Mathematics in Chemical Engineering

Bruce A. Finlayson, Department of Chemical Engineering, University of Washington, Seattle, Washington, United States (Chap. 1, 2, 3, 4, 5, 6, 7, 8, 9, 11 and 12)

Lorenz T. Biegler, Carnegie Mellon University, Pittsburgh, Pennsylvania, United States (Chap. 10)

Ignacio E. Grossmann, Carnegie Mellon University, Pittsburgh, Pennsylvania, United States (Chap. 10)

1. Solution of Equations 5
1.1. Matrix Properties 5
1.2. Linear Algebraic Equations 7
1.3. Nonlinear Algebraic Equations . . . 10
1.4. Linear Difference Equations 12
1.5. Eigenvalues 12
2. Approximation and Integration . . . 13
2.1. Introduction 13
2.2. Global Polynomial Approximation . 13
2.3. Piecewise Approximation 15
2.4. Quadrature 18
2.5. Least Squares 20
2.6. Fourier Transforms of Discrete Data 21
2.7. Two-Dimensional Interpolation and Quadrature 22
3. Complex Variables 22
3.1. Introduction to the Complex Plane . 22
3.2. Elementary Functions 23
3.3. Analytic Functions of a Complex Variable 25
3.4. Integration in the Complex Plane . . 26
3.5. Other Results 28
4. Integral Transforms 29
4.1. Fourier Transforms 29
4.2. Laplace Transforms 33
4.3. Solution of Partial Differential Equations by Using Transforms . . . 37
5. Vector Analysis 40
6. Ordinary Differential Equations as Initial Value Problems 49
6.1. Solution by Quadrature 50
6.2. Explicit Methods 50
6.3. Implicit Methods 54
6.4. Stiffness 55
6.5. Differential – Algebraic Systems . . . 56
6.6. Computer Software 57
6.7. Stability, Bifurcations, Limit Cycles 58
6.8. Sensitivity Analysis 60
6.9. Molecular Dynamics 61
7. Ordinary Differential Equations as Boundary Value Problems 61
7.1. Solution by Quadrature 62
7.2. Initial Value Methods 62
7.3. Finite Difference Method 63
7.4. Orthogonal Collocation 65
7.5. Orthogonal Collocation on Finite Elements 69
7.6. Galerkin Finite Element Method . . 71
7.7. Cubic B-Splines 73
7.8. Adaptive Mesh Strategies 73
7.9. Comparison 73
7.10. Singular Problems and Infinite Domains 74
8. Partial Differential Equations 75
8.1. Classification of Equations 75
8.2. Hyperbolic Equations 77
8.3. Parabolic Equations in One Dimension 79
8.4. Elliptic Equations 84
8.5. Parabolic Equations in Two or Three Dimensions 86
8.6. Special Methods for Fluid Mechanics 86
8.7. Computer Software 88
9. Integral Equations 89
9.1. Classification 89
9.2. Numerical Methods for Volterra Equations of the Second Kind 91
9.3. Numerical Methods for Fredholm, Urysohn, and Hammerstein Equations of the Second Kind 91
9.4. Numerical Methods for Eigenvalue Problems 92
9.5. Green's Functions 92
9.6. Boundary Integral Equations and Boundary Element Method 94
10. Optimization 95
10.1. Introduction 95
10.2. Gradient-Based Nonlinear Programming 96
10.3. Optimization Methods without Derivatives 105
10.4. Global Optimization 106
10.5. Mixed Integer Programming 110
10.6. Dynamic Optimization 121

Ullmann's Modeling and Simulation

ISBN: 978-3-527-31605-2

10.7. Development of Optimization Models 124
11. Probability and Statistics 125
11.1. Concepts 125
11.2. Sampling and Statistical Decisions . 129
11.3. Error Analysis in Experiments 132
11.4. Factorial Design of Experiments and Analysis of Variance 133
12. Multivariable Calculus Applied to Thermodynamics 135
12.1. State Functions 135
12.2. Applications to Thermodynamics . . 136
12.3. Partial Derivatives of All Thermodynamic Functions 137
13. References 138

Symbols

Variables

a	1, 2, or 3 for planar, cylindrical, spherical geometry
a_i	acceleration of i-th particle in molecular dynamics
A	cross-sectional area of reactor; Helmholtz free energy in thermodynamics
Bi	Biot number
Bim	Biot number for mass
c	concentration
C_p	heat capacity at constant pressure
C_s	heat capacity of solid
C_v	heat capacity at constant volume
Co	Courant number
D	diffusion coefficient
Da	Damköhler number
D_e	effective diffusivity in porous catalyst
E	efficiency of tray in distillation column
F	molar flow rate into a chemical reactor
G	Gibbs free energy in thermodynamics
h_p	heat transfer coefficient
H	enthalpy in thermodynamics
J	mass flux
k	thermal conductivity; reaction rate constant
k_e	effective thermal conductivity of porous catalyst
k_g	mass transfer coefficient
K	chemical equilibrium constant
L	thickness of slab; liquid flow rate in distillation column; length of pipe for flow in pipe
m_i	mass of i-th particle in molecular dynamics
M	holdup on tray of distillation column
n	power-law exponent in viscosity formula for polymers
p	pressure
Pe	Peclet number
q	heat flux
Q	volumetric flow rate; rate of heat generation for heat transfer problems
r	radial position in cylinder
$\boldsymbol{r}_i$	position of i-th particle in molecular dynamics
R	radius of reactor or catalyst pellet
Re	Reynolds number
S	entropy in thermodynamics
Sh	Sherwood number
t	time
T	temperature
$\boldsymbol{u}$	velocity
U	internal energyy in thermodynamics
$\boldsymbol{v}_i$	velocity of i-th particle in molecular dynamics
V	volume of chemical reactor; vapor flow rate in distillation column; potential energy in molecular dynamics; specific volume in thermodynamics
x	position
z	position from inlet of reactor

Greek symbols

a	thermal diffusivity
δ	Kronecker delta
Δ	sampling rate
ε	porosity of catalyst pellet
η	viscosity in fluid flow; effectiveness factor for reaction in a catalyst pellet
η_0	zero-shear rate viscosity
ϕ	Thiele modulus
φ	void fraction of packed bed
λ	time constant in polymer flow
ρ	density
ρ_s	density of solid
τ	shear stress
μ	viscosity

Special symbols

|	subject to
:	mapping. For example, $h: R^n \rightarrow R^m$, states that functions h map real numbers into m real numbers. There are m functions h written in terms of n variables
$\in$	member of
$\rightarrow$	maps into

1. Solution of Equations

Mathematical models of chemical engineering systems can take many forms: they can be sets of algebraic equations, differential equations, and/or integral equations. Mass and energy balances of chemical processes typically lead to large sets of *algebraic* equations:

$$a_{11}x_1+a_{12}x_2 = b_1$$
$$a_{21}x_1+a_{22}x_2 = b_2$$

Mass balances of stirred tank reactors may lead to ordinary *differential* equations:

$$\frac{\mathrm{d}\, y}{\mathrm{d}\, t} = f\ [y\,(t)]$$

Radiative heat transfer may lead to *integral equations:*

$$y\,(x) = g\,(x) + \lambda \int_0^1 K\,(x, s)\, f\ (s)\, \mathrm{d}\, s$$

Even when the model is a differential equation or integral equation, the most basic step in the algorithm is the solution of sets of algebraic equations. The solution of sets of algebraic equations is the focus of Chapter 1.

A single linear equation is easy to solve for either x or y:

$$y = ax+b$$

If the equation is nonlinear,

$$f\,(x) = 0$$

it may be more difficult to find the x satisfying this equation. These problems are compounded when there are more unknowns, leading to simultaneous equations. If the unknowns appear in a linear fashion, then an important consideration is the structure of the matrix representing the equations; special methods are presented here for special structures. They are useful because they increase the speed of solution. If the unknowns appear in a nonlinear fashion, the problem is much more difficult. Iterative techniques must be used (i.e., make a guess of the solution and try to improve the guess). An important question then is whether such an iterative scheme converges. Other important types of equations are linear difference equations and eigenvalue problems, which are also discussed.

1.1. Matrix Properties

A matrix is a set of real or complex numbers arranged in a rectangular array.

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix}$$

The numbers a_{ij} are the elements of the matrix $\boldsymbol{A}$, or $(\mathrm{A})_{ij} = a_{ij}$. The transpose of $\boldsymbol{A}$ is $(\boldsymbol{A}^T) = a_{ji}$.

The determinant of a square matrix $\boldsymbol{A}$ is

$$\boldsymbol{A} = \begin{vmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{vmatrix}$$

If the i-th row and j-th column are deleted, a new determinant is formed having $n-1$ rows and columns. This determinant is called the minor of a_{ij} denoted as M_{ij}. The cofactor A'_{ij} of the element a_{ij} is the signed minor of a_{ij} determined by

$$A'_{ij} = (-1)^{i+j} M_{ij}$$

The value of $|\boldsymbol{A}|$ is given by

$$|\boldsymbol{A}| = \sum_{j=1}^{n} a_{ij} A'_{ij} \text{ or } \sum_{i=1}^{n} a_{ij} A'_{ij}$$

where the elements a_{ij} must be taken from a single row or column of $\boldsymbol{A}$.

If all determinants formed by striking out whole rows or whole columns of order greater than r are zero, but there is at least one determinant of order r which is not zero, the matrix has rank r.

The value of a determinant is not changed if the rows and columns are interchanged. If the elements of one row (or one column) of a determinant are all zero, the value of the determinant is zero. If the elements of one row or column are multiplied by the same constant, the determinant is the previous value times that constant. If two adjacent rows (or columns) are interchanged, the value of the new determinant is the negative of the value of the original determinant. If two rows (or columns) are identical, the determinant is zero. The value of a determinant is not changed if one row (or column) is multiplied by a constant and added to another row (or column).

A matrix is *symmetric* if

$$a_{ij}=a_{ji}$$

and it is *positive definite* if

$$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}=\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ij}x_ix_j\geq 0$$

for all $\boldsymbol{x}$ and the equality holds only if $\boldsymbol{x}=0$.

If the elements of $\boldsymbol{A}$ are complex numbers, $\boldsymbol{A}^*$ denotes the complex conjugate in which $(A^*)_{ij}=a^*_{ij}$. If $\boldsymbol{A}=\boldsymbol{A}^*$ the matrix is Hermitian.

The *inverse of a matrix* can also be used to solve sets of linear equations. The inverse is a matrix such that when $\boldsymbol{A}$ is multiplied by its inverse the result is the identity matrix, a matrix with 1.0 along the main diagonal and zero elsewhere.

$$\boldsymbol{A}\boldsymbol{A}^{-1}=\boldsymbol{I}$$

If $\boldsymbol{A}^T=\boldsymbol{A}^{-1}$ the matrix is orthogonal.

Matrices are added and subtracted element by element.

$\boldsymbol{A}+\boldsymbol{B}$ is $a_{ij}+b_{ij}$

Two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are equal if $a_{ij}=b_{ij}$. Special relations are

$$(\boldsymbol{A}\boldsymbol{B})^{-1}=\boldsymbol{B}^{-1}\boldsymbol{A}^{-1},\quad (\boldsymbol{A}\boldsymbol{B})^T=\boldsymbol{B}^T\boldsymbol{A}^T$$
$$(\boldsymbol{A}^{-1})^T=(\boldsymbol{A}^T)^{-1},\quad (\boldsymbol{A}\boldsymbol{B}\boldsymbol{C})^{-1}=\boldsymbol{C}^{-1}\boldsymbol{B}^{-1}\boldsymbol{A}^{-1}$$

A diagonal matrix is zero except for elements along the diagonal.

$$a_{ij}=\begin{cases} a_{ii}, & i=j \\ 0, & i\neq j \end{cases}$$

A tridiagonal matrix is zero except for elements along the diagonal and one element to the right and left of the diagonal.

$$a_{ij}=\begin{cases} 0 \text{ if } j<i-1 \\ a_{ij} \text{ otherwise} \\ 0 \text{ if } j>i+1 \end{cases}$$

Block diagonal and pentadiagonal matrices also arise, especially when solving partial differential equations in two- and three-dimensions.

QR Factorization of a Matrix. If A is an $m \times n$ matrix with $m \geq n$, there exists an $m \times m$ unitary matrix $\boldsymbol{Q}=[\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots \boldsymbol{q}_m]$ and an $m \times n$ right-triangular matrix $\boldsymbol{R}$ such that $\boldsymbol{A}=\boldsymbol{Q}\boldsymbol{R}$. The QR factorization is frequently used in the actual computations when the other transformations are unstable.

Singular Value Decomposition. If $\boldsymbol{A}$ is an $m \times n$ matrix with $m \geq n$ and rank $k \leq n$, consider the two following matrices.

$\boldsymbol{A}\boldsymbol{A}^*$ and $\boldsymbol{A}^*\boldsymbol{A}$

An $m \times m$ unitary matrix $\boldsymbol{U}$ is formed from the eigenvectors $\boldsymbol{u}_i$ of the first matrix.

$$\boldsymbol{U}=[\boldsymbol{u}_1,\boldsymbol{u}_2,\ldots\boldsymbol{u}_m]$$

An $n \times n$ unitary matrix $\boldsymbol{V}$ is formed from the eigenvectors $\boldsymbol{v}_i$ of the second matrix.

$$\boldsymbol{V}=[\boldsymbol{v}_1,\boldsymbol{v}_2,\ldots,\boldsymbol{v}_n]$$

Then the matrix $\boldsymbol{A}$ can be decomposed into

$$\boldsymbol{A}=\boldsymbol{U}\Sigma\boldsymbol{V}^*$$

where Σ is a $k \times k$ diagonal matrix with diagonal elements $d_{ii}=\sigma_i>0$ for $1 \leq i \leq k$. The eigenvalues of $\Sigma^*\Sigma$ are σ_i^2. The vectors $\boldsymbol{u}_i$ for $k+1 \leq i \leq m$ and $\boldsymbol{v}_i$ for $k+1 \leq i \leq n$ are eigenvectors associated with the eigenvalue zero; the eigenvalues for $1 \leq i \leq k$ are σ_i^2. The values of σ_i are called the singular values of the matrix $\boldsymbol{A}$. If $\boldsymbol{A}$ is real, then $\boldsymbol{U}$ and $\boldsymbol{V}$ are real, and hence orthogonal matrices. The value of the singular value decomposition comes when a process is represented by a linear transformation and the elements of $\boldsymbol{A}$, a_{ij}, are the contribution to an output i for a particular variable as input variable j. The input may be the size of a disturbance, and the output is the gain [1]. If the rank is less than n, not all the variables are independent and they cannot all be controlled. Furthermore, if the singular values are widely separated, the process is sensitive to small changes in the elements of the matrix and the process will be difficult to control.

1.2. Linear Algebraic Equations

Consider the $n \times n$ linear system

$$\begin{aligned} a_{11}\,x_1 + a_{12}\,x_2 + \ldots + a_{1n}\,x_n &= f_1 \\ a_{21}\,x_1 + a_{22}\,x_2 + \ldots + a_{2n}\,x_n &= f_2 \\ &\ldots \\ a_{n1}\,x_1 + a_{n2}\,x_2 + \ldots + a_{nn}\,x_n &= f_n \end{aligned}$$

In this equation $a_{11}, \ldots, a_{nn}$ are known parameters, $f_1, \ldots, f_n$ are known, and the unknowns are $x_1, \ldots, x_n$. The values of all unknowns that satisfy every equation must be found. This set of equations can be represented as follows:

$$\sum_{j=1}^{n} a_{ij}\,x_j = f_j \text{ or } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{f}$$

The most efficient method for solving a set of linear algebraic equations is to perform a lower – upper (LU) decomposition of the corresponding matrix $\boldsymbol{A}$. This decomposition is essentially a Gaussian elimination, arranged for maximum efficiency [2, 3].

The LU decomposition writes the matrix as

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

The $\boldsymbol{U}$ is upper triangular; it has zero elements below the main diagonal and possibly nonzero values along the main diagonal and above it (see Fig. 1). The $\boldsymbol{L}$ is lower triangular. It has the value 1 in each element of the main diagonal, nonzero values below the diagonal, and zero values above the diagonal (see Fig. 1). The original problem can be solved in two steps:

$$\boldsymbol{L}\boldsymbol{y} = \boldsymbol{f}, \boldsymbol{U}\boldsymbol{x} = \boldsymbol{y} \text{ solves } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{L}\boldsymbol{U}\boldsymbol{x} = \boldsymbol{f}$$

Each of these steps is straightforward because the matrices are upper triangular or lower triangular.

When f is changed, the last steps can be done without recomputing the LU decomposition. Thus, multiple right-hand sides can be computed efficiently. The number of multiplications and divisions necessary to solve for m right-hand sides is:

$$\text{Operation count } = \frac{1}{3}n^3 - \frac{1}{3}n + m\,n^2$$

The *determinant* is given by the product of the diagonal elements of $\boldsymbol{U}$. This should be calculated as the LU decomposition is performed.

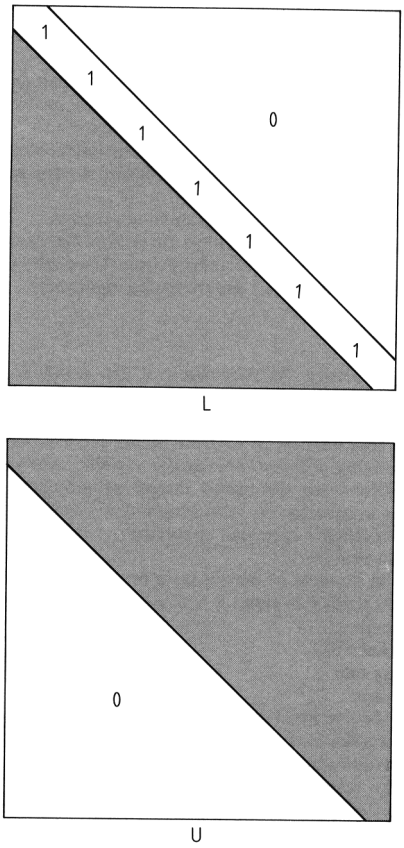

Figure 1. Structure of $\boldsymbol{L}$ and $\boldsymbol{U}$ matrices

If the value of the determinant is a very large or very small number, it can be divided or multiplied by 10 to retain accuracy in the computer; the scale factor is then accumulated separately. The condition number κ can be defined in terms of the singular value decomposition as the ratio of the largest d_{ii} to the smallest d_{ii} (see above). It can also be expressed in terms of the norm of the matrix:

$$\kappa\,(A) = \|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|$$

where the norm is defined as

$$\|\boldsymbol{A}\| \equiv \sup_{x \neq 0} \frac{\|\boldsymbol{A}\boldsymbol{x}\|}{\|\boldsymbol{x}\|} = \max_k \sum_{j=1}^{n} |a_{jk}|$$

If this number is infinite, the set of equations is singular. If the number is too large, the matrix is said to be ill-conditioned. Calculation of

the condition number can be lengthy so another criterion is also useful. Compute the ratio of the largest to the smallest pivot and make judgments on the ill-conditioning based on that.

When a matrix is ill-conditioned the LU decomposition must be performed by using pivoting (or the singular value decomposition described above). With pivoting, the order of the elimination is rearranged. At each stage, one looks for the largest element (in magnitude); the next stages if the elimination are on the row and column containing that largest element. The largest element can be obtained from only the diagonal entries (partial pivoting) or from all the remaining entries. If the matrix is nonsingular, Gaussian elimination (or LU decomposition) could fail if a zero value were to occur along the diagonal and were to be a pivot. With full pivoting, however, the Gaussian elimination (or LU decomposition) cannot fail because the matrix is nonsingular.

The *Cholesky decomposition* can be used for real, symmetric, positive definite matrices. This algorithm saves on storage (divide by about 2) and reduces the number of multiplications (divide by 2), but adds *n* square roots.

The linear equations are solved by

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{f}$$

Generally, the inverse is not used in this way because it requires three times more operations than solving with an LU decomposition. However, if an inverse is desired, it is calculated most efficiently by using the LU decomposition and then solving

$$\boldsymbol{A}\boldsymbol{x}^{(i)} = \boldsymbol{b}^{(i)}$$

$$b_j^{(i)} = \begin{cases} 0 & j \neq i \\ 1 & j = i \end{cases}$$

Then set

$$\boldsymbol{A}^{-1} = \left(\boldsymbol{x}^{(1)} | \boldsymbol{x}^{(2)} | \boldsymbol{x}^{(3)} | \cdots | \boldsymbol{x}^{(n)}\right)$$

Solutions of Special Matrices. Special matrices can be handled even more efficiently. A *tridiagonal matrix* is one with nonzero entries along the main diagonal, and one diagonal above and below the main one (see Fig. 2). The corresponding set of equations can then be written as

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i$$

The LU decomposition algorithm for solving this set is

$b'_1 = b_1$
for $k = 2, n$ do
$a'_k = \frac{a_k}{b'_{k-1}}$, $b'_k = b_k - \frac{a_k}{b'_{k-1}} c_{k-1}$
enddo
$d'_1 = d_1$
for $k = 2, n$ do
$d'_k = d_k - a'_k d'_{k-1}$
enddo
$x_n = d'_n / b'_n$
for $k = n-1, 1$do
$x_k = \frac{d'_k - c_k x_{k+1}}{b'_k}$
enddo

The number of multiplications and divisions for a problem with *n* unknowns and *m* right-hand sides is

$$\text{Operation count} = 2\,(n-1) + m\,(3\,n-2)$$

If

$$|b_i| > |a_i| + |c_i|$$

no pivoting is necessary. For solving two-point boundary value problems and partial differential equations this is often the case.

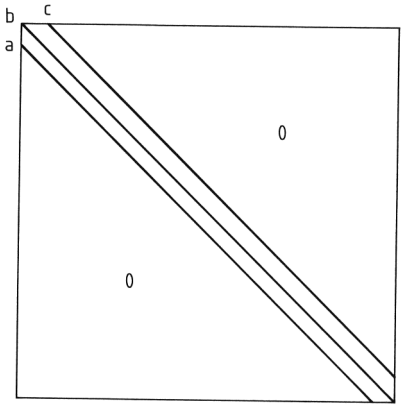

Figure 2. Structure of tridiagonal matrices

Sparse matrices are ones in which the majority of elements are zero. If the zero entries occur in special patterns, efficient techniques can be used to exploit the structure, as was done above for tridiagonal matrices, block tridiagonal

matrices, arrow matrices, etc. These structures typically arise from numerical analysis applied to solve differential equations. Other problems, such as modeling chemical processes, lead to sparse matrices but without such a neatly defined structure—just a lot of zeros in the matrix. For matrices such as these, special techniques must be employed: efficient codes are available [4]. These codes usually employ a symbolic factorization, which must be repeated only once for each structure of the matrix. Then an LU factorization is performed, followed by a solution step using the triangular matrices. The symbolic factorization step has significant overhead, but this is rendered small and insignificant if matrices with exactly the same structure are to be used over and over [5].

The efficiency of a technique for solving sets of linear equations obviously depends greatly on the arrangement of the equations and unknowns because an efficient arrangement can reduce the bandwidth, for example. Techniques for renumbering the equations and unknowns arising from elliptic partial differential equations are available for finite difference methods [6] and for finite element methods [7].

Solutions with Iterative Methods. Sets of linear equations can also be solved by using iterative methods; these methods have a rich historical background. Some of them are discussed in Chapter 8 and include Jacobi, Gauss – Seidel, and overrelaxation methods. As the speed of computers increases, direct methods become preferable for the general case, but for large three-dimensional problems iterative methods are often used.

The *conjugate gradient method* is an iterative method that can solve a set of *n* linear equations in *n* iterations. The method primarily requires multiplying a matrix by a vector, which can be done very efficiently on parallel computers: for sparse matrices this is a viable method. The original method was devised by HESTENES and STIEFEL [8]; however, more recent implementations use a *preconditioned* conjugate gradient method because it converges faster, provided a good "preconditioner" can be found. The system of *n* linear equations

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{f}$$

where $\boldsymbol{A}$ is symmetric and positive definite, is to be solved. A preconditioning matrix $\boldsymbol{M}$ is defined in such a way that the problem

$$\boldsymbol{M}\boldsymbol{t} = \boldsymbol{r}$$

is easy to solve exactly ($\boldsymbol{M}$ might be diagonal, for example). Then the preconditioned conjugate gradient method is

Guess $\boldsymbol{x}_0$

Calculate $\boldsymbol{r}_0 = \boldsymbol{f} - \boldsymbol{A}\boldsymbol{x}_0$

Solve $\boldsymbol{M}\boldsymbol{t}_0 = \boldsymbol{r}_0$, and set $\boldsymbol{p}_0 = \boldsymbol{t}_0$

for $k = 1, n$ (or until convergence)

$a_k = \frac{\boldsymbol{r}_k^{\mathrm{T}}\boldsymbol{t}_k}{\boldsymbol{p}_k^{\mathrm{T}}\boldsymbol{A}\boldsymbol{p}_k}$

$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + a_k\boldsymbol{p}_k$

$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \boldsymbol{a}_k\boldsymbol{A}\boldsymbol{p}_k$

Solve $\boldsymbol{M}t_{k+1} = \boldsymbol{r}_{k+1}$

$b_k = \frac{\boldsymbol{r}_{k+1}^{\mathrm{T}}\boldsymbol{t}_{k+1}}{\boldsymbol{r}_k^{\mathrm{T}}\boldsymbol{t}_k}$

$\boldsymbol{p}_{k+1} = \boldsymbol{t}_{k+1} + b_k\boldsymbol{p}_k$

test for convergence

enddo

Note that the entire algorithm involves only matrix multiplications. The generalized minimal residual method (GMRES) is an iterative method that can be used for nonsymmetric systems and is based on a modified Gram – Schmidt orthonormalization. Additional information, including software for a variety of methods, is available [9 – 13].

In dimensional analysis if the dimensions of each physical variable P_j (there are *n* of them) are expressed in terms of fundamental measurement units m_j (such as time, length, mass; there are *m* of them):

$$[P_j] = m_1^{\alpha_{1j}} m_2^{\alpha_{2j}} \cdots m_m^{\alpha_{mj}}$$

then a matrix can be formed from the α_{ij}. If the rank of this matrix is *r*, $n - r$ independent dimensionless groups govern that phenomenon. In chemical reaction engineering the chemical reaction stoichiometry can be written as

$$\sum_{i=1}^{n} \alpha_{ij}\, C_i = 0,\ j = 1, 2, \ldots, m$$

where there are *n* species and *m* reactions. Then if a matrix is formed from the coefficients α_{ij}, which is an $n \times m$ matrix, and the rank of the matrix is *r*, there are *r* independent chemical reactions. The other $n - r$ reactions can be deduced from those *r* reactions.

1.3. Nonlinear Algebraic Equations

Consider a single nonlinear equation in one unknown,

$$f(x) = 0$$

In Microsoft Excel, roots are found by using Goal Seek or Solver. Assign one cell to be x, put the equation for $f(x)$ in another cell, and let Goal Seek or Solver find the value of x making the equation cell zero. In MATLAB, the process is similar except that a function (m-file) is defined and the command fzero('f',x0) provides the solution x, starting from the initial guess x_0.

Iterative methods applied to single equations include the *successive substitution* method

$$x^{k+1} = x^k + \beta f\left(x^k\right) \equiv g\left(x^k\right)$$

and the Newton – Raphson method.

$$x^{k+1} = x^k - \frac{f\left(x^k\right)}{\mathrm{d}f/\mathrm{d}x\left(x^k\right)}$$

The former method converges if the derivative of $g(x)$ is bounded [3]. The latter method

$$\left|\frac{\mathrm{d}g}{\mathrm{d}x}(x)\right| \leq \mu \text{ for } |x-\alpha| < h$$

is based on a Taylor series of the equation about the k-th iterate:

$$f\left(x^{k+1}\right) = f\left(x^k\right) + \frac{\mathrm{d}f}{\mathrm{d}x}|_{x^k}\left(x^{k+1}-x^k\right) + \frac{\mathrm{d}^2 f}{\mathrm{d}x^2}|_{x^k}\frac{1}{2}\left(x^{k+1}-x^k\right)^2 + \cdots$$

The second and higher-order terms are neglected and $f\left(x^{k+1}\right) = 0$ to obtain the method.

$$\left|\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{x^0} > 0,\ |x^1 - x^0| = \left|\frac{f\left(x^0\right)}{\mathrm{d}f/\mathrm{d}x\left(x^0\right)}\right|_{x^0} \leq b, \text{ and } \left|\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}\right| \leq c$$

Convergence of the Newton–Raphson method depends on the properties of the first and second derivative of $f(x)$ [3, 14]. In practice the method may not converge unless the initial guess is good, or it may converge for some parameters and not others. Unfortunately, when the method is nonconvergent the results look as though a mistake occurred in the computer programming; distinguishing between these situations is difficult, so careful programming and testing are required. If the method converges the difference between successive iterates is something like 0.1, 0.01, 0.0001, 10^{-8}. The error (when it is known) goes the same way; the method is said to be quadratically convergent when it converges. If the derivative is difficult to calculate a numerical approximation may be used.

$$\frac{\mathrm{d}f}{\mathrm{d}x}|_{x^k} = \frac{f\left(x^k+\varepsilon\right) - f\left(x^k\right)}{\varepsilon}$$

In the *secant method* the same formula is used as for the Newton – Raphson method, except that the derivative is approximated by using the values from the last two iterates:

$$\frac{\mathrm{d}f}{\mathrm{d}x}|_{x^k} = \frac{f\left(x^k\right) - f\left(x^{k-1}\right)}{x^k - x^{k-1}}$$

This is equivalent to drawing a straight line through the last two iterate values on a plot of $f(x)$ versus x. The Newton – Raphson method is equivalent to drawing a straight line tangent to the curve at the last x. In the *method of false position* (or regula falsi), the secant method is used to obtain x^{k+1}, but the previous value is taken as either x^{k-1} or x^k. The choice is made so that the function evaluated for that choice has the opposite sign to $f\left(x^{k+1}\right)$. This method is slower than the secant method, but it is more robust and keeps the root between two points at all times. In all these methods, appropriate strategies are required for bounds on the function or when $\mathrm{d}f/\mathrm{d}x = 0$. *Brent's method* combines bracketing, bisection, and an inverse quadratic interpolation to provide a method that is fast and guaranteed to converge, if the root can be bracketed initially [15, p. 251].

In the *method of bisection*, if a root lies between x_1 and x_2 because $f(x_1) < 0$ and $f(x_2) > 0$, then the function is evaluated at the center, $x_c = 0.5(x_1 + x_2)$. If $f(x_c) > 0$, the root lies between x_1 and x_c. If $f(x_c) < 0$, the root lies between x_c and x_2. The process is then repeated. If $f(x_c) = 0$, the root is x_c. If $f(x_1) > 0$ and $f(x_2) > 0$, more than one root may exist between x_1 and x_2 (or no roots).

For systems of equations the Newton – Raphson method is widely used, especially for equations arising from the solution of differential equations.

$$f_i\left(\{x_j\}\right) = 0,\ 1 \leq i, j \leq n,$$
$$\text{where } \{x_j\} = (x_1, x_2, \ldots, x_n) = \boldsymbol{x}$$

Then, an expansion in several variables occurs:

$$\boldsymbol{f}_i\left(\boldsymbol{x}^{k+1}\right)=f_i\left(\boldsymbol{x}^k\right)+\sum_{j=1}^{n}\frac{\partial f_i}{\partial x_j}|_{\boldsymbol{x}^k}\left(x_j^{k+1}-x_j^k\right)+\cdots$$

The Jacobian matrix is defined as

$$\boldsymbol{J}_{ij}^k=\left.\frac{\partial f_i}{\partial x_j}\right|_{\boldsymbol{x}^k}$$

and the Newton – Raphson method is

$$\sum_{j=1}^{n}\boldsymbol{J}_{ij}^k\left(x^{k+1}-x^k\right)=-f_i\left(\boldsymbol{x}^k\right)$$

For convergence, the norm of the inverse of the Jacobian must be bounded, the norm of the function evaluated at the initial guess must be bounded, and the second derivative must be bounded [14, p. 115], [3, p. 12].

A review of the usefulness of solution methods for nonlinear equations is available [16]. This review concludes that the Newton – Raphson method may not be the most efficient. Broyden's method approximates the inverse to the Jacobian and is a good all-purpose method, but a good initial approximation to the Jacobian matrix is required. Furthermore, the rate of convergence deteriorates for large problems, for which the Newton – Raphson method is better. Brown's method [16] is very attractive, whereas Brent's is not worth the extra storage and computation. For large systems of equations, efficient software is available [11 – 13].

Homotopy methods can be used to ensure finding the solution when the problem is especially complicated. Suppose an attempt is made to solve $f(\boldsymbol{x})=0$, and it fails; however, $g(\boldsymbol{x})=0$ can be solved easily, where $g(\boldsymbol{x})$ is some function, perhaps a simplification of $f(\boldsymbol{x})$. Then, the two functions can be embedded in a homotopy by taking

$$\boldsymbol{h}(\boldsymbol{x},t)=t\,f(\boldsymbol{x})+(1-t)\,g(x)$$

In this equation, $\boldsymbol{h}$ can be a $n\times n$ matrix for problems involving n variables; then x is a vector of length n. Then $h(x, t)=0$ can be solved for $t=0$ and t gradually changes until at $t=1$, $\boldsymbol{h}(\boldsymbol{x}, 1)=f(\boldsymbol{x})$. If the Jacobian of $\boldsymbol{h}$ with respect to x is nonsingular on the homotopy path (as t varies), the method is guaranteed to work. In classical methods, the interval from $t=0$ to $t=1$ is broken up into N subdivisions. Set $\Delta t=1/N$ and solve for $t=0$, which is easy by the choice of $g(x)$. Then set $t=\Delta t$ and use the Newton – Raphson method to solve for x. Since the initial guess is presumably pretty good, this has a high chance of being successful. That solution is then used as the initial guess for $t=2\,\Delta t$ and the process is repeated by moving stepwise to $t=1$. If the Newton – Raphson method does not converge, then Δt must be reduced and a new solution attempted.

Another way of using homotopy is to create an ordinary differential equation by differentiating the homotopy equation along the path (where $\boldsymbol{h}=0$).

$$\frac{\mathrm{d}\boldsymbol{h}\left[\boldsymbol{x}(t),t\right]}{\mathrm{d}t}=\frac{\partial\boldsymbol{h}}{\partial\boldsymbol{x}}\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}+\frac{\partial\boldsymbol{h}}{\partial t}=0$$

This can be expressed as an ordinary differential equation for $x(t)$:

$$\frac{\partial\boldsymbol{h}}{\partial\boldsymbol{x}}\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}=-\frac{\partial\boldsymbol{h}}{\partial t}$$

If Euler's method is used to solve this equation, a value x^0 is used, and $\mathrm{d}x/\mathrm{d}t$ from the above equation is solved for. Then

$$\boldsymbol{x}^{1,0}=\boldsymbol{x}^0+\Delta t\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}$$

is used as the initial guess and the homotopy equation is solved for $\boldsymbol{x}^1$.

$$\frac{\partial\boldsymbol{h}}{\partial x}\left(\boldsymbol{x}^{1,k+1}-\boldsymbol{x}^{1,k}\right)=-\boldsymbol{h}\left(\boldsymbol{x}^{1,k},t\right)$$

Then t is increased by Δt and the process is repeated.

In arc-length parameterization, both x and t are considered parameterized by a parameter s, which is thought of as the arc length along a curve. Then the homotopy equation is written along with the arc-length equation.

$$\frac{\partial\boldsymbol{h}}{\partial\boldsymbol{x}}\frac{\mathrm{d}x}{\mathrm{d}\boldsymbol{s}}+\frac{\partial\boldsymbol{h}}{\partial t}\frac{\mathrm{d}t}{\mathrm{d}\boldsymbol{s}}=0$$

$$\frac{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}}{\mathrm{d}\boldsymbol{s}}\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}s}+\left(\frac{\mathrm{d}t}{\mathrm{d}\boldsymbol{s}}\right)^2=1$$

The initial conditions are

$$\boldsymbol{x}(0)=\boldsymbol{x}^0$$
$$t(0)=0$$

The advantage of this approach is that it works even when the Jacobian of $\boldsymbol{h}$ becomes singular because the full matrix is rarely singular. Illustrations applied to chemical engineering are available [17]. Software to perform these computations is available (called LOCA) [18].

1.4. Linear Difference Equations

Difference equations arise in chemical engineering from staged operations, such as distillation or extraction, as well as from differential equations modeling adsorption and chemical reactors. The value of a variable in the *n*-th stage is noted by a subscript *n*. For example, if $y_{n,i}$ denotes the mole fraction of the *i*-th species in the vapor phase on the *n*-th stage of a distillation column, $x_{n,i}$ is the corresponding liquid mole fraction, *R* the reflux ratio (ratio of liquid returned to the column to product removed from the condenser), and $K_{n,i}$ the equilibrium constant, then the mass balances about the top of the column give

$$y_{n+1,i} = \frac{R}{R+1}x_{n,i} + \frac{1}{R+1}x_{0,i}$$

and the equilibrium equation gives

$$y_{n,i} = K_{n,i}x_{n,i}$$

If these are combined,

$$K_{n+1,i}x_{n+1,i} = \frac{R}{R+1}x_{n,i} + \frac{1}{R+1}x_{0,i}$$

is obtained, which is a linear difference equation. This particular problem is quite complicated, and the interested reader is referred to [19, Chap. 6]. However, the form of the difference equation is clear. Several examples are given here for solving difference equations. More complete information is available in [20].

An equation in the form

$$x_{n+1} - x_n = f_{n+1}$$

can be solved by

$$x_n = \sum_{i=1}^{n} f_i$$

Usually, difference equations are solved analytically only for linear problems. When the coefficients are constant and the equation is linear and homogeneous, a trial solution of the form

$$x_n = \varphi^n$$

is attempted; φ is raised to the power *n*. For example, the difference equation

$$cx_{n-1} + bx_n + ax_{n+1} = 0$$

coupled with the trial solution would lead to the equation

$$a\varphi^2 + b\varphi + c = 0$$

This gives

$$\varphi_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

and the solution to the difference equation is

$$x_n = A\varphi_1^n + B\varphi_2^n$$

where *A* and *B* are constants that must be specified by boundary conditions of some kind.

When the equation is nonhomogeneous, the solution is represented by the sum of a particular solution and a general solution to the homogeneous equation.

$$x_n = x_{n,P} + x_{n,H}$$

The general solution is the one found for the homogeneous equation, and the particular solution is any solution to the nonhomogeneous difference equation. This can be found by methods analogous to those used to solve differential equations: the method of undetermined coefficients and the method of variation of parameters. The last method applies to equations with variable coefficients, too. For a problem such as

$$x_{n+1} - f_n x_n = 0$$
$$x_0 = c$$

the general solution is

$$x_n = c\prod_{i=1}^{n} f_{i-1}$$

This can then be used in the method of variation of parameters to solve the equation

$$x_{n+1} - f_n x_n = g_n$$

1.5. Eigenvalues

The $n \times n$ matrix $\boldsymbol{A}$ has *n* eigenvalues λ_i, $i = 1, \ldots, n$, which satisfy

$$\det(\boldsymbol{A} - \lambda_i \boldsymbol{I}) = 0$$

If this equation is expanded, it can be represented as

$$P_n(\lambda) = (-\lambda)^n + a_1(-\lambda)^{n-1} + a_2(-\lambda)^{n-2} + \cdots + a_{n-1}(-\lambda) + a_n = 0$$

If the matrix $\boldsymbol{A}$ has real entries then a_i are real numbers, and the eigenvalues either are real

numbers or occur in pairs as complex numbers with their complex conjugates (for definition of complex numbers, see Chap. 3). The Hamilton – Cayley theorem [19, p. 127] states that the matrix $\boldsymbol{A}$ satisfies its own characteristic equation.

$$P_n(\boldsymbol{A}) = (-\boldsymbol{A})^n + a_1(-\boldsymbol{A})^{n-1} + a_2(-\boldsymbol{A})^{n-2} + \cdots + a_{n-1}(-\boldsymbol{A}) + a_n \boldsymbol{I} = 0$$

A laborious way to find the eigenvalues of a matrix is to solve the n-th order polynomial for the λ_i — far too time consuming. Instead the matrix is transformed into another form whose eigenvalues are easier to find. In the *Givens method* and the *Housholder method* the matrix is transformed into the tridiagonal form; then, in a fixed number of calculations the eigenvalues can be found [15]. The Givens method requires $4\,n^3/3$ operations to transform a real symmetric matrix to tridiagonal form, whereas the Householder method requires half that number [14]. Once the tridiagonal form is found, a Sturm sequence is applied to determine the eigenvalues. These methods are especially useful when only a few eigenvalues of the matrix are desired.

If all the eigenvalues are needed, the $\boldsymbol{Q}\,\boldsymbol{R}$ algorithm is preferred [21].

The eigenvalues of a certain tridiagonal matrix can be found analytically. If $\boldsymbol{A}$ is a tridiagonal matrix with

$$a_{ii} = p, a_{i,i+1} = q, a_{i+1,i} = r, qr > 0$$

then the eigenvalues of $\boldsymbol{A}$ are [22]

$$\lambda_i = p + 2(qr)^{1/2}\cos\frac{i\pi}{n+1} \quad i = 1,2,\ldots,n$$

This result is useful when finite difference methods are applied to the diffusion equation.

2. Approximation and Integration

2.1. Introduction

Two types of problems arise frequently:

1) A *function* is known exactly at a set of points and an interpolating function is desired. The interpolant may be exact at the set of points, or it may be a "best fit" in some sense. Alternatively it may be desired to represent a function in some other way.
2) *Experimental data* must be fit with a mathematical model. The data have experimental error, so some uncertainty exists. The parameters in the model as well as the uncertainty in the determination of those parameters is desired.

These problems are addressed in this chapter. Section 2.2 gives the properties of polynomials defined over the whole domain and Section 2.3 of polynomials defined on segments of the domain. In Section 2.4, quadrature methods are given for evaluating an integral. Least-squares methods for parameter estimation for both linear and nonlinear models are given in Sections 2.5. Fourier transforms to represent discrete data are described in Section 2.7. The chapter closes with extensions to two-dimensional representations.

2.2. Global Polynomial Approximation

A *global* polynomial $P_m(x)$ is defined over the entire region of space

$$P_m(x) = \sum_{j=0}^{m} c_j x^j$$

This polynomial is of degree m (highest power is x^m) and order $m + 1$ ($m + 1$ parameters $\{c_j\}$). If a set of $m + 1$ points is given,

$$y_1 = f(x_1), y_2 = f(x_2), \ldots, y_{m+1} = f(x_{m+1})$$

then Lagrange's formula yields a polynomial of degree m that goes through the $m + 1$ points:

$$P_m(x) = \frac{(x-x_2)(x-x_3)\cdots(x-x_{m+1})}{(x_1-x_2)(x_1-x_3)\cdots(x_1-x_{m+1})} y_1 + \frac{(x-x_1)(x-x_3)\cdots(x-x_{m+1})}{(x_2-x_1)(x_2-x_3)\cdots(x_2-x_{m+1})} y_2 + \cdots + \frac{(x-x_1)(x-x_2)\cdots(x-x_m)}{(x_{m+1}-x_1)(x_{m+1}-x_2)\cdots(x_{m+1}-x_m)} y_{m+1}$$

Note that each coefficient of y_j is a polynomial of degree m that vanishes at the points $\{x_j\}$ (except for one value of j) and takes the value of 1.0 at that point, i.e.,

$$P_m(x_j) = y_j \quad j = 1, 2, \ldots, m+1$$

If the function $f(x)$ is known, the error in the approximation is [23]

$$|\text{error}(x)| \le \frac{|x_{m+1} - x_1|^{m+1}}{(m+1)!} \max_{x_1 \le x \le x_{m+1}} \left| f^{(m+1)}(x) \right|$$

The evaluation of $P_m\,(x)$ at a point other than the defining points can be made with Neville's algorithm [15]. Let P_1 be the value at x of the unique function passing through the point (x_1, y_1); i.e., $P_1 = y_1$. Let P_{12} be the value at x of the unique polynomial passing through the points x_1 and x_2. Likewise, $P_{ijk\ldots r}$ is the unique polynomial passing through the points x_i, x_j, x_k, . . . , x_r. The following scheme is used:

$$\begin{array}{llllll} x_1 & y_1 = P_1 & & & & \\ & & P_{12} & & & \\ x_2 & y_2 = P_2 & & P_{123} & & P_{1234} \\ & & P_{23} & & P_{234} & \\ x_3 & y_3 = P_3 & & & & \\ & & P_{34} & & & \\ x_4 & y_4 = P_4 & & & & \end{array}$$

These entries are defined by using

$$P_{i(i+1)\cdots(i+m)} = \frac{(x-x_{i+m})P_{i(i+1)\cdots(i+m-1)}+(x_i-x)P_{(i+1)(i+2)\cdots(i+m)}}{x_i-x_{i+m}}$$

Consider P_{1234}: the terms on the right-hand side of the equation involve P_{123} and P_{234}. The "parents," P_{123} and P_{234}, already agree at points 2 and 3. Here $i = 1$, $m = 3$; thus, the parents agree at $x_{i+1}, \ldots, x_{i+m-1}$ already. The formula makes $P_{i(i+1)\ldots(i+m)}$ agree with the function at the additional points x_{i+m} and x_i. Thus, $P_{i(i+1)\ldots(i+m)}$ agrees with the function at all the points $\{x_i, x_{i+1}, \ldots, x_{i+m}\}$.

Orthogonal Polynomials. Another form of the polynomials is obtained by defining them so that they are orthogonal. It is required that $P_m\,(x)$ be orthogonal to $P_k\,(x)$ for all $k = 0, \ldots, m-1$.

$$\int_a^b W\,(x)\,P_k\,(x)\,P_m\,(x)\,\mathrm{d}x = 0 \qquad k = 0,1,2,\ldots,m-1$$

The orthogonality includes a nonnegative weight function, $W\,(x) \geq 0$ for all $a \leq x \leq b$. This procedure specifies the set of polynomials to within multiplicative constants, which can be set either by requiring the leading coefficient to be one or by requiring the norm to be one.

$$\int_a^b W\,(x)\,P_m^2\,(x)\,\mathrm{d}x = 1$$

The polynomial $P_m\,(x)$ has m roots in the closed interval a to b.

The polynomial

$$p\,(x) = c_0 P_0\,(x) + c_1 P_1\,(x) + \cdots c_m\,P_m\,(x)$$

minimizes

$$I = \int_a^b W\,(x)\,[f\,(x) - p\,(x)]^2\,\mathrm{d}x$$

when

$$c_j = \frac{\int_a^b W(x)\,f(x)P_j(x)\mathrm{d}x}{W_j},$$

$$W_j = \int_a^b W\,(x)\,P_j^2\,(x)\,\mathrm{d}x$$

Note that each c_j is independent of m, the number of terms retained in the series. The minimum value of I is

$$I_{\min} = \int_a^b W\,(x)\;f^2\,(x)\,\mathrm{d}x - \sum_{j=0}^{m} W_j c_j^2$$

Such functions are useful for continuous data, i.e., when $f\,(x)$ is known for all x.

Typical orthogonal polynomials are given in Table 1. Chebyshev polynomials are used in spectral methods (see Chap. 8). The last two rows of Table 1 are widely used in the orthogonal collocation method in chemical engineering.

Table 1. Orthogonal polynomials [15, 23]

a	b	$W\,(x)$	Name	Recursion relation
-1	1	1	Legendre	$(i+1)\,P_{i+1} = (2\,i+1)xP_i - i\,P_{i-1}$
-1	1	$\frac{1}{\sqrt{1-x^2}}$	Chebyshev	$T_{i+1} = 2xT_i - T_{i-1}$
0	1	$x^{\,q-1}\,(1-x)^{p-q}$	Jacobi (p, q)	
$-\infty$	∞	e^{-x^2}	Hermite	$H_{i+1} = 2xH_i - 2\,i\,H_{i-1}$
0	∞	$x^{\,c}\,\mathrm{e}^{-x}$	Laguerre (c)	$(i+1)\,L_{i+1}^{c} = (-x+2\,i+c+1)\,L_{i}^{c} - (i+c)\,L_{i-1}^{c}$
0	1	1	shifted Legendre	
0	1	1	shifted Legendre, function of x^2	

The last entry (the shifted Legendre polynomial as a function of x^2) is defined by

$$\int_0^1 W\left(x^2\right) P_k\left(x^2\right) P_m\left(x^2\right) x^{a-1}\mathrm{d}x = 0$$
$$k = 0,1,\ldots,m-1$$

where $a = 1$ is for planar, $a = 2$ for cylindrical, and $a = 3$ for spherical geometry. These functions are useful if the solution can be proved to be an even function of x.

Rational Polynomials. Rational polynomials are ratios of polynomials. A rational polynomial $R_{i(i+1)\ldots(i+m)}$ passing through $m + 1$ points

$$y_i = f\left(x_i\right),\ i = 1,\ldots,m+1$$

is

$$R_{i(i+1)\cdots(i+m)} = \frac{P_\mu(x)}{Q_\nu(x)} = \frac{p_0+p_1x+\cdots+p_\mu x^\mu}{q_0+q_1x+\cdots+q_\nu x^\nu},$$
$$m+1 = \mu+\nu+1$$

An alternative condition is to make the rational polynomial agree with the first $m + 1$ terms in the power series, giving a Padé approximation, i.e.,

$$\frac{\mathrm{d}^k R_{i(i+1)\cdots(i+m)}}{\mathrm{d}x^k} = \frac{\mathrm{d}^k f\left(x\right)}{\mathrm{d}x^k}\quad k = 0,\ldots,m$$

The Bulirsch – Stoer recursion algorithm can be used to evaluate the polynomial:

$$R_{i(i+1)\cdots(i+m)} = R_{(i+1)\cdots(i+m)}$$
$$+\frac{R_{(i+1)\cdots(i+m)}-R_{i(i+1)\cdots(i+m-1)}}{\mathrm{Den}}$$
$$\mathrm{Den} = \left(\frac{x-x_i}{x-x_{i+m}}\right)$$
$$\left(1-\frac{R_{(i+1)\cdots(i+m)}R_{i(i+1)\cdots(i+m-1)}}{R_{(i+1)\cdots(i+m)}-R_{(i+1)\cdots(i+m-1)}}\right)-1$$

Rational polynomials are useful for approximating functions with poles and singularities, which occur in Laplace transforms (see Section 4.2).

Fourier series are discussed in Section 4.1. Representation by sums of exponentials is also possible [24].

In summary, for discrete data, Legendre polynomials and rational polynomials are used. For continuous data a variety of orthogonal polynomials and rational polynomials are used. When the number of conditions (discrete data points) exceeds the number of parameters, then see Section 2.5.

2.3. Piecewise Approximation

Piecewise approximations can be developed from difference formulas [3]. Consider a case in which the data points are equally spaced

$$x_{n+1}-x_n = \Delta x$$
$$y_n = y\left(x_n\right)$$

forward differences are defined by

$$\Delta y_n = y_{n+1}-y_n$$
$$\Delta^2 y_n = \Delta y_{n+1}-\Delta y_n = y_{n+2}-2y_{n+1}+y_n$$

Then, a new variable is defined

$$\alpha = \frac{x_\alpha - x_0}{\Delta x}$$

and the finite interpolation formula through the points $y_0, y_1, \ldots, y_n$ is written as follows:

$$y_\alpha = y_0+\alpha\Delta y_0+\frac{\alpha(\alpha-1)}{2!}\Delta^2 y_0+\cdots+$$
$$\frac{\alpha(\alpha-1)\cdots(\alpha-n+1)}{n!}\Delta^n y_0 \qquad (1)$$

Keeping only the first two terms gives a straight line through $(x_0, y_0) - (x_1, y_1)$; keeping the first three terms gives a quadratic function of position going through those points plus (x_2, y_2). The value $\alpha = 0$ gives $x = x_0$; $\alpha = 1$ gives $x = x_1$, etc.

Backward differences are defined by

$$\nabla y_n = y_n-y_{n-1}$$
$$\nabla^2 y_n = \nabla y_n-\nabla y_{n-1} = y_n-2y_{n-1}+y_{n-2}$$

The interpolation polynomial of order n through the points $y_0, y_{-1}, y_{-2}, \ldots$ is

$$y_\alpha = y_0+\alpha\nabla y_0+\frac{\alpha(\alpha+1)}{2!}\nabla^2 y_0+\cdots+$$
$$\frac{\alpha(\alpha+1)\cdots(\alpha+n-1)}{n!}\nabla^n y_0$$

The value $\alpha = 0$ gives $x = x_0$; $\alpha = -1$ gives $x = x_{-1}$. Alternatively, the interpolation polynomial of order n through the points $y_1, y_0, y_{-1}, \ldots$ is

$$y_\alpha = y_1+(\alpha-1)\nabla y_1+\frac{\alpha(\alpha-1)}{2!}\nabla^2 y_1+\cdots+$$
$$\frac{(\alpha-1)\alpha(\alpha+1)\cdots(\alpha+n-2)}{n!}\nabla^n y_1$$

Now $\alpha = 1$ gives $x = x_1$; $\alpha = 0$ gives $x = x_0$.

The *finite element method* can be used for piecewise approximations [3]. In the finite element method the domain $a \le x \le b$ is divided

into elements as shown in Figure 3. Each function $N_i\,(x)$ is zero at all nodes except x_i; $N_i\,(x_i) = 1$. Thus, the approximation is

$$y\,(x) = \sum_{i=1}^{NT} c_i N_i\,(x) = \sum_{i=1}^{NT} y\,(x_i)\,N_i\,(x)$$

where $c_i = y\,(x_i)$. For convenience, the trial functions are defined within an element by using new coordinates:

$$u = \frac{x - x_i}{\Delta x_i}$$

The Δx_i need not be the same from element to element. The trial functions are defined as $N_i\,(x)$ (Fig. 3 A) in the global coordinate system and $N_I\,(u)$ (Fig. 3 B) in the local coordinate system (which also requires specification of the element). For $x_1 < x < x_{i+1}$

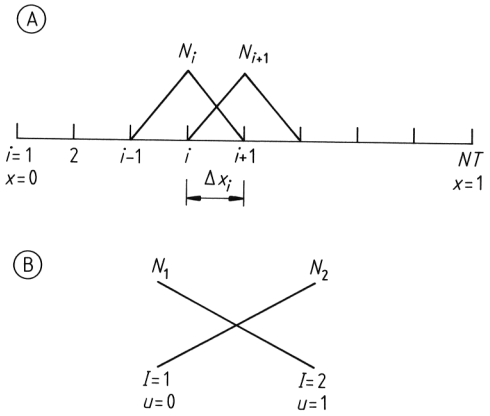

Figure 3. Galerkin finite element method – linear functions
A) Global numbering system; B) Local numbering system

$$y\,(x) = \sum_{i=1}^{NT} c_i N_i\,(x) = c_i N_i\,(x) + c_{i+1} N_{i+1}\,(x)$$

because all the other trial functions are zero there. Thus

$$y\,(x) = c_i N_{I=1}\,(u) + c_{i+1} N_{I=2}\,(u)\,,$$

$$x_i < x < x_{i+1},\ 0 < u < 1$$

Then

$$N_{I=1} = 1 - u,\ N_{I=2} = u$$

and the expansion is rewritten as

$$y\,(x) = \sum_{I=1}^{2} c_I^e N_I\,(u) \qquad (2)$$

x in e-th element and $c_i = c_I^e$ within the element e. Thus, given a set of points $(x_i,\ y_i)$, a finite element approximation can be made to go through them.

Quadratic approximations can also be used within the element (see Fig. 4). Now the trial functions are

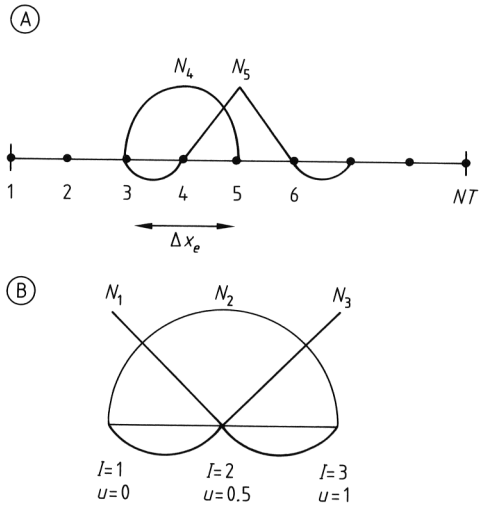

Figure 4. Finite elements approximation – quadratic elements
A) Global numbering system; B) Local numbering system

$$N_{I=1} = 2\,(u-1)\left(u - \tfrac{1}{2}\right)$$

$$N_{I=2} = 4u\,(1-u) \qquad (3)$$

$$N_{I=3} = 2u\left(u - \tfrac{1}{2}\right)$$

The approximation going through an odd number of points $(x_i,\ y_i)$ is then

$$y\,(x) = \sum_{I=1}^{3} c_I^e N_I\,(u) \quad x \text{ in } e\text{−th element}$$

$$\text{with } c_I^e = y\,(x_i)\,,\ i = (e-1)\,2 + I$$

$$\text{in the } e\text{−th element}$$

Hermite cubic polynomials can also be used; these are continuous and have continuous first derivatives at the element boundaries [3].

Splines. Splines are functions that match given values at the points $x_1, \ldots, x_{NT}$, shown in Figure 5, and have continuous derivatives up to some order at the knots, or the points $x_2, \ldots, x_{NT-1}$. *Cubic splines* are most common. In this case the function is represented by a cubic polynomial within each interval and has continuous first and second derivatives at the knots.

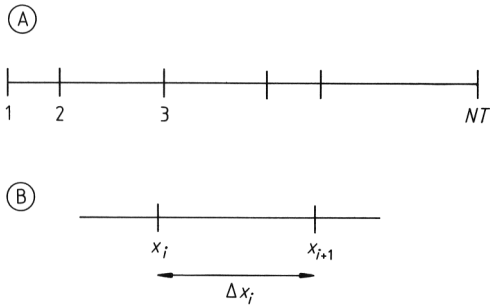

Figure 5. Finite elements for cubic splines
A) Notation for spline knots. B) Notation for one element

Consider the points shown in Figure 5 A. The notation for each interval is shown in Figure 5 B. Within each interval the function is represented as a cubic polynomial.

$$C_i(x) = a_{0i} + a_{1i}x + a_{2i}x^2 + a_{3i}x^3$$

The interpolating function takes on specified values at the knots.

$$C_{i-1}(x_i) = C_i(x_i) = f(x_i)$$

Given the set of values $\{x_i, f(x_i)\}$, the objective is to pass a smooth curve through those points, and the curve should have continuous first and second derivatives at the knots.

$$C'_{i-1}(x_i) = C'_i(x_i)$$

$$C''_{i-1}(x_i) = C''_i(x_i)$$

The formulas for the cubic spline are derived as follows for one region. Since the function is a cubic function the third derivative is constant and the second derivative is linear in x. This is written as

$$C''_i(x) = C''_i(x_i) + \left[C''_i(x_{i+1}) - C''_i(x_i)\right] \frac{x - x_i}{\Delta x_i}$$

and integrated once to give

$$C'_i(x) = C'_i(x_i) + C''_i(x_i)(x - x_i) + \left[C''_i(x_{i+1}) - C''_i(x_i)\right] \frac{(x - x_i)^2}{2\Delta x_i}$$

and once more to give

$$C_i(x) = C_i(x_i) + C'_i(x_i)(x - x_i) + C''_i(x_i) \frac{(x - x_i)^2}{2} + \left[C''_i(x_{i+1}) - C''_i(x_i)\right] \frac{(x - x_i)^3}{6\Delta x_i}$$

Now

$$y_i = C_i(x_i), y'_i = C'_i(x_i), y''_i = C''_i(x_i)$$

is defined so that

$$C_i(x) = y_i + y'_i(x - x_i) + \frac{1}{2}y''_i(x - x_i)^2 + \frac{1}{6\Delta x_i}\left(y''_{i+1} - y''_i\right)(x - x_i)^3$$

A number of algebraic steps make the interpolation easy. These formulas are written for the i-th element as well as the $i-1$-th element. Then the continuity conditions are applied for the first and second derivatives, and the values y'_i and y'_{i-1} are eliminated [15]. The result is

$$y''_{i-1}\Delta x_{i-1} + y''_i\, 2(\Delta x_{i-1} + \Delta x_i) + y''_{i+1}\Delta x_i = -6\left(\frac{y_i - y_{i-1}}{\Delta x_{i-1}} - \frac{y_{i+1} - y_i}{\Delta x_i}\right)$$

This is a tridiagonal system for the set of $\{y''_i\}$ in terms of the set of $\{y_i\}$. Since the continuity conditions apply only for $i = 2, \ldots, NT - 1$, only $NT - 2$ conditions exist for the NT values of y''_i. Two additional conditions are needed, and these are usually taken as the value of the second derivative at each end of the domain, y''_1, y''_{NT}. If these values are zero, the *natural cubic splines* are obtained; they can also be set to achieve some other purpose, such as making the first derivative match some desired condition at the two ends. With these values taken as zero, in the natural cubic spline, an $NT - 2$ system of tridiagonal equations exists, which is easily solved. Once the second derivatives are known at each of the knots, the first derivatives are given by

$$y'_i = \frac{y_{i+1} - y_i}{\Delta x_i} - y''_i \frac{\Delta x_i}{3} - y''_{i+1} \frac{\Delta x_i}{6}$$

The function itself is then known within each element.

Orthogonal Collocation on Finite Elements. In the method of orthogonal collocation on finite elements the solution is expanded in a polynomial of order $NP = NCOL + 2$ within each element [3]. The choice $NCOL = 1$ corresponds to using quadratic polynomials, whereas $NCOL = 2$ gives cubic polynomials. The notation is shown in Figure 6. Set the function to a known value at the two endpoints

$$y_1 = y(x_1)$$

$$y_{NT} = y(x_{NT})$$

and then at the $NCOL$ interior points to each element

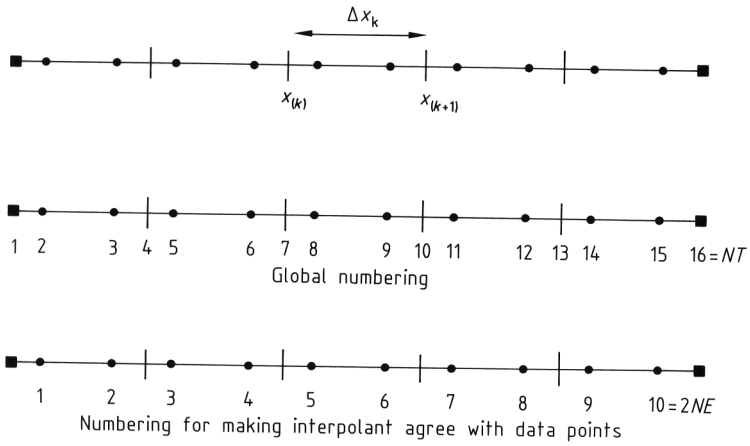

Figure 6. Notation for orthogonal collocation on finite elements
• Residual condition; ■ Boundary conditions; | Element boundary, continuity
NE = total no. of elements.
NT = (*NCOL* + 1) *NE* + 1

$y_i^e = y_i = y(x_i), i = (NCOL+1)\,e+I$

The actual points x_i are taken as the roots of the orthogonal polynomial.

$P_{NCOL}(u) = 0$ gives $u_1, u_2, \ldots, u_{NCOL}$

and then

$x_i = x_{(e)} + \Delta x_e u_I \equiv x_I^e$

The first derivatives must be continuous at the element boundaries:

$$\frac{\mathrm{d}y}{\mathrm{d}x}\Big|_{x=x_{(2)-}} = \frac{\mathrm{d}y}{\mathrm{d}x}\Big|_{x=x_{(2)+}}$$

Within each element the interpolation is a polynomial of degree *NCOL* + 1. Overall the function is continuous with continuous first derivatives. With the choice *NCOL* = 2, the same approximation is achieved as with Hermite cubic polynomials.

2.4. Quadrature

To calculate the value of an integral, the function can be approximated by using each of the methods described in Section 2.3. Using the first three terms in Equation 1 gives

$$\int_{x_0}^{x_0+h} y(x)\,\mathrm{d}x = \int_0^1 y_\alpha h\,\mathrm{d}\alpha$$
$$= \tfrac{h}{2}(y_0+y_1) - \tfrac{1}{12}h^3 y_0''(\xi),\ x_0 \le \xi \le x_0+h$$

This corresponds to passing a straight line through the points (x_0, y_0), (x_1, y_1) and integrating under the interpolant. For equally spaced points at $a = x_0$, $a + \Delta x = x_1$, $a + 2\,\Delta x = x_2$, . . . , $a + N\,\Delta x = x_N$, $a + (N + 1)\,\Delta x = b = x_{n+1}$, the trapezoid rule is obtained.

Trapezoid Rule.

$$\int_a^b y(x)\,\mathrm{d}x = \tfrac{h}{2}(y_0+2y_1+2y_2+\cdots+2y_N$$
$$+y_{N+1}) + O(h^3)$$

The first five terms in Equation 1 are retained and integrated over two intervals.

$$\int_{x_0}^{x_0+2h} y(x)\,\mathrm{d}x = \int_0^2 y_\alpha h\,\mathrm{d}\alpha = \tfrac{h}{3}(y_0+4y_1+y_2)$$
$$-\tfrac{h^5}{90}y_0^{(\mathrm{IV})}(\xi),\ x_0 \le \xi \le x_0+2h$$

This corresponds to passing a quadratic function through three points and integrating. For an even number of intervals and an odd number of points, 2 *N* + 1, with $a = x_0$, $a + \Delta x = x_1$, $a + 2\,\Delta x = x_2$, . . . , $a + 2\,N\,\Delta x = b$, Simpson's rule is obtained.

Simpson's Rule.

$$\int_a^b y(x)\,\mathrm{d}x = \tfrac{h}{3}(y_0+4y_1+2y_2+4y_3+2y_4$$
$$+\cdots+2y_{2N-1}+4y_{2N}+y_{2N+1}) + O(h^5)$$

Within each pair of intervals the interpolant is continuous with continuous derivatives, but only the function is continuous from one pair to another.

If the *finite element representation* is used (Eq. 2), the integral is

$$\int_{x_i}^{x_{i+1}} y(x)\,\mathrm{d}x = \int_0^1 \sum_{I=1}^{2} c_I^e N_I(u)\,(x_{i+1}-x_i)\,\mathrm{d}u$$
$$= \Delta x_i \sum_{I=1}^{2} c_I^e \int_0^1 N_I(u)\,\mathrm{d}u = \Delta x_i \left(c_1^e \tfrac{1}{2} + c_2^e \tfrac{1}{2}\right)$$
$$= \tfrac{\Delta x_i}{2}\,(y_i + y_{i+1})$$

Since $c_1^e = y_i$ and $c_2^e = y_{i+1}$, the result is the same as the trapezoid rule. These formulas can be added together to give *linear elements*:

$$\int_a^b y(x)\,\mathrm{d}x = \sum\nolimits_e \frac{\Delta x_e}{2}\,(y_1^e + y_2^e)$$

If the *quadratic* expansion is used (Eq. 3), the endpoints of the element are x_i and x_{i+2}, and x_{i+1} is the midpoint, here assumed to be equally spaced between the ends of the element:

$$\int_{x_i}^{x_{i+2}} y(x)\,\mathrm{d}x = \int_0^1 \sum_{I=1}^{3} c_I^e N_I(u)\,(x_{i+2}-x_i)\,\mathrm{d}u$$
$$= \Delta x_i \sum_{I=1}^{3} c_I^e \int_0^1 N_I(u)\,\mathrm{d}u$$
$$= \Delta x_e \left(c_1^e \tfrac{1}{6} + c_2^e \tfrac{2}{3} + c_3^e \tfrac{1}{6}\right)$$

For many elements, with different Δx^e, *quadratic elements:*

$$\int_a^b y(x) = \sum\nolimits_e \frac{\Delta x_e}{6}\,(y_1^e + 4y_2^e + y_3^e)$$

If the element sizes are all the same this gives Simpson's rule.

For cubic splines the quadrature rule within one element is

$$\int_{x_i}^{x_{i+1}} C_i(x)\,\mathrm{d}x = \tfrac{1}{2}\Delta x_i\,(y_i + y_{i+1}) - \tfrac{1}{24}\Delta x_i^3 \left(y_i'' + y_{i+1}''\right)$$

For the entire interval the quadrature formula is

$$\int_{x_1}^{x_{NT}} y(x)\,\mathrm{d}x = \tfrac{1}{2}\sum_{i=1}^{NT-1} \Delta x_i (y_i + y_{i+1}) - \tfrac{1}{24}\sum_{i=1}^{NT-1} \Delta x_i^3 (y_i'' + y_{i+1}'')$$

with $y_1'' = 0$, $y''_{NT} = 0$ for natural cubic splines.

When orthogonal polynomials are used, as in Equation 1, the m roots to $P_m(x) = 0$ are chosen as quadrature points and called points $\{x_j\}$. Then the quadrature is *Gaussian:*

$$\int_0^1 y(x)\,\mathrm{d}x = \sum_{j=1}^{m} W_j\, y(x_j)$$

The quadrature is exact when y is a polynomial of degree $2m - 1$ in x. The m weights and m Gauss points result in 2 m parameters, chosen to exactly represent a polynomial of degree $2m - 1$, which has 2 m parameters. The Gauss points and weights are given in Table 2. The weights can be defined with $W(x)$ in the integrand as well.

Table 2. Gaussian quadrature points and weights *

N	x_i	W_i
1	0.5000000000	0.6666666667
2	0.2113248654	0.5000000000
	0.7886751346	0.5000000000
3	0.1127016654	0.2777777778
	0.5000000000	0.4444444445
	0.8872983346	0.2777777778
4	0.0694318442	0.1739274226
	0.3300094783	0.3260725774
	0.6699905218	0.3260725774
	0.9305681558	0.1739274226
5	0.0469100771	0.1184634425
	0.2307653450	0.2393143353
	0.5000000000	0.2844444444
	0.7692346551	0.2393143353
	0.9530899230	0.1184634425

* For a given N the quadrature points $x_2, x_3, \ldots, x_{NP-1}$ are given above. $x_1 = 0$, $x_{NP} = 1$. For $N = 1$, $W_1 = W_3 = 1/6$ and for $N \geq 2$, $W_1 = W_{NP} = 0$.

For orthogonal collocation on finite elements the quadrature formula is

$$\int_0^1 y(x)\,\mathrm{d}x = \sum\nolimits_e \Delta x_e \sum_{j=1}^{NP} W_j\, y(x_j^e)$$

Each special polynomial has its own quadrature formula. For example, Gauss – Legendre polynomials give the quadrature formula

$$\int_0^\infty e^{-x} y(x)\,\mathrm{d}x = \sum_{i=1}^{n} W_i\, y(x_i)$$

(points and weights are available in mathematical tables) [23].

For Gauss – Hermite polynomials the quadrature formula is

$$\int_{-\infty}^{\infty} e^{-x^2} y(x)\,\mathrm{d}x = \sum_{i=1}^{n} W_i\, y(x_i)$$

(points and weights are available in mathematical tables) [23].

Romberg's method uses extrapolation techniques to improve the answer [15]. If I_1 is the value of the integral obtained by using interval size $h = \Delta x$, I_2 the value of I obtained by using interval size $h/2$, and I_0 the true value of I, then the error in a method is approximately h^m, or

$$I_1 \approx I_0 + ch^m$$

$$I_2 \approx I_0 + c\left(\frac{h}{2}\right)^m$$

Replacing the $\approx$ by an equality (an approximation) and solving for c and I_0 give

$$I_0 = \frac{2^m I_2 - I_1}{2^m - 1}$$

This process can also be used to obtain $I_1, I_2, \ldots$, by halving h each time, calculating new estimates from each pair, and calling them $J_1, J_2, \ldots$ (i.e., in the formula above, I_0 is replaced with J_1). The formulas are reapplied for each pair of J's to obtain $K_1, K_2, \ldots$. The process continues until the required tolerance is obtained.

I_1	I_2	I_3	I_4
	J_1	J_2	J_3
		K_1	K_2
			L_1

Romberg's method is most useful for a low-order method (small m) because significant improvement is then possible.

When the integrand has singularities, a variety of techniques can be tried. The integral may be divided into one part that can be integrated analytically near the singularity and another part that is integrated numerically. Sometimes a change of argument allows analytical integration. Series expansion might be helpful, too. When the domain is infinite, Gauss – Legendre or Gauss – Hermite quadrature can be used. Also a transformation can be made [15]. For example, let $u = 1/x$ and then

$$\int_a^b f(x)\,\mathrm{d}x = \int_{1/b}^{1/a} \frac{1}{u^2} f\left(\frac{1}{u}\right)\mathrm{d}u \quad a, b > 0$$

2.5. Least Squares

When fitting experimental data to a mathematical model, it is necessary to recognize that the experimental measurements contain error; the goal is to find the set of parameters in the model that best represents the experimental data. Reference [23] gives a complete treatment relating the least-squares procedure to maximum likelihood.

In a least-squares parameter estimation, it is desired to find parameters that minimize the sum of squares of the deviation between the experimental data and the theoretical equation

$$\chi^2 = \sum_{i=1}^{N} \left[\frac{y_i - y(x_i; a_1, a_2, \ldots, a_M)}{\sigma_i}\right]^2$$

where y_i is the i-th experimental data point for the value x_i, $y(x_i; a_1, a_2, \ldots, a_M)$ the theoretical equation at x_i, σ_i the standard deviation of the i-th measurement, and the parameters $\{a_1, a_2, \ldots, a_M\}$ are to be determined to minimize χ^2. The simplification is made here that the standard deviations are all the same. Thus, we minimize the variance of the curve fit.

$$\sigma^2 = \sum_{i=1}^{N} \frac{[y_i - y(x_i; a_1, a_2, \ldots, a_M)]^2}{N}$$

Linear Least Squares. When the model is a straight line, one is minimizing

$$\chi^2 = \sum_{i=1}^{N} [y_i - a - bx_i]^2$$

The linear correlation coefficient r is defined by

$$r = \frac{\sum_{i=1}^{N} (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{N} (x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{N} (y_i - \overline{y})^2}}$$

and

$$\chi^2 = (1 - r^2)\sum_{i=1}^{N} [y_i - \overline{y}]^2$$

where $\overline{y}$ is the average of y_i values. Values of r near 1 indicate a positive correlation; r near -1 means a negative correlation, and r near zero means no correlation. These parameters are easily found by using standard programs, such as Microsoft Excel.

Polynomial Regression. In polynomial regression, one expands the function in a polynomial in *x*.

$$y(x) = \sum_{j=1}^{M} a_j x^{j-1}$$

The parameters are easily determined using computer software. In Microsoft Excel, the data is put into columns A and B and the graph is created as for a linear curve fit. Then add a trendline and choose the degree of polynomial desired.

Multiple Regression. In multiple regression, any set of functions can be used, not just polynomials.

$$y(x) = \sum_{j=1}^{M} a_j f_j(x)$$

where the set of functions $\{f_j(x)\}$ is known and specified. Note that the unknown parameters $\{a_j\}$ enter the equation linearly. In this case, the spreadsheet can be expanded to have a column for *x*, and then successive columns for $f_j(x)$. In Microsoft Excel, choose Regression under Tools/Data Analysis, and complete the form. In addition to the actual correlation, one gets the expected variance of the unknowns, which allows one to assess how accurately they were determined.

Nonlinear Regression. In nonlinear regression, the same procedure is followed except that an optimization routine must be used to find the minimum χ^2. See Chapter 10.

2.6. Fourier Transforms of Discrete Data [15]

Suppose a signal $y(t)$ is sampled at equal intervals

$y_n = y(n\Delta), n = \ldots, -2, -1, 0, 1, 2, \ldots$

$\Delta =$ sampling rate

(e.g., number of samples per second)

The Fourier transform and inverse transform are

$$Y(\omega) = \int_{-\infty}^{\infty} y(t)\, e^{i\omega t} dt$$

$$y(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} Y(\omega)\, e^{-i\omega t} d\omega$$

(For definition of i, see Chap. 3.) The Nyquist critical frequency or critical angular frequency is

$$f_c = \frac{1}{2\Delta}, \omega_c = \frac{\pi}{\Delta}$$

If a function $y(t)$ is bandwidth limited to frequencies smaller than f_c, i.e.,

$$Y(\omega) = 0 \text{ for } \omega > \omega_c$$

then the function is completely determined by its samples y_n. Thus, the entire information content of a signal can be recorded by sampling at a rate $\Delta^{-1} = 2f_c$. If the function is not bandwidth limited, then aliasing occurs. Once a sample rate Δ is chosen, information corresponding to frequencies greater than f_c is simply aliased into that range. The way to detect this in a Fourier transform is to see if the transform approaches zero at $\pm f_c$; if not, aliasing has occurred and a higher sampling rate is needed.

Next, for *N* samples, where *N* is even

$$y_k = y(t_k),\ t_k = k\Delta,\ k = 0, 1, 2, \ldots, N-1$$

and the sampling rate is Δ; with only *N* values $\{y_k\}$ the complete Fourier transform $Y(\omega)$ cannot be determined. Calculate the value $Y(\omega_n)$ at the discrete points

$$\omega_n = \frac{2\pi n}{N\Delta}, n = -\frac{N}{2}, \ldots, 0, \ldots, \frac{N}{2}$$

$$Y_n = \sum_{k=0}^{N-1} y_k e^{2\pi i k n/N}$$

$$Y(\omega_n) = \Delta Y_n$$

The discrete inverse Fourier transform is

$$y_k = \frac{1}{N} \sum_{k=0}^{N-1} Y_n e^{-2\pi i k n/N}$$

The *fast Fourier transform* (*FFT*) is used to calculate the Fourier transform as well as the inverse Fourier transform. A discrete Fourier transform of length *N* can be written as the sum of two discrete Fourier transforms, each of length *N*/2, and each of these transforms is separated into two halves, each half as long. This continues until only one component is left. For this reason, *N* is taken as a power of 2, $N = 2^p$.

The vector $\{y_j\}$ is filled with zeroes, if need be, to make $N = 2^p$ for some p. For the computer program, see [15, p. 381]. The standard

Fourier transform takes N^2 operations to calculate, whereas the fast Fourier transform takes only $N \log_2 N$. For large N the difference is significant; at N = 100 it is a factor of 15, but for N = 1000 it is a factor of 100.

The discrete Fourier transform can also be used for *differentiating* a function; this is used in the spectral method for solving differential equations. Consider a grid of equidistant points:

$$x_n = n\Delta x, n = 0,1,2,\ldots, 2N-1, \Delta x = \frac{L}{2N}$$

the solution is known at each of these grid points $\{Y(x_n)\}$. First, the Fourier transform is taken:

$$y_k = \frac{1}{2N}\sum_{n=0}^{2N-1} Y(x_n)\, \mathrm{e}^{-2\mathrm{i}k\pi x_n/L}$$

The inverse transformation is

$$Y(x) = \frac{1}{L}\sum_{k=-N}^{N} y_k \mathrm{e}^{2\mathrm{i}k\pi x/L}$$

which is differentiated to obtain

$$\frac{\mathrm{d}Y}{\mathrm{d}x} = \frac{1}{L}\sum_{k=-N}^{N} y_k \frac{2\pi\mathrm{i}k}{L}\mathrm{e}^{2\mathrm{i}k\pi x/L}$$

Thus, at the grid points

$$\left.\frac{\mathrm{d}Y}{\mathrm{d}x}\right|_n = \frac{1}{L}\sum_{k=-N}^{N} y_k \frac{2\pi\mathrm{i}k}{L}\mathrm{e}^{2\mathrm{i}k\pi x_n/L}$$

The process works as follows. From the solution at all grid points the Fourier transform is obtained by using FFT $\{y_k\}$. This is multiplied by $2\pi\,\mathrm{i}\,k/L$ to obtain the Fourier transform of the derivative:

$$y'_k = y_k \frac{2\pi\mathrm{i}k}{L}$$

The inverse Fourier transform is then taken by using FFT, to give the value of the derivative at each of the grid points:

$$\left.\frac{\mathrm{d}Y}{\mathrm{d}x}\right|_n = \frac{1}{L}\sum_{k=-N}^{N} y'_k \mathrm{e}^{2\mathrm{i}k\pi x_n/L}$$

Any nonlinear term can be treated in the same way: evaluate it in real space at N points and take the Fourier transform. After processing using this transform to get the transform of a new function, take the inverse transform to obtain the new function at N points. This is what is done in direct numerical simulation of turbulence (DNS).

2.7. Two-Dimensional Interpolation and Quadrature

Bicubic splines can be used to interpolate a set of values on a regular array, $f(x_i, y_j)$. Suppose NX points occur in the x direction and NY points occur in the y direction. Press et al. [15] suggest computing NY different cubic splines of size NX along lines of constant y, for example, and storing the derivative information. To obtain the value of f at some point x, y, evaluate each of these splines for that x. Then do one spline of size NY in the y direction, doing both the determination and the evaluation.

Multidimensional integrals can also be broken down into one-dimensional integrals. For example,

$$\int_a^b \int_{f_1(x)}^{f_2(x)} z(x,y)\,\mathrm{d}x\mathrm{d}y = \int_a^b G(x)\,\mathrm{d}x;$$
$$G(x) = \int_{f_1(x)}^{f_2(x)} z(x,y)\,\mathrm{d}x$$

3. Complex Variables [25 – 31]

3.1. Introduction to the Complex Plane

A complex number is an ordered pair of real numbers, x and y, that is written as

$$z = x + \mathrm{i}y$$

The variable i is the imaginary unit which has the property

$$\mathrm{i}^2 = -1$$

The real and imaginary parts of a complex number are often referred to:

$$\mathrm{Re}\,(z) = x, \mathrm{Im}\,(z) = y$$

A complex number can also be represented graphically in the complex plane, where the real part of the complex number is the abscissa and the imaginary part of the complex number is the ordinate (see Fig. 7).

