In mathematical computations on a computer, errors are introduced into the solutions. These errors are brought into a calculation in three ways:

1. Error is present at the outset in the original data—*inherent error*.
2. Error results from replacing an infinite process by a finite one—*truncation error*, i.e., representing a function by the first few terms of a Taylor series expansion.
3. Error arises as a result of the finite precision of the numbers that can be represented in a computer—*round-off error*.

Each of these errors is unavoidable in a calculation, and hence the problem is not to prevent their occurrence, but rather to control their magnitude. The control of inherent error is not within the scope of this text, and the truncation errors pertaining to specific methods are discussed in the appropriate chapters. This section outlines computer arithmetic and how it influences round-off errors.

**Computer Number System**

The mathematician or engineer, in seeking a solution to a problem, assumes that all calculations will be performed within the system of real numbers, \( \mathbb{R} \). In \( \mathbb{R} \), the interval between any two real numbers contains infinitely many real numbers. \( \mathbb{R} \) does not exist in a computer because there are only a finite amount
of real numbers within a computer's number system. This is a source of round-off error. In computer memory, each number is stored in a location that consists of a sign (±) plus a fixed number of digits. A discussion of how these digits represent numbers is presented next.

NORMALIZED FLOATING-POINT NUMBER SYSTEM

A floating-point number system is characterized by four parameters:

\[
\begin{align*}
\beta &= \text{number base} \\
t &= \text{precision} \\
L, U &= \text{exponent range}.
\end{align*}
\]

One can denote such a system by

\[
F(\beta, t, L, U)
\]

Each floating-point number, \(x \neq 0\), in \(F\) is represented in the following way:

\[
x = \pm \left( \frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \ldots + \frac{d_t}{\beta^t} \right) \times \beta^e \tag{A.1}
\]

where

\[
\begin{align*}
1 &\leq d_1 < \beta \\
0 &\leq d_s < \beta, \quad 2 \leq s \leq t, \\
L &\leq e \leq U
\end{align*}
\]

The fact that \(d_1 \neq 0\) means that the floating-point number system is normalized.

ROUND-OFF ERRORS

Next, consider the differences between computations in \(F\) versus \(\mathcal{R}\), i.e., round-off errors. The source of the differences lies in the fact that \(F\) is not closed under the arithmetic operations of addition and multiplication (likewise, subtraction and division); the sum or the product of two numbers in \(F\) may not necessarily be an element of \(F\). Hence, to stay in \(F\), the computer replaces the "true" result of an operation by an element of \(F\), and this process produces some error. Several cases can occur [A.4]:

1. The exponent \(e\) of the result can lie outside the range \(L \leq e \leq U\),
   (a) If \(e > U\), overflow; for example, in \(F(2, 3, -1, 2)\)
   \[
   (0.100 \times 2^2) \times (0.110 \times 2^3) = 0.110 \times 2^3 \\
   2 \times 3 = 6
   \]
(b) If $e < L$, underflow; for example, in $F(2, 3, -1, 2)$

$$
(0.100 \times 2^0) \times (0.110 \times 2^{-1}) = 0.110 \times 2^2
$$

$$
\frac{1}{2} \times \frac{3}{8} = \frac{3}{16}
$$

2. The fractional part has more than $t$ digits; for example, consider $F(2, 3, -1, 2)$

$$
(0.110 \times 2^0) + (0.111 \times 2^0) = 0.1101 \times 2^1
$$

$$
\frac{3}{4} + \frac{7}{8} = \frac{13}{8}
$$

(notice that four digits are required to represent the fractional part). Similarly,

$$
(0.111 \times 2^0) \times (0.110 \times 2^0) = 0.10101 \times 2^0
$$

$$
\frac{7}{8} \times \frac{3}{4} = \frac{21}{32}
$$

(while this situation does not arise frequently in addition, it almost invariably does with multiplication).

To define a result that can be represented in the machine, the computer selects a nearby element of $F$. This can be done in two ways: rounding and chopping. Suppose the "true" result of an operation is

$$
\left(\frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \ldots + \frac{d_t}{\beta^t} + \frac{d_{t+1} + \frac{1}{2} \beta}{\beta^{t+1}} + \ldots \frac{d_w}{\beta^w}\right) \times \beta^e
$$

then,

1. Chopping: digits beyond $(d_t)/(\beta^t)$ are dropped.
2. Rounding:

$$
\left(\frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \ldots + \frac{d_{t+1} + \frac{1}{2} \beta}{\beta^{t+1}}\right) \times \beta^e
$$

then chop.

For example, if one considers $F(2, 3, -1, 2)$, the number

$$
0.1101 \times 2^1: \text{chopping}
$$

$$
0.111 \times 2^1: \text{rounding},
$$

while for

$$
0.101 \times 2^0: \text{chopping}
$$

$$
0.101 \times 2^0: \text{rounding}.
$$
Both methods are commonly used on present-day computers. No matter the method, there is some round-off error introduced by the process. If \( f(x) \) represents the machine representation of \( x \), then
\[
\sigma(x) = \text{relative round-off error} = \frac{|x - f(x)|}{x}, \quad x \neq 0
\]

It can be shown that \([A.1]\)
\[
\sigma(x) \leq \text{EPS} = \begin{cases} 
\beta^{1-t}; & \text{chopping} \\
\frac{1}{2} \beta^{2-t}; & \text{rounding}
\end{cases}
\]

As an example, suppose \( x = 12.467 \) with \( F(10, 4, -50, 50) \) and chopping, then \( f(x) = 0.1246 \times 10^2 \) and
\[
\sigma(x) = \left| \frac{12.467 - 0.1246 \times 10^2}{12.467} \right|
\]
or
\[
\sigma(x) = 0.00056 < \text{EPS} = 10^{-3}
\]

For the same system with rounding, \( f(x) = 0.1247 \times 10^{-2} \) and
\[
\sigma(x) = 0.00024 < \text{EPS} = \frac{1}{2} \times 10^{-3}
\]

One can see that the parameter EPS plays an important role in computation with a floating-point number system. EPS is the machine epsilon and is defined to be the smallest positive machine number such that
\[
f(1 + \text{EPS}) > 1,
\]

For example, for \( F(10, 4, -50, 50) \) with chopping
\[
\text{EPS} = 10^{-3} \quad \text{since} \quad f(1 + 0.001) = 0.1001 \times 10^1 > 1
\]
and for rounding
\[
\text{EPS} = 0.0005 \quad \text{since} \quad f(1 + 0.0005) = 0.1001 \times 10^1 > 1
\]

The machine epsilon is an indicator of the attainable accuracy in a floating-point number system and can be used to determine the maximum achievable accuracy of computation.

Take, as a specific example, an IBM 3032 and find EPS. Considering only floating-point number systems, the IBM 3032 uses either of two base 16 systems:

1. \( F_s(16, 6, -64, 63) \): single precision
2. \( F_D(16, 14, -64, 63) \): extended precision

For chopping (\( \text{EPS} = \beta^{1-t} \)):

- \( \text{EPS (single)} = 9.54 \times 10^{-7} \)
- \( \text{EPS (extended)} = 2.22 \times 10^{-16} \)
If one executes the following algorithm (from Forsythe, Malcolm, and Moler [A.1]):

```
DOUBLE PRECISION EPS, EPS1
EPS = 1.0D0
10 EPS = 0.5D0*EPS
     EPS1 = EPS + 1.0D0
     IF (EPS1, GT. 1.0D0) GO TO 10
     WRITE (6,20) EPS
20 FORMAT (5X, 'THE MACHINE EPSILON = ', D17.10)
     STOP
END
```

the result is:

```
THE MACHINE EPSILON = 0.1110223025 D-15
```

This method of finding EPS can differ from the "true" EPS by at most a fraction of 2 (EPS is continually halved in statement number 10). Notice that the calculated value of EPS is half of the value predicted by \( EPS = \beta^{1-t} \), as one would expect. In the course of carrying out a computer calculation of practical magnitude, a very large number of arithmetic operations are performed, and the errors can propagate. It is, therefore, wise to use the number system with the greatest precision.

