KfK 5120
Mai 1993

Solution Methods for Large Systems of Linear Equations in BACCHUS

Ch. Homann, B. Dorr
Institut für Reaktorsicherheit
Projekt Nukleare Sicherheitsforschung

Kernforschungszentrum Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE

Institut für Reaktorsicherheit
Projekt Nukleare Sicherheitsforschung

KfK 5120

Solution Methods for Large Systems of Linear Equations in BACCHUS

Ch. Homann, B. Dorr

Kernforschungszentrum Karlsruhe GmbH, Karlsruhe

Kernforschungszentrum Karlsruhe GmbH
Postfach 3640, 7500 Karlsruhe 1

ISSN 0303-4003

Solution Methods for Large Systems of Linear Equations in BACCHUS

The computer programme BACCHUS is used to describe steady state and transient thermal-hydraulic behaviour of a coolant in a fuel element with intact geometry in a fast breeder reactor. In such computer programmes generally large systems of linear equations with sparse matrices of coefficients, resulting from discretization of coolant conservation equations, must be solved thousands of times giving rise to large demands of main storage and CPU time. Direct and iterative solution methods of the systems of linear equations, available in BACCHUS, are described, giving theoretical details and experience with their use in the programme. Besides use of a method of lines, a Runge-Kutta-method, for the solution of the partial differential equation is outlined.

Lösungsmethoden für große lineare Gleichungssysteme in BACCHUS

Das Computerprogramm BACCHUS wird zur Beschreibung stationärer und transienter thermohydraulischer Kühlmittelzustände in Brennelementen mit intakter Geometrie für Schnelle Brüter benutzt. In solchen Computerprogrammen werden normalerweise viele tausendmal große lineare Gleichungssysteme mit schwach besetzten Koeffizientenmatrizen gelöst, die sich aus der Diskretisierung der Erhaltungsgleichungen für das Kühlmittel ergeben. Deshalb werden viel Speicherplatz und lange Rechenzeiten benötigt. Direkte und iterative Lösungsmethoden für diese linearen Gleichungssysteme, die in BACCHUS verfügbar sind, werden beschrieben. Dabei werden theoretische Einzelheiten und Erfahrungen aus ihrer Benutzung im Programm genannt. Außerdem wird die Benutzung einer Linienmethode, eines Runge-Kutta-Verfahrens, zur Lösung der partiellen Differentialgleichungen dargelegt.

Contents

Figures

Tables

1.0 Introduction

BACCHUS is a computer programme to describe three-dimensional thermal-hydraulic behaviour of a coolant in a fuel element with intact geometry in a fast breeder reactor for nominal and accident conditions. Its first three-dimensional single-phase version has been documented in [1], details being given in [2]. A detailed description of the latest version is given in [3]. It contains all current modelling features of the various programme versions available earlier. Among them are those for two-phase flow and local blockages.

In computer programmes like BACCHUS discretization of the conservation equations for mass, momentum, and energy, to be solved for the coolant, generally leads to sets of linear algebraic equations. In BACCHUS such systems of linear equations are derived for coolant pressure, coolant enthalpy, and, furthermore, for temperature in the fuel pins.

For programme applications, irrespective of whether large fuel elements in a commercial reactor are to be considered or laboratory experiments are dealt with, total CPU times may be several hours even on the fastest computers. A substantial portion of the total CPU time, say 50 to 80 percent, is spent for the solution of the systems of linear equations. Storage demands may be large likewise. For calculations concerning the local blockage experiment Mol 7C/7, documented in [4], nearly the whole available main storage region of 48 MByte of the vector computer VP50 was needed for the programme; more than one half of this region was necessary for the solution of the systems of linear equations. Moreover the precision demands for the solution are large. Details will be given later.

Because of these large demands it is worthwhile to spend some ideas on these numerical aspects of the code. In current textbooks solution methods can easily be found, but it turns out that the large demands of minimum storage, minimum CPU time, and maximum precision are not easily fulfilled. Therefore it is the aim of this report to present solution methods for the systems of linear equations available in BACCHUS, to compare them, to summarize some experience, and so give some suggestions for application of the code which may also be helpful for other codes. As direct methods LU decomposition and Gauss elimination are presented in this report, as iterative methods a block method and the ADI method. As an alternative method for the solution of the partial differential equations a method of lines, a Runge-Kutta method, is outlined. It works without setting up systems of linear equations. Besides some theoretical background about matrices is summarized and some characteristics of the matrix of coefficients as used in BACCHUS are given.

2.0 Derivation of Systems of Linear Equations in BACCHUS

To describe three-dimensional transient thermal-hydraulic behaviour of the coolant in a fast breeder reactor bundle with the computer code BACCHUS the solution domain is divided into a number of control volumes containing fluid and solid material. For the location of the variables for the coolant staggered meshes are used [2], [3]. The radial and azimuthal discretization of the fuel bundle is shown in Figure 1 on page 3. The radial discretization shown in this figure corresponds to the coarse one used usually. For special applications the spacing between subsequent pin rows (except for IC = 1) can be discretized by two control volumes (fine discretization) as it is used in [4]. For a 169-pin bundle as it was to be used for SNR 300 eight control volumes in the radial direction are necessary for coarse radial discretization. In the azimuthal direction the bundle is divided into 6, 12 or 24 sectors according to the problem to be investigated. Due to the coupling of adjacent cells there is also a coupling between the first and the last sector in a ring. Sometimes this gives some difficulties for the solution of the system of linear equations. The number of axial sections in the bundle (planes) is arbitrary, and the axial length of the control volumes need not be equidistant. The first and the last control volume in the axial direction are dummy meshes used to impose boundary conditions. As a minimum about 40 cells in the axial direction are used as in [5]; 120 cells are used in [4].

For the fluid portions in the control volumes the conservation equations for coolant mass, momentum, and energy are solved for every control volume and for every time step. In BACCHUS the coolant continuity and momentum equation are combined to form a Poisson like equation for pressure. In every time step first this pressure equation is solved, then coolant velocity components are derived from the momentum equation and coolant temperature from enthalpy equation. Finally heat conduction equation for the pins is solved giving actual heat fluxes between pins and coolant. In every time step the solution scheme must be repeated several times to satisfy all conservation equations simultaneously. In these iterations always the latest values for physical quantities are used. In literature this iteration process is usually called outer iteration.

As an example for the treatment of the basic equations in BACCHUS we consider the single-phase coolant enthalpy equation in a simple form

$$\frac{\partial \rho h}{\partial t} + \nabla \cdot (\rho \vec{v} h) = \nabla \cdot (\rho \alpha \nabla h) + Q \qquad (1)$$

where h, $\vec{v}$, α, ρ are coolant enthalpy, velocity, thermal diffusivity, and density, respectively, t is time, and Q heat source. The thermal diffusivity is given by

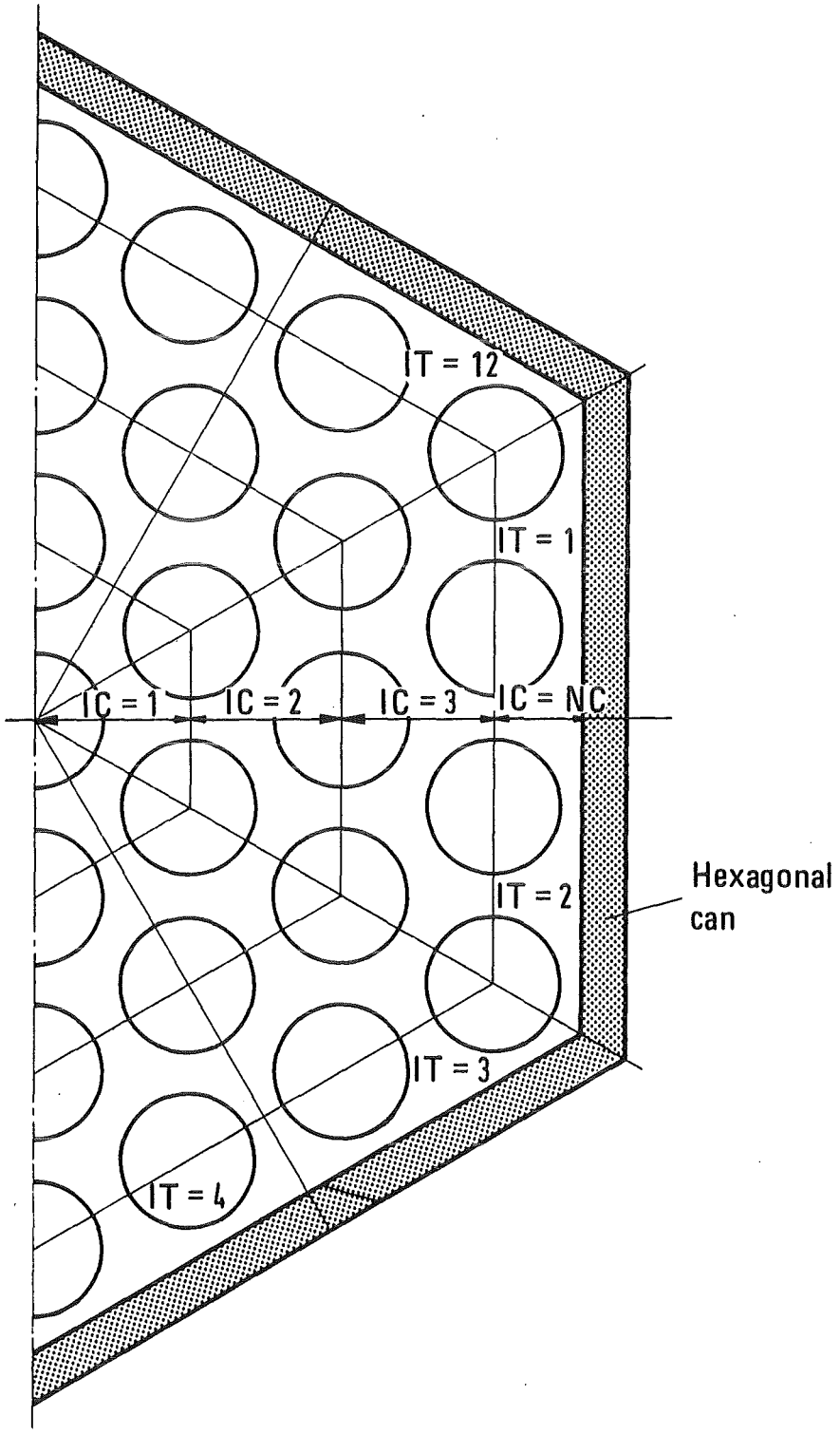

Figure 1. Radial and azimuthal discretization in BACCHUS

$$\alpha = \frac{\lambda}{\rho c_p} \tag{2}$$

where λ and c_p are coolant thermal conductivity and specific heat, respectively. The conservation equations are integrated over the respective fluid control volume using the Gauss theorem and divided by the total cell volume. Explicit time derivative is discretized by forward time difference scheme, in convection and diffusion terms enthalpy is discretized with new time values. This yields

$$\frac{\varepsilon\rho}{\Delta t}\left(h^{n+1} - h^n\right) + \sum_{i=1}^{6}(-1)^i \frac{\gamma_i}{\Delta x_i}\left(\rho v_i h^{n+1} + \rho\alpha \frac{\partial h^{n+1}}{\partial x_i}\right) = \varepsilon\, Q^{n+1} \tag{3}$$

for averaged values in a given control volume. n is the index of the old time step. ε and γ_i are volume porosity and surface permeabilities of the given control volume, Δx_i is the distance between adjacent enthalpy locations in the respective coordinate direction. The sum extends over the six surfaces of the control volume in question and contains values which are averaged over the respective surface. Even values of index i refer to positive coordinate direction. For spatial discretization first order upwind differences for convective terms and central differences for diffusive terms are used. For time discretization forward differences are used; convective and diffusive fluxes are treated implicitly in time. Rearranging gives an equation for averaged enthalpy at the new time level n+1 which has the same form as a discretized Poisson equation

$$a_0\, h_0^{n+1} + \sum_{i=1}^{6} a_i\, h_i^{n+1} = b_0 \tag{4}$$

Enthalpy in the given control volume, here labeled "0", is coupled with enthalpy in the six surrounding control volumes. The coefficients in eq. (4) are given by

$$a_i = -\frac{\gamma_i}{\Delta x_i}\left(k_i \rho v_i + \frac{\rho\alpha}{\Delta x_i}\right) \qquad i = 1(1)6 \tag{5}$$

$$a_0 = \sum_{i=1}^{6}(1 - k_i)\,\rho v_i + \frac{\rho\alpha}{\Delta x_i} + \frac{\varepsilon\rho}{\Delta t} \tag{6}$$

$$b_0 = \varepsilon\, Q^{n+1} + \frac{\varepsilon \rho h^n}{\Delta t} \tag{7}$$

k_i is 0 or 1 according to the prescription of upwind differencing. Evidently the coefficients are not constant for the whole solution domain so that fast Fourier transforms are not applicable.

An equation of similar form as eq. (4) is obtained for pin temperature when heat conduction is treated three-dimensionally as it is possible in BACCHUS since some time [6]. Formally this equation corresponds to coolant enthalpy equation for stagnant sodium.

A similar equation is also obtained for coolant pressure when the coolant continuity and momentum equation are combined. The definition of the coefficients a_i is lengthier than for enthalpy in eq. (5) [3], but $a_i < 0$ for all coefficients. Furthermore

$$a_0 = -\sum_{i=1}^{6} a_i + c_0 \qquad c_0 \geq 0 \tag{8}$$

instead of eq. (6). For the single-phase pressure equation and for the separated phases model $c_0 = 0$, for the two-phase slip model pressure equation this term refers to fluid compressibility.

As an equation of the form of eq. (4) exists for every control volume of the solution domain, a system of linear equations results which can be written in matrix form

$$A\vec{x} = \vec{b} \tag{9}$$

with dim A = n if the given discretization comprises n cells. $\vec{x}$ stands for coolant pressure, coolant enthalpy, or pin temperature.

As a measure for precision of the solution of pressure equation local mass imbalance is used. If it exceeds 10^{-8} kg/s for a total mass flow of the order of magnitude of 1 kg/s, two-phase flow calculations are not possible. As a nearly trivial request to meet this precision 8-byte words must be used for all real variables in the whole programme.

3.0 Some Properties of the Matrix of Coefficients

Definitions and theorems for matrices considered in this and the following sections and some theoretical background for matrices are given in Appendix A.

To set up the matrix of coefficients in eq. (9), we number the control volumes subsequently as follows. For the coolant control volumes we begin at the bundle inlet and number the rings subsequently in clockwise direction beginning with the innermost ring. Having numbered the first plane we continue with the next plane until we reach the bundle outlet. The control volumes in the pins are numbered similarly.

The resulting structure of the matrix of coefficients is shown in Figure 2 on page 7 for the case of three physically relevant axial, four radial, and six azimuthal control volumes. The nonzero entries are marked by crosses. The matrix has a regular block structure. For any number r of physically meaningful axial meshes, i. e. without the dummy meshes, it can be written as a block tridiagonal matrix

$$A = \begin{pmatrix} A_1 & B_1 & & & \\ C_1 & A_2 & B_2 & & \\ & C_2 & A_3 & B_3 & \\ & & \cdots & & \\ & & C_r & A_r & \end{pmatrix} \tag{10}$$

where the submatrices correspond to the large boxes in Figure 2 on page 7. The block matrices A_i are again block tridiagonal matrices

$$A_i = \begin{pmatrix} A'_1 & B'_1 & & & \\ C'_1 & A'_2 & B'_2 & & \\ & C'_2 & A'_3 & B'_3 & \\ & & \cdots & & \\ & & C'_p & A'_p & \end{pmatrix} \tag{11}$$

for p rings in a plane. The submatrices correspond to the small boxes in Figure 2 on page 7. In block matrices A'_i the off-diagonal elements are due to the azimuthal coupling. The singular off-diagonal nonzero entries in the corners of submatrix A'_i are a consequence of the azimuthal coupling of the first and the last sector in a ring. Block matrices B'_i and C'_i in eq. (11) give the radial coupling and block matrices B_i and C_i in eq. (10) are due to the axial coupling of control volumes. B'_i, C'_i, B_i and C_i are therefore diagonal matrices.

