E. Dieterich
G. Sorescu
G. Eigenberger

FEDERAL REPUBLIC OF GERMANY

# Numerical methods for the simulation of chemical engineering processes

*Fundamental aspects and the current state of the art in simulating the dynamic and steady-state behavior of chemical engineering processes are discussed. The discretization of the spatial derivatives in the equations of change leads to a system of differential algebraic equations (DAE), consisting of ordinary differential equations in the time domain, and algebraic equations. The present paper discusses the necessary steps to solve the DAE, and mentions proven standard software for these steps as well as for the solution of the differential algebraic equations as a whole.*

## 1. Introduction

Chemical engineering research and development is based on the analysis and on the purposeful utilization of the relevant physical-chemical interactions. In this context, mathematical modeling of these interactions and their numerical simulation keep gaining an increasing importance; they enable to concentrate experimental investigations on well-defined aspects, to discover complex relationships through the interplay between experiment and numerical simulation, and to commercially exploit these.

From *Chemie-Ingenieur-Technik* **64**, No. 2, pp. 136–147 (1992), with permission. Dipl.-Ing. Erwin Dieterich, Dr.-Ing. Gheorge Sorescu, and Professor Dr.-Ing. Gerhart Eigenberger are associated with the Institut für Chemische Verfahrenstechnik, Universität Stuttgart, Böblinger Str. 72, D70199 Stuttgart 1.



Whereas mathematical analysis of experiments is distinguished by a long and proven tradition in chemical engineering, detailed mathematical modeling and numerical simulation are techniques within the framework of chemical engineering training that are covered with an intensity greatly varying from one place to another. In addition, current development of methods and procedures used in computer simulation is so rapid that it increasingly becomes more difficult to follow its growth.

In this paper, on the one hand, important fundamentals of numerical simulation are sketched, standard methods are explained, and a few sources that are suitable for their study in depth are mentioned. On the other hand, developments are discussed which have an effect on the application of numerical simulation, and which may become significant for its future use. In contrast to the computer scientist, or the (numerical) mathematician, the engineer usually considers numerical simulation as a tool with whose details he wants to become acquainted only up to a point which is necessary

for its practical application. This paper is based on that premise. However, it is apparent that, with the help of standard numerical tools, only standard problems of computer simulation may be solved, and that each foray into new territory forces consideration of the whole spectrum of numerical methods available. The present paper also contains comments on this problem.

Physically meaningful mathematical models of chemical engineering equipment and plants are based on the equations for conservation of mass, momentum, and energy. For most applications, so far they have been limited to the use of one or two spatial coordinates; often, only the steady state is treated. The latter limitation, however, is removed in this treatment, partly because consideration of dynamics becomes increasingly important for the understanding of chemical engineering equipment and processes, and, partly, because inclusion of dynamics often requires only small extra effort as compared with the calculation of the steady state, and sometimes it even assures a better convergence of the solution than a purely steady-state calculation. Above all, however, dynamic simulation may be carried out by using a highly standardized scheme, the so-termed method of lines.

### *1.1. Introductory example*

As an introductory example, the model of an isothermal tubular reactor is considered that can be characterized by the mass balance on the concentration $c$ of a key component for a controlling reaction. Using the spatially one-dimensional diffusion convection equation, the mass balance is as follows:

$$B\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial z^2} - v\frac{\partial c}{\partial z} + S(c, z, t) \tag{1}$$

with the initial condition

$$c(t = 0, z) = c^0(z) \tag{2}$$

and the boundary conditions

$$D\frac{\partial c}{\partial z}\bigg|_{z=0} = v(c(t, z = 0) - c^*) \tag{3}$$

$$\frac{\partial c}{\partial z}\bigg|_{z=L} = 0 \tag{4}$$

The source term $S(c, z, t)$ describes the rate of reaction which depends on the concentration $c$ usually in a nonlinear fashion, and that may also explicitly depend on position and time. The capacity factor $B$ in Equation (1) is equal to unity. Nevertheless, it is convenient to explicitly carry $B$ in the subsequent manipulations.

Equation (1) is a nonlinear partial differential equation (PDE) of the convection-diffusion type, which is widely used for the description of chemical engineering equipment, characterized by one-dimensional flow.

The customary technique used for its numerical solution is the method of lines. This technique is based on discretization of the spatial derivative

$$D\frac{\partial^2 c}{\partial z^2} - v\frac{\partial c}{\partial z} \tag{5}$$

*e.g.*, by a simple equidistant difference approximation at point $l$, using step size $\Delta z$:

$$\frac{\partial c}{\partial z}\bigg|_l \approx \frac{c_{l+1} - c_{l-1}}{2\,\Delta z} \tag{6}$$

$$\frac{\partial^2 c}{\partial z^2}\bigg|_l \frac{\partial\left(\frac{\partial c}{\partial z}\right)}{\partial z}\bigg|_l \approx \frac{1}{\Delta z}\left[\frac{c_{l+1} - c_l}{\Delta z} - \frac{c_l - c_{l-1}}{\Delta z}\right]$$

$$= \frac{c_{l+1} - 2\,c_l + c_{l-1}}{\Delta z^2} \tag{7}$$

This way, within the solution interval, the PDE is transformed into a system of $NZ$ ordinary differential equations. The equation at the point of linearization $l$ becomes

$$B\frac{\mathrm{d}c_l}{\mathrm{d}t} = \alpha c_{l-1} + \beta c_l + \gamma c_{l+1} + S(c_l, z_l, t) \tag{8}$$

where

$$\alpha = \frac{D}{\Delta z^2} + \frac{v}{2\,\Delta z} \tag{9}$$

$$\beta = \frac{-2\,D}{\Delta z^2} \tag{10}$$

$$\gamma = \frac{D}{\Delta z^2} - \frac{v}{2\,\Delta z} \tag{11}$$

In view of coupling between $c_l$, $c_{l-1}$, $c_{l+1}$, the system of equations must be solved simultaneously. Inspection of the solution plane $t$ above $z$ (*cf.* Figure 1) offers an explanation for the term method of lines. After discretization with respect to spatial derivatives, each of the ordinary differential equations (8) obtained is solved along a $z$ = constant line.

Equation (8), written in vectorial notation, is as follows:

$$\boldsymbol{B}\frac{d\boldsymbol{y}}{dt} = \boldsymbol{f}(\boldsymbol{y}, z, t) \tag{12}$$

where $\boldsymbol{B}$ is the capacity matrix; $\boldsymbol{y}$ is the solution vector; and $\boldsymbol{f}$ on the right-hand side is the function vector

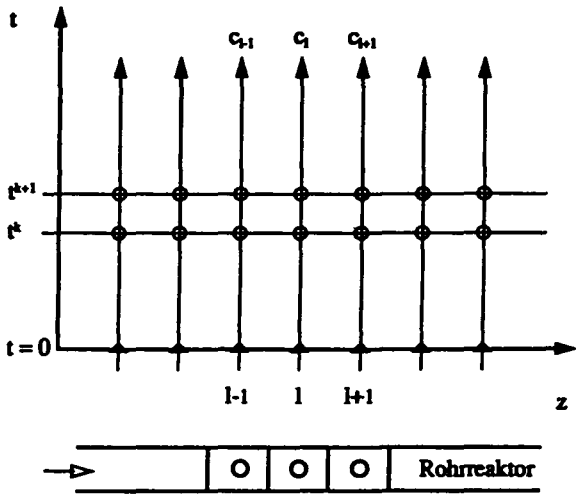


**Fig. 1.** Spatial discretization and integration with respect to time of Equation (1) by the method of lines.

$$\boldsymbol{B} = \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & \ddots & & & & \\ & & & 1 & & & \\ & & & & \ddots & & \\ & & & & & 1 & \\ & & & & & & 1 \end{bmatrix}$$

$$\boldsymbol{y} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_l \\ \vdots \\ c_{NZ-1} \\ c_{NZ} \end{bmatrix}; \boldsymbol{f} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_l \\ \vdots \\ f_{NZ-1} \\ f_{NZ} \end{bmatrix} \qquad (13)$$