Another computational problem involving the inability of the computer to represent numbers of \( \mathcal{R} \) in \( F \) is shown below. Take for instance the number 0.1, which is used frequently in the partition of intervals, and consider whether ten steps of length 0.1 are the same as one step of length 1.0. If one executes the following algorithm on an IBM 3032:

```
DOUBLE PRECISION X
X = 0.D0
N = 0
DO 10 I = 1,10
     X = X + 0.1D0
10 CONTINUE
     IF (X.EQ.1.0D0) N = 1
     WRITE(6,20) N,X
20 FORMAT (10X,' THE VALUE OF N = ',I1,/,10X,  
     ' THE VALUE OF X = ',D17.10)
     STOP
END
```

the result is:

```
THE VALUE OF N = 0
THE VALUE OF X = 0.1000000000D 01.
```
Since the printed value of $x$ is exactly 1.0, then why is the value of $N$ still equal to zero? The answer to this question is as follows. The IBM computer operates with $\beta$ being a power of 2, and because of this, the number 0.1 cannot be exactly represented in $F$ (0.1 does not have a terminating expansion of $\frac{1}{2}$). In fact,

$$\frac{1}{10} = \frac{0}{2} + \frac{0}{2^2} + \frac{0}{2^3} + \frac{1}{2^4} + \frac{1}{2^5} + \frac{0}{2^6} + \frac{0}{2^7} + \ldots$$

or

$$(0.1)_{10} = (0.000110011001100\ldots)_{2} = (0.19999\ldots)_{16}$$

The base 2 or base 16 representations are terminated after $t$ digits since the IBM chops when performing computations, and when ten of these representations of 0.1 are added together, the result is not exactly 1.0. This is why $N$ was not set equal to 1 in the above algorithm. Why then is the printed value of $x$ equal to 1.0? The IBM machine chops when performing computations, but then rounds on output. Therefore, it is the rounding procedure on output that sets $x$ exactly equal to 1.0.

The programmer must be aware of the subtleties discussed in this appendix and many others, which are described in Chapter 2 of [A.1], for effective implementation of computational algorithms.
Systems of nonlinear algebraic equations arise in the discretization of differential equations. In this appendix, we illustrate a technique for solving systems of nonlinear algebraic equations. More detailed discussions of this topic can be found in [A.1–A.4].

Consider the set of nonlinear algebraic equations

\[ f_1(y_1, y_2, \ldots, y_n) = 0 \]
\[ f_2(y_1, y_2, \ldots, y_n) = 0 \]
\[ \vdots \]
\[ f_n(y_1, y_2, \ldots, y_n) = 0 \]  \hspace{1cm} \text{(B.1)}

which can be written as

\[ f_i(y) = 0, \quad i = 1, 2, \ldots, n, \]

or

\[ f(y) = 0 \]

We wish to find that set \( \{y_i| i = 1, \ldots, n\} \) that satisfies (B.1).

Although there are many ways to solve Eq. (B.1), the most common method of practical use is Newton's method (or variants of it). In the case of a single equation, the Newton method consists in linearizing the given equation \( f(y) = 0 \) by approximating \( f(y) \) by

\[ f(y^0) + f'(y^0)(y - y^0) \]  \hspace{1cm} \text{(B.2)}
where \( y^0 \) is believed to be close to the actual solution, and solving the linearized equation
\[
f(y^0) + f'(y^0) \Delta y = 0 \quad (B.3)
\]
The value \( y^1 = y^0 + \Delta y \) is then accepted as a better approximation, and the process is continued if necessary.

Now consider the system \((B.1)\). If the \( i \)th equation is linearized, then
\[
f_i(y_1^k, y_2^k, \ldots, y_n^k) + \sum_{j=1}^{n} \left[ \frac{\partial f_i}{\partial y_j} \right]_{k} (y_j^{k+1} - y_j^k) = 0 \quad (B.4)
\]
where \( k \geq 0 \). The Jacobian is defined as
\[
J_k^i = \left. \frac{\partial f_i}{\partial y_j} \right|_k \quad (B.5)
\]
and \((B.4)\) can be written in matrix form as
\[
J^k \Delta y = -f(y^k) \quad (B.6)
\]
where
\[
\Delta y_j = y_j^{k+1} - y_j^k \\
\Delta y = (\Delta y_1, \Delta y_2, \ldots, \Delta y_n)^T
\]
The procedure is

1. Choose \( y^0 \)
2. Calculate \( \Delta y \) from \((B.6)\)
3. Set \( y^{k+1} = y^k + \Delta y \)
4. Iterate on (2) and (3) until
\[
||\Delta y||_\infty < \text{TOL}
\]
where
\[
||x||_\infty = \max x_i \\
\text{TOL} = \text{arbitrary}
\]
The convergence of the Newton method is proven in [A.2] under certain conditions, and it is shown that the method converges quadratically, i.e.,
\[
||g^* - y^{k+1}||_\infty < m \ ||g^* - y^{k}||_\infty^2 \quad (B.7)
\]
where
\[
f(g^*) = 0
\]
and
\[
m = \text{a constant}
\]
From the main body of this text one can see that all the methods for solving differential equations can yield large sets of equations that can be formulated into a matrix problem. Normally, these equations give rise to a matrix having a special property in that a great many of its elements are zero. Such matrices are called sparse. Typically, there is a pattern of zero and nonzero elements, and special matrix methods have been developed to take these patterns into consideration. In this appendix we begin by discussing a method for solving a general linear system to equations and then proceed by outlining a method for the tridiagonal matrix.

**DENSE MATRIX**

The standard method of solving a linear system of algebraic equations is to do a lower-upper (LU) decomposition on the matrix, or Gaussian elimination. Consider a dense (all elements are nonzero), nonsingular (all rows or columns are independent) \( n \times n \) matrix \( A \) such that

\[
Ax = r
\]
where

\[ x = [x_1, x_2, \ldots, x_n]^T \]
\[ r = [r_1, r_2, \ldots, r_n]^T \]
\[ A = \begin{bmatrix}
  a_{11} & a_{12} & \cdots & a_{1n} \\
  a_{21} & \ddots & & \\
  \vdots & & \ddots & \\
  \vdots & & & \ddots \\
  a_{n1} & & & a_{nn}
\end{bmatrix} \]

The 21 element can be made zero by multiplying the first row by \(-a_{21}/a_{11}\) and adding it to the second row. By multiplying the first row by \(-a_{31}/a_{11}\) and adding to the third row, the 31 element becomes zero. Likewise,

\[
A^{[1]}x = \begin{bmatrix}
  a_{11} & a_{12} & a_{13} & \cdots \\
  0 & a_{22} - \frac{a_{21}}{a_{11}} & a_{23} - \frac{a_{21}}{a_{11}} a_{13} & \cdots \\
  0 & a_{32} - \frac{a_{31}}{a_{11}} & a_{33} - \frac{a_{31}}{a_{11}} a_{13} & \cdots \\
  \vdots & \vdots & \vdots & \ddots \\
  \vdots & \vdots & \vdots & \ddots \\
  \vdots & \vdots & \vdots & \ddots
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
\vdots \\
x_n
\end{bmatrix} = \begin{bmatrix}
r_1 \\
r_2 - \frac{a_{21}}{a_{11}}r_1 \\
r_3 - \frac{a_{31}}{a_{11}}r_1 \\
\vdots \\
\vdots \\
\vdots
\end{bmatrix} = r^{[1]}
\]

\[ (C.2) \]