Another representation of a complex number is the *polar form*, where r is the magnitude and θ is the argument.

$$r = |x + \mathrm{i}\,y| = \sqrt{x^2+y^2},\ \theta = \arg(x + \mathrm{i}\,y)$$

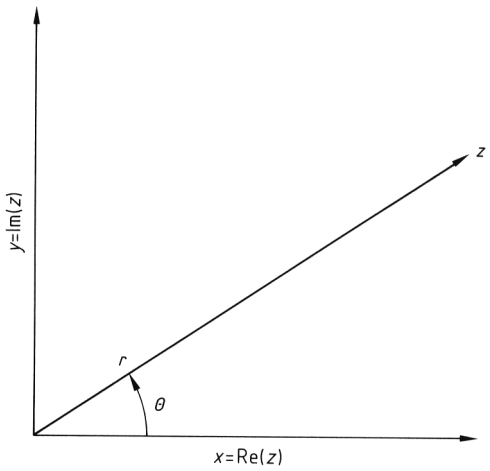

Figure 7. The complex plane

Write

$$z = x + \mathrm{i}\, y = r\,(\cos\theta + \mathrm{i}\sin\theta)$$

so that

$$x = r\cos\theta, y = r\sin\theta$$

and

$$\theta = \arctan\frac{y}{x}$$

Since the arctangent repeats itself in multiples of π rather than $2\,\pi$, the argument must be defined carefully. For example, the θ given above could also be the argument of $-\,(x + \mathrm{i}\, y)$. The function $z = \cos\theta + \mathrm{i}\sin\theta$ obeys $|z| = |\cos\theta + \mathrm{i}\sin\theta| = 1$.

The rules of equality, addition, and multiplication are

$$z_1 = x_1 + \mathrm{i}\, y_1, z_2 = x_2 + \mathrm{i}\, y_2$$

Equality :
$z_1 = z_2$ if and only if $x_1 = x_2$ and $y_1 = y_2$
Addition :
$$z_1 + z_2 = (x_1 + x_2) + \mathrm{i}\,(y_1 + y_2)$$
Multiplication :
$$z_1 z_2 = (x_1 x_2 - y_1 y_2) + \mathrm{i}\,(x_1 y_2 + x_2 y_1)$$

The last rule can be remembered by using the standard rules for multiplication, keeping the imaginary parts separate, and using $\mathrm{i}^2 = -1$. In the complex plane, addition is illustrated in Figure 8. In polar form, multiplication is

$$z_1 z_2 = r_1 r_2\,[\cos(\theta_1 + \theta_2) + \mathrm{i}\sin(\theta_1 + \theta_2)]$$

The magnitude of $z_1 + z_2$ is bounded by

$$|z_1 \pm z_2| \le |z_1| + |z_2| \text{ and } |z_1| - |z_2| \le |z_1 \pm z_2|$$

as can be seen in Figure 8. The magnitude and arguments in multiplication obey

$$|z_1 z_2| = |z_1|\,|z_2|, \arg(z_1 z_2) = \arg z_1 + \arg z_2$$

The *complex conjugate* is $z^* = x - \mathrm{i}\, y$ when $z = x + \mathrm{i}\, y$ and $|z^*| = |z|$, $\arg z^* = -\arg z$

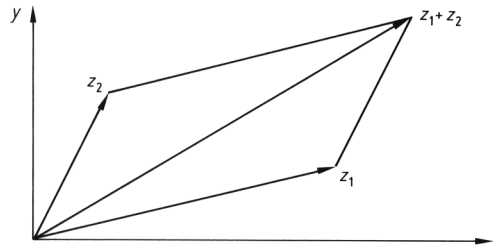

Figure 8. Addition in the complex plane

For complex conjugates then

$$z^*\, z = |z|^2$$

The reciprocal is

$$\frac{1}{z} = \frac{z^*}{|z|^2} = \frac{1}{r}\,(\cos\theta - \mathrm{i}\sin\theta)\,, \arg\left(\frac{1}{z}\right) = -\arg z$$

Then

$$\frac{z_1}{z_2} = \frac{x_1 + \mathrm{i}\, y_1}{x_2 + \mathrm{i}\, y_2} = (x_1 + \mathrm{i}\, y_1)\,\frac{x_2 - \mathrm{i}\, y_2}{x_2^2 + y_2^2}$$
$$= \frac{x_1 x_2 + y_1 y_2}{x_2^2 + y_2^2} + \mathrm{i}\,\frac{x_2 y_1 - x_1 y_2}{x_2^2 + y_2^2}$$

and

$$\frac{z_1}{z_2} = \frac{r_1}{r_2}\,[\cos(\theta_1 - \theta_2) + \mathrm{i}\sin(\theta_1 - \theta_2)]$$

3.2. Elementary Functions

Properties of elementary functions of complex variables are discussed here [32]. When the polar form is used, the argument must be specified because the same physical angle can be achieved with arguments differing by $2\,\pi$. A complex number taken to a real power obeys

$$u = z^n,\ |z^n| = |z|^n, \arg\,(z^n) = n\arg z\,(\mathrm{mod}\, 2\pi)$$

$$u = z^n = r^n\,(\cos n\theta + \mathrm{i}\sin n\theta)$$

Roots of a complex number are complicated by careful accounting of the argument

$z = w^{1/n}$ with $w = R(\cos\Theta + \mathrm{i}\sin\Theta)$, $0 \leq \Theta \leq 2\pi$

then

$$z_k = R^{1/n}\{\cos\left[\frac{\Theta}{n} + (k-1)\frac{2\pi}{n}\right] + \mathrm{i}\sin\left[\frac{\Theta}{n} + (k-1)\frac{2\pi}{n}\right]\}$$

such that

$(z_k)^n = w$ for every k

$$z = r(\cos\theta + \mathrm{i}\sin\theta)$$
$$r^n = R, n\theta = \Theta \pmod{2\pi}$$

The exponential function is

$$e^z = \mathrm{e}^x(\cos y + \mathrm{i}\sin y)$$

Thus,

$$z = r(\cos\theta + \mathrm{i}\sin\theta)$$

can be written

$$z = r\mathrm{e}^{\mathrm{i}\theta}$$

and

$$|\mathrm{e}^z| = \mathrm{e}^x, \arg\mathrm{e}^z = y \pmod{2\pi}$$

The exponential obeys

$\mathrm{e}^z \neq 0$ for every finite z

and is periodic with period 2π:

$$e^{z+2\pi\,\mathrm{i}} = \mathrm{e}^z$$

Trigonometric functions can be defined by using

$$e^{iy} = \cos y + \mathrm{i}\sin y, \text{ and } \mathrm{e}^{-\mathrm{i}y} = \cos y - \mathrm{i}\sin y$$

Thus,

$$\cos y = \frac{\mathrm{e}^{\mathrm{i}\,y} + \mathrm{e}^{-\mathrm{i}\,y}}{2} = \cosh \mathrm{i}\,y$$

$$\sin y = \frac{\mathrm{e}^{\mathrm{i}\,y} - \mathrm{e}^{-\mathrm{i}\,y}}{2\mathrm{i}} = -\mathrm{i}\sinh \mathrm{i}\,y$$

The second equation follows from the definitions

$$\cosh z \equiv \frac{\mathrm{e}^z + \mathrm{e}^{-z}}{2},\ \sinh z \equiv \frac{\mathrm{e}^z - \mathrm{e}^{-z}}{2}$$

The remaining hyperbolic functions are

$$\tanh z \equiv \frac{\sinh z}{\cosh z}, \coth z \equiv \frac{1}{\tanh z}$$

$$\operatorname{sech} z \equiv \frac{1}{\cosh z}, \operatorname{csch} z \equiv \frac{1}{\sinh z}$$

The circular functions with complex arguments are defined

$$\cos z = \frac{\mathrm{e}^{\mathrm{i}\,z} + \mathrm{e}^{-\mathrm{i}\,z}}{2}, \sin z = \frac{\mathrm{e}^{\mathrm{i}\,z} - \mathrm{e}^{-\mathrm{i}\,z}}{2},$$

$$\tan z = \frac{\sin z}{\cos z}$$

and satisfy

$$\sin(-z) = -\sin z, \cos(-z) = \cos z$$

$$\sin(\mathrm{i}\,z) = \mathrm{i}\sinh z, \cos(\mathrm{i}\,z) = -\cosh z$$

All trigonometric identities for real, circular functions with real arguments can be extended without change to complex functions of complex arguments. For example,

$$\sin^2 z + \cos^2 z = 1,$$

$$\sin(z_1 + z_2) = \sin z_1 \cos z_2 + \cos z_1 \sin z_2$$

The same is true of hyperbolic functions. The absolute boundaries of $\sin z$ and $\cos z$ are not bounded for all z.

Trigonometric identities can be defined by using

$$e^{i\theta} = \cos\theta + \mathrm{i}\sin\theta$$

For example,

$$\mathrm{e}^{\mathrm{i}(\alpha+\beta)} = \cos(\alpha+\beta) + \mathrm{i}\sin(\alpha+\beta)$$
$$= \mathrm{e}^{\mathrm{i}\alpha}\mathrm{e}^{\mathrm{i}\beta} = (\cos\alpha + \mathrm{i}\sin\alpha)(\cos\beta + \mathrm{i}\sin\beta)$$
$$= \cos\alpha\cos\beta - \sin\alpha\sin\beta + \mathrm{i}(\cos\alpha\sin\beta + \cos\beta\sin\alpha)$$

Equating real and imaginary parts gives

$$\cos(\alpha+\beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta$$

$$\sin(\alpha+\beta) = \cos\alpha\sin\beta + \cos\beta\sin\alpha$$

The logarithm is defined as

$$\ln z = \ln|z| + \mathrm{i}\arg z$$

and the various determinations differ by multiples of 2π i. Then,

$$\mathrm{e}^{\ln z} = z$$

$$\ln(\mathrm{e}^z) - z \equiv 0 \pmod{2\pi\mathrm{i}}$$

Also,

$$\ln(z_1 z_2) - \ln z_1 - \ln z_2 \equiv 0 \pmod{2\pi\mathrm{i}}$$

is always true, but

$$\ln(z_1 z_2) = \ln z_1 + \ln z_2$$

holds only for some determinations of the logarithms. The principal determination of the argument can be defined as $-\pi < \arg \leq \pi$.

3.3. Analytic Functions of a Complex Variable

Let $f(z)$ be a single-valued continuous function of z in a domain D. The function $f(z)$ is differentiable at the point z_0 in D if

$$\lim_{h\to 0}\frac{f(z_0+h)-f(z_0)}{h}$$

exists as a finite (complex) number and is independent of the direction in which h tends to zero. The limit is called the derivative, $f'(z_0)$. The derivative now can be calculated with h approaching zero in the complex plane, i.e., anywhere in a circular region about z_0. The function $f(z)$ is differentiable in D if it is differentiable at all points of D; then $f(z)$ is said to be an analytic function of z in D. Also, $f(z)$ is analytic at z_0 if it is analytic in some ε neighborhood of z_0. The word analytic is sometimes replaced by holomorphic or regular.

The *Cauchy – Riemann equations* can be used to decide if a function is analytic. Set

$$f(z) = f(x+\mathrm{i}\,y) = u(x,y)+\mathrm{i}\,v(x,y)$$

Theorem [30, p. 51]. Suppose that $f(z)$ is defined and continuous in some neighborhood of $z = z_0$. A necessary condition for the existence of $f'(z_0)$ is that $u(x, y)$ and $v(x, y)$ have first-order partials and that the Cauchy – Riemann conditions (see below) hold.

$$\frac{\partial u}{\partial x}=\frac{\partial v}{\partial y}\ \text{and}\ \frac{\partial u}{\partial y}=-\frac{\partial v}{\partial x}\ \text{at}\ z_0$$

Theorem [30, p. 61]. The function $f(z)$ is analytic in a domain D if and only if u and v are continuously differentiable and satisfy the Cauchy – Riemann conditions there.

If $f_1(z)$ and $f_2(z)$ are analytic in domain D, then $\alpha_1 f_1(z) + \alpha_2 f_2(z)$ is analytic in D for any (complex) constants α_1, α_2.

$f_1(z)+f_2(z)$ is analytic in D

$f_1(z)/f_2(z)$ is analytic in D except where $f_2(z)=0$

An analytic function of an analytic function is analytic. If $f(z)$ is analytic, $f'(z)\neq 0$ in D, $f(z_1)\neq f(z_2)$ for $z_1\neq z_2$, then the inverse function $g(w)$ is also analytic and

$$g'(w)=\frac{1}{f'(z)}\ \text{where}\ w=f(z),$$

$$g(w)=g[f(z)]=z$$

Analyticity implies continuity but the converse is not true: $z^* = x - \mathrm{i}\,y$ is continuous but, because the Cauchy – Riemann conditions are not satisfied, it is not analytic. An entire function is one that is analytic for all finite values of z. Every polynomial is an entire function. Because the polynomials are analytic, a ratio of polynomials is analytic except when the denominator vanishes. The function $f(z) = |z^2|$ is continuous for all z but satisfies the Cauchy – Riemann conditions only at $z = 0$. Hence, $f'(z)$ exists only at the origin, and $|z|^2$ is nowhere analytic. The function $f(z) = 1/z$ is analytic except at $z = 0$. Its derivative is $-1/z^2$, where $z \neq 0$. If $\ln z = \ln|z| + \mathrm{i}\arg z$ in the cut domain $-\pi < \arg z \leq \pi$, then $f(z) = 1/\ln z$ is analytic in the same cut domain, except at $z = 1$, where $\log z = 0$. Because e^z is analytic and $\pm\,\mathrm{i}\,z$ are analytic, $\mathrm{e}^{\pm \mathrm{i}z}$ is analytic and linear combinations are analytic. Thus, the sine and cosine and hyperbolic sine and cosine are analytic. The other functions are analytic except when the denominator vanishes.

The derivatives of the elementary functions are

$$\frac{\mathrm{d}}{\mathrm{d}z}\mathrm{e}^z=\mathrm{e}^z,\ \frac{\mathrm{d}}{\mathrm{d}z}z^n=n\,z^{n-1}$$

$$\frac{\mathrm{d}}{\mathrm{d}z}(\ln z)=\frac{1}{z},\ \frac{\mathrm{d}}{\mathrm{d}z}\sin z=\cos z,$$

$$\frac{\mathrm{d}}{\mathrm{d}z}\cos z=-\sin z$$

In addition,

$$\frac{\mathrm{d}}{\mathrm{d}z}(f\,g)=f\frac{\mathrm{d}g}{\mathrm{d}z}+g\frac{\mathrm{d}f}{\mathrm{d}z}$$

$$\frac{\mathrm{d}}{\mathrm{d}z}f[g(z)]=\frac{\mathrm{d}f}{\mathrm{d}g}\frac{\mathrm{d}g}{\mathrm{d}z}$$

$$\frac{\mathrm{d}}{\mathrm{d}z}\sin w=\cos w\frac{\mathrm{d}w}{\mathrm{d}z},\ \frac{\mathrm{d}}{\mathrm{d}z}\cos w=-\sin w\frac{\mathrm{d}w}{\mathrm{d}z}$$

Define $z^a = \mathrm{e}^{a\,\ln z}$ for complex constant a. If the determination is $-\pi < \arg z \leq \pi$, then z^a is analytic on the complex plane with a cut on the negative real axis. If a is an integer n, then $\mathrm{e}^{2\pi i n} = 1$ and z^n has the same limits approaching the cut from either side. The function can be made continuous across the cut and the function is analytic there, too. If $a = 1/n$ where n is an integer, then

$$z^{1/n}=\mathrm{e}^{(\ln z)/n}=|z|^{1/n}\mathrm{e}^{\mathrm{i}(\arg z)/n}$$

So $w = z^{1/n}$ has n values, depending on the choice of argument.

Laplace Equation. If $f(z)$ is analytic, where

$$f(z) = u(x,y) + \mathrm{i}\, v(x,y)$$

the Cauchy – Riemann equations are satisfied. Differentiating the Cauchy – Riemann equations gives the Laplace equation:

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial^2 v}{\partial x \partial y} = \frac{\partial^2 v}{\partial y \partial x} = -\frac{\partial^2 u}{\partial y^2} \text{ or}$$

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

Similarly,

$$\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} = 0$$

Thus, general solutions to the Laplace equation can be obtained from analytic functions [30, p. 60]. For example,

$$\ln \frac{1}{|z - z_0|}$$

is analytic so that a solution to the Laplace equation is

$$\ln \left[(x-a)^2 + (y-b)^2 \right]^{-1/2}$$

A solution to the Laplace equation is called a harmonic function. A function is harmonic if, and only if, it is the real part of an analytic function. The imaginary part is also harmonic. Given any harmonic function u, a conjugate harmonic function v can be constructed such that $f = u + \mathrm{i}\, v$ is locally analytic [30, p. 290].

Maximum Principle. If $f(z)$ is analytic in a domain D and continuous in the set consisting of D and its boundary C, and if $|f(z)| \leq M$ on C, then $|f(z)| < M$ in D unless $f(z)$ is a constant [30, p. 134].

3.4. Integration in the Complex Plane

Let C be a rectifiable curve in the complex plane

$$C: z = z(t), 0 \leq t \leq 1$$

where $z(t)$ is a continuous function of bounded variation; C is oriented such that $z_1 = z(t_1)$ precedes the point $z_2 = z(t_2)$ on C if and only if $t_1 < t_2$. Define

$$\int_C f(z)\,\mathrm{d}z = \int_0^1 f[z(t)]\,\mathrm{d}z(t)$$

The integral is linear with respect to the integrand:

$$\int_C [\alpha_1 f_1(z) + \alpha_2 f_2(z)]\,\mathrm{d}z$$
$$= \alpha_1 \int_C f_1(z)\,\mathrm{d}z + \alpha_2 \int_C f_2(z)\,\mathrm{d}z$$

The integral is additive with respect to the path. Let curve C_2 begin where curve C_1 ends and $C_1 + C_2$ be the path of C_1 followed by C_2. Then,

$$\int_{C_1 + C_2} f(z)\,\mathrm{d}z = \int_{C_1} f(z)\,\mathrm{d}z + \int_{C_2} f(z)\,\mathrm{d}z$$

Reversing the orientation of the path replaces the integral by its negative:

$$\int_{-C} f(z)\,\mathrm{d}z = -\int_C f(z)\,\mathrm{d}z$$

If the path of integration consists of a finite number of arcs along which $z(t)$ has a continuous derivative, then

$$\int_C f(z)\,\mathrm{d}z = \int_0^1 f[z(t)]\, z'(t)\,\mathrm{d}t$$

Also if $s(t)$ is the arc length on C and $l(C)$ is the length of C

$$\left| \int_C f(z)\,\mathrm{d}z \right| \leq \max_{z \in C} |f(z)|\, l(C)$$

and

$$\left| \int_C f(z)\,\mathrm{d}z \right| \leq \int_C |f(z)|\,|\mathrm{d}z| = \int_0^1 |f[z(t)]|\,\mathrm{d}s(t)$$

Cauchy's Theorem [25, 30, p. 111]. Suppose $f(z)$ is an analytic function in a domain D and C is a simple, closed, rectifiable curve in D such that $f(z)$ is analytic inside and on C. Then

$$\oint_C f(z)\,\mathrm{d}z = 0 \tag{4}$$

If D is simply connected, then Equation 4 holds for every simple, closed, rectifiable curve C in D. If D is simply connected and if a and b are any two points in D, then

$$\int_a^b f(z)\,\mathrm{d}z$$

is independent of the rectifiable path joining a and b in D.

Cauchy's Integral. If C is a closed contour such that $f(z)$ is analytic inside and on C, z_0 is a point inside C, and z traverses C in the counterclockwise direction,

$$f(z_0) = \frac{1}{2\pi i}\oint_C \frac{f(z)}{z-z_0}\,dz$$

$$f'(z_0) = \frac{1}{2\pi i}\oint_C \frac{f(z)}{(z-z_0)^2}\,dz$$

Under further restriction on the domain [30, p. 127],

$$f^{(m)}(z_0) = \frac{m!}{2\pi i}\oint_C \frac{f(z)}{(z-z_0)^{m+1}}\,dz$$

Power Series. If $f(z)$ is analytic interior to a circle $|z - z_0| < r_0$, then at each point inside the circle the series

$$f(z) = f(z_0) + \sum_{n=1}^{\infty} \frac{f^{(n)}(z_0)}{n!}(z-z_0)^n$$

converges to $f(z)$. This result follows from Cauchy's integral. As an example, e^z is an entire function (analytic everywhere) so that the MacLaurin series

$$e^z = 1 + \sum_{n=1}^{\infty} \frac{z^n}{n!}$$

represents the function for all z.

Another result of Cauchy's integral formula is that if $f(z)$ is analytic in an annulus R, $r_1 < |z - z_0| < r_2$, it is represented in R by the Laurent series

$$f(z) = \sum_{n=-\infty}^{\infty} A_n (z-z_0)^n, r_1 < |z-z_0| \le r_2$$

where

$$A_n = \frac{1}{2\pi i}\int_C \frac{f(z)}{(z-z_0)^{n+1}}\,dz,$$

$$n = 0, \pm 1, \pm 2, \ldots,$$

and C is a closed curve counterclockwise in R.

Singular Points and Residues [33, p. 159, 30, p. 180]. If a function in analytic in every neighborhood of z_0, but not at z_0 itself, then z_0 is called an isolated singular point of the function. About an isolated singular point, the function can be represented by a Laurent series.

$$f(z) = \cdots + \frac{A_{-2}}{(z-z_0)^2} + \frac{A_{-1}}{z-z_0} + A_0 + A_1(z-z_0) + \cdots 0 < |z-z_0| \le r_0 \tag{5}$$

In particular,

$$A_{-1} = \frac{1}{2\pi i}\oint_C f(z)\,dz$$

where the curve C is a closed, counterclockwise curve containing z_0 and is within the neighborhood where $f(z)$ is analytic. The complex number A_{-1} is the residue of $f(z)$ at the isolated singular point z_0; $2\pi i A_{-1}$ is the value of the integral in the positive direction around a path containing no other singular points.

If $f(z)$ is defined and analytic in the exterior $|z - z_0| > R$ of a circle, and if

$$v(\zeta) = f\left(z_0 + \frac{1}{\zeta}\right) \text{ obtained by } \zeta = \frac{1}{z-z_0}$$

has a removable singularity at $\zeta = 0$, $f(z)$ is analytic at infinity. It can then be represented by a Laurent series with nonpositive powers of $z - z_0$.

If C is a closed curve within which and on which $f(z)$ is analytic except for a finite number of singular points $z_1, z_2, \ldots, z_n$ interior to the region bounded by C, then the residue theorem states

$$\oint_C f(z)\,dz = 2\pi i(\varrho_1 + \varrho_2 + \cdots \varrho_n)$$

where ϱ_n denotes the residue of $f(z)$ at z_n.

The series of negative powers in Equation 5 is called the principal part of $f(z)$. If the principal part has an infinite number of nonvanishing terms, the point z_0 is an essential singularity. If $A_{-m} \neq 0$, $A_{-n} = 0$ for all $m < n$, then z_0 is called a pole of order m. It is a simple pole if $m = 1$. In such a case,

$$f(z) = \frac{A_{-1}}{z-z_0} + \sum_{n=0}^{\infty} A_n (z-z_0)^n$$

If a function is not analytic at z_0 but can be made so by assigning a suitable value, then z_0 is a removable singular point.

When $f(z)$ has a pole of order m at z_0,

$$\varphi(z) = (z-z_0)^m f(z), 0 < |z-z_0| < r_0$$

has a removable singularity at z_0. If $\varphi(z_0) = A_{-m}$, then $\varphi(z)$ is analytic at z_0. For a simple pole,

$$A_{-1} = \varphi(z_0) = \lim_{z \to z_0} (z-z_0)\, f(z)$$

Also $|f(z)| \to \infty$ as $z \to z_0$ when z_0 is a pole. Let the function $p(z)$ and $q(z)$ be analytic at z_0, where $p(z_0) \neq 0$. Then

$$f(z) = \frac{p(z)}{q(z)}$$

has a simple pole at z_0 if, and only if, $q(z_0) = 0$ and $q'(z_0) \neq 0$. The residue of $f(z)$ at the simple pole is

$$A_{-1} = \frac{p(z_0)}{q'(z_0)}$$

If $q^{(i-1)}(z_0) = 0$, i = 1, . . . , m, then z_0 is a pole of $f(z)$ of order m.

Branch [33, p. 163]. A branch of a multiple-valued function $f(z)$ is a single-valued function that is analytic in some region and whose value at each point there coincides with the value of $f(z)$ at the point. A branch cut is a boundary that is needed to define the branch in the greatest possible region. The function $f(z)$ is singular along a branch cut, and the singularity is not isolated. For example,

$$z^{1/2} = f_1(z) = \sqrt{r}\left(\cos\tfrac{\theta}{2} + i\sin\tfrac{\theta}{2}\right)$$

$$-\pi < \theta < \pi,\ r > 0$$

is double valued along the negative real axis. The function tends to $\sqrt{r}i$ when $\theta \to \pi$ and to $-\sqrt{r}i$ when $\theta \to -\pi$; the function has no limit as $z \to -r$ ($r > 0$). The ray $\theta = \pi$ is a branch cut.

Analytic Continuation [33, p. 165]. If $f_1(z)$ is analytic in a domain D_1 and domain D contains D_1, then an analytic function $f(z)$ may exist that equals $f_1(z)$ in D_1. This function is the analytic continuation of $f_1(z)$ onto D, and it is unique. For example,

$$f_1(z) = \sum_{n=0}^{\infty} z^n,\ |z| < 1$$

is analytic in the domain $D_1 : |z| < 1$. The series diverges for other z. Yet the function is the MacLaurin series in the domain

$$f_1(z) = \frac{1}{1-z},\ |z| < 1$$

Thus,

$$f_1(z) = \frac{1}{1-z}$$

is the analytic continuation onto the entire z plane except for $z = 1$.

An extension of the Cauchy integral formula is useful with Laplace transforms. Let the curve C be a straight line parallel to the imaginary axis and z_0 be any point to the right of that (see Fig. 9). A function $f(z)$ is of order z^k as $|z| \to \infty$ if positive numbers M and r_0 exist such that

$$|z^{-k} f(z)| < M \text{ when } |z| > r_0, \text{ i.e.,}$$

$$|f(z)| < M|z|^k \text{ for } |z| \text{ sufficiently large}$$

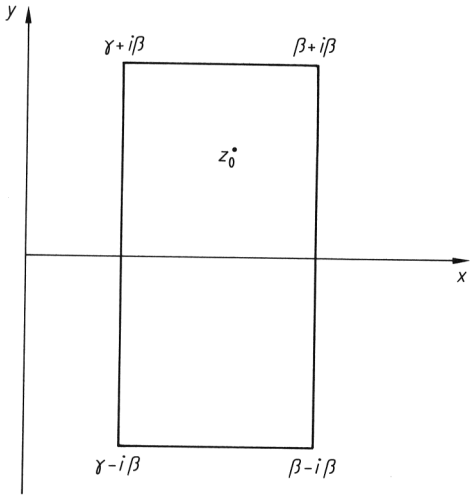

Figure 9. Integration in the complex plane

Theorem [33, p. 167]. Let $f(z)$ be analytic when $R(z) \geq \gamma$ and $O(z^{-k})$ as $|z| \to \infty$ in that half-plane, where γ and k are real constants and $k > 0$. Then for any z_0 such that $R(z_0) > \gamma$

$$f(z_0) = -\frac{1}{2\pi i} \lim_{\beta \to \infty} \int_{\gamma - i\beta}^{\gamma + i\beta} \frac{f(z)}{z - z_0} dz,$$

i.e., integration takes place along the line $x = \gamma$.

3.5. Other Results

Theorem [32, p. 84]. Let $P(z)$ be a polynomial of degree n having the zeroes $z_1, z_2, \ldots, z_n$ and let π be the least convex polygon containing the zeroes. Then $P'(z)$ cannot vanish anywhere in the exterior of π.

If a polynomial has real coefficients, the roots are either real or form pairs of complex conjugate numbers.

The radius of convergence *R* of the Taylor series of $f\ (z)$ about z_0 is equal to the distance from z_0 to the nearest singularity of $f\ (z)$.

Conformal Mapping. Let $u\ (x, y)$ be a harmonic function. Introduce the coordinate transformation

$$x = \hat{x}\,(\xi, \eta)\,,\ y = \hat{y}\,(\xi, \eta)$$

It is desired that

$$U\ (\xi, \eta) = u\,[\hat{x}\,(\xi, \eta)\,, \hat{y}\,(\xi, \eta)]$$

be a harmonic function of ξ and η.

Theorem [30, p. 284]. The transformation

$$z = f\,(\zeta) \tag{6}$$

takes all harmonic functions of x and y into harmonic functions of ξ and η if and only if either $f\ (\zeta)$ or $f^*\,(\zeta)$ is an analytic function of $\zeta = \xi + \mathrm{i}\,\eta$.

Equation 6 is a restriction on the transformation which ensures that

$$\text{if } \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \text{ then } \frac{\partial^2 U}{\partial \zeta^2} + \frac{\partial^2 U}{\partial \eta^2} = 0$$

Such a mapping with $f\ (\zeta)$ analytic and $f'\ (\zeta) \neq 0$ is a conformal mapping.

If Laplace's equation is to be solved in the region exterior to a closed curve, then the point at infinity is in the domain D. For flow in a long channel (governed by the Laplace equation) the inlet and outlet are at infinity. In both cases the transformation

$$\zeta = \frac{a\,z + b}{z - z_0}$$

takes z_0 into infinity and hence maps D into a bounded domain D*.

4. Integral Transforms [34 – 39]

4.1. Fourier Transforms

Fourier Series [40]**.** Let $f\ (x)$ be a function that is periodic on $-\pi < x < \pi$. It can be expanded in a Fourier series

$$f\ (x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos n\,x + b_n \sin n\,x)$$

where

$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} f\ (x)\,\mathrm{d}x,\ a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f\ (x) \cos n\,x\,\mathrm{d}x,$$

$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f\ (x) \sin n\,x\,\mathrm{d}x$$

The values $\{a_n\}$ and $\{b_n\}$ are called the finite cosine and sine transform of *f*, respectively. Because

$$\cos n\,x = \tfrac{1}{2}\left(\mathrm{e}^{\mathrm{i}nx} + \mathrm{e}^{-\mathrm{i}nx}\right)$$

$$\text{and } \sin n\,x = \tfrac{1}{2\mathrm{i}}\left(\mathrm{e}^{\mathrm{i}nx} - \mathrm{e}^{-\mathrm{i}nx}\right)$$

the Fourier series can be written as

$$f\ (x) = \sum_{n=-\infty}^{\infty} c_n \mathrm{e}^{-\mathrm{i}nx}$$

where

$$c_n = \begin{cases} \frac{1}{2}\,(a_n + \mathrm{i}\,b_n) & \text{for} \quad n \geq 0 \\ \frac{1}{2}\,(a_{-n} - \mathrm{i}\,b_{-n}) & \text{for} \quad n < 0 \end{cases}$$

and

$$c_n = \frac{1}{2\,\pi}\int_{-\pi}^{\pi} f\ (x)\,\mathrm{e}^{\mathrm{i}nx}\,\mathrm{d}x$$

If f is real

$$c_{-n} = c_n^*.$$

If f is continuous and piecewise continuously differentiable

$$f'\,(x) = \sum_{-\infty}^{\infty} (-\mathrm{i}\,n)\,c_n \mathrm{e}^{-\mathrm{i}nx}$$

If f is twice continuously differentiable

$$f''\ (x) = \sum_{-\infty}^{\infty} \left(-n^2\right) c_n \mathrm{e}^{-\mathrm{i}nx}$$

Inversion. The Fourier series can be used to solve linear partial differential equations with constant coefficients. For example, in the problem

$$\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2}$$

$$T\ (x, 0) = f\ (x)$$

$$T\ (-\pi, t) = T\ (\pi, t)$$

Let

$$T = \sum_{-\infty}^{\infty} c_n\,(t)\,\mathrm{e}^{-inx}$$

Then,

$$\sum_{-\infty}^{\infty} \frac{\mathrm{d}c_n}{\mathrm{d}t}\,\mathrm{e}^{-inx} = \sum_{-\infty}^{\infty} c_n\,(t)\,\left(-n^2\right)\mathrm{e}^{-inx}$$

Thus, $c_n\,(t)$ satisfies

$$\frac{\mathrm{d}c_n}{\mathrm{d}t} = -n^2\,c_n, \text{ or } c_n = c_n\,(0)\,\mathrm{e}^{-n^2 t}$$

Let $c_n\,(0)$ be the Fourier coefficients of the initial conditions:

$$f\,(x) = \sum_{-\infty}^{\infty} c_n\,(0)\,\mathrm{e}^{-inx}$$

The formal solution to the problem is

$$T = \sum_{-\infty}^{\infty} c_n\,(0)\,\mathrm{e}^{-n^2 t}\mathrm{e}^{-inx}$$

Fourier Transform [40]**.** When the function $f\,(x)$ is defined on the entire real line, the Fourier transform is defined as

$$\boldsymbol{F}\,[f] \equiv \hat{f}\,(\omega) = \int_{-\infty}^{\infty} f\,(x)\,\mathrm{e}^{\mathrm{i}\omega x}\mathrm{d}x$$

This integral converges if

$$\int_{-\infty}^{\infty} |f\,(x)|\,\mathrm{d}x$$

does. The inverse transformation is

$$f\,(x) = \frac{1}{2\,\pi}\int_{-\infty}^{\infty} \hat{f}\,(\omega)\,\mathrm{e}^{-\mathrm{i}\omega x}\mathrm{d}\omega$$

If $f\,(x)$ is continuous and piecewise continuously differentiable,

$$\int_{-\infty}^{\infty} f\,(x)\,\mathrm{e}^{\mathrm{i}\omega x}\mathrm{d}x$$

converges for each ω, and

$$\lim_{x\to\pm\infty} f\,(x) = 0$$

then

$$\boldsymbol{F}\left[\frac{\mathrm{d}\,f}{\mathrm{d}x}\right] = -\mathrm{i}\omega\boldsymbol{F}\,[f]$$

If f is real $F\,[f\,(-\omega)] = F\,[f\,(\omega)^*]$. The real part is an even function of ω and the imaginary part is an odd function of ω.

A function $f\,(x)$ is *absolutely integrable* if the improper integral

$$\int_{-\infty}^{\infty} |f\,(x)|\,\mathrm{d}x$$

has a finite value. Then the improper integral

$$\int_{-\infty}^{\infty} f\,(x)\,\mathrm{d}x$$

converges. The function is *square integrable* if

$$\int_{-\infty}^{\infty} |f\,(x)|^2\mathrm{d}x$$

has a finite value. If $f\,(x)$ and $g\,(x)$ are square integrable, the product $f\,(x)\,g\,(x)$ is absolutely integrable and satisfies the Schwarz inequality:

$$\left|\int_{-\infty}^{\infty} f\,(x)\,g\,(x)\,\mathrm{d}x\right|^2 \leq \int_{-\infty}^{\infty} |f\,(x)|^2\mathrm{d}x \int_{-\infty}^{\infty} |g\,(x)|^2\mathrm{d}x$$

The triangle inequality is also satisfied:

$$\left\{\int_{-\infty}^{\infty} |f+g|^2\mathrm{d}x\right\}^{1/2} \leq \left\{\int_{-\infty}^{\infty} |f|^2\mathrm{d}x\right\}^{1/2}\left\{\int_{-\infty}^{\infty} |g|^2\mathrm{d}x\right\}^{1/2}$$

A sequence of square integrable functions $f_n\,(x)$ converges in the mean to a square integrable function $f\,(x)$ if

$$\lim_{n\to\infty}\int_{-\infty}^{\infty} |f\,(x) - f_n\,(x)|^2\mathrm{d}x = 0$$

The sequence also satisfies the Cauchy criterion

$$\lim_{\substack{n\to\infty\\ m\to\infty}}\int_{-\infty}^{\infty} |f_n - f_m|^2\mathrm{d}x = 0$$

Theorem [40, p. 307]. If a sequence of square integrable functions $f_n\,(x)$ converges to a function $f\,(x)$ uniformly on every finite interval $a \leq x \leq b$, and if it satisfies Cauchy's criterion, then $f\,(x)$ is square integrable and $f_n\,(x)$ converges to $f\,(x)$ in the mean.

Theorem (Riesz – Fischer) [40, p. 308]. To every sequence of square integrable functions $f_n(x)$ that satisfy Cauchy's criterion, there corresponds a square integrable function $f(x)$ such that $f_n(x)$ converges to $f(x)$ in the mean. Thus, the limit in the mean of a sequence of functions is defined to within a null function.

Square integrable functions satisfy the Parseval equation.

$$\int_{-\infty}^{\infty} \left|\hat{f}(\omega)\right|^2 \mathrm{d}\omega = 2\pi \int_{-\infty}^{\infty} |f(x)|^2 \mathrm{d}x$$

This is also the total power in a signal, which can be computed in either the time or the frequency domain. Also

$$\int_{-\infty}^{\infty} \hat{f}(\omega)\, \hat{g}*(\omega)\, \mathrm{d}\omega = 2\pi \int_{-\infty}^{\infty} f(x)\, g*(x)\, \mathrm{d}x$$

Fourier transforms can be used to solve differential equations too. Then it is necessary to find the inverse transformation. If $f(x)$ is square integrable, the Fourier transform of its Fourier transform is $2\pi f(-x)$, or

$$f(x) = \boldsymbol{F}\left[\hat{f}(\omega)\right] = \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{f}(\omega)\, \mathrm{e}^{-\mathrm{i}\omega x}\mathrm{d}\omega$$

$$= \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x)\, \mathrm{e}^{\mathrm{i}\omega x}\mathrm{d}x e^{-\mathrm{i}\omega x}\mathrm{d}\omega$$

$$f(x) = \frac{1}{2\pi}\boldsymbol{F}[\boldsymbol{F}f(-x)] \text{ or } f(-x) = \frac{1}{2\pi}\boldsymbol{F}[\boldsymbol{F}[f]]$$

Properties of Fourier Transforms [40, p. 324], [15].

$$\boldsymbol{F}\left[\frac{\mathrm{d}f}{\mathrm{d}x}\right] = -\mathrm{i}\,\omega\,\boldsymbol{F}[f] = \mathrm{i}\,\omega\hat{f}$$

$$\boldsymbol{F}[\mathrm{i}\,x\,f(x)] = \frac{\mathrm{d}}{\mathrm{d}\omega}\boldsymbol{F}[f] = \frac{\mathrm{d}}{\mathrm{d}\omega}\hat{f}$$

$$\boldsymbol{F}[f(a\,x-b)] = \frac{1}{|a|}\mathrm{e}^{\mathrm{i}\omega b/a}\hat{f}\left(\frac{\omega}{a}\right)$$

$$\boldsymbol{F}\left[\mathrm{e}^{\mathrm{i}cx}f(x)\right] = \hat{f}(\omega+c)$$

$$\boldsymbol{F}[\cos\omega_0 x\, f(x)] = \frac{1}{2}\left[\hat{f}(\omega+\omega_0)+\hat{f}(\omega-\omega_0)\right]$$

$$\boldsymbol{F}[\sin\omega_0 x\, f(x)] = \frac{1}{2\mathrm{i}}\left[\hat{f}(\omega+\omega_0)-\hat{f}(\omega-\omega_0)\right]$$

$$\boldsymbol{F}\left[\mathrm{e}^{-\mathrm{i}\omega_0 x}f(x)\right] = \hat{f}(\omega-\omega_0)$$

If $f(x)$ is real, then $f(-\omega) = \hat{f}*(\omega)$. If $f(x)$ is imaginary, then $\hat{f}(-\omega) = -\hat{f}*(\omega)$. If $f(x)$ is even, then $\hat{f}(\omega)$ is even. If $f(x)$ is odd, then $\hat{f}(\omega)$ is odd. If $f(x)$ is real and even, then $\hat{f}(\omega)$ is real and even. If $f(x)$ is real and odd, then $\hat{f}(\omega)$ is imaginary and odd. If $f(x)$ is imaginary and even, then $\hat{f}(\omega)$ is imaginary and even. If $f(x)$ is imaginary and odd, then $\hat{f}(\omega)$ is real and odd.

Convolution [40, p. 326].

$$f*h(x_0) \equiv \int_{-\infty}^{\infty} f(x_0-x)\; h(x)\; \mathrm{d}x$$

$$= \frac{1}{2\pi}\int_{-\infty}^{\infty} \mathrm{e}^{\mathrm{i}\omega x_0}\hat{f}(-\omega)\,\hat{h}(\omega)\,\mathrm{d}\omega$$

Theorem. The product

$$\hat{f}(\omega)\,\hat{h}(\omega)$$

is the Fourier transform of the convolution product $f*h$. The convolution permits finding inverse transformations when solving differential equations. To solve

$$\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2}$$

$$T(x,0) = f(x), -\infty<x<\infty$$

T bounded

take the Fourier transform

$$\frac{\mathrm{d}\hat{T}}{\mathrm{d}t}+\omega^2\hat{T} = 0$$

$$\hat{T}(\omega,0) = \hat{f}(\omega)$$

The solution is

$$\hat{T}(\omega,t) = \hat{f}(\omega)\,\mathrm{e}^{-\omega^2 t}$$

The inverse transformation is

$$T(x,t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \mathrm{e}^{-\mathrm{i}\omega x}\,\hat{f}(\omega)\,\mathrm{e}^{-\omega^2 t}\mathrm{d}\omega$$

Because

$$\mathrm{e}^{-\omega^2 t} = \boldsymbol{F}\left[\frac{1}{\sqrt{4\pi t}}\,\mathrm{e}^{-x^2/4t}\right]$$

the convolution integral can be used to write the solution as

$$T(x,t) = \frac{1}{\sqrt{4\pi t}}\int_{-\infty}^{\infty} f(y)\,\mathrm{e}^{-(x-y)^2/4t}\mathrm{d}y$$

Finite Fourier Sine and Cosine Transform [41]. In analogy with finite Fourier transforms (on $-\pi$ to π) and Fourier transforms (on $-\infty$ to $+\infty$), finite Fourier sine and cosine transforms (0 to π) and Fourier sine and cosine transforms (on 0 to $+\infty$) can be defined.

The finite Fourier sine and cosine transforms are

$$f_s(n) = \boldsymbol{F}_s^n[f] = \frac{2}{\pi}\int_0^\pi f(x)\sin n\,x\,\mathrm{d}x,$$

$$n = 1, 2, \ldots,$$

$$f_c(n) = \boldsymbol{F}_c^n[f] = \frac{2}{\pi}\int_0^\pi f(x)\cos n\,x\,\mathrm{d}x$$

$$n = 0, 1, 2, \ldots$$

$$f(x) = \sum_{n=1}^{\infty} f_s(n)\sin n\,x,$$

$$f(x) = \tfrac{1}{2} f_c(0) + \sum_{n=1}^{\infty} f_c(n)\cos n\,x$$

They obey the operational properties

$$\boldsymbol{F}_s^n\left[\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}\right] = -n^2\,\boldsymbol{F}_s^n[f]$$

$$+\frac{2n}{\pi}[f(0) - (-1)^n f(\pi)]$$

f, f' are continuous, f'' is piecewise continuous on $0 \le x \le \pi$.

$$\left.\begin{aligned} &f_s(n)\cos nk \\ &= \boldsymbol{F}_s^n\left[\tfrac{1}{2} f_1(x-k) + \tfrac{1}{2} f_1(x+k)\right] \\ &f_s(n)(-1)^{n+1} = \boldsymbol{F}_s^n[f(\pi - x)] \end{aligned}\right\} f \text{ is piecewise continuous on } 0 \le x \le \pi$$

and f_1 is the extension of f, k is a constant

$$\left.\begin{aligned} &f_1(x) = f(x) \quad 0 < x < \pi \\ &f_1(-x) = -f_1(x) \\ &f_1(x + 2\pi) = f_1(x) \end{aligned}\right\} -\infty < x < \infty$$

Also,

$$\boldsymbol{F}_c^n\left[\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}\right] = -n^2\boldsymbol{F}_c^n[f] - \frac{2}{\pi}\frac{\mathrm{d}f}{\mathrm{d}x}(0)$$

$$+(-1)^n\frac{2}{\pi}\frac{\mathrm{d}f}{\mathrm{d}x}(\pi)$$

$$f_c(n)\cos nk = \boldsymbol{F}_c^n\left[\tfrac{1}{2} f_2(x-k) + \tfrac{1}{2} f_2(x+k)\right]$$

$$f_c(n)(-1)^n = \boldsymbol{F}_c^n[f(\pi - x)]$$

and f_2 is the extension of f.

$$\left.\begin{aligned} &f_2(x) = f(x) \quad 0 < x < \pi \\ &f_2(-x) = f_2(x) \\ &f_2(x + 2\pi) = f_2(x) \end{aligned}\right\} -\infty < x < \infty$$

Table 3. Finite sine transforms [41]

$f_s(n) =$	$F(x)$ $(0<x<\pi)$
$\int_0^\pi F(x)\sin n\,x\,\mathrm{d}x$ $(n = 1,2,\ldots)$	
$(-1)^{n+1} f_s(n)$	$F(\pi - x)$
$\frac{1}{n}$	$\frac{(\pi - x)}{\pi}$
$\frac{(-1)^{n+1}}{n}$	$\frac{x}{\pi}$
$\frac{1-(-1)^n}{n}$	1
$\frac{\pi}{n^2}\sin n\,c$ $(0<c<\pi)$	$\begin{cases} (\pi - c)\,x & (x \le c) \\ c\,(\pi - x) & (x \ge c) \end{cases}$
$\frac{\pi}{n}\cos n\,c$ $(0<c<\pi)$	$\begin{cases} -x & (x<c) \\ \pi - x & (x) \end{cases}$
$\frac{\pi^2(-1)^{n-1}}{n} - \frac{2[1-(-1)^n]}{n^3}$	x^2
$\pi(-1)^n\left(\frac{6}{n^3} - \frac{\pi^2}{n}\right)$	x^3
$\frac{n}{n^2+c^2}[1-(-1)^n e^{c\pi}]$	e^{cx}
$\frac{n}{n^2+c^2}$	$\frac{\sinh c(\pi - x)}{\sinh c\,\pi}$
$\frac{n}{n^2-k^2}$ $(\lvert k\rvert \ne 0, 1, 2, \ldots)$	$\frac{\sin k(\pi - x)}{\sin k\,\pi}$
0 $(n \ne m)$; $f_s(m) = \frac{\pi}{2}$	$\sin m\,x$ $(m = 1,2,\ldots)$
$\frac{n}{n^2-k^2}[1-(-1)^n\cos k\,x]$	$\cos k\,x$ $(\lvert k\rvert \ne 1,2,\ldots)$
$\frac{n}{n^2-m^2}[1-(-1)^{n+m}]$, $(n \ne m)$; $f_s(m) = 0$	$\cos m\,x$ $(m = 1,2,\ldots)$

The material is reproduced with permission of McGrawHill, Inc.

Also,

$$\boldsymbol{F}_s^n\left[\frac{\mathrm{d}f}{\mathrm{d}x}\right] = -n\boldsymbol{F}_c^n[f]$$

$$\boldsymbol{F}_c^n\left[\frac{\mathrm{d}f}{\mathrm{d}x}\right] = n\boldsymbol{F}_s^n[f] - \frac{2}{\pi} f(0) + (-1)^n\frac{2}{\pi} f(\pi)$$

When two functions $F(x)$ and $G(x)$ are defined on the interval $-2\pi < x < 2\pi$, the function

$$F(x) * G(x) = \int_{-\pi}^{\pi} f(x-y)\,g(y)\,\mathrm{d}y$$

is the convolution on $-\pi < x < \pi$. If F and G are both even or both odd, the convolution is even; it is odd if one function is even and the other odd. If F and G are piecewise continuous on $0 \le x \le \pi$, then

$$f_s(n)\,g_s(n) = \boldsymbol{F}_c^n\left[-\tfrac{1}{2} F_1(x) * G_1(x)\right]$$

$$f_s(n)\,g_c(n) = \boldsymbol{F}_s^n\left[\tfrac{1}{2} F_1(x) * G_2(x)\right]$$

$$f_c(n)\,g_c(n) = \boldsymbol{F}_c^n\left[\tfrac{1}{2} F_2(x) * G_2(x)\right]$$

where F_1 and G_1 are odd extensions of F and G, respectively, and F_2 and G_2 are even extensions of F and G, respectively. Finite sine and cosine transforms are listed in Tables 3 and 4.

Table 4. Finite cosine transforms [41]

$f_c(n) =$	$F(x)$ $(0<x<\pi)$
$\int_0^\pi F(x)\cos n\,x\,\mathrm{d}x\ (n=0,1,\ldots)$	
$(-1)^n f_c(n)$	$F(\pi-x)$
0 when n=1,2, . . . ; $f_c(0)=\pi$	1
$\frac{2}{n}\sin n\,c$; $f_c(0)=2c-\pi$	$\begin{cases}1 & (0<x<c)\\ -1 & (c<x<\pi)\end{cases}$
$-\frac{1-(-1)^n}{n^2}$; $f_c(0)=\frac{\pi^2}{2}$	x
$\frac{(-1)^n}{n^2}$; $f_c(0)=\frac{\pi^2}{6}$	$\frac{x^2}{2\pi}$
$\frac{1}{n^2}$; $f_c(0)=0$	$\frac{(\pi-x)^2}{2\pi}-\frac{\pi}{6}$
$\frac{(-1)^n e^{cx}-1}{n^2+c^2}$	$\frac{1}{c}e^{cx}$
$\frac{1}{n^2+c^2}$	$\frac{\cosh c(\pi-x)}{c\sinh c\,\pi}$
$\frac{(-1)^n\cos k\,\pi-1}{n^2-k^2}$ $(\lvert k\rvert\neq 0,1,\ldots)$	$\frac{1}{k}\sin k\,x$
$\frac{(-1)^{n+m}-1}{n^2-m^2}$; $f_c(m)=0$ $(m=1,\ldots)$	$\frac{1}{m}\sin m\,x$
$\frac{1}{n^2-k^2}$ $(\lvert k\rvert\neq 0,1,\ldots)$	$-\frac{\cos k(\pi-x)}{k\sin k\,x}$
0 $(n\neq m)$; $f_c(m)=\frac{\pi}{2}$ $(m=1,2,\ldots)$	$\cos m\,x$

On the semi-infinite domain, $0<x<\infty$, the Fourier sine and cosine transforms are

$$\boldsymbol{F}_{\mathrm{s}}^{\omega}[f]\equiv\int_0^\infty f(x)\sin\omega\,x\,\mathrm{d}x,$$

$$\boldsymbol{F}_{\mathrm{c}}^{\omega}[f]\equiv\int_0^\infty f(x)\cos\omega\,x\,\mathrm{d}x\ \text{and}$$

$$f(x)=\tfrac{2}{\pi}\boldsymbol{F}_{\mathrm{s}}^{\omega\prime}\left[\boldsymbol{F}_{\mathrm{s}}^{\omega}[f]\right],\ f(x)=\tfrac{2}{\pi}\boldsymbol{F}_{\mathrm{c}}^{\omega\prime}\left[\boldsymbol{F}_{\mathrm{c}}^{\omega}[f]\right]$$

The sine transform is an odd function of ω, whereas the cosine function is an even function of ω. Also,

$$\boldsymbol{F}_{\mathrm{s}}^{\omega}\left[\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}\right]=f(0)\,\omega-\omega^2\boldsymbol{F}_{\mathrm{s}}^{\omega}[f]$$

$$\boldsymbol{F}_{\mathrm{c}}^{\omega}\left[\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}\right]=-\frac{\mathrm{d}f}{\mathrm{d}x}(0)-\omega^2\boldsymbol{F}_{\mathrm{c}}^{\omega}[f]$$

provided $f(x)$ and $f'(x)\to 0$ as $x\to\infty$. Thus, the sine transform is useful when $f(0)$ is known and the cosine transform is useful when $f'(0)$ is known.

Hsu and Dranoff [42] solved a chemical engineering problem by applying finite Fourier transforms and then using the fast Fourier transform (see Chap. 2).

4.2. Laplace Transforms

Consider a function $F(t)$ defined for $t>0$. The Laplace transform of $F(t)$ is [41]

$$\boldsymbol{L}[F]=f(s)=\int_0^\infty \mathrm{e}^{-st}F(t)\,\mathrm{d}t$$

The Laplace transformation is linear, that is,

$$\boldsymbol{L}[F+G]=\boldsymbol{L}[F]+\boldsymbol{L}[G]$$

Thus, the techniques described herein can be applied only to linear problems. Generally, the assumptions made below are that $F(t)$ is at least piecewise continuous, that it is continuous in each finite interval within $0<t<\infty$, and that it may take a jump between intervals. It is also of exponential order, meaning $\mathrm{e}^{-\alpha t}\lvert F(t)\rvert$ is bounded for all $t>T$, for some finite T.

The unit step function is

$$S_k(t)=\begin{cases}0 & 0\le t<k\\ 1 & t>k\end{cases}$$

and its Laplace transform is

$$\boldsymbol{L}[S_k(t)]=\frac{\mathrm{e}^{-ks}}{s}$$

In particular, if $k=0$ then

$$\boldsymbol{L}[1]=\frac{1}{s}$$

The Laplace transforms of the first and second derivatives of $F(t)$ are

$$\boldsymbol{L}\left[\frac{\mathrm{d}F}{\mathrm{d}t}\right]=s\,f(s)-F(0)$$

$$\boldsymbol{L}\left[\frac{\mathrm{d}^2F}{\mathrm{d}t^2}\right]=s^2 f(s)-s\,F(0)-\frac{\mathrm{d}F}{\mathrm{d}t}(0)$$

More generally,

$$\boldsymbol{L}\left[\frac{\mathrm{d}^n F}{\mathrm{d}t^n}\right]=s^n f(s)-s^{n-1}F(0)$$
$$-s^{n-2}\frac{\mathrm{d}F}{\mathrm{d}t}(0)-\cdots-\frac{\mathrm{d}^{n-1}F}{\mathrm{d}t^{n-1}}(0)$$

The inverse Laplace transformation is

$$F(t)=\boldsymbol{L}^{-1}[f(s)]\ \text{ where }\ f(s)=\boldsymbol{L}[F]$$

The inverse Laplace transformation is not unique because functions that are identical except for

isolated points have the same Laplace transform. They are unique to within a null function. Thus, if

$$\boldsymbol{L}[F_1] = f(s) \text{ and } \boldsymbol{L}[F_2] = f(s)$$

it must be that

$$F_2 = F_1 + N(t)$$

$$\text{where } \int_0^T (N(t))\,\mathrm{d}t = 0 \text{ for every } T$$

Laplace transforms can be inverted by using Table 5, but knowledge of several rules is helpful.

Table 5. Laplace transforms (see [23] for a more complete list)

$L[F]$	$F(t)$
$\frac{1}{s}$	1
$\frac{1}{s^2}$	t
$\frac{1}{s^n}$	$\frac{t^{n-1}}{(n-1)!}$
$\frac{1}{\sqrt{s}}$	$\frac{1}{\sqrt{\pi t}}$
$s^{-3/2}$	$2\sqrt{t/\pi}$
$\frac{\Gamma(k)}{s^k}\ (k>0)$	t^{k-1}
$\frac{1}{s-a}$	e^{at}
$\frac{1}{(s-a)^n}\ (n=1,2,\ldots)$	$\frac{1}{(n-1)!}t^{n-1}\mathrm{e}^{at}$
$\frac{\Gamma(k)}{(s-a)^k}\ (k>0)$	$t^{k-1}\ e^{at}$
$\frac{1}{(s-a)(s-b)}$	$\frac{1}{a-b}\left(\mathrm{e}^{at}-\mathrm{e}^{bt}\right)$
$\frac{s}{(s-a)(s-b)}$	$\frac{1}{a-b}\left(a\mathrm{e}^{at}-b\mathrm{e}^{bt}\right)$
$\frac{1}{s^2+a^2}$	$\frac{1}{a}\sin a\,t$
$\frac{s}{s^2+a^2}$	$\cos a\,t$
$\frac{1}{s^2-a^2}$	$\frac{1}{a}\sinh a\,t$
$\frac{s}{s^2-a^2}$	$\cosh a\,t$
$\frac{s}{(s^2+a^2)^2}$	$\frac{t}{2a}\sin a\,t$
$\frac{s^2-a^2}{(s^2+a^2)^2}$	$t\cos a\,t$
$\frac{1}{(s-a)^2+b^2}$	$\frac{1}{b}\mathrm{e}^{a\,t}\sin b\,t$
$\frac{s-a}{(s-a)^2+b^2}$	$\mathrm{e}^{at}\cos b\,t$

Substitution.

$$f(s-a) = \boldsymbol{L}\left[\mathrm{e}^{at}\,F(t)\right]$$

This can be used with polynomials. Suppose

$$f(s) = \frac{1}{s} + \frac{1}{s+3} = \frac{2s+3}{s(s+3)}$$

Because

$$\boldsymbol{L}[1] = \frac{1}{s}$$

Then

$$F(t) = 1+\mathrm{e}^{-3t},\ t \geq 0$$

More generally, translation gives the following.

Translation.

$$f(as-b) = f\left[a\left(s-\frac{b}{a}\right)\right] = \boldsymbol{L}\left[\frac{1}{a}\,\mathrm{e}^{bt/a}\,F\left(\frac{t}{a}\right)\right],$$

$a>0$

The step function

$$S(t) = \begin{cases} 0 & 0 \leq t < \frac{1}{h} \\ 1 & \frac{1}{h} \leq t < \frac{2}{h} \\ 2 & \frac{2}{h} \leq t < \frac{3}{h} \end{cases}$$

has the Laplace transform

$$\boldsymbol{L}[S(t)] = \frac{1}{s}\frac{1}{1-\mathrm{e}^{-hs}}$$

The Dirac delta function $\delta(t-t_0)$ (see Equation 116 in → Mathematical Modeling) has the property

$$\int_0^\infty \delta(t-t_0)\ F(t)\ \mathrm{d}t = F(t_0)$$

Its Laplace transform is

$$\boldsymbol{L}[\delta(t-t_0)] = \mathrm{e}^{-st_0}, t_0 \geq 0, s>0$$

The square wave function illustrated in Figure 10 has Laplace transform

$$\boldsymbol{L}[F_{\mathrm{c}}(t)] = \frac{1}{s}\tanh\frac{cs}{2}$$

The triangular wave function illustrated in Figure 11 has Laplace transform

$$L[T_{\mathrm{c}}(t)] = \frac{1}{s^2}\tanh\frac{cs}{2}$$

Other Laplace transforms are listed in Table 5.

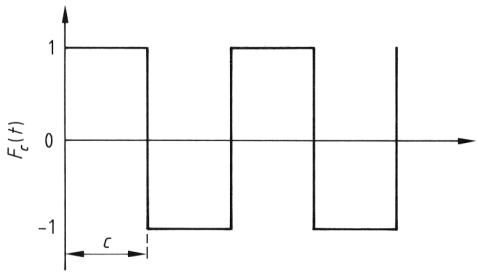

Figure 10. Square wave function

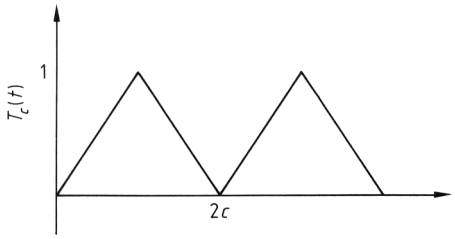

Figure 11. Triangular wave function

Convolution properties are also satisfied:

$$F\ (t)\ *G\ (t) = \int_0^t F\ (\tau)\ G\ (t-\tau)\,\mathrm{d}\tau$$

and

$$f\,(s)\,g\,(s) = \boldsymbol{L}\,[F\,(t)\ *G\,(t)]$$

Derivatives of Laplace Transforms. The Laplace integrals $\boldsymbol{L}\,[F\,(t)]$, $\boldsymbol{L}\,[t\,F\,(t)]$, $\boldsymbol{L}\,[t^2\,F\,(t)]$, . . . are uniformly convergent for $s_1 \geq \alpha$ and

$$\lim_{s\to\infty} f\ (s) = 0,\ \lim_{s\to\infty} \boldsymbol{L}\,[t^n\ F\ (t)] = 0, n = 1, 2,\ldots$$

and

$$\frac{\mathrm{d}^n f}{\mathrm{d}s^n} = \boldsymbol{L}\,[(-t)^n\ F\ (t)]$$

Integration of Laplace Transforms.

$$\int_s^\infty f\ (\xi)\,\mathrm{d}\xi = \boldsymbol{L}\left[\frac{F\ (t)}{t}\right]$$

If $F\ (t)$ is a periodic function, $F\ (t) = F\ (t + a)$, then

$$f\ (s) = \frac{1}{1-\mathrm{e}^{-as}}\int_0^a \mathrm{e}^{-st} F\ (t)\,\mathrm{d}t,$$

where $F\ (t) = F\ (t+a)$

Partial Fractions [43]**.** Suppose $q\ (s)$ has m factors

$$q\ (s) = (s-a_1)\,(s-a_2)\cdots(s-a_m)$$

All the factors are linear, none are repeated, and the a_n are all distinct. If $p\ (s)$ has a smaller degree than $q\ (s)$, the Heaviside expansion can be used to evaluate the inverse transformation:

$$\boldsymbol{L}^{-1}\left[\frac{p\ (s)}{q\ (s)}\right] = \sum_{i=1}^{m} \frac{p\ (a_i)}{q'\ (a_i)}\,\mathrm{e}^{a_i t}$$

If the factor $(s - a)$ is repeated m times, then

$$f\ (s) = \frac{p\,(s)}{q\,(s)} = \frac{A_m}{(s-a)^m} + \frac{A_{m-1}}{(s-a)^{m-1}} + \cdots + \frac{A_1}{s-a} + h\ (s)$$

where

$$\varphi\ (s) \equiv \frac{(s-a)^m\ p\,(s)}{q\,(s)}$$

$$A_m = \varphi\ (a)\,,\ A_k = \frac{1}{(m-k)!}\,\frac{\mathrm{d}^{m-k}\,\varphi\,(s)}{\mathrm{d}s^{m-k}}|_a,$$

$$k = 1,\ldots,m-1$$

The term $h\ (s)$ denotes the sum of partial fractions not under consideration. The inverse transformation is then

$$F\ (t) = \mathrm{e}^{at}\left(A_m\frac{t^{m-1}}{(m-1)\,!} + A_{m-1}\frac{t^{m-2}}{(m-2)\,!} + \cdots + A_2\frac{t}{1!} + A_1\right) + H\ (t)$$

The term in $F\ (t)$ corresponding to

$$s-a \text{ in } q\ (s) \text{ is } \varphi\ (a)\ \mathrm{e}^{a\,t}$$

$$(s-a)^2 \text{ in } q\ (s) \text{ is } [\varphi'\ (a) + \varphi\ (a)\ t]\,\mathrm{e}^{a\,t}$$

$$(s-a)^3 \text{ in } q\ (s) \text{ is } \tfrac{1}{2}\left[\varphi''\,(a) + 2\varphi'\ (a)\ t + \varphi\ (a)\ t^2\right]\mathrm{e}^{a\,t}$$

For example, let

$$f\ (s) = \frac{1}{(s-2)\ (s-1)^2}$$

For the factor $s - 2$,

$$\varphi\ (s) = \frac{1}{(s-1)^2},\ \varphi\ (2) = 1$$

For the factor $(s - 1)^2$,

$$\varphi\ (s) = \tfrac{1}{s-2}, \varphi'\ (s) = -\tfrac{1}{(s-2)^2}$$

$$\varphi\ (1) = -1,\ \varphi'\,(1) = -1$$

The inverse Laplace transform is then

$$F\,(t) = \mathrm{e}^{2t} + [-1-t]\ \mathrm{e}^{t}$$

Quadratic Factors. Let $p\ (s)$ and $q\ (s)$ have real coefficients, and $q\ (s)$ have the factor

$$(s-a)^2+b^2, b>0$$

where a and b are real numbers. Then define $\varphi\ (s)$ and $h\ (s)$ and real constants A and B such that

$$f\ (s) = \frac{p\ (s)}{q\ (s)} = \frac{\varphi\ (s)}{(s-a)^2+b^2}$$

$$= \frac{A\,s+B}{(s-a)^2+b^2} + h\ (s)$$

Let φ_1 and φ_2 be the real and imaginary parts of the complex number $\varphi\ (a + \mathrm{i}\ b)$.

$$\varphi\,(a+\mathrm{i}\,b) \equiv \varphi_1 + \mathrm{i}\,\varphi_2$$

Then

$$f\ (s) = \frac{1}{b}\,\frac{(s-a)\varphi_2+b\varphi_1}{(s-a)^2+b^2} + h\ (s)$$

$$F\ (t) = \frac{1}{b}\,\mathrm{e}^{at}\ (\varphi_2 \cos b\,t + \varphi_1 \sin b\,t) + H\ (t)$$

To solve ordinary differential equations by using these results:

$$Y''\ (t)\ -2Y'\,(t) + Y\,(t) = \mathrm{e}^{2t}$$

$$Y\,(0) = 0, Y'\,(0) = 0$$

Taking Laplace transforms

$$\boldsymbol{L}\left[Y''\ (t)\right] - 2\boldsymbol{L}\left[Y'\,(t)\right] + \boldsymbol{L}\left[Y\ (t)\right] = \frac{1}{s-2}$$

using the rules

$$s^2\,y\ (s) - s\,Y\ (0) - Y'\,(0) - 2\left[s\,y\ (s) - Y\ (0)\right] + y\ (s) = \frac{1}{s-2}$$

and combining terms

$$\left(s^2-2\,s+1\right)\ y\ (s) = \frac{1}{s-2}$$

$$y\ (s) = \frac{1}{(s-2)(s-1)^2}$$

lead to

$$Y\,(t) = \mathrm{e}^{2t} - (1+t)\ \mathrm{e}^{t}$$

To solve an integral equation:

$$Y\ (t) = a + 2\int_0^t Y\ (\tau) \cos\ (t-\tau)\,\mathrm{d}\tau$$

it is written as

$$Y\,(t) = a + Y\,(t)\ * \cos t$$

Then the Laplace transform is used to obtain

$$y\ (s) = \frac{a}{s} + 2\,y\ (s)\,\frac{s}{s^2+1}$$

$$\text{or } y\ (s) = \frac{a\left(s^2+1\right)}{s\,(s-1)^2}$$

Taking the inverse transformation gives

$$Y\,(t) = s\left[1+2t\mathrm{e}^{t}\right]$$

Next, let the variable s in the Laplace transform be complex. $F\ (t)$ is still a real-valued function of the positive real variable t. The properties given above are still valid for s complex, but additional techniques are available for evaluating the integrals. The real-valued function is O [exp $(x_0 t)$]:

$$|F\ (t)| < M\mathrm{e}^{x_0 t},\ z_0 = x_0 + \mathrm{i}\,y_0$$

The Laplace transform

$$f\ (s) = \int_0^\infty \mathrm{e}^{-st}\,F\ (t)\,\mathrm{d}t$$

is an analytic function of s in the half-plane $x > x_0$ and is absolutely convergent there; it is uniformly convergent on $x \geq x_1 > x_0$.

$$\frac{\mathrm{d}^n f}{\mathrm{d}s^n} = \boldsymbol{L}\left[(-t)^n\,F\ (t)\right]\ \ n = 1, 2, \ldots, x > x_0$$

$$\text{and } f^*\ (s) = f\ (s^*)$$

The functions $|f\ (s)|$ and $|x f\ (s)|$ are bounded in the half-plane $x \geq x_1 > x_0$ and $f\ (s) \to 0$ as $|y| \to \infty$ for each fixed x. Thus,

$$|f\ (x+\mathrm{i}\,y)| < M,\ \ |x\,f\ (x+\mathrm{i}\,y)| < M,$$

$$x \geq x_1 > x_0$$

$$\lim_{y \to \pm\infty} f\ (x+\mathrm{i}\,y) = 0,\ x > x_0$$

If $F\ (t)$ is continuous, $F'\ (t)$ is piecewise continuous, and both functions are O [exp $(x_0 t)$], then $|f\ (s)|$ is $O\ (1/s)$ in each half-plane $x \geq x_1 > x_0$.

$$|s\,f\ (s)| < M$$

If $F\ (t)$ and $F'\ (t)$ are continuous, $F''\ (t)$ is piecewise continuous, and all three functions are O [exp $(x_0 t)$], then

$$\left|s^2 f\ (s) - s\,F\,(0)\right| < M, x \geq x_1 > x_0$$

The additional constraint $F\ (0) = 0$ is necessary and sufficient for $|f\ (s)|$ to be $O\ (1/s^2)$.

Inversion Integral [41]. Cauchy's integral formula for $f\ (s)$ analytic and $O\ (s^{-k})$ in a half-plane $x \geq y$, $k > 0$, is

$$f\ (s) = \frac{1}{2\,\pi\mathrm{i}} \lim_{\beta \to \infty} \int_{\gamma-\mathrm{i}\beta}^{\gamma+\mathrm{i}\beta} \frac{f\ (z)}{s-z}\mathrm{d}z,\ \mathrm{Re}\,(s) > \gamma$$

Applying the inverse Laplace transformation on either side of this equation gives

$$F\ (t) = \frac{1}{2\,\pi \mathrm{i}} \lim_{\beta \to \infty} \int_{\gamma - \mathrm{i}\beta}^{\gamma + \mathrm{i}\beta} \mathrm{e}^{zt}\ f\ (z)\,\mathrm{d}z$$

If $F\ (t)$ is of order $O\ [\exp\ (x_0 t)]$ and $F\ (t)$ and $F'\ (t)$ are piecewise continuous, the inversion integral exists. At any point t_0, where $F\ (t)$ is discontinuous, the inversion integral represents the mean value

$$F\ (t_0) = \lim_{\varepsilon \to \infty} \frac{1}{2}\ [F\ (t_0 + \varepsilon) + F\,(t_0 - \varepsilon)]$$

When $t = 0$ the inversion integral represents 0.5 $F\ (O+)$ and when $t < 0$, it has the value zero.

If $f\ (s)$ is a function of the complex variable s that is analytic and of order $O\ (s^{-k-m})$ on $R\ (s) \geq x_0$, where $k > 1$ and m is a positive integer, then the inversion integral converges to $F\ (t)$ and

$$\frac{\mathrm{d}^n F}{\mathrm{d}t^n} = \frac{1}{2\,\pi \mathrm{i}} \lim_{\beta \to \infty} \int_{\gamma - \mathrm{i}\beta}^{\gamma + \mathrm{i}\beta} \mathrm{e}^{zt} z^n\ f\ (z)\,\mathrm{d}z,$$

$$n = 1,2,\ldots,m$$

Also $F\ (t)$ and its n derivatives are continuous functions of t of order $O\ [\exp\ (x_0 t)]$ and they vanish at $t = 0$.

$$F\,(0) = F'\,(0) = \cdots F^{(m)}\ (0) = 0$$

Series of Residues [41]. Let $f\ (s)$ be an analytic function except for a set of isolated singular points. An isolated singular point is one for which $f\ (z)$ is analytic for $0 < |z - z_0| < \varrho$ but z_0 is a singularity of $f\ (z)$. An isolated singular point is either a pole, a removable singularity, or an essential singularity. If $f\ (z)$ is not defined in the neighborhood of z_0 but can be made analytic at z_0 simply by defining it at some additional points, then z_0 is a *removable singularity*. The function $f\ (z)$ has a *pole* of order $k \geq 1$ at z_0 if $(z - z_0)^k f\ (z)$ has a removable singularity at z_0 whereas $(z - z_0)^{k-1} f\ (z)$ has an unremovable isolated singularity at z_0. Any isolated singularity that is not a pole or a removable singularity is an *essential* singularity.

Let the function $f\ (z)$ be analytic except for the isolated singular point $s_1, s_2, \ldots, s_n$. Let $\varrho_n\ (t)$ be the residue of $\mathrm{e}^{zt} f\ (z)$ at $z = s_n$ (for definition of residue, see Section 3.4). Then

$$F\ (t) = \sum_{n=1}^{\infty} \varrho_n\ (t)$$

When s_n is a simple pole

$$\varrho_n\ (t) = \lim_{z \to s_n} (z - s_n)\,\mathrm{e}^{zt}\ f\ (z)$$

$$= \mathrm{e}^{s_n t} \lim_{z \to s_n} (z - s_n)\ f\ (z)$$

When

$$f\ (z) = \frac{p\ (z)}{q\ (z)}$$

where $p\ (z)$ and $q\ (z)$ are analytic at $z = s_n$, $p\ (s_n) \neq 0$, then

$$\varrho_n\ (t) = \frac{p\ (s_n)}{q'\ (s_n)}\,\mathrm{e}^{s_n t}$$

If s_n is a removable pole of $f\ (s)$, of order m, then

$$\varphi_n\ (z) = (z - s_n)^m\ f\,(z)$$

is analytic at s_n and the residue is

$$\varrho_n\ (t) = \frac{\Phi_n\ (s_n)}{(m-1)!} \text{ where } \Phi_n\ (z) = \frac{\partial^{m-1}}{\partial z^{m-1}}\ [\varphi_n\ (z)\,\mathrm{e}^{zt}]$$

An important inversion integral is when

$$f\ (s) = \frac{1}{s} \exp\left(-s^{1/2}\right)$$

The inverse transform is

$$F\ (t) = 1 - \text{erf}\ \left(\frac{1}{2\sqrt{t}}\right) = \text{erfc}\ \left(\frac{1}{2\sqrt{t}}\right)$$

where erf is the error function and erfc the complementary error function.

4.3. Solution of Partial Differential Equations by Using Transforms

A common problem facing chemical engineers is to solve the heat conduction equation or diffusion equation

$$\varrho\, C_p \frac{\partial T}{\partial t} = k\,\frac{\partial^2 T}{\partial x^2} \text{ or } \frac{\partial c}{\partial t} = D\,\frac{\partial^2 c}{\partial x^2}$$

The equations can be solved on an infinite domain $-\infty < x < \infty$, a semi-infinite domain $0 \leq x < \infty$, or a finite domain $0 \leq x \leq L$. At a boundary, the conditions can be a fixed temperature $T\ (0, t) = T_0$ (boundary condition of the first kind, or Dirichlet condition), or a fixed flux $-k\frac{\partial T}{\partial x}\ (0,t) = q_0$ (boundary condition of the second kind, or Neumann condition), or a

combination $-k\frac{\partial T}{\partial x}(0,t) = h[T(0,t) - T_0]$ (boundary condition of the third kind, or Robin condition).

The functions T_0 and q_0 can be functions of time. All properties are constant (ϱ, C_p, k, D, h), so that the problem remains linear. Solutions are presented on all domains with various boundary conditions for the heat conduction problem.

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2},\ \alpha = \frac{k}{\varrho C_p}$$

Problem 1. Infinite domain, on $-\infty < x < \infty$.

$T(x,0) = f(x)$, initial conditions

$T(x,t)$ bounded

Solution is via Fourier transforms

$$\hat{T}(\omega,t) = \int_{-\infty}^{\infty} T(x,t)\, \mathrm{e}^{\mathrm{i}\omega x}\, \mathrm{d}x$$

Applied to the differential equation

$$\boldsymbol{F}\left[\frac{\partial^2 T}{\partial x^2}\right] = -\omega^2 \alpha \boldsymbol{F}[T]$$

$$\frac{\partial \hat{T}}{\partial t} + \omega^2 \alpha \hat{T} = 0, \hat{T}(\omega,0) = \hat{f}(\omega)$$

By solving

$$\hat{T}(\omega,t) = \hat{f}(\omega)\, \mathrm{e}^{-\omega^2 \alpha t}$$

the inverse transformation gives [40, p. 328], [44, p. 58]

$$T(x,t) = \frac{1}{2\pi} \lim_{L\to\infty} \int_{-L}^{L} \mathrm{e}^{-\mathrm{i}\omega x}\, \hat{f}(\omega)\, \mathrm{e}^{-\omega^2 \alpha t} \mathrm{d}\omega$$

Another solution is via Laplace transforms; take the Laplace transform of the original differential equation.

$$s\,t(s,x) - f(x) = \alpha \frac{\partial^2 t}{\partial x^2}$$

This equation can be solved with Fourier transforms [40, p. 355]

$$t(s,x) = \frac{1}{2\sqrt{s\alpha}} \int_{-\infty}^{\infty} \mathrm{e}\left\{-\sqrt{\frac{s}{\alpha}}\,|x-y|\right\} f(y)\, \mathrm{d}y$$

The inverse transformation is [40, p. 357], [44, p. 53]

$$T(x,t) = \frac{1}{2\sqrt{\pi \alpha t}} \int_{-\infty}^{\infty} \mathrm{e}^{-(x-y)^2/4\alpha t} f(y)\, \mathrm{d}y$$

Problem 2. Semi-infinite domain, boundary condition of the first kind, on $0 \le x \le \infty$

$T(x,0) = T_0 =$ constant

$T(0,t) = T_1 =$ constant

The solution is

$$T(x,t) = T_0 + [T_1 - T_0]\left[1 - \text{erf}\left(x/\sqrt{4\alpha t}\right)\right]$$

$$\text{or } T(x,t) = T_0 + (T_1 - T_0)\ \text{erfc}\left(x/\sqrt{4\alpha t}\right)$$

Problem 3. Semi-infinite domain, boundary condition of the first kind, on $0 \le x < \infty$

$T(x,0) = f(x)$

$T(0,t) = g(t)$

The solution is written as

$$T(x,t) = T_1(x,t) + T_2(x,t)$$

where

$T_1(x,0) = f(x), T_2(x,0) = 0$

$T_1(0,t) = 0, T_2(0,t) = g(t)$

Then T_1 is solved by taking the sine transform

$$U_1 = F_s^{\omega}[T_1]$$

$$\frac{\partial U_1}{\partial t} = -\omega^2 \alpha U_1$$

$$U_1(\omega,0) = F_s^{\omega}[f]$$

Thus,

$$U_1(\omega,t) = F_s^{\omega}[f]\, \mathrm{e}^{-\omega^2 \alpha t}$$

and [40, p. 322]

$$T_1(x,t) = \frac{2}{\pi} \int_0^{\infty} F_s^{\omega}[f]\, \mathrm{e}^{-\omega^2 \alpha t} \sin \omega x\, \mathrm{d}\omega$$

Solve for T_2 by taking the sine transform

$$U_2 = F_s^{\omega}[T_2]$$

$$\frac{\partial U_2}{\partial t} = -\omega^2 \alpha U_2 + \alpha g(t)\ \omega$$

$$U_2(\omega,0) = 0$$

Thus,

$$U_2(\omega,t)=\int_0^t \mathrm{e}^{-\omega^2\alpha(t-\tau)}\alpha\omega\, g(\tau)\,\mathrm{d}\tau$$

and [40, p. 435]

$$T_2(x,t)=\frac{2\alpha}{\pi}\int_0^\infty \omega\sin\omega x\int_0^t \mathrm{e}^{-\omega^2\alpha(t-\tau)}\,g(\tau)\,\mathrm{d}\tau\mathrm{d}\omega$$

The solution for T_1 can also be obtained by Laplace transforms.

$$t_1=\boldsymbol{L}[T_1]$$

Applying this to the differential equation

$$st_1-f(x)=\alpha\frac{\partial^2 t_1}{\partial x^2},\ t_1(0,s)=0$$

and solving gives

$$t_1=\frac{1}{\sqrt{s\,\alpha}}\int_0^x \mathrm{e}^{-\sqrt{s/\alpha}(x'-x)}\,f(x')\,\mathrm{d}x'$$

and the inverse transformation is [40, p. 437], [44, p. 59]

$$T_1(x,t)= \tag{7}$$
$$\frac{1}{\sqrt{4\pi\alpha t}}\int_0^\infty\left[\mathrm{e}^{-(x-\xi)^2/4\alpha t}-\mathrm{e}^{-(x+\xi)^2/4\alpha t}\right]f(\xi)\,\mathrm{d}\xi$$

Problem 4. Semi-infinite domain, boundary conditions of the second kind, on $0\le x<\infty$.

$$T(x,0)=0$$

$$-k\frac{\partial T}{\partial x}(0,t)=q_0=\text{constant}$$

Take the Laplace transform

$$t(x,s)=L[T(x,t)]$$

$$st=\alpha\frac{\partial^2 t}{\partial x^2}$$

$$-k\frac{\partial t}{\partial x}=\frac{q_0}{s}$$

The solution is

$$t(x,s)=\frac{q_0\sqrt{\alpha}}{k\,s^{3/2}}\mathrm{e}^{-x\sqrt{s/\alpha}}$$

The inverse transformation is [41, p. 131], [44, p. 75]

$$T(x,t)=\frac{q_0}{k}\left[2\sqrt{\frac{\alpha t}{\pi}}\,\mathrm{e}^{-x^2/4\alpha t}-x\,\mathrm{erfc}\left(\frac{x}{\sqrt{4\alpha t}}\right)\right]$$

Problem 5. Semi-infinite domain, boundary conditions of the third kind, on $0\le x<\infty$

$$T(x,0)=f(x)$$

$$k\frac{\partial T}{\partial x}(0,t)=h\,T(0,t)$$

Take the Laplace transform

$$st-f(x)=\alpha\frac{\partial^2 t}{\partial x^2}$$

$$k\frac{\partial t}{\partial x}(0,s)=h\,t(0,s)$$

The solution is

$$t(x,s)=\int_0^\infty f(\xi)\,g(x,\xi,s)\,\mathrm{d}\xi$$

where [41, p. 227]

$$2\sqrt{s/\alpha}\,g(x,\xi,s)=\exp\left(-|x-\xi|\sqrt{s/\alpha}\right)$$
$$+\frac{\sqrt{s}-h\sqrt{\alpha}/k}{\sqrt{s}+h\sqrt{\alpha}/k}\exp\left[-(x+\xi)\sqrt{s/\alpha}\right]$$

One form of the inverse transformation is [41, p. 228]

$$T(x,t)=\frac{2}{\pi}\int_0^\infty \mathrm{e}^{-\beta^2\alpha t}\cos[\beta x-\mu(\beta)]\int_0^\infty f(\xi)$$
$$\cos[\beta\xi-\mu(\beta)]\,\mathrm{d}\xi\mathrm{d}\beta$$
$$\mu(\beta)=\arg(\beta+i\,h/k)$$

Another form of the inverse transformation when $f = T_0$= constant is [41, p. 231], [44, p. 71]

$$T(x,t)=T_0\left[\mathrm{erf}\left(\frac{x}{\sqrt{4\alpha t}}\right)+\mathrm{e}^{hx/k}\,\mathrm{e}^{h^2\alpha t/k^2}\right.$$
$$\left.\mathrm{erfc}\left(\frac{h\sqrt{\alpha t}}{k}+\frac{x}{\sqrt{4\alpha t}}\right)\right]$$

Problem 6. Finite domain, boundary condition of the first kind

$$T(x,0)=T_0=\text{constant}$$

$$T(0,t)=T(L,t)=0$$

Take the Laplace transform

$$st(x,s)-T_0=\alpha\frac{\partial^2 t}{\partial x^2}$$

$$t(0,s)=t(L,s)=0$$

The solution is

$$t(x,s)=-\frac{T_0}{s}\frac{\sinh\sqrt{\frac{s}{\alpha}}x}{\sinh\sqrt{\frac{s}{\alpha}}L}-\frac{T_0}{s}\frac{\sinh\sqrt{\frac{s}{\alpha}}(L-x)}{\sinh\sqrt{\frac{s}{\alpha}}L}+\frac{T_0}{s}$$

The inverse transformation is [41, p. 220], [44, p. 96]

$$T\ (x,t) = \frac{2}{\pi}\, T_0 \sum\nolimits_{n=1,3,5,\ldots} \frac{2}{n}\, \mathrm{e}^{-n^2\pi^2\alpha t/L^2} \sin\frac{n\,\pi\,x}{L}$$

or (depending on the inversion technique) [40, pp. 362, 438]

$$T\ (x,t) = \frac{T_0}{\sqrt{4\,\pi\,\alpha\,t}} \int_0^L \sum_{n=-\infty}^{\infty} \left[\mathrm{e}^{-[(x-\xi)+2nL]^2/4\alpha t} - \mathrm{e}^{-[(x+\xi)+2nL]^2/4\alpha t}\right] \mathrm{d}\xi$$

Problem 7. Finite domain, boundary condition of the first kind

$$T\,(x,0) = 0$$

$$T\,(0,t) = 0$$

$$T\,(L,0) = T_0 = \text{constant}$$

Take the Laplace transform

$$s\,t\ (x,s) = \alpha\ \frac{\partial^2 t}{\partial x^2}$$

$$t\ (0,s) = 0, t\ (L,s) = \frac{T_0}{s}$$

The solution is

$$t\ (x,s) = \frac{T_0}{s}\,\frac{\sinh\frac{x}{L}\sqrt{s/\alpha}}{\sinh\sqrt{s/\alpha}}$$

and the inverse transformation is [41, p. 201], [44, p. 313]

$$T\ (x,t) = T_0\left[\frac{x}{L} + \frac{2}{\pi}\sum_{n=1}^{\infty}\frac{(-1)^n}{n}\,\mathrm{e}^{-n^2\pi^2\alpha t/L^2}\sin\frac{n\,\pi\,x}{L}\right]$$

An alternate transformation is [41, p. 139], [44, p. 310]

$$T\ (x,t) = T_0 \sum_{n=0}^{\infty}\left[\mathrm{erf}\left(\frac{(2\,n{+}1)\ L{+}x}{\sqrt{4\,\alpha\,t}}\right) - \mathrm{erf}\left(\frac{(2\,n{+}1)\ L{-}x}{\sqrt{4\,\alpha\,t}}\right)\right]$$

Problem 8. Finite domain, boundary condition of the second kind

$$T\,(x,0) = T_0$$

$$\frac{\partial\,T}{\partial x}\,(0,t) = 0, T\ (L,t) = 0$$

Take the Laplace transform

$$s\,t\ (x,s) - T_0 = \alpha\,\frac{\partial^2 t}{\partial x^2}$$

$$\frac{\partial t}{\partial x}\,(0,s) = 0, t\ (L,s) = 0$$

The solution is

$$t\ (x,s) = \frac{T_0}{s}\left[1 - \frac{\cosh x\,\sqrt{s/\alpha}}{\cosh L\,\sqrt{s/\alpha}}\right]$$

Its inverse is [41, p. 138]

$$T\ (x,t) = T_0 - T_0 \sum_{n=0}^{\infty}(-1)^n\left[\mathrm{erfc}\left(\frac{(2n{+}1)\ L{-}x}{\sqrt{4\,\alpha\,t}}\right) + \mathrm{erfc}\left(\frac{(2n{+}1)\ L{+}x}{\sqrt{4\,\alpha\,t}}\right)\right]$$

5. Vector Analysis

Notation. A *scalar* is a quantity having magnitude but no direction (e.g., mass, length, time, temperature, and concentration). A *vector* is a quantity having both magnitude and direction (e.g., displacement, velocity, acceleration, force). A second-order *dyadic* has magnitude and two directions associated with it, as defined precisely below. The most common examples are the stress dyadic (or tensor) and the velocity gradient (in fluid flow). Vectors are printed in boldface type and identified in this chapter by lower-case Latin letters. Second-order dyadics are printed in boldface type and are identified in this chapter by capital or Greek letters. Higher order dyadics are not discussed here. Dyadics can be formed from the components of tensors including their directions, and some of the identities for dyadics are more easily proved by using tensor analysis, which is not presented here (see also, → Transport Phenomena, Chap. 1.1.4). Vectors are also first-order dyadics.