A solution of the system (12) of first-order, nonlinear differential equations can be carried out by first discretizing the time coordinate $t$, using the index $k$ and the time step $\Delta t$:

$$\boldsymbol{B}\frac{\boldsymbol{y}^{k+1} - \boldsymbol{y}^k}{\Delta t} = \boldsymbol{f}^{k+1}(\boldsymbol{y}^{k+1}, z, t + \Delta t) \qquad (14)$$

Next, the nonlinear right-hand side $\boldsymbol{f}$ must be linearized. This linearization can be carried out, *e.g.*, by expanding $\boldsymbol{f}$ about the old solution $\boldsymbol{y}^k$. The relationship is as follows:

$$\boldsymbol{f}^{k+1} \approx \boldsymbol{f}^k + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}(\boldsymbol{y}^{k+1} - \boldsymbol{y}^k) \qquad (15)$$

where $\partial \boldsymbol{f}/\partial \boldsymbol{y}$ is the matrix of partial derivatives of the right-hand side $f$ with respect to all solutions $y$, the so-termed Jacobian matrix, at the known time $k$.

The solution at the new time $k + 1$, therefore, requires solution of the linear system of equations

$$\underbrace{\left(\frac{\boldsymbol{B}}{\Delta t} - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right)}_{\boldsymbol{A}} \boldsymbol{y}^{k+1} = \underbrace{\left(\frac{\boldsymbol{B}}{\Delta t} - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right) \boldsymbol{y}^k = \boldsymbol{f}^k}_{\boldsymbol{b}} \qquad (16)$$

where both the matrix of coefficients $\boldsymbol{A}$ and the vector $\boldsymbol{b}$ on the right-hand side can be determined with the help of model constants and the solutions at the last step $\boldsymbol{y}^k$.

The amount of computation required for the solution of the system of Equations (16) depends on the structure of the matrix $\boldsymbol{A}$, *i.e.*, on the number of elements in $\boldsymbol{A}$ that are different from zero, and on the pattern of the nonzero elements. With ease, it can be shown that the pattern is usually determined by the structure of the Jacobian matrix $\partial \boldsymbol{f}/\partial \boldsymbol{y}$. In the introductory example used, the Jacobian matrix is tridiagonal (*cf.* Figure 2a) because, according to Equation (8), at point of linearization $l$, the concentrations $c_{l-1}$, $c_l$, and $c_{l+1}$ appear in the mass balance. If the model of the tubular reactor under consideration not only consists of a PDE of one concentration but if also, *e.g.*, equations for other concentrations as well as for temperature, pressure, and density are required, then, in the vector $\boldsymbol{y}$ [Equation (13)], a subvector appears instead of $c_l$, consisting of the variables at point $l$, mentioned above. The function $f_l$ in the function vector $\boldsymbol{f}$ is replaced by a subvector of the appropriate spatially discretized right-hand side at point $l$. Then, $\partial \boldsymbol{f}_l/\partial \boldsymbol{y}_l$ is the Jacobian submatrix at point of linearization $l$, and, therefore, $\partial \boldsymbol{f}/\partial \boldsymbol{y}$ and also $\boldsymbol{A}$ obtain a block tridiagonal structure (*cf.* Figure 2b).

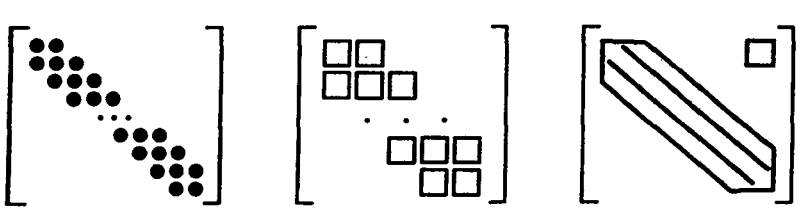


**Fig. 2.** Customary structures of the Jacobian matrix $\boldsymbol{J} = \partial \boldsymbol{f}/\partial \boldsymbol{y}$ or the solution matrix $\boldsymbol{A}$ [Equation (16)]. a) tridiagonal structure of the introductory example; b) block tridiagonal structure when considering several state variables; c) convection-diffusion system with recycle.

If, instead of simple flow through a reactor, a tubular reactor with recycle were considered, then, the first element $l = 1$ (input) would be coupled with the last element $l = NZ$ (output). In the pattern of the Jacobian matrix and of the $\boldsymbol{A}$ matrix, the additional coupling block at position $(1, NZ)$ is shown in Figure 2c. In this way, the peculiarities of spatial coupling are built into the matrix $\boldsymbol{A}$ of the linear system of Equation (16) for each piece of equipment and each process in an unambiguous way.

Also, the capacity matrix $\boldsymbol{B}$ introduced by Equations (1), (12), and (13) plays an important role for an efficient and routine numerical treatment of the model equations. In the introductory example, $\boldsymbol{B}$ corresponds to the unit matrix; often, $\boldsymbol{B}$ is a diagonal matrix. Situations exist, however, where time derivatives of several state variables appear in the same balance equation, *e.g.*, an enthalpy balance for compressible fluids leads to a sum of the time derivatives of temperature $(\partial T/\partial t)$ and total pressure $(\partial p/\partial t)$ that can be considered in $\boldsymbol{B}$ without any additional transformation. From Equation (16), it is apparent that the system of equations may also be solved if $\boldsymbol{B}$ is not regular, *i.e.*, if the system contains rows of zeros. An important example is the calculation of the steady-state solution. Here, all time derivatives disappear and $\boldsymbol{B}$ becomes the zero matrix. Under these conditions, Equation (16) describes an iterative scheme for the steady-state solution $\boldsymbol{y}^k$ by the Newton-Raphson method (see Section 3), where $k$ denotes the iteration step. In any event, the solution of the linear system of Equations (16) must be repeated until $\boldsymbol{y}^{k+1}$ sufficiently well agrees with $\boldsymbol{y}^k$.

In addition, during the course of model formulation, often nonlinear coupling relationships appear between state variables that do not contain time derivatives. Typical examples are equations of state or phase equilibrium relationships; quasisteady-state balance equations also belong here. This means that Equation (12) often does not represent a system of ordinary differential equations but it contains differential equations and nonlinear equations which are coupled with one another by $\boldsymbol{f}$ and, occasionally, also by $\boldsymbol{B}$. It has become customary to refer to this situation as that of differential algebra equations (DAE).

Quite in general, a physically based mathematical model of a chemical engineering process, after discretization of the spatial derivatives, leads to a DAE such as represented by Equation (12). DAE may be solved with the help of so-termed DA solvers (*cf.* Section 4). The introductory example shows a particularly simple procedure for this solution. In all situations, a linear system of equations that has a structure as the one shown by Equation (16) results, and that must be iteratively solved.

In the following section, the individual steps:

1. discretization with respect to the space coordinate;
2. discretization with respect to the time coordinate;
3. solution of the nonlinear system of equations; and
4. solution of the linear system of equations are discussed in greater detail and in the opposite order.