In sequel this is

\[
a_{i,j}^{[k]} = a_{i,j}^{[k-1]} - \frac{a_{i,k-1}^{[k-1]}}{a_{k-1,k-1}^{[k-1]}} a_{k-1,j}^{[k-1]} \\ r_i^{[k]} = r_i^{[k-1]} - \frac{a_{i,k-1}^{[k-1]}}{a_{k-1,k-1}^{[k-1]}} r_{k-1}
\]

\[ (C.3) \quad (C.4) \]

Now make a column of zeros below the diagonal in the second column by doing the same process as before

\[
A^{[2]}x = \begin{bmatrix}
  a_{11} & a_{12} & a_{13} & \cdots \\
  0 & a_{22} & a_{23} & \cdots \\
  0 & 0 & a_{33} & \cdots \\
  \vdots & \vdots & \vdots & \ddots \\
  \vdots & \vdots & \vdots & \ddots \\
  \vdots & \vdots & \vdots & \ddots
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
\vdots \\
x_n
\end{bmatrix} = \begin{bmatrix}
r_1 \\
r_2^{[2]} \\
r_3^{[2]} \\
\vdots \\
\vdots \\
\vdots
\end{bmatrix}
\]

\[ (C.5) \]
Continue the procedure until the lower triangle is filled with zeros and set \( A^{[n]} = U \).

\[
U_x = A^{[n]}x = \begin{bmatrix}
a_{11} & a_{12} & a_{13} & a_{14} & \cdots \\
a_{22} & a_{23} & a_{24} & \cdots \\
a_{33} & a_{34} & \cdots \\
0 & \cdots
\end{bmatrix} \begin{bmatrix}x_1 \\
x_2 \\
\vdots
\end{bmatrix} = \begin{bmatrix}r_1 \\
r_2 \\
\vdots
\end{bmatrix} \tag{C.6}
\]

Define \( L \) as the matrix with zeros in the upper triangle, ones on the diagonal, and the scalar multiples used in the lower triangle to create \( U \),

\[
L = \begin{bmatrix}
1 & & & \\
-\frac{a_{21}}{a_{11}} & 1 & & \\
-\frac{a_{31}}{a_{11}} - \frac{a_{22}^{[2]}}{a_{22}} & \frac{a_{22}^{[2]}}{a_{22}} & 1 \\
\vdots & \vdots & \vdots & \ddots \\
-\frac{a_{n-1}}{a_{n-1,n-1}} & -\frac{a_{n-1}}{a_{n-1,n-1}} & \cdots & 1
\end{bmatrix} \tag{C.7}
\]

If the unit diagonal is understood, then \( L \) and \( U \) can be stored in the same space as \( A \). The solution is now obtained by

\[
x_n = \frac{r_n^{[n]}}{a_{nn}} \\
x_{n-1} = \frac{r_{n-1}^{[n-1]} - a_{n-1,n}^{[n-1]} x_n}{a_{n-1,n-1}^{[n-1]}} \tag{C.8}
\]

\[
x_i = \frac{r_i^{[i]} - \sum_{j=i+1}^{n} a_{i,j}^{[i]} x_j}{a_{i,i}^{[i]}}
\]

It is possible to show that \( A = LU \) \([A.5]\). Thus (C.1) can be represented as

\[
Ax = LUx = r \tag{C.9}
\]
Notice that the sequence

\[ L y = r \]  \hspace{1cm} (C.10)

\[ U x = y \]  \hspace{1cm} (C.11)

gives (C.9) by multiplying (C.11) from the left by \( L \) to give

\[ L U x = L y \]  \hspace{1cm} (C.12)

or

\[ A x = L U x = r \]

One can think of Eq. (C.3) as the LU decomposition of \( A \), Eq. (C.4) as the forward substitution or the solution of (C.10), and Eq. (C.8) as the backward elimination or the solution of (C.11). If a certain problem has a constant matrix \( A \) but different right-hand sides \( r \), then the matrix need only be decomposed once and the iteration would only involve the forward and backward sweeps.

The number of multiplications and divisions needed to do one LU decomposition and \( m \)-forward and backward sweeps for a dense matrix is

\[ \text{OP}_{\text{G.E.}} = \frac{1}{3} n^3 - \frac{1}{3} n + mn^2 \]  \hspace{1cm} (C.13)

This is fewer operations than it takes to calculate an inverse, so the decomposition is more efficient. Notice that the decomposition is proportional to \( n^3 \), whereas the forward and backward sweeps are proportional to \( n^2 \). For large \( n \) the decomposition is a significant cost.

The only way that Gaussian elimination can become unstable and the process break down when \( A \) is nonsingular is if \( a_{ji}^{[-1]} = 0 \) before performing step \( i \) of the decomposition. Since the procedure is being performed on a computer, round-off errors can cause \( a_{ji}^{[-1]} \) to be "close" to zero, likewise, causing instabilities. Often this round-off error problem can be avoided by pivoting; that is, find row \( s \) such that \( \max_{i=\{j\in\{i\}\}} |a_{si}^{[-1]}| = |a_{ji}^{[-1]}| \) and switch row \( s \) and row \( i \) before performing the \( i \)th step. To avoid pivoting, we must impose that matrix \( A \) be diagonally dominant:

\[ |a_{ii}| \geq \sum_{j=1}^{n} |a_{ij}|, \quad i = 1, \ldots, n, \]  \hspace{1cm} (C.14)

where the strict inequality must hold for at least one row. Condition (C.14) insures that \( a_{ji}^{[-1]} \) will not be "close" to zero, and therefore the Gaussian elimination procedure is stable and does not require pivoting.
TRIDIAGONAL MATRIX

The LU decomposition of a tridiagonal matrix is performed by Gaussian elimination. A tridiagonal matrix can be written as

\[
\begin{bmatrix}
b_1 & c_1 & & & \\
a_2 & b_2 & c_2 & & \\
& a_3 & b_3 & c_3 & \\
& & \ddots & \ddots & \ddots \\
& & & a_{n-1} & b_{n-1} & c_{n-1} \\
& & & & a_n & b_n \\
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_i \\
\vdots \\
x_n \\
\end{bmatrix}
\begin{bmatrix}
r_1 \\
r_2 \\
\vdots \\
r_i \\
\vdots \\
r_n \\
\end{bmatrix}
\tag{C.15}
\]

The Thomas algorithm (Gaussian elimination which takes into account the form of the matrix) is

\[
\alpha_1 = \frac{c_1}{b_1} \tag{C.16}
\]

\[
\gamma_1 = \frac{r_1}{b_1} \tag{C.17}
\]

\[
\alpha_i = \frac{c_i}{b_i - a_i\alpha_{i-1}} \quad i = 2, 3, \ldots, n \tag{C.18}
\]

\[
\gamma_i = \frac{r_i - a_i\gamma_{i-1}}{b_i - a_i\alpha_{i-1}} \quad i = 2, 3, \ldots, n \tag{C.19}
\]

with

\[
c_n = 0,
\]

and

\[
x_n = \gamma_n \tag{C.20}
\]

\[
x_i = \gamma_i - \alpha_i x_{i+1}, \quad i = n - 1, n - 2, \ldots, 1 \tag{C.21}
\]