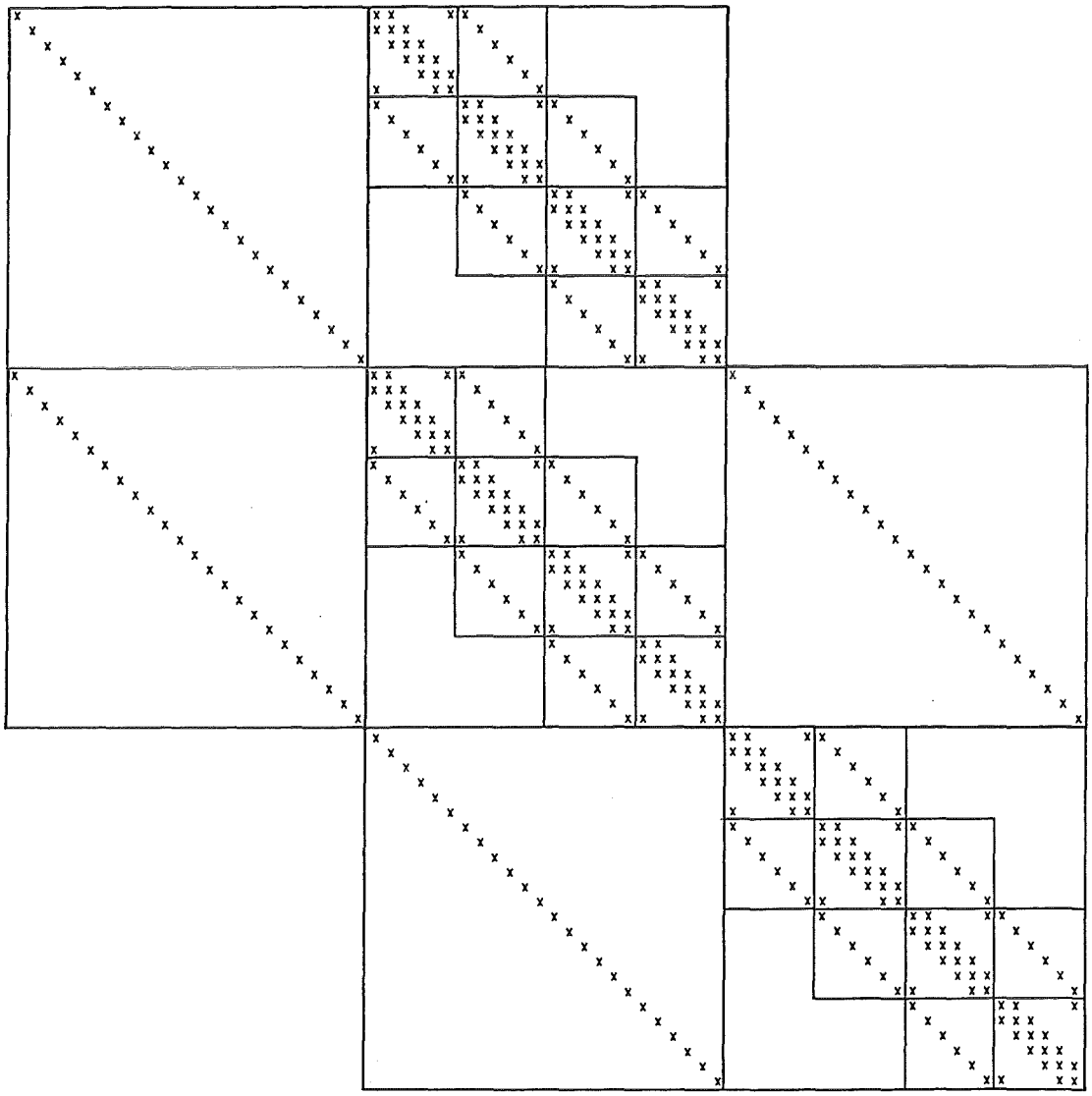

Figure 2. Structure of matrix of coefficients

Evidently the matrix is sparse. No more than seven entries in a row are nonzero for a total of n entries. The matrix is a band matrix. The bandwidth is just the number of control volumes in a plane. Even within the bandwidth the matrix is sparse.

For all systems of linear equations we obtain in BACCHUS

$$a_{ij} \leq 0 \qquad i \neq j \tag{12}$$

The unequality holds only for the six entries corresponding to the terms of the sum in eq. (4). For higher order difference schemes as they are described in [7] for a two-dimensional test problem more than six entries of the matrix of coefficients would be nonzero. We get furthermore

$$a_{ii} > 0 \tag{13}$$

Therefore the matrix of coefficients is an L-matrix.

The matrix of coefficients is asymmetric for pressure and enthalpy equations, and hence it is not positive definite (theorem 14 in Appendix A). Multiplication of the matrix of coefficients with its transposed would give a symmetric matrix but the condition number of the matrix and hence the accuracy of the solution would be deteriorated [8].

The matrix of coefficients for pressure with $c_0 = 0$ in eq. (8) is irreducibly diagonally dominant because it is derived from a boundary value problem. Boundary conditions give unequality in eq. (8) for one plane. For other systems of linear equations it is strictly diagonal dominant. Hence the matrix is nonsingular (theorem 7 in Appendix A). From theorem 16 in Appendix A it follows moreover that the matrix is an M-matrix. Consequently $A^{-1} > 0$, B_J is non-negative, irreducible, and convergent (theorem 33 in Appendix A). The second line of eq. (A.42) holds (theorem 37 in Appendix A).

For all systems of linear equations the matrix of coefficients has Property A because only central and upwind differencing are used for the discretization of the conservation equations. Proof: We denote the cells by the three indices i, j, and k corresponding to the coordinate directions. A cell (i,j,k) is coupled only with its six nearest neighbours. So the ordering vector can be given the components $\gamma_i = \mathrm{mod}((i + j + k), 2)$. If higher order discretization schemes are used, where cell (i,j,k) is also coupled with, say, cell (i-1,j-1,k) or with cell (i-2,j,k), the matrix of coefficients does not have Property A.

4.0 Direct Solution Methods in BACCHUS

In literature generally LU decomposition and Gaussian elimination are described as direct methods. Sometimes the links between these two methods are also outlined. Since both methods are available in BACCHUS, they are described here. It is assumed in this section that the matrix of coefficients is nonsingular because the case of singular matrices is not relevant for BACCHUS.

4.1 LU Decomposition

Instead of solving

$$A\vec{x} = \vec{b} \tag{14}$$

we try a decomposition of A into the product of an upper matrix U and a lower matrix L

$$A = LU \tag{15}$$

From theorem 3 in Appendix A and the fact that

$$\det A = \det L \det U \tag{16}$$

it can be shown that

$$l_{ii}, u_{ii} \neq 0\,, \quad i = 1(1)n \tag{17}$$

Instead of solving eq. (14) we have to solve

$$LU\,\vec{x} = \vec{b} \tag{18}$$

To solve eq. (18) we set

$$U\vec{x} = \vec{y} \tag{19}$$

and first solve

$$L\vec{y} = \vec{b} \tag{20}$$

for $\vec{y}$ and then use eq. (19) to solve for $\vec{x}$.

Once the decomposition of A is done, the solution of eqs. (20) and (19) is simply obtained by forward substitution similarly to eqs. (A.8) and (A.11)

$$y_1 = \frac{b_1}{l_{11}} \qquad y_i = \frac{1}{l_{ii}} \left(b_i - \sum_{k=1}^{i-1} l_{ik} y_k \right) \qquad i = 2(1)n \tag{21}$$

and backsubstitution, respectively.

$$x_n = \frac{y_n}{u_{nn}} \qquad x_i = \frac{1}{u_{ii}} \left(y_i - \sum_{k=i+1}^{n} u_{ik} x_k \right) \qquad i = n-1(-1)1 \tag{22}$$

In literature this method is recommended when several systems of linear equations with the same matrix of coefficients but with different right-hand sides are to be solved. In this case the LU decomposition must only be done once, and the various right-hand sides of eq. (14) can be treated simultaneously in eq. (21) because the right-hand sides in eqs. (20) and (19) mainly depend on the matrix coefficients and not on the right-hand sides of eq. (14).

A decomposition according to eq. (15) is not always possible for a nonsingular matrix A, but it can be shown [9] that for any nonsingular matrix A such a decomposition exists after certain permutations of rows.

From eq. (15) we get n^2 equations for the elements of L and U according to the rules of matrix multiplication, but since the main diagonals of both matrices, L and U, are non-zero, there are $n^2 + n$ unknowns. In practice all diagonals of one of these two matrices are set to one. In literature two methods are known, the Crout method [10] where $u_{ii} = 1$ and the Doolittle method [11] where $l_{ii} = 1$. For the latter the LU decomposition is done as follows.

$$\begin{aligned} u_1 &= a_{1j} \qquad j = 1(1)n \\ u_{ij} &= a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj} \qquad i = 2(1)n \quad j = i(1)n \end{aligned} \tag{23}$$

$$l_{i1} = \frac{a_{i1}}{u_{11}} \qquad i = 2(1)n$$

$$l_{ij} = \frac{1}{u_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj} \right) \qquad i = 3(1)n \quad j = 2(1)i - 1 \tag{24}$$

The i-th row of L and the i-th column of U are calculated alternatingly and stored subsequently, but without the main diagonal of L.

4.2 Gaussian Elimination

Another well-known method is Gaussian elimination, described in many textbooks as [9], [12]. For the time being it is assumed that all main diagonal entries are nonzero. (This assumption is always fulfilled in BACCHUS.) In a first step the first equation is multiplied by

$$l_{i1} = \frac{a_{i1}}{a_{11}} \qquad i = 2(1)n \tag{25}$$

and subtracted from equation i for i = 2(1)n. In a second step the second equation is multiplied by

$$l_{i2} = \frac{a_{i2}}{a_{22}} \tag{26}$$

and subtracted from equation i for i = 3(1)n. This process is continued to transform eq. (14) into

$$U\vec{x} = \vec{c} \tag{27}$$

with an upper diagonal matrix U. The unknowns in eq. (27) are obtained by backsubstitution similar to eq. (22).

The forward elimination can also be described by matrix multiplication: For the first step according to eq. (25) the original matrix is premultiplied by a lower triangular matrix G with

$$g_{ij} = \begin{cases} 1 & i = j \\ -l_{i1} & j = 1,\ i \neq j \\ 0 & \text{else} \end{cases} \tag{28}$$

as it is pointed out in [9]. A matrix like G which differs from a unity matrix only as it can contain nonzero elements in one column is called Frobenius matrix. Its inverse H is given by

$$h_{ij} = \begin{cases} 1 & i = j \\ l_{i1} & j = 1, i \neq j \\ 0 & \text{else} \end{cases} \tag{29}$$

One of the simplest cases for Gaussian elimination is for tridiagonal matrices. These are band matrices with a bandwidth of 1. For $n = 6$ a tridiagonal matrix has the form

```
x x
x x x
  x x x
    x x x
      x x x
        x x
```

It can be seen that x_1 has only to be eliminated from the second equation, then x_2 has only to be eliminated from the third equation and so on. Backsubstitution is also very simple. So tridiagonal systems are very easy to program, need negligibly small storage (only the three diagonals and some auxiliary arrays) and small computing time. The solution algorithm according to Thomas is often cited in literature as [13] and [14]. Details are given in Appendix B. For moredimensional problems, however, it cannot be used directly.

For any matrix of coefficients the solution procedure described above fails evidently if a main diagonal element is zero (see eqs. (25) and (26)). It can be shown, however, that for a nonsingular matrix A this case can be circumvented by interchanging rows or columns (pivoting). This corresponds to a multiplication of the original system of linear equations with permutation matrices. E. g. interchanging rows 1 and r is done by premultiplication of A with permutation matrix P [9] with

$$p_{ij} = \begin{cases} 1 & \begin{cases} i = j, \text{ but } i \neq 1, \ r \\ i = 1 \text{ and } j = r \\ i = r \text{ and } j = 1 \end{cases} \\ 0 & \text{else} \end{cases} \tag{30}$$

It can be shown [15], [16] that there are close connections between LU decomposition and Gaussian elimination. Therefore there are some common features of these two methods: For a nonsingular matrix they can always be used. For diagonal dominant matrices pivoting is not necessary though some people recommend it nevertheless. The matrix of coefficients and the right-hand side of the equation can be overwritten during

the solution procedure. In this way much storage can be saved. The band structure of the matrix of coefficients is preserved during the solution procedure. This feature is used to save storage and computer time: only the matrix band is stored, and matrix multiplications are only done within the band. However, there is a fill-in within the band of the matrix. Therefore the whole band must be stored and handled. As a consequence storage and CPU time requirement are proportional to the total number of meshes and to the bandwidth of the matrix of coefficients. For large problems this means large efforts. Since the bandwidth is given by the numbering conventions of the unknowns, a bad sequence may have drastic effects on computational requirements.

When roundoff errors influence the numerical solution of a system of linear equations, the numerical solution can be improved iteratively [8], [12] in the following manner. Let the numerical solution of eq. (14) be $\vec{x}^{(1)}$. Defining

$$\vec{r}^{(1)} = \vec{b} - A\vec{x}^{(1)} \tag{31}$$

and solving

$$A\,\Delta\vec{x}^{(1)} = \vec{r}^{(1)} \tag{32}$$

for $\Delta\vec{x}^{(1)}$, it can be shown that

$$\vec{x}^{(2)} = \vec{x}^{(1)} + \Delta\vec{x}^{(1)} \tag{33}$$

is a better approximation of the correct solution This procedure can be used iteratively [8]. Up to now it was not thought necessary to use such an improvement in BACCHUS, but it might be necessary for future applications. In such a case LU decomposition would in principle be more favourite, because L und U need only be calculated once. A good vectorization of the respective subroutines is, however, necessary.

4.3 Considerations for BACCHUS

Since for all applications in BACCHUS the number of planes is larger than that of sectors or rings, the numbering conventions for the control volumes are intended to keep storage and CPU time requirements as low as possible. If the numbering would not follow the coordinate directions somewhat smaller bandwidths may be possible, but the additional computational efforts to renumber the meshes would probably outweigh the gain in bandwidth.

As a consequence of the numbering conventions in BACCHUS, storage and CPU time requirements are proportional to the number of meshes in the axial direction and proportional to the square of the number of meshes in the radial and the azimuthal direction. So when the axial discretization is refined computational efforts increase less than when azimuthal discretization is refined or when a bundle with more pin rows is to be considered.

To solve systems of linear equations directly it is also possible to use methods based on the division of the matrix of coefficients into block matrices. A solution procedure is given in [14] for a block tridiagonal matrix and in [17] for general block matrices. In the case of a block tridiagonal matrix decomposition, eq. (10), the inverse of every main diagonal block matrix A_i must be calculated explicitly, and it was thought that this costs rather much computing time. The off-diagonal block matrices are diagonal matrices reducing computational effort for matrix multiplication according to eqs. (A.3) and (A.4).