## 2. Solution of linear systems of equations

### *2.1. Direct solution procedure (LU decomposition)*

Formally, the solution of the linear system of equations

$$\boldsymbol{A}\boldsymbol{y} = \boldsymbol{b} \tag{17}$$

is carried out by finding the inverses $(\boldsymbol{A})^{-1}$:

$$\boldsymbol{y} = \boldsymbol{A}^{-1}\boldsymbol{b} \tag{18}$$

This procedure, however, usually is not carried out explicitly because of the large amount of calculations involved. Rather, $\boldsymbol{A}$ is decomposed by suitable transformation into the product of a lower triangular matrix $\boldsymbol{L}$ (*cf.* Figure 3a) and an upper triangular matrix $\boldsymbol{U}$ (*cf.* Figure 3b) [1, 3–6]:

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U} \tag{19}$$

With the knowledge of $\boldsymbol{L}$ and $\boldsymbol{U}$, the solution of Equation (16) or, equivalently, of

$$\boldsymbol{L}\boldsymbol{U}\boldsymbol{y} = \boldsymbol{b} \tag{20}$$

can be reduced to two simple single steps: using the

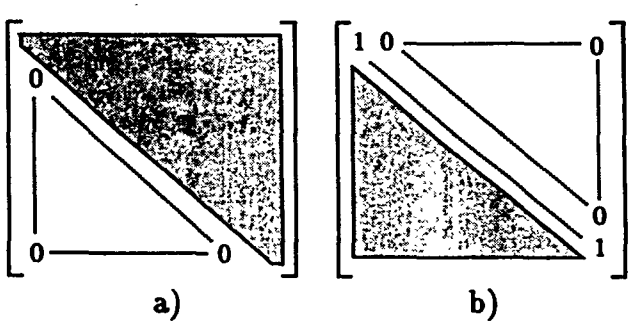


**Fig. 3.** Decomposition of $\boldsymbol{A}$ according to Equation (19) into a) an upper triangular matrix $\boldsymbol{U}$; and b) a lower triangular matrix $\boldsymbol{L}$.

substitution $\boldsymbol{Uy} = \boldsymbol{x}$, first, the system of equations

$$\boldsymbol{Lx} = \boldsymbol{b} \tag{21}$$

is solved by forward substitution for the determination of $\boldsymbol{x}$. If $\boldsymbol{x}$ is known, the solution vector $\boldsymbol{y}$ follows by substitution from

$$\boldsymbol{Uy} = \boldsymbol{x} \tag{22}$$

The decisive advantage of this procedure is that the solution may be easily found for any right-hand side $\boldsymbol{b}$ after carrying out the decomposition of $\boldsymbol{A}$ into $\boldsymbol{LU}$. This advantage may be exploited in various situations in numerical simulation of chemical engineering processes.

In the $\boldsymbol{LU}$ decomposition, attention must be paid to pivoting. The pivoting is necessary in order to avoid division by zero in the transformation of $\boldsymbol{A}$, or rounding off significant digits through small differences of large numbers. Pivoting is frequently facilitated by appropriate scaling of the state variables $\boldsymbol{y}$, *e.g.*, on the interval $[-1, 1]$; *cf.* [2, 4, 6].

Numerous reliable standard routines are available for the purpose of $\boldsymbol{LU}$ decomposition (*cf.* Section 2.3) so that the process engineer only very seldom needs to do any explicit programming. In general, it holds valid that the structure of $\boldsymbol{A}$ decisively influences the ease of solution of the linear system of equations. Therefore, it is logical to group the model equations in such a way that $\boldsymbol{A}$ possesses a standard dominance. This standard dominance includes, in particular, tridiagonal and block tridiagonal dominances, both of which are shown in Figure 2. The tridiagonal and block tridiagonal patterns have the advantage that they keep their structure in the process of $\boldsymbol{LU}$ decomposition, if no pivoting is required.

Sometimes, it happens that model building does not lead to any of the standard patterns shown in Figure 2, and, instead, the nonzero elements are very irregularly distributed over the matrix. This distribution especially occurs for coupling of various pieces of equipment or processes, *i.e.*, for plant simulation. Here, $A$ has only a few nonzero elements, and it is referred to as a sparse matrix. For such sparse matrices, special so-termed sparse solvers exist, which limit the storage place and the necessary calculations to the nonzero elements of the matrix.

### 2.2. *Iterative solution techniques*

The solution techniques discussed up to this point are so-termed direct methods. For dense matrices, these techniques permit the solution of systems consisting of a few thousand equations, and, for distinctly diagonally dominant matrices, even the solutions of many more. Because the amount of calculation increases faster than in direct proportion to the number of equations $N$, an upper limit of $N$ exists beyond which a direct solution is not justified either from the point of view of the amount of calculation to be carried out or from the point of view of the increasing extent of round-off errors. The upper limit depends on the so-termed conditioning of the matrix and on the computing accuracy available. This limit may be easily reached and also exceeded, particularly in the simulation of model equations of two or three spatial dimensions.

One way out of this difficulty is by using iterative solution techniques. However, whereas direct methods always lead to the solution, for a nonexcessive number of equations $N$ and a regular matrix $\boldsymbol{A}$, convergence with the classical iterative methods is only assured for diagonally dominant systems. Modern iterative methods are capable of creating diagonal dominance by using suitable transformations [3, 6].

According to the simplest version of an iterative solution technique, the so-termed Jacobian method, the $n$th equation of the system of equations is solved for the $n$th unknown in each iteration step, independently of the other equations. Then, for Equation (17), *e.g.*,

$$y_n^{i+1} = \frac{1}{a_{nn}} \left( b_n - \sum_{k=1, k \neq n}^{N} a_{n,k}\, y_k^i \right) \tag{23}$$

where $a_{n,k}$ are the elements of $\boldsymbol{A}$. Here, $i$ is the iteration index. This indicates that always the unknowns from the previous iteration are substituted into the right-hand side.

If the unknowns converge, iterative methods have the advantage to be insensitive to rounding off errors, and, for sparse matrices, iterations require only little storage capacity.

### 2.3. *Software for the solution of linear systems of equations*

For a direct solution of systems of equations with block tridiagonal or tridiagonal structure, the library routines of LINPACK [18] or the IMSL library [19] can be recommended. These routines are just as the rest of the numerical software recommended here, written in standard FORTRAN 77, and they are based on $\boldsymbol{LU}$ decomposition and partial pivoting.

For the direct solution of systems of sparse matrices, a generally accepted solution technique is not established. A very frequently used program is the Harwell routine MA30, or its variant MA28 by Duff and Reid [20, 21], which is easier to use and whose

strength lies in the application of information obtained in the first $LU$ decomposition for subsequent solutions of problems with a similar pivotal structure. In recent years, various new algorithms have been published [22–24] that, however, mostly do not reach the speed of MA 30 for repeated solutions of problems of identical structure such as in the simulation of chemical engineering processes [25].

Iterative methods for the solution of linear systems of equations do not yet reach the reliability provided by direct solution. In practice, tests must be carried out in every individual situation whether or not a method converges for the particular problem at hand. On the basis of the developments in recent years, *e.g.*, [26], however, significant progress may be expected in the field using special strategies for improvement in convergence and of combinations of direct and iterative methods.

Basically, standard methods such as those described in this section always result in a slower solution of a system of equations than an algorithm that was optimized for a particular matrix structure; however, the amount of work required for such nonstandard methods is only warranted for very time-critical applications.

Each numerical simulation is ultimately based on repeated solutions of linear systems of equations. From the users point of view, therefore, the developers of general solution packages for nonlinear equations or DA systems should uncouple their programs from the solution of linear systems of equations, and provide general interfaces. Only, then, the incorporation of specific linear equation solvers will be possible without unreasonable expense.