Vectors. Two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ are equal if they have the same magnitude and direction. If they have the same magnitude but the opposite direction, then $\boldsymbol{u} = -\boldsymbol{v}$. The sum of two vectors is identified geometrically by placing the vector $\boldsymbol{v}$ at the end of the vector $\boldsymbol{u}$, as shown in Figure 12. The product of a scalar m and a vector $\boldsymbol{u}$ is a vector $m\,\boldsymbol{u}$ in the same direction as $\boldsymbol{u}$ but with a magnitude that equals the magnitude of $\boldsymbol{u}$ times the scalar m. Vectors obey commutative and associative laws.

$\boldsymbol{u} + \boldsymbol{v} = \boldsymbol{v} + \boldsymbol{u}$	Commutative law for addition
$\boldsymbol{u} + (\boldsymbol{v} + \boldsymbol{w}) = (\boldsymbol{u} + \boldsymbol{v}) + \boldsymbol{w}$	Associative law for addition
$m\,\boldsymbol{u} = \boldsymbol{u}\,m$	Commutative law for scalar multiplication
$m\,(n\,\boldsymbol{u}) = (m\,n)\,\boldsymbol{u}$	Associative law for scalar multiplication
$(m + n)\,\boldsymbol{u} = m\,\boldsymbol{u} + n\,\boldsymbol{u}$	Distributive law
$m\,(\boldsymbol{u} + \boldsymbol{v}) = m\,\boldsymbol{u} + m\,\boldsymbol{v}$	Distributive law

The same laws are obeyed by dyadics, as well.

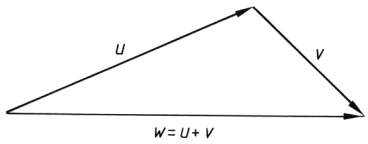

Figure 12. Addition of vectors

A unit vector is a vector with magnitude 1.0 and some direction. If a vector has some magnitude (i.e., not zero magnitude), a unit vector $\boldsymbol{e}_u$ can be formed by

$$\boldsymbol{e}_u = \frac{\boldsymbol{u}}{|\boldsymbol{u}|}$$

The original vector can be represented by the product of the magnitude and the unit vector.

$$\boldsymbol{u} = |\boldsymbol{u}|\,\boldsymbol{e}_u$$

In a cartesian coordinate system the three principle, orthogonal directions are customarily represented by *unit vectors*, such as $\{\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z\}$ or $\{\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}\}$. Here, the first notation is used (see Fig. 13). The coordinate system is right handed; that is, if a right-threaded screw rotated from the x to the y direction, it would advance in the z direction. A vector can be represented in terms of its components in these directions, as illustrated in Figure 14. The vector is then written as

$$\boldsymbol{u} = u_x\boldsymbol{e}_x + u_y\boldsymbol{e}_y + u_z\boldsymbol{e}_z$$

The magnitude is

$$|\boldsymbol{u}| = \sqrt{u_x^2 + u_y^2 + u_z^2}$$

The position vector is

$$\boldsymbol{r} = x\boldsymbol{e}_x + y\boldsymbol{e}_y + z\boldsymbol{e}_z$$

with magnitude

$$|\boldsymbol{r}| = \sqrt{x^2 + y^2 + z^2}$$

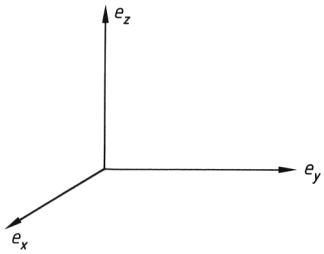

Figure 13. Cartesian coordinate system

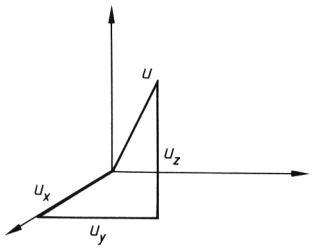

Figure 14. Vector components

Dyadics. The dyadic $\boldsymbol{A}$ is written in component form in cartesian coordinates as

$$\boldsymbol{A} = A_{xx}\boldsymbol{e}_x\boldsymbol{e}_x + A_{xy}\boldsymbol{e}_x\boldsymbol{e}_y + A_{xz}\boldsymbol{e}_x\boldsymbol{e}_z$$

$$+ A_{yx}\boldsymbol{e}_y\boldsymbol{e}_x + A_{yy}\boldsymbol{e}_y\boldsymbol{e}_y + A_{yz}\boldsymbol{e}_y\boldsymbol{e}_z$$

$$+ A_{zx}\boldsymbol{e}_z\boldsymbol{e}_x + A_{zy}\boldsymbol{e}_z\boldsymbol{e}_y + A_{zz}\boldsymbol{e}_z\boldsymbol{e}_z$$

Quantities such as $\boldsymbol{e}_x\boldsymbol{e}_y$ are called *unit dyadics*. They are second-order dyadics and have two directions associated with them, $\boldsymbol{e}_x$ and $\boldsymbol{e}_y$; the order of the pair is important. The components $A_{xx}, \ldots, A_{zz}$ are the components of the tensor A_{ij} which here is a 3×3 matrix of numbers that are transformed in a certain way when variables undergo a linear transformation. The $y\,x$ momentum flux can be defined as the flux of x momentum across an area with unit normal in the y direction. Since two directions are involved, a second-order dyadic (or tensor) is needed to represent it, and because the y momentum across an area with unit normal in the x direction may not be the same thing, the order of the indices must be kept straight. The dyadic $\boldsymbol{A}$ is said to be symmetric if

$$\boldsymbol{A}_{ij} = \boldsymbol{A}_{ji}$$

Here, the indices i and j can take the values x, y, or z; sometimes (x, 1), (y, 2), (x, 3) are identified and the indices take the values 1, 2, or 3. The dyadic $\boldsymbol{A}$ is said to be antisymmetric if

$$\boldsymbol{A}_{ij} = -\boldsymbol{A}_{ji}$$

The transpose of $\boldsymbol{A}$ is

$$\boldsymbol{A}_{ij}^{\mathrm{T}} = \boldsymbol{A}_{ji}$$

Any dyadic can be represented as the sum of a symmetric portion and an antisymmetric portion.

$$\boldsymbol{A}_{ij} = \boldsymbol{B}_{ij} + \boldsymbol{C}_{ij}, \boldsymbol{B}_{ij} \equiv \tfrac{1}{2}\left(\boldsymbol{A}_{ij} + \boldsymbol{A}_{ji}\right),$$

$$\boldsymbol{C}_{ij} \equiv \tfrac{1}{2}\left(\boldsymbol{A}_{ij} - \boldsymbol{A}_{ji}\right)$$

An ordered pair of vectors is a second-order dyadic.

$$\boldsymbol{u}\,\boldsymbol{v} = \sum_i \sum_j \boldsymbol{e}_i \boldsymbol{e}_j u_i\, v_j$$

The transpose of this is

$$(\boldsymbol{u}\,\boldsymbol{v})^{\mathrm{T}} = \boldsymbol{v}\,\boldsymbol{u}$$

but

$$\boldsymbol{u}\,\boldsymbol{v} \neq \boldsymbol{v}\,\boldsymbol{u}$$

The Kronecker delta is defined as

$$\delta_{ij} = \left\{ \begin{array}{l} 1 \text{ if } i = j \\ 0 \text{ if } i \neq j \end{array} \right\}$$

and the unit dyadic is defined as

$$\delta = \sum_i \sum_j \boldsymbol{e}_i \boldsymbol{e}_j\, \delta_{ij}$$

Operations. The *dot* or *scalar product* of two vectors is defined as

$$\boldsymbol{u}\cdot\boldsymbol{v} = |\boldsymbol{u}|\,|\boldsymbol{v}| \cos\theta,\ 0 \leq \theta \leq \pi$$

where θ is the angle between $\boldsymbol{u}$ *and* $\boldsymbol{v}$. The scalar product of two vectors is a scalar, not a vector. It is the magnitude of $\boldsymbol{u}$ multiplied by the projection of $\boldsymbol{v}$ on $\boldsymbol{u}$, or vice versa. The scalar product of $\boldsymbol{u}$ with itself is just the square of the magnitude of $\boldsymbol{u}$.

$$\boldsymbol{u}\cdot\boldsymbol{u} = \left|\boldsymbol{u}^2\right| = u^2$$

The following laws are valid for scalar products

$\boldsymbol{u}\cdot\boldsymbol{v} = \boldsymbol{v}\cdot\boldsymbol{u}$	Commutative law for scalar products
$\boldsymbol{u}\cdot(\boldsymbol{v}+\boldsymbol{w}) = \boldsymbol{u}\cdot\boldsymbol{v} + \boldsymbol{u}\cdot\boldsymbol{w}$	Distributive law for scalar products
$\boldsymbol{e}_x\cdot\boldsymbol{e}_x = \boldsymbol{e}_y\cdot\boldsymbol{e}_y = \boldsymbol{e}_z\cdot\boldsymbol{e}_z = 1$	
$\boldsymbol{e}_x\cdot\boldsymbol{e}_y = \boldsymbol{e}_x\cdot\boldsymbol{e}_z = \boldsymbol{e}_y\cdot\boldsymbol{e}_z = 0$	

If the two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ are written in component notation, the scalar product is

$$\boldsymbol{u}\cdot\boldsymbol{v} = u_x\, v_x + u_y\, v_y + u_z\, v_z$$

If $\boldsymbol{u}\cdot\boldsymbol{v} = 0$ and $\boldsymbol{u}$ and $\boldsymbol{v}$ are not null vectors, then $\boldsymbol{u}$ and $\boldsymbol{v}$ are perpendicular to each other and $\theta = \pi/2$.

The *single dot product* of two dyadics is

$$\boldsymbol{A}\cdot\boldsymbol{B} = \sum_i \sum_j \boldsymbol{e}_i \boldsymbol{e}_j \left(\sum_k A_{ik}\, B_{kj}\right)$$

The *double dot product* of two dyadics is

$$\boldsymbol{A}:\boldsymbol{B} = \sum_i \sum_j A_{ij}\, B_{ji}$$

Because the dyadics may not be symmetric, the order of indices and which indices are summed are important. The order is made clearer when the dyadics are made from vectors.

$$(\boldsymbol{u}\,\boldsymbol{v})\cdot(\boldsymbol{w}\,\boldsymbol{x}) = \boldsymbol{u}\,(\boldsymbol{v}\cdot\boldsymbol{w})\,\boldsymbol{x} = \boldsymbol{u}\,\boldsymbol{x}\,(\boldsymbol{v}\cdot\boldsymbol{w})$$

$$(\boldsymbol{u}\,\boldsymbol{v}):(\boldsymbol{w}\,\boldsymbol{x}) = \ (\boldsymbol{u}\cdot\boldsymbol{x})\,(\boldsymbol{v}\cdot\boldsymbol{w})$$

The dot product of a dyadic and a vector is

$$\boldsymbol{A}\cdot\boldsymbol{u} = \sum_i \boldsymbol{e}_i \left(\sum_j A_{ij}\, u_j\right)$$

The *cross* or *vector product* is defined by

$$\boldsymbol{c} = \boldsymbol{u}\times\boldsymbol{v} = \boldsymbol{a}\,|u|\,|v| \sin\theta,\ 0 \leq \theta \leq \pi$$

where $\boldsymbol{a}$ is a unit vector in the direction of $\boldsymbol{u}\times\boldsymbol{v}$. The direction of $\boldsymbol{c}$ is perpendicular to the plane of $\boldsymbol{u}$ and $\boldsymbol{v}$ such that $\boldsymbol{u}$, $\boldsymbol{v}$, and $\boldsymbol{c}$ form a right-handed system. If $\boldsymbol{u} = \boldsymbol{v}$, or $\boldsymbol{u}$ is parallel to $\boldsymbol{v}$, then $\theta = 0$ and $\boldsymbol{u}\times\boldsymbol{v} = 0$. The following laws are valid for cross products.

$\boldsymbol{u}\times\boldsymbol{v} = -\boldsymbol{v}\times\boldsymbol{u}$	Commutative law fails for vector product
$\boldsymbol{u}\times(\boldsymbol{v}\times\boldsymbol{w}) \neq (\boldsymbol{u}\times\boldsymbol{v})\times\boldsymbol{w}$	Associative law fails for vector product
$\boldsymbol{u}\times(\boldsymbol{v}+\boldsymbol{w}) = \boldsymbol{u}\times\boldsymbol{v} + \boldsymbol{u}\times\boldsymbol{w}$	Distributive law for vector product
$\boldsymbol{e}_x\times\boldsymbol{e}_x = \boldsymbol{e}_y\times\boldsymbol{e}_y = \boldsymbol{e}_z\times\boldsymbol{e}_z = 0$	
$\boldsymbol{e}_x\times\boldsymbol{e}_y = \boldsymbol{e}_z,\ \boldsymbol{e}_y\times\boldsymbol{e}_z = \boldsymbol{e}_x,\ \boldsymbol{e}_z\times\boldsymbol{e}_x = \boldsymbol{e}_y$	

$$\boldsymbol{u}\times\boldsymbol{v} = \det\begin{bmatrix} \boldsymbol{e}_x & \boldsymbol{e}_y & \boldsymbol{e}_z \\ u_x & u_y & u_z \\ v_x & v_y & v_z \end{bmatrix}$$

$$= \boldsymbol{e}_x\,(u_y\, v_z - v_y\, u_z) + e_y\,(u_z\, v_z - u_x\, v_z)$$

$$+ \boldsymbol{e}_z\,(u_x\, v_y - u_y\, v_x)$$

This can also be written as

$$\boldsymbol{u}\times\boldsymbol{v} = \sum_i \sum_j \varepsilon_{kij}\, u_i\, v_j \boldsymbol{e}_k$$

where

$$\varepsilon_{ijk} = \begin{cases} 1 & \text{if } i,j,k \text{ is an even permutation of } 123 \\ -1 & \text{if } i,j,k \text{ is an odd permutation of } 123 \\ 0 & \text{if any two of } i,j,k \text{ are equal} \end{cases}$$

Thus $\varepsilon_{123} = 1$, $\varepsilon_{132} = -1$, $\varepsilon_{312} = 1$, $\varepsilon_{112} = 0$, for example.

The magnitude of $\boldsymbol{u}\times\boldsymbol{v}$ is the same as the area of a parallelogram with sides $\boldsymbol{u}$ and $\boldsymbol{v}$. If $\boldsymbol{u}\times\boldsymbol{v} = \boldsymbol{0}$ and $\boldsymbol{u}$ *and* $\boldsymbol{v}$ are not null vectors, then $\boldsymbol{u}$ *and* $\boldsymbol{v}$ are parallel. Certain triple products are useful.

$$(\boldsymbol{u}\cdot\boldsymbol{v})\,\boldsymbol{w} \neq \boldsymbol{u}\,(\boldsymbol{v}\cdot\boldsymbol{w})$$

$$\boldsymbol{u}\cdot(\boldsymbol{v}\times\boldsymbol{w}) = \boldsymbol{v}\cdot(\boldsymbol{w}\times\boldsymbol{u}) = \boldsymbol{w}\cdot(\boldsymbol{u}\times\boldsymbol{v})$$

$$\boldsymbol{u}\times(\boldsymbol{v}\times\boldsymbol{w}) = (\boldsymbol{u}\cdot\boldsymbol{w})\,v - (\boldsymbol{u}\cdot\boldsymbol{v})\,\boldsymbol{w}$$

$$(\boldsymbol{u}\times\boldsymbol{v})\times\boldsymbol{w} = (\boldsymbol{u}\cdot\boldsymbol{w})\,v - (\boldsymbol{v}\cdot\boldsymbol{w})\,\boldsymbol{u}$$

The cross product of a dyadic and a vector is defined as

$$\boldsymbol{A}\times\boldsymbol{u} = \sum_i \sum_j \boldsymbol{e}_i \boldsymbol{e}_j \left(\sum_k \sum_l \varepsilon_{klj}\, A_{ik}\, u_l\right)$$

The magnitude of a dyadic is

$$|\boldsymbol{A}| = A = \sqrt{\frac{1}{2}\,(\boldsymbol{A}{:}\boldsymbol{A}^{\mathrm{T}})} = \sqrt{\frac{1}{2}\sum_i \sum_j A_{ij}^2}$$

There are three *invariants* of a dyadic. They are called invariants because they take the same value in any coordinate system and are thus an intrinsic property of the dyadic. They are the trace of $\boldsymbol{A}, \boldsymbol{A}^2, \boldsymbol{A}^3$ [45].

$$\mathrm{I} = \mathrm{tr}\boldsymbol{A} = \sum_i A_{ii}$$

$$\mathrm{II} = \mathrm{tr}\boldsymbol{A}^2 = \sum_i \sum_j A_{ij}\, A_{ji}$$

$$\mathrm{III} = \mathrm{tr}\boldsymbol{A}^3 = \sum_i \sum_j \sum_k A_{ij}\, A_{jk}\, A_{ki}$$

The invariants can also be expressed as

$$\mathrm{I}_1 = \mathrm{I}$$

$$\mathrm{I}_2 = \tfrac{1}{2}\,(\mathrm{I}^2 - \mathrm{II})$$

$$\mathrm{I}_3 = \tfrac{1}{6}\,(\mathrm{I}^3 - 3\,\mathrm{I}\cdot\mathrm{II} + 2\,\mathrm{III}) = \det \boldsymbol{A}$$

Invariants of two dyadics are available [46]. Because a second-order dyadic has nine components, the *characteristic equation*

$$\det\,(\lambda\,\boldsymbol{\delta} - \boldsymbol{A}) = 0$$

can be formed where λ is an eigenvalue. This expression is

$$\lambda^3 - I_1\lambda^2 + I_2\lambda - I_3 = 0$$

An important theorem of Hamilton and Cayley [47] is that a second-order dyadic satisfies its own characteristic equation.

$$\boldsymbol{A}^3 - I_1\boldsymbol{A}^2 + I_2\boldsymbol{A} - I_3\boldsymbol{\delta} = 0 \tag{7}$$

Thus $\boldsymbol{A}^3$ can be expressed in terms of $\boldsymbol{\delta}$, $\boldsymbol{A}$, and $\boldsymbol{A}^2$. Similarly, higher powers of $\boldsymbol{A}$ can be expressed in terms of $\boldsymbol{\delta}$, $\boldsymbol{A}$, and $\boldsymbol{A}^2$. Decomposition of a dyadic into a symmetric and an antisymmetric part was shown above. The antisymmetric part has zero trace. The symmetric part can be decomposed into a part with a trace (the isotropic part) and a part with zero trace (the deviatoric part).

$$\boldsymbol{A} = \underbrace{1/3\,\boldsymbol{\delta}\boldsymbol{\delta}:\boldsymbol{A}}_{\text{Isotropic}} + \underbrace{1/2\,[\boldsymbol{A} + \boldsymbol{A}^{\mathrm{T}} - 2/3\,\boldsymbol{\delta}\boldsymbol{\delta}:\boldsymbol{A}]}_{\text{Deviatoric}} + \underbrace{1/2\,[\boldsymbol{A} - \boldsymbol{A}^{\mathrm{T}}]}_{\text{Antisymmetric}}$$

Differentiation. The derivative of a vector is defined in the same way as the derivative of a scalar. Suppose the vector $\boldsymbol{u}$ depends on t. Then

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = \lim_{\Delta t\to 0} \frac{\boldsymbol{u}\,(t+\Delta t) - \boldsymbol{u}\,(t)}{\Delta t}$$

If the vector is the position vector $\boldsymbol{r}\,(t)$, then the difference expression is a vector in the direction of $\Delta\,\boldsymbol{r}$ (see Fig. 15). The derivative

$$\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}t} = \lim_{\Delta t\to 0} \frac{\Delta\boldsymbol{r}}{\Delta t} = \lim_{\Delta t\to 0} \frac{\boldsymbol{r}\,(t+\Delta t) - \boldsymbol{r}\,(t)}{\Delta t}$$

is the velocity. The derivative operation obeys the following laws.

$$\tfrac{\mathrm{d}}{\mathrm{d}t}\,(\boldsymbol{u}+\boldsymbol{v}) = \tfrac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} + \tfrac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t}$$

$$\tfrac{\mathrm{d}}{\mathrm{d}t}\,(\boldsymbol{u}\cdot\boldsymbol{v}) = \tfrac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t}\cdot\boldsymbol{v} + \boldsymbol{u}\cdot\tfrac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t}$$

$$\tfrac{\mathrm{d}}{\mathrm{d}t}\,(\boldsymbol{u}\times\boldsymbol{v}) = \tfrac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t}\times\boldsymbol{v} + \boldsymbol{u}\times\tfrac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t}$$

$$\tfrac{\mathrm{d}}{\mathrm{d}t}\,(\varphi\boldsymbol{u}) = \tfrac{\mathrm{d}\varphi}{\mathrm{d}t}\boldsymbol{u} + \varphi\tfrac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t}$$

If the vector $\boldsymbol{u}$ depends on more than one variable, such as x, y and z, partial derivatives are defined in the usual way. For example,

if $\boldsymbol{u}\,(x, y, z)$, then

$$\frac{\partial\boldsymbol{u}}{\partial x} = \lim_{\Delta t\to 0} \frac{\boldsymbol{u}(x+\Delta x,\, y,\, z) - \boldsymbol{u}(x,\, y,\, z)}{\Delta x}$$

Rules for differentiation of scalar and vector products are

$$\frac{\partial}{\partial x}(\boldsymbol{u}\cdot\boldsymbol{v}) = \frac{\partial \boldsymbol{u}}{\partial x}\cdot\boldsymbol{v}+\boldsymbol{u}\cdot\frac{\partial \boldsymbol{v}}{\partial x}$$

$$\frac{\partial}{\partial x}(\boldsymbol{u}\times\boldsymbol{v}) = \frac{\partial \boldsymbol{u}}{\partial x}\times\boldsymbol{v}+\boldsymbol{u}\times\frac{\partial \boldsymbol{v}}{\partial x}$$

Differentials of vectors are

$$\mathrm{d}\boldsymbol{u} = \mathrm{d}u_x\boldsymbol{e}_x+\mathrm{d}u_y\boldsymbol{e}_y+\mathrm{d}u_z\boldsymbol{e}_z$$

$$\mathrm{d}(\boldsymbol{u}\cdot\boldsymbol{v}) = \mathrm{d}\boldsymbol{u}\cdot\boldsymbol{v}+\boldsymbol{u}\cdot\mathrm{d}\boldsymbol{v}$$

$$\mathrm{d}(\boldsymbol{u}\times\boldsymbol{v}) = \mathrm{d}\boldsymbol{u}\times\boldsymbol{v}+\boldsymbol{u}\times\mathrm{d}\boldsymbol{v}$$

$$\mathrm{d}\boldsymbol{u} = \frac{\partial \boldsymbol{u}}{\partial x}\mathrm{d}x+\frac{\partial \boldsymbol{u}}{\partial y}\mathrm{d}y+\frac{\partial \boldsymbol{u}}{\partial z}\mathrm{d}z$$

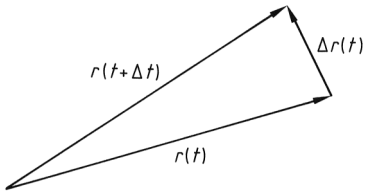

Figure 15. Vector differentiation

If a curve is given by $\boldsymbol{r}(t)$, the length of the curve is [43]

$$L = \int_a^b \sqrt{\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}t}\cdot\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}t}}\,\mathrm{d}t$$

The arc-length function can also be defined:

$$s(t) = \int_a^t \sqrt{\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}t^*}\cdot\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}t^*}}\,\mathrm{d}t^*$$

This gives

$$\left(\frac{\mathrm{d}s}{\mathrm{d}t}\right)^2 = \frac{\mathrm{d}r}{\mathrm{d}t}\cdot\frac{\mathrm{d}r}{\mathrm{d}t} = \left(\frac{\mathrm{d}x}{\mathrm{d}t}\right)^2+\left(\frac{\mathrm{d}y}{\mathrm{d}t}\right)^2+\left(\frac{\mathrm{d}z}{\mathrm{d}t}\right)^2$$

Because

$$d\boldsymbol{r} = \mathrm{d}x\boldsymbol{e}_x+\mathrm{d}y\boldsymbol{e}_y+\mathrm{d}z\boldsymbol{e}_z$$

then

$$ds^2 = \mathrm{d}\boldsymbol{r}\cdot\mathrm{d}\boldsymbol{r} = \mathrm{d}x^2+\mathrm{d}y^2+\mathrm{d}z^2$$

The derivative d$\boldsymbol{r}$/dt is tangent to the curve in the direction of motion

$$\boldsymbol{u} = \frac{\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}t}}{\left|\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}t}\right|}$$

Also,

$$\boldsymbol{u} = \frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}s}$$

Differential Operators. The vector differential operator (del operator) $\boldsymbol{\nabla}$ is defined in cartesian coordinates by

$$\boldsymbol{\nabla} = \boldsymbol{e}_x\frac{\partial}{\partial x}+\boldsymbol{e}_y\frac{\partial}{\partial y}+\boldsymbol{e}_z\frac{\partial}{\partial z}$$

The *gradient* of a scalar function is defined

$$\boldsymbol{\nabla}\varphi = \boldsymbol{e}_x\frac{\partial\varphi}{\partial x}+\boldsymbol{e}_y\frac{\partial\varphi}{\partial y}+\boldsymbol{e}_z\frac{\partial\varphi}{\partial z}$$

and is a vector. If φ is height or elevation, the gradient is a vector pointing in the uphill direction. The steeper the hill, the larger is the magnitude of the gradient.

The *divergence* of a vector is defined by

$$\boldsymbol{\nabla}\cdot\boldsymbol{u} = \frac{\partial u_x}{\partial x}+\frac{\partial u_y}{\partial y}+\frac{\partial u_z}{\partial z}$$

and is a scalar. For a volume element ΔV, the net outflow of a vector $\boldsymbol{u}$ over the surface of the element is

$$\int_{\Delta S}\boldsymbol{u}\cdot\boldsymbol{n}\mathrm{d}S$$

This is related to the divergence by [48, p. 411]

$$\boldsymbol{\nabla}\cdot\boldsymbol{u} = \lim_{\Delta V\to 0}\frac{1}{\Delta V}\int_{\Delta S}\boldsymbol{u}\cdot\boldsymbol{n}\mathrm{d}S$$

Thus, the divergence is the net outflow per unit volume.

The *curl* of a vector is defined by

$$\boldsymbol{\nabla}\times\boldsymbol{u} = \left(\boldsymbol{e}_x\frac{\partial}{\partial x}+\boldsymbol{e}_y\frac{\partial}{\partial y}+\boldsymbol{e}_z\frac{\partial}{\partial z}\right)$$

$$\times(\boldsymbol{e}_x u_x+\boldsymbol{e}_y u_y+\boldsymbol{e}_z u_z)$$

$$= \boldsymbol{e}_x\left(\frac{\partial u_z}{\partial y}-\frac{\partial u_y}{\partial z}\right)+\boldsymbol{e}_y\left(\frac{\partial u_x}{\partial z}-\frac{\partial u_z}{\partial x}\right)$$

$$+\boldsymbol{e}_z\left(\frac{\partial u_y}{\partial x}-\frac{\partial u_x}{\partial y}\right)$$

and is a vector. It is related to the integral

$$\int_C\boldsymbol{u}\cdot\mathrm{d}\boldsymbol{s} = \int_C u_s\mathrm{d}s$$

which is called the circulation of $\boldsymbol{u}$ around path C. This integral depends on the vector and the contour C, in general. If the circulation does not depend on the contour C, the vector is said to be irrotational; if it does, it is rotational. The relationship with the curl is [48, p. 419]

$$\boldsymbol{n}\cdot(\boldsymbol{\nabla}\times\boldsymbol{u}) = \lim_{\Delta\to 0}\frac{1}{\Delta S}\int_C \boldsymbol{u}\cdot\mathrm{d}\boldsymbol{s}$$

Thus, the normal component of the curl equals the net circulation per unit area enclosed by the contour C.

The gradient, divergence, and curl obey a distributive law but not a commutative or associative law.

$$\boldsymbol{\nabla}(\varphi+\psi) = \boldsymbol{\nabla}\varphi+\boldsymbol{\nabla}\psi$$

$$\boldsymbol{\nabla}\cdot(\boldsymbol{u}+\boldsymbol{v}) = \boldsymbol{\nabla}\cdot\boldsymbol{u}+\boldsymbol{\nabla}\cdot\boldsymbol{v}$$

$$\boldsymbol{\nabla}\times(\boldsymbol{u}+\boldsymbol{v}) = \boldsymbol{\nabla}\times\boldsymbol{u}+\boldsymbol{\nabla}\times\boldsymbol{v}$$

$$\boldsymbol{\nabla}\cdot\varphi \neq \varphi\boldsymbol{\nabla}$$

$$\boldsymbol{\nabla}\cdot\boldsymbol{u} \neq \boldsymbol{u}\cdot\boldsymbol{\nabla}$$

Useful formulas are [49]

$$\boldsymbol{\nabla}\cdot(\varphi\boldsymbol{u}) = \boldsymbol{\nabla}\varphi\cdot\boldsymbol{u}+\varphi\boldsymbol{\nabla}\cdot\boldsymbol{u}$$

$$\boldsymbol{\nabla}\times(\varphi\boldsymbol{u}) = \boldsymbol{\nabla}\varphi\times\boldsymbol{u}+\varphi\boldsymbol{\nabla}\times\boldsymbol{u}$$

$$\boldsymbol{\nabla}\cdot(\boldsymbol{u}\times\boldsymbol{v}) = \boldsymbol{v}\cdot(\boldsymbol{\nabla}\times\boldsymbol{u}) - \boldsymbol{u}\cdot(\boldsymbol{\nabla}\times\boldsymbol{v})$$

$$\boldsymbol{\nabla}\times(\boldsymbol{u}\times\boldsymbol{v}) = \boldsymbol{v}\cdot\boldsymbol{\nabla}\boldsymbol{u} - \boldsymbol{v}(\boldsymbol{\nabla}\cdot\boldsymbol{u}) - \boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{v} + \boldsymbol{u}(\boldsymbol{\nabla}\cdot\boldsymbol{v})$$

$$\boldsymbol{\nabla}\times(\boldsymbol{\nabla}\times\boldsymbol{u}) = \boldsymbol{\nabla}(\boldsymbol{\nabla}\cdot\boldsymbol{u}) - \nabla^2\boldsymbol{u}$$

$$\boldsymbol{\nabla}\cdot(\boldsymbol{\nabla}\varphi) = \nabla^2\varphi = \frac{\partial^2\varphi}{\partial x^2}+\frac{\partial^2\varphi}{\partial y^2}+\frac{\partial^2\varphi}{\partial z^2}, \text{ where } \nabla^2$$

is called the Laplacian operator. $\boldsymbol{\nabla}\times(\boldsymbol{\nabla}\varphi) = \boldsymbol{0}$. The curl of the gradient of φ is zero.

$$\boldsymbol{\nabla}\cdot(\boldsymbol{\nabla}\times\boldsymbol{u}) = 0$$

The divergence of the curl of $\boldsymbol{u}$ is zero. Formulas useful in fluid mechanics are

$$\boldsymbol{\nabla}\cdot(\boldsymbol{\nabla}\boldsymbol{v})^{\mathrm{T}} = \boldsymbol{\nabla}(\boldsymbol{\nabla}\cdot\boldsymbol{v})$$

$$\boldsymbol{\nabla}\cdot(\tau\cdot\boldsymbol{v}) = \boldsymbol{v}\cdot(\boldsymbol{\nabla}\cdot\tau)+\tau:\boldsymbol{\nabla}\boldsymbol{v}$$

$$\boldsymbol{v}\cdot\boldsymbol{\nabla}\boldsymbol{v} = \tfrac{1}{2}\boldsymbol{\nabla}(\boldsymbol{v}\cdot\boldsymbol{v}) - \boldsymbol{v}\times(\boldsymbol{\nabla}\times\boldsymbol{v})$$

If a coordinate system is transformed by a rotation and translation, the coordinates in the new system (denoted by primes) are given by

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} l_{11} & l_{12} & l_{13} \\ l_{21} & l_{22} & l_{23} \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

Any function that has the same value in all coordinate systems is an invariant. The gradient of an invariant scalar field is invariant; the same is true for the divergence and curl of invariant vectors fields.

The gradient of a vector field is required in fluid mechanics because the velocity gradient is used. It is defined as

$$\boldsymbol{\nabla}\boldsymbol{v} = \sum_i\sum_j \boldsymbol{e}_i\boldsymbol{e}_j\frac{\partial v_j}{\partial x_i} \text{ and}$$

$$(\boldsymbol{\nabla}\boldsymbol{v})^{\mathrm{T}} = \sum_i\sum_j \boldsymbol{e}_i\boldsymbol{e}_j\frac{\partial v_i}{\partial x_j}$$

The divergence of dyadics is defined

$$\boldsymbol{\nabla}\cdot\boldsymbol{\tau} = \sum_i \boldsymbol{e}_i\left(\sum_j \frac{\partial\tau_{ji}}{\partial x_j}\right) \text{ and}$$

$$\boldsymbol{\nabla}\cdot(\varphi\boldsymbol{u}\,\boldsymbol{v}) = \sum_i \boldsymbol{e}_i\left[\sum_j \frac{\partial}{\partial x_j}(\varphi u_j\, v_i)\right]$$

where $\boldsymbol{\tau}$ is any second-order dyadic.

Useful relations involving dyadics are

$$(\varphi\,\boldsymbol{\delta}:\boldsymbol{\nabla}\boldsymbol{v}) = \varphi(\boldsymbol{\nabla}\cdot\boldsymbol{v})$$

$$\boldsymbol{\nabla}\cdot(\varphi\,\boldsymbol{\delta}) = \boldsymbol{\nabla}\varphi$$

$$\boldsymbol{\nabla}\cdot(\varphi\boldsymbol{\tau}) = \boldsymbol{\nabla}\varphi\cdot\boldsymbol{\tau}+\varphi\boldsymbol{\nabla}\cdot\boldsymbol{\tau}$$

$$\boldsymbol{n}\,\boldsymbol{t}:\boldsymbol{\tau} = \boldsymbol{t}\cdot\boldsymbol{\tau}\cdot\boldsymbol{n} = \boldsymbol{\tau}:\boldsymbol{n}\,\boldsymbol{t}$$

A surface can be represented in the form

$$f(x,y,z) = c = \text{ constant}$$

The normal to the surface is given by

$$\boldsymbol{n} = \frac{\boldsymbol{\nabla} f}{|\boldsymbol{\nabla} f|}$$

provided the gradient is not zero. Operations can be performed entirely within the surface. Define

$$\boldsymbol{\delta}_{\mathrm{II}} \equiv \boldsymbol{\delta} - \boldsymbol{n}\,\boldsymbol{n}, \boldsymbol{\nabla}_{\mathrm{II}} \equiv \boldsymbol{\delta}_{\mathrm{II}}\cdot\boldsymbol{\nabla}, \frac{\partial}{\partial n} \equiv \boldsymbol{n}\cdot\boldsymbol{\nabla}$$

$$\boldsymbol{v}_{\mathrm{II}} \equiv \boldsymbol{\delta}_{\mathrm{II}}\cdot\boldsymbol{v}, v_n \equiv \boldsymbol{n}\cdot\boldsymbol{v}$$

Then a vector and del operator can be decomposed into

$$\boldsymbol{v} = \boldsymbol{v}_{\mathrm{II}}+\boldsymbol{n}v_n, \boldsymbol{\nabla} = \boldsymbol{\nabla}_{\mathrm{II}}+\boldsymbol{n}\frac{\partial}{\partial n}$$

The velocity gradient can be decomposed into

$$\boldsymbol{\nabla}\boldsymbol{v} = \boldsymbol{\nabla}_{\mathrm{II}}\boldsymbol{v}_{\mathrm{II}} + (\boldsymbol{\nabla}_{\mathrm{II}}\boldsymbol{n})\,v_n + \boldsymbol{n}\boldsymbol{\nabla}_{\mathrm{II}}v_n + \boldsymbol{n}\frac{\partial\boldsymbol{v}_{\mathrm{II}}}{\partial n} + \boldsymbol{n}\,\boldsymbol{n}\frac{\partial v_n}{\partial n}$$

The surface gradient of the normal is the negative of the curvature dyadic of the surface.

$$\boldsymbol{\nabla}_{\mathrm{II}}\boldsymbol{n} = -\boldsymbol{B}$$

The surface divergence is then

$$\boldsymbol{\nabla}_{\mathrm{II}}\cdot\boldsymbol{v} = \boldsymbol{\delta}_{\mathrm{II}} : \boldsymbol{\nabla}\boldsymbol{v} = \boldsymbol{\nabla}_{\mathrm{II}}\cdot\boldsymbol{v}_{\mathrm{II}} - 2\,H\,v_n$$

where H is the mean curvature.

$$H = \frac{1}{2}\boldsymbol{\delta}_{\mathrm{II}} : \boldsymbol{B}$$

The surface curl can be a scalar

$$\boldsymbol{\nabla}_{\mathrm{II}} \times \boldsymbol{v} = -\boldsymbol{\varepsilon}_{\mathrm{II}} : \boldsymbol{\nabla} \boldsymbol{v} = -\boldsymbol{\varepsilon}_{\mathrm{II}} : \boldsymbol{\nabla}_{\mathrm{II}} \boldsymbol{v}_{\mathrm{II}} = -\boldsymbol{n} \cdot (\boldsymbol{\nabla} \times \boldsymbol{v}),$$

$$\boldsymbol{\varepsilon}_{\mathrm{II}} = \boldsymbol{n} \cdot \boldsymbol{\varepsilon}$$

or a vector

$$\boldsymbol{\nabla}_{\mathrm{II}} \times \boldsymbol{v} \equiv \boldsymbol{\nabla}_{\mathrm{II}} \times \boldsymbol{v}_{\mathrm{II}} = \boldsymbol{n} \boldsymbol{\nabla}_{\mathrm{II}} \times \boldsymbol{v} = \boldsymbol{n}\, \boldsymbol{n} \cdot \boldsymbol{\nabla} \times v$$

Vector Integration [48, pp. 206 – 212]. If $\boldsymbol{u}$ is a vector, then its integral is also a vector.

$$\int \boldsymbol{u}(t)\,\mathrm{d}t = \boldsymbol{e}_x \int u_x(t)\,\mathrm{d}t + \boldsymbol{e}_y \int u_y(t)\,\mathrm{d}t + \boldsymbol{e}_z \int u_z(t)\,\mathrm{d}t$$

If the vector $\boldsymbol{u}$ is the derivative of another vector, then

$$\boldsymbol{u} = \frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t}, \int \boldsymbol{u}(t)\,\mathrm{d}t = \int \frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t}\,\mathrm{d}t = \boldsymbol{v} + \text{constant}$$

If $\boldsymbol{r}(t)$ is a position vector that defines a curve C, the line integral is defined by

$$\int_C \boldsymbol{u} \cdot \mathrm{d}\boldsymbol{r} = \int_C (u_x \mathrm{d}x + u_y \mathrm{d}y + u_z \mathrm{d}z)$$

Theorems about this line integral can be written in various forms.

Theorem [43]. If the functions appearing in the line integral are continuous in a domain D, then the line integral is independent of the path C if and only if the line integral is zero on every simple closed path in D.

Theorem [43]. If $\boldsymbol{u} = \boldsymbol{\nabla}\varphi$ where φ is single-valued and has continuous derivatives in D, then the line integral is independent of the path C and the line integral is zero for any closed curve in D.

Theorem [43]. If f, g, and h are continuous functions of x, y, and z, and have continuous first derivatives in a simply connected domain D, then the line integral

$$\int_C (f\,\mathrm{d}x + g\mathrm{d}y + h\mathrm{d}z)$$

is independent of the path if and only if

$$\frac{\partial h}{\partial y} = \frac{\partial g}{\partial z}, \frac{\partial f}{\partial z} = \frac{\partial h}{\partial x}, \frac{\partial g}{\partial x} = \frac{\partial f}{\partial y}$$

or if f, g, and h are regarded as the x, y, and z components of a vector $\boldsymbol{v}$:

$$\boldsymbol{\nabla} \times \boldsymbol{v} = 0$$

Consequently, the line integral is independent of the path (and the value is zero for a closed contour) if the three components in it are regarded as the three components of a vector and the vector is derivable from a potential (or zero curl). The conditions for a vector to be derivable from a potential are just those in the third theorem. In two dimensions this reduces to the more usual theorem.

Theorem [48, p. 207]. If M and N are continuous functions of x and y that have continuous first partial derivatives in a simply connected domain D, then the necessary and sufficient condition for the line integral

$$\int_C (M\mathrm{d}x + N\mathrm{d}y)$$

to be zero around every closed curve C in D is

$$\frac{\partial M}{\partial y} = \frac{\partial N}{\partial x}$$

If a vector is integrated over a surface with incremental area $\mathrm{d}S$ and normal to the surface $\boldsymbol{n}$, then the surface integral can be written as

$$\int\int_S \boldsymbol{u} \cdot \mathrm{d}\boldsymbol{S} = \int\int_S \boldsymbol{u} \cdot \boldsymbol{n} \mathrm{d}S$$

If $\boldsymbol{u}$ is the velocity then this integral represents the flow rate past the surface S.

Divergence Theorem [48, 49]. If V is a volume bounded by a closed surface S and $\boldsymbol{u}$ is a vector function of position with continuous derivatives, then

$$\int_V \boldsymbol{\nabla} \cdot \boldsymbol{u} \mathrm{d}V = \int_S \boldsymbol{n} \cdot \boldsymbol{u} \mathrm{d}S = \int_S \boldsymbol{u} \cdot \boldsymbol{n} \mathrm{d}S = \int_S \boldsymbol{u} \cdot \mathrm{d}\boldsymbol{S}$$

where $\boldsymbol{n}$ is the normal pointing outward to S. The normal can be written as

$$\boldsymbol{n} = \boldsymbol{e}_x \cos(x, n) + \boldsymbol{e}_y \cos(y, n) + \boldsymbol{e}_z \cos(z, n)$$

where, for example, cos (x, n) is the cosine of the angle between the normal n and the x axis. Then the divergence theorem in component form is

$$\int_V \left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z} \right) \mathrm{d}x \mathrm{d}y \mathrm{d}z =$$

$$\int_S [u_x \cos(x, n) + u_y \cos(y, n) + u_z \cos(z, n)]\,\mathrm{d}S$$

If the divergence theorem is written for an incremental volume

$$\boldsymbol{\nabla}\cdot\boldsymbol{u} = \lim_{\Delta V\to 0}\frac{1}{\Delta V}\int_{\Delta S} u_n \mathrm{d}S$$

the divergence of a vector can be called the integral of that quantity over the area of a closed volume, divided by the volume. If the vector represents the flow of energy and the divergence is positive at a point *P*, then either a source of energy is present at *P* or energy is leaving the region around *P* so that its temperature is decreasing. If the vector represents the flow of mass and the divergence is positive at a point *P*, then either a source of mass exists at *P* or the density is decreasing at the point *P*. For an incompressible fluid the divergence is zero and the rate at which fluid is introduced into a volume must equal the rate at which it is removed.

Various theorems follow from the divergence theorem.

Theorem. If φ is a solution to Laplace's equation

$$\nabla^2\varphi = 0$$

in a domain D, and the second partial derivatives of φ are continuous in D, then the integral of the normal derivative of φ over any piecewise smooth closed orientable surface *S* in D is zero. Suppose $\boldsymbol{u} = \varphi\,\boldsymbol{\nabla}\psi$ satisfies the conditions of the divergence theorem: then *Green's theorem* results from use of the divergence theorem [49].

$$\int_V \left(\varphi\nabla^2\psi + \boldsymbol{\nabla}\varphi\cdot\boldsymbol{\nabla}\psi\right)\mathrm{d}V = \int_S \varphi\frac{\partial\psi}{\partial n}\mathrm{d}S$$

and

$$\int_V \left(\varphi\nabla^2\psi - \psi\nabla^2\varphi\right)\mathrm{d}V = \int_S \left(\varphi\frac{\partial\psi}{\partial n} - \psi\frac{\partial\varphi}{\partial n}\right)\mathrm{d}S$$

Also if φ satisfies the conditions of the theorem and is zero on *S* then φ is zero throughout D. If two functions φ and ψ both satisfy the Laplace equation in domain D, and both take the same values on the bounding curve *C*, then $\varphi = \psi$; i.e., the solution to the Laplace equation is unique.

The divergence theorem for dyadics is

$$\int_V \boldsymbol{\nabla}\cdot\boldsymbol{\tau}\mathrm{d}V = \int_S \boldsymbol{n}\cdot\boldsymbol{\tau}\,\mathrm{d}S$$

Stokes Theorem [48, 49]. Stokes theorem says that if *S* is a surface bounded by a closed, nonintersecting curve *C*, and if $\boldsymbol{u}$ has continuous derivatives then

$$\oint_C \boldsymbol{u}\cdot\mathrm{d}\boldsymbol{r} = \int\int_S (\boldsymbol{\nabla}\times\boldsymbol{u})\cdot\boldsymbol{n}\mathrm{d}S = \int\int_S (\boldsymbol{\nabla}\times\boldsymbol{u})\cdot\mathrm{d}\boldsymbol{S}$$

The integral around the curve is followed in the counterclockwise direction. In component notation, this is

$$\begin{aligned}&\oint_C \left[u_x\cos\,(x,s) + u_y\cos\,(y,s) + u_z\cos\,(z,s)\right]\mathrm{d}s = \\ &\int\int_S \left[\left(\frac{\partial u_z}{\partial y} - \frac{\partial u_y}{\partial z}\right)\cos(x,n)\right. \\ &+ \left(\frac{\partial u_x}{\partial z} - \frac{\partial u_z}{\partial x}\right)\cos\,(y,n) \\ &\left.+ \left(\frac{\partial u_y}{\partial x} - \frac{\partial u_x}{\partial y}\right)\cos(z,n)\right]\mathrm{d}S\end{aligned}$$

Applied in two dimensions, this results in Green's theorem in the plane:

$$\oint_C (M\mathrm{d}x + N\mathrm{d}y) = \int\int_S \left(\frac{\partial N}{\partial x} - \frac{\partial M}{\partial y}\right)\mathrm{d}x\mathrm{d}y$$

The formula for dyadics is

$$\int\int_S \boldsymbol{n}\cdot(\boldsymbol{\nabla}\times\boldsymbol{\tau})\,\mathrm{d}S = \oint_C \boldsymbol{\tau}^{\mathrm{T}}\cdot\mathrm{d}\boldsymbol{r}$$

Representation. Two theorems give information about how to represent vectors that obey certain properties.

Theorem [48*, p. 422].* The necessary and sufficient condition that the curl of a vector vanish identically is that the vector be the gradient of some function.

Theorem [48*, p. 423].* The necessary and sufficient condition that the divergence of a vector vanish identically is that the vector is the curl of some other vector.

Leibniz Formula. In fluid mechanics and transport phenomena, an important result is the derivative of an integral whose limits of integration are moving. Suppose the region *V* (*t*) is moving with velocity $\boldsymbol{v}_s$. Then Leibniz's rule holds:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int\int\int_{V(t)} \varphi\mathrm{d}V = \int\int\int_{V(t)} \frac{\partial\varphi}{\partial t}\mathrm{d}V \quad + \int\int_S \varphi\boldsymbol{v}_s\cdot\boldsymbol{n}\mathrm{d}S$$

Curvilinear Coordinates. Many of the relations given above are proved most easily by using tensor analysis rather than dyadics. Once proven, however, the relations are perfectly general in any coordinate system. Displayed here are the specific results for cylindrical and spherical geometries. Results are available for a few other geometries: parabolic cylindrical, paraboloidal, elliptic cylindrical, prolate spheroidal, oblate spheroidal, ellipsoidal, and bipolar coordinates [45, 50].

For *cylindrical coordinates*, the geometry is shown in Figure 16. The coordinates are related to cartesian coordinates by

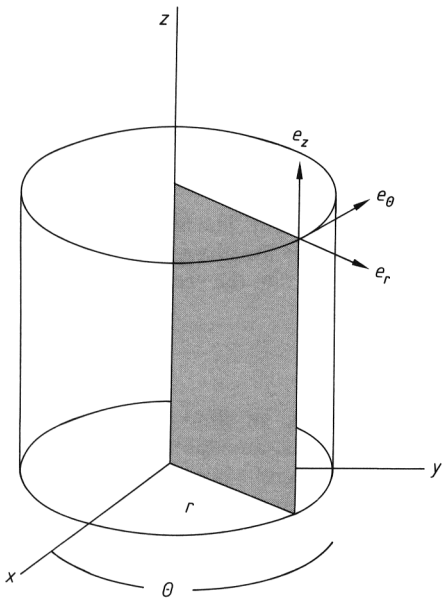

Figure 16. Cylindrical coordinate system

$$x = r\cos\theta \qquad r = \sqrt{x^2+y^2}$$
$$y = r\sin\theta \qquad \theta = \arctan\left(\tfrac{y}{x}\right)$$
$$z = z \qquad z = z$$

The unit vectors are related by

$$\boldsymbol{e}_r = \cos\theta\boldsymbol{e}_x + \sin\theta\boldsymbol{e}_y \qquad \boldsymbol{e}_x = \cos\theta\boldsymbol{e}_r - \sin\theta\boldsymbol{e}_\theta$$
$$\boldsymbol{e}_\theta = -\sin\theta\boldsymbol{e}_x + \cos\theta\boldsymbol{e}_y \qquad \boldsymbol{e}_y = \sin\theta\boldsymbol{e}_r + \cos\theta\boldsymbol{e}_\theta$$
$$\boldsymbol{e}_z = \boldsymbol{e}_z \qquad \boldsymbol{e}_z = \boldsymbol{e}_z$$

Derivatives of the unit vectors are

$$\mathrm{d}\boldsymbol{e}_\theta = -\boldsymbol{e}_r\mathrm{d}\theta,\ \mathrm{d}\boldsymbol{e}_r = \boldsymbol{e}_\theta\mathrm{d}\theta,\ \mathrm{d}\boldsymbol{e}_z = 0$$

Differential operators are given by [45]

$$\boldsymbol{\nabla} = \boldsymbol{e}_r\frac{\partial}{\partial r} + \frac{\boldsymbol{e}_\theta}{r}\frac{\partial}{\partial\theta} + \boldsymbol{e}_z\frac{\partial}{\partial z},$$

$$\boldsymbol{\nabla}\varphi = \boldsymbol{e}_r\frac{\partial\varphi}{\partial r} + \frac{\boldsymbol{e}_\theta}{r}\frac{\partial\varphi}{\partial\theta} + \boldsymbol{e}_z\frac{\partial\varphi}{\partial z}$$

$$\boldsymbol{\nabla}^2\varphi = \tfrac{1}{r}\tfrac{\partial}{\partial r}\left(r\tfrac{\partial\varphi}{\partial r}\right) + \tfrac{1}{r^2}\tfrac{\partial^2\varphi}{\partial\theta^2} + \tfrac{\partial^2\varphi}{\partial z^2}$$

$$= \tfrac{\partial^2\varphi}{\partial r^2} + \tfrac{1}{r}\tfrac{\partial\varphi}{\partial r} + \tfrac{1}{r^2}\tfrac{\partial^2\varphi}{\partial\theta^2} + \tfrac{\partial^2\varphi}{\partial z^2}$$

$$\boldsymbol{\nabla}\cdot\boldsymbol{v} = \frac{1}{r}\frac{\partial}{\partial r}(r\,v_r) + \frac{1}{r}\frac{\partial v_\theta}{\partial\theta} + \frac{\partial v_z}{\partial z}$$

$$\boldsymbol{\nabla}\times\boldsymbol{v} = \boldsymbol{e}_r\left(\tfrac{1}{r}\tfrac{\partial v_z}{\partial\theta} - \tfrac{\partial v_\theta}{\partial z}\right) + \boldsymbol{e}_\theta\left(\tfrac{\partial v_r}{\partial z} - \tfrac{\partial v_z}{\partial r}\right)$$

$$+\boldsymbol{e}_z\left[\tfrac{1}{r}\tfrac{\partial}{\partial r}(r\,v_\theta) - \tfrac{1}{r}\tfrac{\partial v_r}{\partial\theta}\right]$$

$$\boldsymbol{\nabla}\cdot\boldsymbol{\tau} = \boldsymbol{e}_r\left[\tfrac{1}{r}\tfrac{\partial}{\partial r}(r\tau_{rr}) + \tfrac{1}{r}\tfrac{\partial\tau_{\theta r}}{\partial\theta} + \tfrac{\partial\tau_{zr}}{\partial z} - \tfrac{\tau_{\theta\theta}}{r}\right] +$$

$$+\boldsymbol{e}_\theta\left[\tfrac{1}{r^2}\tfrac{\partial}{\partial r}(r^2\tau_{r\theta}) + \tfrac{1}{r}\tfrac{\partial\tau_{\theta\theta}}{\partial\theta} + \tfrac{\partial\tau_{z\theta}}{\partial z} + \tfrac{\tau_{\theta r}-\tau_{r\theta}}{r}\right] +$$

$$+\boldsymbol{e}_z\left[\tfrac{1}{r}\tfrac{\partial}{\partial r}(r\tau_{rz}) + \tfrac{1}{r}\tfrac{\partial\tau_{\theta z}}{\partial\theta} + \tfrac{\partial\tau_{zz}}{\partial z}\right]$$

$$\boldsymbol{\nabla}\boldsymbol{v} = \boldsymbol{e}_r\boldsymbol{e}_r\tfrac{\partial v_r}{\partial r} + \boldsymbol{e}_r\boldsymbol{e}_\theta\tfrac{\partial v_\theta}{\partial r} + \boldsymbol{e}_r\boldsymbol{e}_z\tfrac{\partial v_z}{\partial r} +$$

$$+\boldsymbol{e}_\theta\boldsymbol{e}_r\left(\tfrac{1}{r}\tfrac{\partial v_r}{\partial\theta} - \tfrac{v_\theta}{r}\right) + \boldsymbol{e}_\theta\boldsymbol{e}_\theta\left(\tfrac{1}{r}\tfrac{\partial v_\theta}{\partial\theta} + \tfrac{v_r}{r}\right) +$$

$$+\boldsymbol{e}_\theta\boldsymbol{e}_z\tfrac{1}{r}\tfrac{\partial v_z}{\partial\theta} + \boldsymbol{e}_z\boldsymbol{e}_r\tfrac{\partial v_r}{\partial z} + \boldsymbol{e}_z\boldsymbol{e}_\theta\tfrac{\partial v_\theta}{\partial z} + \boldsymbol{e}_z\boldsymbol{e}_z\tfrac{\partial v_z}{\partial z}$$

$$\nabla^2\boldsymbol{v} = \boldsymbol{e}_r\left[\tfrac{\partial}{\partial r}\left(\tfrac{1}{r}\tfrac{\partial}{\partial r}(r\,v_r)\right) + \tfrac{1}{r^2}\tfrac{\partial^2 v_r}{\partial\theta^2} + \tfrac{\partial^2 v_r}{\partial z^2}\right.$$

$$\left.-\tfrac{2}{r^2}\tfrac{\partial v_\theta}{\partial\theta}\right] +$$

$$+\boldsymbol{e}_\theta\left[\tfrac{\partial}{\partial r}\left(\tfrac{1}{r}\tfrac{\partial}{\partial r}(r\,v_\theta)\right) + \tfrac{1}{r^2}\tfrac{\partial^2 v_\theta}{\partial\theta^2} + \tfrac{\partial^2 v_\theta}{\partial z^2} + \tfrac{2}{r^2}\tfrac{\partial v_r}{\partial\theta}\right] +$$

$$+\boldsymbol{e}_z\left[\tfrac{1}{r}\tfrac{\partial}{\partial r}\left(r\tfrac{\partial v_z}{\partial r}\right) + \tfrac{1}{r^2}\tfrac{\partial^2 v_z}{\partial\theta^2} + \tfrac{\partial^2 v_z}{\partial z^2}\right]$$

For *spherical coordinates*, the geometry is shown in Figure 17. The coordinates are related to cartesian coordinates by

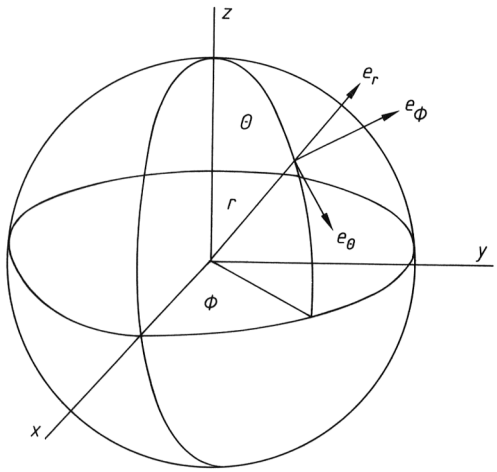

Figure 17. Spherical coordinate system

$x = r\sin\theta\cos\varphi \qquad r = \sqrt{x^2+y^2+z^2}$

$y = r\sin\theta\sin\varphi \qquad \theta = \arctan\left(\sqrt{x^2+y^2}/z\right)$

$z = r\cos\theta \qquad \varphi = \arctan\left(\frac{y}{x}\right)$

The unit vectors are related by

$$\boldsymbol{e}_r = \sin\theta\cos\varphi\boldsymbol{e}_x + \sin\theta\sin\varphi\boldsymbol{e}_y + \cos\theta\boldsymbol{e}_z$$
$$\boldsymbol{e}_\theta = \cos\theta\cos\varphi\boldsymbol{e}_x + \cos\theta\sin\varphi\boldsymbol{e}_y - \sin\theta\boldsymbol{e}_z$$
$$\boldsymbol{e}_\phi = -\sin\varphi\boldsymbol{e}_x + \cos\varphi\boldsymbol{e}_y$$
$$\boldsymbol{e}_x = \sin\theta\cos\varphi\boldsymbol{e}_r + \cos\theta\cos\varphi\boldsymbol{e}_\theta - \sin\varphi\boldsymbol{e}_\phi$$
$$\boldsymbol{e}_y = \sin\theta\sin\varphi\boldsymbol{e}_r + \cos\theta\sin\varphi\boldsymbol{e}_\theta + \cos\varphi\boldsymbol{e}_\phi$$
$$\boldsymbol{e}_z = \cos\theta\boldsymbol{e}_r - \sin\theta\boldsymbol{e}_\theta$$

Derivatives of the unit vectors are

$$\frac{\partial\boldsymbol{e}_r}{\partial\theta} = \boldsymbol{e}_\theta,\ \frac{\partial\boldsymbol{e}_\theta}{\partial\theta} = -\boldsymbol{e}_r$$

$$\frac{\partial\boldsymbol{e}_r}{\partial\varphi} = e_\phi\sin\theta,\ \frac{\partial\boldsymbol{e}_\theta}{\partial\varphi} = \boldsymbol{e}_\phi\cos\theta,$$

$$\frac{\partial\boldsymbol{e}_\phi}{\partial\varphi} = -\boldsymbol{e}_r\sin\theta - \boldsymbol{e}_\theta\cos\theta$$

Others 0

Differential operators are given by [45]

$$\boldsymbol{\nabla} = \boldsymbol{e}_r\frac{\partial}{\partial r} + \boldsymbol{e}_\theta\frac{1}{r}\frac{\partial}{\partial\theta} + \boldsymbol{e}_\phi\frac{1}{r\sin\theta}\frac{\partial}{\partial\varphi},$$

$$\boldsymbol{\nabla}\psi = \boldsymbol{e}_r\frac{\partial\psi}{\partial r} + \boldsymbol{e}_\theta\frac{1}{r}\frac{\partial\psi}{\partial\theta} + \boldsymbol{e}_\phi\frac{1}{r\sin\theta}\frac{\partial\psi}{\partial\varphi}$$

$$\boldsymbol{\nabla}^2\psi = \tfrac{1}{r^2}\tfrac{\partial}{\partial r}\left(r^2\tfrac{\partial\psi}{\partial r}\right) + \tfrac{1}{r^2\sin\theta}\tfrac{\partial}{\partial\theta}\left(\sin\theta\,\tfrac{\partial\psi}{\partial\theta}\right) +$$
$$+\tfrac{1}{r^2\sin^2\theta}\tfrac{\partial^2\psi}{\partial\varphi^2}$$

$$\boldsymbol{\nabla}\cdot\boldsymbol{v} = \tfrac{1}{r^2}\tfrac{\partial}{\partial r}\left(r^2\,v_r\right) + \tfrac{1}{r\sin\theta}\tfrac{\partial}{\partial\theta}\left(v_\theta\sin\theta\right) +$$
$$+\tfrac{1}{r\sin\theta}\tfrac{\partial v_\phi}{\partial\varphi}$$

$$\boldsymbol{\nabla}\times\boldsymbol{v} = \boldsymbol{e}_r\left[\tfrac{1}{r\sin\theta}\tfrac{\partial}{\partial\theta}\left(v_\phi\sin\theta\right) - \tfrac{1}{r\sin\theta}\tfrac{\partial v_\theta}{\partial\varphi}\right] +$$
$$+\boldsymbol{e}_\theta\left[\tfrac{1}{r\sin\theta}\tfrac{\partial v_r}{\partial\varphi} - \tfrac{1}{r}\tfrac{\partial}{\partial r}\left(r\,v_\phi\right)\right] +$$
$$+\boldsymbol{e}_\phi\left[\tfrac{1}{r}\tfrac{\partial}{\partial r}\left(r\,v_\theta\right) - \tfrac{1}{r}\tfrac{\partial v_r}{\partial\theta}\right]$$

$$\boldsymbol{\nabla}\cdot\boldsymbol{\tau} = \boldsymbol{e}_r\left[\tfrac{1}{r^2}\tfrac{\partial}{\partial r}\left(r^2\tau_{rr}\right)\right.$$
$$\left.+\tfrac{1}{r\sin\theta}\tfrac{\partial}{\partial\theta}\left(\tau_{\theta r}\sin\theta\right) + \tfrac{1}{r\sin\theta}\tfrac{\partial\tau_{\phi r}}{\partial\varphi} - \tfrac{\tau_{\theta\theta}+\tau_{\phi\phi}}{r}\right] +$$
$$+\boldsymbol{e}_\theta\left[\tfrac{1}{r^3}\tfrac{\partial}{\partial r}\left(r^3\tau_{r\theta}\right) + \tfrac{1}{r\sin\theta}\tfrac{\partial}{\partial\theta}\left(\tau_{\theta\theta}\sin\theta\right) +\right.$$
$$\left.\tfrac{1}{r\sin\theta}\tfrac{\partial\tau_{\phi\theta}}{\partial\varphi} + \tfrac{\tau_{\theta r}-\tau_{r\theta}-\tau_{\phi\phi}\cot\theta}{r}\right] +$$
$$+\boldsymbol{e}_\phi\left[\tfrac{1}{r^3}\tfrac{\partial}{\partial r}\left(r^3\tau_{r\phi}\right) + \tfrac{1}{r\sin\theta}\tfrac{\partial}{\partial\theta}\left(\tau_{\theta\phi}\sin\theta\right) +\right.$$
$$\left.\tfrac{1}{r\sin\theta}\tfrac{\partial\tau_{\phi\phi}}{\partial\varphi} + \tfrac{\tau_{\phi r}-\tau_{r\phi}+\tau_{\phi\theta}\cot\theta}{r}\right]$$

$$\boldsymbol{\nabla}\boldsymbol{v} = \boldsymbol{e}_r\boldsymbol{e}_r\tfrac{\partial v_r}{\partial r} + \boldsymbol{e}_r\boldsymbol{e}_\theta\tfrac{\partial v_\theta}{\partial r} + \boldsymbol{e}_r\boldsymbol{e}_\phi\tfrac{\partial v_\phi}{\partial r} +$$
$$+\boldsymbol{e}_\theta\boldsymbol{e}_r\left(\tfrac{1}{r}\tfrac{\partial v_r}{\partial\theta} - \tfrac{v_\theta}{r}\right) + \boldsymbol{e}_\theta\boldsymbol{e}_\theta\left(\tfrac{1}{r}\tfrac{\partial v_\theta}{\partial\theta} + \tfrac{v_r}{r}\right) +$$
$$+\boldsymbol{e}_\theta\boldsymbol{e}_\phi\tfrac{1}{r}\tfrac{\partial v_\phi}{\partial\theta} + \boldsymbol{e}_\phi\boldsymbol{e}_r\left(\tfrac{1}{r\sin\theta}\tfrac{\partial v_r}{\partial\varphi} - \tfrac{v_\phi}{r}\right) +$$
$$+\boldsymbol{e}_\phi\boldsymbol{e}_\theta\left(\tfrac{1}{r\sin\theta}\tfrac{\partial v_\theta}{\partial\varphi} - \tfrac{v_\phi}{r}\cot\theta\right) +$$
$$+\boldsymbol{e}_\phi\boldsymbol{e}_\phi\left(\tfrac{1}{r\sin\theta}\tfrac{\partial v_\phi}{\partial\varphi} + \tfrac{v_r}{r} + \tfrac{v_\theta}{r}\cot\theta\right)$$

$$\nabla^2\boldsymbol{v} = \boldsymbol{e}_r\left[\tfrac{\partial}{\partial r}\left(\tfrac{1}{r^2}\tfrac{\partial}{\partial r}\left(r^2\,v_r\right)\right)\right.$$
$$+\tfrac{1}{r^2\sin\theta}\tfrac{\partial}{\partial\theta}\left(\sin\theta\tfrac{\partial v_r}{\partial\theta}\right)$$
$$\left.+\tfrac{1}{r^2\sin^2\theta}\tfrac{\partial^2 v_r}{\partial\varphi^2} - \tfrac{2}{r^2\sin\theta}\tfrac{\partial}{\partial\theta}\left(v_\theta\sin\theta\right) - \tfrac{2}{r^2\sin\theta}\tfrac{\partial v_\phi}{\partial\varphi}\right]$$
$$+\boldsymbol{e}_\theta\left[\tfrac{1}{r^2}\tfrac{\partial}{\partial r}\left(r^2\tfrac{\partial v_\theta}{\partial r}\right) + \tfrac{1}{r^2}\tfrac{\partial}{\partial\theta}\left(\tfrac{1}{\sin\theta}\tfrac{\partial}{\partial\theta}\left(v_\theta\sin\theta\right)\right)\right.$$
$$\left.+\tfrac{1}{r^2\sin^2\theta}\tfrac{\partial^2 v_\theta}{\partial\varphi^2} + \tfrac{2}{r^2}\tfrac{\partial v_r}{\partial\theta} - \tfrac{2}{r^2}\tfrac{\cot\theta}{\sin\theta}\tfrac{\partial v_\phi}{\partial\varphi}\right]$$
$$+\boldsymbol{e}_\phi\left[\tfrac{1}{r^2}\tfrac{\partial}{\partial r}\left(r^2\tfrac{\partial v_\phi}{\partial r}\right) + \tfrac{1}{r^2}\tfrac{\partial}{\partial\theta}\left(\tfrac{1}{\sin\theta}\tfrac{\partial}{\partial\theta}\left(v_\phi\sin\theta\right)\right)\right.$$
$$\left.+\tfrac{1}{r^2\sin^2\theta}\tfrac{\partial^2 v_\phi}{\partial\varphi^2} + \tfrac{2}{r^2\sin\theta}\tfrac{\partial v_r}{\partial\varphi} + \tfrac{2}{r^2}\tfrac{\cot\theta}{\sin\theta}\tfrac{\partial v_\theta}{\partial\varphi}\right]$$

6. Ordinary Differential Equations as Initial Value Problems

A differential equation for a function that depends on only one variable (often time) is called an ordinary differential equation. The general solution to the differential equation includes many possibilities; the boundary or initial conditions are required to specify which of those are desired. If all conditions are at one point, the problem is an initial value problem and can be integrated from that point on. If some of the conditions are available at one point and others at another point, the ordinary differential equations become two-point boundary value problems, which are treated in Chapter 7. Initial value problems as ordinary differential equations arise in control of lumped-parameter models, transient models of stirred tank reactors, polymerization reactions and plug-flow reactors, and generally in models where no spatial gradients occur in the unknowns.

6.1. Solution by Quadrature

When only one equation exists, even if it is nonlinear, solving it by quadrature may be possible. For

$$\frac{dy}{dt} = f(y)$$

$$y(0) = y_0$$

the problem can be separated

$$\frac{dy}{f(y)} = dt$$

and integrated:

$$\int_{y_0}^{y} \frac{dy'}{f(y')} = \int_0^t dt = t$$

If the quadrature can be performed analytically then the exact solution has been found.

For example, consider the kinetics problem with a second-order reaction.

$$\frac{dc}{dt} = -kc^2, c(0) = c_0$$

To find the function of the concentration versus time, the variables can be separated and integrated.

$$\frac{dc}{c^2} = -k dt,$$

$$-\frac{1}{c} = -kt + D$$

Application of the initial conditions gives the solution:

$$\frac{1}{c} = kt + \frac{1}{c_0}$$

For other ordinary differential equations an integrating factor is useful. Consider the problem governing a stirred tank with entering fluid having concentration c_{in} and flow rate F, as shown in Figure 18. The flow rate out is also F and the volume of the tank is V. If the tank is completely mixed, the concentration in the tank is c and the concentration of the fluid leaving the tank is also c. The differential equation is then

$$V\frac{dc}{dt} = F(c_{in} - c), c(0) = c_0$$

Upon rearrangement,

$$\frac{dc}{dt} + \frac{F}{V}c = \frac{F}{V}c_{in}$$

is obtained. An integrating factor is used to solve this equation. The integrating factor is a function that can be used to turn the left-hand side into an exact differential and can be found by using Fréchet differentials [51]. In this case,

$$\exp\left(\frac{Ft}{V}\right)\left[\frac{dc}{dt} + \frac{F}{V}c\right] = \frac{d}{dt}\left[\exp\left(\frac{Ft}{V}\right)c\right]$$

Thus, the differential equation can be written as

$$\frac{d}{dt}\left[\exp\left(\frac{Ft}{V}\right)c\right] = \exp\left(\frac{Ft}{V}\right)\left[\frac{F}{V}c_{in}\right]$$

This can be integrated once to give

$$\exp\left(\frac{Ft}{V}\right)c = c(0) + \frac{F}{V}\int_0^t \exp\left(\frac{Ft'}{V}\right)c_{in}(t')\,dt'$$

or

$$c(t) = \exp\left(-\frac{Ft}{V}\right)c_0$$

$$+\frac{F}{V}\int_0^t \exp\left(-\frac{F(t-t')}{V}\right)c_{in}(t')\,dt'$$

If the integral on the right-hand side can be calculated, the solution can be obtained analytically. If not, the numerical methods described in the next sections can be used. Laplace transforms can also be attempted. However, an analytic solution is so useful that quadrature and an integrating factor should be tried before resorting to numerical solution.

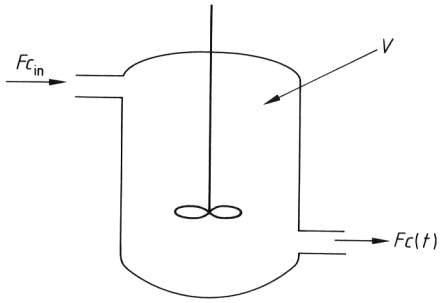

Figure 18. Stirred tank

6.2. Explicit Methods

Consider the ordinary differential equation

$$\frac{dy}{dt} = f(y)$$

Multiple equations that are still initial value problems can be handled by using the same techniques discussed here. A higher order differential equation

$$y^{(n)}+F\left(y^{(n-1)}, y^{(n-2)},\ldots, y', y\right)=0$$

with initial conditions

$$G_i\left(y^{(n-1)}(0), y^{(n-2)}(0),\ldots, y'(0), y(0)\right)=0$$

$$i=1,\ldots,n$$

can be converted into a set of first-order equations. By using

$$y_i \equiv y^{(i-1)}=\frac{\mathrm{d}^{(i-1)}y}{\mathrm{d}t^{(i-1)}}=\frac{\mathrm{d}}{\mathrm{d}t}y^{(i-2)}=\frac{\mathrm{d}y_{i-1}}{\mathrm{d}t}$$

the higher order equation can be written as a set of first-order equations:

$$\frac{\mathrm{d}y_1}{\mathrm{d}t}=y_2$$

$$\frac{\mathrm{d}y_2}{\mathrm{d}t}=y_3$$

$$\frac{\mathrm{d}y_3}{\mathrm{d}t}=y_4$$

$$\ldots$$

$$\frac{\mathrm{d}y_n}{\mathrm{d}t}=-F(y_{n-1}, y_{n-2},\ldots, y_2, y_1)$$

The initial conditions would have to be specified for variables $y_1(0), \ldots, y_n(0)$, or equivalently $y(0), \ldots, y^{(n-1)}(0)$. The set of equations is then written as

$$\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}t}=\boldsymbol{f}(\boldsymbol{y}, t)$$

All the methods in this chapter are described for a single equation; the methods apply to the multiple equations as well. Taking the single equation in the form

$$\frac{\mathrm{d}y}{\mathrm{d}t}=f(y)$$

multiplying by dt, and integrating once yields

$$\int_{t_n}^{t_{n+1}}\frac{\mathrm{d}y}{\mathrm{d}t'}\,\mathrm{d}t'=\sum_{t_n}^{t_{n+1}}f(y(t'))\,\mathrm{d}t'$$

This is

$$y^{n+1}=y^n+\int_{t_n}^{t_{n+1}}\frac{\mathrm{d}y}{\mathrm{d}t'}\,\mathrm{d}t'$$

The last substitution gives a basis for the various methods. Different interpolation schemes for $y(t)$ provide different integration schemes; using low-order interpolation gives low-order integration schemes [3].

Euler's method is first order

$$y^{n+1}=y^n+\Delta t f(y^n)$$

Adams – Bashforth Methods. The *second-order* Adams – Bashforth method is

$$y^{n+1}=y^n+\frac{\Delta t}{2}\left[3f(y^n)-f\left(y^{n-1}\right)\right]$$

The *fourth-order* Adams – Bashforth method is

$$y^{n+1}=y^n+\frac{\Delta t}{24}\left[55f(y^n)-59f\left(y^{n-1}\right)+37f\left(y^{n-2}\right)-9f\left(y^{n-3}\right)\right]$$

Notice that the higher order explicit methods require knowing the solution (or the right-hand side) evaluated at times in the past. Because these were calculated to get to the current time, this presents no problem except for starting the evaluation. Then, Euler's method may have to be used with a very small step size for several steps to generate starting values at a succession of time points. The error terms, order of the method, function evaluations per step, and stability limitations are listed in Table 6. The advantage of the fourth-order Adams – Bashforth method is that it uses only one function evaluation per step and yet achieves high-order accuracy. The disadvantage is the necessity of using another method to start.