Sometimes direct solution routines are published which are said to be very fast or to use a very small amount of additional storage. Some of them, collected by [18], have been tried in BACCHUS. However, the advantages of these solution routines are only true for cases with only little fill-in. In contrast the whole band is filled with non-zero elements for problems under consideration in BACCHUS and therefore no method of this kind was suitable up to now.

5.0 Iterative Solution Methods in BACCHUS

In contrast to direct methods explained in the previous section storage and CPU time requirements for iterative methods are generally proportional to the number of meshes. Therefore storage increases much slower with increasing problem size than for direct methods. CPU time requirement should therefore also increase with the number of meshes. Since, however, convergence for larger problems may be slower than for small ones, it is more difficult to assess CPU times than storage.

5.1 SOR

Though successive overrelaxation is rather old and somewhat obsolete, it is still mentioned in more recent textbooks [19]. It has been studied extensively in [13] for certain classes of matrices. Further classes of matrices for which overrelaxation can improve convergence are given in [15]. It was the first solution method implemented in BACCHUS (which did not contain azimuthal discretization at that time [20]). In practice this method is executed as follows. For every iteration (index k) first a Gauss-Seidel iteration step is made, i. e. equation i of system eq. (9) (i = 1, ... n) is solved for x_i using the latest values for all other unknowns. This yields a provisional solution

$$\hat{x}_i^k = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j \right) \tag{34}$$

for iteration number k, and the final solution is then given by

$$x_i^k = x_i^{k-1} + \omega \left(\hat{x}_i^k - x_i^{k-1} \right) \tag{35}$$

These two equations are equivalent to

$$x_i^k = (1 - \omega) x_i^{k-1} + \frac{\omega}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^k - \sum_{j=i+1}^{n} a_{ij} x_j^{k-1} \right) \tag{36}$$

So the difference between subsequent Gauss-Seidel iterations for x_i is multiplied by a given factor ω. For $\omega > 1$ the provisionally computed difference, eq. (34), is increased (overrelaxation), otherwise it is increased (underrelaxation). The essential task for SOR is to determine ω. In practice this is done approximately. Such methods are described in literature, some of them have been tested in BACCHUS and are documented in [2].

Intuitively it is clear that overrelaxation improves convergence if the difference between subsequent iterations is a monotonous function of iteration number and that underrelaxation gives an improvement if this difference oscillates. Another way of understanding overrelaxation is to consider the residuum

$$r = a_{ii}x_i + \sum_{j=1}^{i-1} a_{ij}x_j^k + \sum_{j=i+1}^{n} a_{ij}x_j^{k-1} - b_i \tag{37}$$

and to apply a Newton-Raphson iteration

$$x_i^k = x_i^{k-1} - \omega \frac{\partial x_i^{k-1}}{\partial r} r \tag{38}$$

The partial derivative in eq. (38) is obtained from eq. (37):

$$\frac{\partial x_i^{k-1}}{\partial r} = \frac{1}{a_{ii}} \tag{39}$$

5.2 ADI Methods

5.2.1 Procedure

When the first two-dimensional version of BACCHUS was extended to a three-dimensional code, SOR was not accurate enough. The reason has not been investigated in detail; it was thought, however, that the alternating direction implicit (ADI) method would improve the situation.

The idea of this method is a matrix splitting. Instead of solving one large problem, a number of smaller ones, more easily to solve, are created. In case of ADI the splitting is related to the coupling of cells in the three coordinate directions. In a first step only axial coupling is kept on the left-hand side of the system of linear equations, the rest being put on the right-hand side. In the second step only radial coupling is kept on the left-hand side, in the third step only azimuthal coupling. Several variants are known in literature [13], [14]. The oldest one has been proposed by Peaceman and Rachford [21]:

$$
\begin{aligned}
(A_z + rI)\,\vec{x}^{(1)} &= \vec{b} - (A_z^- - rI)\,\vec{x}^{(n)} \\
(A_r + rI)\,\vec{x}^{(2)} &= \vec{b} - (A_r^- - rI)\,\vec{x}^{(1)} \\
(A_s + rI)\,\vec{x}^{(3)} &= \vec{b} - (A_s^- - rI)\,\vec{x}^{(2)} \\
\vec{x}^{(n+1)} &= \vec{x}^{(3)}
\end{aligned}
\tag{40}
$$

A_z, A_r, A_s are matrices the nonzero entries of which consist only of coefficients related to the coupling in axial, radial, and azimuthal direction, respectively, formally written as

$$
\begin{aligned}
A_z &= \{a_{z-}, -(a_{z-} + a_{z+}), a_{z+}\} \\
A_r &= \{a_{r-}, -(a_{r-} + a_{r+}), a_{r+}\} \\
A_s &= \{a_{s-}, -(a_{s-} + a_{s+}), a_{s+}\}
\end{aligned}
\tag{41}
$$

Besides

$$
A_i^- = A - A_i \qquad i = z, r, s \tag{42}
$$

and r is a non-negative real number influencing convergence behaviour. For the right-hand side of the eq. (40) always the latest results for the unknowns are used, beginning with the results from iteration n. The third step gives the results for iteration number n+1.

For every step of eq. (40) the equations are rearranged in such a way that the matrix on the left-hand side is tridiagonal so that the Thomas algorithm can be used. The use of this fast solution algorithm is the main reason for the development of ADI methods.

To get an idea of the meaning of the relaxation parameter r we consider the Poisson equation

$$
\Delta u = \rho \tag{43}
$$

as

$$
\lim_{t\to\infty} \frac{\partial u}{\partial t} = \Delta u - \rho \tag{44}
$$

and instead of solving eq. (43) we solve eq. (44). As a simple case we consider a two-dimensional case with equal spacing Δx for both coordinate directions, using forward differences for the left-hand side and central differences for the right-hand side of eq. (44). For the Peaceman-Rachford variant of the ADI method, eq. (40), r turns out to be

$$r = \frac{2\Delta x^2}{\Delta t} \tag{45}$$

and the choice of r in eq. (40) is equivalent to the choice of a time step Δt in eq. (44) [12].

The Peaceman-Rachford variant of ADI method described above has the disadvantage that the matrices on the right-hand side are evidently not tridiagonal. Therefore evaluation of the right-hand sides of eqs. (40) is computer time consuming, and efforts have been made to remove this drawback (see e. g. [14]). The best variant known to the authors is that of Messina and Londrillo [22]. It is the one which is actually available in BACCHUS.

$$\begin{aligned}
(A_z + rI)\,\vec{x}^{(1)} &= \vec{b} - A\,\vec{x}^{(n)} \\
(A_r + rI)\,\vec{x}^{(2)} &= r\,\vec{x}^{(1)} \\
(A_s + rI)\,\vec{x}^{(3)} &= r\,\vec{x}^{(2)} \\
\vec{x}^{(n+1)} &= x^{(n)} + 2\,\vec{x}^{(3)}
\end{aligned} \tag{46}$$

In this variant only in the first step a matrix vector multiplication has to be performed. Inspection shows, however, that the right-hand side of the first step gives the residuum of the last iteration, and this value may be used as a convergence criteria. In this and in other variants of the ADI method the meaning of r is not so simple as for the Peaceman-Rachford variant, but it can be shown that r must be nonnegative for convergence reasons.

5.2.2 Determination of Relaxation Parameter r

For further considerations we write Messina-Londrillo variant of the ADI method, eq. (46), similarly to SOR and other stationary iterative methods according to eq. (A.20) in Appendix A.

$$\vec{x}^{(n+1)} = B\vec{x}^{(n)} + \vec{c} \tag{A.20}$$

According to theorem 26 in Appendix A the ADI method converges if the spectral radius S of the iteration matrix B is less than unity. Now, the iteration matrix for eq. (46) can be written as

$$B = I - 2\,r^2(A_r + rI)^{-1}(A_s + rI)^{-1}(A_z + rI)^{-1}\,A \tag{47}$$

[23] and it can be seen that $S(B) = f(r, ...)$. So the spectral radius of the iteration matrix can be minimized by choosing an appropriate value of r. Theory shows, however, that even in this case convergence is not better than for point SOR [12]. This is disappointing because computational effort is much larger for ADI than for point SOR method. However, convergence is accelerated substantially, if r is varied cyclically. Several cycles may be done before the iterations are ended.

To get an impression of the spectrum of eigenvalues Λ_K of B, it has been computed for a case with 40 axial, 6 azimuthal, and 4 radial meshes in Figure 3 on page 20. In this figure p is the portion of eigenvalues in the interval $\Lambda_K - 0.05 \leq \Lambda_K < \Lambda_K + 0.05$. For $r = 1$ most eigenvalues are greater than 0.5. For larger values of r there is a clustering around $\Lambda_K = 0$ which fades for $r = 1000$. In this case most eigenvalues are positive.

In general it is not possible to calculate the eigenspectrum of the iteration matrix, and since "the determination of an optimal sequence of r's for an arbitrary elliptic problem is an unsolved problem, a heuristic solution should be found" [12]. Under strong assumptions, i. e. for a common set of eigenvectors for the respective matrices, the eigenvalues of B are given by the eigenvalues of A_z, A_s,A_r and by r to be

$$\Lambda = 1 - \frac{2\,r^2\,(\lambda_z + \lambda_r + \lambda_s)}{(\lambda_z + r)\,(\lambda_r + r)\,(\lambda_s + r)} \tag{48}$$

Though these assumptions are surely not fulfilled in general, eq. (48) is used in BACCHUS for a heuristic procedure. It is described in detail in [23] and runs as follows. Minimum, maximum, and average values for A_z, A_s, A_r are assessed from the coefficients of the respective matrices. Out of these values 27 tripels are formed and inserted into eq. (48). It is assumed that this procedure gives representative values Λ_h for the eigenspectrum of B.

To illustrate this procedure values of Λ_h are plotted for various r for three triples in Figure 4 on page 21. For small values of r the curves begin near $\Lambda_h = 1$, have a minimum somewhere, and either take the value $\Lambda_h = 0$ or come close to it. Experience for all computational cases examined up to now shows a similar behaviour for all 27 triples. Experience shows further that the ADI method converges optimally if r is varied cyclically between the minimum and the maximum values of r for which the 27 curves cross the line $\Lambda_h = 0$ or come close to it. As an example we suppose that the three curves given in Figure 4 on page 21 are envelopes for all 27 curves. Then r should be varied cyclically between about 5 and 2000. Since r spans a large range, generally over several orders of magnitude, r is not changed linearly for the cyclic variations but according to

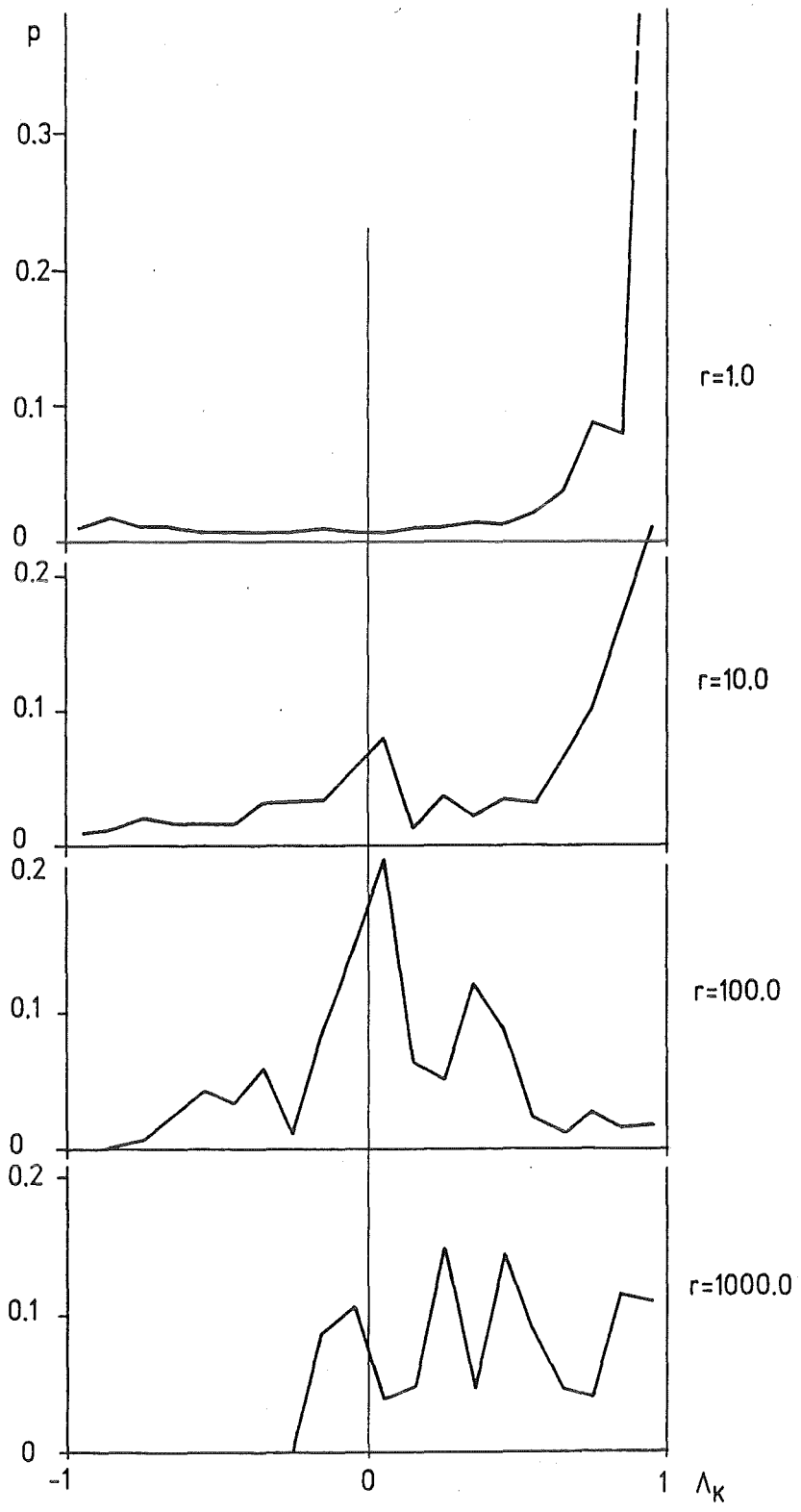

Figure 3. Eigenvalue distribution for various values of r

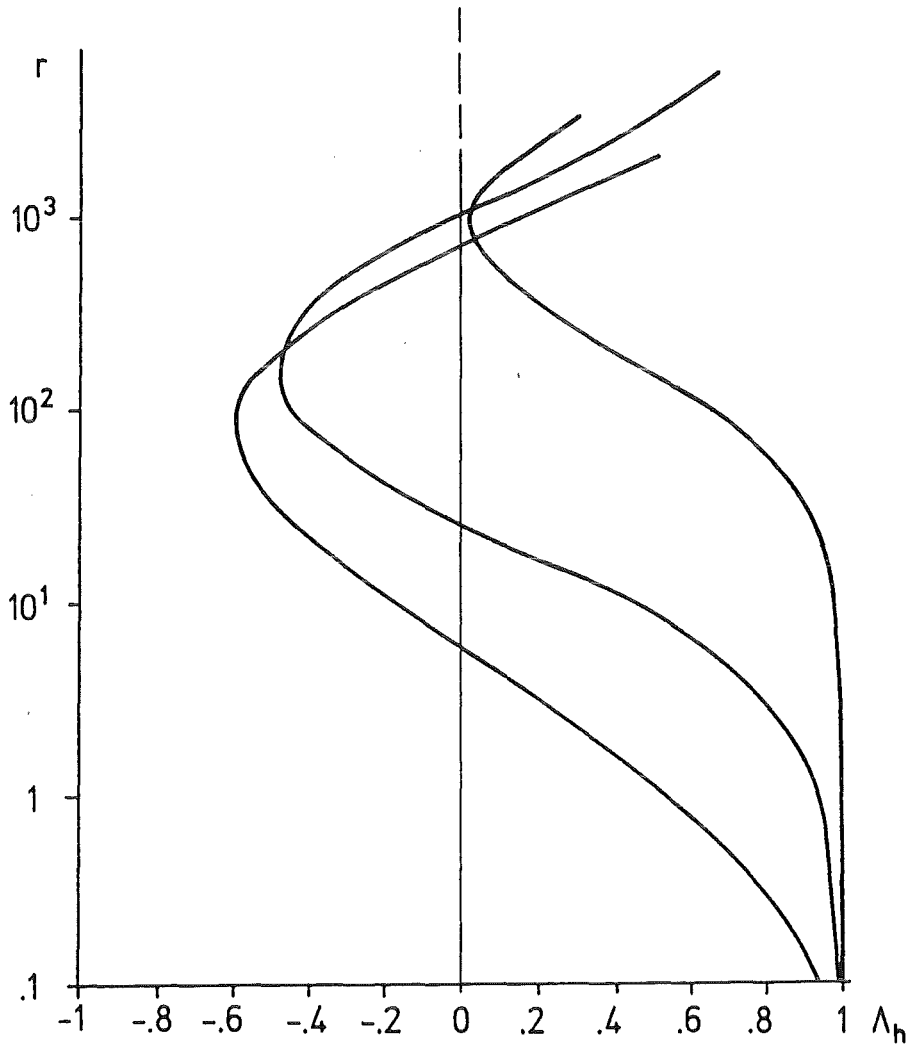

Figure 4. Relation between Λ and r for various triples

$$r_j = r_{min}\left(\frac{r_{min}}{r_{max}}\right)^{\frac{1-j}{m-1}} \qquad j = 1(1)m \tag{49}$$