## 3. Solution of nonlinear systems of equations

The iterative solution of a nonlinear equation in one unknown

$$0 = f(y) \tag{24}$$

by Newton's method, using the iteration index $i$

$$y^{i+1} = y^i - \frac{f(y^i)}{f'(y^i)} \tag{25}$$

is generally known. Generalization of this iteration procedure to systems of equations is the so-termed Newton-Raphson method which can be formally written as a fixed-point iteration:

$$\boldsymbol{y}^{i+1} = \boldsymbol{y}^i - \boldsymbol{J}(\boldsymbol{y}^i)^{-1}\boldsymbol{f}(\boldsymbol{y}^i) \tag{26}$$

Here, $\boldsymbol{J}(\boldsymbol{y})$ is the Jacobian matrix that was already introduced in connection with Equation (15)

$$\boldsymbol{J}(\boldsymbol{y}) = \begin{bmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} & \cdots & \frac{\partial f_1}{\partial y_n} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} & \cdots & \frac{\partial f_2}{\partial y_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial y_1} & \frac{\partial f_n}{\partial y_2} & \cdots & \frac{\partial f_n}{\partial y_n} \end{bmatrix} \tag{27}$$

In practice, instead of a matrix inversion, the linear system of equations

$$\boldsymbol{J}(\boldsymbol{y}^i)\,\Delta\boldsymbol{y}^i = -\boldsymbol{f}(\boldsymbol{y}^i) \tag{28}$$

equivalent to Equation (26), with

$$\boldsymbol{y}^{i+1} = \boldsymbol{y}^i + \Delta\boldsymbol{y}^i \tag{29}$$

is iteratively solved until the residual vector $\Delta\boldsymbol{y}^i$ becomes less than a prescribed value [1, 3, 5]. In the introductory example, only the first step in the Newton-Raphson iteration was used in Equations (15) and (16), where the initial value was the solution vector at the old time step $t^k$.

The reason for frequent use of either the direct or the modified Newton-Raphson method is its simple algorithm and its quadratic, *i.e.*, faster than linear convergence. On the other hand, this convergence is only assured if sufficiently good initial values are used. Hence, the solution of large nonlinear systems of equations may run into great difficulties because if several solutions exist the algorithm does not necessarily converge to the desired or to a physically meaningful solution. In conjunction with the solution of a DAE, these convergence problems, however, are less severe because the previous time step provides a good starting value for the iteration, while the size of the change $\Delta\boldsymbol{y}$ can be arbitrarily decreased by reduction of the size of the time step.

Exceptions to this rule may be expected in two situations: at the start of the program if inconsistent starting values are used (*cf.* Section 4), or for functions with a local or a global minimum in the vicinity of the solution. This type of problem can be easily understood for one unknown. The equation $0 = y^2$ has a (double) zero at $y = 0$. At the same time, the global minimum of the function is at the same place. Thus, $f'(y = 0) = 0$. Here, the iteration procedure (25) leads to a division by zero.

Additional difficulties may arise if the function $\boldsymbol{f}$ is not defined everywhere, because the Newton-Raphson method does not always monotonously converge but often oscillates about the solution. If the

function is not defined in the vicinity of a zero, then, an evaluation might be required in an undefined region. As a simple example, the reaction rate-expression $r = c^{0.5}$ may be used. A reasonable idea is to assign the value of zero to the reaction rate for negative concentrations. However, this assignment may prevent convergence of the Newton-Raphson procedure because here the derivative of the function $f$ in the vicinity of the solution is no more continuous. A logical way out of this difficulty is to continue the function without interruption into the undefined region, *e.g.*, by letting $r(c < 0) = -|c|^{0.5}$.

In the process of determination of the Jacobian, the first question to answer is whether or not to use an analytical solution or an approximate numerical calculation. As a rule, an analytical evaluation of the derivatives leads to more accurate results, and mostly also in a shorter time. However, analytical differentiation of complicated functions requires tedious work which may be subject to errors. Examples for this are implicit relationships for the calculation of multicomponent equilibria, and complicated reaction rate-expressions. Therefore, in many library programs, the option exists of numerically calculating the Jacobian matrix:

$$\frac{\partial f_i}{\partial y_j} \approx \frac{f_i(y_j + \Delta y_j) - f_i(y_j)}{\Delta y_j} \tag{30}$$

In this process, care must be taken to avoid obliteration of significant digits by suitable scaling and by choice of step size $\Delta y_i$. By taking advantage of that the Jacobian matrix usually is sparse, the number of function calls for the calculation of $\boldsymbol{f}(\boldsymbol{y} + \Delta \boldsymbol{y})$ can be significantly reduced by altering before each function call several components of $\boldsymbol{f}$. Thus, in our introductory example for the complete calculation of the tridiagonal Jacobian, four function evaluations are sufficient, independently of the number of discretization points.

Because, in the last few years, powerful programs for symbolic calculations have become available, *e.g.*, MAPLE [35] or MACSYMA [36], today, analytical derivatives can be determined in a simple and accurate manner. The result is available in FORTRAN code, and may directly be transferred to the equation solver.

In the iteration according to Equation (26), primarily, two cost factors appear:

(1) solution of the linear system of equations; and

(2) calculation of the Jacobian matrix $\boldsymbol{J}$.

Efficient procedures for the solution of linear systems of equations were discussed in the previous section. The simplest strategy for decreasing the amount of work required for the calculation of the Jacobian matrix consists of keeping the Jacobian matrix constant throughout several iteration steps, whereby the order and the radius of convergence are diminished. On the other hand, the determination of the Jacobian matrix and the $\boldsymbol{LU}$ decomposition becomes unnecessary. To what extent this possibility can be exploited to advantage depends on the particular situation at hand.

Another possible way consists of not repeatedly calculating the entire Jacobian matrix but to improve in each iteration step the matrix determined at the outset. Procedures along these lines are mostly based on the method developed by Broyden [28]. These procedures have the disadvantage that they create additional nonzero elements in the matrix. Several authors [29, 30] tackled the problem of Broyden updates while maintaining the matrix density almost unchanged.

A convenient possibility for influencing the rate and the radius of convergence in the Newtonian process is the relaxation expressed by

$$\boldsymbol{y}^{i+1} = \boldsymbol{y}^i + \omega \Delta \boldsymbol{y}^i \tag{31}$$

In this process, a step with $\omega > 1$ is referred to as overrelaxation, whereas a step with $\omega < 1$ is termed underrelaxation. Overrelaxation is used to increase the convergence rate. Of practical significance is primarily the underrelaxation whereby the convergence radius can be significantly increased [3, 14].

### *3.1. Standard software for the solution of nonlinear systems of equations*

Newtonian and *quasi*-Newtonian methods are available for the calculation of zeros of nonlinear systems of equations in all the large standard mathematical libraries, *e.g.*, such as IMSL or Harwell. The program NLEQ1 of Deuflhard [31] was found by the authors to be particularly useful in the solution of critical problems such as in the calculation of simultaneous chemical equilibria, and, also, in the process of initialization of DA systems; see Section 4.2. This program has a very effective control of the relaxation parameter $\omega$. Recent reviews of and comparison between various solution methods of nonlinear systems of equations from the point of view of the chemical engineer were presented by Sacham [32] and Sun [33].

## 4. Integration of differential equations

During the course of model building, generally, systems of partial differential equations, ordinary differential equations of first and higher order, and alge-

braic equations are generated. With the help of the method of lines, the partial differential equations can be reduced to a system of ordinary differential equations. Transformation of ordinary differential equations of a higher order into a system of differential equations of first order is carried out by simple substitution.