Equations (C.18) and (C.19) are the LU decomposition and forward substitution, and Eq. (C.21) is the backward elimination. The important point is that there is no fill outside the tridiagonal matrix (structure remains the same). This is an advantage in reducing the work and storage requirements. The operation count
to solve $m$ such systems of size $n$ is

$$\text{OP}_{TD} = 2(n - 1) + m(3n - 2),$$

which is a significant savings over $(\frac{1}{3})n^3$ of (C.13). Since this algorithm is a special form of Gaussian elimination without pivoting, the procedure is stable only when the matrix possesses diagonal dominance.
In finite element methods the manner in which the approximate numerical solution of a differential equation is represented affects the entire solution process. Specifically, we would like to choose an approximating space of functions that is easy to work with and is capable of approximating the solution accurately. Such spaces exist, and bases for these spaces can be constructed by using B-splines [A.6]. The authoritative text on this subject is by deBoor [A.6].

Before defining the B-splines, one must first understand the meaning of a divided difference and a truncated power function. The first-order divided difference of a function $g(x)$, $x_i \leq x \leq x_{i+1}$, is

$$g[x_i, x_{i+1}] = \frac{g(x_{i+1}) - g(x_i)}{x_{i+1} - x_i} \quad (D.1)$$

while the higher-order divided difference (dd) formulas are given by recursion formulas: the $r$th-order dd of $g(x)$ on the points $x_p, x_{i+1}, \ldots, x_{i+r}$ is

$$g[x_p, x_{i+1}, \ldots, x_{i+r}] = \frac{g[x_{i+1}, \ldots, x_{i+r}] - g[x_p, \ldots, x_{i+r-1}]}{x_{i+r} - x_i} \quad (D.2)$$

where

$$g[x_p, \ldots, x_{i+r-1}] = \frac{g[x_{i+1}, \ldots, x_{i+r-1}] - g[x_p, \ldots, x_{i+r-2}]}{x_{i+r-1} - x_i}$$

$$\vdots$$

$$g[x_p, x_{i+1}, x_{i+2}] = \frac{g[x_{i+1}, x_{i+2}] - g[x_p, x_{i+1}]}{x_{i+2} - x_i}$$
Notice that the $i$th-order dd of a function is equal to its $i$th derivative at some point times a constant. Next, define a truncated power function of order $r$ (degree $= r - 1$) as:

$$\phi_r^i(t) = (x - t)^{r-1}_+ = \begin{cases} (x - t)^{r-1}, & x \geq t \\ 0, & x < t \end{cases}$$  \hspace{1cm} (D.3)

This function is illustrated in Figure D.1. The function and all of its derivatives except for the $(r - 1)$st are continuous [(r - 1)th derivative is not continuous at $x = t$]. Now, let the sequence $t_0, \ldots, t_{j+r}$ of $r + 1$ points be a nondecreasing sequence and define

$$Z_j^i(x) = \phi_r^i [t_0, \ldots, t_{j+r}] \hspace{1cm} (D.4)$$

Thus, $Z_j^i(x) = 0$, when $t_0, \ldots, t_{j+r}$ is not in an interval containing $x$, and when the interval does contain $x$, $Z_j^i(x)$ is a linear combination of terms $\phi_r^i(t) = (x - t)^{r-1}_+$ evaluated at $t = t_0, \ldots, t_{j+r}$, that is, the linear combination that results from the dd's.

The $i$th B-spline of order $k$ for the sequence $t = \{t_i \mid i = 1, \ldots, k\}$ (called a knot sequence) is denoted by $B_{i,k,t}$ and is defined by

$$B_{i,k,t}(x) = (-1)^k(t_{i+k} - t_i)Z_i^k(x) \hspace{1cm} (D.5)$$

If $k$ and $t$ are understood, then one can write $B_i(x)$ for $B_{i,k,t}(x)$. The main properties of $B_i(x)$ are:

1. Each $B_i(x) = 0$ when $x < t_i$ or $x > t_{i+k}$ (local support).
2. \[\sum_{i} B_i(x) = 1, \text{ specifically } \sum_{i=q+1-k}^{s} B_i(x) = 1 \text{ for } t_q \leq x \leq t_s.\]
3. Each $B_i(x)$ satisfied $0 \leq B_i(x) \leq 1$ for $t_i \leq x \leq t_{i+k}$ [normalized by the term $(t_{i+k} - t_i)$ in (D.5)] and possesses only one maximum.

![FIGURE D.1 Truncated power function of order r.](image-url)
Consider the approximating space $\mathcal{P}_k^\nu(\pi)$ (described in Chapter 3) and whether the B-splines can serve as a basis for this space. Given the partition $\pi$,

$$a = x_1 < x_2 < \ldots < x_{\ell+1} = b,$$

and the nonnegative integer sequence $\nu = \{v_j | j = 2, \ldots, \ell\}$, which denotes the continuity at the breakpoints $x_i$, then the dimension of $\mathcal{P}_k^\nu(\pi)$ is given by

$$N = \dim \mathcal{P}_k^\nu(\pi) = \sum_{j=1}^{\ell} (k - v_j)$$

with $v_1 = 0$. If $t = \{t_1| i = 1, \ldots, N + k\}$ such that

$$t_1 \leq t_2 \leq \ldots \leq t_k \leq x_1$$

(makes the first B-spline one at $x_1$)

$$x_{\ell+1} \leq t_{N+1} \leq \ldots \leq t_{N+k}$$

(makes the last B-spline one at $x_{\ell+1}$)

and if for $i = 2, \ldots, \ell$, the number $x_i$ occurs exactly $k - v_i$ times in the set $t$, then the sequence $B_i(x)$, $i = 1, \ldots, N$ is a basis for $\mathcal{P}_k^\nu(\pi)$ [A.6]. Therefore a function $f(x)$ can be represented in $\mathcal{P}_k^\nu(\pi)$ by

$$f(x) = \sum_{i=1}^{N} \alpha_i B_i(x)$$

The B-splines have many computational features that are described in [A.6]. For example, they are easy to evaluate. To evaluate a B-spline at a point $x$, $t_j \leq x \leq t_{j+1}$, the following algorithm can be used [A.7] [let $B_{i,k,t}(x)$ be denoted by $B_{i,k}$ and $Z_i^k(x)$ by $Z_{i,k}$]:

$$B_{i,1} = 1$$

DO 20 $\ell = 1, \ldots, k - 1$

$$B_{i-\ell, \ell+1} = 0$$

DO 10 $j = 1, \ldots, \ell$

$$Z_{i+j-\ell, \ell} = B_{i+j-\ell, \ell}(t_{i+j} - t_{i+1} - \ell)$$

$$B_{i+j-\ell-1, \ell+1} = B_{i+j-\ell-1, \ell+1} + (t_{i+j} - x)Z_{i+j-\ell-1, \ell}$$

$$B_{i+j-\ell, \ell+1} = (x - t_{i+j-\ell})Z_{i+j-\ell, \ell}$$

10 CONTINUE

20 CONTINUE

Thus B-splines of lower order are used to evaluate B-splines of higher order. A complete set of algorithms for computing with B-splines is given by deBoor [A.6].