for step number j in a cycle of m variations of r.

5.2.3 Handling of Azimuthal Coupling

In the third step of the ADI method when the azimuthal coupling is on the left-hand side of the equations (see eq. (46)) the system of linear equations is not tridiagonal, but cyclic as it is shown schematically below for one ring with six sectors

```
x x       x
x x x
  x x x
    x x x
      x x x
x       x x
```

To use the Thomas algorithm nevertheless, the coupling between the first and the last sector in a ring, i. e. the singular non-zero elements at the top right and the bottom left are put on the right-hand side of the equation. In this manner there are some convergence difficulties just at the link between the first and the last sector. They can be overcome if in this part of the solution procedure two sweeps are done instead of one. In the first sweep the sectors are numbered as shown below for the case of six sectors in a bundle.

```
    1
6       2
5       3
    4
```

In the second sweep the sectors are renumbered in the following manner.

```
    4
3       5
2       6
    1
```

In this way the coupling between the first and the last sector is positioned at different locations in the bundle cross section, and convergence is improved. As a further improvement this procedure has been replaced by the direct solution of the circular matrix as it is described in Appendix B.

5.3 Iterative Block Method

5.3.1 Procedure

Since Gaussian elimination and LU decomposition need much storage it has been tried to overcome this difficulty by using the group Gauss-Seidel or group overrelaxation method. Physically this means the following. The bundle is divided axially into q blocks as shown in the left part of Figure 5 on page 24 for q = 3. All blocks (generally except the last one) have the same length and so contain the same number Δj of planes. So the matrix of coefficients is partitioned into a block tridiagonal system

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} & & & \\ A_{2,1} & A_{2,2} & A_{2,3} & & \\ & A_{3,2} & A_{3,3} & A_{3,4} & \\ & & \cdots & & \\ & & & A_{q,q-1} & A_{q,q} \end{pmatrix} \tag{50}$$

If a block contains all meshes of a single plane, the submatrices in eq. (50) correspond those of eq. (10).

Irrespective of the block size, i. e. of the value of Δj, the system of linear equations within a block is solved with a direct method, the coupling between adjacent blocks being put on the right-hand side of the system of linear equations. Therefore in a given time step iterations over the blocks are necessary. Convergence is reached in a given time step, when the modulus of the residuum, defined by

$$\vec{r} = A\vec{x} - \vec{b} \tag{51}$$

becomes smaller than a given limit. Convergence may be accelerated by overrelaxation.

In this way a number of smaller systems of linear equations is solved instead of a large one. If a block consists of all nodes in a single plane the bandwidth of the matrix of coefficients is given by the number of sectors, otherwise by the number of cells in a plane as for the matrix of coefficients of the whole system of linear equations. In this way storage demands are decreased but computing time is increased because of the iterations over the blocks.

5.3.2 Theoretical Considerations

It has been tested that $B_J^{(\pi)}$ is a CO-matrix when a block consists of one plane. The ordering vector contains '1' for the first plane, '2' for the second, and '3' for the third plane in analogy to theorem 25 in Appendix A.

Generally, when a block consists of one or several planes, the matrix of coefficients, A, is block tridiagonal. The submatrices on the diagonal of A contain the coupling within a plane and, if several planes form a group, the coupling of the various planes within the group. The off-diagonal submatrices contain the coupling between subsequent groups. If one plane forms a group, these submatrices are diagonal matrices and hence nonsingular. The matrix of coefficients has Property $A^{(\pi)}$ because it is block tridiagonal, and it is a $\pi - CO -$ matrix according to the definition of $\pi - CO -$ matrix. So $\hat{B}^{(\pi)}$ and $\hat{C}^{(\pi)}$ are CO-matrices. Since for our grouping the sequence of the cell numbering is not changed, $\hat{B}^{(\pi)} = B^{(\pi)}$ and $\hat{C}^{(\pi)} = C^{(\pi)}$ (theorem 42 in Appendix A). $D^{(\pi)}$ is nonsingular because its submatrices are irreducibly diagonally dominant. Since A is a $\pi - CO -$ matrix,

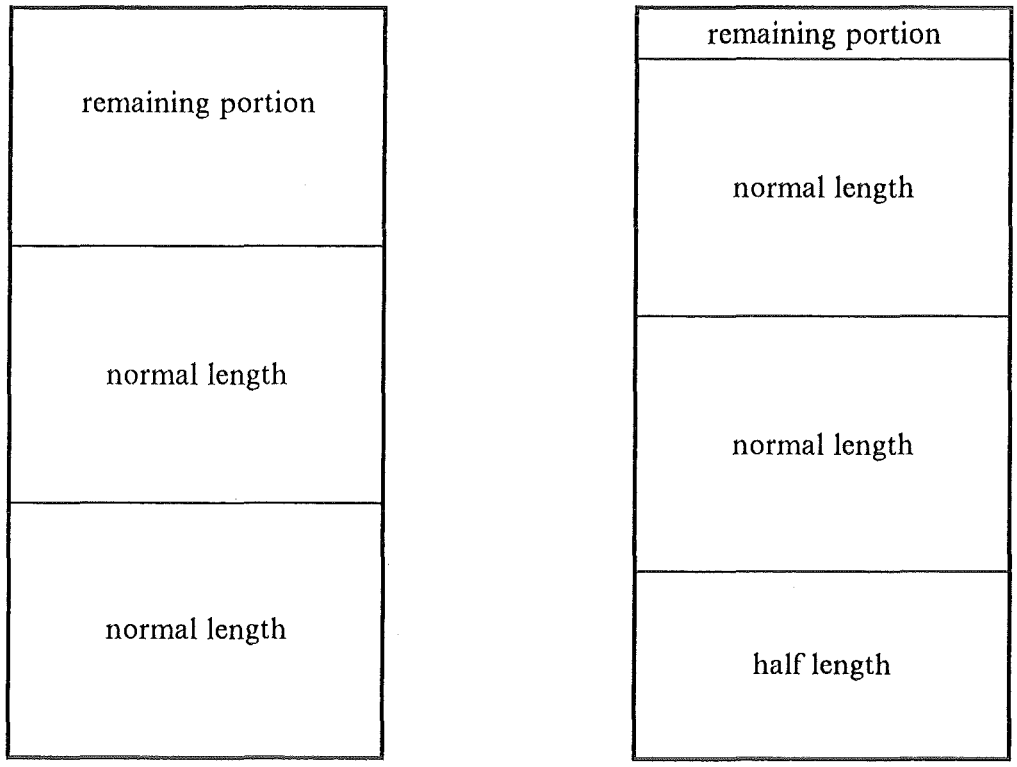

Figure 5. Axial cut through a bundle for iterative block method. The left part shows the partitioning of the bundle into blocks, the right part shows the block arrangement for every second sweep.

A is also a π – GCO – matrix (theorem 44 in Appendix A). The other implications of theorem 44 in Appendix A are also valid. As a consequence the relations between the eigenvalues of the group Jacobi and the group SOR method indicated in theorem 45 in Appendix A are valid.

For the group SOR method it is not clear from the above theorem that the spectral radius of the iteration matrix is less than unity. If it is the case the group SOR method converges. For a given matrix of coefficients the eigenvalues can be calculated. The iteration matrix related to the group Jacobi method can be obtained when the inverse of $D^{(\pi)}$ is calculated. Mathematically this is rather easy as this inverse is composed of the inverses of the respective block matrices D_i of $D^{(\pi)}$ but it gives a result only for a special case. Therefore it is left to inspect the results of a given case to see whether the group SOR method converges.

6.0 Runge-Kutta Methods

To solve systems of linear equations is not the only possibility to solve partial differential equations numerically. So methods of lines [24] consist of discretizing a partial differential equation for all variables but one. In this manner formally an ordinary differential equation is obtained. As an example we consider enthalpy equation (eq. (1)) and write it in a very short form as

$$\frac{\partial}{\partial t}(\rho h) = R(h, \vec{v}, Q...) \tag{52}$$

where the right-hand side contains all other terms, i.e. convective and diffusive fluxes and heat source, all of them discretized in the usual manner (eq. (3)). We differentiate the left-hand side of the equation to get

$$\frac{\partial}{\partial t}(\rho h) = \rho \frac{\partial h}{\partial t} + h \frac{\partial \rho}{\partial t} \tag{53}$$

We substitute eq. (53) into eq. (52), and put the last term of eq. (53) on the right-hand side of eq. (52) using forward differencing. So we get formally an ordinary differential equation in time for enthalpy. Since it can be written for every enthalpy node, we get a set of ordinary differential equations of the following form

$$\vec{y}' = f(x, \vec{y}) \tag{54}$$

To solve eq. (54) we use a method of the theory of ordinary differential equations, namely an explicit Runge-Kutta method of order four. Its idea is as follows. We suppose that a solution of eq. (54) at step i, $\vec{y}_i$ is known and that the solution of eq. (54) at step $i+1$, $\vec{y}_{i+1}$, is to be calculated, the step width being h_i. We try

$$\vec{y}_{i+1} = \vec{y}i + h_i \sum_{j=1}^{m} A_j \vec{k}_j \tag{55}$$

where $\vec{k}_j$ are values of the right-hand side of eq. (54) at given positions:

$$\begin{aligned} \vec{k}_1 &= \vec{f}(x, \vec{y}_i) \\ \vec{k}_j &= \vec{f}(x + a_j h_i,\ \vec{y}_i + h_i \sum_{s=1}^{j-1} b_{js} \vec{k}_s) \qquad j = 2(1)m \end{aligned} \tag{56}$$

Coefficients A_j, a_j, and b_{js} are determined from expansions of eqs. (54) and (55) into Taylor series. For the widely used case m=4 this has been done in [25]. It turns out that some open parameters remain, giving rise to various sets of formula. Parameter sets for a large number of explicit Runge-Kutta methods are given in [17].

In BACCHUS the classical Runge method is implemented:

$$
\begin{aligned}
\vec{y}_{i+1} &= \vec{y}_i + \frac{h_i}{6}\left(\vec{k}_1 + 2\vec{k}_2 + 2\vec{k}_3 + \vec{k}_4\right) \\
\vec{k}_1 &= \vec{f}(x_i, \vec{y}_i) \\
\vec{k}_2 &= \vec{f}(x_i + \frac{h_i}{2}, \vec{y}_i + \frac{\vec{k}_1}{2}) \\
\vec{k}_3 &= \vec{f}(x_i + \frac{h_i}{2}, \vec{y}_i + \frac{\vec{k}_2}{2}) \\
\vec{k}_4 &= \vec{f}(x_i + h_i, \vec{y}_i + \vec{k}_3)
\end{aligned}
\qquad (57)
$$

Other variants are also listed in [25]. Some of them may give a higher accuracy than the Runge formula, but up to now they are not implemented in BACCHUS because there is an approximation used in the code which might influence accuracy more: Neither velocities in convective and diffusive fluxes nor material property data are updated for the calculations of $\vec{k}_1$ ... $\vec{k}_4$ in eq. (57). In the original version updating would have required much computing time. Besides when the Runge-Kutta method was implemented it was not clear whether stability would be decreased by this measure.

Runge-Kutta methods have the great advantage that no results of previous integration steps are needed. Their disadvantage is that no well-defined measure of accuracy exists. So a heuristic method must be used [25].

7.0 Experience gained with BACCHUS

Implementation of the various solution methods extended over many years and was generally done when problems arose for a given application of the programme. So experience is only gained for a given status of programme development, and the advantages or disadvantages of a solution method may change, when further programme development is done. For time reasons it is not possible to repeat a whole comparison of methods when the status of the programme had changed. This makes a direct comparison of the various methods somewhat difficult.

7.1 Test Cases

For comparison between Gaussian elimination and ADI method three test cases have been chosen (Table 1). There are a small problem case with about 1000 meshes, a medium problem case with about 3400 meshes and a large problem case with about 14000 meshes. The small case corresponds to calculations for 37-pin KNS experiments, the intermediate one to a 127-pin KNK fuel element, the large one to calculations for the 19-pin local blockage Mol 7C/7 experiment. Details about these three cases are given in [23].

Problem size		Region/MByte			CPU time/s	
$N_z * N_s * N_r$	nodes	code	Gauss	ADI	Gauss	ADI
40 * 6 * 4	960	2.7	0.3	$\simeq 0$	30	57
41 * 12 * 7	3444	8	4.4	$\simeq 0.1$	513	306
120 * 24 * 5	14400	19	27	2.6		

Table 1. Main storage region and CPU times for three representative cases on VP50. Gauss means Gaussian elimination. CPU times refer to quasi steady state cases.

The implementation of the block iterative method has been made at the same time as essential improvements for ADI method have been made. When the ADI method was available testing of the block iterative method has not been made so extensively for several problem sizes as for ADI method for time reasons. Instead tests have only been made with a small test case. It is rather an artificial case with 23 axial, six azimuthal, and 5 radial meshes. Eight grid spacers and a local blockage were simulated as shown in Figure 6 on page 29. This configuration was chosen because on the one hand it is a small problem quickly to solve and on the other hand it is a hard problem for geometrical reasons.