First, the integration of an ordinary differential equation of first order with a nonlinear right-hand side $f(y, t)$ is considered:

$$y' = \frac{dy}{dt} = f(y, t) \tag{32}$$

The simplest possibility consists of approximating the differential quotient $dy/dt$ by a difference quotient, *e.g.*,

$$\frac{dy}{dt} = \frac{y^{k+1} - y^k}{\Delta t} = f(y, t) \tag{33}$$

The next question is how to use a suitable approximation for the function $f(y, t)$ in the interval $(t^k, t^k + \Delta t)$. A flexible way of accomplishing this is provided by the general Euler method (*cf.* Figure 4).

$$\frac{y^{k+1} - y^k}{\Delta t} = (1-a)f(y^k, t^k) + a\, f(y^{k+1}, t^k + \Delta t) \tag{34}$$

Letting $a = 0$, *i.e.*, $f(y, t) \approx f(y^k, t^k)$, the so-termed explicit Euler method is obtained, permitting solution for $y^{k+1}$:

$$\frac{y^{k+1} - y^k}{\Delta t} = f(y^k, t^k) \tag{35}$$

In contrast to this method, the so-termed implicit Euler method with $a = 1$, *i.e.*, $f(y^{k+1}, t^k + \Delta t)$, in general, results in a nonlinear relationship that must be solved by iterations, using the methods described in the previous section

$$\frac{y^{k+1} - y^k}{\Delta t} = f(y^{k+1}, t^k + \Delta t) \tag{36}$$

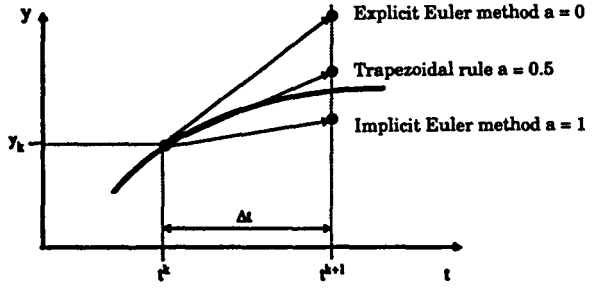


**Fig. 4.** Integration of Equation (32) using the general Euler method [Equation (34)].

Both of these methods reach a first-order accuracy. For $a = 1/2$, from Equation (34), the so-termed trapezoidal rule follows, which is also implicit since $f^{k+1}$ is required, but this rule reaches a second-order accuracy.

In this context, $n$th-order accuracy indicates that the discretization error in the solution of the differential equation is proportional to $\Delta t^n$. By using a higher-order method, the discretization error can be reduced to a greater extent by decreasing $\Delta t$.

Using a central difference approximation of the time derivative, the following explicit procedure is obtained, which is also of the order of two:

$$\frac{y^{k+1} - y^{k-1}}{2\,\Delta t} = f(y^k, t^k) \tag{37}$$

An analysis, however, shows that an unstable parasitic solution is brought in by the discretization process. This analysis leads, independently of the choice of $\Delta t$, after several steps to oscillations of increasing amplitude about the true solution so that, as a result, Equation (37) turns out to be useless [7].

A necessary condition for a solution algorithm is, therefore, first of all, a stable approximation of the time derivative. This condition is satisfied for the general Euler method. Nevertheless, even here undesirable oscillations may occur if $a < 1$, and the time step is greater than a certain value. The undesirable oscillations are shown in Figure 5 on the example of the stable linear differential equation

$$\tau \frac{dy}{dt} = -y \tag{38}$$

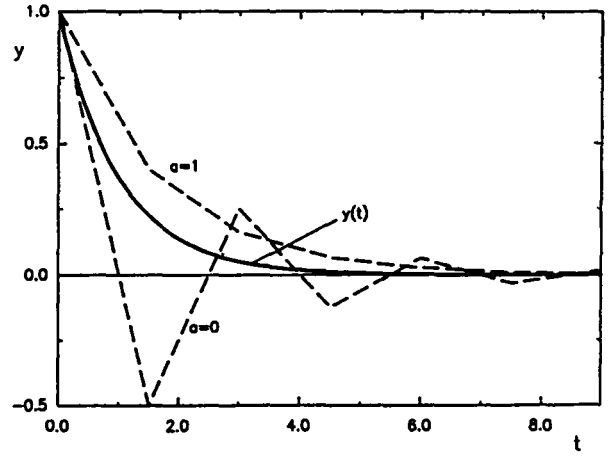


**Fig. 5.** Numerical solution of Equation (38) with $\Delta t = 1.5\tau$ using the implicit ($a = 1$) and the explicit ($a = 0$) Euler method.

with the solution

$$y(t) = y^0 \exp(-t/\tau) \tag{39}$$

With the help of Equations (35) and (36), the development of the numerical solution for large time steps can be easily followed. For the explicit Euler method, the solution starts to oscillate in the example as soon as $\Delta t > \tau$, and it exponentially increases as soon as $\Delta t > 2\tau$. In contrast, the implicit Euler method monotonously approaches the exact solution, even for arbitrary large time steps $\Delta t$.

This result may be generalized in a remarkable way: all purely explicit integration methods for the solution of stable differential equations become unstable above a limiting value of the time-step size $\Delta t_{crit}$, where $\Delta t_{crit}$ is of the order of magnitude of the smallest time constant $\tau$ or the smallest reciprocal eigenvalue of the linearized differential equation. In contrast, implicit solution methods permit time steps that are also significantly greater than $\tau$. This result is of great importance for the solution of systems of differential equations (*cf.* Section 4.1). On the other hand, no reason exists to prefer the implicit to the explicit method in the solution of individual differential equations of first order, since, in order to obtain an accurate solution, the time step must be chosen to be smaller than the value of the time constant anyhow. The amount of calculations involved in the explicit methods is, by the very nature, significantly less compared to the implicit method. For this reason, in the past, mainly explicit solution methods were used for ordinary differential equations. The most popular examples are the classical fourth-order Runge-Kutta method, the Adams-Moulton, and the Adams-Bashford methods [8].

For an efficient use of an integration method, automatic time-step control must be used. Control assures, in addition to providing stability (in the explicit method) and convergence (in the implicit method), also a control of the errors, and minimization of the computing time required for the solution of a problem. In most situations, control of the time-step size is based on an error estimation, *e.g.*, by a comparison of results obtained with different step sizes $\Delta t$.

### *4.1. Integration of a system of ordinary differential equations*

In the integration of a system of ordinary differential equations,

$$\boldsymbol{B}\boldsymbol{y}_1' = \boldsymbol{f}(\boldsymbol{y}, t) \tag{40}$$

with a regular matrix $\boldsymbol{B}$, application of one of the explicit methods described above may run into significant difficulties. These difficulties are illustrated on a short example. The concentration evolution in an isothermal batch reactor of constant volume is to be calculated for a simple situation of two reactions in series:

$$A \xrightarrow{k_1} B \xrightarrow{k_2} C \tag{41}$$

The time dependence of concentration is described by the following first-order differential equations:

$$\frac{dc_A}{dt} = -k_1 c_A \tag{42}$$

$$\frac{dc_B}{dt} = k_1 c_A - k_2 c_B \tag{43}$$

$$\frac{dc_C}{dt} = k_2 c_B \tag{44}$$

Analytical solution of this system is as follows:

$$c_A = c_{A0} \exp(-k_1 t) \tag{45}$$

$$c_B = \frac{k_1 c_{A0}}{k_2 - k_1} (\exp(-k_1 t) - \exp(-k_2 t)) \tag{46}$$

$$c_C = \frac{c_{A0}}{k_2 - k_1} (-k_2 \exp(-k_1 t) + k_1 \exp(-k_2 t)) + c_{A0} \tag{47}$$