A B-spline $B_{i,k,t}(x)$ has support over $[t_i, \ldots, t_{i+k}]$. If each point $x_i$ appears only once in $t$, that is, $v_i = k - 1$, then the support is $k$ subintervals, and the B-spline has continuity of the $k - 2$ derivative and all lower-order derivatives.
To decrease the continuity, one must increase the number of times \( x_i \) appears in \( t \) (this also decreases the support). This loss in continuity results from the loss in order of the \( \text{dd}'s \). To illustrate this point, consider the case of quadratic B-splines \((k = 3)\) corresponding to the knots \( \{0, 1, 1, 3, 4, 6, 6, 6\} \) on the partition \( x_1 = 1, x_2 = 3, x_3 = 4, x_4 = 6 \). For this case notice that \((k = 3, \ell = 3): \)

\[
\begin{align*}
v_2 &= 2, \\
v_3 &= 2 \\
N &= \dim \mathcal{B}_k^p(\pi) = \sum_{j=1}^{3} (3 - \nu_j) = 5 \\
t_1 &\leq \ldots \leq t_k \leq x_1 \quad (0 \leq 1 \leq 1)
\end{align*}
\]

and

\[
x_{\ell+1} \leq t_{N+1} \leq \ldots \leq t_{N+k} \quad (6 \leq 6 \leq 6)
\]

Therefore, there are five B-splines to be calculated, each of order 3. Figure D.2 (from [A.6]) illustrates these B-splines. From this figure one can see the normal parabolic spline, \( B_3(x) \), which comes from the fact that all the knots are distinct. Also, the loss of continuity in the first derivative is illustrated, for example, \( B_2(x) \) at \( x = 1 \) due to the repetition of \( x_1 \).

When using B-splines as the basis functions for finite element methods, one specifies the order \( k \) and the knot sequence. From this information, one can calculate the basis if continuity in all derivatives of order lower than the order of the differential equation is assumed.

---

Consider the solution of the linear algebraic system

\[ Ax = b \tag{E.1} \]

where \( A \) is a given real \( N \times N \) matrix, \( b \) is a given \( N \) component vector, and \( x \) is the unknown vector to be determined. In this appendix we are concerned with systems in which \( N \) is large and \( A \) is sparse. Linear systems of this type arise from the numerical solution of partial differential equations.

After discretizing a partial differential equation by either finite difference or finite element techniques, one is faced with the task of solving a large sparse system of linear equations of form (E.1). Two general procedures used to solve systems of this nature are direct and iterative methods. Direct methods are those that, in the absence of round-off errors, yield the exact solution in a finite number of arithmetic operations. An example of a direct method is Gaussian elimination. Iterative methods are those that start with an initial guess for the solution and that, by applying a suitably chosen algorithm, lead to better approximations. In general, iterative methods require less storage and fewer arithmetic operations than direct methods for large sparse systems (for a comparison of direct and iterative methods, see [A.8]).

An iterative method for the solution of linear systems is obtained by splitting the matrix \( A \) into two parts, say

\[ A = S - T \tag{E.2} \]

To solve (E.1) define a sequence of vectors \( x^\ell \) by

\[ Sx^{\ell+1} = Tx^\ell + b, \quad \ell = 0, 1, \ldots \tag{E.3} \]
where \( x^0 \) is specified. If the sequence of vectors converges, then the limit will be a solution of (E.1).

There are three common splittings of \( A \). The first is known as the point Jacobi method and is

\[
S = D \\
T = D - A \tag{E.4}
\]

where the matrix \( D \) is the diagonal matrix whose main diagonal is that of \( A \). In component form the point Jacobi method is

\[
x_{i}^{\ell+1} = - \sum_{j=1 \atop j \neq i}^{N} \frac{a_{ij}}{a_{ii}} x_{j}^{\ell} + \frac{b_{i}}{a_{ii}}, \quad 1 \leq i \leq N, \quad \ell \geq 0 \tag{E.5}
\]

An obvious necessary condition for (E.5) to work is \( a_{ii} \neq 0 \). If \( A \) is diagonally dominant, then the point Jacobi method converges [A.3]. Examination of (E.5) shows that one must save all the components of \( x^{\ell} \) while computing \( x^{\ell+1} \). The Gauss-Seidel method does not possess this storage requirement. The matrix splitting equations for the Gauss-Seidel method are

\[
S = D + L \\
T = -U \tag{E.6}
\]

where \( D \) is as before, and \( U \) and \( L \) are strictly upper and lower triangular matrices, respectively. In component form this method is

\[
x_{i}^{\ell+1} = - \sum_{j=1 \atop j \neq i}^{i-1} a_{ij} x_{j}^{\ell} - \sum_{j=i+1}^{N} a_{ij} x_{j}^{\ell} + \frac{b_{i}}{a_{ii}}, \quad 1 \leq i \leq N, \quad \ell \geq 0 \tag{E.7}
\]

As with the point Jacobi method, \( A \) must be diagonally dominant for the Gauss-Seidel method to converge [A.3]. Also, in most practical problems the Gauss-Seidel method converges faster than the point Jacobi method. The third common method is closely related to the Gauss-Seidel method. Let the vector \( \hat{x}^{\ell+1} \) be defined by

\[
\hat{x}_{i}^{\ell+1} = - \sum_{j=1 \atop j \neq i}^{i-1} a_{ij} \hat{x}_{j}^{\ell+1} - \sum_{j=i+1}^{N} a_{ij} x_{j}^{\ell} + \frac{b_{i}}{a_{ii}}, \quad 1 \leq i \leq N, \quad \ell \geq 0 \tag{E.8}
\]

from which \( x^{\ell+1} \) is obtained as

\[
x_{i}^{\ell+1} = x_{i}^{\ell} + \omega (\hat{x}_{i}^{\ell+1} - x_{i}^{\ell})
\]

or

\[
x_{i}^{\ell+1} = (1 - \omega) x_{i}^{\ell} + \omega \hat{x}_{i}^{\ell+1} \tag{E.9}
\]

The constant \( \omega, 1 \leq \omega \leq 2 \), is called the relaxation parameter, and is chosen to accelerate the convergence. Equations (E.8) and (E.9) can be combined to give

\[
x_{i}^{\ell+1} = x_{i}^{\ell} + \omega \left\{ - \sum_{j=1 \atop j \neq i}^{i-1} \frac{a_{ij}}{a_{ii}} x_{j}^{\ell+1} - \sum_{j=i+1}^{N} \frac{a_{ij}}{a_{ii}} x_{j}^{\ell} + \frac{b_{i}}{a_{ii}} - x_{i}^{\ell} \right\}
\]

\[
1 \leq i \leq N, \quad \ell \geq 0 \tag{E.10}
\]
Notice that if $\omega = 1$, the method is the Gauss-Seidel method. Equation (E.10) can be written in the split matrix notation as

$$S = \frac{1}{\omega} [D + \omega L]$$

$$T = \frac{1}{\omega} [(1 - \omega)D - \omega U]$$

(E.11)

where $D$, $L$, and $U$ are as previously defined. This method is called successive over relaxation (SOR). In the practical use of SOR, finding the optimal $\omega$ is of major importance. Adaptive procedures have been developed for the automatic determination of $\omega$ as the iterative procedure is being carried out (see, for example [A.9]).

Computer packages are available for the solution of large, sparse linear systems of algebraic equations. One package, ITPACK [A.10], contains research-oriented programs that implement iterative methods.