Runge – Kutta Methods. Runge – Kutta methods are explicit methods that use several function evaluations for each time step. The general form of the methods is

$$y^{n+1}=y^n+\sum_{i=1}^{v}w_i k_i$$

with

$$k_i=\Delta t f\left(t^n+c_i\Delta t, y^n+\sum_{j=1}^{i-1}a_{ij}k_j\right)$$

Runge – Kutta methods traditionally have been writen for $f(t, y)$ and that is done here, too. If these equations are expanded and compared with a Taylor series, restrictions can be placed on the parameters of the method to make it first order, second order, etc. Even so, additional parameters can be chosen. A *second-order* Runge – Kutta method is

$$y^{n+1}=y^n+\frac{\Delta t}{2}\left[f^n+f(t^n+\Delta t, y^n+\Delta t f^n)\right]$$

The midpoint scheme is another second-order Runge – Kutta method:

Table 6. Properties of integration methods for ordinary differential equations

Method	Error term	Order	Function evaluations per step	Stability limit, $\lambda\,\Delta t \leq$
Explicit methods				
Euler	$\frac{h^2}{2}y''$	1	1	2.0
Second-order Adams – Bashforth	$\frac{5}{12}h^3y'''$	2	1	
Fourth-order Adams – Bashforth	$\frac{251}{720}h^5y^{(5)}$	4	1	0.3
Second-order Runge – Kutta (midpoint)		2	2	2.0
Runge – Kutta – Gill		4	4	2.8
Runge – Kutta – Feldberg	$y^{n+1}-z^{n+1}$	5	6	3.0
Predictor – corrector methods				
Second-order Runge – Kutta		2	2	2.0
Adams, fourth-order		2	2	1.3
Implicit methods, stability limit ∞				
Backward Euler		1	many, iterative	∞ *
Trapezoid rule	$-\frac{1}{12}h^3y'''$	2	many, iterative	2 *
Fourth-order Adams – Moulton		4	many, iterative	3 *

* Oscillation limit, $\lambda\,\Delta t \leq$.

$$y^{n+1} = y^n + \Delta t\, f\left(t^n + \frac{\Delta t}{2}, y^n + \frac{\Delta t}{2}\, f^n\right)$$

A popular *fourth-order* method is the Runge – Kutta – Gill method with the formulas

$$k_1 = \Delta t\, f\,(t^n, y^n)$$

$$k_2 = \Delta t\, f\left(t^n + \tfrac{\Delta t}{2}, y^n + \tfrac{k_1}{2}\right)$$

$$k_3 = \Delta t\, f\left(t^n + \tfrac{\Delta t}{2}, y^n + a\,k_1 + b\,k_2\right)$$

$$k_4 = \Delta t\, f\,(t^n + \Delta t, y^n + c\,k_2 + d\,k_3)$$

$$y^{n+1} = y^n + \tfrac{1}{6}\,(k_1 + k_4) + \tfrac{1}{3}\,(b\,k_2 + d\,k_3)$$

$$a = \frac{\sqrt{2}-1}{2}, b = \frac{2-\sqrt{2}}{2},$$

$$c = -\frac{\sqrt{2}}{2}, d = 1 + \frac{\sqrt{2}}{2}$$

Another fourth-order Runge – Kutta method is given by the Runge – Kutta – Feldberg formulas [52]; although the method is fourth-order, it achieves fifth-order accuracy. The popular integration package RKF 45 is based on this method.

$$k_1 = \Delta t\, f\,(t^n, y^n)$$

$$k_2 = \Delta t\, f\left(t^n + \tfrac{\Delta t}{4}, y^n + \tfrac{k_1}{4}\right)$$

$$k_3 = \Delta t\, f\left(t^n + \tfrac{3}{8}\Delta t, y^n + \tfrac{3}{32}k_1 + \tfrac{9}{32}k_2\right)$$

$$k_4 = \Delta t f\left(t^n + \tfrac{12}{13}\Delta t, y^n + \tfrac{1932}{2197}k_1 - \tfrac{7200}{2197}k_2 + \tfrac{7296}{2197}k_3\right)$$

$$k_5 = \Delta t f\left(t^n + \Delta t, y^n + \tfrac{439}{216}k_1 - 8\,k_2 + \tfrac{3680}{513}k_3 - \tfrac{845}{4104}k_4\right)$$

$$k_6 = \Delta t\, f\left(t^n + \tfrac{\Delta t}{2}, y^n - \tfrac{8}{27}k_1 + 2k_2 - \tfrac{3544}{2565}k_3 + \tfrac{1859}{4104}k_4 - \tfrac{11}{40}k_5\right)$$

$$y^{n+1} = y^n + \tfrac{25}{216}k_1 + \tfrac{1408}{2565}k_3 + \tfrac{2197}{4104}k_4 - \tfrac{1}{5}k_5$$

$$z^{n+1} = y^n + \tfrac{16}{135}k_1 + \tfrac{6656}{12\,825}k_3 + \tfrac{28\,561}{56\,430}k_4 - \tfrac{9}{50}k_5 + \tfrac{2}{55}k_6$$

The value of $y^{n+1} - z^{n+1}$ is an estimate of the error in y^{n+1} and can be used in step-size control schemes.

Generally, a high-order method should be used to achieve high accuracy. The Runge – Kutta – Gill method is popular because it is high order and does not require a starting method (as does the fourth-order Adams – Bashforth method). However, it requires four function evaluations per time step, or four times as many as the Adams – Bashforth method. For problems in which the function evaluations are a significant portion of the calculation time this might be important. Given the speed of computers and the widespread availability of desktop computers, the efficiency of a method is most important only for very large problems that are going to be solved many times. For other problems the most important criterion for choosing a method is probably the time the user spends setting up the problem.

The stability of an integration method is best estimated by determining the rational polynomial corresponding to the method. Apply this method to the equation

$$\frac{dy}{dt} = -\lambda y, y(0) = 1$$

and determine the formula for r_{mn}:

$$y^{k+1} = r_{mn}(\lambda \Delta t)\, y^k$$

The rational polynomial is defined as

$$r_{mn}(z) = \frac{p_n(z)}{q_m(z)} \approx e^{-z}$$

and is an approximation to $\exp(-z)$, called a Padé approximation. The stability limits are the largest positive z for which

$$|r_{mn}(z)| \leq 1$$

The method is *A* acceptable if the inequality holds for $Re\, z > 0$. It is *A* (0) acceptable if the inequality holds for z real, $z > 0$ [53]. The method will not induce oscillations about the true solution provided

$$r_{mn}(z) > 0$$

A method is *L* acceptable if it is *A* acceptable and

$$\lim_{z \to \infty} r_{mn}(z) = 0$$

For example, Euler's method gives

$$y^{n+1} = y^n - \lambda \Delta t y^n \text{ or } y^{n+1} = (1 - \lambda \Delta t)\, y^n$$

$$\text{or } r_{mn} = 1 - \lambda \Delta t$$

The stability limit is then

$$\lambda \Delta t \leq 2$$

The Euler method will not oscillate provided

$$\lambda \Delta t \leq 1$$

The stability limits listed in Table 6 are obtained in this fashion. The limit for the Euler method is 2.0; for the Runge – Kutta – Gill method it is 2.785; for the Runge – Kutta – Feldberg method it is 3.020. The rational polynomials for the various explicit methods are illustrated in Figure 19. As can be seen, the methods approximate the exact solution well as $\lambda \Delta t$ approaches zero, and the higher order methods give a better approximation at high values of $\lambda \Delta t$.

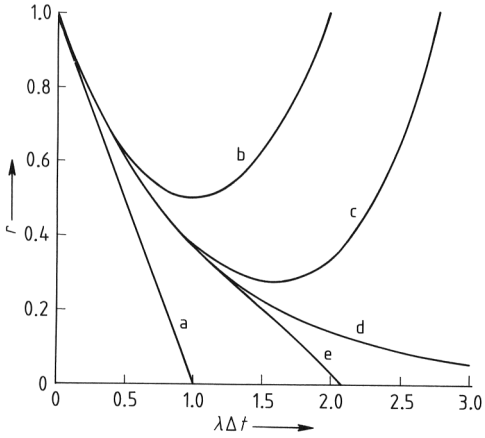

Figure 19. Rational approximations for explicit methods a) Euler; b) Runge – Kutta – 2; c) Runge – Kutta – Gill; d) Exact curve; e) Runge – Kutta – Feldberg

In solving sets of equations

$$\frac{dy}{dt} = \boldsymbol{A}\,\boldsymbol{y} + \boldsymbol{f}, \boldsymbol{y}(0) = y_0$$

all the eigenvalues of the matrix $\boldsymbol{A}$ must be examined. Finlayson [3] and Amundson [54, p. 197 – 199] both show how to transform these equations into an orthogonal form so that each equation becomes one equation in one unknown, for which single equation analysis applies. For linear problems the eigenvalues do not change, so the stability and oscillation limits must be satisfied for every eigenvalue of the matrix $\boldsymbol{A}$. When solving nonlinear problems the equations are linearized about the solution at the local time, and the analysis applies for small changes in time, after which a new analysis about the new solution must be made. Thus, for nonlinear problems the eigenvalues keep changing.

Richardson extrapolation can be used to improve the accuracy of a method. Step forward one step Δt with a p-th order method. Then redo the problem, this time stepping forward from the same initial point but in two steps of length $\Delta t/2$, thus ending at the same point. Call the solution of the one-step calculation y_1 and the solution of the two-step calculation y_2. Then an improved solution at the new time is given by

$$y = \frac{2^p y_2 - y_1}{2^p - 1}$$

This gives a good estimate provided Δt is small enough that the method is truly convergent with

order p. This process can also be repeated in the same way Romberg's method was used for quadrature (see Section 2.4).

The accuracy of a numerical calculation depends on the step size used, and this is chosen automatically by efficient codes. For example, in the Euler method the local truncation error LTE is

$$\text{LTE} = \frac{\Delta t^2}{2} y_n''$$

Yet the second derivative can be evaluated by using the difference formulas as

$$y_n'' = \nabla (\Delta t\, y_n') = \Delta t\ (y_n' - y_{n-1}') =$$

$$\Delta t\ (f_n - f_{n-1})$$

Thus, by monitoring the difference between the right-hand side from one time step to another, an estimate of the truncation error is obtained. This error can be reduced by reducing Δt. If the user specifies a criterion for the largest local error estimate, then Δt is reduced to meet that criterion. Also, Δt is increased to as large a value as possible, because this shortens computation time. If the local truncation error has been achieved (and estimated) by using a step size Δt_1

$$\text{LTE} = c\, \Delta t_1^p$$

and the desired error is ε, to be achieved using a step size Δt_2

$$\varepsilon = c\, \Delta t_2^p$$

then the next step size Δt_2 is taken from

$$\frac{\text{LTE}}{\varepsilon} = \left(\frac{\Delta t_1}{\Delta t_2} \right)^p$$

Generally, things should not be changed too often or too drastically. Thus one may choose not to increase Δt by more than a factor (such as 2) or to increase Δt more than once every so many steps (such as 5) [55]. In the most sophisticated codes the alternative exists to change the order of the method as well. In this case, the truncation error of the orders one higher and one lower than the current one are estimated, and a choice is made depending on the expected step size and work.

6.3. Implicit Methods

By using different interpolation formulas, involving y^{n+1}, implicit integration methods can be derived. Implicit methods result in a nonlinear equation to be solved for y^{n+1} so that iterative methods must be used. The backward Euler method is a first-order method:

$$y^{n+1} = y^n + \Delta t\, f\ (y^{n+1})$$

The trapezoid rule (see Section 2.4) is a second-order method:

$$y^{n+1} = y^n + \frac{\Delta t}{2} \left[f\ (y^n) + f\ (y^{n+1}) \right]$$

When the trapezoid rule is used with the finite difference method for solving partial differential equations it is called the Crank – Nicolson method. Adams methods exist as well, and the fourth-order Adams – Moulton method is

$$y^{n+1} = y^n + \frac{\Delta t}{24} \left[9 f\ (y^{n+1}) + 19 f\ (y^n) - 5 f\ (y^{n-1}) + f\ (y^{n-2}) \right]$$

The properties of these methods are given in Table 6. The implicit methods are stable for any step size but do require the solution of a set of nonlinear equations, which must be solved iteratively. An application to dynamic distillation problems is given in [56].

All these methods can be written in the form

$$y^{n+1} = \sum_{i=1}^{k} \alpha_i\, y^{n+1-i} + \Delta t \sum_{i=0}^{k} \beta_i\, f\ (y^{n+1-i})$$

or

$$y^{n+1} = \Delta t\, \beta_0\, f\ (y^{n+1}) + w^n$$

where w^n represents known information. This equation (or set of equations for more than one differential equation) can be solved by using successive substitution:

$$y^{n+1,\,k+1} = \Delta t\, \beta_0 f \left(y^{n+1,\,k} \right) + w^n$$

Here, the superscript k refers to an iteration counter. The successive substitution method is guaranteed to converge, provided the first derivative of the function is bounded and a small enough time step is chosen. Thus, if it has not converged within a few iterations, Δt can be reduced and the iterations begun again. The *Newton – Raphson method* (see Section 1.2) can also be used.

In many computer codes, iteration is allowed to proceed only a fixed number of times (e.g.,

three) before Δt is reduced. Because a good history of the function is available from previous time steps, a good initial guess is usually possible.

The best software packages for stiff equations (see Section 6.4) use Gear's backward difference formulas. The formulas of various orders are [57].

$$1: y^{n+1} = y^n + \Delta t\, f\ (y^{n+1})$$

$$2: y^{n+1} = \tfrac{4}{3}\, y^n - \tfrac{1}{3}\, y^{n-1} + \tfrac{2}{3}\, \Delta t\, f\ (y^{n+1})$$

$$3: y^{n+1} = \tfrac{18}{11}\, y^n - \tfrac{9}{11}\, y^{n-1} + \tfrac{2}{11}\, y^{n-2} + \tfrac{6}{11}\, \Delta t\, f\ (y^{n+1})$$

$$4: y^{n+1} = \tfrac{48}{25}\, y^n - \tfrac{36}{25}\, y^{n-1} + \tfrac{16}{25}\, y^{n-2} - \tfrac{3}{25}\, y^{n-3} + \tfrac{12}{25}\, \Delta t\, f\ (y^{n+1})$$

$$5: y^{n+1} = \tfrac{300}{137}\, y^n - \tfrac{300}{137}\, y^{n-1} + \tfrac{200}{137}\, y^{n-2} - \tfrac{75}{137}\, y^{n-3} + \tfrac{12}{137}\, y^{n-4} + \tfrac{60}{137}\, \Delta t\, f\ (y^{n+1})$$

The stability properties of these methods are determined in the same way as explicit methods. They are always expected to be stable, no matter what the value of Δt is, and this is confirmed in Figure 20.

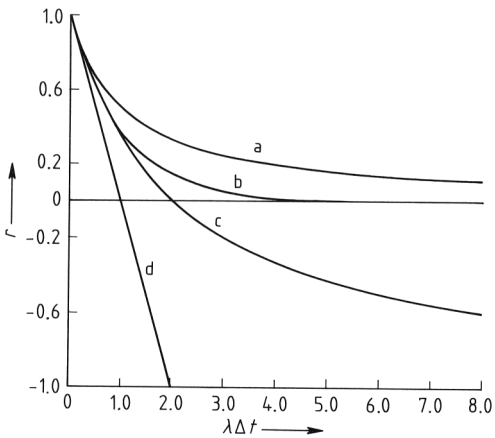

Figure 20. Rational approximations for implicit methods a) Backward Euler; b) Exact curve; c) Trapezoid; d) Euler

Predictor – corrector methods can be employed in which an explicit method is used to predict the value of y^{n+1}. This value is then used in an implicit method to evaluate $f\ (y^{n+1})$.

6.4. Stiffness

Why is it desirable to use implicit methods that lead to sets of algebraic equations that must be solved iteratively whereas explicit methods lead to a direct calculation? The reason lies in the stability limits; to understand their impact, the concept of stiffness is necessary. When modeling a physical situation, the time constants governing different phenomena should be examined. Consider flow through a packed bed, as illustrated in Figure 21.

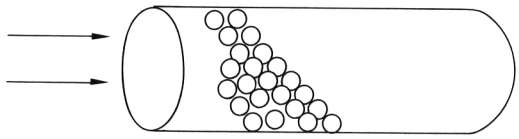

Figure 21. Flow through packed bed

The superficial velocity u is given by

$$u = \frac{Q}{A\,\varphi}$$

where Q is the volumetric flow rate, A is the cross-sectional area, and φ is the void fraction. A time constant for flow through the device is then

$$t_{\text{flow}} = \frac{L}{u} = \frac{\varphi\, A\, L}{Q}$$

where L is the length of the packed bed. If a chemical reaction occurs, with a reaction rate given by

$$\frac{\text{Moles}}{\text{Volume time}} = -k\, c$$

where k is the rate constant (time^{-1}) and c is the concentration (moles/volume), the characteristic time for the reaction is

$$t_{\text{rxn}} = \frac{1}{k}$$

If diffusion occurs inside the catalyst, the time constant is

$$t_{\text{internal diffusion}} = \frac{\varepsilon\, R^2}{D_{\text{e}}}$$

where ε is the porosity of the catalyst, R is the catalyst radius, and D_{e} is the effective diffusion

coefficient inside the catalyst. The time constant for heat transfer is

$$t_{\text{internal heat transfer}} = \frac{R^2}{\alpha} = \frac{\varrho_\text{s} C_\text{s} R^2}{k_\text{e}}$$

where ϱ_s is the catalyst density, C_s is the catalyst heat capacity per unit mass, k_e is the effective thermal conductivity of the catalyst, and α is the thermal diffusivity. The time constants for diffusion of mass and heat through a boundary layer surrounding the catalyst are

$$t_{\text{external diffusion}} = \frac{R}{k_\text{g}}$$

$$t_{\text{external heat transfer}} = \frac{\varrho_\text{s} C_\text{s} R}{h_\text{p}}$$

where k_g and h_p are the mass-transfer and heat-transfer coefficients, respectively. The importance of examining these time constants comes from realization that their orders of magnitude differ greatly. For example, in the model of an automobile catalytic converter [58] the time constant for internal diffusion was 0.3 s, for internal heat transfer 21 s, and for flow through the device 0.003 s. Flow through the device is so fast that it might as well be instantaneous. Thus, the time derivatives could be dropped from the mass balance equations for the flow, leading to a set of differential-algebraic equations (see below). If the original equations had to be solved, the eigenvalues would be roughly proportional to the inverse of the time constants. The time interval over which to integrate would be a small number (e.g., five) multiplied by the longest time constant. Yet the explicit stability limitation applies to all the eigenvalues, and the largest eigenvalue would determine the largest permissible time step. Here, $\lambda = 1/0.003\ \text{s}^{-1}$. Very small time steps would have to be used, e.g., $\Delta t \leq 2 \times 0.003$ s, but a long integration would be required to reach steady state. Such problems are termed stiff, and implicit methods are very useful for them. In that case the stable time constant is not of any interest, because any time step is stable. What is of interest is the largest step for which a solution can be found. If a time step larger than the smallest time constant is used, then any phenomena represented by that smallest time constant will be overlooked — at least transients in it will be smeared over. However, the method will still be stable. Thus, if the very rapid transients of part of the model are not of interest, they can be ignored and an implicit method used [59].

The idea of stiffness is best explained by considering a system of linear equations:

$$\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}t} = \boldsymbol{A}\,\boldsymbol{y}$$

Let λ_i be the eigenvalues of the matrix $\boldsymbol{A}$. This system can be converted into a system of n equations, each of them having only one unknown; the eigenvalues of the new system are the same as the eigenvalues of the original system [3, pp. 39 – 42], [54, pp. 197 – 199]. Then the stiffness ratio SR is defined as [53, p. 32]

$$\text{SR} = \frac{\max_i |\text{Re}\,(\lambda_i)|}{\min_i |\text{Re}\,(\lambda_i)|}$$

SR = 20 is not stiff, SR = 10^3 is stiff, and SR = 10^6 is very stiff. If the problem is nonlinear, the solution is expanded about the current state:

$$\frac{\mathrm{d}y_i}{\mathrm{d}t} = f_i\left[\boldsymbol{y}\left(t^n\right)\right] + \sum_{j=1}^{n} \frac{\partial f_i}{\partial y_j}\left[y_j - y_j\left(t^n\right)\right]$$

The question of stiffness then depends on the eigenvalue of the Jacobian at the current time. Consequently, for nonlinear problems the problem can be stiff during one time period and not stiff during another. Packages have been developed for problems such as these. Although the chemical engineer may not actually calculate the eigenvalues, knowing that they determine the stability and accuracy of the numerical scheme, as well as the step size employed, is useful.

6.5. Differential – Algebraic Systems

Sometimes models involve ordinary differential equations subject to some algebraic constraints. For example, the equations governing one equilibrium stage (as in a distillation column) are

$$M\frac{\mathrm{d}x^n}{\mathrm{d}t} = V^{n+1}\,y^{n+1} - L^n\,x^n - V^n\,y^n + L^{n-1}\,x^{n-1}$$

$$x^{n-1} - x^n = E^n\left(x^{n-1} - x^{*,n}\right)$$

$$\sum_{i=1}^{N} x_i = 1$$

where x and y are the mole fractions in the liquid and vapor, respectively; L and V are liquid and vapor flow rates, respectively; M is the holdup; and the superscript n is the stage number. The efficiency is E, and the concentration in equilibrium with the vapor is x^*. The first equation is

an ordinary differential equation for the mass of one component on the stage, whereas the third equation represents a constraint that the mass fractions add to one. As a second example, the following kinetics problem can be considered:

$$\frac{\mathrm{d}c_1}{\mathrm{d}t} = f\ (c_1, c_2)$$

$$\frac{\mathrm{d}c_2}{\mathrm{d}t} = k_1\, c_1 - k_2\, c_2^2$$

The first equation could be the equation for a stirred tank reactor, for example. Suppose both k_1 and k_2 are large. The problem is then stiff, but the second equation could be taken at equilibrium. If

$$c_1 \rightleftarrows 2c_2$$

The equilibrium condition is then

$$\frac{c_2^2}{c_1} = \frac{k_1}{k_2} \equiv K$$

Under these conditions the problem becomes

$$\frac{\mathrm{d}c_1}{\mathrm{d}t} = f\ (c_1, c_2)$$

$$0 = k_l c_1 - k_2 c_2^2$$

Thus, a differential-algebraic system of equations is obtained. In this case, the second equation can be solved and substituted into the first to obtain differential equations, but in the general case that is not possible.

Differential-algebraic equations can be written in the general notation

$$\boldsymbol{F}\left(t, \boldsymbol{y}, \frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}t}\right) = 0$$

or the variables and equations may be separated according to whether they come primarily from differential [$y(t)$] or algebraic equations [$x(t)$]:

$$\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}t} = \boldsymbol{f}\ (t, \boldsymbol{y}, \boldsymbol{x})\,, \boldsymbol{g}\ (t, \boldsymbol{y}, \boldsymbol{x}) = 0$$

Another form is not strictly a differential-algebraic set of equations, but the same principles apply; this form arises frequently when the Galerkin finite element is applied:

$$\boldsymbol{A}\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}t} = \boldsymbol{f}\ (\boldsymbol{y})$$

The computer program DASSL [60, 61] can solve such problems. They can also be solved by writing the differential equation as

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{A}^{-1}\boldsymbol{f}\ (\boldsymbol{y})$$

When $\boldsymbol{A}$ is independent of $\boldsymbol{y}$, the inverse (from LU decompositions) need be computed only once.

In actuality, higher order backward-difference Gear methods are used in the computer program DASSL [60, 61].

Differential-algebraic systems are more complicated than differential systems because the solution may not always be defined. Pontelides et al. [62] introduced the term "index" to identify possible problems. The index is defined as the minimum number of times the equations must be differentiated with respect to time to convert the system to a set ofordinary differential equations. These higher derivatives may not exist, and the process places limits on which variables can be given initial values. Sometimes the initial values must be constrained by the algebraic equations [62]. For a differential-algebraic system modeling a distillation tower, the index depends on the specification of pressure for the column [62]. Several chemical engineering examples of differential-algebraic systems and a solution for one involving two-phase flow are given in [63].

6.6. Computer Software

Efficient software packages are widely available for solving ordinary differential equations as initial value problems. In each of the packages the user specifies the differential equation to be solved and a desired error criterion. The package then integrates in time and adjusts the step size to achieve the error criterion, within the limitations imposed by stability.

A popular explicit Runge – Kutta package is RKF 45. An estimate of the truncation error at each step is available. Then the step size can be reduced until this estimate is below the user-specified tolerance. The method is thus automatic, and the user is assured of the results. Note, however, that the tolerance is set on the local truncation error, namely, from one step to another, whereas the user is generally interested in the global trunction error, i.e., the error after several steps. The global error is generally made smaller by making the tolerance smaller, but the absolute accuracy is not the same as the tolerance. If the problem is stiff, then very small step sizes are used and the computation becomes

very lengthy. The RKF 45 code discovers this and returns control to the user with a message indicating the problem is too hard to solve with RKF 45.

A popular implicit package is LSODE, a version of Gear's method [57] written by Alan Hindmarsh at Lawrence Livermore Laboratory. In this package, the user specifies the differential equation to be solved and the tolerance desired. Now the method is implicit and, therefore, stable for any step size. The accuracy may not be acceptable, however, and sets of nonlinear equations must be solved. Thus, in practice, the step size is limited but not nearly so much as in the Runge – Kutta methods. In these packages both the step size and the order of the method are adjusted by the package itself. Suppose a k-th order method is being used. The truncation error is determined by the $(k + 1)$-th order derivative. This is estimated by using difference formulas and the values of the right-hand sides at previous times. An estimate is also made for the k-th and $(k + 2)$-th derivative. Then, the errors in a $(k - 1)$-th order method, a k-th order method, and a $(k + 1)$-th order method can be estimated. Furthermore, the step size required to satisfy the tolerance with each of these methods can be determined. Then the method and step size for the next step that achieves the biggest step can be chosen, with appropriate adjustments due to the different work required for each order. The package generally starts with a very small step size and a first-order method — the backward Euler method. Then it integrates along, adjusting the order up (and later down) depending on the error estimates. The user is thus assured that the local truncation error meets the tolerance. A further difficulty arises because the set of nonlinear equations must be solved. Usually a good guess of the solution is available, because the solution is evolving in time and past history can be extrapolated. Thus, the Newton – Raphson method will usually converge. The package protects itself, though, by only doing a few (i.e., three) iterations. If convergence is not reached within these iterations, the step size is reduced and the calculation is redone for that time step. The convergence theorem for the Newton – Raphson method (Chap. 1) indicates that the method will converge if the step size is small enough. Thus, the method is guaranteed to work. Further economies are possible. The Jacobian needed in the Newton – Raphson method can be fixed over several time steps. Then if the iteration does not converge, the Jacobian can be reevaluated at the current time step. If the iteration still does not converge, then the step size is reduced and a new Jacobian is evaluated. The successive substitution method can also be used — wh ich is even faster, except that it may not converge. However, it too will converge if the time step is small enough.

The Runge – Kutta methods give extremely good accuracy, especially when the step size is kept small for stability reasons. If the problem is stiff, though, backward difference implicit methods must be used. Many chemical reactor problems are stiff, necessitating the use of implicit methods. In the MATLAB suite of ODE solvers, ode45 uses a revision of the RKF45 program, while the ode15s program uses an improved backward difference method. Ref. [64] gives details of the programs in MATLAB. Fortunately, many packages are available. On the NIST web page, http://gams.nist.gov/ choose "problem decision tree", and then "differential and integral equations" to find packages which can be downloaded. On the Netlib web site, http://www.netlib.org/, choose "ode" to find packages which can be downloaded. Using Microsoft Excel to solve ordinary differential equations is cumbersome, except for the simplest problems.

6.7. Stability, Bifurcations, Limit Cycles

In this section, bifurcation theory is discussed in a general way. Some aspects of this subject involve the solution of nonlinear equations; other aspects involve the integration of ordinary differential equations; applications include chaos and fractals as well as unusual operation of some chemical engineering equipment. An excellent introduction to the subject and details needed to apply the methods are given in [65]. For more details of the algorithms described below and a concise survey with some chemical engineering examples, see [66] and [67]. Bifurcation results are closely connected with stability of the steady states, which is essentially a transient phenomenon.

Consider the problem

$$\frac{\partial u}{\partial t} = F\ (u, \lambda)$$

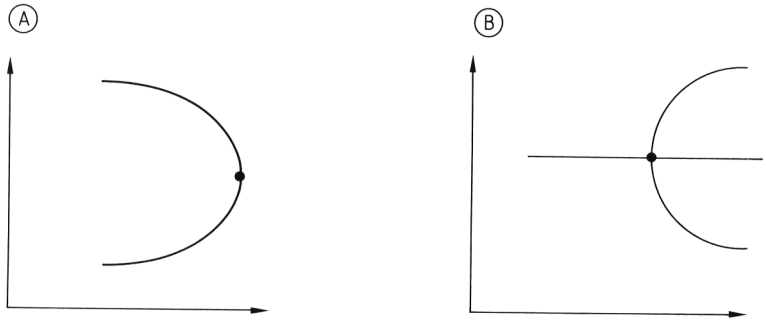

Figure 22. Limit points and bifurcation – limit points
A) Limit point (or turning point); B) Bifurcation-limit point (or singular turning point or bifurcation point)

The variable u can be a vector, which makes F a vector, too. Here, F represents a set of equations that can be solved for the steady state:

$$F\ (u, \lambda) = 0$$

If the Newton – Raphson method is applied,

$$F_u^s\ \delta\ u^s = -F\ (u^s, \lambda)$$

$$u^{s+1} = u^s + \delta u^s$$

is obtained, where

$$F_u^s = \frac{\partial F}{\partial u}\,(u^s)$$

is the Jacobian. Look at some property of the solution, perhaps the value at a certain point or the maximum value or an integral of the solution. This property is plotted versus the parameter λ; typical plots are shown in Figure 22. At the point shown in Figure 22 A, the determinant of the Jacobian is zero:

$$\det F_u = 0$$

For the limit point,

$$\frac{\partial F}{\partial \lambda} \neq 0$$

whereas for the bifurcation-limit point

$$\frac{\partial F}{\partial \lambda} = 0$$

The stability of the steady solutions is also of interest. Suppose a steady solution u_{ss}; the function u is written as the sum of the known steady state and a perturbation u':

$$u = u_{ss} + u'$$

This expression is substituted into the original equation and linearized about the steady-state value:

$$\frac{\partial u_{ss}}{\partial t} + \frac{\partial u'}{\partial t} = F\ (u_{ss} + u', \lambda)$$

$$\approx F\ (u_{ss}, \lambda) + \frac{\partial F}{\partial u}|_{u_{ss}}\, u' + \cdots$$

The result is

$$\frac{\partial u'}{\partial t} = F_u^{ss}\, u'$$

A solution of the form

$$u'\,(x, t) = \mathrm{e}^{\sigma t}\, X\,(x)$$

gives

$$\sigma \mathrm{e}^{\sigma t}\, X = F_u^{ss} \mathrm{e}^{\sigma t} X$$

The exponential term can be factored out and

$$(F_u^{ss} - \sigma\, \delta)\, X = 0$$

A solution exists for X if and only if

$$\det\, |F_u^{ss} - \sigma\, \delta| = 0$$

The σ are the eigenvalues of the Jacobian. Now clearly if Re $(\sigma) > 0$ then u' grows with time, and the steady solution u_{ss} is said to be unstable to small disturbances. If Im $(\sigma) = 0$ it is called stationary instability, and the disturbance would grow monotonically, as indicated in Figure 23 A. If Im $(\sigma) \neq 0$ then the disturbance grows in an oscillatory fashion, as shown in Figure 23 B, and is called oscillatory instability. The case in which Re $(\sigma) = 0$ is the dividing point between stability and instability. If Re $(\sigma) = 0$ and Im $(\sigma) = 0$ — the point governing the onset of stationary instability — then $\sigma = 0$. However, this means that $\sigma = 0$ is an eigenvalue of the Jacobian, and the determinant of the Jacobian is zero. Thus, the points at which the determinant of the Jacobian is zero (for limit points

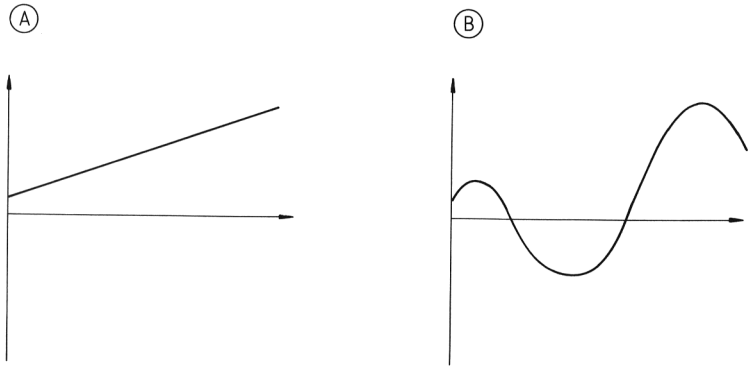

Figure 23. Stationary and oscillatory instability
A) Stationary instability; B) Oscillatory instability

and bifurcation-limit points) are the points governing the onset of stationary instability. When Re (σ) = 0 but Im (σ) $\neq$ 0, which is the onset of oscillatory instability, an even number of eigenvalues pass from the left-hand complex plane to the right-hand complex plane. The eigenvalues are complex conjugates of each other (a result of the original equations being real, with no complex numbers), and this is called a Hopf bifurcation. Numerical methods to study Hopf bifurcation are very computationally intensive and are not discussed here [65].

To return to the problem of solving for the steady-state solution: near the limit point or bifurcation-limit point two solutions exist that are very close to each other. In solving sets of equations with thousands of unknowns, the difficulties in convergence are obvious. For some dependent variables the approximation may be converging to one solution, whereas for another set of dependent variables it may be converging to the other solution; or the two solutions may all be mixed up. Thus, solution is difficult near a bifurcation point, and special methods are required. These methods are discussed in [66].

The first approach is to use *natural continuation* (also known as Euler – Newton continuation). Suppose a solution exists for some parameter λ. Call the value of the parameter λ_0 and the corresponding solution u_0. Then

$$F\ (u_0\ , \lambda_0) = 0$$

Also, compute u_λ as the solution to

$$F_u^{\mathrm{ss}}\, u_\lambda = -F_\lambda$$

at this point [λ_0, u_0]. Then predict the starting guess for another λ using

$$u^0 = u_0 + u_\lambda\ (\lambda - \lambda_0)$$

and apply Newton – Raphson with this initial guess and the new value of λ. This will be a much better guess of the new solution than just u_0 by itself.

Even this method has difficulties, however. Near a limit point the determinant of the Jacobian may be zero and the Newton method may fail. Perhaps no solutions exist at all for the chosen parameter λ near a limit point. Also, the ability to switch from one solution path to another at a bifurcation-limit point is necessary. Thus, other methods are needed as well: arc-length continuation and pseudo-arc-length continuation [66]. These are described in Chapter 1.

6.8. Sensitivity Analysis

Often, when solving differential equations, the solution as well as the sensitivity of the solution to the value of a parameter must be known. Such information is useful in doing parameter estimation (to find the best set of parameters for a model) and in deciding whether a parameter needs to be measured accurately. The differential equation for y (t, α) where α is a parameter, is

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f\ (y, \alpha)\,,\ y\ (0) = y_0$$

If this equation is differentiated with respect to α, then because y is a function of t and α

$$\frac{\partial}{\partial \alpha}\left(\frac{\mathrm{d}y}{\mathrm{d}t}\right) = \frac{\partial f}{\partial y}\frac{\partial y}{\partial \alpha} + \frac{\partial f}{\partial \alpha}$$

Exchanging the order of differentiation in the first term leads to the ordinary differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial y}{\partial \alpha}\right)=\frac{\partial f}{\partial y}\frac{\partial y}{\partial \alpha}+\frac{\partial f}{\partial \alpha}$$

The initial conditions on $\partial y/\partial\alpha$ are obtained by differentiating the initial conditions

$$\frac{\partial}{\partial \alpha}\left[y\ (0,\alpha)=y_0\right], \text{or } \frac{\partial y}{\partial \alpha}\ (0)=0$$

Next, let

$$y_1=y, y_2=\frac{\partial y}{\partial \alpha}$$

and solve the set of ordinary differential equations

$$\frac{\mathrm{d}y_1}{\mathrm{d}t}=f\ (y_1,\alpha) \qquad y_1\ (0)=y_0$$

$$\frac{\mathrm{d}y_2}{\mathrm{d}t}=\frac{\partial f}{\partial y}\ (y_1,\alpha)\ y_2+\frac{\partial f}{\partial \alpha} \qquad y_2\ (0)=0$$

Thus, the solution $y\ (t,\ \alpha)$ and the derivative with respect to α are obtained. To project the impact of α, the solution for $\alpha = \alpha_1$ can be used:

$$y\ (t,\alpha)=y_1\ (t,\alpha_1)+\frac{\partial y}{\partial \alpha}\ (t,\alpha_1)\ (\alpha-\alpha_1)+\cdots$$

$$=y_1\ (t,\alpha_1)+y_2\ (t,\alpha_1)\ (\alpha-\alpha_1)+\cdots$$

This is a convenient way to determine the sensitivity of the solution to parameters in the problem.

6.9. Molecular Dynamics

(→ Molecular Dynamics Simulations)

Special integration methods have been developed for molecular dynamics calculations due to the structure of the equations. A very large number of equations are to be integrated, with the following form based on molecular interactions between molecules

$$m_i\frac{d^2\boldsymbol{r}_i}{dt^2}=\boldsymbol{F}_i\left(\{\boldsymbol{r}\}\right), \boldsymbol{F}_i\left(\{\boldsymbol{r}\}\right)=-\boldsymbol{\nabla}V$$

where m_i is the mass of the *i*-th particle, $\boldsymbol{r}_i$ is the position of the *i*-th particle, $\boldsymbol{F}_i$ is the force acting on the *i*-th particle, and *V* is the potential energy that depends upon the location of all the particles (but not their velocities). Since the major part of the calculation is in the evaluation of the forces, or potentials, a method must be used that minimizes the number of times the forces are calculated to move from one time to another time. Rewrite this equation in the form of an acceleration.

$$\frac{d^2\boldsymbol{r}_i}{dt^2}=\frac{1}{m_i}\boldsymbol{F}_i\left(\{\boldsymbol{r}\}\right)\equiv\boldsymbol{a}_i$$

In the Verlot method, this equation is written using central finite differences (Eq. 12). Note that the accelerations do not depend upon the velocities.

$$\boldsymbol{r}_i\left(t+\Delta t\right)=2\boldsymbol{r}_i\left(t\right)-\boldsymbol{r}_i\left(t-\Delta t\right)+\boldsymbol{a}_i\left(t\right)\Delta t^2$$

The calculations are straightforward, and no explicit velocity is needed. The storage requirement is modest, and the precision is modest (it is a second-order method). Note that one must start the calculation with values of $\{r\}$ at time *t* and $t-\Delta t$.

In the Velocity Verlot method, an equation is written for the velocity, too.

$$\frac{d\boldsymbol{v}_i}{dt}=\boldsymbol{a}_i$$

The trapezoid rule (see page 18) is applied to obtain

$$\boldsymbol{v}_i\left(t+\Delta t\right)=\boldsymbol{v}_i\left(t\right)+\frac{1}{2}\left[\boldsymbol{a}_i\left(t\right)+\boldsymbol{a}_i\left(t+\Delta t\right)\right]\Delta t$$

The position of the particles is expanded in a Taylor series.

$$\boldsymbol{r}_i\left(t+\Delta t\right)=\boldsymbol{r}_i\left(t\right)+\boldsymbol{v}_i\Delta t+\frac{1}{2}\boldsymbol{a}_i\left(t\right)\Delta t^2$$

Beginning with values of $\{\boldsymbol{r}\}$ and $\{\boldsymbol{v}\}$ at time zero, one calculates the new positions and then the new velocities. This method is second order in Δt, too. For additional details, see [68 – 72].

7. Ordinary Differential Equations as Boundary Value Problems

Diffusion problems in one dimension lead to boundary value problems. The boundary conditions are applied at two different spatial locations: at one side the concentration may be fixed and at the other side the flux may be fixed. Because the conditions are specified at two different locations the problems are not initial value in character. To begin at one position and integrate directly is impossible because at least one of the conditions is specified somewhere else and not enough conditions are available to begin the

calculation. Thus, methods have been developed especially for boundary value problems. Examples include heat and mass transfer in a slab, reaction – diffusion problems in a porous catalyst, reactor with axial dispersion, packed beds, and countercurrent heat transfer.

7.1. Solution by Quadrature

When only one equation exists, even if it is nonlinear, it may possibly be solved by quadrature. For

$$\frac{dy}{dt} = f(y)$$

$$y(0) = y_0$$

the problem can be separated

$$\frac{dy}{f(y)} = dt$$

and integrated

$$\int_{y_0}^{y} \frac{dy'}{f(y')} = \int_0^t dt = t$$

If the quadrature can be performed analytically, the exact solution has been found.

As an example, consider the flow of a non-Newtonian fluid in a pipe, as illustrated in Figure 24. The governing differential equation is [73]

$$\frac{1}{r}\frac{d}{dr}(r\tau) = -\frac{\Delta p}{L}$$

where r is the radial position from the center of the pipe, τ is the shear stress, Δp is the pressure drop along the pipe, and L is the length over which the pressure drop occurs. The variables are separated once

$$d(r\tau) = -\frac{\Delta p}{L} r dr$$

and then integrated to give

$$r\tau = -\frac{\Delta p}{L}\frac{r^2}{2} + c_1$$

Proceeding further requires choosing a constitutive relation relating the shear stress and the velocity gradient as well as a condition specifying the constant. For a Newtonian fluid

$$\tau = -\eta \frac{dv}{dr}$$

where v is the velocity and η the viscosity. Then the variables can be separated again and the result integrated to give

$$-\eta v = -\frac{\Delta p}{L}\frac{r^2}{4} + c_1 \ln r + c_2$$

Now the two unknowns must be specified from the boundary conditions. This problem is a two-point boundary value problem because one of the conditions is usually specified at $r = 0$ and the other at $r = R$, the tube radius. However, the technique of separating variables and integrating works quite well.

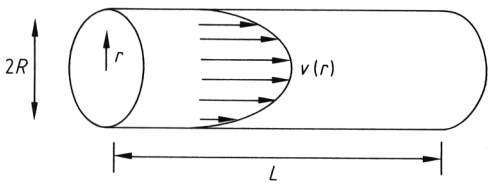

Figure 24. Flow in pipe

When the fluid is non-Newtonian, it may not be possible to do the second step analytically. For example, for the Bird – Carreau fluid [74, p. 171], stress and velocity are related by

$$\tau = \frac{\eta_0}{\left[1 + \lambda\left(\frac{dv}{dr}\right)^2\right]^{(1-n)/2}}$$

where η_0 is the viscosity at $v = 0$ and λ the time constant.

Putting this value into the equation for stress as a function of r gives

$$\frac{\eta_0}{\left[1 + \lambda\left(\frac{dv}{dr}\right)^2\right]^{(1-n)/2}} = -\frac{\Delta p}{L}\frac{r}{2} + \frac{c_1}{r}$$

This equation cannot be solved analytically for dv/dr, except for special values of n. For problems such as this, numerical methods must be used.

7.2. Initial Value Methods

An initial value method is one that utilizes the techniques for initial value problems but allows for an iterative calculation to satisfy all the boundary conditions. Suppose the nonlinear boundary value problem

$$\frac{d^2y}{dx^2} = f\left(x,y,\frac{dy}{dx}\right)$$

with the boundary conditions

$$a_0\, y\,(0) - a_1 \frac{dy}{dx}\,(0) = \alpha,\ a_i \geq 0$$

$$b_0\, y\,(1) - b_1 \frac{dy}{dx}\,(1) = \beta,\ b_i \geq 0$$

Convert this second-order equation into two first-order equations along with the boundary conditions written to include a parameter s.

$$\frac{du}{dx} = v$$

$$\frac{dv}{dx} = f\,(x,u,v)$$

$$u\,(0) = a_1 s - c_1 \alpha$$

$$v\,(0) = a_0 s - c_0 \alpha$$

The parameters c_0 and c_1 are specified by the analyst such that

$$a_1 c_0 - a_0 c_1 = 1$$

This ensures that the first boundary condition is satisfied for any value of parameter s. If the proper value for s is known, $u\,(0)$ and $u'\,(0)$ can be evaluated and the equation integrated as an initial value problem. The parameter s should be chosen iteratively so that the last boundary condition is satisfied.

The model for a *chemical reactor with axial diffusion* is

$$\frac{1}{Pe}\frac{dc^2}{dz^2} - \frac{dc}{dz} = DaR\,(c)$$

$$-\frac{1}{Pe}\frac{dc}{dz}\,(0) + c\,(0) = c_{in},\ \frac{dc}{dz}\,(1) = 0$$

where *Pe* is the Péclet number and *Da* the Damköhler number.

The boundary conditions are due to Danckwerts [75] and to Wehner and Wilhelm [76]. This problem can be treated by using initial value methods also, but the method is highly sensitive to the choice of the parameter s, as outlined above. Starting at $z = 0$ and making small changes in s will cause large changes in the solution at the exit, and the boundary condition at the exit may be impossible to satisfy. By starting at $z = 1$, however, and integrating backwards, the process works and an iterative scheme converges in many cases [77]. However, if the problem is extremely nonlinear the iterations may not converge. In such cases, the methods for boundary value problems described below must be used.

Packages to solve boundary value problems are available on the internet. On the NIST web page, http://gams.nist.gov/ choose "problem decision tree", and then "differential and integral equations", then "ordinary differential equations", "multipoint boundary value problems". On the Netlib web site, http://www.netlib.org/, search on "boundary value problem". Any spreadsheet that has an iteration capability can be used with the finite difference method. Some packages for partial differential equations also have a capability for solving one-dimensional boundary value problems [e.g., Comsol Multiphysics (formerly FEMLAB)].

7.3. Finite Difference Method

To apply the finite difference method, we first spread grid points through the domain. Figure 25 shows a uniform mesh of n points (nonuniform meshes are also possible). The unknown, here $c\,(x)$, at a grid point x_i is assigned the symbol $c_i = c\,(x_i)$. The finite difference method can be derived easily by using a Taylor expansion of the solution about this point.

$$\begin{aligned} c_{i+1} &= c_i + \left.\frac{dc}{dx}\right|_i \Delta x + \left.\frac{d^2c}{dx^2}\right|_i \frac{\Delta x^2}{2} + \ldots \\ c_{i-1} &= c_i - \left.\frac{dc}{dx}\right|_i \Delta x + \left.\frac{d^2c}{dx^2}\right|_i \frac{\Delta x^2}{2} - \ldots \end{aligned} \tag{8}$$

These formulas can be rearranged and divided by Δx to give

$$\left.\frac{dc}{dx}\right|_i = \frac{c_{i+1} - c_i}{\Delta x} - \left.\frac{d^2c}{dx^2}\right|_i \frac{\Delta x}{2} + \ldots \tag{9}$$

$$\left.\frac{dc}{dx}\right|_i = \frac{c_i - c_{i-1}}{\Delta x} - \left.\frac{d^2c}{dx^2}\right|_i \frac{\Delta x}{2} + \ldots \tag{10}$$

which are representations of the first derivative. Alternatively the two equations can be subtracted from each other, rearranged and divided by Δx to give

$$\left.\frac{dc}{dx}\right|_i = \frac{c_{i+1} - c_{i-1}}{2\Delta x} - \left.\frac{d^2c}{dx^3}\right|_i \frac{\Delta x^2}{3!} \tag{11}$$

If the terms multiplied by Δx or Δx^2 are neglected, three representations of the first derivative are possible. In comparison with the Taylor series, the truncation error in the first two expressions is proportional to Δx, and the methods are said to be first order. The truncation error in the last expression is proportional to Δx^2, and the method is said to be second order. Usually, the

last equation is chosen to ensure the best accuracy.

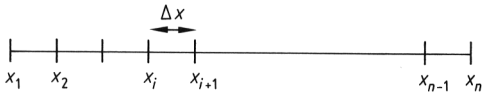

Figure 25. Finite difference mesh; Δx uniform

The finite difference representation of the second derivative can be obtained by adding the two expressions in Equation 8. Rearrangement and division by Δx^2 give

$$\left.\frac{\mathrm{d}^2c}{\mathrm{d}x^2}\right|_i = \frac{c_{i+1}-2c_i+c_{i-1}}{\Delta x^2} - \left.\frac{\mathrm{d}^4c}{\mathrm{d}x^4}\right|_i \frac{\Delta x^2}{4!} + \ldots \qquad (12)$$

The truncation error is proportional to Δx^2.

To see how to solve a differential equation, consider the equation for convection, diffusion, and reaction in a tubular reactor:

$$\frac{1}{Pe}\frac{\mathrm{d}^2c}{\mathrm{d}x^2} - \frac{\mathrm{d}c}{\mathrm{d}x} = Da\, R(c)$$

To evaluate the differential equation at the i-th grid point, the finite difference representations of the first and second derivatives can be used to give

$$\frac{1}{Pe}\frac{c_{i+1}-2c_i+c_{i-1}}{\Delta x^2} - \frac{c_{i+1}-c_{i-1}}{2\Delta x} = Da\, R \qquad (13)$$

This equation is written for $i = 2$ to $n - 1$ (i.e., the internal points). The equations would then be coupled but would involve the values of c_1 and c_n, as well. These are determined from the boundary conditions.

If the boundary condition involves a derivative, the finite difference representation of it must be carefully selected; here, three possibilities can be written. Consider a derivative needed at the point $i = 1$. First, Equation 9 could be used to write

$$\left.\frac{\mathrm{d}c}{\mathrm{d}x}\right|_1 = \frac{c_2-c_1}{\Delta x} \qquad (14)$$

Then a second-order expression is obtained that is one-sided. The Taylor series for the point c_{i+2} is written:

$$c_{i+2} = c_i + \left.\frac{\mathrm{d}c}{\mathrm{d}x}\right|_i 2\Delta x + \left.\frac{\mathrm{d}^2c}{\mathrm{d}x^2}\right|_i \frac{4\Delta x^2}{2!} + \left.\frac{\mathrm{d}^3c}{\mathrm{d}x^3}\right|_i \frac{8\Delta x^3}{3!} + \ldots$$

Four times Equation 8 minus this equation, with rearrangement, gives

$$\left.\frac{\mathrm{d}c}{\mathrm{d}x}\right|_i = \frac{-3c_i+4c_{i+1}-c_{i+2}}{2\Delta x} + O\left(\Delta x^2\right)$$

Thus, for the first derivative at point $i = 1$

$$\left.\frac{\mathrm{d}c}{\mathrm{d}x}\right|_i = \frac{-3c_i+4c_2-c_3}{2\Delta x} \qquad (15)$$

This one-sided difference expression uses only the points already introduced into the domain. The third alternative is to add a false point, outside the domain, as $c_0 = c\ (x = -\Delta x)$. Then the centered first derivative, Equation 11, can be used:

$$\left.\frac{\mathrm{d}c}{\mathrm{d}x}\right|_1 = \frac{c_2-c_0}{2\Delta x}$$

Because this equation introduces a new variable, another equation is required. This is obtained by also writing the differential equation (Eq. 13), for $i = 1$.

The same approach can be taken at the other end. As a boundary condition, any of three choices can be used:

$$\left.\frac{\mathrm{d}c}{\mathrm{d}x}\right|_n = \frac{c_n-c_{n-1}}{\Delta x}$$

$$\left.\frac{\mathrm{d}c}{\mathrm{d}x}\right|_n = \frac{c_{n-2}-4c_{n-1}+3c_n}{2\Delta x}$$

$$\left.\frac{\mathrm{d}c}{\mathrm{d}x}\right|_n = \frac{c_{n+1}-c_{n-1}}{2\Delta x}$$

The last two are of order Δx^2 and the last one would require writing the differential equation (Eq. 13) for $i = n$, too.

Generally, the first-order expression for the boundary condition is not used because the error in the solution would decrease only as Δx, and the higher truncation error of the differential equation (Δx^2) would be lost. For this problem the boundary conditions are

$$-\frac{1}{Pe}\left.\frac{\mathrm{d}c}{\mathrm{d}x}\right|(0) + c(0) = c_{\mathrm{in}}$$

$$\left.\frac{\mathrm{d}c}{\mathrm{d}x}\right|(1) = 0$$

Thus, the three formulations would give first order in Δx

$$-\frac{1}{Pe}\frac{c_2-c_1}{\Delta x} + c_1 = c_{\mathrm{in}}$$

$$\frac{c_n-c_{n-1}}{\Delta x} = 0$$

plus Equation 13 at points $i = 2$ through $n - 1$; second order in Δx, by using a three-point one-sided derivative

$$-\frac{1}{Pe}\frac{-3c_1+4c_2-c_3}{2\Delta x}+c_1 = c_{\text{in}}$$

$$\frac{c_{n-2}-4c_{n-1}+3c_n}{2\Delta x} = 0$$

plus Equation 13 at points $i = 2$ through $n - 1$; second order in Δx, by using a false boundary point

$$-\frac{1}{Pe}\frac{c_2-c_0}{2\Delta x}+c_1 = c_{\text{in}}$$

$$\frac{c_{n+1}-c_{n-1}}{2\Delta x} = 0$$

plus Equation 13 at points $i = 1$ through n.

The sets of equations can be solved by using the Newton – Raphson method, as outlined in Section 1.2.

Frequently, the transport coefficients (e.g., diffusion coefficient D or thermal conductivity) depend on the dependent variable (concentration or temperature, respectively). Then the differential equation might look like

$$\frac{\mathrm{d}}{\mathrm{d}x}\left(D(c)\frac{\mathrm{d}c}{\mathrm{d}x}\right) = 0$$

This could be written as

$$-\frac{\mathrm{d}J}{\mathrm{d}x} = 0 \tag{16}$$

in terms of the mass flux J, where the mass flux is given by

$$J = -D(c)\frac{\mathrm{d}c}{\mathrm{d}x}$$

Because the coefficient depends on c the equations are more complicated. A finite difference method can be written in terms of the fluxes at the midpoints, $i + 1/2$. Thus,

$$-\frac{J_{i+1/2}-J_{i-1/2}}{\Delta x} = 0$$

Then the constitutive equation for the mass flux can be written as

$$J_{i+1/2} = -D\left(c_{i+1/2}\right)\frac{c_{i+1}-c_i}{\Delta x}$$

If these are combined,

$$\frac{D\left(c_{i+1/2}\right)(c_{i+1}-c_i)-D\left(c_{i-1/2}\right)(c_i-c_{i-1})}{\Delta x^2} = 0$$

This represents a set of nonlinear algebraic equations that can be solved with the Newton – Raphson method. However, in this case a viable iterative strategy is to evaluate the transport coefficients at the last value and then solve

$$\frac{D\left(c_{i+1/2}^k\right)\left(c_{i+1}^{k+1}-c_i^{k+1}\right)-D\left(c_{i-1/2}^k\right)\left(c_i^{k+1}-c_{i-1}^{k+1}\right)}{\Delta x^2} = 0$$

The advantage of this approach is that it is easier to program than a full Newton – Raphson method. If the transport coefficients do not vary radically, the method converges. If the method does not converge, use of the full Newton – Raphson method may be necessary.

Three ways are commonly used to evaluate the transport coefficient at the midpoint. The first one employs the transport coefficient evaluated at the average value of the solutions on either side:

$$D\left(c_{i+1/2}\right) \approx D\left[\frac{1}{2}\left(c_{i+1}+c_i\right)\right]$$

The second approach uses the average of the transport coefficients on either side:

$$D\left(c_{i+1/2}\right) \approx \frac{1}{2}\left[D\left(c_{i+1}\right)+D\left(c_i\right)\right] \tag{17}$$

The truncation error of these approaches is also Δx^2 [78, Chap. 14], [3, p. 215]. The third approach employs an "upstream" transport coefficient.

$$D\left(C_{i+1/2}\right) \approx D\left(c_{i+1}\right), \text{ when } D\left(c_{i+1}\right) > D\left(c_i\right)$$

$$D\left(c_{i+1/2}\right) \approx \left(c_i\right), \text{ when } D\left(c_{i+1}\right) < D\left(c_i\right)$$

This approach is used when the transport coefficients vary over several orders of magnitude and the "upstream" direction is defined as the one in which the transport coefficient is larger. The truncation error of this approach is only Δx [78, Chap. 14] , [3, p. 253], but this approach is useful if the numerical solutions show unrealistic oscillations [3, 78].

Rigorous error bounds for linear ordinary differential equations solved with the finite difference method are dicussed by Isaacson and Keller [79, p. 431].

7.4. Orthogonal Collocation

The orthogonal collocation method has found widespread application in chemical engineering, particularly for chemical reaction engineering. In the collocation method [3], the dependent variable is expanded in a series.

$$y(x) = \sum_{i=1}^{N+2} a_i y_i(x) \tag{18}$$

Suppose the differential equation is

$$N[y] = 0$$

Then the expansion is put into the differential equation to form the residual:

$$\text{Residual} = N\left[\sum_{i=1}^{N+2} a_i y_i(x)\right]$$

In the collocation method, the residual is set to zero at a set of points called collocation points:

$$N\left[\sum_{i=1}^{N+2} a_i y_i(x_j)\right] = 0,\ j = 2,\ldots,N+1$$

This provides N equations; two more equations come from the boundary conditions, giving $N + 2$ equations for $N + 2$ unknowns. This procedure is especially useful when the expansion is in a series of orthogonal polynomials, and when the collocation points are the roots to an orthogonal polynomial, as first used by Lanczos [80, 81]. A major improvement was the proposal by Villadsen and Stewart [82] that the entire solution process be done in terms of the solution at the collocation points rather than the coefficients in the expansion. Thus, Equation 18 would be evaluated at the collocation points

$$y(x_j) = \sum_{i=1}^{N+2} a_i y_i(x_j),\ j = 1,\ldots,N+2$$

and solved for the coefficients in terms of the solution at the collocation points:

$$a_i = \sum_{j=1}^{N+2} [y_i(x_j)]^{-1} y(x_j),\ i = 1,\ldots,N+2$$

Furthermore, if Equation 18 is differentiated once and evaluated at all collocation points, the first derivative can be written in terms of the values at the collocation points:

$$\frac{dy}{dx}(x_j) = \sum_{i=1}^{N+2} a_i \frac{dy_i}{dx}(x_j),\ j = 1,\ldots,N+2$$

This can be expressed as

$$\frac{dy}{dx}(x_j) = \sum_{i,k=1}^{N+2} [y_i(x_k)]^{-1} y(x_k) \frac{dy_i}{dx}(x_j),\ j = 1,\ldots,N+2$$

or shortened to

$$\frac{dy}{dx}(x_j) = \sum_{k=1}^{N+2} A_{jk} y(x_k),$$

$$A_{jk} = \sum_{i=1}^{N+2} [y_i(x_k)]^{-1} \frac{dy_i}{dx}(x_j)$$

Similar steps can be applied to the second derivative to obtain

$$\frac{d^2y}{dx^2}(x_j) = \sum_{k=1}^{N+2} B_{jk} y(x_k),$$

$$B_{jk} = \sum_{i=1}^{N+2} [y_i(x_k)]^{-1} \frac{d^2y_i}{dx^2}(x_j)$$

This method is next applied to the differential equation for reaction in a tubular reactor, after the equation has been made nondimensional so that the dimensionless length is 1.0.

$$\begin{aligned} &\tfrac{1}{Pe}\tfrac{d^2c}{dx^2} - \tfrac{dc}{dx} = Da\,R(c), \\ &-\tfrac{dc}{dx}(0) = Pe[c(0) - c_{in}],\ \tfrac{dc}{dx}(1) = 0 \end{aligned} \tag{19}$$

The differential equation at the collocation points is

$$\frac{1}{Pe}\sum_{k=1}^{N+2} B_{jk} c(x_k) - \sum_{k=1}^{N+2} A_{jk} c(x_k) = Da\,R(c_j) \tag{20}$$

and the two boundary conditions are

$$\begin{aligned} &-\sum_{k=1}^{N+2} A_{lk} c(x_k) = Pe(c_1 - c_{in}), \\ &\sum_{k=1}^{N+2} A_{N+2,k} c(x_k) = 0 \end{aligned} \tag{21}$$

Note that 1 is the first collocation point ($x = 0$) and $N + 2$ is the last one ($x = 1$). To apply the method, the matrices A_{ij} and B_{ij} must be found and the set of algebraic equations solved, perhaps with the Newton – Raphson method. If orthogonal polynomials are used and the collocation points are the roots to one of the orthogonal polynomials, the orthogonal collocation method results.

In the orthogonal collocation method the solution is expanded in a series involving orthogonal polynomials, where the polynomials $P_{i-1}(x)$ are defined in Section 2.2.

$$\begin{aligned} y &= a + bx + x(1-x)\sum_{i=1}^{N} a_i P_{i-1}(x) \\ &= \sum_{i=1}^{N+2} b_i P_{i-1}(x) \end{aligned} \tag{22}$$

which is also

$$y = \sum_{i=1}^{N+2} d_i x^{i-1}$$

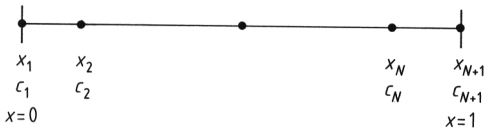

Figure 26. Orthogonal collocation points

The collocation points are shown in Figure 26. There are N interior points plus one at each end, and the domain is always transformed to lie on 0 to 1. To define the matrices A_{ij} and B_{ij} this expression is evaluated at the collocation points; it is also differentiated and the result is evaluated at the collocation points.

$$y(x_j) = \sum_{i=1}^{N+2} d_i x_j^{i-1}$$

$$\frac{dy}{dx}(x_j) = \sum_{i=1}^{N+2} d_i (i-1) x_j^{i-2}$$

$$\frac{d^2y}{dx^2}(x_j) = \sum_{i=1}^{N+2} d_i (i-1)(i-2) x_j^{i-3}$$

These formulas are put in matrix notation, where $\boldsymbol{Q}$, $\boldsymbol{C}$, and $\boldsymbol{D}$ are $N+2$ by $N+2$ matrices.

$$\boldsymbol{y} = \boldsymbol{Q}\boldsymbol{d}, \frac{d\boldsymbol{y}}{dx} = \boldsymbol{C}\boldsymbol{d}, \frac{d^2\boldsymbol{y}}{dx^2} = \boldsymbol{D}\boldsymbol{d}$$

$$Q_{ji} = x_j^{i-1},\ C_{ji} = (i-1)x_j^{i-2},$$

$$D_{ji} = (i-1)(i-2)x_j^{i-3}$$

In solving the first equation for $\boldsymbol{d}$, the first and second derivatives can be written as

$$\boldsymbol{d} = \boldsymbol{Q}^{-1}\boldsymbol{y},\ \frac{d\boldsymbol{y}}{dx} = \boldsymbol{C}\boldsymbol{Q}^{-1}\boldsymbol{y} = \boldsymbol{A}\boldsymbol{y},$$
$$\frac{d^2\boldsymbol{y}}{dx^2} = \boldsymbol{D}\boldsymbol{Q}^{-1}\boldsymbol{y} = \boldsymbol{B}\boldsymbol{y} \quad (23)$$

Thus the derivative at any collocation point can be determined in terms of the solution at the collocation points. The same property is enjoyed by the finite difference method (and the finite element method described below), and this property accounts for some of the popularity of the orthogonal collocation method. In applying the method to Equation 19, the same result is obtained; Equations 20 and 21, with the matrices defined in Equation 23. To find the solution at a point that is not a collocation point, Equation 22 is used; once the solution is known at all collocation points, $\boldsymbol{d}$ can be found; and once $\boldsymbol{d}$ is known, the solution for any x can be found.

To use the orthogonal collocation method, the matrices are required. They can be calculated as shown above for small N ($N < 8$) and by using more rigorous techniques, for higher N (see Chap. 2). However, having the matrices listed explicitly for $N = 1$ and 2 is useful; this is shown in Table 7.

For some reaction diffusion problems, the solution can be an even function of x. For example, for the problem

$$\frac{d^2c}{dx^2} = kc,\ \frac{dc}{dx}(0) = 0,\ c(1) = 1 \quad (24)$$

the solution can be proved to involve only even powers of x. In such cases, an orthogonal collocation method, which takes this feature into

Table 7. Matrices for orthogonal collocation

$N = 1, a = 0.1666666667$

$$x_j = \begin{pmatrix} 0 \\ 0.5 \\ 1 \end{pmatrix},\ W_j = \begin{pmatrix} a \\ 4a \\ a \end{pmatrix},\ A_{ji} = \begin{pmatrix} -3 & 4 & -1 \\ -1 & 0 & 1 \\ 1 & -4 & 3 \end{pmatrix},\ B_{ji} = \begin{pmatrix} 4 & -8 & 4 \\ 4 & -8 & 4 \\ 4 & -8 & 4 \end{pmatrix},\ Q_{ji}^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ -3 & 4 & -1 \\ 2 & -4 & 2 \end{pmatrix}$$

$N = 2$

$$x_j = \begin{pmatrix} 0 \\ e \\ 1-e \\ 1 \end{pmatrix},\ W_j = \begin{pmatrix} 0 \\ 0.5 \\ 0.5 \\ 0 \end{pmatrix},\ Q_{ji}^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -7 & 7+b & -1-b & 1 \\ 12 & -18-f & 12+f & -6 \\ -6 & 10+d & -10-d & 6 \end{pmatrix}$$

$$A_{ji} = \begin{pmatrix} -7 & 7+b & -1-b & 1 \\ -1-a & a & a & 1-a \\ -1+a & -a & -a & 1+a \\ -1 & 1+b & -7-b & 7 \end{pmatrix},\ B_{ji} = \begin{pmatrix} 24 & -37-c & 25+c & -12 \\ 16+d & -24 & 12 & -4-d \\ -4-d & 12 & -24 & 16+d \\ -12 & 25+c & -37-c & 24 \end{pmatrix}$$

where $a = 1.732050808$, $b = 1.196152423$, $c = 0.17691454$, $d = 0.392304846$
$e = 0.2113248654$, $f = 0.58845727$

account, is convenient. This can easily be done by using expansions that only involve even powers of x. Thus, the expansion

$$y\left(x^2\right) = y\left(1\right) + \left(1-x^2\right)\sum_{i=1}^{N} a_i P_{i-1}\left(x^2\right)$$

is equivalent to

$$y\left(x^2\right) = \sum_{i=1}^{N+1} b_i P_{i-1}\left(x^2\right) = \sum_{i=1}^{N+1} d_i x^{2i-2}$$

The polynomials are defined to be orthogonal with the weighting function $W\left(x^2\right)$.

$$\int_0^1 W\left(x^2\right) P_k\left(x^2\right) P_m\left(x^2\right) x^{a-1}\mathrm{d}x = 0 \qquad k \leq m-1 \tag{25}$$

where the power on x^{a-1} defines the geometry as planar or Cartesian (a = 1), cylindrical (a = 2), and spherical (a = 3). An analogous development is used to obtain the $(N + 1)\times(N + 1)$ matrices

$$y\left(x_j\right) = \sum_{i=1}^{N+1} d_i x_j^{2i-2}$$

$$\frac{\mathrm{d}y}{\mathrm{d}x}\left(x_j\right) = \sum_{i=1}^{N+1} d_i\left(2i-2\right) x_j^{2i-3}$$

$$\nabla^2 y\left(x_i\right) = \sum_{i=1}^{N+1} d_i \nabla^2\left(x^{2i-2}\right)\Big|_{x_j}$$

$$\boldsymbol{y} = \boldsymbol{Q}\boldsymbol{d}, \frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}x} = \boldsymbol{C}\boldsymbol{d}, \nabla^2\boldsymbol{y} = \boldsymbol{D}\boldsymbol{d}$$

$$Q_{ji} = x_j^{2i-2},\ C_{ji} = \left(2i-2\right) x_j^{2i-3},$$

$$D_{ji} = \nabla^2\left(x^{2i-2}\right)|_{x_j}$$

$$\boldsymbol{d} = \boldsymbol{Q}^{-1}\boldsymbol{y}, \frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}x} = \boldsymbol{C}\boldsymbol{Q}^{-1}\boldsymbol{y} = \boldsymbol{A}\boldsymbol{y},$$

$$\nabla^2\boldsymbol{y} = \boldsymbol{D}\boldsymbol{Q}^{-1}\boldsymbol{y} = \boldsymbol{B}\boldsymbol{y}$$

In addition, the quadrature formula is

$$\boldsymbol{W}\boldsymbol{Q} = \boldsymbol{f},\ \boldsymbol{W} = \boldsymbol{f}\boldsymbol{Q}^{-1}$$

where

$$\int_0^1 x^{2i-2} x^{a-1}\mathrm{d}x = \sum_{j=1}^{N+1} W_j x_j^{2i-2}$$
$$= \frac{1}{2i-2+a} \equiv f_i$$

As an example, for the problem

$$\frac{1}{x^{a-1}}\frac{\mathrm{d}}{\mathrm{d}x}\left(x^{a-1}\frac{\mathrm{d}c}{\mathrm{d}x}\right) = \varphi^2 R\left(c\right)$$

$$\frac{\mathrm{d}c}{\mathrm{d}x}\left(0\right) = 0,\ c\left(1\right) = 1$$

orthogonal collocation is applied at the interior points

$$\sum_{i=1}^{N+1} B_{ji} c_i = \varphi^2 R\left(c_j\right),\ j = 1,\ldots,N$$

and the boundary condition solved for is

$$c_{N+1} = 1$$

The boundary condition at x = 0 is satisfied automatically by the trial function. After the solution has been obtained, the effectiveness factor η is obtained by calculating

$$\eta \equiv \frac{\int_0^1 R\left[c\left(x\right)\right] x^{a-1}\mathrm{d}x}{\int_0^1 R\left[c\left(1\right)\right] x^{a-1}\mathrm{d}x} = \frac{\sum_{i=1}^{N+1} W_j R\left(c_j\right)}{\sum_{i=1}^{N+1} W_j R\left(1\right)}$$

Note that the effectiveness factor is the average reaction rate divided by the reaction rate evaluated at the external conditions. Error bounds have been given for linear problems [83, p. 356]. For planar geometry the error is

$$\text{Error in } \eta = \frac{\varphi^{2(2N+1)}}{(2N+1)!\,(2N+2)!}$$

This method is very accurate for small N (and small φ^2); note that for finite difference methods the error goes as $1/N^2$, which does not decrease as rapidly with N. If the solution is desired at the center (a frequent situation because the center concentration can be the most extreme one), it is given by

$$c\left(0\right) = d_1 \sum_{i=1}^{N+1} \left[Q^{-1}\right]_{1i} y_i$$

The collocation points are listed in Table 8. For small N the results are usually more accurate when the weighting function in Equation 25 is $1 - x^2$. The matrices for N = 1 and N = 2 are given in Table 9 for the three geometries. Computer programs to generate matrices and a program to solve reaction diffusion problems, OCRXN, are available [3, p. 325, p. 331].

Orthogonal collocation can be applied to distillation problems. Stewart et al. [84, 85] developed a method using Hahn polynomials that retains the discrete nature of a plate-to-plate distillation column. Other work treats problems with multiple liquid phases [86]. Some of the applications to chemical engineering can be found in [87 – 90].

Table 8. Collocation points for orthogonal collocation with symmetric polynomials and W=1

	Geometry		
N	Planar	Cylindrical	Spherical
1	0.5773502692	0.7071067812	0.7745966692
2	0.3399810436	0.4597008434	0.5384693101
	0.8611363116	0.8880738340	0.9061793459
3	0.2386191861	0.3357106870	0.4058451514
	0.6612093865	0.7071067812	0.7415311856
	0.9324695142	0.9419651451	0.9491079123
4	0.1834346425	0.2634992300	0.3242534234
	0.5255324099	0.5744645143	0.6133714327
	0.7966664774	0.8185294874	0.8360311073
	0.9602898565	0.9646596062	0.9681602395
5	0.1488743390	0.2165873427	0.2695431560
	0.4333953941	0.4803804169	0.5190961292
	0.6794095683	0.7071067812	0.7301520056
	0.8650633667	0.8770602346	0.8870625998
	0.9739065285	0.9762632447	0.9782286581

7.5. Orthogonal Collocation on Finite Elements

In the method of orthogonal collocation on finite elements, the domain is first divided into elements, and then within each element orthogonal collocation is applied. Figure 27 shows the domain being divided into *NE* elements, with *NCOL* interior collocation points within each element, and *NP* = *NCOL* + 2 total points per element, giving *NT* = *NE* * (*NCOL* + 1) + 1 total number of points. Within each element a local coordinate is defined

$$u = \frac{x - x_{(k)}}{\Delta x_k},\ \Delta x_k = x_{(k+1)} - x_{(k)}$$

The reaction – diffusion equation is written as

$$\frac{1}{x^{a-1}} \frac{\mathrm{d}}{\mathrm{d}x} \left(x^{a-1} \frac{\mathrm{d}c}{\mathrm{d}x} \right) = \frac{\mathrm{d}^2 c}{\mathrm{d}x^2} + \frac{a-1}{x} \frac{\mathrm{d}c}{\mathrm{d}x} = \varphi^2 R(c)$$

and transformed to give

$$\frac{1}{\Delta x_k^2} \frac{\mathrm{d}^2}{\mathrm{d}u^2} + \frac{a-1}{x_{(k)} + u \Delta x_k} \frac{1}{\Delta x_k} \frac{\mathrm{d}c}{\mathrm{d}u} = \varphi^2 R(c)$$

The boundary conditions are typically

$$\frac{\mathrm{d}c}{\mathrm{d}x}(0) = 0,\ -\frac{\mathrm{d}c}{\mathrm{d}x}(1) = Bi_m [c(1) - c_B]$$

where Bi_m is the Biot number for mass transfer. These become

$$\frac{1}{\Delta x_1} \frac{\mathrm{d}c}{\mathrm{d}u}(u = 0) = 0,$$

in the first element;

$$-\frac{1}{\Delta x_{N\,E}} \frac{\mathrm{d}c}{\mathrm{d}u}(u = 1) = Bi_m [c(u = 1) - c_B],$$

in the last element. The orthogonal collocation method is applied at each interior collocation point.

$$\frac{1}{\Delta x_k^2} \sum_{J=1}^{N\,P} B_{I\,J} c_J + \frac{a-1}{x_{(k)} + u_I \Delta x_k} \frac{1}{\Delta x_k} \sum_{J=1}^{N\,P} A_{I\,J} c_J =$$
$$= \varphi^2 R(c_J),\ I = 2,\ldots,N\,P-1$$

The local points $i = 2, \ldots, NP - 1$ represent the interior collocation points. Continuity of the function and the first derivative between elements is achieved by taking

$$\frac{1}{\Delta x_{k-1}} \sum_{J=1}^{N\,P} A_{N\,P,J} c_J \Big|_{\text{element } k-1}$$
$$= \frac{1}{\Delta x_k} \sum_{J=1}^{N\,P} A_{1,J} c_J \Big|_{\text{element } k}$$

at the points between elements. Naturally, the computer code has only one symbol for the solution at a point shared between elements, but the derivative condition must be imposed. Finally, the boundary conditions at $x = 0$ and $x = 1$ are applied:

$$\frac{1}{\Delta x_k} \sum_{J=1}^{N\,P} A_{1,J} c_J = 0,$$

in the first element;

$$-\frac{1}{\Delta x_{N\,E}} \sum_{J=1}^{N\,P} A_{N\,P,J} c_J = Bi_m [c_{NP} - c_B],$$

in the last element.

These equations can be assembled into an overall matrix problem

$$\boldsymbol{A}\,\boldsymbol{A}\boldsymbol{c} = \boldsymbol{f}$$

Table 9. Matrices for orthogonal collocation with symmetric polynomials and $W=1-x^2$

Planar geometry, $a = 1$

$N = 1$

$$x_j = \begin{pmatrix} 0.447214 \\ 1.000000 \end{pmatrix}, \quad A_{ji} = \begin{pmatrix} -1.118034 & 1.118034 \\ -2.500000 & 2.500000 \end{pmatrix}$$

$$W_j = \begin{pmatrix} 0.833333 \\ 0.166667 \end{pmatrix}, \quad B_{ji} = \begin{pmatrix} -2.5 & 2.5 \\ -2.5 & 2.5 \end{pmatrix}, \quad Q_{ji}^{-1} = \begin{pmatrix} 1.25 & -0.25 \\ -1.25 & 1.25 \end{pmatrix}$$

$N = 2$

$$x_j = \begin{pmatrix} 0.285232 \\ 0.765055 \\ 1.000000 \end{pmatrix}, \quad W_j = \begin{pmatrix} 0.554858 \\ 0.378475 \\ 0.066667 \end{pmatrix}, \quad A_{ji} = \begin{pmatrix} -1.752962 & 2.507614 & -0.754652 \\ -1.370599 & -0.653547 & 2.024146 \\ 1.791503 & -8.791503 & 7.000000 \end{pmatrix}$$

$$B_{ji} = \begin{pmatrix} -4.73987 & 5.67713 & -0.93725 \\ 8.32288 & -23.26013 & 14.93725 \\ 19.07189 & -47.07190 & 28.00000 \end{pmatrix}, \quad Q_{ji}^{-1} = \begin{pmatrix} 1.26430 & -0.38930 & 0.125 \\ -3.42435 & 5.17435 & -1.750 \\ 2.16005 & -4.78505 & 2.625 \end{pmatrix}$$

Cylindrical geometry, $a = 2$

$N = 1$

$$x_j = \begin{pmatrix} 0.577350 \\ 1.000000 \end{pmatrix}, \quad W_j = \begin{pmatrix} 0.375 \\ 0.125 \end{pmatrix}, \quad A_{ji} = \begin{pmatrix} -1.732051 & 1.732051 \\ -3.000000 & 3.000000 \end{pmatrix}$$

$$B_{ji} = \begin{pmatrix} -6 & 6 \\ -6 & 6 \end{pmatrix}, \quad Q_{ji}^{-1} = \begin{pmatrix} 1.5 & -0.5 \\ -1.5 & 1.5 \end{pmatrix}$$

$N = 2$

$$x_j = \begin{pmatrix} 0.39377 \\ 0.80309 \\ 1.000000 \end{pmatrix}, \quad W_j = \begin{pmatrix} 0.18820 \\ 0.25624 \\ 0.05555 \end{pmatrix}, \quad A_{ji} = \begin{pmatrix} -2.53958 & 3.82562 & -1.28603 \\ -1.37768 & -1.24519 & 2.62287 \\ 1.71548 & -9.71548 & 8.00000 \end{pmatrix}$$

$$B_{ji} = \begin{pmatrix} -9.90238 & 12.29966 & -2.39728 \\ 9.03367 & -32.76429 & 23.73061 \\ 22.7575 & -65.42415 & 42.66667 \end{pmatrix}, \quad Q_{ji}^{-1} = \begin{pmatrix} 1.58808 & -0.89141 & 0.33333 \\ -3.97389 & 6.64056 & -2.66667 \\ 2.41582 & -5.74914 & 3.33333 \end{pmatrix}$$

Spherical geometry, $a = 3$

$N = 1$

$$x_j = \begin{pmatrix} 0.654654 \\ 1.000000 \end{pmatrix}, \quad W_j = \begin{pmatrix} 0.233333 \\ 0.100000 \end{pmatrix}, \quad A_{ji} = \begin{pmatrix} -2.291288 & 2.291288 \\ -3.500000 & 3.500000 \end{pmatrix}$$

$$B_{ji} = \begin{pmatrix} -10.5 & 10.5 \\ -10.5 & 10.5 \end{pmatrix}, \quad Q_{ji}^{-1} = \begin{pmatrix} 1.75 & -0.75 \\ -1.75 & 1.75 \end{pmatrix}$$

$N = 2$

$$x_j = \begin{pmatrix} 0.46885 \\ 0.83022 \\ 1.00000 \end{pmatrix}, \quad W_j = \begin{pmatrix} 0.09491 \\ 0.19081 \\ 0.04762 \end{pmatrix}, \quad A_{ji} = \begin{pmatrix} -3.19933 & 5.01517 & -1.81584 \\ -1.40870 & -1.80674 & 3.21544 \\ 1.69677 & -10.69677 & 9.00000 \end{pmatrix}$$

$$B_{ji} = \begin{pmatrix} -15.66996 & 20.03488 & -4.36492 \\ 9.96512 & -44.33004 & 34.36492 \\ 26.93285 & -86.93229 & 60.00000 \end{pmatrix}, \quad Q_{ji}^{-1} = \begin{pmatrix} 1.88193 & -1.50693 & 0.625 \\ -4.61225 & 8.36225 & -3.750 \\ 2.73032 & -6.85532 & 4.125 \end{pmatrix}$$

The form of these equations is special and is discussed by FINLAYSON [3, p. 116], who also gives the computer code to solve linear equations arising in such problems. Reaction – diffusion problems are solved by the program OCFERXN [3, p. 337]. See also the program COLSYS described below.

The error bounds of DEBOOR [91] give the following results for second-order problems solved with cubic trial functions on finite elements with

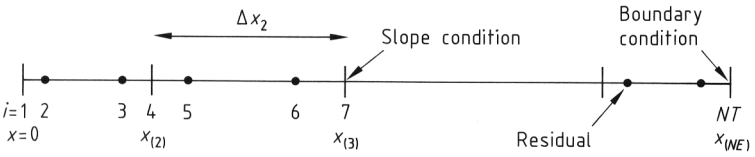

Figure 27. Grid for orthogonal collocation on finite elements

continuous first derivatives. The error at all positions is bounded by

$$\| \frac{\mathrm{d}^i}{\mathrm{d}x^i} (y - y_{\text{exact}}) \|_{\infty} \leq \text{constant} |\Delta x|^2$$

The error at the collocation points is more accurate, giving what is known as superconvergence.

$$\left| \frac{\mathrm{d}^i}{\mathrm{d}x^i} (y - y_{\text{exact}}) \right|_{\text{collocation points}} \leq \text{constant} |\Delta x|^4$$

7.6. Galerkin Finite Element Method

In the finite element method the domain is divided into elements and an expansion is made for the solution on each finite element. In the Galerkin finite element method an additional idea is introduced: the Galerkin method is used to solve the equation. The Galerkin method is explained before the finite element basis set is introduced.