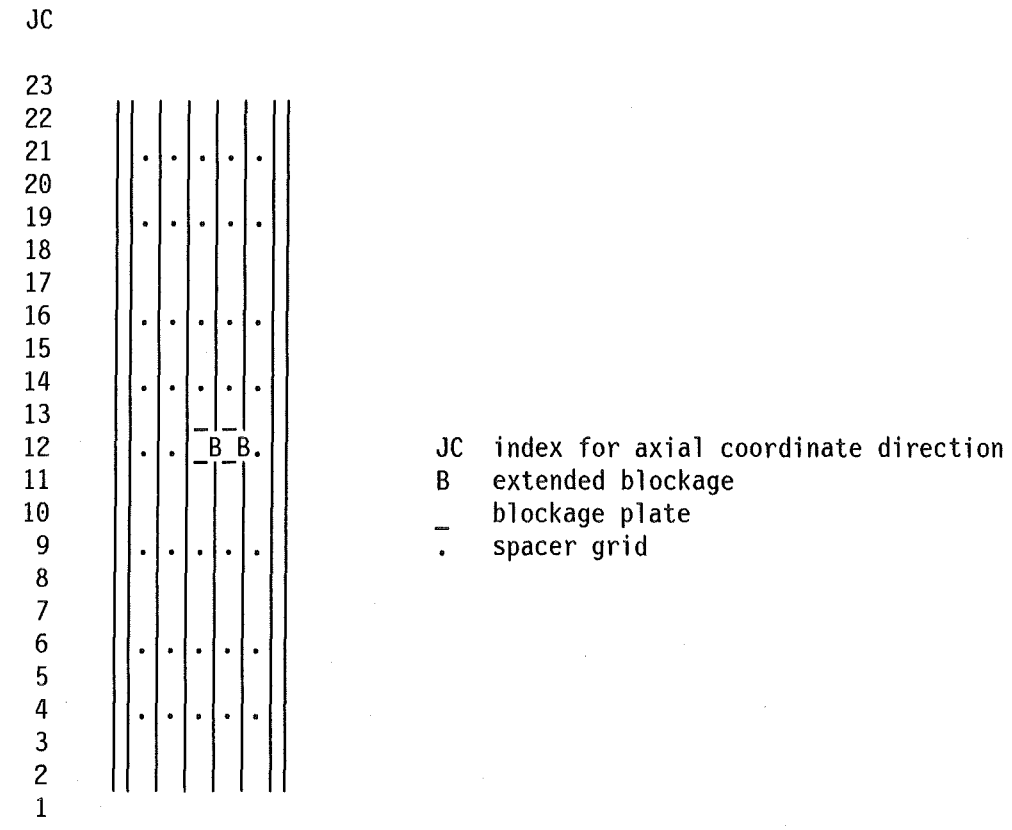

Figure 6. Test case for iterative block method, axial cut. At inlet velocity and temperature are prescribed, at outlet pressure.

7.2 Direct Methods

In BACCHUS both, LU decomposition following the Doolittle method and Gaussian elimination, are implemented. The routines for LU decomposition have been delivered by Centre d'Etudes Nucléaires at Grenoble and are optimized for scalar computers. The programme structure has, however, been found difficult to vectorize [18].

For this reason Gaussian elimination is also implemented. The routine has been taken from SSL2 library available at KfK and has been further optimized for VP50 [18]. For a small problem with 23 axial, 6 azimuthal, and 5 radial meshes it has been seen that even on a scalar computer (IBM 3090-60J) this Gaussian elimination routine is about 20 % faster than LU decomposition.

Storage demands for the code without the solution routines for the systems of linear equations and for the solution routines are listed in Table 1 on page 28 for the three test cases mentioned in the previous section. Storage for LU decomposition is the same as for Gaussian elimination. For the small case storage requirement for Gaussian elimi-

nation is negligible, for the medium case it is about one half of the rest of the code, for the large case it is more than the rest of the code. Further comparison will be given later.

7.3 Iterative Methods

7.3.1 SOR

SOR methods are no longer used because of their bad convergence. When the iterative block method has been developed, however, some investigations have been done on the basis of the test case for the iterative block method (Figure 6 on page 29). In particular it has been tested that the eigenvalues of B_J, calculated from eqs. (A.22) and (A.6) are real and occur in pairs $\pm \lambda$ with $|\lambda| \leq 0.999995507$. This result corresponds to [26]. Since the spectral radius is below unity, the Gauss-Seidel method and underrelaxation converge. The optimum overrelaxation parameter, calculated from eq. (A.43) is $\omega_b = 1.9940$. Generally convergence for our problems is only assured for $\omega \simeq 1$ according to theorems 29 and 34 in Appendix A.

Using a computer program to test Property A and the consistent ordering of a given matrix of coefficients, it has been seen that it is not consistently ordered. By the same way it has been seen that this is a consequence of the azimuthal coupling for IT = 1 and IT = NTH; if this coupling is suppressed, the matrix is consistently ordered. In this case a compatible ordering vector can be constructed in the following manner. The component which corresponds to cell (i,j,k) has the value i+j-1+k-1.

Furthermore it has been tested that because of this azimuthal coupling an indexing of the cells first in the radial and than in the azimuthal direction does not lead to a consistent order, neither. Eqs. (A.43) to (A.46) are not applicable because the matrix of coefficients is not consistently ordered. Only eq. (A.38) is valid.

It can be seen that a ring and hence the whole matrix can be consistently ordered if e. g. for six azimuthal sectors the cells in a ring are not labeled

```
    1                       1
6       2      but      2       3
5       3               4       5
    4                       6
```

This indexing has, however, not been tested. It is possible that the insufficient accuracy which was obtained, when the first two-dimensional version of BACCHUS was extended to a three-dimensional code, was in fact due to a convergence problem as a consequence of the matrix of coefficients not being consistently ordered. Even if the matrix of coeffi-

cients were consistently ordered, ADI method should be much faster [12]so that it is not worthwhile to reactivate the SOR method.

7.3.2 ADI Method

For ADI method experience shows that it can be used with a cyclic variation of the relaxation parameter r for coolant pressure and enthalpy solution. Coolant mass balance is fulfilled to 10^{-10} kg/s instead of 10^{-8} kg/s as it was obtained for the first implementation of the ADI method. Only by this way it is possible to use the ADI method for two-phase cases. For local blockage cases it can also be taken, but for the large case with 120 axial meshes in Table 1 on page 28 convergence was not very satisfactory. The ADI method can, however, well be used for three-dimensional pin temperature calculations.

A good performance can be obtained when for the solution of coolant pressure equation four cycles with 25 steps each are used. For coolant enthalpy and pin temperature one cycle with 30 and 12 to 15 steps, respectively, are taken. Region demands and CPU times on vector processor VP 50 for the BACCHUS calculations are listed in Table 1 on page 28 for the three cases which have already been mentioned. For the large case no value has been given for CPU time for the ADI method because of the poor convergence. In general the ADI method is clearly superior for larger cases as well with respect to storage as to computing times. For the results presented above the old modelling to handle the azimuthal coupling in the ADI method is used, i. e. two sweeps are made with a renumbering of the meshes. As a further improvement this procedure has been replaced by the direct solution of the circular matrix as it is described in Appendix B. It takes about 10 % of CPU time more than one sweep for the standard Thomas algorithm but the second sweep need not be done, and so the solution procedure is faster. Accuracy, measured by mass imbalance, is increased, and it can even be better than with Gaussian elimination when more cycles are performed. Time step can be increased because of better convergence behaviour. These results are only valid if the ADI method is used for the solution of the pressure and the enthalpy equation. If the Runge-Kutta method is used for the solution of the enthalpy equation accuracy may be decreased. As an example Table 2 on page 32 shows a comparison of CPU times on the vector processor VP400 for a case with 40 axial, 6 azimuthal, and 4 radial meshes for a quasi steady state and the begin of a transient calculation. The time step is 40 ms as used for the calculations presented earlier. The begin of this transient is rather steep; otherwise less outer iterations would be necessary for both methods.

	Gauss	ADI old	ADI new
steady state CPU time/s	10.0	19.0	12.5
transient CPU time/Δ t	63.0	210.0	135.0

Table 2. CPU times for a small case on VP400. Gauss means Gaussian elimination. "old" and "new" refer to old and new modelling of azimuthal coupling in a ring according to this section.

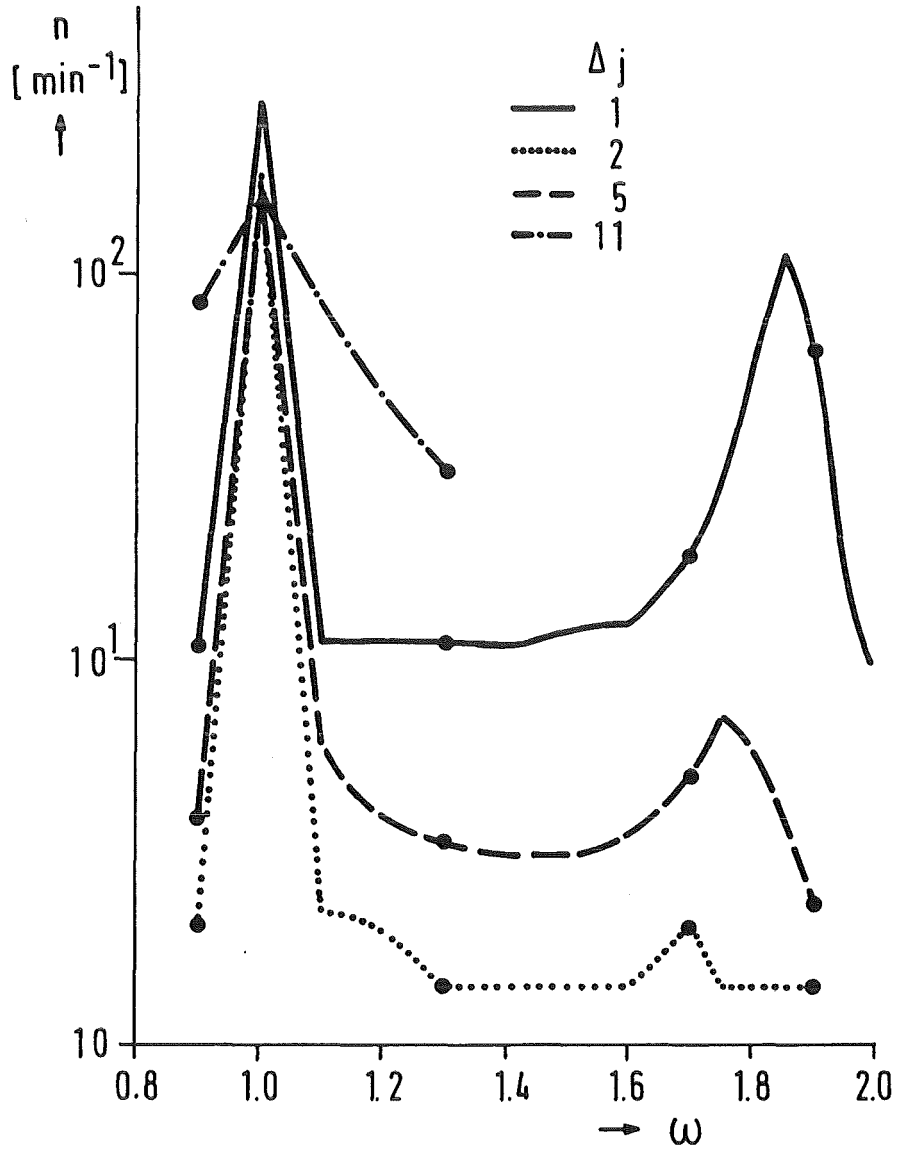

Figure 7. Number of time steps per minute as a function of relaxation parameter

7.3.3 Block Iterative Method

To test the block iterative method runs have been made with the artificial test case explained above. Blocks included one, two, five, and eleven planes, respectively. The last block is generally shorter than the other ones. For the case when every block consists of a single plane it was tested for one time step that for all planes all eigenvalues of B_J are real and occur in pairs $\pm \lambda$ with $|\lambda| < 0.9992$. It was seen that convergence could be reached for all cases. It was further seen that convergence is faster when the block length is increased. Detailed results about convergence behaviour are difficult to obtain because convergence behaviour differs much. When the blocks consist of more than one plane, the calculation converges faster, when for every second iteration the first block has only half of the normal length (see right part of Figure 5 on page 24). In this case the coupling between adjacent blocks is shifted between two iterations.

As an indication for convergence behaviour the number n of time steps per minute of CPU time, which could be calculated on a scalar computer is taken, assuming that the greater n for a given block length, the less iterations per time steps are necessary to reach convergence in the given time step. In Figure 7 on page 32 n as a function of overrelaxation parameter ω is given for various block lengths Δj. For all cases under consideration n has a maximum for $\omega = 0$. It is largest when a block consists of a single plane because in this case the bandwidth of the matrix of coefficients is smaller as stated above. A smaller local maximum exists for $\omega \simeq 1.7$ depending on the block size. Underrelaxation with $\omega = 0.9$ slows down convergence. The results suggest that overrelaxation should not be applied. More experience is necessary to give general guide lines for overrelaxation. Methods to calculate an optimum relaxation factor which are known for point SOR [2] failed for the test case.

7.4 Runge-Kutta Method

For transient calculations enthalpy equation must of course be solved with the same time step Δt used elsewhere in BACCHUS. To increase accuracy the internal step width for Runge-Kutta method may, however, be smaller than Δt. Since there is no well-defined measure of accuracy for Runge-Kutta methods, heuristic procedures as in [25] are necessary to assess the internal step width. In BACCHUS the difference $\vec{k}_3 - \vec{k}_2$ (see eq. (57) is used for this purpose. If one of the components of this vector exceeds 10^5 in value the internal step width for Runge-Kutta method is halved.

In principle the Runge-Kutta method is much faster than Gaussian elimination for all problem sizes of practical interest. However, it has some disadvantages: Though it can be used in principle for the solution of pressure equation many iterations have to be done within a given outer iteration time step: in contrast to enthalpy equation there is

no explicit time derivative of pressure, and the choice which coordinate direction to be retained for explicit derivative similarly to eq. (52) seems arbitrary for general problems. So this method is too slow for pressure solution in practice. In any case accuracy is lower than for Gaussian elimination. So time step is restricted even if a smaller step width is used for Runge-Kutta method within a given outer iteration time step.

As a consequence Runge-Kutta method can only be used for single-phase enthalpy calculations if the transients are not too steep. In such cases CPU time is quite negligible. To extend the range where a method of lines is applicable another solution method than Runge-Kutta would perhaps be better but it would be laborious to try various methods. No method is ideal for all problems [17], [27]. Because of the role of density in eq. (53) the situation might perhaps be improved in the actual programme version if at least density would be updated for the various terms in eq. (57), but for time reasons this has not been tried.

7.5 General remarks

Experience shows that overall convergence behaviour of the code in the outer iteration loop depends on the choice of time step and convergence criteria, hence on parameters which must be chosen not only according to the physical problem but also depending on the solution method. Therefore the results given above indicate trends and should not be mistaken as fundamental constants.

Experience shows further that in principle all solution methods may be combined for the solution of the various systems of linear equations in BACCHUS. However, convergence behaviour is not always the same. Choosing the same solution methods for for pressure and enthalpy equations gives a good convergence behaviour. The same is true when Gaussian elimination or ADI method are combined with Runge-Kutta method for enthalpy equation. If, however, Gaussian elimination is used for pressure equation and ADI method for enthalpy equation, convergence behaviour is less satisfactory.