**APPENDIX REFERENCES**


Adams, J., 226
Addison, C.A., 51
Alexander, R., 29, 50
Allen, R.H., 28, 50
Ames, W.F., 174, 175, 227
Aris, R., 126
Ascher, U., 126
Ariz, A.Z., 95
Bank, R.E., 184, 185, 224, 226
Becker, E.B., 126, 175, 227
Bickart, T.A., 51
Bird, R.B., 94, 126, 137
Birkhoff, G., 194, 224
Bleakney, T., 174, 225
Bramble, J.H., 194, 224
Brown, D.R., 94
Brown, H.G., 51
Bui, T.D., 29, 50
Bui, T.R., 29, 50
Bulirsch, R., 94
Burka, M.K., 29, 50
Burrage, K., 51
Butcher, J.C., 51
Butt, J.B., 5, 49, 173
Byrne, G.D., 51, 174, 225
Caillaud, J.B., 28, 33, 42, 43, 44, 50
Calahan, D., 28, 50
Carberry, J.J., 93
Carey, G.F., 126, 147, 173, 175, 227
Carver, M., 174, 225
Childs, B., 95
Chipman, F.H., 51
Christiansen, J., 126
Conte, S.D., 49
Courant, R., 205, 225
Craige, J.A.I., 51
Crowther, C.R., 51
Daniel, J.W., 95
Davenport, S.M., 39, 51
Davis, M.E., 90, 94, 126, 144, 146, 173
deBoor, C., 49, 120, 125, 126, 247
Dendy, J.E., Jr., 214, 225
Denison, K.S., 224
Denman, E., 95
Deuflhard, P., 93, 94
Dew, P.M., 173
Diaz, J.C., 118, 126, 224
Dixon, A.D., 227
Douglas, J., Jr., 142, 172, 214, 225
DuPont, T., 214, 225
Eisenstat, S., 247
England, R., 94
Enright, W.H., 39, 51
Fairweather, G., 90, 94, 125, 126, 146, 173, 175, 214, 225, 227
Fellen, B.M., 39, 51
Finlayson, B.A., 42, 52, 175, 227, 249
Fix, G.J., 126, 175, 225, 227
Forsythe, G.E., 50, 52, 175, 233, 249
Fox, L., 86, 93, 95, 126
Fox, P.A., 174
Froment, G.F., 174
Gaffney, P.W., 51
Gear, C.W., 36, 49, 52
George, A., 247
Gladwell, 1., 51, 95
Gordon, M.K., 49, 52, 173
Griffiths, D.F., 172, 175, 227
Grimes, R.G., 226, 247, 250
Gunn, J.E., 214, 225
Hall, A.D., 52, 126, 174
Hamrin, C.E., 224
Hicks, J.S., 94
Hilbert, S.R., 194, 224, 225
Hindmarsh, A.C., 51, 173, 226
Hornsby, J., 226
Houstics, E.N., 226
Hueber, K.H., 227
Hull, T.E., 39, 51
Hwang, M., 42, 52
Hyman, J.M., 174
Isaacson, E., 95, 175
Jackson, K.R., 51
Johnston, R.L., 37, 49, 52, 247
Jones, B.F., 142, 172
Keast, P., 126
Kehoe, J.P.G., 5, 49, 173
Keller, H.B., 56, 93, 95, 175, 249
Kincaid, D., 226, 249, 250
Kong, A.K., 51, 147, 173
Krogh, F.T., 37, 50
Lambert, J.D., 50, 52
Lapidus, L., 42, 52
Leaf, G.K., 174, 225
Lees, M., 142, 173
Lentini, M., 89, 94
Lightfoot, E.N., 94, 126, 137
Lindberg, B., 39, 51
Liskovets, O.A., 172
MacGregor, W.J., 250
Madsen, M.K., 174
Malcolm, M.A., 50, 52, 233, 249
Melgaard, D., 225
Michelsen, M.L., 33, 37, 42, 50, 52, 174, 226
Minkoff, M., 174, 225
Mitchell, A.R., 172, 175, 214, 225, 227
Moler, C.B., 50, 52, 233, 249
Murray, J.D., 81, 93
Nelson, P., 95
Norsett, S.P., 29, 50
Oden, J.T., 126, 175, 227
Padmanabhan, L., 28, 33, 42, 43, 44, 50
Papatheodorou, T.S., 226
Peaceman, D.W., 214, 225
Pereyra, V., 86, 89, 93, 94
Pincel, Z., 51
Poots, J., 126
Prater, C.D., 226
Prenter, P.M., 195, 225
Price, T.H., 5, 49
Proskurowski, W., 226
Rachford, H.H., 214, 225
Rice, J., 226
Robertson, A.H., 41, 52
Rose, D.J., 184, 185, 224
Rosenbrock, H.H., 50
Russell, R.D., 95, 126, 195, 225, 247
Saliariya, A.K., 94
Saltzman, J., 174, 225
Sayers, D.K., 95
Schissler, W., 174, 225
Schryer, N.L., 126, 174
Schultz, M.H., 194, 224
Scott, M.R., 93, 94, 95
Sedgwick, A.E., 39, 51
Seinfeld, J.H., 42, 52
Sepehrnoori, K., 147, 173
Sewell, G., 21, 226
Shampine, L.F., 39, 49, 50, 51, 52, 173
Sherman, A.H., 226, 249
Sincovec, R.F., 174, 225
Skeel, R.D., 51, 147, 173
Sorensen, D., 174, 225
Stewart, W.E., 94, 126, 137
Strang, G., 126, 175, 225, 227
Swartz, B., 126
Swarztrauber, P., 226
Sweet, N., 226
Author Index

Taylor, J.C., 226
Taylor, J.V., 226
Tendler, J.M., 51
Varah, J.M., 118
Varga, R.S., 93, 95, 126, 185, 191, 194, 224
Verwer, J.G., 51, 173
Villadsen, J., 42, 50, 52, 174, 226
Wait, R., 175, 225, 227
Walsh, J.E., 173
Watson, W.R., 175
Watts, H.A., 39, 50, 51, 52, 93, 94
Weiss, R., 126
Weisz, P.B., 94
Wilkes, J.O., 137, 172
Yamanis, J., 144, 146, 173
Young, D.M., 250
Zienkiewicz, O.C., 227
Adams-Bashforth predictor-corrector method, 24, 25, 27, 39, 40
Adams-Moulton predictor-corrector method, 26, 33, 39, 40
Adiabatic, 34
Alternating direction implicit methods (ADI), 212–214
Annular bed reactor, 144–146, 187–190
Approximating space, 97–106, 109, 194, 200, 201, 243, 245
Autonomous, 28, 34
Axial dispersion, 187–190
Backward difference approximation, 68, 132, 133, 134, 139–142
Backward elimination, 240, 241
Basis functions, 97–103, 154, 158, 159, 194, 200, 201, 205, 207, 208, 210, 211
Batch still, 21–24
Bilinear basis functions, 201–204
Biot number (Bi), 116, 149, 150, 152
Block tridiagonal, 109, 179, 184
BLSODE, 39
Boundary conditions, 53, 54, 128, 129, 135, 136, 152–161, 177, 195, 196
Boundary value problem (BVP), 1, 2, 53–127
BOUNDS, 88
Breakpoints, 100–102
B-splines, 103, 109, 120, 200, 204, 243–246
BVP software, 87–90, 119–123
Centered difference method, 84
Central difference method, 69, 76, 186, 188
Chopping errors, 213–234
Collocation method, 97, 98, 111–123, 130, 158–166, 192, 194–200, 212, 220
Collocation points, 98, 112, 113, 117, 118, 121–123, 158, 161, 192, 195, 197, 198
COLSYS, 119–123
Computer arithmetic and error control, 229–234
Computer number system, 229, 230
Continuity of approximating space, 100–102, 112, 119, 245, 246
Cooling fin, 72–75
Cost of computation, 11, 16, 18
Crank-Nicolson method, 133, 135, 136, 140–142, 144, 146, 215
DD03AD, 88, 90
Subject Index