To solve the problem

$$\frac{1}{x^{a-1}} \frac{\mathrm{d}}{\mathrm{d}x} \left(x^{a-1} \frac{\mathrm{d}c}{\mathrm{d}x} \right) = \varphi^2 R(c)$$

$$\frac{\mathrm{d}c}{\mathrm{d}x}(0) = 0, \; -\frac{\mathrm{d}c}{\mathrm{d}x}(1) = Bi_m [c(1) - c_B]$$

the unknown solution is expanded in a series of known functions $\{b_i(x)\}$, with unknown coefficients $\{a_i\}$.

$$c(x) = \sum_{i=1}^{NT} a_i b_i(x)$$

The series (the trial solution) is inserted into the differential equation to obtain the residual:

$$\text{Residual} = \sum_{i=1}^{NT} a_i \frac{1}{x^{a-1}} \frac{\mathrm{d}}{\mathrm{d}x} \left(x^{a-1} \frac{\mathrm{d}b_i}{\mathrm{d}x} \right) - \varphi^2 R \left[\sum_{i=1}^{NT} a_i b_i(x) \right]$$

The residual is then made orthogonal to the set of basis functions.

$$\int_0^1 b_j(x) \left\{ \sum_{i=1}^{NT} a_i \frac{1}{x^{a-1}} \frac{\mathrm{d}}{\mathrm{d}x} \left(x^{a-1} \frac{\mathrm{d}b_i}{\mathrm{d}x} \right) - \varphi^2 R \left[\sum_{i=1}^{NT} a_i b_i(x) \right] \right\} x^{a-1} \mathrm{d}x = 0$$

$$j = 1, \ldots, NT$$

This process makes the method a Galerkin method. The basis for the orthogonality condition is that a function that is made orthogonal to each member of a complete set is then zero. The residual is being made orthogonal, and if the basis functions are complete, and an infinite number of them are used, then the residual is zero. Once the residual is zero the problem is solved. It is necessary also to allow for the boundary conditions. This is done by integrating the first term of Equation 26 by parts and then inserting the boundary conditions:

$$\begin{aligned} &\int_0^1 b_j(x) \frac{1}{x^{a-1}} \frac{\mathrm{d}}{\mathrm{d}x} \left(x^{a-1} \frac{\mathrm{d}b_i}{\mathrm{d}x} \right) x^{a-1} \mathrm{d}x = \\ &\int_0^1 \frac{\mathrm{d}}{\mathrm{d}x} \left[b_j(x) x^{a-1} \frac{\mathrm{d}b_i}{\mathrm{d}x} \right] \mathrm{d}x - \int_0^1 \frac{\mathrm{d}b_j}{\mathrm{d}x} \frac{\mathrm{d}b_i}{\mathrm{d}x} x^{a-1} \mathrm{d}x \\ &= \left[b_j(x) x^{a-1} \frac{\mathrm{d}b_i}{\mathrm{d}x} \right]_0^1 - \int_0^1 \frac{\mathrm{d}b_j}{\mathrm{d}x} \frac{\mathrm{d}b_i}{\mathrm{d}x} x^{a-1} \mathrm{d}x \\ &= -\int_0^1 \frac{\mathrm{d}b_j}{\mathrm{d}x} \frac{\mathrm{d}b_i}{\mathrm{d}x} x^{a-1} \mathrm{d}x - Bi_m b_j(1) [b_i(1) - c_B] \end{aligned} \tag{27}$$

Combining this with Equation 26 gives

$$\begin{aligned} &- \sum_{i=1}^{NT} \int_0^1 \frac{\mathrm{d}b_j}{\mathrm{d}x} \frac{\mathrm{d}b_i}{\mathrm{d}x} x^{a-1} \mathrm{d}x a_i \\ &- Bi_m b_j(1) \left[\sum_{i=1}^{NT} a_i b_i(1) - c_B \right] \\ &= \varphi^2 \int_0^1 b_j(x) \left[\sum_{i=1}^{NT} a_i b_i(x) \right] x^{a-1} \mathrm{d}x \end{aligned} \tag{28}$$

$$j = 1, \ldots, NT$$

This equation defines the Galerkin method, and a solution that satisfies this equation (for all $j = 1, \ldots, \infty$) is called a weak solution. For an approximate solution the equation is written once for each member of the trial function, $j = 1, \ldots, NT$. If the boundary condition is

$$c(1) = c_B$$

then the boundary condition is used (instead of Eq. 28) for $j = NT$,

$$\sum_{i=1}^{NT} a_i b_i(1) = c_B$$

The Galerkin finite element method results when the Galerkin method is combined with a finite element trial function. Both linear and quadratic finite element approximations are described in Chapter 2. The trial functions $b_i(x)$ are then generally written as $N_i(x)$.

$$c(x) = \sum_{i=1}^{NT} c_i N_i(x)$$

Each $N_i(x)$ takes the value 1 at the point x_i and zero at all other grid points (Chap. 2). Thus c_i are the nodal values, $c(x_i) = c_i$. The first derivative must be transformed to the local coordinate system, $u = 0$ to 1 when x goes from x_i to $x_i + \Delta x$.

$$\frac{dN_j}{dx} = \frac{1}{\Delta x_e}\frac{dN_J}{du},\ dx = \Delta x_e du$$

in the e-th element. Then the Galerkin method is

$$\begin{aligned}
&-\sum_e \frac{1}{\Delta x_e}\sum_{I=1}^{NP}\int_0^1 \frac{dN_J}{du}\frac{dN_I}{du}(x_e+u\Delta x_e)^{a-1}du c_I^e \\
&-Bi_m\sum_e N_J(1)\left[\sum_{I=1}^{NP} c_I^e N_I(1) - c_1\right] \\
&= \varphi^2\sum_e \Delta x_e \int_0^1 N_J(u)\, R\left[\sum_{I=1}^{NP} c_I^e N_I(u)\right] \\
&(x_e+u\Delta x_e)^{a-1}du
\end{aligned} \tag{29}$$

The element integrals are defined as

$$B_{JI}^e = -\frac{1}{\Delta x_e}\int_0^1 \frac{dN_J}{du}\frac{dN_I}{du}(x_e+u\Delta x_e)^{a-1}du,$$

$$F_J^e = \varphi^2\Delta x_e\int_0^1 N_J(u)\, R\left[\sum_{I=1}^{NP} c_I^e N_I(u)\right] (x_e+u\Delta x_e)^{a-1}du$$

whereas the boundary element integrals are

$$BB_{JI}^e = -Bi_m N_J(1) N_I(1),$$

$$FF_J^e = -Bi_m N_J(1) c_1$$

Then the entire method can be written in the compact notation

$$\sum\nolimits_e B_{JI}^e c_I^e + \sum\nolimits_e BB_{JI}^e c_I^e = \sum\nolimits_e F_J^e + \sum\nolimits_e FF_J^e$$

The matrices for various terms are given in Table 10. This equation can also be written in the form

$$\boldsymbol{AAc} = \boldsymbol{f}$$

where the matrix $\boldsymbol{AA}$ is sparse. If linear elements are used the matrix is tridiagonal. If quadratic elements are used the matrix is pentadiagonal. Naturally the linear algebra is most efficiently carried out if the sparse structure is taken into account. Once the solution is found the solution at any point can be recovered from

$$c^e(u) = c_{I=1}^e(1-u) + c_{I=2}^e u$$

for linear elements

$$c^e(u) = c_{I=1}^e 2(u-1)\left(u-\tfrac{1}{2}\right) + c_{I=2}^e 4u(1-u) + c_{I=3}^e 2u\left(u-\tfrac{1}{2}\right)$$

for quadratic elements

Table 10. Element matrices for Galerkin method

Linear shape functions	*Quadratic shape functions*
$N_1 = 1-u,\ N_2 = u,\ \frac{dN_1}{du} = -1,\ \frac{dN_2}{du} = 1$	$N_1 = 2(u-1)\left(u-\frac{1}{2}\right),\ N_2 = 4u(1-u),\ N_3 = 2u\left(u-\frac{1}{2}\right),$ $\frac{dN_1}{du} = 4u-3,\ \frac{dN_2}{du} = 4-8u,\ \frac{dN_3}{du} = 4u-1$
$\int_0^1 \frac{dN_J}{du}\frac{dN_I}{du}du = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix},\ \int_0^1 N_J\frac{dN_I}{du}du = \begin{pmatrix} -\frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix}$	$\int_0^1 \frac{dN_J}{du}\frac{dN_I}{du}du = \frac{1}{3}\begin{pmatrix} 7 & -8 & 1 \\ -8 & 16 & -8 \\ 1 & -8 & 7 \end{pmatrix},\ \int_0^1 N_J\frac{dN_I}{du}du = \frac{1}{6}\begin{pmatrix} -3 & 4 & -1 \\ -4 & 0 & 4 \\ 1 & -4 & 3 \end{pmatrix}$
$\int_0^1 N_J N_I du = \begin{pmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{pmatrix},\ \int_0^1 N_J du = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix},\ \int_0^1 N_j u\, du = \begin{pmatrix} \frac{1}{6} \\ \frac{1}{3} \end{pmatrix}$	$\int_0^1 N_J N_I du = \frac{1}{30}\begin{pmatrix} 4 & 2 & -1 \\ 2 & 16 & 2 \\ -1 & 2 & 4 \end{pmatrix},\ \int_0^1 N_J du = \frac{1}{6}\begin{pmatrix} 1 \\ 4 \\ 1 \end{pmatrix},\ \int_0^1 N_j u\, du = \frac{1}{6}\begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix}$

Because the integrals in Equation 28 may be complicated, they are usually formed by using Gaussian quadrature. If *NG* Gauss points are used, a typical term would be

$$\int_0^1 N_J(u)\, R\left[\sum_{I=1}^{NP} c_I^e N_I(u)\right](x_e+u\Delta x_e)^{a-1}\mathrm{d}u$$
$$=\sum_{k=1}^{NG} W_k N_J(u_k)\, R\left[\sum_{I=1}^{NP} c_I^e N_I(u_k)\right]$$
$$(x_e+u_k\Delta x_e)^{a-1}$$

7.7. Cubic B-Splines

Cubic B-splines have cubic approximations within each element, but first and second derivatives continuous between elements. The functions are the same ones discussed in Chapter 2, and they can be used to solve differential equations, too. See Sincovec [92].

7.8. Adaptive Mesh Strategies

In many two-point boundary value problems, the difficulty in the problem is the formation of a boundary layer region, or a region in which the solution changes very dramatically. In such cases small mesh spacing should be used there, either with the finite difference method or the finite element method. If the region is known a priori, small mesh spacings can be assumed at the boundary layer. If the region is not known though, other techniques must be used. These techniques are known as adaptive mesh techniques. The general strategy is to estimate the error, which depends on the grid size and derivatives of the solution, and refine the mesh where the error is large.

The adaptive mesh strategy was employed by Ascher et al. [93] and by Russell and Christiansen [94]. For a second-order differential equation and cubic trial functions on finite elements, the error in the *i*-th element is given by

$$\|\text{Error}\|_i = c\Delta x_i^4 \|u^{(4)}\|_i$$

Because cubic elements do not have a nonzero fourth derivative, the third derivative in adjacent elements is used [3, p. 166]:

$$a_i = \frac{1}{\Delta x_i^3}\frac{\mathrm{d}^3 c^i}{\mathrm{d}u^3},\ a_{i+1} = \frac{1}{\Delta x_{i+1}^3}\frac{\mathrm{d}^3 c^{i+1}}{\mathrm{d}u^3}$$
$$\|u^{(4)}\|_i \approx \frac{1}{2}\left[\frac{a_i - a_{i-1}}{\frac{1}{2}(x_{i+1}-x_{i-1})} + \frac{a_{i+1}-a_i}{\frac{1}{2}(x_{i+2}-x_i)}\right]$$

Element sizes are then chosen so that the following error bounds are satisfied

$$C\Delta x_i^4 \|u^{(4)}\|_i \leq \varepsilon \text{ for all } i$$

These features are built into the code COLSYS (http://www.netlib.org/ode/).

The error expected from a method one order higher and one order lower can also be defined. Then a decision about whether to increase or decrease the order of the method can be made by taking into account the relative work of the different orders. This provides a method of adjusting both the mesh spacing (Δx, sometimes called h) and the degree of polynomial (p). Such methods are called $h-p$ methods.

7.9. Comparison

What method should be used for any given problem? Obviously the error decreases with some power of Δx, and the power is higher for the higher order methods, which suggests that the error is less. For example, with linear elements the error is

$$y(\Delta x) = y_{\text{exact}} + c_2 \Delta x^2$$

for small enough (and uniform) Δx. A computer code should be run for varying Δx to confirm this. For quadratic elements, the error is

$$y(\Delta x) = y_{\text{exact}} + c_3 \Delta x^3$$

If orthogonal collocation on finite elements is used with cubic polynomials, then

$$y(\Delta x) = y_{\text{exact}} + c_4 \Delta x^4$$

However, the global methods, not using finite elements, converge even faster [95], for example,

$$y(N) = y_{\text{exact}} + c_N \left(\frac{1}{NCOL}\right)^{NCOL}$$

Yet the workload of the methods is also different. These considerations are discussed in [3]. Here, only sweeping generalizations are given.

If the problem has a relatively smooth solution, then the orthogonal collocation method

is preferred. It gives a very accurate solution, and N can be quite small so the work is small. If the problem has a steep front in it, the finite difference method or finite element method is indicated, and adaptive mesh techniques should probably be employed. Consider the reaction – diffusion problem: as the Thiele modulus φ increases from a small value with no diffusion limitations to a large value with significant diffusion limitations, the solution changes as shown in Figure 28. The orthogonal collocation method is initially the method of choice. For intermediate values of φ, $N = 3 - 6$ must be used, but orthogonal collocation still works well (for η down to approximately 0.01). For large φ, use of the finite difference method, the finite element method, or an asymptotic expansion for large φ is better. The decision depends entirely on the type of solution that is obtained. For steep fronts the finite difference method and finite element method with adaptive mesh are indicated.

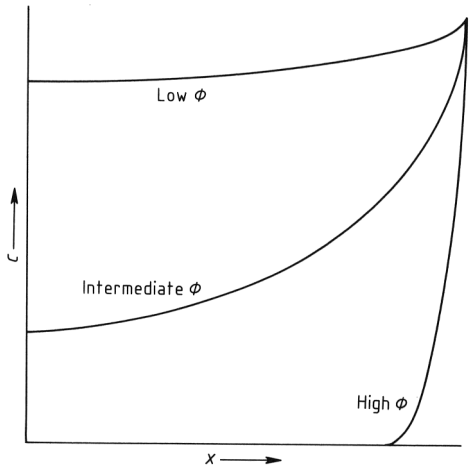

Figure 28. Concentration solution for different values of Thiele modulus

7.10. Singular Problems and Infinite Domains

If the solution being sought has a singularity, a good numerical solution may be hard to find. Sometimes even the location of the singularity may not be known [96, pp. 230 – 238]. One method of solving such problems is to refine the mesh near the singularity, by relying on the better approximation due to a smaller Δx. Another approach is to incorporate the singular trial function into the approximation. Thus, if the solution approaches $f(x)$ as x goes to zero, and $f(x)$ becomes infinite, an approximation may be taken as

$$y(x) = f(x) + \sum_{i=1}^{N} a_i y_i(x)$$

This function is substituted into the differential equation, which is solved for a_i. Essentially, a new differential equation is being solved for a new variable:

$$u(x) \equiv y(x) - f(x)$$

The differential equation is more complicated but has a better solution near the singularity (see [97, pp. 189 – 192], [98, p. 611]).

Sometimes the domain is infinite. Boundary layer flow past a flat plate is governed by the Blasius equation for stream function [99, p. 117].

$$2\frac{\mathrm{d}^3}{\mathrm{d}\eta^3} + f\frac{\mathrm{d}^2}{\mathrm{d}\eta^2} = 0$$

$$f = \frac{\mathrm{d}f}{\mathrm{d}\eta} = 0 \text{ at } \eta = 0$$

$$\frac{\mathrm{d}f}{\mathrm{d}\eta} = 1 \text{ at } \eta \to \infty$$

Because one boundary is at infinity using a mesh with a constant size is difficult! One approach is to transform the domain. For example, let

$$z = \mathrm{e}^{-\eta}$$

Then $\eta = 0$ becomes $z = 1$ and $\eta = \infty$ becomes $z = 0$. The derivatives are

$$\frac{\mathrm{d}z}{\mathrm{d}\eta} = -\mathrm{e}^{-\eta} = -z, \quad \frac{\mathrm{d}^2 z}{\mathrm{d}\eta^2} = \mathrm{e}^{-\eta} = z$$

$$\frac{\mathrm{d}f}{\mathrm{d}\eta} = \frac{\mathrm{d}f}{\mathrm{d}z}\frac{\mathrm{d}z}{\mathrm{d}\eta} = -z\frac{\mathrm{d}f}{\mathrm{d}z}$$

$$\frac{\mathrm{d}^2 f}{\mathrm{d}\eta^2} = \frac{\mathrm{d}^2 f}{\mathrm{d}z^2}\left(\frac{\mathrm{d}z}{\mathrm{d}\eta}\right)^2 + \frac{\mathrm{d}f}{\mathrm{d}z}\frac{\mathrm{d}^2 z}{\mathrm{d}\eta^2} = z^2\frac{\mathrm{d}^2 f}{\mathrm{d}z^2} + z\frac{\mathrm{d}f}{\mathrm{d}z}$$

The Blasius equation becomes

$$2\left[-z^3\frac{\mathrm{d}^3 f}{\mathrm{d}z^3} - 3z^2\frac{\mathrm{d}^2 f}{\mathrm{d}z^2} - z\frac{\mathrm{d}f}{\mathrm{d}z}\right]$$

$$+ f\left[z^2\frac{\mathrm{d}^2}{\mathrm{d}z^2} + z\frac{\mathrm{d}f}{\mathrm{d}z}\right] = 0 \qquad \text{for } 0 \le z \le 1.$$

The differential equation now has variable coefficients, but these are no more difficult to handle than the original nonlinearities.

Another approach is to use a variable mesh, perhaps with the same transformation. For example, use $z = e^{-\eta}$ and a constant mesh size in z. Then with 101 points distributed uniformly from $z = 0$ to $z = 1$, the following are the nodal points:

$$z = 0.,0.01,0.02,\ldots,0.99,1.0$$

$$\eta = \infty,4.605,3.912,\ldots,0.010,0$$

$$\Delta\eta = \infty,0.639,\ldots,0.01$$

Still another approach is to solve on a finite mesh in which the last point is far enough away that its location does not influence the solution. A location that is far enough away must be found by trial and error.

8. Partial Differential Equations

Partial differential equations are differential equations in which the dependent variable is a function of two or more independent variables. These can be time and one space dimension, or time and two or more space dimensions, or two or more space dimensions alone. Problems involving time are generally either hyperbolic or parabolic, whereas those involving spatial dimensions only are often elliptic. Because the methods applied to each type of equation are very different, the equation must first be classified as to its type. Then the special methods applicable to each type of equation are described. For a discussion of all methods, see [100 – 103]; for a discussion oriented more toward chemical engineering applications, see [104]. Examples of hyperbolic and parabolic equations include chemical reactors with radial dispersion, pressure-swing adsorption, dispersion of an effluent, and boundary value problems with transient terms added (heat transfer, mass transfer, chemical reaction). Examples of elliptic problems include heat transfer and mass transfer in two and three spatial dimensions and steady fluid flow.

8.1. Classification of Equations

A set of differential equations may be hyperbolic, elliptic, or parabolic, or it may be of mixed type. The type may change for different parameters or in different regions of the flow. This can happen in the case of nonlinear problems; an example is a compressible flow problem with both subsonic and supersonic regions. *Characteristic curves* are curves along which a discontinuity can propagate. For a given set of equations, it is necessary to determine if characteristics exist or not, because that determines whether the equations are hyperbolic, elliptic, or parabolic.

Linear Problems For linear problems, the theory summarized by Joseph et al. [105] can be used.

$$\frac{\partial}{\partial t}, \frac{\partial}{\partial x_i}, \ldots, \frac{\partial}{\partial x_n}$$

is replaced with the Fourier variables

$$i\xi_0, i\xi_1, \ldots, i\xi_n$$

If the m-th order differential equation is

$$P = \sum\nolimits_{|\alpha|=m} a_\alpha \partial^\alpha + \sum\nolimits_{|\alpha|<m} b_\alpha \partial^\alpha$$

where

$$\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_n), \; |\alpha| = \sum_{i=0}^{n} \alpha_i$$

$$\partial^\alpha = \frac{\partial^{|\alpha|}}{\partial t^{\alpha_0} \partial x_1^{\alpha_1} \ldots \partial x_n^{\alpha_n}}$$

the characteristic equation for P is defined as

$$\sum\nolimits_{|\alpha|=m} a_\alpha \sigma^\alpha = 0, \; \sigma = (\sigma_0, \sigma_1, \ldots, \sigma_n)$$
$$\sigma^\alpha = \sigma_0^{\alpha_0} \sigma_1^{\alpha_1} \ldots \sigma_n^{\alpha_n} \tag{30}$$

where σ represents coordinates. Thus only the highest derivatives are used to determine the type. The surface is defined by this equation plus a normalization condition:

$$\sum_{k=0}^{n} \sigma_k^2 = 1$$

The shape of the surface defined by Equation 30 is also related to the type: elliptic equations give rise to ellipses; parabolic equations give rise to parabolas; and hyperbolic equations give rise to hyperbolas.

$$\frac{\sigma_1^2}{a^2} + \frac{\sigma_2^2}{b^2} = 1, \text{ Ellipse}$$

$$\sigma_0 = a\sigma_1^2, \text{ Parabola}$$

$$\sigma_0^2 - a\sigma_1^2 = 0, \text{ Hyperbola}$$

If Equation 30 has no nontrivial real zeroes then the equation is called elliptic. If all the roots are

real and distinct (excluding zero) then the operator is hyperbolic.

This formalism is applied to three basic types of equations. First consider the equation arising from steady diffusion in two dimensions:

$$\frac{\partial^2 c}{\partial x^2}+\frac{\partial^2 c}{\partial y^2}=0$$

This gives

$$-\xi_1^2-\xi_2^2=-\left(\xi_2^1+\xi_2^2\right)=0$$

Thus,

$$\sigma_1^2+\sigma_2^2=1 \text{ (normalization)}$$

$$\sigma_1^2+\sigma_2^2=0 \text{ (equation)}$$

These cannot both be satisfied so the problem is elliptic. When the equation is

$$\frac{\partial^2 u}{\partial t^2}-\frac{\partial^2 u}{\partial x^2}=0$$

then

$$-\xi_0^2+\xi_1^2=0$$

Now real ξ_0 can be solved and the equation is hyperbolic

$$\sigma_0^2+\sigma_1^2=1 \text{ (normalization)}$$

$$-\sigma_0^2+\sigma_1^2=0 \text{ (equation)}$$

When the equation is

$$\frac{\partial c}{\partial t}=D\left(\frac{\partial^2 c}{\partial x^2}+\frac{\partial^2 c}{\partial y^2}\right)$$

then

$$\sigma_0^2+\sigma_1^2+\sigma_2^2=1 \text{ (normalization)}$$

$$\sigma_1^2+\sigma_2^2=0 \text{ (equation)}$$

thus we get

$$\sigma_0^2=1 \text{ (for normalization)}$$

and the characteristic surfaces are hyperplanes with t = constant. This is a parabolic case.

Consider next the telegrapher's equation:

$$\frac{\partial T}{\partial t}+\beta\frac{\partial^2 T}{\partial t^2}=\frac{\partial^2 T}{\partial x^2}$$

Replacing the derivatives with the Fourier variables gives

$$i\xi_0-\beta\xi_0^2+\xi_1^2=0$$

The equation is thus second order and the type is determined by

$$-\beta\sigma_0^2+\sigma_1^2=0$$

The normalization condition

$$\sigma_0^2+\sigma_1^2=1$$

is required. Combining these gives

$$1-(1+\beta)\,\sigma_0^2=0$$

The roots are real and the equation is hyperbolic. When $\beta = 0$

$$\xi_1^2=0$$

and the equation is parabolic.

First-order quasi-linear problems are written in the form

$$\sum_{l=0}^{n}\boldsymbol{A}_1\frac{\partial \boldsymbol{u}}{\partial x_1}=\boldsymbol{f},\ \boldsymbol{x}=(t,x_1\ldots,x_n)$$
$$\boldsymbol{u}=(u_1,u_2,\ldots,u_k) \tag{31}$$

The matrix entries $\boldsymbol{A}_1$ is a $k\times k$ matrix whose entries depend on $\boldsymbol{u}$ but not on derivatives of $\boldsymbol{u}$. Equation 31 is hyperbolic if

$$\mathrm{A}=\mathrm{A}_\mu$$

is nonsingular and for any choice of real λ_1, l = 0, . . . , n, $l \neq \mu$ the roots α_k of

$$\det\left(\alpha\boldsymbol{A}-\sum_{\substack{l=0\\ l\neq\mu}}^{n}\lambda_1\boldsymbol{A}_1\right)=0$$

are real. If the roots are complex the equation is elliptic; if some roots are real and some are complex the equation is of mixed type.

Apply these ideas to the *advection equation*

$$\frac{\partial u}{\partial t}+F(u)\frac{\partial u}{\partial x}=0$$

Thus,

$$\det(\alpha\boldsymbol{A}_0-\lambda_1\boldsymbol{A}_1)=0 \text{ or } \det(\alpha\boldsymbol{A}_1-\lambda_0\boldsymbol{A}_0)=0$$

In this case,

$$n=1,\ \boldsymbol{A}_0=1,\ \boldsymbol{A}_1=F(u)$$

Using the first of the above equations gives

$$\det(\alpha-\lambda_1 F(u))=0, \text{ or } \alpha=\lambda_1 F(u)$$

Thus, the roots are real and the equation is hyperbolic.

The final example is the heat conduction problem written as

$$\varrho C_p \frac{\partial T}{\partial t} = -\frac{\partial q}{\partial x},\ q = -k\frac{\partial T}{\partial x}$$

In this formulation the constitutive equation for heat flux is separated out; the resulting set of equations is first order and written as

$$\varrho C_p \tfrac{\partial T}{\partial t} + \tfrac{\partial q}{\partial x} = 0$$

$$k\tfrac{\partial T}{\partial x} = -q$$

In matrix notation this is

$$\begin{bmatrix} \varrho C_p & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial T}{\partial t} \\ \frac{\partial q}{\partial t} \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ k & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial T}{\partial x} \\ \frac{\partial q}{\partial t} \end{bmatrix} = \begin{bmatrix} 0 \\ -q \end{bmatrix}$$

This compares with

$$\boldsymbol{A}_0 \frac{\partial \boldsymbol{u}}{\partial x_0} + \boldsymbol{A}_1 \frac{\partial \boldsymbol{u}}{\partial x_1} = \boldsymbol{f}$$

In this case $\boldsymbol{A}_0$ is singular whereas $\boldsymbol{A}_1$ is nonsingular. Thus,

$$\det\left(\alpha \boldsymbol{A}_1 - \lambda_0 \boldsymbol{A}_0\right) = 0$$

is considered for any real λ_0. This gives

$$\begin{vmatrix} -\varrho C_p \lambda_0 & \alpha \\ k\alpha & 0 \end{vmatrix} = 0$$

or

$$\alpha^2 k = 0$$

Thus the α is real, but zero, and the equation is parabolic.

8.2. Hyperbolic Equations

The most common situation yielding hyperbolic equations involves unsteady phenomena with convection. A prototype equation is

$$\frac{\partial c}{\partial t} + \frac{\partial F(c)}{\partial x} = 0$$

Depending on the interpretation of c and $F(c)$, this can represent accumulation of mass and convection. With $F(c) = u\,c$, where u is the velocity, the equation represents a mass balance on concentration. If diffusive phenomenon are important, the equation is changed to

$$\frac{\partial c}{\partial t} + \frac{\partial F(c)}{\partial x} = D\frac{\partial^2 c}{\partial x^2} \tag{32}$$

where D is a diffusion coefficient. Special cases are the *convective diffusive equation*

$$\frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} = D\frac{\partial^2 c}{\partial x^2} \tag{33}$$

and *Burgers viscosity equation*

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \tag{34}$$

where u is the velocity and ν is the kinematic viscosity. This is a prototype equation for the Navier – Stokes equations (→ Fluid Mechanics). For *adsorption phenomena* [106, p. 202],

$$\varphi\frac{\partial c}{\partial t} + \varphi u\frac{\partial c}{\partial x} + (1-\varphi)\frac{\mathrm{d}f}{\mathrm{d}c}\frac{\partial c}{\partial t} = 0 \tag{35}$$

where φ is the void fraction and $f(c)$ gives the equilibrium relation between the concentrations in the fluid and in the solid phase. In these examples, if the diffusion coefficient D or the kinematic viscosity ν is zero, the equations are hyperbolic. If D and ν are small, the phenomenon may be essentially hyperbolic even though the equations are parabolic. Thus the numerical methods for hyperbolic equations may be useful even for parabolic equations.

Equations for several methods are given here, as taken from [107]. If the convective term is treated with a centered difference expression the solution exhibits oscillations from node to node, and these vanish only if a very fine grid is used. The simplest way to avoid the oscillations with a hyperbolic equation is to use upstream derivatives. If the flow is from left to right, this would give the following for Equations (40):

$$\frac{\mathrm{d}c_i}{\mathrm{d}t} + \frac{F(c_i) - F(c_{i-1})}{\Delta x} = D\frac{c_{i+1} - 2c_i + c_{i-1}}{\Delta x^2}$$

for Equation 34:

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} + u_i\frac{u_i - u_{i-1}}{\Delta x} = \nu\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}$$

and for Equation 35:

$$\varphi\frac{\mathrm{d}c_i}{\mathrm{d}t} + \varphi u_i\frac{c_i - c_{i-1}}{\Delta x} + (1-\varphi)\frac{\mathrm{d}f}{\mathrm{d}c}\Big|_i\frac{\mathrm{d}c_i}{\mathrm{d}t} = 0$$

If the flow were from right to left, then the formula would be

$$\frac{\mathrm{d}c_i}{\mathrm{d}t} + \frac{F(c_{i+1}) - F(c_i)}{\Delta x} = D\frac{c_{i+1} - 2c_i + c_{i-1}}{\Delta x^2}$$

If the flow could be in either direction, a local determination must be made at each node i and the appropriate formula used. The effect of using upstream derivatives is to add artificial or numerical diffusion to the model. This can be

ascertained by taking the finite difference form of the convective diffusion equation

$$\frac{dc_i}{dt}+u\frac{c_i-c_{i-1}}{\Delta x}=D\frac{c_{i+1}-2c_i+c_{i-1}}{\Delta x^2}$$

and rearranging

$$\frac{dc_i}{dt}+u\frac{c_{i+1}-c_{i-1}}{2\Delta x}$$

$$=\left(D+\frac{u\Delta x}{2}\right)\frac{c_{i+1}-2c_i+c_{i-1}}{\Delta x^2}$$

Thus the diffusion coefficient has been changed from

$$D \text{ to } D+\frac{u\Delta x}{2}$$

Expressed in terms of the cell Peclet number, $Pe_\Delta=u\Delta x/D$, this is D is changed to $D[1+Pe_\Delta/2]$

The cell Peclet number should always be calculated as a guide to the calculations. Using a large cell Peclet number and upstream derivatives leads to excessive (and artificial) smoothing of the solution profiles.

Another method often used for hyperbolic equations is the *MacCormack method*. This method has two steps; it is written here for Equation 33.

$$c_i^{*n+1}=c_i^n-\frac{u\Delta t}{\Delta x}\left(c_{i+1}^n-c_i^n\right)$$

$$+\frac{\Delta tD}{\Delta x^2}\left(c_{i+1}^n-2c_i^n+c_{i-1}^n\right)$$

$$c_i^{n+1}=\frac{1}{2}\left(c_i^n+c_i^{*n+1}\right)-\frac{u\Delta t}{2\Delta x}\left(c_i^{*n+1}-c_{i-1}^{*n+1}\right)$$

$$+\frac{\Delta tD}{2\Delta x^2}\left(c_{i+1}^{*n+1}-2c_i^{*n+1}+c_{i-1}^{*n+1}\right)$$

The concentration profile is steeper for the MacCormack method than for the upstream derivatives, but oscillations can still be present. The flux-corrected transport method can be added to the MacCormack method. A solution is obtained both with the upstream algorithm and the MacCormack method; then they are combined to add just enough diffusion to eliminate the oscillations without smoothing the solution too much. The algorithm is complicated and lengthy but well worth the effort [107 – 109].

If finite element methods are used, an explicit Taylor – Galerkin method is appropriate. For the convective diffusion equation the method is

$$\frac{1}{6}\left(c_{i+1}^{n+1}-c_{i+1}^n\right)+\frac{2}{3}\left(c_i^{n+1}-c_i^n\right)+\frac{1}{6}\left(c_{i-1}^{n+1}-c_{i-1}^n\right)$$

$$=-\frac{u\Delta t}{2\Delta x}\left(c_{i+1}^n-c_{i-1}^n\right)+\left(\frac{\Delta tD}{\Delta x^2}+\frac{u^2\Delta t^2}{2\Delta x^2}\right)\left(c_{i+1}^n\right.$$

$$\left.-2c_i^n+c_{i-1}^n\right)$$

Leaving out the $u^2\ \Delta t^2$ terms gives the Galerkin method. Replacing the left-hand side with

$$c_i^{n+1}-c_i^n$$

gives the Taylor finite difference method, and dropping the $u^2\ \Delta t^2$ terms in that gives the centered finite difference method. This method might require a small time step if reaction phenomena are important. Then the implicit Galerkin method (without the Taylor terms) is appropriate

A stability diagram for the explicit methods applied to the convective diffusion equation is shown in Figure 29. Notice that all the methods require

$$Co=\frac{u\Delta t}{\Delta x}\leq 1$$

where *Co* is the Courant number. How much *Co* should be less than one depends on the method and on $r = D\ \Delta t/\Delta x^2$, as given in Figure 29. The MacCormack method with flux correction requires a smaller time step than the MacCormack method alone (curve a), and the implicit Galerkin method (curve e) is stable for all values of *Co* and *r* shown in Figure 29 (as well as even larger values).

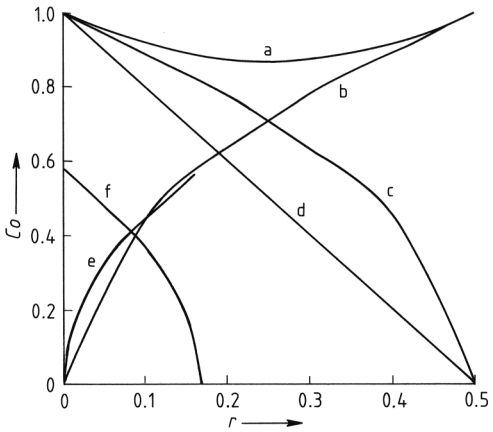

Figure 29. Stability diagram for convective diffusion equation (stable below curve)
a) MacCormack; b) Centered finite difference; c) Taylor finite difference; d) Upstream; e) Galerkin; f) Taylor – Galerkin

Each of these methods tries to avoid oscillations that would disappear if the mesh were

fine enough. For the steady convective diffusion equation these oscillations do not occur provided

$$\frac{u\Delta x}{2D} = \frac{Pe_{\Delta}}{2} <= 1 \tag{36}$$

For large u, Δx must be small to meet this condition. An alternative is to use a small Δx in regions where the solution changes drastically. Because these regions change in time, the elements or grid points must move. The criteria to move the grid points can be quite complicated, and typical methods are reviewed in [107]. The criteria include moving the mesh in a known way (when the movement is known a priori), moving the mesh to keep some property (e.g., first- or second-derivative measures) uniform over the domain, using a Galerkin or weighted residual criterion to move the mesh, and Euler – Lagrange methods which move part of the solution exactly by convection and then add on some diffusion after that.

The final illustration is for adsorption in a packed bed, or chromatography. Equation 35 can be solved when the adsorption phenomenon is governed by a Langmuir isotherm.

$$f(c) = \frac{\alpha c}{1+Kc}$$

Similar numerical considerations apply and similar methods are available [110 – 112].

8.3. Parabolic Equations in One Dimension

In this section several methods are applied to parabolic equations in one dimension: separation of variables, combination of variables, finite difference method, finite element method, and the orthogonal collocation method. Separation of variables is successful for linear problems, whereas the other methods work for linear or nonlinear problems. The finite difference, the finite element, and the orthogonal collocation methods are numerical, whereas the separation or combination of variables can lead to analytical solutions.

Analytical Solutions. Consider the diffusion equation

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2}$$

with boundary and initial conditions

$$c(x,0) = 0$$

$$c(0,t) = 1,\ c(L,t) = 0$$

A solution of the form

$$c(x,t) = T(t)\,X(x)$$

is attempted and substituted into the equation, with the terms separated to give

$$\frac{1}{DT}\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{1}{X}\frac{\mathrm{d}^2X}{\mathrm{d}x^2}$$

One side of this equation is a function of x alone, whereas the other side is a function of t alone. Thus, both sides must be a constant. Otherwise, if x is changed one side changes, but the other cannot because it depends on t. Call the constant $-\lambda$ and write the separate equations

$$\frac{\mathrm{d}T}{\mathrm{d}t} - \lambda DT,\ \frac{\mathrm{d}^2X}{\mathrm{d}x^2} - \lambda X$$

The first equation is solved easily

$$T(t) = T(0)\,\mathrm{e}^{-\lambda Dt}$$

and the second equation is written in the form

$$\frac{\mathrm{d}^2X}{\mathrm{d}x^2} + \lambda X = 0$$

Next consider the boundary conditions. If they are written as

$$c(L,t) = 1 = T(t)\,X(L)$$

$$c(0,t) = 0 = T(t)\,X(0)$$

the boundary conditions are difficult to satisfy because they are not homogeneous, i.e. with a zero right-hand side. Thus, the problem must be transformed to make the boundary conditions homogeneous. The solution is written as the sum of two functions, one of which satisfies the nonhomogeneous boundary conditions, whereas the other satisfies the homogeneous boundary conditions.

$$c(x,t) = f(x) + u(x,t)$$

$$u(0,t) = 0$$

$$u(L,t) = 0$$

Thus, $f(0) = 1$ and $f(L) = 0$ are necessary. Now the combined function satisfies the boundary conditions. In this case the function $f(x)$ can be taken as

$$f(x) = L - x$$

The equation for u is found by substituting for c in the original equation and noting that the $f(x)$ drops out for this case; it need not disappear in the general case:

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

The boundary conditions for u are

$$u(0,t) = 0$$

$$u(L,t) = 0$$

The initial conditions for u are found from the initial condition

$$u(x,0) = c(x,0) - f(x) = \frac{x}{L} - 1$$

Separation of variables is now applied to this equation by writing

$$u(x,t) = T(t)\,X(x)$$

The same equation for $T(t)$ and $X(x)$ is obtained, but with $X(0) = X(L) = 0$.

$$\frac{\mathrm{d}^2 X}{\mathrm{d}x^2} + \lambda X = 0$$

$$X(0) = X(L) = 0$$

Next $X(x)$ is solved for. The equation is an eigenvalue problem. The general solution is obtained by using e^{mx} and finding that $m^2 + \lambda = 0$; thus $m = \pm\mathrm{i}\sqrt{\lambda}$. The exponential term

$$\mathrm{e}^{\pm\mathrm{i}\sqrt{\lambda}x}$$

is written in terms of sines and cosines, so that the general solution is

$$X = B\cos\sqrt{\lambda}x + E\sin\sqrt{\lambda}x$$

The boundary conditions are

$$X(L) = B\cos\sqrt{\lambda}L + E\sin\sqrt{\lambda}L = 0$$

$$X(0) = B = 0$$

If $B = 0$, then $E \neq 0$ is required to have any solution at all. Thus, λ must satisfy

$$\sin\sqrt{\lambda}L = 0$$

This is true for certain values of λ, called eigenvalues or characteristic values. Here, they are

$$\lambda_n = n^2\pi^2/L^2$$

Each eigenvalue has a corresponding eigenfunction

$$X_n(x) = E\sin n\,\pi\,x/L$$

The composite solution is then

$$X_n(x)\,T_n(t) = E\,A\sin\frac{n\,\pi\,x}{L}\,\mathrm{e}^{-\lambda_n\,Dt}$$

This function satisfies the boundary conditions and differential equation but not the initial condition. To make the function satisfy the initial condition, several of these solutions are added up, each with a different eigenfunction, and $E\,A$ is replaced by A_n.

$$u(x,t) = \sum_{n=1}^{\infty} A_n \sin\frac{n\,\pi\,x}{L}\,\mathrm{e}^{-n^2\pi^2\,Dt/L^2}$$

The constants A_n are chosen by making $u(x, t)$ satisfy the initial condition.

$$u(x,0) = \sum_{n=1}^{\infty} A_n \sin\frac{n\,\pi\,x}{L} = \frac{x}{L} - 1$$

The residual $R(x)$ is defined as the error in the initial condition:

$$R(x) = \frac{x}{L} - 1 - \sum_{n=1}^{\infty} A_n \sin\frac{n\,\pi\,x}{L}$$

Next, the Galerkin method is applied, and the residual is made orthogonal to a complete set of functions, which are the eigenfunctions.

$$\int_0^L \left(\tfrac{x}{L} - 1\right)\sin\tfrac{m\,\pi\,x}{L}\,\mathrm{d}x$$

$$= \sum_{n=1}^{\infty} A_n \int_0^L \sin\tfrac{m\,\pi\,x}{L}\,\sin\tfrac{n\,\pi\,x}{L}\,\mathrm{d}x = \tfrac{A_m}{2}$$

The Galerkin criterion for finding A_n is the same as the least-squares criterion [3, p. 183]. The solution is then

$$c(x,t) = 1 - \frac{x}{L} + \sum_{n=1}^{\infty} A_n \sin\frac{n\,\pi\,x}{L}\,\mathrm{e}^{-n^2\pi^2 Dt/L^2}$$

This is an "exact" solution to the linear problem. It can be evaluated to any desired accuracy by taking more and more terms, but if a finite number of terms are used, some error always occurs. For large times a single term is adequate, whereas for small times many terms are needed. For small times the Laplace transform method is also useful, because it leads to solutions that converge with fewer terms. For small times, the method of combination of variables may be used

as well. For nonlinear problems, the method of separation of variables fails and one of the other methods must be used.

The method of combination of variables is useful, particularly when the problem is posed in a semi-infinite domain. Here, only one example is provided; more detail is given in [3, 113, 114]. The method is applied here to the nonlinear problem

$$\frac{\partial c}{\partial t} = \frac{\partial}{\partial x}\left[D\left(c\right)\frac{\partial c}{\partial x}\right] = D\left(c\right)\frac{\partial^2 c}{\partial x^2} + \frac{\mathrm{d}\,D\left(c\right)}{\mathrm{d}c}\left(\frac{\partial c}{\partial x}\right)^2$$

with boundary and initial conditions

$$c\left(x,0\right) = 0$$

$$c\left(0,t\right) = 1,\ c\left(\infty,t\right) = 0$$

The transformation combines two variables into one

$$c\left(x,t\right) = f\left(\eta\right)\ \text{ where }\ \eta = \frac{x}{\sqrt{4D_0 t}}$$

The use of the 4 and D_0 makes the analysis below simpler. The equation for $c\ (x,\ t)$ is transformed into an equation for $f\ (\eta)$

$$\frac{\partial c}{\partial t} = \frac{\mathrm{d}f}{\mathrm{d}\eta}\frac{\partial \eta}{\partial t},\ \frac{\partial c}{\partial x} = \frac{\mathrm{d}f}{\mathrm{d}\eta}\frac{\partial \eta}{\partial x}$$

$$\frac{\partial^2 c}{\partial x^2} = \frac{\mathrm{d}^2 f}{\mathrm{d}\eta^2}\left(\frac{\partial \eta}{\partial x}\right)^2 + \frac{\mathrm{d}f}{\mathrm{d}\eta}\frac{\partial^2 \eta}{\partial x^2}$$

$$\frac{\partial \eta}{\partial t} = -\frac{x/2}{\sqrt{4D_0 t^3}},\ \frac{\partial \eta}{\partial x} = \frac{1}{\sqrt{4D_0 t}},\ \frac{\partial^2 \eta}{\partial x^2} = 0$$

The result is

$$\frac{\mathrm{d}}{\mathrm{d}\eta}\left[K\left(c\right)\frac{\mathrm{d}f}{\mathrm{d}\eta}\right] + 2\eta\frac{\mathrm{d}f}{\mathrm{d}\eta} = 0$$

$$K\left(\mathrm{c}\right) = D\left(c\right)/D_0$$

The boundary conditions must also combine. In this case the variable η is infinite when either x is infinite or t is zero. Note that the boundary conditions on $c\ (x,\ t)$ are both zero at those points. Thus, the boundary conditions can be combined to give

$$f\left(\infty\right) = 0$$

The other boundary condition is for $x = 0$ or $\eta = 0$,

$$f\left(0\right) = 1$$

Thus, an ordinary differential equation must be solved rather than a partial differential equation. When the diffusivity is constant the solution is the well-known complementary error function:

$$c\left(x,t\right) = 1 - \mathrm{erf}\,\eta = \mathrm{erfc}\,\eta$$

$$\mathrm{erf}\,\eta = \frac{\int_0^\eta \mathrm{e}^{-\xi^2}\mathrm{d}\xi}{\int_0^\infty \mathrm{e}^{-\xi^2}\mathrm{d}\xi}$$

This is a tabulated function [23].

Numerical Methods. Numerical methods are applicable to both linear and nonlinear problems on finite and semi-infinite domains. The *finite difference method* is applied by using the method of lines [115]. In this method the same equations are used for the spatial variations of the function, but the function at a grid point can vary with time. Thus the linear diffusion problem is written as

$$\frac{\mathrm{d}c_i}{\mathrm{d}t} = D\frac{c_{i+1} - 2c_i + c_{i-1}}{\Delta x^2} \tag{37}$$

This can be written in the general form

$$\frac{\mathrm{d}\boldsymbol{c}}{\mathrm{d}t} = \boldsymbol{A}\boldsymbol{A}c$$

This set of ordinary differential equations can be solved by using any of the standard methods. The stability of explicit schemes is deduced from the theory presented in Chapter 6. The equations are written as

$$\frac{\mathrm{d}c_i}{\mathrm{d}t} = D\frac{c_{i+1} - 2c_i + c_{i-1}}{\Delta x^2} = \frac{D}{\Delta x^2}\sum_{j=1}^{n+1} B_{ij}c_j$$

where the matrix $\boldsymbol{B}$ is tridiagonal. The stability of the integration of these equations is governed by the largest eigenvalue of $\boldsymbol{B}$. If Euler's method is used for integration,

$$\Delta t\frac{D}{\Delta x^2} \leq \frac{2}{|\lambda|_{\max}}$$

The largest eigenvalue of $\boldsymbol{B}$ is bounded by the Gerschgorin theorem [14, p. 135].

$$|\lambda|_{\max} \leq \max_{2<j<n}\sum_{i=2}^{n}|B_{ji}| = 4$$

This gives the well-known stability limit

$$\Delta t\frac{D}{\Delta x^2} \leq \frac{1}{2}$$

If other methods are used to integrate in time, then the stability limit changes according to the method. It is interesting to note that the eigenvalues of Equation 37 range from $D\,\pi^2/L^2$ (smallest) to $4\,D/\Delta x^2$ (largest), depending on

the boundary conditions. Thus the problem becomes stiff as Δx approaches zero [3, p. 263].

Implicit methods can also be used. Write a finite difference form for the time derivative and average the right-hand sides, evaluated at the old and new times:

$$\frac{c_i^{n+1}-c_i^n}{\Delta t} = D\,(1-\theta)\,\frac{c_{i+1}^n-2c_i^n+c_{i-1}^n}{\Delta x^2}$$

$$+D\theta\frac{c_{i+1}^{n+1}-2c_i^{n+1}+c_{i-1}^{n+1}}{\Delta x^2}$$

Now the equations are of the form

$$-\frac{D\Delta t\theta}{\Delta x^2}c_{i+1}^{n+1}+\left[1+2\frac{D\Delta t\theta}{\Delta x^2}\right]c_i^{n+1}-\frac{D\Delta t\theta}{\Delta x^2}c_{i-1}^{n+1}$$

$$=c_i^n+\frac{D\Delta t(1-\theta)}{\Delta x^2}\left(c_{i+1}^n-2c_i^n+c_{i-1}^n\right)$$

and require solving a set of simultaneous equations, which have a tridiagonal structure. Using $\theta = 0$ gives the Euler method (as above); $\theta = 0.5$ gives the Crank – Nicolson method; $\theta = 1$ gives the backward Euler method. The stability limit is given by

$$\frac{D\Delta t}{\Delta x^2}\leq\frac{0.5}{1-2\theta}$$

whereas the oscillation limit is given by

$$\frac{D\Delta t}{\Delta x^2}\leq\frac{0.25}{1-\theta}$$

If a time step is chosen between the oscillation limit and stability limit, the solution will oscillate around the exact solution, but the oscillations remain bounded. For further discussion, see [3, p. 218].

Finite volume methods are utilized extensively in computational fluid dynamics. In this method, a mass balance is made over a cell accounting for the change in what is in the cell and the flow in and out. Figure 30 illustrates the geometry of the *i*-th cell. A mass balance made on this cell (with area A perpendicular to the paper) is

$$A\Delta x(c_i^{n+1}-c_i^n)=\Delta tA(J_{j-1/2}-J_{i+1/2})$$

where J is the flux due to convection and diffusion, positive in the $+x$ direction.

$$J=uc-D\frac{\partial c}{\partial x},\,J_{i-1/2}=u_{i-1/2}c_{i-1/2}-D\frac{c_i-c_{i-1/2}}{\Delta x}$$

The concentration at the edge of the cell is taken as

$$c_{i-1/2}=\frac{1}{2}(c_i+c_{i-1})$$

Rearrangement for the case when the velocity u is the same for all nodes gives

$$\frac{c_i^{n+1}-c_i^n}{\Delta t}+\frac{u(c_{i+1}-c_{i-1})}{2\Delta x}=\frac{D}{\Delta x^2}(c_{i+1}-2c_i+c_{i-1})$$

This is the same equation as obtained using the finite difference method. This is not always true, and the finite volume equations are easy to derive. In two- and three-dimensions, the mesh need not be rectangular, as long as it is possible to compute the velocity normal to an edge of the cell. The finite volume method is useful for applications involving filling, such as injection molding, when only part of the cell is filled with fluid. Such applications do involve some approximations, since the interface is not tracked precisely, but they are useful engineering approximations.

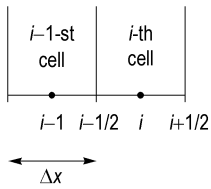

Figure 30.

The *finite element method* is handled in a similar fashion, as an extension of two-point boundary value problems by letting the solution at the nodes depend on time. For the diffusion equation the finite element method gives

$$\sum\nolimits_e\sum\nolimits_I C_{JI}^e\frac{\mathrm{d}c_I^e}{\mathrm{d}t}=\sum\nolimits_e\sum\nolimits_I B_{JI}^e\,c_I^e$$

with the mass matrix defined by

$$C_{JI}^e=\Delta x_e\int_0^1 N_J\,(u)\,N_I\,(u)\,\mathrm{d}u$$

This set of equations can be written in matrix form

$$\boldsymbol{CC}\frac{\mathrm{d}c}{\mathrm{d}t}=\boldsymbol{AA}c$$

Now the matrix $\boldsymbol{C}\,\boldsymbol{C}$ is not diagonal, so that a set of equations must be solved for each time step, even when the right-hand side is evaluated explicitly. This is not as time-consuming as it seems, however. The explicit scheme is written as

$$CC_{ji}\frac{c_i^{n+1}-c_i^n}{\Delta t}=AA_{ji}c_i^n$$

and rearranged to give

$$CC_{ji}\left(c_i^{n+1}-c_i^n\right)=\Delta tAA_{ji}c_i^n \text{ or}$$

$$\boldsymbol{CC}\left(\boldsymbol{c}^{n+1}-\boldsymbol{c}^n\right)=\Delta t\boldsymbol{AAc}$$

This is solved with an *L U* decomposition (see Section 1.1) that retains the structure of the mass matrix $\boldsymbol{C}\,\boldsymbol{C}$. Thus,

$$\boldsymbol{CC}=\boldsymbol{LU}$$

At each step, calculate

$$\boldsymbol{c}^{n+1}-\boldsymbol{c}^n=\Delta t\boldsymbol{U}^{-1}\boldsymbol{L}^{-1}\boldsymbol{AAc}^n$$

This is quick and easy to do because the inverse of $\boldsymbol{L}$ and $\boldsymbol{U}$ are simple. Thus the problem is reduced to solving one full matrix problem and then evaluating the solution for multiple right-hand sides. For implicit methods the same approach can be used, and the LU decomposition remains fixed until the time step is changed.

The *method of orthogonal collocation* uses a similar extension: the same polynomial of x is used but now the coefficients depend on time.

$$\left.\frac{\partial c}{\partial t}\right|_{x_j}=\frac{\mathrm{d}c\left(x_j,t\right)}{\mathrm{d}t}=\frac{\mathrm{d}c_j}{\mathrm{d}t}$$

Thus, for diffusion problems

$$\frac{\mathrm{d}c_j}{\mathrm{d}t}=\sum_{i=1}^{N+2}B_{ji}c_i,\ j=2,\ldots,N+1$$

This can be integrated by using the standard methods for ordinary differential equations as initial value problems. Stability limits for explicit methods are available [3, p. 204].

The method of orthogonal collocation on finite elements can also be used, and details are provided elsewhere [3, pp. 228 – 230].

The maximum eigenvalue for all the methods is given by

$$|\lambda|_{\max}=\frac{LB}{\Delta x^2} \tag{38}$$

where the values of *LB* are as follows:

Finite difference	4
Galerkin, linear elements, lumped	4
Galerkin, linear elements	12
Galerkin, quadratic elements	60
Orthogonal collocation on finite elements, cubic	36

Spectral methods employ the discrete Fourier transform (see 2 and Chebyshev polynomials on rectangular domains [116]).

In the Chebyshev collocation method, $N+1$ collocation points are used

$$x_j=\cos\frac{\pi j}{N},\ j=0,1,\ldots,N$$

As an example, consider the equation

$$\frac{\partial u}{\partial t}+f\left(u\right)\frac{\partial u}{\partial x}=0$$

An explicit method in time can be used

$$\frac{u^{n+1}-u^n}{\Delta t}+f\left(u^n\right)\left.\frac{\partial u}{\partial x}\right|^n=0$$

and evaluated at each collocation point

$$\frac{u_j^{n+1}-u_j^n}{\Delta t}+f\left(u_j^n\right)\left.\frac{\partial u}{\partial x}\right|_j^n=0$$

The trial function is taken as

$$u_j\left(t\right)=\sum_{p=0}^{N}a_p\left(t\right)\cos\frac{\pi pj}{N},\ u_j^n=u_j\left(t^n\right) \tag{39}$$

Assume that the values u_j^n exist at some time. Then invert Equation 39 using the fast Fourier transform to obtain $\{a_p\}$ for $p=0,1,\ldots,N$; then calculate S_p

$$S_p=S_{p+2}+\left(p+1\right)a_{p+1},\ 0\leq p\leq N-1$$

$$S_N=0,\ S_{N+1}=0$$

and finally

$$a_p^{(1)}=\frac{2S_p}{c_p}$$

Thus, the first derivative is given by

$$\left.\frac{\partial u}{\partial x}\right|_j=\sum_{p=0}^{N}a_p^{(1)}\left(t\right)\cos\frac{\pi pj}{N}$$

This is evaluated at the set of collocation points by using the fast Fourier transform again. Once the function and the derivative are known at each collocation point the solution can be advanced forward to the $n+1$-th time level.

The advantage of the spectral method is that it is very fast and can be adapted quite well to parallel computers. It is, however, restricted in the geometries that can be handled.

8.4. Elliptic Equations

Elliptic equations can be solved with both finite difference and finite element methods. One-dimensional elliptic problems are two-point boundary value problems and are covered in Chapter 7. Two-dimensional elliptic problems are often solved with direct methods, and iterative methods are usually used for three-dimensional problems. Thus, two aspects must be considered: how the equations are discretized to form sets of algebraic equations and how the algebraic equations are then solved.

The prototype elliptic problem is steady-state heat conduction or diffusion,

$$k\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}\right)=Q$$

possibly with a heat generation term per unit volume, Q. The boundary conditions can be

Dirichlet or 1st kind:	$T = T_1$ on boundary S_1
Neumann or 2nd kind:	$k\frac{\partial T}{\partial n} = q_2$ on boundary S_2
Robin, mixed, or 3rd kind:	$-k\frac{\partial T}{\partial n} = h\,(T-T_3)$ on boundary S_3

Illustrations are given for constant physical properties k, h, while T_1, q_2, T_3 are known functions on the boundary and Q is a known function of position. For clarity, only a two-dimensional problem is illustrated. The finite difference formulation is given by using the following nomenclature

$$T_{i,j} = T\,(i\Delta x, j\Delta y)$$

The finite difference formulation is then

$$\frac{T_{i+1,j}-2T_{i,j}+T_{i-1,j}}{\Delta x^2} + \frac{T_{i,j+1}-2T_{i,j}+T_{i,j-1}}{\Delta y^2} = Q_{i,j}/k \tag{40}$$

$$T_{i,j} = T_1 \text{ for } i,j \text{ on boundary } S_1$$

$$k\frac{\partial T}{\partial n}\Big|_{i,j} = q_2 \text{ for } i,j \text{ on boundary } S_2$$

$$-k\frac{\partial T}{\partial n}\Big|_{i,j} = h\,(T_{i,j}-T_3) \text{ for } i,j \text{ on boundary } S_3$$

If the boundary is parallel to a coordinate axis the boundary slope is evaluated as in Chapter 7, by using either a one-sided, centered difference or a false boundary. If the boundary is more irregular and not parallel to a coordinate line, more complicated expressions are needed and the finite element method may be the better method.

Equation 40 is rewritten in the form

$$2\left(1+\frac{\Delta x^2}{\Delta y^2}\right)T_{i,j} = T_{i+1,j}+T_{i-1,j}+\frac{\Delta x^2}{\Delta y^2}$$

$$(T_{i,j+1}+T_{i,j-1}) - \Delta x^2\frac{Q_{i,j}}{k}$$

And this is converted to the Gauss–Seidel iterative method.

$$2\left(1+\frac{\Delta x^2}{\Delta y^2}\right)T_{i,j}^{s+1} = T_{i+1,j}^{s}+T_{i-1,j}^{s+1}$$

$$+\frac{\Delta x^2}{\Delta y^2}\left(T_{i,j+1}^{s}+T_{i,j-1}^{s+1}\right) - \Delta x^2\frac{Q_{i,j}}{k}$$

Calculations proceed by setting a low i, computing from low to high j, then increasing i and repeating the procedure. The relaxation method uses

$$2\left(1+\frac{\Delta x^2}{\Delta y^2}\right)T_{i,j}^{*} = T_{i+1,j}^{s}+T_{i-1,j}^{s+1}+\frac{\Delta x^2}{\Delta y^2}$$

$$\left(T_{i,j+1}^{s}-T_{i,j-1}^{s+1}\right) - \Delta x^2\frac{Q_{i,j}}{k}$$

$$T_{i,j}^{s+1} = T_{i,j}^{s}+\beta\left(T_{i,j}^{*}-T_{i,j}^{s}\right)$$

If $\beta = 1$, this is the Gauss – Seidel method. If $\beta > 1$, it is overrelaxation; if $\beta < 1$, it is underrelaxation. The value of β may be chosen empirically, $0 < \beta < 2$, but it can be selected theoretically for simple problems like this [117, p. 100], [3, p. 282]. In particular, the optimal value of the iteration parameter is given by

$$-\ln\left(\beta_{\text{opt}}-1\right) \approx R$$

and the error (in solving the algebraic equation) is decreased by the factor $(1 - R)^N$ for every N iterations. For the heat conduction problem and Dirichlet boundary conditions,

$$R = \frac{\pi^2}{2n^2}$$

(when there are n points in both x and y directions). For Neumann boundary conditions, the value is

$$R = \frac{\pi^2}{2n^2}\,\frac{1}{1+\max\left[\Delta x^2/\Delta y^2, \Delta y^2/\Delta x^2\right]}$$

Iterative methods can also be based on lines (for 2D problems) or planes (for 3D problems).

Preconditioned conjugate gradient methods have been developed (see Chap. 1). In these methods a series of matrix multiplications are done iteration by iteration; and the steps lend

themselves to the efficiency available in parallel computers. In the multigrid method the problem is solved on several grids, each more refined than the previous one. In iterating between the solutions on different grids, one converges to the solution of the algebraic equations. A chemical engineering application is given in [118]. Software for a variety of these methods is available, as described below.

The Galerkin finite element method (FEM) is useful for solving elliptic problems and is particularly effective when the domain or geometry is irregular [119 – 125]. As an example, cover the domain with triangles and define a trial function on each triangle. The trial function takes the value 1.0 at one corner and 0.0 at the other corners, and is linear in between (see Fig. 31). These trial functions on each triangle are pieced together to give a trial function on the whole domain. For the heat conduction problem the method gives [3]

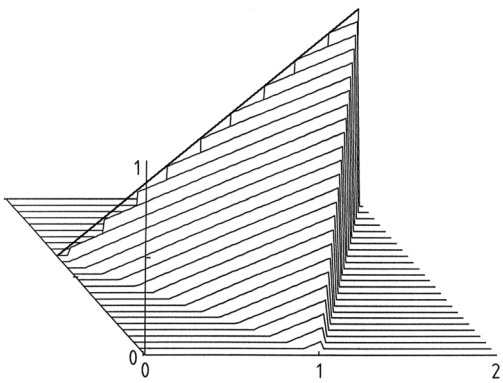

Figure 31. Finite elements trial function: linear polynomials on triangles

$$\sum_e \sum_J A^e_{IJ} T^e_J = \sum_e \sum_J F^e_I \tag{41}$$

where

$$A^e_{IJ} = -\int k \nabla N_I \cdot \nabla N_J \mathrm{d}A - \int_{C_3} h_3 N_I N_J \mathrm{d}C$$

$$F^e_I = \int N_I Q \mathrm{d}A + \int_{C_2} N_I q_2 \mathrm{d}C - \int_{C_3} N_I h_3 T_3 \mathrm{d}C$$

Also, a necessary condition is that

$$T_i = T_1 \text{ on } C_1$$

In these equations *I* and *J* refer to the nodes of the triangle forming element *e* and the summation is made over all elements. These equations represent a large set of linear equations, which are solved using matrix techniques (Chap. 1).

If the problem is nonlinear, e.g., with *k* or *Q* a function of temperature, the equations must be solved iteratively. The integrals are given for a triangle with nodes *I, J*, and *K* in counterclockwise order. Within an element,

$$T = N_I(x,y)\,T_I + N_J(x,y)\,T_J + N_K(x,y)\,T_K$$

$$N_I = \frac{a_I + b_I\,x + c_I\,y}{2\Delta}$$

$$a_I = x_J\,y_K - x_K\,y_J$$

$$b_I = y_I - y_K$$

$$c_I = x_K - x_J$$

plus permutation on I, K, J

$$2\Delta = \det \begin{bmatrix} 1 & x_I & y_I \\ 1 & x_J & y_J \\ 1 & x_K & y_K \end{bmatrix} = 2\,(\text{area of triangle})$$

$$a_I + a_J + a_K = 1$$

$$b_I + b_J + b_K = 0$$

$$c_I + c_J + c_K = 0$$

$$A^e_{IJ} = -\frac{k}{4\Delta}\,(b_I\,b_J + c_I\,c_J)$$

$$F^e_{IJ} = \frac{Q}{2}\,(a_I + b_I\,\overline{x} + c_I\,\overline{y}) = \frac{QD}{3}$$

$$\overline{x} = \frac{x_I + x_J + x_K}{3},\ \overline{y} = \frac{y_I + y_J + y_K}{3}$$

$$a_I + b_I\,\overline{x} + c_I\,\overline{y} = \frac{2}{3}\Delta$$

The trial functions in the finite element method are not limited to linear ones. Quadratic functions, and even higher order functions, are frequently used. The same considerations hold as for boundary value problems: the higher order trial functions converge faster but require more work. It is possible to refine both the mesh (*h*) and power of polynomial in the trial function (*p*) in an hp method. Some problems have constraints on some of the variables. If singularities exist in the solution, it is possible to include them in the basis functions and solve for the difference between the total solution and the singular function [126 – 129].

When applying the Galerkin finite element method, one must choose both the shape of the element and the basis functions. In two dimensions, triangular elements are usually used because it is easy to cover complicated geometries and refine parts of the mesh. However, rectangular elements sometimes have advantages, par-

ticularly when some parts of the solution do not change very much and the elements can be long. In three dimensions the same considerations apply: tetrahedral elements are frequently used, but brick elements are also possible. While linear elements (in both two and three dimensions) are usually used, higher accuracy can be obtained by using quadratic or cubic basis functions within the element. The reason is that all methods converge according to the mesh size to some power, and the power is larger when higher order elements are used. If the solution is discontinuous, or has discontinuous first derivatives, then the lowest order basis functions are used because the convergence is limited by the properties of the solution, not the finite element approximation.

One nice feature of the finite element method is the use of natural boundary conditions. In this problem the natural boundary conditions are the Neumann or Robin conditions. When using Equation 41, the problem can be solved on a domain that is shorter than needed to reach some limiting condition (such as at an outflow boundary). The externally applied flux is still applied at the shorter domain, and the solution inside the truncated domain is still valid. Examples are given in [107] and [131]. The effect of this is to allow solutions in domains that are smaller, thus saving computation time and permitting the solution in semi-infinite domains.

8.5. Parabolic Equations in Two or Three Dimensions

Computations become much more lengthy with two or more spatial dimensions, for example, the unsteady heat conduction equation

$$\varrho C_p \frac{\partial T}{\partial t} = k\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) - Q$$

or the unsteady diffusion equation

$$\frac{\partial c}{\partial t} = D\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 c}{\partial y^2}\right) - R(c)$$

In the finite difference method an explicit technique would evaluate the right-hand side at the n-th time level:

$$\rho C_p \frac{T_{i,j}^{n+1} - T_{i,j}^{n}}{\Delta t}$$

$$= \frac{k}{\Delta x^2}\left(T_{i+1,j}^{n} - 2T_{i-1,j}^{n} + T_{i-1,j}^{n}\right)$$

$$+ \frac{k}{\Delta y^2}\left(T_{i,j+1}^{n} - 2T_{i,j}^{n} + T_{i,j-1}^{n}\right) - Q$$

When $Q = 0$ and $\Delta x = \Delta y$, the time step is limited by

$$\Delta t \leq \frac{\Delta x^2 \varrho C_p}{4k} \text{ or } \frac{\Delta x^2}{4D}$$

These time steps are smaller than for one-dimensional problems. For three dimensions, the limit is

$$\Delta t \leq \frac{\Delta x^2}{6D}$$

To avoid such small time steps, which must be smaller when Δx decreases, an implicit method could be used. This leads to large sparse matrices, rather than convenient tridiagonal matrices.

8.6. Special Methods for Fluid Mechanics

The method of operator splitting is also useful when different terms in the equation are best evaluated by using different methods or as a technique for reducing a larger problem to a series of smaller problems. Here the method is illustrated by using the Navier – Stokes equations. In vector notation the equations are

$$\varrho \frac{\partial \boldsymbol{u}}{\partial t} + \varrho \boldsymbol{u} \cdot \boldsymbol{\nabla} \boldsymbol{u} = \varrho \boldsymbol{f} - \boldsymbol{\nabla} p + \mu \nabla^2 \boldsymbol{u}$$

The equation is solved in the following steps

$$\varrho \frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t} = -\varrho \boldsymbol{u}^n \cdot \boldsymbol{\nabla} \boldsymbol{u}^n + \varrho \boldsymbol{f} + \mu \nabla^2 \boldsymbol{u}^n$$

$$\nabla^2 p^{n+1} = \frac{1}{\Delta t} \boldsymbol{\nabla} \cdot \boldsymbol{u}^*$$

$$\varrho \frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^*}{\Delta t} = -\boldsymbol{\nabla} p$$

This can be done by using the finite difference [132, p. 162] or the finite element method [133 – 135].

In fluid flow problems solved with the finite element method, the basis functions for pressure and velocity are often different. This is required by the LBB condition (named after Ladyshenskaya, Brezzi, and Babuska) [134, 135]. Sometimes a discontinuous basis function is used for pressure to meet this condition. Other times a

penalty term is added, or the quadrature is done using a small number of quadrature points. Thus, one has to be careful how to apply the finite element method to the Navier – Stokes equations. Fortunately, software exists that has taken this into account.

Level Set Methods. Multiphase problems are complicated because the terms in the equations depend on which phase exists at a particular point, and the phase boundary may move or be unknown. It is desirable to compute on a fixed grid, and the level set formulation allows this. Consider a curved line in a two-dimensional problem or a curved surface in a three-dimensional problem. One defines a level set function ϕ, which is the signed distance function giving the distance from some point to the closest point on the line or surface. It defined to be negative on one side of the interface and positive on the other. Then the curve

$$\phi(x,y,z) = 0$$

represents the location of the interface. For example, in flow problems the level set function is defined as the solution to

$$\frac{\partial \phi}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} \phi = 0$$

The physics governing the velocity of the interface must be defined, and this equation is solved along with the other equations representing the problem [130 – 137].

Lattice Boltzmann Methods. Another way to solve fluid flow problems is based on a molecular viewpoint and is called the Lattice Boltzmann method [138 – 141]. The treatment here follows [142]. A lattice is defined, and one solves the following equation for $f_i\,(x,t)$, the probability of finding a molecule at the point x with speed c_i.

$$\frac{\partial f_i}{\partial t} + c_i \cdot \boldsymbol{\nabla} f_i = -\frac{f_i - f_i^{\mathrm{eq}}}{\tau}$$

The right-hand side represents a single time relaxation for molecular collisions, and τ is related to the kinematic viscosity. By means of a simple stepping algorithm, the computational algorithm is

$$f_i(\mathbf{x} + \mathbf{c}_i \Delta t, t + \Delta t) = f_i(\mathbf{x}, t) - \frac{\Delta t}{\tau}(f_i - f_i^{\mathrm{eq}})$$

Consider the lattice shown in Figure 32. The various velocities are

$$c_1 = (1,0), c_3 = (0,1), c_5 = (-1,0), c_7 = (0,-1)$$

$$c_2 = (1,1), c_4 = (-1,1), c_6 = (-1,-1), c_8 = (1,-1)$$

$$c_0 = (0,0)$$

The density, velocity, and shear stress (for some k) are given by

$$\rho = \sum\nolimits_i f_i(\boldsymbol{x},t), \rho \mathbf{u} = \sum\nolimits_i \boldsymbol{c}_i f_i(\boldsymbol{x},t),$$

$$\tau = k \sum\nolimits_i \boldsymbol{c}_i \boldsymbol{c}_i \left[f_i(\boldsymbol{x},t) - f_i^{\mathrm{eq}}(\boldsymbol{x},t) \right]$$

For this formulation, the kinematic viscosity and speed of sound are given by

$$\nu = \frac{1}{3}(\tau - \frac{1}{2}), c_s = \frac{1}{3}$$

The equilibrium distribution is

$$f_i^{\mathrm{eq}} = \rho w_i \left[1 + \frac{\boldsymbol{c}_i \cdot \mathbf{u}}{c_s^2} + \frac{(\boldsymbol{c}_i \cdot \mathbf{u})^2}{2c_s^4} - \frac{\mathbf{u} \cdot \mathbf{u}}{2c_s^2} \right]$$

where the weighting functions are

$$w_0 = \frac{4}{9}, w_1 = w_3 = w_5 = w_7 = \frac{1}{9},$$
$$w_2 = w_4 = w_6 = w_8 = \frac{1}{36}$$

With these conditions, the solution for velocity is a solution of the Navier—Stokes equation. These equations lead to a large computational problem, but it can be solved by parallel processing on multiple computers.

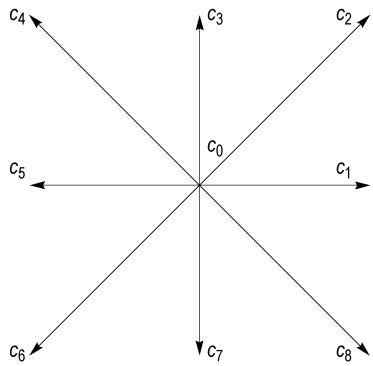

Figure 32.

8.7. Computer Software

A variety of general-purpose computer programs are available commercially. Mathematica (http://www.wolfram.com/), Maple (http://www.maplesoft.com/) and Mathcad (http://www.mathcad.com/) all have the capability of doing symbolic manipulation so that algebraic solutions can be obtained. For example, Mathematica can solve some ordinary and partial differential equations analytically; Maple can make simple graphs and do linear algebra and simple computations, and Mathcad can do simple calculations. In this section, examples are given for the use of Matlab (http://www.mathworks.com/), which is a package of numerical analysis tools, some of which are accessed by simple commands, and some of which are accessed by writing programs in C. Spreadsheets can also be used to solve simple problems. A popular program used in chemical engineering education is Polymath (http://www.polymath-software.com/), which can numerically solve sets of linear or nonlinear equations, ordinary differential equations as initial value problems, and perform data analysis and regression.

The mathematical methods used to solve partial differential equations are described in more detail in [143 – 148]. Since many computer programs are available without cost, consider the following decision points. The first decision is whether to use an approximate, engineering flow model, developed from correlations, or to solve the partial differential equations that govern the problem. Correlations are quick and easy to apply, but they may not be appropriate to your problem, or give the needed detail. When using a computer package to solve partial differential equations, the first task is always to generate a mesh covering the problem domain. This is not a trivial task, and special methods have been developed to permit importation of a geometry from a computer-aided design program. Then, the mesh must be created automatically. If the boundary is irregular, the finite element method is especially well-suited, although special embedding techniques can be used in finite difference methods (which are designed to be solved on rectangular meshes). Another capability to consider is the ability to track free surfaces that move during the computation. This phenomenon introduces the same complexity that occurs in problems with a large Peclet number, with the added difficulty that the free surface moves between mesh points, and improper representation can lead to unphysical oscillations. The method used to solve the equations is important, and both explicit and implicit methods (as described above) can be used. Implicit methods may introduce unacceptable extra diffusion, so the engineer needs to examine the solution carefully. The methods used to smooth unphysical oscillations from node to node are also important, and the engineer needs to verify that the added diffusion or smoothing does not give inaccurate solutions. Since current-day problems are mostly nonlinear, convergence is always an issue since the problems are solved iteratively. Robust programs provide several methods for convergence, each of which is best in some circumstance or other. It is wise to have a program that includes many iterative methods. If the iterative solver is not very robust, the only recourse to solving a steady-state problem may be to integrate the time-dependent problem to steady state. The solution time may be long, and the final result may be further from convergence than would be the case if a robust iterative solver were used.

A variety of computer programs is available on the internet, some of them free. First consider general-purpose programs. On the NIST web page, http://gams.nist.gov/ choose "problem decision tree", and then "differential and integral equations", then "partial differential equations". The programs are organized by type of problem (elliptic, parabolic, and hyperbolic) and by the number of spatial dimensions (one or more than one). On the Netlib web site, http://www.netlib.org/, search on "partial differential equation". The website: http://software.sandia.gov has a variety of programs available. Lau [141, 145] provides many programs in C++ (also see http://www.nr.com/). The multiphysics program Comsol Multiphysics (formerly FEMLAB) also solves many standard equations arising in Mathematical Physics.

Computational fluid dynamics (CFD) (→ Computational Fluid Dynamics) programs are more specialized, and most of them have been designed to solve sets of equations that are appropriate to specific industries. They can then include approximations and correlations for some features that would be difficult to solve for di-

rectly. Four widely used major packages are Fluent (http://www.fluent.com/), CFX (now part of ANSYS), Comsol Multiphysics (formerly FEMLAB) (http://www.comsol.com/), and ANSYS (http://www.ansys.com/). Of these, Comsol Multiphysics is particularly useful because it has a convenient graphical user interface, permits easy mesh generation and refinement (including adaptive mesh refinement), allows the user to add in phenomena and additional equations easily, permits solution by continuation methods (thus enhancing convergence), and has extensive graphical output capabilities. Other packages are also available (see http://cfd-online.com/), and these may contain features and correlations specific to the engineer's industry. One important point to note is that for turbulent flow, all the programs contain approximations, using the k-epsilon models of turbulence, or large eddy simulations; the direct numerical simulation of turbulence is too slow to apply it to very big problems, although it does give insight (independent of any approximations) that is useful for interpreting turbulent phenomena. Thus, the method used to include those turbulent correlations is important, and the method also may affect convergence or accuracy.

9. Integral Equations [149 – 155]

If the dependent variable appears under an integral sign an equation is called an integral equation; if derivatives of the dependent variable appear elsewhere in the equation it is called an integrodifferential equation. This chapter describes the various classes of equations, gives information concerning Green's functions, and presents numerical methods for solving integral equations.

9.1. Classification

Volterra integral equations have an integral with a variable limit, whereas Fredholm integral equations have a fixed limit. Volterra equations are usually associated with initial value or evolutionary problems, whereas Fredholm equations are analogous to boundary value problems. The terms in the integral can be unbounded, but still yield bounded integrals, and these equations are said to be weakly singular. A *Volterra equation of the second kind* is

$$y(t) = g(t) + \lambda \int_a^t K(t,s) y(s)\, \mathrm{d}s \tag{42}$$

whereas a *Volterra equation of the first kind* is

$$y(t) = \lambda \int_a^t K(t,s)\, y(s)\, \mathrm{d}s$$

Equations of the first kind are very sensitive to solution errors so that they present severe numerical problems.

An example of a problem giving rise to a Volterra equation of the second kind is the following heat conduction problem:

$$\varrho C_p \frac{\partial T}{\partial t} = k \frac{\partial^2 T}{\partial x^2},\ 0 \leq x < \infty,\ t > 0$$

$$T(x,0) = 0,\ \frac{\partial T}{\partial x}(0,t) = -g(t),$$

$$\lim_{x \to \infty} T(x,t) = 0,\ \lim_{x \to \infty} \frac{\partial T}{\partial x} = 0$$

If this is solved by using Fourier transforms the solution is

$$T(x,t) = \frac{1}{\sqrt{\pi}} \int_0^t g(s)\, \frac{1}{\sqrt{t-s}} \mathrm{e}^{-x^2/4(t-s)} \mathrm{d}s$$

Suppose the problem is generalized so that the boundary condition is one involving the solution T, which might occur with a radiation boundary condition or heat-transfer coefficient. Then the boundary condition is written as

$$\frac{\partial T}{\partial x} = -G(T,t),\ x = 0,\ t > 0$$

The solution to this problem is

$$T(x,t) = \frac{1}{\sqrt{\pi}} \int_0^t G(T(0,s),s)\, \frac{1}{\sqrt{t-s}}\, \mathrm{e}^{-x^2/4(t-s)} \mathrm{d}s$$

If $T^*(t)$ is used to represent $T(0, t)$, then

$$T^*(t) = \frac{1}{\sqrt{\pi}} \int_0^t G(\boldsymbol{T}(s),s)\, \frac{1}{\sqrt{t-s}} \mathrm{d}s$$

Thus the behavior of the solution at the boundary is governed by an integral equation. NAGEL and KLUGE [156] use a similar approach to solve for adsorption in a porous catalyst.

The existence and uniqueness of the solution can be proved [151, p. 30, 32].

Sometimes the kernel is of the form

$$K(t,s) = K(t-s)$$

Equations of this form are called convolution equations and can be solved by taking the Laplace transform. For the integral equation

$$Y(t) = G(t) + \lambda \int_0^t K(t-\tau)\, Y(\tau)\, \mathrm{d}\tau$$

$$K(t) * Y(t) \equiv \int_0^t K(t-\tau)\, Y(\tau)\, \mathrm{d}\tau$$

the Laplace transform is

$$y(s) = g(s) + k(s)\, y(s)$$

$$k(s)\, y(s) = \boldsymbol{L}[K(t) * Y(t)]$$

Solving this for $y(s)$ gives

$$y(s) = \frac{g(s)}{1-k(s)}$$

If the inverse transform can be found, the integral equation is solved.

A *Fredholm equation of the second kind* is

$$y(x) = g(x) + \lambda \int_a^b K(x,s)\, y(s)\, \mathrm{d}s \tag{43}$$

whereas a *Fredholm equation of the first kind* is

$$\int_a^b K(x,s)\, y(s)\, \mathrm{d}s = g(x)$$

The limits of integration are fixed, and these problems are analogous to boundary value problems. An eigenvalue problem is a homogeneous equation of the second kind.

$$y(x) = \lambda \int_a^b K(x,s)\, y(s)\, \mathrm{d}s \tag{44}$$

Solutions to this problem occur only for specific values of λ, the eigenvalues. Usually the Fredholm equation of the second or first kind is solved for values of λ different from these, which are called regular values.

Nonlinear Volterra equations arise naturally from initial value problems. For the initial value problem

$$\frac{\mathrm{d}y}{\mathrm{d}t} = F(t, y(t))$$

both sides can be integrated from 0 to t to obtain

$$y(t) = y(0) + \int_0^t F(s, y(s))\, \mathrm{d}s$$

which is a nonlinear Volterra equation. The general nonlinear Volterra equation is

$$y(t) = g(t) + \int_0^t K(t,s,y(s))\, \mathrm{d}s \tag{45}$$

Theorem [151*, p. 55].* If $g(t)$ is continuous, the kernel $K(t, s, y)$ is continuous in all variables and satisfies a Lipschitz condition

$$|K(t,s,y) - K(t,s,z)| \leq L\,|y-z|$$

then the nonlinear Volterra equation has a unique continuous solution.

A successive substitution method for its solution is

$$y_{n+1}(t) = g(t) + \int_0^t K[t,s,y_n(s)]\, \mathrm{d}s$$

Nonlinear Fredholm equations have special names. The equation

$$f(x) = \int_0^1 K[x,y,f(y)]\, \mathrm{d}y$$

is called the Urysohn equation [150 p. 208]. The special equation

$$f(x) = \int_0^1 K[x,y]\, F[y,f(y)]\, \mathrm{d}y$$

is called the Hammerstein equation [150, p. 209]. Iterative methods can be used to solve these equations, and these methods are closely tied to fixed point problems. A fixed point problem is

$$x = F(x)$$

and a successive substitution method is

$$x_{n+1} = F(x_n)$$

Local convergence theorems prove the process convergent if the solution is close enough to the answer, whereas global convergence theorems are valid for any initial guess [150, p. 229 – 231]. The successive substitution method for nonlinear Fredholm equations is

$$y_{n+1}(x) = \int_0^1 K[x,s,y_n(s)]\, \mathrm{d}s$$

Typical conditions for convergence include that the function satisfies a Lipschitz condition.

9.2. Numerical Methods for Volterra Equations of the Second Kind

Volterra equations of the second kind are analogous to initial value problems. An initial value problem can be written as a Volterra equation of the second kind, although not all Volterra equations can be written as initial value problems [151, p. 7]. Here the general nonlinear Volterra equation of the second kind is treated (Eq. 45). The simplest numerical method involves replacing the integral by a quadrature using the trapezoid rule.

$$y_n \equiv y(t_n) = g(t_n) + \Delta t \left\{ \frac{1}{2} K(t_n,t_0,y_0) + \sum_{i=1}^{n-1} K(t_n,t_i,y_i) + \frac{1}{2} K(t_n,t_n,y_n) \right\}$$

This equation is a nonlinear algebraic equation for y_n. Since y_0 is known it can be applied to solve for y_1, y_2, ... in succession. For a single integral equation, at each step one must solve a single nonlinear algebraic equation for y_n. Typically, the error in the solution to the integral equation is proportional to Δt^{μ}, and the power μ is the same as the power in the quadrature error [151, p. 97].