8.0 Conclusions

The solution of large systems of linear equations plays a large role in computational fluid dynamics. In literature direct as well as iterative solution procedures are documented. There merits depend, however, on the special problems to be solved. In this report some characteristics of the matrices of coefficients used in the computer programme BACCHUS to describe thermal-hydraulic behaviour in a fuel element are pointed out. It turns out that the matrices have some special features which guarantee a unique solution of the system of equations.

The direct solution methods presented in this paper have the great advantage that they can always be used. The disadvantages of LU decomposition and of Gaussian elimination are the large storage and CPU time demands for large problems. They increase with the number of unknowns and the bandwidth of the matrix of coefficients. The advantages of iterative methods presented here are just the disadvantages of direct methods. However, there are open parameters the determination of which is not always simple. As an alternative a method of lines for the solution of the partial differential equation, a Runge-Kutta method, is not generally applicable for reasons of accuracy or overall convergence behaviour of the code.

Up to now no general purpose method exists for the solution of the systems of linear equations which need nearly no storage and CPU time and give a nearly unlimited accuracy. Therefore several solution methods are available in BACCHUS: Gaussian elimination is useful for coolant pressure and enthalpy solution, and a Runge-Kutta method for single-phase enthalpy solution. As an iterative method ADI is recommended for coolant pressure and enthalpy and for pin temperature solution. The advantage of this method is that its accuracy can be varied by the number of cycles. ADI is preferable for many problems, but Gaussian elimination is always a backup for difficult problems.

An iterative block method is implemented in the code but not much experience has been gained up to now. Convergence could not be proved from the theorems listed in this paper, but is suggested in literature.

Solution methods should harmonize: So in a given problem Gaussian elimination can be used for coolant pressure and Runge-Kutta for enthalpy, but Gaussian elimination for coolant pressure and ADI for enthalpy give less satisfactory overall convergence behaviour. In any case convergence behaviour is also influenced by parameters as time step and convergence criteria for the outer iteration loop. As a basis experience from former calculations should be used but it is worthwhile to verify such data from time to time, even for a given problem, because the situation may change in a given transient. This experience and the fact that programme development, e. g. progress in physical modell-

ing, may change convergence behaviour, imply that recommendations cannot be given once and for all.

The status reached up to now is surely not the final situation in a computer code like BACCHUS. Requests sometimes change quickly. So a three-dimensional solution for pin temperature increases the portion of solution of systems of linear equations with respect to total CPU time drastically, especially when more than one pin has to be considered in this way. Besides applications change; their size generally increases and often increases faster than available computer facilities. Therefore there is always a need for good solution methods, and, if possible, for better ones.

Appendix A. Matrix Definitions and Theorems

The items of this section have been collected for various reasons. They can be used as a theoretical background for previous sections of this report. Some additional information is also listed but it was not intended to elaborate a complete compendium about matrices. In the following $A = (a_{ij})$, $B = (b_{ij})$, and $C = (c_{ij})$ are matrices of order n with real coefficients, if not stated otherwise.

A.1 Special Matrices

A.1.1 Permutation Matrices

Definition: A permutation matrix $P = (p_{ij})$ of order n is a matrix with exactly one non-zero element, namely unity, in each row and each column. P corresponds to a permutation function,

$$\sigma(i) = j \qquad i = 1, \ldots, n \qquad \text{where } p_{ij} = 1 \tag{A.1}$$

Theorem 1: If a permutation matrix P corresponds to a permutation σ, then $P^{-1} = P^T$ corresponds to σ^{-1}. Proof: [13], p. 10.

Premultiplication of a matrix A by a permutation matrix permutes rows, postmultiplication permutes columns of A. For an interchange of rows and corresponding columns of a matrix A according to eq. (A.1) the matrix multiplication is done as P^{-1} A P.

A.1.2 Diagonal, Band, and Triangular Matrices

The product C = A B of two matrices A and B is given by

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \tag{A.2}$$

If A is a diagonal matrix, eq. (A.2) reduces to

$$c_{ij} = a_{ii} b_{ij} \tag{A.3}$$

So every column i of B is multiplied by a_{ii}. Similarly, if B is a diagonal matrix eq. (A.2) gives

$$c_{ij} = a_{ij}b_{jj} \tag{A.4}$$

Every row of A is multiplied by b_{jj}.

If in eq. (A.3) C is the unity matrix I with elements δ_{ij} we get $B = A^{-1}$ with

$$c_{ij} = a_{ii}b_{ij} = \delta_{ij} \tag{A.5}$$

hence

$$b_{ij} = \begin{cases} 0 & i \neq j \\ \dfrac{1}{a_{ii}} & i = j \end{cases} \tag{A.6}$$

Definition: A matrix A is a band matrix, if $a_{ij} = 0$ for $|i - j| \geq m$ where m is the band width. A special case is a tridiagonal matrix where $m = 1$.

Theorem 2: The inverse of a band matrix is not necessarily a band matrix (Murphy). A general proof for space matrices is given in [8].

Definition: $L = (l_{ij})$ is a lower diagonal matrix, if $l_{ij} = 0$ for $j > i$. It is a strictly lower diagonal matrix, if $l_{ij} = 0$ for $j \geq i$.

Theorem 3: Let $L = (l_{ij})$ be a lower diagonal matrix. L is nonsingular if and only if $l_{ii} \neq 0$ for $i = 1, \ldots, n$.
Proof: $\det L = \prod_{i=1}^{n} l_{ii}$, as can be seen by mathematical induction when the determinant is developed by rows or columns.

Theorem 4: The inverse of a lower diagonal matrix is a lower diagonal matrix.
Proof: $L\ L^{-1} = I$ is equivalent to n systems of equations

$$L\vec{x}_j = \vec{b}_j \tag{A.7}$$

where $\vec{x}_j$ is the jth column of L^{-1} and $\vec{b}_j$ the jth column of I. For $j = 1, \ldots, n$ the components x_{ij} of $\vec{x}_j$ can be obtained from forward substitution. We get

$$i = 1: \quad \sum_{k=1}^{n} l_{1k}x_{kj} = l_{11}x_{1j} = b_{1j} \tag{A.8}$$

$$i = 2: \quad \sum_{k=1}^{n} l_{2k}x_{kj} = \sum_{k=1}^{2} l_{2k}x_{kj} = l_{21}x_{1j} + l_{22}x_{2j} = b_{2j} \tag{A.9}$$

yields x_{2j}, since x_{1j} has already been determined. In general, when the first i-1 components of $\vec{x}_j$ have already been determined from the first i-1 equations, we get

$$\sum_{k=1}^{n} l_{ik}x_{kj} = \sum_{k=1}^{i} l_{ik}x_{kj} = b_{ij} \tag{A.10}$$

and hence

$$l_{ii}x_{ij} = b_{ij} - \sum_{k=1}^{i-1} l_{ik}x_{kj} \tag{A.11}$$

Since $b_{ij} = \delta_{ij} = 0$ for $i < j$ it can be seen from eqs. (A.8 - A.11) that $x_{ij} = 0$ for $i < j$. For $i=j$ we get $x_{ii} = \frac{1}{l_{ii}}$. For $i > j$ we normally get $x_{ij} \neq 0$.

Theorem 5: Similarly an upper diagonal matrix U is nonsingular if and only if $u_{ii} \neq 0$ for $i = 1, \ldots, n$. Its inverse is an upper diagonal matrix. Proof: U^T, the transposed matrix, is a lower diagonal matrix. Using $(A^{-1})^T = (A^T)^{-1}$ (see [28], chapter 3.1, p. 35) completes the proof.

A.1.3 Convergent Matrices

Definition: A matrix A of order n is convergent (to zero), if $\lim_{n \to \infty} A^n = 0$ where 0 is the zero matrix.

Theorem 6: A complex matrix A of order n is convergent if and only if $S(A) < 1$ where S(A) is the spectral radius of A. Proof: [14], theorem 1.4, p. 13.

A.1.4 Irreducible and Diagonally Dominant Matrices

Definition: A complex matrix A of order $n > 2$ is reducible if there exists a permutation matrix P of order n such that

$$PAP^{-1} = \begin{bmatrix} A_{1,1} & A_{1,2} \\ 0 & A_{2,2} \end{bmatrix} \tag{A.12}$$

where $A_{1,1}$ is a submatrix (or block matrix) of order r and $A_{2,2}$ is a submatrix of order n-r, where $1 \leq r < n$. If no such permutation matrix exists, then A is irreducible.

Definition: A complex matrix A of order n is diagonally dominant, if

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \tag{A.13}$$

Definition: A complex matrix A of order n is strictly diagonally dominant, if strict inequality in eq. (A.13) holds for all $1 \leq i \leq n$. Similarly A is irreducibly diagonally dominant if A is irreducible and diagonally dominant with strict inequality in eq. (A.13) for at least one i.

Theorem 7: Let A be a complex strictly or irreducibly diagonally dominant matrix. Then A is nonsingular. Proof: [14], theorem 1.8, p. 23.

Theorem 8: Let A be a complex strictly or irreducibly diagonally dominant matrix. Then none of the diagonal elements of A vanishes. Proof: [13], theorem 2-5.3, p. 40.

A.1.5 Eigenvalues

Theorem 9: If A is a square matrix of order n with eigenvalues $\lambda_1, \ldots, \lambda_n$, then

$$\det A = \prod_{i=1}^{n} \lambda_i \qquad \text{trace } A = \sum_{i=1}^{n} a_{ii} = \sum_{i=1}^{n} \lambda_i \tag{A.14}$$

Proof: [13], theorem 2-1.7, p. 14.

Theorem 10: If A is a strictly or irreducibly diagonally dominant complex matrix where all diagonal entries are real and positive, then the eigenvalues λ_i of A satisfy

$$\text{Re } \lambda_i > 0 \quad 1 \leq i \leq n \tag{A.15}$$

Proof: [14], theorem 1.8, p. 23.

A.1.6 Hermitian and Positive Definite Matrices

Definition: For a matrix $A = (a_{ij})$ we define $A^H = (a_{ji}^*)$, where '*' stands for 'complex conjugate'.

Definition: A matrix A is Hermitian if $A = A^H$.

Definition: If furthermore A is a real matrix, A is symmetric.

Theorem 11: A matrix A of order n is an Hermitian matrix if and only if $\vec{v}A\vec{v}$ is real for all vectors $\vec{v}$ of the corresponding vector space of order n. Proof: [13], theorem 2-2.1, p. 18.

Theorem 12: For any matrix L the matrix $L\,L^H$ is Hermitian and nonnegative definite. If L is nonsingular, then $L\,L^H$ is positive definite. Proof: [13], theorem 2-2.6, p. 22.

Theorem 13: A matrix A is positive definite (nonnegative definite) if and only if it is Hermitian and all of its eigenvalues are positive (nonnegative). Proof: [13], p. 21.

Theorem 14: Let A be strictly or irreducibly diagonally dominant and Hermitian with positive real diagonal entries. Then A is positive definite. Proof: [14], corollary to theorem 1.8, p. 23.

A.1.7 L-Matrices and Related Matrices

Definition: A matrix A of order n is an L-matrix if

$$a_{ii} > 0 \quad i = 1, \ldots, n \tag{A.16}$$

$$a_{ij} \leq 0 \quad i \neq j, \quad i, j = 1, \ldots, n \tag{A.17}$$

Definition: A is an M-matrix if A is nonsingular, $A^{-1} \geq 0$, and if eq. (A.17) holds.

Theorem 15: Any M-matrix is an L-matrix. Proof: [13], p. 43.

Theorem 16: If A is an irreducibly diagonally dominant L-matrix, it is an M-matrix. Proof: [14], corollary 1, p. 85.

Definition: A is a Stieltjes matrix if A is positive definite and if eq. (A.17) holds.

Theorem 17: If A is a Stieltjes matrix, then it is also an M-matrix. Proof: [14], corollary 3, p. 85.

Theorem 18: If A is a symmetric and nonsingular irreducible matrix, where $a_{ij} \leq 0$ for all $i \neq j$ then $A^{-1} > 0$ if and only if A is positive definite. Proof: [14], corollary 2, p. 85.

A.1.8 Property A and Consistent Ordering

Definition: A matrix A of order n has Property A if there exist two disjoint subsets S_1 and S_2 of W, the set of the first n positive integers, such that $S_1 + S_2 = W$ and such that if $i \neq j$ and if either $a_{ij} \neq 0$ or $a_{ji} \neq 0$, then $i \in S_1$ and $j \in S_2$ or else $i \in S_2$ and $j \in S_1$.

Definition: For a given matrix A the integers i and j are associated with respect to A if $a_{ij} \neq 0$ or $a_{ji} \neq 0$.

Definition: The matrix A of order n is consistently ordered (A is a CO-matrix) if for some t there exist disjoint subsets $S_1, \ldots, S_t$ of W, the first n integers, such that $\sum_{k=1}^{t} S_k = W$ and such that if i and j are associated, then $j \in S_{k+1}$ if $j > i$ and $j \in S_{k-1}$ if $j < i$, where S_k is the subset containing i.

Definition: The vector $\gamma = (\gamma_1, \ldots, \gamma_n)^T$ where $\gamma_1, \ldots, \gamma_n$ are integers, is an ordering vector for the matrix A of order n if for any pair of associated integers i and j with $i \neq j$ we have $|\gamma_i - \gamma_j| = 1$.

Definition: An ordering vector $\gamma = (\gamma_1, \ldots, \gamma_n)^T$ for the matrix A of order n is a compatible ordering vector for A if
a) $\gamma_i - \gamma_j = 1$ if i and j are associated and $i > j$
b) $\gamma_i - \gamma_j = -1$ if i and j are associated and $i < j$.
In sum $\gamma_i - \gamma_j = \text{sgn}(i - j)$ if $a_{ij} \neq 0$.

Theorem 19: If A has Property A, then the eigenvalues of B_J are either zero or occur in pairs $\pm \lambda_i$. see [26], p. 121.

Theorem 20: If A has Property A, then for any permutation matrix P the matrix $A' = P^{-1} A P$ has Property A. Proof: [13], theorem 5-4.3, p. 149.

Theorem 21: There exists an ordering vector for a matrix A if and only if A has Property A. Moreover, if A is consistently ordered, then A has Property A. Proof: [13], theorem 5-4.1, p. 148.

Theorem 22: If A has Property A, then there exists an ordering vector whose components have at most two different values. Proof: [13], corollary 5-4.2, p. 148.

Theorem 23: A matrix A has Property A if and only if there exists a permutation matrix P such that the matrix $A' = P^{-1} A P$ is consistently ordered. Proof [13], theorem 5-4.5, p. 150.

Theorem 24: A matrix A of order n is consistently ordered if and only if there exists a compatible ordering vector for A. Proof: [13], theorem 5-3.2, p. 146.

Definition: A is a T-matrix if A has a block-tridiagonal form with square diagonal matrices on its diagonal. So A can be decomposed into submatrices A_{ij} where all A_{ii} are square matrices and $A_{ij} = 0$ for $|i-j| > 1$.

Theorem 25: If A is a T-matrix, then A is consistently ordered. Proof: [13], theorem 5-3.1, p. 145. Especially a tridiagonal matrix of order n is consistently ordered. An ordering vector γ has the form $\gamma = (1, 2, \ldots, n)^T$.