DD04AD, 88-90
Deferred corrections, 85-89
Dense matrix, 237
DE/ODE, 39
2DEPEP, 215, 219
deroot/ODERT, 39
DGEAR, 40-44, 47
Diagonally dominant, 240, 242, 248
Diffusion, 1, 58-61, 81-83, 128-130, 137
Dimension of approximating space, 100-103, 112, 119, 200, 245
Dirichlet boundary conditions, 68-71, 73, 77, 110, 111, 129, 135, 178, 179, 182, 190, 191, 195, 205
Discrete numerical solution, 3
Discrete variable methods, 53-67
Discriminant of L, 128
DISPL, 163-166, 215-221
Divided difference, 243, 244, 246
Double precision, 232, 233
DSS/2, 163, 215
DVERK, 40-44, 60-62

Effectiveness factor, 58-62, 120-123
Eigenvalue, 30-32, 34, 42, 147
ELIPTI, 220
Elliptic partial differential equations, 128, 145, 146, 177-211, 219-221
ELLPACK, 220
Enzyme-catalyzed reaction, 81-83
EPDEI, 220
EPISODE, 39-44
Error analysis, 4, 5, 9-11, 16, 20, 21, 71, 74, 75, 105, 106, 109, 130, 132, 133, 139, 152, 155, 159, 165, 166, 178, 184, 200, 204, 210, 214, 216, 217
Error oscillations, 9, 10, 20
Error propagation, 9-11
Error tolerance (TOL), 19, 36-38, 41, 43, 44, 60, 62, 83, 87, 88, 90, 122, 123, 152, 153, 165, 166, 217, 236
Euler method (explicit), 4-12, 16, 17, 19, 20, 25, 29, 30, 36, 130-132, 212
Euler method (implicit), 19-21, 23, 28, 32, 35, 36, 132
Exact solution, 3, 4
Explicit methods, 3-19, 25, 27, 32, 41, 44, 130-132, 213
Explicit methods stability criteria, 10, 130-132
Exponent overflow, 230
Exponent underflow, 231

Extrapolation, 21-24
FFT9, 220
Finite difference methods, 53, 67-90, 105, 128-153, 162, 163, 177-192, 211-220
Finite element methods, 97-123, 128, 154-166, 192-211, 214-221, 243, 246
First degree interpolation, 190, 191
First order accurate, 5, 10, 12, 20, 21, 23, 68
First order systems, 54, 55, 57, 60, 62-64, 83-85
FISPACK, 220
Fixed bed reactor, 155-158
Floating point number system, 230-234
Fluid flow, 1, 88-90, 136-141, 196-200, 216-218
Flux boundary conditions, 71-75, 77, 110, 111, 115, 135-137
FORSIM, 163
FORSIM IV, 215
Forward difference approximation, 68, 130-133, 139-142
Forward substitution, 240, 241
Fourier's law of heat conduction, 71
Fourth order accurate, 13, 14, 16-19, 88, 109, 115, 200, 204, 214, 217

Gaussian elimination, 70, 114, 120, 237-242, 247
Gaussian points, 112, 120, 158, 159, 195, 198
Gauss-Seidel method, 248, 249
GEAR, 39, 162, 163, 215
GEARB, 39, 162, 163, 215
GEARIB, 162, 163
GERK, 39
Global error, 4, 7, 9-11, 38
Global methods, 67
Green's first identity, 193
Green's theorem, 185, 188