The stability of the method [151, p. 111] can be examined by considering the equation

$$y(t) = 1 - \lambda \int_0^t y(s)\,\mathrm{d}s$$

whose solution is

$$y(t) = \mathrm{e}^{-\lambda t}$$

Since the integral equation can be differentiated to obtain the initial value problem

$$\frac{\mathrm{d}y}{\mathrm{d}t} = -\lambda y,\ y(0) = 1$$

the stability results are identical to those for initial value methods. In particular, using the trapezoid rule for integral equations is identical to using this rule for initial value problems. The method is A-stable.

Higher order integration methods can also be used [151, p. 114, 124]. When the kernel is infinite at certain points, i.e., when the problem has a weak singularity, see [151, p. 71, 151].

9.3. Numerical Methods for Fredholm, Urysohn, and Hammerstein Equations of the Second Kind

Whereas Volterra equations could be solved from one position to the next, like initial value differential equations, Fredholm equations must be solved over the entire domain, like boundary value differential equations. Thus, large sets of equations will be solved and the notation is designed to emphasize that.

The methods are also based on quadrature formulas. For the integral

$$I(\varphi) = \int_a^b \varphi(y)\,\mathrm{d}y$$

a quadrature formula is written:

$$I(\varphi) = \sum_{i=0}^{n} w_i \varphi(y_i)$$

Then the integral Fredholm equation can be rewritten as

$$f(x) - \lambda \sum_{i=0}^{n} w_i K(x,y_i) f(y_i) = g(x), \quad a \le x \le b \tag{46}$$

If this equation is evaluated at the points $x = y_j$,

$$f(y_j) - \lambda \sum_{i=0}^{n} w_i K(y_j,y_i) f(y_i) = g(y_i)$$

is obtained, which is a set of linear equations to be solved for $\{f(y_j)\}$. The solution at any point is then given by Equation 46.

A common type of integral equation has a singular kernel along $x = y$. This can be transformed to a less severe singularity by writing

$$\int_a^b K(x,y) f(y)\,\mathrm{d}y = \int_a^b K(x,y)[f(y) - f(x)]\,\mathrm{d}y + \int_a^b K(x,y) f(x)\,\mathrm{d}y = \int_a^b K(x,y)[f(y) - f(x)]\,\mathrm{d}y + f(x) H(x)$$

where

$$H(x) = \int_a^b K(x,y) f(x)\,\mathrm{d}y$$

is a known function. The integral equation is then replaced by

$$f(x) = g(x) + \sum_{i=0}^{n} w_i K(x,y_i)\,[f(y_i) - f(x)] + f(x)\,H(x)$$

Collocation methods can be applied as well [149, p. 396]. To solve integral Equation 43 expand f in the function

$$f = \sum_{i=0}^{n} a_i \varphi_i(x)$$

Substitute f into the equation to form the residual

$$\sum_{i=0}^{n} a_i \varphi_i(x) - \lambda \sum_{i=0}^{n} a_i \int_a^b K(x,y)\,\varphi_i(y)\,\mathrm{d}y = g(x)$$

Evaluate the residual at the collocation points

$$\sum_{i=0}^{n} a_i \varphi_i(x_j) - \lambda \sum_{i=0}^{n} a_i \int_a^b K(x_j,y)\,\varphi_i(y)\,\mathrm{d}y = g(x_j)$$

The expansion can be in piecewise polynomials, leading to a collocation finite element method, or global polynomials, leading to a global approximation. If orthogonal polynomials are used then the quadratures can make use of the accurate Gaussian quadrature points to calculate the integrals. Galerkin methods are also possible [149, p. 406]. Mills et al. [157] consider reaction – diffusion problems and say the choice of technique cannot be made in general because it is highly dependent on the kernel.

When the integral equation is nonlinear, iterative methods must be used to solve it. Convergence proofs are available, based on Banach's contractive mapping principle. Consider the Urysohn equation, with $g(x) = 0$ without loss of generality:

$$f(x) = \int_a^b F[x,y,f(y)]\,\mathrm{d}y$$

The kernel satisfies the Lipschitz condition

$$\max_{a \le x,y \le b} |F[x,y,f(y)] - F[x,z,f(z)]| \le K|y-z|$$

Theorem [150, p. 214]. If the constant K is < 1 and certain other conditions hold, the successive substitution method

$$f_{n+1}(x) = \int_a^b F[x,y,f_n(y)]\,\mathrm{d}y, n = 0,1\ldots$$

converges to the solution of the integral equations.

9.4. Numerical Methods for Eigenvalue Problems

Eigenvalue problems are treated similarly to Fredholm equations, except that the final equation is a matrix eigenvalue problem instead of a set of simultaneous equations. For example,

$$\sum_{i=1}^{n} w_i K(y_i,y_i)\,f(y_i) = \lambda f(y_j),$$

$$i = 0,1,\ldots,n$$

leads to the matrix eigenvalue problem

$$\boldsymbol{K\,D\,f} = \lambda \boldsymbol{f}$$

Where $\boldsymbol{D}$ is a diagonal matrix with $D_{ii} = w_i$.

9.5. Green's Functions [158 – 160]

Integral equations can arise from the formulation of a problem by using Green's function. For example, the equation governing heat conduction with a variable heat generation rate is represented in differential forms as

$$\frac{\mathrm{d}^2 T}{\mathrm{d}x^2} = \frac{Q(x)}{k},\ T(0) = T(1) = 0$$

In integral form the same problem is [149, pp. 57 – 60]

$$T(x) = \frac{1}{k}\int_0^1 G(x,y)\,Q(y)\,\mathrm{d}y$$

$$G(x,y) = \begin{cases} -x(1-y) & x \le y \\ -y(1-x) & y \le x \end{cases}$$

Green's functions for typical operators are given below.

For the Poisson equation with solution decaying to zero at infinity

$$\nabla^2 \psi = -4\pi\varrho$$

the formulation as an integral equation is

$$\psi(\boldsymbol{r}) = \int_V \rho(\boldsymbol{r}_0)\,G(\boldsymbol{r},\boldsymbol{r}_0)\,\mathrm{d}V_0$$

where Green's function is [50, p. 891]

$G(\boldsymbol{r},\boldsymbol{r}_0) = \frac{1}{r}$ in three dimensions

$= -2\ln r$ in two dimensions

where $r = \sqrt{(x-x_0)^2+(y-y_0)^2+(z-z_0)^2}$ in three dimensions

and $r = \sqrt{(x-x_0)^2+(y-x_0)^2}$ in two dimensions

For the problem

$$\frac{\partial u}{\partial t} = D\nabla^2 u,\ u = 0 \text{ on } S,$$

with a point source at x_0,y_0,z_0

Green's function is [44, p. 355]

$$u = \frac{1}{8[\pi D(t-\tau)]^{3/2}}$$

$$\mathrm{e}^{-\left[(x-x_0)^2+(y-y_0)^2+(z-z_0)^2\right]/4\,D(t-\tau)}$$

When the problem is

$$\frac{\partial c}{\partial t} = D\nabla^2 c$$

$$c = f\,(x,y,z) \text{ in } S \text{ at } t = 0$$

$$c = \varphi\,(x,y,z) \text{ on } S, t>0$$

the solution can be represented as [44, p. 353]

$$c = \int\int\int (u)_{\tau=0} f\,(x,y,z)\,\mathrm{d}x\mathrm{d}y\mathrm{d}z$$

$$+D\int_0^t\int\int\int \varphi\,(x,y,z,\tau)\,\frac{\partial u}{\partial n}\mathrm{d}S\mathrm{d}t$$

When the problem is two dimensional,

$$u = \frac{1}{\sqrt{4\pi D(t-\tau)}}\mathrm{e}^{-\left[(x-x_0)^2+(y-y_0)^2\right]/4D(t-\tau)}$$

$$c = \int\int (u)_{\tau=0} f\,(x,y)\,\mathrm{d}x\mathrm{d}y$$

$$+D\int_0^t\int \varphi\,(x,y,\tau)\,\frac{\partial u}{\partial n}\mathrm{d}C\mathrm{d}t$$

For the following differential equation and boundary conditions

$$\frac{1}{x^{a-1}}\frac{\mathrm{d}}{\mathrm{d}x}\left(x^{\mathrm{a}-1}\frac{\mathrm{dc}}{\mathrm{d}x}\right) = f\,[x,c\,(x)]\,,$$

$$\frac{\mathrm{dc}}{\mathrm{d}x}\,(0) = 0,\ \frac{2}{Sh}\frac{\mathrm{dc}}{\mathrm{d}x}\,(1) + c\,(1) = g$$

where *Sh* is the Sherwood number, the problem can be written as a Hammerstein integral equation:

$$c\,(x) = g - \int_0^1 G\,(x,y,Sh)\,f\,[y,c\,(y)]\,y^{a-1}\mathrm{d}y$$

Green's function for the differential operators are [163]

$a = 1$

$$G\,(x,y,Sh) = \begin{cases} 1+\frac{2}{Sh}-x, & y\le x \\ 1+\frac{2}{Sh}-y, & x<y \end{cases}$$

$a = 2$

$$G\,(x,y,Sh) = \begin{cases} \frac{2}{Sh}-\ln x, & y\le x \\ \frac{2}{Sh}-\ln y, & x<y \end{cases}$$

$a = 3$

$$G\,(x,y,Sh) = \begin{cases} \frac{2}{Sh}+\frac{1}{x}-1, & y\le x \\ \frac{2}{Sh}+\frac{1}{y}-1, & x<y \end{cases}$$

Green's functions for the reaction diffusion problem were used to provide computable error bounds by Ferguson and Finlayson [163].

If Green's function has the form

$$K\,(x,y) = \begin{cases} u\,(x)\,v\,(y) & 0\le y\le x \\ u\,(y)\,v\,(x) & x\le y\le 1 \end{cases}$$

the problem

$$f\,(x) = \int_0^1 K\,(x,y)\,F\,[y,f\,(y)]\,\mathrm{d}y$$

may be written as

$$f\,(x) - \int_0^x [u\,(x)\,v\,(y) - u\,(y)\,v\,(x)]\cdot$$

$$f\,[y,f\,(y)]\,\mathrm{d}y = \alpha v\,(x)$$

where

$$\alpha = \int_0^1 u\,(y)\,F\,[y,f\,(y)]\,\mathrm{d}y$$

Thus, the problem ends up as one directly formulated as a fixed point problem:

$$f = \Phi\,(f)$$

When the problem is the diffusion – reaction one, the form is

$$c\,(x) = g - \int_0^x [u\,(x)\,v\,(y) - u\,(y)\,v\,(x)]$$

$$f\,[y,c\,(y)]\,y^{a-1}\mathrm{d}y - \alpha v\,(x)$$

$$\alpha = \int_0^1 u\,(y)\,f\,[y,c\,(y)]\,y^{a-1}\mathrm{d}y$$

Dixit and Taularidis [164] solved problems involving Fischer – Tropsch synthesis reactions in a catalyst pellet using a similar method.

9.6. Boundary Integral Equations and Boundary Element Method

The boundary element method utilizes Green's theorem and integral equations. Here, the method is described briefly for the following boundary value problem in two or three dimensions

$$\nabla^2\varphi = 0,\ \varphi = f_1 \text{ on } S_1,\ \frac{\partial\varphi}{\partial n} = f_2 \text{ on } S_2$$

Green's theorem (see page 46) says that for any functions sufficiently smooth

$$\int_V (\varphi\nabla^2\psi - \psi\nabla^2\varphi)\,\mathrm{d}V = \int_S \left(\varphi\frac{\partial\psi}{\partial n} - \psi\frac{\partial\varphi}{\partial n}\right)\mathrm{d}S$$

Suppose the function ψ satisfies the equation

$$\nabla^2\psi = 0$$

In two and three dimensions, such a function is

$$\psi = \ln r,\ r = \sqrt{(x-x_0)^2+(y-y_0)^2}$$

in two dimensions

$$\psi = \tfrac{1}{r},\ r = \sqrt{(x-x_0)^2+(y-y_0)^2+(z-z_0)^2}$$

in three dimensions

where $\{x_0, y_0\}$ or $\{x_0, y_0, z_0\}$ is a point in the domain. The solution φ also satisfies

$$\nabla^2\varphi = 0$$

so that

$$\int_S \left(\varphi\frac{\partial\psi}{\partial n} - \psi\frac{\partial\varphi}{\partial n}\right)\mathrm{d}S = 0$$

Consider the two-dimensional case. Since the function ψ is singular at a point, the integrals must be carefully evaluated. For the region shown in Figure 33, the domain is $S = S_1 + S_2$; a small circle of radius r_0 is placed around the point P at x_0, y_0. Then the full integral is

$$\int_S \left(\varphi\frac{\partial\ln r}{\partial n} - \ln r\frac{\partial\varphi}{\partial n}\right)\mathrm{d}S + \int_{\theta=0}^{\theta=2\pi} \left(\varphi\frac{\partial\ln r_0}{\partial n} - \ln r_0\frac{\partial\varphi}{\partial n}\right) r_0\mathrm{d}\theta = 0$$

As r_0 approaches 0,

$$\lim_{r_0\to 0} r_0\ln r_0 = 0$$

and

$$\lim_{r_0\to 0} \int_{\theta=0}^{\theta=2\pi} \varphi\frac{\partial\ln r_0}{\partial n} r_0\mathrm{d}\theta = -\varphi(P)\,2\pi$$

Thus for an internal point,

$$\varphi(P) = \frac{1}{2\pi}\int_S \left(\varphi\frac{\partial\ln r}{\partial n} - \ln r\frac{\partial\varphi}{\partial n}\right)\mathrm{d}S \tag{47}$$

If P is on the boundary, the result is [165, p. 464]

$$\varphi(P) = \frac{1}{\pi}\int_S \left(\varphi\frac{\partial\ln r}{\partial n} - \ln r\frac{\partial\varphi}{\partial n}\right)\mathrm{d}S$$

Putting in the boundary conditions gives

$$\pi\varphi(P) = \int_{S_1} \left(f_1\frac{\partial\ln r}{\partial n} - \ln r\frac{\partial\varphi}{\partial n}\right)\mathrm{d}S + \int_{S_2} \left(\varphi\frac{\partial\ln r}{\partial n} - f_2\ln r\right)\mathrm{d}S \tag{48}$$

This is an integral equation for φ on the boundary. Note that the order is one less than the original differential equation. However, the integral equation leads to matrices that are dense rather than banded or sparse, so some of the advantage of lower dimension is lost. Once this integral equation (Eq. 48) is solved to find φ on the boundary, it can be substituted in Equation 47 to find φ anywhere in the domain.

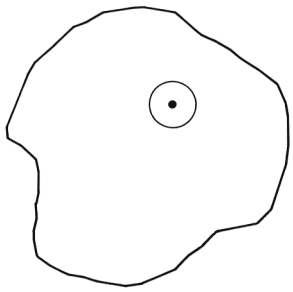

Figure 33. Domain with singularity at P

In the boundary finite element method, both the function and its normal derivative along the boundary are approximated.

$$\varphi = \sum_{j=1}^{N}\varphi_j N_j(\xi),\ \frac{\partial\varphi}{\partial n} = \sum_{j=1}^{N}\left(\frac{\partial\varphi}{\partial n}\right)_j N_j(\xi)$$

One choice of trial functions can be the piecewise constant functions shown in Figure 34. The integral equation then becomes

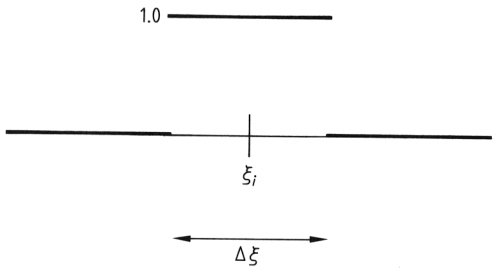

Figure 34. Trial function on boundary for boundary finite element method

$$\pi\varphi_i \sum_{j=1}^{N} \left[\varphi_j \int_{s_j} \frac{\partial \ln r_i}{\partial n} \mathrm{d}S - \left(\frac{\partial \varphi_j}{\partial n} \right) \int_{s_j} \ln r_i \right] \mathrm{d}s$$

The function φ_j is of course known along s_1, whereas the derivative $\partial\varphi_j/\partial n$ is known along s_2. This set of equations is then solved for φ_i and $\partial\varphi_i/\partial n$ along the boundary. This constitutes the boundary integral method applied to the Laplace equation.

If the problem is Poisson's equation

$$\nabla^2\varphi = g\,(x,y)$$

Green's theorem gives

$$\int_S \left(\varphi \frac{\partial \ln r}{\partial n} - \ln r \frac{\partial \varphi}{\partial n} \right) \mathrm{d}S + \int_A g \ln r \, \mathrm{d}A = 0$$

Thus, for an internal point,

$$2\pi\varphi\,(P) = \int_S \left(\varphi \frac{\partial \ln r}{\partial r} - \ln r \frac{\partial \varphi}{\partial n} \right) \mathrm{d}S + \int_A g \ln r \mathrm{d}A \tag{49}$$

and for a boundary point,

$$\pi\varphi\,(P) = \int_S \left(\varphi \frac{\partial \ln r}{\partial n} - \ln r \frac{\partial \varphi}{\partial n} \right) \mathrm{d}S + \int_A g \ln r \mathrm{d}A \tag{50}$$

If the region is nonhomogeneous this method can be used [165, p. 475], and it has been applied to heat conduction by Hsieh and Shang [166]. The finite element method can be applied in one region and the boundary finite element method in another region, with appropriate matching conditions [165, p. 478]. If the problem is nonlinear, then it is more difficult. For example, consider an equation such as Poisson's in which the function depends on the solution as well

$$\nabla^2\varphi = g\,(x,y,\varphi)$$

Then the integral appearing in Equation 50 must be evaluated over the entire domain, and the solution in the interior is given by Equation 49. For further applications, see [167] and [168].

10. Optimization

We provide a survey of systematic methods for a broad variety of optimization problems. The survey begins with a general classification of mathematical optimization problems involving continuous and discrete (or integer) variables. This is followed by a review of solution methods of the major types of optimization problems for continuous and discrete variable optimization, particularly nonlinear and mixed-integer nonlinear programming. In addition, we discuss direct search methods that do not require derivative information as well as global optimization methods. We also review extensions of these methods for the optimization of systems described by differential and algebraic equations.

10.1. Introduction

Optimization is a key enabling tool for decision making in chemical engineering [306]. It has evolved from a methodology of academic interest into a technology that continues to make significant impact in engineering research and practice. Optimization algorithms form the core tools for a) experimental design, parameter estimation, model development, and statistical analysis; b) process synthesis analysis, design, and retrofit; c) model predictive control and real-time optimization; and d) planning, scheduling, and the integration of process operations into the supply chain [307, 308].

As shown in Figure 35, optimization problems that arise in chemical engineering can be classified in terms of continuous and discrete variables. For the former, nonlinear programming (NLP) problems form the most general case, and widely applied specializations include linear programming (LP) and quadratic programming (QP). An important distinction for NLP is whether the optimization problem is convex or nonconvex. The latter NLP problem may

have multiple local optima, and an important question is whether a global solution is required for the NLP. Another important distinction is whether the problem is assumed to be differentiable or not.

Mixed-integer problems also include discrete variables. These can be written as mixed-integer nonlinear programs (MINLP) or as mixed-integer linear programs (MILP) if all variables appear linearly in the constraint and objective functions. For the latter an important case occurs when all the variables are integer; this gives rise to an integer programming (IP) problem. IPs can be further classified into many special problems (e.g., assignment, traveling salesman, etc.), which are not shown in Figure 35. Similarly, the MINLP problem also gives rise to special problem classes, although here the main distinction is whether its relaxation is convex or nonconvex.

The ingredients of formulating optimization problems include a mathematical model of the system, an objective function that quantifies a criterion to be extremized, variables that can serve as decisions, and, optionally, inequality constraints on the system. When represented in algebraic form, the general formulation of discrete/continuous optimization problems can be written as the following mixed-integer optimization problem:

$$\begin{aligned} \text{Min} \quad & f(x,y) \\ \text{s.t.} \quad & h(x,y) = 0 \\ & g(x,y) \leq 0 \\ & x \in \Re^n, y \in \{0,1\}^t \end{aligned} \tag{51}$$

where $f(x, y)$ is the objective function (e.g., cost, energy consumption, etc.), $h(x, y) = 0$ are the equations that describe the performance of the system (e.g., material balances, production rates), the inequality constraints $g(x, y) \leq 0$ can define process specifications or constraints for feasible plans and schedules, and s.t. denotes subject to. Note that the operator Max $f(x)$ is equivalent to Min $-f(x)$. We define the real n-vector x to represent the continuous variables while the t-vector y represents the discrete variables, which, without loss of generality, are often restricted to take 0–1 values to define logical or discrete decisions, such as assignment of equipment and sequencing of tasks. (These variables can also be formulated to take on other integer values as well.) Problem 51 corresponds to a mixed-integer nonlinear program (MINLP) when any of the functions involved are nonlinear. If all functions are linear it corresponds to a mixed-integer linear program (MILP). If there are no 0–1 variables, then problem 51 reduces to a nonlinear program 52 or linear program 65 depending on whether or not the functions are linear.

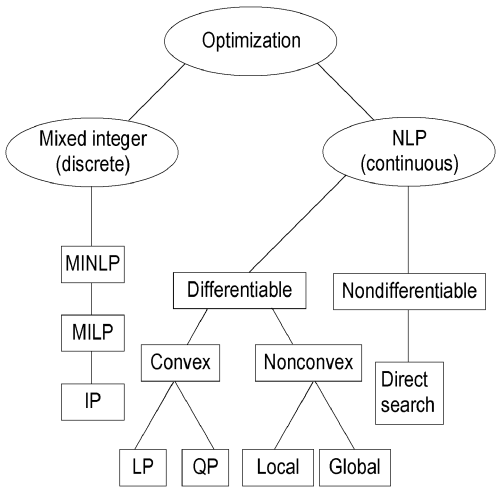

Figure 35. Classes of optimization problems and algorithms

We first start with continuous variable optimization and consider in the next section the solution of NLPs with differentiable objective and constraint functions. If only local solutions are required for the NLP, then very efficient large-scale methods can be considered. This is followed by methods that are not based on local optimality criteria; we consider direct search optimization methods that do not require derivatives, as well as deterministic global optimization methods. Following this, we consider the solution of mixed-integer problems and outline the main characteristics of algorithms for their solution. Finally, we conclude with a discussion of optimization modeling software and its implementation in engineering models.

10.2. Gradient-Based Nonlinear Programming

For continuous variable optimization we consider problem 51 without discrete variables y.

The general NLP problem 52 is presented below:

$$\begin{array}{ll} \text{Min} & f(x) \\ \text{s.t.} & h(x) = 0 \\ & g(x) \leq 0 \end{array} \qquad (52)$$

and we assume that the functions $f(x)$, $h(x)$, and $g(x)$ have continuous first and second derivatives. A key characteristic of problem 52 is whether the problem is convex or not, i.e., whether it has a convex objective function and a convex feasible region. A function $\phi(x)$ of x in some domain X is convex if and only if for all points $x_1, x_2 \in X$:

$$\phi[\alpha x_1 + (1-\alpha)x_2] \leq \alpha\phi(x_1) + (1-\alpha)\phi(x_2) \qquad (53)$$

holds for all $\alpha \in (0, 1)$. If $\phi(x)$ is differentiable, then an equivalent definition is:

$$\phi(x_1) + \nabla\phi(x_1)^T (x - x_1) \leq \phi(x) \qquad (54)$$

Strict convexity requires that the inequalities 53 and 54 be strict. Convex feasible regions require $g(x)$ to be a convex function and $h(x)$ to be linear. If 52 is a convex problem, than any local solution is guaranteed to be a global solution to 52. Moreover, if the objective function is strictly convex, then this solution x^* is unique. On the other hand, nonconvex problems may have multiple local solutions, i.e., feasible solutions that minimize the objective function within some neighborhood about the solution.

We first consider methods that find only local solutions to nonconvex problems, as more difficult (and expensive) search procedures are required to find a global solution. Local methods are currently very efficient and have been developed to deal with very large NLPs. Moreover, by considering the structure of convex NLPs (including LPs and QPs), even more powerful methods can be applied. To study these methods, we first consider conditions for local optimality.

Local Optimality Conditions – A Kinematic Interpretation. Local optimality conditions are generally derived from gradient information from the objective and constraint functions. The proof follows by identifying a local minimum point that has no feasible descent direction. Invoking a theorem of the alternative (e.g., Farkas – Lemma) leads to the celebrated Karush – Kuhn – Tucker (KKT) conditions [169]. Instead of a formal development of these conditions, we present here a more intuitive, kinematic illustration. Consider the contour plot of the objective function $f(x)$ given in Figure 36 as a smooth valley in space of the variables x_1 and x_2. For the contour plot of this unconstrained problem, Min $f(x)$, consider a ball rolling in this valley to the lowest point of $f(x)$, denoted by x^*. This point is at least a local minimum and is defined by a point with zero gradient and at least nonnegative curvature in all (nonzero) directions p. We use the first derivative (gradient) vector $\nabla f(x)$ and second derivative (Hessian) matrix $\nabla_{xx} f(x)$ to state the necessary first- and second-order conditions for unconstrained optimality:

$$\nabla_x f(x^*) = 0 \quad p^T \nabla_{xx} f(x^*) p \geq 0 \quad \text{for all } p \neq 0 \qquad (55)$$

These necessary conditions for local optimality can be strengthened to sufficient conditions by making the inequality in the relations 55 strict (i.e., positive curvature in all directions). Equivalently, the sufficient (necessary) curvature conditions can be stated as: $\nabla_{xx} f(x^*)$ has all positive (nonnegative) eigenvalues and is therefore defined as a positive (semi-)definite matrix.

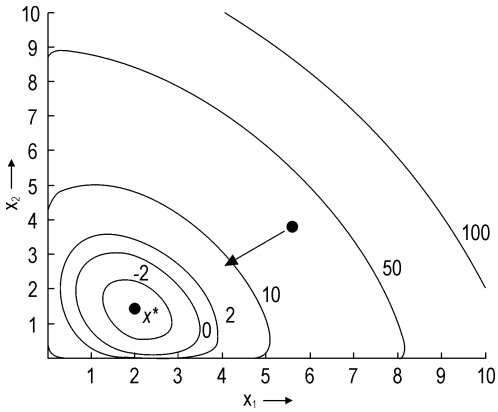

Figure 36. Unconstrained minimum

Now consider the imposition of inequality $[g(x) \leq 0]$ and equality constraints $[h(x) = 0]$ in Figure 37. Continuing the kinematic interpretation, the inequality constraints $g(x) \leq 0$ act as "fences" in the valley, and equality constraints

$h(x)=0$ as "rails". Consider now a ball, constrained on a rail and within fences, to roll to its lowest point. This stationary point occurs when the normal forces exerted by the fences $[-\nabla g\,(x^*)]$ and rails $[-\nabla h\,(x^*)]$ on the ball are balanced by the force of "gravity" $[-\nabla f\,(x^*)]$. This condition can be stated by the following *KKT necessary conditions* for constrained optimality:

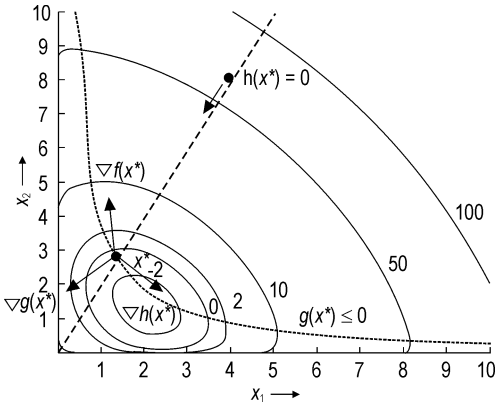

Figure 37. Constrained minimum

Stationarity Condition: It is convenient to define the Lagrangian function $L(x,\ \lambda,\ \nu)= f(x)+g(x)^T\lambda+h(x)^T\nu$ along with "weights" or multipliers λ and ν for the constraints. These multipliers are also known as "dual variables" and "shadow prices". The stationarity condition (balance of forces acting on the ball) is then given by:

$$\nabla L(x,\lambda,\nu)=\nabla f\,(x)+\nabla h\,(x)\,\lambda+\nabla g\,(x)\,\nu=0 \qquad (56)$$

Feasibility: Both inequality and equality constraints must be satisfied (ball must lie on the rail and within the fences):

$$h\,(x)=0,\quad g\,(x)\leq 0 \qquad (57)$$

Complementarity: Inequality constraints are either strictly satisfied (active) or inactive, in which case they are irrelevant to the solution. In the latter case the corresponding KKT multiplier must be zero. This is written as:

$$\nu^T g\,(x)=0,\quad \nu\geq 0 \qquad (58)$$

Constraint Qualification: For a local optimum to satisfy the KKT conditions, an additional regularity condition or constraint qualification (CQ) is required. The KKT conditions are derived from gradient information, and the CQ can be viewed as a condition on the relative influence of constraint curvature. In fact, linearly constrained problems with nonempty feasible regions require no constraint qualification. On the other hand, as seen in Figure 38, the problem Min x_1, s.t. $x_2\geq 0, (x_1)^3\geq x_2$ has a minimum point at the origin but does not satisfy the KKT conditions because it does not satisfy a CQ.

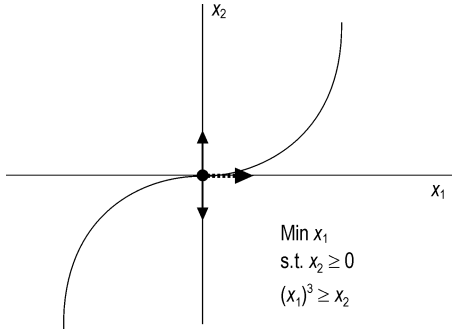

Figure 38. Failure of KKT conditions at constrained minimum (note linear dependence of constraint gradients)

CQs be defined in several ways. For instance, the linear independence constraint qualification (LICQ) requires that the active constraints at x^* be linearly independent, i.e., the matrix $[\nabla h\,(x^*)\,|\nabla g_{\mathrm{A}}(x^*)]$ is full column rank, where g_{A} is the vector of inequality constraints with elements that satisfy $g_{\mathrm{A},i}\,(x^*)=0$. With LICQ, the KKT multipliers (λ, ν) are guaranteed to be unique at the optimal solution. The weaker Mangasarian – Fromovitz constraint qualification (MFCQ) requires only that $\nabla h(x^*)$ have full column rank and that a direction p exist that satisfies $\nabla h(x^*)^T p=0$ and $\nabla g_{\mathrm{A}}(x^*)^T p>0$. With MFCQ, the KKT multipliers (λ, ν) are guaranteed to be bounded (but not necessarily unique) at the optimal solution. Additional discussion can be found in [169].

Second Order Conditions: As with unconstrained optimization, nonnegative (positive) curvature is necessary (sufficient) in all of the allowable (i.e., constrained) nonzero directions p. This condition can be stated in several ways. A typical necessary second-order condition requires a point x^* that satisfies LICQ and first-order conditions 56–58 with multipliers (λ, ν) to satisfy the additional conditions given by:

$$p^T \nabla_{xx} L(x^*,\lambda,\nu) p \geq 0 \quad \text{for all}$$

$$p \neq 0,\ \nabla h(x^*)^T p = 0,$$

$$\nabla g_A(x^*)^T p = 0 \tag{59}$$

The corresponding sufficient conditions require that the inequality in 59 be strict. Note that for the example in Figure 36, the allowable directions p span the entire space for x while in Figure 37, there are *no* allowable directions p.

Example: To illustrate the KKT conditions, consider the following unconstrained NLP:

$$\begin{aligned} &\text{Min } (x_1)^2 - 4x_1 + 3/2(x_2)^2 \\ &-7x_2 + x_1 x_2 + 9 - \ln x_1 - \ln x_2 \end{aligned} \tag{60}$$

corresponding to the contour plot in Figure 36. The optimal solution can be found by solving for the first order conditions 54:

$$\nabla f(x) = \begin{bmatrix} 2x_1 - 4 + x_2 - 1/x_1 \\ 3x_2 - 7 + x_1 - 1/x_2 \end{bmatrix} = 0$$

$$\Rightarrow x^* = \begin{bmatrix} 1.3475 \\ 2.0470 \end{bmatrix} \tag{61}$$

and $f(x^*) = -2.8742$. Checking the second-order conditions leads to:

$$\nabla_{xx} f(x*) = \begin{bmatrix} 2 + 1/(x_1^*)^2 & 1 \\ 1 & 3 + 1/(x_2^*)^2 \end{bmatrix} \Rightarrow$$

$$\nabla_{xx} f(x*) = \begin{bmatrix} 2.5507 & 1 \\ 1 & 3.2387 \end{bmatrix} \text{ (positive definite)} \tag{62}$$

Now consider the constrained NLP:

$$\begin{aligned} \text{Min} \quad & (x_1)^2 - 4x_1 + 3/2(x_1)^2 - 7x_2 + x_1 x_2 \\ & + 9 - \ln x_1 - \ln x_2 \\ \text{s.t.} \quad & 4 - x_1 x_2 \leq 0 \\ & 2x_1 - x_2 = 0 \end{aligned} \tag{63}$$

that corresponds to the plot in Figure 37. The optimal solution can be found by applying the first-order KKT conditions 56–58:

$$\nabla L(x,\lambda,\nu) = \nabla f(x) + \nabla h(x)\,\lambda + \nabla g(x)\,\nu =$$

$$\begin{bmatrix} 2x_1 - 4 + x_2 - 1/x_1 \\ 3x_2 - 7 + x_1 - 1/x_2 \end{bmatrix} + \begin{bmatrix} 2 \\ -1 \end{bmatrix} \lambda + \begin{bmatrix} -x_2 \\ -x_1 \end{bmatrix} \nu = 0$$

$$g(x) = 4 - x_1 x_2 \leq 0, \quad h(x) = 2x_1 - x_2 = 0$$

$$g(x)\,\nu = (4 - x_1 x_2)\nu, \quad \nu \geq 0$$

$$\Downarrow$$

$$x* = \begin{bmatrix} 1.4142 \\ 2.8284 \end{bmatrix}, \lambda* = 1.036, \nu* = 1.068$$

and $f(x^*) = -1.8421$. Checking the second-order conditions 59 leads to:

$$\nabla_{xx} L(x*,\lambda*,\nu*) = \nabla_{xx} [f(x*) + h(x*)\lambda* + g(x*)\nu*] =$$

$$\begin{bmatrix} 2 + 1/(x_1)^2 & 1 - \nu \\ 1 - \nu & 3 + 1/(x_2)^2 \end{bmatrix} = \begin{bmatrix} 2.5 & 0.068 \\ 0.068 & 3.125 \end{bmatrix}$$

$$[\nabla h(x*) | \nabla g_A(x*)]$$

$$p = \begin{bmatrix} 2 & -2.8284 \\ -1 & -1.4142 \end{bmatrix} \quad p = 0, p \neq 0$$

Note that LICQ is satisfied. Moreover, because $[\nabla h(x^*)\,|\nabla g_A(x^*)]$ is square and nonsingular, there are no nonzero vectors p that satisfy the allowable directions. Hence, the sufficient second-order conditions ($p^T \nabla_{xx} L(x^*,\lambda^*,\nu^*) > 0$, for all allowable p) are *vacuously satisfied* for this problem.

Convex Cases of NLP. Linear programs and quadratic programs are special cases of problem 52 that allow for more efficient solution, based on application of KKT conditions 56–59. Because these are convex problems, any locally optimal solution is a global solution. In particular, if the objective and constraint functions in problem 52 are linear then the following linear program (LP):

$$\begin{aligned} \text{Min} \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & Cx \leq d \end{aligned} \tag{65}$$

can be solved in a finite number of steps, and the optimal solution lies at a vertex of the polyhedron described by the linear constraints. This is shown in Figure 39, and in so-called primal degenerate cases, multiple vertices can be alternate optimal solutions with the same values of the objective function. The standard method to solve problem 65 is the simplex method, developed in the late 1940s [170], although, starting from KARMARKAR's discovery in 1984, interior point methods have become quite advanced and competitive for large scale problems [171]. The simplex method proceeds by moving successively from vertex to vertex with improved objective function values. Methods to solve problem 65 are well implemented and widely used,

especially in planning and logistical applications. They also form the basis for MILP methods (see below). Currently, state-of-the-art LP solvers can handle millions of variables and constraints and the application of further decomposition methods leads to the solution of problems that are two or three orders of magnitude larger than this [172, 173]. Also, the interior point method is described below from the perspective of more general NLPs.

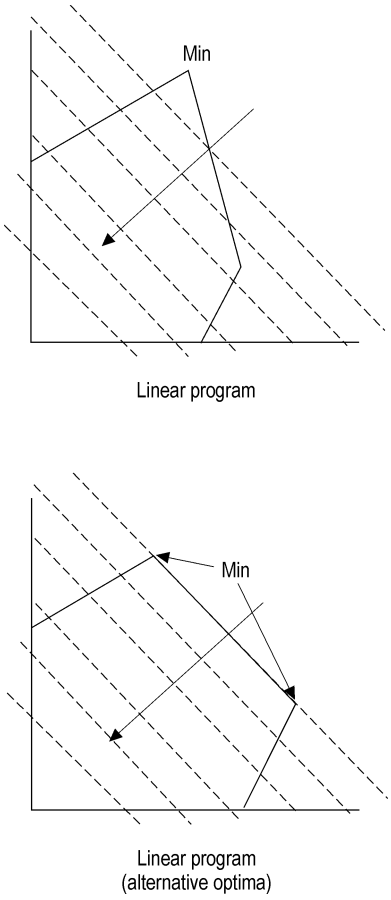

Figure 39. Contour plots of linear programs

Quadratic programs (QP) represent a slight modification of problem 65 and can be stated as:

$$\begin{array}{ll} \text{Min} & c^T x + \frac{1}{2} x^T Q x \\ \text{s.t.} & Ax = b \\ & x \leq d \end{array} \qquad (66)$$

If the matrix Q is positive semidefinite (positive definite), when projected into the null space of the active constraints, then problem 66 is (strictly) convex and the QP is a global (and unique) minimum. Otherwise, local solutions exist for problem 66, and more extensive global optimization methods are needed to obtain the global solution. Like LPs, convex QPs can be solved in a finite number of steps. However, as seen in Figure 40, these optimal solutions can lie on a vertex, on a constraint boundary, or in the interior. A number of active set strategies have been created that solve the KKT conditions of the QP and incorporate efficient updates of active constraints. Popular QP methods include null space algorithms, range space methods, and Schur complement methods. As with LPs, QP problems can also be solved with interior point methods [171].

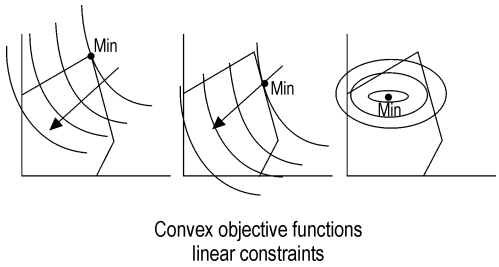

Figure 40. Contour plots of convex quadratic programs

Solving the General NLP Problem. Solution techniques for problem 52 deal with satisfaction of the KKT conditions 56–59. Many NLP solvers are based on successive quadratic programming (SQP) as it allows the construction of a number of NLP algorithms based on the Newton – Raphson method for equation solving. SQP solvers have been shown to require the fewest function evaluations to solve NLPs [174] and they can be tailored to a broad range of process engineering problems with different structure.

The SQP strategy applies the equivalent of a Newton step to the KKT conditions of the nonlinear programming problem, and this leads to a fast rate of convergence. By adding slack variables s the first-order KKT conditions can be rewritten as:

$$\nabla f(x)+\nabla h(x)\lambda+\nabla g(x)\nu=0 \tag{67a}$$

$$h(x)=0 \tag{67b}$$

$$g(x)+s=0 \tag{67c}$$

$$SVe=0 \tag{67d}$$

$$(s,\nu)\geq 0 \tag{67e}$$

where $e=[1,\ 1,\ \ldots,\ 1]^T$, $S=\mathrm{diag}(s)$ and $V=\mathrm{diag}(\nu)$. SQP methods find solutions that satisfy Equations 67a, 67b, 67c, 67d and 67e by generating Newton-like search directions at iteration k. However, equations 67d and active bounds 67e are dependent and serve to make the KKT system ill-conditioned near the solution. SQP algorithms treat these conditions in two ways. In the *active set strategy*, discrete decisions are made regarding the active constraint set, $i\in I=\{i|g_i(x^*)=0\}$, and Equation 67d is replaced by $s_i=0$, $i\in I$, and $v_i=0$, $i\notin I$. Determining the active set is a combinatorial problem, and a straightforward way to determine an estimate of the active set [and also satisfy 67e] is to formulate, at a point x^k, and solve the following QP at iteration k:

$$\begin{array}{ll}\text{Min} & \nabla f(x^k)^T p+\frac{1}{2}p^T\nabla_{xx}L(x^k,\lambda^k,\nu^k)\,p\\ \text{s.t.} & h(x^k)+\nabla h(x^k)^T p=0\\ & g(x^k)+\nabla g(x^k)^T p+s=0,\ s\geq 0\end{array} \tag{68}$$

The KKT conditions of 68 are given by:

$$\nabla f(x^k)+\nabla^2 L(x^k,\lambda^k,\nu^k)\,p+\nabla h(x^k)\lambda+\nabla g(x^k)\nu=0 \tag{69a}$$

$$h(x^k)+\nabla h(x^k)^T p=0 \tag{69b}$$

$$g(x^k)+\nabla g(x^k)^T p+s=0 \tag{69c}$$

$$SVe=0 \tag{69d}$$

$$(s,\nu)\geq 0 \tag{69e}$$

where the Hessian of the Lagrange function $\nabla_{xx}L(x,\lambda,\nu)=\nabla_{xx}\left[f(x)+h(x)^T\lambda+g(x)^T\nu\right]$ is calculated directly or through a quasi-Newtonian approximation (created by differences of gradient vectors). It is easy to show that 69a–69c correspond to a Newton–Raphson step for 67a–67c applied at iteration k. Also, selection of the active set is now handled at the QP level by satisfying the conditions 69d, 69e. To evaluate and change candidate active sets, QP algorithms apply inexpensive matrix-updating strategies to the KKT matrix associated with the QP 68. Details of this approach can be found in [175, 176].

As alternatives that avoid the combinatorial problem of selecting the active set, interior point (or barrier) methods modify the NLP problem 52 to form problem 70

$$\begin{array}{ll}\text{Min} & f(x^k)-\mu\sum_i \ln s_i\\ \text{s.t.} & h(x^k)=0\\ & g(x^k)+s=0\end{array} \tag{70}$$

where the solution to this problem has $s>0$ for the penalty parameter $\mu>0$, and decreasing μ to zero leads to solution of problem 52. The KKT conditions for this problem can be written as Equation 71

$$\begin{array}{l}\nabla f(x*)+\nabla h(x*)\lambda+\nabla g(x*)\nu=0\\ h(x*)=0\\ g(x*)+s=0\\ SVe=\mu e\end{array} \tag{71}$$

and at iteration k the Newton steps to solve 71 are given by:

$$\begin{bmatrix}\nabla_{xx}L(x_k,\lambda_k,\nu_k) & & \nabla h(x_k) & \nabla g(x_k)\\ & S_k^{-1}V_k & & I\\ \nabla h(x_k)^T & & & \\ \nabla g(x_k)^T & I & & \end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta s\\ \Delta\lambda\\ \Delta\nu\end{bmatrix}=-\begin{bmatrix}\nabla_x L(x_k,\lambda_k,\nu_k)\\ \nu_k-S_k^{-1}\mu e\\ h(x_k)\\ g(x_k)+s_k\end{bmatrix} \tag{72}$$

A detailed description of this algorithm, called IPOPT, can be found in [177].

Both active set and interior point methods have clear trade-offs. Interior point methods may require more iterations to solve problem 70 for various values of μ, while active set methods require the solution of the more expensive QP subproblem 68. Thus, if there are few inequality constraints or an active set is known (say from a good starting guess, or a known QP solution from a previous iteration) then solving problem 68 is not expensive and the active set method is favored. On the other hand, for problems with

many inequality constraints, interior point methods are often faster as they avoid the combinatorial problem of selecting the active set. This is especially the case for large-scale problems and when a large number of bounds are active. Examples that demonstrate the performance of these approaches include the solution of model predictive control (MPC) problems [178 – 180] and the solution of large optimal control problems using barrier NLP solvers. For instance, IPOPT allows the solution of problems with more than 10^6 variables and up to 50 000 degrees of freedom [181, 182].

Other Gradient-Based NLP Solvers. In addition to SQP methods, a number of NLP solvers have been developed and adapted for large-scale problems. Generally, these methods require more function evaluations than SQP methods, but they perform very well when interfaced to optimization modeling platforms, where function evaluations are cheap. All of these can be derived from the perspective of applying Newton steps to portions of the KKT conditions.

LANCELOT [183] is based on the solution of bound constrained subproblems. Here an augmented Lagrangian is formed from problem 52 and subproblem 73 is solved.

$$\begin{aligned} &\text{Min } f(x)+\lambda^T h(x) \\ &\quad +\nu\left(g(x)+s\right)+\tfrac{1}{2}\rho\|g(x),g(x)+s\|^2 \\ &\text{s.t.} \quad s\geq 0 \end{aligned} \tag{73}$$

The above subproblem can be solved very efficiently for fixed values of the multipliers λ and v and penalty parameter ρ. Here a gradient-projection, trust-region method is applied. Once subproblem 73 is solved, the multipliers and penalty parameter are updated in an outer loop and the cycle repeats until the KKT conditions for problem 52 are satisfied. LANCELOT works best when exact second derivatives are available. This promotes a fast convergence rate in solving each subproblem and allows a bound constrained trust-region method to exploit directions of negative curvature in the Hessian matrix.

Reduced gradient methods are active set strategies that rely on partitioning the variables and solving Equations 67a, 67b, 67c, 67d and 67e in a nested manner. Without loss of generality, problem 52 can be rewritten as problem 74.

$$\begin{aligned} &\text{Min } f(z) \\ &\text{s.t. } c(z)=0 \\ &a\leq z\leq b \end{aligned} \tag{74}$$

Variables are partitioned as nonbasic variables (those fixed to their bounds), basic variables (those that can be solved from the equality constraints), and superbasic variables (those remaining variables between bounds that serve to drive the optimization); this leads to $z^T=\left[z_N^T,z_S^T,z_B^T\right]$. This partition is derived from local information and may change over the course of the optimization iterations. The corresponding KKT conditions can be written as Equations 75a, 75b, 75c, 75d and 75e

$$\nabla_{\mathrm{N}} f(z)+\nabla_{\mathrm{N}} c(z)\,\gamma=\beta_{\mathrm{a}}-\beta_{\mathrm{b}} \tag{75a}$$

$$\nabla_{\mathrm{S}} f(z)+\nabla_{\mathrm{S}} c(z)\,\gamma=0 \tag{75b}$$

$$\nabla_{\mathrm{B}} f(z)+\nabla_{\mathrm{B}} c(z)\,\gamma=0 \tag{75c}$$

$$c(z)=0 \tag{75d}$$

$$z_{\mathrm{N},j}=a_j \text{ or } b_j,\ \beta_{\mathrm{a},j}\geq 0,\beta_{\mathrm{b},j}=0 \text{ or } \beta_{\mathrm{b},j}\geq 0,\beta_{\mathrm{a},j}=0 \tag{75e}$$

where λ and β are the KKT multipliers for the equality and bound constraints, respectively, and 75e replaces the complementarity conditions 58. Reduced gradient methods work by nesting equations 75b, 75d within 75a, 75c. At iteration k, for fixed values of z_{N}^k and z_{S}^k, we can solve for z_{B} using 75d and for λ using 75b. Moreover, linearization of these equations leads to constrained derivatives or *reduced gradients* (Eq. 76)

$$\frac{df}{dz_{\mathrm{s}}}=\nabla f_{\mathrm{s}}-\nabla c_{\mathrm{s}}(\nabla c_{\mathrm{B}})^{-1}\nabla f_{\mathrm{B}} \tag{76}$$

which indicate how $f(z)$ (and z_{B}) change with respect to z_{S} and z_{N}. The algorithm then proceeds by updating z_{S} using reduced gradients in a Newton-type iteration to solve equation 75c. Following this, bound multipliers β are calculated from 75a. Over the course of the iterations, if the variables z_{B} or z_{S} exceed their bounds or if some bound multipliers β become negative, then the variable partition needs to be changed and the equations 75a, 75b, 75c, 75d and 75e are reconstructed. These reduced gradient methods are embodied in the popular GRG2, CONOPT, and SOLVER codes [173].

The SOLVER code has been incorporated into Microsoft Excel. CONOPT [184] is an efficient and widely used code in several optimization modeling environments.

MINOS [185] is a well-implemented package that offers a variation on reduced gradient strategies. At iteration *k,* equation 75d is replaced by its linearization (Eq. 77)

$$c(z_{\mathrm{N}}^k, z_{\mathrm{S}}^k, z_{\mathrm{B}}^k) + \nabla_{\mathrm{B}} c\left(z^k\right)^T (z_{\mathrm{B}} - z_{\mathrm{B}}^k) + \nabla_{\mathrm{S}} c\left(z^k\right)^T (z_{\mathrm{S}} - z_{\mathrm{S}}^k) = 0 \quad (77)$$

and (75a–75c, 75e) are solved with Equation 77 as a subproblem using concepts from the reduced gradient method. At the solution of this subproblem, the constraints 75d are relinearized and the cycle repeats until the KKT conditions of 75a, 75b, 75c, 75d and 75e are satisfied. The augmented Lagrangian function 73 is used to penalize movement away from the feasible region. For problems with few degrees of freedom, the resulting approach leads to an extremely efficient method even for very large problems. MINOS has been interfaced to a number of modeling systems and enjoys widespread use. It performs especially well on large problems with few nonlinear constraints. However, on highly nonlinear problems it is usually less reliable than other reduced gradient methods.

Algorithmic Details for NLP Methods. All of the above NLP methods incorporate concepts from the Newton – Raphson Method for equation solving. Essential features of these methods are a) providing accurate derivative information to solve for the KKT conditions, b) stabilization strategies to promote convergence of the Newton-like method from poor starting points, and c) regularization of the Jacobian matrix in Newton's method (the so-called KKT matrix) if it becomes singular or ill-conditioned.

a) *Providing first and second derivatives:* The KKT conditions require first derivatives to define stationary points, so accurate first derivatives are essential to determine locally optimal solutions for differentiable NLPs. Moreover, Newton–Raphson methods that are applied to the KKT conditions, as well as the task of checking second-order KKT conditions, necessarily require information on second derivatives. (Note that second-order conditions are not checked by methods that do not use second derivatives). With the recent development of automatic differentiation tools, many modeling and simulation platforms can provide exact first and second derivatives for optimization. When second derivatives are available for the objective or constraint functions, they can be used directly in LANCELOT, SQP and, less efficiently, in reduced gradient methods. Otherwise, for problems with few superbasic variables, reduced gradient methods and reduced space variants of SQP can be applied. Referring to problem 74 with *n* variables and *m* equalities, we can write the QP step from problem 68 as Equation 78.

$$\begin{aligned} \text{Min} \quad & \nabla f\left(z^k\right)^T p + {}^1\!/_2\, p^T \nabla_{xx} L\left(z^k, \gamma^k\right) p \\ \text{s.t.} \quad & c\left(z^k\right) + \nabla c\left(z^k\right)^T p = 0 \\ & a \le z^k + p \le b \end{aligned} \quad (78)$$

Defining the search direction as $p = Z_k p_Z + Y_k p_Y$, where $\nabla c(x^k)^T Z_k = 0$ and $[Y_k | Z_k]$ is a nonsingular $n \times n$ matrix, allows us to form the following reduced QP subproblem (with $n-m$ variables)

$$\begin{aligned} \text{Min} \quad & \left[Z_k^T \nabla f\left(z^k\right) + w_k\right]^T p_Z + \tfrac{1}{2} p z^T B_k p_Z \\ \text{s.t.} \quad & a \le z^k + Z_k p_Z + Y_k p_Y \le b \end{aligned} \quad (79)$$

where $p_Y = -[\nabla c(z^k)^T Y_k]^{-1} c(z^k)$. Good choices of Y_k and Z_k, which allow sparsity to be exploited in $\nabla c(z^k)$, are: $Y_k^T = [0|I]$ and $Z_k^T = \left[I | -\nabla_{\mathrm{N,S}} c\left(z^k\right) \nabla_B c\left(z^k\right)^{-1}\right]$. Here we define the reduced Hessian $B_k = Z_z^T \nabla_{zz} L\left(z^k, \gamma^k\right) Z_k$ and $w_k = Z_k^T \nabla_{zz} L\left(z^k, \gamma^k\right) Y_k p_Y$. In the absence of second-derivative information, B_k can be approximated using positive definite quasi-Newton approximations [175]. Also, for the interior point method, a similar reduced space decomposition can be applied to the Newton step given in 72.

Finally, for problems with least-squares functions, as in data reconciliation, parameter estimation, and model predictive control, one can often assume that the values of the objective function and its gradient at the solution are vanishingly small. Under these conditions, one can show that the multipliers (λ, ν) also vanish and $\nabla_{xx} L(x^*, \lambda, \nu)$ can be substituted

by $\nabla_{xx} f(x^*)$. This *Gauss – Newton approximation* has been shown to be very efficient for the solution of least-squares problems [175].

b) *Line-search and trust-region methods* are used to promote convergence from poor starting points. These are commonly used with the search directions calculated from NLP subproblems such as problem 68. In a *trust-region approach*, the constraint, $\|p\| \leq \Delta$ is added and the iteration step is taken if there is sufficient reduction of some merit function (e.g., the objective function weighted with some measure of the constraint violations). The size of the trust region Δ is adjusted based on the agreement of the reduction of the actual merit function compared to its predicted reduction from the subproblem [183]. Such methods have strong global convergence properties and are especially appropriate for ill-conditioned NLPs. This approach has been applied in the KNITRO code [186]. *Line-search methods* can be more efficient on problems with reasonably good starting points and well-conditioned subproblems, as in real-time optimization. Typically, once a search direction is calculated from problem 68, or other related subproblem, a step size $\alpha \in (0, 1]$ is chosen so that $x^k + \alpha p$ leads to a sufficient decrease of a merit function. As a recent alternative, a novel *filter-stabilization* strategy (for both line-search and trust-region approaches) has been developed based on a bicriterion minimization, with the objective function and constraint infeasibility as competing objectives [187]. This method often leads to better performance than those based on merit functions.

c) *Regularization of the KKT matrix for the NLP subproblem* (e.g., in Equation 72) is essential for good performance of general purpose algorithms. For instance, to obtain a unique solution to Eqn. 72, active constraint gradients must be full rank, and the Hessian matrix, when projected into the null space of the active constraint gradients, must be positive definite. These properties may not hold far from the solution, and corrections to the Hessian in SQP may be necessary [176]. Regularization methods ensure that subproblems like Eqns. 68 and 72 remain well-conditioned; they include addition of positive constants to the diagonal of the Hessian matrix to ensure its positive definiteness, judicious selection of active constraint gradients to ensure that they are linearly independent, and scaling the subproblem to reduce the propagation of numerical errors. Often these strategies are heuristics built into particular NLP codes. While quite effective, most of these heuristics do not provide convergence guarantees for general NLPs.

Table 11 summarizes the characteristics of a collection of widely used NLP codes. Much more information on widely available codes can also be found on the NEOS server (www-neos.mcs.anl.gov) and the NEOS Software Guide.

Table 11. Representative NLP solvers

Method	Algorithm type	Stabilization	Second-order information
CONOPT [184]	reduced gradient	line search	exact and quasi-Newton
GRG2 [173]	reduced gradient	line search	quasi-Newton
IPOPT [177]	SQP, barrier	line search	exact
KNITRO [186]	SQP, barrier	trust region	exact and quasi-Newton
LANCELOT [183]	aug mented Lagrangian, bound constrained	trust region	exact and quasi-Newton
LOQO [188]	SQP, barrier	line search	exact
MINOS [185]	reduced gradient, augmented Lagrangian	line search	quasi-Newton
NPSOL [189]	SQP, Active set	line search	quasi-Newton
SNOPT [190]	reduced space SQP, active set	line search	quasi-Newton
SOCS [191]	SQP, active set	line search	exact
SOLVER [173]	reduced gradient	line search	quasi-Newton
SRQP [192]	reduced space SQP, active set	line search	quasi-Newton

10.3. Optimization Methods without Derivatives

A broad class of optimization strategies does not require derivative information. These methods have the advantage of easy implementation and little prior knowledge of the optimization problem. In particular, such methods are well suited for "quick and dirty" optimization studies that explore the scope of optimization for new problems, prior to investing effort for more sophisticated modeling and solution strategies. Most of these methods are derived from heuristics that naturally spawn numerous variations. As a result, a very broad literature describes these methods. Here we discuss only a few important trends in this area.

Classical Direct Search Methods. Developed in the 1960s and 1970s, these methods include one-at-a-time search and methods based on experimental designs [193]. At that time, these direct search methods were the most popular optimization methods in chemical engineering. Methods that fall into this class include the pattern search [194], the conjugate direction method [195], simplex and complex searches [196], and the adaptive random search methods [197 – 199]. All of these methods require only objective function values for unconstrained minimization. Associated with these methods are numerous studies on a wide range of process problems. Moreover, many of these methods include heuristics that prevent premature termination (e.g., directional flexibility in the complex search as well as random restarts and direction generation). To illustrate these methods, Figure 41 illustrates the performance of a pattern search method as well as a random search method on an unconstrained problem.

Simulated Annealing. This strategy is related to random search methods and derives from a class of heuristics with analogies to the motion of molecules in the cooling and solidification of metals [200]. Here a "temperature" parameter θ can be raised or lowered to influence the probability of accepting points that do not improve the objective function. The method starts with a base point x and objective value $f(x)$. The next point x' is chosen at random from a distribution. If $f(x') < f(x)$, the move is accepted with x' as the new point. Otherwise, x' is accepted with probability $p(\theta,x',x)$. Options include the Metropolis distribution, $p(\theta,x,x') = \exp[-\{f(x')-f(x)\}/\theta]$, and the Glauber distribution, $p(\theta, x, x') = \exp[-\{f(x')-f(x)\}/(1 + \exp[-\{f(x')-f(x)\}]/\theta)]$. The θ parameter is then reduced and the method continues until no further progress is made.

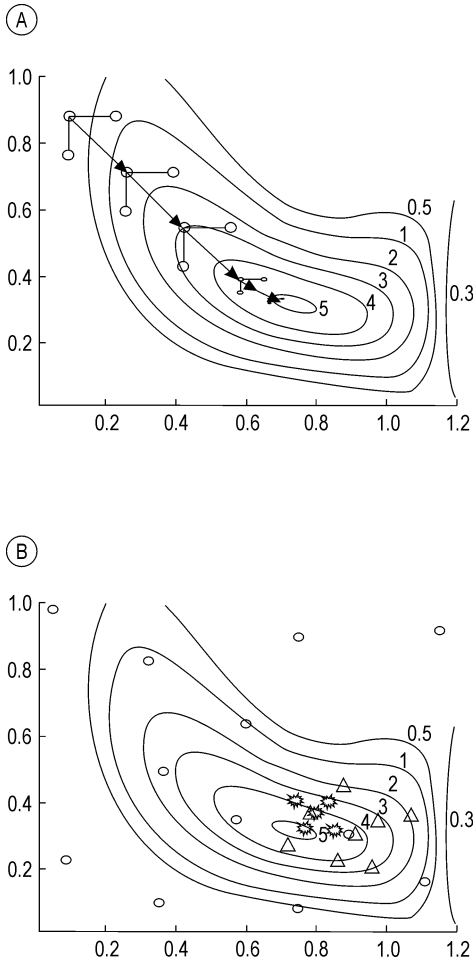

Figure 41. Examples of optimization methods without derivatives
A) Pattern search method; B) Random search method
Circles: 1st phase; Triangles: 2nd phase; Stars: 3rd phase

Genetic Algorithms. This approach, first proposed in [201], is based on the analogy of improving a population of solutions through modifying their gene pool. It also has similar performance characteristics as random search methods and simulated annealing. Two forms of ge-

netic modification, crossover or mutation, are used and the elements of the optimization vector $\boldsymbol{x}$ are represented as binary strings. Crossover deals with random swapping of vector elements (among parents with highest objective function values or other rankings of population) or any linear combinations of two parents. Mutation deals with the addition of a random variable to elements of the vector. Genetic algorithms (GAs) have seen widespread use in process engineering and a number of codes are available. A related GA algorithm is described in [173].

Derivative-Free Optimization (DFO). In the past decade, the availability of parallel computers and faster computing hardware and the need to incorporate complex simulation models within optimization studies have led a number of optimization researchers to reconsider classical direct search approaches. In particular, Dennis and Torczon [202] developed a multidimensional search algorithm that extends the simplex approach [196]. They note that the Nelder–Mead algorithm fails as the number of variables increases, even for very simple problems. To overcome this, their multidimensional pattern-search approach combines reflection, expansion, and contraction steps that act as line search algorithms for a number of linear independent search directions. This approach is easily adapted to parallel computation and the method can be tailored to the number of processors available. Moreover, this approach converges to locally optimal solutions for unconstrained problems and exhibits an unexpected performance synergy when multiple processors are used. The work of Dennis and Torczon [202] has spawned considerable research on analysis and code development for DFO methods. Moreover, Conn et al. [203] construct a multivariable DFO algorithm that uses a surrogate model for the objective function within a trust-region method. Here points are sampled to obtain a well-defined quadratic interpolation model, and descent conditions from trust-region methods enforce convergence properties. A number of trust-region methods that rely on this approach are reviewed in [203]. Moreover, a number of DFO codes have been developed that lead to black-box optimization implementations for large, complex simulation models. These include the DAKOTA package at Sandia National Lab [204], http://endo.sandia.gov/DAKOTA/software.html and FOCUS developed at Boeing Corporation [205].

Direct search methods are easy to apply to a wide variety of problem types and optimization models. Moreover, because their termination criteria are not based on gradient information and stationary points, they are more likely to favor the search for global rather than locally optimal solutions. These methods can also be adapted easily to include integer variables. However, rigorous convergence properties to globally optimal solutions have not yet been discovered. Also, these methods are best suited for unconstrained problems or for problems with simple bounds. Otherwise, they may have difficulties with constraints, as the only options open for handling constraints are equality constraint elimination and addition of penalty functions for inequality constraints. Both approaches can be unreliable and may lead to failure of the optimization algorithm. Finally, the performance of direct search methods scales poorly (and often exponentially) with the number of decision variables. While performance can be improved with the use of parallel computing, these methods are rarely applied to problems with more than a few dozen decision variables.

10.4. Global Optimization

Deterministic optimization methods are available for nonconvex nonlinear programming problems of the form problem 52 that guarantee convergence to the global optimum. More specifically, one can show under mild conditions that they converge to an ε distance to the global optimum on a finite number of steps. These methods are generally more expensive than local NLP methods, and they require the exploitation of the structure of the nonlinear program.

Global optimization of nonconvex programs has received increased attention due to their practical importance. Most of the deterministic global optimization algorithms are based on spatial branch-and-bound algorithm [206], which divides the feasible region of continuous variables and compares lower bound and upper bound for fathoming each subregion. The one that contains the optimal solution is found by eliminating subregions that are proved not to contain the optimal solution.

For nonconvex NLP problems, Quesada and Grossmann [207] proposed a spatial branch-and-bound algorithm for concave separable, linear fractional, and bilinear programs using linear and nonlinear underestimating functions [208]. For nonconvex MINLP, Ryoo and Sahinidis [209] and later Tawarmalani and Sahinidis [210] developed BARON, which branches on the continuous and discrete variables with bounds reduction method. Adjiman et al. [211, 212] proposed the SMIN-αBB and GMIN-αBB algorithms for twice-differentiable nonconvex MINLPs. Using a valid convex underestimation of general functions as well as for special functions, Adjiman et al. [213] developed the αBB method, which branches on both the continuous and discrete variables according to specific options. The branch-and-contract method [214] has bilinear, linear fractional, and concave separable functions in the continuous variables and binary variables, uses bound contraction, and applies the outer-approximation (OA) algorithm at each node of the tree. Smith and Pantelides [215] proposed a reformulation method combined with a spatial branch-and-bound algorithm for nonconvex MINLP and NLP.

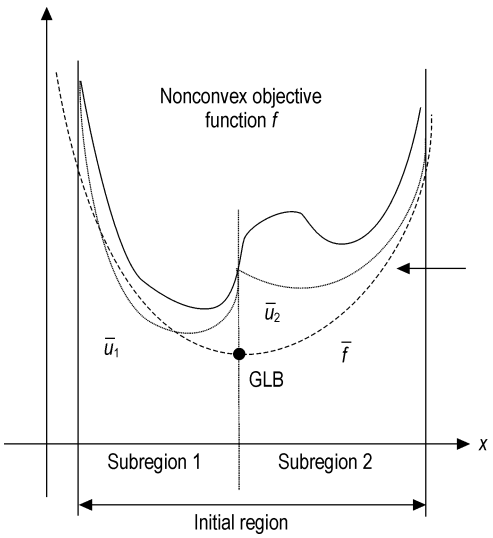

Figure 42. Convex underestimator for nonconvex function

Because in global optimization one cannot exploit optimality conditions like the KKT conditions for a local optimum, these methods work by first partitioning the problem domain (i.e., containing the feasible region) into subregions (see Fig. 42). Upper bounds on the objective function are computed over all subregions of the problem. In addition, lower bounds can be derived from convex underestimators of the objective function and constraints for each subregion. The algorithm then proceeds to eliminate all subregions that have lower bounds that are greater than the least upper bound. After this, the remaining regions are further partitioned to create new subregions and the cycle continues until the upper and lower bounds converge. Below we illustrate the specific steps of the algorithm for nonlinear programs that involve bilinear, linear fractional, and concave separable terms [207, 214].

Nonconvex NLP with Bilinear, Linear Fractional, and Concave Separable Terms. Consider the following specific nonconvex NLP problem,

$$\begin{aligned}
&\underset{x}{\text{Min}} \quad f(x) = \sum_{(i,j)\in BL_0} a_{ij}\, x_i x_j + \sum_{(i,j)\in LF_0} b_{ij} \frac{x_i}{x_j} \\
&+ \sum_{i\in C_0} g_i(x_i) + h(x) \\
&\text{subject to} \\
&f_k(x) = \sum_{(i,j)\in BL_k} a_{ijk} x_i x_j + \sum_{(i,j)\in LF_k} b_{ijk} \frac{x_i}{x_j} \\
&+ \sum_{i\in C_k} g_{i,k}(x_i) + h_k(x) \le 0 \quad k\in K \\
&x\in S\cap \Omega_0 \subset R^n
\end{aligned} \tag{80}$$

where a_{ij}, a_{ijk}, b_{ij}, b_{ijk} are scalars with $i\in I=\{1,2,\cdots,n\}$, $j\in J=\{1,2,\cdots,n\}$, and $k\in K=\{1,2,\cdots,m\}$. BL_0, BL_k, LF_0, LF_k are (i, j)-index sets, with $i\neq j$, that define the bilinear and linear fractional terms present in the problem. The functions $h(x), h_k(x)$ are convex, and twice continuously differentiable. C_0 and C_k are index sets for the univariate twice continuously differentiable concave functions $g_i(x_i), g_{i,k}(x_i)$. The set $S\subset R^n$ is convex, and $\omega_0\subset R^n$ is an n-dimensional hyperrectangle defined in terms of the initial variable bounds $x^{\mathrm{L,in}}$ and $x^{\mathrm{U,in}}$:

$$\begin{aligned}
\Omega_0 = \{x\in R^n : 0 \le x^{\mathrm{L,in}} \le x \le x^{\mathrm{U,in}},\ x_j^{\mathrm{L,in}} > 0 \\
\text{if } (i,j)\in LF_0\cup LF_k, \quad i\in I, j\in J, k\in K\}
\end{aligned}$$

The feasible region of problem 80 is denoted by D. Note that a nonlinear equality constraint of the form $f_k(x)=0$ can be accommodated in

problem 51 through the representation by the inequalities $f_k(x) \le 0$ and $-f_k(x) \le 0$, provided $h_k(x)$ is separable.

To obtain a lower bound $\mathrm{LB}(\Omega)$ for the global minimum of problem 80 over $D \cap \Omega$, where $\Omega = \{x \in R^n : x^{\mathrm{L}} \le x \le x^{\mathrm{U}}\} \subseteq \Omega_0$, the following problem is proposed:

$$\underset{(x,y,z)}{\mathrm{Min}} \quad \hat{f}(x,y,z) = \sum_{(i,j)\in BL_0} a_{ij}\, y_{ij} + \sum_{(i,j)\in LF_0} b_{ij} z_{ij} + \sum_{i\in C_0} \hat{g}_i(x_i) + h(x)$$

subject to

$$\hat{f}_k(x,y,z) = \sum_{(i,j)\in BL_k} a_{ijk} y_{ij} + \sum_{(i,j)\in LF_k} b_{ijk} z_{ij} + \sum_{i\in C_k} \hat{g}_{i,k}(x_i) + h_k(x) \le 0 \quad k \in K$$

$$(x,y,z) \in T(\Omega) \subset R^n \times R^{n_1} \times R^{n_2}$$

$$x \in S \cap \Omega \subset R^n,\ y \in R_+^{n_1},\ z \in R_+^{n_2}, \tag{81}$$

where the functions and sets are defined as follows:

a) $\hat{g}_i(x_i)$ and $\hat{g}_{i,k}(x_i)$ are the convex envelopes for the univariate functions over the domain $x_i \in [x_i^L, x_i^U]$ [216]:

$$\hat{g}_i(x_i) = g_i\left(x_i^{\mathrm{L}}\right) + \left(\frac{g_i\left(x_i^{\mathrm{U}}\right) - g_i\left(x_i^L\right)}{x_i^{\mathrm{U}} - x_i^{\mathrm{L}}}\right)\left(x_i - x_i^{\mathrm{L}}\right) \le g_i(x_i) \tag{82}$$

$$\hat{g}_{i,k}(x_i) = g_{i,k}\left(x_i^{\mathrm{L}}\right) + \left(\frac{g_{i,k}\left(x_i^{\mathrm{U}}\right) - g_{i,k}\left(x_i^L\right)}{x_i^{\mathrm{U}} - x_i^{\mathrm{L}}}\right)\left(x_i - x_i^{\mathrm{L}}\right) \le g_{i,k}(x_i) \tag{83}$$

where $\hat{g}_i(x_i) = g_i(x_i)$ at $x_i = x_i^{\mathrm{L}}$, and $x_i = x_i^{\mathrm{U}}$; likewise, $\hat{g}_{i,k}(x_i) = g_{i,k}(x_i)$ at $x_i = x_i^{\mathrm{L}}$, and $x_i = x_i^{\mathrm{U}}$.

b) $y = \{y_{ij}\}$ is a vector of additional variables for relaxing the bilinear terms in 80, and is used in the following inequalities which determine the convex and concave envelopes of bilinear terms:

$$\begin{aligned} y_{ij} &\ge x_j^{\mathrm{L}} x_i + x_i^{\mathrm{L}} x_j - x_i^{\mathrm{L}} x_j^{\mathrm{L}} && (i,j) \in BL^+ \\ y_{ij} &\ge x_j^{\mathrm{U}} x_i + x_i^{\mathrm{U}} x_j - x_i^{\mathrm{U}} x_j^{\mathrm{U}} && (i,j) \in BL^+ \end{aligned} \tag{84}$$

$$\begin{aligned} y_{ij} &\le x_j^{\mathrm{L}} x_i + x_i^{\mathrm{U}} x_j - x_i^{\mathrm{U}} x_j^{\mathrm{L}} && (i,j) \in BL^- \\ y_{ij} &\le x_j^{\mathrm{U}} x_i + x_i^{\mathrm{L}} x_j - x_i^{\mathrm{L}} x_j^{\mathrm{U}} && (i,j) \in BL^- \end{aligned} \tag{85}$$

where

$$BL^+ = \{(i,j) : (i,j) \in BL_0 \cup BL_k,\ a_{ij} > 0 \text{ or } a_{ijk} > 0,\ k \in K\}$$

$$BL^- = \{(i,j) : (i,j) \in BL_0 \cup BL_k,\ a_{ij} < 0 \text{ or } a_{ijk} < 0,\ k \in K\}$$

The inequalities 84 were first derived by McCormick [208], and along with the inequalities 85 theoretically characterized by Al-Khayyal and Falk [217, 218].

c) $z = \{z_{ij}\}$ is a vector of additional variables for relaxing the linear fractional terms in problem 80; these variables are used in the following inequalities:

$$\begin{aligned} z_{ij} &\ge \frac{x_i}{x_j^{\mathrm{L}}} + x_i^{\mathrm{U}}\left(\frac{1}{x_j} - \frac{1}{x_j^{\mathrm{L}}}\right) && (i,j) \in LF^+ \\ z_{ij} &\ge \frac{x_i}{x_j^{\mathrm{U}}} + x_i^{L}\left(\frac{1}{x_j} - \frac{1}{x_j^{\mathrm{U}}}\right) && (i,j) \in LF^+ \end{aligned} \tag{86}$$

$$z_{ij} \ge \frac{1}{x_j}\left(\frac{x_i + \sqrt{x_i^{\mathrm{L}} x_i^{\mathrm{U}}}}{\sqrt{x_i^{\mathrm{L}}} + \sqrt{x_i^{\mathrm{U}}}}\right)^2 \quad (i,j) \in LF^+ \tag{87}$$

$$\begin{aligned} z_{ij} &\le \frac{1}{x_j^{\mathrm{L}} x_j^{\mathrm{U}}}\left(x_j^{\mathrm{U}} x_i - x_i^{\mathrm{L}} x_j + x_i^{\mathrm{L}} x_j^{\mathrm{L}}\right) && (i,j) \in LF^- \\ z_{ij} &\le \frac{1}{x_j^{\mathrm{L}} x_j^{\mathrm{U}}}\left(x_j^{\mathrm{L}} x_i - x_i^{\mathrm{U}} x_j + x_i^{\mathrm{U}} x_j^{\mathrm{U}}\right) && (i,j) \in LF^- \end{aligned} \tag{88}$$

where

$$LF^+ = \{(i,j) : (i,j) \in LF_0 \cup LF_k,\ b_{ij} > 0 \text{ or } b_{ijk} > 0,\ k \in K\}$$

$$LF^- = \{(i,j) : (i,j) \in LF_0 \cup LF_k,\ b_{ij} < 0 \text{ or } b_{ijk} < 0,\ k \in K\}$$

The inequalities 86 and 87, 88 are convex underestimators due to Quesada and Grossmann [207, 219] and Zamora and Grossmann [214], respectively.

d) $T(\Omega) = \{(x,y,z) \in R^n \times R^{n_1} \times R^{n_2} :$ 82– 87 are satisfied with $x^{\mathrm{L}}, x^{\mathrm{U}}$ as in $\Omega\}$. The feasible region, and the solution of problem 52 are denoted by $M(\Omega)$, and $(\hat{x},\hat{y},\hat{z})_\Omega$, respectively. We define the *approximation gap* $\varepsilon(\Omega)$ at a branch-and-bound node as

$$\varepsilon(\Omega) = \begin{cases} \infty & \text{if OUB} = \infty \\ -\mathrm{LB}(\Omega) & \text{if OUB} = 0 \\ \frac{(\mathrm{OUB} - \mathrm{LB}(\Omega))}{|\mathrm{OUB}|} & \text{otherwise} \end{cases} \tag{89}$$

where the *overall upper bound* (OUB) is the value of $f(x)$ at the best available feasible point $x \in D$; if no feasible point is available, then OUB$=\infty$.

Note that the underestimating problem 81 is a linear program if $LF^{+}=\varnothing$. During the execution of the spatial branch-and-bound algorithm, problem (NBNC) is solved initially over $M(\Omega_0)$ (root node of the branch-and-bound tree). If a better approximation is required, $M(\Omega_0)$ is refined by partitioning Ω_0 into two smaller hyperrectangles Ω_{01} and Ω_{02}, and two children nodes are created with relaxed feasible regions given by $M(\Omega_{01})$ and $M(\Omega_{02})$. Problem 81 might be regarded as a basic underestimating program for the general problem 80. In some cases, however, it is possible to develop additional convex estimators that might strengthen the underestimating problem. See, for instance, the projections proposed by Quesada and Grossmann [207], the reformulation–linearization technique by Sherali and Alameddine [220], and the reformulation–convexification approach by Sherali and Tuncbilek [221].

The Set of Branching Variables. A set of branching variables, characterized by the index set $\mathrm{BV}(\Omega)$ defined below, is determined by considering the optimal solution $(\hat{x},\hat{y},\hat{z})_{\Omega}$ of the underestimating problem:

$$\begin{aligned}&\mathrm{BV}(\Omega)=\{i,j:|\hat{y}_{ij}-\hat{x}_i\hat{x}_j|=\zeta_l \text{ or } |\hat{z}_{ij}-\hat{x}_i/\hat{x}_j|=\zeta_l\\&\text{or } g_i(\hat{x}_i)-\hat{g}_i(\hat{x}_i)=\zeta_l \text{ or } g_{i,k}(\hat{x}_i)-\hat{g}_{i,k}(\hat{x}_i)=\zeta_l,\\&\text{for } i\in I, j\in J, k\in K, l\in L\}\end{aligned} \tag{90}$$

where, for a prespecified number l_n, $L=\{1,2,\ldots,l_n\}$ and ζ_1 is the magnitude of the largest approximation error for a nonconvex term in problem 80 evaluated at $(\hat{x},\hat{y},\hat{z})_{\Omega}$:

$$\begin{aligned}&\xi_1=\mathrm{Max}\ [|\hat{y}_{ij}-\hat{x}_i\hat{x}_j|,\ |\hat{z}_{ij}-\hat{x}_i/\hat{x}_j|,\ g_i(\hat{x}_i)-\hat{g}_i(\hat{x}_i),\\&g_{i,k}(\hat{x}_i)-\hat{g}_{i,k}(\hat{x}_i)]\\&i\in I, j\in J, k\in K\end{aligned}$$

Similarly, we define $\xi_l<\xi_{l-1}$ with $l\in L\setminus\{1\}$ as the l-th largest magnitude for an approximation error; for instance, $\xi_2<\xi_1$ is the second largest magnitude for an approximation error. Note that in some cases it might be convenient to introduce weights in the determination of $\mathrm{BV}(\Omega)$ in order to scale differences in the approximation errors or to induce preferential branching schemes. This might be particularly useful in applications where specific information can be exploited by imposing an order of precedence on the set of complicating variables.

This basic concept in spatial branch-and-bound for global optimization is as follows. *Bounds* are related to the calculation of upper and lower bounds. For the former, any feasible point or, preferably, a locally optimal point in the subregion, can be used. For the lower bound, convex relaxations of the objective and constraint function are derived such as in problem 81. The *refining* step deals with the construction of partitions in the domain and further partitioning them during the search process. Finally, the *selection* step decides on the order of exploring the open subregions. Thus, the feasible region and the objective function are replaced by convex envelopes to form relaxed problems. Solving these convex relaxed problems leads to global solutions that are lower bounds to the NLP in the particular subregion. Finally, we see that gradient-based NLP solvers play an important role in global optimization algorithms, as they often yield the lower and upper bounds for the subregions. The following *spatial branch-and-bound* global optimization algorithm can therefore be given by the following steps:

0. Initialize algorithm: calculate upper bound by obtaining a local solution to problem 80. Calculate a lower bound solving problem 81 over the entire (relaxed) feasible region Ω_0.

For iteration k with a set of partitions $\Omega_{k,j}$ and bounds in each subregion OLB_j and OUB_j:

1) *Bound:* Define best upper bound: $\mathrm{OUB}=\mathrm{Min}_j\ \mathrm{OUB}_j$ and delete (fathom) all subregions j with lower bounds $\mathrm{OLB}_j\geq\mathrm{OUB}$. If $\mathrm{OLB}_j\geq\mathrm{OUB}-\varepsilon$, stop.
2) *Refine:* Divide the remaining active subregions into partitiions $\Omega_{k,j1}$ and $\Omega_{k,j2}$. (Several branching rules are available for this step.)
3) *Select:* Solve the convex NLP 81 in the new partitions to obtain OLB_{j1} and OLB_{j2}. Delete partition if no feasible solution.
4) *Update:* Obtain upper bounds, OUB_{j1} and OUB_{j2} to new partitions if present. Set $k=k+1$, update partition sets and go to step 1.

