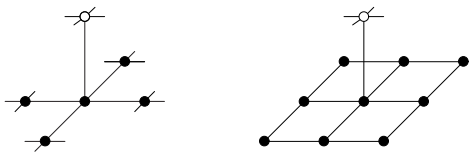


Figure 6.7: Illustration of explicit formulas (6.3.8) and (6.3.9)

Du Fort and Frankel derived a stable explicit method by taking (similarly as for a single spatial coordinate) the unstable Richardson formula

$$u_{i,j}^{m+1} = u_{i,j}^{m-1} + 2\alpha(\delta_x^2 + \delta_y^2)u_{i,j}^m \,. \tag{6.3.12}$$

They replaced $u_{i,j}^m$ by the arithmetic mean $\frac{1}{2}(u_{i,j}^{m-1} + u_{i,j}^{m+1})$ and they got

$$(1+4\alpha)u_{i,j}^{m+1} = (1-4\alpha)u_{i,j}^{m-1} + 2\alpha(u_{i-1,j}^m + u_{i+1,j}^m + u_{i,j-1}^m + u_{i,j+1}^m)\,. \tag{6.3.13}$$

This equation is the Du Fort - Frankel method. The necessary starting values must be computed by another method. The convergence is guaranteed if the parameters of the grid satisfy certain additional condition, e.g. $k/h \to 0$. These conditions decrease the value of this method.

Similarly to the case of a single spatial variable it is possible to derive an explicit - implicit method where the new profile is computed by

$$u_{i,j}^{m+1} = (1 + \alpha(\delta_x^2 + \delta_y^2))u_{i,j}^m\,, \qquad m+i+j \ \text{ even}\,, \tag{6.3.14}$$

$$(1 - \alpha(\delta_x^2 + \delta_y^2))u_{i,j}^{m+1} = u_{i,j}^m\,, \qquad m+i+j \ \text{ odd}\,. \tag{6.3.15}$$

Formula (6.3.14) is an explicit one in the form of (6.3.8) and (6.3.15) is implicit, where we have all the values $u_{i-1,j}^{m+1}, u_{i+1,j}^{m+1}, u_{i,j-1}^{m+1}, u_{i,j+1}^{m+1}$ in the $(m+1)$-th profile computed by (6.3.14), thus (6.3.15) can be used for recurrent evaluation. This algorithm is illustrated in Fig. 6.8. It can be shown that this method is very similar to the Du Fort - Frankel method, so even here we need $k/h \to 0$.

For explicit method the temporal step size k is bounded by the stability condition or by the condition $k/h \to 0$. Thus implicit methods are often used instead. When used for problems described by (6.3.1) - (6.3.3) we need to solve a system of linear algebraic equations for $(n-1)^2$ unknowns in each step. The precise form of this system depends strongly on the type of the problem and on the method used; generally these systems are sparse because in each equation only a small number of unknowns appears. So for large n it is unreasonable to use finite methods (e.g. the Gauss elimination) because of memory and computation time demands.

It is possible to prepare a special algorithm with a finite method for a particular problem, but its applicability is restricted to this particular problem so it is not worth the effort.

Often the method called alternating direction implicit (ADI) is used involving two solutions of a three diagonal system of $(n-1)$ equations. The usage is similar to ADI for elliptic problems. Here, however, the block relaxation ADI is not done for the same time level. Or the point relaxation (upper) method can be used with only a few (usually just one) relaxation cycle for each time level.

Of fundamental meaning is the Crank-Nicolson method (which is always stable for problems (6.3.1) - (6.3.3)) with a five point scheme

$$\left(1 - \frac{\alpha}{2}(\delta_x^2 + \delta_y^2)\right)\boldsymbol{u}^{m+1} = \left(1 + \frac{\alpha}{2}(\delta_x^2 + \delta_y^2)\right)\boldsymbol{u}^m + \mathcal{O}(k^3 + kh^2) \tag{6.3.16}$$