A.2 Iterative Solution of Linear Equations

To solve a system of linear algebraic equations

$$A\vec{x} = \vec{b} \tag{A.18}$$

we can use an iterative method

$$\begin{aligned} &\vec{x}^{(0)} = \Phi_0(A, b) \\ &\vec{x}^{(n+1)} = \Phi_{n+1}(\vec{x}^{(0)}, \ldots, \vec{x}^{(n)}, A, b) \quad n = 0, 1, 2, 3, \ldots \end{aligned} \tag{A.19}$$

Superscripts indicate an iteration index. If for some integer $s > 0$ Φ_n is independent of n for all $n \geq s$, then the method is said to be stationary. The degree of a stationary method is t for $t \leq s$ if, for $n \geq s - 1$, $\vec{x}^{(n+1)}$ depends on $\vec{x}^{(n)}, \vec{x}^{(n-1)}, \ldots, \vec{x}^{(n-t+1)}$ but not on $\vec{x}^{(k)}$ for $k < n\text{-}t+1$. If, for each n, Φ_n is a linear function of $\vec{x}^{(0)}, \ldots, \vec{x}^{(n-1)}$ the method is said to be linear. In the case of a linear stationary iterative method of first degree, the iteration has the form

$$\vec{x}^{(n+1)} = B\vec{x}^{(n)} + \vec{c} \tag{A.20}$$

for some matrix B and for some vector $\vec{c}$.

Definition: The iterative method (A.20) is weakly convergent if for all starting vectors $\vec{u}^{(0)}$ the sequence $\vec{u}^{(0)}, \vec{u}^{(1)} \ldots$ of iterative solutions of eq. (A.20) converges. The method is convergent if for all $\vec{u}^{(0)}$ the sequence converges to a limit independent of $\vec{u}^{(0)}$.

Using conventions as in [14], we decompose A as follows

$$A = D + L + U \tag{A.21}$$

where D is a diagonal matrix, L and U are strictly lower and upper triangular matrices.

For the Jacobi method

$$B_J = -D^{-1}(L+U) \qquad \vec{c} = D^{-1}\vec{b} \tag{A.22}$$

are used. Evidently B_J has zero diagonal entries.

For the Gauss-Seidel method we set

$$B_{GS} = -(L+D)^{-1}U \qquad \vec{c} = (L+D)^{-1}\vec{b} \tag{A.23}$$

For overrelaxation

$$B_{SOR} = -\left(L+\frac{1}{\omega}D\right)^{-1}\left[U+\left(1-\frac{1}{\omega}\right)D\right] \qquad \vec{c} = \left(L+\frac{1}{\omega}D\right)^{-1}\vec{b} \tag{A.24}$$

holds. To facilitate comparison with literature, the definition of matrices used in [13] is given in the following section.

A.2.1 Definition of Iteration Matrices

[13] and [14] are standard textbooks for iterative methods as SOR. Since [14] is used in this report for the definition of matrices for the Jacobi, the Gauss-Seidel, and SOR method, the definitions of [13] are given hereafter to facilitate comparison of formulas in the various publications.

In [13] the matrix of coefficients A for a system of linear equations

$$A\vec{x} = \vec{b} \tag{A.25}$$

is decomposed to be

$$A = D - C_y \tag{A.26}$$

$$C_y = C_L + C_U \tag{A.27}$$

where the indices L and U mean strictly upper and lower matrices. Here and in the following the index y stands for Young.

$$B_y = D^{-1} C_y \tag{A.28}$$

$$L_y = D^{-1} C_L \tag{A.29}$$

$$U_y = D^{-1} C_U \tag{A.30}$$

The iteration matrix for the SOR method in [13] is

$$B_{SOR} = (I - \omega L_y)^{-1} [\omega U_y + (1 - \omega)I] \tag{A.31}$$

From a comparison of eqs. (A.26) and (A.27) with eq. (A.21) we find

$$L = -C_L \quad U = -C_U \tag{A.32}$$

From eq. (A.31), using eqs. (A.28) to (A.30) we get

$$(I + \omega D^{-1} L)\, \vec{u}^{(n+1)} = [(1 - \omega)I - \omega D^{-1} U]\, \vec{u}^{(n)} + \omega D^{-1} \vec{b} \tag{A.33}$$

Multiplication of eq. (A.33) with D gives

$$(D + \omega L)\, \vec{u}^{(n+1)} = [(1 - \omega)D - \omega U]\, \vec{u}^{(n)} + \omega \vec{b} \tag{A.34}$$

Dividing by ω yields eq. (A.24). So the formulas in [14] and [13] are equivalent as it should be.

A.2.2 General Theorems

Theorem 26: The iterative method eq. (A.20) converges if and only if

$$S(B) < 1 \tag{A.35}$$

where S(B) is the spectral radius of B. Proof: [13], theorem 3-5.1, p. 77.

Theorem 27:

$$S(B_{SOR}) \geq |\omega - 1| \tag{A.36}$$

with equality only if all eigenvalues of B_{SOR} are of modulus $|\omega - 1|$ (Kahan). Proof: [14], theorem 3.5, p. 75.

Theorem 28: Moreover, if the SOR method converges, then

$$0 < \omega < 2 \tag{A.37}$$

Proof: [13], theorem 4-1.2, p. 107.

A.2.3 Convergence for Special Matrices

Theorem 29: Let A be an irreducible matrix with weak diagonal dominance. Then
a) The Jacobi method converges.
b) The Gauss-Seidel method converges.
c) The SOR method converges for $0 < \omega \leq 1$.
Proof: [13], theorem 4-2.1, p. 107.

Theorem 30: Let A be a Hermitian matrix of order n, where $a_{ii} > 0$ and $D + \omega L$ is nonsingular for $0 \leq \omega \leq 2$. Then $S(B_{SOR}) < 1$ if and only if A is positive definite and $0 < \omega < 2$. Proof: [14], theorem 3.6, p. 77.

Theorem 31: Let A be a symmetric matrix with positive diagonal elements. Then the SOR method converges if and only if A is positive definite and $0 < \omega < 2$. Proof: [13], theorem 4-3.6, p. 113.

Theorem 32: If A is an L-matrix, then A is an M-matrix if and only if $S(B_J) < 1$. Proof: [13], theorem 2-7.2, p. 43.

Theorem 33: A is an M-matrix, if and only if B_J is non-negative, irreducible, and convergent. Proof: [14], theorem 3.11, p. 84.

Theorem 34: If A is an M-matrix and if

$$0 < \omega < \frac{2}{1 + S(B_J)} \tag{A.38}$$

then $S(B_{SOR}) < 1$. Proof: [13], theorem 4-5.9, p. 126.

A.2.4 Convergence for Consistently Ordered Matrices

Theorem 35: If A is a consistently ordered matrix, then

$$\det(\alpha L + \alpha^{-1}U - kD) \tag{A.39}$$

is independent of α for all $\alpha \neq 0$ and for all k. Proof: [13], theorem 5-3.3, p. 147.

Theorem 36: If A is consistently ordered and if A has nonvanishing diagonal elements, then
a) if μ is any eigenvalue of B_J of multiplicity p, then - μ is also an eigenvalue of B_J of multiplicity p.
b) λ satisfies

$$(\lambda + \omega - 1)^2 = \omega^2\mu^2\lambda \tag{A.40}$$

for some eigenvalue μ of B_J if and only if λ satisfies

$$\lambda + \omega - 1 = \omega\mu\sqrt{\lambda} \tag{A.41}$$

for some eigenvalue μ of B_J.
c) if λ satisfies either, and hence both of the relations eqs. (A.40) and (A.41), then λ is an eigenvalue of B_{SOR}.
d) If λ is an eigenvalue of B_{SOR}, then there exists an eigenvalue μ of B_J such that (A.40) and (A.41) hold. Proof: [13], theorem 5-3.4, p. 147.

Theorem 37: Let the Jacobi matrix B_J be a non-negative matrix of order n. Then, one and only one of the following mutually exclusive relations is valid:

$$\begin{aligned} &S(B_J) = S(B_{GS}) = 0 \\ &S(B_{GS}) < S(B_J) < 1 \\ &S(B_J) = S(B_{GS}) = 1 \\ &1 < S(B_J) < S(B_{GS}) \end{aligned} \tag{A.42}$$

Thus, the Jacobi matrix and the Gauss-Seidel matrix are either both convergent, or both divergent (Stein and Rosenberg). Proof: [14], theorem 3.3, p. 70.

Theorem 38: If A is a consistently ordered matrix with nonvanishing diagonal elements such that B_J has real eigenvalues, then $S(B_{SOR}) < 1$ if and only if $0 < \omega < 2$ and $S(B_J) < 1$. Proof: [13], theorem 6-2.2, p. 172.

Theorem 39: Let A be a consistently ordered matrix with non-vanishing diagonal elements such that the matrix B_J has real eigenvalues and such that $\bar{\mu} = S(B_J) < 1$. If

$$\omega_b = \frac{2}{1 + \sqrt{1 - \bar{\mu}^2}} \tag{A.43}$$

then

$$S(B_{SOR}) = \omega_b - 1 \tag{A.44}$$

and if $\omega \neq \omega_b$ then

$$S(B_{SOR}) > S(B_{SOR}(\omega_b)) \tag{A.45}$$

Moreover, for any ω in the range $0 < \omega < 2$ we have

$$S(B_{SOR}) = \begin{cases} \left[\dfrac{\omega\bar{\mu} + \sqrt{\omega^2\bar{\mu}^2 - 4(\omega - 1)}}{2} \right]^2 & \text{if } 0 < \omega \leq \omega_b \\ \omega - 1 & \text{if } \omega_b \leq \omega < 2 \end{cases} \tag{A.46}$$

Finally, if $0 < \omega < \omega_b$ then $S(B_{SOR})$ is a strictly decreasing function of ω. Proof: [13], theorem 6-2.3, p. 172.

Theorem 40: If A is a positive definite consistently ordered matrix, then the matrix B_J has real eigenvalues. Moreover, $S(B_J) < 1$. Proof: [13], lemma 6-2.5, p. 175.

Theorem 41: Let A be a matrix which has Property A and whose diagonal elements are positive. If, for some diagonal matrix E the matrix $A' = E\,A\,E^{-1}$ is positive definite, then B_J has real eigenvalues and $S(B_J) < 1$. Proof: [13], theorem 6-2.6, p. 175.

A.3 Block Methods

A.3.1 Concept

The following section is summarized on the basis of [13], chapter 14. Arrows, indicating vectors, have been dropped in this section.

An ordered grouping π of W, the first n integers, is a subdivision of W into disjoint subsets $R_1, \ldots R_q$, such that $R_1 + \ldots + R_q = W$. Two ordered groupings π and π',

defined by $R_1, \ldots R_q$ and $R'_1, \ldots, R'_q$, respectively, are identical if $q = q'$ and if $R_1 = R'_1, \ldots, R_q = R'_q$.

For a given matrix A and an ordered grouping π, consisting of q groups, we define the submatrices $A_{r,s}$ for r, s = 1, 2, q as follows: $A_{r,s}$ is formed from A by deleting all rows except those corresponding to R_r and all columns except those corresponding to R_s. Given a column vector u we define column vectors $U_1, \ldots, U_q$ where U_r is formed from u by deleting all elements of u except those corresponding to R_r. Similarly we define column vectors $B_1, \ldots, B_q$ given the column vector b. For example, , if n = 3 and π is defined by $R_1 = \{1, 3\}$, $R_2 = \{2\}$, we have

$$A_{1,1} = \begin{pmatrix} a_{1,1} & a_{1,3} \\ a_{3,1} & a_{3,3} \end{pmatrix}, \quad A_{1,2} = \begin{pmatrix} a_{1,2} \\ a_{3,2} \end{pmatrix}, \quad U_1 = \begin{pmatrix} u_1 \\ u_3 \end{pmatrix}, \quad B_1 = \begin{pmatrix} b_1 \\ b_3 \end{pmatrix} \tag{A.47}$$

$$A_{2,1} = (a_{2,1}\ a_{2,3}), \quad A_{2,2} = (a_{2,2}), \quad U_2 = (u_2), \quad B_2 = (b_2)$$

so that the system of equations $A\vec{u} = \vec{b}$ becomes

$$\begin{pmatrix} a_{1,1} & a_{1,3} \\ a_{3,1} & a_{3,3} \end{pmatrix} \begin{pmatrix} u_1 \\ u_3 \end{pmatrix} + \begin{pmatrix} a_{1,2} \\ a_{3,2} \end{pmatrix} (u_2) = \begin{pmatrix} b_1 \\ b_3 \end{pmatrix} \tag{A.48}$$

$$(a_{2,1}\ a_{2,3}) \begin{pmatrix} u_1 \\ u_3 \end{pmatrix} + (a_{2,2})(u_2) = (b_2)$$

which is equivalent to the original system. In general the system of equations can be written in the form

$$\sum_{s=1}^{q} A_{r,s} U_s = B_r \quad r = 1, 2, \ldots, q \tag{A.49}$$

If $A_{r,r}$ is nonsingular for all r, the group Jacobi method can be written in the form

$$A_{r,r} U_r^{(n+1)} + \sum_{\substack{s=r \\ s \neq r}}^{q} A_{r,s} U_s^{(n)} = B_r \tag{A.50}$$

or, equivalently

$$U_r^{(n+1)} = \sum_{\substack{s=r \\ s \neq r}}^{q} B_{r,s} U_s^{(n)} + C_r \tag{A.51}$$

where

$$B_{r,s} = \begin{cases} -A_{r,r}^{-1} A_{r,s} & \text{if } r \neq s \\ 0 & \text{if } r = s \end{cases} \tag{A.52}$$

Evidently, we may write eq. (A.51) in the matrix form

$$u^{(n+1)} = B_J^{(\pi)} u^{(n)} + c^{(\pi)} \tag{A.53}$$

where

$$B_J^{(\pi)} = \left(D^{(\pi)}\right)^{-1} C^{(\pi)} \tag{A.54}$$

$$c^{(\pi)} = \left(D^{(\pi)}\right)^{-1} b \tag{A.55}$$

$$C^{(\pi)} = D^{(\pi)} - A \tag{A.56}$$

$$D^{(\pi)} = \text{diag}_{\pi} A \tag{A.57}$$

where $\text{diag}_{\pi} A$ is the matrix formed from A by replacing by zeros all $a_{i,j}$ unless i and j belong to the same group. Examples are given in [13], chapter 14.1.

The group Gauss-Seidel method can be written in the matrix form

$$u^{(n+1)} = B_{GS}^{(\pi)} u^{(n)} + (I - L^{(\pi)})^{-1} c^{(\pi)} \tag{A.58}$$

where

$$B_{GS}^{(\pi)} = (I - L^{(\pi)})^{-1} U^{(\pi)} \tag{A.59}$$

$$L^{(\pi)} = (D^{(\pi)})^{-1} C_L^{(\pi)} \tag{A.60}$$

$$U^{(\pi)} = (D^{(\pi)})^{-1} C_U^{(\pi)} \tag{A.61}$$

$C_L^{(\pi)}$ and $C_U^{(\pi)}$ are formed from A by replacing all elements of A by zero except those $a_{i,j}$ such that i and j belong to different groups and such that the group containing i comes after and before, respectively, the group containing j.

The matrix form of the group SOR method is

$$u^{(n+1)} = B_{SOR}{}^{(\pi)} u^{(n)} + (I - \omega L^{(\pi)})^{-1} \omega c^{(\pi)} \tag{A.62}$$

where

$$B_{SOR}{}^{(\pi)} = (I - \omega L^{(\pi)})^{-1} [\omega U^{(\pi)} + (1 - \omega) I] \tag{A.63}$$

A.3.2 Theorems

Given a matrix A and an ordered grouping π with q groups, we define the matrix Z of order q by

$$z_{r,s} = \begin{cases} 0 & \text{if } A_{r,s} = 0 \\ 1 & \text{if } A_{r,s} \neq 0 \end{cases} \tag{A.64}$$

The matrix A has Property $A^{(\pi)}$ if Z has Property A.