Example: To illustrate the spatial branch-and-bound algorithm, consider the global solution of:

$$\begin{aligned}&\mathrm{Min}\ f(x)=5/2\,x^4-20\,x^3+55\,x_2-57\,x\\&\text{s.t. } 0.5\leq x\leq 2.5\end{aligned} \tag{91}$$

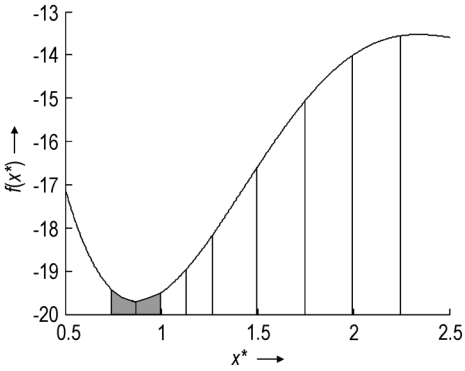

Figure 43. Global optimization example with partitions

As seen in Figure 43, this problem has local solutions at $x^* = 2.5$ and at $x^* = 0.8749$. The latter is also the global solution with $f(x^*) = -19.7$. To find the global solution we note that all but the $-20\,x^3$ term in problem 91 are convex, so we replace this term by a new variable and a linear underestimator within a particular subregion, i.e.:

$$\begin{aligned} \text{Min} \quad & f_L\,(x) = 5/2\,x^4 - 20\,w + 55\,x_2 - 57\,x \\ \text{s.t.} \quad & x_1 \le x \le x_u \\ & w = (x_l)^3\,\tfrac{(x_u - x)}{(x_u - x_l)} + (x_u)^3\,\tfrac{(x - x_l)}{(x_u - x_l)} \end{aligned} \tag{92}$$

In Figure 43 we also propose subregions that are created by simple bisection partitioning rules, and we use a "loose" bounding tolerance of $\varepsilon = 0.2$. In each partition the lower bound, f_{L} is determined by problem 92 and the upper bound f_{U} is determined by the local solution of the original problem in the subregion. Figure 44 shows the progress of the spatial branch-and-bound algorithm as the partitions are refined and the bounds are updated. In Figure 43, note the definitions of the partitions for the nodes, and the sequence numbers in each node that show the order in which the partitions are processed. The grayed partitions correspond to the deleted subregions and at termination of the algorithm we see that $f_{\mathrm{L}j} \ge f_{\mathrm{U}} - \varepsilon$ (i.e., $-19.85 \ge -19.7 - 0.2$), with the gray subregions in Figure 43 still active. Further partitioning in these subregions will allow the lower and upper bounds to converge to a tighter tolerance.

A number of improvements can be made to the bounding, refinement, and selection strategies in the algorithm that accelerate the convergence of this method. A comprehensive discussion of all of these options can be found in [222 – 224]. Also, a number of efficient global optimization codes have recently been developed, including αBB, BARON, LGO, and OQNLP. An interesting numerical comparison of these and other codes can be found in [225].

10.5. Mixed Integer Programming

Mixed integer programming deals with both discrete and continuous decision variables. For simplicity in the presentation we consider the most common case where the discrete decisions are binary variables, i.e., $y_i = 0$ or 1, and we consider the mixed integer problem 51. Unlike local optimization methods, there are no optimality conditions, like the KKT conditions, that can be applied directly.

Mixed Integer Linear Programming. If the objective and constraint functions are all linear in problem 51, and we assume 0–1 binary variables for the discrete variables, then this gives rise to a mixed integer linear programming (MILP) problem given by Equation 93.

$$\begin{aligned} \text{Min} \quad & Z = a^T x + c^T y \\ \text{s.t.} \quad & Ax + By \le b \\ & x \ge 0,\ y \in \{0,1\}^t \end{aligned} \tag{93}$$

As is well known, the (MILP) problem is NP-hard. Nevertheless, an interesting theoretical result is that it is possible to transform it into an LP with the convexification procedures proposed by Lovacz and Schrijver [226], Sherali and Adams [227], and Balas et al. [228]. These procedures consist of sequentially lifting the original relaxed $x - y$ space into higher dimension and projecting it back to the original space so as to yield, after a finite number of steps, the integer convex hull. Since the transformations have exponential complexity, they are only of theoretical interest, although they can be used as a basis for deriving cutting planes (e.g. lift and project method by [228]).

As for the solution of problem (MILP), it should be noted that this problem becomes an LP problem when the binary variables are relaxed as continuous variables $0 \le y \le 1$. The most common solution algorithms for problem

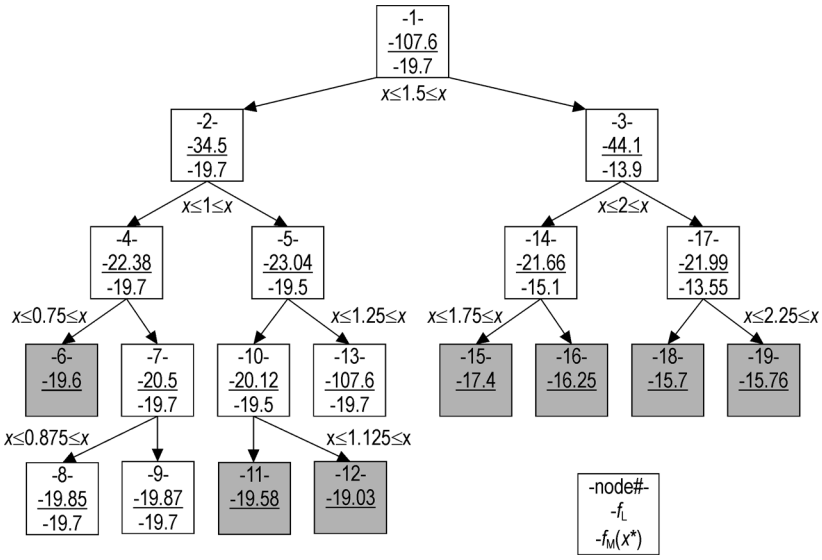

Figure 44. Spatial branch-and-bound sequence for global optimization example

(MILP) are LP-based branch-and-bound methods, which are enumeration methods that solve LP subproblems at each node of the search tree. This technique was initially conceived by Land and Doig [229], Balas [230], and later formalized by Dakin, [231]. Cutting-plane techniques, which were initially proposed by Gomory [232], and consist of successively generating valid inequalities that are added to the relaxed LP, have received renewed interest through the works of Crowder et al. [233], van Roy and Wolsey [234], and especially the lift-and-project method of Balas et al. [228]. A recent review of branch-and-cut methods can be found in [235]. Finally, Benders decomposition [236] is another technique for solving MILPs in which the problem is successively decomposed into LP subproblems for fixed 0–1 and a master problem for updating the binary variables.

LP-Based Branch and Bound Method. We briefly outline in this section the basic ideas behind the branch-and-bound method for solving MILP problems. Note that if we relax the t binary variables by the inequalities $0 \leq y \leq 1$ then 93 becomes a linear program with a (global) solution that is a lower bound to the MILP 93. There are specific MILP classes in which the LP relaxation of 93 has the same solution as the MILP. Among these problems is the well-known *assignment problem*. Other MILPs that can be solved with efficient special-purpose methods are the *knapsack* problem, the *set-covering* and *set-partitioning* problems, and the *traveling salesman* problem. See [237] for a detailed treatment of these problems.

The branch-and-bound algorithm for solving MILP problems [231] is similar to the spatial branch-and-bound method of the previous section that explores the search space. As seen in Figure 45, binary variables are successively fixed to define the search tree and a number of bounding properties are exploited in order to fathom nodes in to avoid exhaustive enumeration of all the nodes in the tree.

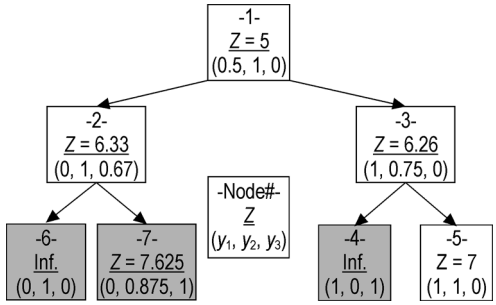

Figure 45. Branch and bound sequence for MILP example

The basic idea in the search is as follows. The top, or root node, in the tree is the solution to the linear programming relaxation of 93. If all the

y variables take on 0–1 values, the MILP problem is solved, and no further search is required. If at least one of the binary variables yields a fractional value, the solution of the LP relaxation yields a lower bound to problem 93. The search then consists of branching on that node by fixing a particular binary variable to 0 or 1, and the corresponding restricted LP relaxations are solved that in turn yield a lower bound for any of their descendant nodes. In particular, the following properties are exploited in the branch-and-bound search:

- Any node (initial, intermediate, leaf node) that leads to feasible LP solution corresponds to a valid upper bound to the solution of the MILP problem 93.
- Any intermediate node with an infeasible LP solution has infeasible leaf nodes and can be fathomed (i.e., all remaining children of this node can be eliminated).
- If the LP solution at an intermediate node is not less than an existing integer solution, then the node can be fathomed.

These properties lead to pruning of the nodes in the search tree. Branching then continues in the tree until the upper and lower bounds converge.

The basic concepts outlined above lead to a branch-and-bound algorithm with the following features. LP solutions at intermediate nodes are relatively easy to calculate since they can be effectively updated with the dual simplex method. The selection of binary variables for branching, known as the *branching rule,* is based on a number of different possible criteria; for instance, choosing the fractional variable closest to 0.5, or the one involving the largest of the smallest pseudocosts for each fractional variable. *Branching strategies* to navigate the tree take a number of forms. More common *depth-first strategies* expand the most recent node to a leaf node or infeasible node and then backtrack to other branches in the tree. These strategies are simple to program and require little storage of past nodes. On the other hand, *breadth-first strategies* expand all the nodes at each level of the tree, select the node with the lowest objective function, and then proceed until the leaf nodes are reached. Here, more storage is required, but generally fewer nodes are evaluated than in depth-first search. In practice, a combination of both strategies is commonly used: branch on the dichotomy 0–1 at each node (i.e., like breadth-first), but expand as in depth-first. Additional description of these strategies can be found in [237].

Example: To illustrate the branch-and-bound approach, we consider the MILP:

$$\begin{aligned} \text{Min} \quad & Z = x + y_1 + 2y_2 + 3y_3 \\ \text{s.t.} \quad & -x + 3y_1 + y_2 + 2y_3 \le 0 \\ & -4y_1 - 8y_2 - 3y_3 \le -10 \\ & x \ge 0, y_1, y_2, y_3 = \{0,1\} \end{aligned} \tag{94}$$

The solution to problem 94 is given by $x = 4$, $y_1 = 1$, $y_2 = 1$, $y_3 = 0$, and $Z = 7$. Here we use a depth-first strategy and branch on the variables closest to zero or one. Figure 45 shows the progress of the branch-and-bound algorithm as the binary variables are selected and the bounds are updated. The sequence numbers for each node in Figure 45 show the order in which they are processed. The grayed partitions correspond to the deleted nodes and at termination of the algorithm we see that $Z = 7$ and an integer solution is obtained at an intermediate node where coincidentally $y_3 = 0$.

Mixed integer linear programming (MILP) methods and codes have been available and applied to many practical problems for more than twenty years (e.g., [237]. The LP-based branch-and-bound method [231] has been implemented in powerful codes such as OSL, CPLEX, and XPRESS. Recent trends in MILP include the development of branch-and-price [238] and branch-and-cut methods such as the lift-and-project method [228] in which cutting planes are generated as part of the branch-and-bound enumeration. See also [235] for a recent review on MILP. A description of several MILP solvers can also be found in the NEOS Software Guide.

Mixed Integer Nonlinear Programming The most basic form of an MINLP problem when represented in algebraic form is as follows:

$$\begin{aligned} & \min z = f(x,y) \\ & s.t.\ g_j(x,y) \le 0 \; j \in J \qquad \text{(P1)} \\ & x \in X, \; y \in Y \end{aligned} \tag{95}$$

where $f(\cdot)$, $g(\cdot)$ are *convex, differentiable* functions, J is the index set of inequalities, and x and y are the continuous and discrete

variables, respectively. The set X is commonly assumed to be a convex compact set, e.g., $X=\{x|x\in\mathbf{R}^n, Dx\leq d, x^L\leq x\leq x^U\}$; the discrete set Y corresponds to a polyhedral set of integer points, $Y=\{y|x\in Z^m, Ay\leq a\}$, which in most applications is restricted to 0–1 values, $y\in\{0,1\}^m$. In most applications of interest the objective and constraint functions $f(\cdot)$, $g(\cdot)$ are linear in y (e.g., fixed cost charges and mixed-logic constraints): $f(x,y)=c^T y+r(x)$, $g(x,y)=By+h(x)$.

Methods that have addressed the solution of problem 95 include the branch-and-bound method (BB) [239 – 243], generalized Benders decomposition (GBD) [244], outer-approximation (OA) [245 – 247], LP/NLP based branch-and-bound [248], and extended cutting plane method (ECP) [249].

There are three basic *NLP subproblems* that can be considered for problem 95:

a) NLP relaxation

$$\begin{aligned} &\text{Min } Z_{LB}^k = f(x,y)\\ &s.t.\ g_j(x.y)\leq 0\ j\in J\\ &\quad x\in X, y\in Y_R \qquad \text{(NLP1)}\\ &\quad y_i\leq\alpha_i^k\ i\in I_{FL}^k\\ &\quad y_i\geq\beta_i^k\ i\in I_{FU}^k \end{aligned} \tag{96}$$

where Y_{R} is the continuous relaxation of the set Y, and $I_{\mathrm{FL}}^k, I_{\mathrm{FU}}^k$ are index subsets of the integer variables y_i, $i\in I$ which are restricted to lower and upper bounds α_i^k, β_i^k at the k-th step of a branch-and-bound enumeration procedure. Note that $\alpha_i^k=\lfloor y_i^l\rfloor, \beta_i^k=\lceil y_i^m\rceil$, $l<k$, $m<k$, where y_i^l, y_i^m are noninteger values at a previous step, and $\lfloor\cdot\rfloor$, $\lceil\cdot\rceil$ are the floor and ceiling functions, respectively.

Also note that if $I_{\mathrm{FU}}^k=I_{\mathrm{FL}}^k=\varnothing\ (k=0)$, problem 96 corresponds to the continuous NLP relaxation of problem 95. Except for few and special cases, the solution to this problem yields in general a noninteger vector for the discrete variables. Problem 96 also corresponds to the k-th step in a branch-and-bound search. The optimal objective function $Z_{\mathrm{LB}}^{\mathrm{o}}$ provides an absolute lower bound to problem 95; for $m\geq k$, the bound is only valid for $I_{\mathrm{FL}}^k\subset I_{\mathrm{FL}}^m$, $I_{\mathrm{FU}}^k\subset I_{\mathrm{FL}}^m$.

b) NLP subproblem for fixed y^k:

$$\begin{aligned} &\text{Min}\quad Z_{\mathrm{U}}^k = f(x,y^k)\\ &\text{s.t.}\quad g_j(x,y^k)\leq 0\quad j\in J\\ &\qquad x\in X \end{aligned} \tag{97}$$

which yields an upper bound Z_{U}^k to problem 95 provided problem 97 has a feasible solution. When this is not the case, we consider the next subproblem:

c) Feasibility subproblem for fixed y^{k}:

$$\begin{aligned} &\text{Min } u\\ &\text{s.t. } g_j(x,y^k)\leq u\ j\in J\\ &\quad x\in X, u\in R^1 \end{aligned} \tag{98}$$

which can be interpreted as the minimization of the infinity-norm measure of infeasibility of the corresponding NLP subproblem. Note that for an infeasible subproblem the solution of problem 98 yields a strictly positive value of the scalar variable u.

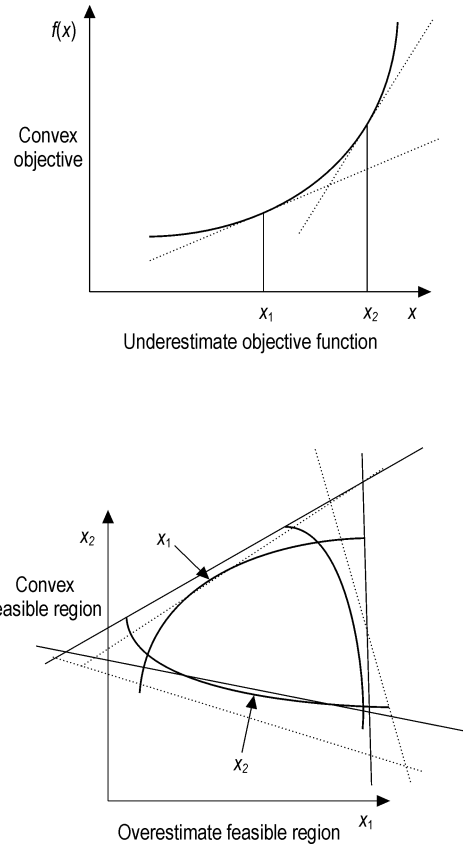

Figure 46. Geometrical interpretation of linearizations in master problem 99

The convexity of the nonlinear functions is exploited by replacing them with supporting hyperplanes, which are generally, but not necessarily, derived at the solution of the NLP subproblems. In particular, the new values y^K (or (x^K, y^K) are obtained from a *cutting-plane MILP problem* that is based on the K points (x^k, y^k), $k = 1 \ldots K$, generated at the K previous steps:

$$\begin{aligned} &\text{Min } Z_L^K = \alpha \\ &\text{st } \left.\begin{aligned} &\alpha \geq f(x^k,y^k) \\ &\quad + \nabla f(x^k,y^k)^T \begin{bmatrix} x-x^k \\ y-y^k \end{bmatrix} \\ &g_j(x^k,y^k) \\ &\quad + \nabla g_j(x^k,y^k)^T \begin{bmatrix} x-x^k \\ y-y^k \end{bmatrix} \leq 0 \; j \in J^k \end{aligned} \right\} k=1,\ldots K \\ &\qquad x \in X, \; y \in Y \end{aligned} \tag{99}$$

where $J^k \subseteq J$. When only a subset of linearizations is included, these commonly correspond to violated constraints in problem 95. Alternatively, it is possible to include all linearizations in problem 99. The solution of 99 yields a valid lower bound Z_L^k to problem 95. This bound is nondecreasing with the number of linearization points K. Note that since the functions $f(x,y)$ and $g(x,y)$ are convex, the linearizations in problem 99 correspond to outer approximations of the nonlinear feasible region in problem 95. A geometrical interpretation is shown in Figure 46, where it can be seen that the convex objective function is being underestimated, and the convex feasible region overestimated with these linearizations.

Algorithms. The different methods can be classified according to their use of the subproblems 96–98, and the specific specialization of the MILP problem 99, as seen in Figure 47. In the GBD and OA methods (case b), as well in the LP/NLP-based branch-and-bound mehod (case d), problem 98 is solved if infeasible subproblems are found. Each of the methods is explained next in terms of the basic subproblems.

Branch and Bound. While the earlier work in branch and bound (BB) was aimed at linear problems [231] this method can also be applied to nonlinear problems [239 – 243]. The BB method starts by solving first the continuous NLP relaxation. If all discrete variables take integer values the search is stopped. Otherwise, a tree search is performed in the space of the integer variables y_i, $i \in I$. These are successively fixed at the corresponding nodes of the tree, giving rise to relaxed NLP subproblems of the form (NLP1) which yield lower bounds for the subproblems in the descendant nodes. Fathoming of nodes occurs when the lower bound exceeds the current upper bound, when the subproblem is infeasible, or when all integer variables y_i take on discrete values. The last-named condition yields an upper bound to the original problem.

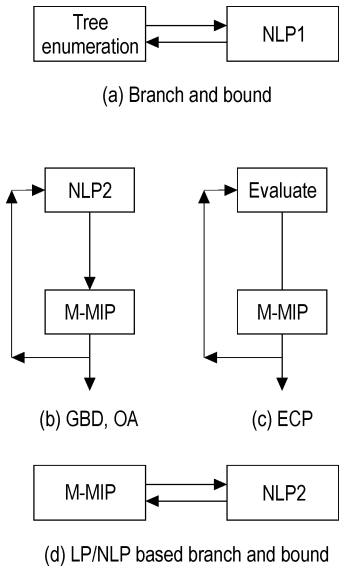

Figure 47. Major steps in the different algorithms

The BB method is generally only attractive if the NLP subproblems are relatively inexpensive to solve, or when only few of them need to be solved. This could be either because of the low dimensionality of the discrete variables, or because the integrality gap of the continuous NLP relaxation of 95 is small.

Outer Approximation [245 – 247]**.** The OA method arises when NLP subproblems 97 and MILP master problems 99 with $J^k = J$ are solved successively in a cycle of iterations to generate the points (x^k, y^k). Since the master problem 99 requires the solution of all feasible discrete variables y^k, the following MILP

relaxation is considered, assuming that the solution of K different NLP subproblems ($K = |\mathrm{KFS} \cup \mathrm{KIS}|$), where KFS is a set of solutions from problem 97, and KIS set of solutions from problem 98 is available:

$$\begin{aligned} &\text{Min } Z_{\mathrm{L}}^{K}=\alpha \\ \text{st}\quad &\left.\begin{aligned} &\alpha \geq f(x^k,y^k) \\ &+\nabla f(x^k,y^k)^T \begin{bmatrix} x-x^k \\ y-y^k \end{bmatrix} \\ &g_j(x^k,y^k) \\ &+\nabla g_j(x^k,y^k)^T \begin{bmatrix} x-x^k \\ y-y^k \end{bmatrix} \leq 0 \quad j\in J \end{aligned}\right\} k=1,\ldots K \\ &x\in X,\, y\in Y \end{aligned} \tag{100}$$

Given the assumption on convexity of the functions $f(x,y)$ and $g(x,y)$, it can be proved that the solution of problem 100 Z_{L}^{k} corresponds to a lower bound of the solution of problem 95. Note that this property can be verified in Figure 46. Also, since function linearizations are accumulated as iterations proceed, the master problems 100 yield a nondecreasing sequence of lower bounds $Z_{\mathrm{L}}^{1}\ldots\leq Z_{\mathrm{L}}^{k}\ldots Z_{\mathrm{L}}^{k}$ since linearizations are accumulated as iterations k proceed.

The OA algorithm as proposed by Duran and Grossmann consists of performing a cycle of major iterations, $k = 1,..K$, in which problem 97 is solved for the corresponding y^K, and the relaxed MILP master problem 100 is updated and solved with the corresponding function linearizations at the point (x^k,y^k) for which the corresponding subproblem NLP2 is solved. If feasible, the solution to that problem is used to construct the first MILP master problem; otherwise a feasibility problem 98 is solved to generate the corresponding continuous point [247]. The initial MILP master problem 100 then generates a new vector of discrete variables. The subproblems 97 yield an upper bound that is used to define the best current solution, $\mathrm{UB}^k=\min\limits_{k}\left\{Z_{\mathrm{U}}^{k}\right\}$. The cycle of iterations is continued until this upper bound and the lower bound of the relaxed master problem Z_{L}^{k} are within a specified tolerance. One way to avoid solving the feasibility problem 98 in the OA algorithm when the discrete variables in problem 95 are 0–1, is to introduce the following integer cut whose objective is to make infeasible the choice of the previous 0–1 values generated at the K previous iterations [245]:

$$\sum_{i\in B^k} y_i - \sum_{i\in N^k} y_i \leq \left|B^k\right| - 1 \quad k=1,\ldots K \tag{101}$$

where $B^k=\left\{i|y_i^k=1\right\}$, $N^k=\left\{i|y_i^k=0\right\}$, $k=1,\ldots K$. This cut becomes very weak as the dimensionality of the 0–1 variables increases. However, it has the useful feature of ensuring that new 0–1 values are generated at each major iteration. In this way the algorithm will not return to a previous integer point when convergence is achieved. Using the above integer cut the termination takes place as soon as $Z_{\mathrm{L}}^{K}\geq \mathrm{UB}^K$.

The OA algorithm trivially converges in one iteration if $f(x,y)$ and $g(x,y)$ are linear. This property simply follows from the fact that if $f(x,y)$ and $g(x,y)$ are linear in x and y the MILP master problem 100 is identical to the original problem 95. It is also important to note that the MILP master problem need not be solved to optimality.

Generalized Benders Decomposition (GBD) [244]**.** The GBD method [250] is similar to the outer-approximation method. The difference arises in the definition of the MILP master problem 99. In the GBD method only active inequalities are considered $J^k=\{j|\, g_j(x^k,y^k)=0\}$ and the set $x\in X$ is disregarded. In particular, consider an outer-approximation given at a given point (x^k, y^k)

$$\begin{aligned} &\alpha \geq f\left(x^k,y^k\right)+\nabla f\left(x^k,y^k\right)^T \begin{bmatrix} x-x^k \\ y-y^k \end{bmatrix} \\ &g\left(x^k,y^k\right)+\nabla g\left(x^k,y^k\right)^T \begin{bmatrix} x-x^k \\ y-y^k \end{bmatrix} \leq 0 \end{aligned} \tag{102}$$

where for a fixed y^k the point x^k corresponds to the optimal solution to problem 97. Making use of the Karush – Kuhn – Tucker conditions and eliminating the continuous variables x, the inequalities in 102 can be reduced as follows [248]:

$$\begin{aligned} &\alpha \geq f\left(x^k,y^k\right)+\nabla_y f\left(x^k,y^k\right)^T\left(y-y^k\right)+ \\ &\left(\mu^k\right)^T\left[g\left(x^k,y^k\right)+\nabla_y g\left(x^k,y^k\right)^T\left(y-y^k\right)\right] \end{aligned} \tag{103}$$

which is the Lagrangian cut projected in the y-space. This can be interpreted as a surrogate constraint of the equations in 102, because it is obtained as a linear combination of these.

For the case when there is no feasible solution to problem 97, then if the point x^k is obtained from the feasibility subproblem (NLPF), the following feasibility cut projected in y can be obtained using a similar procedure.

$$\left(\lambda^k\right)^T \left[g\left(x^k,y^k\right)+\nabla_y g\left(x^k,y^k\right)^T\left(y-y^k\right)\right]\leq 0 \tag{104}$$

In this way, problem 99 reduces to a problem projected in the y-space:

$$\begin{aligned} &\text{Min} \quad Z_L^K=\alpha \\ &\text{st}\,\alpha \geq \quad f\left(x^k,y^k\right)+\nabla_y f\left(x^k,y^k\right)^T\left(y-y^k\right)+ \\ &\left(\mu^k\right)^T\left[g\left(x^k,y^k\right)+\nabla_y g\left(x^k,y^k\right)^T\left(y-y^k\right)\right] \\ &\qquad k\in KFS \\ &\left(\lambda^k\right)^T\left[g\left(x^k,y^k\right)+\nabla_y g\left(x^k,y^k\right)^T\left(y-y^k\right)\right]\leq 0 \\ &\qquad k\in KIS \\ &x\in X,\ \alpha\in R^1 \end{aligned} \tag{105}$$

where KFS is the set of feasible subproblems 97, and KIS the set of infeasible subproblems whose solution is given by problem 98. Also $|\text{KFS}\cup\text{KIS}| = K$. Since master problem 105 can be derived from master problem 100, in the context of problem 95, GBD can be regarded as a particular case of the outer-approximation algorithm. In fact one can prove that given the same set of K subproblems, the lower bound predicted by the relaxed master problem 100 is greater than or equal to that predicted by the relaxed master problem 105 [245]. This proof follows from the fact that the Lagrangian and feasibility cuts 103 and 104 are surrogates of the outer-approximations 102. Given the fact that the lower bounds of GBD are generally weaker, this method commonly requires a larger number of cycles or major iterations. As the number of 0–1 variables increases this difference becomes more pronounced. This is to be expected since only one new cut is generated per iteration. Therefore, user-supplied constraints must often be added to the master problem to strengthen the bounds. Also, it is sometimes possible to generate multiple cuts from the solution of an NLP subproblem in order to strengthen the lower bound [251]. As for the OA algorithm, the trade-off is that while it generally predicts stronger lower bounds than GBD, the computational cost for solving the master problem (M-OA) is greater, since the number of constraints added per iteration is equal to the number of nonlinear constraints plus the nonlinear objective.

If problem 95 has zero integrality gap, the GBD algorithm converges in one iteration once the optimal (x^*, y^*) is found [252]. This property implies that the only case in which one can expect the GBD method to terminate in one iteration is that in which the initial discrete vector is the optimum, and when the objective value of the NLP relaxation of problem 95 is the same as the objective of the optimal mixed-integer solution.

Extended Cutting Plane (ECP) [249]**.** The ECP method, which is an extension of Kelley's cutting-plane algorithm for convex NLP [253], does not rely on the use of NLP subproblems and algorithms. It relies only on the iterative solution of problem 99 by successively adding a linearization of the most violated constraint at the predicted point (x^k, y^k):

$$J^k=\left\{\hat{j}\in\arg\left\{\max_{j\in J} g_j\left(x^k,y^k\right)\right\}\right\}$$

Convergence is achieved when the maximum constraint violation lies within the specified tolerance. The optimal objective value of problem 99 yields a nondecreasing sequence of lower bounds. It is of course also possible to either add to problem 99 linearizatons of all the violated constraints in the set J^k, or linearizations of all the nonlinear constraints $j \in J$. In the ECP method the objective must be defined as a linear function, which can easily be accomplished by introducing a new variable to transfer nonlinearities in the objective as an inequality.

Note that since the discrete and continuous variables are converged simultaneously, the ECP method may require a large number of iterations. However, this method shares with the OA method Property 2 for the limiting case when all the functions are linear.

LP/NLP-Based Branch and Bound [248]**.** This method is similar in spirit to a branch-and-cut method, and avoids the complete solution of the MILP master problem (M-OA) at each major iteration. The method starts by solving an initial NLP subproblem, which is linearized as in (M-OA). The basic idea consists then of performing an LP-based branch-and-bound method for (M-OA) in which NLP subproblems 97 are solved at those nodes in which feasible integer solutions

are found. By updating the representation of the master problem in the current open nodes of the tree with the addition of the corresponding linearizations, the need to restart the tree search is avoided.

This method can also be applied to the GBD and ECP methods. The LP/NLP method commonly reduces quite significantly the number of nodes to be enumerated. The trade-off, however, is that the number of NLP subproblems may increase. Computational experience has indicated that often the number of NLP subproblems remains unchanged. Therefore, this method is better suited for problems in which the bottleneck corresponds to the solution of the MILP master problem. Leyffer [254] has reported substantial savings with this method.

Example: Consider the following MINLP problem, whose objective function and constraints contain nonlinear convex terms

$$\begin{aligned} \text{Min } & Z = y_1 + 1.5y_2 + 0.5y_3 + x_1^2 + x_2^2 \\ s.t. \quad & (x_1 - 2)^2 - x_2 \leq 0 \\ & x_1 - 2y_1 \geq 0 \\ & x_1 - x_2 - 4(1 - y_2) \geq 0 \\ & x_2 - y_2 \geq 0 \\ & x_1 + x_2 \geq 3y_3 \\ & y_1 + y_2 + y_3 \geq 1 \\ & 0 \leq x_1 \leq 4, 0 \leq x_2 \leq 4 \\ & y_1, y_2, y_3 = 0, 1 \end{aligned} \tag{106}$$

The optimal solution of this problem is given by $y_1 = 0$, $y_2 = 1$, $y_3 = 0$, $x_1 = 1$, $x_2 = 1$, $Z = 3.5$. Figure 48 shows the progress of the iterations with the OA and GBD methods, while the table lists the number of NLP and MILP subproblems that are solved with each of the methods. For the case of the MILP problems the total number of LPs solved at the branch-and-bound nodes are also reported.

Extensions of MINLP Methods. Extensions of the methods described above include a quadratic approximation to (RM-OAF) [247] using an approximation of the Hessian matrix. The quadratic approximations can help to reduce the number of major iterations, since an improved representation of the continuous space is obtained. This, however, comes at the price of having to solve an MIQP instead of an MILP at each iteration.

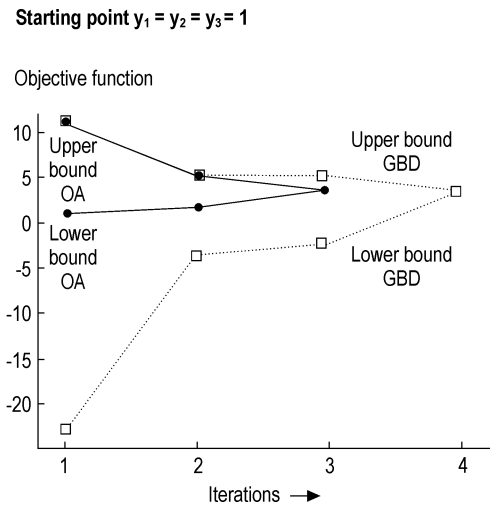

Figure 48. Progress of iterations with OA and GBD methods, and number of subproblems for the BB, OA, GBD, and ECP methods.

The master problem 100 can involve a rather large number of constraints, due to the accumulation of linearizations. One option is to keep only the last linearization point, but this can lead to nonconvergence even in convex problems, since then the monotonic increase of the lower bound is not guaranteed. As shown [248], linear approximations to the nonlinear objective and constraints can be aggregated with an MILP master problem that is a hybrid of the GBD and OA methods.

For the case when linear equalities of the form $h(x, y) = 0$ are added to problem 95 there is no major difficulty, since these are invariant to the linearization points. If the equations are nonlinear, however, there are two difficulties. First, it is not possible to enforce the linearized equalities at K points. Second, the nonlinear equations may generally introduce nonconvexities, unless they relax as convex inequalities [255]. Kocis and Grossmann [256] proposed an equality relaxation strategy in which the nonlinear equalities are replaced by the inequalities

$$T^k \nabla h\left(x^k, y^k\right)^T \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \leq 0 \tag{107}$$

where $T^k = \{t_{ii}^k\}$, and t_{ii}^k=sign (λ_i^k) sign in which λ_i^k is the multiplier associated to the equation $h_i(x, y) = 0$. Note that if these equations relax as the inequalities $h(x, y) \leq 0$ for all y and $h(x, y)$ is convex, this is a rigorous procedure. Otherwise, nonvalid supports may be generated. Also, in the master problem 105 of GBD, no special provision is required to handle equations, since these are simply included in the Lagrangian cuts. However, similar difficulties as in OA arise if the equations do not relax as convex inequalities.

When $f(x,y)$ and $g(x,y)$ are nonconvex in problem 95, or when nonlinear equalities $h(x, y) = 0$ are present, two difficulties arise. First, the NLP subproblems 96 – 98 may not have a unique local optimum solution. Second, the master problem (M-MIP) and its variants (e.g., M-MIPF, M-GBD, M-MIQP) do not guarantee a valid lower bound Z_L^K or a valid bounding representation with which the global optimum may be cut off.

Rigorous global optimization approaches for addressing nonconvexities in MINLP problems can be developed when special structures are assumed in the continuous terms (e.g. bilinear, linear fractional, concave separable). Specifically, the idea is to use convex envelopes or underestimators to formulate lower-bounding convex MINLP problems. These are then combined with global optimization techniques for continuous variables [206, 207, 209, 214, 216, 222, 257], which usually take the form of spatial branch-and-bound methods. The lower bounding MINLP problem has the general form,

$$\begin{aligned} &\text{Min } Z = \overline{f}(x,y) \\ &\text{s.t. } \overline{g}_j(x,y) \leq 0 \; j \in J \\ &\quad x \in X, \; y \in Y \end{aligned} \tag{108}$$

where $\overline{f}, \overline{g}$ are valid convex underestimators such that $\overline{f}(x,y) \leq f(x,y)$ and the inequalities $\overline{g}(x,y) \leq 0$ are satisfied if $g(x,y) \leq 0$. A typical example of convex underestimators are the convex envelopes for bilinear terms [208].

Examples of global optimization methods for MINLP problems include the branch-and-reduce method [209, 210], the α-BB method [212], the reformulation/spatial branch-and-bound search method [258], the branch-and-cut method [259], and the disjunctive branch-and-bound method [260]. All these methods rely on a branch-and-bound procedure. The difference lies in how to perform the branching on the discrete and continuous variables. Some methods perform the spatial tree enumeration on both the discrete and continuous variables of problem 108. Other methods perform a spatial branch and bound on the continuous variables and solve the corresponding MINLP problem 108 at each node using any of the methods reviewed above. Finally, other methods branch on the discrete variables of problem 108, and switch to a spatial branch and bound on nodes where a feasible value for the discrete variables is found. The methods also rely on procedures for tightening the lower and upper bounds of the variables, since these have a great effect on the quality of the underestimators. Since the tree searches are not finite (except for ε convergence), these methods can be computationally expensive. However, their major advantage is that they can rigorously find the global optimum. Specific cases of nonconvex MINLP problems have been handled. An example is the work of Pörn and Westerlund [261], who addressed the solution of MINLP problems with pseudoconvex objective function and convex inequalities through an extension of the ECP method.

The other option for handling nonconvexities is to apply a heuristic strategy to try to reduce as much as possible the effect of nonconvexities. While not being rigorous, this requires much less computational effort. We describe here an approach for reducing the effect of nonconvexities at the level of the MILP master problem.

Viswanathan and Grossmann [262] proposed to introduce slacks in the MILP master problem to reduce the likelihood of cutting off feasible solutions. This master problem (augmented penalty/equality relaxation) has the form:

$$\begin{aligned} &\min Z^K = \alpha + \sum_{k=1}^{K} w_p^k p^k + w_q^k q^k \\ &\text{s.t.} \quad \left.\begin{aligned} &\alpha \geq f(x^k, y^k) \\ &\quad + \nabla f(x^k, y^k)^T \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \\ &T^k \nabla h(x^k, y^k)^T \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \leq p^k \\ &g(x^k, y^k) + \nabla g(x^k, y^k)^T \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \\ &\quad \leq q^k \end{aligned} \right\} k = 1, \ldots K \end{aligned} \tag{109}$$

$$\sum_{i\in B^k} y_i - \sum_{i\in N^k} y_i \le \left|B^k\right| - 1 \; k=1,\ldots K$$
$$x\in X, y\in Y, \alpha\in R^1, p^k, q^k \ge 0$$

where w_p^k, w_q^k are weights that are chosen sufficiently large (e.g., 1000 times the magnitude of the Lagrange multiplier). Note that if the functions are convex then the MILP master problem 109 predicts rigorous lower bounds to problem 95 since all the slacks are set to zero.

Computer Codes for MINLP. Computer codes for solving MINLP problems include the following. The program DICOPT [262] is an MINLP solver that is available in the modeling system GAMS [263]. The code is based on the master problem 109 and the NLP subproblems 97. This code also uses relaxed 96 to generate the first linearization for the above master problem, with which the user need not specify an initial integer value. Also, since bounding properties of problem 109 cannot be guaranteed, the search for nonconvex problems is terminated when there is no further improvement in the feasible NLP subproblems. This is a heuristic that works reasonably well in many problems. Codes that implement the branch-and-bound method using subproblems 96 include the code MINLP_BB, which is based on an SQP algorithm [243] and is available in AMPL, the code BARON [264], which also implements global optimization capabilities, and the code SBB, which is available in GAMS [263]. The code a-ECP implements the extended cutting-plane method [249], including the extension by PÖRN and WESTERLUND [261]. Finally, the code MINOPT [265] also implements the OA and GBD methods, and applies them to mixed-integer dynamic optimization problems. It is difficult to derive general conclusions on the efficiency and reliability of all these codes and their corresponding methods, since no systematic comparison has been made. However, one might anticipate that branch-and-bound codes are likely to perform better if the relaxation of the MINLP is tight. Decomposition methods based on OA are likely to perform better if the NLP subproblems are relatively expensive to solve, while GBD can perform with some efficiency if the MINLP is tight and there are many discrete variables. ECP methods tend to perform well on mostly linear problems.

Logic-Based Optimization. Given difficulties in the modeling and solution of mixed integer problems, the following major approaches based on logic-based techniques have emerged: generalized disjunctive programming 110 [266], mixed-logic linear programming (MLLP) [267], and constraint programming (CP) [268]. The motivations for this logic-based modeling has been to facilitate the modeling, reduce the combinatorial search effort, and improve the handling of nonlinearities. In this section we mostly concentrate on generalized disjunctive programming and provide a brief reference to constraint programming. A general review of logic-based optimization can be found in [269, 309].

Generalized disjunctive programming in 110 [266] is an extension of disjunctive programming [270] that provides an alternative way of modeling MILP and MINLP problems. The general formulation 110 is as follows:

$$\begin{aligned}
&\text{Min } Z = \sum_{k\in K} c_k + f(x)\\
&s.t. \quad g(x) \le 0\\
&\bigvee_{j\in J_k} \begin{bmatrix} Y_{jk} \\ h_{jk}(x) \le 0 \\ c_k = \gamma_{jk} \end{bmatrix}, \; k\in K \qquad (110)\\
&\Omega(Y) = True\\
&x\in R^n, c\in R^m, Y\in \{true,\ false\}^m
\end{aligned}$$

where Y_{jk} are the Boolean variables that decide whether a term j in a disjunction $k \in K$ is true or false, and x are continuous variables. The objective function involves the term $f(x)$ for the continuous variables and the charges c_k that depend on the discrete choices in each disjunction $k \in K$. The constraints $g(x) \le 0$ hold regardless of the discrete choice, and $h_{jk}(x) \le 0$ are conditional constraints that hold when Y_{jk} is true in the j-th term of the k-th disjunction. The cost variables c_k correspond to the fixed charges, and are equal to γ_{jk} if the Boolean variable Y_{jk} is true. $\Omega(Y)$ are logical relations for the Boolean variables expressed as propositional logic.

Problem 110 can be reformulated as an MINLP problem by replacing the Boolean variables by binary variables y_{jk},

$$\begin{aligned}
&\text{Min } Z=\textstyle\sum_{k\in K}\sum_{j\in J_k}\gamma_{jk}y_{jk}+f(x)\\
&\text{s.t.}\quad g(x)\le 0\\
&h_{jk}(x)\le M_{jk}\left(1-y_{jk}\right),\ j\in J_k,\ k\in K \qquad \text{(BM)}\\
&\textstyle\sum_{j\in J_k}y_{jk}=1,\ k\in K\\
&Ay\le a\\
&0\le x\le x^U,\ y_{jk}\in\{0,1\},\ j\in J_k,\ k\in K
\end{aligned} \tag{111}$$

where the disjunctions are replaced by "Big-M" constraints which involve a parameter M_{jk} and binary variables y_{jk}. The propositional logic statements $\Omega(Y)$ = True are replaced by the linear constraints $Ay \le a$ [271] and [272]. Here we assume that x is a nonnegative variable with finite upper bound x^{U}. An important issue in model 111 is how to specify a valid value for the Big-M parameter M_{jk}. If the value is too small, then feasible points may be cut off. If M_{jk} is too large, then the continuous relaxation might be too loose and yield poor lower bounds. Therefore, finding the smallest valid value for M_{jk} is desired. For linear constraints, one can use the upper and lower bound of the variable x to calculate the maximum value of each constraint, which then can be used to calculate a valid value of M_{jk}. For nonlinear constraints one can in principle maximize each constraint over the feasible region, which is a nontrivial calculation.

Lee and Grossmann [273] have derived the *convex hull relaxation* of problem 110. The basic idea is as follows. Consider a disjunction $k \in K$ that has convex constraints

$$\begin{aligned}
&\bigvee_{j\in J_k}\begin{bmatrix} Y_{jk}\\ h_{jk}(x)\le 0\\ c=\gamma_{jk}\end{bmatrix}\\
&0\le x\le x^U, c\ge 0
\end{aligned} \tag{112}$$

where $h_{jk}(x)$ are assumed to be convex and bounded over x. The convex hull relaxation of disjunction 112 [242] is given as follows:

$$\begin{aligned}
&x=\textstyle\sum_{j\in J_k}v^{jk},\quad c=\sum_{j\in J}\lambda_{jk}\gamma_{jk}\\
&0\le v^{jk}\le\lambda_{jk}x^{\mathrm{U}}_{jk},\ j\in J_k\\
&\textstyle\sum_{j\in J_k}\lambda_{jk}=1,\ 0\le\lambda_{jk}\le 1,\ j\in J_k \qquad \text{(CH)}\\
&\lambda_{jk}h_{jk}\left(v^{jk}/\lambda_{jk}\right)\le 0,\ j\in J_k\\
&x,c,v^{jk}\ge 0,\ j\in J_k
\end{aligned} \tag{113}$$

where v^{jk} are disaggregated variables that are assigned to each term of the disjunction $k \in K$, and λ_{jk} are the weight factors that determine the feasibility of the disjunctive term. Note that when λ_{jk} is 1, then the j-th term in the k-th disjunction is enforced and the other terms are ignored. The constraints $\lambda_{jk}h_{jk}\left(v^{jk}/\lambda_{jk}\right)$ are convex if $h_{jk}(x)$ is convex [274, p. 160]. A formal proof can be found in [242]. Note that the convex hull 113 reduces to the result by Balas [275] if the constraints are linear. Based on the convex hull relaxation 113, Lee and Grossmann [273] proposed the following convex relaxation program of problem 110.

$$\begin{aligned}
&\text{Min } Z^L=\textstyle\sum_{k\in K}\sum_{j\in J_k}\gamma_{jk}\lambda_{jk}+f(x)\\
&\text{s.t.}\quad g(x)\le 0\\
&x=\textstyle\sum_{j\in J_k}v^{jk},\ \sum_{j\in J_k}\lambda_{jk}=1,\ k\in K \qquad \text{(CRP)}\\
&0\le x,\ v^{jk}\le x^{\mathrm{U}},\ 0\le\lambda_{jk}\le 1,\ j\in J_k,\ k\in K\\
&\lambda_{jk}h_{jk}\left(v^{jk}/\lambda_{jk}\right)\le 0,\ j\in J_k,\ k\in K\\
&A\lambda\le a\\
&0\le x,\ v^{jk}\le x^{\mathrm{U}},\ 0\le\lambda_{jk}\le 1,\ j\in J_k,\ k\in K
\end{aligned} \tag{114}$$

where x^{U} is a valid upper bound for x and v. Note that the number of constraints and variables increases in problem 114 compared with problem 110. Problem 114 has a unique optimal solution and it yields a valid lower bound to the optimal solution of problem 110 [273]. Grossmann and Lee [276] proved that problem 114 has the useful property that the lower bound is greater than or equal to the lower bound predicted from the relaxation of problem 111.

Further description of algorithms for disjunctive programming can be found in [277].

Constraint Programming. Constraint programming (CP) [268, 269] is a relatively new modeling and solution paradigm that was originally developed to solve feasibility problems, but it has been extended to solve optimization problems as well. Constraint programming is very expressive, as continuous, integer, and Boolean variables are permitted and, moreover, variables can be indexed by other variables. Constraints can be expressed in algebraic form (e.g., $h(x) \le 0$), as disjunctions (e.g., $[A_1x \le b_1] \vee A_2x \le b_2]$), or as conditional logic statements (e.g., If $g(x) \le 0$ then $r(x) \le 0$). In addition, the language can support special implicit functions such as the all-different (x_1, x_2, ... x_n) constraint for

assigning different values to the integer variables $x_1, x_2, \ldots x_n$. The language consists of C ++ procedures, although the recent trend has been to provide higher level languages such as OPL. Other commercial CP software packages include ILOG Solver [278], CHIP [279], and ECLiPSe [280].

10.6. Dynamic Optimization

Interest in dynamic simulation and optimization of chemical processes has increased significantly during the last two decades. Chemical processes are modeled dynamically using differential-algebraic equations (DAEs), consisting of differential equations that describe the dynamic behavior of the system, such as mass and energy balances, and algebraic equations that ensure physical and thermodynamic relations. Typical applications include control and scheduling of batch processes; startup, upset, shut-down, and transient analysis; safety studies; and the evaluation of control schemes. We state a general differential-algebraic optimization problem 115 as follows:

$$
\begin{aligned}
\text{Min} \quad & \Phi\left(z\left(t_{\mathrm{f}}\right); y\left(t_{\mathrm{f}}\right); u\left(t_{\mathrm{f}}\right); t_{\mathrm{f}}; p\right) \\
\text{s.t.} \quad & F\left(dz/dt; z(t); u(t); t; p\right) = 0, \; z(0) = z_0 \\
& G_{\mathrm{s}}\left[z\left(t_{\mathrm{s}}\right); y\left(t_{\mathrm{s}}\right); u\left(t_{\mathrm{s}}\right); t_{\mathrm{s}}; p\right)] = 0 \\
& z^{\mathrm{L}} \le z(t) \le x^{\mathrm{U}} \\
& y^{\mathrm{L}} \le y(t) \le y^{\mathrm{U}} \\
& u^{\mathrm{L}} \le u(t) \le y^{\mathrm{U}} \\
& p^{\mathrm{L}} \le p \le p^{\mathrm{U}} \\
& t_{\mathrm{f}}^{t} \le t_{\mathrm{f}} \le t_{\mathrm{f}}^{U}
\end{aligned}
\tag{115}
$$

where Φ is a scalar objective function at final time t_{f}, and F are DAE constraints, G_{s} additional point conditions at times t_{s}, $z(t)$ differential state profile vectors, $y(t)$ algebraic state profile vectors, $u(t)$ control state profile vectors, and p is a time-independent parameter vector.

We assume, without loss of generality, that the index of the DAE system is one, consistent initial conditions are available, and the objective function is in the above Mayer form. Otherwise, it is easy to reformulate problems to this form. Problem 115 can be solved either by the variational approach or by applying some level of discretization that converts the original continuous time problem into a discrete problem. Early solution strategies, known as indirect methods, were focused on solving the classical variational conditions for optimality. On the other hand, methods that discretize the original continuous time formulation can be divided into two categories, according to the level of discretization. Here we distinguish between the methods that discretize only the control profiles (partial discretization) and those that discretize the state and control profiles (full discretization). Basically, the partially discretized problem can be solved either by dynamic programming or by applying a nonlinear programming (NLP) strategy (direct-sequential). A basic characteristic of these methods is that a feasible solution of the DAE system, for given control values, is obtained by integration at every iteration of the NLP solver. The main advantage of these approaches is that, for the NLP solver, they generate smaller discrete problems than full discretization methods.

Methods that fully discretize the continuous time problem also apply NLP strategies to solve the discrete system and are known as direct-simultaneous methods. These methods can use different NLP and discretization techniques, but the basic characteristic is that they solve the DAE system only once, at the optimum. In addition, they have better stability properties than partial discretization methods, especially in the presence of unstable dynamic modes. On the other hand, the discretized optimization problem is larger and requires large-scale NLP solvers, such as SOCS, CONOPT, or IPOPT.

With this classification we take into account the degree of discretization used by the different methods. Below we briefly present the description of the variational methods, followed by methods that partially discretize the dynamic optimization problem, and finally we consider full discretization methods for problem 115.

Variational Methods. These methods are based on the solution of the first-order necessary conditions for optimality that are obtained from Pontryagin's maximum principle [281, 282]. If we consider a version of problem 115 without bounds, the optimality conditions are formulated as a set of DAEs:

$$
\frac{\partial F\left(z,y,u,p,t\right)}{\partial z'} \lambda' = \frac{\partial H}{\partial z} = \frac{\partial F\left(z,y,u,p,t\right)}{\partial z} \lambda \tag{116a}
$$

$$F(z,y,u,p,t)=0 \tag{116b}$$

$$G_f(z,y,u,p,t_f)=0 \tag{116c}$$

$$G_s(z,y,u,p,t_s)=0 \tag{116d}$$

$$\frac{\partial H}{\partial y}=\frac{\partial F(z,y,u,p,t)}{\partial y}\lambda=0 \tag{116e}$$

$$\frac{\partial H}{\partial u}=\frac{\partial F(z,y,u,p,t)}{\partial u}\lambda=0 \tag{116f}$$

$$\int_0^{t_f}\frac{\partial F(z,y,u,p,t)}{\partial p}\lambda\,dt=0 \tag{116g}$$

where the Hamiltonian H is a scalar function of the form $H(t)=F(z, y, u, p, y)^T\lambda(t)$ and $\lambda(t)$ is a vector of adjoint variables. Boundary and jump conditions for the adjoint variables are given by:

$$\begin{aligned}&\tfrac{\partial F}{\partial z'}\lambda(t_f)+\tfrac{\partial \Phi}{\partial z}+\tfrac{\partial G_f}{\partial z}v_f=0\\&\tfrac{\partial F}{\partial z'}\lambda\left(t_s^-\right)+\tfrac{\partial G_s}{\partial z}v_s=\tfrac{\partial F}{\partial z'}\lambda\left(t_s^+\right)\end{aligned} \tag{117}$$

where v_f, v_s are the multipliers associated with the final time and point constraints, respectively. The most expensive step lies in obtaining a solution to this boundary value problem. Normally, the state variables are given as initial conditions, and the adjoint variables as final conditions. This formulation leads to boundary value problems (BVPs) that can be solved by a number of standard methods including single shooting, invariant embedding, multiple shooting, or some discretization method such as collocation on finite elements or finite differences. Also the point conditions lead to an additional calculation loop to determine the multipliers v_f and v_s. On the other hand, when bound constraints are considered, the above conditions are augmented with additional multipliers and associated complementarity conditions. Solving the resulting system leads to a combinatorial problem that is prohibitively expensive except for small problems.

Partial Discretization. With partial discretization methods (also called sequential methods or control vector parametrization), only the control variables are discretized. Given the initial conditions and a given set of control parameters, the DAE system is solved with a differential algebraic equation solver at each iteration. This produces the value of the objective function, which is used by a nonlinear programming solver to find the optimal parameters in the control parametrization ν. The sequential method is reliable when the system contains only stable modes. If this is not the case, finding a feasible solution for a given set of control parameters can be very difficult. The time horizon is divided into time stages and at each stage the control variables are represented with a piecewise constant, a piecewise linear, or a polynomial approximation [283, 284]. A common practice is to represent the controls as a set of Lagrange interpolation polynomials.

For the NLP solver, gradients of the objective and constraint functions with respect to the control parameters can be calculated with the sensitivity equations of the DAE system, given by:

$$\begin{aligned}&\frac{\partial F}{\partial z'}^T s_k'+\frac{\partial F}{\partial z}^T s_k+\frac{\partial F}{\partial y}^T w_k+\frac{\partial F}{\partial q_k}^T=0,\\&s_k(0)=\frac{\partial z(0)}{\partial q_k}\quad k=1,\ldots N_q\end{aligned} \tag{118}$$

where $s_k(t)=\frac{\partial z(t)}{\partial q_k}$, $w_k(t)=\frac{\partial y(t)}{\partial q_k}$, and $q^T=[p^T,\nu^T]$. As can be inferred from Equation 118, the cost of obtaining these sensitivities is directly proportional to N_q, the number of decision variables in the NLP. Alternately, gradients can be obtained by integration of the adjoint Equations 116a, 116e, 116g [282, 285, 286] at a cost independent of the number of input variables and proportional to the number of constraints in the NLP.

Methods that are based on this approach cannot treat directly the bounds on state variables, because the state variables are not included in the nonlinear programming problem. Instead, most of the techniques for dealing with inequality path constraints rely on defining a measure of the constraint violation over the entire horizon, and then penalizing it in the objective function, or forcing it directly to zero through an end-point constraint [287]. Other techniques approximate the constraint satisfaction (constraint aggregation methods) by introducing an exact penalty function [286, 288] or a Kreisselmeier–Steinhauser function [288] into the problem.

Finally, initial value solvers that handle path constraints directly have been developed [284].

The main idea is to use an algorithm for constrained dynamic simulation, so that any admissible combination of the control parameters produces an initial value problem that is feasible with respect to the path constraints. The algorithm proceeds by detecting activation and deactivation of the constraints during the solution, and solving the resulting high-index DAE system and their related sensitivities.

Full Discretization. Full discretization methods explicitly discretize all the variables of the DAE system and generate a large-scale nonlinear programming problem that is usually solved with a successive quadratic programming (SQP) algorithm. These methods follow a simultaneous approach (or infeasible path approach); that is, the DAE system is not solved at each iteration; it is only solved at the optimum point. Because of the size of the problem, special decomposition strategies are used to solve the NLP efficiently. Despite this characteristic, the simultaneous approach has advantages for problems with state variable (or path) constraints and for systems where instabilities occur for a range of inputs. In addition, the simultaneous approach can avoid intermediate solutions that may not exist, are difficult to obtain, or require excessive computational effort. There are mainly two different approaches to discretize the state variables explicitly, multiple shooting [289, 290] and collocation on finite elements [181, 191, 291].

With *multiple shooting*, time is discretized into P stages and control variables are parametrized using a finite set of control parameters in each stage, as with partial discretization. The DAE system is solved on each stage, $i = 1,\ldots P$ and the values of the state variables $z(t_i)$ are chosen as additional unknowns. In this way a set of relaxed, decoupled initial value problems (IVP) is obtained:

$$\begin{aligned} &F\left(dz/dt; z\left(t\right); y\left(t\right); \nu_i; p\right) = 0, \\ &t \in \left[t_{i-1}, t_i\right], z\left(t_{i-1}\right) = z_i \\ &z_{i+1} - z\left(t_i; z_i; \nu_i; p\right) = 0, i = 1, \ldots P-1 \end{aligned} \tag{119}$$

Note that continuity among stages is treated through equality constraints, so that the final solution satisfies the DAE system. With this approach, inequality constraints for states and controls can be imposed directly at the grid points, but path constraints for the states may not be satisfied between grid points. This problem can be avoided by applying penalty techniques to enforce feasibility, like the ones used in the sequential methods.

The resulting NLP is solved using SQP-type methods, as described above. At each SQP iteration, the DAEs are integrated in each stage and objective and constraint gradients with respect to p, z_i, and ν_i are obtained using sensitivity equations, as in problem 118. Compared to sequential methods, the NLP contains many more variables, but efficient decompositions have been proposed [290] and many of these calculations can be performed in parallel.

In *collocation methods*, the continuous time problem is transformed into an NLP by approximating the profiles as a family of polynomials on finite elements. Various polynomial representations are used in the literature, including Lagrange interpolation polynomials for the differential and algebraic profiles [291]. In [191] a Hermite–Simpson collocation form is used, while Cuthrell and Biegler [292] and Tanartkit and Biegler [293] use a monomial basis for the differential profiles. All of these representations stem from implicit Runge–Kutta formulae, and the monomial representation is recommended because of smaller condition numbers and smaller rounding errors. Control and algebraic profiles, on the other hand, are approximated using Lagrange polynomials.

Discretizations of problem 115 using collocation formulations lead to the largest NLP problems, but these can be solved efficiently using large-scale NLP solvers such as IPOPT and by exploiting the structure of the collocation equations. Biegler et al. [181] provide a review of dynamic optimization methods using simultaneous methods. These methods offer a number of advantages for challenging dynamic optimization problems, which include:

- Control variables can be discretized at the same level of accuracy as the differential and algebraic state variables. The KKT conditions of the discretized problem can be shown to be consistent with the variational conditions of problem 115. Finite elements allow for discontinuities in control profiles.
- Collocation formulations allow problems with unstable modes to be handled in an

efficient and well-conditioned manner. The NLP formulation inherits stability properties of boundary value solvers. Moreover, an elementwise decomposition has been developed that pins down unstable modes in problem 115.

- Collocation formulations have been proposed with moving finite elements. This allows the placement of elements both for accurate breakpoint locations of control profiles as well as accurate DAE solutions.

Dynamic optimization by collocation methods has been used for a wide variety of process applications including batch process optimization, batch distillation, crystallization, dynamic data reconciliation and parameter estimation, nonlinear model predictive control, polymer grade transitions and process changeovers, and reactor design and synthesis. A review of this approach can be found in [294].

10.7. Development of Optimization Models

The most important aspect of a successful optimization study is the formulation of the optimization model. These models must reflect the real-world problem so that meaningful optimization results are obtained, and they also must satisfy the properties of the problem class. For instance, NLPs addressed by gradient-based methods require functions that are defined in the variable domain and have bounded and continuous first and second derivatives. In mixed integer problems, proper formulations are also needed to yield good lower bounds for efficient search. With increased understanding of optimization methods and the development of efficient and reliable optimization codes, optimization practitioners now focus on the *formulation* of optimization models that are realistic, well-posed, and inexpensive to solve. Finally, convergence properties of NLP, MILP, and MINLP solvers require accurate first (and often second) derivatives from the optimization model. If these contain numerical errors (say, through finite difference approximations) then performance of these solvers can deteriorate considerably. As a result of these characteristics, modeling platforms are essential for the formulation task. These are classified into two broad areas: optimization modeling platforms and simulation platforms with optimization.

Optimization modeling platforms provide general purpose interfaces for optimization algorithms and remove the need for the user to interface to the solver directly. These platforms allow the general formulation for all problem classes discussed above with direct interfaces to state of the art optimization codes. Three representative platforms are GAMS (General Algebraic Modeling Systems), AMPL (A Mathematical Programming Language), and AIMMS (Advanced Integrated Multidimensional Modeling Software). All three require problem-model input via a declarative modeling language and provide exact gradient and Hessian information through automatic differentiation strategies. Although possible, these platforms were not designed to handle externally added procedural models. As a result, these platforms are best applied on optimization models that can be developed entirely within their modeling framework. Nevertheless, these platforms are widely used for large-scale research and industrial applications. In addition, the MATLAB platform allows the flexible formulation of optimization models as well, although it currently has only limited capabilities for automatic differentiation and limited optimization solvers. More information on these and other modeling platforms can be found on the NEOS server www-neos.mcs.anl.gov

Simulation platforms with optimization are often dedicated, application-specific modeling tools to which optimization solvers have been interfaced. These lead to very useful optimization studies, but because they were not originally designed for optimization models, they need to be used with some caution. In particular, most of these platforms do not provide exact derivatives to the optimization solver; often they are approximated through finite difference. In addition, the models themselves are constructed and calculated through numerical procedures, instead of through an open declarative language. Examples of these include widely used process simulators such as Aspen/Plus, PRO/II, and Hysys. More recent platforms such as Aspen Custom Modeler and gPROMS include declarative models and exact derivatives.

For optimization tools linked to procedural models, reliable and efficient automatic differ-

entiation tools are available that link to models written, say, in FORTRAN and C, and calculate exact first (and often second) derivatives. Examples of these include ADIFOR, ADOL-C, GRESS, Odyssee, and PADRE. When used with care, these can be applied to existing procedural models and, when linked to modern NLP and MINLP algorithms, can lead to powerful optimization capabilities. More information on these and other automatic differentiation tools can be found on: http://www-unix.mcs.anl.gov/autodiff/AD_Tools/.

Finally, the availability of automatic differentiation and related sensitivity tools for differential equation models allows for considerable flexibility in the formulation of optimization models. In [295] a seven-level modeling hierarchy is proposed that matches optimization algorithms with models that range from completely open (fully declarative model) to fully closed (entirely procedural without sensitivities). At the lowest, fully procedural level, only derivative-free optimization methods are applied, while the highest, declarative level allows the application of an efficient large-scale solver that uses first and second derivatives. Depending on the modeling level, optimization solver performance can vary by several orders of magnitude.

11. Probability and Statistics [15, 296 – 303]

The models treated thus far have been deterministic, that is, if the parameters are known the outcome is determined. In many situations, all the factors cannot be controlled and the outcome may vary randomly about some average value. Then a range of outcomes has a certain probability of occurring, and statistical methods must be used. This is especially true in quality control of production processes and experimental measurements. This chapter presents standard statistical concepts, sampling theory and statistical decisions, and factorial design of experiments or analysis of variances. Multivariant linear and nonlinear regression is treated in Chapter 2.

11.1. Concepts

Suppose N values of a variable y, called $y_1, y_2, \ldots, y_N$, might represent N measurements of the same quantity. The *arithmetic mean* $E(y)$ is

$$E(y) = \frac{\sum_{i=1}^{N} y_i}{N}$$

The *median* is the middle value (or average of the two middle values) when the set of numbers is arranged in increasing (or decreasing) order. The *geometric mean* $\bar{y}_G$ is

$$\overline{y}_G = (y_1 y_2 \ldots y_N)^{1/N}$$

The *root-mean-square* or quadratic mean is

$$\text{Root}-\text{mean}-\text{square} = \sqrt{E(y^2)} = \sqrt{\sum_{i=1}^{N} y_i^2 / N}$$

The *range* of a set of numbers is the difference between the largest and the smallest members in the set. The *mean deviation* is the mean of the deviation from the mean.

$$\text{Mean}-\text{deviation} = \frac{\sum_{i=1}^{N} |y_i - E(y)|}{N}$$

The *variance* is

$$var(y) = \sigma^2 = \frac{\sum_{i=1}^{N} (y_i - E(y))^2}{N}$$

and the *standard deviation* σ is the square root of the variance.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N} (y_i - E(y))^2}{N}}$$

If the set of numbers $\{y_i\}$ is a small sample from a larger set, then the *sample average*

$$\overline{y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

is used in calculating the *sample variance*

$$s^2 = \frac{\sum_{i=1}^{n} (y_i - \overline{y})^2}{n-1}$$

and the sample standard deviation

$$s = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \overline{y})^2}{n-1}}$$

The value $n-1$ is used in the denominator because the deviations from the sample average must total zero:

$$\sum_{i=1}^{n}(y_i-\overline{y})=0$$

Thus, knowing $n-1$ values of $y_i-\bar{y}$ and the fact that there are n values automatically gives the n-th value. Thus, only $n-1$ degrees of freedom ν exist. This occurs because the unknown mean $E(y)$ is replaced by the sample mean y derived from the data.

If data are taken consecutively, running totals can be kept to permit calculation of the mean and variance without retaining all the data:

$$\sum_{i=1}^{n}(y_i-\overline{y})^2=\sum_{i=1}^{n}y_1^2-2\overline{y}\sum_{i=1}^{n}y_i+(\overline{y})^2$$

$$\overline{y}=\sum_{i=1}^{n}y_i/n$$

Thus,

$$n,\sum_{i=1}^{n}y_1^2,\ \text{and}\sum_{i=1}^{n}y_i$$

are retained, and the mean and variance are computed when needed.

Repeated observations that differ because of experimental error often vary about some central value in a roughly symmetrical distribution in which small deviations occur more frequently than large deviations. In plotting the number of times a discrete event occurs, a typical curve is obtained, which is shown in Figure 49. Then the probability p of an event (score) occurring can be thought of as the ratio of the number of times it was observed divided by the total number of events. A continuous representation of this probability density function is given by the *normal distribution*

$$p(y)=\frac{1}{\sigma\sqrt{2\pi}}\mathrm{e}^{-[y-E(y)]^2/2\sigma^2}. \qquad (120)$$

This is called a normal probability distribution function. It is important because many results are insensitive to deviations from a normal distribution. Also, the central limit theorem says that if an overall error is a linear combination of component errors, then the distribution of errors tends to be normal as the number of components increases, almost regardless of the distribution of the component errors (i.e., they need not be normally distributed). Naturally, several sources of error must be present and one error cannot predominate (unless it is normally distributed). The normal distribution function is calculated easily; of more value are integrals of the function, which are given in Table 12; the region of interest is illustrated in Figure 50.

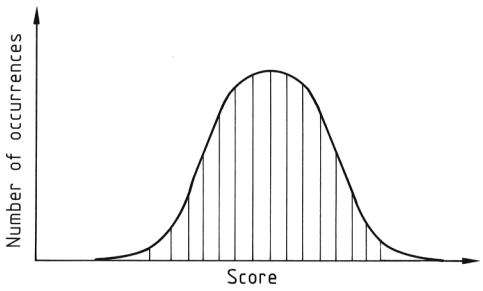

Figure 49. Frequency of occurrence of different scores

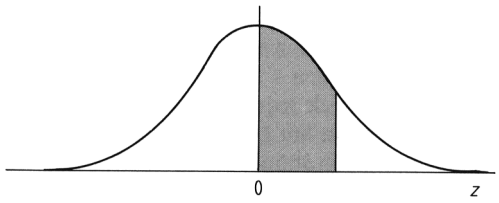

Figure 50. Area under normal curve

For a small sample, the variance can only be estimated with the sample variance s^2. Thus, the normal distribution cannot be used because σ is not known. In such cases *Student's t-distribution*, shown in Figure 51 [303, p. 70], is used:

$$p(y)=\frac{y_0}{\left(1+\frac{t^2}{n-1}\right)^{n/2}},\ t=\frac{\overline{y}-E(y)}{s/\sqrt{n}}$$

and y_0 is chosen such that the area under the curve is one. The number $\nu=n-1$ is the degrees of freedom, and as ν increases, Student's t-distribution approaches the normal distribution. The normal distribution is adequate (rather than the t-distribution) when $\nu>15$, except for the tails of the curve which require larger ν. Integrals of the t-distribution are given in Table 13, the region of interest is shown in Figure 52.

Table 12. Area under normal curve *

$$F(z) = \frac{1}{\sqrt{2\pi}} \int_0^z \mathrm{e}^{-z^2/2} \mathrm{d}z$$

z	$F(z)''$	z	$F(z)''$
0.0	0.0000	1.5	0.4332
0.1	0.0398	1.6	0.4452
0.2	0.0793	1.7	0.4554
0.3	0.1179	1.8	0.4641
0.4	0.1554	1.9	0.4713
0.5	0.1915	2.0	0.4772
0.6	0.2257	2.1	0.4821
0.7	0.2580	2.2	0.4861
0.8	0.2881	2.3	0.4893
0.9	0.3159	2.4	0.4918
1.0	0.3413	2.5	0.4938
1.1	0.3643	2.7	0.4965
1.2	0.3849	3.0	0.4987
1.3	0.4032	4.0	0.499968
1.4	0.4192	5.0	0.4999997

* Table gives the probability F that a random variable will fall in the shaded region of Figure 50. For a more complete table (in slightly different form), see [23, Table 26.1]. This table is obtained in Microsoft Excel with the function NORMDIST(z,0,1,1)-0.5.

Table 13. Percentage points of area under Students t-distribution *

ν	α=0.10	α=0.05	α=0.01	α=0.001
1	6.314	12.706	63.657	636.619
2	2.920	4.303	9.925	31.598
3	2.353	3.182	5.841	12.941
4	2.132	2.776	4.604	8.610
5	2.015	2.571	4.032	6.859
6	1.943	2.447	3.707	5.959
7	1.895	2.365	3.499	5.408
8	1.860	2.306	3.355	5.041
9	1.833	2.262	3.250	4.781
10	1.812	2.228	3.169	4.587
15	1.753	2.131	2.947	4.073
20	1.725	2.086	2.845	3.850
25	1.708	2.060	2.787	3.725
30	1.697	2.042	2.750	3.646
∞	1.645	1.960	2.576	3.291

* Table gives t values such that a random variable will fall in the shaded region of Figure 52 with probability α. For a one-sided test the confidence limits are obtained for $\alpha/2$. For a more complet table (in slightly different form), see [23, Table 26.10]. This table is obtained in Microsoft Excel with the function TINV(α,ν).

Other probability distribution functions are useful. Any distribution function must satisfy the following conditions:

$$0 \leq F(x) \leq 1$$

$$F(-\infty) = 0, F(+\infty) = 1$$

$$F(x) \leq F(y) \quad \text{when } x \leq y$$

The *probability density function* is

$$p(x) = \frac{\mathrm{d}F(x)}{\mathrm{d}x}$$

where

$$\mathrm{d}F = p\mathrm{d}x$$

is the probability of x being between x and x + dx. The probability density function satisfies

$$p(x) \geq 0$$

$$\int_{-\infty}^{\infty} p(x)\,\mathrm{d}x = 1$$

The *Bernoulli distribution* applies when the outcome can take only two values, such as heads or tails, or 0 or 1. The probability distribution function is

$$p(x = k) = p^k (1-p)^{1-k}, \; k = 0 \text{ or } 1$$

and the mean of a function $g(x)$ depending on x is

$$E[g(x)] = g(1)\,p + g(0)\,(1-p)$$

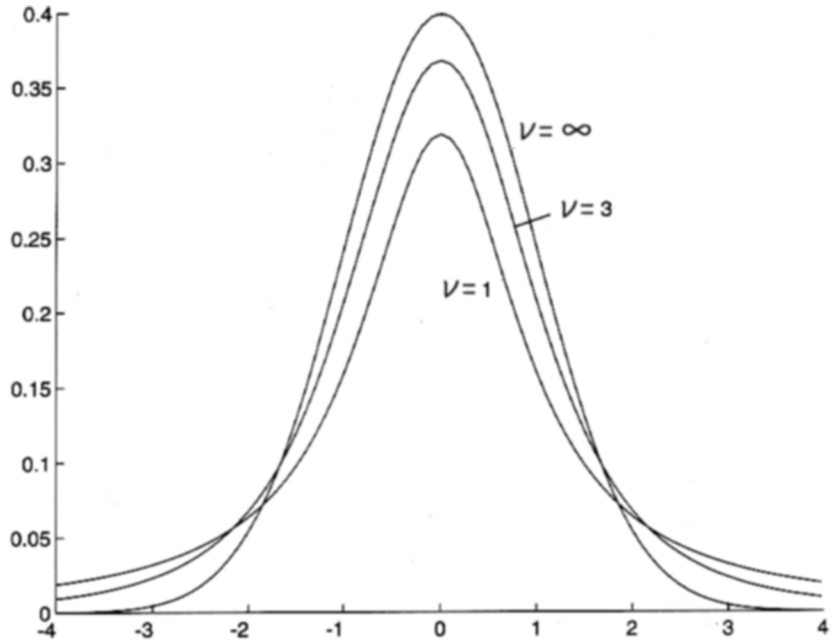

Figure 51. Student's *t*-distribution.
For explanation of ν see text

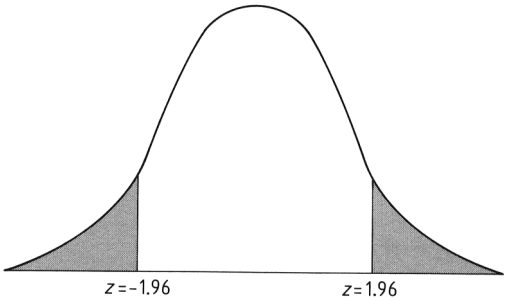

Figure 52. Percentage points of area under Student's *t*-distribution

The *binomial distribution function* applies when there are *n* trials of a Bernoulli event; it gives the probability of *k* occurrences of the event, which occurs with probability *p* on each trial

$$p\,(x = k) = \frac{n!}{k\,(n-k)\,!} p^k (1-p)^{n-k}$$

The mean and variance are

$$E\,(x) = np$$

$$var\,(x) = np\,(1-p)$$

The *hypergeometric distribution function* applies when there are *N* objects, of which *M* are of one kind and $N - M$ are of another kind. Then the objects are drawn one by one, without replacing the last draw. If the last draw had been replaced the distribution would be the binomial distribution. If *x* is the number of objects of type *M* drawn in a sample of size *n*, then the probability of $x = k$ is

$$p\,(x = k) =$$

$$\frac{M!(N-M)!n!(N-n)!}{k!(M-k)!(n-k)!(N-M-n+k)!N!}$$

The mean and variance are

$$E\,(x) = \frac{nM}{N}$$

$$var\,(x) = np\,(1-p)\,\frac{N-n}{N-1}$$

The *Poisson distribution* is

$$p\,(x = k) = \mathrm{e}^{-\lambda} \frac{\lambda^k}{k!}$$

with a parameter λ. The mean and variance are

$$E\,(x) = \lambda$$

$$var\,(x) = \lambda$$

The simplest continuous distribution is the uniform distribution. The probability density function is

$$p = \begin{cases} \frac{1}{b-a} & a<x<b \\ 0 & x<a,\ x>b \end{cases}$$

and the probability distribution function is

$$F(x) = \begin{cases} 0 & x<a \\ \frac{x-a}{b-a} & a<x<b \\ 1 & b<x \end{cases}$$

The mean and variance are

$$E(x) = \frac{a+b}{2}$$

$$var(x) = \frac{(b-a)^2}{12}$$

The *normal distribution* is given by Equation 120 with variance σ^2.

The *log normal probability density* function is

$$p(x) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left[-\frac{(\log x-\mu)^2}{2\sigma^2}\right]$$

and the mean and variance are [305, p. 89]

$$E(x) = \exp\left(\mu+\frac{\sigma^2}{2}\right)$$

$$var(x) = \exp\left(\sigma^2-1\right)\exp\left(2\mu+\sigma^2\right)$$

11.2. Sampling and Statistical Decisions

Two variables can be statistically dependent or independent. For example, the height and diameter of all distillation towers are statistically independent, because the distribution of diameters of all columns 10 m high is different from that of columns 30 m high. If y_B is the diameter and y_A the height, the distribution is written as

$$p(y_B|y_A = \text{constant}), \text{ or here}$$

$$p(y_B|y_A = 10) \neq p(y_B|y_A = 30)$$

A third variable y_C, could be the age of the operator on the third shift. This variable is probably unrelated to the diameter of the column, and for the distribution of ages is

$$p(y_C|y_A) = p(y_C)$$

Thus, variables y_A and y_C are distributed independently. The joint distribution for two variables is

$$p(y_A,y_B) = p(y_A)\,p(y_B|y_A)$$

if they are statistically dependent, and

$$p(y_A,y_B) = p(y_A)\,p(y_B)$$

if they are statistically independent. If a set of variables y_A, y_B, . . . is independent and identically distributed,

$$p(y_A,y_B,\ldots) = p(y_A)\,p(y_B)\ldots$$

Conditional probabilities are used in hazard analysis of chemical plants.

A measure of the linear dependence between variables is given by the *covariance*

$$Cov(y_A,y_B) = E\{[y_A-E(y_A)]\,[(y_B-E(y_B))]\}$$

$$= \frac{\sum_{i=1}^{N}[y_{Ai}-E(y_A)][y_{Bi}-E(y_B)]}{N}$$

The correlation coefficient ϱ is

$$\varrho(y_A,y_B) = \frac{Cov(y_A,y_B)}{\sigma_A\sigma_B}$$

If y_A and y_B are independent, then *Cov* (y_A, y_B) = 0. If y_A tends to increase when y_B decreases then *Cov* (y_A, y_B) < 0. The sample correlation coefficient is [15, p. 484]

$$r(y_A,y_B) = \frac{\sum_{i=1}^{n}(y_{Ai}-\overline{y}_A)(y_{Bi}-\overline{y}_B)}{(n-1)\,s_A s_B}$$

If measurements are for independent, identically distributed observations, the errors are independent and uncorrelated. Then $\overline{y}$ varies about E (y) with variance σ^2/n, where n is the number of observations in $\overline{y}$. Thus if something is measured several times today and every day, and the measurements have the same distribution, the variance of the means decreases with the number of samples in each day's measurement n. Of course, other factors (weather, weekends) may cause the observations on different days to be distributed nonidentically.

Suppose Y, which is the sum or difference of two variables, is of interest:

$$Y = y_A \pm y_B$$

Then the mean value of Y is

$$E(Y) = E(y_A) \pm E(y_B)$$

and the variance of Y is

$$\sigma^2(Y) = \sigma^2(y_A) + \sigma^2(y_B)$$

More generally, consider the random variables $y_1, y_2, \ldots$ with means $E(y_1), E(y_2), \ldots$ and variances $\sigma^2(y_1), \sigma^2(y_2), \ldots$ and correlation coefficients ϱ_{ij}. The variable

$$Y = \alpha_1 y_1 + \alpha_2 y_2 + \ldots$$

has a mean

$$E(Y) = \alpha_1 E(y_1) + \alpha_2 E(y_2) + \ldots$$

and variance [303, p. 87]

$$\sigma^2(Y) = \sum_{i=1}^{n} \alpha_i^2 \sigma^2(y_i) + 2 \sum_{i=1}^{n} \sum_{j=i+1}^{n} \alpha_i \alpha_j \sigma(y_i)\, \sigma(y_j)\, \varrho_{ij}$$

or

$$\sigma^2(Y) = \sum_{i=1}^{n} \alpha_i^2 \sigma^2(y_i) + 2 \sum_{i=1}^{n} \sum_{j=i+1}^{n} \alpha_i \alpha_j Cov(y_i, y_j) \tag{120}$$

If the variables are uncorrelated and have the same variance, then

$$\sigma^2(Y) = \left(\sum_{i=1}^{n} \alpha_i^2 \right) \sigma^2$$

This fact can be used to obtain more accurate cost estimates for the purchased cost of a chemical plant than is true for any one piece of equipment. Suppose the plant is composed of a number of heat exchangers, pumps, towers, etc., and that the cost estimate of each device is $\pm$ 40 % of its cost (the sample standard deviation is 20 % of its cost). In this case the α_i are the numbers of each type of unit. Under special conditions, such as equal numbers of all types of units and comparable cost, the standard deviation of the plant costs is

$$\sigma(Y) = \frac{\sigma}{\sqrt{n}}$$

and is then $\pm (40/\sqrt{n})$ %. Thus the standard deviation of the cost for the entire plant is the standard deviation of each piece of equipment divided by the square root of the number of units. Under less restrictive conditions the actual numbers change according to the above equations, but the principle is the same.

Suppose modifications are introduced into the manufacturing process. To determine if the modification causes a significant change, the mean of some property could be measured before and after the change; if these differ, does it mean the process modification caused it, or could the change have happened by chance? This is a statistical decision. A hypothesis H_0 is defined; if it is true, action *A* must be taken. The reverse hypothesis is H_1; if this is true, action *B* must be taken. A correct decision is made if action *A* is taken when H_0 is true or action *B* is taken when H_1 is true. Taking action *B* when H_0 is true is called a type I error, whereas taking action *A* when H_1 is true is called a type II error.

The following test of hypothesis or test of significance must be defined to determine if the hypothesis is true. The level of significance is the maximum probability that an error would be accepted in the decision (i.e., rejecting the hypothesis when it is actually true). Common levels of significance are 0.05 and 0.01, and the test of significance can be either one or two sided. If a sampled distribution is normal, then the probability that the z score

$$z = \frac{y - \bar{y}}{s_y}$$

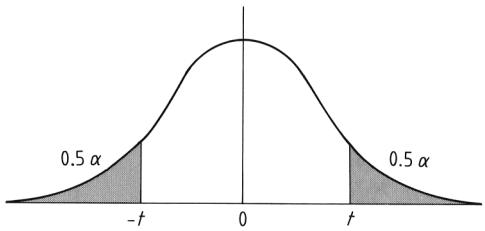

Figure 53. Two-sided statistical decision

is in the unshaded region is 0.95. Because a two-sided test is desired, $F = 0.95/2 = 0.475$. The value given in Table 12 for $F = 0.475$ is $z = 1.96$. If the test was one-sided, at the 5 % level of significance, 0.95 = 0.5 (for negative z) + F (for positive z). Thus, $F = 0.45$ or $z = 1.645$. In the two-sided test (see Fig. 53), if a single sample is chosen and $z < -1.96$ or $z > 1.96$, then this could happen with probability 0.05 if the hypothesis were true. This z would be significantly different from the expected value (based on the chosen level of significance) and the tendency would be to reject the hypothesis. If the value of z was between -1.96 and 1.96, the hypothesis would be accepted.