where $c_{A0}$ is the concentration of the educt $A$ at time $t = 0$. If reaction (2) is much faster than reaction (1), then, the term $\exp(-k_2 t)$ rapidly vanishes and the dynamic behavior of the system is only determined by $\exp(-k_1 t)$. Figure 6 shows the concentration evolution for values of the rate constant $k_1 = 1$ and $k_2 = 10^6$. Note should be made that the solution is almost exclusively determined by the slow dynamics with $\tau_1 = 1/k_1 = 1$ after a very short initial time period. Nevertheless, when using the explicit method of integration, for the reasons explained above, the time-step size of the order of magnitude of the least time constant must be chosen, *i.e.*, $\tau_2 = 1/k_2 = 10^6$ (!), in order to keep the solution stable. Such a system is referred to as stiff. As a measure of stiffness, generally, the ratio of the least to the greatest real part of the eigenvalues of the system of differential equations can be used [5]. Integration of a stiff system of the form of Equation (40) with an explicit method is not very effective: the number of the integration steps which are only necessary for the sake of stability and not for the sake of accuracy becomes so great that the integration can only be conveniently carried out with implicit methods.

For nonlinear differential equations, the eigenvalues follow from the equations linearized about the respective solution. Therefore, the stiffness can strongly change during the course of the solution process. The least eigenvalue of a system of equations

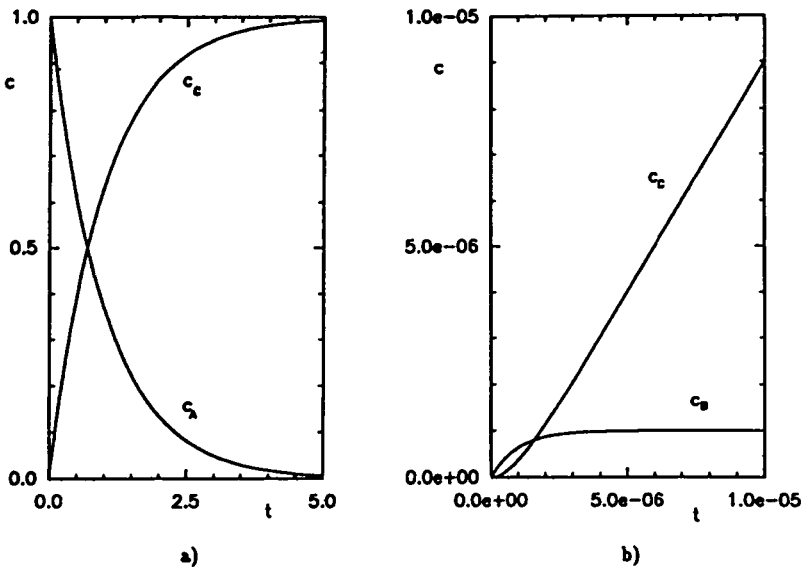


**Fig. 6.** a) Concentration *versus* time curves for a consecutive reaction [Equation (41)] for $k_1 = 1$, $k_2 = 10^6$; b) short time dynamics of the consecutive reaction.

arising from spatial discretization of a PDE also depends on the size chosen for the spatial step. For these and the other reasons mentioned in Section 4.2, it is advisable not to use the explicit integration methods in connection with the method of lines.

The semiimplicit Euler method [37] used in the introductory example

$$\boldsymbol{B}\frac{\boldsymbol{y}^{k+1}-\boldsymbol{y}^{k}}{\Delta t}=\boldsymbol{f}(\boldsymbol{y}^{k})+\left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}^{k}}(\boldsymbol{y}^{k+1}-\boldsymbol{y}^{k}) \qquad (48)$$

can be obtained from the implicit Euler method, Equation (36), by linearizing the term $\boldsymbol{f}^{f+k}$ about the old time point $t^k$, and leads to the formally explicit integration procedure

$$\left(\boldsymbol{B}-\Delta t\left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}^{k}}\right)\boldsymbol{y}^{k+1} = \left(\boldsymbol{B}-\Delta t\left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}^{k}}\right)\boldsymbol{y}^{k}+\Delta t\boldsymbol{f}(\boldsymbol{y}^{k}) \qquad (49)$$

without losing the stability properties of the implicit Euler method [17]. Therefore, it is also suitable for stiff systems. This example forms the basis of LIMEX [47], an efficient standard procedure for the solution of DAEs; *cf.* Section 4.4.

### *4.2. Integration of differential algebra systems*

In general, the model of chemical engineering equipment consists of differential equations that describe its dynamic behavior, and of algebraic coupling, DAE. The model may also be expressed in the form of Equation (40), where the matrix $\boldsymbol{B}$ is singular, *i.e.*, it contains empty rows and/or columns. Typical examples for linear and nonlinear coupling relationships are represented by the boundary conditions of PDEs, phase equilibria, equations of state, and PDEs assumed to be in *quasi*-steady state. The DAE may be solved as a stiff system with the help of implicit integration methods [38], provided its index (*cf.* next section) is less or equal to unity. Algebraic relationships can be visualized as differential equations with an infinitely short oscillation time; DAEs are, therefore, infinitely stiff systems of differential equations.

For the numerical solution of a DAE, the initial conditions must be specified. However, the number of the initial conditions that can be freely chosen, *i.e.*, the degrees of freedom of the DAE may be less than the number of variables whose time derivatives are contained in the system. This issue will be discussed in greater detail in the next section. In any event, for the start of integration, the additional (not freely selectable) variables must be determined in an initialization step in such a way that they satisfy the DAE at the initial point of time. For this purpose, a transformation of the DAE which is described in the next section should be used.

### *4.3. Index problems*

In most situations, a DAE may be solved without any difficulty with the help of implicit or semiimplicit integration methods. In some situations, however, a particular structural property of DAEs, the so-termed index, has a detrimental effect. Various ways exist of defining the index, among which the most widely used is to define a differential index [40, 41].

According to its definition, the differential index $k_d$ of a DAE is the number of derivatives necessary in order to transform the DA system into a system of ordinary differential equations (ODE). A DAE with an index $k_d \leq 1$ may be integrated with the help of the above-described methods for stiff systems of differential equations. The integration of systems with $k_d \geq 2$, however, may lead to severe problems so that standard methods often do not work.

For a demonstration of these deficiencies, the following simple example is considered:

$$y_1' = y_1 + 2y_2 + 2z \tag{50}$$

$$y_2' = y_1 - y_2 - z \tag{51}$$

$$0 = y_1 + y_2 \tag{52}$$

The integration is characterized by the peculiarity that $y_1$ and $y_2$ are algebraically coupled through Equation (52). Evidently, therefore, only one of these equations can be freely specified as an initial condition, whereas the initial value of the other, along with the variable $z$, must be determined in an initialization step. However, this initialization is not immediately possible because no explicit relationship exists for the calculation of $z$. For the determination of $z$, Equation (52) can first be differentiated:

$$y_1' + y_2' = 0 \tag{53}$$

After substitution of (50) and (51) into (53), the following algebraic relationship which was hidden in the original system is obtained:

$$0 = 2y_1 + y_2 + z \tag{54}$$

Now, $z$ can be calculated. After differentiation of Equation (54) and substitution of Equations (50) and (51), the following differential equation for $z$ is obtained:

$$z' = -3(y_1 + y_2 + z) \tag{55}$$

Since two differentiations were necessary for the transformation of the original DAEs (50) to (52) into the system of ordinary differential equations (50), (51), and (55), the differential index of the original system is two.