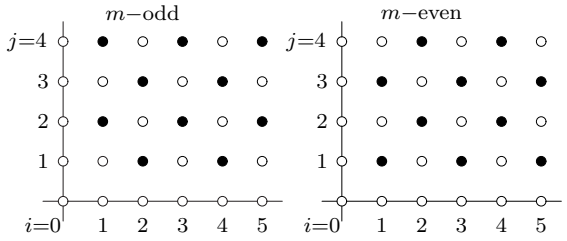


Figure 6.8: Explicit implicit method
• - values computed by (6.3.14)
○ - values computed by (6.3.15)
or from boundary condition

or a nine point scheme

$$\left(1-\frac{\alpha}{2}\delta_x^2\right)\left(1-\frac{\alpha}{2}\delta_y^2\right)\boldsymbol{u}^{m+1}=\left(1+\frac{\alpha}{2}\delta_x^2\right)\left(1+\frac{\alpha}{2}\delta_y^2\right)\boldsymbol{u}^{m}+\mathcal{O}(k^3+kh^2)\,. \tag{6.3.17}$$

They both are of order $\mathcal{O}(k^2+h^2)$. We get the ADI method by introducing additional profile $\boldsymbol{u}^+$ and by appropriate splitting the formula (6.3.16). This way we get the Peaceman-Rachford method

$$\left(1-\frac{\alpha}{2}\delta_x^2\right)\boldsymbol{u}^{+} = \left(1+\frac{\alpha}{2}\delta_y^2\right)\boldsymbol{u}^{m}\,, \tag{6.3.18}$$

$$\left(1-\frac{\alpha}{2}\delta_y^2\right)\boldsymbol{u}^{m+1} = \left(1+\frac{\alpha}{2}\delta_x^2\right)\boldsymbol{u}^{+}\,. \tag{6.3.19}$$

If we eliminate the profile $\boldsymbol{u}^+$, from (6.3.18) and (6.3.19) by simple manipulation we get (6.3.16). Fig. 6.9 illustrates the Peaceman-Rachford method.

There are other methods using alternating directions (Djakon method, Douglas-Rachford method etc.). The interested reader is invited to use the original literature.

Figure 6.9: Peaceman-Rachford method
• - known values,
○ - unknown values

As the number of unknowns and the number of equations for implicit methods depends heavily on n, namely as $(n-1)^2$, we try to reduce the number of nodes while keeping the accuracy. This can be done by using more nodes to approximate the spatial derivatives e.g.

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j} \approx \frac{-u_{i-2,j}+16u_{i-1,j}-30u_{i,j}+16u_{i+1,j}-u_{i+2,j}}{12h^2} \tag{6.3.20}$$

or at the boundary

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{1,j} \approx \frac{11u_{0,j}-20u_{1,j}+6u_{2,j}+4u_{3,j}-u_{4,j}}{12h^2}\,. \tag{6.3.21}$$

This method is illustrated in Fig. 6.10 for both explicit and implicit methods. The order in x and y is $\mathcal{O}(h^4)$, again Crank-Nicolson averaging can be used. Difference formulas of a very high order can be constructed, using up to all $(n-1)$ values of u so that even for small n a good accuracy can be reached in certain cases.

Solution of nonlinear parabolic equations in three dimensions is similar to two dimensional problems, the resulting implicit linear problems are solved by some method given above, e.g. upper relaxation or ADI.

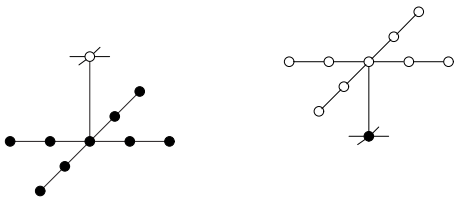


Figure 6.10: Explicit and implicit formula of higher order
• - known values,
◦ - unknown values

Similarly as for two independent variables, the method of lines can be used. Consider a quasi-linear equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + R(u)$$

with initial and boundary conditions (6.3.2), (6.3.3). Denoting $u_{i,j}(t) \doteq u(x_i, y_j, t)$, and using the simplest three point formulas we get

$$\begin{aligned} \frac{\mathrm{d}u_{i,j}}{\mathrm{d}t} &= \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2} + \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{h^2} + R(u_{i,j})\,, \\ u_{i,j}(0) &= \varphi(x_i, y_j)\,, \qquad i = 1, \ldots, n-1\,,\ j = 1, \ldots, n-1\,. \end{aligned}$$

The number of ordinary differential equations is in this case large, proportional to n^2. The advantage of this approach is that it is easy.