The matrix A is a π-consistently ordered matrix (a $\pi - CO -$ matrix) if Z is consistently ordered.

The matrix $\hat{A}^{(\pi)}$ is obtained from A by reordering the rows and columns of A according to the ordered grouping π using the submatrices:

$$\hat{A}^{\pi} = P^{-1} A P = \begin{pmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,q} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,q} \\ \cdots & \cdots & \cdots & \cdots \\ A_{q,1} & A_{q,2} & \cdots & A_{q,q} \end{pmatrix} \tag{A.65}$$

where $P = P^{(\pi)}$ is the corresponding permutation matrix.

Theorem 42: If A has Property $A^{(\pi)}$ then $C^{(\pi)}$ has Property A, and if $D^{(\pi)}$ is nonsingular, $B_J{}^{(\pi)}$ has Property A. If A is a $\pi - CO -$ matrix, then $\hat{C}^{(\pi)}$ is a CO-matrix, and if $D^{(\pi)}$ is nonsingular, $\hat{B}^{(\pi)}$ is a CO-matrix. Proof: [13], theorem 14-3.1, p. 446.

Theorem 43: Let A be a matrix and π an ordered grouping such that $\hat{C}^{(\pi)}$ is consistently ordered. Then

$$\Delta = \det(\alpha C_L^{(\pi)} + \alpha^{-1} C_U^{(\pi)} - kD^{(\pi)}) \tag{A.66}$$

is independent of α for all $\alpha \neq 0$ and for all k. Proof: [13], theorem 14-3.2, p. 448.

A matrix A is a generalized π-consistently ordered matrix (a $\pi - GCO -$ matrix) if Δ, given by eq. (A.66), is independent of α for all $\alpha \neq 0$ and for all k.

Theorem 44: If A is a $\pi - CO -$ matrix, then A is a $\pi - GCO -$ matrix. More generally, if $\hat{C}^{(\pi)}$ is a CO-matrix or if $D^{(\pi)}$ is nonsingular and $\hat{B}^{(\pi)}$ is a CO-matrix, then A is a $\pi - GCO -$ matrix. Proof: [13], theorem 14-3.3, p. 449.

Theorem 45: If A is a $\pi - GCO -$ matrix such that $D^{(\pi)}$ is nonsingular, then the conclusions of theorem 36 are valid, if we replace B_J by $B_J{}^{(\pi)}$ and B_{SOR} by $B_{SOR}{}^{(\pi)}$ (see [13], theorem 14-3.4, p. 451.

Theorem 46: If A is a $\pi - GCO -$ matrix which is an irreducible M-matrix, then $\hat{B}^{(\pi)}$ and $\hat{C}^{(\pi)}$ and are CO-matrices. Proof: [13], theorem 14-3.5, p. 451.

Appendix B. Special Systems of Equations

B.1 Tridiagonal Systems of Equations

Gaussian elimination is very simple for tridiagonal matrices. The procedure is well-known in literature and attributed to Thomas. It is recalled in this section and used as a basis for the solution of cyclic tridiagonal systems (see next section), i. e. for systems of linear equations which differ from tridiagonal systems by a_{1n}, $a_{n1} \neq 0$.

Given a tridiagonal system of equations

$$\begin{aligned} a_{11}\ y_1 \quad &+ a_{12}\ y_2 \qquad\qquad\qquad\quad &&= b_1 \\ a_{ii-1}\, y_{i-1} &+ a_{ii}\ y_i \quad + a_{ii+1}\, y_{i+1} &&= b_i \qquad i = 2, \ldots, n-1 \\ &\quad a_{nn-1}\, y_{n-1} + a_{nn}\ y_n &&= b_n \end{aligned} \tag{B.1}$$

of n equations with n unknowns. The matrix of coefficients has the form

```
x x
x x x
  x x x
    ...
      x x x
        x x x
          x x
```

We suppose that at least for one equation $b_i \neq 0$ and that the matrix of coefficients is nonsingular so that there is a unique solution of eq. (B.1). Gaussian elimination is equivalent to transforming equation i (i = 1, 2, ... , n - 1) into an equation i'

$$y_i = b'_i - a'_{ii+1}\, y_{i+1} \tag{B.2}$$

and substituting into equation i+1. We obtain

$$\begin{aligned} a'_{12} &= \frac{a_{12}}{a_{11}} \\ b'_1 &= \frac{b_1}{a_{11}} \end{aligned} \tag{B.3}$$

$$a'_{ii} = a_{ii} - a_{ii-1}\, a'_{i-1\,i}$$

$$a'_{ii+1} = \frac{a_{ii+1}}{a'_{ii}}$$

$$b'_i = \frac{b_i - a_{ii-1}\, b'_{i-1}}{a'_{ii}}$$

for i = 2 , ... , n - 1. Proof of eqs. (B.3) is by mathematical induction. The same coefficients a′ and b′ would of course result from a LU decomposition of the original matrix of coefficients [17]. Substitution fails for $a_{11} = 0$ or $a'_{ii} = 0$. This situation is discussed in [24], but is not interesting for BACCHUS. Substituting equation (n-1)′ into equation n yields y_n to be

$$y_n = \frac{b_n - a_{nn-1}\, b'_{n-1}}{a_{nn} - a_{nn-1}\, a'_{n-1\,n}} \tag{B.4}$$

and for i = n - 1 , ... , 1 one obtains y_i from eq. (B.2) by backsubstitution.

The advantage of the solution algorithm of tridiagonal system is that y_1 exists only in the first and the second equation. Once it has been substituted in the second equation, y_2 exists only in the second and the third equation and so on. This can be programmed without a DO-loop and so gives a quick solution routine. In practice a′ and b′ are stored thus needing only little additional storage.

B.2 Cyclic Systems of Equations

Cyclic systems of equations are an extension of tridiagonal systems, i. e. in addition to tridiagonal systems a_{1n}, $a_{n1} \neq 0$. In BACCHUS such systems of linear equations arise because of the coupling of the first and the last control volume in a ring.

A solution procedure can be used similarly to the Thomas algorithm for tridiagonal systems of equations. In contrast to the Thomas algorithm an equation is not only combined with the following one but also with the last one. In the following this procedure is described in detail.

Given a system of equations

$$\begin{array}{llllllll} a_{11} & y_1 & + a_{12} & y_2 & + a_{1n} & y_n & = b_1 & \\ a_{ii-1} & y_{i-1} & + a_{ii} & y_i & + a_{ii+1} & y_{i+1} & = b_i & i = 2, \ldots, n-1 \\ a_{n1} & y_1 & + a_{nn-1} & y_{n-1} & + a_{nn} & y_n & = b_n & \end{array} \tag{B.5}$$

similarly to eq. (B.1), but a_{1n} and a_{n1} are allowed to be nonzero. So the matrix of coefficients has the form

```
X X         X
X X X
  X X X
    ...
      X X X
        X X X
X         X X
```

The first equation is divided by a_1 to yield

$$y_1 = b'_1 - a'_{12} y_2 - \bar{a}_{1n} y_n \tag{B.6}$$

with

$$\begin{aligned} a'_{12} &= \frac{a_{12}}{a_{11}} \\ \bar{a}_{1n} &= \frac{a_{1n}}{a_{11}} \\ b'_1 &= \frac{b_1}{a_{11}} \end{aligned} \tag{B.7}$$

and introduced into the second and the last equation of system (B.5).

In the i-th substitution step (i = 2, ... , n - 1) equation i of system (B.5) is transformed into an equation i'

$$y_i = b'_i - a'_{ii+1} y_{i+1} - \bar{a}_{in} y_n \tag{B.8}$$

with

$$\begin{aligned} a'_{ii} &= a_{ii} - a_{ii-1} a'_{i-1i} \\ a'_{ii+1} &= \frac{a_{ii+1}}{a'_{ii}} \\ \bar{a}_{in} &= -\frac{a_{ii-1} \bar{a}_{i-1n}}{a'_{ii}} \\ b'_i &= \frac{b_i - a_{ii-1} b'_{i-1}}{a'_{ii}} \end{aligned} \tag{B.9}$$

and introduced into equation i+1.

Furthermore at the beginning of the ith substitution step the last equation (number n) of the original system (B.5) has been transformed to form equation $n^{(i)}$

$$a'_{ni}\, y_i + a_{n\,n-1}\, y_{n-1} + a^{(i)}_{nn}\, y_n = b^{(i)}_n \tag{B.10}$$

with

$$\begin{aligned} a'_{ni} &= -a'_{n\,i-1}\, a'_{i-1\,i} \\ a^{(i)}_{nn} &= a^{(i-1)}_{nn} - a'_{n\,i-1}\, \bar{a}_{i-1\,n} \\ b^{(i)}_n &= b^{(i-1)}_n - a'_{n\,i-1}\, b'_{i-1} \end{aligned} \tag{B.11}$$

For eq. (B.11) we set

$$\begin{aligned} a'_{n1} &= a_{n1} \\ a^{(1)}_{nn} &= a_{nn} \\ b^{(1)}_n &= b_n \end{aligned} \tag{B.12}$$

Proof of eqs. (B.9) and (B.11) for the new coefficients can be done by mathematical induction. For $a_{1n} = a_{n1} = 0$ eqs. (B.7), (B.8) and (B.9) give eqs. (B.2) and (B.3), and eqs. (B.10) and (B.11) are dropped. During the i-th substitution step eq. (B.8) is introduced into eq. (B.10) for i = 2, ... , n - 2.

For i = n - 1 we get by this forward elimination from eqs. (B.8) and (B.10)

$$y_{n-1} = b'_{n-1} - (a'_{n-1\,n} + \bar{a}_{n-1\,n})\, y_n \tag{B.13}$$

$$(a'_{n\,n-1} + a_{n\,n-1})\, y_{n-1} + a^{(n-1)}_{nn}\, y_n = b^{(n-1)}_n$$

which gives

$$y_n = \frac{b^{(n-1)}_n - (a'_{n\,n-1} + a_{n\,n-1})\, b'_{n-1}}{a^{(n-1)}_{nn} - (a'_{n\,n-1} + a_{n\,n-1})(a'_{n-1\,n} + \bar{a}_{n-1\,n})} \tag{B.14}$$

The solution scheme is completed by backsubstitution. y_{n-1} is calculated of one the two equations of (B.13). For i = n-2(1)1 several strategies are possible.

1. y_i is calculated from eq. (B.8).

2. y_i is calculated from eq. (B.10).

3. y_i is calculated from the original system (B.5).

In the first case the coefficients of eq. (B.9) must be stored, in the second case those of eq. (B.11). In the third case no additional arrays are available. However, for the second and the third case one additional division must be made for every equation. In practice the first strategy is used in BACCHUS, because the Thomas algorithm is already available. The solution procedure described above has been derived independently of [17].

References

[1] M. Bottoni, B. Dorr, Ch. Homann, D. Struwe: BACCHUS-3D/SP, a Computer Programme to Describe Transient Three-Dimensional Single Phase Flow in LMFBR Rod Bundles, Nuclear Technology 71 (1985), 43-67.

[2] M. Bottoni, B. Dorr, Ch. Homann, D. Struwe: BACCHUS-3D/SP, a Computer Programme for Three-Dimensional Description of Sodium Single Phase Flow in Bundle Geometry, KfK 3376, July 1983.

[3] M. Bottoni, B. Dorr, Ch. Homann: The Three-Dimensional Transient Two-Phase Flow Computer Programme BACCHUS-3D/TP, KfK 4760, April 1992.

[4] Ch. Homann, M. Bottoni, B. Dorr, D. Struwe: Prediction of Thermal-Hydraulic Behaviour of Wall Blockages in a Fuel Subassembly with the Code BACCHUS-3D, Fourth International Topical Meeting on Nuclear Reactor Thermal-Hydraulics (NURETH-4), October 10-13, 1989, Karlsruhe, West Germany, pp. 1176 - 1182.

[5] M. Bottoni, B. Dorr, Ch. Homann, F. Huber, K. Mattes, W. Peppler, D. Struwe: Experimental and Numerical Investigations of Sodium Boiling Experiments in Pin Bundle Geometry, Nuclear Technology 89 (1990), pp. 56-82.

[6] Ch. Homann, B. Dorr: Prediction of Thermal-Hydraulic Behaviour of Central Blockages in an EFR Breeder Subassembly, Fifth International Topical Meeting on Nuclear Reactor Thermal-Hydraulics (NURETH-5), September 21-24, 1992. Salt Lake City, Utah, USA, pp. 242 - 249.

[7] C. Günther: Vergleich verschiedener Differenzenverfahren zur numerischen Lösung der 2-d-Konvektions-Diffusionsgleichung anhand eines Beispiels mit bekannter exakter Lösung, KfK 4439, August 1988.

[8] I. S. Duff, A. M. Erisman, J. K. Reid: Direct Methods for Sparse Matrices, Clarendon Press, Oxford, 1986.

[9] J. Stoer: Einführung in die Numerische Mathematik I, Springer-Verlag, Berlin, 1979.

[10] F. B. Hildebrand: Introduction to Numerical Analysis, McGraw-Hill, New York, 1974.

[11] L. Fox: An Introduction to Numerical Linear Algebra, Oxford University Press, New York, 1964.

[12] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling: Numerical Recipes, the Art of Schientific Computing, Cambridge University Press, Cambridge, 1988.

[13] D. M. Young: Iterative Solution of Large Linear Systems, Academic Press, New York, 1971.

[14] R. S. Varga: Matrix Iterative Analysis, Prentice-Hall, N. J., 1962.

[15] U. Schendel: Sparse Matrices: Numerical Aspects with Applications for Scientists and Engineers, John Wiley, New York, 1989.

[16] G. Strang: Linear Algebra and its Applications, Academic Press, New York, 1976.
[17] G. Engeln-Müllges, F. Reutter: Formelsammlung zur numerischen Mathematik mit Quick-BASIC-Programmen, BI Wissenschaftsverlag, Mannheim, 1991.
[18] F. Schmitz: Private communication.
[19] W. J. Minkowycz, E. M. Sparrow, G. E. Schneider, R. H. Pletcher: Handbook of Numerical Heat Transfer, Wiley, New York, 1988.
[20] Le code BACCHUS T, Notice de présentation, No. SYFRA 90416, document from CEA/DEDR/DRE/STT/STML, internal reference number TT/STML/82-3-L.
[21] D. W. Peaceman, H. H. Rachford: The Numerical Solution of Parabolic and Elliptic Differential Equations, J. of Soc. Indust. Appl. Math. 3 (1955), 28 - 41.
[22] A. Messina, P. Londrillo: A N-Body Code for Elliptic Galaxies, Supercomputing in astrophysics, Astronet Special Publication 88/1, Rome 1988.
[23] M. Bottoni, B. Dorr, Ch. Homann: unpublished report, 1991.
[24] G. E. Forsythe: Finite-Difference Methods for Partial Differential Equations, Wiley, New York, 1967.
[25] A. Ralston, H. S. Wilf: Mathematische Methoden für Digitalrechner I, R. Oldenbourg Verlag, München, 1972.
[26] W. F. Ames: Numerical Methods for Partial Differential Equations, Nelson, London, 1969.
[27] J. Stoer, R. Bulirsch: Einführung in die Numerische Mathematik II, Springer-Verlag, Berlin, 1978.
[28] R. Zurmühl, S. Falk: Matrizen und ihre Anwendungen, Springer-Verlag, Berlin, 1984.