For a general handling of index problems for $k_d \geq 2$, Bachmann *et al.* [45, 46] suggested developing the hidden algebraic coupling from the original DAE by differentiation and substitution, and to solve the algebraic equations along with the required number of differential equations. In the above-mentioned example, the substitute system consists of Equations (52) and (54), in addition to one of the three differential equations, namely, Equations (50), (51), and (55), and the system has a differential index of $k_d = 1$.

The example shows that, due to the structure of the DAE, all three variables $y_1$, $y_2$, and $z$ play an equal role, and that to which of these an initial condition (degree of freedom) should be assigned is a matter of arbitrary choice. In practice, which of these variables will be used as the initial condition depends on the nature of the physical problem.

Transformation of the original DAE by differentiation and substitution may require a great deal of calculations for complicated systems. Under such conditions, it is convenient to substitute a new (algebraic) unknown for the derivative of one of the variables to be eliminated, with the result that the derivatives of the algebraic coupling conditions supply the additional equations necessary for the solution of the problem. In the above-mentioned example, the substitution of $y_2' = z_2$ in Equation (53) results in the equivalent DAE with the index $k_d = 1$:

$$y_1' = y_1 + 2y_2 + 2z \tag{56}$$

$$0 = y_1 - y_2 - z - z_2 \tag{57}$$

$$0 = y_1 + y_2 \tag{58}$$

$$y_1' = -z_2 \tag{59}$$

Although the system now has an additional new variable, the amount of work necessary for transformation was significantly reduced.

For dynamic models of chemical engineering processes, index problems may always arise whenever the state variables are coupled with one another by equations of state, phase equilibrium relationships, or special transport expressions. In this connection, a problem may arise already in the formulation of the fundamental partial differential equations [42]. The way to proceed here, *i.e.*, before spatial discretization has taken place, is subject to ongoing research.

### *4.4. Standard software for DAEs*

Using the integration routines contained in the large program libraries of numerical mathematics, *e.g.*, IMSL, NAG, and Harwell, simple problems can be solved without any difficulties and in a short time. However, in general, to use programs in which more recent results of mathematical research are implemented is more effective, since, particularly in the field of numerical integration of DAEs, significant progress has been made in the eighties. The following methods were found to be very effective:

(1) backward differential formula (BDF) methods;
(2) extrapolation methods; and
(3) implicit Runge-Kutta methods.

The programs DASSL [43] and LSODE/LSODI [44], based on the BDF method, found the most widespread use. The extrapolation codes developed by Deuflhard [3], *e.g.*, EULSIM and LIMEX [47], were found to be extremely effective in our applications, and were found to be equally valuable compared to DASSL in the solution of problems of practical relevance. In recent years, a strong interest developed in the implicit Runge-Kutta method [48–50]. The respective codes, *e.g.*, RADAU5 [50], did not yet reach the level of the above-named programs, but, in the near future, further progress may be expected.

All the above-mentioned programs contain automatic error and step-size monitoring; in addition, LIMEX is in a position to detect problems with an index $k_{d\geq 2}$, and to automatically reject these. In general, these routines solve a problem in comparable lengths of time, and, in each individual situation, it must be established which method is the most suitable. If, in a system, a large number of switching processes exist, which manifest themselves in discontinuities of the right-hand side $f(y, t)$ of the mathematical models, then, one-step methods such as the extrapolation or the Runge-Kutta methods are to be preferred because, with multistep methods such as BDF, a restart must be performed with low order.

The simultaneous methods of dynamic plant simulation such as DIVA [51] or SPEEDUP [52] are based on DAE solvers of high performance, where the models of individual plant parts are discretized in space, and combined to one single large DAE (40), which is solved with the methods described above.

## 5. Integration of partial differential equations

As has been shown in the introductory example, systems of PDEs can be transformed into a DAE by discretization with respect to the position coordinates. Whereas, for the solution of a DAE, the above-described comprehensive and tested method of solution exists, permitting the solution with a predetermined accuracy, the technique of spatial discretization is not equally well developed. In particular, the chemical engineer himself must usually also carry out spatial discretization when using commercial simulation software, unless he or she prefers premanufactured process modules with a limited scope. As a rule, in these modules, the errors arising from spatial discretization are not considered or limited.

A large number of methods is available for the discretization of spatial derivatives. The best known for chemical engineering applications are the finite difference methods and the finite element methods. In the one-dimensional situation, both yield either identical or similar systems of equations. Due to its simplicity of handling, the difference method is preferred in the one-dimensional situation and sometimes also in the two-dimensional case. As an example, the difference approximation carried out in Equations (5) to (7) of the introductory example may be mentioned.

The problems arising in the discretization of spatial derivatives and their convenient handling will not be discussed in this paper, but it will be presented in a separate publication. At this point, only reference is made to some introductory and comprehensive monographs [9, 10]. Introductory chapters dealing with the treatment of PDEs may also be found in some standard texts on numerical mathematics [5, 6]. The methods used for the solution of chemical engineering problems are discussed in [11, 12]. An introduction to the so-termed finite volume method may be found in [13] and to the finite element method in [14 to 16].

## 6. Sources of information and codes

This survey of numerical methods for the simulation of dynamic and stationary behavior of chemical engineering processes was meant to review the procedures that were established and tested in recent years. More detailed information may be found in the references indicated. It was pointed out that, for the solution of linear systems of equations, the solution of nonlinear systems of equations, and the solution of DAEs, proven standard codes are available so that individual programming of these steps may be limited to a few special situations.

The standard codes are usually contained as FORTRAN 77 subprograms, on the one hand, in the large commercially operated numerical libraries requiring licensing fees, such as IMSL, Harwell, or NAG. On the other hand, in recent years, a number of interesting codes have become available without fees as public domain software, *i.e.*, these subprograms may be freely copied and used. An example of this is the LAPACK collection for the solution of linear systems of equations, along with the BLAS routines simultaneously developed for the purpose of fundamental vector and matrix operations. (BLAS = Basic Linear Algebra Subroutines.) These routines are available, and can be copied without charge at most university computer centers throughout the world. Additional general or specialized codes are centrally collected at various locations. The best known server of programs of numerical math is NETLIB. The largest similar collection in Germany is the eLib of the Konrad-Zuse-Zentrum in Berlin. In addition to a

copy of NETLIB, also numerous recent developments may be found in eLib, such as the routines LIMEX and NLEQ1 mentioned in the present paper. This collection of programs is easily obtained by way of INTERNET, and various mail protocols. Information on the best way to proceed may be obtained from administrators of networks of scientific computer centers.

## Acknowledgments

The material presented in this paper partly originates from the joint project DIVA of dynamic simulation of chemical engineering plants involving four laboratories of the Chemical Engineering Faculty of the University of Stuttgart. Support of this work by the VW Foundation and DFG is gratefully acknowledged.

## Nomenclature*

| | |
|---|---|
| $\boldsymbol{A}$ | coefficient matrix |
| $\boldsymbol{b}$ | vector of right-hand side |
| $B, \boldsymbol{B}$ | capacity term |
| $c$ | concentration |
| $D$ | diffusivity |
| $f, \boldsymbol{f}$ | function of the right-hand side |
| $\boldsymbol{J}$ | Jacobian matrix |
| $k_1, k_2$ | rate constants |
| $\boldsymbol{L}$ | lower triangular matrix |
| $L$ | length of reactor |
| $N$ | number of equations |
| $NZ$ | number of mesh points in space |
| $p$ | pressure |
| $S$ | source function |
| $t$ | time |
| $T$ | temperature |
| $\boldsymbol{U}$ | upper triangular matrix |
| $v$ | flow velocity |
| $y, \boldsymbol{y}$ | unknown |
| $z$ | position |
| $z$ | algebraic unknown (in Section 4.3) |

### *Greek letters*

| | |
|---|---|
| $\Delta t$ | time-step size |
| $\Delta z$ | spatial step size |
| $\tau$ | time constant |
| $\omega$ | relaxation factors |

### *Subscripts*

| | |
|---|---|
| $i$ | iteration index |
| $k$ | time index or summation index |
| $l$ | position index |
| * | input |

*Vectors and matrices are in boldface.