* * *

For further study see [1], [5], [7], [18], [19], [21], [23], [27], [28].

Bibliography

[1] Berezin M.S., Židkov N.P.: Mětody vyčislenij I,II. Fizmatgiz, Moskva 1962.

[2] Brenan K.E., Campbell S.L., Petzold L.R.: Numerical Solution of Initial Value Problems in Differential-Algebraic Equations. North-Holland, Elsevier, New York 1989.

[3] Ciarlet P.G.: Introduction to Numerical Linear Algebra and Optimisation. Cambridge Univ. Press, Cambridge 1989.

[4] de Boor C.: A Practical Guide to Splines. Springer Verlag, New York 1978.

[5] Děmidovič B.P., Maron I.A., Šuvalova E.Z.: Čislennyje mětody analiza. Fizmatgiz, Moskva 1967.

[6] Flaherty J.E., Paslow P.J., Stephard M.S., Vasilakis J.D., Eds.: Adaptive Methods for Partial Differential Equations. SIAM, Philadelphia 1989.

[7] Forsythe G.E., Wasow W.R.: Finite Difference Methods for Partial Differential Equations. Wiley, New York 1960.

[8] Golub G.H., Van Loan C.F.: Matrix Computations. The Johns Hopkins Univ. Press, Baltimore 1996.

[9] Hairer E., Norsett S.P., Wanner G.: Solving Ordinary Differential Equations I. Nonstiff Problems. Springer Verlag, Berlin 1987.

[10] Hairer E., Wanner G.: Solving Ordinary Diferential Equations II. Stiff and Differential-Algebraic Problems. Springer Verlag, Berlin 1991.

[11] Hairer E., Lubich C., Roche M.: The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods. Springer Verlag, Berlin 1989.

[12] Henrici P.: Discrete Varible Methods in Ordinary Differential Equations. J.Wiley, New York 1962.

[13] Himmelblau D.M.: Process Analysis and Statistical Methods. Wiley, New York 1970.

[14] Horn R. A., Johnnson C. R.: Matrix Analysis. Cambridge University Press, Cambridge, 1985.

[15] Kubíček M., Hlaváček V.: Numerical Solution of Nonlinear Boundary Value Problems with Applications. Prentice-Hall, Englewood Cliffs 1983.

[16] Lambert J.D.: Computational Methods in Ordinary Differential Equations. J.Wiley, London 1973.

[17] Lapidus L., Seinfeld. J.H.: Numerical Solution of Ordinary Differential Equations. Academic Press, New York 1971.

[18] Mitchell A.R.: Computational Methods in Partial Differential Equations. J.Wiley, London 1970.

[19] Mitchell A.R., Griffiths D.F.: The Finite Difference Methods in Partial Differential Equations. J.Wiley, New York 1980.

[20] Ortega J.M., Rheinboldt W.C.: Iterative Solution of Nonlinear Equations in Several Variables. Academic Press, New York 1970.

[21] Richtmyer R.D.: Difference Methods for Initial-Value Problems. Interscience, New York 1956.

[22] Roberts S.M., Shipman J.S.: Two Point Boundary Value Problems: Shooting Methods. Elsevier, New York 1971.

[23] Rosenbrock H.H., Storey C.: Computational Techniques for Chemical Engineers. Pergamon Press, London 1966.

[24] Saad Y.: Iterative Methods for Sparse Linear Systems. PWS Publ. Co., Boston 1996.

[25] Shampine L.F., Allen R.C.Jr., Pruess S.: Fundamentals of Numerical Computing. J.Wiley, New York 1997.

[26] Stoer J., Bulirsch R.: Introduction to Numerical Analysis. Springer Verlag, New York 1992.

[27] Villadsen J., Michelsen M.L.: Solution of Differential Equation Models by Polynomial Approximation. Prentice-Hall, Englewood Cliffs 1978.

[28] Von Rosenberg D.U.: Methods for the Numerical Solution of Partial Differential Equations. American Elsevier, New York 1969.

[29] Wilkinson J.H.: The Algebraic Eigenvalue Problem. Oxford University Press, London, 1965.

[30] Wilkinson J.H., Reinsch C.: Handbook for Automatic Computing. Linear Algebra. Springer Verlag, New York 1971.

Index