The same type of decisions can be made for other distributions. Consider Student's *t*-distri-

bution. At a 95 % level of confidence, with ν = 10 degrees of freedom, the t values are $\pm$ 2.228. Thus, the sample mean would be expected to be between

$$\overline{y} \pm t_c \frac{s}{\sqrt{n}}$$

with 95 % confidence. If the mean were outside this interval, the hypothesis would be rejected.

The *chi-square distribution* is useful for examining the variance or standard deviation. The statistic is defined as

$$\chi^2 = \frac{ns^2}{\sigma^2}$$

$$= \frac{(y_1-\overline{y})^2+(y_2-\overline{y})^2+\ldots+(y_n-\overline{y})^2}{\sigma^2}$$

and the chi-square distribution is

$$p(y) = y_0 \chi^{\nu-2} \mathrm{e}^{-\chi^2/2}$$

$\nu = n - 1$ is the number of degrees of freedom and y_0 is chosen so that the integral of p (y) over all y is 1. The probability of a deviation larger than χ^2 is given in Table 14; the area in question, in Figure 54. For example, for 10 degrees of freedom and a 95 % confidence level, the critical values of χ^2 are 0.025 and 0.975. Then

$$\frac{s\sqrt{n}}{\chi_{0.975}} < \sigma < \frac{s\sqrt{n}}{\chi_{0.025}}$$

or

$$\frac{s\sqrt{n}}{20.5} < \sigma < \frac{s\sqrt{n}}{3.25}$$

with 95 % confidence.

Tests are available to decide if two distributions that have the same variance have different means [15, p. 465]. Let one distribution be called x, with N_1 samples, and the other be called y, with N_2 samples. First, compute the standard error of the difference of the means:

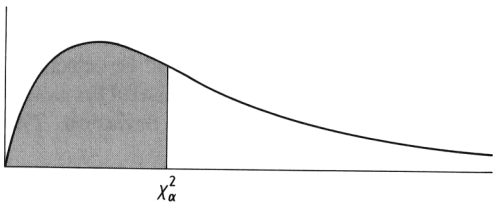

Figure 54. Percentage points of area under chi-squared distribution with ν degrees of freedom

$$s_D = \sqrt{\frac{\sum_{i=1}^{N_1}(x_i-\overline{x})^2+\sum_{i=1}^{N_2}(y_i-\overline{y})^2}{N_1+N_2-2}\left(\frac{1}{N_1}+\frac{1}{N_2}\right)}$$

Next, compute the value of t

$$t = \frac{\overline{x}-\overline{y}}{s_D}$$

and evaluate the significance of t using Student's t-distribution for $N_1 + N_2 - 2$ degrees of freedom.

If the samples have different variances, the relevant statistic for the t-test is

$$t = \frac{\overline{x}-\overline{y}}{\sqrt{var\,(x)\,/N_1+var\,(y)\,/N_2}}$$

The number of degrees of freedom is now taken approximately as

$$\nu = \frac{\left(\frac{var(x)}{N_1}+\frac{var(y)}{N_2}\right)^2}{\frac{[var(x)/N_1]^2}{N_1-1}+\frac{[var(y)/N_2]^2}{N_2-1}}$$

There is also an F-test to decide if two distributions have significantly different variances. In this case, the ratio of variances is calculated:

$$F = \frac{var\,(x)}{var\,(y)}$$

where the variance of x is assumed to be larger. Then, a table of values is used to determine the significance of the ratio. The table [23, Table 26.9] is derived from the formula [15, p. 169]

$$Q\,(F|\nu_1,\nu_2) = I_{\nu_2/(\nu_2+\nu_1 F)}\left(\frac{\nu_2}{2},\frac{\nu_1}{2}\right)$$

where the right-hand side is an incomplete beta function. The F table is given by the Microsoft Excel function FINV(fraction, ν_x, ν_y), where fraction is the fractional value (≤ 1) representing the upper percentage and ν_x and ν_y are the degrees of freedom of the numerator and denominator, respectively.

Example. For two sample variances with 8 degrees of freedom each, what limits will bracket their ratio with a midarea probability of 90 %? FINV(0.95,8,8) = 3.44. The 0.95 is used to get both sides to toal 10 %. Then

$$P\,[1/3.44 \leq \mathrm{var}\,(x)\,/\mathrm{var}\,(y) \leq 3.44] = 0.90.$$

Table 14. Percentage points of area under chi-square distribution with ν degrees of freedom *

ν	α=0.995	α=0.99	α=0.975	α=0.95	α=0.5	α=0.05	α=0.025	α=0.01	α=0.005
1	7.88	6.63	5.02	3.84	0.455	0.0039	0.0010	0.0002	0.00004
2	10.6	9.21	7.38	5.99	1.39	0.103	0.0506	0.0201	0.0100
3	12.8	11.3	9.35	7.81	2.37	0.352	0.216	0.115	0.072
4	14.9	13.3	11.1	9.49	3.36	0.711	0.484	0.297	0.207
5	16.7	15.1	12.8	11.1	4.35	1.15	0.831	0.554	0.412
6	18.5	16.8	14.4	12.6	5.35	1.64	1.24	0.872	0.676
7	20.3	18.5	16.0	14.1	6.35	2.17	1.69	1.24	0.989
8	22.0	20.1	17.5	15.5	7.34	2.73	2.18	1.65	1.34
9	23.6	21.7	19.0	16.9	8.34	3.33	2.70	2.09	1.73
10	25.2	23.2	20.5	18.3	9.34	3.94	3.25	2.56	2.16
12	28.3	26.2	23.3	21.0	11.3	5.23	4.40	3.57	3.07
15	32.8	30.6	27.5	25.0	14.3	7.26	6.26	5.23	4.60
17	35.7	33.4	30.2	27.6	16.3	8.67	7.56	6.41	5.70
20	40.0	37.6	34.2	31.4	19.3	10.9	9.59	8.26	7.43
25	46.9	44.3	40.6	37.7	24.3	14.6	13.1	11.5	10.5
30	53.7	50.9	47.0	43.8	29.3	18.5	16.8	15.0	13.8
40	66.8	63.7	59.3	55.8	39.3	26.5	24.4	22.2	20.7
50	79.5	76.2	71.4	67.5	49.3	34.8	32.4	29.7	28.0
60	92.0	88.4	83.3	79.1	59.3	43.2	40.5	37.5	35.5
70	104.2	100.4	95.0	90.5	69.3	51.7	48.8	45.4	43.3
80	116.3	112.3	106.6	101.9	79.3	60.4	57.2	53.5	51.2
90	128.3	124.1	118.1	113.1	89.3	69.1	65.6	61.8	59.2
100	140.2	135.8	129.6	124.3	99.3	77.9	74.2	70.1	67.3

* Table value is χ^2_α; $\chi^2 < \chi^2_\alpha$ with probability α. For a more complete table (in slightly different form), see [23, Table 26.8]. The Microsoft Excel function CHIINV(1-α,ν) gives the table value.

11.3. Error Analysis in Experiments

Suppose a measurement of several quantities is made and a formula or mathematical model is used to deduce some property of interest. For example, to measure the thermal conductivity of a solid k, the heat flux q, the thickness of the sample d, and the temperature difference across the sample ΔT must be measured. Each measurement has some error. The heat flux q may be the rate of electrical heat input $\dot{Q}$ divided by the area A, and both quantities are measured to some tolerance. The thickness of the sample is measured with some accuracy, and the temperatures are probably measured with a thermocouple, to some accuracy. These measurements are combined, however, to obtain the thermal conductivity, and the error in the thermal conductivity must be determined. The formula is

$$k = \frac{d}{A\Delta T}\,\dot{Q}$$

If each measured quantity has some variance, what is the variance in the thermal conductivity?

Suppose a model for Y depends on various measurable quantities, $y_1, y_2, \ldots$ Suppose several measurements are made of $y_1, y_2, \ldots$ under seemingly identical conditions and several different values are obtained, with means $E(y_1)$, $E(y_2)$, ... and variances σ_1^2, σ_2^2, ... Next suppose the errors are small and independent of one another. Then a change in Y is related to changes in y_i by

$$\mathrm{d}Y = \frac{\partial Y}{\partial y_1}\mathrm{d}y_1 + \frac{\partial Y}{\partial y_2}\mathrm{d}y_2 + \ldots$$

If the changes are indeed small, the partial derivatives are constant among all the samples. Then the expected value of the change is

$$E(\mathrm{d}Y) = \sum_{i=1}^{N}\left(\frac{\partial Y}{\partial y_i}\right) E(\mathrm{d}y_i)$$

Naturally $E(\mathrm{d}y_i) = 0$ by definition so that $E(\mathrm{d}Y) = 0$, too. However, since the errors are independent of each other and the partial derivatives are assumed constant because the errors are small, the variances are given by Equation 121 [296, p. 550]

$$\sigma^2(\mathrm{d}Y) = \sum_{i=1}^{N}\left(\frac{\partial Y}{\partial y_i}\right)^2 \sigma_i^2 \qquad (121)$$

Thus, the variance of the desired quantity Y can be found. This gives an independent estimate of the errors in measuring the quantity Y from the errors in measuring each variable it depends upon.

11.4. Factorial Design of Experiments and Analysis of Variance

Statistically designed experiments consider, of course, the effect of primary variables, but they also consider the effect of extraneous variables, the interactions among variables, and a measure of the random error. Primary variables are those whose effect must be determined. These variables can be quantitative or qualitative. Quantitative variables are ones that may be fit to a model to determine the model parameters. Curve fitting of this type is discused in Chapter 2. Qualitative variables are ones whose effect needs to be known; no attempt is made to quantify that effect other than to assign possible errors or magnitudes. Qualitative variables can be further subdivided into type I variables, whose effect is determined directly, and type II variables, which contribute to performance variability, and whose effect is averaged out. For example, in studying the effect of several catalysts on yield in a chemical reactor, each different type of catalyst would be a type I variable, because its effect should be known. However, each time the catalyst is prepared, the results are slightly different, because of random variations; thus, several batches may exist of what purports to be the same catalyst. The variability between batches is a type II variable. Because the ultimate use will require using different batches, the overall effect including that variation should be known, because knowing the results from one batch of one catalyst precisely might not be representative of the results obtained from all batches of the same catalyst. A randomized block design, incomplete block design, or Latin square design, for example, all keep the effect of experimental error in the blocked variables from influencing the effect of the primary variables. Other uncontrolled variables are accounted for by introducing randomization in parts of the experimental design. To study all variables and their interaction requires a factorial design, involving all possible combinations of each variable, or a fractional factorial design, involving only a selected set. Statistical techniques are then used to determine the important variables, the important interactions and the error in estimating these effects. The discussion here is a brief overview of [303].

If only two methods exist for preparing some product, to see which treatment is best, the sampling analysis discussed in Section 11.2 can be used to deduce if the means of the two treatments differ significantly. With more treatments, the analysis is more detailed. Suppose the experimental results are arranged as shown in Table 15, i.e., several measurements for each treatment. The objective is to see if the treatments differ significantly from each other, that is, whether their means are different. The samples are assumed to have the same variance. The hypothesis is that the treatments are all the same, and the null hypothesis is that they are different. Deducing the statistical validity of the hypothesis is done by an analysis of variance.

Table 15. Estimating the effect of four treatments

Treatment	1	2	3	4
	–	–	–	–
	–	–	–	–
	–	–	–	–
		–	–	–
			–	–
				–
Treatment average, $\overline{y}_t$	–	–	–	–
Grand average, $\overline{y}$		–		

The data for $k = 4$ treatments are arranged in Table 15. Each treatment has n_t experiments, and the outcome of the i-th experiment with treatment t is called y_{ti}. The treatment average is

$$\overline{y}_t = \frac{\sum_{i=1}^{n_t} y_{ti}}{n_t}$$

and the grand average is

$$\overline{y} = \frac{\sum_{t=1}^{k} n_t \overline{y}_t}{N}, \; N = \sum_{t=1}^{k} n_t$$

Next, the sum of squares of deviations is computed from the average within the t-th treatment

$$S_t = \sum_{i=1}^{n_t} (y_{ti} - \overline{y}_t)^2$$

Since each treatment has n_t experiments, the number of degrees of freedom is $n_t - 1$. Then the sample variances are

$$s_t^2 = \frac{S_t}{n_t - 1}$$

The within-treatment sum of squares is

$$S_R = \sum_{t=1}^{k} S_t$$

and the within-treatment sample variance is

$$s_R^2 = \frac{S_R}{N-k}$$

Now, if no difference exists between treatments, a second estimate of σ^2 could be obtained by calculating the variation of the treatment averages about the grand average. Thus, the between-treatment mean square is computed:

$$s_T^2 = \frac{S_T}{k-1},\ S_T = \sum_{t=1}^{k} n_t(\overline{y}_t - \overline{y})^2$$

Basically the test for whether the hypothesis is true or not hinges on a comparison between the within-treatment estimate s_R^2 (with $\nu_R = N - k$ degrees of freedom) and the between-treatment estimate s_T^2 (with $\nu_T = k - 1$ degrees of freedom). The test is made based on the F distribution for ν_R and ν_T degrees of freedom [23, Table 26.9], [303, p. 636].

Table 16. Block design with four treatments and five blocks

Treatment	1	2	3	4	Block average
Block 1	–	–	–	–	–
Block 2	–	–	–	–	–
Block 3	–	–	–	–	–
Block 4	–	–	–	–	–
Block 5	–	–	–	–	–
Treatment average	–	–	–	–	grand average

Next consider the case in which *randomized blocking* is used to eliminate the effect of some variable whose effect is of no interest, such as the batch-to-batch variation of the catalysts in the chemical reactor example. With k treatments and n experiments in each treatment, the results from $n\ k$ experiments can be arranged as shown in Table 16; within each block, various treatments are applied in a random order. The block average, the treatment average, and the grand average are computed as before. The following quantities are also computed for the analysis of variance table:

Name	Formula	Degrees of freedom
Average	$S_A = nk\overline{y}^2$	1
Blocks	$S_B = k\sum_{i=1}^{n}(\overline{y}_t - \overline{y})^2$	n-1
Treatments	$S_T = n\sum_{t=1}^{k}(\overline{y}_t - \overline{y})^2$	k-1
Residuals	$S_R = \sum_{t=1}^{k}\sum_{i=1}^{n}(y_{ti} - \overline{y}_i - \overline{y}_t + \overline{y})^2$	(n-1)(k-1)
Total	$S = \sum_{t=1}^{k}\sum_{i=1}^{n} y_{ti}^2$	$N = n\ k$

The key test is again a statistical one, based on the value of

s_T^2/s_R^2, where $s_T^2 = \frac{S_T}{k-1}$

and $s_R^2 = \frac{S_R}{(n-1)(k-1)}$

and the F distribution for ν_R and ν_T degrees of freedom [303, p. 636]. The assumption behind the analysis is that the variations are linear [303, p. 218]. Ways to test this assumption as well as transformations to make if it is not true are provided in [303], where an example is given of how the observations are broken down into a grand average, a block deviation, a treatment deviation, and a residual. For two-way factorial design, in which the second variable is a real one rather than one you would like to block out, see [303, p. 228].

Table 17. Two-level factorial design with three variables

Run	Variable 1	Variable 2	Variable 3
1	–	–	–
2	+	–	–
3	–	+	–
4	+	+	–
5	–	–	+
6	+	–	+
7	–	+	+
8	+	+	+

To measure the effects of variables on a single outcome, a *factorial design* is appropriate. In a two-level factorial design, each variable is considered at two levels only, a high and low value, often designated as a + and a −. The two-level factorial design is useful for indicating trends and showing interactions; it is also the basis for a fractional factorial design. As an example, consider a 2^3 factorial design, with 3 variables and 2 levels for each. The experiments are indicated in Table 17. The main effects are calculated by determining the difference between results from

all high values of a variable and all low values of a variable; the result is divided by the number of experiments at each level. For example, for the first variable, calculate

$$\text{Effect of variable 1} = [(y_2+y_4+y_6+y_8) - [(y_1+y_3+y_5+y_7)]]/4$$

Note that all observations are being used to supply information on each of the main effects and each effect is determined with the precision of a fourfold replicated difference. The advantage of a one-at-a-time experiment is the gain in precision if the variables are additive and the measure of nonadditivity if it occurs [303, p. 313].

Interaction effects between variables 1 and 2 are obtained by comparing the difference between the results obtained with the high and low value of 1 at the low value of 2 with the difference between the results obtained with the high and low value 1 at the high value of 2. The 12-interaction is

$$12\,\text{interaction} = [(y_4-y_3+y_8-y_7) - [(y_2-y_1+y_6-y_5)]]/2$$

The key step is to determine the errors associated with the effect of each variable and each interaction so that the significance can be determined. Thus, standard errors need to be assigned. This can be done by repeating the experiments, but it can also be done by using higher order interactions (such as 123 interactions in a 2^4 factorial design). These are assumed negligible in their effect on the mean but can be used to estimate the standard error [303, pp. 319 – 328]. Then calculated effects that are large compared to the standard error are considered important, whereas those that are small compared to the standard error are considered due to random variations and are unimportant.

In a fractional factorial design, only part of the possible experiments is performed. With k variables, a factorial design requires 2^k experiments. When k is large, the number of experiments can be large; for $k = 5$, $2^5 = 32$. For k this large, Box et al. [296, p. 235] do a fractional factorial design. In the fractional factorial design with $k = 5$, only 8 experiments are chosen. Cropley [298] gives an example of how to combine heuristics and statistical arguments in application to kinetics mechanisms in chemical engineering.

12. Multivariable Calculus Applied to Thermodynamics

Many of the functional relationships required in thermodynamics are direct applications of the rules of multivariable calculus. In this short chapter, those rules are reviewed in the context of the needs of thermodynamics. These ideas were expounded in one of the classic books on chemical engineering thermodynamics [299].

12.1. State Functions

State functions depend only on the state of the system, not on its past history or how one got there. If z is a function of two variables x and y, then $z\,(x, y)$ is a state function, because z is known once x and y are specified. The differential of z is

$$\mathrm{d}z = M\mathrm{d}x+N\mathrm{d}y$$

The line integral

$$\int_C (M\mathrm{d}x+N\mathrm{d}y)$$

is independent of the path in $x – y$ space if and only if

$$\frac{\partial M}{\partial y} = \frac{\partial N}{\partial x} \tag{122}$$

Because the total differential can be written as

$$\mathrm{d}z = \left(\frac{\partial z}{\partial x}\right)_y \mathrm{d}x+\left(\frac{\partial z}{\partial y}\right)_x \mathrm{d}y \tag{123}$$

for path independence

$$\frac{\partial}{\partial y}\left(\frac{\partial z}{\partial x}\right)_y = \frac{\partial}{\partial x}\left(\frac{\partial z}{\partial y}\right)_x$$

or

$$\frac{\partial^2 z}{\partial y \partial x} = \frac{\partial^2 z}{\partial x \partial y} \tag{124}$$

is needed.

Various relationships can be derived from Equation 123. If z is constant,

$$\left[0 = \left(\frac{\partial z}{\partial x}\right)_y \mathrm{d}x+\left(\frac{\partial z}{\partial y}\right)_x \mathrm{d}y\right]_z$$

Rearrangement gives

$$\left(\frac{\partial z}{\partial x}\right)_y = -\left(\frac{\partial y}{\partial x}\right)_z \left(\frac{\partial z}{\partial y}\right)_x = -\frac{(\partial y/\partial x)_z}{(\partial y/\partial z)_x} \quad (125)$$

Alternatively, if Equation 123 is divided by dy while some other variable w is held constant,

$$\left(\frac{\partial z}{\partial y}\right)_w = \left(\frac{\partial z}{\partial x}\right)_y \left(\frac{\partial x}{\partial y}\right)_w + \left(\frac{\partial z}{\partial y}\right)_x \quad (126)$$

Dividing both the numerator and the denominator of a partial derivative by dw while holding a variable y constant yields

$$\left(\frac{\partial z}{\partial x}\right)_y = \frac{(\partial z/\partial w)_y}{(\partial x/\partial w)_y} = \left(\frac{\partial z}{\partial w}\right)_y \left(\frac{\partial w}{\partial x}\right)_y \quad (127)$$

In thermodynamics the state functions include the internal energy U, the enthalpy H, and the Helmholtz and Gibbs free energies A and G, respectively, which are defined as follows:

$$H = U + pV$$

$$A = U - TS$$

$$G = H - TS = U + pV - TS = A + pV$$

where S is the entropy, T the absolute temperature, p the pressure, and V the volume. These are also state functions, in that the entropy is specified once two variables (e.g., T and p) are specified. Likewise V is specified once T and p are specified, and so forth.

12.2. Applications to Thermodynamics

All of the following applications are for closed systems with constant mass. If a process is reversible and only $p - V$ work is done, one form of the first law states that changes in the internal energy are given by the following expression

$$dU = T\mathrm{d}S - p\mathrm{d}V \quad (128)$$

If the internal energy is considered a function of S and V, then

$$\mathrm{d}U = \left(\frac{\partial U}{\partial S}\right)_V \mathrm{d}S + \left(\frac{\partial U}{\partial V}\right)_S \mathrm{d}V$$

This is the equivalent of Equation 123 and

$$T = \left(\frac{\partial U}{\partial S}\right)_V,\ p = -\left(\frac{\partial U}{\partial V}\right)_S$$

Because the internal energy is a state function, Equation 124 is required:

$$\frac{\partial^2 U}{\partial V \partial S} = \frac{\partial^2 U}{\partial S \partial V}$$

which here is

$$\left(\frac{\partial T}{\partial V}\right)_S = -\left(\frac{\partial p}{\partial S}\right)_V \quad (129)$$

This is one of the Maxwell relations and is merely an expression of Equation 124.

The differentials of the other energies are

$$dH = T\,\mathrm{d}S + V\,\mathrm{d}p \quad (130)$$

$$dA = -S\mathrm{d}T - p\mathrm{d}V \quad (131)$$

$$dG = -S\,\mathrm{d}T + V\,\mathrm{d}p \quad (132)$$

From these differentials, other Maxwell relations can be derived in a similar fashion by applying Equation 124.

$$\left(\frac{\partial T}{\partial p}\right)_S = \left(\frac{\partial V}{\partial S}\right)_p \quad (133)$$

$$\left(\frac{\partial S}{\partial V}\right)_T = \left(\frac{\partial p}{\partial T}\right)_V \quad (134)$$

$$\left(\frac{\partial S}{\partial p}\right)_T = -\left(\frac{\partial V}{\partial T}\right)_p \quad (135)$$

The heat capacity at constant pressure is defined as

$$C_p = \left(\frac{\partial H}{\partial T}\right)_p$$

If entropy and enthalpy are taken as functions of T and p, the total differentials are

$$\mathrm{d}S = \left(\frac{\partial S}{\partial T}\right)_p \mathrm{d}T + \left(\frac{\partial S}{\partial p}\right)_T \mathrm{d}p$$

$$\mathrm{d}H = \left(\frac{\partial H}{\partial T}\right)_p \mathrm{d}T + \left(\frac{\partial H}{\partial p}\right)_T \mathrm{d}p$$

$$= C_p \mathrm{d}T + \left(\frac{\partial H}{\partial p}\right)_T \mathrm{d}p$$

If the pressure is constant,

$$\mathrm{d}S = \left(\frac{\partial S}{\partial T}\right)_p \mathrm{d}T \text{ and } \mathrm{d}H = C_p \mathrm{d}T$$

When enthalpy is considered a function of S and p, the total differential is

$$\mathrm{d}H = T\,\mathrm{d}S + V\,\mathrm{d}p$$

When the pressure is constant, this is

$$\mathrm{d}H = T\mathrm{d}S$$

Thus, at constant pressure

$$\mathrm{d}H = C_p \mathrm{d}T = T\mathrm{d}S = T\left(\frac{\partial S}{\partial T}\right)_p \mathrm{d}T$$

which gives

$$\left(\frac{\partial S}{\partial T}\right)_p = \frac{C_p}{T}$$

When p is not constant, using the last Maxwell relation gives

$$\mathrm{d}S = \frac{C_p}{T}\mathrm{d}T - \left(\frac{\partial V}{\partial T}\right)_p \mathrm{d}p \tag{136}$$

Then the total differential for H is

$$\mathrm{d}H = T\mathrm{d}S + V\mathrm{d}p = C_p\mathrm{d}T - T\left(\frac{\partial V}{\partial T}\right)_p \mathrm{d}p + V\mathrm{d}p$$

Rearranging this, when H (T, p), yields

$$\mathrm{d}H = C_p\mathrm{d}T + \left[V - T\left(\frac{\partial V}{\partial T}\right)_p\right]\mathrm{d}p \tag{137}$$

This equation can be used to evaluate enthalpy differences by using information on the equation of state and the heat capacity:

$$H(T_2,p_2) - H(T_1,p_1) = \int_{T_1}^{T_2} C_p(T,p_1)\,\mathrm{d}T + \int_{p_1}^{p_2}\left[V - T\left(\frac{\partial V}{\partial T}\right)_p\right]|_{T_2,p}\mathrm{d}p \tag{138}$$

The same manipulations can be done for internal energy:

$$\left(\frac{\partial S}{\partial T}\right)_V = \frac{C_v}{T} \tag{139}$$

$$\mathrm{d}S = -\left[\frac{(\partial V/\partial T)_p}{(\partial V/\partial p)_T}\right]\mathrm{d}V + \frac{C_v}{T}\mathrm{d}T \tag{140}$$

$$\mathrm{d}U = C_v\mathrm{d}T - \left[p + T\frac{(\partial V/\partial T)_p}{(\partial V/\partial p)_T}\right]\mathrm{d}V$$

12.3. Partial Derivatives of All Thermodynamic Functions

The various partial derivatives of the thermodynamic functions can be classified into six groups. In the general formulas below, the variables *U, H, A, G*, or *S* are denoted by Greek letters, whereas the variables *V, T*, or *p* are denoted by Latin letters.

Type 1 (3 possibilities plus reciprocals).

General: $\left(\frac{\partial a}{\partial b}\right)_c$, Specific: $\left(\frac{\partial p}{\partial T}\right)_V$

Equation 125 yields

$$\left(\frac{\partial p}{\partial T}\right)_V = -\left(\frac{\partial V}{\partial T}\right)_p\left(\frac{\partial p}{\partial V}\right)_T = -\frac{(\partial V/\partial T)_p}{(\partial V/\partial p)_T} \tag{141}$$

This relates all three partial derivatives of this type.

Type 2 (30 possibilities plus reciprocals).

General: $\left(\frac{\partial \alpha}{\partial b}\right)_c$, Specific: $\left(\frac{\partial G}{\partial T}\right)_V$

Using Equation 132 gives

$$\left(\frac{\partial G}{\partial T}\right)_V = -S + V\left(\frac{\partial p}{\partial T}\right)_V$$

Using the other equations for *U, H, A*, or *S* gives the other possibilities.

Type 3 (15 possibilities plus reciprocals).

General: $\left(\frac{\partial a}{\partial b}\right)_\alpha$, Specific: $\left(\frac{\partial V}{\partial T}\right)_S$

First the derivative is expanded by using Equation 125, which is called expansion without introducing a new variable:

$$\left(\frac{\partial V}{\partial T}\right)_S = -\left(\frac{\partial S}{\partial T}\right)_V\left(\frac{\partial V}{\partial S}\right)_T = -\frac{(\partial S/\partial T)_V}{(\partial S/\partial V)_T}$$

Then the numerator and denominator are evaluated as type 2 derivatives, or by using Equations (99) and (100):

$$\left(\frac{\partial V}{\partial T}\right)_S = -\frac{C_v/T}{-(\partial V/\partial T)_p(\partial p/\partial V)_T} = \frac{C_v}{T}\frac{\left(\frac{\partial V}{\partial p}\right)_T}{\left(\frac{\partial V}{\partial T}\right)_p} \tag{142}$$

These derivatives are important for reversible, adiabatic processes (e.g., in an ideal turbine or compressor) because the entropy is constant. Similar derivatives can be obtained for isenthalpic processes, such as a pressure reduction at a valve. In that case, the Joule – Thomson coefficient is obtained for constant H:

$$\left(\frac{\partial T}{\partial p}\right)_H = \frac{1}{C_p}\left[-V + T\left(\frac{\partial V}{\partial T}\right)_p\right]$$

Type 4 (30 possibilities plus reciprocals).

General: $\left(\frac{\partial\alpha}{\partial\beta}\right)_c$, Specific: $\left(\frac{\partial G}{\partial A}\right)_p$

Now, expand through the introduction of a new variable using Equation 127:

$$\left(\frac{\partial G}{\partial A}\right)_p = \left(\frac{\partial G}{\partial T}\right)_p \left(\frac{\partial T}{\partial A}\right)_p = \frac{(\partial G/\partial T)_p}{(\partial A/\partial T)_p}$$

This operation has created two type 2 derivatives. Substitution yields

$$\left(\frac{\partial G}{\partial A}\right)_p = \frac{S}{S+p(\partial V/\partial T)_p}$$

Type 5 (60 possibilities plus reciprocals).

General: $\left(\frac{\partial\alpha}{\partial b}\right)_\beta$, Specific: $\left(\frac{\partial G}{\partial p}\right)_A$

Starting from Equation 132 for dG gives

$$\left(\frac{\partial G}{\partial p}\right)_A = -S\left(\frac{\partial T}{\partial p}\right)_A + V$$

The derivative is a type 3 derivative and can be evaluated by using Equation 125.

$$\left(\frac{\partial G}{\partial p}\right)_A = S\frac{(\partial A/\partial p)_T}{(\partial A/\partial T)_p} + V$$

The two type 2 derivatives are then evaluated:

$$\left(\frac{\partial G}{\partial p}\right)_A = \frac{S_p(\partial V/\partial p)_T}{S+p(\partial V/\partial T)_p} + V$$

These derivatives are also of interest for free expansions or isentropic changes.

Type 6 (30 possibilities plus reciprocals).

General: $\left(\frac{\partial\alpha}{\partial\beta}\right)_\gamma$, Specific: $\left(\frac{\partial G}{\partial A}\right)_H$

Equation 127 is used to obtain two type 5 derivatives.

$$\left(\frac{\partial G}{\partial A}\right)_H = \frac{(\partial G/\partial T)_H}{(\partial A/\partial T)_H}$$

These can then be evaluated by using the procedures for type 5 derivatives.

The difference in molar heat capacities ($C_p - C_v$) can be derived in similar fashion. Using Equation 139 for C_v yields

$$C_v = T\left(\frac{\partial S}{\partial T}\right)_V$$

To evaluate the derivative, Equation 126 is used to express dS in terms of p and T:

$$\left(\frac{\partial S}{\partial T}\right)_V = -\left(\frac{\partial V}{\partial T}\right)_p \left(\frac{\partial p}{\partial T}\right)_V + \frac{C_p}{T}$$

Substitution for $(\partial p/\partial T)_V$ and rearrangement give

$$\begin{aligned} C_p - C_v &= T\left(\frac{\partial V}{\partial T}\right)_p \left(\frac{\partial p}{\partial T}\right)_V \\ &= -T\left(\frac{\partial V}{\partial T}\right)_p^2 \left(\frac{\partial p}{\partial V}\right)_T \end{aligned}$$

Use of this equation permits the rearrangement of Equation 142 into

$$\left(\frac{\partial V}{\partial T}\right)_S = \frac{(\partial V/\partial T)_p^2 + \frac{C_p}{T}(\partial V/\partial p)_T}{(\partial V/\partial T)_p}$$

The ratio of heat capacities is

$$\frac{C_p}{C_v} = \frac{T(\partial S/\partial T)_p}{T(\partial S/\partial T)_V}$$

Expansion by using Equation 125 gives

$$\frac{C_p}{C_v} = \frac{-(\partial p/\partial T)_S(\partial S/\partial p)_T}{-(\partial V/\partial T)_S(\partial S/\partial V)_T}$$

and the ratios are then

$$\frac{C_p}{C_v} = \left(\frac{\partial p}{\partial V}\right)_S \left(\frac{\partial V}{\partial p}\right)_T$$

Using Equation 125 gives

$$\frac{C_p}{C_v} = -\left(\frac{\partial p}{\partial V}\right)_S \left(\frac{\partial T}{\partial p}\right)_V \left(\frac{\partial V}{\partial T}\right)_p$$

Entropy is a variable in at least one of the partial derivatives.

13. References

Specific References

1. D. E. Seborg, T. F. Edgar, D. A. Mellichamp: *Process Dynamics and Control*, 2nd ed., John Wiley & Sons, New York 2004.
2. G. Forsyth, C. B. Moler: *Computer Solution of Lineart Algebraic Systems*, Prentice-Hall, Englewood Cliffs 1967.
3. B. A. Finlayson: *Nonlinear Analysis in Chemical Engineering*, McGraw-Hill, New York 1980; reprinted, Ravenna Park, Seattle 2003.

4. S. C. Eisenstat, M. H. Schultz, A. H. Sherman: "Algorithms and Data Structures for Sparse Symmetric Gaussian Elimination," *SIAM J. Sci. Stat. Comput.* **2** (1981) 225 – 237.
5. I. S. Duff: *Direct Methods for Sparse Matrices*, Charendon Press, Oxford 1986.
6. H. S. Price, K. H. Coats, "Direct Methods in Reservoir Simulation," *Soc. Pet. Eng. J.* **14** (1974) 295 – 308.
7. A. Bykat: "A Note on an Element Re-Ordering Scheme," *Int. J. Num. Methods Egn.* **11** (1977) 194 –198.
8. M. R. Hesteness, E. Stiefel: "Methods of conjugate gradients for solving linear systems," *J. Res. Nat. Bur. Stand* **29** (1952) 409 – 439.
9. Y. Saad: *Iterative Methods for Sparse Linear Systems*, 2nd ed., Soc. Ind. Appl. Math., Philadelphia 2003.
10. Y. Saad, M. Schultz: "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems." *SIAM J. Sci. Statist. Comput.* **7** (1986) 856–869.
11. http://mathworld.wolfram.com/GeneralizedMinimalResidualMethod.html.
12. http://www.netlib.org/linalg/html_templates/Templates.html.
13. http://software.sandia.gov/.
14. E. Isaacson, H. B. Keller, *Analysis of Numerical Methods*, J. Wiley and Sons, New York 1966.
15. W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling: *Numerical Recipes*, Cambridge University Press, Cambridge 1986.
16. R. W. H. Sargent: "A Review of Methods for Solving Non-linear Algebraic Equations," in R. S. H. Mah, W. D. Seider (eds.): *Foundations of Computer-Aided Chemical Process Design*, American Institute of Chemical Engineers, New York 1981.
17. J. D. Seader: "Computer Modeling of Chemical Processes," *AIChE Monogr. Ser.* **81** (1985) no. 15.
18. http://software.sandia.gov/trilinos/packages/nox/loca_user.html
19. N. R. Amundson: *Mathematical Methods in Chemical Engineering*, Prentice-Hall, Englewood Cliffs, N.J. 1966.
20. R. H. Perry, D. W Green: *Perry's Chemical Engineers' Handbook*, 7th ed., McGraw-Hill, New York 1997.
21. D. S. Watkins: "Understanding the QR Algorithm," *SIAM Rev.* **24** (1982) 427 – 440.
22. G. F. Carey, K. Sepehrnoori: "Gershgorin Theory for Stiffness and Stability of Evolution Systems and Convection-Diffusion," *Comp. Meth. Appl. Mech.* **22** (1980) 23 – 48.
23. M. Abranowitz, I. A. Stegun: *Handbook of Mathematical Functions*, National Bureau of Standards, Washington, D.C. 1972.
24. J. C. Daubisse: "Some Results about Approximation of Functions of One or Two Variables by Sums of Exponentials, " *Int. J. Num. Meth. Eng.* **23** (1986) 1959 – 1967.
25. O. C. McGehee: *An Introduction to Complex Analysis*, John Wiley & Sons, New York 2000.
26. H. A. Priestley: *Introduction to complex analysis*, Oxford University Press, New York 2003.
27. Y. K. Kwok: *Applied complex variables for scientists and engineers,* Cambridge University Press, New York 2002.
28. N. Asmar, G. C. Jones: *Applied complex analysis with partial differential equations*, Prentice Hall, Upper Saddle River, NJ, 2002.
29. M. J. Ablowitz, A. S. Fokas: *Complex variables: Introduction and applications*, Cambridge University Press, New York 2003.
30. J. W. Brown, R. V. Churchill: *Complex variables and applications*, 6th ed., McGraw-Hill, New York 1996; 7th ed. 2003.
31. W. Kaplan: *Advanced calculus*, 5th ed., Addison-Wesley, Redwood City, Calif., 2003.
32. E. Hille: *Analytic Function Theory*, Ginn and Co., Boston 1959.
33. R. V. Churchill: *Operational Mathematics*, McGraw-Hill, New York 1958.
34. J. W. Brown, R. V. Churchill: *Fourier Series and Boundary Value Problems*, 6th ed., McGraw-Hill, New York 2000.
35. R. V. Churchill: *Operational Mathematics*, 3rd ed., McGraw-Hill, New York 1972.
36. B. Davies: *Integral Transforms and Their Applications*, 3rd ed., Springer, Heidelberg 2002.
37. D. G. Duffy: *Transform Methods for Solving Partial Differential Equations*, Chapman & Hall/CRC, New York 2004.
38. A. Varma, M. Morbidelli: *Mathematical Methods in Chemical Engineering*, Oxford, New York 1997.
39. Bateman, H., *Tables of Integral Transforms, vol. I*, McGraw-Hill, New York 1954.
40. H. F. Weinberger: *A First Course in Partial Differential Equations*, Blaisdell, Waltham, Mass. 1965.
41. R. V. Churchill: *Operational Mathematics*, McGraw-Hill, New York 1958.

42. J. T. Hsu, J. S. Dranoff: "Numerical Inversion of Certain Laplace Transforms by the Direct Application of Fast Fourier Transform (FFT) Algorithm," *Comput. Chem. Eng.* **11** (1987) 101 – 110.
43. E. Kreyzig: *Advanced Engineering Mathematics*, 9th ed., John Wiley & Sons, New York 2006.
44. H. S. Carslaw, J. C. Jaeger: *Conduction of Heat in Solids*, 2nd ed., Clarendon Press, Oxford – London 1959.
45. R. B. Bird, R. C. Armstrong, O. Hassager: *Dynamics of Polymeric Liquids*, 2nd ed., Appendix A, Wiley-Interscience, New York 1987.
46. R. S. Rivlin, *J. Rat. Mech. Anal.* **4** (1955) 681 – 702.
47. N. R. Amundson: *Mathematical Methods in Chemical Engineering; Matrices and Their Application*, Prentice-Hall, Englewood Cliffs, N.J. 1966.
48. I. S. Sokolnikoff, E. S. Sokolnikoff: *Higher Mathematics for Engineers and Physicists*, McGraw-Hill, New York 1941.
49. R. C. Wrede, M. R. Spiegel: *Schaum's outline of theory and problems of advanced calculus*, 2nd ed, McGraw Hill, New York 2006.
50. P. M. Morse, H. Feshbach: *Methods of Theoretical Physics*, McGraw-Hill, New York 1953.
51. B. A. Finlayson: *The Method of Weighted Residuals and Variational Principles*, Academic Press, New York 1972.
52. G. Forsythe, M. Malcolm, C. Moler: *Computer Methods for Mathematical Computation*, Prentice-Hall, Englewood Cliffs, N.J. 1977.
53. J. D. Lambert: *Computational Methods in Ordinary Differential Equations*, J. Wiley and Sons, New York 1973.
54. N. R. Amundson: *Mathematical Methods in Chemical Engineering*, Prentice-Hall, Englewood Cliffs, N.J. 1966.
55. J. R. Rice: *Numerical Methods, Software, and Analysis*, McGraw-Hill, New York 1983.
56. M. B. Bogacki, K. Alejski, J. Szymanowski: "The Fast Method of the Solution of Reacting Distillation Problem," *Comput. Chem. Eng.* **13** (1989) 1081 – 1085.
57. C. W. Gear: *Numerical Initial-Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, N.J. 1971.
58. N. B. Ferguson, B. A. Finlayson: "Transient Modeling of a Catalytic Converter to Reduce Nitric Oxide in Automobile Exhaust," *AIChE J* **20** (1974) 539 – 550.
59. W. F. Ramirez: *Computational Methods for Process Simulations*, 2nd ed., Butterworth-Heinemann, Boston 1997.
60. U. M. Ascher, L. R. Petzold: *Computer methods for ordinary differential equations and differential-algebraic equations*, SIAM, Philadelphia 1998.
61. K. E. Brenan, S. L. Campbell, L. R. Petzold: *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, Elsevier, Amsterdam 1989.
62. C. C. Pontelides, D. Gritsis, K. R. Morison, R. W. H. Sargent: "The Mathematical Modelling of Transient Systems Using Differential-Algebraic Equations," *Comput. Chem. Eng.* **12** (1988) 449 – 454.
63. G. A. Byrne, P. R. Ponzi: "Differential-Algebraic Systems, Their Applications and Solutions," *Comput. Chem. Eng.* **12** (1988) 377 – 382.
64. L. F. Shampine, M. W. Reichelt: The MATLAB ODE Suite, *SIAM J. Sci. Comp.* **18** (1997) 122.
65. M. Kubicek, M. Marek: *Computational Methods in Bifurcation Theory and Dissipative Structures*, Springer Verlag, Berlin – Heidelberg – New York – Tokyo 1983.
66. T. F. C. Chan, H. B. Keller: "Arc-Length Continuation and Multi-Grid Techniques for Nonlinear Elliptic Eigenvalue Problems," *SIAM J. Sci. Stat. Comput.* **3** (1982) 173 – 194.
67. M. F. Doherty, J. M. Ottino: "Chaos in Deterministic Systems: Strange Attractors, Turbulence and Applications in Chemical Engineering," *Chem. Eng. Sci.* **43** (1988) 139 – 183.
68. M. P. Allen, D. J. Tildesley: *Computer Simulation of Liquids*, Clarendon Press, Oxford 1989.
69. D. Frenkel, B. Smit: *Understanding Molecular Simulation*, Academic Press, San Diego 2002.
70. J. M. Haile: *Molecular Dynamics Simulation*, John Wiley & Sons, New York 1992.
71. A. R. Leach: *Molecular Modelling: Principles and Applications*, Prentice Hall, Englewood Cliffs, NJ, 2001.
72. T. Schlick: *Molecular Modeling and Simulations*, Springer, New York 2002.
73. R. B. Bird, W. E. Stewart, E. N. Lightfoot: *Transport Phenomena*, 2nd ed., John Wiley & Sons, New York 2002.
74. R. B. Bird, R. C. Armstrong, O. Hassager: *Dynamics of Polymeric Liquids*, 2nd ed., Wiley-Interscience, New York 1987.

75. P. V. Danckwerts: "Continuous Flow Systems," *Chem. Eng. Sci.* **2** (1953) 1 – 13.
76. J. F. Wehner, R. Wilhelm: "Boundary Conditions of Flow Reactor," *Chem. Eng. Sci.* **6** (1956) 89 – 93.
77. V. Hlavácek, H. Hofmann: "Modeling of Chemical Reactors-XVI-Steady-State Axial Heat and Mass Transfer in Tubular Reactors. An Analysis of the Uniqueness of Solutions," *Chem. Eng. Sci.* **25** (1970) 173 – 185.
78. B. A. Finlayson: *Numerical Methods for Problems with Moving Fronts*, Ravenna Park Publishing Inc., Seattle 1990.
79. E. Isaacson, H. B. Keller: *Analysis of Numerical Methods*, J. Wiley and Sons, New York 1966.
80. C. Lanczos: "Trigonometric Interpolation of Empirical and Analytical Functions," *J. Math. Phys. (Cambridge Mass.) 17* (1938) 123 – 199.
81. C. Lanczos: *Applied Analysis*, Prentice-Hall, Englewood Cliffs, N.J. 1956.
82. J. Villadsen, W. E. Stewart: "Solution of Boundary-Value Problems by Orthogonal Collocation," *Chem. Eng. Sci.* **22** (1967) 1483 – 1501.
83. M. L. Michelsen, J. Villadsen: "Polynomial Solution of Differential Equations" pp. 341 – 368 in R. S. H. Mah, W. D. Seider (eds.): *Foundations of Computer-Aided Chemical Process Design*, Engineering Foundation, New York 1981.
84. W. E. Stewart, K. L. Levien, M. Morari: "Collocation Methods in Distillation," in A. W. Westerberg, H. H. Chien (eds.): *Proceedings of the Second Int. Conf. on Foundations of Computer-Aided Process Design*, Computer Aids for Chemical Engineering Education (CACHE), Austin Texas, 1984, pp. 535 – 569.
85. W. E. Stewart, K. L. Levien, M. Morari: "Simulation of Fractionation by Orthogonal Collocation," *Chem. Eng. Sci.* **40** (1985) 409 – 421.
86. C. L. E. Swartz, W. E. Stewart: "Finite-Element Steady State Simulation of Multiphase Distillation," *AIChE J.* **33** (1987) 1977 – 1985.
87. K. Alhumaizi, R. Henda, M. Soliman: "Numerical Analysis of a reaction-diffusion-convection system," *Comp. Chem. Eng.* **27** (2003) 579–594.
88. E. F. Costa, P. L. C. Lage, E. C. Biscaia, Jr.: "On the numerical solution and optimization of styrene polymerization in tubular reactors," *Comp. Chem. Eng.* **27** (2003) 1591–1604.
89. J. Wang, R. G. Anthony, A. Akgerman: "Mathematical simulations of the performance of trickle bed and slurry reactors for methanol synthesis," *Comp. Chem. Eng.* **29** (2005) 2474–2484.
90. V. K. C. Lee, J. F. Porter, G. McKay, A. P. Mathews: "Application of solid-phase concentration-dependent HSDM to the acid dye adsorption system," *AIChE J.* **51** (2005) 323–332.
91. C. deBoor, B. Swartz: "Collocation at Gaussian Points," *SIAM J. Num. Anal.* **10** (1973) 582 – 606.
92. R. F. Sincovec: "On the Solution of the Equations Arising From Collocation With Cubic B-Splines," *Math. Comp.* **26** (1972) 893 – 895.
93. U. Ascher, J. Christiansen, R. D. Russell: "A Collocation Solver for Mixed-Order Systems of Boundary-Value Problems," *Math. Comp.* **33** (1979) 659 – 679.
94. R. D. Russell, J. Christiansen: "Adaptive Mesh Selection Strategies for Solving Boundary-Value Problems," *SIAM J. Num. Anal.* **15** (1978) 59 – 80.
95. P. G. Ciarlet, M. H. Schultz, R. S. Varga: "Nonlinear Boundary-Value Problems I. One Dimensional Problem," *Num. Math.* **9** (1967) 394 – 430.
96. W. F. Ames: *Numerical Methods for Partial Differential Equations*, 2nd ed., Academic Press, New York 1977.
97. J. F. Botha, G. F. Pinder: *Fundamental Concepts in The Numerical Solution of Differential Equations*, Wiley-Interscience, New York 1983.
98. W. H. Press, B. P. Flanner, S. A. Teukolsky, W. T. Vetterling: *Numerical Recipes*, Cambridge Univ. Press, Cambridge 1986.
99. H. Schlichting, *Boundary Layer Theory*, 4th ed. McGraw-Hill, New York 1960.
100. R. Aris, N. R. Amundson: *Mathematical Methods in Chemical Engineering, vol. 2, First-Order Partial Differential Equations with Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1973.
101. R. Courant, D. Hilbert: *Methods of Mathematical Physics, vol. I and II*, Intersicence, New York 1953, 1962.
102. P. M. Morse, H. Feshbach: *Methods of Theoretical Physics, vol. I and II*, McGraw-Hill, New York 1953.
103. A. D. Polyanin: *Handbook of Linear Partial Differential Equations for Engineers and*

Scientists, Chapman and Hall/CRC, Boca Raton, FL 2002.
104. D. Ramkrishna, N. R. Amundson: *Linear Operator Methods in Chemical Engineering with Applications to Transport and Chemical Reaction Systems*, Prentice Hall, Englewood Cliffs, NJ, 1985.
105. D. D. Joseph, M. Renardy, J. C. Saut: "Hyperbolicity and Change of Type in the Flow of Viscoelastic Fluids," *Arch. Rational Mech. Anal.* **87** (1985) 213 – 251.
106. H. K. Rhee, R. Aris, N. R. Amundson: *First-Order Partial Differential Equations*, Prentice-Hall, Englewood Cliffs, N.J. 1986.
107. B. A. Finlayson: *Numerical Methods for Problems with Moving Fronts*, Ravenna Park Publishing Inc., Seattle 1990.
108. D. L. Book: *Finite-Difference Techniques for Vectorized Fluid Dynamics Calculations*, Springer Verlag, Berlin – Heidelberg – New York – Tokyo 1981.
109. G. A. Sod: *Numerical Methods in Fluid Dynamics*, Cambridge University Press, Cambridge 1985.
110. G. H. Xiu, J. L. Soares, P. Li, A. E. Rodriques: "Simulation of five-step one-bed sorption-endanced reaction process," *AIChE J.* **48** (2002) 2817–2832.
111. A. Malek, S. Farooq: "Study of a six-bed pressure swing adsorption process," *AIChE J.* **43** (1997) 2509–2523.
112. R. J. LeVeque: *Numerical Methods for Conservation Laws*, Birkhäuser, Basel 1992.
113. W. F. Ames: "Recent Developments in the Nonlinear Equations of Transport Processes," *Ind. Eng. Chem. Fundam.* **8** (1969) 522 – 536.
114. W. F. Ames: *Nonlinear Partial Differential Equations in Engineering*, Academic Press, New York 1965.
115. W. E. Schiesser: *The Numerical Method of Lines*, Academic Press, San Diego 1991.
116. D. Gottlieb, S. A. Orszag: *Numerical Analysis of Spectral Methods: Theory and Applications*, SIAM, Philadelphia, PA 1977.
117. D. W. Peaceman: *Fundamentals of Numerical Reservoir Simulation*, Elsevier, Amsterdam 1977.
118. G. Juncu, R. Mihail: "Multigrid Solution of the Diffusion-Convection-Reaction Equations which Describe the Mass and/or Heat Transfer from or to a Spherical Particle," *Comput. Chem. Eng.* **13** (1989) 259 – 270.
119. G. R. Buchanan: *Schaum's Outline of Finite Element Analysis*," McGraw-Hill, New York 1995.
120. M. D. Gunzburger: *Finite Element Methods for Viscous Incompressible Flows*, Academic Press, San Diego 1989.
121. H. Kardestuncer, D. H. Norrie: *Finite Element Handbook*, McGraw-Hill, New York 1987.
122. J. N. Reddy, D. K. Gartling: *The Finite Element Method in Heat Transfer and Fluid Dynamics*, 2nd ed., CRC Press, Boca Raton, FL 2000.
123. O. C. Zienkiewicz, R. L. Taylor, J. Z. Zhu: *The Finite Element Method: Its Basis & Fundamentals*, vol. **1**, 6th ed., Elsevier Butterworth-Heinemann, Burlington, MA 2005.
124. O. C. Zienkiewicz, R. L. Taylor: *The Finite Element Method*, Solid and Structural Mechanics, vol. **2**, 5th ed., Butterworth-Heinemann, Burlington, MA 2000.
125. O. C. Zienkiewicz, R. L. Taylor: *The Finite Element Method*, Fluid Dynamics, vol. **3**, 5th ed., Butterworth-Heinemann, Burlington, MA 2000.
126. Z. C. Li: *Combined Methods for Elliptic Equations with Singularities, Interfaces and Infinities*, Kluwer Academic Publishers, Boston, MA, 1998.
127. G. J. Fix, S. Gulati, G. I. Wakoff: "On the use of singular functions with finite element approximations," *J. Comp. Phys.* **13** (1973) 209–238.
128. N. M. Wigley: "On a method to subtract off a singularity at a corner for the Dirichlet or Neumann problem," *Math. Comp.* **23** (1968) 395–401.
129. H. Y. Hu, Z. C. Li, A. H. D. Cheng: "Radial Basis Collocation Methods for Elliptic Boundary Value Problems," *Comp. Math. Applic.* **50** (2005) 289–320.
130. S. J. Osher, R. P. Fedkiw:, *Level Set Methods and Dynamic Implicit Surfaces*, Springer, New York 2002.
131. M. W. Chang, B. A. Finlayson: "On the Proper Boundary Condition for the Thermal Entry Problem," *Int. J. Num. Methods Eng.* **15** (1980) 935 – 942.
132. R. Peyret, T. D. Taylor: *Computational Methods for Fluid Flow*, Springer Verlag, Berlin – Heidelberg – New York – Tokyo 1983.
133. P. M. Gresho, S. T. Chan, C. Upson, R. L. Lee: "A Modified Finite Element Method for Solving the Time-Dependent, Incompressible Navier – Stokes Equations," *Int. J. Num.*

Method. Fluids (1984) "Part 1. Theory": 557 – 589; "Part 2: Applications": 619 – 640.
134. P. M. Gresho, R. L. Sani: *Incompressible Flow and the Finite Element Method*, Advection-Diffusion, vol. **1**, John Wiley & Sons, New York 1998.
135. P. M. Gresho, R. L. Sani: *Incompressible Flow and the Finite Element Method* , Isothermal Laminar Flow, vol. **2**, John Wiley & Sons, New York 1998.
136. J. A. Sethian, *Level Set Methods*, Cambridge University Press, Cambridge 1996.
137. C. C. Lin, H. Lee, T. Lee, L. J. Weber: "A level set characteristic Galerkin finite element method for free surface flows," *Int. J. Num. Methods Fluids* **49** (2005) 521–547.
138. S. Succi: *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*, Oxford University Press, Oxford 2001.
139. M. C. Sukop, D. T. Thorne, Jr.: *Lattice Boltzmann Modeling: An Introduction for Geoscientists and Engineers*, Springer, New York 2006.
140. S. Chen, S. G. D. Doolen: "Lattice Boltzmann method for fluid flows," *Annu. Rev. Fluid Mech.* **30** (2001) 329–364.
141. H. T. Lau: *Numerical Library in C for Scientists and Engineers*, CRC Press, Boca Raton, FL 1994.
142. Y. Y. Al-Jaymany, G. Brenner, P. O. Brum: "Comparative study of lattice-Boltzmann and finite volume methods for the simulation of laminar flow through a 4:1 contraction," *Int. J. Num. Methods Fluids* **46** (2004) 903–920.
143. R. L. Burden, R. L J. D. Faires, A. C. Reynolds: *Numerical Analysis*, 8th ed., Brooks Cole, Pacific Grove, CA 2005.
144. S. C. Chapra, R. P. Canal: *Numerical Methods for Engineers*," 5th ed., McGraw-Hill, New York 2006.
145. H. T. Lau: *Numerical Library in Java for Scientists and Engineers*, CRC Press, Boca Raton, FL 2004.
146. K. W. Morton, D. F. Mayers: *Numerical Solution of Partial Differential Equations*, Cambridge University Press, New York 1994.
147. A. Quarteroni, A. Valli: *Numerical Approximation of Partial Differential Equations*, Springer, Heidelberg 1997.
148. F. Scheid: "Schaum's Outline of Numerical Analysis," 2nd ed., McGraw-Hill, New York 1989.
149. C. T. H. Baker: *The Numerical Treatment of Integral Equations*, Clarendon Press, Oxford 1977.
150. L. M. Delves, J. Walsh (eds.): *Numerical Solution of Integral Equations*, Clarendon Press, Oxford 1974.
151. P. Linz: *Analytical and Numerical Methods for Volterra Equations*, SIAM Publications, Philadelphia 1985.
152. M. A. Golberg (ed.): *Numerical Solution of Integral Equations*, Plenum Press, New York 1990.
153. C. Corduneanu: *Integral Equations and Applications*, Cambridge Univ. Press, Cambridge 1991.
154. R. Kress: *Linear Integral Equations*, 2nd ed. Springer, Heidelberg 1999.
155. P. K. Kythe P. Purl: *Computational Methods for Linear Integral Equations*, Birkhäuser, Basel 2002.
156. G. Nagel, G. Kluge: "Non-Isothermal Multicomponent Adsorption Processes and Their Numerical Treatment by Means of Integro-Differential Equations," *Comput. Chem. Eng.* **13** (1989) 1025 – 1030.
157. P. L. Mills, S. Lai, M. P. Duduković, P. A. Ramachandran: "A Numerical Study of Approximation Methods for Solution of Linear and Nonlinear Diffusion-Reaction Equations with Discontinuous Boundary Conditions," *Comput. Chem. Eng.* **12** (1988) 37 – 53.
158. D. Duffy: *Green's Functions with Applications*, Chapman and Hall/CRC, New York 2001.
159. I. Statgold: *Green's Functions and Boundary Value Problems*, 2nd ed., Interscience, New York 1997.
160. B. Davies: *Integral Transforms and Their Applications*, 3rd ed., Springer, New York 2002.
161. J. M. Bownds, "Theory and performance of a subroutine for solving Volterra integral equations," *J. Comput.* **28** 317–332 (1982).
162. J. G. Blom, H. Brunner, "Algorithm 689; discretized collocation and iterated collocation for nonlinear Volterra integral equations of the second kind," *ACM Trans. Math. Software* **17** (1991) 167–177.
163. N. B. Ferguson, B. A. Finlayson: "Error Bounds for Approximate Solutions to Nonlinear Ordinary Differential Equations," *AIChE J.* **18** (1972) 1053 – 1059.
164. R. S. Dixit, L. L. Taularidis: "Integral Method of Analysis of Fischer – Tropsch Synthesis Reactions in a Catalyst Pellet," *Chem. Eng. Sci.* **37** (1982) 539 – 544.

165. L. Lapidus, G. F. Pinder: *Numerical Solution of Partial Differential Equations in Science and Engineering*, Wiley-Interscience, New York 1982.
166. C. K. Hsieh, H. Shang: "A Boundary Condition Dissection Method for the Solution of Boundary-Value Heat Conduction Problems with Position-Dependent Convective Coefficients," *Num. Heat Trans. Part B: Fund.* **16** (1989) 245 – 255.
167. C. A. Brebbia, J. Dominguez: *Boundary Elements – An Introductory Course*, 2nd ed., Computational Mechanics Publications, Southhamtpon 1992.
168. J. Mackerle, C. A. Brebbia (eds.): *Boundary Element Reference Book*, Springer Verlag, Berlin – Heidelberg – New York – Tokyo 1988.
169. O. L. Mangasarian: *Nonlinear Programming*, McGraw-Hill, New York 1969.
170. G. B. Dantzig: *Linear Programming and Extensions*, Princeton University Press, Princeton, N.J, 1963.
171. S. J. Wright: *Primal-Dual Interior Point Methods*, SIAM, Philadelphia 1996.
172. F. Hillier, G. J. Lieberman: *Introduction to Operations Research*, Holden-Day, San Francisco, 1974.
173. T. F. Edgar, D. M. Himmelblau, L. S. Lasdon: *Optimization of Chemical Processes*, McGraw-Hill Inc., New York 2002.
174. K. Schittkowski: *More Test Examples for Nonlinear Programming Codes, Lecture notes in economics and mathematical systems no. 282*, Springer-Verlag, Berlin 1987.
175. J. Nocedal, S. J. Wright: *Numerical Optimization*, Springer, New York 1999.
176. R. Fletcher: *Practical Optimization*, John Wiley & Sons, Ltd., Chichester, UK 1987
177. A. Wächter, L. T. Biegler: "On the Implementation of an Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming," **106** *Mathematical Programming* (2006) no. 1, 25 – 57.
178. C. V. Rao, J. B. Rawlings S. Wright: On the Application of Interior Point Methods to Model Predictive Control. *J. Optim. Theory Appl.* **99** (1998) 723.
179. J. Albuquerque, V. Gopal, G. Staus, L. T. Biegler, B. E. Ydstie: "Interior point SQP Strategies for Large-scale Structured Process Optimization Problems," *Comp. Chem. Eng.* **23** (1997) 283.
180. T. Jockenhoevel, L. T. Biegle, A. Wächter: "Dynamic Optimization of the Tennessee Eastman Process Using the OptControlCentre," *Comp. Chem. Eng.* **27** (2003) no. 11, 1513–1531.
181. L. T. Biegler, A. M. Cervantes, A. Wächter: "Advances in Simultaneous Strategies for Dynamic Process Optimization," *Chemical Engineering Science* **57** (2002) no. 4, 575–593.
182. C. D. Laird, L. T. Biegler, B. van Bloemen Waanders, R. A. Bartlett: "Time Dependent Contaminant Source Determination for Municipal Water Networks Using Large Scale Optimization," *ASCE Journal of Water Resource Management and Planning* **131** (2005) no. 2, 125.
183. A. R. Conn, N. Gould, P. Toint: *Trust Region Methods*, SIAM, Philadelphia 2000.
184. A. Drud: "CONOPT – A Large Scale GRG Code," *ORSA Journal on Computing* **6** (1994) 207–216.
185. B. A. Murtagh, M. A. Saunders: MINOS 5.1 User's Guide, Technical Report SOL 83-20R, Stanford University, Palo Alto 1987.
186. R. H. Byrd, M. E. Hribar, J. Nocedal: An Interior Point Algorithm for Large Scale Nonlinear Programming, *SIAM J. Opt.* **9** (1999) no. 4, 877.
187. R. Fletcher, N. I. M. Gould, S. Leyffer, Ph. L. Toint, A. Waechter: "Global Convergence of a Trust-region (SQP)-filter Algorithms for General Nonlinear Programming," *SIAM J. Opt.* **13** (2002) no. 3, 635–659.
188. R. J. Vanderbei, D. F. Shanno: An Interior Point Algorithm for Non-convex Nonlinear Programming. Technical Report SOR-97-21, CEOR, Princeton University, Princeton, NJ, 1997.
189. P. E. Gill, W. Murray, M. Wright: *Practical Optimization*, Academic Press, New York 1981.
190. P. E. Gill, W. Murray, M. A. Saunders: *User's guide for SNOPT: A FORTRAN Package for Large-scale Nonlinear Programming, Technical report, Department of Mathematics*, University of California, San Diego 1998.
191. J. T. Betts: *Practical Methods for Optimal Control Using Nonlinear Programming, Advances in Design and Control 3*, SIAM, Philadelphia 2001.
192. gPROMS User's Guide, PSE Ltd., London 2002.
193. G. E. P. Box: "Evolutionary Operation: A Method for Increasing Industrial Productivity," *Applied Statistics* **6** (1957) 81–101.
194. R. Hooke, T. A. Jeeves: "Direct Search Solution of Numerical and Statistical Problems," *J. ACM* **8** (1961) 212.

195. M. J. D. Powell: "An Efficient Method for Finding the Minimum of a Function of Several Variables without Calculating Derivatives, *Comput. J.* **7** (1964) 155.
196. J. A. Nelder, R. Mead: "A Simplex Method for Function Minimization," *Computer Journal* **7** (1965) 308.
197. R. Luus, T. H. I. Jaakola: "Direct Search for Complex Systems," *AIChE J* **19** (1973) 645–646.
198. R. Goulcher, J. C. Long: "The Solution of Steady State Chemical Engineering Optimization Problems using a Random Search Algorithm," *Comp. Chem. Eng.* **2** (1978) 23.
199. J. R. Banga, W. D. Seider: "Global Optimization of Chemical Processes using Stochastic Algorithms," C. Floudas and P. Pardalos (eds.): *State of the Art in Global Optimization*, Kluwer, Dordrecht 1996, p. 563.
200. P. J. M. van Laarhoven, E. H. L. Aarts: *Simulated Annealing: Theory and Applications*, Reidel Publishing, Dordrecht 1987.
201. J. H. Holland: *Adaptations in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor 1975.
202. J. E. Dennis, V. Torczon:, "Direct Search Methods on Parallel Machines," *SIAM J. Opt.* **1** (1991) 448.
203. A. R. Conn, K. Scheinberg, P. Toint: "Recent Progress in Unconstrained Nonlinear Optimization without Derivatives," *Mathemtatical Programming, Series B*, **79** (1997) no. 3, 397.
204. M. Eldred: "DAKOTA: A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis," 2002, http://endo.sandia.gov/DAKOTA/software.html
205. A. J. Booker et al.: "A Rigorous Framework for Optimization of Expensive Functions by Surrogates," CRPC Technical Report 98739, Rice University, Huston TX, February 1998.
206. R. Horst, P. M. Tuy: "Global Optimization: Deterministic Approaches," 3rd ed., Springer Verlag, Berlin 1996.
207. I. Quesada, I. E. Grossmann: "A Global Optimization Algorithm for Linear Fractional and Bilinear Programs," *Journal of Global Optimization* **6** (1995) no. 1, 39–76.
208. G. P. McCormick: "Computability of Global Solutions to Factorable Nonconvex Programs: Part I – Convex Underestimating Problems," *Mathematical Programming* **10** (1976) 147–175.
209. H. S. Ryoo, N. V. Sahinidis: "Global Optimization of Nonconvex NLPs and MINLPs with Applications in Process Design," *Comp. Chem. Eng.* **19** (1995) no. 5, 551–566.
210. M. Tawarmalani, N. V. Sahinidis: "Global Optimization of Mixed Integer Nonlinear Programs: A Theoretical and Computational Study," *Mathematical Programming* **99** (2004) no. 3, 563–591.
211. C. S. Adjiman., I.P. Androulakis, C.A. Floudas: "Global Optimization of MINLP Problems in Process Synthesis and Design. *Comp. Chem. Eng.*, **21** (1997) Suppl., S445–S450.
212. C.S. Adjiman, I.P. Androulakis and C.A. Floudas, "Global Optimization of Mixed-Integer Nonlinear Problems," *AIChE Journal*, **46** (2000) no. 9, 1769–1797.
213. C. S. Adjiman, I.P. Androulakis, C.D. Maranas, C. A. Floudas: "A Global Optimization Method, αBB, for Process Design," *Comp. Chem. Eng.*, **20** (1996) Suppl., S419–S424.
214. J. M. Zamora, I. E. Grossmann: "A Branch and Contract Algorithm for Problems with Concave Univariate, Bilinear and Linear Fractional Terms," *Journal of Gobal Optimization* **14** (1999) no. 3, 217–249.
215. E. M. B. Smith, C. C. Pantelides: "Global Optimization of Nonconvex NLPs and MINLPs with Applications in Process Design," *Comp. Chem. Eng.*, **21** (1997) no. 1001, S791–S796.
216. J. E. Falk, R. M. Soland: "An Algorithm for Separable Nonconvex Programming Problems," *Management Science* **15** (1969) 550–569.
217. F. A. Al-Khayyal, J. E. Falk: "Jointly Constrained Biconvex Programming," *Mathematics of Operations Research* **8** (1983) 273–286.
218. F. A. Al-Khayyal: "Jointly Constrained Bilinear Programs and Related Problems: An Overview," *Computers and Mathematics with Applications*, **19** (1990) 53–62.
219. I. Quesada, I. E. Grossmann: "A Global Optimization Algorithm for Heat Exchanger Networks," *Ind. Eng. Chem. Res.* **32** (1993) 487–499.
220. H. D. Sherali, A. Alameddine: "A New Reformulation-Linearization Technique for

Bilinear Programming Problems," *Journal of Global Optimization* **2** (1992) 379–410.
221. H. D. Sherali, C. H. Tuncbilek: "A Reformulation-Convexification Approach for Solving Nonconvex Quadratic Programming Problems," *Journal of Global Optimization* **7** (1995) 1–31.
222. C. A. Floudas: *Deterministic Global Optimization: Theory, Methods and Applications*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.
223. M. Tawarmalani, N. Sahinidis: *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*, Kluwer Academic Publishers, Dordrecht 2002.
224. R. Horst, H. Tuy: *Global optimization: Deterministic approaches*. Springer-Verlag, Berlin 1993.
225. A. Neumaier, O. Shcherbina, W. Huyer, T. Vinko: "A Comparison of Complete Global Optimization Solvers," *Math. Programming B* **103** (2005) 335–356.
226. L. Lovász A. Schrijver: "Cones of Matrices and Set-functions and 0-1 Optimization," *SIAM J. Opt.* **1** (1991) 166–190.
227. H. D. Sherali, W.P. Adams: "A Hierarchy of Relaxations Between the Continuous and Convex Hull Representations for Zero-One Programming Problems," *SIAM J. Discrete Math.* **3** (1990) no. 3, 411–430.
228. E. Balas, S. Ceria, G. Cornuejols: "A Lift-and-Project Cutting Plane Algorithm for Mixed 0-1 Programs," *Mathematical Programming* **58** (1993) 295–324.
229. A. H. Land, A.G. Doig: "An Automatic Method for Solving Discrete Programming Problems," *Econometrica* **28** (1960) 497–520.
230. E. Balas: "An Additive Algorithm for Solving Linear Programs with Zero-One Variables," *Operations Research* **13** (1965) 517–546.
231. R. J. Dakin: "A Tree Search Algorithm for Mixed-Integer Programming Problems", *Computer Journal* **8** (1965) 250–255.
232. R. E. Gomory: "Outline of an Algorithm for Integer Solutions to Linear Programs," *Bulletin of the American Mathematics Society* **64** (1958) 275–278.
233. H. P. Crowder, E. L. Johnson, M. W. Padberg: "Solving Large-Scale Zero-One Linear Programming Problems," *Operations Research* **31** (1983) 803–834.
234. T. J. Van Roy, L. A. Wolsey: "Valid Inequalities for Mixed 0-1 Programs," *Discrete Applied Mathematics* **14** (1986) 199–213.
235. E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh: "Progress in Linear Programming Based Branch-and-Bound Algorithms: Exposition," *INFORMS Journal of Computing* **12** (2000) 2–23.
236. J. F. Benders: "Partitioning Procedures for Solving Mixed-variables Programming Problems," *Numeri. Math.* **4** (1962) 238–252.
237. G. L. Nemhauser, L. A. Wolsey: *Integer and Combinatorial Optimization*, Wiley-Interscience, New York 1988.
238. C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, P. H. Vance: "Branch-and-price: Column Generation for Solving Huge Integer Programs," *Operations Research* **46** (1998) 316–329.
239. O. K. Gupta, V. Ravindran: "Branch and Bound Experiments in Convex Nonlinear Integer Programming," *Management Science* **31** (1985) no. 12, 1533–1546.
240. S. Nabar, L. Schrage: "Modeling and Solving Nonlinear Integer Programming Problems," Presented at Annual AIChE Meeting, Chicago 1991.
241. B. Borchers, J.E. Mitchell: "An Improved Branch and Bound Algorithm for Mixed Integer Nonlinear Programming," *Computers and Operations Research* **21** (1994) 359–367.
242. R. Stubbs, S. Mehrotra: "A Branch-and-Cut Method for 0-1 Mixed Convex Programming," *Mathematical Programming* **86** (1999) no. 3, 515–532.
243. S. Leyffer: "Integrating SQP and Branch and Bound for Mixed Integer Noninear Programming," *Computational Optimization and Applications* **18** (2001) 295–309.
244. A. M. Geoffrion: "Generalized Benders Decomposition," *Journal of Optimization Theory and Applications* **10** (1972) no. 4, 237–260.
245. M. A. Duran, I.E. Grossmann: "An Outer-Approximation Algorithm for a Class of Mixed-integer Nonlinear Programs," *Math Programming* **36** (1986) 307.
246. X. Yuan, S. Zhang, L. Piboleau, S. Domenech : "Une Methode d'optimisation Nonlineare en Variables Mixtes pour la Conception de Procedes," *RAIRO* **22** (1988) 331.
247. R. Fletcher, S. Leyffer: "Solving Mixed Integer Nonlinear Programs by Outer Approximation," *Math Programming* **66** (1974) 327.

248. I. Quesada, I.E. Grossmann: "An LP/NLP Based Branch and Bound Algorithm for Convex MINLP Optimization Problems," *Comp. Chem. Eng.* **16** (1992) 937–947.
249. T. Westerlund, F. Pettersson: "A Cutting Plane Method for Solving Convex MINLP Problems," *Comp. Chem. Eng.* **19** (1995) S131–S136.
250. O. E. Flippo, A. H. G. R. Kan: "Decomposition in General Mathematical Programming," *Mathematical Programming* **60** (1993) 361–382.
251. T. L. Magnanti, R. T. Wong: "Acclerated Benders Decomposition: Algorithm Enhancement and Model Selection Criteria," *Operations Research* **29** (1981) 464–484.
252. N. V. Sahinidis, I. E. Grossmann: "Convergence Properties of Generalized Benders Decomposition," *Comp. Chem. Eng.* **15** (1991) 481.
253. J. E. Kelley Jr.: "The Cutting-Plane Method for Solving Convex Programs," *J. SIAM* **8** (1960) 703–712.
254. S. Leyffer: "Deterministic Methods for Mixed-Integer Nonlinear Programming," Ph.D. thesis, Department of Mathematics and Computer Science, University of Dundee, Dundee 1993.
255. M. S. Bazaraa, H. D. Sherali, C. M. Shetty: *Nonlinear Programming*, John Wiley & Sons, Inc., New York 1994.
256. G. R. Kocis, I. E. Grossmann: "Relaxation Strategy for the Structural Optimization of Process Flowsheets," *Ind. Eng. Chem. Res.* **26** (1987) 1869.
257. I. E. Grossmann: "Mixed-Integer Optimization Techniques for Algorithmic Process Synthesis", *Advances in Chemical Engineering*, vol. **23**, Process Synthesis, Academic Press, London 1996, pp. 171–246.
258. E. M. B. Smith, C. C. Pantelides: "A Symbolic Reformulation/Spatial Branch and Bound Algorithm for the Global Optimization of Nonconvex MINLPs," *Comp. Chem. Eng.* **23** (1999) 457–478.
259. P. Kesavan P. P. I. Barton: "Generalized Branch-and-cut Framework for Mixed-integer Nonlinear Optimization Problems," *Comp. Chem. Eng.* **24** (2000) 1361–1366.
260. S. Lee I. E. Grossmann: "A Global Optimization Algorithm for Nonconvex Generalized Disjunctive Programming and Applications to Process Systems," *Comp. Chem. Eng.* **25** (2001) 1675–1697.
261. R. Pörn, T. Westerlund: "A Cutting Plane Method for Minimizing Pseudo-convex Functions in the Mixed-integer Case," *Comp. Chem. Eng.* **24** (2000) 2655–2665.
262. J. Viswanathan, I. E. Grossmann: "A Combined Penalty Function and Outer-Approximation Method for MINLP Optimization," *Comp. Chem. Eng.* **14** (1990) 769.
263. A. Brooke, D. Kendrick, A. Meeraus, R. Raman: GAMS – A User's Guide, www.gams.com, 1998.
264. N. V. A. Sahinidis, A. Baron: "A General Purpose Global Optimization Software Package," *Journal of Global Optimization* **8** (1996) no.2, 201–205.
265. C. A. Schweiger, C. A. Floudas: "Process Synthesis, Design and Control: A Mixed Integer Optimal Control Framework," Proceedings of DYCOPS-5 on Dynamics and Control of Process Systems, Corfu, Greece 1998, pp. 189–194.
266. R. Raman, I. E. Grossmann: "Modelling and Computational Techniques for Logic Based Integer Programming," *Comp. Chem. Eng.* **18** (1994) no.7, 563.
267. J. N. Hooker, M. A. Osorio: "Mixed Logical.Linear Programming," *Discrete Applied Mathematics* **96-97** (1994) 395–442.
268. P. V. Hentenryck: *Constraint Satisfaction in Logic Programming*, MIT Press, Cambridge, MA, 1989.
269. J. N. Hooker: *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*, John Wiley & Sons, New York 2000.
270. E. Balas: "Disjunctive Programming," *Annals of Discrete Mathematics* **5** (1979) 3–51.
271. H. P. Williams: *Mathematical Building in Mathematical Programming*, John Wiley, Chichester 1985.
272. R. Raman, I. E. Grossmann: "Relation between MILP Modelling and Logical Inference for Chemical Process Synthesis," *Comp. Chem. Eng.* **15** (1991) no. 2, 73.
273. S. Lee, I. E. Grossmann: "New Algorithms for Nonlinear Generalized Disjunctive Programming," *Comp. Chem. Eng.* **24** (2000) no. 9-10, 2125–2141.
274. J. Hiriart-Urruty, C. Lemaréchal: *Convex Analysis and Minimization Algorithms*, Springer-Verlag, Berlin, New York 1993.
275. E. Balas: "Disjunctive Programming and a Hierarchy of Relaxations for Discrete

Optimization Problems," *SIAM J. Alg. Disc. Meth.* **6** (1985) 466–486.
276. I. E. Grossmann, S. Lee: "Generalized Disjunctive Programming: Nonlinear Convex Hull Relaxation and Algorithms," *Computational Optimization and Applications* **26** (2003) 83–100.
277. S. Lee, I. E. Grossmann: "Logic-based Modeling and Solution of Nonlinear Discrete/Continuous Optimization Problems," *Annals of Operations Research* **139** 2005 267–288.
278. ILOG, Gentilly Cedex, France 1999, www.ilog.com,
279. M. Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf, F. Berthier: The Constraint Logic Programming Language CHIP, FGCS-88: Proceedings of International Conference on Fifth Generation Computer Systems, Tokyo, 693–702.
280. M. Wallace, S. Novello, J. Schimpf: "ECLiPSe: a Platform for Constraint Logic Programming," *ICL Systems Journal* **12** (1997) no.1, 159–200.
281. V. Pontryagin, V. Boltyanskii, R. Gamkrelidge, E. Mishchenko: *The Mathematical Theory of Optimal Processes*, Interscience Publishers Inc., New York, NY, 1962.
282. A. E. Bryson, Y. C. Ho: "Applied Optimal Control: Optimization, Estimation, and Control," Ginn and Company, Waltham, MA, 1969.
283. V. Vassiliadis, PhD Thesis, Imperial College, University of London 1993.
284. W. F. Feehery, P. I. Barton: "Dynamic Optimization with State Variable Path Constraints," *Comp. Chem. Eng.*, **22** (1998) 1241–1256.
285. L. Hasdorff:. *Gradient Optimization and Nonlinear Control*, Wiley-Interscience, New York, NY, 1976.
286. R. W. H. Sargent, G. R. Sullivan: "Development of Feed Changeover Policies for Refinery Distillation Units," *Ind. Eng. Chem. Process Des. Dev.* **18** (1979) 113–124.
287. V. Vassiliadis, R. W. H. Sargent, C. Pantelides: "Solution of a Class of Multistage Dynamic Optimization Problems," *I & EC Research* **33** (1994) 2123.
288. K. F. Bloss, L. T. Biegler, W. E. Schiesser: "Dynamic Process Optimization through Adjoint Formulations and Constraint Aggregation,". *Ind. Eng. Chem. Res.* **38** (1999) 421–432.
289. H. G. Bock, K. J. Plitt: A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems, 9th IFAC World Congress, Budapest 1984.
290. D. B. Leineweber, H. G. Bock, J. P. Schlöder, J. V. Gallitzendörfer, A. Schäfer, P. Jansohn: "A Boundary Value Problem Approach to the Optimization of Chemical Processes Described by DAE Models," *Computers & Chemical Engineering*, April 1997. (IWR-Preprint 97-14, Universität Heidelberg, March 1997.
291. J. E. Cuthrell, L. T. Biegler: "On the Optimization of Differential- algebraic Process Systems," *AIChE Journal* **33** (1987) 1257–1270.
292. A. Cervantes, L. T. Biegler: "Large-scale DAE Optimization Using Simultaneous Nonlinear Programming Formulations. *AIChE Journal* **44** (1998) 1038.
293. P. Tanartkit, L. T. Biegler: "Stable Decomposition for Dynamic Optimization," *Ind. Eng. Chem. Res.* **34** (1995) 1253–1266.
294. S. Kameswaran, L. T. Biegler: "Simultaneous Dynamic Optimization Strategies: Recent Advances and Challenges," *Chemical Process Control – 7*, to appear 2006.
295. B. van Bloemen Waanders, R. Bartlett, K. Long, P. Boggs, A. Salinger: "Large Scale Non-Linear Programming for PDE Constrained Optimization," Sandia Technical Report SAND2002-3198, October 2002.
296. G. P. Box, J. S. Hunter, W. G. Hunter: *Statistics for Experimenters: Design, Innovation, and Discovery*, 2nd ed., John Wiley & Sons, New York 2005.
297. D. C. Baird: *Experimentation: An Introduction to Measurement Theory and Experiment Design*, 3rd ed., Prentice Hall, Engelwood Cliffs, NJ, 1995.
298. J. B. Cropley: "Heuristic Approach to Comples Kinetics," *ACS Symp. Ser.* **65** (1978) 292 – 302.
299. S. Lipschutz, J. J. Schiller, Jr: *Schaum's Outline of Theory and Problems of Introduction to Probability and Statistics*, McGraw-Hill, New York 1988.
300. D. S. Moore, G. P. McCabe: *Introduction to the Practice of Statistics*," 4th ed., Freeman, New York 2003.
301. D. C. Montgomery, G. C. Runger: *Applied Statistics and Probability for Engineers*, 3 rd ed., John Wiley & Sons, New York 2002.
302. D. C. Montgomery, G. C. Runger, N. F. Hubele: *Engineering Statistics*, 3rd ed., John Wiley & Sons, New York 2004.

303. G. E. P. Box, W. G. Hunter, J. S. Hunter: *Statistics for Experimenters*, John Wiley & Sons, New York 1978.
304. B. W. Lindgren: *Statistical Theory*, Macmillan, New York 1962.
305. O. A. Hougen, K. M. Watson, R. A. Ragatz: *Chemical Process Principles*, 2nd ed., part II, "Thermodynamics," J. Wiley & Sons, New York 1959.
306. L. T. Biegler, I. E. Grossmann, A. W. Westerberg: *Systematic Methods for Chemical Process Design*, Prentice-Hall Englewood Cliffs, NJ 1997.
307. I. E. Grossmann (ed.), "Global Optimization in Engineering Design", Kluwer, Dordrecht 1996.
308. J. Kallrath, "Mixed Integer Optimization in the Chemical Process Industry: Experience, Potential and Future," *Trans. I .Chem E.*, **78** (2000) Part A, 809–822.
309. J. N. Hooker: *Logic-Based Methods for Optimization*, John Wiley & Sons, New York 1999.