## Literature cited

1. Stoer, J., *Numerische Mathematik* (Numerical Mathematics) 1, 5th Edition, Springer-Verlag, Berlin (1989).
2. Stoer, J., *Numerische Mathematik* (Numerical Mathematics) **2**, Third Edition, Springer-Verlag, Berlin (1990).
3. Deuflhard, P. and Hohmann, A., *Numerische Mathematik* (Numerical Mathematics), W. de Gruyter, Berlin (1991).
4. Engeln-Müllges, G., *Numerische Mathematik für Ingenieure* (Numerical Mathematics for Engineers), 5th Edition, Bibliographisches Institut (1987).
5. Schwarz, H. R., *Numerische Mathematik* (Numerical Mathematics), Second Edition, B. G. Teubner Verlag, Stuttgart (1988).
6. Press, W. H., *et al.*, *Numerical Recipes*, Third Edition, Cambridge University Press, Cambridge (1988).
7. Burden, L. B. and Faires, J. D., *Numerical Analysis*, Third Edition, Prindle, Weber, and Schmidt, Boston (1989).
8. Gear, G. W., *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice Hall, Englewood Cliffs (1971).
9. Meis, T. and and Marcowitz, U., *Numerische Behandlung partieller Differentialgleichungen* (Numerical Treatment of Partial Differential Equations), Springer-Verlag, Berlin (1978).
10. Ames, W. F., *Numerical Methods for Partial Differential Equations*, Second Edition, Academic Press, New York (1977).
11. Davis, M. E., *Numerical Methods and Modeling for Chemical Engineers*, John Wiley & Sons, New York (1964).
12. Finlayson, B. A., *Nonlinear Analysis in Chemical Engineering*, McGraw-Hill, New York (1980).
13. Patankar, S. V., *Numerical Heat Transfer and Fluid Flow*, McGraw-Hill, New York (1980).
14. Richter, W., *Numerische Lösung partieller Differential-gleichungen mit der Finite-Element-Methode* (Numerical Solution of Partial Dif-

ferential Equations with the Finite Element Method), F. Vieweg & Sohn, Braunschweig (1986).
15. Villadsen, J. V. and Michelsen, M. L., *Solution of Differential Equation Models by Polynomial Approximation*, Prentice-Hall, Englewood Cliffs (1980).
16. Schwarz, H. R., *Methode der finiten Elemente* (The Finite Elements Method), Second Edition, B. G. Teubner Verlag, Stuttgart (1984).
17. Deuflhard, P., *Numerik von Anfangswertmethoden für gewöhnliche Differentialgleichungen* (Numerical Initial Value Methods for Ordinary Differential Equations), FU Berlin, TR 89-2 (1989).
18. Dongarra, J. J., *et al.*, *LINPACK User's Guide*, SIAM, Philadelphia (1979).
19. IMSL, *User's Manual*, IMSL Mathematics Library, Houston (1989).
20. Duff, I. S. and Reid, J. K., *ACM Trans. Math. Software* **5**, pp. 18–35 (1979).
21. Duff, I. S., Erisman, A. M., and Reid, J. K., *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford (1987).
22. Stadtherr, M. A. and Wood, E. S., *Comput. Chem. Eng.* **8**, pp. 9–33 (1984).
23. Gilbert, J. and Peierls, T., *SIAM J. Sci. Stat. Comp.* **9**, pp. 862–874 (1988).
24. Osterby, O. and Zlatev, Z., *Direct Methods for Sparse Matrices* Daimi PB-123, Computer Science Department, Aarhus University (1980).
25. Ng. E., A Comparison of Some Methods for Solving Sparse Nonsymmetric Linear Systems, Oak Ridge National Laboratory, unpublished draft (1991).
26. Saadi, Y. and Schultz, M. H., *SIAM J. Sci. Stat. Comput.* **7**, pp. 856–870 (1986).
27. Crowe, C. M. and Nishio, M., *AIChE J.* **21**, pp. 3–12 (1975).
28. Broyden, C. G., *Math. Comp.* **19**, pp. 577–593 (1965).
29. Schubert, L. K., *Math. Comp.* **25**, pp. 27–39 (1970).
30. Bogle, I. D. L. and Perkins, J. D., *Comp. Chem. Eng.* **12**, pp. 791–812 (1988).
31. Deuflhard, P., *Newton Techniques for Highly Nonlinear Problems—Theory, Algorithms, Codes*, Academic Press, New York (in print).
32. Shacham, M., *Comp. Chem. Eng.* **9**, pp. 103–124 (1985).
33. Sun, E. T. and Stadtherr, M. A., *Comp. Chem. Eng.* **12**, pp. 1129–1145 (1988).
34. Stoer, J., in *Computational Mathematical Programming*, Klaus Schittkowski, Ed., NATO ASI Series, Springer Verlag, Berlin (1985).
35. Char, B. W., *et al.*, *MAPLE Reference Manual*, Watcom Publ., Ltd. (1988).
36. *MACSYMA Reference Manual*, Symbolics, Inc. (1988).
37. Deuflhard, P., *SIAM Rev.* **27**, pp. 505–535 (1988).
38. Gear, C. W., *IEEE Trans. Circuit Theory CT-18*, pp. 89–95 (1971).
39. Pantelides, C. C., *SIAM J. Sci. Stat. Comput.* **9**, pp. 213–231 (1988).
40. Petzold, L. R., *SIAM J. Sci. Stat. Comput.* **3**, pp. 367–384 (1982).
41. Gear, C. W. and Petzold, L. R., *SIAM J. Numer. Anal.* **21**, pp. 716–728 (1984).
42. Hindmarsh, A. C. and Johnson, S. H., *Chem. Eng. Sci.* **43**, pp. 3235–3258 (1988).
43. Brenan, K. E., Campbell, S. L., and Petzold, L. R., *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, North-Holland Publ. Co., Amsterdam (1989).
44. Hindmarsh, A. C., *ACM-Signum Newsletter* **15**, pp. 10–27 (1980).
45. Bachmann, R., Mrziglod, T., *et al.*, in *DECHEMA-Monogr.* **116**, VCh, Weinheim (1989).
46. Bachmann, R., Mrziglod, T., *et al.*, *Comp. Chem. Eng.* **14**, pp. 1271–1273 (1990).
47. Deuflhard, P., Hairer, E., and Zugck, J., *Numer. Math.* **51**, 51, pp. 501–516 (1987).
48. Brenan, K. E. and Petzold, L. R., *SIAM J. Numer. Anal.* **26**, pp. 976–996 (1989).
49. Ascher, U. M. and Petzold, L. R., *SIAM J. Numer. Anal.* **28**, pp. 1097–1120 (1991).
50. Hairer, E., Lubich, C., and Roche, M., *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods*, Springer Verlag, Berlin (1989).
51. Kröner, A., *et al.*, *Comp. Chem. Eng.* **14**, pp. 1289–1295 (1990).
52. Perkins, J. D. and Sargent, R. W. H., *AIChE Symp. Ser.* **214**, pp. 1–11 (1982).