HARWELL library, 38, 39, 88, 162, 163, 215
Hermite cubic basis, 102, 103, 106-109, 112-117, 194-200, 204
Higher order methods for solving BVP's, 85-87
Higher order methods for solving IVP's, 28, 29, 33
Higher order time approximations, 145-153
HLMHLZ/HELMIT/HELSIX/HELSYM, 220
Hyperbolic partial differential equations, 128
Identity matrix, 33, 134, 179, 180
Implicit backward formula, 27
<table>
<thead>
<tr>
<th>Subject Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implicit methods, 19-21, 25-28, 32, 40, 44, 132, 213</td>
</tr>
<tr>
<td>IMSL (DPDES), 162, 163</td>
</tr>
<tr>
<td>IMSL library, 38-40, 60, 88, 162, 163, 215</td>
</tr>
<tr>
<td>Incompatibility of boundary conditions and initial conditions, 136-140</td>
</tr>
<tr>
<td>Inherent error, 229</td>
</tr>
<tr>
<td>Inhomogeneous media, 75-79, 142-146, 187-190</td>
</tr>
<tr>
<td>Initial conditions, 128-131, 134-137</td>
</tr>
<tr>
<td>Initial value methods for solving BVP's, 53-67</td>
</tr>
<tr>
<td>Initial value problems (IVP), 1-52, 55-57, 59, 61-65, 87, 88, 127, 129, 146, 147, 151, 155-162, 212</td>
</tr>
<tr>
<td>Initial value problem software, 37-39</td>
</tr>
<tr>
<td>Inner products, 90, 104-111, 193, 194</td>
</tr>
<tr>
<td>Integral method, 75-79, 135, 142-146</td>
</tr>
<tr>
<td>Irregular boundaries (geometries), 190-192, 200, 205-211, 215, 220</td>
</tr>
<tr>
<td>Isoparametric method, 211, 215</td>
</tr>
<tr>
<td>Isothermal, 5, 58, 196, 221</td>
</tr>
<tr>
<td>Iterative matrix methods, 247-249</td>
</tr>
<tr>
<td>ITPACK/REGION, 220</td>
</tr>
<tr>
<td>Jacobian, 32, 34, 40-42, 67, 80, 82, 87, 147, 152-154, 236</td>
</tr>
<tr>
<td>Jacobi method, 248</td>
</tr>
<tr>
<td>Kinetics of benzene hydrogenation 5-8, 10, 11, 147-154</td>
</tr>
<tr>
<td>Knot sequence, 244-246</td>
</tr>
<tr>
<td>Lacquer application, 89, 90</td>
</tr>
<tr>
<td>Laplace's equation, 128, 178-184, 220</td>
</tr>
<tr>
<td>Lewis number, 149, 150, 152</td>
</tr>
<tr>
<td>L'Hopital's rule, 82</td>
</tr>
<tr>
<td>Linear function space, 100</td>
</tr>
<tr>
<td>Linear interpolation, 207-210</td>
</tr>
<tr>
<td>Linear second order equations, 67-79, 83-87</td>
</tr>
<tr>
<td>Linearly independent, 66, 100, 180</td>
</tr>
<tr>
<td>Line integral, 185, 186, 188, 191</td>
</tr>
<tr>
<td>Local support, 103, 104, 155, 158, 159, 244-246</td>
</tr>
<tr>
<td>Local truncation error, 4, 5, 9, 16, 23, 25, 26, 36, 37, 39, 41, 70, 86, 229</td>
</tr>
<tr>
<td>L-operator, 54, 55, 68, 69, 79, 83, 113, 127</td>
</tr>
<tr>
<td>Lower-upper decomposition (LU), 237-242</td>
</tr>
<tr>
<td>Low order time approximations, 130-146</td>
</tr>
<tr>
<td>LSODE, 39, 41-44, 62, 152, 162</td>
</tr>
<tr>
<td>LSODI, 162</td>
</tr>
<tr>
<td>Machine epsilon (EPS), 232, 233</td>
</tr>
<tr>
<td>Machine number, 9, 229-234</td>
</tr>
<tr>
<td>Mass transfer, 1, 58, 59, 136, 137, 147-158</td>
</tr>
<tr>
<td>Material (mass) balance, 5-8, 34, 35, 58-62, 115-118, 136, 137, 144-166, 187-190, 196-200, 219-221</td>
</tr>
<tr>
<td>Mathematical software, 37-44, 87-90, 111, 119-123, 162-166, 214-221</td>
</tr>
<tr>
<td>Mesh points, 67, 68, 89</td>
</tr>
<tr>
<td>Mesh refinement, 120, 122</td>
</tr>
<tr>
<td>Mesh size, 68, 85, 87</td>
</tr>
<tr>
<td>Method of false (fictitious) boundaries, 71, 72, 135, 136, 151, 179, 180</td>
</tr>
<tr>
<td>Method of lines (MOL), 128-156, 159, 160, 162, 211, 212, 215</td>
</tr>
<tr>
<td>Method parameter (MF), 40, 41, 43, 44</td>
</tr>
<tr>
<td>MOL1D, 163</td>
</tr>
<tr>
<td>M3RK, 39, 147</td>
</tr>
<tr>
<td>MSHOOT, 88</td>
</tr>
<tr>
<td>Multiple shooting, 56, 63-65, 88-90</td>
</tr>
<tr>
<td>Multistep methods, 24-28, 39, 40, 146, 152</td>
</tr>
<tr>
<td>NAG library, 38, 39, 88, 162, 163, 215, 219, 220</td>
</tr>
<tr>
<td>Neumann boundary conditions, 129, 135, 178-182, 191, 192</td>
</tr>
<tr>
<td>Newton backward formula, 24</td>
</tr>
<tr>
<td>Newton iteration, 19, 21, 27, 55, 62, 65, 67, 79, 83, 87, 89, 110, 118, 120, 122, 141, 184, 185, 235, 236</td>
</tr>
<tr>
<td>Newton's law of heating and cooling, 15</td>
</tr>
<tr>
<td>Non-autonomous, 33, 34</td>
</tr>
<tr>
<td>Nonlinear, 53, 54, 67, 79-83, 87, 88, 109, 110, 112, 118, 120, 140-142, 147, 155, 156, 184, 185, 200, 220, 235, 236</td>
</tr>
<tr>
<td>Nonstiff, 32, 34, 39, 41-44, 62, 147, 152, 154, 162</td>
</tr>
<tr>
<td>Nonuniform mesh (grid), 84, 85, 87, 185-191, 205-211</td>
</tr>
<tr>
<td>Nuclear fuel rods, 78, 79</td>
</tr>
<tr>
<td>Numerical integration, 24</td>
</tr>
<tr>
<td>ODE, 40-44, 147</td>
</tr>
<tr>
<td>Order of accuracy, 4, 5, 11, 13, 14, 20, 23, 104-106, 109, 113-115, 181, 204</td>
</tr>
<tr>
<td>Order of approximating space (k), 100, 101, 103, 105, 112, 115, 118-120, 155, 158, 159, 165, 200, 245, 246</td>
</tr>
<tr>
<td>Ordinary differential equations, 1-125, 128, 129, 142</td>
</tr>
<tr>
<td>Orthogonal, 67</td>
</tr>
<tr>
<td>Orthonormalization, 67, 88</td>
</tr>
<tr>
<td>Parabolic partial differential equations in one space dimension, 127-166</td>
</tr>
<tr>
<td>Parabolic partial differential equations in two space dimensions, 177, 211-218</td>
</tr>
<tr>
<td>Partial differential equations (PDE), 127-227</td>
</tr>
</tbody>
</table>
Subject Index

PDECOL, 162–166
PDEPACK, 163
PDETW0, 215–217
Peclet number (Pe), 156–158
Piecewise continuous, 75
Piecewise linear function, 100–106, 200–204
Piecewise polynomial approximations, 97–126, 154–166, 192–211
Piecewise polynomial functions, 97–103
Pivoting, 240
PLTMG, 220
Polymer reaction, 159–162
POST, 163
Prater number, 149, 150, 152, 219–221
Predictor-corrector methods, 26, 27, 142
Pth-order accurate, 4, 5, 11, 13, 18, 36

Quadratic B-splines, 246
Quadratic piecewise polynomial interpolation, 210, 211, 215
Quadrature, 110
Quasilinearization, 67, 88
Radial dispersion, 187–190
Relative round-off error, 232
Reverse shooting, 90
Reynold’s number (Re), 145, 146
Richardson’s extrapolation, 85, 86
RKF45, 39
Robertson’s problem, 41–44
Robin boundary conditions, 129, 135, 179, 180–184
Rounding errors, 231–234
Rounding off errors, 56, 229–234, 240, 247
Runga-Kutta-Fehlberg method, 18, 32–35, 39, 88
Runga-Kutta-Gill method, 14–20
Runga-Kutta method, 11–20, 28, 29, 39, 88, 146, 147, 215

Schmidt number, 145, 146
SECDER, 39
Second order accurate, 13, 15, 16, 20, 69–71, 75, 105, 130, 132, 133, 139, 144, 147, 155, 184, 187–190, 204, 210, 214, 216
Second order difference approximations, 69–71, 75, 79, 178, 187
Semi-implicit methods, 28, 29, 37
Sherwood number (Sh), 166
Shooting methods, 53–66, 88
Shooting root(s), 55–58, 60, 62, 63, 65
SHOOT1, 88
SHOOT2, 88
Single precision, 232
Slope functions (s), 102, 103, 106–108, 114, 117, 118, 195, 198, 199
Sparse matrix, 237, 247
Spatial discretization, 128–152, 162
Stability, 7–11, 14, 15, 18–21, 23, 24, 27, 28, 70, 131–133, 139–141, 212, 213
Step size strategies, 36, 37
Stiff, 36, 37, 39, 41–44, 62, 147, 152, 154, 162
Stiffness, 29–32
Stiffness ratio (SR), 30–32, 34, 147
STIFF3, 39–44
STINT, 39
STRIDE, 39
Successive over relaxation method, 248, 249
Superposition, 53, 65–67, 88
SUPORQ, 88
SUPORT, 88
System of differential equations, 29, 32–35
Taylor series expansion (theorem), 4, 12, 13, 19, 24, 25, 68, 229
Tensor products, 194, 195, 200, 201, 204
Thermocouple, 14–16
Theta method, 133–136, 139–141
Thomas algorithm, 241
Time integration of PDE’s, 128–153, 162
Trapezoidal rule, 20, 21, 23, 25–28, 32, 83, 85, 86, 89, 133, 155
Triangulation (triangular grid), 205–211, 215, 220
Tricomi’s equation, 128
Tridiagonal matrix, 70, 72, 80, 83, 105–109, 132, 142, 213, 237, 241, 242
Truncated power function, 244
Tubular reactor, 9, 34
Unconditionally stable, 10, 20, 21, 28, 132, 133, 214
Value functions (v), 102, 103, 113, 114, 117, 118, 194, 198, 199
Vector notation (derivation), 2, 3
Velocity profile, 138–141, 215–219
Wave equation, 128
Weak form, 99, 104, 107, 110, 154, 157, 158, 194, 202–205
Wetted-wall column, 163–166
Zero degree interpolation, 190

258